ALIBABA CLOUD

Alibaba Cloud

Elastic Compute Service

Best Practices

Document Version: 20201012

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	Danger: Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	Notice: If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Quick reference 07
2.Best practices for instance type selection 12
3.Provisioning methods of ECS instances 20
4.Best practices for cost management 23
5.Security 27
5.1. Best practices of the security group (part 1) 27
5.2. Best practices of the security group (part 2) 29
5.3. Best practices of the security group (part 3) 34
5.4. ECS data security best practices 37
5.5. Improve ECS instance security 39
5.6. Configure interconnection of instances over the classic ... 41
5.7. Modify the default remote port of an instance 44
5.8. Use logs in Windows instances 46
5.9. Best practices for Windows Firewall with Advanced Secu... 47
5.10. Isolation of instances within a security group 50
5.11. Security group quintuple rules 51
5.12. Enable or disable SELinux 53
5.13. Revoke the authorization for internal network commun... 55
5.14. Authorize internal network communication between EC... 57
6.Data recovery 60
6.1. Restore data deleted by mistake 60
6.2. Handle low disk space on Windows instances 62
6.3. Restore data in Linux instances 65
6.4. Restore data in Windows instances 71
7.Configuration preference 74
7.1. Transfer ECS instance data 74

7.2. Increase data throughput through read/write splitting 79
7.3. Change the preferred language of a Windows instance 84
7.4. Boot a Linux ECS instance into single user mode 86
8.Block Storage 89
8.1. Resize partitions and file systems of Linux system disks 89
8.2. Resize partitions and file systems of Linux data disks 94
8.3. Use LVs for Linux 111
8.3.1. Create an LV 111
8.3.2. Resize an LV 114
8.4. Create RAID arrays for Linux 116
8.5. Decrease the size of a disk 120
9.Best practices for tag design 123
10.Best practices for using custom images 126
10.1. Overview 126
10.2. Packer: machine images as code 127
10.2.1. Benefits of using Packer to create custom images 127
10.2.2. Alicloud Image Builder parameters used to implem... 133
10.3. Create and import a custom image 137
11.Monitor 139
11.1. Use CloudMonitor to monitor ECS instances 139
12.Use RAM roles to access other Alibaba Cloud services 141
13.GPU instances 147
13.1. Deploy an NGC environment on instances with GPU cap... 147
13.2. Use RAPIDS to accelerate machine learning tasks on a... 149
14.FaaS instances best practices 156
14.1. Use RTL Compiler on an f1 instance 156
14.2. Use OpenCL on an f1 instance 159
14.3. Use OpenCL on an f3 instance 163

14.4. Use the RTL design on an f3 instance 170
14.5. faascmd tool 174
14.5.1. faascmd overview 174
14.5.2. Install faascmd 175
14.5.3. Configure faascmd 176
14.5.4. Use faascmd 176
14.5.5. faascmd FAQ 181
15.Process ECS status change events 189
16.DevOps tutorials 196
16.1. DevOps for small and medium web apps 196
16.1.1. General introduction 196
16.1.2. Install and configure GitLab 196
16.1.3. Continuous integration 211
16.1.4. Code quality 219
16.1.5. Continuous delivery 237
16.2. HTTPS configuration 283
16.3. Log management 315
16.4. Speeding up CI and CD pipeline 328
16.5. Getting started with DevOps with Kubernetes 339
16.6. Getting started with Rancher 345
17.Disaster recovery solutions 353
18.Deploy a highly available architecture 355
18.1. Deploy a highly available architecture 355
18.2. Replicate ECS instances 355
18.3. Configure an SLB instance 357
18.4. Migrate user-created databases to RDS instances 359

1.Quick reference

This topic provides a quick reference guide for common operations of ECS instances. This topic also introduces common operations on instance resources.

This guide offers solutions for scenarios such as how to connect to ECS instances, change operating systems, resize cloud disks, upgrade or downgrade configurations, and use snapshots or images.

Limits

- For more information about considerations of ECS instances, see Usage notes.
- For more information about limits of ECS resources, see Limits and View quotas (old version).
- To apply for ICP filings for websites that are deployed on your ECS instance, ensure that the instance meets ICP filing requirements. You can apply for a limited number of ICP filing service numbers for each ECS instance. For more information, see 备案服务器（接入信息）准备与检查. For information about how to apply for ICP filings, see ICP filing application overview.

Create and manage ECS instances

- You can perform the following steps to manage the lifecycle of an ECS instance:
  i. Create an instance by using the provided wizard
  ii. Connect to an ECS instance
  iii. Stop an instance
  iv. Release an instance
- If the current instance type or network configuration is unsuitable for your business, you can change the instance type, IP address, and peak Internet bandwidth:
  - Subscription instances:
    - Upgrade configurations of subscription instances
    - Downgrade instance configurations during renewal
  - Pay-as-you-go instances:
    - Change the instance type of a pay-as-you-go instance
    - Change the Internet bandwidth of a pay-as-you-go instance
  - IP addresses of ECS instances:
    - Change the public IP address of an ECS instance
    - Convert the public IP address of a VPC-type instance to an Elastic IP address
- If the current operating system is unsuitable for your business, you can change the operating system. For more information, see Change the operating system.
- You can use the following features to control and manage ECS instances in a fine-grained manner:
  - User data
  - Metadata
  - Instance identity
  - Instance RAM roles

Manage the billing method

- Subscription instances:

  You can use one of the following methods to renew subscription instances:

  - Manually renew an instance
  - Enable auto-renewal
  - Downgrade instance configurations during renewal

- Pay-as-you-go instances:

  You can enable the No Fees for Stopped Instances (VPC-Connected) feature for pay-as-you-go instances. For more information, see No Fees for Stopped Instances (VPC-Connected).

- Change the billing method of ECS instances:
  - Switch the billing method from pay-as-you-go to subscription
  - Change the billing method of instances from subscription to pay-as-you-go

Improve cost-effectiveness

- You can purchase preemptible instances to reduce costs and achieve automatic scaling by combining with auto provisioning. For more information, see Create an auto provisioning group and Create a preemptible instance.
- You can purchase reserved instances to improve the flexibility of paying for instances and reduce costs. For more information, see Purchase reserved instances.

Create and manage cloud disks

If you want to use a cloud disk as a data disk, you can perform the following steps:

1. Create a pay-as-you-go disk.
2. Attach a cloud disk.
3. Format a data disk for a Linux instance or Format a data disk for a Windows ECS instance.
4. Create a snapshot to back up data. For more information, see Create a snapshot.
5. If the capacity of the existing system disk or data disk cannot meet your requirements, you can resize the system or data disk. For more information, see Resize disks online for Linux instances and Resize disks offline for Linux instances. If you want to resize the data disk, perform one of the following operations based on your operating system:
   - Resize disks online for Windows instances
   - Resize partitions and file systems of Linux system disks
   - Resize partitions and file systems of Linux data disks
6. If data error occurs on a cloud disk, you can use a snapshot from a specified point in time to roll back the cloud disk. For more information, see Roll back a disk by using a snapshot.
7. If you want to restore a cloud disk to its initial status, you can reinitialize the disk. For more information, see Re-initialize a data disk.
8. Detach a cloud disk.
9. Release a cloud disk.

Create and manage snapshots

You can perform the following steps to use a snapshot:

1. Create a snapshot by using one of the following methods:
   - Create a normal snapshot.
   - Use an automatic snapshot policy to create snapshots automatically on a regular basis. For more information, see Apply or disable an automatic snapshot policy.
2. View the snapshot size.
3. Delete unnecessary snapshots to save storage space. For more information, see Reduce snapshot fees.

The common application scenarios for snapshots are as follows:

- To copy or back up data: You can use a snapshot to create or roll back a cloud disk. For more information, see Create a disk from a snapshot and Roll back a disk by using a snapshot.
- To ease environment deployment: You can use a system disk snapshot to create a custom image and use the custom image to create instances. For more information, see Create a custom image from a snapshot and Create an ECS instance by using a custom image.

Create and manage custom images

Only custom images can be managed in the ECS console. You can use a custom image to quickly deploy a business environment. You can use one of the following methods to obtain a custom image.

- Create a custom image from a snapshot.
- Create a custom image from an instance.
- Create a custom image by using Packer.
- Copy custom images across regions. For more information, see Copy custom images.
- Share custom images across accounts. For more information, see Share custom images.
- Import custom images.
- Create and import on-premises images by using Packer.

You can export custom images to back up environments. For more information, see Export custom images.

Create and manage security groups

You can perform the following steps to create and manage a security group.

1. Create a security group.
2. Add security group rules.
3. Add an ECS instances to a security group.
4. Delete a security group rule.
5. Delete security groups.

You can clone a security group across regions and network types to simplify business deployment. For more information, see Clone a security group.

If new security group rules disrupt your online business, you can perform a complete or partial restoration of the security group rules. For more information, see Restore security group rules.

Create and bind instance RAM roles

You can perform the following steps to create and bind an instance RAM role.

1. Optional. Authorize a RAM user to use an instance RAM role. For more information, see Authorize a RAM user to use an instance RAM role.
2. Create and bind an instance RAM role. For more information, see Bind an instance RAM role.
3. Replace the instance RAM role based on your needs. For more information, see Replace an instance RAM role.

Create and manage SSH key pairs

You can perform the following steps to create and manage an SSH key pair:

1. Create an SSH key pair or Import an SSH key pair.
2. Bind an SSH key pair to an instance.
3. Connect to a Linux instance by using an SSH key pair.
4. Unbind an SSH key pair.
5. Delete an SSH key pair.

Create and manage ENIs

You can perform the following steps to create and manage an ENI.

1. Create an ENI.
2. Attach an ENI to an instance or Attach an ENI when you create an instance.
3. Optional. Configure an ENI.
4. Assign secondary private IP addresses.
5. Detach an ENI from an instance.
6. 删除弹性网卡.

Use tags

You can use tags to manage resources to enhance efficiency. You can perform the following steps to use tags:

1. Create or bind a tag.
2. Search for resources by tag.
3. Delete or unbind a tag.

Create and manage launch templates

Launch templates help you create ECS instances that have the same configurations. You can perform the following steps to create a launch template:

1. Create a launch template.
2. 创建实例启动模板的新版本.
3. 删除实例启动模板和版本.

Create and manage deployment sets

Deployment sets help you implement high availability for underlying applications. You can perform the following steps to create and manage a deployment set:

1. Create a deployment set.
2. Create an ECS instance in a deployment set.
3. Change the deployment set of an instance.
4. Delete a deployment set.

Use Cloud Assistant

Cloud Assistant allows you to send remote commands to ECS instances without the need to configure jump servers. You can perform the following steps to use Cloud Assistant:

1. Optional. Manually install and configure the Cloud Assistant client on some ECS instances. For more information, see Install the Cloud Assistant client.
2. Create a command.
3. Run commands.
4. Query execution results and fix common problems.

2.Best practices for instance type selection

To select an instance type, you must know the key features of each instance type. Knowing the features of each instance type allows you to select a variety of instance types to manage scenarios such as insufficient resources, instance families phase-out, and the use of preemptible instances and take full advantages of the elasticity and flexibility of ECS.

Alibaba Cloud ECS select a server first purchase
This topic describes how to select enterprise-level instance families, instead of entry-level instance families (also called shared instance families). For more information about the selection of entry-level instance families, see Overview.

Learn instance families

When you create an ECS instance, you want to select the most cost-effective and stable instance type to suit your performance, prices, and workload needs. Alibaba Cloud provides a variety of instance families suited for different business scenarios with different specifications of vCPUs, memory, network performance, and throughput. An instance family is further divided into multiple instance types. Instance family names are in the ecs. <Instance family> format, and instance type names are in the ecs. <Instance family>.<nx>large format.

- ecs: the product code of ECS.
- <Instance family>: consists of lowercase letters and digits.
  - Lowercase letters are abbreviations of the performance field of the instance family. Some of the abbreviations are described as follows:
    - c: compute optimized (computational)
    - g: general purpose (general)
    - r: memory optimized (ram)
    - ne: enhanced network performance (network enhanced)
  - Digits identify the release dates of instance types within an instance family. A larger digit represents a newer generation of the instance family and is cheaper, more effective, and has higher performance.
- <nx>large: the number of vCPUs of the instance type. The larger the digit n is, the more vCPUs the instance type has.

For example, ecs.g6.2xlarge represents an instance type that belongs to the general purpose instance family g6 and has eight vCPUs. g6 is a newer-generation instance family than g5, g4, and sn2ne.

Select instance types

You can use one of the following methods to view details about instance families and types to select an appropriate instance type:

- Instance families: See this topic to learn details about instance families without logging on with an Alibaba Cloud account.
- DescribeInstanceTypes: Call this API operation to obtain the latest performance specifications of instance types. You must log on with an Alibaba Cloud account to perform this operation.

```
aliyun ecs DescribeInstanceTypes --InstanceTypeFamily ecs.g6
```

- Pricing: View the pricing information, latest special offers, and the estimated cost on the pricing page.

Select instance families based on scenarios

The following figure shows certain ECS instance families and the business scenarios for which they are suited.

Select instance families based on applications

If you are using software or applications similar to those in the following figure, select corresponding instance families from the right side of the picture.

Select instance families based on user-created services

If you are using user-created services, select corresponding instance families based on your services and the selection principles.

Application type	Common application	Selection principle	Recommended instance family
Load balancing	NGINX	Connections can be frequently established. • CPU computing power: high. • Memory: not high.	Instance families: c6 and hfc6
RPC	• SOFA • Dubbo	A huge amount of memory is available for network connection-intensive workloads.	Instance families: g6
Cache	• Redis • Memcache • Solo	• CPU computing power: not high. • Memory: high.	• Instance families: r6 and re6 • Block Storage: standard SSDs or enhanced SSDs (ESSDs)
Configuration center	ZooKeeper	A large number of I/O operations generated when applications initiate negotiations can be handled. • CPU computing power: not high. • Memory: not high.	• Instance families: c6 • Block Storage: standard SSDs or enhanced SSDs

Application type	Common application	Selection principle	Recommended instance family
Message queues	• Kafka • RabbitMQ	Disks are preferred for the purpose of message integrity. • CPU computing power: not high. • vCPU-to-memory ratio: 1:1. • Storage: not high.	• Instance families: c6 • Block Storage: standard SSDs or enhanced SSDs
Container orchestration	Kubernetes	The X-Dragon architecture and containers are combined to maximize computing power.	Instance families: ebmc6 and ebmg6
Large table storage	HBase	• Typically, d series instance families are suitable. • If your business requires ultra-high IOPS, you can select i series instance families.	• Instance families: d1, d1ne, and d2s • i2
Database	• MySQL • NoSQL	• If your business requires scalable storage, you can select enhanced SSDs. • If your business is I/O-sensitive, i series instance families are preferred.	• Instance families: c6, g6, and r6 Block Storage: enhanced SSDs • i2
	SQLServer	• Windows features single-channel I/O configuration while high I/O capabilities are required. Therefore, enhanced SSDs are recommended. • The logical and physical sectors of ECS instances are set to 4 KB in size.	• Instance families: c6, g6, and r6 • Block Storage: enhanced SSDs
Text search	Elasticsearch	• Instance types that have high vCPU-to-memory ratios are recommended. • I/O capabilities can meet the requirements for exporting database data in the .es format.	• Instance families: g6 Block Storage: enhanced SSDs • d1, d1ne, and d2s
Real-time computing	• Flink • Blink	You can select general purpose ECS instances attached with disks or select d series instances.	d1, d1ne, and d2s

Application type	Common application	Selection principle	Recommended instance family
Offline computing	• Hadoop • HDFS • CDH	d series instance families are preferred.	d1, d1ne, and d2s

Recommended instance families for scenarios such as common scenarios, game servers, and live streaming

These scenarios are CPU-compute intensive. We recommend that you select an instance type with a relatively balanced CPU-to-memory ratio, typically 1:2, use an ultra disk as the system disk, and use standard SSDs or enhanced SSDs as data disks. For scenarios that require higher network performance such as on-screen video comments, you can select an instance type with higher specifications to improve the packet forwarding rate.

Scenario category	Specific scenario	Recommended instance family	Performance requirement	CPU-to-memory ratio
Common scenarios	Balanced performance applications and backend applications	g series instance families, such as g6	Medium clock speed, compute-intensive	1:4
	Applications with high packet forwarding rates	g series instance families, such as g6	High packet forwarding rates, compute-intensive	1:4
	High-performance computing	c series instance families, such as c6	High clock speed, compute-intensive	1:2
Game applications	High-performance client games	c series instance families, such as c6	High clock speed	1:2
	Mobile or web games	g series instance families, such as g6	Medium clock speed	1:4
Live streaming applications	Video forwarding	g series instance families, such as g6	Medium clock speed, compute-intensive	1:4
	On-screen video comments	g series instance families, such as g6	High packet forwarding rates, compute-intensive	1:4

Recommended instance families for big data scenarios such as Hadoop, Spark, and Kafka

These scenarios have different performance requirements for each node. You must balance the performance of each node, including computing performance, storage throughput, and network performance.

- Management nodes: Select an instance family in the same manner as you would in common scenarios. For more information, see the Recommended instance families for scenarios such as common scenarios, game servers, and live streaming section.
- Compute nodes: Select an instance family in the same manner as you would in common scenarios. For more information, see the Recommended instance families for scenarios such as common scenarios, game servers, and live streaming section. You must select instance types based on the cluster size. For example, you can select ecs.g6.4xlarge for clusters with less than 100 nodes and ecs.g6.8xlarge for clusters with more than 100 nodes.

  > **Note** Compute nodes can be billed on preemptible instances to improve cost-effectiveness. For more information, see Overview.

- Data nodes: require high storage throughput, high network throughput, and balanced CPU-to-memory ratios. We recommend that you use the d series big data instance family. For example, you can select ecs.d1ne.6xlarge for MapReduce and Hive, or ecs.d1ne.8xlarge for Spark and MLib.

Recommended instance families for scenarios such as databases, caches, and search scenarios

Typically, these scenarios require that the CPU-to-memory ratio is greater than 1:4. Some software in these scenarios is sensitive to latency and storage I/O capabilities. We recommend that you select instance families whose unit price of memory is more reasonable.

Scenario category	Specific scenario	Recommended instance family	CPU-to-memory ratio	Data disk
Relational databases	High performance and dependent on high availability in the application layer	i series instance families	1:4	Local SSDs, ultra disks, and standard SSDs
	Small and medium-sized databases	g series instance families, or other instance families with CPU-to-memory ratios of 1:4	1:4	Ultra disks and standard SSDs
	High-performance databases	r series instance families	1:8	Ultra disks and standard SSDs

Scenario category	Specific scenario	Recommended instance family	CPU-to-memory ratio	Data disk
Distributed caches	Medium memory usage scenarios	g series instance families, or other instance families with CPU-to-memory ratios of 1:4	1:4	Ultra disks and standard SSDs
	High memory usage scenarios	r series instance families	1:8	Ultra disks and standard SSDs
NoSQL databases	High performance and high availability in the application layer	i series instance families	1:4	Local SSDs, ultra disks, and standard SSDs
	Small and medium-sized databases	g series instance families, or other instance families with CPU-to-memory ratios of 1:4	1:4	Ultra disks and standard SSDs
	High-performance databases	r series instance families	1:8	Ultra disks and standard SSDs
ElasticSearch	Small clusters that depend on disks to ensure high data availability	g series instance families, or other instance families with CPU-to-memory ratios of 1:4	1:4	Ultra disks and standard SSDs
	Large clusters and dependent on high availability in the application layer	d series instance families	1:4	Local SSDs, ultra disks, and standard SSDs

Take databases as an example. Traditionally, business systems directly connect to online transaction processing (OLTP) databases and data redundancy is mainly achieved by RAID. However, if you use ECS provided by Alibaba Cloud, you can flexibly deploy both lightly and heavily loaded databases.

- Lightly loaded databases: use enterprise-level instance families with disks, which are more cost-effective.
- Heavily load databases: require high storage IOPS and low read/write latency. We recommend that you use i series instance families with local SSDs. The local SSDs are high I/O NVMe SSDs that can meet the requirements of large heavily-loaded databases.

Recommended instance families for scenarios such as deep learning and image processing

In these scenarios, applications require high-performance GPU accelerators. The following section describes recommendations for the GPU-to-CPU ratio:

- Deep learning training: The recommended GPU-to-CPU ratio is 1:8 to 1:12.
- General-purpose deep learning: The recommended GPU-to-CPU ratio is 1:4 to 1:48.
- Image recognition and reasoning: The recommended GPU-to-CPU ratio is 1:4 to 1:12.
- Speech recognition and synthesis reasoning: The recommended GPU-to-CPU ratio is 1:16 to 1:48.

Check and adjust your selection

After you select an instance type and start using the instance, we recommend that you check whether the instance type is suitable based on the performance monitoring information.

If you select the ecs.g6.xlarge instance type and find that the CPU utilization is low, we recommend that you log on to the instance to check whether the memory usage is high. If the memory usage is high, you can change to another instance family with a more suitable CPU-to-memory ratio. For more information, see the following topics:

- ECS monitoring service
- View the monitoring data of a cloud disk
- Host monitoring overview

For scenarios such as insufficient resources, instance families phase-out, change of the current instance family to a more cost-effective one, you can upgrade or downgrade the instance configurations based on the instance family features. The following table lists some recommendations for instance configuration changes. For more information, see Overview of instance upgrade and downgrade and Change instance types of instances.

Current instance families	Preferred target instance family	Alternative target instance family
sn1 and sn2	• c6 • g6 • r6	• c5 and sn1ne • g5 and sn2ne • r5 and se1ne
c4	hfc6 and c6	hfc5 and c5
ce4	r6	r5 and se1ne
cm4	hfc6	hfc5 and g5
n1, n2, and e3	• c6 • g6 • r6	• c5 and sn1ne • g5 and sn2ne • r5 and se1ne

Current instance families	Preferred target instance family	Alternative target instance family
• t1 • s1, s2, and s3 • m1 and m2 • c1 and c2	• c6 • g6 • r6	• c5 and sn1ne • g5 and sn2ne • r5 and se1ne

References

For more information about application scenarios, see Selection of enterprise-level instance types.

3.Provisioning methods of ECS instances

ECS instances can be provisioned by using multiple methods, such as individual provisioning, batch provisioning, high-availability deployment, and automatic cluster creation. These provisioning methods can be made in the ECS console or through API operation calls to meet your instance creation requirements in various scenarios.

Manually create instances

Scenarios: create multiple ECS instances of the same instance type and billing method in the same zone.

Creation methods:

- Use the ECS console:
  - Create an instance by using the provided wizard

    You can use the Wizard to select configurations.
  - Create an ECS instance by using a custom image

    You can use custom images under your Alibaba Cloud account and use the Wizard to select configurations.
  - Purchase an ECS instance of the same configuration

    You can use Wizard to confirm configurations of existing instances.
  - Create an instance by using a launch template

    You can use a launch template and use the Wizard to confirm configurations.
- Call the RunInstances operation:
  - RunInstances
  - Batch create ECS instances

Number of instances that can be created: If you create instances in the ECS console, the number of instances that can be created at a time depends on the usage amount of your ECS instances. If you create instances by calling the RunInstances operation, you can create 1 to 100 instances per call.

The following figure shows the instance lifecycle when you create ECS instances in the console or by calling the RunInstances operation:

You can also call the CreateInstance operation to create an ECS instance. However, the instance enters the Stopped state after the instance is created, and you must manually start the instance.

Deploy instances on multiple physical servers to improve availability of the cluster (deployment sets)

Scenarios: deploy multiple ECS instances across physical servers. The instance creation mode is suitable for providing computing power to applications that require high availability and disaster recovery of underlying databases.

Creation method: create a deployment set first and then specify the deployment set when you create ECS instances. You can create instances in the ECS console or by calling the RunInstances or CreateInstance operation.

Number of instances that can be created: It depends on the creation method. If you create instances in the console or by calling the RunInstances operation, you can create one to seven instances at a time. If you create instances by calling the CreateInstance operation, you can create only one instance at a time.

Limits:

- A maximum of seven ECS instances can be created for each deployment set in a zone.
- Only specific ECS instance types are supported. For more information, see Overview.
- Billing methods: Subscription and pay-as-you-go are supported. Preemptible instances are not supported.

Procedure:

- Use the ECS console:
  i. Create a deployment set
  ii. Create an ECS instance in a deployment set
- Call API operations:
  i. CreateDeploymentSet
  ii. RunInstances or CreateInstance

Automatically create instance clusters at minimal costs (auto provisioning)

Scenarios: deploy instance clusters of different billing methods and instance types in different zones with one click. The instance creation mode is suitable for scenarios that require stable computing power to be provisioned quickly and preemptible instances to reduce costs.

Creation method: create an auto provisioning group, which automatically creates multiple ECS instances at a time.

Number of instances that can be created: 1 to 1,000 instances can be created for each auto provisioning group.

Limits: Pay-as-you-go instances and preemptible instances are supported. Subscription instances are not supported.

Procedure:

- Use the ECS console: Create an auto provisioning group
- Call API operations: CreateAutoProvisioningGroup

Automatically create and release instances (Auto Scaling)

Scenarios: maintain instance clusters of different billing methods and instance types in different zones. The creation mode is suitable for scenarios where service workloads fluctuate from time to time.

Creation method: create a scaling group and a trigger task. The scaling group automatically creates or releases multiple ECS instances at a time.

Number of instances that can be created:

- A maximum of 1,000 ECS instances can be created within a scaling activity.
- A maximum of 1,000 ECS instances can be created by using one scaling group.

Limits: Pay-as-you-go instances and preemptible instances are supported. You can manually add an existing subscription instance to a scaling group, but cannot create a subscription instance in the scaling group.

Procedure:

- Use the ECS console:
  i. Create a scaling group and a scaling configuration
  ii. Automatically add ECS instances or Automatically remove ECS instances
- Call API operations:
  i. CreateScalingGroup
  ii. CreateScalingConfiguration
  iii. CreateScalingRule
  iv. CreateScheduledTask

Auto Scaling also provides simplified functions which improves provisioning efficiency, shortening the time from requirement to provisioning. For example, you can configure an ECS instance to automatically associate itself with an SLB instance and an RDS instance. You can also configure lifecycle hooks to perform custom operations on the instance. You can use Auto Scaling to obtain the scalability that meets your business needs. For information about best practices of Auto Scaling, see:

- Build a scalable web application
- Use Auto Scaling to reduce costs
- Deploy a high-availability compute cluster

4.Best practices for cost management

This topic describes the cost components and benefits of ECS and the recommended cost management solutions. These solutions can maximize cost effectiveness while ensuring rapid business development.

Cost components

The cost of using ECS consists of the following components:

- Ownership cost: the costs of resources and resource plans.
- O&M cost: the labor costs incurred when you use ECS.

Cost benefits of data migration to cloud

To build an Internet data center (IDC), you must consider direct costs such as hardware, network, electricity, and O&M. You must also consider scaling costs from upgrades and capacity increase, and the risk costs from the implementation of data backup and high availability. When you scale out your IDC to meet business needs, the unit cost of resources increases, and the IDC becomes more complex while the fault tolerance of the IDC decreases. Additional costs occur if you select improper models.

Compared with operating your own IDCs, cloud resources do not require investments such as hardware, physical environments, and labors. The unit cost of cloud resources is relatively linear and all resources allow on-demand access and delivery. Cloud resources also support multiple billing methods to further optimize costs.

Cost optimization suggestions

We recommend that you manage ECS costs from the following aspects:

- Optimize resources
- Upgrade instance families
- Regular cost saving measures
- Implement automated O&M

Optimize resources

If you find resources with high costs, you can monitor the resources from multiple aspects, determine the reasons for the high costs, and then take targeted optimization measures.

1. Monitor resource usage.
    i. Monitor the usage of resources such as CPU, memory, disks, and bandwidth. Assess whether the current configuration is higher than required.
    ii. Monitor idle resources to avoid waste. Idle resources include instances that are upgraded but not restarted, reserved instances that do not match pay-as-you-go instances, disks that are not attached to instances, and EIPs that are not associated with instances.
    iii. Monitor resource usage periods. If you use resources such as pay-as-you-go instances and disks for an extended period of time, we recommend that you use subscription resources or resource plans to reduce cost.

iv. Monitor the lifecycle of resources. Pay attention to the expiration dates of subscription resources such as subscription instances, reserved instances, and storage capacity units. Renew the resources in a timely manner.

2. Select appropriate instance types.

   Instance types have significant impacts on ECS costs. You need to select the most cost-effective instance type and adjust the instance quantity based on business scenarios. This way, you can maximize resource utilization and reduce costs while meeting business needs.

   For example, you are using 60 d1ne.14xlarge instances for short videos. The instance monitoring result shows that the memory usage is reasonable but the CPU utilization is relatively low. The specifications of instance families show that the CPU-to-memory ratio is 1:4 for d1ne and 1:5.5 for d2.

   To reduce the CPU-to-memory ratio to increase CPU utilization, you can use 85 d2s.8xlarge instances to replace 60 d1ne.14xlarge instances. The instance type is downgraded from 14xlarge to 8xlarge, reducing costs by about 23%.

   For more information about instance configurations, see Best practices for instance type selection.

3. Combine multiple billing methods.

   Different types of business have different requirements for the resource usage cycle. Select an appropriate billing method for each type of business and combine billing methods to optimize costs.

   - Subscription and reserved instances are used for stable business workloads.
   - Pay-as-you-go instances are used for stateful and dynamic business workloads.
   - Preemptible instances are used for stateless and fault-tolerant business workloads.

4. Use dedicated hosts to reuse ECS instance resources.

   For scenarios that have minimal requirements for CPU stability, such as development and test environments, you can use CPU overprovisioned dedicated hosts to deploy more ECS instances of the same specifications and reduce the unit deployment cost.

   Stopped ECS instances that are deployed on dedicated hosts do not occupy resources. Therefore, you can stop some ECS instances during off-peak hours of the production environment. Then you can use other resources in the production environment to run test tasks whose periods are predictable, such as offline computing and automated tests.

Upgrade instance families

Hardware such as processors is continuously updated to improve performance while reducing costs. ECS is also updated to provide you with more cost-effective services.

Later instance types are more cost-effective than earlier instance types. For example, the following table describes the difference between g5.2xlarge and g6.2xlarge in performance and price.

Performance	Price
• The integer computing performance is improved by 40%. • The floating-point computing performance is improved by 30%. • The memory bandwidth is increased by 15%. • The memory idle latency is decreased by 40%. • The internal bandwidth is increased by 220%.	• The cost of annual subscription ECS resources is reduced by 6%. • The cost of pay-as-you-go ECS resources is reduced by 43%.

To ensure that you can use the next-generation instance types in time, we recommend that you:

- Design robust applications that can run on different instance types.
- Assess whether to upgrade instance types based on the updates in the official website of Alibaba Cloud.

Examples of instance family upgrade

By using the following upgrade scheme, you can improve business performance while ensuring same CPU and memory configurations and save at least 15% of instance costs.

Current instance families	Preferred target instance family	Alternative target instance family
sn1 and sn2	• c6 • g6 • r6	• c5 and sn1ne • g5 and sn2ne • r5 and se1ne
c4	hfc6 and c6	hfc5 and c5
ce4	r6	r5 and se1ne
cm4	hfc6	hfc5 and g5
n1, n2, and e3	• c6 • g6 • r6	• c5 and sn1ne • g5 and sn2ne • r5 and se1ne
• t1 • s1, s2, and s3 • m1 and m2 • c1 and c2	• c6 • g6 • r6	• c5 and sn1ne • g5 and sn2ne • r5 and se1ne

Regular cost saving measures

Cloud resources are on-demand and save you the investment and cost of operating your own IDC. However, you need to constantly optimize costs in your daily work to obtain the ideal results. You can refine the following typical operations to make a practical scheme.

- Hold regular cost meetings. Review budget implementation with cost-related parties such as finance and R&D teams, evaluate optimization results, and improve optimization strategies on a regular basis.
- Enforce the use of tags. Mark resources with tags of business, environment, and owners to track daily costs.
- Classify resources and select appropriate usage methods. For example, pay-as-you-go instances are preferred for the deployment of the development and testing environments for short-term projects, and the instances are released in a timely manner after the projects are complete.
- Avoid idle resources. Check resource usage on a regular basis and determine the notification and disposal workflows of idle resources.
- Renew resources in a timely manner. Apply for a budget for subscription resources in advance to avoid the additional cost of purchasing and deploying new resources after existing resources are released upon expiration.

Implement automated O&M

Alibaba Cloud provides a variety of O&M services to help you improve O&M efficiency and reduce O&M labor costs. For example:

- Auto Scaling: constantly maintains instance clusters of different billing methods and instance types in different zones. This service is ideal for the scenarios where business workloads fluctuate from time to time.
- Auto Provisioning: quickly deploys instance clusters of different billing methods and instance types in different zones. This service is ideal for the scenarios where stable computing power needs to be provisioned quickly and preemptible instances are needed to reduce costs.
- Resource Orchestration Service: Deploys and maintains stacks that contain multiple cloud resources and dependencies. This service is ideal for the scenarios where delivery of an integrated system or environment clone is required.

5.Security

5.1. Best practices of the security group (part 1)

This article introduces how to configure the inbound rules of security groups.

Like a virtual firewall, a security group controls network access for one or more ECS instances. It is an important means of security isolation. When creating an ECS instance, you must select a security group. You can also add security group rules to control outbound and inbound access for all ECS instances in the same security group.

Before configuring the inbound rules for a security group, you should have learnt about the following information:

- Security group restrictions
- Default security group rules
- Set the inbound access of a security group
- Set the outbound access of a security group

General suggestions for security group practices

Before you work with security groups, read the following suggestions:

- The most important rule: A security group should be used as a whitelist.
- The "minimum authorization" principle should be observed when you configure the inbound or outbound rules for applications. For example, you can allow a specific port (such as port 80).
- It is not recommended to use one security group to manage all applications, because requirements must be different at different layers.
- For distributed applications, different security groups should be used for different application types. For example, you should use different security groups for the Web, Service, Database and Cache layers to apply different inbound/outbound rules and permissions.
- There is no need to set a separate security group for every instance, as this would unnecessarily add to management costs.
- VPC should be preferred.
- Do not assign Internet addresses to resources that require no Internet access.
- Keep the rules of each security group as concise as possible. A single instance can join up to five security groups, and a security group can contain up to 100 security group rules, so an instance may be subject to hundreds of security group rules at the same time. You can aggregate all the assigned security rules to determine whether inbound or outbound traffic is permitted or not. However, overly complicated rules for a single security group can increase management complexity. For this reason, it is recommended to keep the rules of each security group as concise as possible.
- The ECS console allows you to clone a security group and security group rules. If you want to modify an active security group and its rules, you should clone the security group and modify the cloned security group, avoiding any impacts on online applications.

> **Note** Adjusting inbound or outbound rules of active security groups can be risky. Therefore, do not update those rules at will unless you know what you are doing.

Set inbound access rules of security groups

The following are some suggestions about inbound rules of a security group.

Do not use the 0.0.0.0/0 inbound rule

It is a common mistake to permit all inbound access without any restrictions. Using 0.0.0.0/0 means that all ports are open to external access. This is extremely insecure. The correct practice is to deny external access to all the ports first. Whitelist items should be configured for security groups. For example, if you need to expose web services, you should only open common TCP ports such as 80, 8080 and 443 by default. All other ports should be disabled.

```
{ "IpProtocol" : "tcp", "FromPort" : "80", "ToPort" : "80", "SourceCidrIp" : "0.0.0.0/0", "Policy": "accept"} ,
{ "IpProtocol" : "tcp", "FromPort" : "8080", "ToPort" : "8080", "SourceCidrIp" : "0.0.0.0/0", "Policy": "accept
"} ,
{ "IpProtocol" : "tcp", "FromPort" : "443", "ToPort" : "443", "SourceCidrIp" : "0.0.0.0/0", "Policy": "accept"} ,
```

Disable unneeded inbound rules

If your current inbound rules include 0.0.0.0/0, review the ports and services that must be exposed for your applications. If you do not want some ports to directly provide services for external applications, add denial rules for them. For example, if you have installed MySQL database services on the server, port 3306 should not be exposed to the Internet by default. You can add a denial rule, as shown below. Set the priority value to 100, which is the lowest priority.

```
{ "IpProtocol" : "tcp", "FromPort" : "3306", "ToPort" : "3306", "SourceCidrIp" : "0.0.0.0/0", "Policy": "drop",
Priority: 100} ,
```

This setting prevents any other ports from accessing port 3306. However, this can block normal service requests as well. For this reason, you can authorize resources of another security group for inbound access.

Authorize another security group for inbound access

Different security groups adopt inbound and outbound rules in accordance with the minimum authorization principle. Different application layers should use different security groups with corresponding inbound and outbound rules.

For example, different security groups are configured for distributed applications. However, directly authorizing IP addresses or CIDR network segments can be very difficult as different security groups cannot intercommunicate on the Internet. In this situation, you can authorize all resources of another security group to be directly accessible. For example, sg-web and sg-database security groups are created respectively for the Web and Database layers of your applications. In sg-database, you can add the following rule to authorize all resources in the sg-web security group to access port 3306.

```
{ "IpProtocol" : "tcp", "FromPort" : "3306", "ToPort" : "3306", "SourceGroupId" : "sg-web", "Policy": "accep
t", Priority: 2} ,
```

Authorize another CIDR for inbound access

In classic networks, controlling network segments is difficult and you are recommended to use security group IDs to authorize inbound rules.

In VPC networks, you can plan IP addresses on your own and use different VSwitches to set different IP domains. Therefore, in VPC networks, you can deny any access by default but authorize access for your own VPC, namely directly authorizing trusted CIDR network segments.

```
{ "IpProtocol" : "icmp", "FromPort" : "-1", "ToPort" : "-1", "SourceCidrIp" : "10.0.0.0/24", Priority: 2} ,
{ "IpProtocol" : "tcp", "FromPort" : "0", "ToPort" : "65535", "SourceCidrIp" : "10.0.0.0/24", Priority: 2} ,
{ "IpProtocol" : "udp", "FromPort" : "0", "ToPort" : "65535", "SourceCidrIp" : "10.0.0.0/24", Priority: 2} ,
```

Steps and instructions for changing security group rules

Changing security group rules can interrupt network communication among instances. To prevent required network communication from being impacted, try to permit required instances with the method below and then execute security group policies to narrow down your changes.

> **Note** After narrowing down the changes, check that service applications are running correctly before performing other required changes.

- Create a new security group, add instances that need mutual access to it, and then perform the changes.
- If the authorization type is **Security Group**, add the bound security group IDs of peer instances that require intercommunication into the authorization rules of the security group.
- If the authorization type is **CIDR**, add Intranet IP addresses of peer instances that require intercommunication into the authorization rules of the security group.

For detailed instructions, see Add security group rules.

5.2. Best practices of the security group (part 2)

This document introduces the following:

- Authorize and revoke security groups.
- Join and leave security groups.

Alibaba Cloud provides two types of networks, namely **classic networks** and **VPC networks**. They support different security group rules:

- For classic networks, you can set the following rules: intranet inbound, intranet outbound, Internet inbound and Internet outbound.
- For VPC networks, you can set the following rules: intranet inbound and intranet outbound.

Basic knowledge of intranet communication for security groups

Firstly, learn about the following points about intranet communication for security groups:

- By default, only the ECS instances in the same security group can access each other. In other words, the instances of the same account in different security groups are inaccessible to each other on the intranet. This applies to both classic and VPC networks. Therefore, the ECS instances in classic networks are secure over the intranet.
- If you have two ECS instances in different security groups, and you want them to be inaccessible to each other over the intranet but they are actually accessible, you should check the intranet rule settings of your security group. If the intranet rules include the following items, you are recommended to reconfigure them.
  - Allow all ports;
  - The authorized object is a CIDR segment (SourceCidrIp): *0.0.0.0/0* or *10.0.0.0/8*. For classic networks, the above rules can expose your intranet to external access.
- If you want to implement network intercommunication among the resources of different security groups, you should adopt security group authorization. For intranet access, you are recommended to adopt the source security group authorization, instead of CIDR segment authorization.

Attributes of security rules

Security rules mainly describe different access permissions with the following attributes:

- Policy: authorization policies. The parameter value can be *accept* or *drop*.
- Priority: priority levels. The priority levels are sorted by creation time in descending order. The rule priority ranges from 1 to 100. The default value is 1, which is the highest priority. A greater value indicates a lower priority.
- NicType: network type. In security group authorization (namely SourceGroupId is specified while SourceCidrIp is not), you must specify NicType as *intranet*.
- Description:
  - IpProtocol: IP protocol. Values: *tcp*, *udp*, *icmp*, *gre* or *all*. The value "all" indicates all the protocols.
  - PortRange: the range of port numbers related to the IP protocol:
    - When the value of IpProtocol is *tcp* or *udp*, the port range is 1-65535. The format must be "starting port number/ending port number". For example, "1/200" indicates that the port range is 1-200. If the input value is "200/1", an error will be reported when the interface is called.
    - When the value of IpProtocol is *icmp*, *gre* or *all*, the port range is -1/-1, indicating no restriction on ports.
  - If security group authorization is adopted, the SourceGroupId (namely the source security group ID) should be specified. In this case, you can choose to set SourceGroupOwnerAccount based on whether it is cross-account authorization. SourceGroupOwnerAccount indicates the account to which the source security group belongs.
  - If CIDR authorization is adopted, SourceCidrIp should be specified. SourceCidrIp is the source IP address segment, which must be in the CIDR format.

Create a rule to authorize inbound requests

When you create a security group in the console or through the API, the default inbound rule is *deny all*, that is, all the inbound requests are rejected by default. This does not apply to all the situations, so you need to configure inbound rules accordingly.

If you need to enable port 80 on the Internet to provide HTTP services for external applications, do not impose any restrictions on IP network segments but set it to *0.0.0.0/0* in order to allow all inbound requests. For this purpose, you can refer to the following properties where console parameters are outside of brackets and OpenAPI parameters are within brackets (no difference is made if both parameters are the same).

- NIC Type (NicType): Internet (internet). For VPCs, simply enter intranet to implement Internet access through EIP.
- Action (Policy): allow (accept).
- Rule Direction (NicType): inbound.
- Protocol Type (IpProtocol): TCP (tcp).
- Port Range (PortRange): 80/80.
- Authorized Objects (SourceCidrIp): 0.0.0.0/0.
- Priority (Priority): 1.

> **Note** These recommended values apply only to the Internet. For intranet requests, you are not recommended to use CIDR network segments. Please refer to Do not use CIDR or IP authorization for intranet security group rules of classic networks.

Create a rule to deny inbound requests

To deny inbound requests, you only need to configure a denial policy with a low priority. In this way, you can configure another rule with a higher priority to overwrite this rule when needed. For example, the following explains how to deny access to port 6379.

- NIC Type (NicType): Intranet (intranet).
- Action (Policy): forbid (drop).
- Rule Direction (NicType): inbound.
- Protocol Type (IpProtocol): TCP (tcp).
- Port Range (PortRange): 6379/6379.
- Authorized Objects (SourceCidrIp): 0.0.0.0/0.
- Priority (Priority): 100.

Do not use CIDR or IP authorization for intranet security group rules of classic networks

For ECS instances in classic networks, no intranet inbound rules are enabled by default. Always exercise caution for intranet authorization.

> **Note** For the sake of security, it is not recommended to enable any authorization that is based on CIDR network segments.

For elastic computing, intranet IP addresses change frequently and the network segment to which the IP addresses map varies dynamically. For this reason, you are only recommended to authorize intranet access through security groups in classic networks.

For example, if you build a Redis cluster in the sg-redis security group and only permit certain computers (such as those in sg-web) to access the servers of this Redis cluster, you do not need to configure CIDR. Instead, you only need to add an inbound rule to specify relevant security group IDs.

- NIC Type (NicType): Intranet (intranet).
- Action (Policy): allow (accept).
- Rule Direction (NicType): inbound.
- Protocol Type (IpProtocol): TCP (tcp).
- Port Range (PortRange): 6379/6379.
- Authorized Objects (SourceGroupId): sg-web.
- Priority (Priority): 1.

For instances in a VPC, if you have planned an IP address range through multiple VSwitches, you can use the CIDR settings as the security group inbound rules. However, if your VPC network segment is ambiguous, you are recommended to prioritize security groups for inbound rules.

Add ECS instances requiring intercommunication into the same security group

A single ECS instance can join up to five security groups, and the ECS instances in the same security group can intercommunicate over the intranet. If you have created multiple security groups during planning and directly setting multiple security rules is too complicated, you can create a security group and add the instances that require intranet communication to it.

Different security groups may have different network types. More importantly, an ECS instance in a classic network can only join a security group created for classic networks. An ECS instance in a VPC can only join a security group created for the same VPC.

Additionally, you are not recommended to add all the ECS instances into the same security group as this will make the configuration of security group rules quite messy. For a large or medium-sized application, each server group has a different role and it is important to plan inbound and outbound requests in a rational manner.

In the console, you can add an instance to a security group by following the description in Join a security group.

If you are quite familiar with Alibaba Cloud OpenAPI, you can perform batch operations through OpenAPI. For more information, see Query an ECS instance. The corresponding Python snippets are as follows.

```
def join_sg(sg_id, instance_id):
    request = JoinSecurityGroupRequest()
    request.set_InstanceId(instance_id)
    request.set_SecurityGroupId(sg_id)
    response = _send_request(request)
    return response
# send open api request
def _send_request(request):
    request.set_accept_format('json')
    try:
        response_str = clt.do_action(request)
        logging.info(response_str)
        response_detail = json.loads(response_str)
        return response_detail
    except Exception as e:
        logging.error(e)
```

Remove an ECS instance from a security group

If an ECS instance is added to an inappropriate security group, your services may be exposed or blocked. In this case, you can remove the ECS instance from the security group. Before the removal, however, you must ensure that your ECS instance has been added to another security group.

> **Note** You are recommended to perform sufficient tests before the removal as this may cause intercommunication failure between the instance and other instances in the current security group.

The corresponding Python snippets are as follows:

```
def leave_sg(sg_id, instance_id):
    request = LeaveSecurityGroupRequest()
    request.set_InstanceId(instance_id)
    request.set_SecurityGroupId(sg_id)
    response = _send_request(request)
    return response
# send open api request
def _send_request(request):
    request.set_accept_format('json')
    try:
        response_str = clt.do_action(request)
        logging.info(response_str)
        response_detail = json.loads(response_str)
        return response_detail
    except Exception as e:
        logging.error(e)
```

Define reasonable names and tags for security groups

Reasonable names and descriptions for security groups help you quickly identify the meanings of complicated rule combinations. You can change security group names and descriptions as needed.

Also, you can set tags for security groups. You can manage your own security groups by grouping them with tags. To set tags, you can directly configure them in the console or by using APIs.

Delete undesired security groups

Security rules of security groups are like whitelist and blacklist items. Therefore, you are recommended to delete unnecessary security groups to prevent unexpected problems caused by adding an ECS instance to those groups by mistake.

5.3. Best practices of the security group (part 3)

In practice, all instances may be placed in the same security group, thus reducing the configuration workload in the initial period. In the long run, however, interactions of the business systems will become complicated and uncontrollable. When you modify a security group, you will be unable to clearly identify the impact scope of adding or removing a rule.

Rational planning and differentiation of security groups makes it easy to adjust your systems, sort out the services provided by the applications, and arrange applications at different layers. We recommend that you plan different security groups and set different security group rules for different businesses.

Distinguish between different security groups

- **Use different security groups for ECS instances on the Internet and those on the intranet**

  ECS instances that provide Internet services, either through exposure of some ports for external access (such as 80 and 443) or through provision of port forwarding rules (for example, instances configured with forwarding rules for Internet IP address, EIP address or NAT ports), will expose their applications to the Internet.

  For the two scenarios above, the relevant security groups should adopt the strictest rules. We recommend that Internet access should be rejected first. Specifically, all ports and protocols should be disabled by default except the ports needed to provide external services, such as 80 and 443. As the security group only contains the ECS instances that provide Internet access, it is easier to adjust the security group rules.

  For a group of ECS instances that provide Internet access, their responsibilities should be clear and simple, avoiding offering other external services on the same instances. For MySQL, Redis, and more, for example, it is recommended to install such services on ECS instances that disable Internet access, and then enable access to them through security group authorization.

  Assume you have an ECS instance that provides Internet access, which is in the security group SG_CURRENT as the instances of other applications. You can make changes by performing the steps below.

  i. Sort out the ports and protocols exposed by the current Internet services, such as 80 and 443.

  ii. Create a new security group such as SG_WEB and add corresponding ports and rules.

  > Note Action: Allow; Protocol Type: All; Port Range: 80/80; Authorization Objects: 0.0.0.0/0; Action: Allow; Protocol Type: All; Port Range: 443/443; Authorization Objects: 0.0.0.0/0.

  iii. Select the security group SG_CURRENT and add a rule for security group authorization, that is, allowing the resources in SG_WEB to access the resources in SG_CURRENT.

  > Note Action: Allow; Protocol Type: All; Port Range: -1/-1; Authorization Objects: SG_WEB; Priority: Choose from [1-100] according to actual conditions.

  iv. Add ECS_WEB_1 to the new security group. It is an instance that needs to switch its security group.

  a. In the ECS console, select **Security groups**.

  b. Select **SG_WEB > Manage Instances > Add Instance**. Add the instance ECS_WEB_1 to the new security group SG_WEB. Make sure ECS_WEB_1 works normally.

  v. Remove the instance ECS_WEB_1 from the original security group.

  a. In the ECS console, select **Security Groups**.

  b. Select **SG_WEB > Manage Instances > Add Instance**. Select ECS_WEB_1 and remove it from SG_CURRENT. Verify that the traffic and network are in normal condition.

  c. If errors occur, add ECS_WEB_1 back to the security group SG_CURRENT. Check whether the ports of SG_WEB are exposed as expected, and then make adjustments accordingly.

  vi. Make other changes to the security group.

- **Use different security groups for different applications**

  In production environments, different operating systems generally do not belong to the same application group to provide load balancing services. Providing different services means that exposed ports are different from rejected ports. Therefore, it is recommended that instances with different operating systems belong to different security groups.

  For example, TCP port 22 may be exposed for implementing SSH in Linux, while TCP port 3389 may be exposed for implementing remote desktop connection in Windows.

  In addition, for instances that have the same type of images but provide different services, it is recommended to put them into different security groups if they do not need to access each other over the intranet. This facilitates decoupling and future changes to security group rules as the rules can be as simple as possible.

  When planning and adding new applications, you should reasonably organize the security groups apart from dividing different VSwitches to configure subnets. You can use network segments and security groups to distinguish yourself as the service provider or consumer.

  For specific change procedures, see the operations above.

- **Use different security groups for production environments and testing environments**

  To better isolate systems, you may build multiple testing environments and one online environment during actual development. For better network isolation, you need to configure different security policies for different environments, preventing changes to the testing environment from being synchronized to the online environment, which may affect the stability of online services.

  By creating different security groups, you can restrict the access domains of applications and avoid interoperability between the production environment and testing environment. Also, you can create different security groups for different test environments, thus avoiding interference between test environments and improving development efficiency.

Only assign Internet addresses to subnets or instances that require Internet access

Whether it is a classic network or a VPC, rational allocation of Internet addresses facilitates Internet management of the system and reduces the risk of attack. For VPCs, we recommend that you place the IP segments of instances requiring Internet access onto several dedicated VSwitches (subnet CIDR) when creating a VSwitch. This facilitates auditing and differentiation and helps avoid accidental Internet access.

Most distributed applications have different layers and groups. For ECS instances that offer no Internet access, try your best not to provide Internet addresses for them. If there are multiple instances that provide Internet access, we recommend you to configure the Server Load Balancer to distribute traffic of Internet services, thus improving system availability and avoiding a single point of failure.

For ECS instances that require no Internet access, try your best not to assign Internet addresses to them. In VPCs, when your ECS instances need to access the Internet, we recommend you to use the NAT gateway to provide Internet proxy services for ECS instances without Internet addresses in the VPC. By simply configuring the corresponding SNAT rules, you can enable a specific CIDR segment or subnet to access the Internet. For specific configurations, see SNAT. In this way, exposure of services to the Internet can be avoided after Elastic IP (EIP) addresses are allocated when only outbound access is required.

Minimum principle

A security group should work as a whitelist. Therefore, try your best to open and expose as few ports as possible, and allocate as few Internet addresses as possible. Although allocating Internet addresses or binding EIPs makes it easy to access online instances for troubleshooting, it exposes the entire instance to the Internet after all. A safer policy is to manage IP addresses by using the Jump Server.

Use the Jump Server

As the Jump Server has much higher permissions, relevant operations should be well recorded and audited through tools. In addition, it is recommended to choose a dedicated VSwitch for the Jump Server in VPCs, providing the corresponding EIP or NAT port forwarding tables to it.

First, create a dedicated security group SG_BRIDGE by enabling the corresponding port such as TCP 22 in Linux or RDP 3389 in Windows. To restrict the inbound access, you can limit the authorization objects to the Internet egress ports of your company, lowering the probability of being scanned and accessed.

After that, you can add the Jumper Server instance to that security group. In order for that Jumper Server to access other appropriate instances, you can configure appropriate group authorization. For example, add a rule for SG_CURRENT, allowing SG_BRIDGE to access certain ports and protocols.

When you use the Jumper Server for SSH communication, it is recommended to use the SSH key pair for logon, instead of the password.

In summary, reasonable planning of security groups makes it easy for you to expand the applications and makes your system more secure.

5.4. ECS data security best practices

This topic describes the recommended best practices for improving the data security of ECS instances from an O&M perspective.

Best practices

- Back up data regularly
- Design security domains
- Set security group rules
- Set logon passwords
- Improve server port security
- Protect application vulnerabilities
- Collect security information

Back up data regularly

Backing up data regularly reduces the risks of data loss due to system failures, operation errors, and security problems. The snapshot backup function of ECS instances can be used as a means to backup data regularly. To use this function, you must first customize a backup policy by performing one of the following operations:

- Create a snapshot.
- Define automatic snapshot policies and apply automatic snapshot policies to disks.

Next, you need to create automatic snapshots regularly, such as on a daily basis, and then store snapshots for a period of at least seven days. This will significantly increase disaster tolerance, minimizing potential data losses.

Design security domains

You can use VPCs to build private networks, or intranets, which separate servers of different security levels in your enterprise. This prevents servers from affecting each other over an interconnected network.

Set security group rules

You can use security groups to set network access control for one or more ECS instances. By using security groups, you can set firewall policies at the instance level, which allow you to control the access to and from an instance over the network. Specifically, you can restrict the inbound and outbound access on a port, and allow or deny access to and from specific IP addresses.

Consider the following example. By default, the remote access port for Linux ECS instances is port 22. This port does not provide direct access to the Internet. To enable an instance to access the Internet, you can set up a security group to control the access authorization of ECS instances, such as by authorizing the access of fixed IP addresses to the Internet.

To learn more about security groups, see Application cases. If you require additional security options, we recommend that you use third-party VPN products to encrypt the logon data.

Set strong logon passwords

We recommend that you set a strong server password to make your services more secure. To meet the password conventions of most Alibaba Cloud services, we recommend that you set each password to at least eight characters in length and include the following character types: uppercase letters, lowercase letters, numbers, and special characters. We also recommend that you change your password at least once every six months.

Improve server port security

To make sure that your servers are securely connected to the Internet, we recommend that you open only required ports as needed, and that you use only custom ports (port number 30000 or higher). Additionally, we recommend that access control is also implemented on service ports.

For example, we recommend that you restrict database services to an intranet for most general scenarios. However, if your services require access to the Internet, we recommend that you change the original connection port from 3306 to a higher port number and authorize the relevant IP addresses according to your service requirements.

Install a Web Application Firewall (WAF)

We recommend that you install a Web Application Firewall (WAF) to prevent potential attacks due to application vulnerabilities. Installing one makes sure the security and availability of your services in the cloud.

Application vulnerability are security defects that hackers may exploit to illegally access data. Some common application vulnerabilities include SQL injection, XSS attacks, illegal HTTP requests, and unauthorized access to core files.

Application design alone cannot guarantee protection from such vulnerabilities, therefore it is important that you make sure to install a WAF for this purpose.

5.5. Improve ECS instance security

An instance is a virtual machine. Security protection is generally implemented at the instance level to protect the virtual machine against attacks and intrusions. ECS instances also need security protection. You must implement effective security measures in conjunction with the inherent protection of Alibaba Cloud.

Prerequisites

You must have registered an Alibaba Cloud account before you follow the instructions provided in the tutorial. If not, create a new Alibaba Cloud account first.

Context

Lack of security protection for ECS instances may cause many adverse effects. For example,

- DDoS attacks interrupt your business.
- Trojans tamper or attack your webpages.
- Data leak caused by injection affects the normal operation of ECS.

You can use the following methods to improve the security of your ECS instances:

- Configure security groups
- Enable Anti-DDoS Basic
- Access Security Center
- Access Web Application Firewall

Configure security groups

A security group is a virtual firewall that provides stateful packet inspection (SPI). Security groups can serve the following purposes:

- Control access to one or more ECS instances. Security group rules can allow or deny inbound or outbound traffic for ECS instances associated with security groups.
- If security groups are not planned properly or do not contain strict enough rules, they will be at a much greater risk of attack.

To add a rule to the security group to which the ECS instance is bound, perform the following operations:

1. Log on to the ECS console.
2. In the left-side navigation pane, choose **Network & Security > Security Groups**.
3. In the top navigation bar, select a region.
4. Find the target security group. Click **Add Rules** in the **Actions** column.
5. Click **Add Security Group Rule**.
6. In the dialog box that appears, configure the parameters. For example, if you want to allow only a specified IP address to remotely log on to an ECS instance, configure the inbound rules for Internet access. Assume that the instance runs the Linux operating system and you want to allow only the specified IP address to access the instance over port 22.

i. Set an Internet inbound security group rule as follows:

- **Action** to **Allow**.
- **Protocol Type** to **Custom TCP**.
- **Port Range** to *22/22*.
- **Authorization Type** to **IPv4 CIDR Block**.
- **Authorization Object** to the CIDR block allowed to remotely log on to the ECS instance, in the format of x.x.x.x/xx (IP address/subnet mask). For this example, the CIDR block is *10.x.x.x/32*. The priority is *1*.

ii. Set another Internet inbound security group rule as follows:

- **Action** to **Forbid**.
- **Protocol Type** to **Custom TCP**.
- **Port Range** to *22/22*.
- **Authorization Type** to **IPv4 CIDR Block**.
- **Authorization Object** to *0.0.0.0/0*. The priority is *2*.

7. Click **OK**.
   Two rules are created:
   - The allow rule with a priority of 1 takes precedence for traffic from 10.x.x.x to port 22.
   - The deny rule with a priority of 2 takes precedence for traffic from other IP addresses to port 22.

Enable Anti-DDoS Basic

Distributed denial of service (DDoS) attacks use client or server technologies to combine multiple computers into an attack platform and attack one or more targets simultaneously so that the impact of the denial-of-service (DoS) attack is multiplied.

Alibaba Cloud Security can defend against Layer 3 to Layer 7 DDoS attacks, including SYN Flood, UDP Flood, ACK Flood, ICMP Flood, DNS Flood, and HTTP Flood attacks. Anti-DDoS Basic provides up to 5 Gbit/s default DDoS protection free of charge. By default, Anti-DDoS Basic is enabled for ECS. Anti-DDoS eliminates the need to purchase expensive traffic scrubbing devices, while allowing you to maintain the access speed during DDoS attacks. With Anti-DDoS, your bandwidth is guaranteed regardless of other affected users, ensuring the availability and stability of your business. After an ECS instance is created, you can set the scrubbing thresholds. For more information, see Configure a cleaning threshold.

Alibaba Cloud has also launched the Security Credibility program, which provides improved DDoS protection based on a security credit score. Users that meet the criteria can obtain free protection against DDoS attacks up to 100 Gbit/s. In the Anti-DDoS Basic console, you can check your current security credibility score and details, as well as scoring criteria. For more information, see Security Credibility.

Access Security Center

Security Center is a unified security management system that recognizes, analyzes, and warns of security threats in real-time. With security capabilities such as ransomware protection, anti-virus protection, web tamper protection, and compliance assessments, users can automate security operations, responses, and threat tracing to secure cloud and local servers and meet regulatory compliance requirements.

The Security Center agent is a security plug-in installed on your local servers. You must install this agent on your servers before you can enable Security Center features. For more information about how to install the Security Center agent, see Install the Security Center agent.

> **Note** When you purchase an ECS instance, you can select the **Security Enhancement** check box to automatically install the agent and activate Security Center Basic edition.

The Basic edition of Security Center is available by default. The Basic edition only scans for the following risks: unusual logons to servers, vulnerabilities, and configuration risks in cloud services. To use advanced features such as vulnerability fixing and virus detection and removal, you must log on to the Security Center console.

Access Web Application Firewall

Web Application Firewall (WAF) is implemented based on the big data capabilities of Apsara Stack Security. This module protects web applications against common attacks reported by OWASP, such as SQL injections, XSS, vulnerability exploits in web server plugins, trojan attacks, and unauthorized access. WAF blocks malicious visits to avoid data from being compromised and ensure the security and availability of your websites.

WAF has the following benefits:

- WAF can handle various web application attacks to ensure web security and availability of a website without installing any software or hardware or modifying website configuration and code. In addition to powerful web protection capabilities, WAF can customize protection for specific websites. WAF is used to protect web applications in fields such as finance, e-commerce, O2O, Internet Plus, gaming, governments, and insurance.
- Without WAF, you may be vulnerable to web intrusions such as data leaks, HTTP floods, and trojans.

For more information about how to access WAF, see Deploy WAF.

Alibaba Cloud provides multiple security services to safeguard ECS instances. You can choose appropriate methods as needed to enhance systems and data protection, prevent intrusion into ECS instances, and ensure stability and reliability.

5.6. Configure interconnection of instances over the classic network

A security group functions similar to a firewall for an instance. To ensure security, you must follow the principle of least privilege when setting security group rules for instances. This topic describes four recommended methods to configure interconnection of instances over the classic network.

Prerequisites

An Alibaba Cloud account is created. To create an Alibaba Cloud account, go to the account registration page.

Method 1: Authorize access to IP addresses

- Scenario: This method is applicable to interconnection between a small number of instances over the internal network.
- Advantages: Authorizing access to IP addresses involves simple and clear security group rules.
- Disadvantages: When attempting to interconnect a large number of instances over the internal network, you will be limited by the 200 security group rule quota and be burdened by a high maintenance workload.

Procedure:

1. Find the target instance and click its ID. In the left-side navigation pane, click **Security Groups**.
2. Find the target security group and click **Add Rules** in the Actions column.
3. Click the **Inbound** tab. Click **Add Security Group Rule**.
4. Set the security group rule as follows:
   - **Action**: Allow.
   - **Protocol Type**: Select the protocol type as needed.
   - **Port Range**: Set the port range as needed. The format is *start port number/end port number*.
   - **Authorization Type**: IPv4 CIDR Block.
   - **Authorization Object**: Enter the desired private IP addresses of the instances to be interconnected. The format must be *a.b.c.d/32*. Note that the subnet mask must be */32*.

Method 2: Add instances to the same security group

- Scenario: If your application architecture is relatively simple, you can add all target instances to the same security group. Such instances need no special rules as they can access each other over the internal network by default.
- Advantages: Security group rules are simple and clear.
- Disadvantage: This method is only applicable to a simple network architecture. When the network architecture is adjusted, the authorization method must be modified accordingly.

For more information about the procedure, see Add an ECS instances to a security group.

Method 3: Add instances to a security group created solely for interconnection

- Scenario: You can add the target instances to a dedicated security group for interconnection. This method is recommended for a network architecture with multiple layers of applications.
- Advantages: This method is easy to implement and allows you to quickly establish interconnection between instances. The method is applicable in complicated network architectures.
- Disadvantages: The instances are added to multiple security groups and the security group rules may be complex.

Procedure:

1. Create a new security group with a name like security group for interconnection. No rules are required for the new security group.
2. Add the target instances to the new security group. The instances are then interconnected over the internal network by default.

Method 4: Authorize security groups

- Scenario: You can add the target instances to a dedicated security group for interconnection. This method is recommended for a network architecture with multiple layers of applications.
- Advantages: This method is easy to implement and allows you to quickly establish interconnection between instances. The method is applicable to complicated network architectures.
- Disadvantages: The instances are added to multiple security groups and the security group rules may be complex.

Procedure:

1. Find the target instance and click its ID. In the left-side navigation pane, click **Security Groups**.
2. Find the target security group and click **Add Rules** in the Actions column.
3. Click the **Inbound** tab. Click **Add Security Group Rule**.
4. Set the security group rule as follows:
   - **Action**: Allow.
   - **Protocol Type**: Select the protocol type as needed.
   - **Port Range**: Set the port range as needed.
   - **Authorization Type**: Security Group.
   - **Authorization Object**:
     - If you select the **Allow Current Account** check box in the Authorization Type field, select the security group IDs of the peer instances for interconnection over the internal network in the **Authorization Object** field based on your networking requirements.

       
     - If you select the **Allow Other Accounts** check box in the Authorization Type field, enter the security group IDs of the peer instances in the **Authorization Object** field. Enter the peer account ID in the **Account ID** field. You can query the account ID by choosing **Account Management > Security Settings**.

Suggestions

If you determine that an inappropriate level of access has been granted through the applied security group, we recommend that you downgrade the level of access through the following procedure.

In the preceding figure, **Delete 0.0.0.0** means deleting the original security group rule that allows inbound traffic from *0.0.0.0/0*.

If one or more security group rules are improperly configured, the interconnection between your instances may be affected. We recommend that you back up security group rules before changing them so that you can easily recover the rules.

A security group maps the role of an instance in the overall application architecture. We recommend that you plan the firewall rules based on the application architecture. For example, in a common three-tier Web application architecture, you can plan three security groups and associate them to instances deployed with applications or databases as follows:

- Web layer security group: allows port 80.
- Application layer security group: allows port 8080.
- Database layer security group: allows port 3306.

5.7. Modify the default remote port of an instance

This topic describes how to modify the default remote ports of Windows and Linux instances.

Prerequisites

An Alibaba Cloud account is created. To create an Alibaba Cloud account, go to the account registration page.

Modify the default remote port of a Windows instance

This section uses Windows Server 2012 as an example to describe how to modify the default remote port of a Windows instance.

1. Establish a remote connection and log on to the Windows instance. For more information, see Connect to a Windows instance from the console.
2. Modify the value of the **PortNumber** registry subkey.
   i. Press the shortcut keys *Win + R* to open the **Run** dialog box.
   ii. Enter *regedit.exe* and press the Enter key to open the registry editor.
   iii. In the left-side navigation pane, choose **HKEY_LOCAL_MACHINE > System > CurrentControlSet > Control > Terminal Server > WinStations > RDP-Tcp**.
   iv. Find and right-click **PortNumber** in the list, and choose **Modify...** from the shortcut menu.
   v. In the Edit DWORD (32-bit) Value dialog box, enter a new remote port number in the **Value data** field. This example uses 3399 as the port number. Select **Decimal** in the **Base** section and click **OK**.

   
3. (Optional)If you have enabled the firewall, add the new remote port number to the firewall whitelist and allow connections to the port.
4. Restart the instance in the ECS console. For more information, see Restart an instance.
5. Add security group rules for the instance to allow connections to the new remote port. For more information, see Add security group rules.
6. Establish a remote connection to the instance. Add a colon (:) followed by the new remote port number to the end of the remote IP address to connect to the instance. For example, 192.168.1.2:3399.

> **Note** If you use Mac Remote Desktop Connection to connect to the instance, you can connect only through the default port 3389 after the remote port is modified.

Modify the default remote port of a Linux instance

This section uses CentOS 6.8 as an example to describe how to modify the default remote port of a Linux instance.

> **Note** Add a new default remote port without modifying port 22. This allows you to use the port 22 to log on to the instance if you fail to connect to the instance through the new remote port.

1. Establish a remote connection and log on to the Linux instance. For more information, see Connect to a Linux instance from the console.
2. Run the **vim /etc/ssh/sshd_config** command.
3. Press the `I` key to enter the editing state. Add a new remote port such as port 1022 in this example. Enter `Port 1022` under `Port 22`.
4. Press the `Esc` key, enter **:wq**, and then exit the editing state.
5. Run the following command to restart the instance. After the instance is restarted, you can log on to the Linux instance by using SSH through both port 22 and port 1022.

   ```
   /etc/init.d/sshd restart
   ```

6. Configure the firewall. If your system version is earlier than CentOS 7 and you have enabled the default firewall iptables, note that iptables does not block access traffic by default. You need to run the **iptables -A INPUT -p tcp --dport 1022 -j ACCEPT** command to configure the firewall. Then, run the **service iptables restart** command to restart the firewall.

   > **Note** By default, firewalld is installed for CentOS 7 and later. If you have enabled firewalld.service, run the `firewall-cmd --add-port=1022/tcp --permanent` command to allow traffic on TCP port 1022. If `success` is returned, traffic on TCP port 1022 is allowed.

7. Add security group rules for the instance to allow connections to the new remote port. For more information, see Add security group rules.
8. Use an SSH client to connect to the new remote port to check whether the remote port is modified.
   i. Enter the new remote port number in the **Port** field, which is 1022 in this example.

   ii. If you can connect to the instance through port 1022, run the **vim /etc/ssh/sshd_config** command to delete port 22.
   iii. Run the **/etc/init.d/sshd restart** command to restart the instance and the changes take effect after the instance restarts. You can start to use the new port to log on to the instance.

5.8. Use logs in Windows instances

Logs monitor events occurring in the system and record hardware, software, and system issues. When an instance is attacked or an issue occurs on an application, you can locate the critical problems based on logs. This improves work efficiency and instance security. This topic uses Windows Server 2012 R2 as an example to describe how to use and analyze four types of logs.

Prerequisites

You must have registered an Alibaba Cloud account before you follow the instructions provided in the tutorial. If not, create a new Alibaba Cloud account first.

Context

Windows logs can be further divided into system logs, application logs, security logs, and application and service logs.

View logs in Windows Event Viewer

Perform the following steps to open Event Viewer:

1. Click **Start** and open the **Run** dialog box.
2. Run the `eventvwr` command in the dialog box to open **Event Viewer**.
3. View the following four types of logs in **Event Viewer**:

   > **Note** For IDs of all error logs found by using log-viewing methods described in this topic, you can find the corresponding solutions in the Microsoft Knowledge Base.

   - System logs

     In the left-side navigation pane, choose **Windows Logs** > **System** to view system logs.

     System logs contain events recorded by Windows system components. For example, the failure of a driver or other system components to load during startup is recorded in the system log.

     The types of events recorded by system components are predetermined by Windows.

   - Application logs

     In the left-side navigation pane, choose **Windows Logs** > **Application** to view application logs.

     Application logs contain events logged by applications. For example, a database program can record a file error in the application log.

     The types of the events recorded in application logs are determined by developers.

   - Security logs

     In the left-side navigation pane, choose **Windows Logs** > **Security** to view security logs.

     Security logs contain valid and invalid logon attempts and events related to resource use, such as creating, opening, or deleting files or other objects.

     The types of the events recorded in security logs are determined by administrators. For example, if logon auditing is enabled, the security logs will record logon attempts.

- Application and service logs

  An application and service log is a new type of event log. Application and service logs contain events from a single application program or component rather than events that can affect the whole system.

Modify the log path and back up logs

By default, logs are stored on the system disk. The maximum size of logs is 20 MB. If the limit is exceeded, previous events will be overwritten. You can modify the log path as needed.

Perform the following steps to modify the log path and back up logs:

1. In the left-side navigation pane of **Event Viewer**, click **Windows Logs**.
2. In the right-side list, right-click a log name and choose **Properties** from the shortcut menu.
3. In the **Log Properties** dialog box that appears, modify the following parameters:
   - Log path
   - Maximum log size (KB)
   - When maximum event log size is reached:

5.9. Best practices for Windows Firewall with Advanced Security

If vulnerable ports such as Windows remote port 3389 and Linux remote port 22 are exposed, malicious parties can scan for and initiate attacks over these ports. You can prevent these attacks by modifying the default remote port or restricting remote access sources. This topic takes an ECS instance running Windows Server 2012 R2 as an example to describe how to use Windows Firewall with Advanced Security (WFAS) to restrict IP addresses of remote access.

Prerequisites

Context

WFAS is an important part of a layered security model. WFAS provides host-based bidirectional network traffic filtering to block unauthorized network traffic flowing into or out of the local computer. WFAS also works with Network Awareness to apply corresponding security settings to the network to which the computer is connected. WFAS integrates Windows Firewall and Internet Protocol Security (IPsec) configuration settings into a single Microsoft Management Console (MMC), becoming an important part of the network isolation strategy.

> **Note** The procedure described in this topic is not applicable to ECS instances that run Windows Server 2016. For ECS instances running Windows Server 2016, we recommend that you restrict remote access sources by adding security group rules. For more information, see the "Scenario 4: Allow your instance to access only specific public IP addresses" section in Scenarios for security groups. For more information about how to add security group rules, see Add security group rules.

Use MMC to configure WFAS

1.

2. Enable the firewall.
   i. Press the shortcut keys *Win + R* to open the **Run** dialog box.
   ii. Enter *firewall.cpl* and press the Enter key.

   iii. Click **Turn Windows Firewall on or off** to view the firewall status. The firewall is disabled by default.
   iv. Enable Windows firewalls for each network type and click **OK**.

3. Check the remote RDP port 3389.
   i. Press the shortcut keys *Win + R* to open the **Run** dialog box.
   ii. Enter *wf.msc* and press the Enter key.

   iii. Click **Inbound Rules**. In the Open RDP Port 3389 inbound rule, you can find that the default allowed port is 3389.

4. Add the remote RDP port 3389 to **Windows Firewall with Advanced Security**.
   i. Click **New Rule**, and the **New Inbound Rule Wizard** dialog box appears.
   ii. In the **Rule Type** step, select **Port** and click **Next**.
   iii. In the Protocol and Ports step, select *TCP* as the protocol, select the Specific local ports option button and enter *3389* in the field, and then click **Next**.
   iv. Select **Allow the connection** and click **Next**.
   v. Keep the default configuration and click **Next**.
   vi. Set the rule name such as RemoteDesktop, and click **Finish**.
5. Configure the scope.
   i. Right-click the created inbound rule RemoteDesktop and choose **Properties** from the shortcut menu.
   ii. On the **Scope** tab, select **These IP addresses:** in the **Remote IP address** section, add one or more IP addresses or CIDR blocks, and then click **OK**.

   > **Notice** After the scope parameter is configured, remote connection is only allowed from the remote IP address that you have set in the scope.

6. Validate the scope. Add any other IP address to the remote IP addresses and click **OK**.
   The remote connection is automatically interrupted, indicating that the scope parameter has taken effect.
   If the remote connection continues, right-click the Open RDP Port 3389 inbound rule and choose **Disable Rule** from the shortcut menu.
7. In the ECS Console, change the remote IP addresses in the scope to the public IP address of the office environment to restore the remote connection.
   i. Log on to the ECS console.

ii. In the instance list, find the target instance. In the **Actions** column, click **Connect**.

iii. In the **Enter VNC password:** field, enter the password and click **OK**.

iv. Modify the remote IP address in the scope of the RemoteDesktop inbound rule. Change the original 1.1.1.1 IP address to the IP address that you want to authorize.

Use CLI to configure WFAS

You can also run the **netsh** command in CLI to configure WFAS. The following list shows examples of the **netsh** command:

- Export the firewall configuration file.

```
netsh advfirewall export c:\adv.pol
```

- Import the firewall configuration file.

```
netsh advfirewall import c:\adv.pol
```

- Restore the default settings of the firewall.

```
netsh advfirewall reset
```

- Disable the firewall.

```
netsh advfirewall set allprofiles state off
```

- Enable the firewall.

```
netsh advfirewall set allprofiles state on
```

- Set the default firewall policy for all configuration files to block inbound traffic and allow outbound traffic.

```
netsh advfirewall set allprofiles firewallpolicy blockinbound,allowoutbound
```

- Delete the ftp rule.

```
netsh advfirewall firewall delete rule name=ftp
```

- Delete all inbound rules for the local port 80.

```
netsh advfirewall firewall delete rule name=all protocol=tcp localport=80
```

- Add an inbound rule for the remote desktop to allow traffic from port 3389.

```
netsh advfirewall firewall add rule name=remote desktop (TCP-In-3389) protocol=TCP dir=in localport=3389 action=allow
```

Related information

-
-

5.10. Isolation of instances within a security group

A security group is a virtual firewall that provides Stateful Packet Inspection (SPI) and packet filtering. It contains instances in the same region with the same security requirements and mutual trust. Alibaba Cloud provides various access control policies to allow you isolate instances within a security group.

Intra-group isolation rules

- Network isolation in a security group is implemented between network interfaces, not between instances. If multiple Elastic Network Interfaces (ENIs) are bound to an instance, you need to set isolation rules for each ENI.
- Instances in a security group can access each other by default, which is not changed by the isolation rules.

  Intra-group isolation rules are user-defined access control policies, and are invalid for the default security groups and new security groups. The default access control policy for a security group is: instances in the same security group can access each other over the intranet, while instances in different security groups cannot.
- Intra-group isolation rules have the lowest priority.

  To isolate instances in a security group, make sure no intercommunication rules apply to them except for the isolation rules. In the following cases, instances can still access each other even though intra-group isolation rules are set:

  - Intra-group isolation rules are set in a security group, while an Access Control List (ACL) that permits intra-group communication between instances is set at the same time.
  - Intra-group isolation rules are set in a security group, while intra-group intercommunication is configured at the same time.
- Intra-group isolation rules only apply to the instances in the current security group.

Modify the access control policy

You can use the ModifySecurityGroupPolicy interface to modify the access control policy within a security group.

Case analysis

The following figure shows the relationship between three instances and their security groups.

In this example, Group1, Group2, and Group3 are three different security groups. ECS1, ECS2, and ECS3 are three different ECS instances. ECS1 and ECS2 belong to Group1 and Group2. ECS2 and ECS3 belong to Group3.

The intra-group intercommunication policies of the three security groups are as follows:

Security group	Intra-group intercommunication policy	Instances included
Group1	Isolated	ECS1 and ECS2
Group2	Interconnected	ECS1 and ECS2
Group3	Interconnected	ECS2 and ECS3

The communication status between instances is as follows:

Instance	Interconnected or isolated?	Reason
ECS1 and ECS2	Interconnected	ECS1 and ECS2 belong to both Group1 and Group2. The policy of Group1 is "isolated", while that of Group2 is "interconnected". As intra-group isolation has the lowest priority, ECS1 and ECS2 are interconnected.
ECS2 and ECS3	Interconnected	Both ECS2 and ECS3 belong to Group3. The policy of Group3 is "interconnected", so ECS2 and ECS3 are interconnected.
ECS1 and ECS3	Isolated	ECS1 and ECS3 belong to different security groups. Instances in different security groups are not interconnected by default. To permit access between instances in two security groups, you can authorize security groups through security group rules.

5.11. Security group quintuple rules

Security groups are used to configure network access control for one or more ECS instances. As an important means of security isolation, security groups logically isolate security domains on the cloud. Security group quintuple rules allow you to precisely control the following five parameters: source IP address, source port, destination IP address, destination port, and transport layer protocol.

Background information

Previously, security group rules have the following characteristics:

- The inbound rules support only the settings of the source IP address, destination port, and transport layer protocol.
- The outbound rules support only the settings of the destination IP address, destination port, and transport layer protocol.

In most scenarios, these security group rules provide simple configurations, but have the following drawbacks:

- You cannot specify a range of source ports in an inbound rule. Inbound traffic over all ports is allowed by default.
- You cannot specify the destination IP address in an inbound rule. Inbound traffic from all IP addresses within a security group is allowed by default.
- You cannot specify a range of source ports in an outbound rule. Outbound traffic over all ports is allowed by default.

- You cannot specify the source IP address in an outbound rule. Outbound traffic from all IP addresses within a security group is allowed by default.

Definition

A quintuple rule includes the following parameters: source IP address, source port, destination IP address, destination port, and transport layer protocol.

Quintuple rules are designed to provide more fine-grained control over the preceding five parameters, while completely compatible with existing security group rules.

Example quintuple outbound rule:

```
Source IP address: 172.16.1.0/32
Source port: 22
Destination IP address: 10.0.0.1/32
Destination port: no restriction
Transport layer protocol: TCP
Action: Forbid
```

The example outbound rule indicates that TCP access from `172.16.1.0/32` to `10.0.0.1/32` over port 22 is denied.

Scenarios

- Some platform products use solutions from third-party vendors to provide users with network services. To prevent unauthorized access from these products to ECS instances of the users, quintuple rules are required to control inbound and outbound traffic more precisely.
- If ECS instances in a security group are configured to be isolated from each other and you want to allow specified ECS instances to communicate with each other, you must configure quintuple rules.

Configure quintuple rules

You can call API operations to configure quintuple rules.

- To create an inbound security group rule, see AuthorizeSecurityGroup.
- To create an outbound security group rule, see AuthorizeSecurityGroupEgress.
- To delete an inbound security group rule, see RevokeSecurityGroup.
- To delete an outbound security group rule, see RevokeSecurityGroupEgress.

Parameters

The following table describes the parameters of security group rules.

Parameter	Meaning in inbound rules	Meaning in outbound rules
SecurityGroupId	The ID of the security group to which the current inbound rule belongs. This is the ID of the destination security group.	The ID of the security group to which the current outbound rule belongs. This is the ID of the source security group.

Parameter	Meaning in inbound rules	Meaning in outbound rules
DestCidrIp	Optional. The range of destination IP addresses. • If DestCidrIp is specified, you can control the range of destination IP addresses in an inbound rule more precisely. • If DestCidrIp is not specified, the range of IP addresses in an inbound rule includes all IP addresses in the security group with the specified SecurityGroupId.	The range of destination IP addresses. Either DestGroupId or DestCidrIp must be specified. If both are specified, DestCidrIp takes priority.
PortRange	Required. The range of destination ports.	Required. The range of destination ports.
DestGroupId	Manual input is not allowed. The value of DestGroupId must be the same as that of SecurityGroupId.	The ID of the destination security group. You must specify either DestGroupId or DestCidrIp. If you specify both parameters, DestCidrIp takes priority.
SourceGroupId	The ID of the source security group ID. You must specify either SourceGroupId or SourceCidrIp. If you specify both parameters, SourceCidrIp takes priority.	Manual input is not allowed. The value of SourceGroupId must be the same as that of SecurityGroupId.
SourceCidrIp	The range of source IP addresses. You must specify either SourceGroupId or SourceCidrIp. If you specify both parameters, SourceCidrIp takes priority.	Optional. The range of source IP addresses. • If SourceCidrIp is specified, you can control the range of source IP addresses in an outbound rule more precisely. • If SourceCidrIp is not specified, the range of IP addresses in an outbound rule includes all IP addresses in the security group with the specified SecurityGroupId.
SourcePortRange	Optional. The range of source ports. If this parameter is not specified, no source ports are restricted.	Optional. The range of source ports. If this parameter is not specified, no source ports are restricted.

5.12. Enable or disable SELinux

Security-enhanced Linux (SELinux) is a Linux kernel feature that provides a security policy-based protection mechanism for access control. This topic describes how to enable or disable SELinux and avoid system boot failures.

Prerequisites

An ECS instance is created from an Alibaba Cloud public image or a custom image.

> **Note** If the custom image that you used was created from imported local files or the source server migration through Server Migration Center (SMC), ensure that SELinux is disabled on the source server before migration.

Context

Typically, enabling SELinux can enhance system security. However, it can damage files in the operating system and lead to system boot failures. If your enterprise or team has high requirements on security and SELinux must be enabled for your operating systems, you can follow operations in this topic to enable this feature without system boot failures. This topic uses CentOS 7.2 64-bit as an example.

Enable SELinux

1. Remotely connect to an ECS instance as a root user. For more information, see Overview.
2. Run the following command on an instance to modify the `config` file of SELinux:

   ```
   vi /etc/selinux/config
   ```

3. Find `SELINUX=disabled`, press the `i` key to enter edit mode, and then enable SELinux by modifying this parameter.

   You can modify the parameter to one of the following modes as needed:

   - `SELINUX=enforcing`: indicates that all security policy violations will be prohibited.
   - `SELINUX=permissive`: indicates that security policy violations will not be prohibited but will be recorded in the operation logs.

4. After the parameter has been modified, press the `Esc` key and run the `:wq` command to save and close the file.

   > **Note** You must restart the instance before you modify the `config` file. However, if you restart the instance directly, the system will fail to start. Therefore, you must create an autorelabel file in the root directory before you restart the system.

5. Create an `autorelabel` file. After the instance is restarted, SELinux will automatically relabel all system files.

   ```
   touch /.autorelabel
   ```

6. Restart the ECS instance.

   ```
   shutdown -r now
   ```

Check SELinux status

1. Remotely connect to an ECS instance as a root user. For more information, see Overview.

2. Run the `getenforce` command to check the status of SELinux. The return value can be `enforcing` or `permissive`. The return value in this topic is `enforcing`.
3. Run the `sestatus` command to query more information about SELinux.

   If the return value of `SELinux status` is `enabled`, SELinux is enabled.

Disable SELinux

1. Remotely connect to an ECS instance as a root user. For more information, see Overview.
2. Run the `getenforce` command to check the status of SELinux. If the return value is `enforcing`, SELinux is enabled.
3. Disable SELinux temporarily or permanently.
   - Run the `setenforce 0` command to disable SELinux temporarily.
   - Disable SELinux permanently.
     a. Run the following command to edit the `config` file of SELinux:

     ```
     vi /etc/selinux/config
     ```

     b. Find `SELINUX=enforcing`, press the `i` key to enter edit mode, and then modify the parameter to `SELINUX=disabled`.

     c. After that, press the `Esc` key and run the `:wq` command to save and close the file.

     d. Restart the ECS instance.

     ```
     shutdown -r now
     ```

     e. After the instance has been restarted, run the `getenforce` command to check the status of SELinux. If the return value is `disabled`, SELinux is disabled.

What's next

You can create a custom image from an ECS instance that has SELinux enabled. Then, you can create more SELinux-enabled instances from this custom image as needed.

5.13. Revoke the authorization for internal network communication between ECS instances in different accounts through the API

If you have authorized internal network communication between ECS instances across different accounts within the same region, you can revoke security group authorization by calling the API operation.

Prerequisites

- An Alibaba Cloud account is created. To create an Alibaba Cloud account, go to the account registration page.
- Alibaba Cloud Command-Line Interface (CLI) is installed for the ECS instance. For more information about how to install Alibaba Cloud CLI in different operating systems, see the following topics:
  - Windows
  - Linux
  - MacOS

Context

In this topic, the RevokeSecurityGroup operation is used to revoke authorized security group rules. Before you start, you must prepare the following information:

- Account name: the name of the account that you use to log on to the ECS console.
- Security group IDs of the ECS instances: the IDs of the security groups to which the instances involved belong.

  You can query the security group IDs in the ECS console or by calling the DescribeSecurityGroupReferences operation.
- Region IDs of the ECS instances: See . *cn-beijing* is used in this example.

Assume that the information of the two accounts is as follows.

Account	Account name	Security group	Security group ID
Account A	a@aliyun.com	sg1	sg-bp1azkttqpldxgtedXXX
Account B	b@aliyun.com	sg2	sg-bp15ed6xe1yxeycg7XXX

Procedure

1. Run the following command for Account A:

```
aliyun ecs RevokeSecurityGroup --SecurityGroupId sg-bp1azkttqpldxgtedXXX --RegionId cn-beijing --IpProtocol all --PortRange -1/-1 --SourceGroupId sg-bp15ed6xe1yxeycg7XXX --SourceGroupOwnerAccount b@aliyun.com --NicType intranet
```

2. Run the following command for Account B:

```
aliyun ecs RevokeSecurityGroup --SecurityGroupId sg-bp15ed6xe1yxeycg7XXX --RegionId cn-beijing --IpProtocol all --PortRange -1/-1 --SourceGroupId sg-bp1azkttqpldxgtedXXX --SourceGroupOwnerAccount a@aliyun.com --NicType intranet
```

Related information

- RevokeSecurityGroup

- DescribeSecurityGroupReferences

5.14. Authorize internal network communication between ECS instances under different accounts by using the API

This topic describes how to authorize internal network communication between ECS instances that are in the same region but belong to different accounts.

Prerequisites

Alibaba Cloud CLI is used to call the ECS API. Make sure that you have installed and configured Alibaba Cloud CLI. For more information, see Install CLI and Configure CLI.

Context

You can authorize internal network communication in one of the following modes:

- Authorize internal network communication between ECS instances: You can authorize internal communication between two ECS instances that belong to the same account.
- Authorize internal network communication between accounts: You can authorize internal network communication between ECS instances that belong to two different accounts across two different security groups that are in the same region, including those purchased after authorization is completed.

> Note To enable internal network communication between different accounts, you need to authorize communication between security groups in each account. These ECS instances can then communicate over the internal network. Modifying the configurations of a security group will affect all instances in the security group as well as the services running on these instances. Use caution when performing this operation.

Security groups are virtual firewalls for ECS instances. Security groups do not provide communication and networking capabilities. After you authorize internal network communication between instances that belong to different security groups, ensure that the instances can establish internal network connection.

- If all instances are of the classic network type, they must be in the same region to communicate with each other.
- VPCs are isolated by default. If all instances are of the VPC type, these instances cannot communicate with each other. We recommend that you allow ECS instances to communicate over a public network or through Express Connect, VPN Gateway, or Cloud Enterprise Network (CEN). For more information, see Express Connect, VPN Gateway, and CEN.
- If instances are of different network types, establish a ClassicLink connection to allow communication between these instances. For more information, see 经典网络和专有网络互通.
- If instances are in different regions, we recommend that you allow ECS instances to communicate over a public network or through Express Connect, VPN Gateway, or CEN. For more information, see Express Connect, VPN Gateway, and CEN.

Authorize internal network communication between ECS instances

1. Query internal IP addresses and security group IDs of the two ECS instances. You can use the console or call the DescribeInstances operation to obtain security group IDs of the instances. The following table lists the information of the two ECS instances.

Instance	IP address	Security group	Security group ID
Instance A	10.0.0.1	sg1	sg-bp1azkttqpldxgtedXXX
Instance B	10.0.0.2	sg2	sg-bp15ed6xe1yxeycg7XXX

2. Add a rule to sg1 to allow inbound traffic from 10.0.0.2.

```
aliyun ecs AuthorizeSecurityGroup --SecurityGroupId sg-bp1azkttqpldxgtedXXX --RegionId cn-qingdao --IpProtocol all  --PortRange=-1/-1. --SourceCidrIp 10.0.0.2 --NicType intranet
```

3. Add a rule to sg2 to allow inbound traffic from 10.0.0.1.

```
aliyun ecs AuthorizeSecurityGroup --SecurityGroupId sg-bp15ed6xe1yxeycg7XXX --RegionId cn-qingdao --IpProtocol all  --PortRange=-1/-1. --SourceCidrIp 10.0.0.1 --NicType intranet
```

> **Note**
>
> - In the preceding commands, the region ID *cn-qingdao* is for reference only. Replace it with your actual region ID.
> - In the preceding commands, the AuthorizeSecurityGroup operation is called to add inbound rules to security groups. Specify the SecurityGroupId and SourceCidrIp parameters.

4. After a few minutes, run the ping command to check whether the two ECS instances can communicate with each other over the internal network.

Authorize internal network communication between accounts

1. Query names and security group IDs of the two accounts. You can use the console or call the DescribeInstances operation to obtain security group IDs of the ECS instances. The following table lists the information of two accounts.

Account	Account ID	Security group	Security group ID
Account A	a@aliyun.com	sg1	sg-bp1azkttqpldxgtedXXX
Account B	b@aliyun.com	sg2	sg-bp15ed6xe1yxeycg7XXX

2. Add a rule to sg1 to allow inbound traffic from sg2.

```
aliyun ecs AuthorizeSecurityGroup --SecurityGroupId sg-bp1azkttqpldxgtedXXX --RegionId cn-qingdao --IpProtocol all  --PortRange=-1/-1. --SourceGroupId sg-bp15ed6xe1yxeycg7XXX --SourceGroupOwnerAccount b@aliyun.com --NicType intranet
```

3. Add a rule to sg2 to allow inbound traffic from sg1.

```
aliyun ecs AuthorizeSecurityGroup --SecurityGroupId sg-bp15ed6xe1yxeycg7XXX --RegionId cn-qin
gdao --IpProtocol all  --PortRange=-1/-1. --SourceGroupId sg-bp1azkttqpldxgtedXXX --SourceGrou
pOwnerAccount a@aliyun.com --NicType intranet
```

> **Note**
>
> - In the preceding commands, the region ID *cn-qingdao* is for reference only. Replace it with your actual region ID.
> - In the preceding commands, the AuthorizeSecurityGroup operation is called to add inbound rules to security groups. Specify the SecurityGroupId, SourceGroupId, and SourceGroupOwnerAccount parameters.

4. After a few minutes, run the ping command to check whether the ECS instances can communicate with each other over the internal network.

6.Data recovery

6.1. Restore data deleted by mistake

This topic describes how to use extundelete to restore data that was accidentally deleted. CentOS 7 is used in the examples.

Prerequisites

An Alibaba Cloud account is created. To create an Alibaba Cloud account, go to the account registration page.

Context

In practice, data may be accidentally deleted. In this case, you need to restore the data quickly and effectively. Alibaba Cloud offers several ways to restore data, including the following ways:

- In the ECS console, roll back snapshots or restore custom images to restore data.
- Implement load balancing and high availability for your business by purchasing multiple instances. For more information, see What is Server Load Balancer?.
- Use Object Storage Service (OSS) to store a large amount of data such as web pages, images, and videos. For more information, see What is OSS?.

There are multiple open source data recovery tools for Linux, such as debugfs, R-Linux, ext3grep, and extundelete. Although ext3grep and extundelete are both popular and adopt similar recovery techniques, extundelete has a higher performance. Linux systems do not have a Recycle Bin function built-in. You can install extundelete on Linux systems to restore data that has been deleted by accident.

extundelete can find and recover the deleted data by combining the inode information and logs to find the inode block. extundelete can recover deleted data from ext3 or ext4 partitions.

If you accidentally delete data, you must first detach the disk or disk partition on which the deleted data was stored. This is because after a file is deleted, the inode pointers of the file are cleared but the file will remain on the disk. If the disk is attached in read/write mode, data blocks of the deleted file may be reallocated by the operating system. After the data blocks are overwritten by new data, the original data will be lost completely and cannot be restored. To reduce the risk of data overwriting and improve chance of data restoration, you can store files on disks in read-only mode.

> **Note** When you restore deleted data online, do not install extundelete on the disk where the deleted data was located. Otherwise, you may overwrite the data that you want to restore. Back up the disk by taking a snapshot before you perform any operations.

This topic is applicable to the following users:

- Users who have accidentally deleted files from a disk and have not since performed write operations on the disk.
- Users whose websites have low traffic and who have few ECS instances.

Required software releases: e2fsprogs-devel, e2fsprogs, gcc-c++, make (a utility for compiling and linking multi-file projects), and extundelete-0.2.4.

> **Note** libext2fs 1.39 or later is required to run extundelete. However, to restore data from ext4 partitions, make sure that you have access to e2fsprogs 1.41 or later. You can run the `dumpe2fs` command and record the version of e2fsprogs.

The preceding releases were available at the time of writing. The versions that you download may be different in your actual running environment.

Procedure

Perform the following operations to use extundelete to restore data that was deleted by accident:

1. Step 1: Deploy extundelete
2. Step 2: Use extundelete to stimulate the restoration process

Step 1: Deploy extundelete

Run the following commands to deploy extundelete:

```
wget http://zy-res.oss-cn-hangzhou.aliyuncs.com/server/extundelete-0.2.4.tar.bz2
yum -y install bzip2 e2fsprogs-devel e2fsprogs gcc-c++ make # Install related dependencies and li
braries.
tar -xvjf extundelete-0.2.4.tar.bz2
cd extundelete-0.2.4 # Go to the program directory.
./configure # If the output in the following figure is generated, extundelete is in
stalled.
```

```
make && make install
```

At this point, the *src* directory appears. It contains an extundelete executable file and the corresponding path. The default path is *usr/local/bin*. The following step for data restoration is performed in the *usr/local/bin* directory.

Step 2: Use extundelete to stimulate the restoration process

Perform the following steps to use extundelete to stimulate the restoration process:

1. Check the available disks and partitions of your ECS instance, and then format and partition the */dev/vdb* disk. For more information, see Format a data disk for a Linux-based ECS instance.

   ```
   fdisk -l
   ```

2. Attach the partitioned disk to the */zhuyun* directory, create a file named *hello* in the */zhuyun* directory, and then write `test` to the file.

```
mkdir /zhuyun                    # Create the zhuyun directory.
mount /dev/vdb1 /zhuyun          # Attach the disk to the zhuyun directory.
echo test > hello                # Create a test file.
```

3. Record the MD5 values of the *hello* file. The **md5sum** command is used to generate and verify the MD5 values of the file before and after deletion.

```
md5sum hello
```

4. Delete the hello file in simulation mode.

```
rm -rf hello
cd ~
fuser -k /zhuyun                 # Terminate the process tree that uses a specified partition. Skip this step if no resources are occupied.
```

5. Detach a data disk.

```
umount /dev/vdb1                 # Before you use any file restoration tool, unmount or mount the partitions to be restored in read-only mode to prevent their data from being overwritten.
```

6. Use extundelete to restore the file.

```
extundelete --inode 2 /dev/vdb1      # Query the content in a specified inode. The "2" indicates that the entire partition is queried. To query a directory, you can specify the inode and the directory. The deleted file and the inode are displayed.
```

```
/usr/local/bin/extundelete  --restore-inode 12  /dev/vdb1    # Restore the deleted file.
```

At this point, the *RECOVERED_FILES* directory appears under the directory where the command is executed.

7. Run the **md5sum** command to check the MD5 value of the *RECOVERED_FILES* file after restoration.

```
md5sum RECOVERED_FILES
```

Check whether the MD5 values of the *hello* and *RECOVERED_FILES* files are the same. If they are, the data is restored.

6.2. Handle low disk space on Windows instances

This topic describes how to handle low disk space on Windows instances and the best practices of daily disk management for Windows instances.

Prerequisites

An Alibaba Cloud account is created. To create an Alibaba Cloud account, go to the account registration page.

Context

The methods described in this topic apply to Windows Server 2003 and later. Windows Server 2012 R2 is used as an example.

Solutions and best practices

When you are running out of available disk space on a Windows instance, you can take one of the following measures:

- Release disk space
- Resize the disk

Develop good habits for using disk space with these best practices:

- Compress files for archiving
- Delete unneeded applications on a regular basis
- Configure disk monitoring

Release disk space

When you are running out of available disk space on a Windows instance, you can clear unnecessary files from the disk. Perform the following steps:

1. Find the files that are taking up the most disk space.
   i. Connect to the Windows instance. For more information, see Connect to a Windows instance.
   ii. Click **Start** and then click **This PC**.
   iii. Click the disk to be cleared, and press **Ctrl+F**.
   iv. In the top navigation bar, choose **Search** > **Size**, and filter out large files on the specified disk.

   > **Note** You can also customize a file size range for retrieval. For example:
   > - Enter *Size: > 500 MB* to search for files that are larger than 500 MB in size.
   > - Enter *Size: > 100 MB < 500 MB* to search for files that are between 100 MB and 500 MB in size.

2. Delete files that you no longer need. We recommend that you use the Windows Disk Cleanup utility to delete unneeded files such as log files, and empty the recycle bin. The Windows Disk Cleanup utility is not installed on the instance by default and must be installed manually. Perform the following steps to install the utility and use it to delete files:
   i. In the taskbar, click the **Server Manager** icon.

ii. In the upper-right corner, choose **Manage** > **Add Roles and Features**.

iii. In the Add Roles and Features Wizard dialog box, keep the default settings and click **Next** until the **Features** module is displayed. Select **Ink and Handwriting Services** and **Desktop Experience**, and click **Next**.

iv. Click **Install**.

v. After the utility is installed, the system will prompt you to restart the instance. You must restart the instance manually. After the instance restarts, confirm that Desktop Experience has been installed.

vi. Click **Start**. In the top search box, enter **Disk Cleanup**. In the dialog box that appears, select the disk that you want to delete and click **OK**.

Resize the disk

When you are running out of available disk space on a Windows instance, you can resize the disk. For more information, see Resize disks online for Linux instances.

Compress files for archiving

For files that are generated on a regular basis, you can compress them for archiving to improve disk usage. We recommend that you use WinRAR to compress files. The following example describes how to configure a backup policy for compressing files:

1. Download and install WinRAR. Download link: WinRAR and RAR archiver downloads. WinRAR is used in this example.
2. After WinRAR is installed, find the file to be compressed, right-click the file, and then select **Add to archive...**.
3. In the **Settings** dialog box that appears, click the **Backup** tab and select **Generate archive name by mask**. Do not click **OK**.
4. Click the **General** tab and then click **Browse** to specify the path in which to save the archive file. Click **Profiles...** and select **Save current settings to a new profile...**.
5. In the **Profile parameters** dialog box, set the **Profile name** parameter and select **Save archive name**, **Save selected file names**, and **Create shortcut on desktop**. Click **OK**.
6. In the **Archive name and parameters** dialog box, click **OK**.
   A shortcut key to the archive file is generated on the desktop.
7. Press **Win+R** to open the **Run** dialog box. Run the `control` command to open the **Control Panel** page. On the **Control Panel** page, click **System and Security** and then click **Scheduled tasks**. In the **Task Scheduler** dialog box, click **Create Basic Task...**.
8. In the Create Basic Task Wizard dialog box, enter a new task name and click **Next**.
9. Select a trigger for the task and click **Next**. Select **Start a program** and click **Next**.
10. In the Start a Program step, set the **Program/script:** parameter. To set this parameter, find the previously generated shortcut key. Right-click the shortcut key and then select **Properties**. In the dialog box that appears, copy the value of the **Target:** field.
11. In the Create Basic Task Wizard dialog box, paste the copied value in the **Program/script:** field of the **Start a Program** step. Click **Finish**.

After you configure the backup policy, you can delete expired archive files on a regular basis to avoid taking up too much disk space.

Delete unneeded applications on a regular basis

You can delete unneeded applications on a regular basis by clicking **Programs and Features** on the **Control Panel** page.

Configure disk monitoring

Monitoring plug-ins are pre-installed on ECS instances. You can log on to the CloudMonitor console to create disk alert rules. By doing so, you can receive alerts when disk usage exceeds a configured threshold and clean the disk in a timely manner. For more information, see Manage alert rules.

6.3. Restore data in Linux instances

When you troubleshoot disks, you may encounter the loss of data disk partitions. This topic describes data disk partition loss in Linux and the solutions. This topic also describes the common mistakes and best practices for using disks to avoid the risk of data loss.

Prerequisites

- A snapshot is created for the data disk that lost a partition. If errors occur during data restoration, you can use the snapshot to roll back the data disk to the state before the restoration. For more information, see Create a normal snapshot and Roll back a disk by using a snapshot.
- An Alibaba Cloud account is created. To create an Alibaba Cloud account, go to the account registration page.

Context

In a Linux instance, you can use one of the following tools to restore data on a data disk:

- fdisk: a tool provided by Linux for partitioning disks.
- testdisk: a tool used to restore disk partitions or data in Linux. By default, the tool is not provided in Linux. You must install it on your own. For example, you can run the **yum install -y testdisk** command to install testdisk in CentOS.
- partprobe: a tool provided by Linux. The tool is used to enable the kernel to re-read partitions without restarting the system.

Methods

After you restart a Linux instance, the partition or data of data disks may be lost. This may be because you did not set the partition to be mounted automatically upon startup of the instance in the *etc/fstab* file. In this case, you can manually mount the data disk partition. If the system prompts partition table loss when you manually mount the data disk partition, you can use one of the following methods to restore the partition or data:

- Restore a partition by using fdisk
- Restore a partition by using testdisk
- Restore data by using testdisk

Restore a partition by using fdisk

Typically, default values apply to the starting and ending sectors of a partition when you partition a data disk. You can first run the **fdisk** command to restore the partition. For more information, see Format a data disk for a Linux instance.

```
[root@Aliyun ~]# fdisk /dev/xvdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-10485759, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-10485759, default 10485759):
Using default value 10485759
Partition 1 of type Linux and of size 5 GiB is set

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
[root@Aliyun ~]# mount /dev/xvd
xvda   xvda1  xvdb   xvdb1
[root@Aliyun ~]# mount /dev/xvdb
xvdb   xvdb1
[root@Aliyun ~]# mount /dev/xvdb1 /mnt/
[root@Aliyun ~]# ls /mnt/
123.sh  configclient  data  diamond  install_edsd.sh  install.sh  ip.qz
```

If the preceding operation cannot restore the partition, you can use testdisk to restore the partition.

Restore a partition by using testdisk

A disk device named */dev/xvdb* is used in this example. To use testdisk to restore a partition, follow these steps:

1. Run the **testdisk /dev/xvdb** command (you can replace the device name), select **Proceed** (default value), and then press the Enter key:

```
TestDisk 7.0, Data Recovery Utility, April 2015
Christophe GRENIER <grenier@cgsecurity.org>
http://www.cgsecurity.org

  TestDisk is free software, and
comes with ABSOLUTELY NO WARRANTY.

Select a media (use Arrow keys, then press Enter):
>Disk /dev/xvdb - 5368 MB / 5120 MiB


>[Proceed ]  [  Quit  ]

Note: Disk capacity must be correctly detected for a successful recovery.
If a disk listed above has incorrect size, check HD jumper settings, BIOS
detection, and install the latest OS patches and disk drivers.
```

2. Select the partition table type for scanning. Typically, the default value *Intel* is selected. Select *EFI GPT* if your data disk uses the GPT format.

```
TestDisk 7.0, Data Recovery Utility, April 2015
Christophe GRENIER <grenier@cgsecurity.org>
http://www.cgsecurity.org

Disk /dev/xvdb - 5368 MB / 5120 MiB

Please select the partition table type, press Enter when done.
>[Intel  ] Intel/PC partition
 [EFI GPT] EFI GPT partition map (Mac i386, some x86_64...)
 [Humax  ] Humax partition table
 [Mac    ] Apple partition map
 [None   ] Non partitioned media
 [Sun    ] Sun Solaris partition
 [XBox   ] XBox partition
 [Return ] Return to disk selection


Note: Do NOT select 'None' for media with only a single partition. It's very
rare for a disk to be 'Non-partitioned'.
```

3. Select *Analyse* and press the Enter key.

```
Disk /dev/xvdb - 5368 MB / 5120 MiB
     CHS 652 255 63 - sector size=512

>[ Analyse  ] Analyse current partition structure and search for lost partitions
 [ Advanced ] Filesystem Utils
 [ Geometry ] Change disk geometry
 [ Options  ] Modify options
 [ MBR Code ] Write TestDisk MBR code to first sector
 [ Delete   ] Delete all data in the partition table
 [ Quit     ] Return to disk selection


Note: Correct disk geometry is required for a successful recovery. 'Analyse'
process may give some warnings if it thinks the logical geometry is mismatched.
```

4. Select *Quick Search* and press the Enter key if the partition information is not displayed.

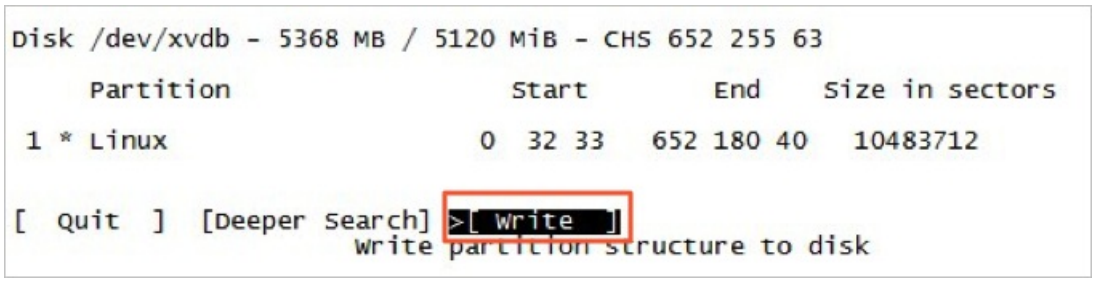
```
Disk /dev/xvdb - 5368 MB / 5120 MiB - CHS 652 255 63
Current partition structure:
     Partition                  Start        End    Size in sectors

No partition is bootable


*=Primary bootable  P=Primary  L=Logical  E=Extended  D=Deleted
>[Quick Search]                Try to locate partition
```

The partition information is displayed in the command output, as shown in the following figure.

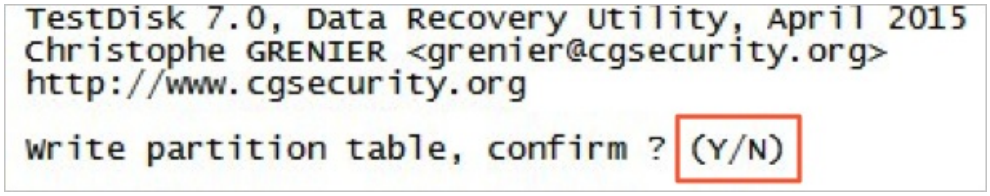
```
Disk /dev/xvdb - 5368 MB / 5120 MiB - CHS 652 255 63
     Partition                  Start        End    Size in sectors
>* Linux                    0  32 33   652 180 40   10483712


Structure: Ok.  Use Up/Down Arrow keys to select partition.
Use Left/Right Arrow keys to CHANGE partition characteristics:
*=Primary bootable  P=Primary  L=Logical  E=Extended  D=Deleted
Keys A: add partition, L: load backup, T: change type, P: list files,
     Enter: to continue
```

5. Select the partition and press the Enter key.
6. Select *Write* to save the partition.

> Note Select *Deeper Search* to continue to search if the expected partition is not listed.



7. Press the *Y* key to save the partition.



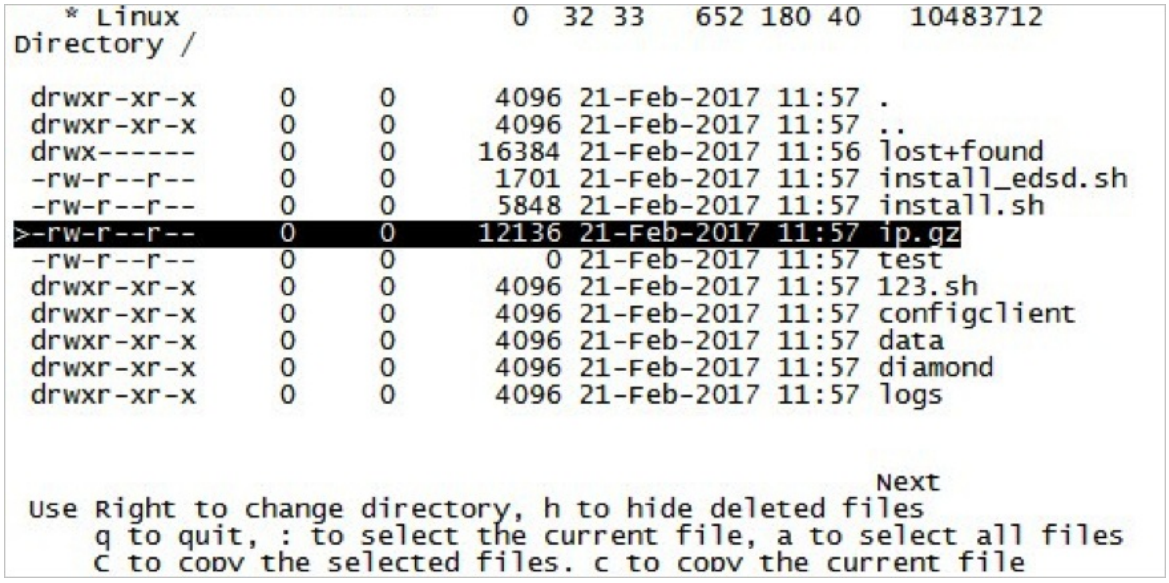
8. Run the **partprobe /dev/xvdb** command (you can replace the device name) to manually refresh the partition table.
9. Mount the partition again and view the data in the data disk.

```
[root@Aliyun home]# mount /dev/xvdb1 /mnt/
[root@Aliyun home]# ls /mnt/
123.sh  configclient  data  diamond  install_edsd.sh  install.sh  ip.gz  logs  lost+found  test
```

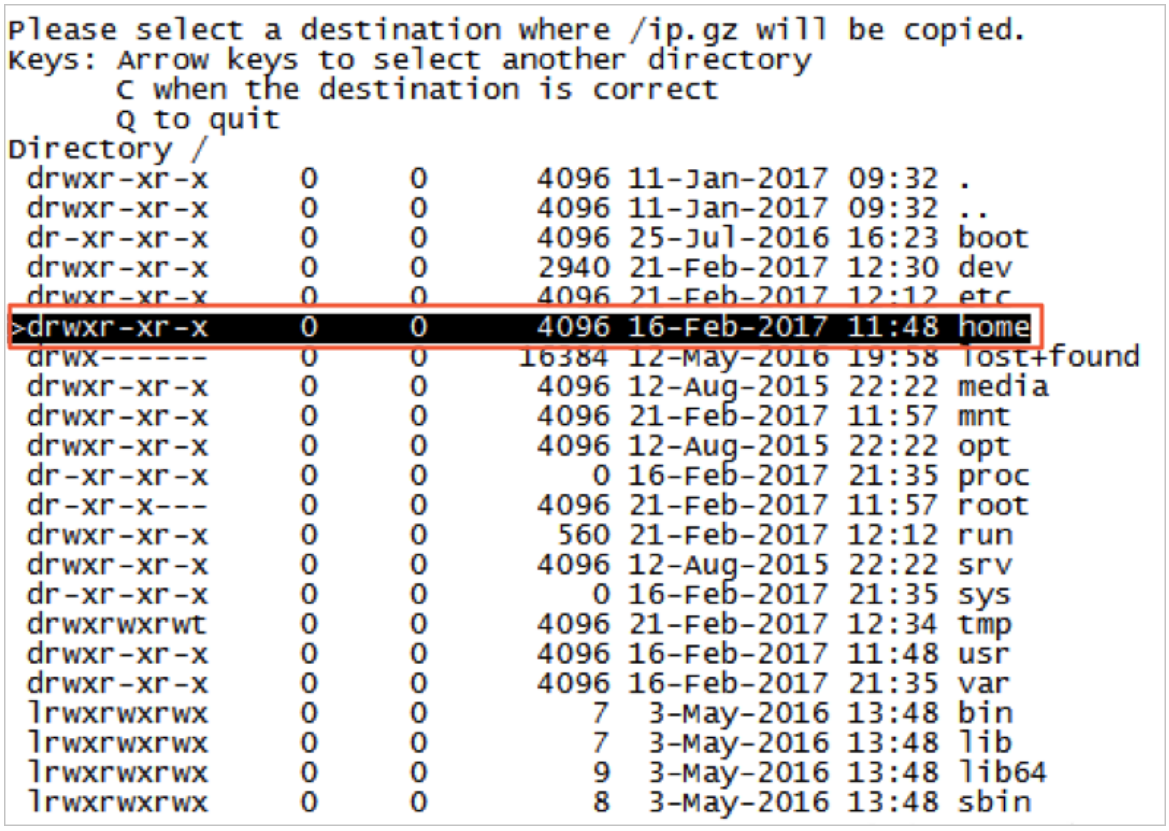
Restore data by using testdisk

In some cases, you can scan and locate a disk partition by using testdisk. However, you cannot save the partition. In this case, you can directly restore the data. Follow these steps:

1. Scan and locate a disk partition by using testdisk. For more information, see Step 1 to Step 4 in Restore a partition by using testdisk.
2. Press the *P* key to list files.
   The following figure shows the command output.



3. Select the file that you want to restore and press the *C* key.
4. Select the destination directory. In this example, the file is restored to the */home* directory.



If `Copy done! 1 ok, 0 failed` is displayed, the file is copied, as shown in the following figure.

```
   * Linux                          0   32 33    652 180 40    10483712
Directory /
Copy done! 1 ok, 0 failed
 drwxr-xr-x     0     0      4096 21-Feb-2017 11:57 .
 drwxr-xr-x     0     0      4096 21-Feb-2017 11:57 ..
 drwx------     0     0     16384 21-Feb-2017 11:56 lost+found
 -rw-r--r--     0     0      1701 21-Feb-2017 11:57 install_edsd.sh
 -rw-r--r--     0     0      5848 21-Feb-2017 11:57 install.sh
>-rw-r--r--     0     0     12136 21-Feb-2017 11:57 ip.gz
 -rw-r--r--     0     0         0 21-Feb-2017 11:57 test
 drwxr-xr-x     0     0      4096 21-Feb-2017 11:57 123.sh
 drwxr-xr-x     0     0      4096 21-Feb-2017 11:57 configclient
 drwxr-xr-x     0     0      4096 21-Feb-2017 11:57 data
 drwxr-xr-x     0     0      4096 21-Feb-2017 11:57 diamond
 drwxr-xr-x     0     0      4096 21-Feb-2017 11:57 logs
```

5. Switch to the */home* directory to view details.
   If the file is displayed, as shown in the following figure, the file is restored.

```
[root@Aliyun /]# ls /home/
admin  ip.gz
[root@Aliyun /]#
```

Common mistakes and best practices

Data is the core asset of users. A large number of users build websites and databases such as MySQL, MongoDB, and Redis on ECS instances. Data loss may cause huge risks to businesses. This section describes the common mistakes and best practices in data security.

- Common mistakes

  The underlying storage of Alibaba Cloud is based on triplicate technology. Therefore, some users consider that no risk of data loss in the operating system exists. This is a misunderstanding. The three copies of data stored in the underlying layer provide physical layer protection for data disks. However, if errors occur to the cloud disk logic in the system, such as infection with viruses, accidental data deletion, and file system damage, the data may still be lost. You must use technologies such as snapshots and geo-redundancy to ensure data security. For more information about three copies, see Triplicate storage.

- Best practices

  Data disk partition restoration and data restoration are the final solutions to data loss problems, but they may not restore data as expected. We recommend that you follow the best practices to create automatic or manual snapshots for data and run different backup schemes to maximize your data security.

  - Apply automatic snapshot policies

    Automatic snapshot policies are applied to system and data disks to create automatic snapshots for the disks. Note that after the system disk is replaced, the instance expires, or the disk is manually released, the corresponding automatic snapshots may be released.

    If you want automatic snapshots of a disk to be released along with the disk, you can select **Delete Automatic Snapshots While Releasing Disk** in the **Modify Disk Property** dialog box in the ECS console. If you want to retain the automatic snapshots, you can clear this option.

    For more information, see Snapshot FAQ and Create an automatic snapshot policy.

- Create manual snapshots

  Before you perform important or high-risk operations, you must manually create snapshots for the disk. These operations include:

  - Update the kernel.
  - Upgrade or change applications.
  - Restore data on disks.

  Before you restore a disk, you must create a snapshot for the disk. After the snapshot is created, you can perform other operations.

- OSS backup, offline backup, and geo-redundancy

  You can back up important data by using OSS backup, offline backup, or geo-redundancy.

6.4. Restore data in Windows instances

When you troubleshoot disks, you may encounter the loss of data disk partitions. This topic describes data disk partition loss in Windows and the solutions. This topic also describes the common mistakes and best practices for using disks to avoid the risk of data loss.

Prerequisites

- An Alibaba Cloud account is created. To create an Alibaba Cloud account, go to the account registration page.
- A snapshot is created for the data disk that lost a partition. If errors occur during data restoration, you can use the snapshot to roll back the data disk to the state before the restoration. For more information, see Create a normal snapshot and Roll back a disk by using a snapshot.

Context

In a Windows instance, you can use one of the following tools to restore data on a data disk:

- Disk Management: a tool provided by Windows for partitioning and formatting data disks.
- Data restoration software: typically, commercial software. You can download the software from the official websites. The software is used for restoring data in file systems to which exceptions occur.

Status of the disk is Foreign and no partitions are displayed

In **Disk Management** of Windows, the disk is in the **Foreign** state and no partitions are displayed.

Solution:

Right-click the space next to **Foreign**, select **Import Foreign Disks**, and click **OK**.

Status of the disk is Offline and no partitions are displayed

In **Disk Management** of Windows, the disk is in the **Offline** state and no partitions are displayed.

Solution:

Right-click the space next to **Offline**, select **Online**, and then click **OK**.

No driver letter assigned

In **Disk Management** of Windows, you can find the information of the data disk, but no drive letter is assigned to the data disk.

Solution:

Right-click the primary partition of the disk (such as **Disk 1**), select **Change Drive Letter and Paths**, and then follow the prompt to complete other operations.

Error occurred during storage enumeration

In **Disk Management** of Windows, you cannot view data disks. An error message similar to **An error occurred during storage enumeration** is reported in the system log.

> Note The reported content may be An error occurred during enumeration of volumes based on your operating system version.

Solution:

1. Start Windows PowerShell.
2. Run the **winrm quickconfig** command to restore data. When **Make these changes [y/n]?** is displayed, enter *y* to run the command.

After the restoration, you can find the data disk in **Disk Management**.

Data disk in the RAW format

In some cases, you may find that the data disk in Windows is in the RAW format.

If the file system of a disk is unrecognizable to Windows, the disk is displayed in the RAW format. Typically, this occurs when the partition table or boot sector that records the type or location of the file system is lost or damaged. The following common causes may lead to the loss or damage:

- **Safely remove hardware** is not used when the external disk is disconnected.
- Disk problems caused by power outages or unexpected shutdowns.
- Hardware faults occur.
- Underlying disk-related drivers or applications. For example, DiskProbe can be used to directly modify the disk table structure.
- Computer viruses.

For information about how to restore data disks, visit Dskprobe Overview in the Microsoft documentation.

Windows also contains a large variety of free or commercial data restoration software to restore lost data. For example, you can use Disk Genius to scan and restore expected documents.

Common mistakes and best practices

Data is the core asset of users. A large number of users build websites and databases such as MySQL, MongoDB, and Redis on ECS instances. Data loss may cause huge risks to businesses. The following section describes the common mistakes and best practices in data security.

- Common mistakes

The underlying storage of Alibaba Cloud is based on triplicate technology. Therefore, some users consider that data loss will not occur in the operating system. This is a misunderstanding. The three copies of data stored in the underlying layer provide physical layer protection for data disks. However, if errors occur to the cloud disk logic in the system, such as infection with viruses, accidental data deletion, and file system damage, the data may still be lost. In this case, you must use technologies such as snapshots and geo-redundancy to ensure data security.

- Best practices

  Data disk partition restoration and data restoration are the final solutions to data loss problems, but they may not restore data as expected. We recommend that you follow the best practices to create automatic or manual snapshots for data and run different backup schemes to maximize your data security.

  - Apply automatic snapshot policies

    Automatic snapshot policies are applied to system and data disks to create automatic snapshots for the disks. Note that after the system disk is replaced, the instance expires, or the disk is manually released, the corresponding automatic snapshots may be released.

    If you want automatic snapshots of a disk to be released along with the disk, you can select **Delete Automatic Snapshots While Releasing Disk** in the **Modify Disk Property** dialog box in the ECS console. If you want to retain the automatic snapshots, you can clear this option.

    For more information, see Snapshot FAQ and Delete automatic snapshots while releasing a disk.

  - Create manual snapshots

    Before you perform important or high-risk operations, you must manually create snapshots for disks. These operations include:

    - Update the kernel.
    - Upgrade or change applications.
    - Restore data on disks.

    Before you restore a disk, you must create a snapshot for the disk. After the snapshot is created, you can perform other operations.

  - OSS backup, offline backup, and geo-redundancy

    You can back up important data by using OSS backup, offline backup, or geo-redundancy.

7.Configuration preference

7.1. Transfer ECS instance data

This topic describes the file transfer principle and typical transfer methods on Unix-like, Linux, and Windows platforms in Alibaba Cloud ECS. Additionally, this topic compares these methods to help you choose appropriate file transfer methods that meet your specific requirements.

File transfer principle

File transfer, also known as file data communication, is a form of information transfer that transmits file data between data sources and data sinks. In a file transfer process, the OS extracts file data to the memory for temporary storage, and then duplicates the data to the destination. Encryption adds a secure layer to a file, while duplication transfers the encrypted file as a whole to another location. Decryption is needed only when a compressed package is opened. A large file cannot be transferred as a whole between hosts immediately because the transfer is a continuous process. If any interruption occurs during the transfer, the file will not exist in the destination path. If multiple files are transferred, they are transferred separately and sequentially. If any interruption occurs during the transfer, the files that are being transferred or have not yet been transferred will fail, but the transferred files are transferred successfully. A compressed package is considered as one file regardless of how many files the package contains.

Multiple file transfer tools, such as Netcat, FTP, SCP, and NFS, can be used to transfer files. The following sections describe the features and usage of some typical file transfer tools.

Netcat

Netcat is a powerful and versatile networking tool with optimal file transfer capabilities.

Parameter descriptions

Parameter	Description
-g <gateway>	Specifies up to eight long-distance communication gateways for the router.
-G <number of indicators>	Specifies the number of source routing indicators. The value is a multiple of 4.
-i <delay in seconds>	Specifies the time interval for sending messages and scanning the communications port.
-l	Enables the listening mode to control received data.
-o <output file>	Specifies the name of the file where the transferred data is dumped and saved in hexadecimal character codes.
-P <communication port>	Specifies the communication port used by the local host.
-r	Specifies the communication port between the local host and the remote host.
-u	Enables the UDP transfer protocol.

Parameter	Description
-v	Shows the command running process.
-w <timeout in seconds>	Specifies the waiting time for a connection.
-z	Enables the zero input/output mode, which is used only for scanning the communications port.
-n	Uses IP addresses instead of the DNS.

Examples of usage

1. Scan ports 21-24 (for example, IP address 192.168.2.34).

```
nc -v -w 2 192.168.2.34 -z 21-24
```

Response example:

```
nc: connect to 192.168.2.34 port 21 (tcp) failed: Connection refused
Connection to 192.168.2.34 22 port [tcp/ssh] succeeded!
nc: connect to 192.168.2.34 port 23 (tcp) failed: Connection refused
nc: connect to 192.168.2.34 port 24 (tcp) failed: Connection refused
```

2. Copy files from 192.168.2.33 to 192.168.2.34.

- Run the following command at 192.168.2.34: `nc-l 1234 > test.txt` .
- Run the following command at 192.168.2.33: `nc192.168.2.34 < test.txt` .

3. Run the following nc commands to operate Memcached as needed:

- To store data, run the command `printf “set key 0 10 6rnresultrn” |nc 192.168.2.34 11211` .
- To obtain data, run the command `printf “get keyrn” |nc 192.168.2.34 11211` .
- To delete data, run the command `printf “delete keyrn” |nc 192.168.2.34 11211` .
- To view the status, run the command `printf “statsrn” |nc 192.168.2.34 11211` .
- To simulate the top command to view the status, run the command `watch “echo stats” |nc 192.168.2.34 11211` .
- To clear the cache, run the following command:

```
printf “flush_allrn” |nc 192.168.2.34 11211     # This operation cannot be undone.
```

SCP

The use of Secure Copy (SCP) commands is similar to that of RCP commands. The difference is that SCP commands provide higher security protection by prompting users to enter a password for verification. Therefore, we recommend that you use SCP commands instead of RCP commands. SCP commands use SSH to transfer data and use the same authentication model as SSH to provide the same security protection. SSH is a reliable protocol that provides security for remote logon sessions and other network services. With SSH, you can effectively prevent information leakage during remote management. SCP is an SSH-based application. Therefore, it requires that the machines involved in data transfer support SSH.

Features

Similar to RCP, SCP can retain the file attributes on a specific file system and retain the copied subdirectories that need recursion.

SCP provides better file transfer confidentiality. Overall, SCP is suitable for users with high data security requirements.

Examples of usage

If you do not want to enter your username and password every time you use SCP commands to copy files between two machines, you can configure SSH.

To generate an RSA key, run the following command:

```
```

When you are prompted to enter the path and password to save the key, you can press Enter to use the default path and a null password. Then, the generated public key is saved in /.ssh/id_rsa.pub, and the private key is saved in /.ssh/id_rsa. You can copy the content of the public key from this key pair to the /.ssh/authorized_keys file in the machine that you want to access. In this way, you do not need to enter your password when you next access this machine.

Copy a file between two Linux hosts

Basic command format:

```
scp [optional parameter] file_source file_target
```

To copy a file from a local directory to a remote directory, run one of the following four commands:

```
scp local_file remote_username@remote_ip:remote_folder
scp local_file remote_username@remote_ip:remote_file
scp local_file remote_ip:remote_folder
scp local_file remote_ip:remote_file
```

> **Note** In the first and second commands, user names are specified and the password must be entered after the commands are executed. In the first command, a remote directory is specified and the file name remains the same. In the second command, a file name is specified.

In the third and fourth commands, user names are not specified and the password must be entered after the commands are executed. In the third command, a remote directory is specified and the file name remains the same. In the fourth command, a file name is specified.

To copy a file from a remote directory to a local directory, run the following commands:

```
scp root@www.cumt.edu.cn:/home/root/others/music /home/space/music/i.mp3
scp -r www.cumt.edu.cn:/home/root/others/ /home/space/music/
```

Rsync

Rsync is a file synchronization and transfer tool for Linux or Unix. As an alternative to RCP, Rsync may be used through RSH or SSH, or run in daemon mode. In daemon mode, the Rsync server opens port 873 for client connections. During client connections, the Rsync server will verify the password. If the password is correct, the file can be transferred. During the first connection, the entire file is transferred. During the subsequent connections, only incremental data of the file is synchronized.

Rsync Installation methods

> **Note** You can use the package manager of your OS to install Rsync.

```
sudo apt-get  install  rsync     # Install Rsync online in Debian and Ubuntu.
slackpkg  install  rsync         # Install Rsync online using Slackware.
yum install rsync                # Install Rsync in Fedora and Red Hat.
```

Install Rsync through source code compilation:

```
wget http://rsync.samba.org/ftp/rsync/src/rsync-3.0.9.tar.gz
tar xf rsync-3.0.9.tar.gz
cd rsync-3.0.9
./configure && make && make install
```

Parameter descriptions

Parameter	Description
-v	Specifies the output mode.
-a	Specifies the archive mode. It meas that files are transferred recursively and all file attributes are retained. This parameter is equivalent to the combined parameter -rlptgoD.
-r	Transfers subdirectories recursively.
-l	Retains soft links.
-p	Retains file permissions.
-t	Retains file time information.
-g	Retains file group information.

Parameter	Description
-o	Retains file owner information.
-D	Retains device file information.
-H	Retains hard links.
-S	Processes sparse files explicitly to save space for DST files.
-z	Compresses backup files during transfer.

Six work modes of Rsync

- To copy local files from the /home/coremail directory to the /cmbak directory, run the following command:

```
rsync -avSH /home/coremail/ /cmbak/
```

- To copy files from a local machine to a remote machine, run the following command:

```
rsync -av /home/coremail/ 192.168.11.12:/home/coremail/
```

- To copy files from a remote machine to a local machine, run the following command:

```
rsync -av 192.168.11.11:/home/coremail/ /home/coremail/
```

- To copy files from a remote Rsync server (running in daemon mode) to a local machine, run the following command:

```
rsync -av root@172.16.78.192::www /databack
```

- To copy files from a local machine to a remote Rsync server (running in daemon mode), run the following command. This work mode is started when the DST path information contains the “::” delimiter.

```
rsync -av /databack root@172.16.78.192::www
```

- To show the file list of a remote machine, run the following command:

```
rsync -v rsync://192.168.11.11/data
```

Description of the Rsync configuration file

```
cat/etc/rsyncd.conf          # The contents are as follows:
port = 873                   # Specify the port number.
uid = nobody                 # Specify the UID of the daemon process when the module transfers files.
gid = nobody                 # Specify the GID of  the daemon process when the module transfers files.
use chroot = no              # Use chroot to enter the directories in the file system.
max connections = 10         # Specify the maximum concurrent connections.
strict modes = yes           # Specify whether to check the permissions of password-protected files.
pid file = /usr/local/rsyncd/rsyncd.pid      # Specify PID files.
lock file = /usr/local/rsyncd/rsyncd.lock    # Specify the lock file that supports the maximum concurren
t connections. By default, the lock file is /var/run/rsyncd.lock.
motd file = /usr/local/rsyncd/rsyncd.motd    #Define server information and write the rsyncd.motd file
.
log file = /usr/local/rsyncd/rsync.log       # Specify the log of the Rsync server.
log format = %t %a %m %f %b
syslog facility = local3
timeout = 300
[conf]                       # custom module
path = /usr/local/nginx/conf         # Specify the directory to be backed up.
comment = Nginx conf
ignore errors                # Ignore some IO errors.
read only = no               # To allow the client to upload files, set the value to no. Otherwise, set t
he value to yes.
write only = no              # To allow the client to download files, set the value to no. Otherwise, se
t the value to yes.
hosts allow = 192.168.2.0/24         # Specify an allowed IP address.
hosts deny = *               # Specify a denied IP address.
list = false                 # Use the module list upon request.
uid = root
gid = root
auth users = backup          # Specify a connection user name, which is irrelevant to Linux user na
mes.
secrets file = /etc/rsyncd.pass      # Specify the password file.
```

7.2. Increase data throughput through read/write splitting

Typically, system performance decreases when reads and writes occur in the same database server. To improve overall system performance and optimize user experience, you can reduce the load of your primary database through read/write splitting. This topic describes how to use MySQL Proxy to split read and write operations.

Prerequisites

An Alibaba Cloud account is created. To create an Alibaba Cloud account, go to the account registration page.

Context

At the application layer, read/write splitting is implemented through coding. Before you enter the service layer, Aspect-Oriented Programming (AOP) is used to determine whether to use the read database or the write database. The method names can be used to implement the target action. For example, the read database is used for method names that start with query, find, or get, and the write database is used for others.

Advantages:

- The program automatically switches among multiple data sources with ease.
- Middleware is not required.
- Theoretically, all databases are supported.

Disadvantages:

- Manual operations are not supported.
- Data sources cannot be dynamically added.

You can use one of the following methods to split read and write operations at the system layer:

- Distributed Relational Database Service (DRDS)
- MySQL Proxy

The following section describes how to use MySQL Proxy to split read and write operations.

MySQL Proxy is a simple program that is situated between your client and MySQL server and can monitor, analyze, or transform communication between the server and the client. It can serve a wide variety of purposes, such as load balancing, fault query and analysis, and query filtering and modification.

The following figure shows the principle of MySQL Proxy.

MySQL Proxy is an intermediate-layer proxy that acts as a connection pool to forward connection requests from frontend applications to the backend database. MySQL Proxy can perform complex connection control and filtering to implement read/write splitting and load balancing by using the Lua script. MySQL Proxy allows applications to access the backend database smoothly. The applications only need to be connected to the listening port of MySQL Proxy. In this case, the proxy server may become a single point of failure (SPOF). You can use multiple proxy servers to implement redundancy. Therefore, you only need to configure multiple proxy connections in the connection pool of the application server.

Advantages:

- Read/write splitting can be implemented without modifying the source program.
- Data sources can be added dynamically without restarting the program.

Disadvantages:

- The program relies on the middleware, which makes it difficult to switch databases.
- Performance decreases because the middleware serves as a forwarding proxy.

Procedure

Perform the following operations to use MySQL Proxy to split read and write operations:

1. Step 1. Preparations
2. Step 2. Configure read/write splitting
3. Step 3. Grant permissions
4. Step 4. Test read/write splitting

Step 1. Preparations

The following section describes the environment:

- Primary database IP address: 121.40.xx.xx
- Secondary database IP address: 101.37.xx.xx
- MySQL Proxy IP address: 116.62.xx.xx

Perform the following operations to prepare for the installation:

1. Create three ECS instances and install MySQL.
2. Build primary/secondary databases and ensure data consistency between them.
3. Modify the MySQL configuration file of the primary/secondary environment.
   - Primary environment:

     ```
     vim /etc/my.cnf
     [mysqld]
     server-id=202              #Set the unique ID of the server. The default ID is 1.
     log-bin=mysql-bin          # Enable binary logs.
     ```

   - Secondary environment:

     ```
     [mysqld]
     server-id=203
     ```

4. Restart the MySQL service on the primary/secondary servers.

   ```
   /etc/init.d/mysqld restart
   ```

5. Create an account on the primary server and grant permissions to the secondary server.

   ```
   mysql -uroot -p95c7586783
   grant replication slave on *.* to 'syncms'@'Enter secondary-IP address' identified by '123456';
   flush privileges;
   ```

6. Check the status of the primary database.

   ```
   mysql> show master status;
   ```

7. Configure the secondary database.

```
change master to master_host='Enter primary-IP address', master_user='syncms',     master_pass
word='123456', master_log_file='mysql-bin.000005', master_log_pos=602;
```

8. Start the secondary synchronization process and check the status.

```
start slave;
show slave status\G
```

9. Verify the synchronization between the primary and secondary databases.
   i. Write data to the *testproxy.test1* table in the primary database.

```
mysql> create database testproxy;
mysql> create table testproxy.test1(ID int primary key,name char(10) not null);
mysql> insert into testproxy.test1 values(1,'one');
mysql> insert into testproxy.test1 values(2,'two');
mysql> select * from testproxy.test1;
```

   ii. Run the following command in the secondary database to query data in the *testproxy.test1* table:

```
select * from testproxy.test1;
```

   If the content in *testproxy.test1* is the same as that in the primary database, data is synchronized between the the primary and secondary databases.

Step 2. Configure read/write splitting

Perform the following operations to configure read/write splitting:

1. Install MySQL Proxy.

```
wget https://cdn.mysql.com/archives/mysql-proxy/mysql-proxy-0.8.5-linux-glibc2.3-x86-64bit.tar.g
z
mkdir /alidata
tar xvf mysql-proxy-0.8.5-linux-glibc2.3-x86-64bit.tar.gz
mv mysql-proxy-0.8.5-linux-glibc2.3-x86-64bit/  /alidata/mysql-proxy-0.8.5
```

2. Set environment variables.

```
vim /etc/profile                #Add the following information:
PATH=$PATH:/alidata/mysql-proxy-0.8.5/bin
export $PATH
source /etc/profile             #Validate the environment variables.
mysql-proxy -V
```

3. Set the read/write splitting parameters.

```
cd /alidata/mysql-proxy-0.8.5/share/doc/mysql-proxy/
vim  rw-splitting.lua
```

MySQL Proxy will detect client connections. If the number of connections does not exceed the preset value of min_idle_connections, read/write splitting will not be performed. By default, read/write splitting will be performed for at least four connections and at most eight connections. To simplify the test of read/write splitting, set the number of connections to one at least and two at most. For the production environment, you can set the number based on the actual conditions.

Before the modification:

After the modification:

4. Copy the Lua administration script *admin.lua* to the directory where the read/write splitting script *rw-splitting.lua* is located.

```
cp /alidata/mysql-proxy-0.8.5/lib/mysql-proxy/lua/admin.lua /alidata/mysql-proxy-0.8.5/share/doc/mysql-proxy/
```

Step 3. Grant permissions

Perform the following operations to grant permissions:

1. Grant permissions in the primary database. The permissions will also be granted in the secondary database due to synchronization between the primary and secondary databases.

```
mysql -uroot -p95c7586783
grant all on *.* to 'mysql-proxy'@'Enter <MySQL Proxy IP>' identified by '123456';
flush privileges;
```

2. Start MySQL Proxy.

```
mysql-proxy --daemon --log-level=debug --log-file=/var/log/mysql-proxy.log --plugins=proxy -b Enter <primary-IP address>:3306 -r Enter secondary-IP:3306 --proxy-lua-script="/alidata/mysql-proxy-0.8.5/share/doc/mysql-proxy/rw-splitting.lua" --plugins=admin --admin-username="admin" --admin-password="admin" --admin-lua-script="/alidata/mysql-proxy-0.8.5/share/doc/mysql-proxy/admin.lua"
```

3. Check the port and related processes.

```
netstat -tpln
```

```
ps -ef | grep mysql
```

Step 4. Test read/write splitting

Perform the following operations to test read/write splitting:

1. Disable secondary replication.

```
stop slave;
```

2. Log on to the backend of MySQL Proxy.

```
mysql  -u  admin -padmin -P 4041 -h MySQL-Proxy-IP
select * from backends;              #Check the status.
```

The first connection will be established to the primary database.

```
mysql -umysql-proxy -p123456 -h 116.62.xx.xx -P 4040
insert into testproxy.test1 values(3,'three');          #Add a data record. Secondary replication is di
sabled. Therefore, the record exists in the primary database but does not exist in the secondary
database.
```

Create additional test connections. If the data displayed in the *testproxy.test1* table in the primary database is the same as that in the secondary database, read/write splitting is successful.

```
mysql -umysql-proxy -p123456 -h 116.62.xx.xx -P 4040
select * from testproxy.test1;
```

7.3. Change the preferred language of a Windows instance

This topic describes how to download a language pack from Windows Update and set the language as the preferred language of a Windows instance. An ECS instance that runs Windows Server 2016 (English) is used in the examples in this topic.

Prerequisites

An Alibaba Cloud account is created. To create an Alibaba Cloud account, go to the account registration page.

Context

Alibaba Cloud ECS provides only Chinese and English editions of Windows Server public images. If you want to use another language on an ECS instance, such as Arabic, German, Russian, or Japanese, you can download the language pack from Windows Update and set the language as the preferred language of the instance. German is used in the examples in the following procedure. This procedure applies to instances that run Windows Server 2012 or later. After the preferred language of the instance is changed to German, you can use the instance to create a custom image. The custom image also uses German and German keyboard settings. You can then create as many instances as required from this custom image.

Procedure

1. Connect to the target Windows instance. For more information, see Overview.
2. Open the PowerShell module.
3. Run the following commands to temporarily disable Windows Server Update Services (WSUS):

```
Set-ItemProperty -Path 'HKLM:\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate\AU' -Na
me UseWUServer -Value 0
Restart-Service -Name wuauserv
```

4. Open the Control Panel, and choose **Clock, Language, and Region** > **Language** > **Add a language**.
5. In the **Add languages** dialog box, select a language and click **Add**. In this example, **Deutsch (German)** > **Deutsch (Deutschland)** is selected.

6. Select the language and click **Move up** to change the language priority. In this example, **Deutsch (Deutschland)** is selected.
7. Click **Options** next to the selected language to check for language updates online.

8. Wait for three minutes while the instance checks for updates. If an update is available for download, click **Download and install language pack**.

Wait until the installation is complete.

9. Restart the instance by using the ECS console. For more information, see Restart an instance.
10. Connect to the Windows instance again.
The display language is now Deutsch (German).
11. Open the PowerShell ISE module and run the following commands to enable WSUS:

```
Set-ItemProperty -Path 'HKLM:\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate\AU' -Na
me UseWUServer -Value 1
Restart-Service -Name wuauserv
```

12. Open **Windows Update**, check for security updates, and re-install all the security updates that are already installed before the language change.

What's next

Create multiple instances with the same language settings.

1. Log on to the ECS console.
2. Create a custom image from the Windows instance with the new display language. For more information, see Create a custom image from an instance.
3. Create a specified number of instances from the custom image. For more information, see Create an ECS instance by using a custom image.

7.4. Boot a Linux ECS instance into single user mode

This topic describes how to boot an ECS instance that is created from a CentOS, Debian, SUSE Linux Enterprise Server (SLES), or Ubuntu image into single user mode.

Prerequisites

- An Alibaba Cloud account is created. To create an Alibaba Cloud account, go to the account registration page.
- An ECS instance is created. For more information, see Creation method overview. In this example, an ECS instance of the ecs.g6.large instance type is created.

Context

Single user mode is one of the modes in which Linux distributions are booted. GRUB can be used to boot a Linux distribution into single user mode. After the system enters single user mode, you will have system administrator permissions and can modify all system configurations. This mode is usually used in the following scenarios:

- Change the system password
- Troubleshoot boot failures
- Fix system exceptions
- Maintain partitions of hard disk drives (HDD)

> **Notice** In single user mode, you can modify critical system configurations. We recommend that you set this mode only when necessary and proceed with caution.

References

- Example 1: How a CentOS ECS instance enters single user mode
- Example 2: How a Debian ECS instance enters single user mode
- Example 3: How a SLES instance enters single user mode
- Example 4: How an Ubuntu ECS instance enters single user mode

Example 1: How a CentOS ECS instance enters single user mode

In this example, an ECS instance running a CentOS 8.0 64-bit operating system is used.

1. Connect to the ECS instance.For more information, see Overview.
2. Run the `reboot` command to restart the ECS instance. When the interface for selecting a boot system appears, press the *E* key to go to the configuration interface for boot options.The following figure shows the configuration interface for boot options.

   
3. Move the pointer to the line that starts with `linux` by using the arrow keys on the keyboard. Replace the content from `ro` to the end of the line with `rw init=/bin/sh crashkernel=auto` .

The following figure shows the information after the change is made.

4. Press the *Ctrl+X* composite key or the *F10* key.The system enters single user mode. The following figure shows the interface for resetting the system password.

Example 2: How a Debian ECS instance enters single user mode

In this example, an ECS instance running a Debian 10.2 64-bit operating system is used.

1. Connect to the ECS instance.For more information, see Overview.
2. Run the `reboot` command to restart the ECS instance. When the configuration interface for kernel options appears, press the *E* key to go to the GRUB interface.The following figure shows the GRUB interface.

3. Move the pointer to the line that starts with `linux` by using the arrow keys on the keyboard. Append `single` to the end of the line.
   The following figure shows the information after the change is made.

4. Press the *Ctrl+X* composite key or the *F10* key to start the system. Enter the password of the root user.The system enters single user mode.

Example 3: How a SLES instance enters single user mode

In this example, an ECS instance running a SLES 15 SP1 64-bit operating system is used.

1. Connect to the ECS instance.For more information, see Overview.
2. Run the `reboot` command to restart the ECS instance. When the configuration interface for kernel options appears, press the *E* key to go to the GRUB interface.The following figure shows the GRUB interface.

3. Move the pointer to the line that starts with `linux` by using the arrow keys on the keyboard. Append `single` to the end of the line.
   The following figure shows the information after the change is made.

4. Press the *Ctrl+X* composite key or the *F10* key to start the system. Enter the password of the root user.The system enters single user mode.

Example 4: How an Ubuntu ECS instance enters single user mode

In this example, an ECS instance running an Ubuntu 18.04 64-bit operating system is used.

1. Connect to the ECS instance.For more information, see Overview.
2. Run the `reboot` command to restart the ECS instance. During the restarting process, press

the *Shift* key to go to the GRUB interface.The following figure shows the GRUB interface.

3. Select the Advanced options for Ubuntu in the second line of the GRUB interface and press the *Enter* key.
4. Select the recovery mode in the second line on the interface that appears and press the *E* key to edit the boot options.

5. On the editing page, move the pointer to the line that starts with `linux` by using the arrow keys on the keyboard. Replace the content from `ro` to the end of the line with `rw single init=/bin/bash` .The following figure shows the information after the change is made.

6. Press the *Ctrl+X* composite key or the *F10* key.The system enters single user mode. The following figure shows the interface for resetting the system password.

8.Block Storage

8.1. Resize partitions and file systems of Linux system disks

This topic describes how to use the growpart and resize2fs tools to resize partitions and file systems of Linux system disks.

Prerequisites

Before you resize partitions and file systems of system disks, make sure that the following requirements are met:

1. A snapshot is created to back up data.

   To prevent data loss caused by incorrect operations, we recommend that you create a snapshot to back up your data. For more information, see Create a normal snapshot.

2. A data disk is resized in the console.

   If no data disks are resized, see Resize disks online for Linux instances or Resize disks offline for Linux instances to resize a data disk.

3. Remotely connect to an ECS instance. For more information, see Overview.
4. The growpart or xfsprogs tool is installed based on your operating system.
   - In CentOS 7 or Alibaba Cloud Linux, run the following commands to install the tools:

     ```
     yum install cloud-utils-growpart
     yum install xfsprogs
     ```

   - In Ubuntu 14, Ubuntu 16, Ubuntu 18, or Debian 9, run the following commands to install the tools:

     ```
     apt install cloud-guest-utils
     apt install xfsprogs
     ```

   - In Debian 8, OpenSUSE 42.3, OpenSUSE 13.1, or SUSE Linux Enterprise Server 12 SP2, use an upstream version of growpart or xfsprogs.

     > **Note** If the partition fails to be resized due to a formatting tool error, you can reinstall the tool.

5. The kernel version of your instance is checked, such as by running the `uname -a` command.
   - For more information about kernels 3.6.0 or later, see Procedure for instances with kernels 3.6.0 or later.
   - For more information about kernels earlier than 3.6.0, see Procedure for instances with kernels earlier than 3.6.0. If your instance runs a Linux distribution such as CentOS 6, Debian 7, or SUSE Linux Enterprise Server 11 SP4, you can resize partitions after you restart the instance from the console or by calling an API operation.

Context

The procedures in this topic apply to system disks with the following partition and file system formats:

- Partition formats: MBR and GPT
- File system formats: ext*, XFS, and Btrfs

Procedure for instances with kernels 3.6.0 or later

This section uses an instance running CentOS 7 as an example to describe how to resize a system disk partition and file system.

1. Run the `fdisk -l` command to check the current disk size.

   The following example shows that the */dev/vda* disk size is 100 GiB:

   ```
   [root@ecshost ~]# fdisk -l
   Disk /dev/vda: 107.4 GB, 107374182400 bytes, 209715200 sectors
   Units = sectors of 1 * 512 = 512 bytes
   Sector size (logical/physical): 512 bytes / 512 bytes
   I/O size (minimum/optimal): 512 bytes / 512 bytes
   Disk label type: dos
   Disk identifier: 0x000bad2b

     Device Boot      Start         End      Blocks   Id  System
   /dev/vda1   *        2048    83886046    41941999+  83  Linux
   ```

2. Run the `df -h` command to check the partition size and the file system type.

   The following example shows that the */dev/vda1* partition size is 40 GiB, and the file system type is ext4:

   ```
   [root@ecshost ~]# df -Th
   Filesystem     Type      Size  Used Avail Use% Mounted on
   devtmpfs       devtmpfs  869M     0  869M   0% /dev
   tmpfs          tmpfs     879M     0  879M   0% /dev/shm
   tmpfs          tmpfs     879M  460K  878M   1% /run
   tmpfs          tmpfs     879M     0  879M   0% /sys/fs/cgroup
   /dev/vda1      ext4       40G  1.8G   36G   5% /
   tmpfs          tmpfs     176M     0  176M   0% /run/user/0
   ```

3. Run the `growpart <DeviceName> <PartionNumber>` command to resize the partition.

   The following example shows that the first partition */dev/vda1* of the system disk is resized:

```
[root@ecshost ~]# growpart /dev/vda 1
CHANGED: partition=1 start=2048 old: size=83883999 end=83886047 new: size=209713119 end=209
715167
```

4. Resize the file system.

   You can use one of the following methods based on your file system type:

   - If the file system is ext*, such as ext3 and ext4, run the `resize2fs <PartitionName>` command.

     The following example shows that the file system of the */dev/vda1* partition is resized:

     ```
     [root@ecshost ~]# resize2fs /dev/vda1
     resize2fs 1.42.9 (28-Dec-2013)
     Filesystem at /dev/vda1 is mounted on /; on-line resizing required
     old_desc_blocks = 3, new_desc_blocks = 7
     The filesystem on /dev/vda1 is now 26214139 blocks long.
     ```

   - If the file system is XFS, run the `xfs_growfs <mountpoint>` command.

     The following example shows that the file system of the */dev/vda1* partition is resized. The mount point of the */dev/vda1* partition is the root directory */*.

     ```
     [root@ecshost ~]# xfs_growfs /
     meta-data=/dev/vda1              isize=512    agcount=13, agsize=1310656 blks
          =                       sectsz=512   attr=2, projid32bit=1
          =                       crc=1        finobt=1, sparse=1, rmapbt=0
          =                       reflink=1
     data     =                       bsize=4096   blocks=15728379, imaxpct=25
          =                       sunit=0      swidth=0 blks
     naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
     log      =internal log           bsize=4096   blocks=2560, version=2
          =                       sectsz=512   sunit=0 blks, lazy-count=1
     realtime =none                   extsz=4096   blocks=0, rtextents=0
     data blocks changed from 15728379 to 20971259
     ```

     > **Note** The xfs_growfs commands may vary by version. Run `xfs_growfs --help` to check the corresponding commands.

5. Run the `df -h` command to check the size of the disk partition.

   The following example shows that the */dev/vda1* partition size is 100 GiB, indicating that the partition is resized:

```
[root@ecshost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        869M     0  869M   0% /dev
tmpfs           879M     0  879M   0% /dev/shm
tmpfs           879M  492K  878M   1% /run
tmpfs           879M     0  879M   0% /sys/fs/cgroup
/dev/vda1        99G  1.8G   93G   2% /
tmpfs           176M     0  176M   0% /run/user/0
```

Procedure for instances with kernels earlier than 3.6.0

This section uses an instance running CentOS 6 as an example to describe how to resize a system disk partition and file system.

1. Install the dracut-modules-growroot tool.

   ```
   [root@ecshost ~]# yum install -y dracut-modules-growroot
   ```

   If a package manager other than yum is used, change **yum** in the preceding command based on the package manager.

2. Run the following command to overwrite the existing initramfs file:

   ```
   [root@ecshost ~]# dracut -f
   ```

3. Run the `fdisk -l` command to check the current disk size.

   The following example shows that the */dev/vda1* disk size is 100 GiB:

   ```
   [root@ecshost ~]# fdisk -l
   Disk /dev/vda: 107.4 GB, 107374182400 bytes
   255 heads, 63 sectors/track, 13054 cylinders
   Units = cylinders of 16065 * 512 = 8225280 bytes
   Sector size (logical/physical): 512 bytes / 512 bytes
   I/O size (minimum/optimal): 512 bytes / 512 bytes
   Disk identifier: 0x0003a7b4

     Device Boot      Start         End      Blocks   Id  System
   /dev/vda1   *           1        2611    20970496   83  Linux
   ```

4. Run the `df -h` command to check the size of the disk partition.

   The following example shows that the */dev/vda1* partition size is 20 GiB:

```
[root@ecshost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           7.8G     0  7.8G   0% /dev/shm
/dev/vda1        20G  1.1G   18G   6% /
```

5. Run the `growpart <DeviceName> <PartionNumber>` command to resize the partition.

   The following example shows that the first partition */dev/vda1* of the system disk is resized:

   ```
   [root@ecshost ~]# growpart /dev/vda 1
   CHANGED: partition=1 start=2048 old: size=41940992 end=41943040 new: size=209710462,end=209
   712510
   ```

6. Restart the instance from the console or by calling the RebootInstance operation. For more information about the procedure, see Restart an instance and RebootInstance.
7. Connect to the instance.
8. Resize the file system.

   You can use one of the following methods based on your file system type:

   - If the file system is ext*, such as ext3 and ext4, run the `resize2fs <PartitionName>` command.

     The following example shows that the file system of the */dev/vda1* partition is resized:

     ```
     [root@ecshost ~]# resize2fs /dev/vda1
     resize2fs 1.41.12 (17-May-2010)
     Filesystem at /dev/vda1 is mounted on /; on-line resizing required
     old desc_blocks = 2, new_desc_blocks = 7
     Performing an on-line resize of /dev/vda1 to 26213807 (4k) blocks.
     The filesystem on /dev/vda1 is now 26213807 blocks long.
     ```

   - If the file system is XFS, run the `xfs_growfs <mountpoint>` command.

     The following example shows that the file system of the */dev/vda1* partition is resized. The mount point of the */dev/vda1* partition is the root directory */*.

     ```
     [root@ecshost ~]# xfs_growfs /
     ```

     > **Note** The xfs_growfs commands may vary by version. Run `xfs_growfs --help` to check the corresponding commands.

9. Run the `df -h` command to check the size of the disk partition.

   The following example shows that the */dev/vda1* partition size is 100 GiB, indicating that the partition is resized:

```
[root@ecshost ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1       99G  1.1G   93G   2% /
tmpfs          7.8G     0  7.8G   0% /dev/shm
```

Related information

- Resize partitions and file systems of Linux data disks

8.2. Resize partitions and file systems of Linux data disks

When you resize disks, only the storage capacity of the disks is extended. File systems of the ECS instances are not affected. Follow the steps in this topic to resize file systems to extend the capacity of ECS instances.

Prerequisites

1. A snapshot is created to back up data.

   To prevent data loss caused by incorrect operations, we recommend that you create a snapshot to back up your data. For more information, see Create a normal snapshot.

2. A data disk is resized in the console.

   If no data disks are resized, see Resize disks online for Linux instances or Resize disks offline for Linux instances to resize a data disk.

3. Remotely connect to an ECS instance. For more information, see Overview.

Context

The following section describes the configurations that are used in this example:

- Operating system of the ECS instance: public image Alibaba Cloud Linux 2.1903 LTS 64-bit
- Data disk: an ultra disk
- Device name of the data disk: */dev/vdb*

Adjust the command and parameter configurations based on the actual operating system and device name of the data disk.

Check the partition table format and the file system type

1. Run the following command to check the partition table format of the data disk:

   ```
   fdisk -lu /dev/vdb
   ```

   In this example, the disk has a partition named */dev/vdb1*.

   - In the case of `"System"="Linux"`, the data disk uses the MBR partition table format.
   - In the case of `"System"="GPT"`, the data disk uses the GPT partition table format.

```
[root@ecshost ~]# fdisk -lu /dev/vdb
Disk /dev/vdb: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x9277b47b

Device Boot Start End Blocks Id System
/dev/vdb1 2048 41943039 20970496 83 Linux
```

2. Run the following command to check the file system type of the existing partition:

```
blkid /dev/vdb1
```

In this example, the file system type of */dev/vdb1* is *ext4*.

```
[root@ecshost ~]# blkid /dev/vdb1
/dev/vdb1: UUID="e97bf1e2-fc84-4c11-9652-73********24" TYPE="ext4"
```

> **Note** No results are returned if a data disk does not have partitions or file systems, or if a data disk has partitions but not file systems.

3. Run the following command to check the status of the file system:
   - ext* file system: `e2fsck -n /dev/vdb1`
   - XFS file system: `xfs_repair -n /dev/vdb1`

> **Notice** In this example, the file system is in the clean state, indicating that the file system is normal. If the file system is not in the clean state, troubleshoot the file system.

```
[root@ecshost ~]# e2fsck -n /dev/vdb1
Warning! /dev/vdb1 is mounted.
Warning: skipping journal recovery because doing a read-only filesystem check.
/dev/vdb1: clean, 11/1310720 files, 126322/5242624 blocks
```

Select a method to resize partitions or file systems

Select a resizing method based on the partition format and file system type.

Scenario	Resizing method

Scenario	Resizing method
The data disk has partitions and file systems.	• To resize the existing MBR partitions of the data disk, see Option 1: Resize existing MBR partitions. • If you want to resize the disk to increase MBR partitions, see Option 2: Add and format MBR partitions. • To resize the existing GPT partitions of the data disk, see Option 3: Resize existing GPT partitions. • If you want to resize the disk to increase GPT partitions, see Option 4: Add and format GPT partitions.
The new data disk does not have partitions or file systems.	After you resize the data disk in the console, see Partition and format a data disk or Partition and format a data disk larger than 2 TiB.
The raw data disk has a file system but no partitions.	After you resize the data disk in the console, see Option 5: Resize the file system of a raw data disk.
The data disk is not attached to any instance.	After you attach the data disk to an instance, follow the steps in this topic to resize the data disk.

Note

- If a data disk contains an MBR partition, the data disk cannot be resized to 2 TiB or larger. To prevent data loss, we recommend that you create a disk larger than 2 TiB. Format a GPT partition and copy the data in the MBR partition to the GPT partition. For more information, see Partition and format a data disk larger than 2 TiB.
- If an error occurs during resizing due to the problem of the resizing and formatting tool, you can upgrade the tool version, or uninstall and reinstall the tool.

Option 1: Resize existing MBR partitions

Note To prevent data loss, we recommend that you do not resize partitions and file systems that are mounted to ECS instances. Unmount the mounted partitions (**umount**) first. After you resize the partitions and they can be normally used, mount the partitions (**mount**) again. We recommend that you use the following methods for different Linux kernel versions:

- If the instance kernel version is earlier than 3.6, unmount the partition, modify the partition table, and then resize the file system.
- If the instance kernel version is 3.6 or later, modify the corresponding partition table, notify the kernel to update the partition table, and then resize the file system.

Perform the following operations to resize an existing MBR partition:

1. **Modify the partition table.**

i. Run the following command to view the partition information and record the start and end sectors of the existing partition:

```
fdisk -lu /dev/vdb
```

In this example, the start sector number of the */dev/vdb1* partition is 2048 and the end sector number is 41943039.

```
[root@ecshost ~]# fdisk -lu /dev/vdb
Disk /dev/vdb: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x9277b47b

Device Boot Start End Blocks Id System
/dev/vdb1 2048 41943039 20970496 83 Linux
```

ii. View the mount path of the data disk. Unmount the partition based on the returned file path and wait until the partition is fully unmounted.

```
[root@ecshost ~]# mount | grep "/dev/vdb"
/dev/vdb1 on /mnt type ext4 (rw,relatime,data=ordered)
[root@ecshost ~]# umount /dev/vdb1
[root@ecshost ~]# mount | grep "/dev/vdb"
```

iii. Run the **fdisk** command to delete the existing partition.

> **Warning** If errors occur when you delete a partition, data stored on the partition may be deleted. To avoid data loss, back up important data such as user data in a database before you delete a partition.

a. Run the `fdisk -u /dev/vdb` command to partition the data disk.
b. Enter *p* to show the partition table.
c. Enter *d* to delete the partition.
d. Enter *p* to confirm that the partition has been deleted.
e. Enter *w* to save changes and exit.

```
[root@ecshost ~]# fdisk -u /dev/vdb
Welcome to fdisk (util-linux 2.23.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.
Command (m for help): p
Disk /dev/vdb: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x9277b47b
Device Boot Start End Blocks Id System
/dev/vdb1 2048 41943039 20970496 83 Linux
Command (m for help): d
Selected partition 1
Partition 1 is deleted
Command (m for help): p
Disk /dev/vdb: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x9277b47b
Device Boot Start End Blocks Id System
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
```

iv. Run the **fdisk** command to create a partition.

a. Run the `fdisk -u /dev/vdb` command to partition the data disk.

b. Enter *p* to show the partition table.

c. Enter *n* to create a partition.

d. Enter *p* to select the primary partition type.

e. Enter *<partition number>* to select the partition number. In this example, 1 is selected.

f. Set the start and end sector numbers for the new partition.

> **Warning** The start sector number of the new partition must be the same as that of the existing partition. The end sector number must be greater than that of the existing partition. Otherwise, the resizing operation will fail.

g. Enter *w* to save changes and exit.

In this example, the */dev/vdb1* partition is resized from 20 GiB to 40 GiB.

```
[root@ecshost ~]# fdisk -u /dev/vdb
Welcome to fdisk (util-linux 2.23.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.
Command (m for help): p
Disk /dev/vdb: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x9277b47b
Device Boot Start End Blocks Id System
Command (m for help): n
Partition type:
p primary (0 primary, 0 extended, 4 free)
e extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-83886079, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-83886079, default 83886079):
Partition 1 of type Linux and of size 40 GiB is set
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
```

v. Notify the kernel that the partition table of the data disk has been updated.Run the `partprobe /dev/vdb` or `partx -u /dev/vdb1` command to notify the kernel that the partition table of the data disk has been modified and the kernel must be synchronized.

vi. Run the `lsblk /dev/vdb` command to verify that the partition table has been resized.

vii. Run the `e2fsck -f /dev/vdb1` command to check the file system again and verify that the file system is in the clean state after the partition is resized.

2. Resize the file system.
   - For an ext* file system such as ext3 or ext4, run the following commands in sequence to resize the ext* file system and remount the partition:

```
[root@ecshost ~]# resize2fs /dev/vdb1
resize2fs 1.43.5 (04-Aug-2017)
Resizing the filesystem on /dev/vdb1 to 7864320 (4k) blocks.
The filesystem on /dev/vdb1 is now 7864320 blocks long.
[root@ecshost ~]# mount /dev/vdb1 /mnt
```

   - For an XFS file system, run the following commands in sequence to remount the partition and then resize the XFS file system:

> **Note** The new version xfs_growfs identifies the device to be resized based on the mount point. Example: `xfs_growfs /mnt`. You can run the `xfs_growfs --help` command to check how to use xfs_growfs of different versions.

```
[root@ecshost ~]# mount /dev/vdb1 /mnt/
[root@ecshost ~]# xfs_growfs /mnt
meta-data=/dev/vdb1              isize=512    agcount=4, agsize=1310720 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=0 spinodes=0
data     =                       bsize=4096   blocks=5242880, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal               bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 5242880 to 7864320
```

Option 2: Add and format MBR partitions

If you want to resize the disk to increase MBR partitions, perform the following operations to resize the file system of the instance:

1. Run the `fdisk -u /dev/vdb` command to create a partition.

   In this example, the */dev/vdb2* partition is created for the newly added 20 GiB disk space.

```
[root@ecshost ~]# fdisk -u /dev/vdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write commad.

Command (m for help): p

Disk /dev/vdb: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x2b31a2a3

   Device Boot      Start         End      Blocks   Id  System
/dev/vdb1            2048    41943039    20970496   83  Linux

Command (m for help): n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): p
Partition number (2-4, default 2): 2
First sector (41943040-83886079, default 41943040):
Using default value 41943040
Last sector, +sectors or +size{K,M,G} (41943040-83886079, default 83886079):
Using default value 83886079
Partition 2 of type Linux and of size 20 GiB is set

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

2. Run the `lsblk /dev/vdb` command to view the partition.

```
[root@ecshost ~]# lsblk /dev/vdb
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
vdb    253:16   0  40G  0 disk
├─vdb1 253:17   0  20G  0 part
└─vdb2 253:18   0  20G  0 part
```

3. Format the new partition.
   - To create an ext4 file system, run the `mkfs.ext4 /dev/vdb2` command.

     ```
     [root@ecshost ~]# mkfs.ext4 /dev/vdb2
     Filesystem label=
     OS type: Linux
     Block size=4096 (log=2)
     Fragment size=4096 (log=2)
     Stride=0 blocks, Stripe width=0 blocks
     1310720 inodes, 5242880 blocks
     262144 blocks (5.00%) reserved for the super user
     First data block=0
     Maximum filesystem blocks=2153775104
     160 block groups
     32768 blocks per group, 32768 fragments per group
     8192 inodes per group
     Superblock backups stored on blocks:
         32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
         4096000

     Allocating group tables: done
     Writing inode tables: done
     Creating journal (32768 blocks): done
     Writing superblocks and filesystem accounting information: done
     [root@ecshost ~]# blkid /dev/vdb2
     /dev/vdb2: UUID="e3f336dc-d534-4fdd-****-b6ff1a55bdbb" TYPE="ext4"
     ```

   - To create an ext3 file system, run the `mkfs.ext3 /dev/vdb2` command.
   - To create an XFS file system, run the `mkfs.xfs -f /dev/vdb2` command.
   - To create a btrfs file system, run the `mkfs.btrfs /dev/vdb2` command.
4. Run the `mount /dev/vdb2 /mnt` command to mount the partition.
5. Run the `df -h` command to check the current capacity and usage of the data disk.

   If information about the new file system is displayed, the partition is mounted.

```
[root@ecshost ~]# df -h
Filesystem Size Used Avail Use% Mounted on
/dev/vda1 40G 1.6G 36G 5% /
devtmpfs 3.9G 0 3.9G 0% /dev
tmpfs 3.9G 0 3.9G 0% /dev/shm
tmpfs 3.9G 460K 3.9G 1% /run
tmpfs 3.9G 0 3.9G 0% /sys/fs/cgroup
/dev/vdb2 9.8G 37M 9.2G 1% /mnt
tmpfs 783M 0 783M 0% /run/user/0
```

Option 3: Resize existing GPT partitions

Perform the following operations to resize an existing GPT partition:

1. View the mount path of the data disk. Unmount the partition based on the returned file path and wait until the partition is fully unmounted.

   ```
   [root@ecshost ~]# mount | grep "/dev/vdb"
   /dev/vdb1 on /mnt type ext4 (rw,relatime,data=ordered)
   [root@ecshost ~]# umount /dev/vdb1
   [root@ecshost ~]# mount | grep "/dev/vdb"
   ```

2. Use the Parted tool to allocate capacity for the existing GPT partition.

   i. Run the `parted /dev/vdb` command to start the **parted** tool.

   To view the Parted tool instructions, run the `help` command.

   ii. Run the `print` command to view the partition information and record the partition number and start sector value of the existing partition.

   If `Fix/Ignore/Cancel?` or `Fix/Ignore?` is displayed, enter *Fix*.

   In this example, the size of the existing partition is 1 TiB. The partition number (the value of `Number`) is `1`. The start sector number (the value of `Start`) is `1049kB`.

   iii. Run the `rm <Partition number>` command to detach the existing partition.

   In this example, the partition number of the existing partition is `1`. Therefore, the following command is used:

   ```
   rm 1
   ```

iv. Run the `mkpart primary <the start sector number of the existing partition> <the percentage of allocated capacity>` command to recreate the primary partition.

In this example, the start sector number of the existing partition is `1049kB` and the 3 TiB total capacity is allocated to the partition. Therefore, the following command is used:

```
mkpart primary 1049kB 100%
```

v. Run the `print` command to check whether the new partition is created.

The following figure shows that the new GPT partition number is still 1, but the capacity of the partition has been increased to 3 TiB.

vi. Run the `quit` command to exit the Parted tool.

The following section shows the complete sample code:

```
[root@ecshost ~]# parted /dev/vdb
GNU Parted 3.1
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Error: The backup GPT table is not at the end of the disk, as it should be.
This might mean that another operating system believes the disk is smaller.
Fix, by moving the backup to the end (and removing the old backup)?
Fix/Ignore/Cancel? Fix
Warning: Not all of the space available to /dev/vdb appears to be used, you can
fix the GPT to use all of the space (an extra 4294967296 blocks) or continue
with the current setting?
Fix/Ignore? Fix
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 3299GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  1100GB  1100GB  ext4         primary

(parted) rm 1
(parted) mkpart primary 1049kB 100%
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 3299GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  3299GB  3299GB  ext4         primary

(parted) quit
Information: You may need to update /etc/fstab.
```

3. Run the `fsck -f /dev/vdb1` command to check the file system for consistency.

```
[root@ecshost ~]# fsck -f /dev/vdb1
fsck from util-linux 2.23.2
e2fsck 1.43.5 (04-Aug-2017)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/vdb1: 11/67108864 files (0.0% non-contiguous), 4265369/268434944 blocks
```

4. Resize the file system corresponding to the partition and remount the partition.
   - For an ext* file system such as ext3 or ext4, run the following commands in sequence to resize the ext* file system of the new partition and remount the partition:

     ```
     [root@ecshost ~]# resize2fs /dev/vdb1
     resize2fs 1.43.5 (04-Aug-2017)
     Resizing the filesystem on /dev/vdb1 to 805305856 (4k) blocks.
     The filesystem on /dev/vdb1 is now 805305856 blocks long.
     [root@ecshost ~]# mount /dev/vdb1 /mnt
     ```

   - For an XFS file system, run the following commands in sequence to remount the partition and then resize the XFS file system.

     > **Note** The new version xfs_growfs identifies the device to be resized based on the mount point. Example: `xfs_growfs /mnt`. You can run the `xfs_growfs --help` command to check how to use xfs_growfs of different versions.

     ```
     [root@ecshost ~]# mount /dev/vdb1 /mnt/
     [root@ecshost ~]# xfs_growfs /mnt
     ```

Option 4: Add and format GPT partitions

If new disk space is added to increase GPT partitions, perform the following operations to resize the file system of the instance: In this example, a data disk of 32 TiB is used. The existing partition */dev/vdb1* has a 4.8 TiB capacity. The */dev/vdb2* partition is to be created.

1. Run the **fdisk** command to view information about the existing partition.

```
[root@ecshost ~]# fdisk -l
Disk /dev/vda: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000b1b45

   Device Boot      Start         End      Blocks   Id  System
/dev/vda1   *        2048    83875364    41936658+  83  Linux
WARNING: fdisk GPT support is currently new, and therefore in an experimental phase. Use at you
r own discretion.

Disk /dev/vdb: 35184.4 GB, 35184372088832 bytes, 68719476736 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: gpt
Disk identifier: BCE92401-F427-45CC-8B0D-B30EDF279C2F

#         Start          End    Size  Type            Name
 1          2048  10307921919    4.8T  Microsoft basic mnt
```

2. Use the **parted** tool to create a new partition and allocate capacity for it.
   i. Run the `parted /dev/vdb` command to start the Parted tool.
   ii. Run the `print free` command to view the disk capacity to be allocated. Record the start and end sectors and capacity of the existing partition.In this example, the start sector number of */dev/vdb1* is 1,049 KB, the end sector number is 5,278 GB, and the capacity is 5,278 GiB.

```
(parted) print free
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 35.2TB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
        17.4kB  1049kB  1031kB  Free Space
 1      1049kB  5278GB  5278GB  ext4         mnt
        5278GB  35.2TB  29.9TB  Free Space
```

iii. Run the `mkpart <the partition name> <the start sector number> <the percentage of allocated capacity>` command.

In this example, the */dev/vdb2* partition named test is created. The start sector number of the new partition is the end sector number of the existing partition. The new capacity is allocated to the new partition.

iv. Run the `print` command to check whether the capacity (Size) of the partition is changed.

```
(parted) mkpart test 5278GB 100%
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 35.2TB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
 1      1049kB  5278GB  5278GB  ext4         mnt
 2      5278GB  35.2TB  29.9TB               test
```

v. Run the `quit` command to exit the **parted** tool.

3. Create a file system for the new partition.
   - To create an ext4 file system, run the `mkfs.ext4 /dev/vdb2` command.
   - To create an ext3 file system, run the `mkfs.ext3 /dev/vdb2` command.
   - To create an XFS file system, run the `mkfs.xfs -f /dev/vdb2` command.
   - To create a btrfs file system, run the `mkfs.btrfs /dev/vdb2` command.

   In this example, an XFS file system is created.

```
[root@ecshost ~]# mkfs -t xfs /dev/vdb2
meta-data=/dev/vdb2              isize=512    agcount=28, agsize=268435455 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=0, sparse=0
data     =                       bsize=4096   blocks=7301444096, imaxpct=5
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal log           bsize=4096   blocks=521728, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

4. Run the `fdisk -l` command to view the change of the partition capacity.

```
[root@ecshost ~]# fdisk -l
Disk /dev/vda: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000b1b45

   Device Boot      Start         End      Blocks   Id  System
/dev/vda1   *        2048    83875364    41936658+  83  Linux
WARNING: fdisk GPT support is currently new, and therefore in an experimental phase. Use at you
r own discretion.

Disk /dev/vdb: 35184.4 GB, 35184372088832 bytes, 68719476736 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: gpt
Disk identifier: BCE92401-F427-45CC-8B0D-B30EDF279C2F

#         Start          End    Size  Type            Name
 1         2048  10307921919    4.8T  Microsoft basic mnt
 2  10307921920  68719474687   27.2T  Microsoft basic test
```

5. Run the **blkid** command to view the file system types.

```
[root@ecshost ~]# blkid
/dev/vda1: UUID="ed95c595-4813-480e-****-85b1347842e8" TYPE="ext4"
/dev/vdb1: UUID="21e91bbc-7bca-4c08-****-88d5b3a2303d" TYPE="ext4" PARTLABEL="mnt" PARTU
UID="576235e0-5e04-4b76-****-741cbc7e98cb"
/dev/vdb2: UUID="a7dcde59-8f0f-4193-****-362a27192fb1" TYPE="xfs" PARTLABEL="test" PARTUUI
D="464a9fa9-3933-4365-****-c42de62d2864"
```

6. Mount the new partition.

```
[root@ecshost ~]# mount /dev/vdb2 /mnt
```

Option 5: Resize the file system of a raw data disk

If a raw data disk contains a file system but no partitions, perform the following operations to directly resize the file system:

1. View the mount path of the data disk and unmount the partition based on the returned file path.

```
[root@ecshost ~]# mount | grep "/dev/vdb"
/dev/vdb on /mnt type ext4 (rw,relatime,data=ordered)
[root@ecshost ~]# umount /dev/vdb
[root@ecshost ~]# mount | grep "/dev/vdb"
```

2. Run different commands based on the file system type.
   - For an ext* file system, run the resize2fs command as the root user to resize the file system. Example:

     ```
     resize2fs /dev/vdb
     ```

   - For an XFS file system, run the xfs_growfs command as the root user to resize the file system.

     > **Note** The new version xfs_growfs identifies the device to be resized based on the mount point. Example: `xfs_growfs /mnt`. You can run the `xfs_growfs --help` command to check how to use xfs_growfs of different versions.

     - xfs_growfs of the new version

       ```
       xfs_growfs /mnt
       ```

     - xfs_growfs that is not updated

       ```
       xfs_growfs /dev/vdb
       ```

3. Mount the disk to the mount point.

   ```
   mount /dev/vdb /mnt
   ```

4. Run the `df -h` command to view the result of data disk resizing.

   The file system has a larger capacity, indicating that the file system is resized.

   ```
   [root@ecshost ~]# df -h
   Filesystem Size Used Avail Use% Mounted on
   /dev/vda1 40G 1.6G 36G 5% /
   devtmpfs 3.9G 0 3.9G 0% /dev
   tmpfs 3.9G 0 3.9G 0% /dev/shm
   tmpfs 3.9G 460K 3.9G 1% /run
   tmpfs 3.9G 0 3.9G 0% /sys/fs/cgroup
   /dev/vdb 98G 37G 61G 37% /mnt
   tmpfs 783M 0 783M 0% /run/user/0
   ```

Related information

- Resize disks online for Windows instances
- Resize partitions and file systems of Linux system disks

8.3. Use LVs for Linux

8.3.1. Create an LV

This topic describes how to use the Logical Volume Manager (LVM) to create a Logical Volume (LV) that consists of multiple cloud disks on a Linux ECS instance.

LVM LV Dynamic management Database Volume

Prerequisites

You have created multiple cloud disks and attached them to the ECS instance. For more information, see Create a disk and Attach a data disk.

Context

The LVM is a disk partition management mechanism for Linux ECS instances. It is featured by dynamic management of hard disks. In this mechanism, a logical layer is created on hard disks and partitions to improve flexible management of hard disks and partitions. You can dynamically adjust the LV size without losing existing data. The existing LV remains unchanged even if you add new data disks.

> **Notice** A snapshot backs up only the data of a single cloud disk. When the LVM is used, data differences may occur when you roll back cloud disks.

Step 1: Create a Physical Volume (PV)

1. Remotely connect to an ECS instance as a root user. For more information, see Overview.
2. Run the **lsblk** command to view information of all cloud disks on an ECS instance.

   If the following information is displayed, you can use the LVM to create an LV that consists of five cloud disks.

   ```
   root@lvs06:~# lsblk
   NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
   vda    252:0040G  0 disk
   └─vda1 252:1040G  0 part /
   vdb    252:1601T  0 disk
   vdc    252:3201T  0 disk
   vdd    252:4801T  0 disk
   vde    252:6401T  0 disk
   vdf    252:8001T  0 disk
   ```

3. Run the **pvcreate** command to create a PV.

   ```
   pvcreate <Data disk name 1> ... <Data disk name N>
   ```

   If you want to add multiple data disks, you can add multiple data disk names separated with spaces.

```
root@lvs06:~# pvcreate /dev/vdb /dev/vdc /dev/vdd /dev/vde /dev/vdf
  Physical volume "/dev/vdb" successfully created.
  Physical volume "/dev/vdc" successfully created.
  Physical volume "/dev/vdd" successfully created.
  Physical volume "/dev/vde" successfully created.
  Physical volume "/dev/vdf" successfully created.
```

4. Run the **lvmdiskscan** command to view information of the PV that has been created on the ECS instance.

```
root@lvs06:~# lvmdiskscan | grep LVM
  /dev/vdb  [       1.00 TiB] LVM physical volume
  /dev/vdc  [       1.00 TiB] LVM physical volume
  /dev/vdd  [       1.00 TiB] LVM physical volume
  /dev/vde  [       1.00 TiB] LVM physical volume
  /dev/vdf  [       1.00 TiB] LVM physical volume
  5 LVM physical volume whole disks
  0 LVM physical volumes
```

Step 2: Create a Volume Group (VG)

1. Run the **vgcreate** command to create a VG.

```
vgcreate <VG name> <PV name 1> ..... <PV name N>
```

If you want to add multiple PVs, you can add multiple PV names separated with spaces. You can customize the VG name, such as lvm_01.

```
root@lvs06:~# vgcreate lvm_01 /dev/vdb /dev/vdc /dev/vdd /dev/vde /dev/vdf
  Volume group "lvm_01" successfully created
```

2. Run the **vgextend** command to add a PV to the VG.

```
vgrecord VG name <PV name 1> ..... <PV name N>
```

If you add a new PV to lvm_01, the command output is as follows:

```
root@lvs06:~# vgextend lvm_01 /dev/vdb /dev/vdc /dev/vdd /dev/vde /dev/vdf
  Volume group "lvm_01" successfully extended
```

3. Run the **vgs** or **vgdisplay** command to view the VG information.

```
root@lvs06:~# vgs
  VG    #PV #LV #SN Attr   VSize  VFree
  lvm_01   600 wz--n- <6.00t <6.00t
```

Step 3: Create an LV

1. Run the **lvcreate** command to create an LV.

```
lvcreate [-L <LV size>][ -n <LV name>] <VG name>
```

If you create a 5 TiB LV, the command output is as follows:

```
root@lvs06:~# lvcreate -L 5T -n lv01 lvm_01
  Logical volume "lv01" created.
```

> **Note**
>
> - LV size: The LV size must be smaller than the free space of the VG. The unit can be MiB, GiB, or TiB.
> - LV name: You can customize the name.
> - VG name: the name of an existing VG.

2. Run the **lvdisplay** command to view the LV details.

```
root@lvs06:~# lvdisplay
  --- Logical volume ---
  LV Path                /dev/lvm_01/lv01
  LV Name                lv01
  VG Name                lvm_01
  LV UUID                svB00x-l6Ke-ES6M-ctsE-9P6d-dVj2-o0h***
  LV Write Access        read/write
  LV Creation host, time lvs06, 2019-06-0615:27:19 +0800
  LV Status              available
  # open                 0
  LV Size                5.00 TiB
  Current LE             1310720
  Segments               6
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:0
```

Step 4: Create a file system and attach it

1. Run the **mkfs** command to create a file system on the LV.

```
mkfs. <File system format> <LV path>
```

If you create an ext4 file system, the command output is as follows:

```
root@lvs06:~# mkfs.ext4 /dev/lvm_01/lv01
mke2fs 1.44.1 (24-Mar-2018)
Creating filesystem with13421772804k blocks and 167772160 inodes
Filesystem UUID: 2529002f-9209-4b6a-9501-106c1145c***
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424, 20480000, 23887872, 71663616, 78675968,
        102400000, 214990848, 512000000, 550731776, 644972544

Allocating group tables: done
Writing inode tables: done
Creating journal (262144 blocks): done
Writing superblocks and filesystem accounting information:
done
```

2. (Optional) Create a mount point such as /media/lv01. You can also attach the cloud disk to an existing directory.

```
mkdir /media/lv01
```

3. Run the **mount** command to attach the file system.

```
root@lvs06:~# mount /dev/lvm_01/lv01 /media/lv01
```

4. Run the **df** command to check whether the LV is attached.

```
root@lvs06:~# df -h
Filesystem           Size  Used Avail Use% Mounted on
udev                 12G    012G   0% /dev
tmpfs                2.4G  3.7M  2.4G   1% /run
/dev/vda1              40G  3.6G   34G  10% /
tmpfs                12G    0   12G   0% /dev/shm
tmpfs                5.0M    05.0M   0% /run/lock
tmpfs                12G    012G   0% /sys/fs/cgroup
tmpfs                2.4G    02.4G   0% /run/user/0
/dev/mapper/lvm_01-lv01  5.0T   89M  4.8T   1% /media/lv01
```

Related information

- Resize an LV

8.3.2. Resize an LV

This topic describes how to use the Logical Volume Manager (LVM) to resize a Logical Volume (LV) that consists of multiple cloud disks on a Linux ECS instance.

LVM LV Dynamic management Database Volume

Prerequisites

You have created an LV. For more information, see Create an LV.

The cloud disks have free space, or the cloud disks have been resized. For more information, see Overview.

Procedure

1. Remotely connect to an ECS instance as a root user. For more information, see Overview.
2. Run the **lvdisplay** command to view information of the LV that has been created in the ECS instance.

   You have created the /dev/lvm_01/lv01 LV, which have a physical capacity of 5 TiB.

```
root@lvs06:~# lvdisplay
  --- Logical volume ---
  LV Path                /dev/lvm_01/lv01
  LV Name                lv01
  VG Name                lvm_01
  LV UUID                svB00x-l6Ke-ES6M-ctsE-9P6d-dVj2-o0h***
  LV Write Access        read/write
  LV Creation host, time lvs06, 2019-06-0615:27:19 +0800
  LV Status              available
  # open                 0
  LV Size                5.00 TiB
  Current LE             1310720
  Segments               6
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:0
```

3. Run the **pvs** command to view the Physical Volume (PV) usage.

   The /dev/vdg data disk has a free space of 500 GiB.

```
root@lvs06:~# pvs
  PV         VG     Fmt  Attr PSize     PFree
  /dev/vdb   lvm_01 lvm2 a--  <1024.00g      0
  /dev/vdc   lvm_01 lvm2 a--  <1024.00g      0
  /dev/vdd   lvm_01 lvm2 a--  <1024.00g      0
  /dev/vde   lvm_01 lvm2 a--  <1024.00g      0
  /dev/vdf   lvm_01 lvm2 a--  <1024.00g      0
  /dev/vdg   lvm_01 lvm2 a--  <1024.00g <523.98g
```

4. Run the **lvextend** command to resize the LV.

```
lvextend [-L +/- <Capacity to be resized>] <LV name>
```

> Note
>
> - Capacity to be resized: the LV can be resize only when the Volume Group (VG) has sufficient free space. In this example, you resize the LV from 5 TiB to 5.5 TiB.
> - LV name: the name of the LV to be resized.

You resize the /dev/lvm_01/lv01 LV by a physical capacity of 500 GiB.

```
root@lvs06:~# lvextend -L +500GB /dev/lvm_01/lv01
Size of logical volume lvm_01/lv01 changed from5.00 TiB (1310720 extents) to <5.49 TiB (1438720 ex
tents).
Logical volume lvm_01/lv01 successfully resized.
```

5. Run the **resize2fs** command to resize the file system of the LV.

```
root@lvs06:~# resize2fs /dev/lvm_01/lv01
resize2fs 1.44.1 (24-Mar-2018)
Filesystem at /dev/lvm_01/lv01 is mounted on /media/lv01; on-line resizing required
old_desc_blocks = 640, new_desc_blocks = 703
The filesystem on /dev/lvm_01/lv01 is now 1473249280 (4k) blocks long.
```

6. Run the **df** command to check the file system result.

   The total capacity of the LV is 5.5 TiB, indicating that the resize operation is successful.

```
root@lvs06:~# df -h
Filesystem          Size  Used Avail Use% Mounted on
udev                12G    012G   0% /dev
tmpfs              2.4G  3.7M  2.4G   1% /run
/dev/vda1            40G  3.6G   34G  10% /
tmpfs               12G    0   12G   0% /dev/shm
tmpfs              5.0M    05.0M   0% /run/lock
tmpfs               12G    012G   0% /sys/fs/cgroup
tmpfs              2.4G    02.4G   0% /run/user/0
/dev/mapper/lvm_01-lv01  5.5T   83M  5.2T   1% /media/lv01
```

Related information

- Create an LV

8.4. Create RAID arrays for Linux

This topic uses an ECS instance running the Ubuntu operating system to show how to use the msadm command of the Linux operating system to create a 100-GiB RAID array for multiple data disks.

Prerequisites

You have created multiple cloud disks and attached them to the ECS instance. We recommend that you create cloud disks of the same capacity and type. For more information about the procedure to create pay-as-you-go cloud disks, see Create a disk and Attach a data disk. For more information about the procedure to create subscription cloud disks, see Create a subscription disk.

Context

Independent Array of Independent Disks (RAID) combines multiple cloud disks into a disk array group. Compared with a cloud disk, a RAID array offers higher capacity, read/write bandwidth, reliability, and availability.

We recommend that you use RAID 0 or RAID 1 mode and partition cloud disks in the same size to improve usage of disk space. We recommend that you do not use RAID 5 or RAID 6 mode, because parity check data in RAID5 or RAID6 mode consumes the IOPS of cloud disks and causes performance deterioration.

The following table lists the advantages, disadvantages and application scenarios of RAID 0 and RAID 1 modes.

Mode	Advantage	Disadvantage	Scenario
RAID 0	Uses striping technology to allocate I/O loads to different cloud disks. High disk space increases throughput directly. The capacity and bandwidth in the array are the sum of the capacity and bandwidth of different disks.	A damaged disk may cause loss of all data because no data redundancy is available.	Has high requirements for I/O performance and is applicable when data is backed up in other methods or no data backup is needed.
RAID 1	Provides higher data redundancy because data is stored in different cloud disks in mirroring mode. The lowest capacity and bandwidth of cloud disks are the capacity and bandwidth of the RAID array.	The write performance is poor because data must be written to multiple cloud disks at the same time.	Focuses more on fault tolerance than I/O performance in key applications.

Procedure

1. Remotely connect to an ECS instance as a root user. For more information, see Overview.
2. Run the **lsblk** command to view information of all cloud disks on the ECS instance.

   ```
   root@lvs06:~# lsblk
   ```

3. Run the **mdadm** command to create the /dev/md0 RAID array.
   - Run the following command to create RAID 0 mode:

     ```
     root@raid06:~# mdadm --create /dev/md0 --level=0 --raid-devices=5 /dev/vd[bcdef]
     ices=5 /dev/vd[bcdef]
     mdadm: Defaulting to version 1.2 metadata
     mdadm: array /dev/md0 started
     ```

`--level=0` indicates RAID 0 mode and */dev/vd[bcdef]* indicates that the RAID array consists of the /dev/vdb, /dev/vdc, /dev/vdd, /dev/vde, and/dev/vdf cloud disks.

- Run the following command to create RAID 1 mode:

```
root@raid06:~# mdadm --create /dev/md0 --level=1 --raid-devices=5 /dev/vd[bcdef]
```

`--level=1` indicates RAID 1 mode.

4. Run the **mdadm** command to view information of the newly created /dev/md0 RAID array.

```
root@raid06:~# mdadm --detail /dev/md0
/dev/md0:
      Version : 1.2
 Creation Time : Sun May 1912:31:532019
    Raid Level : raid0
    Array Size : 104775680 (99.92 GiB 107.29 GB)
  Raid Devices : 5
 Total Devices : 5
   Persistence : Superblock is persistent

   Update Time: Sun May 1912: 31: 532019
         State : clean
 Active Devices : 5
Working Devices : 5
 Failed Devices : 0
  Spare Devices : 0

    Chunk Size : 512K

       Name : raid06:0  (local to host raid06)
       UUID : 59b65ca6:ad8ffc30:ee439c6b:db6ba***
     Events : 0Number   Major   Minor   RaidDevice State
    0253160     active sync   /dev/vdb
    1253321     active sync   /dev/vdc
    2253482     active sync   /dev/vdd
    3253643     active sync   /dev/vde
    4253804     active sync   /dev/vdf
```

5. Run the **mkfs** command to create a file system such as ext4 on the RAID array.

   You can also create a file system of other types.

```
root@raid06:~# mkfs.ext4 /dev/md0
mke2fs 1.42.13 (17-May-2015)
Creating filesystem with261939204k blocks and 6553600 inodes
Filesystem UUID: 4fc55c24-d780-40d5-a077-03b484519***
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

6. Run the following command using root permissions to create a profile containing RAID information and configure the RAID array to be automatically reassembled when the ECS instance starts.

```
root@raid06:~# sudo mdadm --detail --scan | sudo tee -a /etc/mdadm/mdadm.conf
```

7. (Optional) Create a mount point such as /media/raid0. You can also attach the cloud disk to an existing directory.

```
root@raid06:~# mkdir /media/raid0
```

8. Run the **mount** command to attach a file system. For example, you can attach the /dev/md0 file system to the /media/raid0 directory.

```
root@raid06:~# mount /dev/md0 /media/raid0
```

9. Run the **df** command to view the mount point information of the RAID array.

   In the returned information, the file system must be attached to the specified mount point.

```
root@raid06:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            7.9G     07.9G   0% /dev
tmpfs           1.6G  3.5M  1.6G   1% /run
/dev/vda1        40G   23G   15G  61% /
tmpfs           7.9G     0  7.9G   0% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           7.9G     07.9G   0% /sys/fs/cgroup
tmpfs           1.6G     01.6G   0% /run/user/0
/dev/md0         99G   60M   94G   1% /media/raid0
```

What's next

To configure the RAID array to be automatically loaded each time the ECS instance starts, add the following information to the /etc/fstab profile.

1. Add auto-start settings to the/etc/fstab profile.

```
root@raid06:~# echo /dev/md0 /media/rad0 efaults,nofail,nobootwait 02 >> /etc/fstab
```

> **Note** To configure the ECS instance to start when the RAID array is not attached, add nofail settings. Even if an error occurs when you stall a cloud disk, nofail settings allow the ECS instance to start. If you are using the Ubuntu operating system, you also can add nobootwait settings.

2. Run the **mount** command to attach all file systems to the /etc/fstab profile:

```
root@raid06:~# mount -a
```

8.5. Decrease the size of a disk

You can use the Alibaba Cloud Migration tool to decrease the size of a disk. Currently, you cannot decrease the size of system or data disks that are attached to ECS instances.

Prerequisites

Before you decrease the size of a disk, make sure that the following requirements are met:

- The remote data synchronization tool rsync is installed in the instance when the disk is attached to a Linux instance. To install rsync, use one of the following commands based on the operating system:
  - CentOS: Run the `yum install rsync -y` command.
  - Ubuntu: Run the `apt-get install rsync -y` command.
  - Debian: Run the `apt-get install rsync -y` command.
  - Other distributions: For information about the installation, visit the official website.
- An AccessKey pair to be exported into the *user_config.json* configuration file is created in the console. For more information, see Create an AccessKey.

  > **Note** To avoid disclosing the AccessKey pair of your Alibaba Cloud account, we recommend that you create a RAM user and use the credentials of the RAM user to create an AccessKey pair. For more information, see Create a RAM user and Create an AccessKey.
- An Alibaba Cloud account is created. To create an Alibaba Cloud account, go to the account registration page.
- Other prerequisites and requirements are met. For more information, see Migrate your server to Alibaba Cloud by using the Cloud Migration tool.

Context

The Cloud Migration tool is designed to balance the cloud-based and offline workloads of Alibaba Cloud users.

The Cloud Migration tool allows you to create a custom image from an ECS instance. During the process of creating a custom image, you can reset the size of a disk to decrease its size. The target object for which we want to decrease the disk size is an ECS instance. You can use the Cloud Migration tool with the same procedure and limits for disk size decreasing and data migration. Data migration between ECS instances is more efficient and has a lower error rate.

However, this method causes some attributes of the original ECS instance to change, such as the instance ID (`InstanceId`) and public IP address. If your instance is in a VPC, you can retain the public IP address by changing it to an Elastic IP address (EIP). We recommend that you use this method when instances are associated with EIPs, or when you are less dependent on public IP addresses.

> **Note**
>
> The Cloud Migration tool is upgraded to Server Migration Center (SMC). Alibaba Cloud no longer provides version updates and technical support to the Cloud Migration tool. We recommend that you use SMC for a better cloud migration experience. SMC provides services, which include full migration, incremental migration, batch migration, and VPC-connected migration. For more information, see Server Migration Center.

Procedure

1. Connect to the target ECS instance by using the administrator or root account. For more information, see Connect to a Linux instance by using a username and password.
2. Download the ZIP file of Alibaba Cloud Migration Tool.
3. Decompress the ZIP file, access the client file directory of the corresponding operating system and version, and find the *user_config.json* configuration file.
4. Complete the configuration. For more information, see Step 2. Configure the migration source and destination. The following figure shows the configuration file of a Linux instance.

   When you decrease the size of a disk, take note of the following parameters:

   - system_disk_size: Set this parameter to the expected size of the system disk in GB. The value cannot be less than the actual size of the system disk.
   - data_disks: Set this parameter to the expected size of the data disk in GB. The value cannot be less than the actual size of the data disk.

   > **Note**
   >
   > - If a Linux instance is attached with a data disk, the *data_disks* parameter is required even if you do not want to decrease the size of the data disk.
   > - If a Windows instance is attached with a data disk, the *data_disks* parameter is optional if you do not want to decrease the size of the data disk.

5. Run the *go2aliyun_client.exe* program.
   - Windows instance: Right-click *go2aliyun_client.exe*, and then select **Run as administrator**.
   - Linux instance:
     a. Run `chmod +x go2aliyun_client` to grant execute permissions to the client.

b. Run `./ go2aliyun_client` to run the client.

6. Wait for the operation results.
   - If `Goto Aliyun Finished!` is displayed, go to the Images page in the ECS console to view the custom image after the disk size is decreased. If the custom image is generated, you can release the original ECS instance and create an ECS instance from the generated custom image. After the instance is created, the new instance will have a reduced disk size. For more information, see Create an ECS instance by using a custom image.
   - If `Goto Aliyun Not Finished!` is displayed, check the files in the *Logs* folder of the same directory for troubleshooting. For more information, see Troubleshooting.

     After the problems are solved, run the Cloud Migration tool again to resume the process to decrease the disk size.

Related information

- Overview of the Cloud Migration tool
- Migrate a server to Alibaba Cloud by using the Cloud Migration tool

9.Best practices for tag design

This topic introduces the best practices for tag design in ECS. Tags can be used to manage, categorize, and search for resources.

Scenarios

Tags can be used to categorize and manage resources by group. The resources include personnel, finance, and cloud services. Tags are applicable in the following common scenarios:

- Management of application publishing procedures
- Resource tracking and tag-based group search and resource management
- Tag- and group-based automated O&M by using Alibaba Cloud services such as Operation Orchestration Service (OOS), Resource Orchestration Service (ROS), Auto Scaling, and Cloud Assistant
- Tag-based cost management and cost allocation
- Resource- or role-based access control

Principles

You must implement the best tagging practices based on the following principles:

- Mutual exclusivity
- Collective exhaustion
- Limited values
- Considering ramifications of future changes
- Simplified design

Mutual exclusivity

Mutual exclusivity is implemented to ensure that multiple tags cannot be assigned to the same resource attribute. For example, if you use the `key="owner"` tag key to represent the owner attribute, you cannot use other tag keys such as *own*, *belonger*, or *owner* to represent this attribute again.

Collective exhaustion

Collective exhaustion means that when you plan resources, you must plan tags at the same time and prioritize the tag keys. All resources must be bound with the planned tag keys and their corresponding values.

- Each tag key-value pair must be named in a standard format.
- Collective exhaustion is a prerequisite for future tag-based access control, cost tracking, automated O&M, and group search.

Limited values

Limited values are implemented to remove excess tag values and retain only core tag values.

This principle simplifies procedures such as resource management, access control, automated O&M, and cost allocation. You can also use tags and automation tools under this principle to manage resources. ECS allows you to control tags by using API operations in SDKs, which makes it easy to automatically manage, retrieve, and filter resources.

Considering ramifications of future changes

During the planning stage, you must consider the impact of adding or deleting tag values to improve the flexibility of modifying tags.

When you modify tags, the tag-based access control, automated O&M, and related billing reports may be changed. For corporate or personal business, the best practice is to create business-related tag groups to manage resources in technical, business, and security dimensions. When you use automated O&M tools to manage resources and services, you can add automation-specific tags to aid in the automation efforts.

Simplified design

Simplified design allows you to simplify the use of tag keys by creating tag keys that have fixed dimensions during the tag planning stage. This principle can reduce operation errors caused by redundant tag keys.

- You can create business-related tag groups to manage resources in technical, business, and security dimensions.
- When you use automated O&M tools to manage resources and services, you can add automation-specific tags.

Examples of designing tag keys

The following table lists the tag naming examples with common dimensions. We recommend that you use lowercase letters to name tags.

Dimension	Tag key	Tag value
Organization	• company • department • organization • team • group	Organization-specific names
Business	• product • business • module • service	Business-specific names
Role	• role • user	• network administrator • application administrator • system administrator • opsuser • devuser • testuser
Purpose	• purpose • use	Specific purposes

Dimension	Tag key	Tag value

Project	• From project dimensions:
	  ◦ project
	  ◦ risk
	  ◦ schedule
	  ◦ subtask
	  ◦ environment
	• From personnel dimensions:
	  ◦ sponsor
	  ◦ member
	  ◦ decisionmaker or owner
	  ◦ creator
		Project-related values

Business department (to implement cost allocation and business tracking)	• costcenter • businessunit • biz • financecontact	Department-related values
Owner from the financial dimension (to identify the resource owner)	owner	Names or emails
Customer from the financial dimension (to identify the clients that a particular group of resources serves)	Custom values or true values	Customer names
Project from the financial dimension (to determine the resource-supported projects)	project	Project names
Order from the financial dimension	order	Order category IDs

References

- Search for resources by tag

Related API operations

- TagResources
- ListTagResources
- UntagResources

10.Best practices for using custom images

10.1. Overview

Alibaba Cloud offers a variety of methods for you to create custom images. You can import a local custom image or create custom images by using snapshots, ECS instances, or Packer. If you frequently use custom images, we recommend that you update them on a regular basis to ensure data security. For example, you can upgrade the OS patches, update the middleware or certificates, or install the latest third-party software for custom images.

Image creation methods

The following table compares the image creation methods offered by Alibaba Cloud.

Method	Advantage	Disadvantage	Billing
• Create a custom image from a snapshot • Create a custom image from an instance	Easy operations in the ECS console	• The image creation procedure becomes complex as more pre-installed software is added. • Difficult to ensure correct and consistent manual operations. • High maintenance cost.	You may be charged for snapshots. For more information, see Snapshot billing.
Create a custom image by using Packer	Open source tool that can be used by most cloud service providers	• Installation and maintenance required. • Script writing required.	You may be charged for resources such as instances, disks, and snapshots. For more information, see Billing overview.
Create and import a custom image	On-cloud application deployment	Difficult operations. You must know how to use the GUI and configure virtualization platform drivers.	You may be charged for data stored in Object Storage Service (OSS) and snapshots. For more information, see Snapshot billing and Overview in *OSS documentation*.

Image creation procedure

The preceding image creation methods (except importing a local custom image) depend on the status and application data of an ECS instance at a specific point in time. OOS and Packer automate the image creation procedure by automatically creating a temporary instance and releasing it after the creation, making them more suitable for agile development.

The image creation procedure varies depending on the method you choose:

- If you plan to create a custom image by using a snapshot or an ECS instance, you must use an existing snapshot or a running ECS instance in the ECS console. After the image is created, you must release the temporary ECS instance you created for building the custom image.
- If you use Packer to create a custom image, you must write scripts by using a JSON template such as the image builder.
- If you plan to create and import a local custom image, you must first configure a virtual machine (such as VirtualBox VM) by following the custom image importing instructions, and then use OSS to import the image to ECS. For more information, see Instructions for importing images.

10.2. Packer: machine images as code

10.2.1. Benefits of using Packer to create custom images

This topic describes the benefits of using Packer to create custom images. Packer is an easy-to-use tool that automates the creation of images. To use Packer to create a custom image, you only need to specify the basic image information, the software to be installed, and the relevant configurations in a JSON template.

create an image patch upgrade version upgrade application update elasticity compose image

Prerequisites

An Alibaba Cloud account is created. To create an Alibaba Cloud account, go to the account registration page.

Context

Packer is a tool provided by HashiCorp to automate the creation of images. A JSON template is used to ensure that the images created each time are identical. Packer simplifies image testing and updating and minimizes the image O&M and management costs. For more information, visit the official Packer website.

Operation conditions

This topic highlights the benefits of Packer in DevOps scenarios by comparing the procedure to create a custom image from an ECS instance and the procedure to create a custom image by using Packer. The examples provided in this topic are based on the following conditions:

- Region: China (Beijing). For more information, see Regions and zones.
- Operating system: CentOS 7.3 64-bit. The public image centos_7_03_64_20G_alibase_20170818.vhd is used in the examples. You can view the image lists of other operating systems in the ECS console or by calling the DescribeImages operation.
- Custom service: Redis.

- Whether to retain temporary resources: No.

> **Note** Paid resources such as instances, public IP addresses, and snapshots will be created in the following procedures. To avoid additional fees, we recommend that you release these resources after custom images are created.

Create a custom image from an ECS instance

This example shows how to create a custom image from an ECS instance by using the ECS console.

1. Log on to the ECS console.
2. In the left-side navigation pane, choose **Instances & Images > Instances**.
3. In the top navigation bar, select a region.
4. Create an ECS instance. For more information, see Create an instance by using the provided wizard. To minimize charges and simplify the procedure, you can create the instance with the following configurations:
   - Billing Method: Pay-As-You-Go. For more information, see Pay-as-you-go.
   - Instance Type: ecs.t5-lc1m1.small. For more information, see Instance families.
   - Public Image: CentOS 7.3 64-bit.
   - VPC: the default VPC.
   - Security Group: the default security group.
   - Public IP Address: If your instance does not need to access the Internet, you can choose not to assign a public IP address to the instance, and then connect to the instance by using the management terminal VNC. For more information, see Connect to a Linux instance by using VNC.
5. Remotely connect to an ECS instance. For more information, see Overview.
6. Run the `yum install redis.x86_64 -y` command to install Redis.
7. Return to the homepage of the ECS console and select the China (Beijing) region.
8. Create an image from the instance you created. For more information, see Create a custom image from an instance.
9. In the left-side navigation pane, choose **Instances & Images > Images**.
10. On the **Images** page, view the progress of image creation.
11. (Optional)After the image is created, release temporary resources created in this procedure, including the instance. If an Elastic IP address (EIP) is used, you can also release this EIP.

Create a custom image by using Packer

This example shows how to use Packer to create a custom image. Before you perform the operations, make sure you have installed Packer. For more information, visit Install Packer or see Use Packer to create a custom image. To create a custom image by using Packer, follow these steps:

1. Create the *alicloud.json* file that contains the following information:

```
{
 "variables": {
  "access_key": "{{env `ALICLOUD_ACCESS_KEY`}}",
  "secret_key": "{{env `ALICLOUD_SECRET_KEY`}}"
 },
 "builders": [{
  "type":"alicloud-ecs",
  "access_key":"{{user `access_key`}}",
  "secret_key":"{{user `secret_key`}}",
  "region":"cn-beijing",
  "image_name":"packer_basic",
  "source_image":"centos_7_03_64_20G_alibase_20170818.vhd",
  "ssh_username":"root",
  "instance_type":"ecs.t5-lc1m1.small",
  "internet_charge_type":"PayByTraffic",
  "io_optimized":"true"
 }],
 "provisioners": [{
  "type": "shell",
  "inline": [
   "sleep 30",
   "yum install redis.x86_64 -y"
  ]
 }]
}
```

Packer parameters

Parameter	Example	Description
variables{"variable1":"value"}	variables{"access_key":"{{env `ALICLOUD_ACCESS_KEY`}}"}	Defines the variables used in the image builder. For more information, visit Builders. These variables are used to prevent information such as AccessKey pairs from being disclosed. The variables use user-defined values at runtime.
builders{"type":"value"}	builders{"type":"alicloud-ecs"}	The image builder defined by Packer. For more information, visit Builders. Alibaba Cloud supports alicloud-ecs, also known as Alicloud Image Builder, which is used to create custom images in Alibaba Cloud ECS.

Parameter	Example	Description
provisioners{"type":"value"}	provisioners{"type":"shell"}	The image provisioners defined by Packer to specify the operations that need to be performed on the temporary instance. The shell provisioner is used in this example. After Packer connects to the Linux instance, Packer automatically runs shell commands to install services based on the shell provisioner configuration. For example, Packer runs the `yum install redis.x86_64 -y` command to install Redis.

Alibaba Cloud parameters

Parameter	Data type	Example	Description	Importance
access_key	String	LTAInPyXXXXQXXXX	The AccessKey ID of your account. For more information, see Create an AccessKey pair. Note To avoid disclosing the AccessKey pair of your Alibaba Cloud account, we recommend that you create a RAM user and use the credentials of the RAM user to create an AccessKey pair. For more information, see Create a RAM user.	High
secret_key	String	CM1ycKrrCekQ0dhXXXXXXXXXl7yavUT	The AccessKey secret of your account.	High
region	String	cn-beijing	The region in which to create the custom image. For more information, see Regions and zones.	High
image_name	String	packer_basic	The name of the custom image to be created. This name must be globally unique in Alibaba Cloud ECS.	Low
source_image	String	centos_7_03_64_20G_alibase_20170818.vhd	The ID of the Alibaba Cloud public image based on which to create the custom image. The created custom image will contain the same operating system as the public image.	High

Parameter	Data type	Example	Description	Importance
instance_type	String	ecs.t5-lc1m1.small	The instance type of the temporary ECS instance used to create the custom image. For more information, see Instance families.	Low
internet_charge_type	String	PayByTraffic	The billing method for network usage of the temporary instance. Recommended value: PaybyTraffic.	Low
io_optimized	Boolean	true	Specifies whether the temporary instance is I/O optimized. Recommended value: true.	Low

2. Run the following command to create a custom image:

```
packer build alicloud.json
```

> Note It may take a few minutes for the image to be created. After the image is created, it is displayed in the image list of the specified Alibaba Cloud region. You can view the image in the ECS console or by calling the DescribeImages operation.

A log is generated during the image creation process. This log records all the image creation operations, including checking parameters, creating temporary resources, pre-installing software, creating target resources, and releasing temporary resources.

```
alicloud-ecs output will be in this color.

==> alicloud-ecs: Prevalidating image name...
    alicloud-ecs: Found image ID: centos_7_03_64_20G_alibase_20170818.vhd
==> alicloud-ecs: Creating temporary keypair: packer_xxx
==> alicloud-ecs: Creating vpc
==> alicloud-ecs: Creating vswitch...
==> alicloud-ecs: Creating security groups...
==> alicloud-ecs: Creating instance.
==> alicloud-ecs: Allocating eip
==> alicloud-ecs: Allocated eip xxx
    alicloud-ecs: Attach keypair packer_xxx to instance: i-xxx
==> alicloud-ecs: Starting instance: i-xxx
==> alicloud-ecs: Using ssh communicator to connect: ***
==> alicloud-ecs: Waiting for SSH to become available...
==> alicloud-ecs: Connected to SSH!
==> alicloud-ecs: Provisioning with shell script: /var/folders/k_/nv2r4drx3xxxxxxxxxxxndb40000gn
/T/packer-shell260049331
    alicloud-ecs: Loaded plugins: fastestmirror
```

```
  alicloud-ecs: Determining fastest mirrors
  alicloud-ecs: Resolving Dependencies
  alicloud-ecs: --> Running transaction check
  alicloud-ecs: ---> Package redis.x86_64 0:3.2.12-2.el7 will be installed
  alicloud-ecs: --> Processing Dependency: libjemalloc.so.1()(64bit) for package: redis-3.2.12-2.el
7.x86_64
  alicloud-ecs: --> Running transaction check
  alicloud-ecs: ---> Package jemalloc.x86_64 0:3.6.0-1.el7 will be installed
  alicloud-ecs: --> Finished Dependency Resolution
  alicloud-ecs:
  alicloud-ecs: Dependencies Resolved
  alicloud-ecs:
  alicloud-ecs: ================================================================================
===============
  alicloud-ecs:  Package         Arch          Version              Repository     Size
  alicloud-ecs: ================================================================================
===============
  alicloud-ecs: Installing:
  alicloud-ecs:  redis           x86_64        3.2.12-2.el7         epel        544 k
  alicloud-ecs: Installing for dependencies:
  alicloud-ecs:  jemalloc        x86_64        3.6.0-1.el7          epel        105 k
  alicloud-ecs:
  alicloud-ecs: Transaction Summary
  alicloud-ecs: ================================================================================
===============
  alicloud-ecs: Install  1 Package (+1 Dependent package)
  alicloud-ecs:
  alicloud-ecs: Total download size: 648 k
  alicloud-ecs: Installed size: 1.7 M
  alicloud-ecs: Downloading packages:
  alicloud-ecs: --------------------------------------------------------------------------------
  alicloud-ecs: Total                                       2.2 MB/s | 648 kB  00:00
  alicloud-ecs: Running transaction check
  alicloud-ecs: Running transaction test
  alicloud-ecs: Transaction test succeeded
  alicloud-ecs: Running transaction
  alicloud-ecs:   Installing : jemalloc-3.6.0-1.el7.x86_64                       1/2
  alicloud-ecs:   Installing : redis-3.2.12-2.el7.x86_64                         2/2
  alicloud-ecs:   Verifying  : redis-3.2.12-2.el7.x86_64                         1/2
  alicloud-ecs:   Verifying  : jemalloc-3.6.0-1.el7.x86_64                       2/2
```

```
    alicloud-ecs:
    alicloud-ecs: Installed:
    alicloud-ecs:   redis.x86_64 0:3.2.12-2.el7
    alicloud-ecs:
    alicloud-ecs: Dependency Installed:
    alicloud-ecs:   jemalloc.x86_64 0:3.6.0-1.el7
    alicloud-ecs:
    alicloud-ecs: Complete!
==> alicloud-ecs: Stopping instance: i-xxx
==> alicloud-ecs: Waiting instance stopped: i-xxx
==> alicloud-ecs: Creating image: packer_basic
    alicloud-ecs: Detach keypair packer_xxx from instance: i-xxx
==> alicloud-ecs: Cleaning up 'EIP'
==> alicloud-ecs: Cleaning up 'instance'
==> alicloud-ecs: Cleaning up 'security group'
==> alicloud-ecs: Cleaning up 'vSwitch'
==> alicloud-ecs: Cleaning up 'VPC'
==> alicloud-ecs: Deleting temporary keypair...
Build 'alicloud-ecs' finished.

==> Builds finished. The artifacts of successful builds are:
--> alicloud-ecs: Alicloud images were created:

cn-beijing: m-bp67acfmxazb4ph***
```

Related information

References

- DescribeImages
- Alicloud Image Builder
- Examples
- Alicloud Image Builder parameters used to implement DevOps

10.2.2. Alicloud Image Builder parameters used to implement DevOps

This topic describes the Alicloud Image Builder parameters that are used to implement DevOps when you use Packer to create Alibaba Cloud ECS custom images.

Parameter used to tag custom images

- Field: tags{"key":"value"}.
- Applicable scenario: If you have multiple custom images, you can tag them for easy

management and retrieval. Alicloud Image Builder provides the tags parameter. If you set this parameter when you use Packer to create a custom image, the generated image will be bound with the tag you specified. For more information, see Tag overview.

- Function: When you query images in the ECS console or by calling the DescribeImages operation, images are displayed with their tags. You can also filter images by tag. Tags bound to images can be used together with Terraform to standardize enterprise-grade DevOps processes.
- Example: The following configuration file contains the `version=v1.0.0` and `app=web` tags bound to the generated image and relative snapshot:

```
{
 "variables": {
  "access_key": "{{env `ALICLOUD_ACCESS_KEY`}}",
  "secret_key": "{{env `ALICLOUD_SECRET_KEY`}}"
 },
 "builders": [{
  "type":"alicloud-ecs",
  "access_key":"{{user `access_key`}}",
  "secret_key":"{{user `secret_key`}}",
  "region":"cn-beijing",
  "image_name":"packer_basic",
  "source_image":"centos_7_03_64_20G_alibase_20170818.vhd",
  "ssh_username":"root",
  "instance_type":"ecs.t5-lc1m1.small",
  "internet_charge_type":"PayByTraffic",
  "io_optimized":"true",
  "tags": {
   "version": "v1.0.0",
   "app": "web"
  }
 }]
}
```

Parameter used to control whether to create an image only based on a system disk

- Field: image_ignore_data_disks. Data type: boolean.
- Application scenario: By default, Packer creates images directly from ECS instances. If the ECS instances have data disks, the generated images contain data disk snapshots. You can use one of the following methods to create an instance that has data disks:
  - Method 1: Set data disk parameters in image_disk_mappings. For more information, see Alicloud Image Builder in *Packer documentation*.

  - Method 2: Select an instance type that comes with data disks by default, such as ecs.d1ne.2xlarge. The data disks that instance types come with by default are typically local disks. Snapshots cannot be created for local disks. Therefore, you cannot create images directly from instances of such instance types.

- Function: If you want to select an instance type that comes with data disks by default, but you do not want the image created to contain data disk snapshots, you can add `image_ignore_data_disks": "true"` in the configuration file. This ensures that the image is created only based on the system disk.

Parameter used to set the snapshot timeout period

- Field: wait_snapshot_ready_timeout. Data type: integer. Default value: 3600. Unit: seconds.
- Applicable scenario: Images are created based on snapshots. The time it takes to create a snapshot for a disk depends on the disk size. The larger a disk is, the more time it takes.
- Function: If a timeout error occurs when you create a snapshot for a large disk, specify a larger value for `wait_snapshot_ready_timeout` to prolong the snapshot timeout period.

Parameter used to control whether to connect to an instance by using a private IP address

- Field: ssh_private_ip. Data type: boolean.
- Applicable scenario: By default, Packer creates an Elastic IP address (EIP) and associates this EIP with the temporary ECS instance when Packer creates a custom image. Then, Packer uses the public IP address that corresponds to the EIP to connect to the instance and then installs software or runs commands on the instance. If Packer can use private IP addresses to connect to the instance, the public IP address is not needed.
- Function: If you set `ssh_private_ip` to true, Packer does not assign an EIP or a public IP address to the temporary instance but uses private IP addresses to connect to the instance.

Parameter used to control whether to stop an instance

- Field: disable_stop_instance. Data type: boolean.
- Applicable scenario: By default, after Packer runs provisioners, it stops instances and then creates images from the instances. However, in some scenarios (for example, when you run Sysprep in Windows instances), instances must be in the Running state.

  For information about the applicable scenarios of Sysprep, see Modify the SID of a Windows ECS instance to create a domain environment.

- Function: After you set `disable_stop_instance` to true, Packer does not stop instances but assumes that the command provided in the configuration (provisioners) automatically stops the instances.

Parameter used to specify the UserData file path for enabling WinRM

- Field: user_data_file.
- Applicable scenario: For security purposes, the Windows Remote Management (WinRM) function is disabled in Windows images by default. However, Packer must use the WinRM function to connect to a Windows instance and run commands on the instance. You can use the UserData file to enable WinRM when you create a Windows instance.

- Function: You can use the `user_data_file` parameter to specify the path to the UserData file. For example, you can set the value of user_data_file to examples.ps1.
- Example: In the following sample code, the UserData file is located in the relative path *examples/alicloud/basic/winrm_enable_userdata.ps1*.

```
{
 "variables": {
  "access_key": "{{env `ALICLOUD_ACCESS_KEY`}}",
  "secret_key": "{{env `ALICLOUD_SECRET_KEY`}}"
 },
 "builders": [{
  "type":"alicloud-ecs",
  "access_key":"{{user `access_key`}}",
  "secret_key":"{{user `secret_key`}}",
  "region":"cn-beijing",
  "image_name":"packer_test",
  "source_image":"win2008r2_64_ent_sp1_zh-cn_40G_alibase_20181220.vhd",
  "instance_type":"ecs.n1.tiny",
  "io_optimized":"true",
  "internet_charge_type":"PayByTraffic",
  "image_force_delete":"true",
  "communicator": "winrm",
  "winrm_port": 5985,
  "winrm_username": "Administrator",
  "winrm_password": "Test1234",
  "user_data_file": "examples/alicloud/basic/winrm_enable_userdata.ps1"
 }],
 "provisioners": [{
  "type": "powershell",
  "inline": ["dir c:\\"]
 }]
}
```

Note

- In the preceding sample code:
  - `"communicator": "winrm"` indicates that you are connected to the instance through WinRM.
  - `"winrm_port": 5985` indicates that the communication port is port 5985.
  - `"winrm_username": "Administrator"` indicates that you are connected to the instance as an administrator.
  - `"winrm_password": "Test1234"` indicates that Password Test1234 is used.
- `"image_force_delete":"true"` indicates that existing images are deleted if they have the same name as the image to be created.

Parameters used to create an image based on an on-premises ISO file and import the image to Alibaba Cloud ECS

- Fields: builders{"type":"qemu"} and post-processors{"type":"alicloud-import"}.
- Applicable scenario: If an on-premises ISO file runs in a non-QEMU-based virtualization environment, you can use Packer to create an image based on the file and import the image to Alibaba Cloud ECS.
- Example: If the on-premises environment is based on QEMU, you can use Packer to create an image and then import the image to Alibaba Cloud ECS. For more information, see Create and import on-premises images by using Packer, which includes two important steps:
  i. Use an on-premises virtualization environment or a builder, such as QEMU Builder, to create an on-premises image.
  ii. Define Alicloud Import Post-Processor to import the generated on-premises image to Alibaba Cloud ECS.

To import an ISO file to Alibaba Cloud ECS, you must first install a virtualization environment on premises and then create an image based on the file. The image must be in a format supported by Alibaba Cloud, such as the QCOW2, VHD, or RAW format. Then, you can import the image to Alibaba Cloud ECS. For more information, see Instructions for importing images.

References

Alicloud Image Builder and Examples.

10.3. Create and import a custom image

If you cannot find the operating system that you need in the console when you create an instance, you can create a custom image and import it to the console to create an instance. This topic describes how to create a custom image and import it to Alibaba Cloud.

Prerequisites

- VirtualBox is installed. If not, click here to download VirtualBox.
- The network connection is stable.
-

- The ISO or binary file of the operating system is installed. Example: Red Hat Enterprise Linux.
- ossbrowser is installed. For more information, see Quick start.

Procedure

1. Create a new virtual machine (VM) on VirtualBox.
   i. Start VirtualBox and click **New**.
   ii. Enter the name of the new VM and select the corresponding operating system and its version. For example, set Type to **Microsoft Windows** and set Version to **Windows 7 (64-bit)**.
   iii. Configure the memory size.
   iv. Create a virtual hard disk.
   v. Select **VHD (Virtual Hard Disk)** as the file type of the hard disk. Alibaba Cloud supports the RAW, VHD, and qcow2 formats.
   vi. Select **Dynamically allocated** as the storage type of the hard disk.
   vii. Enter the name of the virtual hard disk and click **Create**.
   viii. After the VHD file is created, double-click the new VM to start it.
2. Run the following command to install KVM:

   ```
   yum install qemu-kvm qemu-img libvirt
   ```

3. Run the following command to disable the firewall of the VM:

   ```
   service firewalld stop
   ```

4. Upload the VHD file to Object Storage Service (OSS) and use ossbrowser to upload the file to the region where you want to create an instance. For more information, see Quick start.
5. Import the custom image.
   i. Log on to the ECS console.
   ii. In the left-side navigation pane, choose **Instances & Images > Images**.
   iii. In the top navigation bar, select the region where the VHD file is uploaded and click **Import Image**. For information about how to allow ECS to access OSS resources, see Import custom images.
   iv. Configure the parameters and click **OK**.

   > **Note** You can log on to the OSS console to obtain the OSS object URL. For more information, see Download objects.

   After the custom image is imported, it is displayed in the image list.

11.Monitor

11.1. Use CloudMonitor to monitor ECS instances

Many businesses are moving to cloud computing because it is cost-effective, and saves customers of heavy lifting. This can be greatly attributed to the leverage of monitoring. Monitoring service provides real-time operation data for you to identify risks in advance, avoid potential loss, and troubleshoot as quickly as possible.

This article takes a website for example (the website architecture is shown as follows) to illustrate how to configure CloudMonitor. The example website uses Alibaba Cloud services such as ECS, RDS, OSS, and Server Load Balancer.

Prerequisites

Before you begin, you must complete the following operations:

- Make sure that your ECS monitoring agents are functional to collect metric data. Otherwise, you must install the agent manually. For more information, see How to install CloudMonitor agent.
- Add alarm contacts and contact groups. We recommend that you add at least two contacts to make sure real-time responses to monitoring alarms. For more information about metrics, see Cloud service overview and alarm overview.
- With CloudMonitor Dashboard, you can gain system-wide visibility into resource utilization and operational health. You can select a metrics dimension. You can choose per-instance metrics dimension if you only have several instances.

  Otherwise, you can choose ECS groups dimension or user dimension, and choose the average value.

Setting alarm threshold

We recommend that you set the alarm threshold according to your business status. A much lower threshold may trigger alarm too often and render monitoring meaningless, while a much higher threshold may leave you with no time to respond to a major event.

Set alarm rules

Take CPU utilization as an example. We have to reserve some processing capacity to guarantee the normal function, so you can set the threshold to 70% and to trigger an alarm when the threshold is exceeded by three times in a row, as shown in the following figure.

If you have to set alarm rules for other metrics, click **Add Alarm Rule**.

Set process monitoring

For Web applications, you can add monitoring for process . For more information, see Process monitoring.

Set site monitoring

Site monitoring is at the network access layer to test the availability.

Set RDS monitoring

We recommend that you set the RDS CPU utilization alarm threshold to 70% and to trigger an alarm when the threshold is exceeded by three times in a row. You can set the disk utilization , IOPS utilization, total connections and other metrics as needed.

Set Server Load Balancer monitoring

Before you begin, make sure that you have enabled health check for your Server Load Balancer instance.

You can use Custom monitoring metrics if the metrics you need are not covered.

12.Use RAM roles to access other Alibaba Cloud services

This topic describes how to enable applications on ECS instances to access other Alibaba Cloud services by using STS temporary credentials through RAM roles. The examples show how to enable Python to access Object Storage Service (OSS).

Prerequisites

- An Alibaba Cloud account is created. To create an Alibaba Cloud account, go to the account registration page.
- An instance is created. For more information, see Create an instance by using the provided wizard.

Context

Previously, applications deployed on an ECS instance needed to use AccessKey pairs (AKs) to access other Alibaba Cloud services. An AK allows you to access Alibaba Cloud APIs with full permissions for your account. To facilitate the management of the AK by applications, you must store the AK in the application configuration files or otherwise store the AK in an ECS instance. These operations makes it more complicated to manage the AK and keep it confidential. If you need consistent deployment across regions, the AK is spread along with the images or instances created from the images. In these cases, you must update and redeploy each instance and image individually whenever you make changes to the AK.

You can attach a RAM role to an ECS instance, and use an STS temporary credential to access other cloud services from the applications within the instance. STS temporary credentials are generated and updated automatically. Applications can obtain the STS temporary credentials by using the instance metadata URL. You can use RAM roles and authorization policies to grant ECS instances with different or identical permissions to access other cloud services.

> **Notice** All operations in this topic are performed in OpenAPI Explorer to help you get started with the examples. OpenAPI Explorer obtains a temporary AK of the current account through the information of the logged user, and initiates online resource operations on the current account. You will be charged for creating an instance. Release the instance after the operations are complete.

Procedure

To enable Python to access OSS by using the STS temporary credential through RAM roles, perform the following operations:

1. Step 1. Create a RAM role and configure an authorization policy.
2. Step 2. Create an ECS instance and attach the RAM role to the instance.
3. Step 3. Access the instance metadata URL within the instance to obtain the STS temporary credential.
4. Step 4. Use SDK for Python to access OSS by using the STS temporary credential.

Step 1. Create a RAM role and configure an authorization policy.

Perform the following operations to create a RAM role and configure an authorization policy:

1. Create a RAM role. Call the CreateRole operation to create a RAM role.
   - RoleName: Set the name of the RAM role. Set the parameter based on your needs. *EcsRamRoleTest* is used in this example.
   - AssumeRolePolicyDocument: Specify a policy as follows to set the created role as a service role and authorize an Alibaba Cloud service (ECS in this example) to assume the role.

```
{
  "Statement": [
    {
      "Action": "sts:AssumeRole",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "ecs.aliyuncs.com"
        ]
      }
    }
  ],
  "Version": "1"
}
```

2. Create an authorization policy. Call the CreatePolicy operation to create an authorization policy.
   - PolicyName: the name of the authorization policy. *EcsRamRolePolicyTest* is used in this example.
   - PolicyDocument: the content of the authorization policy. The following content is used in the example, indicating that the role has the OSS read-only permission.

```
{
  "Statement": [
    {
      "Action": [
        "oss:Get*",
        "oss:List*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ],
  "Version": "1"
}
```

3. Grant permissions to the role. Call the AttachPolicyToRole operation to grant permissions to the role.
   - PolicyType: Set the parameter to *Custom*.
   - PolicyName: Set the parameter to the name of the authorization policy created in the second step. *EcsRamRolePolicyTest* is used in this example.
   - RoleName: Set the parameter to the name of the role created in the first step. *EcsRamRoleTest* is used in this example.

Step 2. Create an ECS instance and attach the RAM role to the instance.

You can use one of the following methods to create an ECS instance and attach the RAM role to the instance:

- Attach the RAM role to an existing ECS.

  Call the AttachInstanceRamRole operation to attach a RAM role to an existing VPC-type instance. The parameters are configured as follows:

  - RegionId: the region ID of the ECS instance.
  - RamRoleName: the name of the RAM role. *EcsRamRoleTest* is used in this example.
  - InstanceIds: the IDs of VPC-type instances to which you want to attach the RAM role. *["i-bXXXXXXX"]* is used in this example.

- Create an ECS instance and attach the RAM role to the instance.

  To create an ECS instance and attach the RAM role to the instance, perform the following steps:

  i. Create an instance.

     Call the CreateInstance operation and set parameters based on your actual needs. The following parameters are required:

     - RegionId: the region ID of the instance. *cn-hangzhou* is used in this example.
     - ImageId: the image of the instance. *centos_7_03_64_40G_alibase_20170503.vhd* is used in this example.

- InstanceType: the instance type of the instance. *ecs.g6.large* is used in this example.
- VSwitchId: the ID of the VSwitch in the VPC to which the instance belongs. RAM roles can be attached to only VPC-type ECS instances. Therefore, the VSwitchId parameter is required.
- RamRoleName: the name of the RAM role. *EcsRamRoleTest* is used in this example.

If you want to authorize a RAM user to create an ECS instance with the specified RAM role attached, in addition to the permission to create an ECS instance, the RAM user must have the PassRole permission. Therefore, you must customize an authorization policy as follows and attach it to the RAM user.

- If you want to configure the RAM user to create an ECS instance, [ECS RAM Action] filed can be replaced with `ecs:CreateInstance`. You can also grant more permissions to the RAM user based on your actual needs.
- If you want to grant all ECS permissions to the RAM user, [ECS RAM Action] must be replaced with `ecs:*`.

> **Note** For more information about values of [ECS RAM Action], see Authentication rules.

```
{
  "Statement": [
    {
      "Action": "[ECS RAM Action]",
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": "ram:PassRole",
      "Resource": "*",
      "Effect": "Allow"
    }
  ],
  "Version": "1"
}
```

ii. Configure the password and start the instance.

iii. Configure the instance to access the Internet in the ECS console or by calling an API operation.

Step 3. Access the instance metadata URL within the instance to obtain the STS temporary credential.

To obtain the STS temporary credential of the instance, perform the following steps.

> **Notice** A new STS temporary credential is generated 30 minutes before the current one expires. Both STS credentials can be used during this period of time.

1. Remotely connect to an ECS instance. For more information, see Overview.
2. Access `http://100.100.100.200/latest/meta-data/ram/security-credentials/EcsRamRoleTest` to obtain the STS temporary credential. The last part of the URL is the RAM role name, which must be replaced with the name you specified.

   > **Note** In this example, the `curl` command is used to access the preceding URL. If your instance is a Windows instance, see Metadata.

   The following content shows the sample response:

```
[root@local ~]# curl http://100.100.100.200/latest/meta-data/ram/security-credentials/EcsRamRol
eTest
{
"AccessKeyId" : "STS.J8XXXXXXXXX4",
"AccessKeySecret" : "9PjfXXXXXXXXBf2XAW",
"Expiration" : "2017-06-09T09:17:19Z",
"SecurityToken" : "CAIXXXXXXXXXXXwmBkleCTkyI+",
"LastUpdated" : "2017-06-09T03:17:18Z",
"Code" : "Success"
}
```

Step 4. Use SDK for Python to access OSS by using the STS temporary credential.

In this example, SDK for Python is used to list 10 files in a specified OSS bucket located within the same region as the instance through the STS temporary credential.

Prerequisites:

- The ECS instance is connected.
- The ECS instance is installed with Python. If your instance is a Linux instance, pip is required.
- An OSS bucket is created in the same region as the instance and the name and endpoint of the bucket are obtained. In this example, the bucket name is `ramroletest`, and the endpoint is `oss-cn-hangzhou.aliyuncs.com`.

Perform the following steps to use SDK for Python to access OSS:

1. Run the `pip install oss2` command to install OSS SDK for Python.
2. Run the following commands to test whether SDK for Python can be used to list 10 files in the specified OSS bucket:

```
import oss2
from itertools import islice
auth = oss2.StsAuth(<AccessKeyId>, <AccessKeySecret>, <SecurityToken>)
bucket = oss2.Bucket(auth, <Your endpoint>, <Your bucket name>)
for b in islice(oss2.ObjectIterator(bucket), 10):
  print(b.key)
```

where:

- The three parameters of `oss2.StsAuth` correspond to the AccessKeyId, AccessKeySecret, and SecurityToken values returned in the Step 3. Access the instance metadata URL within the instance to obtain the STS credential section.
- The last two parameters of `oss2.Bucket` correspond to the name and endpoint of the bucket.

The following content shows the sample response:

```
[root@local ~]# python
Python 2.7.5 (default, Nov  6 2016, 00:28:07)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import oss2
>>> from itertools import islice
>>> auth = oss2.StsAuth("STS.J8XXXXXXXXXX4", "9PjfXXXXXXXXXBf2XAW", "CAIXXXXXXXXXXXXwmBkl
eCTkyI+")
>>> bucket = oss2.Bucket(auth, "oss-cn-hangzhou.aliyuncs.com", "ramroletest")
>>> for b in islice(oss2.ObjectIterator(bucket), 10):
...     print(b.key)
...
ramroletest.txt
test.shh
```

13.GPU instances

13.1. Deploy an NGC environment on instances with GPU capabilities

The TensorFlow deep learning framework is used in this topic to describe how to deploy a NVIDIA GPU Cloud (NGC) environment on instances with GPU capabilities.

Prerequisites

Before you build a TensorFlow deep learning framework, you must complete the following preparations:

- An Alibaba Cloud account is created and the real-name verification is completed.
- An NGC account is created from the NGC website.
- The NGC API key is obtained from the NGC website and saved locally. The NGC API key will be verified when you log on to the NGC container environment.

Context

As a deep learning ecosystem from NGC allows developers to access the deep learning software stack free of charge and is fit for creating a deep learning development environment.

At present, NGC has been fully deployed in members of the gn5 instance family. Moreover, Alibaba Cloud Marketplace provides NGC container images optimized for NVIDIA Pascal GPUs. Developers can deploy NGC container images from Alibaba Cloud Marketplace to quickly build container environments and access optimized deep learning frameworks while reducing the time spent on product development and business deployment. Other benefits include pre-installation of the development environment, support for optimized algorithm frameworks, and continuous updates.

The NGC website provides images of different versions of the current mainstream deep learning frameworks such as Caffe, Caffe2, CNTK, MXNet, TensorFlow, Theano, and Torch. You can select the desired image to deploy the environment.

The following instance families can be deployed with an NGC environment:

- gn4, gn5, gn5i, gn6v, gn6i, and gn6e
- ebmgn5i, ebmgn6i, ebmgn6v, and ebmgn6e

The following example shows how to create an instance with GPU capabilities and deploy an NGC environment on the instance. A gn5 instance is used in this example.

Procedure

1. Create a gn5 instance. For more information, see Create an instance by using the provided wizard. When you configure parameters, take note of the following items:
   - **Region**: Only China (Qingdao), China (Beijing), China (Hohhot), China (Hangzhou), China (Shanghai), China (Shenzhen), China (Hongkong), Singapore, Australia (Sydney), US (Silicon Valley), US (Virginia), and Germany (Frankfurt) are available.
   - **Instance Type**: Select a gn5 instance type.
   - **Image**: Click **Marketplace Image**, find **NVIDIA GPU Cloud Virtual Machine Image** in the

dialog box that appears, and then click **Continue**.

- **Public IP Address**: Select **Assign Public IP Address**.

  > **Note** If you do not select Assign Public IP Address, you can bind an Elastic IP address to the instance after creation.

- **Security Group**: Select a security group. Access to TCP port 22 must be allowed in the security group. If your instance needs to support HTTPS or DIGITS 6, access to TCP port 443 (for HTTPS) or TCP port 5000 (for DIGITS 6) must be allowed.

After the ECS instance is created, log on to the ECS console and note down the public IP address of the instance.

2. Connect to the ECS instance. Use one of the following methods based on the logon credential that you selected during instance creation:
   - Connect to an ECS instance by using a password. For more information, see Connect to a Linux instance by using a username and password.
   - Connect to an ECS instance by using an SSH key pair. For more information, see Connect to a Linux instance by using an SSH key pair.
3. Enter the NGC API Key that you obtained from the NGC website, and then press the Enter key to log on to the NGC container environment.
4. Run the `nvidia-smi` command. You can view information about the current GPU, including the GPU model and the driver version, as shown in the following figure.
5. Complete the following steps to build the TensorFlow deep learning framework.
   i. Log on to the NGC website, go to the TensorFlow image page, and then obtain the `docker pull` command.
   ii. Download the TensorFlow image.

   ```
   docker pull nvcr.io/nvidia/tensorflow:18.03-py3
   ```

   iii. View the downloaded image.

   ```
   docker image ls
   ```

   iv. Run the container to deploy the TensorFlow development environment.

   ```
   nvidia-docker run --rm -it nvcr.io/nvidia/tensorflow:18.03-py3
   ```

6. Use one of the following methods to test TensorFlow:
   - Simple test of TensorFlow.

   ```
   $python
   ```

```
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
>>> sess.run(hello)
```

If TensorFlow loads the GPU correctly, the result is as shown in the following figure.

- Download the TensorFlow model and test TensorFlow.

```
git clone https://github.com/tensorflow/models.git
cd models/tutorials/image/alexnet
python alexnet_benchmark.py --batch_size 128 --num_batches 100
```

The running status is as shown in the following figure.

7. Save the changes made to the TensorFlow image. Otherwise, the changes will be lost the next time you log on.

13.2. Use RAPIDS to accelerate machine learning tasks on a GPU-accelerated instance

This topic describes how to use the NGC-based Real-time Acceleration Platform for Integrated Data Science (RAPIDS) libraries that are installed on a GPU-accelerated instance to accelerate tasks for data science and machine learning as well as improve the efficiency of computing resources.

Background information

RAPIDS is an open source suite of data processing and machine learning libraries developed by NVIDIA to enable GPU-acceleration for data science and machine learning. For more information about RAPIDS, visit the RAPIDS website.

NVIDIA GPU Cloud (NGC) is a deep learning ecosystem developed by NVIDIA to provide developers with free access to deep learning and machine learning software stacks to build corresponding development environments. The NGC website provides RAPIDS Docker images, which come pre-installed with development environments.

JupyterLab is an interactive development environment that makes it easy to browse, edit, and run code files on your servers.

Dask is a lightweight big data framework that can improve the efficiency of parallel computing.

This topic provides sample code that is modified based on the NVIDIA RAPIDS Demo code and dataset and demonstrates how to use RAPIDS to accelerate an end-to-end task from the Extract-Transform-Load (ETL) phase to the Machine Learning (ML) Training phase on a GPU-accelerated instance. The RAPIDS cuDF library is used in the ETL phase and the XGBoost model is used in the ML Training phase. The sample code is based on the Dask framework and runs on a single machine.

> **Note** To obtain the official RAPIDS Demo code of NVIDIA, visit Mortgage Demo.

Prerequisites

- An Alibaba Cloud account is created and real-name verification is completed.
- An NGC account is created on the NGC registration page.
- An NGC API key is obtained by performing the following steps:
  i. Log on to the NGC website.
  ii. In the left-side navigation pane, click **CONFIGURATION**. On the Setup page that appears, click **Get API Key**.
  iii. On the API Key page that appears, click **Generate API Key**.
  iv. In the **Generate a New API Key** message that appears, click **Confirm**.

  > **Note** A new NGC API key overwrites the previous API key. Before you obtain a new API key, you must make sure that the previous API key is no longer needed.

  v. Copy the API key and save it to your local storage device.

Procedure

If you do not use a RAPIDS pre-installed image to create a GPU-accelerated instance, perform the following steps to use RAPIDS to accelerate machine learning tasks:

1. Obtain the RAPIDS image download command.
2. Deploy the RAPIDS environment.
3. Run RAPIDS Demo.

Step 1: Obtain the RAPIDS image download command

1. Log on to the NGC website.
2. On the page that appears, click the **MACHINE LEARNING** tab. On the MACHINE LEARNING tab that appears, click the **RAPIDS** image.
3. Obtain the docker pull command.

   The sample code in this topic is based on the RAPIDS v0.8 image. Note that if you use another image, the corresponding commands may differ.

   i. On the RAPIDS page that appears, click the **Tags** tab.
   ii. Copy the tag information. `0.8-cuda10.0-runtime-ubuntu16.04-gcc5-py3.6` is copied in this example.

iii. Return to the top of the page, find the **Pull Command** section, and copy the displayed command. Paste the copied command to the text editor. Then, replace the image version with the tag information obtained in the preceding step and save the TXT file. In this example, `cuda9.2-runtime-ubuntu16.04` is replaced with `0.8-cuda10.0-runtime-ubuntu16.04-gcc5-py3.6`.

The saved docker pull command is used to download the RAPIDS image in Step 2: Deploy the RAPIDS environment.

Step 2: Deploy the RAPIDS environment

1. Create a GPU-accelerated instance.

   For more information about how to create a GPU-accelerated instance, see Create an instance by using the provided wizard.

   - **Instance Type**: RAPIDS applies only to GPU models that use the NVIDIA Pascal or a later architecture. Therefore, you must select an instance type that meets the GPU requirements. The following instance families are available: gn6i, gn6v, gn5, and gn5i. For more information, see Instance families. We recommend that you select an instance family with larger memory, such as gn6i, gn6v, or gn5. The GPU-accelerated instance that has a 16 GB memory is used in this example.
   - **Image**: Select `NVIDIA GPU Cloud Virtual Machine Image` in the Image Marketplace dialog box.
   - **Public IP Address**: Select **Assign Public IP Address** or attach an elastic IP address (EIP) after you create a GPU-accelerated instance. For more information, see Attach an EIP.
   - **Security Group**: Select a security group for which the following ports are enabled:
     - TCP port 22, used for SSH logon
     - TCP port 8888, used to access JupyterLab
     - TCP port 8786 and TCP port 8787, used to access Dask

2. Connect to the GPU-accelerated instance.

   For more information, see Connect to a Linux instance.

3. Enter the NGC API key and press the Enter key to log on to the NGC container.

4. (Optional) Run the **nvidia-smi** command to view GPU information, such as the GPU model and GPU driver version.

   We recommend that you check the GPU information to identify potential issues. For example, if an earlier NGC driver version is used, it may not be supported by the target Docker image.

5. Run the docker pull command obtained in Step 1: Obtain the RAPIDS image download command to download the RAPIDS image.

   ```
   docker pull nvcr.io/nvidia/rapidsai/rapidsai:0.8-cuda10.0-runtime-ubuntu16.04-gcc5-py3.6
   ```

6. (Optional) Check the information of the downloaded image.

   We recommend that you check the Docker image information to make sure that the correct image is downloaded.

```
docker images
```

7. Run the NGC container to deploy the RAPIDS environment.

```
docker run --runtime=nvidia \
    --rm -it \
    -p 8888:8888 \
    -p 8787:8787 \
    -p 8786:8786 \
    nvcr.io/nvidia/rapidsai/rapidsai:0.8-cuda10.0-runtime-ubuntu16.04-gcc5-py3.6
```

Step 3: Run RAPIDS Demo

1. Download the dataset and the Demo file on the GPU-accelerated instance.

```
# Get apt source address and download demos.
source_address=$(curl http://100.100.100.200/latest/meta-data/source-address|head -n 1)
source_address="${source_address}/opsx/ecs/linux/binary/machine_learning/"
cd /rapids
wget $source_address/rapids_notebooks_v0.8.tar.gz
tar -xzvf rapids_notebooks_v0.8.tar.gz
cd /rapids/rapids_notebooks_v0.8/xgboost
wget $source_address/data/mortgage/mortgage_2000_1gb.tgz
```

2. Start JupyterLab on the GPU-accelerated instance by running the following commands.

   We recommend that you run these commands to start JupyterLab directly.

```
# Run the following command to start JupyterLab and set the password.
cd /rapids/rapids_notebooks_v0.8/xgboost
jupyter-lab --allow-root --ip=0.0.0.0 --no-browser --NotebookApp.token='YOUR PASSWORD'
# Exit JupyterLab.
sh ../utils/stop-jupyter.sh
```

   - You can also run the `sh ../utils/start-jupyter.sh` script to start JupyterLab. However, you cannot set the logon password in this case.
   - You can also press `Ctrl+C` twice to exit JupyterLab.

3. Open your browser and enter `http://`*IP address of your GPU-accelerated instance*`:8888` in the address bar to access JupyterLab.

   > **Note** We recommend that you use Google Chrome.

   If you set the logon password when you start JupyterLab, you will be directed to a page to enter your password.

4. Run the NoteBook code.

A mortgage repayment task is used in this example. For more information, see Code running. The NoteBook code includes the following content:

- *xgboost_E2E.ipynb*: an XGBoost Demo file. You can double-click this file to view its details, or click the Execute icon to run one cell at a time.
- *mortgage_2000_1gb.tgz*: a file named mortgage_2000_1gb.tgz, which contains the mortgage repayment training data of year 2000. Files in the perf folder are split into files of 1 GB to maximize the usage of GPU memory.

Code running

In this example, XGBoost is used to demonstrate the end-to-end code running process from data pre-processing to training the XGBoost data model. The process involves the following phases:

- Extract-Transform-Load (ETL): performed on the GPU-accelerated instance in most cases. Data is extracted, transformed, and then loaded onto the data warehouse.
- Data Conversion: performed on the GPU-accelerated instance. Data processed in the ETL phase is converted into the DMatrix format so that it can be used by XGBoost to train the data model.
- ML-Training: performed on the GPU-accelerated instance by default. XGBoost is used to train the gradient boosting decision tree (GBDT).

Perform the following steps to run the NoteBook code:

1. Prepare the dataset.

   In this example, the shell script downloads the mortgage repayment training data of year 2000 by default, which is the *mortgage_2000_1gb.tgz* file.

   If you want to obtain more data for XGBoost model training, you can set the download_url parameter to specify the URL. For more information, visit Mortgage Data.

   The following figure shows an example.

2. Set relevant parameters.

Parameter	Description
start_year	Specifies the start year from which training data is selected. In the ETL phase, data generated between the start_year and end_year range is processed.
end_year	Specifies the end year from which training data is selected. In the ETL phase, data generated between the start_year and end_year range is processed.
train_with_gpu	Specifies whether to use GPU for XGBoost model training. Default value: *True*.
gpu_count	Specifies the number of workers to be started. Default value: *1*. You can set the parameter based on your actual need but the value must be less than the number of GPUs in the GPU-accelerated instance.

Parameter	Description
part_count	Specifies the number of performance files used for data model training. Default value: *2 × gpu_count*. If the value is too large, an insufficient memory error occurs in the Data Conversion phase and the error message is stored in the background of NoteBook.

The following figure shows an example.

3. Start Dask.

   The NoteBook code starts Dask Scheduler, and also starts workers based on the gpu_count value for ETL and data model training. After you start Dask, you can monitor tasks on the Dask Dashboard. For more information about how to start Dask, see Dask Dashboard.

   The following figure shows an example.

4. Start the ETL phase.

   In this phase, tables are joined, grouped, aggregated, and sliced. The data format is DataFrame of the cuDF library, which is similar to Pandas DataFrame.

   The following figure shows an example.

5. Start the Data Conversion phase.

   In this phase, DataFrame-format data is converted into the DMatrix-format data for XGBoost model training. Each worker processes one DMatrix object.

   The following figure shows an example.

6. Start the ML Training phase.

   In this phase, data model training is started by dask-xgboost, which supports collaborative communication among Dask workers. At the bottom layer, dask-xgboost is also called to execute data model training.

   The following figure shows an example.

Dask Dashboard

Dask Dashboard supports task progress tracking, task performance problem identification, and fault debugging.

After Dask is started, you can enter `http://IP address of your GPU-accelerated instance:8787/status` in the address bar of your browser to go to the Dask Dashboard.

Related functions

Operation	Function name
Download a file.	def download_file_from_url(url, filename):
Decompress a file.	def decompress_file(filename, path):

Operation	Function name
Obtain the number of GPUs in the current machine.	def get_gpu_nums():
Manage the GPU memory.	• def initialize_rmm_pool(): • def initialize_rmm_no_pool(): • def run_dask_task(func, **kwargs):
Submit a Dask task.	• def process_quarter_gpu(year=2000, quarter=1, perf_file=""): • def run_gpu_workflow(quarter=1, year=2000, perf_file="", **kwargs):
Use cuDF to load data from a CSV file.	• def gpu_load_performance_csv(performance_path, **kwargs): • def gpu_load_acquisition_csv(acquisition_path, **kwargs): • def gpu_load_names(**kwargs):
Process and extract characteristics of data for training machine learning models.	• def null_workaround(df, **kwargs): • def create_ever_features(gdf, **kwargs): • def join_ever_delinq_features(everdf_tmp, delinq_merge, **kwargs): • def create_joined_df(gdf, everdf, **kwargs): • def create_12_mon_features(joined_df, **kwargs): • def combine_joined_12_mon(joined_df, testdf, **kwargs): • def final_performance_delinquency(gdf, joined_df, **kwargs): • def join_perf_acq_gdfs(perf, acq, **kwargs): • def last_mile_cleaning(df, **kwargs):

14.FaaS instances best practices

14.1. Use RTL Compiler on an f1 instance

This topic describes how to use Register Transfer Level (RTL) Compiler on an f1 instance.

Prerequisites

Before you perform the operations, make sure that the following requirements are met:

- An Alibaba Cloud account is created. To create an Alibaba Cloud account, go to the account registration page.
- An f1 instance is created and can access the Internet.

  > **Note** Only FaaS F1 basic images from Alibaba Cloud Marketplace can be used in f1 instances. For more information, see Create an f1 instance.

- The rule for allowing traffic on SSH port 22 is configured for the security groups where the f1 instance resides.
- The ID of the f1 instance is obtained from the Instances page in the ECS console.
- An OSS bucket for FaaS is created.

  The OSS bucket and the f1 instance must be in the same account and the same region. For more information, see Create buckets.

- To encrypt files, activate Key Management Service (KMS) first.

  For more information, see Activate KMS.

- You must complete the following operations before you can manage FPGA-accelerated instances as a RAM user:
  - Create a RAM user and grant permissions to the RAM user. For more information, see Create a RAM user and Grant permissions to a RAM user.

    The permissions you must grant to the RAM user include AliyunECSReadOnlyAccess, AliyunOSSFullAccess, and AliyunRAMFullAccess.
  - Go to the Cloud Resource Access Authorization page to authorize FaaS to access your resources.
  - Obtain the AccessKey ID and AccessKey secret of the RAM user.

Context

Before you perform the operations, take note of the following items:

- All operations described in this topic must be performed by a single account within the same region.
- We strongly recommend that you manage FPGA-accelerated instances as a RAM user. To avoid unwanted operations, you must only authorize the RAM user to perform required actions. You must create a role for the RAM user and grant temporary permissions to the role to access the specified OSS bucket. Then, you can download the original DCP project from the OSS bucket and manage the FPGA image. If you want to encrypt the IP address, you must authorize the RAM user to use KMS. If you want the RAM user to check permissions of Alibaba Cloud

accounts, you must authorize the RAM user to view the resources of Alibaba Cloud accounts.

Procedure

1. Connect to an f1 instance. For more information, see Connect to a Linux instance by using a username and password.
2. Run the following command to configure the basic environment:

```
source /opt/dcp1_1/script/f1_env_set.sh
```

3. Run the following commands in sequence to compile the project:

```
cd /opt/dcp1_1/hw/samples/dma_afu
```

```
afu_synth_setup --source hw/rtl/filelist.txt build_synth
```

```
cd build_synth/
```

```
run.sh
```

> **Note** The compilation process may take some time.

4. Create an image.
   i. Run the following commands in sequence to configure the `PATH` environment variable and grant execute permissions to the *faascmd* file:

```
export PATH=$PATH:/opt/dcp1_1/script/
```

```
chmod +x /opt/dcp1_1/script/faascmd
```

   ii. Run the following commands in sequence to initialize the **faascmd** configuration:

```
# Replace hereIsYourSecretId with your AccessKey ID. Replace hereIsYourSecretKey with you
r AccessKey secret.
faascmd config --id=hereIsYourSecretId --key=hereIsYourSecretKey
```

```
# Replace hereIsYourBucket with the OSS bucket name in the China (Hangzhou) region.
faascmd auth --bucket=hereIsYourBucket
```

   iii. Run the following command in the */opt/dcp1_1/hw/samples/dma_afu* directory to upload the GBS file:

```
faascmd upload_object --object=dma_afu.gbs --file=dma_afu.gbs
```

iv. Run the following command to create an image:

```
# Replace hereIsYourImageName with your image name.
faascmd create_image --object=dma_afu.gbs --fpgatype=intel --name=hereIsYourImageName
--tags=hereIsYourImageTag --encrypted=false --shell=V1.1
```

5. Download the image.

i. Run the following command to check whether the image is created:

```
faascmd list_images
```

If `"State":"success"` is displayed in the command output, the image is created. Record the value of FpgaImageUUID in the command output for later use.

ii. Run the following command to obtain the FPGA ID:

```
# Replace hereIsYourInstanceId with the ID of your f1 instance.
faascmd list_instances --instanceId=hereIsYourInstanceId
```

The following figure shows the command output. Record the value of FpgaUUID.

iii. Run the following command to download the FPGA image to the f1 instance:

```
# Replace hereIsYourInstanceID with the instance ID that you recorded. Replace hereIsFpgaU
UID with the value of FpgaUUID that you recorded. Replace hereIsImageUUID with the value of
FpgaImageUUID that you recorded.
faascmd download_image --instanceId=hereIsYourInstanceID --fpgauuid=hereIsFpgaUUID --f
pgatype=intel --imageuuid=hereIsImageUUID --imagetype=afu --shell=V1.1
```

iv. Run the following command to check whether the image is downloaded:

```
# Replace hereIsYourInstanceID with the instance ID that you recorded. Replace hereIsFpgaU
UID with the value of FpgaUUID that you recorded.
faascmd fpga_status --instanceId=hereIsYourInstanceID --fpgauuid=hereIsFpgaUUID
```

If `"TaskStatus":"operating"` is displayed in the command output, and the value of FpgaImageUUID is the same as that of FpgaImageUUID, the image is downloaded.

6. (Optional)Run the following command to enable HugePages if HugePages is disabled:

```
sudo bash -c "echo 20 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages"
```

7. Run the following commands in sequence to perform a test:

```
cd /opt/dcp1_1/hw/samples/dma_afu/sw
```

```
make
```

```
sudo LD_LIBRARY_PATH=/opt/dcp1_1/hw/samples/dma_afu/sw:$LD_LIBRARY_PATH ./fpga_dma_test 0
```

If the following command output is displayed, the test is completed.

14.2. Use OpenCL on an f1 instance

This topic describes how to use Open Computing Language (OpenCL) on an f1 instance to create an image and download the image to a Field Programmable Gate Array (FPGA).

Prerequisites

- An Alibaba Cloud account is created. To create an Alibaba Cloud account, go to the account registration page.
- An f1 instance is created and can access the Internet.

  Note Only FaaS F1 basic images from Alibaba Cloud Marketplace can be used in f1 instances. For more information, see Create an f1 instance.
- The rule for allowing traffic on SSH port 22 is configured for the security groups where the f1 instance resides.
- The ID of the f1 instance is obtained from the Instances page in the ECS console.
- An OSS bucket for FaaS is created.

  The OSS bucket and the f1 instance must be in the same account and the same region. For more information, see Create buckets.
- To encrypt files, activate Key Management Service (KMS) first.

  For more information, see Activate KMS.
- You must complete the following operations before you can manage FPGA-accelerated instances as a RAM user:
  - Create a RAM user and grant permissions to the RAM user. For more information, see Create a RAM user and Grant permissions to a RAM user.

    The permissions you must grant to the RAM user include AliyunECSReadOnlyAccess, AliyunOSSFullAccess, and AliyunRAMFullAccess.
  - Go to the Cloud Resource Access Authorization page to authorize FaaS to access your resources.
  - Obtain the AccessKey ID and AccessKey secret of the RAM user.

Context

Before you perform the operations, take note of the following items:

- All operations described in this topic must be performed by a single account within the same region.
- We strongly recommend that you manage FPGA-accelerated instances as a RAM user. To avoid

unwanted operations, you must only authorize the RAM user to perform required actions. You must create a role for the RAM user and grant temporary permissions to the role to access the specified OSS bucket. Then, you can download the original DCP project from the OSS bucket and manage the FPGA image. If you want to encrypt the IP address, you must authorize the RAM user to use KMS. If you want the RAM user to check permissions of Alibaba Cloud accounts, you must authorize the RAM user to view the resources of Alibaba Cloud accounts.

Procedure

Perform the following operations to use the OpenCL example on an f1 instance to create an image and download the image to an FPGA:

1. Step 1. Connect to an f1 instance
2. Step 2. Install the basic environment
3. Step 3. Download the OpenCL example
4. Step 4. Upload the configuration file
5. Step 5. Download the image to the f1 instance
6. Step 6. Download the FPGA image to an FPGA

Step 1. Connect to an f1 instance

For more information, see Connect to a Linux instance by using a username and password.

Step 2. Install the basic environment

Run the following script to install the basic environment:

```
source /opt/dcp1_1/script/f1_env_set.sh
```

Step 3. Download the OpenCL example

1. Run the following commands in sequence to create and switch to the */opt/tmp* directory:

   ```
   mkdir -p /opt/tmp
   ```

   ```
   cd /opt/tmp
   ```

   You are now in the */opt/tmp* directory.

   Enter the tmp directory

2. Run the following commands in sequence to download and decompress the example file:

   ```
   wget https://www.altera.com/content/dam/altera-www/global/en_US/others/support/examples/download/exm_opencl_matrix_mult_x64_linux.tgz
   ```

   ```
   tar -zxvf exm_opencl_matrix_mult_x64_linux.tgz
   ```

   The following figure shows the directory after decompression.

   Decompress the example file

3. Run the following commands in sequence to access the *matrix_mult* directory for

compilation:

```
cd matrix_mult
```

```
aoc -v -g --report ./device/matrix_mult.cl
```

The compilation process may take a few hours. You can start another session, and run the top command to monitor system resource usage and view the compilation status.

Step 4. Upload the configuration file

1. Run the following commands in sequence to initialize the faascmd tool:

```
#Configure the environment variable.
export PATH=$PATH:/opt/dcp1_1/script/
```

```
#Grant execute permissions to the faascmd tool.
chmod +x /opt/dcp1_1/script/faascmd
```

```
#Replace <hereIsYourSecretId> in the command with your AccessKey ID. Replace <hereIsYourSecr
etKey> with your AccessKey secret.
faascmd config --id=<hereIsYourSecretId> --key=<hereIsYourSecretKey>
```

```
#Replace <hereIsYourBucket> in the command with your bucket name.
faascmd auth --bucket=<hereIsYourBucket>
```

2. Run the following commands in sequence to access the *matrix_mult/output_files* directory and upload the configuration file:

```
cd matrix_mult/output_files
```

You are now in the /opt/tmp/matrix_mult/matrix_mult/output_files directory.

```
faascmd upload_object --object=afu_fit.gbs --file=afu_fit.gbs
```

3. Run the following command to use the GBS file to create an FPGA image:

```
#Replace <hereIsYourImageName> in the command with your image name. Replace <hereIsYourIm
ageTag> with your image tag.
faascmd create_image --object=dma_afu.gbs --fpgatype=intel --name=<hereIsYourImageName> -
-tags=<hereIsYourImageTag> --encrypted=false --shell=V1.1
```

4. Run the following command to check whether the image is created:

```
faascmd list_images
```

If `"State":"success"` is displayed in the command output, the image is created. Record the value of FpgaImageUUID in the command output to be used in a later step.

Step 5. Download the image to the f1 instance

1. Run the following command to obtain the ID of your FPGA instance:

```
#Replace <hereIsYourInstanceId> in the command with the ID of your FPGA instance.
faascmd list_instances --instanceId=<hereIsYourInstanceId>
```

   The following figure shows the command output. Record the value of FpgaUUID.

2. Run the following command to download the image to the f1 instance.

```
#Replace <hereIsYourInstanceID> in the command with the instance ID that you recorded. Replace <hereIsFpgaUUID> with the value of FpgaUUID that you recorded. Replace <hereIsImageUUID> with the value of FpgaImageUUID that you recorded.
faascmd download_image  --instanceId=<hereIsYourInstanceID> --fpgauuid=<hereIsFpgaUUID> --fpgatype=intel --imageuuid=<hereIsImageUUID> --imagetype=afu --shell=V0.11
```

3. Run the following command to check whether the image is downloaded:

```
# Replace <hereIsYourInstanceID> in the command with the instance ID that you recorded. Replace <hereIsFpgaUUID> with the value of FpgaUUID that you recorded.
faascmd fpga_status --fpgauuid=<hereIsFpgaUUID> --instanceId=<hereIsYourInstanceID>
```

   If `"TaskStatus":"operating"` is displayed in the command output, the image is downloaded.

Step 6. Download the FPGA image to an FPGA

1. Open the environment window in Step 2. If the window is closed, perform the operations in Step 2 again.
2. Run the following command to configure the runtime environment for OpenCL:

```
sh /opt/dcp1_1/opencl/opencl_bsp/linux64/libexec/setup_permissions.sh
```

3. Run the following command to go back to the parent directory:

```
cd .. /..
```

   You are now in the /opt/tmp/matrix_mult directory.

4. Run the following command for compilation:

```
make
# Show the environment configuration.
export CL_CONTEXT_COMPILER_MODE_ALTERA=3
cp matrix_mult.aocx ./bin/matrix_mult.aocx
cd bin
host matrix_mult.aocx
```

If the following output is displayed, the configuration is completed. Note that the last line must be `Verification: PASS`.

```
[root@iZbpXXXXXZ bin]# ./host matrix_mult.aocx
Matrix sizes:
 A: 2048 x 1024
 B: 1024 x 1024
 C: 2048 x 1024
Initializing OpenCL
Platform: Intel(R) FPGA SDK for OpenCL(TM)
Using 1 device(s)
 skx_fpga_dcp_ddr : SKX DCP FPGA OpenCL BSP (acl0)
Using AOCX: matrix_mult.aocx
Generating input matrices
Launching for device 0 (global size: 1024, 2048)
Time: 40.415 ms
Kernel time (device 0): 40.355 ms
Throughput: 106.27 GFLOPS
Computing reference output
Verifying
Verification: PASS
```

14.3. Use OpenCL on an f3 instance

This topic describes how to use Open Computing Language (OpenCL) on an f3 instance to create an image and download the image to a Field Programmable Gate Array (FPGA).

Prerequisites

- An Alibaba Cloud account is created. To create an Alibaba Cloud account, go to the account registration page.
- An f3 instance is created and assigned a public IP address.

  For more information about how to create an f3 instance, see Create an f3 instance.

  > **Note** Only images shared by Alibaba Cloud can be used on f3 instances.

- A rule is added to the security group to which the f3 instance belongs to allow access over SSH port 22.
- The ID of the f3 instance is obtained on the Instances page of the ECS console.
- An Object Storage Service (OSS) bucket dedicated to FPGA as a Service (FaaS) is created.

  The bucket and the f3 instance belong to the same account and reside in the same region. For more information, see Create buckets.

- You must complete the following operations before you can manage FPGA-accelerated instances as a RAM user:

- Create a RAM user and grant permissions to the RAM user. For more information, see Create a RAM user and Grant permissions to a RAM user.

  The permissions you must grant to the RAM user include AliyunECSReadOnlyAccess, AliyunOSSFullAccess, and AliyunRAMFullAccess.
- Go to the Cloud Resource Access Authorization page to authorize FaaS to access your resources.
- Obtain the AccessKey ID and AccessKey secret of the RAM user.

Context

Before you perform the operations, take note of the following items:

- All operations described in this topic must be performed by one account in the same region.
- We recommend that you manage an f3 instance as a RAM user. You must create a role for the RAM user and grant the role temporary permissions to access the specified OSS bucket.
- The operations and commands described in this topic are based on the SDAccel development environment 2018.2. If you use the SDAccel development environment of other versions, the operations and commands may vary.

Procedure

Perform the following operations to use OpenCL to create an image and download the image to an FPGA:

1. Step 1. Set up the environment
2. Step 2. Compile a binary file
3. Step 3. Check the packaging script
4. Step 4. Create an image
5. Step 5. Download the image
6. Step 6. Run the host program

Step 1. Set up the environment

1. Connect to an f3 instance.

   > **Note** The subsequent compilation process may take a few hours. We recommend that you log on using Screen or nohup to avoid forced logout due to an SSH timeout.

2. Run the following command to install Screen:

   ```
   yum install screen -y
   ```

3. Run the following command to start Screen:

   ```
   screen -S f3opencl
   ```

4. Run the following command to set up the environment:

   ```
   source /root/xbinst_oem/F3_env_setup.sh xocl  #Run the command each time you open a new ter
   minal window.
   ```

> **Note**
>
> - The process to set up the environment involves installing the xocl driver, setting the vivado environment variable, checking the vivado license, detecting the aliyun-f3 sdaccel platform, configuring 2018.2 runtime, and checking the faascmd version.
> - If you want to run an emulation of SDAccel, do not run the preceding commands to set up the environment. You only need to configure the environment variable for vivado separately.
> - We recommend that you use Makefile for emulation.

Step 2. Compile a binary file

Perform the following operations to compile the vadd and kernel_global_bandwidth binary files:

- Example 1: vadd
  i. Copy the *example* directory.

  ```
  cp -rf /opt/Xilinx/SDx/2018.2/examples . /
  ```

  ii. Go to the *vadd* directory.

  ```
  cd examples/vadd/
  ```

  iii. Run the **cat sdaccel.mk | grep "XDEVICE="** command to check the value of XDEVICE. Make sure that the value is `xilinx_aliyun-f3_dynamic_5_0` .

  iv. Perform the following operations to modify the *common.mk* file:
    a. Run the **vim ../common/common.mk** command to open the file.
    b. At the end of code line 61, add the compilation parameter --xp param:compiler.acceleratorBinaryContent=dcp. The parameter may be added in code line 60, 61, or 62. The modified code is as follows:

    ```
    CLCC_OPT += $(CLCC_OPT_LEVEL) ${DEVICE_REPO_OPT} --platform ${XDEVICE} ${KERNEL_DEFS} ${KERNEL_INCS} --xp param:compiler.acceleratorBinaryContent=dcp
    ```

    > **Note** You must add the compilation parameter --xp param:compiler.acceleratorBinaryContent=dcp so that Xilinx® OpenCL™ Compiler (xocc) generates a DCP file after the placement and routing is completed. The file cannot be a bit file. Then, you can submit the DCP file to the compilation server.

  v. Run the following command to compile the program:

  ```
  make -f sdaccel.mk xbin_hw
  ```

  If the following similar information is displayed, the compilation of the binary file is started. The compilation process may take a few hours.

- Example 2: kernel_global_bandwidth

i. Run the following commands in sequence to clone `xilinx 2018.2 example`:

```
git clone https://github.com/Xilinx/SDAccel_Examples.git
```

```
cd SDAccel_Examples/
```

```
git checkout 2018.2
```

> **Note** The Git branch version must be 2018.2.

ii. Run the **cd getting_started/kernel_to_gmem/kernel_global_bandwidth/** command to access the directory.

iii. Perform the following operations to modify the *Makefile* file:

a. Run the **vim Makefile** command to open the file.

b. Set `DEVICES` to xilinx_aliyun-f3_dynamic_5_0.

c. In code line 33, add the compilation parameter --xp param:compiler.acceleratorBinaryContent=dcp. The modified code is as follows:

```
CLFLAGS +=--xp "param:compiler.acceleratorBinaryContent=dcp" --xp "param:compiler.pres
erveHlsOutput=1" --xp "param:compiler.generateExtraRunData=true" --max_memory_ports
bandwidth -DNDDR_BANKS=$(ddr_banks)
```

iv. Run the following command to compile the program:

```
make TARGET=hw
```

If the following similar information is displayed, the compilation of the binary file is started. The compilation process may take a few hours.

Step 3. Check the packaging script

Run the following command to check whether the packaging script exists:

```
file /root/xbinst_oem/sdaccel_package.sh
```

If `cannot open (No such file or directory)` is displayed in the command output, the script does not exist. You must run the following command to manually download the packaging script:

```
wget http://fpga-tools.oss-cn-shanghai.aliyuncs.com/sdaccel_package.sh
```

Step 4. Create an image

1. Initialize the faascmd tool and set up the OSS environment.

i. Run the following command to configure the AccessKey pair of the RAM user:

```
#Replace <hereIsYourSecretId> with the AccessKey ID, and <hereIsYourSecretKey> with the AccessKey secret of the RAM user.
faascmd config --id=<hereIsYourSecretId> --key=<hereIsYourSecretKey>
```

ii. Run the following command to configure the OSS bucket dedicated to FaaS:

```
#Replace <hereIsYourBucket> with the name of the OSS bucket that you created.
faascmd auth --bucket=<hereIsYourBucket>
```

2. Run the ls command to obtain the file suffixed by `.xclbin` .

3. Run the following command to package the binary file:

```
/root/xbinst_oem/sdaccel_package.sh -xclbin=/opt/Xilinx/SDx/2018.2/examples/vadd/bin_vadd_hw.xclbin
```

After the binary file is packaged, you can find the packaged file in the same directory, as shown in the following figure.

Step 5. Download the image

You can use a scripted or step-by-step process to upload the netlist files and download the FPGA image.

- Scripted process: This process applies only to f3 instances with a single FPGA.

i. Run the following command to upload and generate the image:

```
sh /root/xbinst_oem/tool/faas_upload_and_create_image.sh <bit.tar.gz - name of the package to be uploaded>
```

ii. Run the following command to download the image:

```
sh /root/xbinst_oem/tool/faas_download_image.sh <bit.tar.gz - package name> <0/1>  #The last number <0/1> stands for the FPGA serial number in the instance.
```

> **Note** 0 indicates the first FPGA of the f3 instance. For single-FPGA instances, the FPGA serial number is always 0. For instances with multiple FPGAs, such as an instance with four FPGAs, the serial numbers are 0, 1, 2, and 3.

If you want to download the same image to multiple FPGAs, add the serial number to the end of each command line. For example, run the following commands to download the same image to four FPGAs:

```
sh /root/xbinst_oem/tool/faas_download_image.sh <bit.tar.gz - package name> 0
```

```
sh /root/xbinst_oem/tool/faas_download_image.sh <bit.tar.gz - package name> 1
```

```
sh /root/xbinst_oem/tool/faas_download_image.sh <bit.tar.gz - package name> 2
```

```
sh /root/xbinst_oem/tool/faas_download_image.sh <bit.tar.gz - package name> 3
```

- Step-by-step process: Use the faascmd tool to perform operations. For more information, see Use faascmd.
  i. Run the following commands in sequence to upload the package to your OSS bucket, and then upload the GBS file from your OSS bucket to the OSS bucket in the FaaS administrative unit:

```
faascmd upload_object --object=bit.tar.gz --file=bit.tar.gz
```

```
faascmd create_image --object=bit.tar.gz --fpgatype=xilinx --name=<hereIsFPGAImageName> --tags=<hereIsFPGAImageTag> --encrypted=false --shell=<hereIsShellVersionOfFPGA>
```

  ii. Run the following command to check whether the FPGA image is downloadable:

```
faascmd list_images
```

  The command output is as follows:

  - If `"State" : "compiling"` is displayed, the FPGA image is being compiled.
  - If `"State" : "success"` is displayed, the FPGA image is downloaded. You must find and record the value of FpgaImageUUID.

  iii. Run the following command. Find and record the value of FpgaUUID in the command output.

```
faascmd list_instances --instanceId=<hereIsYourInstanceId> #Replace <hereIsYourInstanceId> with the ID of the f3 instance.
```

  iv. Run the following command to download the FPGA image:

```
#Replace <hereIsYourInstanceId> with the ID the f3 instance. Replace <hereIsFpgaUUID> with the value of FpgaUUID that you recorded. Replace <hereIsImageUUID> with the value of FpgaImageUUID that you recorded.
faascmd download_image --instanceId=<hereIsYourInstanceId> --fpgauuid=<hereIsFpgaUUID> --fpgatype=xilinx --imageuuid=<hereIsImageUUID> --imagetype=afu --shell=<hereIsShellVersionOfFpga>
```

  v. Run the following command to check whether the image is downloaded:

```
faascmd fpga_status --fpgauuid=<hereIsFpgaUUID> --instanceId=<hereIsYourInstanceId> #Replace <hereIsFpgaUUID> with the value of FpgaUUID that you recorded earlier. Replace <hereIsYourInstanceId> with the ID of the f3 instance.
```

The following figure shows the command output. If the value of FpgaImageUUID in the command output is the same as that of FpgaImageUUID that you recorded, and `"TaskStatus":"valid"` is displayed, the image is downloaded.

Step 6. Run the host program

1. Run the following command to set up the environment:

   ```
   source /root/xbinst_oem/F3_env_setup.sh xocl #Run the command each time you open a new terminal window.
   ```

2. Configure the *sdaccel.ini* file.

   In the directory where the Host binary file is located, run the **vim sdaccel.ini** command to create the *sdaccel.ini* file and enter the following content:

   ```
   [Debug]
   profile=true
   [Runtime]
   runtime_log = "run.log"
   hal_log = hal.log
   ert=false
   kds=false
   ```

3. Run the host program.
   - For vadd, run the following commands:

     ```
     make -f sdaccel.mk host
     ```

     ```
     ./vadd bin_vadd_hw.xclbin
     ```

   - For kernel_global_bandwidth, run the following command:

     ```
     ./kernel_global
     ```

   If `Test Passed` is displayed in the command output, the test is passed.

Other common commands

The following table describes some common commands for f3 instances.

Task	Command
View the help documentation	`make -f ./sdaccel.mk help`

Task	Command
Run software emulation	`make -f ./sdaccel.mk run_cpu_em`
Run hardware emulation	`make -f ./sdaccel.mk run_hw_em`
Compile only the host code	`make -f ./sdaccel.mk host`
Compile and generate downloadable files	`make -f sdaccel.mk xbin_hw`
Clean a work directory	`make -f sdaccel.mk clean`
Forcibly clean a work directory	`make -f sdaccel.mk cleanall`

> **Note**
> - During emulation, follow the Xilinx emulation process. You do not need to set up the F3_env_setup environment.
> - The SDAccel runtime and SDAccel development platform are available in the official f3 images provided by Alibaba Cloud. You can also click SDAccel runtime and SDAccel development platform to download them.

14.4. Use the RTL design on an f3 instance

This topic describes how to implement the Register Transfer Level (RTL) design on an f3 instance.

Prerequisites

- An Alibaba Cloud account is created. To create an Alibaba Cloud account, go to the account registration page.
- An f3 instance is created and can access the Internet. For more information, see Create an f3 instance.
- A rule is added to the security group to which the f3 instance belongs to allow access over SSH port 22. For more information, see Add security group rules.
- The ID of the f3 instance is obtained on the Instances page of the ECS console.
- An Object Storage Service (OSS) bucket dedicated to FPGA as a Service (FaaS) is created in the China (Shanghai) region. For more information, see Create buckets.

  > **Note** The bucket grants read and write permissions to the FaaS administrator account. We recommend that you do not store information that is not related to FaaS.

- You must complete the following operations before you can manage FPGA-accelerated instances as a RAM user:

- Create a RAM user and grant permissions to the RAM user. For more information, see Create a RAM user and Grant permissions to a RAM user.

  The permissions you must grant to the RAM user include AliyunECSReadOnlyAccess, AliyunOSSFullAccess, and AliyunRAMFullAccess.
- Go to the Cloud Resource Access Authorization page to authorize FaaS to access your resources.
- Obtain the AccessKey ID and AccessKey secret of the RAM user.

Context

Before you perform the operations, take note of the following items:

- All operations described in this topic must be performed by a single account within the same region.
- We recommend that you perform operations on an f3 instance as a RAM user. We recommend that you grant the RAM user necessary permissions based on the principle of least privilege, such as permissions to access DCP/xclbin files in OSS buckets, upload the Vivado compilation log, and manage specified ECS instances. You must also specify the RAM role AliyunFAASDefaultRole. By default, FaaS uses the role to access resources hosted in other cloud services. The AliyunFAASRolePolicy policy of the role includes the permissions on Key Management Service (KMS) through which you can encrypt IP addresses.

Procedure

1. Connect to an f3 instance.

   > **Note** The compilation process may take two to three hours. We recommend that you use nohup or Virtual Network Computing (VNC) to connect to the instance to avoid unexpected disconnection.

2. Download and decompress the RTL design.
3. Set up the environment.
   - If the driver is `xdma`, run the following command to set up the environment:

     ```
     source /root/xbinst_oem/F3_env_setup.sh xdma #Run the command each time you open a new terminal window.
     ```

   - If the driver is `xocl`, run the following command to set up the environment:

     ```
     source /root/xbinst_oem/F3_env_setup.sh xocl #Run the command each time you open a new terminal window.
     ```

     > **Note** The process to set up the environment involves installing the xdma or xocl driver, setting the vivado environment variable, checking the vivado license, detecting the aliyun-f3 sdaccel platform, configuring 2018.2 runtime, and checking the faascmd version.

4. Specify an OSS bucket.

```
faascmd config --id=<hereIsYourSecretId> --key=<hereIsYourSecretKey> #Replace <hereIsYourSecretId> with the AccessKey ID, and <hereIsYourSecretKey> with the AccessKey secret of the RAM user.
faascmd auth --bucket=hereIsYourBucket # Replace <hereIsYourBucket> with the name of the OSS bucket that you created.
```

5. Run the following command to compile the RTL project:

```
cd <decompressed directory>/hw/ # Access your decompressed hw directory.
sh compiling.sh
```

> **Note** The compilation process takes about two to three hours.

6. Upload the netlist files and download the FPGA image. You can use a scripted or step-by-step process to complete this step.
   - Scripted process: This process applies only to f3 instances with a single FPGA.
     a. Run the following command to upload the package and generate the image file:

```
sh /root/xbinst_oem/tool/faas_upload_and_create_image.sh <package to be uploaded>
```

     b. Download the image file.

```
sh /root/xbinst_oem/tool/faas_download_image.sh <package name> <0/1> # The last number <0/1> stands for the FPGA serial number in the instance.
```

     0 indicates the first FPGA of the f3 instance. For single-FPGA instances, the FPGA serial number is always 0. For instances with multiple FPGAs, such as an instance with four FPGAs, the serial numbers are 0, 1, 2 and 3.

     If you want to download the same image to multiple FPGAs, add the serial number to the end of each command line. For example, run the following commands to download the same image to four FPGAs:

```
sh /root/xbinst_oem/tool/faas_download_image.sh <package name> 0
sh /root/xbinst_oem/tool/faas_download_image.sh <package name> 1
sh /root/xbinst_oem/tool/faas_download_image.sh <package name> 2
sh /root/xbinst_oem/tool/faas_download_image.sh <package name> 3
```

   - Step-by-step process: Use the faascmd tool to perform operations. For more information, see Use faascmd.

a. Run the following commands in sequence to upload the package to your OSS bucket, and then upload the GBS file from your OSS bucket to the OSS bucket in the FaaS administrative unit:

```
faascmd upload_object --object=bit.tar.gz --file=bit.tar.gz
faascmd create_image --object=bit.tar.gz --fpgatype=xilinx --name=<hereIsFPGAImageName> --tags=<hereIsFPGAImageTag> --encrypted=false --shell=<hereIsShellVersionOfFPGA>
```

b. Run the following command to check whether the FPGA image can be downloaded:

```
faascmd list_images
```

The command output is as follows:

- If `"State" : "compiling"` is displayed, the FPGA image is being compiled.
- If `"State" : "success"` is displayed, the FPGA image is downloaded. You must find and record the value of FpgaImageUUID.

c. Run the following command. Find and record the value of FpgaUUID in the command output.

```
faascmd list_instances --instanceId=<hereIsYourInstanceId> # Replace <hereIsYourInstanceId> with the ID of the f3 instance.
```

d. Run the following command to download the FPGA image:

```
#Replace <hereIsYourInstanceId> with the ID the f3 instance. Replace <hereIsFpgaUUID> with the value of FpgaUUID that you recorded. Replace <hereIsImageUUID> with the value of FpgaImageUUID that you recorded.
faascmd download_image --instanceId=<hereIsYourInstanceId> --fpgauuid=<hereIsFpgaUUID> --fpgatype=xilinx --imageuuid=<hereIsImageUUID> --imagetype=afu --shell=<hereIsShellVersionOfFpga>
```

e. Run the following command to check whether the image is downloaded:

```
# Replace <hereIsFpgaUUID> with the value of FpgaUUID. Replace <hereIsYourInstanceId> with the ID of the f3 instance.
faascmd fpga_status --fpgauuid=<hereIsFpgaUUID> --instanceId=<hereIsYourInstanceId>
```

The following figure shows the command output. If the value of FpgaImageUUID in the command output is the same as that of FpgaImageUUID that you recorded, and `"TaskStatus":"valid"` is displayed, the image is downloaded.

FAQ

Questions 1: How do I view the details of errors that occur during the image upload?

If your project reports errors during the image upload, such as compilation errors, you can view the details of errors by using one of the following methods:

- Check *faas_compiling.log*. When the upload script *faas_upload_and_create_image.sh* is used, *faas_compiling.log* is automatically downloaded and displayed onto the terminal if the compilation fails.
- Run the `sh /root/xbinst_oem/tool/faas_checklog.sh <bit.tar.gz - previously uploaded package name>` command to view the log file.

Question 2: How do I reload the image?

You can perform the following operations to reload the image:

1. Uninstall the driver.
   - If you installed the `xdma` driver, run the `sudo rmmod xdma` command in the instance to uninstall the driver.
   - If you installed the `xocl` driver, run the `sudo rmmod xocl` command in the instance to uninstall the driver.
2. Download the image. You can use one of the following methods to download the image:
   - Use the script:

     ```
     sh faas_download_image.sh bit.tar.gz <0/1> #The last number stands for the FPGA serial numb
     er in the instance.
     ```

   - Use faascmd:

     ```
     faascmd download_image --instanceId=hereIsYourInstanceId --fpgauuid=hereIsFpgaUUID --fpg
     atype=xilinx --imageuuid=hereIsImageUUID --imagetype=afu --shell=hereIsShellVersionOfFpga
     ```

3. Install the driver.
   - Run the following commands to install the `xdma` driver:

     ```
     sudo depmod
     sudo modprobe xdma
     ```

   - Run the following commands to install the `xocl` driver:

     ```
     sudo depmod
     sudo modprobe xocl
     ```

14.5. faascmd tool

14.5.1. faascmd overview

faascmd is a command-line tool provided by the Alibaba Cloud FPGA cloud server. It is a script that is developed based on the Python SDK.

You can use faascmd to:

- Perform authorization and related operations.
- Manage and operate FPGA images.
- View and upload objects.
- Obtain information about FPGA instances.

14.5.2. Install faascmd

This topic describes how to download and install faascmd.

Prerequisites

Before you install faascmd, make sure that the following operations are complete:

1. The AccessKey ID and AccessKey secret of the RAM user are obtained. For more information, see .
2. Python is installed and the version of SDK for Python is checked. The procedure is as follows:
   i. Run the `python -V` command to check whether the Python version is 2.7.x.
   ii. Run the following commands to install Python modules:

```
pip -q install oss2
pip -q install aliyun-python-sdk-core
pip -q install aliyun-python-sdk-faas
pip -q install aliyun-python-sdk-ram
```

   iii. Run the following command to check whether the version of aliyun-python-sdk-core is 2.11.0 or later:

```
cat /usr/lib/python2.7/site-packages/aliyunsdkcore/__init__.py
```

> **Note** If the version is earlier than 2.11.0, run the `pip install --upgrade aliyun-python-sdk-core` command to upgrade aliyun-python-sdk-core to the latest version.

Procedure

1. Log on to the instance. Run the `wget http://fpga-tools.oss-cn-shanghai.aliyuncs.com/faascmd` command in any directory to download faascmd.

> **Note** Record the directory. When you configure the faascmd tool, you must add the absolute path of the directory where faascmd is located to the PATH variable.

2. Run the following command to make faascmd executable:

```
chmod +x faascmd
```

14.5.3. Configure faascmd

Before using faascmd, you need to configure the related environment variable and the AccessKey of the RAM user.

Procedure

1. Log on to your instance and configure the PATH environment variable by running the following command:

```
export PATH=$PATH:<path where faascmd is located>
```

2. Configure the AccessKey (that is, the AccessKeyId and AccessKeySecret) by running the following command:

```
faascmd config --id=<yourAccessKeyID> --key=<yourAccessKeySecret>
```

14.5.4. Use faascmd

This topic describes how to use faascmd commands.

Prerequisites

The faascmd tool is configured. For more information, see Configure faascmd.

The following conditions are met before you use the authorization command:

1. An Object Storage Service (OSS) bucket is created for FPGA as a Service (FaaS) to upload the originally compiled DCP file.
2. A folder named *compiling_logs* is created in the bucket.

Context

The description of the faascmd command syntax is as follows:

- All commands and parameters provided by faascmd are case-sensitive.
- In faascmd commands, extra spaces are not allowed among parameters, `equal signs (=)` , and values.

This topic describes the following faascmd commands:

- Authorize RAM users
- View an authorization policy
- Delete an authorization policy
- View all objects in an OSS bucket
- Upload an originally compiled file
- Download an object from an OSS bucket
- Create an FPGA image
- View FPGA images
- Delete an FPGA image

- Download an FPGA image
- View the download status of an FPGA image
- Publish an FPGA image
- View information of an FPGA-based ECS instance

Authorize RAM users

You can run the `faascmd auth` command to authorize a RAM user to access your OSS buckets as an FaaS administrator.

The command format:

```
faascmd auth --bucket=<YourFaasOSSBucketName>
```

Sample code

> **Note** If your Alibaba Cloud account has multiple RAM users, we recommend that you have the OSS bucket shared among the RAM users to prevent authorization policies from being repeatedly modified or overwritten.

View an authorization policy

You can run the `faascmd list_policy` command to check whether the specified OSS bucket is contained in the corresponding authorization policy (faasPolicy).

The command format:

```
faascmd list_policy
```

Sample code

> **Note** You need to check whether your OSS bucket and the compiling_logs folder in the bucket are displayed in the policy information.

Delete an authorization policy

You can run the `faascmd delete_policy` command to delete an authorization policy (faasPolicy).

The command format:

```
faascmd delete_policy
```

Sample code

Note If your Alibaba Cloud account has multiple RAM users, we recommend that you perform this operation in the RAM console to avoid unexpected policy deletion.

View all objects in an OSS bucket

You can run the `faascmd list_objects` command to view all objects in an OSS bucket.

The command format:

```
faascmd list_objects
```

Sample code

Note You can combine this command with the grep command to filter files. Example: `faascmd list_objects | grep "xxx"` .

Upload an originally compiled file

You can run the `faascmd upload_object` command to upload original files that are compiled on your local PC to a specified OSS bucket.

The command format:

```
faascmd upload_object --object=<NewFileNameInOSSBucket> --file= <YourFilePath>/<FileNameYouWa
ntToUpload>
```

Sample code

Note

- If the target file is stored in the current directory, you do not need to specify a path.
- Locally compiled original files provided by Intel FPGA are in the .gbs format and those provided by Xilinx FPGA are compressed as packages in the .tar format after script processing.

Download an object from an OSS bucket

You can run the `faascmd get_object` command to download the specified object from an OSS bucket.

The command format:

```
faascmd get_object --object=<YourObjectName> --file=<YourLocalPath>/<YourFileName>
```

Sample code

Note If you do not specify a path, the object is downloaded to the current folder.

Create an FPGA image

You can run the `faascmd create_image` command to submit a request to create an FPGA image. If the request succeeds, fpga imageuuid is returned.

The command format:

```
faascmd create_image --object=<YourObjectName>
--fpgatype=<intel/xilinx>  --encrypted=<true/false>
--kmskey=<key/ If encrypted is set to true, this parameter is required. Otherwise, this parameter is op
tional.>
--shell=<Shell Version/Required> --name=<name/Optional>
--description=<description/Optional> --tags=<tags/Optional>
```

Sample code

View FPGA images

You can run the `faascmd list_images` command to view information of all FPGA images that you created.

The command format:

```
faascmd list_images
```

Sample code

Note Each RAM user can have up to ten FPGA images.

Delete an FPGA image

You can run the `faascmd delete_image` command to delete an FPGA image.

The command format:

```
faascmd delete_image --imageuuid=<yourImageuuid>
```

Sample code

Download an FPGA image

You can run the `faascmd download_image` command to submit a request to download an FPGA image.

The command format:

```
faascmd download_image  --instanceId=<YourInstanceId>
--fpgauuid=<Yourfpgauuid> --fpgatype=<intel/xilinx>
--imageuuid=<YourImageuuid> --imagetype=<afu>
--shell=<YourImageShellVersion>
```

Sample code

```
faascmd download_image --instanceId=XXXXX --fpgauuid=XXXX --fpgatype=intel --imageuuid=XXXX
```

View the download status of an FPGA image

You can run the `faascmd fpga_status` command to view the status of the current FPGA board card or download progress of the FPGA image.

The command format:

```
faascmd fpga_status --fpgauuid=<Yourfpgauuid> --instanceId=<YourInstanceId>
```

Sample code

Publish an FPGA image

You can run the `faascmd publish_image` command to submit a request to publish an FPGA image.

The command format:

```
faascmd publish_image --imageuuid=<YourImageuuid> --imageid=<YourFPGAImageid>
```

> **Note**
>
> - imageuuid indicates the ID of the image that you want to publish to Alibaba Cloud Marketplace. You can run the `faascmd list_images` command to view this image ID.
> - imageid indicates the image ID of the current FPGA-based instance. You can go to the product page in the ECS console to view this image ID.

View information of an FPGA-based ECS instance

You can run the `faascmd list_instances` command to view basic information of an FPGA-based ECS instance, including the instance ID, FPGA board card information, and shell version.

The command format:

```
faascmd list_instances --instanceId=<YourInstanceId>
```

Sample code

14.5.5. faascmd FAQ

This topic provides answers to commonly asked questions about the faascmd tool.

- FAQ
  - What do I do if the "Name Error:global name'ID' is not defined." error message is returned?
  - What do I do if the "SDK.InvalidRegionId. Can not find endpoint to access." error message is returned?
  - What do I do if the "HTTP Status: 404 Error: EntityNotExist." Role Error. The specified Role not exists . error message is returned?
  - When I attempt to download an FPGA image, the "HTTP Status:404 Error:SHELL NOT MATCH. The image Shell is not match with fpga Shell! Request ID:D7D1AB1E-8682-4091-8129-C17D54FD10D4" error message is returned. What do I do?
  - When I attempt to download an FPGA image, the "HTTP Status:503 Error:ANOTHER TASK RUNNING. Another task has not finished yet, please retry later! Request ID: 5FCB6F75-8572-4840-9BDC-87C57174F26D" error message is returned. What do I do?
  - What do I do if the image status is displayed as failed when I run the faascmd list_images command?
- Error codes

What do I do if the "Name Error:global name'ID' is not defined." error message is returned?

Cause: faascmd cannot obtain your AccessKey ID or AccessKey secret.

Solution: Run the `faascmd config` command to save the AccessKey ID and AccessKey secret information you entered to the */root/.faascredentials* file.

What do I do if the "SDK.InvalidRegionId. Can not find endpoint to access." error message is returned?

Cause: faascmd cannot obtain the endpoint of FPGA as a Service (FaaS).

Solution: Perform the following steps to check whether faascmd configurations meet the specified requirements:

- Run the `python -V` command to check whether the installed version of Python is 2.7.x.
- Run the `which python` command to check whether the default installation path of Python is `/usr/bin/python`.
- Run the `cat /usr/lib/python2.7/site-packages/aliyunsdkcore/__init__.py` command to check whether the installed version of aliyunsdkcore is 2.11.0 or later.

> **Note** If the aliyunsdkcore version is earlier than 2.11.0, run the `pip install --upgrade aliyun-python-sdk-core` command to upgrade aliyunsdkcore to the latest version.

What do I do if the "HTTP Status: 404 Error: EntityNotExist." Role Error. The specified Role not exists . error message is returned?

Cause: AliyunFAASDefaultRole does not exist in your Alibaba Cloud account.

Solution: Log on to theRAM console to check whether AliyunFAASDefaultRole exists.

- If AliyunFAASDefaultRole does not exist, run the `faascmd config` and `faascmd auth` commands to create the role and grant permissions to it.
- If AliyunFAASDefaultRole exists, submit a ticket.

When I attempt to download an FPGA image, the "HTTP Status:404 Error:SHELL NOT MATCH. The image Shell is not match with fpga Shell! Request ID:D7D1AB1E-8682-4091-8129-C17D54FD10D4" error message is returned. What do I do?

Cause: The shell versions of the target FPGA image and the specified FPGA do not match.

Solution: Perform the following steps:

- Run the `faascmd list_instances --instance=xxx` command to check the shell version of the current FPGA.
- Run the `faascmd list_images` command to check the shell version of the specified FPGA image.

> **Note**
> - If the two shell versions are different, you must create a new FPGA image of the same shell version as the FPGA, and then download the image.
> - If the two shell versions are the same, submit a ticket.

When I attempt to download an FPGA image, the "HTTP Status:503 Error:ANOTHER TASK RUNNING. Another task has not finished yet, please retry later! Request ID: 5FCB6F75-8572-4840-9BDC-87C57174F26D" error message is returned. What do I do?

Cause: The FPGA is still in the operating state due to an unexpected failure or interruption of the download request you submitted.

Solution: We recommend that you wait 10 minutes until the download task ends, and then resubmit an image download request.

> **Note** If this problem persists, submit a ticket.

What do I do if the image status is displayed as failed when I run the faascmd list_images command?

Run the following commands to obtain the compilation log for troubleshooting:

```
faascmd list_objects|grep vivado
faascmd get_object --object=<YourObjectName> --file=<YourLocalPath>/vivado.log  #If no path is specified, the compilation log is downloaded to the current folder.
```

Error codes

HTTP status code	Error code	Error message	Description	Application scope
400	PARAMETER INVALIDATE	Specify parameters are invalid.	The error message returned because input parameters are invalid.	• All faascmd commands • All API operations
500	InternalError	The request processing has failed due to some unknown error.	The error message returned because an unknown error has occurred. Submit a ticket.	
404	InvalidProduct.NotFound	Cannot find product according to your specified domain.	The error message returned because the FaaS product does not exist. Check whether the endpoint configuration of Python Core SDK is correct.	
404	InvalidAccessKeyId.NotFound	Specified access key is not found.	The error message returned because the specified AccessKey ID does not exist.	
400	InvalidAccessKeyId.Inactive	Specified access key is disabled.	The error message returned because the specified AccessKey pair is disabled.	
400	InvalidSecurityToken.Expired	Specified SecurityToken is expired.	The error message returned because the specified security token has expired.	
400	InvalidSecurityToken.Malformed	Specified SecurityToken is malformed.	The error message returned because the specified security token is invalid.	

HTTP status code	Error code	Error message	Description	Application scope
400	InvalidSecurityToken.MismatchWithAccessKey	Specified SecurityToken mismatch with the AccessKey.	The error message returned because the specified security token does not match the AccessKey pair.	
403	NoPermisson	You are not authorized to do this action.	The error message returned because the current RAM user is not authorized to perform this operation.	• faascmd command: auth • API operation: auth
401	IMAGE NUMBER EXCEED	The user is allowed to have no more than 30 images.	The error message returned because the number of images has reached the upper limit of 30. Delete images that are no longer needed.	• faascmd command: create_image • API operation: CreateFpgaImage
503	FREQUENCY ERROR	CreateFpgaImage task is allowed to take every half an hour.	The error message returned because the interval between two consecutive image creation requests is less than 30 minutes.	
404	SHELL NOT SUPPORT	The shellUUID is not supported, please check your input shellUUID.	The error message returned because the specified shell version is not supported.	
404	EntityNotExist.RoleError	The specified Role not exists.	The error message returned because AliyunFAASDefaultRole does not exist in your account.	
403	AccessDeniedError	The bucket you visit does not belong to you.	The error message returned because the RAM user used to configure faascmd is not authorized to access the current bucket.	

HTTP status code	Error code	Error message	Description	Application scope
403	CALLER TYPE NOT SUPPORT	The callerType is not supported, please use sub user's AK.	The error message returned because the specified user identity credential is not supported. Only the identity credentials of RAM users are supported.	
404	NoSuchBucketError	The specified bucket does not exist.	The error message returned because the specified OSS bucket does not exist. Check whether the specified bucket name is correct.	
404	OSS OBJECT NOT FOUND	The specified oss object does not exist.	The error message returned because the specified OSS object does not exist or you have not authorized the FaaS RAM role to access the object.	

404	IMAGE NOT FOUND	The specify image does not found.	The error message returned because the specified fpgaImage parameter does not exist.	• faascmd command: delete_image
				• API operations:
				  ◦ DeleteFpgaImage
				  ◦ DeletePublishFpgaImage

401	NOT AUTHORIZED	You are not allowed to access this instance.	The error message returned because you are not authorized to access the specified instance. Check whether the permission policy of your account includes the permission to call the DescribeInstances operation.	

HTTP status code	Error code	Error message	Description	Application scope
				• faascmd command: list_instances • API operation: DescribeFpgaInstances
403	CALLER TYPE NOT SUPPORT	The callerType is not supported.	The error message returned because the specified user identity credential is not supported. Only STS tokens and the AccessKey pairs of RAM users are supported.	
404	INSTANCE INVALIDATE	The instance you specify is not FPGA type.	The error message returned because the specified instance is not an FPGA-based ECS instance. If you are sure that the instance is an FPGA-based ECS instance, submit a ticket.	
401	NOT AUTHORIZED	You are not allowed to access this instance.	The error message returned because the specified instance ID does not exist. Check the input parameters.	• faascmd command: fpga_status • API operation: DescribeLoadTaskStatus
404	FPGA NOT FOUND	The fpga you specify is not found.	The error message returned because the specified fpgauuid parameter does not exist. Check the input parameters.	
503	ANOTHER TASK RUNNING	Another task is running, user is allowed to take this task half an hour.	The error message returned because the previous image download task you submitted is still in the operating state.	
401	IMAGE ACCESS ERROR	You are not allowed to access this fpga image.	The error message returned because the specified image does not belong to your account.	

HTTP status code	Error code	Error message	Description	Application scope
401	YOU HAVE NO ACCESS TO THIS INSTANCE	You are not allowed to access this instance.	The error message returned because the specified instance does not belong to your account.	• faascmd command: download_image • API operation: LoadFpgaImage
404	IMAGE NOT FOUND	The fpga image you specify is not found.	The error message returned because the specified fpgaImage parameter does not exist.	
404	FPGA NOT FOUND	The fpga you specify is not found.	The error message returned because the specified FPGA-based ECS instance does not exist.	
404	SHELL NOT MATCH	The imageShell is not match with fpgaShell.	The error message returned because the shell version of the specified image does not match that of the specified FPGA-based ECS instance.	
403	ASSUME ROLE USER NOT SUPPORT	AssumeRoleUser only support loading market fpga images.	The error message returned because the STS token is used to download an FPGA image that is not an Alibaba Cloud Marketplace image. STS tokens can only be used to download Alibaba Cloud Marketplace images.	
404	Image not in success state	The fpga image you specify is not in success state.	The error message returned because the specified FPGA image is not in the success state. Only images in the success state can be downloaded.	

HTTP status code	Error code	Error message	Description	Application scope
404	FPGA IMAGE STATE ERROR	The specify fpga image is not in success state.	The error message returned because the specified FPGA image is not in the success state.	• faascmd command: publish_image • API operation: PublishFpgaImage
404	FPGA IMAGE NOT FOUND	The specify fpga image does not found.	The error message returned because the specified image does not exist or does not belong to your account.	

15.Process ECS status change events

This topic describes how CloudMonitor automatically processes ECS status change events by using MNS message queues.

Overview

An ECS instance status change event is triggered when the instance status changes. Specifically, a status change event can indicate changes resulting from operations on the console, the usage of APIs or SDKs, automatic scaling, detection of overdue payments, system exceptions, and more.

To automate the processing of ECS status change events, CloudMonitor provides two methods: function calculation formulas and MNS message queues. This topic describes three best practice cases that use MNS message queues.

Preparations

- **Create a message queue.**
  i. Log on to the MNS Console.
  ii. On the **Queue List** page, select the target region, and click **Create Queue** in the upper-right corner.
  iii. In the **New Queue** dialog box, enter the queue name (for example, ecs-cms-event) and other required information, and then click **OK**.
- **Create an alarm rule for status change events.**
  i. Log on to the CloudMonitor Console.
  ii. In the left-side navigation pane, click **Event Monitoring**.
  iii. Switch to the **Alarm Rules** tab page, and then click **Create Event Alerts**.
  iv. In the **Basic Information** area, enter a name for the alarm rule, for example, ecs-test-rule.
  v. In the **Event alert** area, set the parameters as follows:
    - Set **Event Type** to **System Event**.
    - Set **Product Type** to **ECS** and **Event Type** to **StatusNotiifcation**, and set other parameters as needed.
    - If **Resource Range** is set to **All Resources**, change events of any resource will trigger notifications. If **Resource Range** is set to **Application Groups**, only change events of the resources within the specified group will trigger notifications.
  vi. In the **Alarm Type** area, select **MNS queue**, and then specify **Region** and **Queue** (for example, ecs-cms-event).
  vii. Click **OK**.
- **Install Python dependencies.**

  The following code is tested in Python 3.6. You can use other programming languages, such as Java, as needed.

  Use PyPi to install the following Python dependencies:

  - aliyun-python-sdk-core-v3 of 2.12.1 or later

- aliyun-python-sdk-ecs of 4.16.0 or later
- aliyun-mns of 1.1.5 or later

Procedure

CloudMonitor sends all status change events of ECS instances to MNS. You can then obtain the notifications from MNS and process them by running code. The following practice sections overview a complete tutorial of the preceding methods.

Practice 1: Records of all ECS creation and release events

Currently, you cannot query instances that have been released on the ECS console. If you need to perform these queries, you need to record the life cycle of all ECS instances in your own database or log through an ECS status change event. Specifically, whenever an ECS instance is created, a Pending event will be sent, and whenever an ECS instance is released, a Deleted event will be sent. You can record these two events by performing the following steps:

1. Create a *Conf* file, which must include the MNS endpoint, AccessKeyId and AccessKeySecret of your Alibaba Cloud account, region ID (for example, cn-beijing), and the MNS queue name.

   > **Note** To view the MNS endpoint, you can log on to the MNS console, and click **Get Endpoint** on the **Queue List** page.

   ```
   class Conf:
     endpoint = 'http://<id>.mns.<region>.aliyuncs.com/'
     access_key = '<access_key>'
     access_key_secret = '<access_key_secrect>'
     region_id = 'cn-beijing'
     queue_name = 'test'
     vsever_group_id = '<your_vserver_group_id>'
   ```

2. Use the MNS SDK to compile an MNS client to receive MNS messages.

   ```
   # -*- coding: utf-8 -*-
   import json
   from mns.mns_exception import MNSExceptionBase
   import logging
   from mns.account import Account
   from . import Conf


   class MNSClient(object):
     def __init__(self):
       self.account =  Account(Conf.endpoint, Conf.access_key, Conf.access_key_secret)
       self.queue_name = Conf.queue_name
       self.listeners = dict()

     def regist_listener(self, listener, eventname='Instance:StateChange'):
   ```

```
    def register_listener(self, listener, eventname='instance.StateChange'):
        if eventname in self.listeners.keys():
            self.listeners.get(eventname).append(listener)
        else:
            self.listeners[eventname] = [listener]

    def run(self):
        queue = self.account.get_queue(self.queue_name)
        while True:
            try:
                message = queue.receive_message(wait_seconds=5)
                event = json.loads(message.message_body)
                if event['name'] in self.listeners:
                    for listener in self.listeners.get(event['name']):
                        listener.process(event)
                queue.delete_message(receipt_handle=message.receipt_handle)
            except MNSExceptionBase as e:
                if e.type == 'QueueNotExist':
                    logging.error('Queue %s not exist, please create queue before receive message.', self.
queue_name)
                else:
                    logging.error('No Message, continue waiting')


class BasicListener(object):
    def process(self, event):
        pass
```

The preceding code is used only to pull MNS messages and delete the messages after the listener consumption message is called.

3. Register a listener to use a specified event. When this listener determines that it has received a Pending or Deleted event, it prints a row in the log file.

```
 # -*- coding: utf-8 -*-
import logging
from .mns_client import BasicListener



class ListenerLog(BasicListener):
    def process(self, event):
        state = event['content']['state']
        resource_id = event['content']['resourceId']
        if state == 'Panding':
            logging.info(f'The instance {resource_id} state is {state}')
        elif state == 'Deleted':
            logging.info(f'The instance {resource_id} state is {state}')
```

The following Main function can also be used:

```
mns_client = MNSClient()


mns_client.regist_listener(ListenerLog())


mns_client.run()
```

In your actual scenario, you can store events in your database or use SLS to facilitate search and audit tasks at a later date.

Practice 2: Automatic restart of ECS servers

In some scenarios, ECS servers may shut down unexpectedly. In this case, you need to set automatic restart for the servers.

Use the MNS client in Practice 1 and create a new listener. Then, when the listener receives a Stopped event, the listener executes a `Start` command on the target ECS server.

```
# -*- coding: utf-8 -*-
import logging
from aliyunsdkecs.request.v20140526 import StartInstanceRequest
from aliyunsdkcore.client import AcsClient
from .mns_client import BasicListener
from .config import Conf


class ECSClient(object):
    def __init__(self, acs_client):
        self.client = acs_client

    #Start the ECS instance
    def start_instance(self, instance_id):
        logging.info(f'Start instance {instance_id} ...')
        request = StartInstanceRequest.StartInstanceRequest()
        request.set_accept_format('json')
        request.set_InstanceId(instance_id)
        self.client.do_action_with_exception(request)


class ListenerStart(BasicListener):
    def __init__(self):
        acs_client = AcsClient(Conf.access_key, Conf.access_key_secret, Conf.region_id)
        self.ecs_client = ECSClient(acs_client)

    def process(self, event):
        detail = event['content']
        instance_id = detail['resourceId']
        if detail['state'] == 'Stopped':
            self.ecs_client.start_instance(instance_id)
```

In your actual scenario, after the `Start` command is executed, you will receive Starting, Running, or Stopped event notifications. In this case, you can proceed with the procedure upon command execution for more detailed O&M with the help of a timer and a counter.

Practice 3: Automatic removal of preemptible instances from SLB before they are released

A release alarm event will be sent five minutes before a preemptible instance is released. During these five minutes, you can run some processes without your services being interrupted. For example, you can manually remove the target preemptible instance from the backend SLB server.

Use the MNS client in Practice 1 and create a new listener. Then, when the listener receives the preemptible instance release alarm, the listener calls an SLB SDK.

```
# -*- coding: utf-8 -*-
from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.request import CommonRequest
from .mns_client import BasicListener
from .config import Conf


class SLBClient(object):
    def __init__(self):
        self.client = AcsClient(Conf.access_key, Conf.access_key_secret, Conf.region_id)
        self.request = CommonRequest()
        self.request.set_method('POST')
        self.request.set_accept_format('json')
        self.request.set_version('2014-05-15')
        self.request.set_domain('slb.aliyuncs.com')
        self.request.add_query_param('RegionId', Conf.region_id)

    def remove_vserver_group_backend_servers(self, vserver_group_id, instance_id):
        self.request.set_action_name('RemoveVServerGroupBackendServers')
        self.request.add_query_param('VServerGroupId', vserver_group_id)
        self.request.add_query_param('BackendServers',
                                     "[{'ServerId':'" + instance_id + "','Port':'80','Weight':'100'}]")
        response = self.client.do_action_with_exception(self.request)
        return str(response, encoding='utf-8')


class ListenerSLB(BasicListener):
    def __init__(self, vsever_group_id):
        self.slb_caller = SLBClient()
        self.vsever_group_id = Conf.vsever_group_id

    def process(self, event):
        detail = event['content']
        instance_id = detail['instanceId']
        if detail['action'] == 'delete':
            self.slb_caller.remove_vserver_group_backend_servers(self.vsever_group_id, instance_id)
```

> **Notice**
>
> The event name of the preemptible instance release alarm is Instance:PreemptibleInstanceInterruption", mns_client.regist_listener(ListenerSLB(Conf.vsever_group_id), 'Instance:PreemptibleInstanceInterruption').

In your actual scenario, you need to apply for a new preemptible instance and attach it to SLB to guarantee that your services can run normally.

16.DevOps tutorials

16.1. DevOps for small and medium web apps

16.1.1. General introduction

The intended audience of this document are independent development teams that need to develop and maintain a small/medium web application on Alibaba Cloud. The goal is to keep things simpl. Necessary technologies and best practices are introduced step by step.

Introduction

More complex tooling is mentioned near the middle of this tutorial, for example infrastructure as code tools are explained in Continuous delivery.

The sample web application that comes with this tutorial is composed of two parts:

- A backend written in Java with Spring Boot.
- A frontend written in Javascript with React.

This document addresses the following points:

- How to automate compilation, testing, code analysis and packaging with a CI pipeline.
- How to extend this pipeline to deploy the application automatically.
- How to setup a highly-available architecture on Alibaba Cloud.
- How to backup periodically (and restore!) the database and the version control system.
- How to upgrade the application and the database.
- How to centralize logs and monitor your cluster.

Prerequisites

To follow this tutorial:

- Familiarize yourself with Git and install it on your computer.
- Make sure you have an Alibaba Cloud account.
- Download the related resources before moving to the next part.

16.1.2. Install and configure GitLab

This topic describes how to install and configure GitLab.

Introduction

GitLab CE edition is a free open-source tool that helps you host Git repositories and run your CI/CD pipeline.

To keep it simple, you can install GitLab on an ECS instance with a direct access to Internet. Although the servers will be protected via encryption and restrictive security group rules, you might also want to isolate your virtual machines from Internet by using a VPN Gateway.

The following diagram illustrates the architecture for GitLab.

Create cloud resources

The first step is to buy a domain name. This is necessary if you want to enable security on your servers:

1. Log on to the Domain console.
2. Click **Purchase**.
3. Choose a domain, such as **my-sample-domain.xyz** and follow the instructions to buy it.
4. Return to the console and refresh the page in order to see your new domain.

> **Note** Due to a limitation in Direct Mail, choose a domain name with less than 28 characters.

The second step is to create ECS instances and related resources:

1. Log on to the VPC console.
2. Select the region where you want to create the VPC on top of the page, for example, Singapore.
3. Click **Create VPC**.
4. Fill in the new form with the following information:
   - VPC name = devops-simple-app-vpc
   - VPC destination CIDR Block = “192.168.0.0/16”
   - VSwitch name = devops-simple-app-vswitch
   - VSwitch zone = first zone of the list
   - VSwitch destination CIDR Block = “192.168.0.0/24”
5. Click **OK** to create the VPC and the VSwitch.
6. In the VPC list, click the VPC you have just created.
7. Scroll down and click **0** at the right of **Security Group**.
8. In the new page, click **Create Security Group**.
9. Fill in the new form with the following information:
   - Template = Web Server Linux
   - Security Group Name = devops-simple-app-security-group
   - Network Type = VPC
   - VPC = select the VPC you just created (with the name devops-simple-app-vpc)
10. Click **OK** to create the security group and the rules from the template. Note that the rules open the ports for SSH, HTTP, HTTPS and ICMP to any computer on Internet.
11. Log on to the ECS console.
12. Click **Create Instance**.
13. If needed, select **Advanced Purchase** (also named **Custom**).
14. Fill in the wizard with the following information:
    - Billing Method = Pay-As-You-Go
    - Region = same as your VPC and the same availability zone as the VSwitch

- Instance Type = filter by vCPU = 2, Memory = 4 GiB, Current Generation tab, and select a remaining type such as **ecs.n4.large**
- Image = Ubuntu 18.04 64bit
- System Disk = Ultra Disk 40 GiB
- Network = VPC, select the VPC and VSwitch you have just created
- Do NOT assign a public IP (we will create an EIP instead, which is more flexible)
- Security Group = select the group you have just created
- Log on Credentials = select **Password** and choose one
- Instance Name = devops-simple-app-gitlab
- Host = devops-simple-app-gitlab
- Read and accept the terms of service

15. Finish the instance creation by clicking **Create Instance**.
16. Go back to the console, click **Instances** from the left-side navigation pane, and select a region. Your new instance is displayed.
17. Click **EIP** in the left-side navigation pane.
18. On the new page, click **Create EIP**.
19. Fill in the wizard with the following information:
    - Region = the region where you have created your ECS
    - Max Bandwidth = 1 Mbps
    - Quantity = 1
20. Click **Buy Now**, check the agreement of service, and click **Activate**.
21. Go back to the console and check your new EIP.
22. Next to your new EIP, click **Bind**.
23. Fill in the new form with the following information:
    - Instance Type = ECS Instance
    - ECS Instance = devops-simple-app-gitlab/i-generatedstring
    - Click **OK** to bind the EIP to your ECS instance.
24. Copy the IP address of your EIP (for example, 47.88.155.70).

The ECS instance is ready for GitLab. Now register a sub-domain for this machine:

1. Log on to the Domain console.
2. On the row corresponding to your domain (for example, my-sample-domain.xyz), click **Resolve**.
3. Click **Add Record**.
4. Fill in the new form with the following information:
    - Type = A- IPV4 address
    - Host = gitlab
    - ISP Line = Outside mainland China
    - Value = The EIP IP Address (for example, 47.88.155.70)

- TTL = 10 minute(s)

5. Click **OK** to add the record.

Install GitLab

Open a terminal on your computer and type:

```
# Connect to the ECS instance
ssh root@gitlab.my-sample-domain.xyz # Use the password you set when you have created the ECS in
stance

# Update the machine
apt-get update
apt-get upgrade

# Add the GitLab repository for apt-get
cd /tmp
curl -LO https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.deb.sh
bash /tmp/script.deb.sh

# Install GitLab
apt-get install gitlab-ce

# Open GitLab configuration
nano /etc/gitlab/gitlab.rb
```

> **Note** If you use MAC OSX, you must first disable the setting **Set locale environment variables on startup** in **Preferences > Profiles > Advanced**.

In the GitLab configuration file, replace the value of `external_url` by `http://gitlab.my-sample-domain.xyz` (the domain you have just purchased and configured), and then save and quit by pressing **Ctrl+X**.

Now start GitLab and try it. In your terminal, run the following command:

```
gitlab-ctl reconfigure
```

Open your web browser on `http://gitlab.my-sample-domain.xyz`. The following figure is displayed.

If the preceding figure is not displayed, first make sure you did not miss a step, and then raise an issue if the problem persists.

Do not enter your new password because you are using an unencrypted connection. Now fix this problem.

Configure HTTPS

Open your terminal and enter the following commands:

```
# Connect to the ECS instance
ssh root@gitlab.my-sample-domain.xyz # Use the password you set when you have created the ECS in
stance

# Install dependencies
apt-get install ca-certificates openssh-server
apt-get install postfix # During the installation, select "Internet Site" and set your domain (for exampl
e, gitlab.my-sample-domain.xyz)

# Open GitLab configuration
nano /etc/gitlab/gitlab.rb
```

The last command allows you to edit GitLab configuration:

1. Modify the value of `external_url` by adding an `s` to `http://` into `https://` (for example, https://gitlab.my-sample-domain.xyz).
2. Scroll to **Let's Encrypt integration** and insert the following lines:

   ```
   letsencrypt['enable'] = true
   letsencrypt['contact_emails'] = ["john.doe@your-company.com"] # Your email address
   letsencrypt['auto_renew'] = true
   letsencrypt['auto_renew_hour'] = 11
   letsencrypt['auto_renew_minute'] = 42
   letsencrypt['auto_renew_day_of_month'] = "*/14"
   ```

   Quit and save the file by pressing **Ctrl+X**, and then apply the configuration change and restart GitLab:

   ```
   gitlab-ctl reconfigure
   ```

   Check it worked by opening your web browser and visit https://gitlab.my-sample-domain.xyz (with the `s` in https).

You can now enter your new password and sign in with the username `root` and your new password. You can now access the GitLab dashboard.

Before going further, you still need to configure:

- An email server so that GitLab can send emails.
- Automatic backup to avoid losing data.

Configure the mail server

> **Note** Direct Mail is not available in all regions, but you can configure it in a different one from where you have created your ECS instance. Direct Mail is available in China (Hangzhou), Singapore, and Australia (Sydney). Contact us if you need it in another region.

Go back to the Alibaba Cloud web console and perform the following steps:

1. Log on to the Direct Mail console.
2. Select the region on top of the page.
3. Click **Email Domains** in the left-side navigation pane.
4. Click **New Domain**.
5. In the new form, set the domain name to `mail.my-sample-domain.xyz` (the domain you chose earlier with the prefix `mail`).
6. The page must be refreshed with your new email domain. Click the **Configure** link on its right side.
7. The new page explains you how to configure your domain. Keep this web browser tab opened, open a new one, and go to the Domain console.
8. Click the **Resolve** link next to your domain.
9. Click **Add Record**.
10. Fill in the new form with the following information:
    - Type = TXT- Text
    - Host = the **Host record** column under **1,Ownership verification** in the Direct Mail tab (for example, aliyundm.mail)
    - ISP Line = Outside mainland China
    - Value = the **Record value** column under **1,Ownership verification** in the Direct Mail tab (for example, 3cdb41a3351449c2af6f)
    - TTL = 10 minute(s)
11. Click **OK** and click **Add Record** again.
12. Fill in the new form with the following information:
    - Type = TXT- Text
    - Host = the **Host record** column under **2,SPF verification** in the Direct Mail tab (for example, mail)
    - ISP Line = Outside mainland China
    - Value = the **Record value** column under **2,SPF verification** in the Direct Mail tab (for example, v=spf1 include:spfdm-ap-southeast-1.aliyun.com -all)
    - TTL = 10 minute(s)
13. Click **OK** and click **Add Record** again.
14. Fill in the new form with the following information:
    - Type = MX- Mail exchange
    - Host = the **Host record** column under **3,MX Record Verification** in the Direct Mail tab (for example, mail)
    - ISP Line = Outside mainland China

- Value = the **Record value** column under **3,MX Record Verification** in the Direct Mail tab (for example, mxdm-ap-southeast-1.aliyun.com)
- MX Priority = 10
- TTL = 10 minute(s)
- Synchronize the Default Line = checked

15. Click **OK** and click **Add Record** again.
16. Fill in the new form with the following information:
    - Type = CNAME- Canonical name
    - Host = the **Host record** column under **4,CNAME Record Verification** in the Direct Mail tab (for example, dmtrace.mail)
    - ISP Line = Outside mainland China
    - Value = the **Record value** column under **4,CNAME Record Verification** in the Direct Mail tab (for example, tracedm-ap-southeast-1.aliyuncs.com)
    - TTL = 10 minute(s)
17. Click **OK**.

You probably have a domain configuration that looks like the following figure.

Continue with the email server configuration:

1. Go back to the Direct Mail console (the web browser tab you kept opened).
2. Click **Cancel** to go back to the email domain list.
3. Click **Verify** next to your new domain, and confirm when the prompt appears.
4. Refresh the page after 20 seconds. If the status of your domain is still **To Be Verified**, click **Configure** and check which step is still in the **To Be Verified** status, fix your domain configuration, and re-do the previous step (**Verify**). Sometimes the verification step is a bit slow and you need to retry several times. When the email domain status is **Verification successful**, you can go to the next step.
5. Click **Sender Addresses** in the left-side navigation pane.
6. Click **Create Sender Address**.
7. Fill in the new form with the following information:
    - Email Domains = mail.my-sample-domain.xyz (the email domain you just configured)
    - Account = gitlab
    - Reply-To Address = your email address (for example, john.doe@your-company.com)
    - Mail Type = Triggered Emails
8. Click **OK** to close the form.
9. Your new sender address must be added to the list. Click **Set SMTP password** next to it.
10. Set the SMTP password and click **OK**.
11. Click **Verify the reply-to address** next to your new sender address, and confirm when the prompt appears.
12. Check your mailbox corresponding to the address you set in the **Reply-To Address** field. You should have received an email from **directmail**.

13. Click the link in this email to check a confirmation message.
14. Go back to the sender addresses page and save the SMTP address and port at the end of the description. It should be something like **SMTP service address: smtpdm-ap-southeast-1.aliyun.com . SMTP service ports: 25, 80 or 465(SSL encryption).**

Now that the email server is ready. Configure GitLab to use it. Open a terminal on your computer and enter the following commands:

```
# Connect to the ECS instance
ssh root@gitlab.my-sample-domain.xyz # Use the password you set when you have created the ECS in
stance


# Open GitLab configuration
nano /etc/gitlab/gitlab.rb
```

Scroll down to **### Email Settings** and insert the following lines:

```
gitlab_rails['gitlab_email_enabled'] = true
gitlab_rails['gitlab_email_from'] = 'gitlab@mail.my-sample-domain.xyz' # The sender address you have j
ust created
gitlab_rails['gitlab_email_display_name'] = 'GitLab'
gitlab_rails['gitlab_email_reply_to'] = 'gitlab@mail.my-sample-domain.xyz'
```

Scroll down to **### GitLab email server settings** and insert the following lines:

```
gitlab_rails['smtp_enable'] = true
gitlab_rails['smtp_address'] = "smtpdm-ap-southeast-1.aliyun.com"   # SMTP address written in the Dir
ect Mail console
gitlab_rails['smtp_port'] = 465                          # SMTP port written in the Direct Mail console
gitlab_rails['smtp_user_name'] = "gitlab@mail.my-sample-domain.xyz" # Sender address
gitlab_rails['smtp_password'] = "HangzhouMail2018"               # SMTP password for the sender addres
s
gitlab_rails['smtp_domain'] = "mail.my-sample-domain.xyz"        # Your email domain
gitlab_rails['smtp_authentication'] = "login"
gitlab_rails['smtp_enable_starttls_auto'] = false
gitlab_rails['smtp_tls'] = true
```

Apply the configuration change and restart GitLab:

```
gitlab-ctl reconfigure
```

You can test the configuration like this:

1. Go to GitLab and sign in as root: https://gitlab.my-sample-domain.xyz/

2. Click **Admin area** in the top menu (the wrench icon).
3. Click **Users** in the left-side navigation pane.
4. Click **Administrator**.
5. Click **Edit**.
6. Change the **Email** field to your personal email address.
7. Click **Save changes**.
8. Sign out by clicking your profile picture on the upper-right corner of the page and by selecting **Sign out**.
9. Click the **Forgot your password?** link.
10. Set your personal email address and click **Reset password**.
11. Check your personal mailbox and verify you have received an email (it may be in the spam folder).

Automatically back up configuration

Backups are important because they prevent data loss in case of accident and allow you to migrate to another ECS instance if you need.

To run backups automatically, open a terminal and run the following commands:

> **Note** The GitLab documentation requires TAR version of 1.30 or later.

Create an OSS bucket for you to store your backups:

1. Log on to the OSS console.
2. Click **Create Bucket**.
3. Fill in the new form with the following information:
   - Bucket Name = gitlab-my-sample-domain-xyz (you can set the name you want, but it must be unique)
   - Region = the same as your ECS instance (for example, Asia Pacific SE 1 (Singapore))
   - Storage Class = Standard
   - Access Control List (ACL) = Private
4. Click **OK**.
5. The page must show the bucket you have created. Save the last **Endpoint** for VPC Network Access (something like `oss-ap-southeast-1-internal.aliyuncs.com`). It contains your bucket name and the region ID, for example, ap-southeast-1.

You will also need an access key id and secret:

1. Log on to the user management center by clicking on your profile on the upper-right corner of the page and by selecting AccessKey.
2. Click **Create Access Key**.
3. Note the AccessKeyID and the AccessKeySecret, and click **Save AccessKey Information**.

In your terminal, mount your OSS bucket as a folder:

```
# Save your bucket name, access key id and access key secret in the file /etc/passwd-ossfs
# The format is my-bucket:my-access-key-id:my-access-key-secret
echo gitlab-my-sample-domain-xyz:LTAI********ujwZ:rc15yggaCX08A********X49wNUGpk > /etc/passwd-ossfs
chmod 640 /etc/passwd-ossfs

# Create a folder where we will mount the OSS bucket
mkdir /mnt/gitlab-bucket

# Mount the OSS bucket
# The -ourl come from the last "Endpoint" for VPC Network Access
ossfs gitlab-my-sample-domain-xyz /mnt/gitlab-bucket -ourl=http://oss-ap-southeast-1-internal.aliyuncs.com

# Check it works
echo "It works" > /mnt/gitlab-bucket/test.txt

# Unmount the OSS bucket
umount /mnt/gitlab-bucket
```

Check that the test file is present in your bucket:

1. Log on to the OSS console.
2. Click your bucket name in the left-side navigation pane.
3. Select **Files** from the top menu.
4. The file *test.txt* should be present and should contain **It works**.
5. Delete this file.

Configure the OSS bucket so that it is automatically mounted when the ECS instance starts. Create the following file:

Adapt and copy the following content:

Make sure you set the right bucket name and endpoint. Quit and save by pressing **CTRL+X**. Configure Systemd to run this script at startup.

Log on to the OSS console, and check that the *test2.txt* file is present in your bucket and delete it.

Configure GitLab to store its backup files in the mounted folder. Open the terminal and run the following command:

```
# Open GitLab configuration
nano /etc/gitlab/gitlab.rb
```

Scroll to **### Backup Settings** and insert the following line:

```
gitlab_rails['backup_path'] = "/mnt/gitlab-bucket/backup/"
```

Quit and save by pressing **CTRL+X**, and then check if it works:

```
# Apply GitLab configuration
gitlab-ctl reconfigure


# Manually launch a first backup
gitlab-rake gitlab:backup:create
```

The last command should have created a backup. Log on to the OSS console and check you have a file with a path like *backup/1540288854_2018_10_23_11.3.6_gitlab_backup.tar*.

Configure automatic backup so that it is started automatically every night. For that we will create two types of cron jobs: one to execute the preceding backup command and the other to save the GitLab configuration files.

Open your terminal and run the following command:

```
# Edit the CRON configuration file. Select nano as the editor.
crontab -e
```

Add the following lines into this file:

```
0 2 * * * /opt/gitlab/bin/gitlab-rake gitlab:backup:create CRON=1
0 2 * * * /bin/cp /etc/gitlab/gitlab.rb "/mnt/gitlab-bucket/backup/$(/bin/date '+\%s_\%Y_\%m_\%d')_gi
tlab.rb"
0 2 * * * /bin/cp /etc/gitlab/gitlab-secrets.json "/mnt/gitlab-bucket/backup/$(/bin/date '+\%s_\%Y_\%
m_\%d')_gitlab-secrets.json"
```

Save and quit by pressing **CTRL+X**.

You now have configured automatic backup every night at 02:00. If you want to test this configuration, you can replace **0 2 * * *** by the current time plus 2 minutes. For example, if the current time is 14:24, then set **26 14 * * ***. After that, you need to wait about 2 minutes and check whether new files have been created in your OSS bucket.

The restoration process is well described in the official documentation (section **Restore for Omnibus installations**). Note that it is considered as a best practice to test your backups from time to time.

Install and configure GitLab runner

It is a best practice to run CI/CD jobs (including code compilation, unit tests execution, and application packing) on a different machine from the one that runs GitLab.

Thus, we need to set up one runner on a new ECS instance. Follow these steps:

1. Log on to the VPC console.

2. Select the region of the GitLab ECS instance (on the top of the page).
3. Click the VPC **devops-simple-app-vpc**.
4. Click **1** next to **Security Group**.
5. Click **Create Security Group**.
6. Fill in the new form with the following information:
   - Template = Customize
   - Security Group Name = devops-simple-app-security-group-runner
   - Network Type = VPC
   - VPC = select the VPC **devops-simple-app-vpc**
7. Click **OK** to create the group. We will not add any rule in order to be as restrictive as possible (to improve security).
8. Log on to the ECS console.
9. Click **Create Instance**.
10. If needed, select **Advanced Purchase** (also named **Custom**).
11. Fill in the wizard with the following information:
    - Billing Method = Pay-As-You-Go
    - Region = the same as the ECS instance where you have installed GitLab
    - Instance Type = filter by vCPU = 2, Memory = 4 GiB, Current Generation tab, and select a remaining type such as **ecs.n4.large**
    - Image = Ubuntu 18.04 64bit
    - System Disk = Ultra Disk 40 GiB
    - Network = VPC, select the VPC and VSwitch of the GitLab ECS instance
    - Assign a public IP (no need of an EIP this time)
    - Security Group = select **devops-simple-app-security-group-runner**
    - Log on Credentials = select **Password** and choose one
    - Instance Name = devops-simple-app-gitlab-runner
    - Host = devops-simple-app-gitlab-runner
    - Read and accept the terms of service
12. Finish the instance creation by clicking **Create Instance**.
13. Go back to the ECS console, click **Instances** in the left-side navigation pane, and choose your region on top of the page. You should be able to see your new instance **devops-simple-app-gitlab-runner**.
14. Click **Connect** on the right of your ECS instance, copy the VNC Password (something like **667078**) and enter it immediately.
15. You can see a terminal in your web browser inviting you to log in. Authenticate as root with the password you have just created.

Run the following commands in this web-terminal:

```
# Update the machine
apt-get update
apt-get upgrade

# Add a new repository for apt-get for GitLab Runner
curl -L https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.deb.sh | sudo bas
h

# Add a new repository for apt-get for Docker
apt-get install software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
add-apt-repository \
  "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) \
  stable"

# Update the machine
apt-get update

# Install GitLab runner
apt-get install gitlab-runner

# Install dependencies for Docker
apt-get install apt-transport-https ca-certificates curl software-properties-common

# Install Docker
apt-get install docker-ce
```

As you can see we set up two applications: GitLab Runner and Docker. We will keep things very simple with Docker: it is a verypowerful tool, but for the moment we will just use it as a super installer, for example we will not set up any tool, compiler or SDK on this machine. Instead, we will be lazy and let Docker download the right images for us. Things will become clearer later in this tutorial when we will configure our CI/CD pipeline.

Connect to the runner with GitLab:

1. Open GitLab on another web browser tab (the URL must be like https://gitlab.my-sample-domain.xyz/).
2. Sign in if necessary.
3. Choose **Admin area** from the top (the wrench icon).
4. Choose **Runners** from the left.

The bottom of the page contains an URL and a token:

Go back to the web-terminal connected to the runner machine, and type:

```
gitlab-runner register
```

This tool needs several information to register the runner. Enter the following responses:

1. Enter the gitlab-ci coordinator URL (for example, https://gitlab.com): copy the URL from the GitLab page above (for example, https://gitlab.my-sample-domain.xyz/)
2. Enter the gitlab-ci token for this runner: copy the token from the GitLab page above (for example, gXppo8ZyDgqdFb1vPG-w)
3. Enter the gitlab-ci description for this runner: devops-simple-app-gitlab-runner
4. Enter the gitlab-ci tags for this runner (comma separated): (keep it empty)
5. Enter the executor: docker
6. Enter the default Docker image (for example, ruby:2.1): alpine:latest

After the tool gives you back the hand, you should be able to see this runner on the GitLab web browser tab. Refresh the page and check at the bottom, you should see something like this.

Our GitLab is now ready to be used! But there are few more points to consider before creating our first project:

Manage users

As administrator, you can follow these steps to improve your GitLab account:

1. Open GitLab in your web browser (the URL must be like https://gitlab.my-sample-domain.xyz/).
2. Click your avatar on the upper-right corner of the page and select **Settings**.
3. Correctly set the **Full name** and **Email** fields and click **Edit profile settings**.
4. Click **Account** from the left.
5. Change your username and click **Update username**, and then confirm it again when the prompt appears (this step improves security as attackers would have to guess your username in addition to your password).

You may also want to control who can register on your GitLab server (the default configuration allows anyone on the Internet to register):

1. Click **Admin area** from the top (the wrench icon).
2. Click **Settings** from the left.
3. Expand the **Sign-up restrictions** section.
4. Uncheck the **Sign-up enabled** field.
5. Click **Save changes**.

Now only administrators can create new users. This can be done by navigating to the **Overview > Users** in the **Admin area**.

Maintain the GitLab

Linux servers need to be upgraded from time to time: security patches must be installed as soon as possible and applications should be updated to their latest versions.

On Ubuntu instances, the following commands allow you to safely update your server:

```
apt-get update
apt-get upgrade
```

Other commands such as `apt-get dist-upgrade` or `do-release-upgrade` are less safe, especially the last one because it can update Ubuntu to a later LTS version that is not yet supported by Alibaba Cloud.

For more complex upgrade, it may be more practical to replace the ECS instance:

1. Create a backup of the existing GitLab data.
2. Create a new ECS instance and install GitLab.

   > **Note** The GitLab version on the new ECS instance must be the same as the old one, if not the backup-restore process fails.

3. Restore the backups to the new machine.
4. Check whether the new instance works.
5. Unbind the EIP from the old ECS instance and bind it to the new one.
6. Release the old ECS instance.

Security updates can be automatically installed thanks to `unattended-upgrades`. For each ECS instance (GitLab and its runner), open a terminal (using SSH or the web-terminal console) and enter the following commands:

```
# Install unattended-upgrades
apt-get install unattended-upgrades

# Check the default configuration is fine for you. Press CTRL+X to quit.
nano /etc/apt/apt.conf.d/50unattended-upgrades

# Enable automatic upgrades
dpkg-reconfigure --priority=low unattended-upgrades

# Edit the related configuration
nano /etc/apt/apt.conf.d/20auto-upgrades
```

The last configuration file can be modified and the result looks like this:

```
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Unattended-Upgrade "1";
APT::Periodic::Download-Upgradeable-Packages "1";
APT::Periodic::AutocleanInterval "7";
```

Save and quit by pressing **CTRL+X**. You can launch `unattended-upgrades` manually for testing:

```
unattended-upgrade -d
```

The logs of `unattended-upgrades` are printed in `/var/log/unattended-upgrades`.

More information about automatic update can be found here.

Upgrade the GitLab

The described architecture for GitLab is fine as long as the number of users is not too large. However, there are several solutions when things start to get slow:

- If pipeline jobs take too much time to run, maybe adding more runners or using ECS instances with higher specifications can help.
- If GitLab itself becomes slow, the simplest solution is to migrate it to an ECS instance of a higher instance type.

If a single GitLab instance becomes unavailable due to performance issues or high-availability requirements, the architecture can evolve into a distributed system involving the following cloud resources:

- Additional ECS instances.
- A server load balancer to distribute the load across ECS instances.
- A NAS to let multiple ECS instances share a common file storage system.
- An external database.

As you can see the complexity can quickly increase. Tools such as Packer (virtual machine image builder), Terraform (infrastructure as code software) or Chef / Puppet / Ansible / SaltStack(configuration management) can greatly help managing it: they require an initial investment but allow organizations to better manage their systems.

Another solution is to let other companies manage this complexity for you. There are many SaaS vendors such as GitLab.com or GitHub. Alibaba Cloud offers Codepipeline, but it is currently only available in China.

16.1.3. Continuous integration

Introduction

This topic introduces a simple Continuous Integration pipeline based on GitLab CI/CD. Although we keep it simple now, this pipeline will be extended in the next topic.

Simple application

This topic is based on a simple web application written on top of Spring Boot (for the backend) and React (for the frontend).

The application consists in a todo list where a user can add or remove items. The goal is to have a simple 3-tier architecture with enough features that allow us to explore important concepts:

- The file organization shows a way to combine backend and frontend code into a single module (to keep it simple).
- The backend is stateless, which means that it does not store any data (for example, no

shared variable in the code). Instead, the data is saved in a database. This architecture is particularly useful for horizontal scaling.

- Because a relational database is involved, this project demonstrates how to use Flyway to help to upgrade the schema when the application evolves.
- The build process involves Npm, Babel, Webpack and Maven to compile and package the application for production.
- Code quality is achieved thanks to SonarQube, a tool that can detect bugs in the code and help us to maintain the project over time.

GitLab project creation

Let's start by creating a project on GitLab:

1. Open GitLab in your web browser (the URL must be like https://gitlab.my-sample-domain.xyz/);
2. Click **New...** from the top (with a + icon) and select **New project**.
3. Fill the new form with the following information:
   - Project name = todolist
   - Project slug = todolist
   - Visibility Level = Private
4. Click **Create project**.

We now have a project but we cannot download it on our computer yet, for that we need to generate and register a SSH key:

1. In your GitLab web browser tab, click your avatar (top-right of the page) and select **Settings**.
2. Click **SSH Keys** from the left.
3. Open a terminal and type the following commands:

```
# Generate a SSH certificate (set the email address you set in your GitLab profile)
ssh-keygen -o -t rsa -C "john.doe@your-company.com" -b 4096

# Display the public key
cat ~/.ssh/id_rsa.pub
```

4. Copy the result of the `cat` command and paste in the **Key** field (in the GitLab web browser tab).
5. The **Title** field should be automatically filled with your email address. The page looks like this:
6. Click **Add key** to register your SSH key.

You can now configure git and clone the project on your computer. Enter the following commands in your terminal:

```
# Set your real name
git config --global user.name "John Doe"

# Set the same email address as the one you set in your GitLab profile
git config --global user.email "john.doe@your-company.com"

# Create a directory for your projects
mkdir ~/projects
cd ~/projects

# Clone the empty project on your computer (set your GitLab domain name and username)
git clone git@gitlab.my-sample-domain.xyz:johndoe/todolist.git

# Change directory and check the ".git" folder is present
cd todolist
ls -la
```

Copy all the files from the folder sample-app/version1/* of this tutorial into *~/projects/todolist*. You should have a directory with the following top files:

- .git: Folder containing information for git.
- .gitignore: List of files to ignore for Git.
- .gitlab-ci.yml: GitLab CI pipeline configuration (more information about this file later).
- package.json: Npm configuration for the frontend: it declares dependencies such as React, Babel and Webpack.
- webpack.config.js: Webpack configuration for the frontend: it contains information about how to transpile the JSX code into standard JavaScript supported by all modern web browsers. It also describes how to package the frontend code and place it into a folder where Spring Boot can pick it and serves it via HTTP.
- pom.xml: Maven configuration for the backend: it declares dependencies, how to compile the code, how to run the tests, and how to package the complete application.
- src: Source code of the application.

The src folder is organized like this:

- src/main/java: Backend code in Java. The entry-point is `com/alibaba/intl/todolist/Application.java`.
- src/main/js: Frontend code. The entry-point is `app.js`.
- src/main/resources/application.properties: Backend configuration (for example, database url).
- src/main/resources/static: Frontend code (HTML, CSS and JavaScript). The built folder is generated by Webpack.
- src/main/resources/db/migration: Database scripts for Flyway (more on this later).
- src/test/java: Backend tests.

- src/test/resources: Backend tests configuration.

Run the application locally

Install the JDK 8 and Maven on your computer, and build your application with the following command:

```
mvn clean package
```

This command should end with a **BUILD SUCCESS** message: it compiles and runs the tests and packages the application.

Note

- The application source code organization is based on this tutorial. You can read this document if you are interested in HATEOAS, WebSockets and Spring Security.
- Although the application needs a database, the tests pass because they use H2, an in-memory database.

The next step is to setup a database locally:

1. Download and install MySQL Community Server v5.7. Note that it will normally give you a temporary root password.
2. MySQL should have installed the MySQL Command-Line Tool. You may need to configure your PATH environment variable if the `mysql` command is not available on your terminal. On Mac OSX you can do the following:

   ```
   # Add the MySQL tools into the PATH variable
   echo 'export PATH=/usr/local/mysql/bin:$PATH' >> ~/.bash_profile


   # Reload .bash_profile
   . ~/.bash_profile
   ```

3. Launch MySQL on your computer and connect to it with your terminal:

   ```
   # Connect to the database (use the password you received during the installation)
   mysql -u root -p
   ```

4. The command above should display a prompt. You can now configure your database:

```
-- Change the root password if you never did it before on this database
ALTER USER 'root'@'localhost' IDENTIFIED BY 'YouNewRootPassword';

-- Create a database for our project
CREATE DATABASE todolist;

-- Create a user for our project and grant him the rights
CREATE USER 'todolist'@'localhost' IDENTIFIED BY 'P@ssw0rd';
GRANT ALL PRIVILEGES ON todolist.* TO 'todolist'@'localhost';

-- Exit
QUIT;
```

Now that we have a database up and running, we need to configure the application. Have a look at the backend configuration file "src/main/resources/application.properties" and check that the DB configuration corresponds to your installation:

```
spring.datasource.url=jdbc:mysql://localhost:3306/todolist?useSSL=false
spring.datasource.username=todolist
spring.datasource.password=P@ssw0rd
```

Note

- The `spring.datasource.url` property is in the format `jdbc:mysql://HOSTNAME:PORT/DATABASE_NAME?useSSL=false`.
- If you modified this file you need to re-run `mvn clean package`.

You can now launch the application locally with the following command:

```
mvn spring-boot:run
```

If everything went well, the application should print several lines of logs in the console. Look at the two last lines:

```
2018-11-02 13:56:18.139  INFO 87329 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2018-11-02 13:56:18.145  INFO 87329 --- [main] com.alibaba.intl.todolist.Application    : Started Application in 5.305 seconds (JVM running for 17.412)
```

Open a new tab in your web browser and open the URL `http://localhost:8080`. You should normally get something like this:

> **Note** You can add new tasks by filling a description and by clicking Add.

Congratulation if you managed to get the application up and running! The source code has been written with the IntelliJ IDEA IDE (the ultimate edition is necessary for frontend development, you can evaluate it for free for 30 days).

Before we move on and create our first CI pipeline, there is still an important point to talk about: we didn't create any table in the database, so how does the application work? Let's have a look at our database with a terminal:

```
# Connect to the database (use your new root password)
mysql -u root -p
```

The command above opens a prompt; please enter the following instructions:

```
-- Use our database
USE todolist;

-- Display the tables
SHOW TABLES;
```

The last command should display something like this:

```
+----------------------+
| Tables_in_todolist   |
+----------------------+
| flyway_schema_history |
| task                 |
+----------------------+
2 rows in set (0.00 sec)
```

Now we can understand why the application works: because the database schema has been created. The **task** table corresponds to the Java class `src/main/java/com/alibaba/intl/todolist/model/Task.java`. Let's study `flyway_schema_history`:

```
-- Look at the content of the flyway_schema_history table
SELECT * FROM flyway_schema_history;
```

The result should look like this:

```
+----------------+---------+-------------------+------+---------------------------+------------+--------------+-----
---------------+---------------+---------+
| installed_rank | version | description       | type | script                    | checksum   | installed_by | installed
_on       | execution_time | success |
+----------------+---------+-------------------+------+---------------------------+------------+--------------+-----
---------------+---------------+---------+
|              1 | 001     | Create task table | SQL  | V001__Create_task_table.sql | -947603613 | todolist     | 2018
-10-31 17:57:51 |            24 |       1 |
+----------------+---------+-------------------+------+---------------------------+------------+--------------+-----
---------------+---------------+---------+
1 row in set (0.00 sec)
```

The **flyway_schema_history** table has been created by Flyway, a tool that allows us to create and update our database schema. As you can see, the table contains the names of the scripts from *src/main/resources/db/migration* that have been successfully executed.

Working with Flyway requires us to follow this procedure:

1. During the development of the application, when we want to upgrade our database schema, we need to add a new script in the *src/main/resources/db/migration* folder with a higher prefix number (we cannot modify existing scripts).
2. When Flyway starts, it checks what are the scripts that have been already executed (thanks to the **flyway_schema_history** table), and run the new ones.

Flyway is automatically started when the applications starts, if you check the application logs, you can see that Spring calls Flyway during its initialization. For more information about this integration, please read the official documentation.

Commit and first CI pipeline

It is now time to save the project in the git repository. Please enter the following command in your terminal:

```
# Go to the project folder
cd ~/projects/todolist


# Check files to commit
git status
```

The last command should print something like this:

```
On branch master

No commits yet

Untracked files:
 (use "git add <file>..." to include in what will be committed)

 .gitignore
 .gitlab-ci.yml
 package.json
 pom.xml
 src/
 webpack.config.js
```

Add all these files and commit them:

```
# Add the files
git add .gitignore .gitlab-ci.yml package.json pom.xml src/ webpack.config.js

# Commit the files and write a comment
git commit -m "Initial commit."

# Push the commit to the GitLab server
git push origin master
```

Pushing your code to GitLab triggers something interesting:

1. Open GitLab in your web browser (the URL must be like https://gitlab.my-sample-domain.xyz/);
2. Click **Projects** from the top and select **Your projects**.
3. Click the **todolist** project to see your files.
4. Click **CI / CD** from the left and select **Pipelines**.

You should see something like this:

Clicking **Artifacts** on the left allows you to download the generated **.jar** file containing your ready-for-production application.

Clicking the icon in the **Stages** column and then selecting **build** allows you to see the commands and logs used to compile and package the application.

This pipeline is triggered when somebody pushes code to the server. It is configured by the **.gitlab-ci.yml** file:

```
image: maven:3.6.0-jdk-8

variables:
 MAVEN_OPTS: "-Dmaven.repo.local=./.m2/repository"

cache:
 paths:
  - ./.m2/repository

stages:
 - build

build:
 stage: build
 script: "mvn package"
 artifacts:
  paths:
   - target/*.jar
```

The first line `image: maven:3.6.0-jdk-8` defines the Docker image used to execute the build command (as you can see, using Docker relieves us to setup the JDK 8 and Maven on the GitLab runner manually).

The `MAVEN_OPTS` variable and the `cache` block are an optimization: because Maven takes a lot of time to download dependencies, these definitions allow us to re-use these dependencies among pipelines.

The `stages` block defines only one stage `build`, we will add new ones later in this tutorial.

The `build` block is the most important one: it instructs the GitLab runner to execute `mvn package` in order to compile and run the tests and package the application. The `artifacts` block instructs GitLab to save the generated `.jar` file.

> **Note** Even if this pipeline is simple, it is already quite useful for a team since it can immediately inform the team that somebody committed something bad (for example he missed a file, or some test fail unexpectedly). GitLab automatically sends an email to the person who made the mistake: this rapid feedback can save us a lot of time because the error cause has a great chance to be located in the code that we just modified.

16.1.4. Code quality

Introduction

Before we continue on the way to deployment, it is important to add a stage in our pipeline to improve the code quality of our application. In this tutorial we are introducing SonarQube, a tool that can help us to find bugs before they arrive in production, and help us to manage the technical debt.

SonarQube infrastructure

Let's create an ECS instance with SonarQube:

1. Log on to the ECS console.
2. Click **Create Instance**.
3. If needed, select **Advanced Purchase** (also named **Custom**).
4. Fill the wizard with the following information:
   - Billing Method = Pay-As-You-Go
   - Region = the same region and availability zone as your GitLab server
   - Instance Type = filter by vCPU = 2, Memory = 4 GiB, Current Generation tab, and select a remaining type such as **ecs.n4.large**
   - Image = Ubuntu 18.04 64bit
   - System Disk = Ultra Disk 40 GiB
   - Network = VPC, select the same VPC and VSwitch as the GitLab server
   - Do NOT assign a public IP (we will create an EIP instead, which is more flexible)
   - Security Group = select the group **devops-simple-app-security-group**
   - Log on Credentials = select **Password** and choose one
   - Instance Name = devops-simple-app-sonar
   - Host = devops-simple-app-sonar
   - Read and accept the terms of service
5. Finish the instance creation by clicking **Create Instance**.
6. Go back to the ECS console, select **Instances** from the left-side navigation pane, and choose your region on top the screen. You can see your new instance.
7. Click **EIP** from the left-side navigation pane.
8. On the new page, click **Create EIP**.
9. Fill the wizard with the following information:
   - Region = the region where you have created you ECS
   - Max Bandwidth = 1 Mbps
   - Quantity = 1
10. Click **Buy Now**, select the agreement of service, and click **Activate**.
11. Go back to the EIP console and check your new EIP.
12. Next to you new EIP, click **Bind**.
13. In the new form, select:
    - Instance Type = ECS Instance
    - ECS Instance = devops-simple-app-sonar/i-generatedstring

14. Click **OK** to bind the EIP to you ECS instance.
15. Copy the IP Address of your EIP (it should be something like 47.74.253.23).

The ECS instance is ready, let's register a sub-domain for this machine:

1. Log on to the Domain console.
2. On the row corresponding to your domain (for example, **my-sample-domain.xyz**), click **Resolve**.
3. Click **Add Record**.
4. Fill the new form with the following information:
   - Type = A- IPV4 address
   - Host = sonar
   - ISP Line = Outside mainland China
   - Value = The EIP IP Address (for example 47.74.253.23)
   - TTL = 10 minute(s)
5. Click **OK** to add the record.

SonarQube requires a database, let's create a PostgreSQL RDS instance:

1. Log on to the ApsaraDB for RDS console.
2. Click **Create Instance**.
3. Fill the form with the following information:
   - Select **Pay-As-You-Go**
   - Region = the same as your ECS instance
   - DB Engine = PostgreSQL
   - Version = 9.4
   - Edition = High-availability
   - Zone = the same as your ECS instance
   - Network type = VPC, select the same VPC and availability zone as your ECS instance
   - Type = 2 cores, 4 GB (type rds.pg.s2.large)
   - Capacity = 20GB
   - Quantity = 1
4. Click **Buy Now**, accept the **Product Terms of Service** and **Service Level Notice and Terms of Use**, and click **Pay Now**.
5. Go back to the ApsaraDB for RDS console and wait for the RDS instance to start (it can take few minutes).
6. Set a name for your RDS instance by moving your mouse cursor over it and by clicking the pen icon. Set the name **devops-simple-app-sonar-rds** and confirm.
7. Click the instance ID.
8. Click **Set Whitelist** in the **Basic Information** > **Intranet Address** section.
9. Click **Add a Whitelist Group**.
10. Click **Upload ECS Intranet IP Address**.
11. In the **Whitelist** field, move your mouse cursor on top of the first IP address.

12. A bubble appears. Move your mouse cursor on top of the **Instance Name** bubble field.
13. If the instance name is devops-simple-app-sonar, select this IP address. If not, repeat on the next IP address.
14. After you have selected exactly one IP address, set the group name devops_simple_app_sonar_wlg.
15. Click **OK** to close the pop-up window.

> **Note** The whitelist is a security feature: only the ECS instances in this list can access the database.

Let's now create a database account and collect connection information:

1. Click **Accounts** from the left-side navigation pane.
2. Click **Create Initial Account**.
3. Fill the form with the following information:
   - Database Account = sonarqube
   - Password = YourS0narP@ssword
   - Re-enter Password = YourS0narP@ssword
4. Click **OK** to create the account.
5. Click **Connection Options** from the left-side navigation pane, and save the **Intranet Address** in the **Connection Information** section (it should be something like **rm-gs5wm687b2e3uc770.pgsql.singapore.rds.aliyuncs.com**).

SonarQube installation

We can now install SonarQube. Open a terminal and enter the following commands:

```
# Connect to the ECS instance
ssh root@sonar.my-sample-domain.xyz # Use the password you set when you have created the ECS in
stance

# Update the machine
apt-get update
apt-get upgrade

# Install tools
apt-get install unzip default-jdk postgresql-client

# Connect to the database (use the "Intranet Address" you saved in the paragraph above)
psql postgresql://rm-gs5wm687b2e3uc770.pgsql.singapore.rds.aliyuncs.com:3433/postgres -U sonarq
ube
```

The new command line allows you to configure the PostgreSQL database:

```
-- Create a database
CREATE DATABASE sonarqube;

-- Quit
\q
```

Back to Bash, continue the installation:

```
# Create a Linux user for SonarQube
adduser --system --no-create-home --group --disabled-login sonarqube

# Create directories where we will put SonarQube files
mkdir /opt/sonarqube
mkdir -p /var/sonarqube/data
mkdir -p /var/sonarqube/temp

# Download and unzip SonarQube (LTS version)
cd /opt/sonarqube
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-6.7.5.zip # URL from https://www.sonarqube.org/downloads/
unzip sonarqube-6.7.5.zip
rm sonarqube-6.7.5.zip

# Change the SonarQube file owner
chown -R sonarqube:sonarqube /opt/sonarqube
chown -R sonarqube:sonarqube /var/sonarqube

# Configure SonarQube
nano sonarqube-6.7.5/conf/sonar.properties
```

In the configuration file:

1. Scroll to **# User credentials**, uncomment and set the properties:
   - sonar.jdbc.username=sonarqube
   - sonar.jdbc.password=YourS0narP@ssword
2. Scroll to **#—- PostgreSQL 8.x or greater**, uncomment and set the property:
   - sonar.jdbc.url=jdbc:postgresql://rm-gs5wm687b2e3uc770.pgsql.singapore.rds.aliyuncs.com:3433/sonarqube # Set the **Intranet Address**
3. Scroll to **# WEB SERVER**, uncomment and set the property:
   - sonar.web.javaAdditionalOpts=-server

4. Scroll to **# OTHERS**, uncomment and set the properties:
   - sonar.path.data=/var/sonarqube/data
   - sonar.path.temp=/var/sonarqube/temp

Save and quit by pressing CTRL + X, then continue the installation:

```
# Create a service file for Systemd
nano /etc/systemd/system/sonarqube.service
```

Copy the following content in this new file (set the right path in “ExecStart” and “ExecStop”):

```
[Unit]
Description=SonarQube service
After=syslog.target network.target

[Service]
Type=forking

ExecStart=/opt/sonarqube/sonarqube-6.7.5/bin/linux-x86-64/sonar.sh start
ExecStop=/opt/sonarqube/sonarqube-6.7.5/bin/linux-x86-64/sonar.sh stop

User=sonarqube
Group=sonarqube
Restart=always

[Install]
WantedBy=multi-user.target
```

Save and quit by pressing CTRL+X. Back to Bash, continue the installation:

```
# Start SonarQube
systemctl start sonarqube.service

# Wait few seconds and check it worked (the text must finish with "SonarQube is up")
cat sonarqube-6.7.5/logs/sonar.log

# You can also check that the following command returns some HTML
curl http://localhost:9000

# Configure SonarQube to automatically start when the machine reboot
systemctl enable sonarqube.service
```

Now that SonarQube is started, we need to configure a reverse proxy to let users to connect to SonarQube via HTTPS. Enter the following commands in your terminal:

```
# Install Nginx and Let's Encrypt tooling
apt-get install software-properties-common
add-apt-repository ppa:certbot/certbot
apt-get update
apt-get install nginx python-certbot-nginx

# Configure Nginx to act as a reverse proxy for SonarQube
nano /etc/nginx/sites-available/sonarqube
```

Copy the following content in the new file (set the correct "server_name" according to your domain):

```
server {
    listen 80;
    server_name sonar.my-sample-domain.xyz;

    location / {
        proxy_pass http://127.0.0.1:9000;
    }
}
```

Back to Bash, continue the installation:

```
# Enable the new configuration file
ln -s /etc/nginx/sites-available/sonarqube /etc/nginx/sites-enabled/sonarqube

# Disable the default Nginx configuration
rm /etc/nginx/sites-enabled/default

# Check the configuration syntax
nginx -t

# Start Nginx
systemctl start nginx
```

To check if the installation is successful, open a new web browser tab to `http://sonar.my-sample-domain.xyz/` (adapt the URL for your domain). If everything went well, you should see something like this:

We now need to configure HTTPS. Enter the following commands in your terminal:

```
# Install the Let's Encrypt certificate (adapt for your domain)
certbot --nginx -d sonar.my-sample-domain.xyz
# Note: set your email address and accept the HTTP-to-HTTPS redirection

# The certificate will be automatically renewed. If you want, you can check the Cron configuration:
nano /etc/cron.d/certbot

# Check the renewal process with the following command
certbot renew --dry-run
# The logs should contain "Congratulations, all renewals succeeded" with your domain name (for exam
ple, sonar.my-sample-domain.xyz)

# Restart Nginx
systemctl restart nginx

# Configure Nginx to automatically start when the machine reboot
systemctl enable nginx
```

Refresh your web browser tab with SonarQube and check the URL: the protocol HTTPS must replace HTTP.

SonarQube configuration

We now need to change the administrator password:

1. Open your web browser tab with SonarQube (URL like https://sonar.my-sample-domain.xyz/).
2. Click **Log in** on the top-right of the page.
3. Fill the new form like this:
   - Login = admin
   - Password = admin
4. Click **Log in**.
5. Click your avatar on the top-right of the page and select **My Account**.
6. Click **Security**.
7. Change the password with the following values:
   - Old Password = admin
   - New Password = YourS0narQubeP@ssword
   - Confirm Password = YourS0narQubeP@ssword
8. Click **Change password**, and the message The **password has been changed!** is displayed.

Let's create a normal user:

1. Click **Administration** from the top.
2. Click **Security** in the top-sub-menu and select **Users**.
3. Click **Create User**.
4. Fill the new form like this (adapt the values):
   - Login = johndoe
   - Name = John Doe
   - Email = john.doe@your-company.com
   - Password = JohnDoeP@ssw0rd
5. Click **Create**.

Let's now force users to log in in order to work on SonarQube:

1. Click **Configuration** from the top-sub-menu and select **General Settings**.
2. Click **Security** from the left-side navigation pane.
3. Enable the switch in the **Force user authentication** property and confirm by clicking **Save**.

Now that user configuration is done, let's create our quality gate (the set of conditions to meet in order to let SonarQube to consider a code analysis as successful):

1. Click **Quality Gates** from the top.
2. Click **SonarQube way** on the left panel.
3. Click **Copy** on the top-right of the page.
4. Set the name **Stricter SonarQube way** and click **Copy**.
5. Add the following conditions (by clicking **Add Condition** widget below the existing conditions):
   - Metric = Coverage, Operator = is less than, Error = 70
   - Metric = Unit Test Errors, Operator = is not, Error = 0
   - Metric = Unit Test Failures, Operator = is not, Error = 0
   - Metric = Blocker Issues, Operator = is not, Error = 0
   - Metric = Critical Issues, Operator = is not, Error = 0
   - Metric = Major Issues, Operator = is not, Error = 0
6. Do not forget to click **Add** next to the conditions you just added.
7. Click **Set as Default** on the top-right of the page.

The quality gate should look like this:

SonarQube is now ready! Let's integrate it with our CI pipeline.

Code analysis pipeline stage

The first step is to obtain a token from SonarQube:

1. Open your web browser tab with SonarQube (URL like https://sonar.my-sample-domain.xyz/);
2. If you are still logged in as **admin**, log out by clicking your avatar on the top-right of the page and select **Log out**.

3. Login with your username and password (do not use the **admin** user).
4. Click your avatar on the top-right of the screen and select **My Account**.
5. Click **Security** from the top-sub-menu.
6. Next to **Generate New Token**, set the name **todolist** and click **Generate**.
7. You should see a new token appearing (something like **cfe2e3d7d7a15df20e3ecb7de53b6a23b3757474**).

> **Note** The following part of this section will modify two files: pom.xml and .gitlab-ci.yml. You can see the results by browsing in the sample-app/version2 folder.

The second step is to modify the pom.xml file by adding two Maven plugins:

- JaCoCo, used to analyze the code coverage of our unit tests.
- SonarQube, used to communicate with our SonarQube server.

JaCoCo is independent from SonarQube, it allows us to check which part of our code is covered by our tests. The following code contains the additions to our pom.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <!-- ... -->
  <properties>
    <!-- ... -->
    <jacoco-maven-plugin.version>0.8.2</jacoco-maven-plugin.version>
    <!-- ... -->
  </properties>
  <!-- ... -->
  <build>
    <plugins>
      <!-- ... -->
      <plugin>
        <groupId>org.jacoco</groupId>
        <artifactId>jacoco-maven-plugin</artifactId>
        <version>${jacoco-maven-plugin.version}</version>
        <configuration>
          <append>true</append>
        </configuration>
        <executions>
          <execution>
            <id>agent-for-ut</id>
            <goals>
              <goal>prepare-agent</goal>
            </goals>
          </execution>
          <execution>
            <id>jacoco-site</id>
            <phase>verify</phase>
            <goals>
              <goal>report</goal>
            </goals>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>
```

JaCoCo Maven plugin is executed during the **verify** phase, which happens between **package** and **install**. After its execution, this plugin generates several files:

- target/site/jacoco - A report containing coverage results in multiple formats (HTML, XML and CSV). You can check it by running `mvn clean install` from your project directory and by opening `target/site/jacoco/index.html` in your web browser.
- target/jacoco.exec - Data used to generate the HTML, XML and CSV reports.

The SonarQube Maven plugin reads the reports generated by JaCoCo and Surefire (the Maven plugin that runs our JUnit tests). The following code contains the addition into our pom.xml file for this plugin:

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <!-- ... -->
  <properties>
    <!-- ... -->
    <sonar-maven-plugin.version>3.5.0.1254</sonar-maven-plugin.version>
    <sonar.sources>src/main/java,src/main/js,src/main/resources</sonar.sources>
    <sonar.exclusions>src/main/resources/static/built/*</sonar.exclusions>
    <sonar.coverage.exclusions>src/main/js/**/*</sonar.coverage.exclusions>
    <!-- ... -->
  </properties>
  <!-- ... -->
  <build>
    <plugins>
      <!-- ... -->
      <plugin>
        <groupId>org.sonarsource.scanner.maven</groupId>
        <artifactId>sonar-maven-plugin</artifactId>
        <version>${sonar-maven-plugin.version}</version>
      </plugin>
    </plugins>
  </build>
</project>
```

This plugin is not automatically executed when running `mvn clean install`. You can run it **manually** with the following command:

```
mvn clean install sonar:sonar \
  -Dsonar.host.url=https://sonar.my-sample-domain.xyz \
  -Dsonar.login=cfe2e3d7d7a15df20e3ecb7de53b6a23b3757474 \
  -Dsonar.branch=master \
  -Dmaven.test.failure.ignore=true
```

As you can see, the plugin needs to be configured with the following properties:

- `sonar.host.url` must be set to your SonarQube server URL.
- `sonar.login` must contain the token that SonarQube generated for you.
- `sonar.branch` must contain the Git branch name. Please note that this feature is working but deprecated in SonarQube version 6.x (and removed in the version 7.x). It is now replaced by `sonar.branch.name` and `sonar.branch.target`. However, this functionality is not free anymore. You can read the official documentation if you are interested in purchasing the Developer Edition. A cheaper alternative for the versions 7.x is to set the `sonar.projectKey` property with a name that contains the branch name.
- `maven.test.failure.ignore` must be set to `true` to run the SonarQube analysis even when some tests fail.

The third step is to modify the .gitlab-ci.yml file with the following changes:

```
image: maven:3.6.0-jdk-8

variables:
 MAVEN_OPTS: "-Dmaven.repo.local=./.m2/repository"
 SONAR_URL: "https://your_sonarqube.url"
 SONAR_LOGIN: "token_generated_by_sonarqube"

cache:
 paths:
  - ./.m2/repository
  - ./.sonar/cache

stages:
 - build
 - quality

build:
 stage: build
 script: "mvn package -DskipTests=true"

quality:
 stage: quality
 script:
  - "mvn clean install sonar:sonar -Dsonar.host.url=$SONAR_URL -Dsonar.login=$SONAR_LOGIN -Dson
ar.branch=$CI_COMMIT_REF_NAME -Dmaven.test.failure.ignore=true -Duser.home=."
  - "wget https://github.com/gabrie-allaigre/sonar-gate-breaker/releases/download/1.0.1/sonar-gat
e-breaker-all-1.0.1.jar"
  - "java -jar sonar-gate-breaker-all-1.0.1.jar -u $SONAR_LOGIN"
 artifacts:
  paths:
   - target/*.jar
```

This file contains the following modifications:

- A new stage **quality** has been added under the `stages` block.
- The `build` block has been simplified: the unit test execution has been disabled thanks to the `-DskipTests=true` parameter, and the `artifacts` block has been removed.
- The new `quality` block contains 3 commands: the first one runs the unit tests and launch the SonarQube analysis, and the second and third ones wait for the analysis to complete and break the pipeline if there is a problem (for example, a unit test failed or the quality gate is not respected).

- Two `variables` have been added:
  - SONAR_URL will contain the URL to your SonarQube server.
  - SONAR_LOGIN will contain the generated SonarQube token.

Before committing these two files, we need to properly set the `SONAR_URL` and `SONAR_LOGIN` variables:

1. Open GitLab (the URL must be like https://gitlab.my-sample-domain.xyz/).
2. Sign in if necessary;
3. Click **Projects** in the top menu and select **Your projects**.
4. Click the **todolist** project.
5. In the left-side navigation pane, select **Settings > CI/CD**.
6. Expand the **Variables** panel, and create the following variables:
   - SONAR_URL = your SonarQube server URL (for example, https://sonar.my-sample-domain.xyz)
   - SONAR_LOGIN = your SonarQube token (for example, cfe2e3d7d7a15df20e3ecb7de53b6a23b3757474)
7. Click **Save variables**.

You can now commit the two modified files and let GitLab to run your new pipeline! Please execute the following commands in your terminal:

```
# Go to the project folder
cd ~/projects/todolist

# Check the files to commit
git status

# Add the files
git add .gitlab-ci.yml pom.xml

# Commit the files and write a comment
git commit -m "Add a quality stage in the pipeline."

# Push the commit to the GitLab server
git push origin master
```

Check your new GitLab pipeline: in your GitLab web browser tab, click **CI/CD** from the left-side navigation pane. You should get something like this:

As you can see, there is now two stages in the pipeline. You can click them to check detailed logs.

Have a look at your SonarQube server: open your web browser tab with SonarQube (URL like https://sonar.my-sample-domain.xyz/). You should see your project:

Click your project name. You should see something like this:

Explore this interface by yourself, for example click on the coverage percentage: you will get a list of Java files with their coverage percentage. If you click one of these files you can see which line is covered and which one is not.

> Note Code coverage is a good indicator before you attempt to execute a major code refactoring: like a safety net, a good code coverage means that you have a greater chance that your unit tests will catch bugs before they hit production.

CI pipeline testing

Let's break our pipeline!

Let's start with a unit test:

```
# Go to the project folder
cd ~/projects/todolist

# Open a test file
nano src/test/java/com/alibaba/intl/todolist/controllers/TaskControllerTest.java
```

At the line 87 of this file, change:

```
assertEquals("Task 2", createdTask2.getDescription());
```

Into:

```
assertEquals("Task 2222222222", createdTask2.getDescription());
```

Save by pressing CTRL + X, then commit the change:

```
# Check the files to commit
git status

# Add the file
git add src/test/java/com/alibaba/intl/todolist/controllers/TaskControllerTest.java

# Commit the file and write a comment
git commit -m "Break a unit test on purpose."

# Push the commit to the GitLab server
git push origin master
```

Have a look at your GitLab pipeline:

And your SonarQube project:

Let's now fix the test:

```
# Open the file to fix
nano src/test/java/com/alibaba/intl/todolist/controllers/TaskControllerTest.java
```

Restore the the line 87 (set **Task 2** instead of **Task 2222222222**), save with CTRL + X, and commit:

```
# Check the files to commit
git status

# Add the file
git add src/test/java/com/alibaba/intl/todolist/controllers/TaskControllerTest.java

# Commit the file and write a comment
git commit -m "Fix the unit test."

# Push the commit to the GitLab server
git push origin master
```

Your GitLab pipeline and SonarQube project should be successful.

Now let's break something else:

```
# Open another file to break
nano src/main/java/com/alibaba/intl/todolist/controllers/MachineController.java
```

Insert the following lines at the end of the class (line 71, before the last brace):

```
private String dummy = "example";

    public synchronized String getDummy() {
        return dummy;
    }

    public void setDummy(String dummy) {
        this.dummy = dummy;
    }
```

Save with CTRL + X and continue:

```
# Check the files to commit
git status

# Add the file
git add src/main/java/com/alibaba/intl/todolist/controllers/MachineController.java

# Commit the file and write a comment
git commit -m "Add a potential data race issue."

# Push the commit to the GitLab server
git push origin master
```

Have a look at your GitLab pipeline:

And your SonarQube project:

This time the problem comes from a bug inside the code.

> **Note** Thread-safety issues are usually quite hard to fix because the bugs are not easy to reproduce.

Let's fix the code:

```
# Open the file to fix
nano src/main/java/com/alibaba/intl/todolist/controllers/MachineController.java
```

Remove the added lines (starting from line 71), then save with CTRL + X and continue:

```
# Check the files to commit
git status

# Add the file
git add src/main/java/com/alibaba/intl/todolist/controllers/MachineController.java

# Commit the file and write a comment
git commit -m "Fix the potential data race issue."

# Push the commit to the GitLab server
git push origin master
```

The GitLab pipeline and SonarQube project should be green again.

16.1.5. Continuous delivery

Introduction

In this part we will finally deploy our application in the cloud!

We will create 3 environments:

- dev.my-sample-domain.xyz - The **development** environment with the latest features.
- pre-prod.my-sample-domain.xyz - The **pre-production** environment for testing.
- www.my-sample-domain.xyz - The **production** environment for all users.

The two first sections are quite theoretical as they deal with cloud infrastructure design and development workflow.

The next two sections introduces Terraform and Packer: as you can see in the previous parts of this tutorial, creating our environment for GitLab and SonarQube with the web console is quite slow. Since we will have to create 3 nearly identical environments, we will use Terraform and Packer to speed-up the process.

In the five last sections we will use Terraform, Packer and GitLab to create an highly-available architecture that will be automatically built and updated with a new pipeline stage.

Highly available architecture

The goal is to be able to serve our web application to users even in case of hardware or network failure.

The following diagram shows a simplified view of our architecture:

As you can see we are duplicating each cloud resource into two availability zones (zone A and zone B): since these zones are independents, a problem in one zone (for example, machine/network failure) can be compensated via the other one.

Our application will run on two ECS instances. The traffic from internet is redirected thanks to a server load balancer installed in front of them.

For the data storage layer we use ApsaraDB RDS for MySQL, a managed database service that handles server installation, maintenance, automatic backup, and so on.

> **Note** With this diagram you can understand why a stateless application is advantageous: the only place where data is shared is the database, we do not need to establish a direct link between the application servers. Moreover, if two users are modifying the same data (for example, by deleting the same item), the database will handle transactions for us, keep the data consistent and reject one user modification.

GitLab flow

Until now our development workflow was simple: modify some source code, commit it into the master branch and push it to GitLab. This is fine at the beginning (single developer, no deployment), but we need to enrich this process in order to properly manage our releases. For that GitLab Flow is a good solution: simple but rich enough for our needs.

The following diagram illustrates how we will use GitLab Flow in this tutorial:

The long horizontal arrows corresponds to long-lived branches (master, pre-production and production), a circle represents a commit, and the timeline goes from the left to the right (a commit on the left is older than a commit on the right).

The two short horizontal branches on the top correspond to short-lived branches: they are used to implement new features or bug fixes. In this example, two developers work on two features in parallel (**feature_a** and **feature_b**). When the **feature_a** is finished, the developer emits a merge request from his branch to the master. This is usually a good time for code review: another developer can check the modifications and accept/reject the request. If accepted, the feature branch is merged into the master and closed. In this example, the developer on the **feature_b** merges the new commit from the master branch corresponding to the **feature_a** to his own branch. He later can emit a merge request to merge his branch to the master.

On the right, the blocks correspond to environments (one environment contains an EIP, a server load balancer, two ECS instances and a RDS instance). Everytime a commit is done in the master branch, the CI/CD pipeline compiles, tests, analyzes the code, and build/update cloud resources with our application. The process is the same with the pre-production and production branches: it allows us to manage releases by emitting a merge request from the master branch to the pre-production one and from the pre-production one to the production one.

Infrastructure-as-code with Terraform

The problem with creating cloud resources with the web console is that it is quite tedious and error prone, especially when this process must be repeated for 3 environments. An elegant solution is to use Terraform: we write a script that describes our architecture (in the HCL language) and we ask Terraform to create / update our cloud environment accordingly.

Let's discover Terraform before using it for our project. Please install it on your computer (download the binary package and add it to your PATH variable), then open a terminal and run:

```
# Create a folder for our test
mkdir -p ~/projects/terraform-test

cd ~/projects/terraform-test

# Create a sample script
nano test.tf
```

Copy the following content into your script:

```
// Use Alibaba Cloud provider (https://github.com/terraform-providers/terraform-provider-alicloud)
provider "alicloud" {}

// Sample VPC
resource "alicloud_vpc" "sample_vpc" {
  name = "sample-vpc"
  cidr_block = "192.168.0.0/16"
}
```

Save and quit with CTRL + X, and execute:

```
# Download the latest stable version of the Alibaba Cloud provider
terraform init

# Configure the Alibaba Cloud provider
export ALICLOUD_ACCESS_KEY="your-accesskey-id"
export ALICLOUD_SECRET_KEY="your-accesskey-secret"
export ALICLOUD_REGION="your-region-id"

# Create the resources in the cloud
terraform apply
```

Note The values to set in `ALICLOUD_ACCESS_KEY` and `ALICLOUD_SECRET_KEY` are your access key ID and secret, you have already used them when you configured automatic backup for GitLab in Install and configure GitLab. For `ALICLOUD_REGION`, the available values can be found in .

The last command should print something like this:

```
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  + alicloud_vpc.sample_vpc
      id:             <computed>
      cidr_block:     "192.168.0.0/16"
      name:           "sample-vpc"
      route_table_id: <computed>
      router_id:      <computed>
      router_table_id: <computed>


Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value:
```

Terraform displays its plan of its modifications. As you can see only one resource will be added (the VPC). Enter the value `yes` and press ENTER. The result should be something like this:

```
alicloud_vpc.sample_vpc: Creating...
  cidr_block:     "" => "192.168.0.0/16"
  name:           "" => "sample-vpc"
  route_table_id: "" => "<computed>"
  router_id:      "" => "<computed>"
  router_table_id: "" => "<computed>"
alicloud_vpc.sample_vpc: Creation complete after 7s (ID: vpc-t4nhi7y0wpzkfr2auxc0p)

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Let's check the result:

1. Log on to the VPC console.
2. Select your region on top of the page.
3. Check the VPC table, and you can see **sample-vpc**:

A very interesting feature of Terraform is its idempotence. We can check that with the following command:

```
# Run Terraform again
terraform apply
```

Terraform interacts with the Alibaba Cloud APIs to check what are the existing resources, then compares them to our script and decides that no modification is needed. You can see it in the console logs:

```
alicloud_vpc.sample_vpc: Refreshing state... (ID: vpc-t4nhi7y0wpzkfr2auxc0p)


Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
```

Behind the scene Terraform creates two files **terraform.tfstate** and **terraform.tfstate.backup**. The first one contains all the resources information that have been created. These files are important and should usually be shared among the team (on an OSS bucket for example).

Another interesting feature is that Terraform is able to update an existing architecture. Let's check it by ourselves:

```
# Open our sample script
nano test.tf
```

Add the following **vswitch** block in order to obtain the following result:

```
// Use Alibaba Cloud provider (https://github.com/terraform-providers/terraform-provider-alicloud)
provider "alicloud" {}

// Sample VPC
resource "alicloud_vpc" "sample_vpc" {
  name = "sample-vpc"
  cidr_block = "192.168.0.0/16"
}

// Query Alibaba Cloud about the availability zones in the current region
data "alicloud_zones" "az" {
  network_type = "Vpc"
}

// Sample VSwitch
resource "alicloud_vswitch" "sample_vswitch" {
  name = "sample-vswitch"
  availability_zone = "${data.alicloud_zones.az.zones.0.id}"
  cidr_block = "192.168.1.0/24"
  vpc_id = "${alicloud_vpc.sample_vpc.id}"
}
```

As you can see, we can use placeholders like `${a.variable}` to refer the resources with each others. We can also use a data source to query some information from Alibaba Cloud.

Save and quit with CTRL + X, and run the following command:

```
# Update our cloud resources
terraform apply
```

This time Terraform understands that it does not need to re-create the VPC, only the VSwitch:

```
licloud_vpc.sample_vpc: Refreshing state... (ID: vpc-t4nhi7y0wpzkfr2auxc0p)
data.alicloud_zones.az: Refreshing state...

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  + alicloud_vswitch.sample_vswitch
      id:                <computed>
      availability_zone: "ap-southeast-1a"
      cidr_block:        "192.168.1.0/24"
      name:              "sample-vswitch"
      vpc_id:            "vpc-t4nhi7y0wpzkfr2auxc0p"


Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value:
```

Enter `yes` and press ENTER. The VSwitch should be created in few seconds:

```
alicloud_vswitch.sample_vswitch: Creating...
  availability_zone: "" => "ap-southeast-1a"
  cidr_block:        "" => "192.168.1.0/24"
  name:              "" => "sample-vswitch"
  vpc_id:            "" => "vpc-t4nhi7y0wpzkfr2auxc0p"
alicloud_vswitch.sample_vswitch: Creation complete after 7s (ID: vsw-t4nvtqld0ktk4kddxq709)

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Check it worked with the console:

1. Refresh your web browser tab with the VPC console.
2. If necessary, select your region on top of the page.
3. Click the ID of your VPC sample-vpc.

4. Scroll down and click 1 next to **VSwitch**.

You should be able to see your sample VSwitch:

Congratulation if you managed to get this far! For more information about available resources and datasources, please read the Alicloud provider documentation.

Let's release our cloud resources. With your terminal execute the following command:

```
# Release our cloud resources
terraform destroy
```

Terraform prints its plan as usual:

```
alicloud_vpc.sample_vpc: Refreshing state... (ID: vpc-t4nhi7y0wpzkfr2auxc0p)
data.alicloud_zones.az: Refreshing state...
alicloud_vswitch.sample_vswitch: Refreshing state... (ID: vsw-t4nvtqld0ktk4kddxq709)

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  - alicloud_vpc.sample_vpc

  - alicloud_vswitch.sample_vswitch


Plan: 0 to add, 0 to change, 2 to destroy.

Do you really want to destroy?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value:
```

Enter `yes` and press ENTER. The resources should be released in few seconds:

```
alicloud_vswitch.sample_vswitch: Destroying... (ID: vsw-t4nvtqld0ktk4kddxq709)
alicloud_vswitch.sample_vswitch: Destruction complete after 1s
alicloud_vpc.sample_vpc: Destroying... (ID: vpc-t4nhi7y0wpzkfr2auxc0p)
alicloud_vpc.sample_vpc: Destruction complete after 3s

Destroy complete! Resources: 2 destroyed.
```

As you can see, the fact that Terraform checks existing cloud resources and then compares them to our scripts allows us to create a new pipeline stage to run **terraform apply**: at the first execution cloud resources will be created, at the next executions Terraform will update them if needed.

We will commit the Terraform scripts in the same repository as the application source code. Like this, modifications in the application code will always be in sync with the infrastructure code.

However this approach has one drawback: like scripts that modifies database schemas, we need to make sure we do not break things and stay backward compatible, in case we need to rollback our application to an old version.

VM image generation with Packer

Packer is a tool made by the same company as the one who develops Terraform. It allows us to create an image containing our already-configured application. The goal is to be able to create an ECS instance with an image where everything is already configured (no need to login to the machine via SSH and install or execute applications). This solution is particularly handy for auto scaling.

Let's discover Packer before using it for our project. Please install it on your computer (download the binary package and add it to your PATH variable), then open a terminal and run:

```
alicloud_vswitch.sample_vswitch: Destroying... (ID: vsw-t4nvtqld0ktk4kddxq709)
alicloud_vswitch.sample_vswitch: Destruction complete after 1s
alicloud_vpc.sample_vpc: Destroying... (ID: vpc-t4nhi7y0wpzkfr2auxc0p)
alicloud_vpc.sample_vpc: Destruction complete after 3s

Destroy complete! Resources: 2 destroyed.
```

Copy the following content into your script:

```
{
  "variables": {
    "access_key": "{{env `ALICLOUD_ACCESS_KEY`}}",
    "secret_key": "{{env `ALICLOUD_SECRET_KEY`}}",
    "region_id": "{{env `ALICLOUD_REGION`}}",
    "source_image": "{{env `SOURCE_IMAGE`}}",
    "instance_type": "{{env `INSTANCE_TYPE`}}"
  },
```

```
"builders": [
  {
    "type": "alicloud-ecs",
    "access_key": "{{user `access_key`}}",
    "secret_key": "{{user `secret_key`}}",
    "region": "{{user `region_id`}}",
    "image_name": "sample-image",
    "image_description": "Sample image for testing Packer.",
    "image_version": "1.0",
    "source_image": "{{user `source_image`}}",
    "ssh_username": "root",
    "instance_type": "{{user `instance_type`}}",
    "io_optimized": "true",
    "internet_charge_type": "PayByTraffic",
    "image_force_delete": "true",
    "system_disk_mapping": {
      "disk_category": "cloud_ssd",
      "disk_size": 20
    }
  }
],
"provisioners": [
  {
    "type": "shell",
    "inline": [
      "export DEBIAN_FRONTEND=noninteractive",
      "apt-get -y update",
      "apt-get -y upgrade",
      "apt-get -y install nginx",
      "systemctl start nginx",
      "systemctl enable nginx",
      "sleep 10",
      "curl http://localhost"
    ],
    "pause_before": "30s"
  }
]
}
```

Save and quit with CTRL + X.

Before we can run this script we need to know the exact source image and instance type available in your region. Open a new web browser tab and follow these instructions:

1. Go to the OpenAPI Explorer.
2. If it is not already the case, select **ECS** from the left-side navigation pane, and then **DescribeInstanceTypes** service from the sub-menu.
3. Enter your region ID in the **RegionId** field (for example, ap-southeast-1).
4. Click **Submit Request** at the bottom.
5. If needed, this website will ask you to login with your Alibaba Cloud account.
6. The **Response Result** panel on the right should contain a tree of instance types; expand each instance type until you find one with **MemorySize** equals to 1 or more, and then save the value of its **InstanceTypeId** (for example, ecs.n1.small).
7. Select **DescribeImages** from the sub-menu.
8. Enter your region ID in the **RegionId** field (for example, ap-southeast-1).
9. Enter **ubuntu*64*** in the **ImageName** field.
10. Enter **system** in the **ImageOwnerAlias** field.
11. Click **Submit Request** at the bottom.
12. The **Response Result** panel should contain a tree of available images. Expand each image and save the value of the most recent **ImageId** (for example, ubuntu_18_04_64_20G_alibase_20181212.vhd).

Now that we have the **InstanceTypeId** and **ImageId**, go back to your terminal and type:

```
# Configure the Alibaba Cloud provider
export ALICLOUD_ACCESS_KEY="your-accesskey-id"
export ALICLOUD_SECRET_KEY="your-accesskey-secret"
export ALICLOUD_REGION="your-region-id"
export SOURCE_IMAGE="your-ImageId"
export INSTANCE_TYPE="your-InstanceTypeId"

# Create the image in the cloud
packer build test.json
```

Packer should output something like this:

```
alicloud-ecs output will be in this color.


==> alicloud-ecs: Force delete flag found, skipping prevalidating image name.
    alicloud-ecs: Found image ID: ubuntu_18_04_64_20G_alibase_20181212.vhd
==> alicloud-ecs: Creating temporary keypair: packer_5bea5aa2-e524-1af8-80d1-1db78347ed15
==> alicloud-ecs: Creating vpc
==> alicloud-ecs: Creating vswitch...
==> alicloud-ecs: Creating security groups...
==> alicloud-ecs: Creating instance.
```

```
==> alicloud-ecs: Allocating eip
==> alicloud-ecs: Allocated eip 47.74.178.35
    alicloud-ecs: Attach keypair packer_5bea5aa2-e524-1af8-80d1-1db78347ed15 to instance: i-t4nhcv8q
x069trkfgye6
==> alicloud-ecs: Starting instance: i-t4nhcv8qx069trkfgye6
==> alicloud-ecs: Using ssh communicator to connect: 47.74.178.35
==> alicloud-ecs: Waiting for SSH to become available...
==> alicloud-ecs: Connected to SSH!
==> alicloud-ecs: Pausing 30s before the next provisioner...
==> alicloud-ecs: Provisioning with shell script: /var/folders/v1/jvjz3zmn64q0j34yc9m9n4w00000gn/T/
packer-shell047404213
    alicloud-ecs: Get:1 http://mirrors.cloud.aliyuncs.com/ubuntu xenial InRelease [247 kB]
    alicloud-ecs: Get:2 http://mirrors.cloud.aliyuncs.com/ubuntu xenial-updates InRelease [109 kB]
[...]
    alicloud-ecs: 142 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
[...]
    alicloud-ecs: The following NEW packages will be installed:
    alicloud-ecs:   fontconfig-config fonts-dejavu-core libfontconfig1 libgd3 libvpx3 libxpm4
    alicloud-ecs:   libxslt1.1 nginx nginx-common nginx-core
    alicloud-ecs: 0 upgraded, 10 newly installed, 0 to remove and 4 not upgraded.
[...]
    alicloud-ecs: Executing /lib/systemd/systemd-sysv-install enable nginx
    alicloud-ecs:   % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
    alicloud-ecs:                                  Dload  Upload   Total   Spent    Left  Speed
    alicloud-ecs: 100   612  100   612    0     0  81415      0 --:--:-- --:--:-- --:--:-- 87428
    alicloud-ecs: <!DOCTYPE html>
    alicloud-ecs: <html>
    alicloud-ecs: <head>
    alicloud-ecs: <title>Welcome to nginx!</title>
    alicloud-ecs: <style>
    alicloud-ecs:     body {
    alicloud-ecs:         width: 35em;
    alicloud-ecs:         margin: 0 auto;
    alicloud-ecs:         font-family: Tahoma, Verdana, Arial, sans-serif;
    alicloud-ecs:     }
    alicloud-ecs: </style>
    alicloud-ecs: </head>
    alicloud-ecs: <body>
    alicloud-ecs: <h1>Welcome to nginx!</h1>
    alicloud-ecs: <p>If you see this page, the nginx web server is successfully installed and
    alicloud-ecs: working. Further configuration is required.</p>
```

```
    alicloud-ecs: working. Further configuration is required.</p>
    alicloud-ecs:
    alicloud-ecs: <p>For online documentation and support please refer to
    alicloud-ecs: <a href="http://nginx.org/">nginx.org</a>.<br/>
    alicloud-ecs: Commercial support is available at
    alicloud-ecs: <a href="http://nginx.com/">nginx.com</a>.</p>
    alicloud-ecs:
    alicloud-ecs: <p><em>Thank you for using nginx.</em></p>
    alicloud-ecs: </body>
    alicloud-ecs: </html>
==> alicloud-ecs: Stopping instance: i-t4nhcv8qx069trkfgye6
==> alicloud-ecs: Waiting instance stopped: i-t4nhcv8qx069trkfgye6
==> alicloud-ecs: Creating image: sample-image
    alicloud-ecs: Detach keypair packer_5bea5aa2-e524-1af8-80d1-1db78347ed15 from instance: i-t4nhc
v8qx069trkfgye6
==> alicloud-ecs: Cleaning up 'EIP'
==> alicloud-ecs: Cleaning up 'instance'
==> alicloud-ecs: Cleaning up 'security group'
==> alicloud-ecs: Cleaning up 'vSwitch'
==> alicloud-ecs: Cleaning up 'VPC'
==> alicloud-ecs: Deleting temporary keypair...
Build 'alicloud-ecs' finished.

==> Builds finished. The artifacts of successful builds are:
--> alicloud-ecs: Alicloud images were created:

ap-southeast-1: m-t4n938t1plplyl7akeor
```

The last line contains the ID of the image we have just created (here **m-t4n938t1plplyl7akeor**). Before we go further, let's study what Packer did with our script:

1. Create an ECS instance and necessary cloud resources (key pair, VPC, VSwitch, security group, and ENI).
2. Connect to the ECS instance using SSH.
3. Wait for 30 seconds (to make sure the VM is completely started).
4. Update the machine (`apt-get -y update` and `apt-get -y upgrade`).
5. Install Nginx (`apt-get -y install nginx`).
6. Start Nginx and configure SystemD to start it when the machine boots (`systemctl start nginx` and `systemctl enable nginx`).
7. Wait for 10 seconds (to make sure Nginx is started).
8. Test Nginx by sending a HTTP request to http://localhost (`curl http://localhost`).

9. Stop the ECS instance.
10. Create a normal snapshot of the system disk and convert it to an image.
11. Release all cloud resources (EIP, ECS, security group, VSwitch, VPC, and key pair).

You can check the newly created image using the console:

1. Log on to the ECS console.
2. Select **Images** from the left-side navigation pane.
3. If necessary, select your region on the top of the page.
4. You can see your new image:

5. If you want, you can test this image by clicking **Create Instance** on the left.
6. When you are done, you can delete this image by selecting its checkbox and by clicking **Delete** at the bottom of the page.

Health check web service

Before we start with infrastructure scripts, we first need to modify the application to add a **/health** REST service that just responds **OK**. It will be useful for the health check process performed by the server load balancer. Open a terminal and execute:

```
# Go to the project folder
cd ~/projects/todolist

# Create a REST controller
nano src/main/java/com/alibaba/intl/todolist/controllers/HealthController.java
```

Copy the following content into the new file:

```
package com.alibaba.intl.todolist.controllers;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * Inform other systems that the application is healthy.
 *
 * @author Alibaba Cloud
 */
@RestController
public class HealthController {

    @RequestMapping("/health")
    public String health() {
        return "OK";
    }
}
```

Save and quit by pressing CTRL + X, then create another file:

```
# Create the corresponding test
nano src/test/java/com/alibaba/intl/todolist/controllers/HealthControllerTest.java
```

Copy the following content into the new file:

```
package com.alibaba.intl.todolist.controllers;

import com.alibaba.intl.todolist.AbstractTest;
import org.junit.Before;
import org.junit.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.setup.MockMvcBuilders;
import org.springframework.web.context.WebApplicationContext;

import static org.junit.Assert.assertEquals;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

/**
 * Test the REST API behind "/health".
 *
 * @author Alibaba Cloud
 */
public class HealthControllerTest extends AbstractTest {

    @Autowired
    private WebApplicationContext wac;

    private MockMvc mockMvc;

    @Before
    public void setup() {
        mockMvc = MockMvcBuilders.webAppContextSetup(wac).build();
    }

    @Test
    public void testHealth() throws Exception {
        String response = mockMvc.perform(get("/health"))
                .andExpect(status().isOk())
                .andReturn().getResponse().getContentAsString();
        assertEquals("OK", response);
    }
}
```

Save and quit by pressing CTRL + X, and then continue:

```
# Compile and run the tests
mvn clean package
# Note: the resulting logs should contain "BUILD SUCCESS".

# Check the files to commit
git status

# Add the files
git add src/main/java/com/alibaba/intl/todolist/controllers/HealthController.java
git add src/test/java/com/alibaba/intl/todolist/controllers/HealthControllerTest.java

# Commit the files and write a comment
git commit -m "Add a /health REST controller."

# Push the commit to the GitLab server
git push origin master
```

Check that everything worked in your GitLab pipeline (the URL should be something like https://gitlab.my-sample-domain.xyz/marcplouhinec/todolist/pipelines) and in your SonarQube dashboard (the URL should be something like https://sonar.my-sample-domain.xyz/dashboard?id=com.alibaba.intl%3Atodo-list%3Amaster).

> Note The code modifications above are included in the sample-app/version3 folder.

Application infrastructure

In this section we will create Packer and Terraform scripts that will create the following cloud resources for one environment:

- 1 VPC
- 2 VSwitches (one per availability zone)
- 1 Security group
- 2 ECS instances (one per availability zone)
- 1 Multi-zone MySQL RDS
- 1 SLB instance
- 1 EIP

We will organize the scripts in 3 main groups:

- The basis group that setups VPC, VSwitches, Security group, EIP, SLB instance, and domain records.
- The application group that setups RDS, VM image, and ECS instances.
- The Let's Encrypt group responsible for obtaining and updating our SSL certificate.

> **Note** We will deal with the third group in the next topic.

In addition, we will commit our infrastructure scripts alongside the application source code. The logic behind this design choice is to make sure both code bases are synchronized.

Because the scripts are quite large, we will copy them from the sample-app/version3 folder. Open a terminal and execute the following commands:

```
# Go to the project folder
cd ~/projects/todolist

# Copy the scripts from this tutorials (adapt the path to where you copied this tutorial)
cp -R path/to/sample-app/version3/infrastructure .

# Check the content of this folder
ls -l infrastructure
ls -l infrastructure/10_webapp
```

As you can see the scripts are organized like this:

- 05_vpc_slb_eip_domain - **basis group** (setup VPC, VSwitches, and so on)
- 10_webapp - folder that contains the application group
  - 05_rds - setup the MySQL database
  - 10_image - build the VM image configured to connect to the MySQL database
  - 15_ecs - setup the ECS instances with the VM image

> **Note** The prefix xx_ in the folder names is just a way to have them sorted when displayed with the `ls` command.

Let's have a look at the files in the **05_vpc_slb_eip_domain** folder. The **variables.tf** file contains 3 entries:

```
variable "env" {
  description = "Environment (dev, pre-prod, prod)"
  default = "dev"
}

variable "domain_name" {
  description = "Domain name of the project."
  default = "my-sample-domain.xyz"
}

variable "sub_domain_name" {
  description = "Sub-domain name corresponding to the environment (dev, pre-prod, www)."
  default = "dev"
}
```

The description of each variable should be self-explanatory. We will pass the variable values when invoking the `terraform apply` command.

Let's check the **main.tf** file. The first part declares a VPC, one VSwitch per availability zone, and a security group that accepts incoming traffic from the port 8080 (the default port of our application):

```
// ...
resource "alicloud_vpc" "app_vpc" { /* ... */ }

// One VSwitch per availability zone
resource "alicloud_vswitch" "app_vswitch_zone_0" {
  availability_zone = "... Zone A ..."
  vpc_id = "${alicloud_vpc.app_vpc.id}"
  // ...
}
resource "alicloud_vswitch" "app_vswitch_zone_1" {
  availability_zone = "... Zone B ..."
  vpc_id = "${alicloud_vpc.app_vpc.id}"
  // ...
}

// Security group and rule
resource "alicloud_security_group" "app_security_group" {
  vpc_id = "${alicloud_vpc.app_vpc.id}"
  // ...
}
resource "alicloud_security_group_rule" "accept_8080_rule" {
  type = "ingress"
  ip_protocol = "tcp"
  nic_type = "intranet"
  policy = "accept"
  port_range = "8080/8080"
  priority = 1
  security_group_id = "${alicloud_security_group.app_security_group.id}"
  cidr_ip = "0.0.0.0/0"
}
```

The next part declares the server load balancer:

```
resource "alicloud_slb" "app_slb" {
  // ...
  vswitch_id = "${alicloud_vswitch.app_vswitch_zone_0.id}"
}

resource "alicloud_slb_listener" "app_slb_listener_http" {
  load_balancer_id = "${alicloud_slb.app_slb.id}"

  backend_port = 8080
  frontend_port = 80
  bandwidth = -1
  protocol = "http"

  health_check = "on"
  health_check_type = "http"
  health_check_connect_port = 8080
  health_check_uri = "/health"
  health_check_http_code = "http_2xx"
}
```

Note

- The SLB architecture is composed of a master and a slave. The `vswitch_id` corresponds to the availability zone where the master is located, the slave is automatically created in another zone. If the master fails, HTTP requests are transferred to the slave (within a delay of 30 sec). For more information about failover scenarios, see Scenarios.
- As you can see, the port redirection (80 to 8080) is defined in the SLB listener. You can also see how the SLB uses our Health check web service to determine whether a particular ECS instance is behaving normally or not.

The last part of the **main.tf** file declares an EIP, attaches it to our SLB and registers a DNS entry:

```
resource "alicloud_eip" "app_eip" { /* ... */ }

resource "alicloud_eip_association" "app_eip_association" {
  allocation_id = "${alicloud_eip.app_eip.id}"
  instance_id = "${alicloud_slb.app_slb.id}"
}

resource "alicloud_dns_record" "app_record_oversea" {
  name = "${var.domain_name}"
  type = "A"
  host_record = "${var.sub_domain_name}"
  routing = "oversea"
  value = "${alicloud_eip.app_eip.ip_address}"
  ttl = 600
}
resource "alicloud_dns_record" "app_record_default" {
  name = "${var.domain_name}"
  type = "A"
  host_record = "${var.sub_domain_name}"
  routing = "default"
  value = "${alicloud_eip.app_eip.ip_address}"
  ttl = 600
}
```

Let's build the **basis group** of our infrastructure. Open your terminal and run:

```
# Go to the 05_vpc_slb_eip_domain folder
cd ~/projects/todolist/infrastructure/05_vpc_slb_eip_domain

# Configure Terraform
export ALICLOUD_ACCESS_KEY="your-accesskey-id"
export ALICLOUD_SECRET_KEY="your-accesskey-secret"
export ALICLOUD_REGION="your-region-id"

# Initialize Terraform (download the latest version of the alicloud provider)
terraform init

# Create our infrastructure (note: adapt the domain_name according to your setup)
terraform apply  \
   -var 'env=dev' \
   -var 'domain_name=my-sample-domain.xyz' \
   -var 'sub_domain_name=dev'
```

You can check that your cloud resources have been successfully created by browsing the VPC console and by following links to related resources.

You can also check your new domain:

```
# Note: use your top domain name
nslookup dev.my-sample-domain.xyz
```

It should output something like this:

```
Server:  30.14.129.245
Address: 30.14.129.245#53

Non-authoritative answer:
Name: dev.my-sample-domain.xyz
Address: 161.117.2.245
```

The last IP address should be your EIP.

Let's study the **application group**, open **10_webapp/05_rds/variables.tf**:

```
variable "env" {
  description = "Environment (dev, pre-prod, prod)"
  default = "dev"
}

variable "db_account_password" {
  description = "MySQL database user password."
  default = "P@ssw0rd"
}
```

Note When creating the database, we set the database name to **todolist** and the user name to **todolist** as well. We only let the user password to be configurable (db_account_password variable).

Open 10_webapp/05_rds/main.tf:

```
// ...

resource "alicloud_db_instance" "app_rds" {
  // ...
  instance_type = "rds.mysql.t1.small"
  zone_id = "... Zone A + B ..."
  vswitch_id = "... ID of the VSwitch in zone A ..."
  security_ips = [
    "... VPC IP address range ..."
  ]
}

resource "alicloud_db_database" "app_rds_db" {
  instance_id = "${alicloud_db_instance.app_rds.id}"
  name = "todolist"
  character_set = "utf8"
}

resource "alicloud_db_account" "app_rds_db_account" {
  instance_id = "${alicloud_db_instance.app_rds.id}"
  name = "todolist"
  password = "${var.db_account_password}"
  type = "Normal"
}

resource "alicloud_db_account_privilege" "app_rds_db_account_privilege" {
  instance_id = "${alicloud_db_instance.app_rds.id}"
  account_name = "${alicloud_db_account.app_rds_db_account.name}"
  privilege = "ReadWrite"
  db_names = [
    "${alicloud_db_database.app_rds_db.name}"
  ]
}
```

Note Like with the SLB, the RDS database uses a master/slave architecture. The `zone_id` is set through a datasource (which provides a value like `ap-southeast-1MAZ1(a,b)`). The master is created in the availability zone of the given `vswitch_id`.

Let's create and configure the database. Execute the following instructions in your terminal:

```
# Go to the 10_webapp/05_rds folder
cd ../10_webapp/05_rds

# Initialize Terraform
terraform init

# Create the database
terraform apply  \
   -var 'env=dev' \
   -var 'db_account_password=YourD@tabasePassw0rd'

# Display the DB connection string
export RDS_CONNECTION_STRING=$(terraform output app_rds_connection_string)
echo $RDS_CONNECTION_STRING
```

The last command should print something like **rm-gs522kuv3u5m91256.mysql.singapore.rds.aliyuncs.com**. This value comes from the **output.tf** file:

```
output "app_rds_connection_string" {
  value = "${alicloud_db_instance.app_rds.connection_string}"
}
```

This is the hostname we will use in our ECS instances to connect them to the database.

The next sub-group **10_webapp/10_image** is a bit different: the Terraform scripts are only used to obtain information from Alibaba Cloud (the image ID of Ubuntu Linux and an ECS instance type). The **main.tf** file only contains datasources:

```
data "alicloud_images" "ubuntu_images" {
  owners = "system"
  name_regex = "ubuntu_18[a-zA-Z0-9_]+64"
  most_recent = true
}

data "alicloud_instance_types" "instance_types_zone_0" {
  cpu_core_count = 1
  memory_size = 2
  // ...
}
```

The **output.tf** file allows us to extract information from these datasources:

```
output "image_id" {
  value = "${data.alicloud_images.ubuntu_images.images.0.id}"
}

output "instance_type" {
  value = "${data.alicloud_instance_types.instance_types_zone_0.instance_types.0.id}"
}
```

The **app_image.json** file is a Packer script:

```
{
  "variables": {
    "access_key": "{{env `ALICLOUD_ACCESS_KEY`}}",
    "secret_key": "{{env `ALICLOUD_SECRET_KEY`}}",
    "region_id": "{{env `ALICLOUD_REGION`}}",
    "source_image": "{{env `SOURCE_IMAGE`}}",
    "image_version": "{{env `IMAGE_VERSION`}}",
    "instance_type": "{{env `INSTANCE_TYPE`}}",
    "application_path": "{{env `APPLICATION_PATH`}}",
    "properties_path": "{{env `PROPERTIES_PATH`}}",
    "environment": "{{env `ENVIRONMENT`}}",
    "rds_connection_string": "{{env `RDS_CONNECTION_STRING`}}",
    "rds_database": "{{env `RDS_DATABASE`}}",
    "rds_account": "{{env `RDS_ACCOUNT`}}",
    "rds_password": "{{env `RDS_PASSWORD`}}"
  },
  "builders": [
    {
      "type": "alicloud-ecs",
      "access_key": "{{user `access_key`}}",
      "secret_key": "{{user `secret_key`}}",
      "region": "{{user `region_id`}}",
      "image_name": "sample-app-image-{{user `environment`}}-{{user `image_version`}}",
      "image_description": "To-Do list web application ({{user `environment`}} environment).",
      "image_version": "{{user `image_version`}}",
      "source_image": "{{user `source_image`}}",
      "ssh_username": "root",
      "instance_type": "{{user `instance_type`}}",
      "io_optimized": "true",
      "internet_charge_type": "PayByTraffic",
      "image_force_delete": "true",
```

```
    "system_disk_mapping": {
     "disk_category": "cloud_ssd",
     "disk_size": 20
    }
   }
  ],
  "provisioners": [
   {
    "type": "shell",
    "inline": [
     "export DEBIAN_FRONTEND=noninteractive",
     "apt-get -y update",
     "apt-get -y upgrade",
     "apt-get -y install default-jdk",
     "mkdir -p /opt/todo-list",
     "mkdir -p /etc/todo-list"
    ],
    "pause_before": "30s"
   },
   {
    "type": "file",
    "source": "{{user `application_path`}}",
    "destination": "/opt/todo-list/todo-list.jar"
   },
   {
    "type": "file",
    "source": "{{user `properties_path`}}",
    "destination": "/etc/todo-list/application.properties"
   },
   {
    "type": "file",
    "source": "resources/todo-list.service",
    "destination": "/etc/systemd/system/todo-list.service"
   },
   {
    "type": "shell",
    "inline": [
     "export RDS_CONNECTION_STRING=\"{{user `rds_connection_string`}}\"",
     "export RDS_DATABASE=\"{{user `rds_database`}}\"",
     "export RDS_ACCOUNT=\"{{user `rds_account`}}\"",
     "export RDS_PASSWORD=\"{{user `rds_password`}}\""
```

```
      "export RDS_PASSWORD=\"{{user `rds_password`}}\"",
      "export DATASOURCE_URL=\"jdbc:mysql://$RDS_CONNECTION_STRING:3306/$RDS_DATABASE?useSSL=false\"",
      "export ESCAPED_DATASOURCE_URL=$(echo $DATASOURCE_URL | sed -e 's/\\\\/\\\\\\\\\\/g; s/\\//\\\\\\//g; s/&/\\\\\\&/g')",
      "export ESCAPED_RDS_ACCOUNT=$(echo $RDS_ACCOUNT | sed -e 's/\\\\/\\\\\\\\\\/g; s/\\//\\\\\\//g; s/&/\\\\\\&/g')",
      "export ESCAPED_RDS_PASSWORD=$(echo $RDS_PASSWORD | sed -e 's/\\\\/\\\\\\\\\\/g; s/\\//\\\\\\//g; s/&/\\\\\\&/g')",
      "sed -i \"s/\\(spring\\.datasource\\.url=\\).*\\$/\\1${ESCAPED_DATASOURCE_URL}/\" /etc/todo-list/application.properties",
      "sed -i \"s/\\(spring\\.datasource\\.username=\\).*\\$/\\1${ESCAPED_RDS_ACCOUNT}/\" /etc/todo-list/application.properties",
      "sed -i \"s/\\(spring\\.datasource\\.password=\\).*\\$/\\1${ESCAPED_RDS_PASSWORD}/\" /etc/todo-list/application.properties",
      "systemctl enable todo-list.service"
    ]
  }
 ]
}
```

This script creates a VM image by executing the following steps:

1. Create an ECS instance based on Ubuntu Linux.
2. Upgrade the existing packages.
3. Install Java JDK.
4. Copy our packaged application.
5. Copy our application configuration file (application.properties).
6. Copy a Systemd script (see the next paragraph for more info).
7. Set correct values in our application configuration file.
8. Enable our Systemd script in order to run our application automatically when the ECS instance starts.

The systemd script is located in the **resources** folder:

```
[Unit]
Description=todo-list
After=syslog.target
After=network.target

[Service]
ExecStart=/usr/bin/java -Xmx1800m -jar /opt/todo-list/todo-list.jar --spring.config.location=file:/etc/t
odo-list/application.properties
SuccessExitStatus=143
TimeoutStopSec=10
Restart=on-failure
RestartSec=5
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=todo-list

[Install]
WantedBy=multi-user.target
```

This script instructs Systemd about how to start the application, how to restart it automatically if it crashes, and where to print the logs (through syslog).

Let's create our VM image; in your terminal run:

```
# Go to the 10_webapp/10_image folder
cd ../10_image

# Initialize Terraform
terraform init

# Request some information for the next step
terraform apply -var 'env=dev'
export SOURCE_IMAGE=$(terraform output image_id)
export INSTANCE_TYPE=$(terraform output instance_type)

# Go to the application root folder and package it
cd ~/projects/todolist
mvn clean package -DskipTests=true
export APPLICATION_PATH=$(pwd)/$(ls target/*.jar)
export PROPERTIES_PATH=$(pwd)/src/main/resources/application.properties

# Go back to the 10_webapp/10_image folder
cd infrastructure/10_webapp/10_image

# Create the VM image
export IMAGE_VERSION=1
export ENVIRONMENT=dev
export RDS_DATABASE=todolist
export RDS_ACCOUNT=todolist
export RDS_PASSWORD="YourD@tabasePassw0rd"
packer build app_image.json
```

You can check the newly created image using the console:

1. Log on to the ECS console.
2. Select **Images** from the left-side navigation pane.
3. If necessary, select your region on the top of the page.
4. You should be able to see your new image named **sample-app-image-dev-1**.

Now comes the final step: to create ECS instances with our image and attach them to the SLB. Open the file **10_webapp/15_ecs/main.tf**:

```
// ...

// Our custom application image
data "alicloud_images" "app_images" {
  owners = "self"
  name_regex = "sample-app-image-${var.env}"
  most_recent = true
}

// ...

// One ECS instance per availability zone
resource "alicloud_instance" "app_ecs_zone_0" {
  // ...
  image_id = "${data.alicloud_images.app_images.images.0.id}"
  // ...
  vswitch_id = "... VSwitch in zone A ..."
  // ...
}
resource "alicloud_instance" "app_ecs_zone_1" {
  // ...
  image_id = "${data.alicloud_images.app_images.images.0.id}"
  // ...
  vswitch_id = "... VSwitch in zone B ..."
  // ...
}

// SLB attachments
resource "alicloud_slb_attachment" "app_slb_attachment" {
  load_balancer_id = "... SLB ID ..."
  instance_ids = [
    "${alicloud_instance.app_ecs_zone_0.id}",
    "${alicloud_instance.app_ecs_zone_1.id}"
  ]
}
```

Let's complete our infrastructure. Run the following instructions in your terminal:

```
# Go to the 10_webapp/15_ecs folder
cd ../15_ecs

# Initialize Terraform
terraform init

# Create the ECS instances and attach them to our SLB
terraform apply  \
   -var 'env=dev' \
   -var 'ecs_root_password=YourR00tP@ssword' \
   -parallelism=1
```

> **Note** As you can see, the last command set the parallelism parameter to one. This is necessary because we configured our application to update the database schema during its initialization (with Flyway). By creating one ECS instance at a time, we avoid potential data race issues (the first instance updates the schema, then the next one simply checks that nothing needs to be done).

Let's check the deployment of our application:

1. Log on to the SLB console.
2. Select your region if necessary.

Your new SLB looks like this:

Click the chevron icon next to **Default Server Group 2** and click the first ECS instance. You should see some information about this instance. The **Network (Internal)** graph is interesting:

The small waves are the result of the SLB health check (HTTP requests to the **/health** endpoint).

Let's play with the application! Open a new web browser tab and navigate to your domain (like http://dev.my-sample-domain.xyz/). You should obtain something like this:

Look at the top-right of the page: the hostname and instance ID allow you to know which ECS instance responded to your HTTP request. Refresh the page several times and look what happens:

As you can see, your HTTP requests are distributed among your two ECS instances.

> **Note** If you wish, you can enable Add an HTTP listener when configuring your SLB listener. That would allow a user to stick to the same ECS instance for all his HTTP requests, which is nice if you want to better exploit a local cache on your application server. However the disadvantage of this solution is that it might unbalance the load on your ECS instances. There are other solutions for caching, such as Memcached or Redis.

After you have finished to study your environment, you need to delete it (it will be the responsibility of the CI/CD pipeline to re-create and update it). Open a terminal and run:

```
# Go to the last sub-group folder
cd ~/projects/todolist/infrastructure/10_webapp/15_ecs/

# Configure Terraform
export ALICLOUD_ACCESS_KEY="your-accesskey-id"
export ALICLOUD_SECRET_KEY="your-accesskey-secret"
export ALICLOUD_REGION="your-region-id"

# Delete the ECS instances
terraform destroy

# Delete the database
cd ../05_rds/
terraform destroy

# Delete the vpc and other basis group resources
cd ../../05_vpc_slb_eip_domain
terraform destroy
```

Terraform state files management

Terraform generates tfstate files when we run the `terraform apply` command; they allow Terraform to keep track of existing cloud resources it has created during previous executions.

In the context of pipeline execution, it is crucial to store tfstate files into an external location, because local files are deleted when a pipeline job terminates. As a solution we will use the OSS bucket we have already created to store our GitLab backups.

The tfstate files are managed by Terraform backends. The default one is the local backend, its default configuration is to store tfstate files alongside our scripts. There are other types of backends but unfortunately none of them is directly compatible with OSS. One solution is to combine the local backend with OSSFS: we mount our OSS bucket as a local folder and save the tfstate files inside.

To implement this solution, we need to give the permissions to our Docker containers (the ones that run our pipeline jobs) to use FUSE, the underlying technology used by OSSFS:

1. Log on to the ECS console.
2. Click **Instance** from the left-side navigation pane.
3. Select your region if necessary.
4. Search for your instance named **devops-simple-app-gitlab-runner**.
5. Click **Connect** on the right side of your instance.
6. The VNC console should appear: copy the VNC password displayed in the popup and paste it

to the next one.

7. Authenticate yourself with the root user and the password you set when you configured GitLab.
8. Edit the GitLab Runner configuration file with this command:

```
nano /etc/gitlab-runner/config.toml
```

9. The configuration file should look like this:

```
concurrent = 1
check_interval = 0

[[session_server]]
  session_timeout = 1800

[[runners]]
  name = "devops-simple-app-gitlab-runner"
  url = "https://gitlab.my-sample-domain.xyz/"
  token = "8943412dd85a41002f9f803f21bdbf"
  executor = "docker"
  [runners.docker]
    tls_verify = false
    image = "alpine:latest"
    privileged = false
    disable_entrypoint_overwrite = false
    oom_kill_disable = false
    disable_cache = false
    volumes = ["/cache"]
    shm_size = 0
  [runners.cache]
    [runners.cache.s3]
    [runners.cache.gcs]
```

10. Change privileged = false to privileged = true. The file should now look like this:

```
concurrent = 1
 check_interval = 0

 [[session_server]]
  session_timeout = 1800

 [[runners]]
  name = "devops-simple-app-gitlab-runner"
  url = "https://gitlab.my-sample-domain.xyz/"
  token = "8943412dd85a41002f9f803f21bdbf"
  executor = "docker"
  [runners.docker]
   tls_verify = false
   image = "alpine:latest"
   privileged = true
   disable_entrypoint_overwrite = false
   oom_kill_disable = false
   disable_cache = false
   volumes = ["/cache"]
   shm_size = 0
  [runners.cache]
   [runners.cache.s3]
   [runners.cache.gcs]
```

11. Save and quit by pressing CTRL + X.
12. Restart the GitLab Runner using the following command:

```
gitlab-runner restart
```

13. Quit the VNC session by entering the command exit and by closing the web browser tab.

Delivery pipeline stage

We can now put everything together and integrate our infrastructure scripts into our CI/CD pipeline.

For that we need to open the **.gitlab-ci.yml** file and apply the following changes:

```
# ...
variables:
  # ...
  ALICLOUD_ACCESS_KEY: "your-accesskey-id"
  ALICLOUD_SECRET_KEY: "your-accesskey-secret"
  ALICLOUD_REGION: "your-region-id"
  GITLAB_BUCKET_NAME: "gitlab-my-sample-domain-xyz"
  GITLAB_BUCKET_ENDPOINT: "http://oss-ap-southeast-1-internal.aliyuncs.com"
  ECS_ENDPOINT: "ecs.aliyuncs.com"
  DOMAIN_NAME: "my-sample-domain.xyz"
  DB_ACCOUNT_PASSWORD: "your-db-password"
  ECS_ROOT_PASSWORD: "your-ecs-root-password"
  OSSFS_VERSION: "1.80.5"
  TERRAFORM_VERSION: "0.11.11"
  PACKER_VERSION: "1.3.3"
# ...
stages:
  - build
  - quality
  - deploy
# ...
deploy:
  stage: deploy
  image: ubuntu:16.04
  script:
    - "export ENV_NAME=$(./gitlab-ci-scripts/deploy/get_env_name_by_branch_name.sh $CI_COMMIT_RE
F_NAME)"
    - "export SUB_DOMAIN_NAME=$(./gitlab-ci-scripts/deploy/get_sub_domain_name_by_branch_name.s
h $CI_COMMIT_REF_NAME)"
    - "export BUCKET_LOCAL_PATH=/mnt/oss_bucket"
    - "./gitlab-ci-scripts/deploy/install_tools.sh"
    - "./gitlab-ci-scripts/deploy/mount_ossfs.sh"
    - "./gitlab-ci-scripts/deploy/build_basis_infra.sh"
    - "./gitlab-ci-scripts/deploy/build_webapp_infra.sh"
    - "umount $BUCKET_LOCAL_PATH"
    - "sleep 10"
  only:
    - master
    - pre-production
    - production
```

Note The complete version of this file can be found in "sample-app/version3/.gitlab-ci.yml".

As you can see, we have added a third stage named **deploy** after **build** and **quality**. The `only` property means that this stage is only executed for the branches **master**, **pre-production** and **production**. It means that a commit in a feature branch only triggers the **build** and **quality** stages.

In addition, because this stage is quite large, it has been split into multiple Bash scripts located in the folder **gitlab-ci-scripts/deploy**:

- get_env_name_by_branch_name.sh is quite simple: it gives the environment name (**dev**, **pre-prod** and **prod**) for the current branch (**master**, **pre-production** and **production**):

```
#!/usr/bin/env bash
#
# Print the environment name according to the branch name.
#
# Parameters:
#   - $1 = BRANCH_NAME
#

BRANCH_NAME=$1
ENV_NAME_MASTER="dev"
ENV_NAME_PRE_PRODUCTION="pre-prod"
ENV_NAME_PRODUCTION="prod"

if [[ ${BRANCH_NAME} == "production" ]]; then
   echo ${ENV_NAME_PRODUCTION};
elif [[ ${BRANCH_NAME} == "pre-production" ]]; then
   echo ${ENV_NAME_PRE_PRODUCTION};
else
   echo ${ENV_NAME_MASTER};
fi
```

- get_sub_domain_name_by_branch_name.sh is similar to **get_env_name_by_branch_name.sh**, but it gives the sub-domain name instead (**dev**, **pre-prod** and **www**):

```
#!/usr/bin/env bash
#
# Print the environment name according to the branch name.
#
# Parameters:
#   - $1 = BRANCH_NAME
#

BRANCH_NAME=$1
ENV_NAME_MASTER="dev"
ENV_NAME_PRE_PRODUCTION="pre-prod"
ENV_NAME_PRODUCTION="prod"

if [[ ${BRANCH_NAME} == "production" ]]; then
   echo ${ENV_NAME_PRODUCTION};
elif [[ ${BRANCH_NAME} == "pre-production" ]]; then
   echo ${ENV_NAME_PRE_PRODUCTION};
else
   echo ${ENV_NAME_MASTER};
fi
```

- install_tools.sh installs OSSFS, Terraform and Packer on top of the Ubuntu Docker image:

```
#!/usr/bin/env bash
#
# Install OSSFS, Terraform and Packer.
#
# Required global variables:
#   - OSSFS_VERSION
#   - TERRAFORM_VERSION
#   - PACKER_VERSION
#

echo "Installing OSSFS version ${OSSFS_VERSION}, Terraform version ${TERRAFORM_VERSION} and Pa
cker version ${PACKER_VERSION}..."

# Create a temporary folder
mkdir -p installation_tmp
cd installation_tmp

# Install OSSFS
apt-get -y update
```

```
apt-get -y update
apt-get -y install gdebi-core wget unzip
wget "https://github.com/aliyun/ossfs/releases/download/v${OSSFS_VERSION}/ossfs_${OSSFS_VERSION}_ubuntu16.04_amd64.deb"
gdebi -n "ossfs_${OSSFS_VERSION}_ubuntu16.04_amd64.deb"

# Install Terraform
wget "https://releases.hashicorp.com/terraform/${TERRAFORM_VERSION}/terraform_${TERRAFORM_VERSION}_linux_amd64.zip"
unzip "terraform_${TERRAFORM_VERSION}_linux_amd64.zip" -d /usr/local/bin/

# Install Packer
wget "https://releases.hashicorp.com/packer/${PACKER_VERSION}/packer_${PACKER_VERSION}_linux_amd64.zip"
unzip "packer_${PACKER_VERSION}_linux_amd64.zip" -d /usr/local/bin/

# Display the version of installed tools
echo "Installed OSSFS version:"
ossfs --version
echo "Installed Terraform version:"
terraform version
echo "Installed Packer version:"
packer version

# Delete the temporary folder
cd ..
rm -rf installation_tmp

echo "Installation of OSSFS, Terraform and Packer completed."
```

- mount_ossfs.sh makes our OSS bucket accessible like a normal folder:

```
#!/usr/bin/env bash
#
# Mount an OSS bucket with OSSFS.
#
# Required global variables:
#   - ALICLOUD_ACCESS_KEY
#   - ALICLOUD_SECRET_KEY
#   - GITLAB_BUCKET_NAME
#   - GITLAB_BUCKET_ENDPOINT
#   - BUCKET_LOCAL_PATH
#

echo "Mounting the OSS bucket ${GITLAB_BUCKET_NAME} (endpoint ${GITLAB_BUCKET_ENDPOINT}) in
to ${BUCKET_LOCAL_PATH}..."

# Configure OSSFS
echo "$GITLAB_BUCKET_NAME:$ALICLOUD_ACCESS_KEY:$ALICLOUD_SECRET_KEY" > /etc/passwd-oss
fs
chmod 640 /etc/passwd-ossfs

# Mount our bucket
mkdir -p "$BUCKET_LOCAL_PATH"
ossfs "$GITLAB_BUCKET_NAME" "$BUCKET_LOCAL_PATH" -ourl="$GITLAB_BUCKET_ENDPOINT"

echo "OSS bucket ${GITLAB_BUCKET_NAME} mounted with success into ${BUCKET_LOCAL_PATH}."
```

- build_basis_infra.sh runs the Terraform scripts to build the basis infrastructure:

```
#!/usr/bin/env bash
#
# Build the basis infrastructure (VPC, VSwitches, ...)
#
# Required global variables:
#   - ALICLOUD_ACCESS_KEY
#   - ALICLOUD_SECRET_KEY
#   - ALICLOUD_REGION
#   - ENV_NAME
#   - DOMAIN_NAME
#   - SUB_DOMAIN_NAME
#   - BUCKET_LOCAL_PATH
#

echo "Building the basis infrastructure (environment: ${ENV_NAME},\
 region: ${ALICLOUD_REGION},\
 domain: ${DOMAIN_NAME},\
 sub-domain: ${SUB_DOMAIN_NAME})..."

# Set values for Terraform variables
export TF_VAR_env=${ENV_NAME}
export TF_VAR_domain_name=${DOMAIN_NAME}
export TF_VAR_sub_domain_name=${SUB_DOMAIN_NAME}

# Run the Terraform scripts in 05_vpc_slb_eip_domain
cd infrastructure/05_vpc_slb_eip_domain
export BUCKET_DIR_PATH="$BUCKET_LOCAL_PATH/infrastructure/$ENV_NAME/05_vpc_slb_eip_domai
n"
mkdir -p ${BUCKET_DIR_PATH}
cp ${BUCKET_DIR_PATH}/*.tfstate* .
terraform init -input=false
terraform apply -input=false -auto-approve
rm -f ${BUCKET_DIR_PATH}/*
cp *.tfstate* ${BUCKET_DIR_PATH}

cd ../..

echo "Basis infrastructure successfully built (environment: ${ENV_NAME}, region: ${ALICLOUD_REGIO
N})."
```

> **Note**
>
> - You can see how we download and then replace tfstate files on our OSS bucket before and after running Terraform.
> - It is possible to directly configure the local backend to directly target the folder mounted by OSSFS (using the `terraform init` command). However there is a bug that corrupts tfstate files.

- build_webapp_infra.sh runs the Terraform scripts to build the application infrastructure:

```
#!/usr/bin/env bash
#
# Build the web application infrastructure (RDS, VM image, ECS, ...)
#
# Required global variables:
#   - ALICLOUD_ACCESS_KEY
#   - ALICLOUD_SECRET_KEY
#   - ALICLOUD_REGION
#   - ENV_NAME
#   - DB_ACCOUNT_PASSWORD
#   - ECS_ROOT_PASSWORD
#   - BUCKET_LOCAL_PATH
#   - CI_PIPELINE_IID
#

echo "Building the application infrastructure (environment: ${ENV_NAME}, region: ${ALICLOUD_REGIO
N})..."

# Set values for Terraform and Packer variables
export TF_VAR_env=${ENV_NAME}
export TF_VAR_db_account_password=${DB_ACCOUNT_PASSWORD}
export TF_VAR_ecs_root_password=${ECS_ROOT_PASSWORD}

export APPLICATION_PATH=$(pwd)/$(ls target/*.jar)
export PROPERTIES_PATH=$(pwd)/src/main/resources/application.properties
export IMAGE_VERSION=${CI_PIPELINE_IID}
export ENVIRONMENT=${ENV_NAME}
export RDS_DATABASE=todolist
export RDS_ACCOUNT=todolist
export RDS_PASSWORD=${DB_ACCOUNT_PASSWORD}

# Create/update the application database
```

```
cd infrastructure/10_webapp/05_rds
export BUCKET_DIR_PATH="$BUCKET_LOCAL_PATH/infrastructure/$ENV_NAME/10_webapp/05_rds"
mkdir -p ${BUCKET_DIR_PATH}
cp ${BUCKET_DIR_PATH}/*.tfstate* .
terraform init -input=false
terraform apply -input=false -auto-approve
rm -f ${BUCKET_DIR_PATH}/*
cp *.tfstate* ${BUCKET_DIR_PATH}
export RDS_CONNECTION_STRING=$(terraform output app_rds_connection_string)

# Extract Alibaba Cloud information for building the application image
cd ../10_image
export BUCKET_DIR_PATH="$BUCKET_LOCAL_PATH/infrastructure/$ENV_NAME/10_webapp/10_image
"
mkdir -p ${BUCKET_DIR_PATH}
cp ${BUCKET_DIR_PATH}/*.tfstate* .
terraform init -input=false
terraform apply -input=false -auto-approve
rm -f ${BUCKET_DIR_PATH}/*
cp *.tfstate* ${BUCKET_DIR_PATH}
export SOURCE_IMAGE=$(terraform output image_id)
export INSTANCE_TYPE=$(terraform output instance_type)

# Build the application image
packer build app_image.json

# Create/update the ECS instances
cd ../15_ecs
export BUCKET_DIR_PATH="$BUCKET_LOCAL_PATH/infrastructure/$ENV_NAME/10_webapp/15_ecs"
mkdir -p ${BUCKET_DIR_PATH}
cp ${BUCKET_DIR_PATH}/*.tfstate* .
terraform init -input=false
terraform apply -input=false -auto-approve -parallelism=1
rm -f ${BUCKET_DIR_PATH}/*
cp *.tfstate* ${BUCKET_DIR_PATH}

cd ../../..

echo "Application infrastructure successfully built (environment: ${ENV_NAME}, region: ${ALICLOUD_
REGION})."
```

> **Note** The `CI_PIPELINE_IID` variable is set by GitLab. It contains the pipeline number. We use it to create unique VM image names.

Before we commit these scripts, we first need to add new variables in our GitLab project configuration:

1. Open GitLab (the URL must be like https://gitlab.my-sample-domain.xyz/).
2. Sign in if necessary.
3. Click **Projects** from the top menu and select **Your projects**.
4. Click the **todolist** project.
5. In the left-side navigation pane, select **Settings** > **CI/CD**.
6. Expand the **Variables** panel, and create the following variables:
   - ALICLOUD_ACCESS_KEY = your Alibaba Cloud access key ID (for example, LTBIgF7wiMozeRIa)
   - ALICLOUD_SECRET_KEY = your Alibaba Cloud access key secret (for example, rdp59SYSKtNdDg3PYlTfjlrxu12fbp)
   - ALICLOUD_REGION = your Alibaba Cloud region (for example, ap-southeast-1)
   - GITLAB_BUCKET_NAME = your OSS bucket name (for example, gitlab-my-sample-domain-xyz)
   - GITLAB_BUCKET_ENDPOINT = the OSS bucket endpoint (for example, http://oss-ap-southeast-1-internal.aliyuncs.com). You can get it from the OSS console, by selecting your bucket and by copying the endpoint next to **VPC Network Access from ECS (Internal Network)**
   - ECS_ENDPOINT = the ECS Service endpoint. You can find the complete list in Request syntax. If you are unsure, set it to **ecs.aliyuncs.com**. This variable is used by Packer when it creates a VM image. Setting this variable to your region improves the performance and reduces the probability of timeout errors.
   - DOMAIN_NAME = your domain name (for example, my-sample-domain.xyz)
   - DB_ACCOUNT_PASSWORD = the password of the **todolist** user in the MySQL database (for example, YourSecretPassw0rdForRds)
   - ECS_ROOT_PASSWORD = the root password of your ECS instances (for example, YourSecretPassw0rdForEcs)
   - OSSFS_VERSION = latest OSSFS version (for example, 1.80.5)
   - TERRAFORM_VERSION = latest Terraform version (for example, 0.11.11)
   - PACKER_VERSION = latest Packer version (for example, 1.3.3)
7. Click **Save variables**.

Let's commit and push our changes. Open your terminal and type:

```
# Go to the project folder
cd ~/projects/todolist

# Check files to commit
git status

# Add the modified and new files
git add .gitignore infrastructure/ gitlab-ci-scripts/

# Commit and push to GitLab
git commit -m "Add the deploy stage."
git push origin master
```

In your GitLab web browser tab, select **CI/CD > Pipelines** from the left-side navigation pane. You should get something like this:

Check that your cloud resources have been successfully created by browsing to the VPC console and by following links to related resources.

Check your application by opening a new web browser tab and by navigating to your domain (like http://dev.my-sample-domain.xyz/).

Congratulation if you managed to deploy your application automatically! From now on, any commit on the master branch automatically launches a pipeline that builds, tests, analyzes and deploys the change!

Pre-production and production environments

As you can see in the scripts you have committed in the previous section, the two variables ENV_NAME and SUB_DOMAIN_NAME are set to different values according to the current branch name. Let's create new branches for the pre-production and production environments. Enter the following commands with your terminal:

```
# Go to the project folder
cd ~/projects/todolist

# Check files to commit
git status

# Add the modified and new files
git add .gitignore infrastructure/ gitlab-ci-scripts/

# Commit and push to GitLab
git commit -m "Add the deploy stage."
git push origin master
```

Check your GitLab CI/CD pipeline (**CI/CD** > **Pipelines** from the leftt-side navigation pane): the process should work successfully.

Check your cloud resources by browsing the VPC console: this time they should have names containing **pre-prod** instead of **dev**.

You can also check your web application with the **pre-prod** sub-domain (for example, http://pre-prod.my-sample-domain.xyz/). As you can see this new environment is completely isolated and independent from the development one.

Let's do the same with the production environment:

```
# Create a production branch
git checkout -b production
git push origin production
```

Check your GitLab CI/CD pipeline, then your cloud resources and finally your web application with the **www** sub-domain: http://www.my-sample-domain.xyz/.

Congratulation: you have 3 environments! From now on, the process to follow to deploy a new version of your application is the following:

1. Regularly commit improvements and new features into the master branch (through feature branches).
2. When the master branch is stable enough for a release, create a merge request from the master branch into the pre-production one:
3. Let another person to check and accept the merge request to start the deployment into pre-production.
4. Test the pre-production version, fix bugs and re-test. Note that it may be necessary to merge the master into the pre-production branch several times until the bugs are fixed.
5. When the pre-production branch is ready, create a merge request from the pre-production branch into the production one:
6. Let another person to check and accept the merge request to start the deployment into production.

16.2. HTTPS configuration

Introduction

HTTPS is a requirement for any professional website that needs to receive input from users, because it prevents man-in-the-middle and eavesdropping attacks.

There are several ways to configure HTTPS for our sample application, the easiest one is to buy an SSL/TLS certificate and Upload certificates. However, we will choose a more complex approach by using SSL/TLS certificates from Let's Encrypt.

Let's Encrypt is a certificate authority founded by organizations such as the Electronic Frontier Foundation, the Mozilla Foundation and Cisco Systems. It is provided free of charge and is 100% automated. Although it only provides domain-validated certificates (no organization validation or extended validation), but this is enough for most scenarios.

Architecture

HTTPS works by encrypting HTTP traffic through the TLS protocol. To configure HTTPS, we first need to obtain an SSL/TLS certificate and configure it on our SLB (by adding an HTTPS listener).

Once configured, the SLB handles the HTTPS complexities and continues to communicate with backend servers through unencrypted HTTP. Thus, a typical HTTPS request works like this:

1. A user opens an HTTPS connection with our web application;
2. The SLB uses its configured SSL/TLS certificate to establish a secured connection;
3. The user sends an HTTPS request;
4. The SLB converts the HTTPS request into an HTTP one (unencrypted) and sends it to one of the backend servers;
5. The backend server receives the HTTP request and sends back an HTTP response;
6. The SLB converts the HTTP response into an HTTPS one (encrypted) and sends it to the user.

Configuring an HTTPS listener for our SLB is relatively easy (we will add an alicloud_slb_server_certificate and a new alicloud_slb_listener in our Terraform script). Unfortunately obtaining an SSL/TLS certificate from Let's Encrypt requires us to modify our architecture:

Let's Encrypt needs a way to automatically check that our domain name belongs to us before providing us a certificate. For that we need to set up a program called certbot on our system, then execute this application so that it cans communicate with Let's Encrypt servers, run a challenge, and obtain the certificate. There are several types of challenges, we will use the HTTP-01 challenge and include it in the following process:

1. Create a new ECS instance named **certificate manager** and configure the SLB through an slb_rule so that every HTTP request with an URL that starts with **http://dev.my-sample-domain.xyz/.well-known/** is forwarded to this new ECS instance.
2. On this new ECS instance, install certbot and Nginx, and then configure the later to serve files from */var/www/html/certman/.well-known/* on the port 8080 (We keep this port to re-use the application security group. The SLB will redirect Internet traffic to this port).
3. Execute Certbot like this:

   ```
   certbot certonly --webroot -w /var/www/html/certman/.well-known/ -d dev.my-sample-domain.xyz
   ```

   This command runs the HTTP-01 challenge by executing the following steps:

   i. Certbot creates a file with a unique name in the folder **/var/www/html/certman/.well-known/acme-challenge/**, so that Nginx cans serve this file when the URL path is **/.well-known/acme-challenge/unique-name**.

   ii. Certbot contacts a Let's Encrypt server and asks it the to make an HTTP request to this file with the URL **http://dev.my-sample-domain.xyz/.well-known/acme-challenge/unique-name**.

iii. The Let's Encrypt server tries to download this file. If it succeeds it means that we indeed own the domain name so the challenge is passed with success.

Once the challenge is passed, the Let's Encrypt server generates an SSL/TLS certificate and sends it to certbot, which then stores it in the PEM format in the folder **/etc/letsencrypt/live/dev.my-sample-domain.xyz/**.

> **Note** Let's Encrypt has rate limits, so we should take care to run certbot only when necessary.

SLB configuration

Let's start by adding a listener to our SLB to let it manage HTTPS connections. For that we will generate a temporary self-signed certificate and update our Terraform script.

> **Note** The complete project files with the modifications of this tutorial part are available in the sample-app/version4 folder.

Open **gitlab-ci-scripts/deploy/build_basis_infra.sh** and insert the following block before `# Set values for Terraform variables` :

```
# Generate SSL/TLS certificate if it doesn't exist
export CERT_FOLDER_PATH=${BUCKET_LOCAL_PATH}/certificate/${ENV_NAME}/selfsigned
export CERT_PUBLIC_KEY_PATH=${CERT_FOLDER_PATH}/public.crt
export CERT_PRIVATE_KEY_PATH=${CERT_FOLDER_PATH}/private.key

mkdir -p ${CERT_FOLDER_PATH}
if [[ ! -f ${CERT_PUBLIC_KEY_PATH} ]]; then
  openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
    -keyout ${CERT_PRIVATE_KEY_PATH} \
    -out ${CERT_PUBLIC_KEY_PATH} \
    -subj "/C=CN/ST=Zhejiang/L=Hangzhou/O=Alibaba Cloud/OU=Project Delivery/CN=${SUB_DOMAIN_NAME}.${DOMAIN_NAME}"
fi
```

As you can see, we use OpenSSL to generate the certificate that we store on the OSS bucket. The certificate is composed of a public key **public.crt** and a private key **private.key**. They are all in the PEM format.

We then create two new Terraform variables at the end of **infrastructure/05_vpc_slb_eip_domain/variables.tf**:

```
variable "certificate_public_key_path" {
  description = "Path to the public key of the SSL/TLS certificate."
}

variable "certificate_private_key_path" {
  description = "Path to the private key of the SSL/TLS certificate."
}
```

Then we modify the script **gitlab-ci-scripts/deploy/build_basis_infra.sh** again by adding the following lines under `export TF_VAR_sub_domain_name=${SUB_DOMAIN_NAME}` :

```
export TF_VAR_certificate_public_key_path=${CERT_PUBLIC_KEY_PATH}
export TF_VAR_certificate_private_key_path=${CERT_PRIVATE_KEY_PATH}
```

Finally, we add the resources `alicloud_slb_server_certificate` and `alicloud_slb_listener` into **infrastructure/05_vpc_slb_eip_domain/main.tf:**

```
// ...
// Server load balancer
resource "alicloud_slb" "app_slb" {
  // ...
}

// SLB server certificate
resource "alicloud_slb_server_certificate" "app_slb_certificate" {
  name = "sample-app-slb-certificate-self-${var.env}"
  server_certificate = "${file(var.certificate_public_key_path)}"
  private_key = "${file(var.certificate_private_key_path)}"
}

// SLB listeners
resource "alicloud_slb_listener" "app_slb_listener_http" {
  // ...
}
resource "alicloud_slb_listener" "app_slb_listener_https" {
  load_balancer_id = "${alicloud_slb.app_slb.id}"

  backend_port = 8080
  frontend_port = 443
  bandwidth = -1
  protocol = "https"
  ssl_certificate_id = "${alicloud_slb_server_certificate.app_slb_certificate.id}"
  tls_cipher_policy = "tls_cipher_policy_1_0"

  health_check = "on"
  health_check_type = "http"
  health_check_connect_port = 8080
  health_check_uri = "/health"
  health_check_http_code = "http_2xx"
}

// EIP
// ...
```

> **Note**
>
> - An SLB can manage two types of certificate resources: server certificates and CA certificates. We only deal with the first type (the second type can be used to authenticate users with client certificates).
> - The HTTPS listener is very similar to the HTTP one. The main differences are the frontend port (443 instead of 80) and the presence of the `ssl_certificate_id`.

Commit and push these changes to GitLab:

```
# Go to the project folder
cd ~/projects/todolist

# Check files to commit
git status

# Add the modified and new files
git add gitlab-ci-scripts/deploy/build_basis_infra.sh
git add infrastructure/05_vpc_slb_eip_domain/variables.tf
git add infrastructure/05_vpc_slb_eip_domain/main.tf

# Commit and push to GitLab
git commit -m "Add a SLB HTTPS listener."
git push origin master
```

Check your CI/CD pipeline on GitLab, in particularly the logs of the **deploy** stage and make sure there is no error.

You can then test the results from your computer with the following command:

```
# Check that the SLB is configured to accept HTTPS requests
curl https://dev.my-sample-domain.xyz/
```

The `curl` command should fail with the following error:

```
curl: (60) SSL certificate problem: self signed certificate
More details here: https://curl.haxx.se/docs/sslcerts.html

curl performs SSL certificate verification by default, using a "bundle"
 of Certificate Authority (CA) public keys (CA certs). If the default
 bundle file isn't adequate, you can specify an alternate file
 using the --cacert option.
If this HTTPS server uses a certificate signed by a CA represented in
 the bundle, the certificate verification probably failed due to a
 problem with the certificate (it might be expired, or the name might
 not match the domain name in the URL).
If you'd like to turn off curl's verification of the certificate, use
 the -k (or --insecure) option.
HTTPS-proxy has similar options --proxy-cacert and --proxy-insecure.
```

Which is normal because self-signed certificates are considered insecure (a hacker performing a man-in-the-middle attack can generate its own self-signed certificate), but it validates that our SLB listener is configured for HTTPS.

> **Note** We can force curl to accept our self-signed certificate with the following command:
>
> ```
> # Force curl to accept our self-signed certificate
> curl -k https://dev.my-sample-domain.xyz/
> ```

The `curl` command should output something like this:

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8">
   <title>To-Do list</title>
   <link rel="stylesheet" href="css/index.css" media="screen">
</head>
<body>
<div id="react"></div>
<script src="built/bundle.js"></script>
</body>
</html>
```

VM image

Let's set up our Certificate Manager so that we can get a proper certificate from Let's Encrypt.

The first step is to create the VM image for the ECS instance that will manage our certificate. Open a terminal and execute the following instructions:

```
# Go to the project folder
cd ~/projects/todolist

# Create the folder that will contain the new scripts
mkdir -p infrastructure/15_certman/05_image
cd infrastructure/15_certman/05_image

# Copy the Terraform scripts that allow us to extract info about our Alibaba Cloud region
cp ../../10_webapp/10_image/variables.tf .
cp ../../10_webapp/10_image/main.tf .
cp ../../10_webapp/10_image/output.tf .

# Create a resources folder for scripts we want to include into the image
mkdir resources
```

The Certificate Manager needs the following configuration to obtain a certificate and update the SLB HTTPS listener configuration:

- Nginx must be installed and configured to serve files from the **/var/www/html/certman/.well-known/** folder.
- Nginx must also responds **OK** when the SLB health check system queries its **/health**.
- OSSFS must be installed and configured to allow the certificate to be stored on our OSS bucket (the goal is to avoid reaching rate limits of Let's Encrypt).
- Certbot must be installed and called through a Python script that will regularly check whether the current certificate is up to date, renew it when necessary and update the SLB HTTPS listener configuration.

The Packer script will become quite large so we will write it step by step. Let's start with Nginx installation and configuration. Enter the following command in your terminal:

```
# Create the packer script
nano certman_image.json
```

Enter the following content into the new file:

```
{
  "variables": {
    "access_key": "{{env `ALICLOUD_ACCESS_KEY`}}",
    "secret_key": "{{env `ALICLOUD_SECRET_KEY`}}",
    "region_id": "{{env `ALICLOUD_REGION`}}",
    "source_image": "{{env `SOURCE_IMAGE`}}",
```

```
  "image_version": "{{env `IMAGE_VERSION`}}",
  "instance_type": "{{env `INSTANCE_TYPE`}}",
  "environment": "{{env `ENVIRONMENT`}}"
 },
 "builders": [
  {
   "type": "alicloud-ecs",
   "access_key": "{{user `access_key`}}",
   "secret_key": "{{user `secret_key`}}",
   "region": "{{user `region_id`}}",
   "image_name": "sample-app-certman-image-{{user `environment`}}-{{user `image_version`}}",
   "image_description": "Certificate manager ({{user `environment`}} environment).",
   "image_version": "{{user `image_version`}}",
   "source_image": "{{user `source_image`}}",
   "ssh_username": "root",
   "instance_type": "{{user `instance_type`}}",
   "io_optimized": "true",
   "internet_charge_type": "PayByTraffic",
   "image_force_delete": "true",
   "system_disk_mapping": {
    "disk_category": "cloud_ssd",
    "disk_size": 20
   }
  }
 ],
 "provisioners": [
  {
   "type": "shell",
   "inline": [
    "export DEBIAN_FRONTEND=noninteractive",
    "apt-get -y update",
    "apt-get -y upgrade",
    "apt-get -y install nginx",
    "mkdir -p /var/www/html/certman/.well-known/acme-challenge",
    "echo \"OK\" > /var/www/html/certman/health",
    "echo \"It works!\" > /var/www/html/certman/.well-known/index.html",
    "echo \"It works!\" > /var/www/html/certman/.well-known/acme-challenge/index.html"
   ],
   "pause_before": "30s"
  },
  {
```

```
    {
      "type": "file",
      "source": "resources/nginx-conf-certman",
      "destination": "/etc/nginx/sites-available/certman"
    },
    {
      "type": "shell",
      "inline": [
        "ln -sf /etc/nginx/sites-available/certman /etc/nginx/sites-enabled/certman",
        "nginx -t",
        "systemctl enable nginx"
      ]
    }
  ]
}
```

This script executes the following actions:

- Update the default Ubuntu installation and install Nginx.
- Create the folders and files that will be used to respond to HTTP requests for **/health** and **/.well-known/**.
- Upload a Nginx configuration file (we will create it in a moment,).
- Activate the uploaded configuration and configure SystemD to automatically start Nginx when the machine starts.

Save and quit by pressing CTRL + X, and then create the Nginx configuration file:

```
# Create the Nginx configuration file
nano resources/nginx-conf-certman
```

Enter the following content into the new file:

```
server {
        listen 8080 default_server;
        listen [::]:8080 default_server;

        root /var/www/html/certman;

        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                try_files $uri $uri/ =404;
        }
}
```

The most interesting parts of this file are the listening port (8080, the same as our application in order to reuse our existing configuration) and the root folder **/var/www/html/certman** (were we have already created the **health** file and **.well-known** folder).

Save and quit by pressing CTRL + X. Now let's extend our Packer script:

```
# Edit the packer script
nano certman_image.json
```

Edit the content with the following changes:

```
{
  "variables": {
    // ...
    "ossfs_version": "{{env `OSSFS_VERSION`}}",
    "bucket_name": "{{env `BUCKET_NAME`}}",
    "bucket_endpoint": "{{env `BUCKET_ENDPOINT`}}"
  },
  // ...
  "provisioners": [
    // ...
    {
      "type": "file",
      "source": "resources/ossfs.service",
      "destination": "/etc/systemd/system/ossfs.service"
    },
    {
      "type": "shell"
```

```
    "type": "shell",
    "inline": [
      "export OSSFS_VERSION=\"{{user `ossfs_version`}}\"",
      "export ALICLOUD_ACCESS_KEY=\"{{user `access_key`}}\"",
      "export ALICLOUD_SECRET_KEY=\"{{user `secret_key`}}\"",
      "export BUCKET_NAME=\"{{user `bucket_name`}}\"",
      "export BUCKET_ENDPOINT=\"{{user `bucket_endpoint`}}\"",
      "export DEBIAN_FRONTEND=noninteractive",
      "apt-get -y install gdebi-core wget",
      "cd /tmp",
      "wget \"https://github.com/aliyun/ossfs/releases/download/v${OSSFS_VERSION}/ossfs_${OSSFS_VERSION}_ubuntu16.04_amd64.deb\"",
      "gdebi -n \"ossfs_${OSSFS_VERSION}_ubuntu16.04_amd64.deb\"",
      "echo \"$BUCKET_NAME:$ALICLOUD_ACCESS_KEY:$ALICLOUD_SECRET_KEY\" > /etc/passwd-ossfs",
      "chmod 640 /etc/passwd-ossfs",
      "mkdir -p /mnt/oss_bucket",
      "export ESCAPED_BUCKET_NAME=$(echo $BUCKET_NAME | sed -e 's/\\\\/\\\\\\\\/g; s/\\//\\\\\\//g; s/&/\\\\\\&/g')",
      "export ESCAPED_BUCKET_ENDPOINT=$(echo $BUCKET_ENDPOINT | sed -e 's/\\\\/\\\\\\\\/g; s/\\//\\\\\\//g; s/&/\\\\\\&/g')",
      "sed -i \"s/%BUCKET_NAME%/${ESCAPED_BUCKET_NAME}/\" /etc/systemd/system/ossfs.service",
      "sed -i \"s/%BUCKET_ENDPOINT%/${ESCAPED_BUCKET_ENDPOINT}/\" /etc/systemd/system/ossfs.service",
      "systemctl enable ossfs.service"
    ]
  }
 ]
}
```

This addition adds two provisioners that:

- Upload a SystemD file **ossfs.service**.
- Install OSSFS, configure it, and configure SystemD to start OSSFS when the machine boots.

Save and close by pressing CTRL + X, then create the SystemD file:

```
# Create the SystemD configuration file for OSSFS
nano resources/ossfs.service
```

Copy the following content to this new file:

```
[Unit]
Description=ossfs
After=syslog.target
After=network.target

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/local/bin/ossfs %BUCKET_NAME% /mnt/oss_bucket -ourl=%BUCKET_ENDPOINT%
ExecStop=/bin/umount /mnt/oss_bucket
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=ossfs

[Install]
WantedBy=multi-user.target
```

The most important part of this file is the `ExecStart` property: it mounts our OSS bucket in **/mnt/oss_bucket**.

> **Note** The `%BUCKET_NAME%` and `%BUCKET_ENDPOINT%` placeholders are replaced by using sedin the Packer script.

Save and close the file with CTRL + X. Let's continue with the certbot installation and our certificate update script:

```
# Edit the packer script
nano certman_image.json
```

Edit the content with the following changes:

```
{
 "variables": {
   // ...
   "domain": "{{env `DOMAIN_NAME`}}",
   "sub_domain": "{{env `SUB_DOMAIN_NAME`}}",
   "email_address": "{{env `EMAIL_ADDRESS`}}",
   "aliyun_python_sdk_core_version": "2.11.1",
   "aliyun_python_sdk_slb_version": "3.2.7"
 },
 // ...
 "provisioners": [
```

```
// ...
{
 "type": "shell",
 "inline": [
  "mkdir -p /etc/certificate-updater/",
  "mkdir -p /opt/certificate-updater/"
 ]
},
{
 "type": "file",
 "source": "resources/certificate-updater-config.ini",
 "destination": "/etc/certificate-updater/config.ini"
},
{
 "type": "file",
 "source": "resources/certificate-updater.py",
 "destination": "/opt/certificate-updater/certificate-updater.py"
},
{
 "type": "file",
 "source": "resources/certificate-manager.sh",
 "destination": "/opt/certificate-updater/certificate-updater.sh"
},
{
 "type": "file",
 "source": "resources/certificate-updater.service",
 "destination": "/etc/systemd/system/certificate-updater.service"
},
{
 "type": "file",
 "source": "resources/certificate-updater-cron",
 "destination": "/etc/cron.d/certificate-updater"
},
{
 "type": "shell",
 "inline": [
  "export DEBIAN_FRONTEND=noninteractive",
  "export ALICLOUD_ACCESS_KEY=\"{{user `access_key`}}\"",
  "export ALICLOUD_SECRET_KEY=\"{{user `secret_key`}}\"",
  "export ALICLOUD_REGION=\"{{user `region_id`}}\"",
  "export ENVIRONMENT=\"{{user `environment`}}\"",
```

```
      "export DOMAIN=\"{{user `domain`}}\"",
      "export SUB_DOMAIN=\"{{user `sub_domain`}}\"",
      "export EMAIL_ADDRESS=\"{{user `email_address`}}\"",
      "apt-get -y install software-properties-common",
      "add-apt-repository -y ppa:certbot/certbot",
      "apt-get -y update",
      "apt-get -y install python-certbot-nginx",
      "cd /opt/certificate-updater",
      "pip install pipenv --upgrade",
      "pipenv install aliyun-python-sdk-core==${ALIYUN_PYTHON_SDK_CORE_VERSION}",
      "pipenv install aliyun-python-sdk-slb==${ALIYUN_PYTHON_SDK_SLB_VERSION}",
      "pipenv install pyopenssl",
      "pipenv install pytz",
      "export ESCAPED_ACCESS_KEY=$(echo $ALICLOUD_ACCESS_KEY | sed -e 's/\\\\/\\\\\\\\/g; s/\\//\
\\\\\//g; s/&/\\\\\\&/g')",
      "export ESCAPED_SECRET_KEY=$(echo $ALICLOUD_SECRET_KEY | sed -e 's/\\\\/\\\\\\\\/g; s/\\//\\
\\\\\//g; s/&/\\\\\\&/g')",
      "export ESCAPED_REGION=$(echo $ALICLOUD_REGION | sed -e 's/\\\\/\\\\\\\\/g; s/\\//\\\\\\\//g; s
/&/\\\\\\&/g')",
      "export ESCAPED_ENVIRONMENT=$(echo $ENVIRONMENT | sed -e 's/\\\\/\\\\\\\\/g; s/\\//\\\\\\\//
g; s/&/\\\\\\&/g')",
      "export ESCAPED_DOMAIN=$(echo $DOMAIN | sed -e 's/\\\\/\\\\\\\\/g; s/\\//\\\\\\\//g; s/&/\\\\\\
&/g')",
      "export ESCAPED_SUB_DOMAIN=$(echo $SUB_DOMAIN | sed -e 's/\\\\/\\\\\\\\/g; s/\\//\\\\\\\//g;
s/&/\\\\\\&/g')",
      "export ESCAPED_EMAIL_ADDRESS=$(echo $EMAIL_ADDRESS | sed -e 's/\\\\/\\\\\\\\/g; s/\\//\\\\\
\//g; s/&/\\\\\\&/g')",
      "sed -i \"s/%access-key-id%/${ESCAPED_ACCESS_KEY}/\" /etc/certificate-updater/config.ini",
      "sed -i \"s/%access-key-secret%/${ESCAPED_SECRET_KEY}/\" /etc/certificate-updater/config.ini",
      "sed -i \"s/%region-id%/${ESCAPED_REGION}/\" /etc/certificate-updater/config.ini",
      "sed -i \"s/%environment%/${ESCAPED_ENVIRONMENT}/\" /etc/certificate-updater/config.ini",
      "sed -i \"s/%domain%/${ESCAPED_DOMAIN}/\" /etc/certificate-updater/config.ini",
      "sed -i \"s/%sub-domain%/${ESCAPED_SUB_DOMAIN}/\" /etc/certificate-updater/config.ini",
      "sed -i \"s/%email-address%/${ESCAPED_EMAIL_ADDRESS}/\" /etc/certificate-updater/config.ini",
      "chmod +x /opt/certificate-updater/certificate-updater.sh",
      "systemctl enable certificate-updater.service"
    ]
  }
 ]
}
```

As you can see, we are adding a new variable email_address that we will need to add in the GitLab configuration. It must contain an email address where we want to receive messages from Let's Encrypt when our certificate is going to expire.

We are also uploading many new files:

- certificate-updater.py - a script written in Python that checks whether the current certificate it up to date, renews it if necessary, and changes the SLB configuration.
- certificate-updater.sh - a Bash script that sets the correct working directory and then calls certificate-updater.py.
- certificate-updater-config.ini - the configuration file for **certificate-updater.py**.
- certificate-updater.service - a SystemD script to execute **certificate-updater.py** when the VM starts.
- certificate-updater-cron - a Cron script to run **certificate-updater.py** periodically.

The last provisioner installs certbot and libraries for our Python script, updates the Python script configuration, and configures SystemD to start OSSFS when the machine boots.

> **Note** You might have remarked that we install Python packages with Pipenv, but not with pip. The reason behind this decision is that we need to create a separate virtual environment for our script as a workaround: there are other Python scripts injected in each ECS instance called **cloud-init**. These cloud-init scripts set things such as the hostname, password, and so on. Unfortunately, the packages needed for our Python script are incompatible with the ones needed by the cloud-init scripts, so we need to separate environments.

Save with CTRL + X, then create the Python script configuration file:

```
# Create the configuration file for the Python script
nano resources/certificate-updater-config.ini
```

Put the following content into this file:

```
#
# Configuration file file for certificate-updater.
#
# This file must be located at /etc/certificate-updater/config.ini
#

[AlibabaCloud]
AccessKeyId : %access-key-id%
AccessKeySecret : %access-key-secret%
RegionId : %region-id%

[Environment]
# Environment (dev, pre-prod, prod)
Environment : %environment%

# Main domain name
Domain : %domain%

# Sub-domain name (dev, pre-prod, www)
SubDomain : %sub-domain%

# Email address of the person to warn when the certificate will expire soon
EmailAddress : %email-address%
```

The content is straight forward. The placeholders are replaced with `sed` in the Packer script. Save with CTRL + X and create the Python script:

```
# Create the Python script
nano resources/certificate-updater.py
```

Enter the following content into the file:

```
#!/usr/bin/env python
# coding=utf-8

import ConfigParser
import OpenSSL
import glob
import json
import os
import pytz
```

```
import shutil
import subprocess
from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.request import CommonRequest
from datetime import datetime
from datetime import timedelta

# Read the configuration
config = ConfigParser.ConfigParser()
config.read("/etc/certificate-updater/config.ini")

accessKeyId = config.get("AlibabaCloud", "AccessKeyId")
accessKeySecret = config.get("AlibabaCloud", "AccessKeySecret")
regionId = config.get("AlibabaCloud", "RegionId")
environment = config.get("Environment", "Environment")
domain = config.get("Environment", "Domain")
subDomain = config.get("Environment", "SubDomain")
emailAddress = config.get("Environment", "EmailAddress")

print("Certificate Updater started (environment: " + environment + ", " +
    "domain: " + domain + ", sub-domain: " + subDomain + ", email address: " + emailAddress + ")")

# Check if we need to run certbot
certFolderPath = "/mnt/oss_bucket/certificate/" + environment + "/letsencrypt"
publicKeyPath = certFolderPath + "/cert.pem"
privateKeyPath = certFolderPath + "/privkey.pem"
certbotCertFolderPath = "/etc/letsencrypt/live/" + subDomain + "." + domain

publicKeyExists = os.path.exists(publicKeyPath)
privateKeyExists = os.path.exists(privateKeyPath)
certExpireSoon = False

if publicKeyExists:
  publicKey = open(publicKeyPath, "rt").read()
  x509 = OpenSSL.crypto.load_certificate(OpenSSL.crypto.FILETYPE_PEM, publicKey)
  expirationDate = datetime.strptime(x509.get_notAfter(), "%Y%m%d%H%M%SZ").replace(tzinfo=pytz.
UTC)
  now = datetime.now(pytz.utc)
  certExpireSoon = now + timedelta(weeks=1) > expirationDate

runCertBot = not publicKeyExists or not privateKeyExists or certExpireSoon
```

```
certbotCronConfigured = not os.path.exists("/etc/cron.d/certbot")
print("Let's Encrypt certificate status:")
print("    publicKeyPath         = %s" % publicKeyPath)
print("    privateKeyPath        = %s" % privateKeyPath)
print("    publicKeyExists       = %s" % publicKeyExists)
print("    privateKeyExists      = %s" % privateKeyExists)
print("    certExpireSoon        = %s" % certExpireSoon)
print("    certbotCronConfigured   = %s" % certbotCronConfigured)
print("    runCertBot            = %s" % runCertBot)

# Run certbot if necessary
if runCertBot:
    print("Executing certbot...")
    returnCode = subprocess.call(
        "certbot certonly --webroot -w /var/www/html/certman/ -d \"%s.%s\" --non-interactive "
        "--agree-tos --email \"%s\"" % (subDomain, domain, emailAddress), shell=True)
    if returnCode != 0:
        print("Unable to run certbot, quitting...")
        quit(1)

    print("Replace the certificate on the OSS bucket...")
    if not os.path.exists(certFolderPath):
        os.makedirs(certFolderPath)
    for f in glob.glob(certFolderPath + "/*"):
        os.remove(f)
    for f in glob.glob(certbotCertFolderPath + "/*"):
        shutil.copy2(f, certFolderPath + "/")

# Check if the SLB certificate needs to be updated
print("Getting information about the SLB sample-app-slb-" + environment + "...")
client = AcsClient(accessKeyId, accessKeySecret, regionId)
request = CommonRequest()
request.set_accept_format("json")
request.set_domain("slb.aliyuncs.com")
request.set_method("POST")
request.set_version("2014-05-15")
request.set_action_name("DescribeLoadBalancers")
request.add_query_param("LoadBalancerName", "sample-app-slb-" + environment)
request.add_query_param("RegionId", regionId)
jsonResponse = client.do_action_with_exception(request)
```

```
jsonResponse = client.do_action_with_exception(request)
response = json.loads(jsonResponse)
if response["TotalCount"] != 1:
    print("Unable to find the SLB. Response:")
    print(response)
    quit(1)
slbInfo = response["LoadBalancers"]["LoadBalancer"][0]
slbId = slbInfo["LoadBalancerId"]

print("SLB found: %s. Loading HTTPS listener information..." % slbId)
request = CommonRequest()
request.set_accept_format("json")
request.set_domain("slb.aliyuncs.com")
request.set_method("POST")
request.set_version("2014-05-15")
request.set_action_name("DescribeLoadBalancerHTTPSListenerAttribute")
request.add_query_param("ListenerPort", "443")
request.add_query_param("LoadBalancerId", slbId)
jsonResponse = client.do_action_with_exception(request)
response = json.loads(jsonResponse)
if "ServerCertificateId" not in response:
    print("Unable to find the SLB HTTPS certificate. Response:")
    print(response)
    quit(1)
slbCertId = response["ServerCertificateId"]

print("SLB HTTPS listener information found. Loading information about the certificate " + slbCertId + "
...")
request = CommonRequest()
request.set_accept_format("json")
request.set_domain("slb.aliyuncs.com")
request.set_method("POST")
request.set_version("2014-05-15")
request.set_action_name("DescribeServerCertificates")
request.add_query_param("RegionId", regionId)
request.add_query_param("ServerCertificateId", slbCertId)
jsonResponse = client.do_action_with_exception(request)
response = json.loads(jsonResponse)
if not response["ServerCertificates"]["ServerCertificate"]:
    print("Unable to find the certificate " + slbCertId + ". Response:")
    print(response)
```

```
    quit(1)
slbCertInfo = response["ServerCertificates"]["ServerCertificate"][0]
slbCertFingerprint = slbCertInfo["Fingerprint"].upper()

# Compute the fingerprint of the current certificate from Let's Encrypt
print("Computing the Let's Encrypt certificate fingerprint...")
publicKey = open(publicKeyPath, "rt").read()
x509 = OpenSSL.crypto.load_certificate(OpenSSL.crypto.FILETYPE_PEM, publicKey)
certFingerprint = x509.digest("sha1")

# Check if the SLB listener certificate needs to be updated
updateListenerCert = slbCertFingerprint != certFingerprint
print("Certificates information:")
print("    slbCertFingerprint = %s" % slbCertFingerprint)
print("    certFingerprint    = %s" % certFingerprint)
print("    updateListenerCert = %s" % updateListenerCert)

if not updateListenerCert:
    print("SLB listener certificate is up to date.")
    quit(0)

# Upload the SLB listener certificate
now = datetime.now()
certName = "sample-app-slb-certificate-" + environment + "-" + now.strftime("%Y%m%d%H%M%S")
print("Upload the Let's Encrypt certificate " + certName + "...")
request = CommonRequest()
request.set_accept_format("json")
request.set_domain("slb.aliyuncs.com")
request.set_method("POST")
request.set_version("2014-05-15")
request.set_action_name("UploadServerCertificate")
privateKey = open(privateKeyPath, "rt").read()
privateKey = privateKey.replace("BEGIN PRIVATE", "BEGIN RSA PRIVATE")
privateKey = privateKey.replace("END PRIVATE", "END RSA PRIVATE")
request.add_query_param("ServerCertificate", publicKey)
request.add_query_param("PrivateKey", privateKey)
request.add_query_param("ServerCertificateName", certName)
jsonResponse = client.do_action_with_exception(request)
response = json.loads(jsonResponse)
if not response["ServerCertificateId"]:
    print("Unable to upload the certificate " + certName + ". Response:")
```

```
    print(response)
    quit(1)
certId = response["ServerCertificateId"]

# Update the HTTPS listener with the new certificate
print("Certificate " + certName + " (id: " + certId + ") uploaded with success. Updating the HTTP listener
...")
request = CommonRequest()
request.set_accept_format("json")
request.set_domain("slb.aliyuncs.com")
request.set_method("POST")
request.set_version("2014-05-15")
request.set_action_name("SetLoadBalancerHTTPSListenerAttribute")
request.add_query_param("ListenerPort", "443")
request.add_query_param("LoadBalancerId", slbId)
request.add_query_param("ServerCertificateId", certId)
jsonResponse = client.do_action_with_exception(request)
response = json.loads(jsonResponse)
if "Code" in response:
    print("Unable to update the SLB HTTPS certificate. Response:")
    print(response)
    quit(1)
print("SLB listener certificate updated with success.")
```

This script is quite long unfortunately, but it is easy to read. It executes the following operations:

- Read the configuration file.
- Check if a certificate already exists, and if yes, check its expiration date.
- Run certbot if the certificate does not exist or if it will be expired soon. The new certificate is stored into the OSS bucket.
- Get information about our SLB configuration thanks to Alibaba Cloud OpenAPI.
- Compare the SLB certificate fingerprint with the one we got with certbot, and stop the script here if they are equal.
- Upload the new certificate and update the SLB HTTPS listener configuration.

Save this file with CTRL + X and create the SystemD configuration file:

```
# Create the SystemD configuration file
nano resources/certificate-updater.service
```

Enter the following text into this file:

```
[Unit]
Description=certificate-updater
After=syslog.target
After=network.target
After=ossfs.service

[Service]
Type=simple
RemainAfterExit=yes
ExecStart=/usr/local/bin/pipenv run python2 /opt/certificate-updater/certificate-updater.py
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=certificate-updater
WorkingDirectory=/opt/certificate-updater

[Install]
WantedBy=multi-user.target
```

As you can see, this service will starts after we have mounted our OSS bucket. The `ExecStart` property run our script with Python 2.7.

Save this file with CTRL + X and create the Cron configuration file:

```
# Create the Cron configuration file
nano resources/certificate-updater-cron
```

Write the following content:

```
#
# Execute the certificate updater.
#
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

15 */12 * * * root systemd-cat -t "certificate-updater" /opt/certificate-updater/certificate-updater.sh
```

This file configures Cron to run **certificate-updater.sh** every day at 12h/15h AM/PM. The console output is sent to syslog with the systemd-cat command.

Save this file with CTRL + X. The file **certificate-updater.sh** does only 2 things: set the right working directory for Pipenv and invoke our Python script:

```
# Create the script that invokes the certificate updater
nano resources/certificate-manager.sh
```

Enter the following content:

```
#!/usr/bin/env bash
cd /opt/certificate-updater
/usr/local/bin/pipenv run python2 /opt/certificate-updater/certificate-updater.py
```

Save and quit by pressing CTRL + X.

Cloud resources

Now that we can generate an image, let's create the ECS instance and other related cloud resources.

Open your terminal and execute:

```
# Go to the project folder
cd ~/projects/todolist

# Create the folder that will contain the new scripts
mkdir -p infrastructure/15_certman/10_ecs_slb_rule
cd infrastructure/15_certman/10_ecs_slb_rule

# Declare the variables for Terraform
nano variables.tf
```

Put the following content into this new file:

```
variable "env" {
  description = "Environment (dev, pre-prod, prod)"
  default = "dev"
}

variable "ecs_root_password" {
  description = "ECS root password (simpler to configure than key pairs)"
  default = "YourR00tP@ssword"
}
```

Save and close with CTRL + X, and then continue with the main script:

```
# Create the main Terraform script
nano main.tf
```

The main script must contain the following code:

```
// Alibaba Cloud provider (source: https://github.com/terraform-providers/terraform-provider-alicloud)
provider "alicloud" {}

// Our custom certificate manager image
data "alicloud_images" "certman_images" {
  owners = "self"
  name_regex = "sample-app-certman-image-${var.env}"
  most_recent = true
}

// VSwitches in the first zone
data "alicloud_vswitches" "app_vswitches_zone_0" {
  name_regex = "sample-app-vswitch-zone-0-${var.env}"
}

// Security group
data "alicloud_security_groups" "app_security_groups" {
  name_regex = "sample-app-security-group-${var.env}"
}

// Load balancer
data "alicloud_slbs" "app_slbs" {
  name_regex = "sample-app-slb-${var.env}"
}

// Instance type with 1 vCPU, 0.5 GB or RAM
data "alicloud_instance_types" "instance_types_zone_0" {
  cpu_core_count = 1
  memory_size = 0.5
  availability_zone = "${data.alicloud_vswitches.app_vswitches_zone_0.vswitches.0.zone_id}"
  network_type = "Vpc"
}

// One ECS instance in the first availability zone
resource "alicloud_instance" "certman_ecs" {
  instance_name = "sample-app-certman-ecs-${var.env}"
  description = "Certificate manager (${var.env} environment)."
```

```
  host_name = "sample-app-certman-ecs-${var.env}"
  password = "${var.ecs_root_password}"

  image_id = "${data.alicloud_images.certman_images.images.0.id}"
  instance_type = "${data.alicloud_instance_types.instance_types_zone_0.instance_types.0.id}"

  internet_max_bandwidth_out = 1

  vswitch_id = "${data.alicloud_vswitches.app_vswitches_zone_0.vswitches.0.id}"
  security_groups = [
    "${data.alicloud_security_groups.app_security_groups.groups.0.id}"
  ]
}

// SLB VServer group
resource "alicloud_slb_server_group" "certman_server_group" {
  name = "sample-app-certman-slb-server-group-${var.env}"
  load_balancer_id = "${data.alicloud_slbs.app_slbs.slbs.0.id}"
  servers = [
    {
      server_ids = [
        "${alicloud_instance.certman_ecs.id}"
      ]
      port = 8080
      weight = 100
    }
  ]
}

// SLB forwarding rule
resource "alicloud_slb_rule" "rule" {
  name = "sample-app-certman-slb-rule-${var.env}"
  load_balancer_id = "${data.alicloud_slbs.app_slbs.slbs.0.id}"
  frontend_port = 80
  url = "/.well-known"
  server_group_id = "${alicloud_slb_server_group.certman_server_group.id}"
}
```

As you can see, this script creates an `alicloud_instance` resource based on our VM image. This ECS instance has a public IP address (`internet_max_bandwidth_out = 1`); without it. The instance would not be able to connect to internet, which is necessary for certbot.

Our SLB configuration is extended with a forwarding rule (`alicloud_slb_rule` resource) that redirects HTTP requests with URLs that starts with **/.well-known** to our new ECS instance. This redirection is made possible through a Add ECS instances to a VServer group (`alicloud_slb_server_group` resource).

Save and close this file with CTRL + X.

GitLab pipeline

Let's integrate our new scripts to our GitLab pipeline. Let's create a Bash script that calls Packer and Terraform:

```
# Go to the project folder
cd ~/projects/todolist

# Create a new pipeline script for the Certificate Manager
nano gitlab-ci-scripts/deploy/build_certman_infra.sh
```

Copy the following content into this new file:

```
#!/usr/bin/env bash
#
# Build the certificate manager infrastructure (RDS, VM image, ECS, ...)
#
# Required global variables:
#   - ALICLOUD_ACCESS_KEY
#   - ALICLOUD_SECRET_KEY
#   - ALICLOUD_REGION
#   - ENV_NAME
#   - DOMAIN_NAME
#   - SUB_DOMAIN_NAME
#   - EMAIL_ADDRESS
#   - ECS_ROOT_PASSWORD
#   - GITLAB_BUCKET_NAME
#   - GITLAB_BUCKET_ENDPOINT
#   - BUCKET_LOCAL_PATH
#   - CI_PIPELINE_IID
#   - OSSFS_VERSION
#

echo "Building the certificate manager infrastructure (environment: ${ENV_NAME}, region: ${ALICLOUD
 REGION}) ..."
```

```
_REGION}}...

# Set values for Terraform and Packer variables
export TF_VAR_env=${ENV_NAME}
export TF_VAR_db_account_password=${DB_ACCOUNT_PASSWORD}
export TF_VAR_ecs_root_password=${ECS_ROOT_PASSWORD}

export IMAGE_VERSION=${CI_PIPELINE_IID}
export ENVIRONMENT=${ENV_NAME}
export BUCKET_NAME=${GITLAB_BUCKET_NAME}
export BUCKET_ENDPOINT=${GITLAB_BUCKET_ENDPOINT}

# Extract Alibaba Cloud information for building the application image
cd infrastructure/15_certman/05_image
export BUCKET_DIR_PATH="$BUCKET_LOCAL_PATH/infrastructure/$ENV_NAME/15_certman/05_image"
mkdir -p ${BUCKET_DIR_PATH}
cp ${BUCKET_DIR_PATH}/*.tfstate* .
terraform init -input=false
terraform apply -input=false -auto-approve
rm -f ${BUCKET_DIR_PATH}/*
cp *.tfstate* ${BUCKET_DIR_PATH}
export SOURCE_IMAGE=$(terraform output image_id)
export INSTANCE_TYPE=$(terraform output instance_type)

# Build the certificate manager image
packer build certman_image.json

# Create/update the ECS, SLB server group and forward rule
cd ../10_ecs_slb_rule
export BUCKET_DIR_PATH="$BUCKET_LOCAL_PATH/infrastructure/$ENV_NAME/15_certman/10_ecs_slb_
rule"
mkdir -p ${BUCKET_DIR_PATH}
cp ${BUCKET_DIR_PATH}/*.tfstate* .
terraform init -input=false
terraform apply -input=false -auto-approve
rm -f ${BUCKET_DIR_PATH}/*
cp *.tfstate* ${BUCKET_DIR_PATH}

cd ../../..

echo "Certificate manager infrastructure successfully built (environment: ${ENV_NAME}, region: ${ALIC
```

```
LOUD_REGION})."
```

This script is composed of two main parts:

- Build the VM image (Terraform is used to obtain information from our Alibaba Cloud region).
- Create/update our cloud resources with Terraform.

Save this file with CTRL + X and edit **.gitlab-ci.yml**:

```
# Edit the GitLab pipeline definition
nano .gitlab-ci.yml
```

Add the following changes to **.gitlab-ci.yml**:

```
// ...
variables:
 // ...
 EMAIL_ADDRESS: "john.doe@example.org"
 // ...

// ...
deploy:
 // ...
 script:
  // ...
  - "./gitlab-ci-scripts/deploy/build_webapp_infra.sh"
  - "./gitlab-ci-scripts/deploy/build_certman_infra.sh"
  - "umount $BUCKET_LOCAL_PATH"
  // ...
 // ...
```

Only two lines need to be added:

- The new variable `EMAIL_ADDRESS`.
- A call to our new Bash script **./gitlab-ci-scripts/deploy/build_certman_infra.sh**.

Save your changes with CTRL + X.

Before we commit and push our modifications to GitLab, let's first add the new variable in the GitLab pipeline configuration:

1. Open GitLab (the URL must be like https://gitlab.my-sample-domain.xyz/).
2. Sign in if necessary.
3. Click **Projects** from the top menu and select **Your projects**.
4. Click the **todolist** project.
5. In the left-side navigation pane, select **Settings > CI/CD**.

6. Expand the **Variables** panel, and create the following variable:
   - EMAIL_ADDRESS = the email address where Let's Encrypt will send messages when the certificate is going to expire.
7. Click **Save variables**.

We can now commit our new scripts:

```
# Check files to commit
git status

# Add the modified and new files
git add infrastructure/15_certman/
git add gitlab-ci-scripts/deploy/build_certman_infra.sh
git add .gitlab-ci.yml

# Commit and push to GitLab
git commit -m "Add the Certificate Manager."
git push origin master
```

Verification

Check the logs of the **deploy** stage on your CI/CD pipeline on GitLab and make sure there is no error.

Let's check the status of our Certificate Manager ECS instance:

1. Log on to the ECS console.
2. Click **Instance** from the left-side navigation pane.
3. Select your region if necessary.
4. Search for your instance named **sample-app-certman-ecs-dev**.
5. Click **Connect** on the right side of your instance.
6. Authenticate yourself with the root user and the password you set in your ECS_ROOT_PASSWORD variable (in the GitLab pipeline settings).
7. Check that the services **ossfs**, **nginx**, and **certificate-updater** are running:

   ```
   # Check the running services configured with SystemD
   systemctl
   ```

   > **Note** You can scroll down by pressing the SPACE bar and quit by pressing Q.

8. Check Nginx is working as expected:

```
# Check the "/health" request (the response must be "OK")
curl http://localhost:8080/health

# Check the "/.well-known/" request (don't forget the last '/', the response must be "It works!")
curl http://localhost:8080/.well-known/
```

9. Check the OSS bucket is mounted properly:

```
# Check that OSSFS is working properly. It should contain the folders "backup", "certificate" and "i
nfrastructure"
ls /mnt/oss_bucket
```

10. Check the logs of the certificate updater:

```
# Check the logs of the certificate updater
journalctl --unit=certificate-updater
```

You can then test the web application from your computer with the following command:

```
# Check that the SLB is well configured with the new certificate
curl https://dev.my-sample-domain.xyz/
```

The `curl` command should succeed with the following logs:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>To-Do list</title>
  <link rel="stylesheet" href="css/index.css" media="screen">
</head>
<body>
<div id="react"></div>
<script src="built/bundle.js"></script>
</body>
</html>
```

Open your application in your web browser with the HTTPS URL (that is, https://dev.my-sample-domain.xyz/) and click the padlock icon on the left of the URL bar. It should indicate that the connection is secured:

Pre-production and production environments

Let's apply the changes on the pre-production:

1. Open GitLab (the URL must be like https://gitlab.my-sample-domain.xyz/).
2. Click **Projects** from the top menu and select **Your projects**.
3. Click the **todolist** project.
4. In the left-side navigation pane, select **Merge Requests**.
5. Click **New merge request**.
6. In the source branch, select **master**.
7. In the target branch, select **pre-production**.
8. Click **Compare branches and continue**.
9. Set the title field to **HTTPS configuration** and click **Submit merge request**.
10. Click **Merge**.
11. Follow the pipeline by clicking the **CI/CD** from the left-side navigation pane.

The pipeline should run with success (unfortunately it now takes about 1h to execute the complete process, mainly because of the Packer scripts).

After the pipeline execution, you can quickly check that it worked with curl:

```
# Check that the pre-production environment is properly updated
curl https://pre-prod.my-sample-domain.xyz/
```

The `curl` command should succeed with the following output:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>To-Do list</title>
  <link rel="stylesheet" href="css/index.css" media="screen">
</head>
<body>
<div id="react"></div>
<script src="built/bundle.js"></script>
</body>
</html>
```

Let's do the same with the production environment:

1. In your GitLab tab, select **Merge Requests** from the left-side navigation pane.
2. Click **New merge request**.
3. In the source branch, select **pre-production**.
4. In the target branch, select **production**.
5. Click **Compare branches and continue**.
6. Set the title field to **HTTPS configuration** and click **Submit merge request**.

7. Click **Merge**.
8. Follow the pipeline by clicking **CI/CD** from the left-side navigation pane.

Again, the pipeline should succeed like the other branches. After its execution, check the result with curl:

```
# Check that the production environment is well configured
curl https://www.my-sample-domain.xyz/
```

The `curl` command should succeed as well:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>To-Do list</title>
  <link rel="stylesheet" href="css/index.css" media="screen">
</head>
<body>
<div id="react"></div>
<script src="built/bundle.js"></script>
</body>
</html>
```

16.3. Log management

Introduction

Working with application logs become more complex when the number of servers increase: for example when there is only one server, an administrator just needs to connect to this machine and reads the */var/logs* folder and execute commands such as `journalctl --unit=todo-list`. But when the number of servers increase, the same administrator must connect to each machine to find the information he's looking for. This become even worse when auto-scaling is enabled, because servers are automatically created and released.

A solution to this problem is to use the Log Service: its role is to collect logs from servers and let administrators/developers to search in them.

> **Note** You can find the source code containing the modifications described in this part in the folder sample-app/version5.

Architecture

Configuring Alibaba Cloud Log Service is a bit complex. The following diagram illustrates how it works:

In this diagram we can see that in each ECS instance, an application generates logs and sends them to Rsyslog (this is the case of our java application, thanks to the SystemD configuration file that specifies `StandardOutput=syslog` and `StandardError=syslog`).

Rsyslog must then be configured to forward the logs to Logtail, a log collection agent similar to LogStash, responsible for sending logs to the Log Service.

The Log Service is organized in log project that contains log stores. In our case we just need one log project and one log store. The Log Service provides endpoints in each region for Logtail (such as http://logtail.ap-southeast-1-intranet.log.aliyuncs.com).

Both the log project and Logtail must be configured:

- Logtail needs a configuration to understand how to parse logs from Rsyslog (the fields / columns in each log line) and how to send them to the Log Service (the endpoint, buffer size, and so on)
- The log project needs to be configured to know what are the logs that needs to be stored (for example, from which data source). This configuration is assigned to the ECS instances though machine groups.

Infrastructure improvements

Cloud resources

The first step is to add a log project, a log store and a log machine in our basis infrastructure. Open a terminal on your computer and type:

```
# Go to the project folder
cd ~/projects/todolist


# Edit the basis infrastructure definition
nano infrastructure/05_vpc_slb_eip_domain/main.tf
```

Add the following code at the end of the file:

```
// Log project, store and machine group
resource "alicloud_log_project" "app_log_project" {
  name = "sample-app-log-project-${var.env}"
  description = "Sample web application log project (${var.env} environment)."
}
resource "alicloud_log_store" "app_log_store" {
  project = "${alicloud_log_project.app_log_project.name}"
  name = "sample-app-log-store-${var.env}"
}
resource "alicloud_log_machine_group" "app_log_machine_group" {
  project = "sample-app-log-project-${var.env}"
  name = "sample-app-log-machine-group-${var.env}"
  identify_type = "userdefined"
  identify_list = [
    "logtail-id-${var.env}"
  ]
}
```

We should also add an ingress security group rule in order to open the port 11111 (used by Logtail). Add the following block under `accept_8080_rule` :

```
resource "alicloud_security_group_rule" "accept_11111_rule" {
  type = "ingress"
  ip_protocol = "tcp"
  nic_type = "intranet"
  policy = "accept"
  port_range = "11111/11111"
  priority = 1
  security_group_id = "${alicloud_security_group.app_security_group.id}"
  cidr_ip = "0.0.0.0/0"
}
```

Save the changes by pressing CTRL + X.

> **Note** If you check the Terraform documentation about `alicloud_log_machine_group`, you can see that the `identify_type` can take 2 values: **ip** and **userdefined**. The **ip** one is less flexible and a bit problematic for our CI/CD pipeline, as it requires us to create the ECS instances first (to get the private IP addresses), and then to configure the log machine group and finally to complete the Log Service configuration. This is problematic because the ECS instances would start without a complete logging configuration (at that time the CI/CD pipeline is not finished yet and the Log Service is not ready), so the logtail application running on the ECS instances will fail to initialize.

For more information about user-defined identity, see Create a custom ID-based machine group.

Logtail configuration on the log project

There are two Logtail configurations: one on the ECS instance side, one on the log project side. This section deals with the log project side.

Unfortunately the Terraform provider for Alibaba Cloud does not support Logtail configuration on the log project side, so we will manage it automatically with the API service.

There are several ways to call this API, one solution is to use the Python SDK to create a script that will be called by GitLab:

```
# Create the Python script that will update the Logtail configuration on the log project side
nano gitlab-ci-scripts/deploy/update_logtail_config.py
```

Copy the following content into this file:

```
#!/usr/bin/python3

import sys
from aliyun.log.logclient import LogClient
from aliyun.log.logexception import LogException
from aliyun.log.logtail_config_detail import SyslogConfigDetail

# Read the arguments
accessKeyId = sys.argv[1]
accessKeySecret = sys.argv[2]
regionId = sys.argv[3]
environment = sys.argv[4]
print("Update the Logtail configuration on the log project (environment = " + environment +
    ", region = " + regionId + ")")

endpoint = regionId + ".log.aliyuncs.com"
logProjectName = "sample-app-log-project-" + environment
logStoreName = "sample-app-log-store-" + environment
logtailConfigName = "sample-app-logtail-config-" + environment
```

```
logMachineGroupName = "sample-app-log-machine-group-" + environment

# Load the existing Logtail configuration
print("Loading existing Logtail configuration (endpoint = " + endpoint +
    ", logProjectName = " + logProjectName + ", logtailConfigName = " + logtailConfigName + ")...")

client = LogClient(endpoint, accessKeyId, accessKeySecret)
existingConfig = None
try:
    response = client.get_logtail_config(logProjectName, logtailConfigName)
    existingConfig = response.logtail_config
    print("Existing logtail configuration found: ", existingConfig.to_json())
except LogException:
    print("No existing logtail configuration found.")

# Create or update the logtail configuration
configDetail = SyslogConfigDetail(logstoreName=logStoreName, configName=logtailConfigName, tag="
sys_tag")
if existingConfig is None:
    print("Create the logtail configuration:", configDetail.to_json())
    client.create_logtail_config(logProjectName, configDetail)
else:
    print("Update the logtail configuration:", configDetail.to_json())
    client.update_logtail_config(logProjectName, configDetail)

# Apply the configuration to the machine group
print("Apply the logtail configuration to the machine group", logMachineGroupName)
client.apply_config_to_machine_group(logProjectName, logtailConfigName, logMachineGroupName)
```

Save and quit by pressing CTRL + X.

As you can see this script creates or updates the Logtail configuration and then links it to the machine group.

VM images

The next step is to modify our Packer scripts to install Logtail and configure it:

```
# Edit the application image script
nano infrastructure/10_webapp/10_image/app_image.json
```

Add the following provisioner at the end of the `provisioners` array:

```
{
  "type": "shell",
  "inline": [
    "export REGION=\"{{user `region_id`}}\"",
    "export ENVIRONMENT=\"{{user `environment`}}\"",
    "mkdir -p /etc/ilogtail",
    "echo \"logtail-id-${ENVIRONMENT}\" > /etc/ilogtail/user_defined_id",
    "wget \"http://logtail-release-${REGION}.oss-${REGION}.aliyuncs.com/linux64/logtail.sh\" -O logtail.s
h",
    "chmod 755 logtail.sh",
    "./logtail.sh install auto",
    "export STREAMLOG_FORMATS='[{\"version\": \"0.1\", \"fields\": []}]'",
    "sed -i \"s/\\(\\\"streamlog_open\\\" : \\).*\\$/\\1true,/\" /usr/local/ilogtail/ilogtail_config.json",
    "sed -i \"s/\\(\\\"streamlog_formats\\\":\\).*\\$/\\1${STREAMLOG_FORMATS},/\" /usr/local/ilogtail/
ilogtail_config.json",
    "/etc/init.d/ilogtaild stop",
    "rm /usr/local/ilogtail/app_info.json",
    "rm /etc/init.d/ilogtaild",
    "systemctl enable logtail"
  ]
}
```

Save and exit with CTRL + X. Then do the same with the certificate manager image:

```
# Edit the certificate manager image script
nano infrastructure/15_certman/05_image/certman_image.json
```

Add the same provisioner as above, and then save and exit with CTRL + X.

As you can see, this provisioner executes the following actions:

1. Create the file **/etc/ilogtail/user_defined_id** and put **logtail-id-${ENVIRONMENT}** inside. This is a necessary step to inform Logtail that it is running inside an ECS instance that belongs to the machine group **sample-app-log-machine-group-${var.env}** (created through Terraform).
2. Download and install Logtail (also called ilogtail). Note that this installation script automatically starts Logtail on the machine.
3. Modify the the properties `streamlog_open` and `streamlog_formats` in the Logtail configuration file **/usr/local/ilogtail/ilogtail_config.json**. Note that Logtail has configuration files in two locations: */etc/ilogtail/* and */usr/local/ilogtail/*.
4. Stop Logtail and remove the configuration file **/usr/local/ilogtail/app_info.json**, as it contains the hostname and private ip address of the ECS instance used by Packer to create the VM image. Logtail will automatically re-create this file when the VM image is used to start our real ECS instances.

5. Remove the Logtail default startup script (/etc/init.d/ilogtaild) and replace it by our own (we will create it in a moment). We need to do that because we need to control the moment when Logtail starts: when our ECS instance starts for the first time, cloud-init scripts reconfigure the system by setting attributes such as the hostname. We need to make sure that Logtail starts after cloud-init, that's why we create our own SystemD script.

Let's create this SystemD script now:

```
# Create our own SystemD script for Logtail
nano infrastructure/10_webapp/10_image/resources/logtail.service
```

Copy the following content into this file:

```
[Unit]
Description=logtail
After=syslog.target
After=network.target
After=cloud-config.service
After=cloud-final.service
After=cloud-init-local.service
After=cloud-init.service
After=cloudmonitor.service
After=cloud-config.target

[Service]
Type=simple
RemainAfterExit=yes
ExecStartPre=/bin/sleep 5
ExecStart=/usr/local/ilogtail/ilogtail
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=logtail
WorkingDirectory=/usr/local/ilogtail

[Install]
WantedBy=multi-user.target
```

Save and quit by pressing CTRL + X.

As you can see at the beginning of this script, we start Logtail after the cloud-init scripts. We even wait for 5 seconds with `ExecStartPre=/bin/sleep 5` to make sure the cloud-init scripts have completed their tasks.

Copy this file for the certificate manager machine:

```
# Copy the Logtail startup script
cp infrastructure/10_webapp/10_image/resources/logtail.service infrastructure/15_certman/05_image
/resources/
```

We also need to configure Rsyslog to forward logs to Logtail:

```
# Create the Rsyslog configuration script
nano infrastructure/10_webapp/10_image/resources/rsyslog-logtail.conf
```

Enter the following content into this file:

```
$ActionQueueFileName fwdRule1 # unique name prefix for spool files
$ActionQueueMaxDiskSpace 1g # 1gb space limit (use as much as possible)
$ActionQueueSaveOnShutdown on # save messages to disk on shutdown
$ActionQueueType LinkedList # run asynchronously
$ActionResumeRetryCount -1 # infinite retries if host is down

# Defines the fields of log data
$template ALI_LOG_FMT,"0.1 sys_tag %timegenerated:::date-unixtimestamp% %fromhost-ip% %hostn
ame% %pri-text% %protocol-version% %app-name% %procid% %msgid% %msg:::drop-last-lf%\n"
*.* @@127.0.0.1:11111;ALI_LOG_FMT
```

Save and exit by pressing CTRL + X. Copy the same file for the certificate manager:

```
# Copy the Rsyslog configuration script
cp infrastructure/10_webapp/10_image/resources/rsyslog-logtail.conf infrastructure/15_certman/05_i
mage/resources/
```

Add provisioners into the application Packer script to upload the configuration files:

```
# Edit the application image script
nano infrastructure/10_webapp/10_image/app_image.json
```

Add the following provisioners BEFORE the shell one we have just created above:

```
{
  "type": "file",
  "source": "resources/rsyslog-logtail.conf",
  "destination": "/etc/rsyslog.d/80-logtail.conf"
}
```

```
{
  "type": "file",
  "source": "resources/logtail.service",
  "destination": "/etc/systemd/system/logtail.service"
}
```

Save and exit with CTRL + X. Edit in a similar way the certificate manager image script:

```
# Edit the certificate manager image script
nano infrastructure/15_certman/05_image/certman_image.json
```

Add the same provisioners as above then save and quit with CTRL + X.

The last step is to force our applications to start after Logtail is started, in order to make sure that their logs are completely collected. Let's edit the SystemD scripts:

```
# Edit the SystemD script for our web application
nano infrastructure/10_webapp/10_image/resources/todo-list.service
```

Add the following content under `After=network.target` :

```
After=logtail.service
```

Save and quit by pressing CTRL + X. Let's edit the certificate updater as well:

```
# Edit the SystemD script for our certificate updater
nano infrastructure/15_certman/05_image/resources/certificate-updater.service
```

Add the following content under `After=ossfs.service` :

```
After=logtail.service
```

Save and quit by pressing CTRL + X.

CI/CD pipeline update

We need to update our pipeline definition file (.gitlab-ci.yml) to run our Python script **update_logtail_config.py**. But before we need to create a script that installs its dependencies:

```
# Create a script that installs the dependencies for update_logtail_config.py nano gitlab-ci-scripts/de
ploy/install_python_packages.sh
```

Copy the following script into the editor:

```
#!/usr/bin/env bash
#
# Install PIP and aliyun-log-python-sdk.
#

echo "Installing Python packages..."

export DEBIAN_FRONTEND=noninteractive
apt-get -y update
apt-get -y install python3-pip

pip3 install -U aliyun-log-python-sdk

echo "Python packages installed with success."
```

Save and quit by pressing CTRL + X; then modify the file **.gitlab-ci.yml**:

```
# Edit the pipeline definition file
nano .gitlab-ci.yml
```

Modify the `deploy` block accordingly:

```
deploy:
  # ...
  script:
    - "export ENV_NAME=$(./gitlab-ci-scripts/deploy/get_env_name_by_branch_name.sh $CI_COMMIT_REF_NAME)"
    - "export SUB_DOMAIN_NAME=$(./gitlab-ci-scripts/deploy/get_sub_domain_name_by_branch_name.sh $CI_COMMIT_REF_NAME)"
    - "export BUCKET_LOCAL_PATH=/mnt/oss_bucket"
    - "./gitlab-ci-scripts/deploy/install_tools.sh"
    - "./gitlab-ci-scripts/deploy/install_python_packages.sh"
    - "./gitlab-ci-scripts/deploy/mount_ossfs.sh"
    - "./gitlab-ci-scripts/deploy/build_basis_infra.sh"
    - "python3 ./gitlab-ci-scripts/deploy/update_logtail_config.py $ALICLOUD_ACCESS_KEY $ALICLOUD_SECRET_KEY $ALICLOUD_REGION $ENV_NAME"
    - "./gitlab-ci-scripts/deploy/build_webapp_infra.sh"
    - "./gitlab-ci-scripts/deploy/build_certman_infra.sh"
    - "umount $BUCKET_LOCAL_PATH"
    - "sleep 10"
  # ...
```

Save and exit with CTRL + X.

As you can see we have added two commands:

- ./gitlab-ci-scripts/deploy/install_python_packages.sh
- python3 ./gitlab-ci-scripts/deploy/update_logtail_config.py $ALICLOUD_ACCESS_KEY $ALICLOUD_SECRET_KEY $ALICLOUD_REGION $ENV_NAME

The final step is to commit and push the changes to GitLab:

```
# Check files to commit
git status

# Add the modified and new files
git add .gitlab-ci.yml
git add gitlab-ci-scripts/deploy/install_python_packages.sh
git add gitlab-ci-scripts/deploy/update_logtail_config.py
git add infrastructure/05_vpc_slb_eip_domain/main.tf
git add infrastructure/10_webapp/10_image/app_image.json
git add infrastructure/10_webapp/10_image/resources/logtail.service
git add infrastructure/10_webapp/10_image/resources/todo-list.service
git add infrastructure/10_webapp/10_image/resources/rsyslog-logtail.conf
git add infrastructure/15_certman/05_image/certman_image.json
git add infrastructure/15_certman/05_image/resources/certificate-updater.service
git add infrastructure/15_certman/05_image/resources/logtail.service
git add infrastructure/15_certman/05_image/resources/rsyslog-logtail.conf

# Commit and push to GitLab
git commit -m "Collect logs with the Log Service."
git push origin master
```

Check your CI/CD pipeline on GitLab, in particularly the logs of the **deploy** stage and make sure there is no error.

Check that the Log Service is correctly configured:

1. Log on to the Log Service console.
2. You should see the log project **sample-app-log-project-dev**. Click it.
3. You should be able to see the log store **sample-app-log-store-dev**.
4. In the left-side navigation pane, click **Log Machine Group**.
5. You should see the group **sample-app-log-machine-group-dev**. Click **Status** on the right.
6. A popup should open with three IP addresses: one like **192.168.0.x** and two like **192.168.1.x**. The heartbeat column must contain **OK** (this value means that Logtail is running on the ECS instance):

7. Close the popup and click **Logtail Config** from the left-side navigation pane. You should see one configuration **sample-app-logtail-config-dev** with **syslog** as data source. Click this configuration.
8. The new page should display a form with a field **Tag Settings** containing **sys_tag**. This value must be exactly the same as the one in the Rsyslog configuration file. Click the **Next**: the machines group **sample-app-log-machine-group-dev** must be displayed and checked. Click **Cancel** to close this wizard.
9. Click **Logstores** from the left-side navigation pane.

10. Click **Preview** next to the **sample-app-log-store-dev** log store.
11. A new web browser tab should open and show collected logs like this:

> **Note** If there is no log, try to switch to the **Shard: 1** and click **Preview**. You can generate logs by yourself by opening your web application (http://dev.my-sample-domain.xyz/) and by creating and deleting tasks. Sometime it may be necessary to wait for few minutes for the logs to appear.

Log search

Now that we collect logs, let's check how to search into them:

1. Open your web application (http://dev.my-sample-domain.xyz/) and create 4 tasks (Task 1, Task 2, Task 3 and Task 4), and then delete them one by one.
2. Log on to the the Log Service console.
3. Click the log project **sample-app-log-project-dev**.
4. Click **Search** next to the **sample-app-log-store-dev** log store.
5. The new page should display logs with a search bar on top. Enter `app-name=todo-list` in this bar and click **Search & Analysis**.
6. The **Raw Logs** panel should now only contains the logs of our application:

We can see the following messages:

- Time: 12-17 10:06:43
  - hostname: sample-app-ecs-zone-1-dev
  - msg: 2018-12-17 10:06:44.111 INFO 705 — [nio-8080-exec-4] c.a.i.t.controllers.TaskController : Delete the task with the UUID: 4d3d5eb5-e21b-42f5-9681-fec5b52bf266
- Time: 12-17 10:06:42
  - hostname: sample-app-ecs-zone-0-dev
  - msg: 2018-12-17 10:06:43.584 INFO 702 — [nio-8080-exec-5] c.a.i.t.controllers.TaskController : Delete the task with the UUID: 960562e1-9806-4df5-95fd-1403b8d8e186
- Time: 12-17 10:06:41
  - hostname: sample-app-ecs-zone-1-dev
  - msg: 2018-12-17 10:06:42.966 INFO 705 — [nio-8080-exec-9] c.a.i.t.controllers.TaskController : Delete the task with the UUID: 8145a65b-c1d4-4a24-b1b9-2af3a53b324c
- Time: 12-17 10:06:41
  - hostname: sample-app-ecs-zone-0-dev
  - msg: 2018-12-17 10:06:42.464 INFO 702 — [nio-8080-exec-1] c.a.i.t.controllers.TaskController : Delete the task with the UUID: efa997bf-f101-429a-bc01-badd69241d5a
- Time: 12-17 10:06:39
  - hostname: sample-app-ecs-zone-1-dev

  - msg: 2018-12-17 10:06:40.802 INFO 705 — [nio-8080-exec-5] c.a.i.t.controllers.TaskController : Create a new task: Task{uuid='4d3d5eb5-e21b-42f5-9681-fec5b52bf266', description='Task 4'}

- Time: 12-17 10:06:35
  - hostname: sample-app-ecs-zone-0-dev
  - msg: 2018-12-17 10:06:36.970 INFO 702 — [nio-8080-exec-3] c.a.i.t.controllers.TaskController : Create a new task: Task{uuid='960562e1-9806-4df5-95fd-1403b8d8e186', description='Task 3'}
- Time: 12-17 10:06:32
  - hostname: sample-app-ecs-zone-1-dev
  - msg: 2018-12-17 10:06:33.274 INFO 705 — [nio-8080-exec-4] c.a.i.t.controllers.TaskController : Create a new task: Task{uuid='8145a65b-c1d4-4a24-b1b9-2af3a53b324c', description='Task 2'}
- Time: 12-17 10:06:25
  - hostname: sample-app-ecs-zone-0-dev
  - msg: 2018-12-17 10:06:26.112 INFO 702 — [nio-8080-exec-8] c.a.i.t.controllers.TaskController : Create a new task: Task{uuid='efa997bf-f101-429a-bc01-badd69241d5a', description='Task 1'}

It is interesting to see how the load balancer distributes HTTP requests to each ECS instance. This is due to the fact that we configured its scheduling algorithm to weighted round robin and set the same weight for all ECS instances.

We can also remark that each row is organized by fields (app-name, hostname, msg, and so on). We can make search easier and faster by adding our own fields (for example, by splitting a message **2018-12-17 10:06:44.111 INFO 705 — [nio-8080-exec-4] c.a.i.t.controllers.TaskController : Delete the task with the UUID: 4d3d5eb5-e21b-42f5-9681-fec5b52bf266** into datetime, level, process-id, thread-name, logger-name, and message). For that we can modify Rsyslog and Logtail configurations (attribute `streamlog_formats` in **/usr/local/ilogtail/ilogtail_config.json**) or directly modify our Java application to use the aliyun-log-log4j-appender.

> **Note** If you let your system running for one day, you can also check the logs of the certificate manager by searching for the query `app-name=certificate-updater`.

Pre-production and production environments

Now that the development environment is ready, let's apply the changes to the pre-production and production environments as well.

Follow the instructions described in the previous topic to create merge requests between the master branch into the pre-production one, and then from the pre-production to the production one.

You can check the configuration by browsing to the Log Service console and by exploring the log projects **sample-app-log-project-pre-prod** and **sample-app-log-project-prod**.

16.4. Speeding up CI and CD pipeline

Introduction

Until now we have been focusing on adding new functionalities to our application (HTTPS and centralized logs). However, in doing so we have slowed down substantially our CI/CD pipeline, as it now takes about one hour to complete the full process.

The goal of this tutorial part is to focus on this slow pipeline problem and to find ways to accelerate it.

> **Note** You can find the source code containing the modifications described in this part in the folders sample-app/version6 and deployment-toolbox/version1.

Deployment Docker image

The slowest stage of our pipeline is the one responsible for deployment, and its first task is to download and install tools. It usually takes several minutes to complete and unnecessarily wastes resources such as network bandwidth.

A way to speed up this first task is to create our own Docker image, and then use it in our pipeline.

Docker repository creation

The first step is to create a repository through the Container Registry service to host our own Docker images. Open a web browser tab and execute the following instructions:

1. Log on to the Container Registry console.
2. If necessary, select your region on top of the page.
3. Click **Namespace** from the left-side navigation pane.
4. Click **Create Namespace**.
5. In the popup form, set a field value corresponding to your domain name such as **my-sample-domain-xyz** (replace dots **.** by dashes **-**) and click **Confirm**. Note that we use the domain name because namespaces must be unique among all accounts in Alibaba Cloud.
6. Click **Repositories** from the left-side navigation pane.
7. Click **Create Repository**.
8. Fill the popup form with the following values:
   - Region = your region
   - Namespace = your namespace such as my-sample-domain-xyz
   - Repository Name = deployment-toolbox
   - Summary = Ubuntu with deployment tools (Terraform, Packer, and so on)
   - Repository Type = Private
9. Click **Next**.
10. Select **Local Repository** and click **Create Repository**.

We then need to create a RAM user in order to let Docker to access to our repository:

1. Copy the URL next to **RAM User Logon Link**. We will use it later.
2. Click **Users** from the left-side navigation pane.

3. Click **Create User**.
4. In the pop-up window form set **sample-app-gitlab** in the **User Name** field and click **OK**.
5. The page should refresh itself and display our **sample-app-gitlab** user. Click **Authorize** on the right.
6. In the new pop-up window, select the policy name **AliyunContainerRegistryFullAccess** and click the button with an arrow pointing to the right.
7. Click **OK** to close the pop-up window.
8. Click **Manage** on the right of the user **sample-app-gitlab**.
9. Click **Enable Console Logon**.
10. In the pop-up form, enter twice the same password, uncheck the checkbox **On your next logon you must reset the password**, and click **OK**.

We now need to set the Docker password for this RAM user:

1. Open a private web browser window and browse to the **RAM User Logon Link URL** you copied earlier (it should be something like **http://signin-intl.aliyun.com/5939306421830******/login.htm**).
2. Login with your ram username and password (the username should be something like **sample-app-gitlab@5939306421******. The password is the one you set earlier.
3. Log on to the Container Registry console.
4. Click **Reset Docker Login Password**.
5. Set a new password and click **OK**.
6. Close your private web browser window.

If you have Docker installed on your computer, you can test your configuration like this:

1. Log on to the Container Registry console (with your normal account).
2. If necessary, select your region on top of the page.
3. The repository **deployment-toolbox** should be displayed. Move your mouse cursor on top of the icon that looks like an arrow going into a box under the **Repository Address** column. A pop-up window should open with multiple URLs:

4. Click the first address (next to **Internet**) to copy it (it should be like **registry-intl.ap-southeast-1.aliyuncs.com/my-sample-domain-xyz/deployment-toolbox**).
5. Open a terminal and type:

```
# Test your repository configuration
docker login --username=sample-app-gitlab@593930642183**** registry-intl.ap-southeast-1.aliyuncs.com
```

This command should prompt for the password you set earlier when you clicked **Reset Docker Login Password**. If the configuration is good, the command should print **Login Succeeded**.

> **Note**
>
> - The `--username` argument should be `sample-app-gitlab@your-user-id-or-enterprise-alias`. You can find your user ID or enterprise alias inside the **RAM User Logon Link** you copied earlier (for example, if the link is **http://signin-intl.aliyun.com/593930642183****/login.htm**, then the user ID is **593930642183****).
> - The next argument is the domain name of your repository address (for example, if the repository address is **registry-intl.ap-southeast-1.aliyuncs.com/my-sample-domain-xyz/deployment-toolbox**, then the argument is **registry-intl.ap-southeast-1.aliyuncs.com**).

Docker image project

The next step is to create a new GitLab project where we will host our Dockerfile:

1. Open GitLab (the URL must be like https://gitlab.my-sample-domain.xyz/).
2. In the home page, click **New project**.
3. Fill the new form with the following information:
   - Project name = deployment-toolbox
   - Project slug = deployment-toolbox
   - Visibility Level = Private
4. Click **Create project**.
5. In the new page, copy the URL for git (such asgit@gitlab.my-sample-domain.xyz:marcplouhinec/deployment-toolbox.git).

Open a terminal on your computer and run:

```
# Go to the projects directory
cd projects

# Git clone our new project (adapt the URL)
git clone git@gitlab.my-sample-domain.xyz:marcplouhinec/deployment-toolbox.git

# Go to the new project folder
cd deployment-toolbox

# Create our Docker image definition file
nano Dockerfile
```

Copy the following content into the editor:

```
FROM ubuntu:18.04

ENV OSSFS_VERSION=1.80.5
ENV TERRAFORM_VERSION=0.11.11
ENV PACKER_VERSION=1.3.3

# Install OSSFS
RUN apt-get -y update
RUN apt-get -y install gdebi-core wget unzip libssl1.0.0
RUN wget "https://github.com/aliyun/ossfs/releases/download/v${OSSFS_VERSION}/ossfs_${OSSFS_V
ERSION}_ubuntu16.04_amd64.deb"
RUN gdebi -n "ossfs_${OSSFS_VERSION}_ubuntu16.04_amd64.deb"

# Install Terraform
RUN wget "https://releases.hashicorp.com/terraform/${TERRAFORM_VERSION}/terraform_${TERRAFOR
M_VERSION}_linux_amd64.zip"
RUN unzip "terraform_${TERRAFORM_VERSION}_linux_amd64.zip" -d /usr/local/bin/

# Install Packer
RUN wget "https://releases.hashicorp.com/packer/${PACKER_VERSION}/packer_${PACKER_VERSION}_lin
ux_amd64.zip"
RUN unzip "packer_${PACKER_VERSION}_linux_amd64.zip" -d /usr/local/bin/

# Install Python packages
RUN apt-get -y install python3-pip
RUN pip3 install -U aliyun-log-python-sdk

CMD ["/bin/bash"]
```

Save and quit by pressing CTRL + X. If you have Docker on your machine, you can test this Dockerfile with the following commands:

```
# Build the Docker image
docker build -t deployment-toolbox:latest .

# Create a container with our new image
docker run -it deployment-toolbox:latest
```

The last command executes Bash inside the container. Let's check that our tools are correctly installed:

```
# Check OSSFS version
ossfs --version

# Check Terraform version
terraform version

# Check Packer version
packer version

# Check our Python dependency version
pip3 show aliyun-log-python-sdk

# Exit and kill the container
exit
```

Let's create the GitLab pipeline definition file:

```
# Create the pipeline definition file
nano .gitlab-ci.yml
```

Put the following text into this file:

```
image: docker:stable

variables:
  DOCKER_HOST: tcp://docker:2375/
  DOCKER_DRIVER: overlay2
  REGISTRY_USERNAME: sample-app-gitlab@your-user-id-or-enterprise-alias
  REGISTRY_PASSWORD: your-docker-login-password
  REGISTRY_URL: registry-intl.ap-southeast-1.aliyuncs.com
  IMAGE_URL: registry-intl.ap-southeast-1.aliyuncs.com/my-sample-domain-xyz/deployment-toolbox

services:
  - docker:dind

stages:
  - build

build:
  stage: build
  before_script:
    - docker login -u $REGISTRY_USERNAME -p $REGISTRY_PASSWORD $REGISTRY_URL
  script:
    - docker pull $IMAGE_URL:latest || true
    - docker build --cache-from $IMAGE_URL:latest --tag $IMAGE_URL:$CI_PIPELINE_IID --tag $IMAGE_URL
:latest .
    - docker push $IMAGE_URL:$CI_PIPELINE_IID
    - docker push $IMAGE_URL:latest
```

Save and quit with CTRL + X.

Before we commit and push our changes to GitLab, we first need to add new variables:

1. Open your web browser tab with GitLab. The deployment-toolbox project should be displayed.
2. In the left-side navigation pane, select **Settings > CI/CD**.
3. Expand the **Variables** panel, and create the following variables:
   - REGISTRY_USERNAME = the username you already used in the previous section when you have tested your configuration with `docker login`.
   - REGISTRY_PASSWORD = the password is the one you set when you clicked **Reset Docker Login Password**.
   - REGISTRY_URL = the domain name of your repository address.
   - IMAGE_URL = your repository address.
4. Click **Save variables**.

Let's commit the changes to GitLab:

```
# Check the files to commit
git status

# Add the new files
git add .gitlab-ci.yml
git add Dockerfile

# Commit and push to GitLab
git commit -m "Create the Dockerfile."
git push origin master
```

Check your CI/CD pipeline (for the **deployment-toolbox** project) and make sure there is no error.

You can also check on the Container Registry web console that the Docker image has been successfully pushed:

1. Log on to the Container Registry console.
2. Click **Manage** next to the **deployment-toolbox** repository.
3. Click **Tags** from the left-side navigation pane.

The page should display your image tags:

Pipeline update

Let's update our pipeline in order to use our Docker image. Open your terminal and run:

```
# Go to the web application project folder
cd ~/projects/todolist

# Remove the tool installation scripts
rm gitlab-ci-scripts/deploy/install_tools.sh
rm gitlab-ci-scripts/deploy/install_python_packages.sh

# Edit the pipeline definition file
nano .gitlab-ci.yml
```

Apply the following modifications to this file:

1. Remove `TERRAFORM_VERSION: 0.11.11` and `PACKER_VERSION: 1.3.3` from the `variables` block.
2. In the `deploy` block, replace the `ubuntu:18.04` image by your image; it should be something like `registry-intl.ap-southeast-1.aliyuncs.com/my-sample-domain-xyz/deployment-toolbox:latest`.
3. In the `deploy` block, remove the two scripts - `./gitlab-ci-scripts/deploy/install_tools.sh` and -

`./gitlab-ci-scripts/deploy/install_python_packages.sh` .

Save and quit with CTRL + X.

Before we commit our changes, we should configure GitLab because our Docker repository is private:

1. Open GitLab (the URL must be like https://gitlab.my-sample-domain.xyz/).
2. Switch to the **todolist** project.
3. In the deploy block, remove the two scripts - ./gitlab-ci-scripts/deploy/install_tools.sh and - ./gitlab-ci-scripts/deploy/install_python_packages.sh.
4. Expand the **Variables** panel, and create the variable DOCKER_AUTH_CONFIG with the following content:

```
"auths": { "registry-intl.ap-southeast-1.aliyuncs.com": { "auth": "3dFtcGxlLXFwcC1naXRsYWJAMTkzOTMwNjQyMTgzDMg2ODpIYW5nemhvdTEw" } }
```

> **Note**
>
> - The URL `registry-intl.ap-southeast-1.aliyuncs.com` must be adapted to your registry domain name.
> - The `auth` value `3dFtcGxlLXFwcC1naXRsYWJAMTkzOTMwNjQyMTgzDMg2ODpIYW5nemhvdTEw` is a base64 string build like this:
>
> ```
> echo -n "sample-app-gitlab@your-user-id-or-enterprise-alias:your-docker-login-password" | base64
> ```

5. Click **Save variables**.

We can now commit the changes:

```
# Check the files to commit
git status

# Add the modified and deleted files
git add .gitlab-ci.yml
git add gitlab-ci-scripts/deploy/install_tools.sh
git add gitlab-ci-scripts/deploy/install_python_packages.sh

# Commit and push to GitLab
git commit -m "Replace the Ubuntu image by our deployment-toolbox."
git push origin master
```

Check your CI/CD pipeline on GitLab, the **deploy** stage should be slightly faster.

Parallelization

The main reason the deploy stage takes so much time is because of the creation of the VM images. Fortunately this stage can be done in parallel: after we deploy the basis infrastructure (VPC, SLB, and so on), we can create/update the web application and the certificate manager cloud resources at the same time. Open your terminal and execute the following commands:

```
# Go to the web application project folder
cd ~/projects/todolist

# Edit the pipeline definition file
nano .gitlab-ci.yml
```

Let's start by replacing the `deploy` stage by `deploy_basis` and `deploy_apps`:

```
stages:
  - build
  - quality
  - deploy_basis
  - deploy_apps
```

Then split the `deploy` job into 3 blocks:

```
deploy_basis:
  stage: deploy_basis
  image: registry-intl.ap-southeast-1.aliyuncs.com/my-sample-domain-xyz/deployment-toolbox:latest
  script:
    - "export ENV_NAME=$(./gitlab-ci-scripts/deploy/get_env_name_by_branch_name.sh $CI_COMMIT_REF_NAME)"
    - "export SUB_DOMAIN_NAME=$(./gitlab-ci-scripts/deploy/get_sub_domain_name_by_branch_name.sh $CI_COMMIT_REF_NAME)"
    - "export BUCKET_LOCAL_PATH=/mnt/oss_bucket"
    - "./gitlab-ci-scripts/deploy/mount_ossfs.sh"
    - "./gitlab-ci-scripts/deploy/build_basis_infra.sh"
    - "python3 ./gitlab-ci-scripts/deploy/update_logtail_config.py $ALICLOUD_ACCESS_KEY $ALICLOUD_SECRET_KEY $ALICLOUD_REGION $ENV_NAME"
    - "umount $BUCKET_LOCAL_PATH"
    - "sleep 10"
  only:
    - master
    - pre-production
    - production

deploy_webapp:
```

```
  stage: deploy_apps
  image: registry-intl.ap-southeast-1.aliyuncs.com/my-sample-domain-xyz/deployment-toolbox:latest
  script:
    - "export ENV_NAME=$(./gitlab-ci-scripts/deploy/get_env_name_by_branch_name.sh $CI_COMMIT_RE
F_NAME)"
    - "export BUCKET_LOCAL_PATH=/mnt/oss_bucket"
    - "./gitlab-ci-scripts/deploy/mount_ossfs.sh"
    - "./gitlab-ci-scripts/deploy/build_webapp_infra.sh"
    - "umount $BUCKET_LOCAL_PATH"
    - "sleep 10"
  only:
    - master
    - pre-production
    - production

deploy_certman:
  stage: deploy_apps
  image: registry-intl.ap-southeast-1.aliyuncs.com/my-sample-domain-xyz/deployment-toolbox:latest
  script:
    - "export ENV_NAME=$(./gitlab-ci-scripts/deploy/get_env_name_by_branch_name.sh $CI_COMMIT_RE
F_NAME)"
    - "export SUB_DOMAIN_NAME=$(./gitlab-ci-scripts/deploy/get_sub_domain_name_by_branch_name.s
h $CI_COMMIT_REF_NAME)"
    - "export BUCKET_LOCAL_PATH=/mnt/oss_bucket"
    - "./gitlab-ci-scripts/deploy/mount_ossfs.sh"
    - "./gitlab-ci-scripts/deploy/build_certman_infra.sh"
    - "umount $BUCKET_LOCAL_PATH"
    - "sleep 10"
  only:
    - master
    - pre-production
    - production
```

As you can see the `deploy_apps` stage has 2 jobs: `deploy_webapp` and `deploy_certman`. We did not change the scripts, just execute them in parallel.

Save the modifications and quit with CTRL + X.

Before we commit we need to modify the GitLab Runner configuration to allow it to run multiple jobs at the same time:

1. Log on to the ECS console.
2. Click **Instance** from the left-side navigation pane.

3. Select your region if necessary.
4. Search for your instance named **devops-simple-app-gitlab-runner**.
5. Click **Connect** on the right side of your instance.
6. The VNC console should appear. Authenticate yourself with the root user and the password you set when you configured GitLab.
7. Edit the GitLab Runner configuration file with this command:

```
nano /etc/gitlab-runner/config.toml
```

8. Edit the GitLab Runner configuration file with this command: nano /etc/gitlab-runner/config.toml
9. Save and quit by pressing CTRL + X.
10. Restart the GitLab Runner via the following command:

```
gitlab-runner restart
```

11. Quit the VNC session by entering the command `exit` and by closing the web browser tab.

Go back to your terminal and commit the changes to GitLab:

```
# Check the files to commit
git status

# Add the modified file
git add .gitlab-ci.yml

# Commit and push to GitLab
git commit -m "Parallelize deployment."
git push origin master
```

This time the GitLab pipeline contains 4 stages with 2 parallels jobs for the last one:

As usual, you can now merge the master branch to pre-production, and then pre-production to production.

16.5. Getting started with DevOps with Kubernetes

Introduction

The goal of this tutorial is to explain how to create a CI/CD pipeline to deploy an application in Kubernetes running on top of Alibaba Cloud.

The procedure can be summarized in two mains steps:

1. Installing the tooling environment (Gitlab and Kubernetes).

2. Creating a small Java web application and configuring a CI/CD pipeline around it.

Prerequisite

The very first step is to create an Alibaba Cloud account and obtain an AccessKey ID and Secret.

Cloud resources are created with Terraform scripts. If you do not know this tool, follow this tutorial and familiarize yourself with the alicloud provider.

Make sure you are familiarized with Kubernetes. If you need, you can follow this awesome tutorial to learn the basics. You will also need to setup the command line tool kubectl.

You should also have Git installed on your computer.

Preparation

Please download the related resources on your computer by cloning this Git repository. Open a terminal and enter the following commands:

```
# Navigate to a folder where you want to clone this tutorial
cd ~/projects

# Clone this repository
git clone git@github.com:alibabacloud-howto/devops.git

# Navigate to the folder of this tutorial
cd devops/tutorials/getting_started_with_devops_with_kubernetes/
```

Gitlab environment

This tutorial uses Gitlab to manage Git repositories and to run CI/CD pipelines. The community edition is free, simple to use and have all the features that we need for this demo.

Open a terminal and enter the following commands with your own AccessKey and region information:

```
export ALICLOUD_ACCESS_KEY="your-accesskey-id"
export ALICLOUD_SECRET_KEY="your-accesskey-secret"
export ALICLOUD_REGION="your-region-id"

cd environment/gitlab
terraform init
terraform apply -var 'gitlab_instance_password=YourSecretR00tPassword'
```

Note This script is a bit too simple to be used in production, for example, an SSL certificate should be configured to allow HTTPS. Here is a more complete tutorial about Gitlab installation.

The output of the script should contain the IP addresses of the newly installed Gitlab instance and a Gitlab runner:

```
Outputs:


gitlab_runner_public_ip = w.x.y.z
gitlab_public_ip = a.b.c.d
```

Open the page `http://a.b.c.d` (from `gitlab_public_ip`) in your web browser and:

1. Set a new password.
2. Sign in as root (with your new password).

You should be able to see a welcome screen. You can now generate and upload an SSH key:

1. Click your user's avatar on the top-right of the page and select **Settings**.
2. In the left-side navigation pane, select **SSH Keys**.
3. If necessary, generate your SSH key as instructed.
4. Copy your public key in the textarea (it should be in the file **~/.ssh/id_rsa.pub**).
5. Click **Add key**.

The next step is to register the Gitlab runner (the ECS instance that runs CI/CD scripts):

1. In the top menu, select **Admin area** (the wrench icon).
2. In the left-side navigation pane, select **Overview > Runners**.
3. The new page must provide a URL and a token under the section **Setup a shared Runner manually**. Keep this page opened, open a terminal, and run the following commands:

   ```
   ssh root@w.x.y.z # The IP address is from `gitlab_runner_public_ip`. The password is the one you set with `gitlab_instance_password`.
   gitlab-runner register
   ```

   Enter the URL and the token from the web browser tab, set **docker-runner** as the description, choose **docker** as executor and set **alpine:latest** as the default Docker image.

   > **Note** Do not set any tag, or your GitLab runner will not execute your jobs.

4. Refresh the web browser tab and check the runner is displayed.

Go back to the home page by clicking on the Gitlab icon on the top-left side of the screen and keep this web browser tab opened, we will return to it later.

Kubernetes environment

To keep it simple, this tutorial creates a single-AZ cluster. However, a multi-AZ one is preferred for production. Read the official documentation for more information.

Open a terminal and enter the following commands with your own AccessKey and region information:

```
export ALICLOUD_ACCESS_KEY="your-accesskey-id"
export ALICLOUD_SECRET_KEY="your-accesskey-secret"
export ALICLOUD_REGION="your-region-id"

cd environment/kubernetes
terraform init
terraform apply -var 'k8s_password=YourSecretR00tPassword'
```

> **Note** It takes about 20 minutes to create a Kubernetes cluster.

The output of the script should contain the master node public IP address:

```
Outputs:

k8s_master_public_ip = e.f.g.h
```

Execute the following commands to configure **kubectl** (the password is the one you set with the variable `k8s_password`):

```
mkdir $HOME/.kube
scp root@e.f.g.h:/etc/kubernetes/kube.conf $HOME/.kube/config # The IP address is the one from `k8s_master_public_ip`

# Check that it works
kubectl cluster-info
```

If the configuration went well, the result of the last command should be something like:

```
Kubernetes master is running at https://161.117.97.242:6443
Heapster is running at https://161.117.97.242:6443/api/v1/namespaces/kube-system/services/heapster/proxy
KubeDNS is running at https://161.117.97.242:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
monitoring-influxdb is running at https://161.117.97.242:6443/api/v1/namespaces/kube-system/services/monitoring-influxdb/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

Container registry

Unfortunately it is not yet possible to create a container registry on Alibaba Cloud through Terraform. Instead, this step must be done manually through the web console:

1. Log on to the Container Registry console.
2. In the left-side navigation pane, select **Products** > **Container Registry**.
3. Select your region on top of the page.
4. Click **Namespace** from the left-side navigation pane.
5. Click **Create Namespace**, set a name such as **cicd-k8s-tutorial** and click **Confirm**.
6. Click **Repositories** from the left-side navigation pane.
7. If you do not remember it, you can click **Reset Docker Login Password**.

Keep this page opened in the web browser, we will need to create a repository in the next step.

CI/CD Pipeline

This tutorial uses a very simple Spring Boot app as an example. You can find the source code in the folder `app/simple-rest-service`.

Docker image repository

The first step is to create a repository where Docker images will be saved. Open you web browser tab from the Container registry section and execute the following instructions:

1. Click **Create Repository**, select your namespace, set the repository name to **simple-rest-service**, set a summary, set the type as **Public**, click **Next**, select **Local Repository** as Code Source, and click **Create Repository**.
2. You should now see a list of repositories. Click **Manage** for your new repository.
3. On the new page, copy the **Internet** repository address and keep it on the side for the moment.

Gitlab project

Open the web browser tab you created in the Gitlab environment section and create a new project:

1. From the home page, click **Create a project**.
2. Set the name **simple-rest-service** and click **Create project**.
3. Once the project is created, in the left-side navigation pane, select **Settings** > **CI/CD**.
4. Expand the **Variables** panel, and create the following variables:
   - DOCKER_REGISTRY_IMAGE_URL = **Internet** repository address you got from the Docker image repository section
   - DOCKER_REGISTRY_USERNAME = Alibaba Cloud account username
   - DOCKER_REGISTRY_PASSWORD = Docker Login Password you might have reset in the Container registry section
   - K8S_MASTER_PUBLIC_IP = The Kubernetes cluster public IP (from `k8s_master_public_ip`)
   - K8S_PASSWORD = The Kubernetes password (from `k8s_password`)

The project repository is now ready to host files:

1. Open a terminal on your local machine and type:

```
mkdir -p $HOME/projects
cd $HOME/projects
git clone git@a.b.c.d:root/simple-rest-service.git # The IP address comes from `gitlab_public_ip`
cd simple-rest-service
```

2. Copy the following files from **app/simple-rest-service** into the new folder **$HOME/projects/simple-rest-service**:
   - src - Sample application source code
   - pom.xml - Maven project descriptor (declares dependencies and packaging information for the application)
   - deployment.yml - Kubernetes deployment descriptor (describes the deployment and a load balancer service)
   - .gitlab-ci.yml - CI/CD descriptor (used by Gitlab to create a pipeline)
3. In your terminal, type:

```
git add .gitlab-ci.yml deployment.yml pom.xml src/
git commit -m "Initial commit"
git push origin master
```

> **Note** If you have an error when you try to push your code, it may be due to the fact that the master branch is automatically protected. In this case, go to the Gitlab web browser tab, select **Settings > Repository > protected Branches**, type `master` in the Branch attribute, and select **Create Wildcard Master**.

Gitlab automatically recognizes the file **.gitlab-ci.yml** and create a pipeline with 3 steps:

1. Compile and execute unit tests.
2. Create the Docker image with JIB and upload it to the Docker image repository.
3. Deploy the application in Kubernetes.

You can see the pipeline in Gitlab by selecting **CI/CD > Pipelines** from the left-side navigation pane.

After you pipeline has been executed completely, you can check you Kubernetes cluster with the following commands:

- Check the deployments:

```
kubectl get deployments
```

The result should be something like:

```
NAME              DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
simple-rest-service   2         2         2            2           21m
```

- Check the pods:

```
kubectl get pods
```

The result should be something like:

```
NAME                      READY   STATUS    RESTARTS   AGE
simple-rest-service-5b9c496d5d-5n6vl   1/1     Running   0          18m
simple-rest-service-5b9c496d5d-h758n   1/1     Running   0          18m
```

- Check the services:

```
kubectl get services
```

The result should be something like:

```
NAME                 TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)        AGE
simple-rest-service-svc   LoadBalancer   10.1.214.231   161.117.73.86   80:30571/TCP   23m
```

- Check the logs of one pod:

```
kubectl logs simple-rest-service-5b9c496d5d-5n6vl
```

The result should start with:

```
.   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.0.5.RELEASE)
```

Test the application by yourself:

1. Open a new tab in your web browser and visit http://161.117.73.86 (the service EXTERNAL-IP). You should see **Hello world!**
2. Add `?name=Seven` to the URL (so it should be something like **http://161.117.73.86?name=Seven**). You should see the **Hello Seven!**

16.6. Getting started with Rancher

Introduction

Rancher is a multi-cluster Kubernetes management platform. The goal of this tutorial is to explain how to set up Rancher on a single node and how to integrate it with Alibaba Cloud Container Service.

Prerequisites

To follow this topic, you need to create an Alibaba Cloud account and obtain an AccessKey ID and Secret.

Cloud resources are created with Terraform scripts. If you do not know this tool, follow this topic and familiarize yourself with the Alicloud Provider.

You are familiarized with Kubernetes. You can follow this tutorial to learn the basics. You need to set up the command line tool kubectl.

The related resources are downloaded.

Rancher installation

There are two ways to set up Rancher:

- Single-node configuration.
- High-Availability configuration.

We will choose the first way as it makes things simpler.

Open a terminal on your computer and execute the following instructions:

```
# Go to the folder where you have downloaded this tutorial
cd path/to/this/tutorial

# Go to the Rancher environment folder
cd environment/rancher

# Download the latest stable version of the Alibaba Cloud provider
terraform init

# Configure the Alibaba Cloud provider
export ALICLOUD_ACCESS_KEY="your-accesskey-id"
export ALICLOUD_SECRET_KEY="your-accesskey-secret"
export ALICLOUD_REGION="your-region-id"

# Configure variables for the Terraform scripts
export TF_VAR_ecs_root_password="YourR00tP@ssword"

# Create the resources in the cloud
terraform apply
```

The last command should ask you to confirm by entering `yes` and should print logs that end like this:

```
Apply complete! Resources: 9 added, 0 changed, 0 destroyed.

Outputs:

rancher_eip_ip_address = 161.117.4.26
```

Open a web browser tab and enter the URL corresponding to https://rancher_eip_ip_address (for example, https://161.117.4.26/). Your web browser will complain that the connection is unsecured (which is normal because we did not configure any SSL/TLS certificate). Make an exception and continue browsing.

> Note If using an invalid certificate bothers you, follow this documentation to set up HTTPS.

The following welcome page is displayed.

Set an administrator password and click **Continue**. Keep the default value of the server URL and click **Save URL**.

The **Clusters** page is displayed.

Kubernetes cluster

Unfortunately the integration with Alibaba Cloud Container Service is not yet supported by the current version of Rancher (v2.1.3). However, we can create a Kubernetes cluster with Terraform and import it manually to Rancher.

Cluster sizing

Before creating our cluster we need to size it correctly. Currently in Alibaba Cloud, a Kubernetes cluster must have exactly 3 master nodes, but the node instance types (number of CPUs and amount of RAM) and the number of worker nodes are flexible.

> Note This document is a good introduction about the master and worker node concepts in Kubernetes.

This document in Chinese gives advices about which instance type to choose for master nodes; it also provides general tips about cluster administration. Concerning our sizing problem, this article proposes the following configurations:

- 1-5 worker nodes, master specification: 4 vCPUs and 8 GB of RAM
- 6-20 worker nodes, master specification: 4 vCPUs and 16 GB of RAM
- 21-100 worker nodes, master specification: 8 vCPUs and 32 GB of RAM
- 100-200 worker nodes, master specification: 16 vCPUs and 64 GB of RAM

According to the same article, the disk size for each master node does not need to be large, as it mainly contains the OS (about 3 GB), docker images, system and application logs, temporary data, and so on.

This second document in Chinese explains how to choose the number and the type of workers nodes. It also provides information about network driver, disk size selection and other management tasks.

The first important advice this topic provides is to prefer few large workers instead of many small ones:

- A small number of large workers increases the chance of having interdependent containers running on the same machine, which greatly reduces network transmission.
- Large resources (such as network bandwidth or physical RAM) concentrated on few nodes allow better resource utilization. For example if two applications need 1 GB of RAM, it is better

to collocate them on one worker with 3 GB of physical RAM instead of distributing them on two workers with 1.5 GB of physical RAM each; in the first case the large worker is able to accept a third application that would also need 1 GB of RAM, whereas the two small workers cannot.

- Pulling Docker images is more efficient on a smaller number of workers, because images are downloaded, stored on the local disk, and then re-used between containers.

However a too small number of workers is not a good idea, because a system should continue to function even if a worker node is down. The exact number of workers depends on the total number of required vCPUs and on the acceptable fault tolerance.

Let's consider the following example where a system needs a total of 160 vCPUs:

- If the fault tolerance is 10%, we cannot lose more than 16 vCPUs, so a valid configuration is 10 workers with 16 vCPUs.
- If the fault tolerance is 20%, we cannot lose more than 32 vCPUs, so a valid configuration is 5 workers with 32 vCPUs.

About the amount of RAM for each worker, the document gives the following rule of thumb in case of applications that are relatively greedy in memory, such as Java applications: a good ratio is 8 GB of RAM per vCPU, so if we choose an instance type with 4 vCPUs, then we need to take about 32 GB of RAM.

Cluster creation

We will create a Kubernetes cluster in multiple availability zones. This decision increases the availability of the system, but it adds the following constraints:

- Alibaba Cloud Container Service is designed to support either 1 or 3 availability zones.
- To be compatible with most of the regions, we can only use 2 availability zones, so we will need to configure our Kubernetes cluster to use twice the same availability zone.
- The minimum number of worker nodes is 3.

Open a terminal on your computer and execute the following instructions:

```
# Go to the folder where you have downloaded this tutorial
cd path/to/this/tutorial

# Go to the Rancher environment folder
cd environment/kubernetes-cluster

# Download the latest stable version of the Alibaba Cloud provider
terraform init

# Configure the Alibaba Cloud provider
export ALICLOUD_ACCESS_KEY="your-accesskey-id"
export ALICLOUD_SECRET_KEY="your-accesskey-secret"
export ALICLOUD_REGION="your-region-id"

# Configure variables for the Terraform scripts
export TF_VAR_ecs_root_password="YourR00tP@ssword"
export TF_VAR_master_instance_cpu_count=4
export TF_VAR_master_instance_ram_amount=8 # in GB
export TF_VAR_master_instance_disk_size=40 # in GB
export TF_VAR_worker_instance_count=3 # Must be >= 3
export TF_VAR_worker_instance_cpu_count=4
export TF_VAR_worker_instance_ram_amount=32 # in GB
export TF_VAR_worker_instance_disk_size=80 # in GB

# Create the resources in the cloud
terraform apply
```

The last command should ask you to confirm by entering `yes` and should end with similar logs:

```
Apply complete! Resources: 16 added, 0 changed, 0 destroyed.

Outputs:

rancher_k8s_cluster_ip_address = 161.117.96.245
```

> **Note** Do not worry if the operation takes some time. Creating a cluster typically takes about 15 minutes.

Let's configure kubectl locally so that it can communicate with the new cluster. Execute the following commands in your terminal:

```
mkdir $HOME/.kube
scp root@161.117.96.245:/etc/kubernetes/kube.conf $HOME/.kube/config
# Note 0: the IP address is the one from `rancher_k8s_cluster_ip_address`.
# Note 1: the password is the one that was set in `TF_VAR_ecs_root_password`.

# Check that it worked
kubectl cluster-info
```

If the configuration went well, the result of the last command should be something like:

```
Kubernetes master is running at https://161.117.96.245:6443
Heapster is running at https://161.117.96.245:6443/api/v1/namespaces/kube-system/services/heapster/proxy
KubeDNS is running at https://161.117.96.245:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
monitoring-influxdb is running at https://161.117.96.245:6443/api/v1/namespaces/kube-system/services/monitoring-influxdb/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

Importing the Kubernetes Cluster into Rancher

Now that we have our Kubernetes Cluster, let's import it into Rancher so that we can manage it from there.

Open your web browser tab with Rancher (the page you got when you finished the Rancher installation section) and follow these instructions:

1. The current page must be the **Clusters** one. Click **Add Cluster**.
2. Select **Import existing cluster**.
3. Set the **Cluster Name** field to **alibabacloud-cluster**.
4. Click **Create**.

You should get a page like this:

Let's execute the last command. Copy it and paste it in your terminal:

```
# Command from Rancher in order to import the cluster
curl --insecure -sfL https://161.117.4.26/v3/import/nlz588gctmkkkpc8jsntrht8ff65gbp4d629smqzbcjpvxzltfdmph.yaml | kubectl apply -f -
```

This command should output the following logs:

```
namespace/cattle-system created
serviceaccount/cattle created
clusterrolebinding.rbac.authorization.k8s.io/cattle-admin-binding created
secret/cattle-credentials-1d26caa created
clusterrole.rbac.authorization.k8s.io/cattle-admin created
deployment.extensions/cattle-cluster-agent created
daemonset.extensions/cattle-node-agent created
```

Go back to the web browser tab and click **Done**. You should now see your cluster:

On this page, click the cluster name (alibabacloud-cluster). You should obtain a dashboard similar to this one:

Testing

Let's play a bit with Rancher by deploying a small application. Open your web browser tab with the cluster dashboard and execute the following actions:

1. In the top menu, select **Projects/Namespaces**.
2. This page displays two projects **Default** and **System**. Click **Default**.
3. The new page displays the workloads, but it is empty for the moment. Click **Deploy**.
4. In this form, set the fields like this:
   - Name = hello-workload
   - Docker Image = rancher/hello-world
5. Scroll down and click **Show advanced options**.
6. Expand **Labels & Annotations** and click **Add Label**.
7. Set the key app and the value hello-app.
8. Click **Launch**.

After few seconds you should see your workload named **hello-workload** with the **Active** status:

Let's create a load balancer to expose this application to internet:

1. Click **Import YAML**.
2. Copy the following content in the dark text area:

```
apiVersion: v1
kind: Service
metadata:
 name: hello-app-load-balancer
 labels:
  app: hello-app
spec:
 type: LoadBalancer
 ports:
 - port: 80
  protocol: TCP
  targetPort: 80
 selector:
  app: hello-app
```

3. Click **Import**.
4. Click **Load Balancing**.

You can see your load balancer:

To get the IP address of this load balancer, click the menu on the right of **hello-app-load-balancer** (with 3 vertical dots) and select **View/Edit YAML**. You can see a large YAML file. Scroll down until you see the `status` :

```
status:
 loadBalancer:
  ingress:
  - ip: 161.117.96.212
```

Copy this IP address and paste it into the URL bar of a new web browser tab. You can access to your application:

Congratulation if you managed to get this far! If you want to continue to learn about Kubernetes and Rancher, see the official documentation.

17.Disaster recovery solutions

Disaster recovery solutions help ensure the running stability and security of your services and IT systems by incorporating data backup and disaster recovery. Alibaba Cloud ECS allows you to use snapshots and images to back up data.

Disaster recovery methods

- Snapshot backup

  Alibaba Cloud ECS allows you to back up system disks and data disks with snapshots. Alibaba Cloud provides the Snapshot 2.0 service, which features a higher snapshot quota and a more flexible automatic task strategy than previous snapshot services, to help reduce the impacts on business I/O. When snapshots are used for data backup, the first snapshot of a disk is a full backup, and subsequent snapshots are incremental backups. Incremental snapshots can be created quickly and have small sizes. The amount of time required for backup depends on the amount of incremental data to be backed up.

  > **Note** Snapshots are created on an incremental basis. To improve backup speed, we recommend that you create a new snapshot before deleting the most recent one.

  The preceding figure shows how incremental snapshots work. In the figure, Snapshots 1, 2, and 3 represent the first, second, and third snapshot of a disk. The file system checks the disk data block by block. When a snapshot is created, only the blocks with changed data are copied to the snapshot. Alibaba Cloud ECS allows you to configure manual or automatic snapshots of disks. To create automatic snapshots of a disk, you can configure and apply an automatic snapshot policy to the disk. You can specify the hour of the day (on the hour), day of week (Monday through Sunday), and retention period for snapshot creation in the policy. You can customize the retention period to a value from 1 to 65,536 days, or choose to save snapshots permanently.

- Snapshot rollback

  When exceptions occur in your system and you want to roll a disk back to a previous state, you can roll the disk back to a created snapshot. For more information, see Roll back a disk by using a snapshot. Note the following points:

  - Rollback operations are irreversible. After a rollback is complete, data before the rollback cannot be restored. Exercise caution when you perform this operation.
  - When a disk is rolled back, all data created or modified between the current time and the snapshot creation time is lost.

- Image backup

  An image works as a copy that stores data from one or more disks. An ECS image may store data from a system disk or from both system and data disks. All image backups are full backups and can only be triggered manually.

- Image recovery

You can create a custom image from a snapshot to include the operating system and data environment of the snapshot in the image. Then, you can use the custom image to create multiple instances with the same operating system and data environment. For more information about the configuration of snapshots and images, see Create a normal snapshot and Create a custom image from a snapshot.

> Note Custom images cannot be used across regions.

Technical metrics

RTO and RPO are related to the amount of data, typically on an hourly basis.

Scenarios

- Backup and recovery

  Alibaba Cloud ECS allows you to back up system disks and data disks with snapshots and images. If incorrect data is stored on a disk due to application errors or hackers' malicious access through application vulnerabilities, you can use the snapshot service to restore the disk to a desired state. In addition, Alibaba Cloud ECS allows you to reinitialize disks with images or create ECS instances from custom images.

- Disaster recovery

  Alibaba Cloud ECS supports the implementation of disaster recovery architecture. For example, you can buy and use an SLB instance at the frontend of an application, and deploy at least two ECS instances at the backend of the same application. Alternatively, you can use Auto Scaling provided by Alibaba Cloud to perform auto scaling by defining how to use ECS resources. This way, even if one of the ECS instances fails or is overloaded, disaster recovery can be implemented to ensure business continuity. The following figure provides an example in which ECS instances are deployed in data centers in two zones within the same region. All communications are implemented in the Alibaba Cloud Gigabit internal network to ensure fast response and reduce Internet traffic costs.

  - SLB: SLB instances are used for load balancing between the two zones. Traffic is distributed to two or more data centers where ECS instance clusters are deployed.
  - ECS cluster: ECS instances deployed in the two data centers are equivalent. The failure of a single instance does not affect data layer applications and the ECS control function. If a failure occurs, the system automatically performs hot migration so that other ECS instances can continue to provide services. This can prevent service interruptions caused by a single point of failure or hot migration failures. If hot migration fails, you will receive a notification about the failures based on system events so that you can deploy new nodes in a timely manner.
  - Data layer: OSS is deployed at the region level. ECS nodes in data centers in different zones can access objects in OSS. For database applications, multi-zone ApsaraDB for RDS service is used. Primary nodes can perform read and write operations across zones without conflicting with application-layer traffic. In addition, secondary nodes can perform read operations across zones to prevent inability of ECS instances to read data in case of failures of the primary nodes.

18.Deploy a highly available architecture

18.1. Deploy a highly available architecture

Highly available architectures provide functions such as service distribution, auto scaling, and multi-zone deployment. Compared with a single ECS instance, a highly available architecture is more stable and scalable when databases and applications are deployed.

Features

Highly available architectures have the following features:

- A multi-zone, highly available SLB instance distributes traffic to multiple ECS instances, increasing the external service capability of application systems, eliminating single point of failures, and improving the availability of application systems. SLB is used for automatic multi-zone deployment, enhancing disaster recovery capabilities of services.
- You can use custom images to create identical ECS instances, and then add these instances to the backend server group of the SLB instance, implementing high availability for your services. SLB can be configured with Layer-4 and Layer-7 listeners at the same time, and with multiple algorithms, such as round robin, weighted round robin, and weighted least connections, properly allocating computing resources to backend ECS instances.
- Relational Database Service (RDS) can be optimized for high concurrency scenarios, ensuring the constant stability and high throughput of the system through thread pools, parallel replication, and hidden primary keys. CloudDBA provides comprehensive performance monitoring metrics to monitor the usage of instances and hardware and slow SQL queries in real time, and gives optimization suggestions to help you locate and solve problems.

Deployment process

If you have created an ECS instance and deployed databases and applications on the instance, you can change the single-instance deployment mode to single-zone or multi-zone, highly available architecture. This topic shows you how to use ECS, EIP, SLB, and RDS to deploy a multi-zone, highly available architecture.

1. Use a custom image to create multiple identical ECS instances. For more information, see Replicate ECS instances.
2. Create an SLB instance and add the ECS instances to the SLB backend server group to mount ECS instances from different zones, achieving the high availability of services. For more information, see Configure an SLB instance.
3. Use DTS to migrate a user-created database from an ECS instance to an RDS instance, ensuring that the business database is not interrupted during migration and data is automatically backed up. For more information, see Migrate user-created databases to RDS instances.

18.2. Replicate ECS instances

This topic describes how to use a custom image created from a source instance to create three ECS instances for multi-zone disaster recovery. One instance is assigned to the same zone as the source instance, and the other two instances are assigned to a different zone in the same region as the source instance.

Prerequisites

- An Alibaba Cloud account is created. To create an Alibaba Cloud account, go to the account registration page.
- You have an ECS instance to be replicated.

Procedure

1. Create a custom image from an ECS instance.
   i. Log on to the ECS console.
   ii. In the left-side navigation pane, choose **Instances & Images > Instances**.
   iii. In the top navigation bar, select a region.
   iv. Find the instance from which you want to create a custom image. In the **Actions** column, choose **More > Disk and Image > Create Custom Image**.
   v. Enter an image name and description.
   vi. Click **Create**.

   > **Note** Image creation may take a while.

   In the left-side navigation pane, choose **Instances & Images > Images**. When the progress of the image reaches 100% in the **Progress** column and the status becomes **Available**, the image is created.

2. Create three ECS instances from the created custom image.
   i. In the left-side navigation pane, choose **Instances & Images > Images**.
   ii. On the **Custom Images** tab of the Images page, find the custom image created in the previous step. Click **Create Instance** in the **Actions** column corresponding to the image.
   iii. On the **Custom Launch** tab, scroll down to the **Image** section. The custom image you selected is automatically selected. Continue with other settings as prompted and set Quantity to 1.where:
      - **Region**: Select the same region as the source instance.
      - **Zone**: Select the same zone as the source instance.
      - **Public IP Address**: Clear **Assign Public IP Address**.

      For more information, see Create an instance by using the provided wizard.

iv. Repeat Step a and Step b. On the **Custom Launch** tab, scroll down to the **Image** section. The custom image you selected is automatically selected. Continue with other settings as prompted and set Quantity to 2.where:

- **Region**: Select the same region as the source instance.
- **Zone**: Select a zone different from that of the source instance.
- **Instance Type**: Set **Quantity** to 2.
- **Public IP Address**: Clear **Assign Public IP Address**.

For more information, see Create an instance by using the provided wizard.

Result

In the left-side navigation pane, choose **Instances & Images** > **Instances**. On the **Instances** page, the four ECS instances are in the **Running** state. Two instances are located in a zone, and the other two are located in another zone.

What's next

Configure an SLB instance

18.3. Configure an SLB instance

After the ECS instances are replicated, you can create a multi-zone SLB instance in a region that supports multiple zones and bind multi-zone ECS instances to the SLB instance. This task can extend the external service capabilities of application systems, eliminate single point of failures, and improve the availability of application systems. This topic describes how to deploy an SLB instance.

Prerequisites

- Three ECS instances are replicated. For more information, see Replicate ECS instances.
- The web services of the four ECS instances are started and are running normally.

> **Notice** If the web services are not running, the SLB instance and the ECS instances cannot communicate normally.

Procedure

1. Create an SLB instance. For more information, see Create an SLB instance. The following settings are used in this topic:
   - **Region**: Select the same region as the ECS instances.
   - **Zone Type**: Select **Multi-zone**.
   - **Instance Type**: Select **Internal Network**.
   - **Network Type**: Select **VPC**.
   - **Primary Zone** and **Backup Zone**: Configure as needed.
2. Convert the public IP address of the source instance into an Elastic IP Address. For more information, see Convert the public IP address of a VPC-type instance to an Elastic IP address.

> **Note** The IP address of the source instance must remain unchanged so that services are not affected. Therefore, you must first convert the public IP address of the source instance into an Elastic IP Address, unbind the Elastic IP Address from the source instance, and then bind the Elastic IP Address to the multi-zone SLB instance.

3. Unbind the Elastic IP Address from the source instance.
   i. In the **IP Address** column of the source instance, click the link of the Elastic IP Address.
   ii. On the **Elastic IP Addresses** page, click **Unbind**.

   
   iii. Click **OK**. For more information, see Disassociate an EIP from a cloud resource.
4. Bind the Elastic IP Address to the SLB instance.
   i. On the **Elastic IP Addresses** page, find the Elastic IP Address that was unbound from the source instance.

   
   ii. In the **Actions** column, click **Bind**.
   iii. Select **SLB Instance** for **Instance Type**, select the SLB instance that you created for **SLB Instance**, and then click **OK**. For more information, see Associate an EIP with an SLB instance.
5. Configure an SLB instance. For more information, see Configure an SLB instance. Perform the following steps to complete the basic settings:
   i. On the **Protocol and Listener** tab, complete the following configuration:
      - **Select Listener Protocol**: Select **TCP**.
      - **Listening Port**: Enter `80`.
      - **Scheduling Algorithm**: Set this parameter as needed. In this topic, Scheduling Algorithm is set to **Round-Robin (RR)**.
      - Use the default values for other settings.

      
   ii. Click **Next**. On the **Backend Servers** tab, select **Default Server Group**, and click **Add More** to add the ECS instance.

      
   iii. Select the source instance and the three replicated ECS instances, and click **Next: Set Weight and Port**. Set **Port** to 80 and remain the default values for other settings. Click **Next**.
   iv. On the **Health Check** tab, use the default values and then click **Next**.
   v. On the **Submit** tab, verify the information and click **Submit**.

vi. Click **OK** to go back to the **Server Load Balancer** page, and click . If the health check is **Normal**, the backend ECS instance is working properly and able to accept requests.

> **Note** It takes a few minutes to perform the health check. Wait and click the refresh icon to view the status.

Result

In this topic, a static web page is built on each of the four ECS instances to identify each instance. Enter the endpoint of the SLB instance in the browser to test whether the SLB is working properly. Because **Scheduling Algorithm** was set to **Round-Robin (RR)**, requests are sent to each ECS instance in turn.

What's next

Migrate user-created databases to RDS instances

18.4. Migrate user-created databases to RDS instances

You can migrate databases from ECS to ApsaraDB for RDS High-availability Edition to ensure high availability, reliability, security, and convenience of database services. This topic uses a MySQL database as an example to describe how to use DTS to migrate a user-created database from an ECS instance to an RDS instance.

Prerequisites

- An SLB instance is configured. For more information, see Configure an SLB instance.
- An RDS High-availability Edition instance is created. For more information about how to create an RDS instance, see Create an ApsaraDB RDS for MySQL instance.
- An account is created for the ApsaraDB for RDS instance. For more information, see Create accounts and databases for an ApsaraDB RDS for MySQL instance.
- Another account besides the root account is created for the user-created database hosted on the ECS instance. The account is used for data migration over Data Transmission Service (DTS).

  For example, you can run the following command to create an account with its username as dts and password as 123456 for a MySQL database:

```
grant all on *.* to 'dts'@'%' IDENTIFIED BY '123456';
```

Context

DTS supports data migration between homogeneous and heterogeneous data sources. It also supports ETL features such as data mapping at three levels (databases, tables, and columns) and data filtering. You can use DTS for zero-downtime data migration. During data migration, the source database continues to provide services normally, minimizing the impact of data migration on your business. For more information about the database types supported by DTS, see Data migration.

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Data Migration**.
3. Select the region where the ApsaraDB for RDS instance is located, and click **Create Migration Task**.
4. Configure the migration task.
   i. Configure a task name.

      You can use the default name or customize a name.

   ii. Configure the source database. DTS supports user-created databases that are accessible through the Internet, VPN Gateway, an Express Connect circuit, or Smart Access Gateway. The source database used in this topic is a user-created database in an ECS instance. For migration solutions for other types of databases, see DTS documentation.

Parameter	Description
Instance Type	A user-created database hosted on the ECS instance.
Instance Region	The region where the source ECS instance is located.
ECS Instance ID	The ID of the source ECS instance. DTS supports ECS instances in the classic network and VPCs.
Database Type	The type of user-created database hosted on the source ECS instance. The database type is MySQL in this example.
Port Number	The listening port number for the MySQL database.
Database Account	The non-root account to access the MySQL database hosted on the source ECS instance. Note The account must not be the root user. If the account is a root user, an error occurs when you test connectivity.
Database Password	The password of the account.

   iii. Click **Test Connectivity** in the lower-right corner of the **Source Database** section.

      If **Test Passed** is returned, the connection to the source database is normal.

iv. Configure the destination database.

Parameter	Description
Instance Type	An ApsaraDB for RDS instance.
Instance Region	The region where the ApsaraDB for RDS instance is located.
ApsaraDB for RDS Instance ID	The ID of the ApsaraDB for RDS instance.
Database Account	The account of the ApsaraDB for RDS instance. For more information, see Create accounts and databases for an ApsaraDB RDS for MySQL instance. Note The account must not be the root user. If the account is a root user, an error occurs when you test connectivity.
Database Password	The password of the account.

v. Click **Test Connectivity** in the lower-right corner of the **Destination Database** section.

If the **Test Passed** message is returned, the connection to the destination database is normal.

vi. Click **Set Whitelist and Next**.

5. Configure migration types and objects.

i. Configure migration types.

- To perform zero-downtime data migration, select Schema Migration, Full Data Migration, and Incremental Data Migration.
- To perform full data migration, select Schema Migration and Full Data Migration.

ii. Configure migration objects.

In the **Available** list, select the database objects to be migrated, such as databases, tables, or columns. Then click the > icon to add them to the **Selected** list.

Note

By default, the object names remain the same as those in the on-premises MySQL database after objects are migrated to a user-created MySQL database hosted on an ECS instance. If a migrated object has different names in the source and destination instances, you must use the object name mapping feature provided by DTS. For more information, see Object name mapping.

6. Click **Precheck**.

Before the migration task starts, DTS checks items such as connectivity, permissions, and log formats. The following figure shows that the precheck is successful.

If the precheck is successful, you can see the status and progress of the migration task in the **Migration Tasks** section.

What's next

Configure the endpoint, account, and password of the RDS instance in the application to remotely connect to the RDS instance. You can also use Data Management (DMS) or the client to manage the RDS instance. For more information, see Connect to an ApsaraDB RDS for MySQL instance.