

Alibaba Cloud Elastic Compute Service

Best Practices

Issue: 20190918

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 due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall 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 the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. 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 bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, 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 content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, 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 the Alibaba Cloud website, product programs, or content 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 contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information 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.
	This warning information 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 restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus, page names, and other UI elements.	Click OK.
Courier font	It is used for commands.	Run the <code>cd / d C :/ windows</code> command to enter the Windows system folder.
<i>Italics</i>	It is used for parameters and variables.	<code>bae log list --instanceid Instance_ID</code>
[] or [a b]	It indicates that it is an optional value, and only one item can be selected.	<code>ipconfig [-all -t]</code>

Style	Description	Example
<code>{}</code> or <code>{a b}</code>	It indicates that it is a required value, and only one item can be selected.	<code>switch {stand slave}</code>

Contents

Legal disclaimer.....	I
Generic conventions.....	I
1 Security.....	1
1.1 Best practices of the security group (part 1).....	1
1.2 Best practices of the security group (part 2).....	4
1.3 Best practices of the security group (part 3).....	10
1.4 ECS data security best practices.....	14
1.5 Configure instances to access each other in classic networks.....	17
1.6 Modify the default remote access port.....	24
1.7 Use logs in Windows instances.....	30
1.8 Overview and best practices of Windows Firewall with Advanced Security.....	32
1.9 Isolation of instances within a security group.....	51
1.10 Security group quintuple rules.....	53
1.11 Revoke the authorization for internal network communication between ECS instances for different accounts through the API.....	56
1.12 Authorize internal network communication between ECS instances for different accounts through the API.....	57
2 Disaster recovery solutions.....	61
3 Data recovery.....	64
3.1 How to restore the data that is deleted by mistake.....	64
3.2 Data restoration in Linux instances.....	68
3.3 Data restoration in Windows instances.....	77
4 Configuration preference.....	81
4.1 Transfer ECS instance data.....	81
4.2 Increase data throughput through read/write split.....	88
4.3 Change the language settings of a Windows instance.....	96
5 Packer: machine images as code.....	102
5.1 Comparison between creating a custom image through ECS against using the tool Packer.....	102
5.2 Configure DevOps parameters by using Packer.....	110
6 Use CloudMonitor to monitor ECS instances.....	115
7 Monitor.....	118
8 Access other Cloud Product APIs by the Instance RAM Role.....	119
9 GPU instances.....	125
9.1 Deploy an NGC on gn5 instances.....	125

9.2 Accelerate machine learning tasks on a GPU instance by using RAPIDS....	130
10 FaaS instances best practices.....	143
10.1 Use RTL compiler on an f1 instance.....	143
10.2 Use OpenCL on an f1 instance.....	147
10.3 Use OpenCL on an f3 instance.....	151
10.4 Best practices for RTL design on an f3 instance.....	162
10.5 faascmd tool.....	171
10.5.1 faascmd overview.....	171
10.5.2 Install faascmd.....	172
10.5.3 Configure faascmd.....	173
10.5.4 Use faascmd.....	173
10.5.5 FAQ.....	180
11 Shrink disk volume.....	186
12 Process ECS status change events.....	189
13 DevOps tutorials.....	198
14 DevOps for small and medium web apps.....	199
14.1 General introduction.....	199
14.2 Install and configure GitLab.....	200
14.3 Continuous integration.....	218
14.4 Code quality.....	228
14.5 Continuous delivery.....	246
14.6 HTTPS configuration.....	286
14.7 Log management.....	315
14.8 Speeding up CI and CD pipeline.....	329
15 Getting started with DevOps with Kubernetes.....	342
16 Getting started with Rancher.....	350

1 Security

1.1 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
", " SourceCidr Ip " : " 0 . 0 . 0 . 0 / 0 ", " Policy ": " accept
" } ,
{ " IpProtocol " : " tcp ", " FromPort " : " 8080 ", " ToPort " : "
8080 ", " SourceCidr Ip " : " 0 . 0 . 0 . 0 / 0 ", " Policy ": "
accept " } ,
```



```
{ " IpProtocol " : " tcp ", " FromPort " : " 443 ", " ToPort " : " 443 ", " SourceCidr Ip " : " 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 ", " SourceCidr Ip " : " 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 ", " SourceGroup Id " : " sg - web ", " Policy ": " accept ", 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",  
  " SourceCidr Ip " : " 10 . 0 . 0 . 0 / 24 ", " Priority " : 2 } ,  
{ " IpProtocol " : " tcp ", " FromPort " : " 0 ", " ToPort " : " 65535",  
  " SourceCidr Ip " : " 10 . 0 . 0 . 0 / 24 ", " Priority " : 2 } ,  
{ " IpProtocol " : " udp ", " FromPort " : " 0 ", " ToPort " : " 65535",  
  " SourceCidr Ip " : " 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 [#unique_9](#).

1.2 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 [Manage ECS instances elastically by using OpenAPI](#). 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_AcceptFormat ( ' 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_AcceptFormat ( ' 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 )
```

```
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.

1.3 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.

1. Sort out the ports and protocols exposed by the current Internet services, such as 80 and 443.
2. 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.

3. 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.

4. 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.
 5. 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.
 6. 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.

1.4 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.

We recommend that you [create a VPC](#), and then set the IP address range, network segments, route tables, and gateways of the VPC. You can store important data in the private network, logically isolated from the Internet. Then, you can use an EIP or the Jumper Server to manage data for daily O&M purposes.

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.

1.5 Configure instances to access each other in classic networks

A security group functions similar to a firewall for an instance. To ensure the security of an instance, the principle of least privilege should be observed when setting security group rules for it. This topic describes four recommended methods of enabling intranet intercommunication for instances by means of applying security groups.

Method 1. Authorize access to IP addresses

- **Application scenario:** Intercommunication of a small number of instances over an intranet.
- **Advantage:** Authorizing access to IP addresses makes the security group rules clear and easy to understand.
- **Disadvantage:** When a large number of instances need to access each other over the intranet, the quota of 100 security group rules will become a restriction. In addition, the maintenance workload will be high.

- **Configuration:**

1. Click the ID of the instance that requires intranet intercommunication, and click **Security Groups**.
2. Select the target security group and click **Add Rules**.
3. Click **Ingress** and then click **Add Security Group Rule**.
4. Add the following security group rules:
 - **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:** CIDR.
 - **Authorization Objects:** Enter the target intranet IP address for intranet intercommunication. The format must be *a.b.c.d/32*. Note that the subnet mask must be */32*.

Add Security Group Rule

NIC Type:

Internal Network

Rule Direction:

Inbound

Action:

Allow

Protocol Type:

Customized TCP

* Port Range:

Example: 22/22

Priority:

1

Authorization Type:

IPv4 CIDR Bloc

* Authorization Objects:

Example: 10.x.y.z/32. You can specify up to 10 authorization objects. Separate multiple authorization

Description:

It must be 2 to 256 characters in length and cannot start with "http://" or "https://".

OK

Cancel

Method 2. Join the same security group

- **Application scenario:** If your application architecture is relatively simple, you can add all of your instances to the same security group. Such instances then require no custom rules as they can access each other over the intranet by default.
- **Advantage:** Security group rules are clear and easy to understand.
- **Disadvantage:** It is only applicable to a simple network architecture. When the network architecture is adjusted, the authorization method should be modified accordingly.
- For more information, see [Add an instance to a security group](#).

Method 3. Add instances to a security group that is created solely for intercommunication

- **Application scenario:** You can add the target instances to a dedicated security group for intercommunication. This method is recommended for a network architecture with multiple layers of applications.
- **Advantage:** This method is easy to implement, and allows you to quickly establish interconnection between instances. Additionally, it is applicable to complicated network architectures.
- **Disadvantage:** The instances need to be added to multiple security groups and the security group rules may be complex.
- **Configuration:**
 1. Create a new security group with the name of “security group for intercommunication” . No rules are required for the new security group.
 2. Add the target instances to the newly created “security group for intercommunication” . The instances are then interconnected over the intranet by default.

Method 4. Security group authorization

- **Application scenario:** If your network architecture is complicated, and the applications deployed on different instances have different service roles, you can select security group authorization.
- **Advantage:** The security group rules are clear and easy to understand, and intercommunication can be implemented across accounts.
- **Disadvantage:** A large number of security group rules need to be correctly configured.

- **Configuration:**

1. Select the target instance, and enter the Security Groups page.
2. Select the target security group and click Add Rules.
3. Click Ingress, and then click Add Security Group Rule.
4. Add security group rules as described below:
 - **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:** Security Group.
 - **Authorization Objects:**
 - **Allow Current Account:** Based on your networking requirements, select the security group IDs of the peer instances for intranet intercommunication in Authorized Objects.
 - **Allow Other Accounts:** Enter the security group IDs of the peer instances in Authorized Objects. Enter the peer account ID in Account ID. You can query it in Account Management > Security Settings.

Add Security Group Rule

NIC Type:

Internal Network

Rule Direction:

Inbound

Action:

Allow

Protocol Type:

Customized TCP

* Port Range:

Example: 22/22

i

Priority:

1

i

Authorization Type:

Security Group

☒ Allow Current Account

☐ Allow Other Accounts

* Authorization Objects:

Select Security Group

Description:

It must be 2 to 256 characters in length and cannot start with "http://" or "https://".

OK

Cancel

Add Security Group Rule

NIC Type:

Internal Network

Rule Direction:

Inbound

Action:

Allow

Protocol Type:

Customized TCP

* Port Range:

Example: 22/22

i

Priority:

1

i

Authorization Type:

Security Group

☐ Allow Current Account

☒ Allow Other Accounts

* Authorization Objects:

sg-xxxxxxxxxxxxxxxxxxxxxxxx

* Account ID:

xxxxxxxxxxxxxxxxxxxxxxxx

Enter an account ID. To query your account ID, go to [Account Center](#)

Description:

It must be 2 to 256 characters in length and cannot start with "http://" or "https://".

OK

Cancel

Suggestions

If you determine that a disproportionate level of access is granted by the applied security group, we recommend that you reduce the scope of authorization according to the following procedure.



In the figure, Delete 0.0.0.0 means to delete the original security group rule that allows the inbound access from the 0.0.0.0/0 address segment.

If one or some security group rules are improperly configured, the communication between your instances may be affected. We recommend that you back up the security group rules you want to change before changing the settings so that you can easily recover the rules if an issue occurs.

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: Open port 80.
- Application layer security group: Open port 8080.
- DB layer security group: Open port 3306.

1.6 Modify the default remote access port

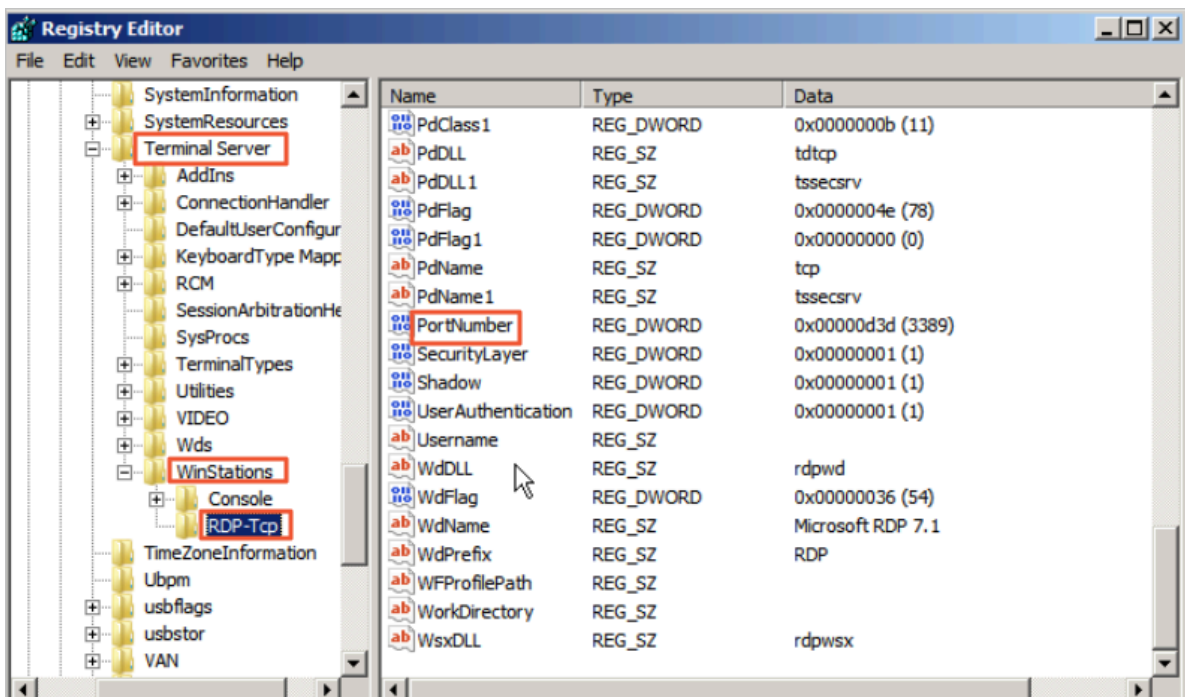
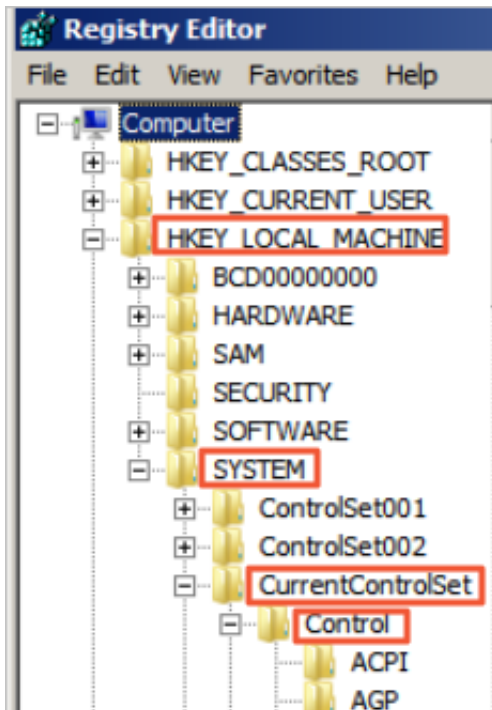
This topic describes how to modify the remote port of a Windows or Linux instance.

Modify the default remote port of a Windows instance

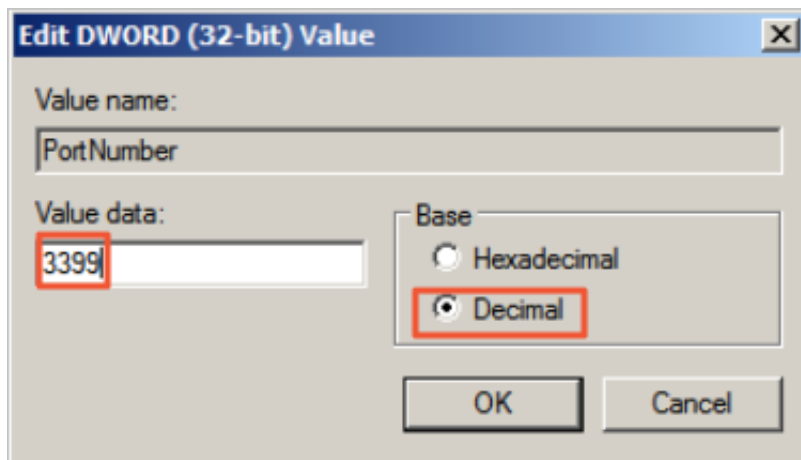
This section describes how to modify the remote port of a Windows instance running Windows Server 2008.

1. [Connect to the Windows instance.](#)
2. Run `regedit.exe` to open Registry Editor.

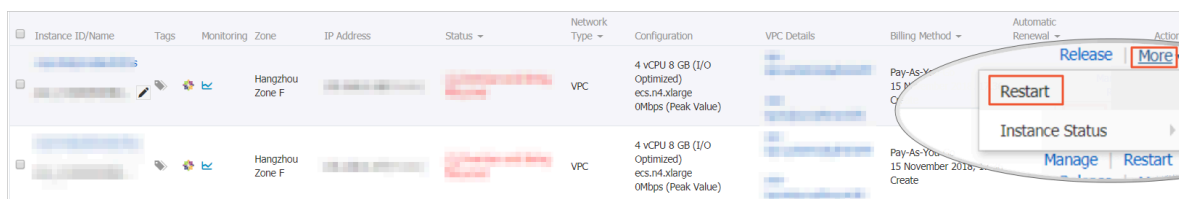
3. On the left-side navigation pane of the Registry Editor, find `HKEY_LOCAL_MACHINE \ System \ CurrentControlSet \ Control \ Terminal Server \ WinStations \ RDP - Tcp \ PortNumber`.



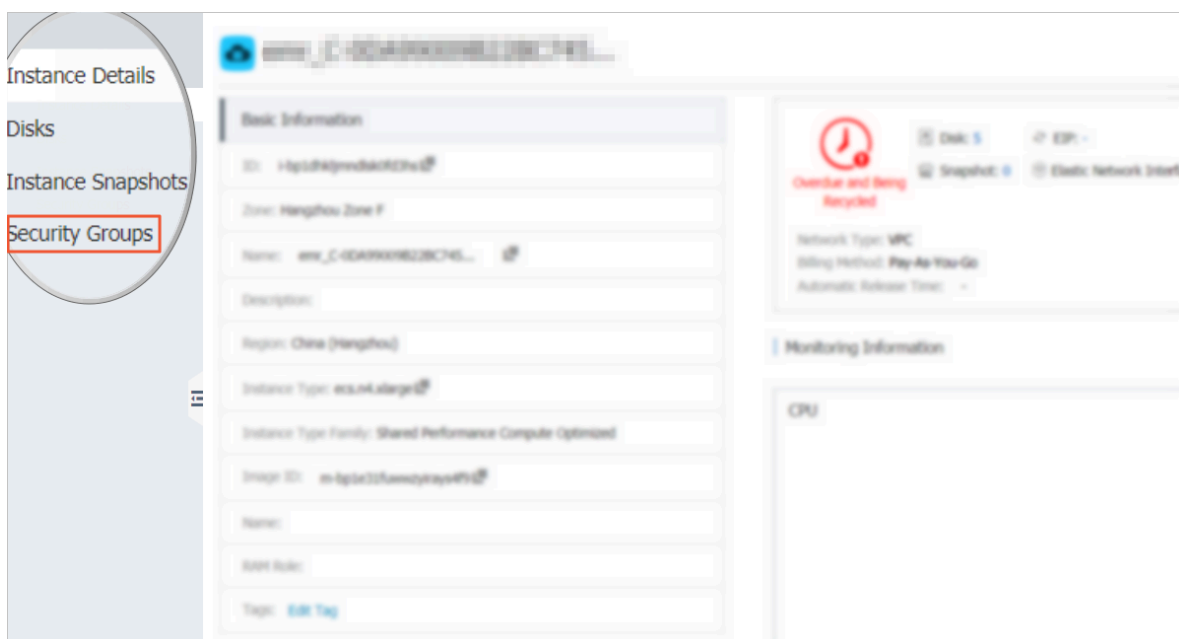
4. In the dialog box, select Decimal as Base, and then type a number in the Value data field as the new remote port number, which is 3399 in this example. Click OK.



5. (Optional) If you have enabled firewall, open the new port on the firewall.
6. Log on to the [ECS console](#), find the instance, and then select More > Restart.



7. After the instance is restarted, click the Manage of the instance to enter the Instance Details page. Click Security Groups.



8. On the Security Groups page, click Add Rules.

9. On the Security Group Rules page, click Add Security Group Rule. Add a new security group rule to allow access to the new remote port. For more information about adding security group rules, see [Add security group rules](#).

Add Security Group Rule ? Add security group rules

NIC: Internal Network

Rule Direction: Ingress

Action: Allow

Protocol Type: Customized TCP

Port Range: 3399/3399

Priority: 1

Authorization Type: IPv4 CIDR Block

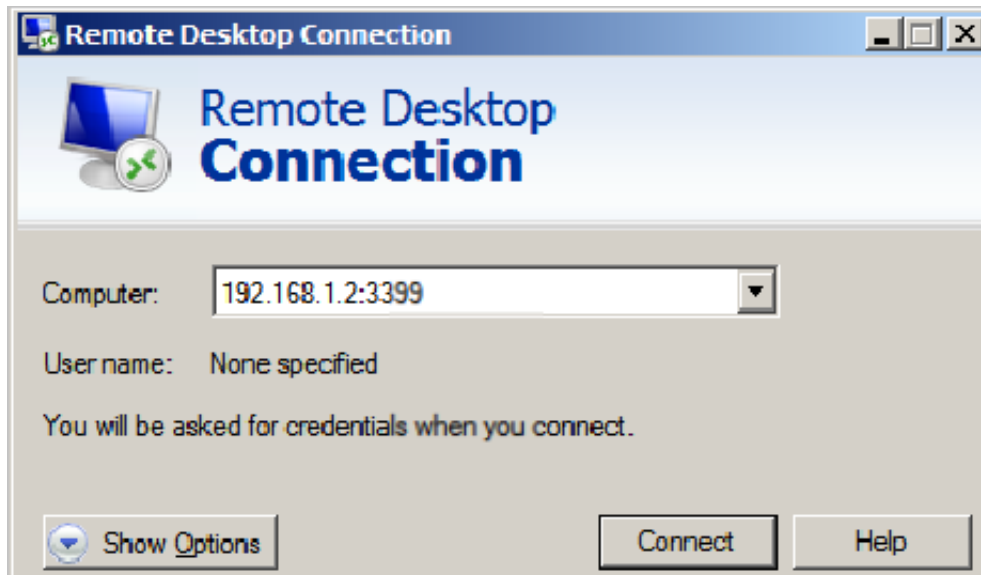
* Authorization Objects: Example: 10.x.y.z/32. You can specify up to 10 authorization objects separated with commas (,)

Description: It can be 2 to 256 characters in length and cannot start with http:// or https://.

OK

Cancel

10. Connect to the instance by accessing the IP address ending with the new port number. For example, 192.168.1.2:3399 in this example.



Note:

Only the default port 3389 can be used for access by Mac remote desktop users.

Modify the default remote port of a Linux instance

This section describes how to modify the remote port of a Linux instance running CentOS 6.8.



Note:

Do not modify the 22 port directly, first add the new default remote port. Set two ports first and delete one after the test succeeds. It ensures that you can use port 22 to debug any problems if you cannot connect the instance through the new port.

1. [Connect to the Linux instance.](#)
2. Run `vim /etc/ssh/sshd_config`.
3. Press the "I" key on the keyboard to enter the Edit mode. Add new remote service port (for example, Port 1022). Enter *Port 1022* under *Port 22*.
4. Press "ESC" and enter `: wq` to exit the editing.

- Restart the instance by executing the following command. You can then log on to the Linux instance through 22 port and 1022 port.

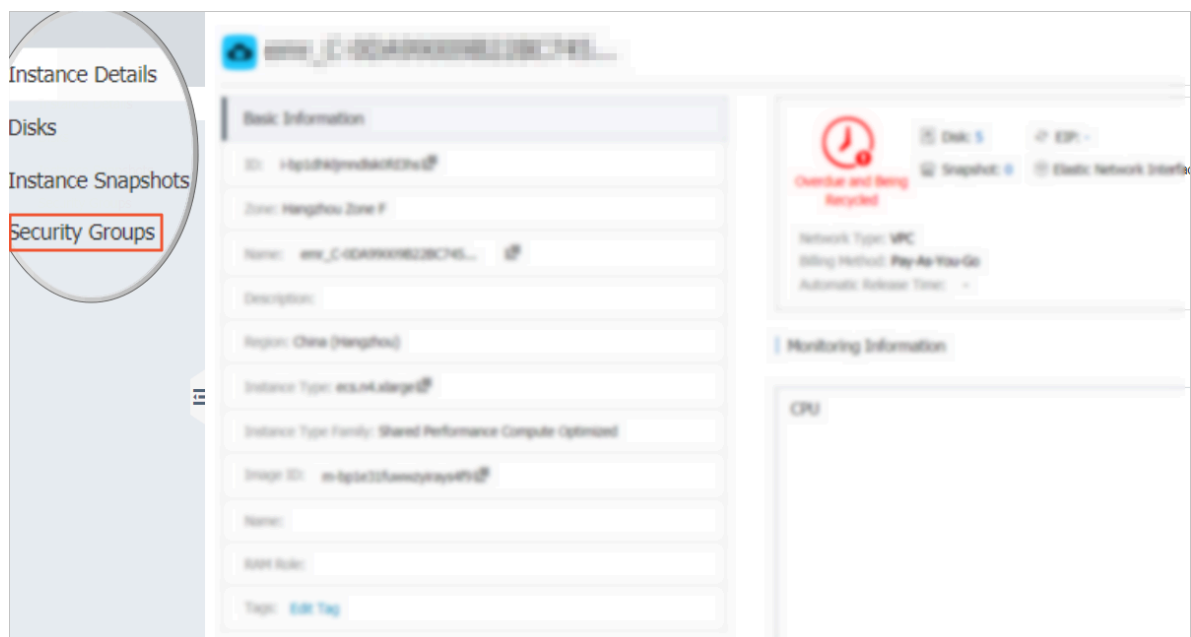
```
/ etc / init . d / ssh restart
```

- (Optional) Configure the firewall. When you use Linux versions earlier than CentOS 7 and has enabled firewall iptables, note that iptables do not intercept access by default. If you configured iptables rules, run `iptables -A INPUT -p tcp --dport 1022 -j ACCEPT` to configure the firewall. Then perform `service iptables restart` to restart the firewall.

**Note:**

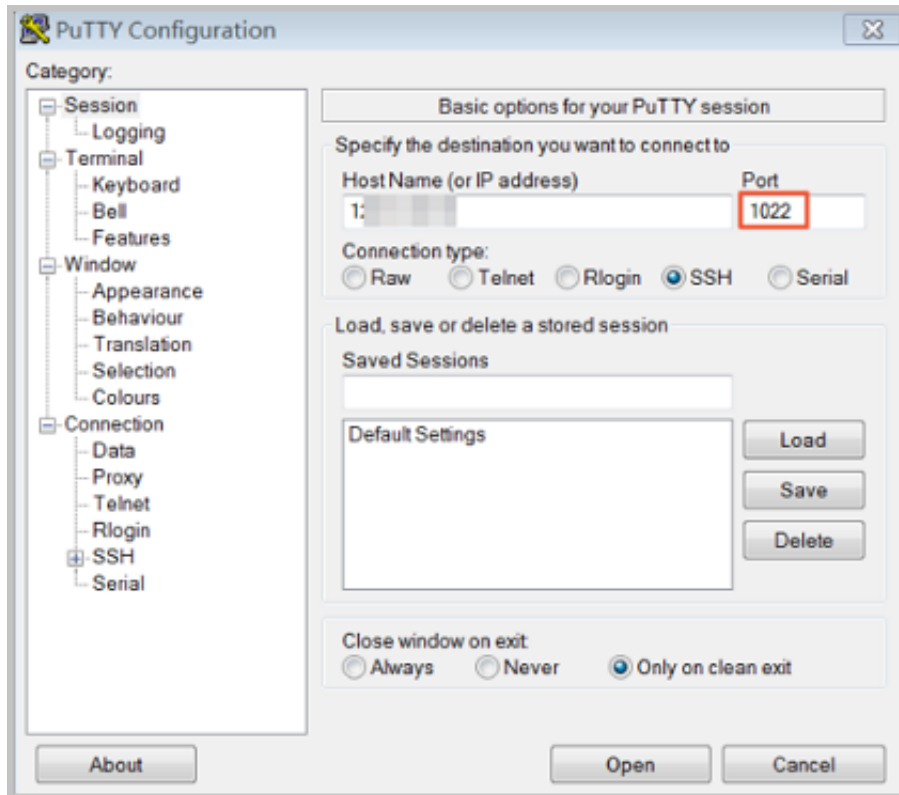
Firewalld is installed by default on CentOS 7 and later versions. If you have enabled `firewalld.service`, open TCP port 1022 by running the command `firewall-cmd --add-port=1022/tcp --permanent`. If success is returned, TCP port 1022 is opened.

- Log on to the [ECS console](#), find the instance, and then select Manage.
- Enter the Instance Details page. Click Security Groups.



- On the Security Groups page, click Add Rules.
- On the Security Group Rules page, click Add Security Group Rule. Add a new security group rule to allow access to the new remote port. For more information about adding security group rules, see [Add security group rules](#).

11. Use the SSH tool to connect to the new port to test if the default remote port is modified successfully. Enter the new port number in Port when logging on to the instance, which is 1022 in this example.



12. Once you successfully connect the instance through port 1022, run `vim /etc/ssh/sshd_config` again to remove port 22.
13. Run `/etc/init.d/sshd restart` to restart the instance and the default remote port is successfully modified. Connect to the instance by accessing the IP address ending with the new port number.

1.7 Use logs in Windows instances

Logs are records of hardware and software in the system, and system error information. They can also be used to monitor system events. When a server intrusion or system (application) error occurs, administrators can quickly locate the problems by using logs and solve the problems quickly, which improves work efficiency and server security substantially. Windows logs can be mainly divided into four categories: system logs, application logs, security logs, and applications and services logs. In this example, we use Windows Server 2008 R2 to introduce the use and analysis of the four categories of logs.

Open the Event Viewer

Follow these steps to open Event Viewer: Open the Run window, type `eventvwr` , and then click OK to open the Event Viewer.

Then, you can view the following four categories of logs in Event Viewer.



Note:

You can find the solutions to any error event ID that you can find in these logs in Microsoft knowledge base.

- System Logs

System logs include events recorded by Windows system components. For example , system logs record failures that occur when loading drivers or other system components during startup.

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

- Application logs

Application logs include events recorded by applications or programs. For example , a database application can record file errors in application logs.

The types of events recorded are determined by developers.

- Security logs

Security logs include events such as valid and invalid logon attempts, and resource usage related events such as creation, opening, or deletion of files or other objects.

Administrators can specify the types of events recorded in security logs. For example, if logon has been set to be audited, logon attempts are recorded in security logs.

- Application and service logs

Application and service logs are a new type of event logs. These logs store events from a single application or component, rather than events that may affect the global system.

Modify log path and back up logs

Logs are stored on the system disk by default. The maximum log size is 20 MB by default, and the earliest events are overwritten when 20 MB is exceeded. You can modify the maximum log size according to your needs.

Follow these steps to modify the log path and back up logs:

1. In the left-side navigation pane of Event Viewer, click Windows Logs.
2. Right click a log name, such as Application and click Properties.
3. In the Log Properties dialog box, you can modify the following settings:
 - Log path
 - Maximum log size
 - Operations executed when maximum event log size is reached

1.8 Overview and best practices of Windows Firewall with Advanced Security

This topic describes Windows Firewall with Advanced Security (WFAS), its application scenarios, and common operations.

Overview

As an important part of the hierarchical security model, WFAS was launched after Windows NT6.0 by Microsoft. WFAS blocks unauthorized traffic that flows in or out of local computers by providing bi-directional filtering based on the current connection status. WFAS also uses Network Location Awareness (NLA) to apply the corresponding firewall profile to the computer based on its current connection status. The security rules of Windows Firewall and Internet Protocol Security (IPsec) are configured in the Microsoft Management Console (MMC) snap-in, and WFAS is also an important part of the network isolation policy.

Application Scenario

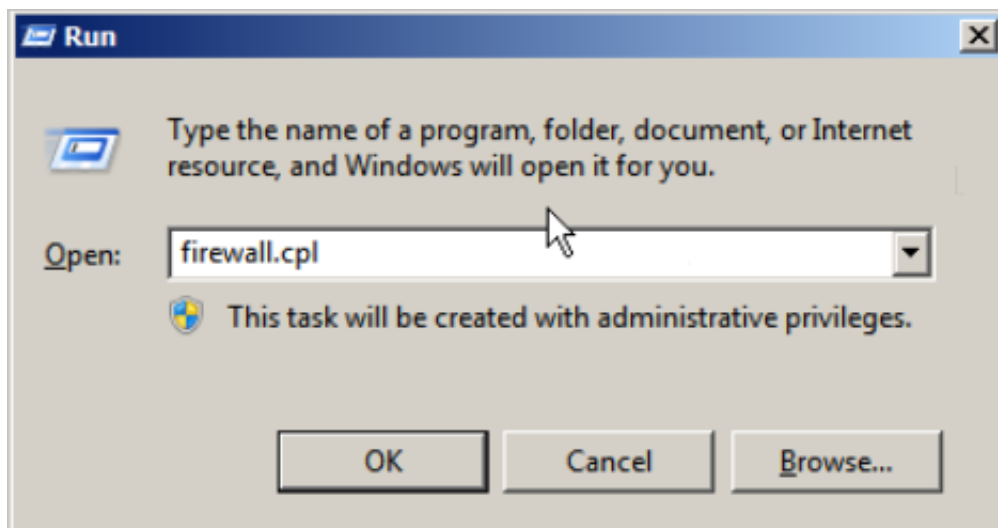
More and more O&M personnel are reporting that servers are attacked and passwords are cracked, which in most cases, are due to the “backdoor” left open to “intruders”. Intruders scan open ports on your computers and penetrate them through vulnerable ports, for example, the remote port 3389 in Windows and the remote port 22 in Linux. Now that we know where the problem is, we can take the effective countermeasure. Specifically, we can close these “backdoors” by modifying the

default remote ports and restricting remote access. So how do we restrict remote access? Now let's demonstrate how to restrict the remote desktop connection by taking an ECS instance (Windows Server 2008 R2) for example.

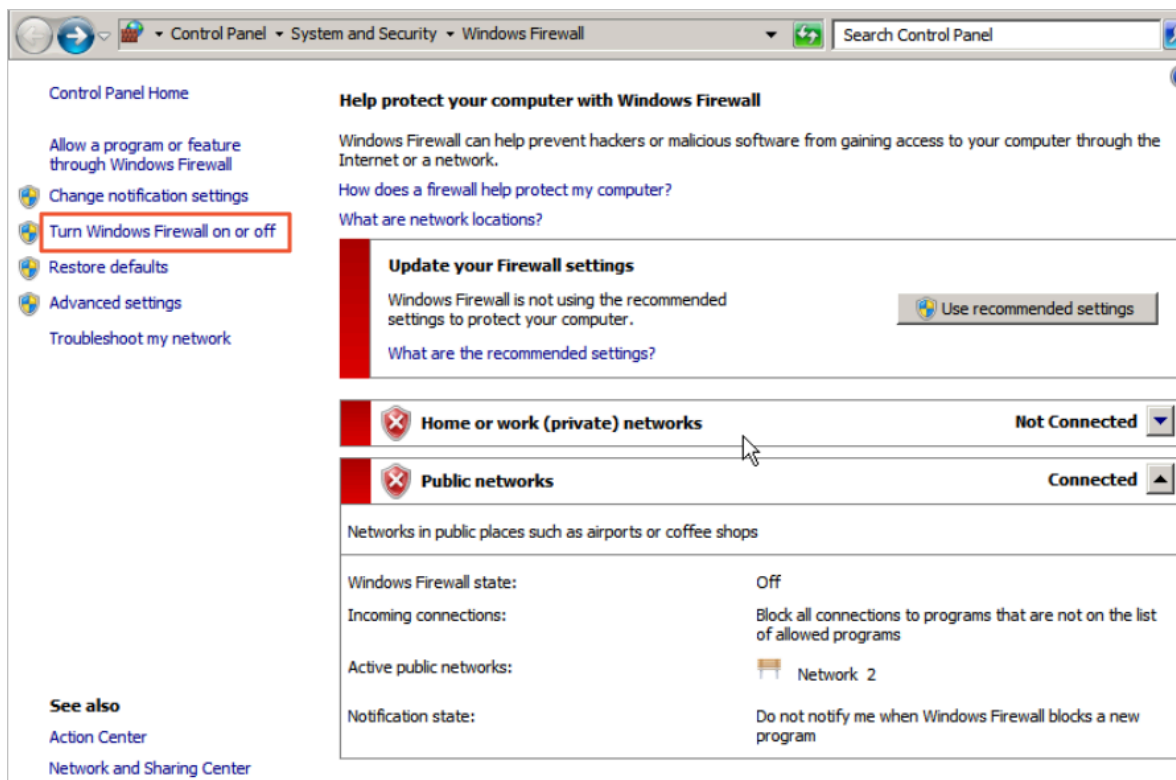
Procedure

1. View the Windows Firewall status

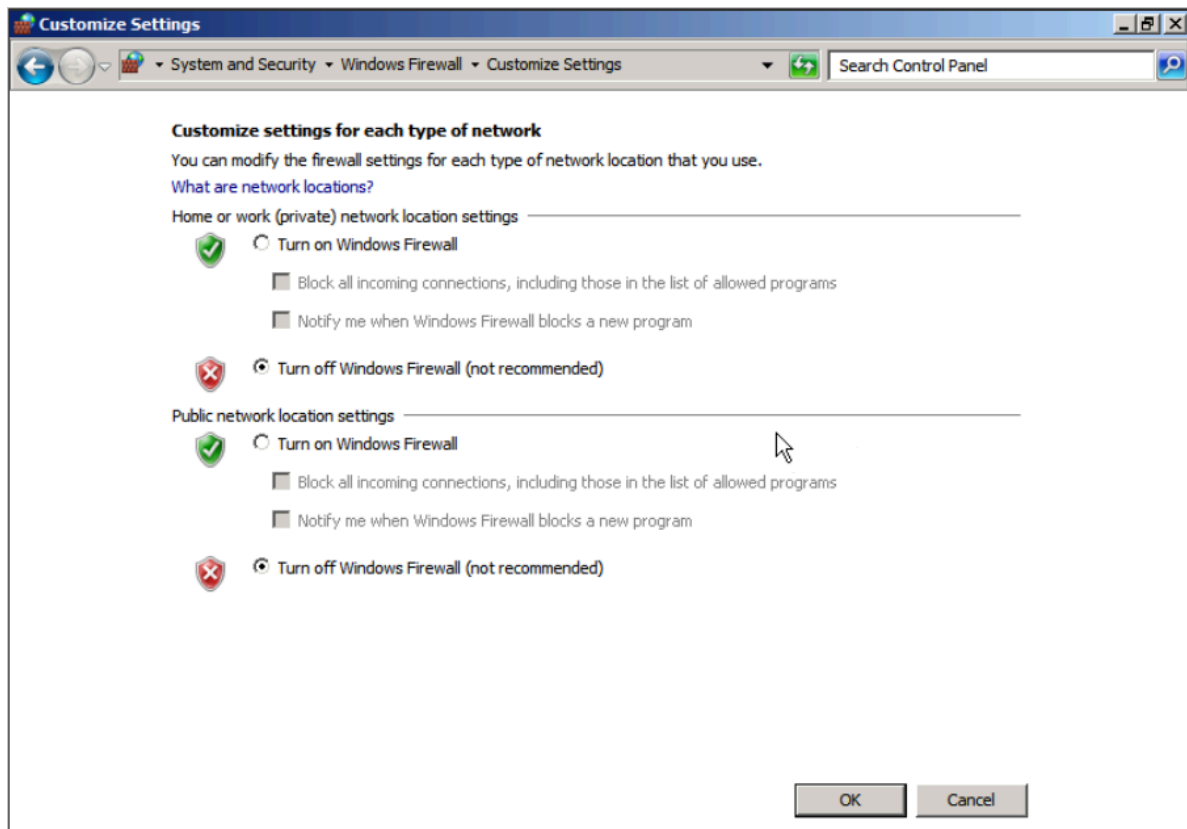
Windows Firewall of the ECS instance is disabled by default. You can press Win+R to open the Run window, enter *firewall.cpl*, and then press Enter to open the Windows Firewall console, as shown below.



Enable or disable Windows Firewall.

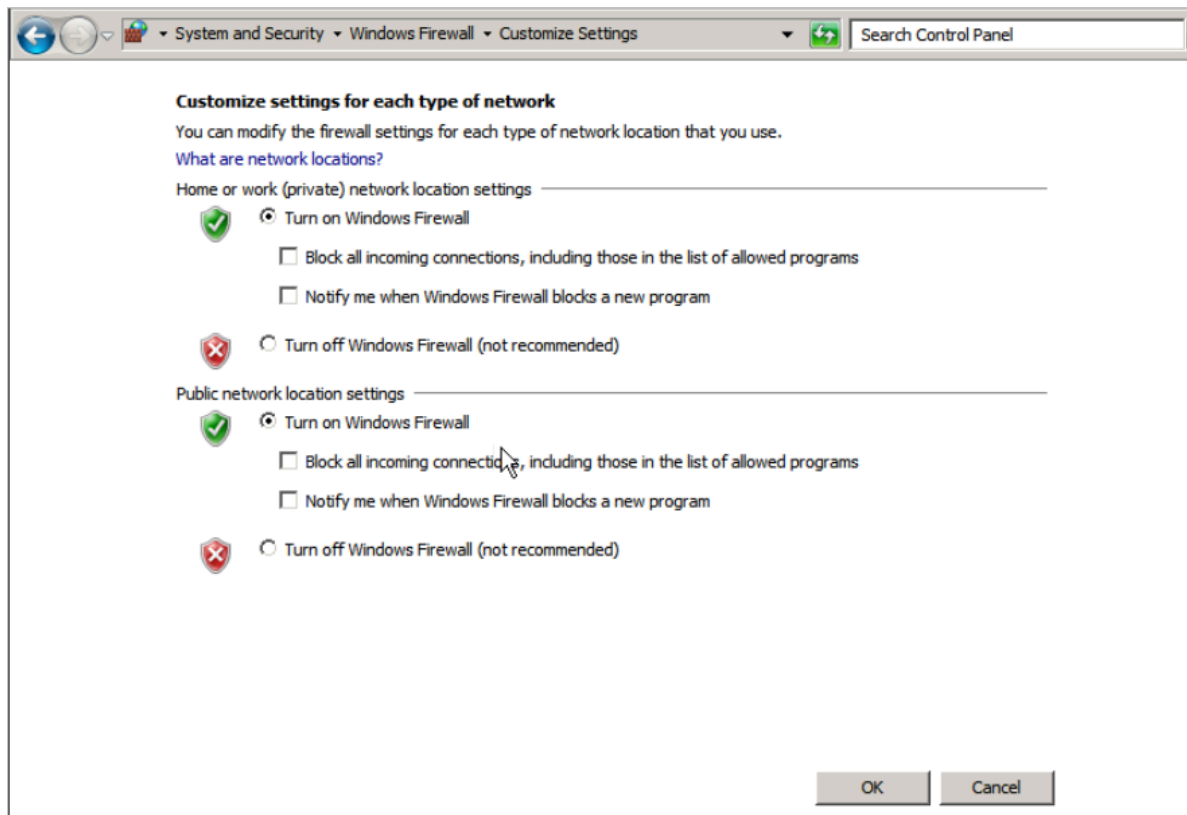


As shown below, Windows Firewall is disabled by default.



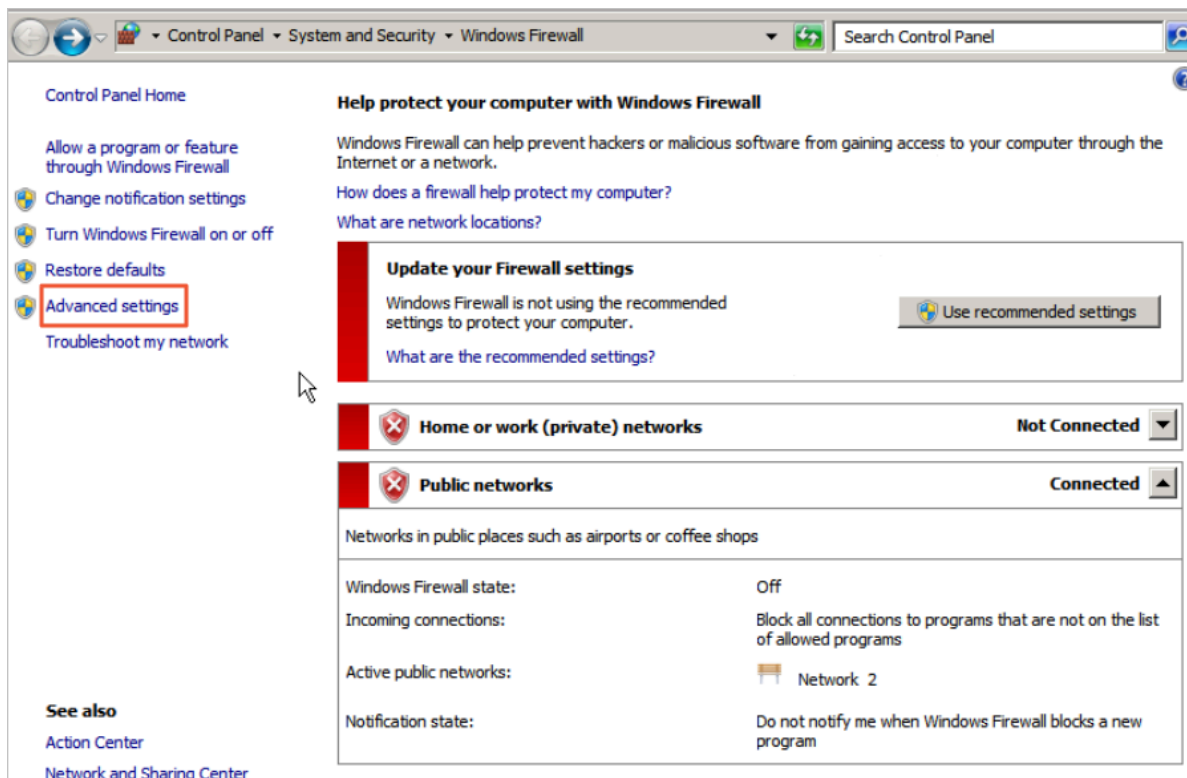
2. Enable Windows Firewall

Enable Windows Firewall through the previous steps, as shown below.

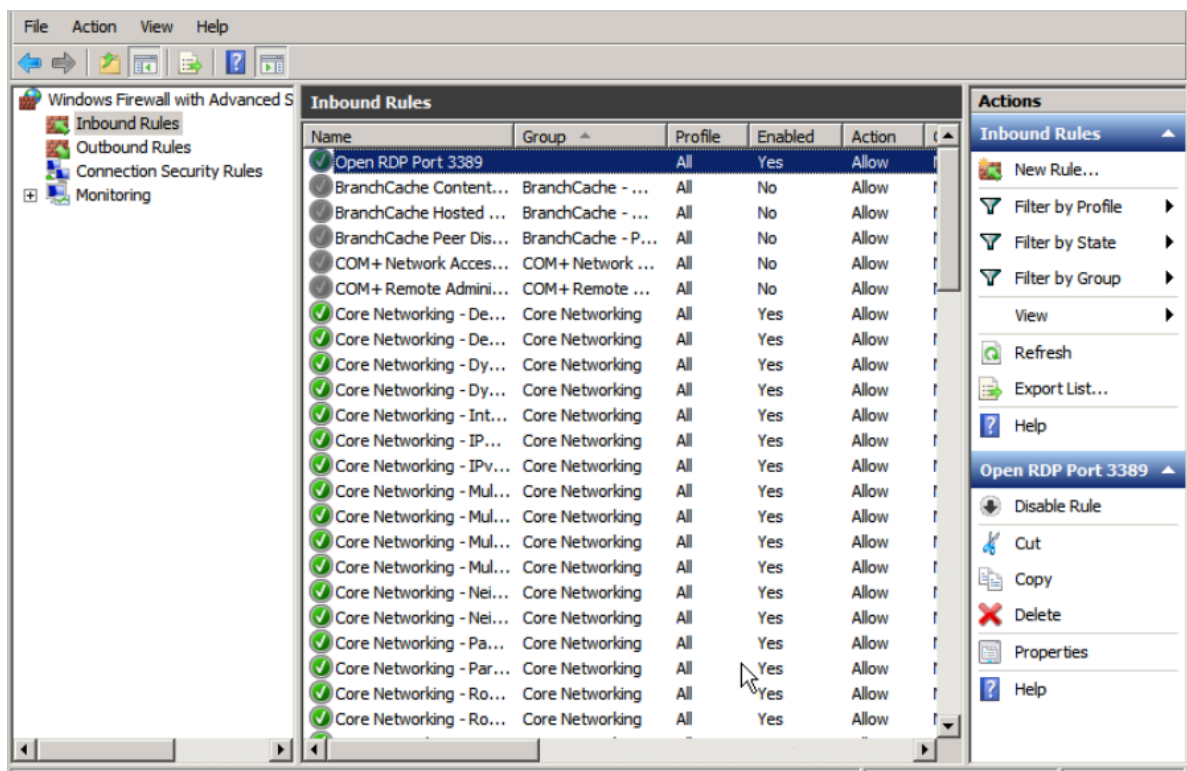


Before enabling Windows Firewall, make sure the remote port is open in the inbound rules, or you cannot establish the remote connection even yourself. WFAS

, however, opens RDP port 3389 in its inbound rules by default. Select Advanced settings.

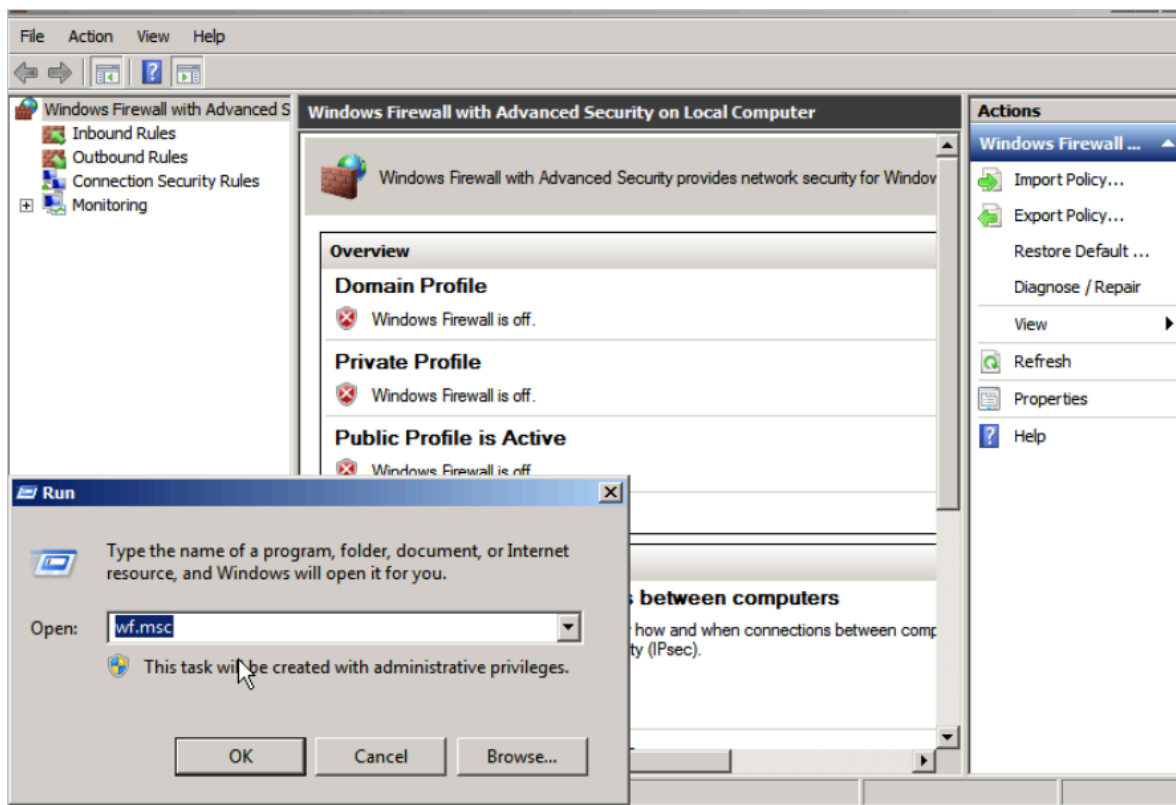


Select Inbound Rules. We can see that the Open RDP Port 3389 rule is enabled by default.

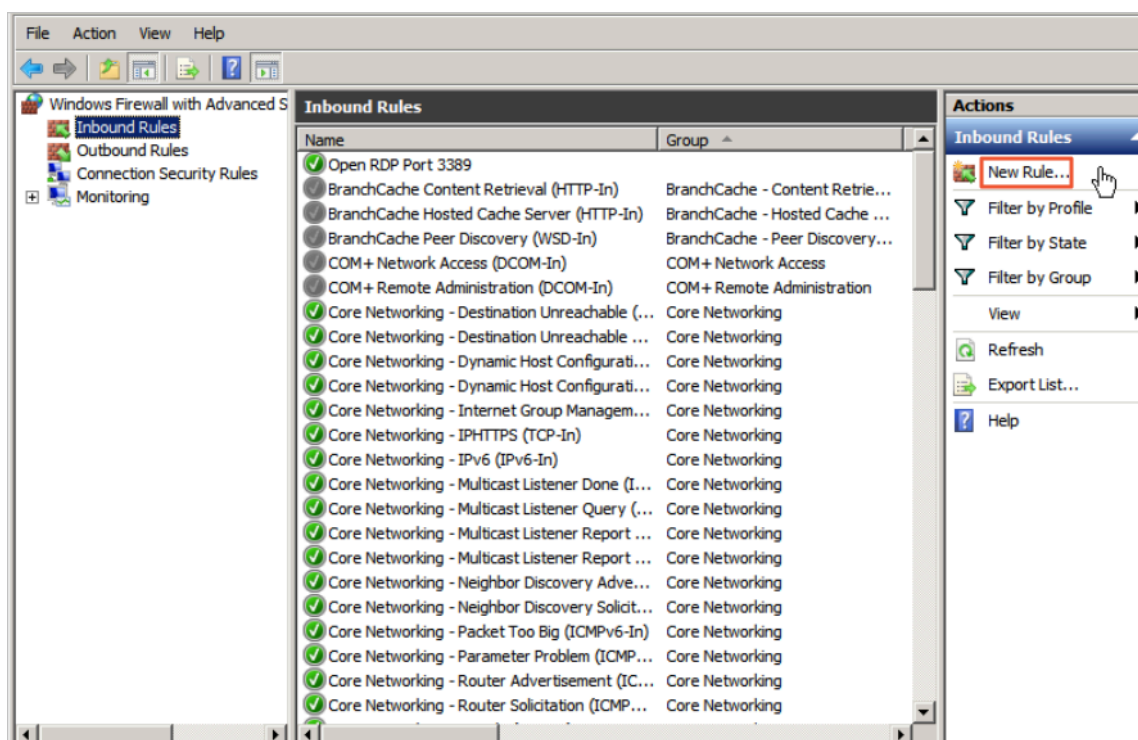


3. Configure WFAS

Press Win+R to open the Run window, enter `wf.msc`, and then press Enter to open the WFAS window, as shown below.



a. Create an inbound rule manually



In the New Inbound Rule Wizard window, select Port and click Next.

Rule Type
Select the type of firewall rule to create.

Steps:

- Rule Type
- Protocol and Ports
- Action
- Profile
- Name

What type of rule would you like to create?

☐ **Program**
Rule that controls connections for a program.

☒ **Port**
Rule that controls connections for a TCP or UDP port.

☐ **Predefined:**
BranchCache - Content Retrieval (Uses HTTP)
Rule that controls connections for a Windows experience.

☐ **Custom**
Custom rule.

[Learn more about rule types](#)

< Back Next > Cancel

Select TCP and set Specific Local Ports to 3389.

Protocol and Ports

Specify the protocols and ports to which this rule applies.

Steps:

- Rule Type
- Protocol and Ports
- Action
- Profile
- Name

Does this rule apply to TCP or UDP?

☒ **TCP**

☐ **UDP**

Does this rule apply to all local ports or specific local ports?

☐ **All local ports**

☒ **Specific local ports:**

Example: 80, 443, 5000-5010

[Learn more about protocol and ports](#)

< Back Next > Cancel

Click Next and select Allow the connection.

Action

Specify the action to be taken when a connection matches the conditions specified in the rule.

Steps:

- Rule Type
- Protocol and Ports
- Action
- Profile
- Name

What action should be taken when a connection matches the specified conditions?

☒ **Allow the connection**

This includes connections that are protected with IPsec as well as those are not.

☐ **Allow the connection if it is secure**

This includes only connections that have been authenticated by using IPsec. Connections will be secured using the settings in IPsec properties and rules in the Connection Security Rule node.

☐ **Block the connection**

[Learn more about actions](#)

< Back Next > Cancel

Click Next and keep the default configurations.

Profile
Specify the profiles for which this rule applies.

Steps:

- Rule Type
- Protocol and Ports
- Action
- Profile
- Name

When does this rule apply?

- ☒ **Domain**
Applies when a computer is connected to its corporate domain.
- ☒ **Private**
Applies when a computer is connected to a private network location.
- ☒ **Public**
Applies when a computer is connected to a public network location.

[Learn more about profiles](#)

< Back Next > Cancel

Click Next and enter the rule name (for example, "RemoteDesktop"), and click Finish.

Name

Specify the name and description of this rule.

Steps:

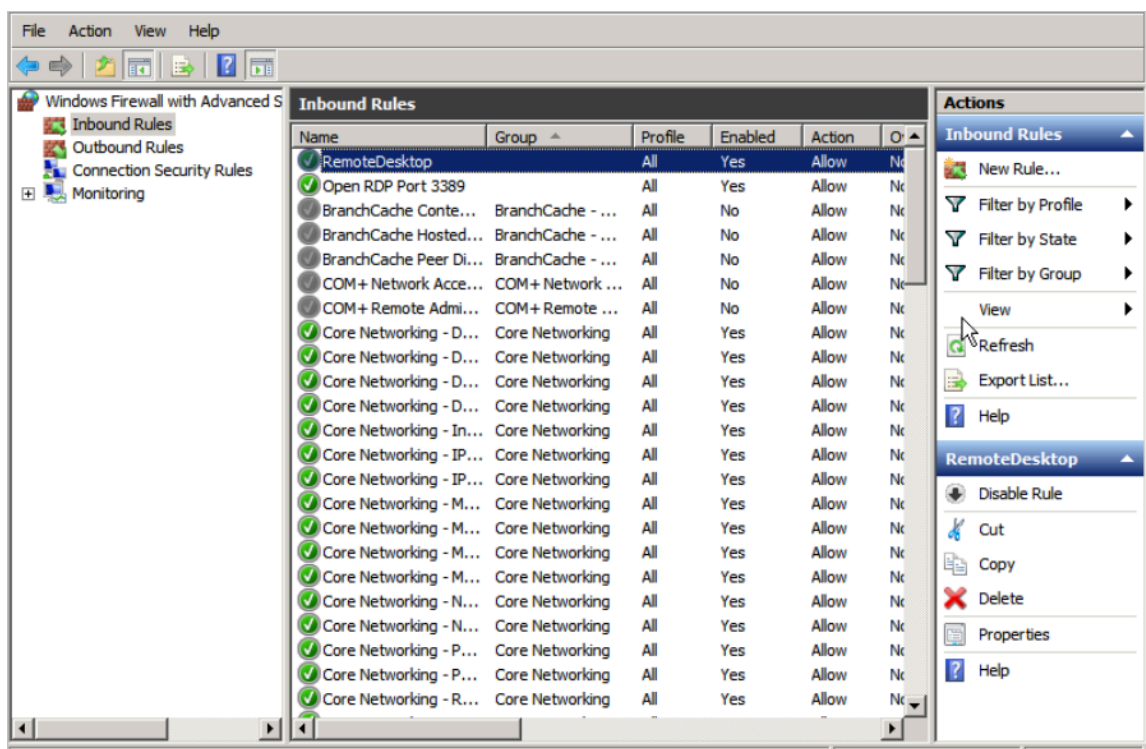
- Rule Type
- Protocol and Ports
- Action
- Profile
- Name**

Name: RemoteDesktop

Description (optional):

< Back Finish Cancel

The new rule is shown in the Inbound Rule list.

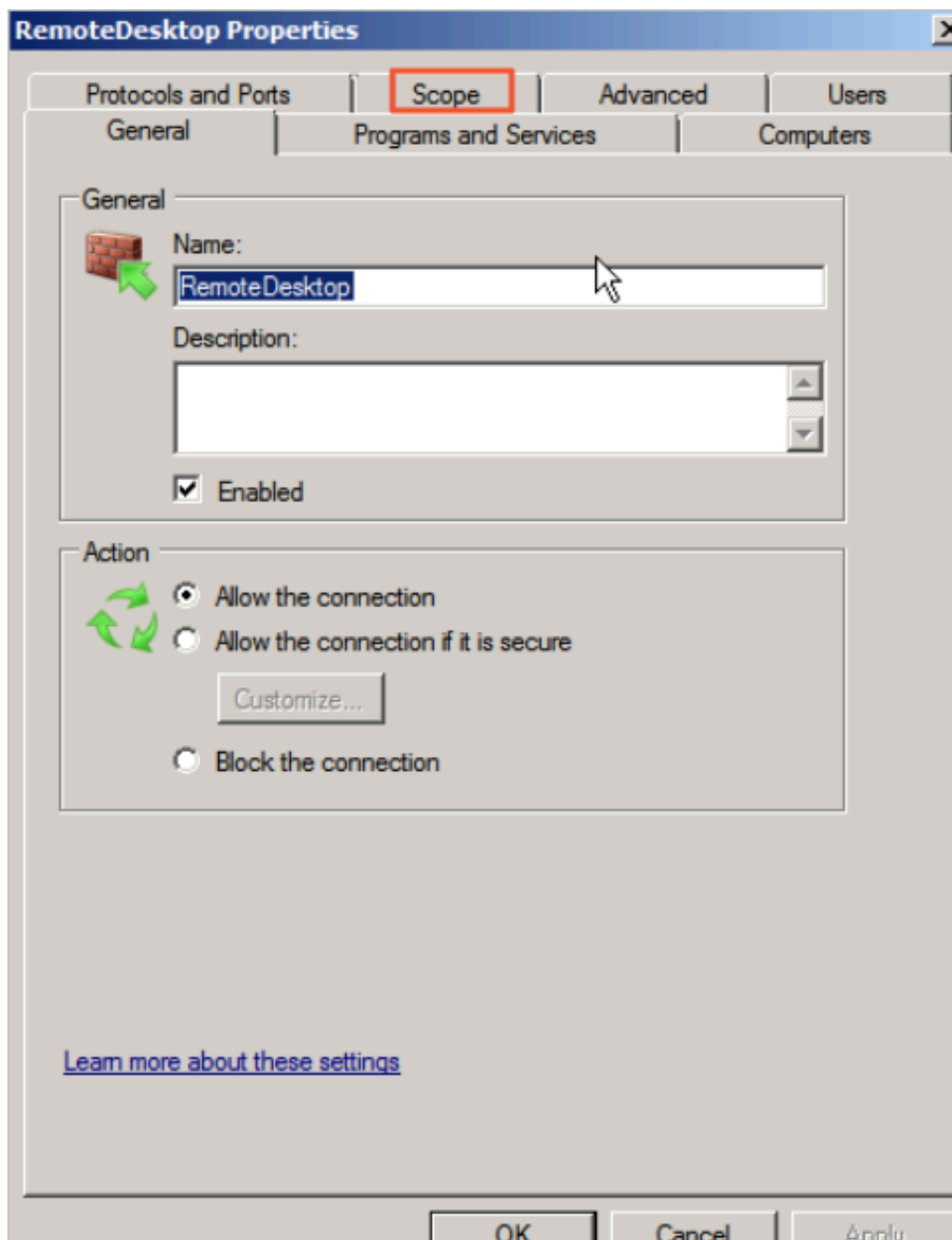


With the above steps, the remote port is added to WFAS, but access restriction is still not implemented. Let's implement it now.

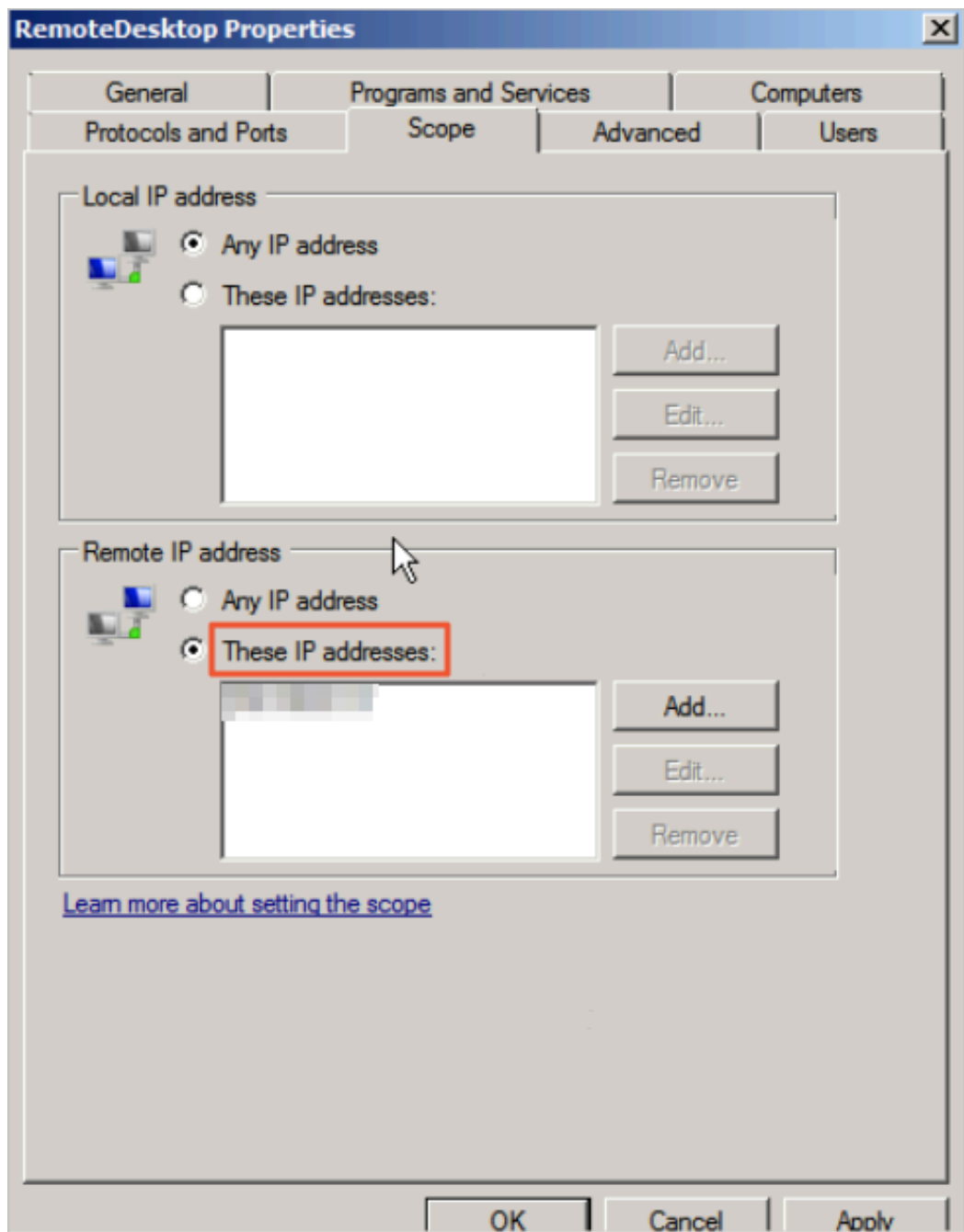
b. Configure the IP address scope

Right-click the just created inbound rule, and select Properties in the context menu. In the displayed dialog box, click the Scope tab. Then add the remote IP addresses that can access this ECS instance. Note that once the IP address

settings here are enabled, other IP addresses will be unable to access this ECS instance.

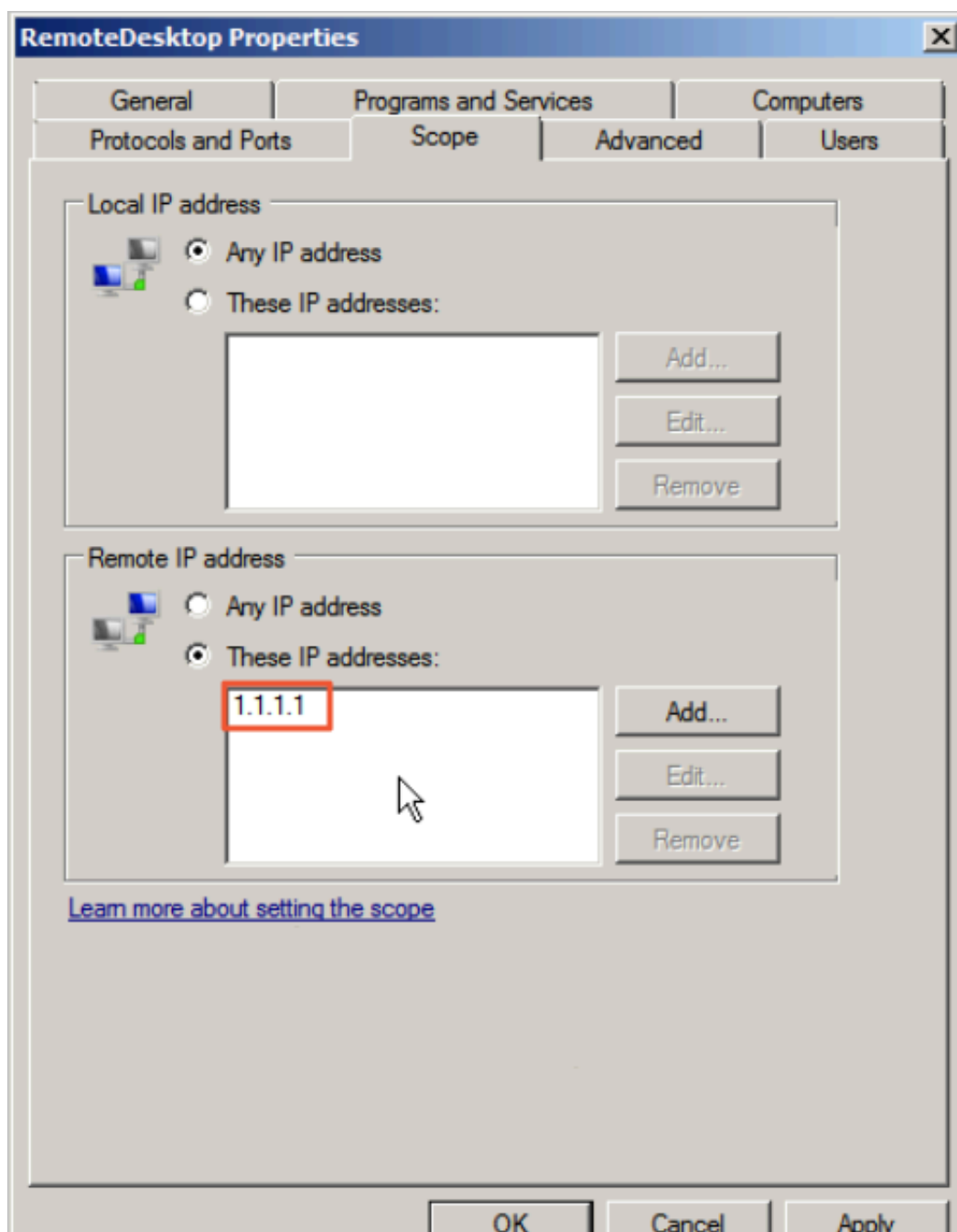


Add remote IP addresses.

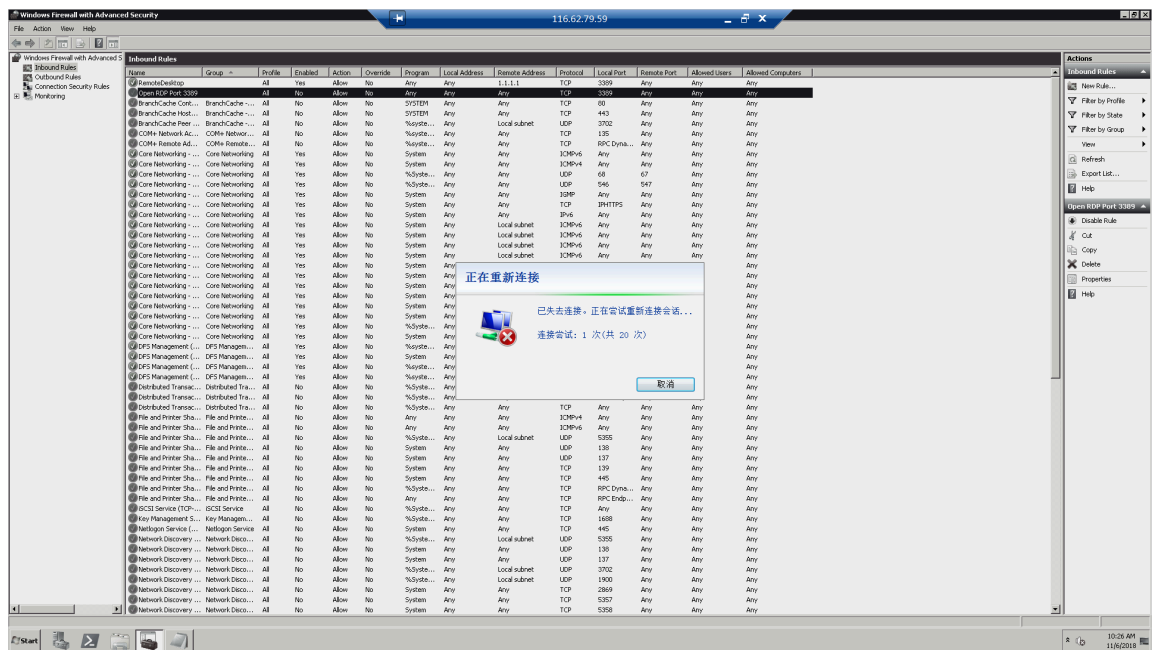


c. Validate the IP address scope

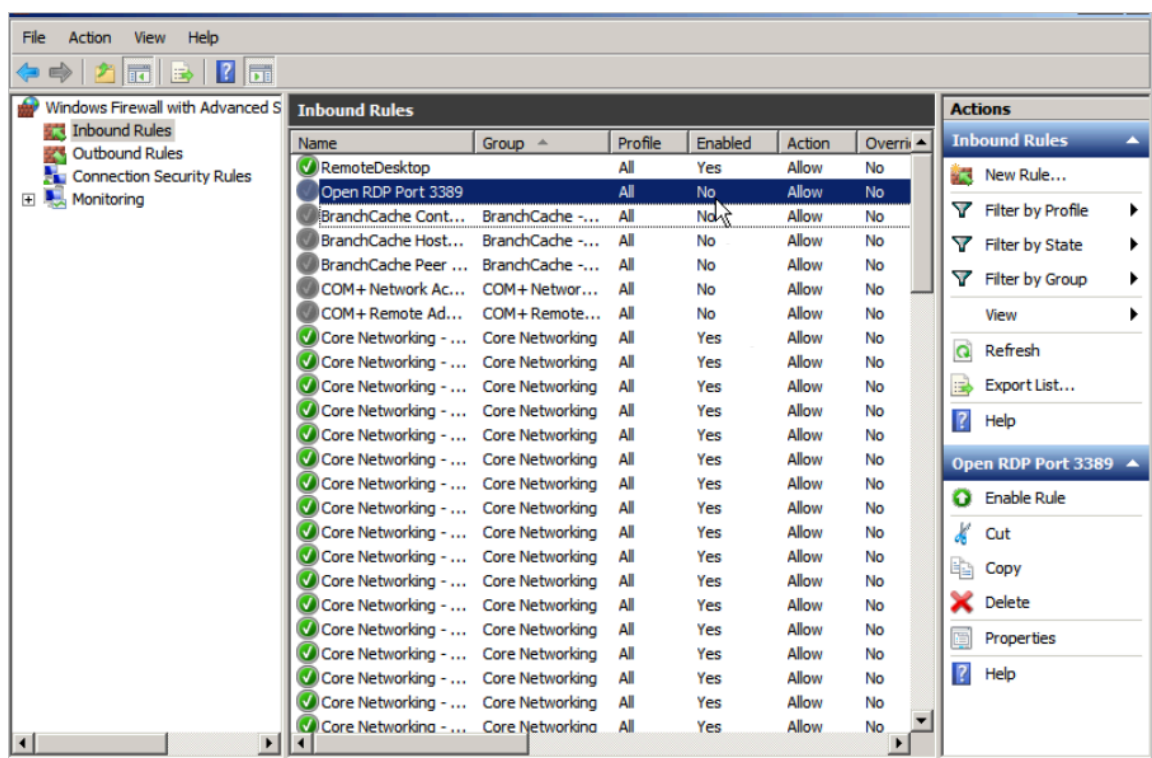
Let's add an IP address arbitrarily in the Remote IP address box and see what happens to the remote connection.



The remote connection is down.



If the remote connection is still up, we can just disable the Open RDP Port 3389 rule.

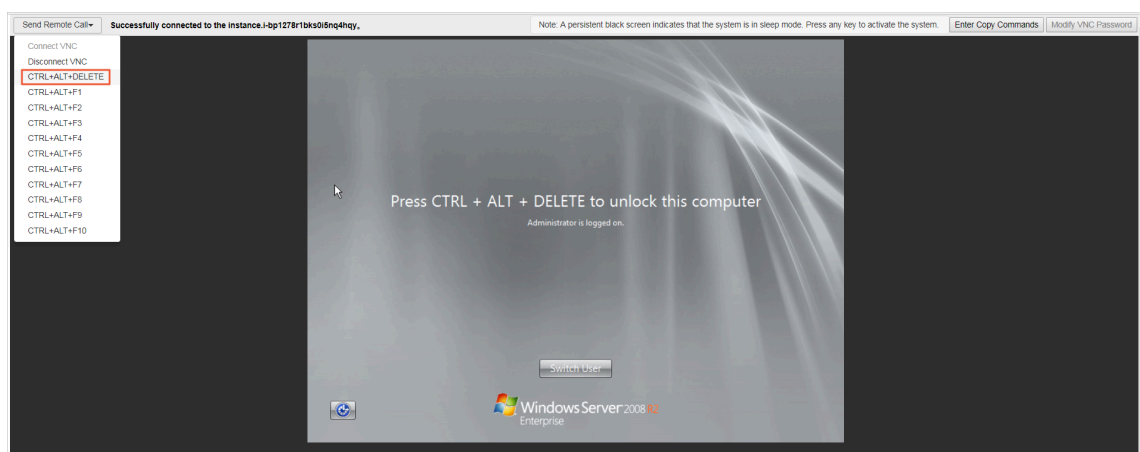


If the remote connection is down, it means that the IP address scope has taken effect. However, we cannot connect to the ECS instance ourselves now. What should we do? We now can turn to the ECS console. Log on to the ECS console, and replace the remote IP address previously configured in the Scope tab with

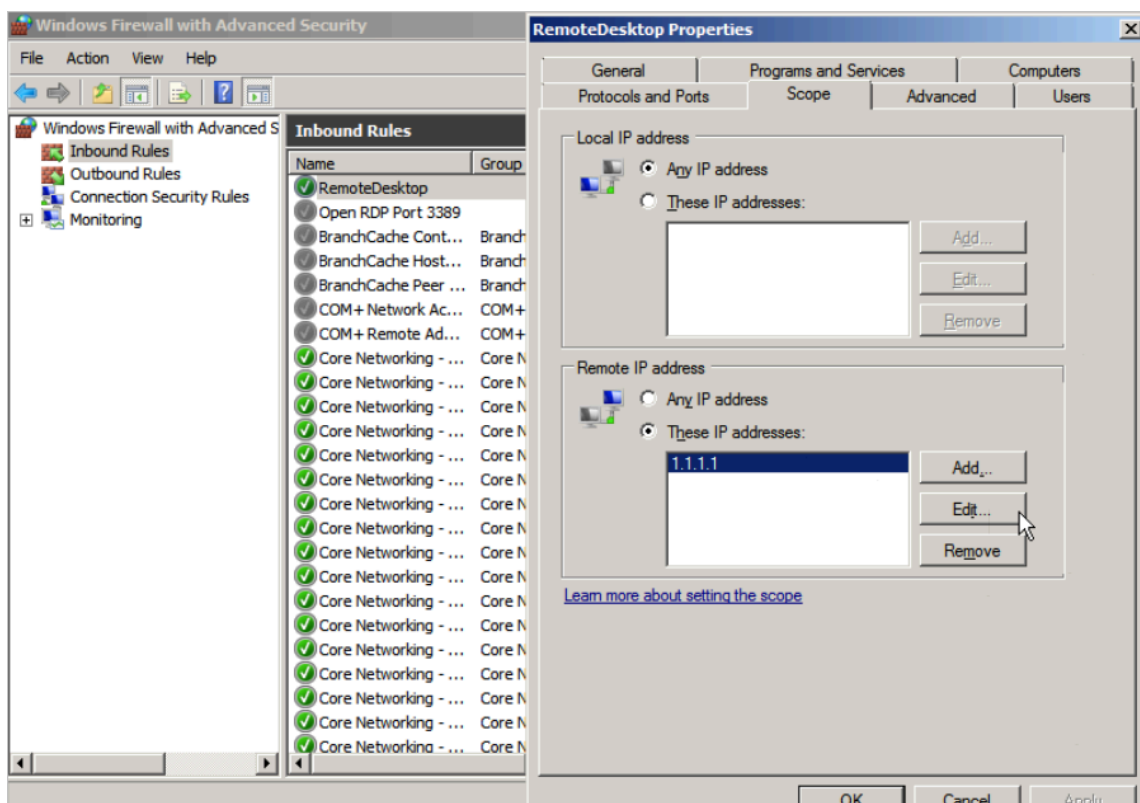
our own address (enter the Internet address unless your work environment is connected to Alibaba Cloud). You can connect to the ECS instance again now. Enter the ECS console, find the corresponding instance, and then connect to it.



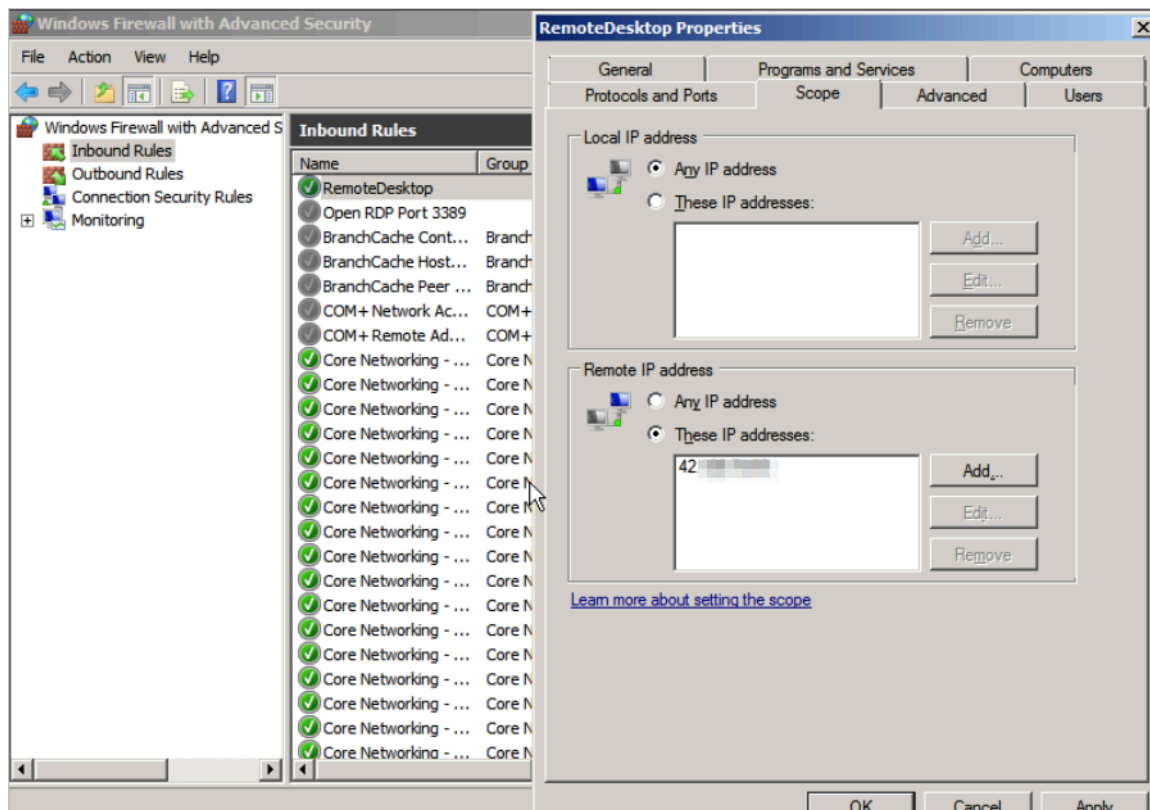
Log on to the ECS instance.



Modify the remote IP address in the Scope tab of the RemoteDesktop rule in the same way. Specifically, replace 1.1.1.1 with our own IP address.



Now we can connect to the ECS instance normally after adding our IP address. If you do not know your Internet address, you can [click here](#) to view it.



The above steps implement remote access restriction on an ECS instance through WFAS. For other services and ports, restrictions can be implemented in the same way, for example, disabling ports 135, 137, 138, and 445 that are not used frequently, limiting access to FTP and related services, and more, thus maximizing the protection of ECS instances.

Command line operations

1. Export the firewall configurations to a file.

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

2. Import the firewall configuration file to the system.

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

3. Restore the default firewall settings.

```
Netsh advfirewall l reset
```

4. Disable the firewall.

```
netsh advfirewall l set allprofile s state off
```

5. Enable the firewall.

```
netsh advfirewall l set allprofile s state on
```

6. Configure to block inbound traffic and allow outbound traffic by default in all configuration files.

```
netsh advfirewall l set allprofile s firewallpo licy  
blockinbou nd , allowoutbo und
```

7. Delete the rule named “ftp” .

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

8. Delete all inbound rules for local port 80.

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

9. Add the RemoteDesktop rule to allow port 3389.

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

References

[How to restrict the access of ports/IP addresses/applications using Windows 2008/2012 Firewall?](#)

More open source software are available at [Alibaba Cloud Marketplace](#).

1.9 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 to 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:

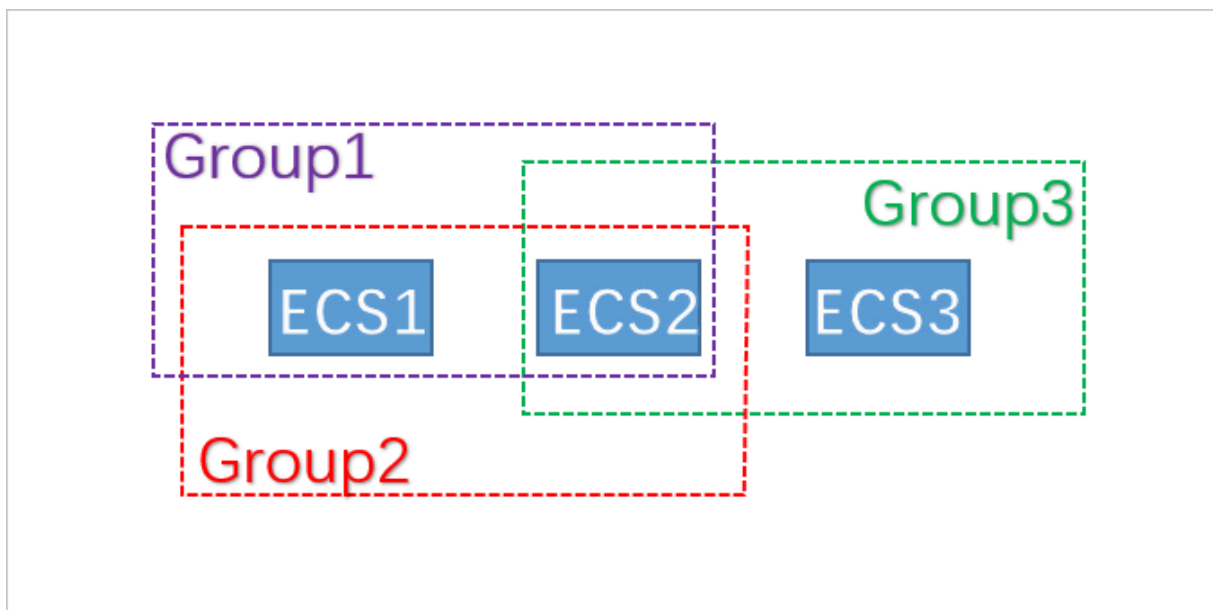
- Intra-group isolation rules are set in a security group, while an Access Control List (ACL) that permits intra-group communication between instances is set at the same time.
- Intra-group isolation rules are set in a security group, while intra-group intercommunication is configured at the same time.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

Instance	Interconnected or isolated?	Reason
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.

1.10 Security group quintuple rules

Security groups are used to set network access control for one or more ECS instances. As an important means of security isolation, security groups allow you to divide security domains on the cloud. Security group quintuple rules let you precisely control the following five parameters: the 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 ingress rules only support the settings of the source IP address, the destination port, and the transport layer protocol.
- The egress rules only support the settings of the destination IP address, the destination port, and the transport layer protocol.

In most scenarios, these types of security group rules simplify the setup process, but possess the following disadvantages:

- The source port range of an ingress rule is not restricted. That is, all source ports are permitted by default.
- The destination IP address of an ingress rule is not restricted. That is, all IP addresses in a security group are permitted by default.
- The source port range of an egress rule is not restricted. That is, all source ports are permitted by default.
- The source IP address of an egress rule is not restricted. That is, all IP addresses in a security group are permitted by default.

Definition of a quintuple rule

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

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

The following shows an example quintuple 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 : Drop
```

The example egress rule indicates that 172.16.1.0/32 is prohibited from accessing 10.0.0.1/32 from port 22 through TCP.

Scenarios

- Some platform products are connected to the solutions of third-party vendors to provide them with network services. To prevent these products from illegally accessing users' ECS instances, it is needed to set quintuple rules in the security group to control the inbound and outbound traffic more precisely.
- If your instances are isolated within a security group due to settings, and you want to precisely control the access between several ECS instances in the group, you can set security group quintuple rules to meet your needs.

How to configure quintuple rules

You can use OpenAPI to set quintuple rules.

- To add a security group ingress rule, see [AuthorizeSecurityGroup](#).
- To add a security group egress rule, see [AuthorizeSecurityGroupEgress](#).
- To delete a security group ingress rule, see [RevokeSecurityGroup](#).
- To delete a security group egress rule, see [RevokeSecurityGroupEgress](#).

Parameters

The following table describes the parameters.

Parameter	Meaning in ingress rules	Meaning in egress rules
SecurityGroupId	The ID of the security group to which the current ingress rule belongs (that is, the ID of the destination security group).	The ID of the security group to which the current egress rule belongs (that is, the ID of the source security group).
DestCidrIp	Destination IP address range; optional. <ul style="list-style-type: none"> If DestCidrIp is specified, you can control the destination IP address range in an ingress rule more precisely. If DestCidrIp is not specified, the IP address range in an ingress rule includes all the IP addresses in the security group indicated by the SecurityGroupId. 	Destination IP addresses. Either DestGroupId or DestCidrIp must be specified. If both are specified, DestCidrIp takes priority.
PortRange	Destination port range; required.	Destination port range; required.
DestGroupId	Input not allowed. The destination security group ID must be a SecurityGroupId.	The destination security group ID. Either DestGroupId or DestCidrIp must be specified. If both are specified, DestCidrIp takes priority.
SourceGroupId	The source security group ID. Either SourceGroupId or SourceCidrIp must be specified. If both are specified, SourceCidrIp takes priority.	Input not allowed. The source security group ID in an egress rule must be a SecurityGroupId.

Parameter	Meaning in ingress rules	Meaning in egress rules
SourceCidrIp	Source IP address range. Either SourceGroupId or SourceCidrIp must be specified. If both are specified, SourceCidrIp takes a higher priority.	Source IP address range; optional. <ul style="list-style-type: none"> · If SourceCidrIp is specified, you can control the source IP address range in an egress rule more precisely. · If SourceCidrIp is not specified, the source IP addresses in an egress rule include all the IP addresses in the security group indicated by the SecurityGroupId.
SourcePort Range	Source port range; optional. If it is not specified, source ports are not restricted.	Source port range; optional. If it is not specified, source ports are not restricted.

1.11 Revoke the authorization for internal network communication between ECS instances for different accounts through the API

If you have authorized internal network communication between ECS instances for different accounts in the same region, you can revoke security group authorization as specified in this topic.

Prerequisites

Alibaba Cloud CLI is used to call ECS APIs. Make sure that you have installed Alibaba Cloud CLI. For more information, see [Alibaba Cloud CLI Installation Guide](#).

Context

As specified in this topic, you can call [#unique_39](#) to revoke an authorized security group rule. You must prepare the following items:

- Account name: the name of the account that you use to log on to the ECS console.
- Security group ID for the ECS instance: the ID of the security group to which the instance involved belongs. You can view this item in the ECS console or by calling [#unique_40](#).
- Region ID for the ECS instance: See [#unique_41](#). `cn - beijing` is used in this example.

The following table lists the information of two accounts.

Account	Account name	Security group	Security group ID
Account A	a@aliyun.com	sg1	sg-bp1azkttqp ldxgtedXXX
Account B	b@aliyun.com	sg2	sg-bp15ed6xe1 yxeycg7XXX

After revoking the authorization for internal network communication between ECS instances for different accounts, you can re-authorize the communication.

Procedure

1. Run the following command for Account A:

```
aliyun ecs RevokeSecurityGroup -- SecurityGroupId sg -
bp1azkttqp ldxgtedXXX -- RegionId cn - beijing -- IpProtocol
all -- PortRange - 1 /- 1 -- SourceGroupId sg - bp15ed6xe1
yxeycg7XXX -- SourceGroupOwnerAccount b @ aliyun . com --
NicType intranet
```

2. Run the following command for Account B:

```
aliyun ecs RevokeSecurityGroup -- SecurityGroupId sg -
bp15ed6xe1 yxeycg7XXX -- RegionId cn - beijing -- IpProtocol
all -- PortRange - 1 /- 1 -- SourceGroupId sg - bp1azkttqp
ldxgtedXXX -- SourceGroupOwnerAccount a @ aliyun . com --
NicType intranet
```

1.12 Authorize internal network communication between ECS instances for different accounts through the API

This topic describes how to authorize internal network communication between ECS instances for different accounts in the same region.

Background

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

- **Communication between ECS instances:** You can authorize internal network communication between two ECS instances that belong to the same account.
- **Communication between accounts:** You can authorize internal network communication between all ECS instances that belong to two different security groups within the same region, including those purchased after authorization.



Note:

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

Limits

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 in different security groups, ensure that the instances can establish internal network communication.

- If all instances are of the classic network type, they must be in the same region.
- If all instances are of the VPC type, instances in different VPCs cannot communicate with each other. We recommend that you establish public network communication methods, or use [Express Connect](#), [VPN Gateway](#), or [Cloud Enterprise Network](#).
- If instances are of different network types, enable [ClassicLink](#) to allow communication between instances.
- If instances are located in different regions, we recommend that you establish public network communication methods, or use [Express Connect](#), [VPN Gateway](#), or [Cloud Enterprise Network](#).

Prerequisites

Alibaba Cloud CLI is used to call ECS APIs. Make sure that you have [installed](#) Alibaba Cloud CLI and [configured](#) it for use.

Authorize internal network communication between ECS instances

1. Query the internal IP addresses and security group IDs of the two ECS instances.

You can obtain the IDs of the security groups to which the ECS instances belong through the console or by calling the [#unique_47](#) operation. The following table lists the information of two ECS instances.

Instance	IP address	Security group	Security group ID
Instance A	10.0.0.1	sg1	sg-bp1azkttqp ldxgtedXXX

Instance	IP address	Security group	Security group ID
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-bp1azkttqp ldxgtdXXX -- RegionId cn - qingdao --
IpProtocol all -- PortRange -- 1 /- 1 . -- SourceCidr Ip 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 . -- SourceCidr Ip 10
. 0 . 0 . 1 -- NicType intranet
```



Note:

- In the preceding examples, region ID `cn - qingdao` is for reference only. Replace it with the actual region ID.
- In the preceding examples, the [#unique_48](#) operation is called to add security group rules. The key parameters for this operation include `SecurityGroupId` and `SourceCidr Ip`.

4. Wait for a short period of time and 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 the names and security group IDs of the two accounts.

You can obtain the IDs of the security groups to which the accounts belong by using the console or calling the [#unique_47](#) operation. The following table lists the information of two accounts.

Account	Account ID	Security group	Security group ID
Account A	a@aliyun.com	sg1	sg-bp1azkttqp ldxgtdXXX
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-bp1azkttqp ldxgtdXXX -- RegionId cn - qingdao --
```

```

IpProtocol    all    -- PortRange =- 1 /- 1 . -- SourceGroup    pId
sg - bp15ed6xe1 yxeycg7XXX -- SourceGroup    pOwnerAcco    unt    b @
aliyun . com -- NicType    intranet

```

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

```

aliyun    ecs    AuthorizeSecurityGroup    up    -- SecurityGroup    oupId
sg - bp15ed6xe1 yxeycg7XXX -- RegionId    cn - qingdao --
IpProtocol    all    -- PortRange =- 1 /- 1 . -- SourceGroup    pId
sg - bplazkttqp ldxgtedXXX -- SourceGroup    pOwnerAcco    unt    a @
aliyun . com -- NicType    intranet

```



Note:

- In the preceding examples, region ID `cn - qingdao` is for reference only. Replace it with the actual region ID.
- In the preceding examples, the [#unique_48](#) operation is called to add security group rules. The key parameters for this operation include `SecurityGroup` `oupId` , `SourceGroup` `pId` , and `SourceGroup` `pOwnerAcco` `unt` .

4. Wait for a short period of time and run the `ping` command to check whether the ECS instances in two security groups can communicate with each other over the internal network.

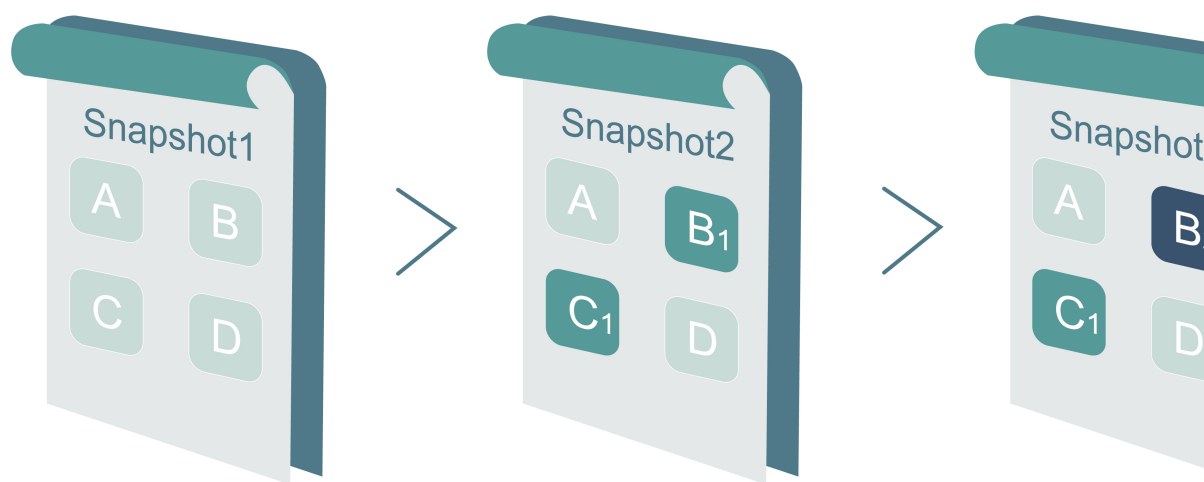
2 Disaster recovery solutions

Disaster recovery solutions help guarantee the running stability and data security of your IT system. Specifically, the solutions incorporate data backup and disaster recovery of systems and applications. Alibaba Cloud ECS allows you to use snapshots and images for data backup.

Disaster recovery methods

- Snapshot backup

Alibaba Cloud ECS allows you to back up system disks and data disks with snapshots. Currently, 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, helping to reduce impact on business I/O. When snapshots are used for data backup, the first backup is a full backup, followed by incremental backups. The backup duration depends on the amount of data to be backed up.



As shown in the preceding figure, Snapshot 1, Snapshot 2, and Snapshot 3, are the first, second, and third snapshots of a disk. The file system checks the disk data by blocks. 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 snapshot backup. With automatic backup, you can specify the time of day (24 options, on the hour), recurring day of week (Monday through Sunday), and retention time for snapshot creation. The retention time is customizable, and you can set a value from 1 to 65,536 days or choose to save snapshots permanently.

- Snapshot rollback

If exceptions occur in your system and you need to roll back a disk to a previous state, you can [roll back the disk](#) so long as it has a corresponding snapshot created.

Note:

- Rolling back a disk is an irreversible action. After disk rollback is completed, data cannot be restored. Exercise caution when performing this action.
- After a disk is rolled back, data will be irretrievably lost from the creation time of the snapshot to the current time.

- Image backup

An image file is equivalent to a replica file that contains all the data from one or more disks (a system disk or both the system disk and data disks). All image backups are full backups and can only be triggered manually.

- Image recovery

You can create custom images from snapshots to include the operating system and data environment in the image. The custom images can then be used to create multiple instances with the same operating system and data environment. For the configuration of snapshots and images, see [Snapshots](#) and [Images](#).



Note:

Custom images cannot be used across regions.

Technical metrics

RTO and RPO: related to the amount of data, usually at an hourly level.

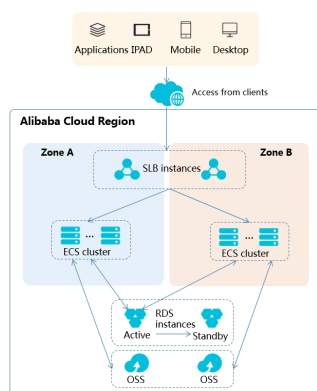
Scenarios

- Backup and restoration

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 data errors caused by application errors, or hackers exploiting application vulnerabilities for malicious access, 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 purchase new ECS instances with a custom image.

- Disaster recovery application

Alibaba Cloud ECS supports the implementation of disaster recovery architecture. For example, you can buy and use a Server Load Balancer (SLB) at the front end of an application, and deploy at least two ECS instances at the back end of the same application. Alternatively, you can implement an Auto Scaling solution using the auto scaling technology provided by Alibaba Cloud by defining how to use the ECS resources. In this way, even if one of the ECS instances fails or resources are overused, business continuity will not be interrupted, thus realizing disaster recovery. Take the deployment of ECS instances in two Internet Data Centers (IDCs) in the same city for example:



- A cluster of ECS instances is deployed in both IDCs. At the access side, SLBs are used for load balancing between the two IDCs.
- The Region Master nodes in both IDCs are identical and operate in active/standby mode. The failure of one node does not affect the ECS control function.
- To switch the control node of ECS instances in the case of IDC failure, the middleware domain name is associated anew as it is used for controlling the cluster. If the IDC of the control node experiences problems, the middleware domain name needs to be associated with the control node of the other IDC.

3 Data recovery

3.1 How to restore the data that is deleted by mistake

By taking CentOS 7 for example, this document introduces how to use Extundelete, an open source tool, to quickly restore accidentally deleted data.

Overview

In practice, data may be deleted accidentally. In this case, how to restore the data quickly and effectively? Alibaba Cloud offers several ways to restore data, for example :

- Roll back a [snapshot](#) or [custom image](#) through the ECS console.
- Purchase several ECS instances to implement [load balancing](#) and high availability for your services.
- Use [Object Storage Service \(OSS\)](#) to store a massive amount of data such as web pages, images, and videos.

There are a variety of open source data recovery tools for Linux, such as debugfs, R-Linux, ext3grep, Extundelete, and more. Of them, ext3grep and Extundelete are generally used. Both tools adopt the same recovery techniques, just that Extundelete is more powerful.

Extundelete is a Linux-based open source data recovery software. When using Linux instances, you can install this tool conveniently to quickly restore the data deleted accidentally as no Recycle Bin is available in Linux.

Extundelete can locate the position of an inode block by combining the inode information and logs so as to search for and restore the desired data. This powerful tool supports the disk-wide restoration of ext3/ext4 dual-format partitions.

Once data is deleted accidentally, firstly you need to unmount the disk or disk partition that contains the deleted data. This is because after a file is deleted, only the inode pointers of the file are zeroed while the actual file is still stored on the disk. If the disk is mounted in read/write mode, data blocks of the deleted file may be reallocated by the operating system. Once the data blocks are overwritten by new data, the original data will be lost completely, and cannot be restored by any means.

Therefore, mounting a disk in read-only mode can reduce the risk of overwriting the data in blocks, thus improving the chances of restoring the data successfully.



Note:

During the online restoration process, do not install Extundelete on the disk that has the deleted file. Otherwise, the data to be restored might be overwritten. Keep in mind to back up the disk by taking a snapshot before any operations.

Intended audience

- Users who accidentally delete files on a disk and no write operations have been performed on the disk after the deletion.
- Users whose websites have low traffic and who have few ECS instances.

Procedure

Software release: e2fsprogs-devel e2fsprogs gcc-c++ make (compiler and more)
Extundelete-0.2.4.



Note:

The libext2fs 1.39 or above is required for the normal operation of Extundelete. For ext4 support, however, make sure e2fsprogs 1.41 or higher is provided (you can run the command `dumpe2fs` to check the version output).

The above releases are available when this document is being written. Your downloads may be different.

- Deploy Extundelete

```
wget http://zy-res.oss-cn-hangzhou.aliyuncs.com/server/extundelet_e-0.2.4.tar.bz2
yum -y install bzip2 e2fsprogs-devel e2fsprogs gcc-c++ make # Install related dependencies and libraries
tar -xvzf extundelet_e-0.2.4.tar.bz2
cd extundelet_e-0.2.4 #
Enter the program directory
```

```
./ configure                                     # Installed
successful ly      as      shown      below
```

```
extundelete-0.2.4/src/Makefile.am
extundelete-0.2.4/configure.ac
extundelete-0.2.4/depcomp
extundelete-0.2.4/Makefile.in
extundelete-0.2.4/Makefile.am
[root@i-XXXXXXXXXX ~]# cd extundelete-0.2.4
[root@i-XXXXXXXXXX extundelete-0.2.4]# ./configure
Configuring extundelete 0.2.4
Writing generated files to disk
[root@i-XXXXXXXXXX extundelete-0.2.4]#
```

```
make && make install
```

At this point, the src directory appears. It contains an Extundelete executable file and a corresponding path. As shown below, the default file is installed in `usr / local / bin`, and the following demo is made in the `usr / local / bin` directory.

- Delete a file and use Extundelete to restore it

1. Check the available disks and partitions of your ECS instance, then format and partition the `/dev/vdb` partition. For more information about formatting and partitioning, see [Format and mount a data disk](#).

```
fdisk -l
```

```
Disk label type: dos
Disk identifier: 0x0000efd2

   Device Boot      Start         End      Blocks    Id  System
/dev/vda1  *           2048     83886079     41942016   83  Linux

Disk /dev/vdb: 21.5 GB, 21474836480 bytes, 41943040 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
```

2. Mount the partitioned disk under the `/zhuyun` directory, and then create a file named `hello`.

```
mkdir / zhuyun                                     # Create the
zhuyun directory .
mount / dev / vdb1 / zhuyun                       # Mount the
disk under the zhuyun directory .
```



```
echo test > hello # Create a
test file .
```

3. Run the `md5sum` command to generate the MD5 value of the file and note it down. You can compare the MD5 values of the file before and after the deletion to verify its integrity.

```
md5sum hello
```



```
[root@iZ zhuyun]# md5sum hello
d8e...49 hello
```

4. Delete the `hello` file.

```
rm -rf hello
cd ~
fuser -k / zhuyun # Terminate the
process tree that uses a certain partition (skip
this if you are sure that no resources are
occupied ).
```

5. Unmount the data disk

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

6. Use `Extundelete` to restore the file.

```
extundelet e -- inode 2 / dev / vdb1 # Query the
contents in a certain inode . Using " 2 " means to
search the entire partition . To search a directory
```

, just specify the inode and directory . Now you can see the deleted file and inode .

```
Direct blocks: 127754, 4, 0, 0, 1, 9252, 0, 0, 0, 0, 0, 0
Indirect block: 0
Double indirect block: 0
Triple indirect block: 0
```

File name	Inode number	Deleted status
..	2	
..	2	
lost+found	11	
hello	12	Deleted

```
/usr/local/bin/extundelet e --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. Check whether the file is restored.

```
[root@izbp13micdqs12364umm8aZ /]# ll RECOVERED_FILES/
total 4
-rw-r--r-- 1 root root 5 Mar  8 14:20 hello
```

Check the MD5 values of the files before and after deletion. If they are the same, restoration is successful.

```
--restore -inode 12 # --restore -inode
Restore by the specified inode .
--extundelet e --restore -all # --restore -all
Restore all .
```

3.2 Data restoration in Linux instances

When solving problems related to disks, you may frequently encounter the loss of data disk partitions. This article describes common data partition loss problems and corresponding solutions in Linux, and provides common mistakes and best practices for cloud disks to avoid possible risks of data loss.

Before restoring data, you must create snapshots for data disks that lose partitions. If problems occur during the restoration process, you can roll back data disks to the status before restoration.

Prerequisites

Before restoring data, you must create snapshots for data disks that lose partitions. If problems occur during the restoration process, you can roll back data disks to the status before restoration.

Introduction to disk management tools

You can select one of the following tools to fix the disk partition and restore the data in a Linux instance:

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

Handle data disk partition loss and data restoration in Linux

After you restart a Linux instance, you may encounter data disk partition loss or data loss issues. This may be because you have not set the partitions to be mounted automatically on startup of the instance in the `etc / fstab` file. In this case, you can manually mount the data disk partition first. If the system prompts partition table loss when you manually mount the data disk, you can try to solve the problem through the following three methods: [Restore partitions by using fdisk](#), [Restore partitions by using testdisk](#), or [Restore data by using testdisk](#).

- Restore partitions by using fdisk

Default values usually apply to the starting and ending sectors of the partition when you partition a data disk. You can then directly use fdisk to restore the

partition. For more information about this tool, see [Linux Format and mount a data disk](#).

```
[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 operations do not help, you can try testdisk for the restoration.

- Restore partitions by using testdisk

Here we suppose the cloud disk device is named `/dev/xvdb`. Follow these steps to restore the partitions by using testdisk:

1. Run `testdisk /dev/xvdb` (replace the device name as appropriate), and then select `Proceed` (default value) and 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: *Intel* by default. If your data disk uses the GPT format, select *EFI GPT*.

```
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 then 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. If you cannot see any partition, select *Quick Search* and then press the Enter key for a quick search.

```

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 returned result, 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 searching if the expected partition is not listed.

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

[ quit ] [Deeper search] >[ write ]
                        write partition structure to disk
```

7. Press the Y key to save the partition.

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

write partition table, confirm ? (Y/N)
```

8. Run `partprobe /dev/xvdb` (replace the device name as appropriate) to refresh the partition table manually.
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.qz  logs  lost+found  test
```


- Restore data by using testdisk

In some cases, you can use testdisk to scan and locate the disk partition, but you cannot save the partition. In this case, you can try to restore files directly. Follow these steps:

- Find the partition following Step 1 to Step 4 described in [Restore partitions by using testdisk](#).
- List files by pressing the P key. The returned result is shown in the following figure.

```
* Linux
Directory /
0 32 33 652 180 40 10483712
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

Next
Use Right to change directory, h to hide deleted files
q to quit, : to select the current file, a to select all files
C to copy the selected files. c to copy the current file
```

- Select the files to restore, and press the C key.
- Select a directory. In this example, the file is restored and copied to the / home directory.


```

Please select a destination where /ip.gz will be copied.
Keys: Arrow keys to select another directory
      C when the destination is correct
      Q to quit
Directory /
drwxr-xr-x  0  0  4096 11-Jan-2017 09:32 .
drwxr-xr-x  0  0  4096 11-Jan-2017 09:32 ..
dr-xr-xr-x  0  0  4096 25-Jul-2016 16:23 boot
drwxr-xr-x  0  0  2940 21-Feb-2017 12:30 dev
drwxr-xr-x  0  0  4096 21-Feb-2017 12:12 etc
>drwxr-xr-x  0  0  4096 16-Feb-2017 11:48 home
drwx-----  0  0 16384 12-May-2016 19:58 lost+found
drwxr-xr-x  0  0  4096 12-Aug-2015 22:22 media
drwxr-xr-x  0  0  4096 21-Feb-2017 11:57 mnt
drwxr-xr-x  0  0  4096 12-Aug-2015 22:22 opt
dr-xr-xr-x  0  0    0 16-Feb-2017 21:35 proc
dr-xr-xr-x  0  0  4096 21-Feb-2017 11:57 root
drwxr-xr-x  0  0   560 21-Feb-2017 12:12 run
drwxr-xr-x  0  0  4096 12-Aug-2015 22:22 srv
dr-xr-xr-x  0  0    0 16-Feb-2017 21:35 sys
drwxrwxrwt  0  0  4096 21-Feb-2017 12:34 tmp
drwxr-xr-x  0  0  4096 16-Feb-2017 11:48 usr
drwxr-xr-x  0  0  4096 16-Feb-2017 21:35 var
lrwxrwxrwx  0  0    7  3-May-2016 13:48 bin
lrwxrwxrwx  0  0    7  3-May-2016 13:48 lib
lrwxrwxrwx  0  0    9  3-May-2016 13:48 lib64
lrwxrwxrwx  0  0    8  3-May-2016 13:48 sbin

```

If you see `Copy done ! 1 ok , 0 failed` , it indicates that copy was successful, 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 you can see files, it indicates that files have been restored successfully.

```

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

```

Common mistakes and best practices

Data is users' core asset. Many users establish websites and databases (MySQL/MongoDB/Redis) on ECS. Huge risks to the users' services may occur when data is lost. Common mistakes and best practices are summarized as follows.

- Common mistakes

The bottom layer of Alibaba Cloud block-level storage is based on [triplicate technology](#). Therefore, some users consider that no risk of data loss in the operating system exists. It is actually a misunderstanding. The three copies of data stored in the bottom layer provide physical layer protection for data disks. However, if problems occur to the cloud disk logic in the system, such as viruses, accidental data deletion, and file system damage, the data may still be lost. To guarantee data security, you have to use technologies such as Snapshot and backup.

- Best practices

Data disk partition restoration and data restoration are the final solutions for solving data loss problems, but it is never guaranteed. We strongly recommend that you follow the best practices to perform auto or manual snapshot on data and run different backup schemes to maximize your data security.

- Enable automatic snapshots

Automatic snapshots are enabled for the system disk and data disk based on actual service conditions. Note that automatic snapshot may be released when

the system disk is changed, the instance is expired, or the disk is manually released.

You log on to the ECS console to change the attributes of the disks to enable snapshot release with the disk. Disable snapshot release with the disk if you want to retain the snapshots.

For more information, see [FAQ about automatic snapshots](#).

- Create manual snapshots

Create snapshots manually before any important or risky operations such as:

- Upgrade the kernel
- Upgrade or change of applications
- Restoration of disk data

You must create snapshots for disks before restoring them. After the snapshots are completed, you can perform other operations.

- OSS, offline, or offsite backup

You can back up important data by means of OSS, offline, or offsite backup based on actual conditions.

3.3 Data restoration in Windows instances

When solving problems related to disks, you may frequently encounter the loss of data disk partitions. This article describes common data partition loss problems and corresponding solutions in Windows, and provides common mistakes and best practices for cloud disks to avoid possible risks of data loss.

Prerequisites

Before restoring data, you must create snapshots for data disks that lose partitions. If problems occur during the restoration process, you can roll back data disks to the status before restoration.

Introduction to disk management tools

In Windows instances, you can select either of the following tools for restoring data disk data:

- **Disk Management:** A tool provided by Windows for partitioning and formatting the disk.

- **Data restoration software:** Generally, they are commercial software, and can be downloaded from the providers' official websites. They are mainly used for restoring data in an abnormal file system.

Status of the disk is Foreign and no partitions are displayed

In the Disk Management of Windows, the disk is in the Foreign status and displays no partitions.

Solution:

Right click the Foreign disk, select Import Foreign Disks, and then click OK.

Status of the disk is Offline and no partitions are displayed

In the Disk Management of Windows, the disk is in the Offline status and displays no partitions.

Solution:

Right click the Offline disk (for example, Disk 1), select Online, and then click OK.

No drive letter assigned

In the Disk Management of Windows, you can view data disk information, but no drive letter is allocated to the data disk.

Solution:

Right click primary partition of the disk (for example, Disk 1), click Change drive letter and paths, and then complete operations by prompt.

Error occurred during storage enumeration

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



Note:

Some versions may report Error occurred during enumeration of volumes. They are the same.

Solution:

1. Start Windows PowerShell.
2. Run `winrm quickconfig` for restoring. When “Make these changes [y/n]?” is displayed on the interface, you must type `y` to run the command.

After the restoration, you can have the data disks in the Disk Management.

Data disk is in RAW format

In some special circumstances, the disk in Windows is in RAW format.

If the file system of a disk is unrecognizable to Windows, it is displayed as a RAW disk. This usually occurs when the partition table or boot sector that records the type or location of the file system is lost or damaged. Common causes are listed as follows:

- Safely remove hardware is not used when disconnecting the external disk.
- Disk problems caused by power outages or unexpected shutdown.
- Hardware layer failure may also cause information loss of the disk partition.
- Bottom layer drivers or disk-related applications. For example, DiskProbe can be used to directly modify the disk table structure.
- Computer viruses.

For more information about how to fix these problems, see [Dskprobe Overview](#) document.

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

Common mistakes and best practices

Data is users' core asset. Many users establish websites and databases (MYSQL/MongoDB/Redis) on ECS. Huge risks to the users' services may occur when data is lost. Common mistakes and best practices are summarized as follows.

- Common mistakes

The bottom layer of Alibaba Cloud block-level storage is based on [triplicate technology](#). Therefore, some users consider that no risk of data loss in the operating system exists. It is actually a misunderstanding. The three copies of data stored in the bottom layer provide physical layer protection for data disks. However, if problems occur to the cloud disk logic in the system, such as viruses, accidental data deletion, and file system damage, the data may still be lost. To guarantee data security, you have to use technologies such as Snapshot and backup.

- Best practices

Data disk partition restoration and data restoration are the final solutions for solving data loss problems, but it is never guaranteed. We strongly recommend that you follow the best practices to perform auto or manual snapshot on data and run different backup schemes to maximize your data security.

- Enable automatic snapshots

Automatic snapshots are enabled for the system disk and data disk based on actual service conditions. Note that automatic snapshot may be released when the system disk is changed, the instance is expired, or the disk is manually released.

You log on to the ECS console to change the attributes of the disks to enable snapshot release with the disk. Disable snapshot release with the disk if you want to retain the snapshots.

For more information, see [FAQ about automatic snapshots](#).

- Create manual snapshots

Create snapshots manually before any important or risky operations such as:

- Upgrade the kernel
- Upgrade or change of applications
- Restoration of disk data

You must create snapshots for disks before restoring them. After the snapshots are completed, you can perform other operations.

- OSS, offline, or offsite backup

You can back up important data by means of OSS, offline, or offsite backup based on actual conditions.

4 Configuration preference

4.1 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.

Parameter	Description
-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.
-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: `nc 192 . 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 6rnresult r n" | nc 192 . 168 . 2 . 34 11211 .`
- To obtain data, run the command `printf "get key r n" | nc 192 . 168 . 2 . 34 11211 .`
- To delete data, run the command `printf "delete key r n" | nc 192 . 168 . 2 . 34 11211 .`
- To view the status, run the command `printf "stats r n" | 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_all r n" | 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:

```
[root@babu> /tsmserv] $ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (//.ssh/id_rsa):
Created directory ''.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in //.ssh/id_rsa.
Your public key has been saved in //.ssh/id_rsa.pub.
The key fingerprint is:
01:18:ba:b1:1d:27:3a:35:3c:8f:ed:11:49:57:9b:04 root@babu
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      .oo Eoo      |
|     o.. + . o     |
|    o B + . o     |
|   B X . .       |
|  = o + S        |
|   . . .         |
|   .             |
|                 |
+-----+
[root@babu> /tsmserv] $
```

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 e file_target t
```

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

```
scp local_file remote_use rname @ remote_ip : remote_fol der
scp local_file remote_use rname @ remote_ip : remote_fil e
scp local_file remote_ip : remote_fol der
scp local_file remote_ip : remote_fil e
```



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 means 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.
-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 connection s = 10 # Specify the maximum concurrent connection s .
strict modes = yes # Specify whether to check the permission s 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 concurrent connection s . By default , the lock file is /var/run/rsyncd.lock .
```

```

motd file = / usr / local / rsyncd / rsyncd . motd      # Define
server informatio n 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 the value to yes .
write only = no                                      # To allow the
client to download files , set the value to no .
Otherwise , set 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 names .
secrets file = / etc / rsyncd . pass                 # Specify the
password file .

```

4.2 Increase data throughput through read/write split

This topic describes how to split read and write operations at the application layer and system layer. To improve overall system performance and optimize user experience, you can reduce the load of your primary database through read/write split.

Split read and write at the application layer

At the application layer, read/write split is implemented through coding. Before the Service method is called, AOP is used to decide 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 starting with `query` , `find` or `get` , and the write database is used for others.

Advantages

- The program easily switches between multiple data sources automatically.
- Middleware is not required.

- Currently, any databases are supported.

Disadvantages

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

Split read and write at the system layer

You can use either the Distributed Relational Database Service (DRDS) or the middleware MySQL Proxy to split read and write operations at the system layer.

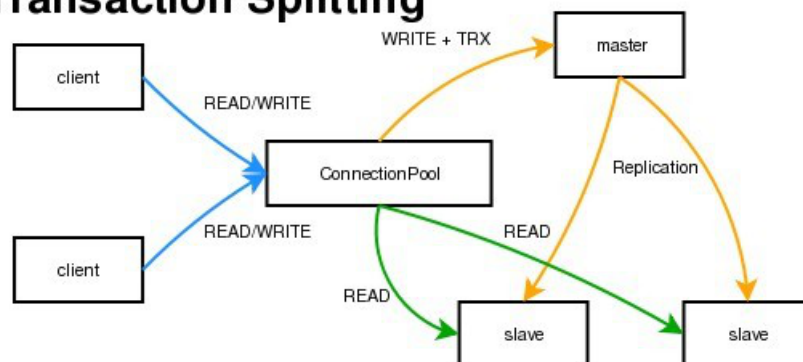
The following procedure describes how to implement read/write split by using MySQL Proxy.

MySQL Proxy overview

MySQL Proxy is a simple program that is situated between your client and MySQL server and that can monitor, analyze, or transform their communication. Its flexibility allows for a wide variety of uses, including load balancing, failover, query analysis, and query filtering and modification.

MySQL Proxy principle

Transaction Splitting



MySQL Proxy is an intermediate-layer proxy that acts as a connection pool to forward connection requests from the front-end application to the back-end database. By using the lua script, MySQL Proxy can perform complex connection control and filtering to implement read/write split and load balancing. MySQL Proxy is transparent to the application, which only needs to be connected to the listening port of MySQL Proxy. In this case, the proxy server may be considered as a Single Point of Failure (SPOF), but the use of multiple proxy servers implements redundancy. Therefore, you only need to configure multiple proxy connections in the connection pool of the application server.

Advantages

- Read/write split 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, making it difficult to switch databases.
- Performance decreases because the middleware serves as a forwarding proxy.

Procedure

Environment

- Master database IP address: 121.40.18.26
- Slave database IP address: 101.37.36.20
- MySQL Proxy IP address: 116.62.101.76

Preparations

1. Create three ECS instances and install MySQL.
2. Build master/slave databases and ensure data consistency.

Primary environment

1. Modify the MySQL configuration file.

```
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
```

2. Restart the MySQL service on the master/slave servers.

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

3. Create an account on the master server and grant permissions to the slave server.

```
mysql - uroot - p95c758678 3
grant replicatio n slave on *.* to ' syncms '@' Enter
slave - IP ' identified by ' 123456 ';
```



```
flush privileges ;
```

4. View the primary database status.

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

```
mysql> show master status;
+-----+-----+-----+-----+-----+
| File           | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+-----+-----+-----+-----+-----+
| mysql-bin.000005 | 602      |              |                  |                  |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

5. Configure the secondary database.

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

6. Start the slave synchronization process and view the status.

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

```
mysql> show slave status\G
***** 1. row *****
      Slave_IO_State: Waiting for master to send event
      Master_Host: 
      Master_User: syncms
      Master_Port: 3306
      Connect_Retry: 60
      Master_Log_File: mysql-bin.000007
      Read_Master_Log_Pos: 154
      Relay_Log_File: iZ...Z-relay-bin.000003
      Relay_Log_Pos: 367
      Relay_Master_Log_File: mysql-bin.000007
      Slave_IO_Running: Yes
      Slave_SQL_Running: Yes
      Replicate_Do_DB: 
      Replicate_Ignore_DB: 
      Replicate_Do_Table: 
      Replicate_Ignore_Table: 
      Replicate_Wild_Do_Table: 
      Replicate_Wild_Ignore_Table: 
      Last_Errno: 0
      Last_Error:
```

7. Verify master/slave synchronization.

Operations on 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 ;
```

```
mysql> create database testproxy;
Query OK, 1 row affected (0.01 sec)

mysql> create table testproxy.test1(ID int primary key,name char(10) not null);
Query OK, 0 rows affected (0.07 sec)

mysql> insert into testproxy.test1 values(1,'one');
Query OK, 1 row affected (0.02 sec)

mysql> insert into testproxy.test1 values(2,'two');
Query OK, 1 row affected (0.03 sec)

mysql> select * from testproxy.test1;
+----+-----+
| ID | name |
+----+-----+
| 1 | one |
| 2 | two |
+----+-----+
2 rows in set (0.01 sec)
```

Operations on the secondary database

Search the secondary database for the data of the `testproxy . test1` table. If the data matches that in the primary database, master/slave synchronization is successful.

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

```
mysql> select * from testproxy.test1;
+----+-----+
| ID | name |
+----+-----+
| 1 | one |
| 2 | two |
+----+-----+
2 rows in set (0.00 sec)
```

Configure read/write split

1. Install MySQL Proxy.

```
wget https://cdn.mysql.com/archives/mysql-proxy/mysql-proxy-0.8.5-linux-glibc2.3-x86-64bit.tar.gz
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
```

```
[root@: ~]# mysql-proxy -V
mysql-proxy 0.8.5
chassis: 0.8.5
glib2: 2.16.6
libevent: 2.0.21-stable
LUA: Lua 5.1.4
package.path: /alidata/mysql-proxy-0.8.5/lib/mysql-proxy/lua/?..lua;
package.cpath: /alidata/mysql-proxy-0.8.5/lib/mysql-proxy/lua/?..so;
-- modules
proxy: 0.8.5
```

3. Set the read-write split 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 split will not be performed. By default, read/write split will be performed for at least four connections and at most eight connections. To simplify testing read/write split, the number of connections is modified to one at least and two at most, and can be adjusted in real application environments based on your specific needs.

Before the modification

```
-- connection pool
if not proxy.global.config.rwsplit then
    proxy.global.config.rwsplit = {
        min_idle_connections = 4,
        max_idle_connections = 8,

        is_debug = false
    }
end
```

After the modification

```
-- connection pool
if not proxy.global.config.rwsplit then
    proxy.global.config.rwsplit = {
        min_idle_connections = 1,
        max_idle_connections = 2,

        is_debug = true
    }
end
```

4. Copy the lua administration script (*admin . lua*) to the directory where the read-write split 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 /
```

Grant permissions

1. Grant permissions in the master database. The permissions will also be granted in the slave database due to master/slave synchronization.

```
mysql - uroot - p95c758678 3
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
master - IP : 3306 - r Enter slave - 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. View the port and relevant processes.

```
netstat - tln
```

```
[root@ ~]# netstat -tln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN      826/sshd
tcp        0      0 0.0.0.0:4040           0.0.0.0:*               LISTEN      22767/mysql-proxy
tcp        0      0 0.0.0.0:4041           0.0.0.0:*               LISTEN      22767/mysql-proxy
```

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

```
[root@ ~]# ps -ef | grep mysql
root      22767      1  0 10:59 ?        00:00:00 /alidata/mysql-proxy-0.8.5/libexec/mysql-proxy --daemon --l
og-level=debug --log-file=/var/log/mysql-proxy.log --plugins=proxy -b :3306 -r :330
6 --proxy-lua-script=/alidata/mysql-proxy-0.8.5/share/doc/mysql-proxy/rw-splitting.lua --plugins=admin --ad
min-username=admin --admin-password=admin --admin-lua-script=/alidata/mysql-proxy-0.8.5/share/doc/mysql-pro
xy/admin.lua
root      22794  22602  0 11:02 pts/0    00:00:00 grep --color=auto mysql
```

Test read/write split

1. Disable slave replication.

```
stop slave ;
```

2. Log on to the back end of MySQL Proxy.

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

```
MySQL [(none)]> select * from backends;
+-----+-----+-----+-----+-----+-----+
| backend_ndx | address          | state  | type | uuid | connected_clients |
+-----+-----+-----+-----+-----+-----+
| 1 | [REDACTED]:3306 | unknown | rw  | NULL | 0 |
| 2 | [REDACTED]:3306 | unknown | ro  | NULL | 0 |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

The first connection will access the master database.

```
mysql -umysql - proxy - p123456 - h 116 . 62 . 101 . 76 - P 4040
insert into testproxy . test1 values ( 3 , ' three ' );
# Add a data record . Slave replication is disabled . Therefore , the record exists in the primary database but does not exist in the secondary database .
```

```
[root@~]# mysql -umysql-proxy -p123456 -h [REDACTED] -P 4040
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.7.17-log MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> insert into testproxy.test1 values(3,'three');
Query OK, 1 row affected (0.03 sec)

MySQL [(none)]>
```

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

```
mysql -umysql - proxy - p123456 - h 116 . 62 . 101 . 76 - P 4040
```

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

```
MySQL [(none)]> select * from testproxy.test1
-> ;
+----+-----+
| ID | name |
+----+-----+
| 1 | one  |
| 2 | two  |
+----+-----+
2 rows in set (0.00 sec)

MySQL [(none)]> insert into testproxy.test1 values(9,'nine')
-> ;
Query OK, 1 row affected (0.02 sec)

MySQL [(none)]> select * from testproxy.test1
-> ;
+----+-----+
| ID | name |
+----+-----+
| 1 | one  |
| 2 | two  |
+----+-----+
2 rows in set (0.00 sec)
```

4.3 Change the language settings of a Windows instance

This topic uses an instance running the Windows Server 2016 public image as an example to describe how to download the German language pack from Windows Update and configure the language settings of an ECS instance by using the downloaded language pack. After you change the language settings for a target ECS instance, you can then use that instance to create a custom image based on the changed language settings.

Context

Alibaba Cloud ECS provides only Chinese and English editions of Windows Server public images by default. However, you can use other language editions (such as Arabic, German, Russian, or Japanese) by downloading language packs from Windows Update to change the language settings of an ECS instance. Note that the information provided in this topic applies to instances running the Windows Server 2012 or later operating systems.

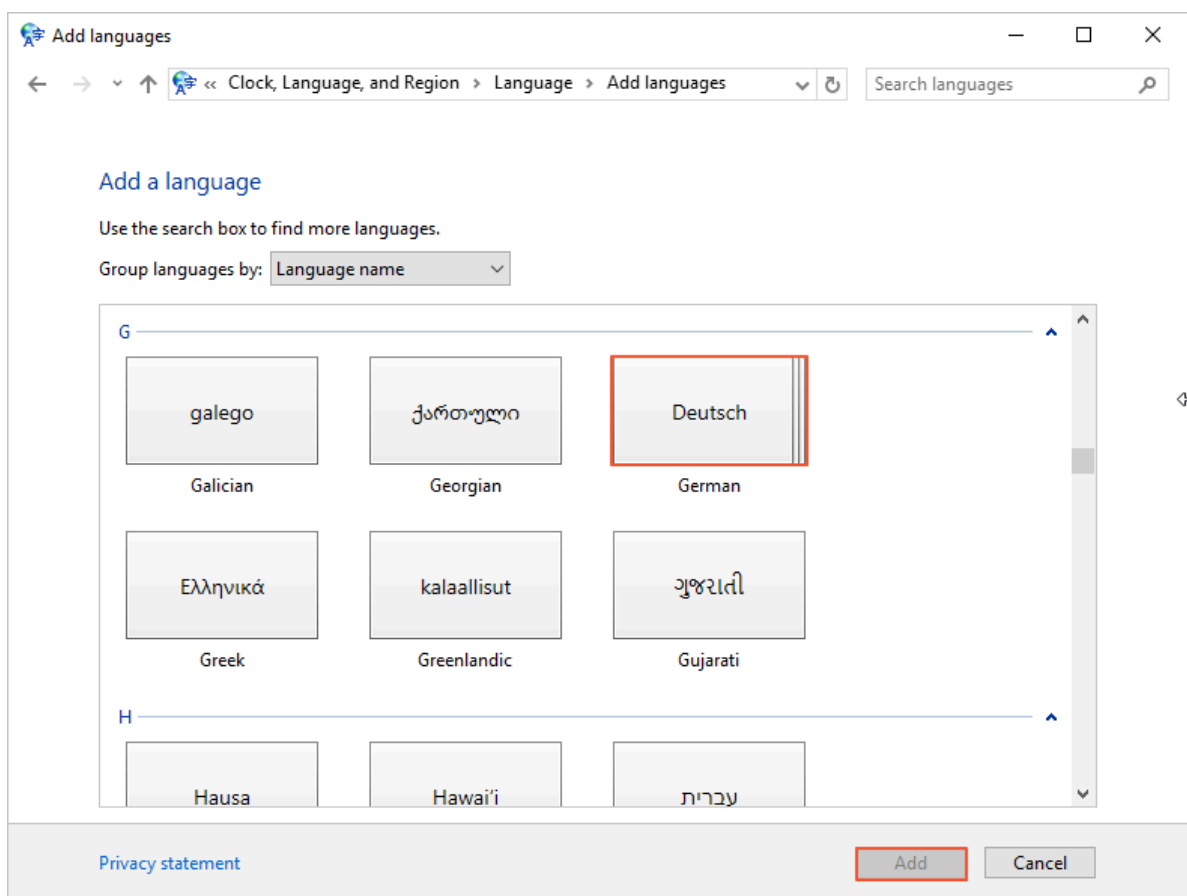
Procedure

1. Connect to the target Windows instance. For more information, see [#unique_66](#).
2. Open the PowerShell module.

3. Run the following commands to temporarily disable Windows Server Update Services (WSUS).

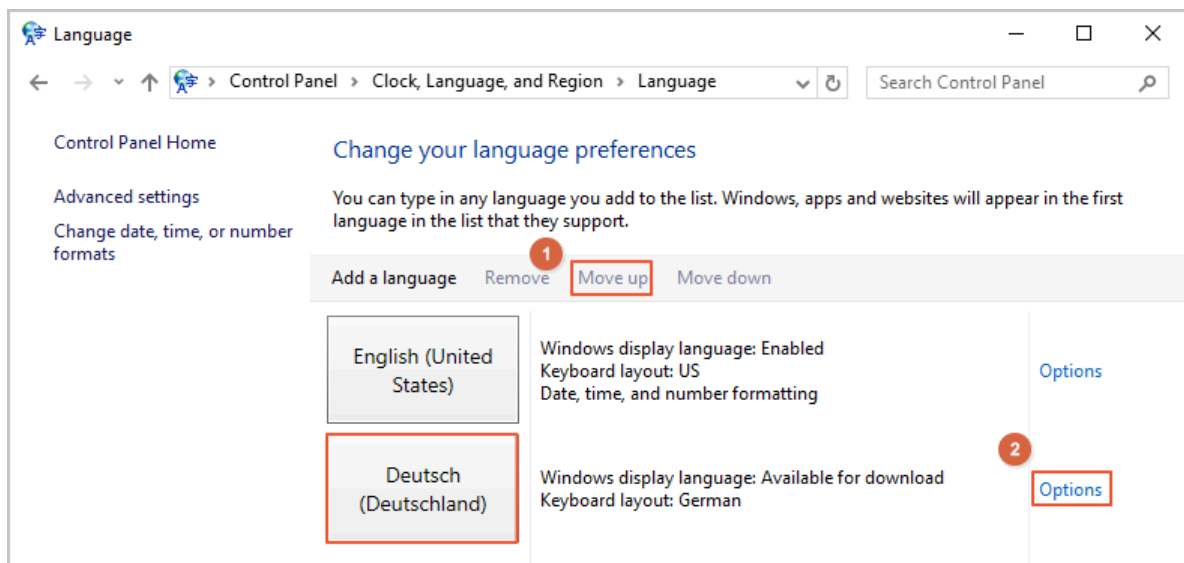
```
Set - ItemProperty - Path 'HKLM:\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate\AU' - Name UseWUService - Value 0
Restart - Service - Name wuauserv
```

4. Find the Control Panel, click Clock, Language, and Region > Language > Add a language.
5. In the Add languages dialog box, select a language, for example, Deutsch (German) > Deutsch (Deutschland), and click Add.

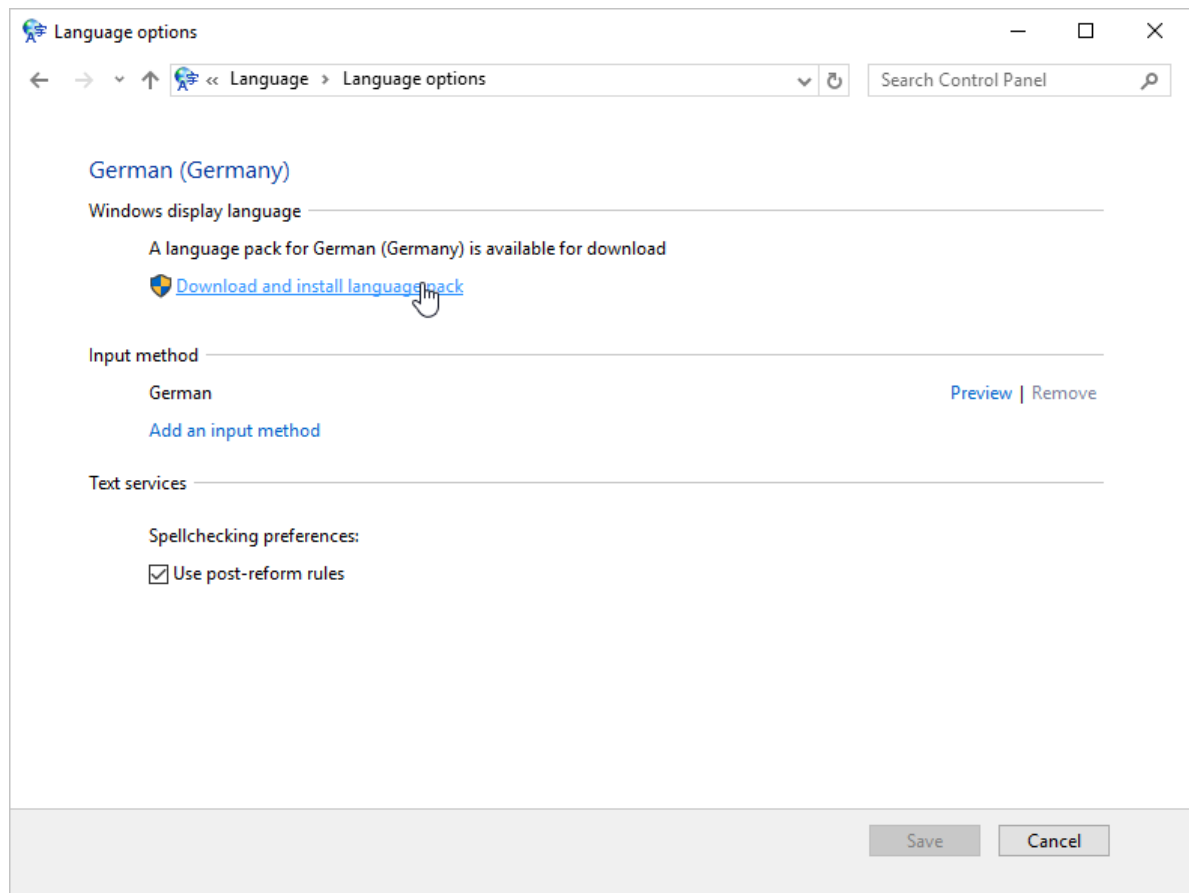


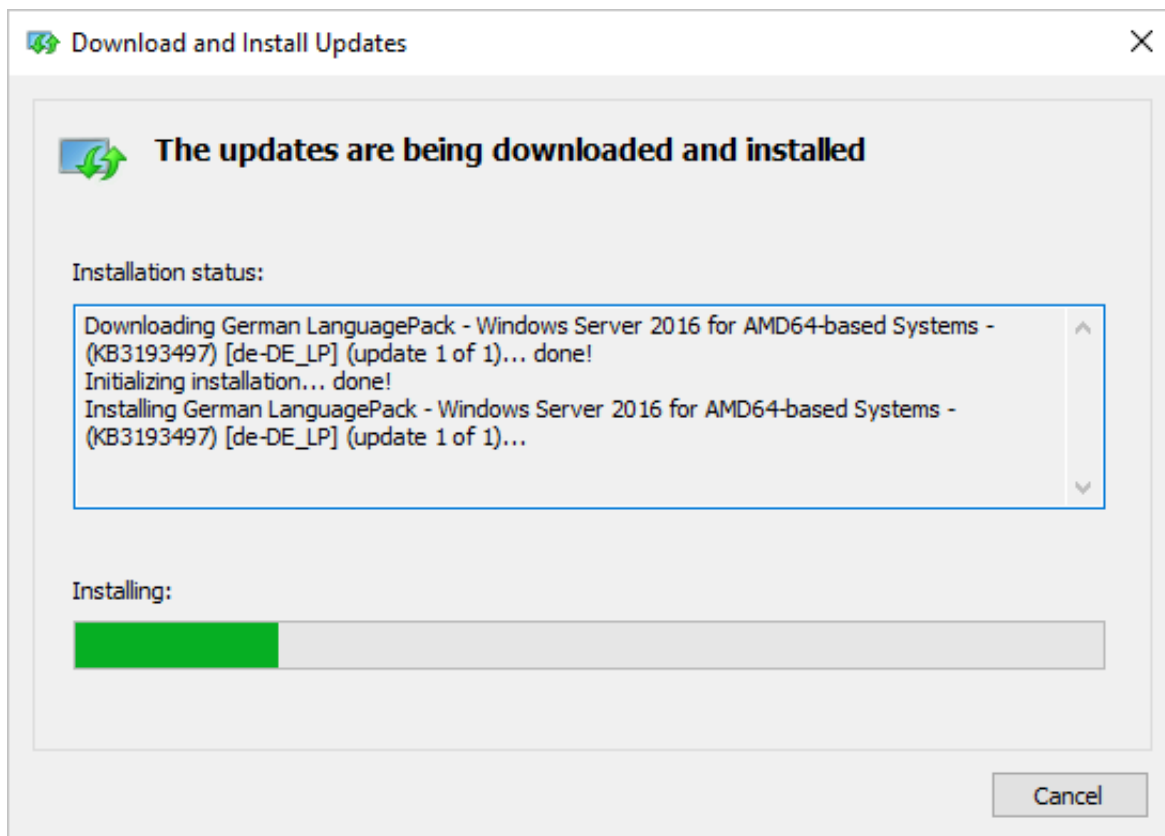
6. Select the language, such as Deutsch (Deutschland), and click Move up to change the language priority.

7. Click Options next to the selected language to check online for language updates.



8. Wait for about three minutes while the instance checks for updates. Once the update is available for download, click Download and install language pack and wait until the installation is complete.





9. [Restart your instance](#) in the ECS console.
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' -Name UseWUService -Value 1
Restart-Service -Name wuauserv
```

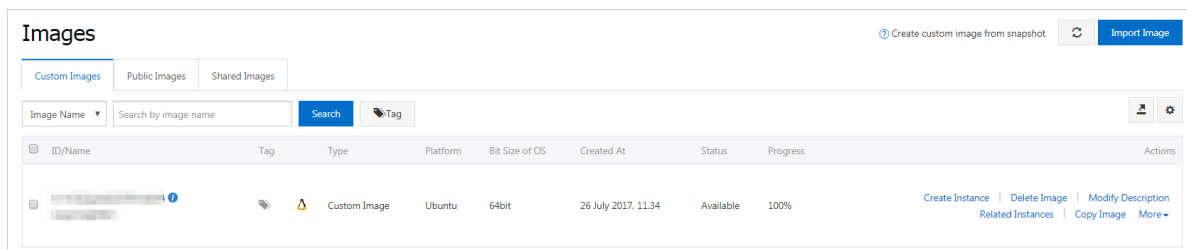
12. Open Windows Update, check for security updates, and re-install all the security updates that are already done before the language settings.

What's next

Create multiple instances with the same language settings

1. Log on to the [ECS console](#).
2. [Create a custom image](#) by using the Windows instance with the new display language.

3. Create a specified number of instances from the custom image.



5 Packer: machine images as code

5.1 Comparison between creating a custom image through ECS against using the tool Packer

This topic compares the procedures of how to create a custom image by using an instance and how to build a custom image by using Packer. With Packer, you can automate the creation of ECS images by specifying basic image building information, software, and settings in a JSON configuration file.

What is Packer?

Packer is an open source image building tool created by HashiCorp that automates the creation of images. The image building process is based on a JSON configuration file. This means that you can build an unlimited number of identical images. Packer also facilitates the testing and updating images, which helps to reduce O&M time and lower overall costs. For more information, visit [the official Packer website](#).

Image building methods

The following table describes the image building methods supported by Alibaba Cloud.

Table 5-1: Image building methods

Method	Prerequisite	Advantage	Disadvantage
#unique_73	If you have created a system disk snapshot, you can create a custom image by using the ECS console or calling the corresponding API.	<ul style="list-style-type: none">• The operations are easy to complete.• Resources are reused.• Images are created in real time based on the target production environment.• Images are based on Alibaba Cloud public images.	<ul style="list-style-type: none">• Software pre-installation and subsequent settings can be complicated.• Correct and consistent manual operations cannot be guaranteed.• High maintenance costs may be incurred.
#unique_74	If you have created an ECS instance , you can create a custom image by using the ECS console or calling the corresponding API.		

Method	Prerequisite	Advantage	Disadvantage
#unique_75	Packer requests can be identified. User information is verified by using the AccessKey.	<ul style="list-style-type: none"> • You do not need to create an instance or a snapshot. • The JSON file is reusable and modifiable. • Complete operation logs are recorded. • Temporary resources are automatically released. • ISO files are automatically converted and then immediately imported to Alibaba Cloud ECS. • Images are based on Alibaba Cloud public images and local ISO files. 	Users are recommended to have prior knowledge of Packer.

Background information

The examples provided in this topic use the following information:

- Target region: China (Beijing). For more information, see [#unique_41](#).
- Operating system: CentOS 7.3 64-bit. The two procedures in this topic use the public image centos_7_03_64_20G_alibase_20170818.vhd as an example. You can query the image ID list of other operating systems by using the [ECS console](#) or calling [DescribeImages](#).
- Custom service: Redis.
- Whether to reserve temporary resources: No.

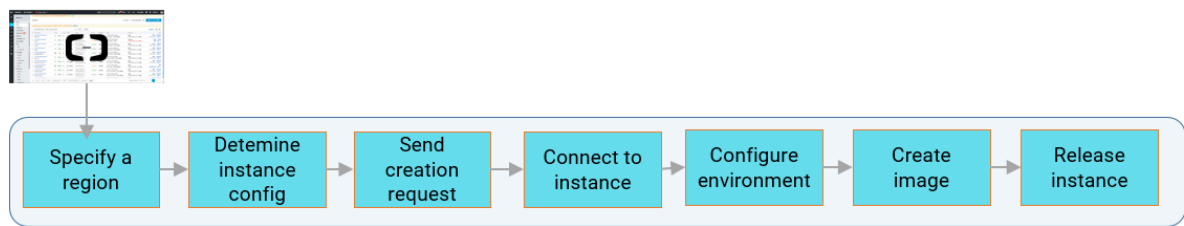


Note:

Resources created in the following procedures (such as an instance, public IP address, and snapshot) may incur fees if retained under your account. We recommend that you release these resources after you have created a custom image to avoid excessive fees.

Create a custom image by using an instance

This example demonstrates how to create a custom image by using the ECS console. The following figure shows the procedure.



1. Log on to the [ECS console](#).
2. In the left-side navigation pane, click Instances.
3. Select the target region.
4. Purchase an ECS instance. For more information, see [#unique_77](#).

To minimize charges and simplify the procedure, you can use the following settings:

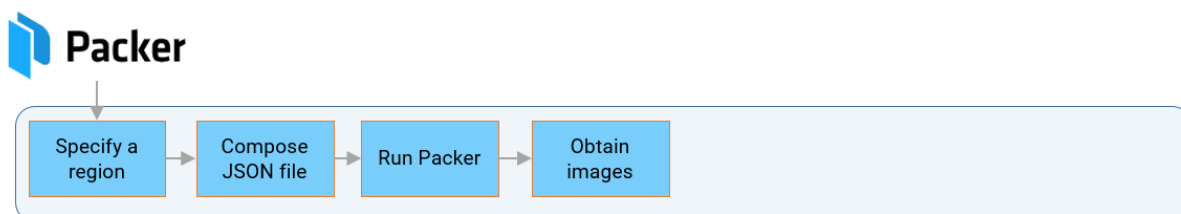
- Billing Method: Pay-As-You-Go. For more information, see [#unique_78](#).
- Instance Type: ecs.t5-1c1m1.small. For more information, see [#unique_79](#).
- Public Image: CentOS 7.3 64bit.
- Network: VPC.
- Security Group: default security group.
- Network Billing Method: If your instance does not need to access the Internet, you do not need to assign a public IP address. For more information, see [#unique_80](#).

5. Connect to the ECS instance. For more information, see [#unique_66](#)
6. Run the `yum install redis . x86_64 - y` command to install Redis.
7. Select China (Beijing) in the ECS console.
8. Create an image. For more information, see [#unique_74](#).
9. Choose Snapshots and Images > Custom Images to view the image status.

10. Optional. After the image is created, release the temporary resources, such as the created instance and EIP.

Build a custom image by using Packer

This example demonstrates how to create a custom image by using Packer. The following figure shows the procedure.



Prerequisites

Packer is installed. For more information, see [Install Packer](#) or [#unique_75](#).

Procedure

1. Create an `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_base_image",
      "source_image": "centos_7_0_3_64_20G_alibase_20170818.vhd",
      "ssh_username": "root",
      "instance_type": "ecs.t5-1c1m1.small",
      "internet_charge_type": "PayByTraffic",
      "io_optimized": "true"
    }
  ],
  "provisioners": [
    {
      "type": "shell",
      "inline": [
        "sleep 30",
        "yum install redis.x86_64 -y"
      ]
    }
  ]
}
```




```
}
```

Table 5-2: Packer parameter description

Parameter	Example value	Description
<code>variables{"variable1":"value"}</code>	<code>variables{"access_key":"{{env `ALICLOUD_ACCESS_KEY`}}"</code>	Defines the variables used in image builders . These variables are used to eliminate the disclosure of such information as the AccessKey (access_key and secret_key). The values of variables are input by the system during runtime.
<code>builders{"type":"value"}</code>	<code>builders{"type":"alicloud-ecs"}</code>	The image builders defined by Packer. Alibaba Cloud supports alicloud-ecs, also known as Alicloud Image Builder, which is used to create custom images in Alibaba Cloud ECS.
<code>provisioners{"type":"value"}</code>	<code>provisioners{"type":"shell"}</code>	The image provisioners defined by Packer to specify operations that need to be performed in the temporary instance. In this example, Shell Provisioner is used to indicate a shell command (for example, <code>yum install redis . x86_64 - y</code>) that is automatically run to install Redis after you connect to the Linux instance.

Table 5-3: Alibaba Cloud parameter description

Parameter	Data type	Example value	Description
access_key	String	LTAInPyXXX XQXXXX	Your AccessKey ID. For more information, see #unique_81 .  Note: To avoid disclosing the AccessKey of your Alibaba Cloud account, we recommend that you #unique_82 .
secret_key	String	CM1ycKrrCe kQ0dhXXXXX XXXXl7yavUT	Your AccessKeySecret.
region	String	cn-beijing	The region to which the target custom image belongs. For more information, see #unique_41 .

Parameter	Data type	Example value	Description
image_name	String	packer_basic	The name of the target custom image. This name must be globally unique.
source_image	String	centos_7_03_64_20G_alibase_20170818.vhd	The ID of the Alibaba Cloud public image that runs the specified operating system.
instance_type	String	ecs.t5-lc1m1.small	The type of the temporary instance used to create the custom image. For more information, see #unique_79 .
internet_charge_type	String	PaybyTraffic	The billing method for the Internet bandwidth of the temporary instance. Recommended value: PaybyTraffic.
io_optimized	Boolean	true	The I/O-optimized attribute of the temporary instance. Recommended value: true.

2. Run the following command to build an image:

```
packer build alicloud . json
```



Note:

It takes a few minutes to build an image. After the image is built, it is displayed in the corresponding Alibaba Cloud region. You can view the image by using the ECS console or calling [DescribeImages](#).

At the same time the image is built, a log is generated. This log includes all image building actions, such as checking the 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 communicat or to connect :
***
==> alicloud - ecs : Waiting for SSH to become available
...
==> alicloud - ecs : Connected to SSH !
==> alicloud - ecs : Provisioni ng with shell script : /
var / folders / k_ / nv2r4drx3b s08l6tcx06 ndb40000gn / T /
packer - shell26004 9331
alicloud - ecs : Loaded plugins : fastestmir ror
alicloud - ecs : Determinin g fastest mirrors
alicloud - ecs : Resolving Dependenci es
alicloud - ecs : --> Running transactio n check
alicloud - ecs : ---> Package redis . x86_64 0 : 3 . 2 .
12 - 2 . el7 will be installed
alicloud - ecs : --> Processing Dependency : libjemallo c
. so . 1 ()( 64bit ) for package : redis - 3 . 2 . 12 - 2 . el7
. x86_64
alicloud - ecs : --> Running transactio n 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 : Dependenci es Resolved
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 dependenci es :
alicloud - ecs : jemalloc x86_64 3 . 6
. 0 - 1 . el7 epel 105 k
alicloud - ecs :
alicloud - ecs : Transactio n 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 : Downloadin g packages :
alicloud - ecs :
-----
alicloud - ecs : Total
2 . 2 MB / s | 648 kB 00 : 00
alicloud - ecs : Running transactio n check
alicloud - ecs : Running transactio n test
alicloud - ecs : Transactio n test succeeded
alicloud - ecs : Running transactio n
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_base ic
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 - xxx

```

What to do next

- Learn more about [Alicloud Image Builder](#) and [Examples](#).
- [#unique_83](#).

5.2 Configure DevOps parameters by using Packer

This topic describes how to configure the basic DevOps parameters of a custom image by using Packer in Alibaba Cloud ECS.

Tag a custom image

- Field name: `tags{"key":"value"}`.
- Applicable scenario: If you have created multiple custom images, you can tag such items as image versions and applications for easy image management and retrieval. Alicloud Image Builder provides the `tags` parameter. By default, the generated images contain Alibaba Cloud ECS tags. For more information, see [#unique_85](#).
- Configuration: When you query images either through the [ECS console](#) or by calling the [DescribeImages](#) API action, the image tags are displayed together

with the target images. These images can also be filtered by tags. Image tags and [Terraform](#) can then help you standardize an enterprise-class DevOps process. We recommend that you read [Alibaba Cloud DevOps tutorials](#). For information about Terraform and Packer, see [Continuous Delivery](#).

- **Example:** The following configuration file contains the generated image and the corresponding snapshot, each of which is attached with the `version = v1 . 0 .`

`0` tag and the `app = web` tag:

```
{
  "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_base_image",
      "source_image": "centos_7_0_3_64_20G_alibase_20170818.vhd",
      "ssh_username": "root",
      "instance_type": "ecs.t5-1c1m1-small",
      "internet_charge_type": "PayByTraffic",
      "io_optimized": "true",
      "tags": {
        "version": "v1.0.0",
        "app": "web"
      }
    }
  ]
}
```

Build an image by only using a system disk snapshot

- **Field name:** `image_ignore_data_disks`.
- **Data type:** Boolean.
- By default, Packer directly creates images based on ECS instances. If the target ECS instances have data disks, the corresponding images will include data disk snapshots. To build an image from an instance that has data disks, you can use either of the following methods:
 - **Method 1:** Set the data disk parameters by using the `image_disk_mappings` parameter. For more information, see [Alicloud Image Builder](#).
 - **Method 2:** Select an instance type that has data disks by default. However, note that data disks are typically local disks, for example, `ecs.d1ne.2xlarge`. Local disks cannot be used to create snapshots. Therefore, you cannot directly create images by using such instances.

- **Configuration:** If you need to select an instance type that has data disks by default, but you do not want to include these data disks during image creation, you can set `"image_ignore_data_disks": "true"` in the configuration file to create a system disk image.

Set a snapshot timeout value

- **Field name:** `wait_snapshot_ready_timeout`.
- **Data type:** Integer.
- **Default value:** 3600s.
- **Applicable scenario:** Images are created based on snapshots. The time required to create a snapshot depends on the disk size.
- **Configuration:** If the disk size is too large and causes a timeout error, run the `wait_snapshot_ready_timeout` command to set a higher timeout value.

Connect to an instance by using a private IP address

- **Field name:** `ssh_private_ip`.
- **Data type:** Boolean.
- **Applicable scenario:** By default, Packer create an EIP and attach this EIP to the target instance. Then, Packer connects to the instance to install software or run commands by using the public IP address corresponding to the EIP. If you use a private IP address to connect to the instance directly, you can disassociate the EIP from the instance.
- **Configuration:** After you set `"ssh_private_ip": "true"`, Packer does not assign an EIP or a public IP address, but connect to the instance by using the private IP address.

Stop an instance

- **Field name:** `disable_stop_instance`.
- **Data type:** Boolean.
- **Applicable scenario:** By default, after Packer runs provisioners, it stops instances and then create images. However, in some scenarios (for example, when you run Sysprep in a Windows instance), the corresponding instance must be in the Running state. For information about the scenarios of Sysprep, see [ECS Windows SID modification operating instructions](#).

- **Configuration:** After you set `"disable_stop_instance": "true"`, Packer does not stop the instance, but determines that the command provided in the configuration (provisioners) automatically stops the instance.

Enable WinRM by using the UserData file

- **Field name:** `user_data_file`.
- **Applicable scenario:** For security purposes, the Windows Remote Management (WinRM) function is disabled in Windows images by default. However, when you connect to a Windows instance and run commands in the instance, you need to use the WinRM function. To make sure the function is enabled when you create an instance, you can enable WinRM by using the UserData file.
- **Configuration:** You can set `"user_data_file": "examples . ps1"` to specify the path of the UserData file.
- **Example:** In the following 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 example, `"communicator": "winrm"` means you are connected to the instance through WinRM, `"winrm_port": 5985` means the communication port is port 5985, `"winrm_username": "Administrator"` means you are connected to the instance as an administrator, and `"winrm_password": "Test1234"` is the password used to connect to the instance.
- The `image_force_delete` parameter setting means to delete the existing same-name image (if any).

Create an image by using a local ISO file

- Field name: `builders{"type":"qemu"}`, `post-processors{"type":"alicloud-import"}`.
- Applicable scenario: If the local ISO file runs in a virtualized environment, you can use Packer to create images.
- Example: If the on-premises environment is QEMU, you can [create and import on-premises images by using Packer](#), as follows:
 1. Use the on-premises virtualized environment or a corresponding Builder, for example, [Qemu Builder](#).
 2. Define [Alicloud Import Post-Processor](#) to import the generated local image file to Alibaba Cloud ECS.

If you want to import a custom image, you must first install an on-premises virtualized environment, and then convert the ISO file to an image file (for example, a QCOW2, VHD, or RAW file) supported by Alibaba Cloud. For more information, see [#unique_88](#).

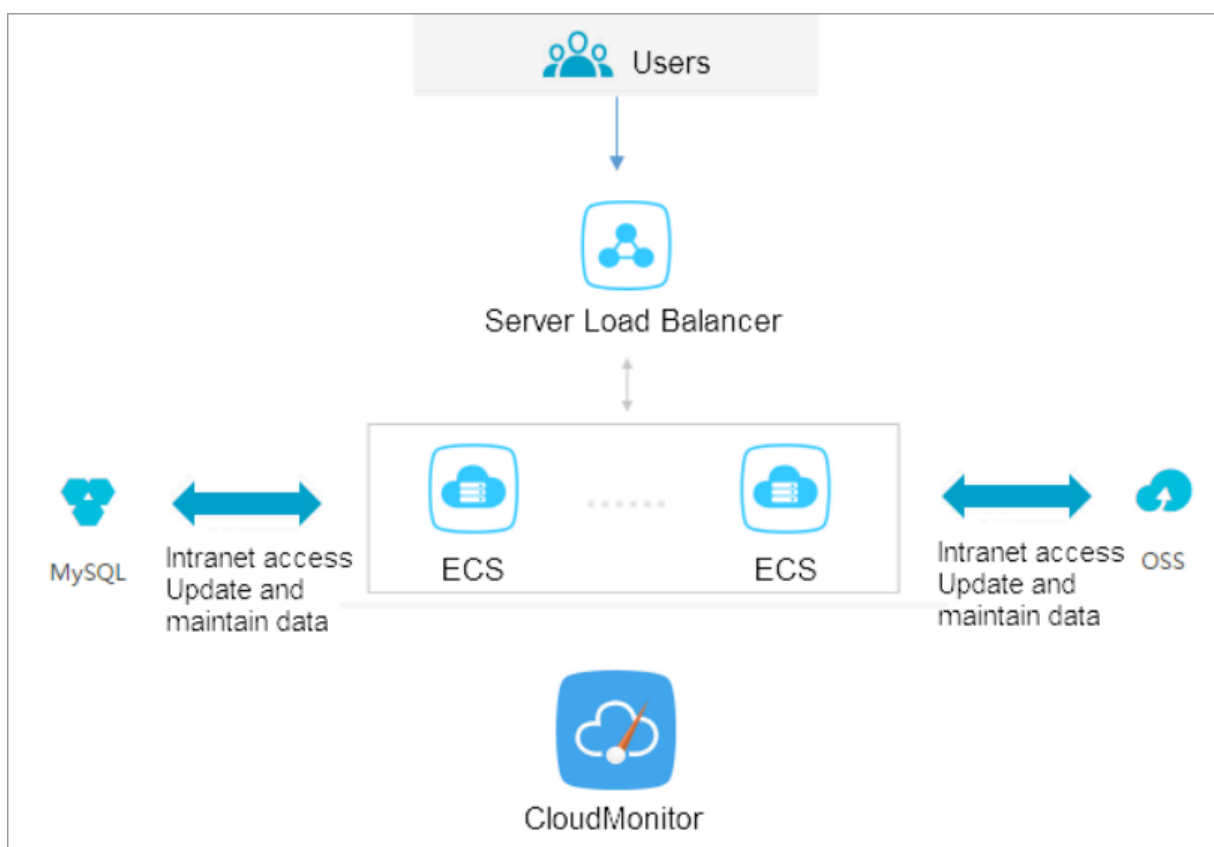
What to do next

Learn more about [Alicloud Image Builder](#) and [Examples](#).

6 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.

2 Set Alarm Rules

Alarm Type : **Threshold Value Alarm** Event Alarm

Alarm Rule : CPU Alarm

Rule Describe : (ECS) CPU Usage 5mins Average >= 70 %

[+Add Alarm Rule](#)

Mute for : 24h ?

Triggered when threshold is exceeded for : 3 ?

Effective Period : 00:00 To: 23:59

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.

7 Monitor

8 Access other Cloud Product APIs by the Instance RAM Role

Previously, applications deployed on an ECS Instance usually needed to use AccessKey ID and AccessKey Secret (AK) to access APIs of other Alibaba Cloud products. AK is the key to accessing Alibaba Cloud APIs and has all of the permissions of the corresponding accounts. To help applications manage the AK, you have to save AK in the configuration files of the application or save it in an ECS instance by using other methods, which makes it more complicated to manage the AK and reduces its confidentiality. What's more, if you need concurrent deployment across regions, the AK is diffused along with the images or instances created by the image, which makes you have to update and re-deploy the instances and images one by one when changing the AK.

Now with the help of the instance [RAM role](#), you can assign a RAM role to an ECS instance. The applications on the instance can then access APIs of other cloud products with the STS credential. The STS credential is automatically generated and updated by the system, and the applications can use the specified [meta data](#) URL to obtain the STS credential without special management. Meanwhile, you can modify the RAM role and the authorization policy to grant different or identical access permissions to an instance to different Alibaba Cloud products.

This article introduces how to create an ECS instance that plays a RAM role and how to enable applications on the ECS instance to access other Alibaba Cloud products with the STS credential.



Note:

To make it easy for you to get started with the example in this article, all of the operations in the document are done in [OpenAPI Explorer](#). OpenAPI Explorer obtains the temporary AK of the current account through the logged user information, and initiates online resource operation to the current account. Please execute operations carefully. Creating an instance will incur charges. Please release the instance soon after completing the operation.

Procedure

To enable python on an instance to access an OSS bucket under the same account by using the instance RAM role, follow these steps:

Step 1: Create a RAM role and attach it to an authorization policy.

Step 2: Create an ECS instance playing the RAM role to create.

Step 3: Within the instance, access the metadata URL to obtain the STS credential.

Step 4: Use Python to access OSS using the STS credential.

Step 1: Create a RAM role and attach it to an authorization policy

1. Use the `CreateRole` API to create a RAM role. The required request parameters are:

- **RoleName:** Specify a name for the role. *EcsRamRoleTestis* is used in this example.
- **AssumeRolePolicyDocument:** Specify a policy as follows, which indicates that the role to be created is a service role and an Alibaba Cloud product (ECS in this example) is assigned to play this role.

```
{
  "Statement": [
    {
      "Action": "sts : AssumeRole ",
      "Effect": "Allow ",
      "Principal": {
        "Service": [
          "ecs . aliyuncs . com "
        ]
      }
    }
  ],
  "Version": " 1 "
}
```

2. Use the `CreatePolicy` API to create an authorization policy. The required request parameters are:

- **PolicyName:** Specify a name for the authorization policy. *EcsRamRolePolicyTest* is used in this example.
- **PolicyDocument:** Specify a policy as follows, which indicates that the role has OSS read-only permission.

```
{
  "Statement": [
    {
      "Action": [
        "oss : Get *",
        "oss : List *"
      ]
    }
  ],
```

```
" Effect ": " Allow ",
" Resource ": "*"
},
" Version ": " 1 "
}
```

3. Use the `AttachPolicyToRole` API to attach the authorization policy to the role. The required request parameters are:

- **PolicyType:** Set it to *Custom*.
- **PolicyName:** Use the policy name specified in step 2. Use *EcsRamRolePolicyTest* in this example.
- **RoleName:** Use the role name specified in step 1. Use *EcsRamRoleTest* in this example.

Step 2: You can use either method to create an ECS instance playing the RAM role:

Attach a RAM role to an existing VPC-Connected ECS instance.

- Create a VPC-Connected ECS instance with the RAM role
- Attach a RAM role to an existing VPC-Connected ECS instance

Create a VPC-Connected ECS instance with the RAM role

Use the `AttachInstanceRamRole` API to attach a RAM role to an existing VPC-Connected ECS instance. The parameters are as follows:

- **RegionId:** The ID of the region where the instance is located.
- **RamRoleName:** The name of a RAM role. In this example, *EcsRamRoleTest* is used. In this example, *EcsRamRoleTest*.
- **InstanceIds:** The IDs of VPC-Connected ECS instances that you want to attach the RAM role to, in the format of ["i-bXXXXXXXX"] for one instance, or ["i-bXXXXXX" , "i-cXXXXXX" , ["i-bXXXXXXXX"]] for multiple instances.

Create a VPC-Connected ECS instance with the RAM role

You must have a VPC network before creating an ECS instance with the RAM role.

1. To create a VPC-Connected ECS instance with the RAM role, follow these steps: Use the `CreateInstance` API to create an ECS instance. The required request parameters are:

- **RegionId**: The region of the instance. In this example, `cn-hangzhou` is used. In this example, `cn-hangzhou` is used.
- **ImageId**: The image of the instance. In this example, `centos_7_03_64_40G_alibase_20170503.vhd` is used. In this example, `centos_7_03_64_40G_alibase_20170503.vhd` is used.
- **InstanceType**: The type of the instance. In this example, `ecs.xn4.small` is used.
- **VSwitchId**: The virtual switch of the VPC network where the instance is located. Because the instance RAM role only supports VPC network, `VSwitchId` is required.
- **RamRoleName**: The name of RAM Role. In this example, `EcsRamRoleTest` is used.

If you want to authorize a sub account to create an ECS instance playing the specified RAM role, besides the permission to create an ECS instance, the sub account must have the `PassRole` permission. Therefore, you must customize an authorization policy as follows and attach it to the sub account. If the action is creating an ECS instance only, set [ECS RAM Action] to `ecs : CreateInstance`. If you want to grant all ECS action permissions to the sub account, set [ECS RAM Action] to `ecs : *`.

```
{
  "Statement": [
    {
      "Action": "ecs : CreateInstance",
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": "ram : PassRole",
      "Resource": "*",
      "Effect": "Allow"
    }
  ],
  "Version": "1"
}
```

2. Set the password and start the instance.

3. Set the ECS instance to access the Internet by using API or in the ECS console.

Step 3: Access the metadata URL within the instance to obtain the STS credential

To obtain the STS credential of the instance, follow these steps:

**Note:**

A new STS credential is generated 30 minutes before the current one expires. Both STS credentials can be used during this period of time.

1. [Connect to the instance.](#)2. Access the following URL to obtain the STS credential. `http://100.100.100.200/latest/meta-data/ram/security-credentials/EcsRamRoleTest`

The last part of the URL is the RAM role name, which must

be replaced with the one you create. The last part of the path is the RAM role name which should be replaced by one you create.

**Note:**

In this example, use the curly command to access the above `curl`. In this example, we run the curl command to access the URL. If you are using a Windows ECS instance, see [Use metadata of an instance](#) in ECS the User Guide to obtain the STS credential.

The return parameters are as follows.

```
[root@local ~]# curl http://100.100.100.200/latest/meta-data/ram/security-credentials/EcsRamRoleTest
{
  "AccessKeyId": "XXXXXXXXX",
  "AccessKeySecret": "XXXXXXXXX",
  "Expiration": "2017-06-09T09:17:19Z",
  "SecurityToken": "CAIXXXXXXXXXXXXXwmBkleCTkyI+",
  "LastUpdated": "2017-10-31T23:20:01Z",
  "Code": "Success"
}
```

Step 4: Use Python SDK to access OSS with the STS credential

In this example, with the STS credential, we use Python to list 10 files in an OSS bucket that is in the same region with the instance.

Prerequisites

You have remotely connected to the ECS instance.

Python has been installed on the ECS instance. If you are using a Linux ECS instance, pip must be installed.

A bucket has been created in the region of the instance, and the bucket name and the Endpoint have been acquired. In this example, the bucket name is `ramroletes t`, and the endpoint is `oss - cn - hangzhou . aliyuncs . com`.

Procedure

To use Python to access the OSS bucket, follow these steps:

1. Run the command `pip install oss2` to install OSS Python SDK.
2. Run the following commands to test, of which:
 - The three parameters in `oss2 . StsAuth` correspond respectively to `AccessKeyId`, `AccessKeySecret` and `SecurityToken` returned by the above URL.
 - The last two parameters in `oss2 . Bucket` are the bucketcodeph name and the endpoint.

```
import oss2
from itertools import islice
auth = oss2 . StsAuth (< AccessKeyI d >, < AccessKeyS
ecret >, < SecurityTo ken >)
bucket = oss2 . Bucket ( auth , < your
Endpoint >, < your Bucket name >)
for b in islice ( oss2 . ObjectIter ator ( bucket ),
10 ):
    print ( b . key )
```

Output results are as follows:

```
[ 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 . J8XXXXXXXXX XX4 ", "
9PjfXXXXXX XXXBf2XAW ", " CAIXXXXXXXXX XXXXwmBkle CTkyI +")
>>> bucket = oss2 . Bucket ( auth , " oss - cn - hangzhou .
aliyuncs . com ", " ramroletes t ")
>>> for b in islice ( oss2 . ObjectIter ator ( bucket ),
10 ):
...     print ( b . key )
...
ramroletes t . txt
test . sh
```

9 GPU instances

9.1 Deploy an NGC on gn5 instances

As a deep learning ecosystem from NVIDIA, NVIDIA GPU CLOUD (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 the gn5 instances. Moreover, the image market also provides NGC container images optimized for NVIDIA Pascal GPU . By deploying NGC container images from the image market, developers can build an NGC container environment conveniently, and access optimized deep learning frameworks instantly, thus reducing the product development and business deployment time considerably. 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 build the environment. By taking the TensorFlow deep learning framework for example, this article describes how to build an NGC environment on gn5 instances.

Before building a TensorFlow environment, you must do the following:

- Sign up with Alibaba Cloud and finish real-name registration.
- Log on to the [NGC website](#) and create your NGC account.
- Log on to the [NGC website](#), get the NGC API Key and save it locally. The NGC API Key will be verified when you log on to the NGC container environment.

Procedure

1. Create a gn5 instance by referring to [create an ECS instance](#). Pay attention to the following configurations:
 - Region: Only China (Qingdao), China (Beijing), China (Hohhot), China (Hangzhou), China (Shanghai), China (Shenzhen), China (Hong Kong),

Singapore, Australia (Sydney), US (Silicon Valley), US (Virginia), and Germany (Frankfurt) are available.

- **Instance:** Select a gn5 instance type.
- **Image:** Select Marketplace Image. In the displayed dialog box, search for NVIDIA GPU Cloud VM Image, and then click Continue.
- **Network Billing Method:** Select Assign Public IP.



Note:

If you do not assign a public IP address here, you can bind an EIP address after the instance is created successfully.

- **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 successfully, [log on to the ECS console](#) and note down the public IP address of the instance.

2. **Connect to the ECS instance:** Based on the logon credentials selected during instance creation, you can [connect to an ECS instance by using a password](#) or [connect to an ECS instance by using an SSH key pair](#).

3. Enter the NGC API Key obtained from the NGC website, and then press the Enter key to log on to the NGC container environment.

```

? MobaXterm 8.4 ?
(SSH client, X-server and networking tools)

> SSH session to [redacted]
? SSH compression : ✓
? SSH-browser      : ✓
? X11-forwarding   : ✓ (remote display is forwarded through SSH)
? DISPLAY          : ✓ (automatically set on remote server)

> For more info, ctrl+click on help or visit our website

Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-116-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

Welcome to the NVIDIA GPU Cloud Virtual Machine. This environment is provided
to enable you to easily run the Deep Learning containers from the NGC Registry.
All of the documentation for how to use NGC and this VM are found at
http://docs.nvidia.com/deeplearning/ngc

Welcome to Alibaba Cloud Elastic Compute Service !

/usr/bin/xauth:  file /root/.Xauthority does not exist

Please enter your NGC APIkey to login to the NGC Registry:

```

4. Run `nvidia - smi` . You can view the information about the current GPU, including the GPU model, the driver version, and more, as shown below.

```

root@ [redacted] # nvidia-smi
Thu Mar 29 20:50:01 2018

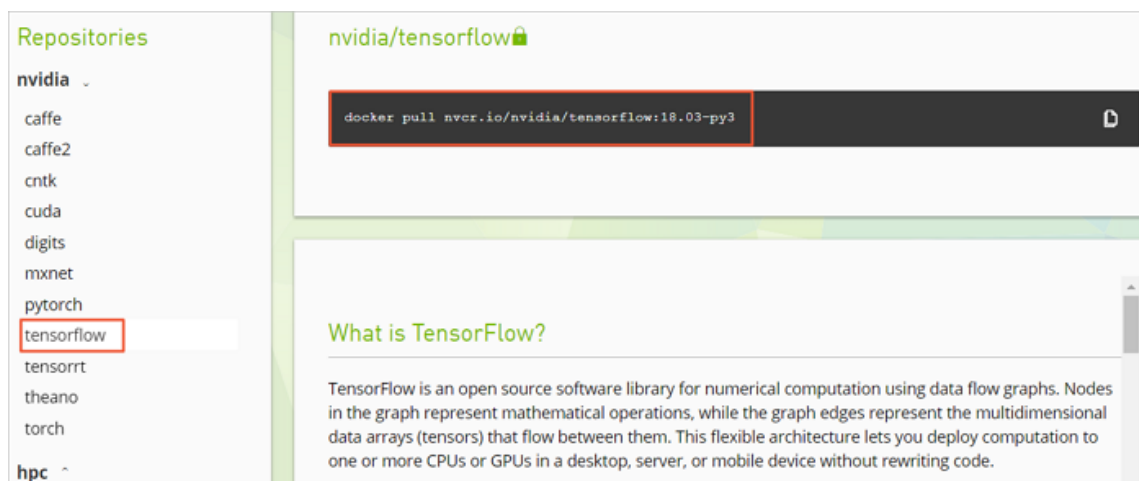
+-----+
| NVIDIA-SMI 384.111                Driver Version: 384.111          |
+-----+-----+
| GPU  Name            Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+
|  0  Tesla P100-PCIE...    Off   | 00000000:00:08.0 Off  |             0        |
| N/A   29C    P0      27W / 250W | 0MiB / 16276MiB |      0%    Default   |
+-----+-----+

Processes:
GPU      PID     Type    Process name                      GPU Memory
Usage
+-----+
| No running processes found |
+-----+

```

5. Follow the steps below to build the TensorFlow environment:

- a. Log on to the [NGC website](#), go to the TensorFlow image page, and then get the `docker pull` command.



- b. Download the TensorFlow image.**

```
docker pull nvcr.io/nvidia/tensorflow:18.03-py3
```

- c. View the downloaded image.**

```
docker image ls
```

- d. Run the container to deploy the TensorFlow development environment.

```
nvidia - docker    run -- rm    - it    nvcr . io / nvidia /  
tensorflow : 18 . 03 - py3
```

```
root@ :~# nvidia-docker run --rm -it nvcr.io/nvidia/tensorflow:18.03-py3

=====
== TensorFlow ==
=====

NVIDIA Release 18.03 (build 349854)

Container image Copyright (c) 2018, NVIDIA CORPORATION. All rights reserved.
Copyright 2017 The TensorFlow Authors. All rights reserved.

Various files include modifications (c) NVIDIA CORPORATION. All rights reserved.
NVIDIA modifications are covered by the license terms that apply to the underlying project or file.
```

6. Test TensorFlow by using one of the following methods:

- 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 device correctly, the result is as shown below.

```
root@-----:~# python
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
2018-03-30 03:37:53.682157: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:892] s
be at least one NUMA node, so returning NUMA node zero
2018-03-30 03:37:53.682544: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Foun
name: Tesla P100-PCI-E-16GB major: 6 minor: 0 memoryClockRate(GHz): 1.3285
pciBusID: 0000:00:08.0
totalMemory: 15.89GiB freeMemory: 15.60GiB
2018-03-30 03:37:53.682583: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1120] Crea
16GB, pci bus id: 0000:00:08.0, compute capability: 6.0)
>>> sess.run(hello)
b'Hello, TensorFlow!'
>>>
```

- 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 below.

```
conv1 [128, 56, 56, 64]
pool1 [128, 27, 27, 64]
conv2 [128, 27, 27, 192]
pool2 [128, 13, 13, 192]
conv3 [128, 13, 13, 384]
conv4 [128, 13, 13, 256]
conv5 [128, 13, 13, 256]
pool5 [128, 6, 6, 256]
2018-03-30 03:40:13.357785: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:892] successful NUMA node read from SysFS
be at least one NUMA node, so returning NUMA node zero
2018-03-30 03:40:13.358207: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Found device 0 with properties:
name: Tesla P100-PCI-E-16GB major: 6 minor: 0 memoryClockRate(GHz): 1.3285
pciBusID: 0000:00:08.0
totalMemory: 15.89GiB freeMemory: 15.60GiB
2018-03-30 03:40:13.358245: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1120] Creating TensorFlow device (/device:GPU:
16GB, pci bus id: 0000:00:08.0, compute capability: 6.0)
2018-03-30 03:40:15.916471: step 0, duration = 0.038
2018-03-30 03:40:16.299169: step 10, duration = 0.038
2018-03-30 03:40:16.682881: step 20, duration = 0.038
2018-03-30 03:40:17.065379: step 30, duration = 0.038
2018-03-30 03:40:17.448118: step 40, duration = 0.038
2018-03-30 03:40:17.830372: step 50, duration = 0.038
2018-03-30 03:40:18.213018: step 60, duration = 0.038
2018-03-30 03:40:18.595734: step 70, duration = 0.038
2018-03-30 03:40:18.978311: step 80, duration = 0.038
2018-03-30 03:40:19.361063: step 90, duration = 0.038
2018-03-30 03:40:19.705396: Forward across 100 steps, 0.038 +/- 0.000 sec / batch
2018-03-30 03:40:21.164735: step 0, duration = 0.090
2018-03-30 03:40:22.062778: step 10, duration = 0.090
2018-03-30 03:40:22.962202: step 20, duration = 0.090
2018-03-30 03:40:23.860856: step 30, duration = 0.090
2018-03-30 03:40:24.758891: step 40, duration = 0.090
2018-03-30 03:40:25.657170: step 50, duration = 0.090
2018-03-30 03:40:26.555194: step 60, duration = 0.090
2018-03-30 03:40:27.452843: step 70, duration = 0.090
2018-03-30 03:40:28.351092: step 80, duration = 0.090
2018-03-30 03:40:29.249606: step 90, duration = 0.090
2018-03-30 03:40:30.058809: Forward-backward across 100 steps, 0.090 +/- 0.000 sec / batch
```

7. Save the changes made to the TensorFlow image. Otherwise, the configuration will be lost the next time you log on.

9.2 Accelerate machine learning tasks on a GPU instance by using RAPIDS

This topic describes how to use RAPIDS libraries (based on the NGC environment) that are installed on a GPU instance to accelerate tasks for data science and machine learning and improve the efficiency of computing resources.

Background information

The following concepts are used in the example provided in this topic:

- Real-time Acceleration Platform for Integrated Data Science (RAPIDS) is a software suite of GPU acceleration libraries developed by NVIDIA for data science and machine learning. For more information, visit [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 that allows them to quickly build corresponding environments. The [NGC website](#) provides RAPIDS Docker images, which come with pre-installed environments.
- JupyterLab is an interactive development environment that helps you browse, edit, and run code files on your servers.
- Dask is a lightweight big data frame that can improve the efficiency of parallel computing.
- In the example provided in this topic, modified code that is based on the NVIDIA RAPIDS Demo and corresponding dataset is provided to demonstrate how to use RAPIDS to accelerate an end-to-end task from ETL to ML Training on a GPU instance. The cuDF library of RAPIDS is used in the Extract-Transform-Load (ETL) phase whereas the XGBoost model is used in the ML Training phase. The example code is based on the Dask frame and runs on a single machine.



Note:

To obtain the official RAPIDS Demo code of NVIDIA, see [Mortgage Demo](#).

Preparations

- Register an Alibaba Cloud account and complete the real-name verification. For more information, see [Account management FAQs](#) and [Real-name registration FAQs](#).
- Go to the [NGC registration page](#) and register an account.

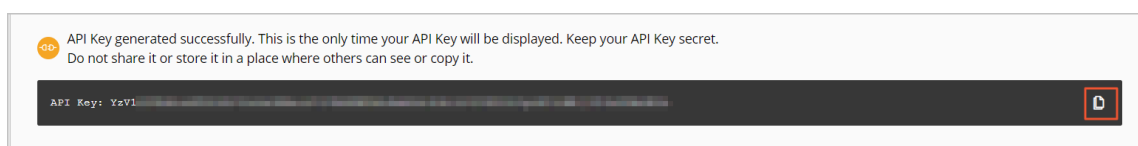
- Obtain an NGC API Key by following these steps:

1. Log on to the [NGC website](#).
2. Go to the CONFIGURATION page, and then click Get API Key.
3. Click Generate API Key.
4. In the displayed dialog box, click Confirm.

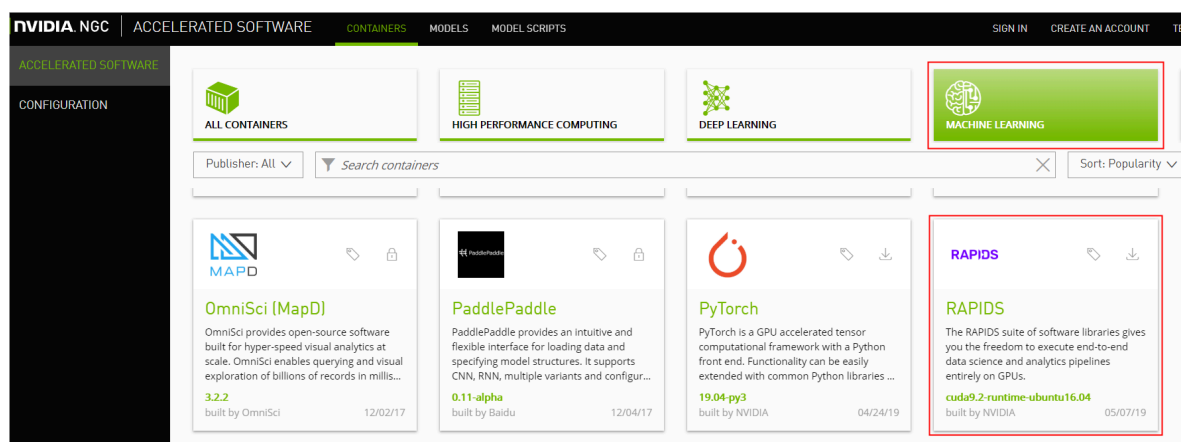
**Note:**

A new NGC API Key overwrites any previous API key. Before you generate a new API Key, you must make sure that the previous API key is no longer being used.

5. Copy the API Key to your local disk.

**Procedure 1: Obtain the RAPIDS image download command**

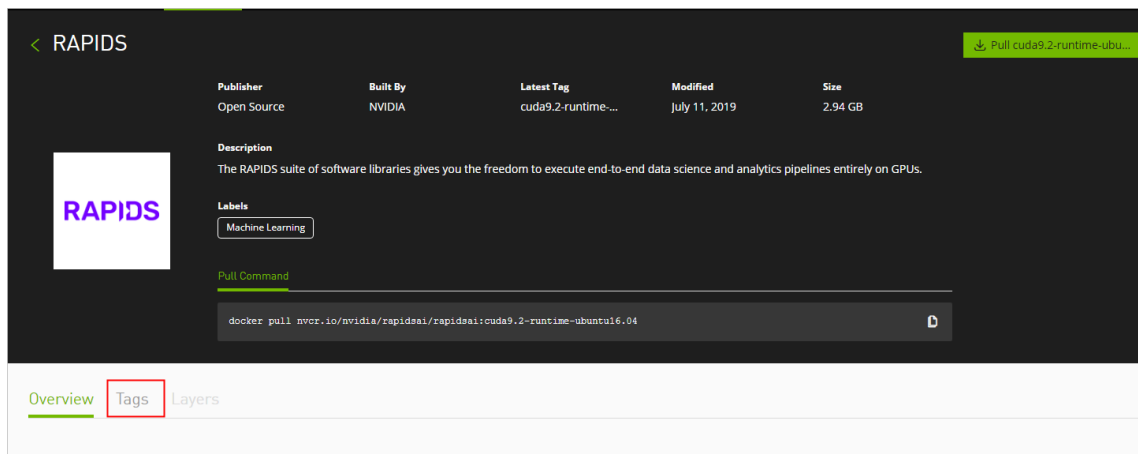
1. Log on to the [NGC website](#).
2. Go to the MACHINE LEARNING page, and then click the RAPIDS image.



3. Obtain the `docker pull` command.

The example code in this topic is based on the RAPIDS v0.6 image. Note that if you use another image, the corresponding command may differ.

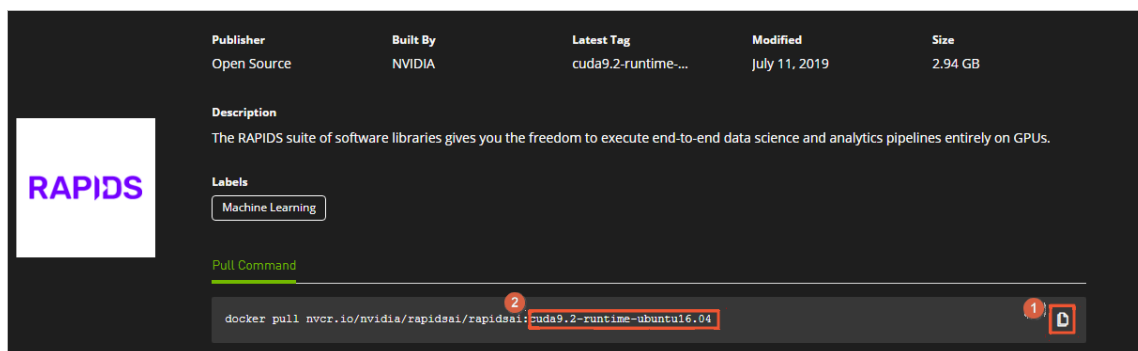
a. Click the Tags tab.



b. Locate and copy the Tag information. In this example, select `0.6 - cuda10.0 - runtime - ubuntu16.04 - gcc5 - py3.6`. Then, open a text editor and paste the Tag information.

0.8-cuda9.2-devel-ubuntu18.04-gcc7-py3.7	July 11, 2019	3.57 GB	↓
0.8-cuda9.2-runtime-ubuntu16.04-gcc5-py3.7	July 11, 2019	2.94 GB	↓
0.8-cuda10.0-runtime-ubuntu16.04-gcc5-py3.6	July 11, 2019	2.96 GB	↓
0.8-cuda9.2-base-centos7-gcc7-py3.6	July 11, 2019	3.04 GB	↓
0.8-cuda9.2-devel-ubuntu16.04-gcc5-py3.7	July 11, 2019	3.74 GB	↓
0.8-cuda9.2-runtime-centos7-gcc7-py3.7	July 11, 2019	3.41 GB	↓

c. Find the Pull Command area and copy the displayed command. Then, paste the command to the text editor. After that, replace the image version with the Tag information from the preceding step, and save the TXT file. In this example, replace `cuda9.2 - runtime - ubuntu16.04` with `0.6 - cuda10.0 - runtime - ubuntu16.04 - gcc5 - py3.6`.

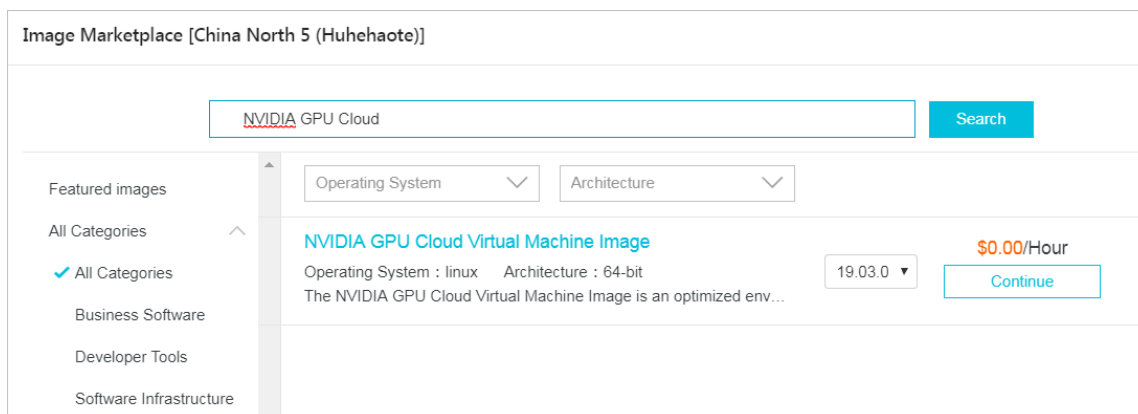


Procedure 2: Deploy the RAPIDS environment

1. Create a GPU instance.

For more information, see [#unique_77](#).

- **Instance Type:** RAPIDS can only be deployed on GPU instances that use NVIDIA Pascal or a later architecture. Currently, you can select the following instance types: gn6i, gn6v, gn5, and gn5i. For more information, see [#unique_79](#). We recommend that you select an instance type that has a larger memory, such as gn6i, gn6v, and gn5. In this example, select the GPU instance that has a 16 GB memory.
- **Image:** In this example, use the `NVIDIA GPU Cloud VM Image`.

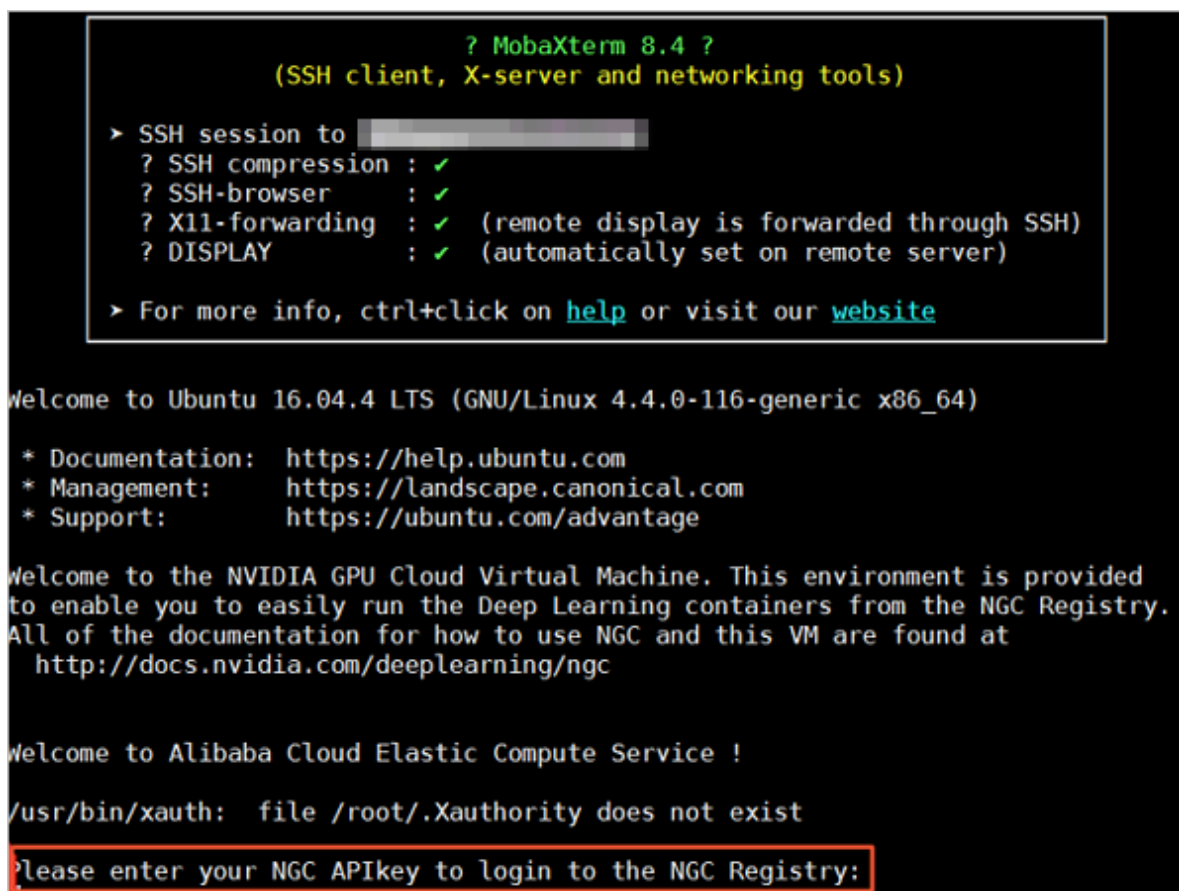


- **Network Billing Method:** Select Assign public IP, or [#unique_105](#) after you create a GPU instance.
- **Security Group:** Select a security group that enables the following ports:
 - TCP port 22, which is used to enable logon through SSH
 - TCP port 8888, which is used to access JupyterLab
 - TCP port 8786 and TCP port 8787, which are used to access Dask

2. Connect to the GPU instance.

For more information, see [#unique_66/unique_66_Connect_42_section_fjm_rgx_wdb](#).

3. Enter the NGC API Key and press Enter to log on to the NGC container.



```

? MobaXterm 8.4 ?
(SSh client, X-server and networking tools)

> SSH session to [redacted]
? SSH compression : ✓
? SSH-browser      : ✓
? X11-forwarding   : ✓ (remote display is forwarded through SSH)
? DISPLAY          : ✓ (automatically set on remote server)

> For more info, ctrl+click on help or visit our website

Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-116-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

Welcome to the NVIDIA GPU Cloud Virtual Machine. This environment is provided
to enable you to easily run the Deep Learning containers from the NGC Registry.
All of the documentation for how to use NGC and this VM are found at
http://docs.nvidia.com/deeplearning/ngc

Welcome to Alibaba Cloud Elastic Compute Service !

/usr/bin/xauth:  file /root/.Xauthority does not exist

Please enter your NGC APIkey to login to the NGC Registry:

```

4. Optional. Run the `nvidia - smi` command to view GPU information, such as GPU model and GPU driver version.

We recommend that you check the GPU information to identify any 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 [Procedure 1: Obtain the RAPIDS image download command](#) to download the RAPIDS image.

```
docker pull nvcr . io / nvidia / rapidsai / rapidsai : 0 . 6 -
cuda10 . 0 - runtime - ubuntu16 . 04 - gcc5 - py3 . 6
```

6. Optional. Check the information of the downloaded image to ensure 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 \
nvr . io / nvidia / rapidsai / rapidsai : 0 . 6 - cuda10 .
0 - runtime - ubuntu16 . 04 - gcc5 - py3 . 6
```

Procedure 3: Run RAPIDS Demo

1. On the GPU instance, download the dataset and the Demo file.

```
# Obtain the apt source address and download the
script ( used to download training data , notebook , and
utils ).
$ source_add ress=$( curl http :// 100 . 100 . 100 . 200 /
latest / meta - data / source - address | head - n 1 )
$ source_add ress="${ source_add ress }/ opsx / ecs / linux /
binary / machine_learning /"
$ wget $ source_add ress / rapids_notebooks_v0 . 6 / utils /
download_v 0 . 6 . sh
# Run the downloaded script .
$ sh ./ download_v 0 . 6 . sh
# Go to the download directory to view the
downloaded file .
$ apt update
$ apt install tree
$ tree / rapids / rapids_notebooks_v0 . 6 /
```

We recommend that you check the downloaded file contains five folders and 16 files.

```
(rapids) root@: /rapids/notebooks# tree /rapids/rapids_notebooks_v0.6/
/rapids/rapids_notebooks_v0.6/
|-- utils
|   |-- download_v0.6.sh
|   |-- start-docker.sh
|   |-- start-jupyter.sh
|   |-- stop-jupyter.sh
|-- xgboost
|   |-- mortgage_2000_lgb
|   |   |-- acq
|   |   |   |-- Acquisition_2000Q1.txt
|   |   |   |-- Acquisition_2000Q2.txt
|   |   |   |-- Acquisition_2000Q3.txt
|   |   |   |-- Acquisition_2000Q4.txt
|   |   |-- names.csv
|   |   |-- perf
|   |   |   |-- Performance_2000Q1.txt_0
|   |   |   |-- Performance_2000Q2.txt_0
|   |   |   |-- Performance_2000Q3.txt_0
|   |   |   |-- Performance_2000Q4.txt_0
|   |   |   |-- Performance_2000Q4.txt_1
|   |-- mortgage_2000_lgb.tgz
|   |-- xgboost_E2E.ipynb
5 directories, 16 files
```

2. Start JupyterLab on the GPU instance by running the following commands:

```
# Go to the working directory .
$ cd / rapids / rapids_notebooks_v0 . 6 / xgboost
# Run the following command to start JupyterLab and
set the logon password :
$ jupyter - lab -- allow - root -- ip = 0 . 0 . 0 . 0 -- no -
browser -- NotebookApp . token = ' logon password '
# Exit .
```

```
$ 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 if you run the script. To exit, press Ctrl+C twice.


3. Open your browser and enter `http :// IP address of your GPU instance : 8888` to access JupyterLab. If a password for JupyterLab is set, you need to enter the password as prompted.



Note:

We recommend that you use Google Chrome.

If you set the logon password when you start JupyterLab, you will be prompted to enter your password.

 jupyter

Password or token:

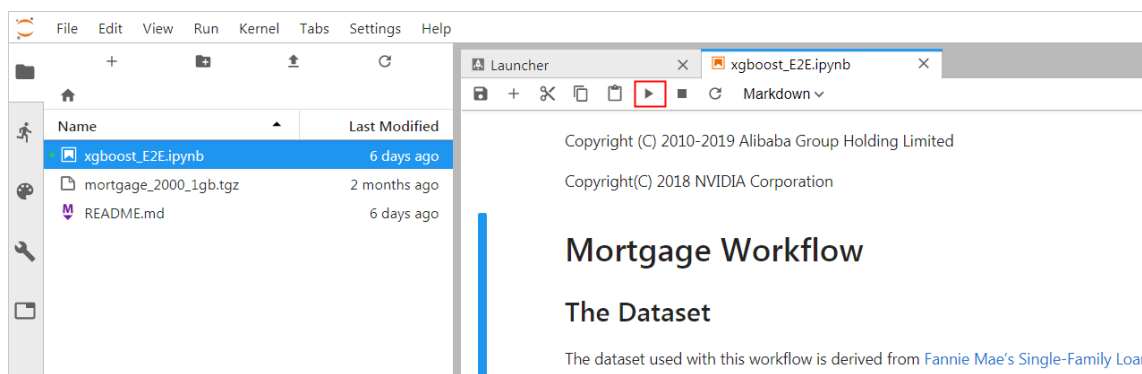
Token authentication is enabled

If no password has been configured, you need to open the notebook server with its login token in the URL, or paste it above. This requirement will be lifted if you [enable a password](#).

4. Run the Notebook code.

Log on to JupyterLab and view the Notebook code. A mortgage repayment task is used in this example. For more information, see [Code running process](#). Details of the example code include:

- A folder named `mortgage_2000_1gb` that contains decompressed training data. Specifically, this folder contains the `acq` folder, `perf` folder and `names.csv` file.
- A file named `xgboost_E2E.ipynb`, which is an XGBoost Demo file. You can double-click this file to view its details, or click the Execute button to execute one cell at a time.



- A file named `mortgage_2000_1gb.tgz`, which contains the mortgage repayment training data of the year 2000 (files in the `perf` folder are split into 1 GB sizes, namely, each file is no larger than 1 GB. This method helps utilize the GPU memory more efficiently).

Code running process

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 three phases:

- ETL (Extract-Transform-Load), which is completed on the GPU instance to extract and transform the data and then load the data to the data warehouse.
- Data Conversion, which is completed on the GPU instance to convert the data processed in the ETL phase into DMatrix-format data so that XGBoost can be used to train the data model.
- ML-Training, which is completed on the GPU instance by default so that the XGBoost training gradient boosting decision tree (GBDT) is used.

The Notebook code is run as follows:

1. Prepare the dataset.

In this example, the Shell script downloads the mortgage repayment training data (`mortgage_2000_1gb.tgz`) and decompress the data to the `mortgage_2000_1gb` file.

If you want to obtain more data for XGBoost model training, you can set the `download_url` parameter to specify the required URL. For more information, see [Mortgage Data](#).

The following figure shows an example.

```
[ ]: # Open: https://docs.rapids.ai/datasets/mortgage-data in your browser, On the displayed page, two datasets are provided: "Dataset" and "1GB Splits",
# 1GB splits are the same data with the individual performance data files split into 1GB pieces.
# This is useful for GPUs with less memory. We recommend that you download the "1GB Splits" dataset for this example.
# You can download and decompress datasets by setting the "download_url" parameter to the actual download URL.
# eg: download_url = 'http://rapidsai-data.s3-website.us-east-2.amazonaws.com/notebook-mortgage-data/mortgage_2000-xxxx_1gb.tgz'
download_url = '' # if download_url = '', use downloaded dataset(mortgage_2000-2001_1gb).

if download_url != '':
    # get download filename from url
    download_filename = download_url.split('/')[-1]
    # the directory to which the file is decompressed.
    mortgage_dir = download_filename.split('.')[0]
    # download dataset with url and filename, if the current directory exist filename, the filename will not be downloaded again.
    download_file_from_url(download_url, download_filename)
    # decompress xxx.tgz to mortgage_dir. if the folder mortgage_dir already exist, the filename will not be decompressed again.
    decompress_file(download_filename, mortgage_dir)
else:
    # use default downloaded data
    mortgage_dir = 'mortgage_2000_1gb'
    # decompress xxx.tgz to mortgage_dir. if the folder mortgage_dir already exist, the filename will not be decompressed again.
    decompress_file('mortgage_2000_1gb.tgz', mortgage_dir)
```

2. Set one or more parameters as needed.

Parameter	Description
<code>start_year</code>	Specify the start year from which training data is selected. In the ETL phase, data generated between the <code>start_year</code> and the <code>end_year</code> is processed.
<code>end_year</code>	Specify the end year from which training data is selected. In the ETL phase, data generated between the <code>start_year</code> and <code>end_year</code> is processed.
<code>train_with_gpu</code>	Set whether to use GPU for XGBoost model training. Default value: <code>True</code> .
<code>gpu_count</code>	Specify the number of workers to be started. Default value: <code>1</code> . You can set the parameter to a value that is less than the number of GPUs in the GPU instance.

Parameter	Description
part_count	Specify the number of performance files used for data model training. Default value: $2 \times \text{gpu_count}$. If the value is too large, an insufficient memory error occurs in Data Conversion phase and an error message is displayed on the backend of Notebook.

The following figure shows an example.

```
[5]: acq_data_path = "{}acq".format(mortgage_dir)
perf_data_path = "{}perf".format(mortgage_dir)
col_names_path = "{}names.csv".format(mortgage_dir)

start_year = 2000
end_year = 2000 # the end_year is inclusive

# whether use GPUs for XGBoost training
train_with_gpu = True

# The number of GPUs to be used, value range: 1 to get_gpu_nums(). Default value: 1.
# This parameter would use for starting dask-worker, doing ETL, doing Conversion and training model(if train_with_gpu=True).
gpu_count = 1

# The number of performance files in the perf folder
part_number = len(os.listdir(perf_data_path))

# if you download 1GB Splits train data(the filename end with 'lgb.tgz'), each performance file is no large than 1GB,
# in this example, a 16G GPU can process 2 or 3 performance files. By default, one GPU is set to process 2 files.
part_count = 2 * gpu_count if part_number >= 2 * gpu_count else part_number

print('>>> Using "{}" GPU(GPUs)'.format(gpu_count))
print('>>> ETL - process performance files from "{}" to "{}"'.format(start_year, end_year))
print('>>> Data Conversion - select "{}" performance data processed in the ETL phase and convert to DMatrix-format for XGBoost training.'.format(part_count))
print('>>> ML - Whether to use the GPU for XGBoost training: "{}"'.format(train_with_gpu))

>>> Using "1" GPU(GPUs).
>>> ETL - process performance files from "2000" to "2000".
>>> Data Conversion - select "2" performance data processed in the ETL phase and convert to DMatrix-format for XGBoost training.
>>> ML - Whether to use the GPU for XGBoost training: "True".
```

3. Start Dask.

The Notebook code starts Dask Scheduler, and also starts workers according to the setting of the `gpu_count` parameter for ETL and data model training.

The following figure shows an example.

```
# run dask-worker
cmd = "hostname --all-ip-addresses"
process = subprocess.Popen(cmd.split(), stdout=subprocess.PIPE)
output, error = process.communicate()
IPADDR = str(output.decode()).split()[0]

cluster = LocalCUDACluster(n_workers=gpu_count, ip=IPADDR)
client = Client(cluster)
client
```

Client	Cluster
• Scheduler: tcp://172.17.0.2:43894	• Workers: 1
• Dashboard: http://172.17.0.2:8787/status	• Cores: 1
	• Memory: 507.25 GB

4. Start the ETL phase.

In this phase, tables are associated, grouped, integrated, and split. The data format is DataFrame of the cuDF library (similar to DataFrame of pandas).

The following figure shows an example.

```
ETL

Perform all of ETL with a single call to

process_quarter_gpu(year=year, quarter=quarter, perf_file=file)

]: %%time

# NOTE: The ETL calculates additional features which are then dropped before creating the XGBoost DMatrix.
# This can be optimized to avoid calculating the dropped features.

gpu_dfs = []
gpu_time = 0
quarter = 1
year = start_year
count = 0
while year <= end_year:
    for file in glob(os.path.join(perf_data_path + "/Performance_" + str(year) + "Q" + str(quarter) + "*")):
        gpu_dfs.append(process_quarter_gpu(year=year, quarter=quarter, perf_file=file))
        count += 1
    quarter += 1
    if quarter == 5:
        year += 1
        quarter = 1
wait(gpu_dfs)

CPU times: user 560 ms, sys: 28 ms, total: 588 ms
Wall time: 20.9 s
```

5. Start the Data Conversion phase.

In this phase, DataFrame-format data is converted into DMatrix-format data for XGBoost model training. Each worker processes one DMatrix object.

The following figure shows an example.

```
Load the data from host memory, and convert to CSR

%%time

gpu_dfs = [delayed(DataFrame.from_arrow)(gpu_df) for gpu_df in gpu_dfs[:part_count]]
gpu_dfs = [gpu_df for gpu_df in gpu_dfs]
wait(gpu_dfs)

tmp_map = [(gpu_df, list(client.who_has(gpu_df).values())[0]) for gpu_df in gpu_dfs]
new_map = {}
for key, value in tmp_map:
    if value not in new_map:
        new_map[value] = [key]
    else:
        new_map[value].append(key)

del(tmp_map)
gpu_dfs = []
for list_delayed in new_map.values():
    gpu_dfs.append(delayed(cudf.concat)(list_delayed))

del(new_map)
gpu_dfs = [(gpu_df[['delinquency_12']], gpu_df[delayed(list)(gpu_df.columns.difference(['delinquency_12']))) for gpu_df in gpu_dfs]
gpu_dfs = [(gpu_df[0].persist(), gpu_df[1].persist()) for gpu_df in gpu_dfs]

gpu_dfs = [dask.delayed(xgb.DMatrix)(gpu_df[1], gpu_df[0]) for gpu_df in gpu_dfs]
gpu_dfs = [gpu_df.persist() for gpu_df in gpu_dfs]
gc.collect()
wait(gpu_dfs)

CPU times: user 200 ms, sys: 4 ms, total: 204 ms
Wall time: 4.3 s
```

6. Start the ML Training phase.

In this phase, data model training is started by dask-xgboost, which supports collaborative communication among Dask workers. On the bottom layer, dask-xgboost is also called to execute data model training.

The following figure shows an example.

```
Train the Gradient Boosted Decision Tree with a single call to

dask_xgboost.train(client, params, data, labels, num_boost_round=dxgb_gpu_params['nround'])

[ ]: if train_with_gpu:
    print('>>> Training with {} GPU.'.format(gpu_count))
else:
    print('>>> Training with {} CPU.'.format(gpu_count))
    dxgb_gpu_params['tree_method'] = 'hist'

print('>>> Worker number: {}'.format(gpu_count))
print('>>> part_count: {}'.format(part_count))

rows = sum([dmatrix.num_row().compute() for dmatrix in gpu_dfs])
cols = gpu_dfs[0].num_col().compute()
print('>>> Train data rows: {},\tcols: {}'.format(rows, cols))

[ ]: %%time
labels = None
bst = dxgb_gpu.train(client, dxgb_gpu_params, gpu_dfs, labels, num_boost_round=dxgb_gpu_params['nround'])
```

Related functions

Operation	Function name
Download a file.	<code>def download_file_from_url(url, filename):</code>
Decompress a file.	<code>def decompress_file(filename, path):</code>
Obtain the number of GPUs in the current machine.	<code>def get_gpu_nums():</code>
Manage the GPU memory.	<ul style="list-style-type: none"> • <code>def initialize_rmm_pool():</code> • <code>def initialize_rmm_no_pool():</code> • <code>def run_dask_task(func, **kwargs):</code>
Submit a Dask task.	<ul style="list-style-type: none"> • <code>def process_quarter_gpu(year=2000, quarter=1, perf_file=""):</code> • <code>def run_gpu_workflow(quarter=1, year=2000, perf_file="", **kwargs):</code>
Use cuDF to load data from a CSV file.	<ul style="list-style-type: none"> • <code>def gpu_load_performance_csv(performance_path, **kwargs):</code> • <code>def gpu_load_acquisition_csv(acquisition_path, **kwargs):</code> • <code>def gpu_load_names(**kwargs):</code>
Process and extract characteristics of data for training machine learning models.	<ul style="list-style-type: none"> • <code>def null_workaround(df, **kwargs):</code> • <code>def create_ever_features(gdf, **kwargs):</code> • <code>def join_ever_delinq_features(everdf_tmp, delinq_merge, **kwargs):</code> • <code>def create_joined_df(gdf, everdf, **kwargs):</code> • <code>def create_12_mon_features(joined_df, **kwargs):</code> • <code>def combine_joined_12_mon(joined_df, testdf, **kwargs):</code> • <code>def final_performance_delinquency(gdf, joined_df, **kwargs):</code> • <code>def join_perf_acq_gdfs(perf, acq, **kwargs):</code> • <code>def last_mile_cleaning(df, **kwargs):</code>

10 FaaS instances best practices

10.1 Use RTL compiler on an f1 instance

This topic describes how to use Register Transfer Level (RTL) compiler on an f1 instance.



Note:

- All the operations described in this topic must be performed by one account in the same region.
- We strongly recommend that you use an f1 instance as a RAM user. To avoid unwanted operations, you must authorize the RAM user to perform required actions only. You must create a role for the RAM user and grant temporary permissions to the role to access the OSS buckets. If you want to encrypt the IP address, grant the RAM user to use Key Management Service (KMS). If you want the RAM user to check permissions, authorize the RAM user to view the resources of an account.

Prerequisites

- Create an f1 instance and add a security group rule to allow Internet access to SSH Port 22 of the instance.



Note:

Only the image we share with you can be used on an f1 instance. For more information, see [Create an f1 instance](#).

- Log on to the [ECS console](#) to obtain the instance ID.
- Activate OSS and [create an OSS bucket](#) to upload your files. The OSS bucket and the f1 instance must be owned by one account and operated in the same region.
- For encryption, activate [Key Management Service \(KMS\)](#).
- To operate FPGA as a RAM user, do the following in advance:
 - [Create a RAM](#) and [grant permissions](#).
 - [Create a RAM](#) and [grant permissions](#).
 - Use the AccessKey to complete the authentication.

Procedure

To use RTL compiler on an f1 instance, follow these steps.

Step 1. Connect to the f1 instance

[Connect to your f1 instance.](#)

Step 2. Configure the basic environment

Run the script to configure the basic environment.

```
source /opt/dcp1_1/script/f1_env_set.sh
```

Step 3. Compile the project

Run the following commands to compile the project.

```
cd /opt/dcp1_1/hw/samples/dma_afu
afu_synth_setup --source hw/rtl/filelist.txt
build_synt h
cd build_synt h /
run . sh
```



Note:

It takes a long time to compile the project.

Step 4. Create an image

To create an image, follow these steps:

1. Run the following commands to initialize `faascmd`.

```
# If needed, add the environment variable and grant
  permission to run the commands.
export PATH=$PATH:/opt/dcp1_1/script/
chmod +x /opt/dcp1_1/script/faascmd
# Replace hereIsMySecretId with your AccessKey ID.
Replace hereIsMySecretKey with your AccessKey Secret.
faascmd config --id=hereIsMySecretId --key=hereIsMySecretKey
faascmd config --id=hereIsYourSecretId --key=hereIsYourSecretKey
# Replace hereIsYourBucket with the OSS bucket name
in the China (Hangzhou) region.
```

```
faascmd auth -- bucket = hereIsYour Bucket
```

2. Make sure you are at the `/opt/dcp1_1/hw/samples/dma_afu` directory, and run the command to upload the gbs file.

```
faascmd upload_object -- object = dma_afu . gbs -- file =
dma_afu . gbs
```

- ### 3. Run the command to create an image.

```
# Replace hereIsYour ImageName with your image name .
faascmd create_image -- object = dma_afu . gbs -- fpgatype
= intel -- name = hereIsYour ImageName -- tags = hereIsYour
ImageTag -- encrypted = false -- shell = V1 . 1
```

Step 5. Download the image

To download the image, follow these steps:

1. Run the `faascmd list_image s` command to check whether the image is created.

If " State ":" success " exists in the returned result, it means the image is created. Record the FpgaImageUUID. Record the FpgaImageUUID.

```
[root@redhat ~]# faascmd list_images
{"FpgaImages":{"fpgaImage":[{"Name":"Image_1_dma_afu","Tags":"ImageTag_1_dma_afu","ShellUUID":"V8-8-8-8","Description":"None","FpgaImageUUID":"intel98db1d1-023-8","State":"success","CreateTime":"Fri Jan 26 2018 10:15:59 GMT+0800 (CST)","Encrypted":"false","UpdateTime":"Fri Jan 26 2018 10:17:08 GMT+0800 (CST)"}]}
```

- 2. Run the command to obtain FPGA ID.**

```
# Replace hereIsYour InstanceId with your f1 instance ID .
faascmd list instances -- instanceId = hereIsYour InstanceId
```

Record FpgaUUID in the returned result.

```
root@i2b:~/2 output_files# faascmd list_instances --instanceId=i-bp15
{"instances":[{"instance":{"ShellUUID":"V...", "FpgaType":"intel", "FpgaUID":"0x...", "InstanceID":"i-bp15", "DeviceID": "05:00:00..."}, "FpgaStatus":"valid"}]}
```

- 3. Run the command to download the image to your f1 instance.**

```
# Replace hereIsYour InstanceID with your f1 instance
  ID . Replace hereIsFpga UUID with your FpgaUUID .
Replace hereIsImag eUUID with your FpgaImageU  UUID .
faascmd download_i mage -- instanceId = hereIsYour InstanceID
-- fpgauuid = hereIsFpga UUID -- fpgatype = intel -- imageuuid =
hereIsImag eUUID -- imagetype = afu -- shell = V0 . 11
```

4. Run the command to check whether the image is downloaded.

```
# Replace hereIsYour InstanceID with your f1 instance
ID . Replace hereIsFpga UUID with your FpgaUUID .
```

```
faascmd fpga_status s -- instanceId = hereIsYour InstanceID --
fpgauuid = hereIsFpga UUID
```

If "TaskStatus ":" operating " exists in the returned result, and the displayed FpgaImageUUID is identical with your recorded FpgaImageUUID, the image is downloaded.

```
[root@ ~]# faascmd fpga_status --instanceId=i-bp1-3e6s --fpgauuid=0x40500
{"shellUUID":"V...", "FpgaImageUUID":"intel98db1...", "FpgaUUID":"0x40500", "InstanceId":"i-bp1-3e6s", "CreateTime":"Fri Jan 26 2018 10:40:41 GMT+0800 (CST)", "TaskStatus":"operating", "Encrypted":"false"}
0.291(s) elapsed
```

Step 6. Test

Run the commands one by one for 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 result is returned, the test is completed.

```
[root@iZ...Z sw]# ./fpga_dma_test use_ase=0
Running test in HW mode
Buffer Verification Success!
Buffer Verification Success!
Running DDR sweep test
Allocated test buffer
Fill test buffer
DDR Sweep Host to FPGA
Measured bandwidth = 5726.623061 Megabytes/sec
Clear buffer
DDR Sweep FPGA to Host
Measured bandwidth = 4473.924267 Megabytes/sec
Verifying buffer..
Buffer Verification Success!
```



Note:

If the Huge pages feature is not enabled, run the following command to enable it.

```
sudo bash -c "echo 20 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages"
```

10.2 Use OpenCL on an f1 instance

This topic introduces how to use Open Computing Language (OpenCL) to create an image file, and then download the image to an FPGA chip.



Note:

- All the operations described in this topic must be performed by one account in the same region.
- We strongly recommend that you use an f1 instance as a RAM user. To avoid unwanted operations, you must authorize the RAM user to perform required actions only. You must create a role for the RAM user and grant temporary permissions to the role to access the OSS buckets. If you want to encrypt the IP address, grant the RAM user to use Key Management Service (KMS). If you want the RAM user to check permissions, authorize the RAM user to view the resources of an account. Before you begin, complete the following:

Prerequisites

- Create an f1 instance and add a security group rule to allow Internet access to SSH Port 22 of the instance.



Note:

Only the image we share with you can be used on an f1 instance. For more information, see [Create an f1 instance](#).

- Log on to the [ECS console](#) to obtain the instance ID.
- [Create an OSS bucket](#) to upload your custom bitstream files. The OSS bucket and the f1 instance must be owned by one account and in the same region.
- To encrypt your bitstream, activate Key Management Service (KMS).
- To operate an f1 instance as a RAM user, you must do the following operations:
 - [Create a RAM user](#) and [grant permissions](#).
 - [Create a RAM role](#) and [grant permissions](#).
 - Create an AccessKey.

Procedure

To configure the environment of FPGA Server Example, follow these steps.

Step 1. Connect to your f1 instance

Connect to the Linux instance.

Step 2. Install the basic environment

Run the following script to install the base environment.

```
source /opt/dcp1_1/script/fl_env_set.sh
```

Step 3. Download the OpenCL Example

Follow these steps to download the official openc1 example.

1. Create the `/opt/tmp` directory, and change the current directory to it.

```
mkdir -p /opt/tmp
cd /opt/tmp
```

Now, you are at the `/opt/tmp` directory.

```
[root@iz[REDACTED]Z tmp]# pwd
/opt/tmp
```

2. Run the commands one by one to download and decompress the OpenCL Example file.

```
wget https://www.altera.com/content/dam/altera-  
www/global/en_US/others/support/examples/download/  
exm_openc1_matrix_mu_lt_x64_lin_ux.tgz  
tar -zxvf exm_openc1_matrix_mu_lt_x64_lin_ux.tgz
```

The following figure displays the directory after decompression.

```
[root@iz...Z tmp]# tree -L 1
.
├── common
├── exm_opencl_matrix_mult_x64_linux.tgz
└── matrix_mult

2 directories, 1 file
```

3. Change the current directory to the `matrix_mul_t` directory and run the command for compilation.

```
cd matrix_mul_t
aoc -v -g --report ./device/matrix_mul_t.cl
```

The process of compilation takes several hours. You can open a new console, and run the `top` command to monitor processes and system resource usage on the instance and view the status of the compilation process.

Step 4. Upload the configuration file to the OSS bucket

Follow these steps to upload the configuration file.

1. Run the commands to initialize the `faascmd`.

```
# If needed, add the environment variable and grant
the permission to run the commands
export PATH=$PATH:/opt/dcp1_1/script/
chmod +x /opt/dcp1_1/script/faascmd
# Replace hereIsYour SecretId with your AccessKey ID.
Replace hereIsYour SecretKey with your AccessKey Secret
faascmd config --id=hereIsYour SecretId --key=
hereIsYour SecretKey
# Replace hereIsYour Bucket with the bucket name of
your OSS in the Region China (Hangzhou).
faascmd auth --bucket=hereIsYour Bucket
```

2. Change the current directory to the `matrix_mul_t/output_files` directory, and upload the configuration file.

```
cd matrix_mul_t/output_files # Now you are accessing
/opt/tmp/matrix_mul_t/matrix_mul_t/output_files
faascmd upload_object --object=afu_fit.gbs --file=
afu_fit.gbs
```

3. Use `gbs` to create an FPGA image.

```
# Replace hereIsYour ImageName with your image name.
Replace hereIsYour ImageTag with your image tag.
faascmd create_image --object=dma_afu.gbs --fpctype=
intel --name=hereIsYour ImageName --tags=hereIsYour
ImageTag --encrypted=false --shell=V1.1
```

4. Run the `faascmd list_images` command to check whether the image is created. In the returned result, if "State": "success" is displayed, it means the image is created. Record the `FpgaImageUUID`.

```
[root@f2op ~]# faascmd list_images
{"FpgaImages":[{"FpgaImage":{"Name":"Image_1_dma_afu","Tags":{"ImageTag_1_dma_afu":{"ShellUUID":"V0.11","Description":"None","FpgaImageUUID":"inteld98db1d1-0238","State":"success","CreateTime":"Fri Jan 26 2018 10:15:59 GMT+0800 (CST)","Encrypted":"false","UpdateTime":"Fri Jan 26 2018 10:17:08 GMT+0800 (CST)"}}

```

Step 5. Download the image to your f1 instance

To download the image to your f1 instance, follow these steps:

1. Run the command to obtain FPGA ID.

```
# Replace hereIsYour InstanceId with your f1 instance
ID .
faascmd list_instances -- instanceId = hereIsYour InstanceId
```

Returned results sample: Record FpgaUUID in the returned result.

```
[root@iz-2 output_files]# faascmd list_instances --instanceId=i-bp15n6gzt...
{"Instances":[{"instance":{"ShellUUID":"V0.11","FpgaType":"intel","FpgaUUID":"0xe...","InstanceId":"i-bp15n6gzt...","DeviceBDF":"05:00.0","FpgaStatus":"valid"}}]}
```

2. Run the command to download the image to your f1 instance.

```
# Replace hereIsYour InstanceID with your f1 instance
ID . Replace hereIsFpga UUID with your FPGA UUID .
Replace hereIsImage eUUID with your image UUID .
faascmd download_image -- instanceId = hereIsYour InstanceID
-- fpgauid = hereIsFpga UUID -- fpgatype = intel -- imageuid
= hereIsImage eUUID -- imagetype = afu -- shell = V0 . 11
```

3. Run the command to check whether the image is downloaded.

```
# Replace hereIsYour InstanceID with your f1 instance
ID . Replace hereIsFpga UUID with your FPGA UUID .
faascmd fpga_status -- fpgauid = hereIsFpga UUID --
instanceId = hereIsYour InstanceID
```

If "TaskStatus": "operating" exists in the returned result, it means the image is downloaded.

```
[root@iz-2 output_files]# faascmd fpga_status --instanceId=i-bp15n6gzt... --fpgauid=0xe...
{"shellUUID":"V0.11","FpgaImageUUID":"inteld98db1...","FpgaUUID":"0xe...","InstanceId":"i-bp15n6gzt...","CreateTime":"Fri Jan 26 2018 10:40:41 GMT+0800 (CST)","TaskStatus":"operating","Encrypted":"false"}
0.291(s) elapsed
```

Step 6. Download the FPGA image to an FPGA chip

To download the FPGA image to an FPGA chip, follow these steps:

1. Open the console in Step 1. If it is closed, repeat Step 1.

2. Run the following command to configure the runtime environment for OpenCL.

```
sh /opt/dcp1_1/openc1/openc1_bsp/linux64/libexec/
setup_permissions.sh
```

3. Run the command to go back to the parent directory.

```
cd ../../ # Now, you are at the /opt/tmp/
matrix_multiply directory
```

4. Run the command to compile.

```
make
# Output the environment configuration
export CL_CONTEXT_COMPILER_MODE_ALTERA = 3
cp matrix_multiply.aocx ./bin/matrix_multiply.aocx
cd bin
host matrix_multiply.aocx
```

If the following result is returned, it means the configuration is successful. Note that the last line must be `Verification : PASS`.

```
[root@iZbpXXXXXZ bin]# ./host matrix_multiply.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_0 : SKX DCP FPGA OpenCL BSP (acl0)
Using AOCX : matrix_multiply.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
```

10.3 Use OpenCL on an f3 instance

This topic describes how to use Open Computing Language (OpenCL) to create, upload, and download an image file to an FPGA on an f3 instance.

Prerequisites

You can use OpenCL on an f3 instance if the following requirements are met:

- An f3 instance is created. For more information, see [#unique_117](#).



Note:

- Only the image we share with you can be used on an f3 instance.
 - Select Assign Public IP Address when creating an instance, so that the instance can access the Internet.
 - The rule for allowing access to SSH port 22 has been configured for the security groups where the f3 instance resides.
- You have obtained the ID of your f3 instance on the Instances page of the ECS console.
 - You have created an OSS bucket in the same region as your f3 instance by using the same account. For more information, see [Activate OSS](#) and [Create a bucket](#).
 - You have completed the following operations if you need to operate FPGA as a RAM user:
 - [Create a RAM user](#) and [grant permissions](#).
 - [Create a RAM role](#) and [grant permissions](#).
 - Obtain the AccessKey ID and AccessKey Secret.

Precautions

Before you use OpenCL on an f3 instance, be aware of the following:

- All the operations described in this topic must be performed by one account in the same region.
- We recommend that you use 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 buckets.
- The operations and commands described in this topic are based on the SDAccel development environment 2018.2. If you use SDAccel development environment of other versions, the operations and commands may vary.

Procedure

To use OpenCL to create, upload, and download an image file to an FPGA on an f3 instance, follow these steps:

- [Step 1. Set up the environment](#)
- [Step 2. Compile a binary file](#)
- [Step 3. Check the packaging script](#)
- [Step 4. Create an image](#)
- [Step 5. Download the image](#)

- **Step 6: Run the Host program**

Step 1. Set up the environment

To set up the environment, follow these steps:

1. Connect to the f3 instance.**Note:**

The subsequent compilation process may take a few hours. We recommend that you log on through screen or nohub, so as 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 enter screen:

```
screen - S f3openc1
```

4. Run the following command to set up the environment:

```
source / root / xbinst_oem / F3_env_set up . sh xocl # Run  
the command each time you open a new terminal  
window
```

**Note:**

- Configuring 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 detecting the faascmd version.
- If you want to run an emulation of SDAccel, do not run the preceding command to set up the environment. Instead, 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

To compile the vadd and kernel_global_bandwidth binary files, follow these steps:

• **Example 1: vadd**

1. Copy the `example` directory.

```
cp -rf /opt/Xilinx/SDx/2018.2/examples ./
```

2. Enter the `vadd` directory.

```
cd examples/vadd/
```

3. Run the command `cat sdaccel.mk | grep "XDEVICE ="` to view the value of `XDEVICE`. Make sure its configuration is `XDEVICE = xilinx_aliyun`.

```
yun -f3_dynamic_5_0.
```

4. Follow these steps to modify the `common.mk` file.

a. Run the `vim ../common/common.mk` command to open the file.

b. At the end of the code line 61, add the compilation parameter `--xp param : compiler . accelerator rBinaryContent = dcp` (the parameter may be in the line 60-62, depending on your file). The modified code is:

```
CLCC_OPT += $(CLCC_OPT_L EVEL) ${DEVICE_REPO_OPT}
--platform ${XDEVICE} -o ${XCLBIN} ${KERNEL_DEF} S
${KERNEL_INC} S } --xp param : compiler . accelerator
rBinaryContent = dcp
```



Note:

Given that you must submit a DCP file to the compilation server, you need to add the parameter `--xp param : compiler . accelerator rBinaryContent = dcp`, so that Xilinx® OpenCL™ Compiler (xocc) generates a DCP file (instead of a bit file) after the placement and routing is complete.

5. Run the following command to compile the program:

```
make -f sdaccel.mk xbin_hw
```

If the following information is displayed, the compilation of the binary file has started. This process may take several hours.

```
[root@ ~]# cd vadd/ && make -f sdaccel.mk xbin_hw
make SDA_FLOW=hw xbin -f sdaccel.mk
make[1]: Entering directory '/root/xilinx_example/examples/vadd'
xocc -c -t hw --platform xilinx_aliyun-f3_dynamic_5_0 --xp param:compiler.acceleratorBinaryContent=dcp -s --kernel krnl_vadd krnl_vadd.cl -o bin_vadd_hw.xo
***** xocc v2018.2 (64-bit)
***** SW Build 2258646 on Thu Jun 14 20:02:38 MDT 2018
***** Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.
Attempting to get a license: ap_openc1
Feature available: ap_openc1
INFO: [XOCC 60-585] Compiling for hardware target
Running SDx Rule Check Server on port:39876
INFO: [XOCC 60-895] Target platform: /opt/Xilinx/SDx/2018.2/platforms/xilinx_aliyun-f3_dynamic_5_0/xilinx_aliyun-f3_dynamic_5_0.xpfm
INFO: [XOCC 60-423] Target device: xilinx_aliyun-f3_dynamic_5_0
```


• Example 2: kernel_global_bandwidth

Follow these steps to compile the kernel_global_bandwidth binary file:

1. 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 must be the 2018.2 version.

2. Run the `cd getting_started / kernel_to_gmem / kernel_global_bandwidth /` command to enter the directory.

3. Follow these steps to modify the `Makefile` file.

a. Run the `vim Makefile` command to modify the file.

b. Set `DEVICES = xilinx_aliyun - f3_dynamic_5_0`.

c. In the code line 33, add the compilation parameter `-- xp param :`

`compiler . acceleratorBinaryContent = dcp`. The code after modification is:

```
CLFLAGS +=-- xp " param : compiler . acceleratorBinaryContent = dcp " -- xp " param : compiler . preserveHlsOutput = 1 " -- xp " param : compiler . generateExtraRunData = true " -- max_memory_ports bandwidth - DNDDBANKS=$(ddr_banks)
```

4. Run the following command to compile the program:

```
make TARGET = hw
```

If the following information is displayed, the compilation of the binary file has started. This process may take several hours.

```

[1]# make TARGET=hw
mkdir -p ./xclbin
/opt/Xilinx/SDx/2018.2/bin/xccp -I /opt/Xilinx/SDx/2018.2/runtime/include/1_2/ -I /opt/Xilinx/SDx/2018.2/Vivado_HLS/include/ -O0 -g -Wall -fmessage-length=0 -std=c++14 -DNDDBANKS=4 -I../
../libs/xcl2 src/kernel_global_bandwidth.cpp ../libs/xcl2/xcl2.cpp -o 'kernel_global' -lOpenCL -lpthread -lrt -lstdc++ -L/opt/Xilinx/SDx/2018.2/runtime/lib/x86_64
src/kernel_global_bandwidth.cpp: In function 'int main(int, char**)':
src/kernel_global_bandwidth.cpp:260:89: warning: value computed is not used [-Wunused-value]
    nsduration = OCL_CHECK(err, event.getProfilingInfo(CL_PROFILING_COMMAND_END>(&err)) - OCL_CHECK(err, event.getProfilingInfo(CL_PROFILING_COMMAND_START>(&err));
                                                                    ^
mkdir -p ./xclbin
/opt/Xilinx/SDx/2018.2/bin/xocc -t hw --platform xilinx_aliyun-f3_dynamic_5_0 --save-temps --xp "param:compiler.acceleratorBinaryContent=dcp" --xp "param:compiler.preserveHlsOutput=1" --xp
"param:compiler.generateExtraRunData=true" --max_memory_ports bandwidth -DNDDBANKS=4 -c -k bandwidth -I'src' -o'xclbin/bandwidth.hw.xilinx_aliyun-f3_dynamic_5_0.xo' 'src/kernel.cl'
***** xocc v2018.2 (64-bit)
***** SW Build 2258646 on Thu Jun 14 20:02:38 MDT 2018
** Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.
```

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 the returned message contains `cannot open (No such file or directory)`, the file does not exist. You need to download the script by running the following command:

```
wget http://fpga-tools.oss-cn-shanghai.aliyuncs.com/sdaccel_package.sh
```

Step 4. Create an image

To create an image, follow these steps:

1. Run the following commands to set up the OSS environment.

```
faascmd config --id=hereIsMySecretId --key=hereIsMySecretKey # Replace hereIsMySecretId, hereIsMySecretKey with your AccessKeyId, AccessKeySecret
faascmd auth --bucket=hereIsMyBucket # Replace hereIsMyBucket with your bucket name
```

2. Run the `ls` command to obtain the file suffixed by `.xclbin`.

```
[root@vadd]# ls
bin_vadd_hw.xclbin      krnl_vadd.cl  vadd.cpp
description.json        README.md     vadd.h
Export_Compliance_Notice.md sdaccel.mk    _xocc_krnl_vadd_bin_vadd_hw.dir
```

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 packaging is completed, you can find a package file in the same directory, as shown in the following figure.

```
[root@vadd]# ls
17_10_28-021904-primary.bit      krnl_vadd.cl
17_10_28-021904-SDAccel_Kernel.tar.gz README.md
17_10_28-021904-xclbin.xml        sdaccel.mk
bin_vadd_hw.xclbin                to_aliyun
description.json                  vadd.cpp
Export_Compliance_Notice.md       vadd.h
header.bin                        _xocc_krnl_vadd_bin_vadd_hw.dir
```

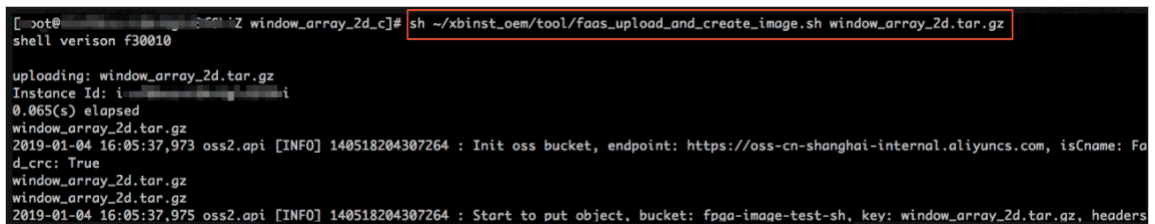
Step 5. Download the image

You can use a scripted process or step-by-step process to upload the package file and download the FPGA image.

- Scripted process: Only applicable to f3 instances with one FPGA.

- Run the following command to upload the package and generate the image file.

```
sh / root / xbinst_oem / tool / faas_upload_and_create_image
. sh < bit . tar . gz - the package to upload >
```



```
[root@ ~]# sh ~/xbinst_oem/tool/faas_upload_and_create_image.sh window_array_2d.tar.gz
shell version f30010
uploading: window_array_2d.tar.gz
Instance Id: i-xxxxxxx
0.065(s) elapsed
window_array_2d.tar.gz
2019-01-04 16:05:37,973 oss2.api [INFO] 140518204307264 : Init oss bucket, endpoint: https://oss-cn-shanghai-internal.aliyuncs.com, isCName: Fa
d_crc: True
window_array_2d.tar.gz
window_array_2d.tar.gz
2019-01-04 16:05:37,975 oss2.api [INFO] 140518204307264 : Start to put object, bucket: fpga-image-test-sh, key: window_array_2d.tar.gz, headers
```

- Download the image file.

```
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 No . in
the instance
```

0 indicates the first FPGA of the f3 instance. For single-FPGA instances, the FPGA serial No. is always 0. For instances with multiple FPGAs, such as an instance with four FPGAs, the serial No. are 0, 1, 2 and 3.

To download the same image to multiple FPGAs, add the serial No. 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.

- Run the following commands to upload the package to your OSS bucket. Then, upload gbs in your OSS bucket to the OSS bucket in the FaaS administrative unit.

```
faascmd upload_obj ect -- object = bit . tar . gz -- file =
bit . tar . gz
faascmd create_ima ge -- object = bit . tar . gz --
fpgatype = xilinx -- name = hereIsFPGA ImageName -- tags
```

```
= hereIsFPGA ImageTag -- encrypted = false -- shell =  
hereIsShell lVersionOf FPGA
```

```
[root@iz-... ~]# faascmd upload_object --object=rion.zj_test_SDAccel_Kernel.tar.gz --file=18_05_03-222718_SDAccel_Kernel.tar.gz
rion.zj_test_SDAccel_Kernel.tar.gz
18_05_03-222718_SDAccel_Kernel.tar.gz
4.735(s) elapsed

[root@iz-... ~]# faascmd create_image --object=rion.zj_test_SDAccel_Kernel.tar.gz --fpgatype=xilinx --name=rion.zj_xilinx_f3_test --tags=hereIsFPGAImageTag --encrypted=false --shell=f30001
{"Name": "rion.zj_xilinx_f3_test", "CreateTime": "Fri May 04 2018 20:24:21 GMT+0800 (CST)", "ShellUUID": "f30001", "Description": "None", "FpgaImageUUID": "xilinx1...", "State": "queued"}
0.221(s) elapsed
```

2. Run the following command to view if the FPGA image is downloadable.

```
faascmd list_image s
```

If the returned message shows `State : compiling`, the FPGA image is being compiled. If the returned message shows `State : success`, the FPGA image is ready for downloading. Find `FpgaImageUUID` and note it down.

```
[root@... ~]# faascmd list_images
{
  "FpgaImages": {
    "fpgaImage": [
      {
        "CreateTime": "Fri Jan 04 2019 16:05:43 GMT+0800 (CST)",
        "Description": "None",
        "Encrypted": "false",
        "FpgaImageUUID": "xilinx8858a3c1-...",
        "Name": "window_array_2d.tar.gz",
        "ShellUUID": "f30010",
        "State": "compiling",
        "Tags": "hereIsFPGAImageTag",
        "UpdateTime": "Fri Jan 04 2019 16:05:44 GMT+0800 (CST)"
      },
      {
        "CreateTime": "Thu Jan 03 2019 15:58:58 GMT+0800 (CST)",
        "Description": "None",
        "Encrypted": "false",
        "FpgaImageUUID": "xilinx6cbd48c1-...",
        "Name": "vadd.tar.gz",
        "ShellUUID": "f30010",
        "State": "success",
        "Tags": "hereIsFPGAImageTag",
        "UpdateTime": "Thu Jan 03 2019 16:32:32 GMT+0800 (CST)"
      }
    ]
  }
}
```

3. Run the following command. In the returned message, find and note down `FpgaUUID`.

```
faascmd list_instances --instanceId = hereIsYour
InstanceId # Replace hereIsYour InstanceId with the f3
instance ID
```

4. Run the following command to download the FPGA image.

```
faascmd download_image --instanceId = hereIsYour
InstanceId --fpgauuid = hereIsFpga UUID --fpgatype = xilinx
```

```
-- imageuuid = hereIsImag eUUID -- imagetype = afu -- shell =
hereIsShell lVersionOf Fpga
# Replace hereIsYour InstanceId with the f3 instance
ID, hereIsFpga UUID with the FpgaUUID, and
hereIsImag eUUID with the FpgaImageU UID
```

```
[root@iZ... ~]# faascmd download_image --instanceId=i-u... 4 --fpgauid=0x... 30 --fpgatype=xilinx
--imageuuid=xilinx12... i5 --imagetype=afu --shell=f30001
{"FpgaImageUUID":"xilinx12... 5","FpgaUUID":"0xc... 30","InstanceId":"i-u... 4"} "TaskStat
us": "committed"
0.223(s) elapsed
```

5. Run the following command to view if the image is downloaded successfully.

```
faascmd fpga_statu s -- fpgauid = hereIsFpga UUID --
instanceId = hereIsYour InstanceId # Replace hereIsFpga
UUID with the obtained FpgaUUID, and hereIsYour
InstanceId with the f3 instance ID
```

Below is an example of the returned message. If the FpgaImageUUID in the message is the same as the FpgaImageUUID you note down and the message shows " TaskStatus ":" valid ", the image is downloaded successfully.

```
[root@iZ... ~]# faascmd fpga_status --fpgauid=0xe... 0 --instanceId=i-u... 4
{"shellUUID":"f30001","FpgaImageUUID":"xilinx1... 5","FpgaUUID":"0xe... 0","InstanceId":"i-u... 4",
"CreateTime":"Fri May 04 2018 21:25:53 GMT+0800 (CST)","TaskStatus":"valid","Encrypted":"false"}
0.263(s) elapsed
```

Step 6: Run the Host program

To run the Host program, follow these steps:

1. Run the following command to configure the environment.

```
source / root / xbinst_oem / F3_env_set up . 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_lo g = " run . log "
hal_log = hal . log
ert = false
kds = false
```

3. Run the Host.

- For vadd, run the following commands:

```
make - f sdaccel . mk host
```

```
./ vadd bin_vadd_h w . xclbin
```

- For kernel_global_bandwidth, run the following command:

```
./ kernel_glo bal
```

If `Test Passed` is returned, the test is successful.

Other common commands

This section introduces some common commands for f3 instances.

Task	Command
View the help document	<code>make -f ./ sdaccel . mk help</code>
Run software emulation	<code>make -f ./ sdaccel . mk run_cpu_em</code>
Run hardware emulation	<code>make -f ./ sdaccel . mk run_hw_em</code>
Compile the host code only	<code>make -f ./ sdaccel . mk host</code>
Compile and generate files for downloading	<code>make -f sdaccel . mk xbin_hw</code>
Clean a work directory	<code>make -f sdaccel . mk clean</code>
Forcibly clean a work directory	<code>make -f sdaccel . mk cleanall</code>



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 download them at [SDAccel runtime](#) and [SDAccel development platform](#).

10.4 Best practices for RTL design on an f3 instance

This topic describes how to implement the Register Transfer Level (RTL) design on an f3 instance.

Prerequisites

- [Create an f3 instance](#) and add a security group rule to allow Internet access to SSH port 22 of the instance.
- Log on to the [ECS console](#) to obtain the instance ID on the details page of the f3 instance.
- [Create an OSS bucket](#) in China (Shanghai) for the FaaS service.



Note:

The bucket will provide read and write access to the FaaS administrative account. We recommend that you do not store objects that are not related to FaaS.

- To operate an f3 instance as a RAM user, do the following:
 - [Create a RAM user](#) and [grant permissions](#).
 - [Create a RAM role](#) and [grant permissions](#).
 - Create the AccessKey ID and AccessKey Secret.

Background information

All the operations described in this topic must be performed by one account in the same region.

We recommend that you use an f3 instance as a RAM user. For security purposes, we also recommend that you grant the RAM user the minimum level of necessary permissions, for example, the permissions to access DCP/xclbin files in OSS buckets, upload the Vivado compilation log, and operate specified ECS instances. Additionally, you need to specify the RAM role AliyunFAASDefaultRole, which is used by the FaaS service to access your resources hosted in other Alibaba Cloud products. The policy (AliyunFAASRolePolicy) of this RAM role includes permissions on the Key Management Service (KMS), through which you can encrypt IP addresses.

Procedure

1. [Connect to your f3 instance](#).



Note:

It takes two or three hours to compile the project. We recommend that you use `nohup` or VNC to connect to the instance to avoid unexpected disconnection.

2. Download and decompress the [RTL reference design](#).

3. Configure the f3 environment.

- If the driver is `xdma`, run the following command to configure the environment:

```
source / root / xbinst_oem / F3_env_set up . sh xdma # Run
this command each time you open a new terminal
window
```

- If the driver is `xocl`, run the following command to configure the environment:

```
source / root / xbinst_oem / F3_env_set up . sh xocl # Run
this command each time you open a new terminal
window
```



Note:

Configuring the environment mainly includes mounting 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 detecting the `faascmd` version.

4. Specify an OSS bucket.

```
faascmd config -- id = hereIsYour SecretId -- key =
hereIsYour SecretKey # Replace hereIsYour SecretId and
hereIsYour SecretKey with your RAM user AccessKey
faascmd auth -- bucket = hereIsYour Bucket # Replace
hereIsYour Bucket with your OSS bucket name
```

5. Run the following commands to compile the RTL project:

```
cd < decompress ed directory >/ hw / # Enter the
decompress ed hw directory
sh compiling . sh
```



Note:

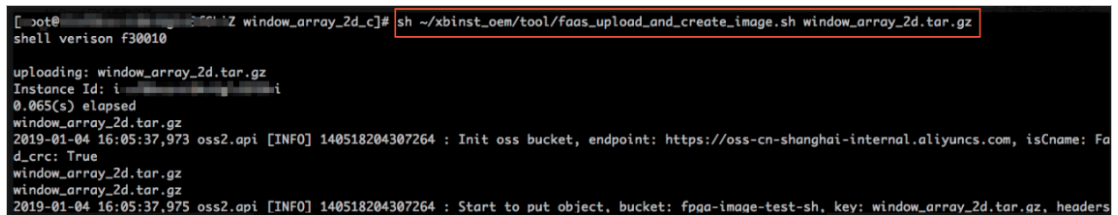
It takes two or three hours to compile the project.

6. Upload the Netlist files and download the FPGA image. You can use the scripted process or the step-by-step process to finish this task.

- Scripted process: Applicable to the f3 instances with a single FPGA chip.

- a. Run the following commands to upload the package and generate the image file:

```
sh / root / xbinst_oem / tool / faas_upload_and_create_image . sh < bit . tar . gz - the package to upload >
```



```
[root@ ~]# sh ~/xbinst_oem/tool/faas_upload_and_create_image.sh window_array_2d.tar.gz
shell version f30010

uploading: window_array_2d.tar.gz
Instance Id: i-00000000
0.065(s) elapsed
window_array_2d.tar.gz
2019-01-04 16:05:37,973 oss2.api [INFO] 140518204307264 : Init oss bucket, endpoint: https://oss-cn-shanghai-internal.aliyuncs.com, isCName: False, d_crc: True
window_array_2d.tar.gz
window_array_2d.tar.gz
2019-01-04 16:05:37,975 oss2.api [INFO] 140518204307264 : Start to put object, bucket: fpga-image-test-sh, key: window_array_2d.tar.gz, headers: {}
```

- b. Download the image file.

```
sh / root / xbinst_oem / tool / faas_download_image . sh < bit . tar . gz - the package filename > < 0 / 1 > # The last number < 0 / 1 > stands for the FPGA serial No . of the instance
```

0 indicates the first FPGA of the f3 instance. For single-FPGA instances, the FPGA serial No. is always 0. For instances with multiple FPGAs, such as an instance with four FPGAs, the serial No. are 0, 1, 2 and 3.

To download the same image to multiple FPGAs, add the serial No. to the end of the command. For example, to download the same image to four FPGAs, use the following commands:

```
sh / root / xbinst_oem / tool / faas_download_image . sh < bit . tar . gz - package filename > 0
sh / root / xbinst_oem / tool / faas_download_image . sh < bit . tar . gz - package filename > 1
sh / root / xbinst_oem / tool / faas_download_image . sh < bit . tar . gz - package filename > 2
sh / root / xbinst_oem / tool / faas_download_image . sh < bit . tar . gz - package filename > 3
```

- Step-by-step process: [Use the faascmd tool](#) to perform the operations.

- a. Run the following commands to upload the package to your OSS bucket, and then upload gbs in your OSS bucket to the OSS bucket of the FaaS unit:

```
faascmd upload_object -- object = bit . tar . gz -- file = bit . tar . gz
faascmd create_image -- object = bit . tar . gz -- fpgatype = xilinx -- name = hereIsFPGA ImageName -- tags
```

```
= hereIsFPGA ImageTag -- encrypted = false -- shell =
hereIsShell lVersionOf FPGA
```

```
[root@iz-... ~]# faascmd upload_object --object=rion.zj_test_SDAccel_Kernel.tar.gz --file=18_05_03-222718_SDAccel_Kernel.tar.gz
rion.zj_test_SDAccel_Kernel.tar.gz
18_05_03-222718_SDAccel_Kernel.tar.gz
4.735(s) elapsed
```

```
[root@iz-... ~]# faascmd create_image --object=rion.zj_test_SDAccel_Kernel.tar.gz --fpgatype=xilinx --name=rion.zj_xilinx_f3_test --tags=hereIsFPGAImageTag --encrypted=false --shell=f30001
{"Name": "rion.zj_xilinx_f3_test", "CreateTime": "Fri May 04 2018 20:24:21 GMT+0800 (CST)", "ShellUUID": "f30001", "Description": "None", "FpgaImageUUID": "xilinx1...5", "State": "queued"}
0.221(s) elapsed
```

- b. Run the following command to check if the FPGA image is ready for downloading:

```
faascmd list_image s
```

If the returned message shows `State : compiling`, the FPGA image is being compiled, and you still need to wait. If the returned message shows

State : success , the FPGA image is ready for downloading. Find the FpgaImageUUID and note it down.

```
[root@ ~]# faascmd list_images
{
  "FpgaImages": {
    "fpgaImage": [
      {
        "CreateTime": "Fri Jan 04 2019 16:05:43 GMT+0800 (CST)",
        "Description": "None",
        "Encrypted": "false",
        "FpgaImageUUID": "xilinx8858a3c1-...",
        "Name": "window_array_2d.tar.gz",
        "ShellUUID": "f30010",
        "State": "compiling",
        "Tags": "hereIsFPGAImageTag",
        "UpdateTime": "Fri Jan 04 2019 16:05:44 GMT+0800 (CST)"
      },
      {
        "CreateTime": "Thu Jan 03 2019 15:58:58 GMT+0800 (CST)",
        "Description": "None",
        "Encrypted": "false",
        "FpgaImageUUID": "xilinx6cbd48c1-...",
        "Name": "vadd.tar.gz",
        "ShellUUID": "f30010",
        "State": "success",
        "Tags": "hereIsFPGAImageTag",
        "UpdateTime": "Thu Jan 03 2019 16:32:32 GMT+0800 (CST)"
      }
    ]
  }
}
```

- c. Run the following command. In the returned message, note down the FpgaUUID.

```
faascmd list_instances -- instanceId = hereIsYour
InstanceId # Replace hereIsYour InstanceId with the
f3 instance ID
```

- d. Run the following command to download the FPGA image:

```
faascmd download_image -- instanceId = hereIsYour
InstanceId -- fpgauid = hereIsFpga UUID -- fpgatype =
xilinx -- imageuid = hereIsImageUUID -- imagetype = afu
-- shell = hereIsShellVersionOf Fpga
# Replace hereIsYour InstanceId with the f3
instance ID , hereIsFpga UUID with the obtained
```

FpgaUUID , and hereIsImageUUID with the obtained FpgaImageUUID

```
[root@iz-... ~]# faascmd download_image --instanceId=i-u... --fpgauid=0xe... --fpgatype=xilinx
--imageuid=xilinx12 --imagetype=afu --shell=f30001
{"FpgaImageUUID":"xilinx12", "FpgaUUID":"0xe...", "InstanceId":"i-u...", "TaskStatus": "committed"}
0.223(s) elapsed
```

- e. Run the following command to check whether the image has been successfully downloaded:

```
faascmd fpga_status --fpgauid = hereIsFpgaUUID --instanceId = hereIsYourInstanceId # Replace hereIsFpgaUUID with the obtained FpgaUUID , and hereIsYourInstanceId with the f3 instance ID
```

The following is an example of the returned message. If the FpgaImageUUID in the message is identical to the FpgaImageUUID you note down, and the message shows " TaskStatus ":" valid ", the image has been successfully downloaded.

```
[root@iz-... ~]# faascmd fpga_status --fpgauid=0xe... --instanceId=i-u...
{"shellUUID":"f30001", "FpgaImageUUID":"xilinx12", "FpgaUUID":"0xe...", "InstanceId":"i-u...", "TaskStatus": "valid", "Encrypted": "false"}
0.263(s) elapsed
```

Create a RAM user and grant permissions

To create a RAM user and grant permissions to this RAM user, follow these steps:

1. Log on to the [RAM console](#).
2. In the left-side navigation pane, click Users.
3. On the User Management page, click Create User.

4. Set the User Name, Display Name, and Email, select Automatically generate an AccessKey for this user, and then click OK.

Create User

* User Name :

The name can contain 1 to 64 characters, including lowercase letters a-z, uppercase letters A-Z, digits 0-9, and only these special characters: period (.), underscore (_), and hyphen (-).

Display Name :

Display names must contain 1-128 characters. They may include Chinese characters, lowercase letters a-z, numbers 0-9, and these special characters: (@) (.) (_) (-).

Email :

Country/Region :

China(+86)

Phone :

Description :

RAM user for operating FaaS

☒ Automatically generate an Access key for this user.

OK

Cancel

5. Click Save AccessKey Information.




Notice:

Each AccessKey can be downloaded only once. You must keep your AccessKeyID and AccessKeySecret safe and confidential. If you lose your AccessKey, you must create a new one. For more information, see [#unique_122](#).

Create User

This is the only time a user's Access key can be downloaded. Save the Access key now.

 Access key successfully created.

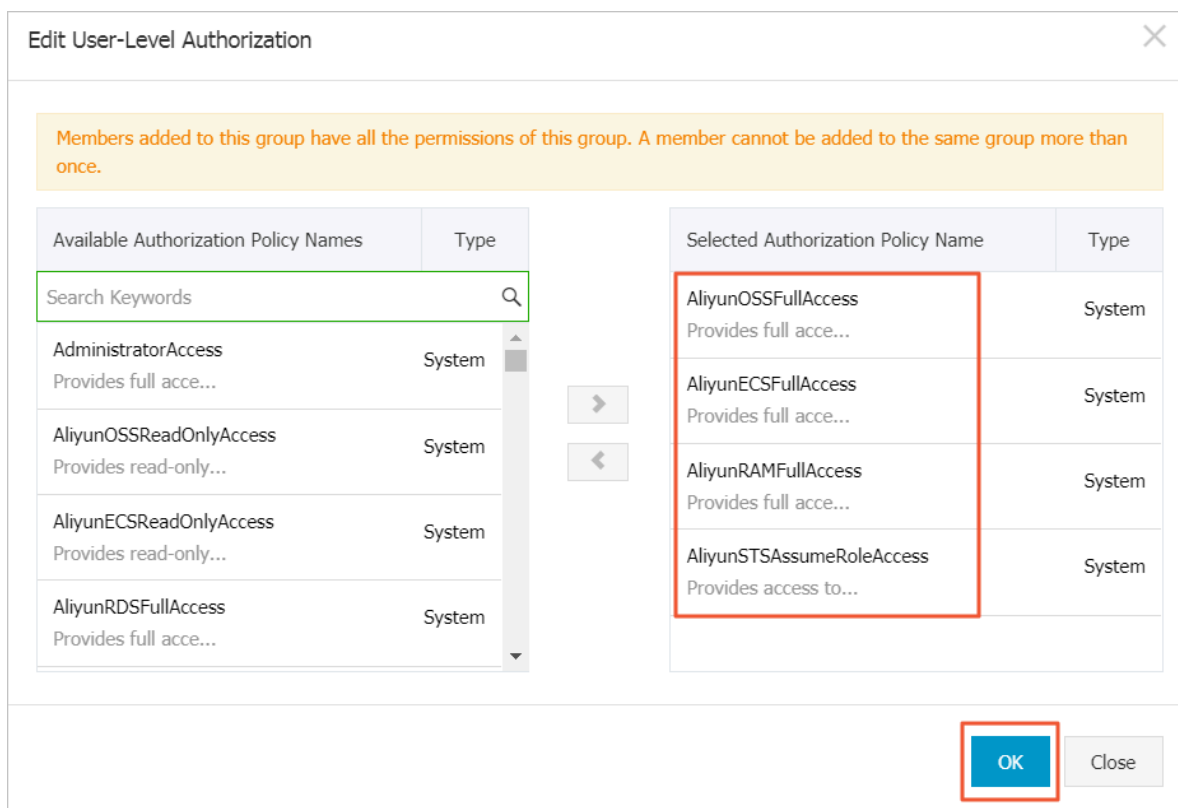
Access Key Details

▼

Save Access Key Information

6. On the User Management page, click Authorize in the Actions column of the new RAM user.

7. Grant permissions to the RAM user, including AliyunOSSFullAccess, AliyunECSFullAccess, AliyunRAMFullAccess and AliyunSTSAssumeRoleAccess, and then click OK.



FAQ

Question 1: How do I view the details of errors that occur during image upload?

If your project reports errors during image upload, such as compilation errors, you can view the error details in two ways:

- Check `faas_compiling.log`. When the upload script `faas_upload_and_create_image.sh` is used, `faas_compiling.log` is automatically downloaded and printed onto the terminal if compilation fails.
- Run the command to view the log file: `sh / root / xbinst_oem / tool / faas_check_log . sh < bit . tar . gz - package uploaded previously >`

Question 2: How do I reload the image?

To reload the image, follow these steps:

1. Uninstall the driver.

- If you have installed the `xdma` driver, run the command `sudo rmmod xdma` in the instance to uninstall it.
- If you have installed the `xocl` driver, run the command `sudo rmmod xocl` in the instance to uninstall it.

2. Download the image in either of the two ways :

- Use the script.

```
sh faas_download_image . sh bit . tar . gz < 0 / 1 > # The
last number stands for the FPGA serial No . of
the instance
```

- Use `faascmd`.

```
faascmd download_image -- instanceId = hereIsYour
InstanceId -- fpgauid = hereIsFpga UUID -- fpgatype = xilinx
-- imageuid = hereIsImageUUID -- imagetype = afu -- shell =
hereIsShellVersionOf Fpga
```

3. Install the driver.

- To install the `xdma` driver, run the following command:

```
sudo depmod
sudo modprobe xdma
```

- To install the `xocl` driver, run the following command:

```
sudo depmod
sudo modprobe xocl
```

10.5 faascmd tool

10.5.1 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.

10.5.2 Install faascmd

This topic describes how to download and install faascmd.

Preparations

- Perform the following steps on the instance for which you want to run faascmd:

- Run the following command to check that the Python version is 2.7.x.

```
python -V
```

```
[root@testhost script]# python -V
Python 2.7.5
```

- Install the Python module by running the following commands:

```
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
```

- Run the following command to check that the aliyun-python-sdk-core version is 2.11.0 or later.

```
cat /usr/lib/python2.7/site-packages/aliyun-sdk-core/
__init__.py
```

```
[root@testhost python2.7]# cat /usr/lib/python2.7/site-packages/aliyun-sdk-core/
__init__.py
version = "2.11.0" [root@testhost python2.7]#
```



Note:

If the version is earlier than 2.11.0, run `pip install --upgrade aliyun-python-sdk-core` to upgrade aliyun-python-sdk-core to the latest version.

- [Obtain the AccessKeyID and AccessKeySecret of the RAM user.](#)

Procedure

- Log on to your instance and run `wget http://fpga-tools.oss-cn-shanghai.aliyuncs.com/faascmd` in the current or any other directory to download faascmd.



Note:

When you [configure faascmd](#), you need to add the absolute path of the directory where faascmd is installed to the PATH variable.

2. Add executable permissions to faascmd by running the following command:

```
chmod +x faascmd
```

10.5.3 Configure faascmd

Before using faascmd, you need to configure the related environment variable and the AccessKey of the RAM user.

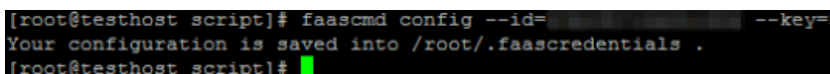
Procedure

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=< yourAccess KeyID > --key=<
yourAccess KeySecret >
```



```
[root@testhost script]# faascmd config --id= --key=
Your configuration is saved into /root/.faascredentials .
[root@testhost script]#
```

10.5.4 Use faascmd

This topic describes how to use faascmd commands.

Prerequisite

You have [configured faascmd](#) before using it.

Syntax description

- All commands and parameters provided by faascmd are case-sensitive.
- There must be no space before and after equal signs (=) in the parameters of faascmd commands.

Authorize users

The `faascmd auth` command is used to authorize the faas admin user to access the users' OSS buckets.

Prerequisites

1. You have created an OSS bucket for FaaS to upload the originally compiled DCP file.
2. You have created a folder named `compiling_logs` in the FaaS OSS bucket.

Command format

```
faascmd auth -- bucket =< yourFaasOS SBucketName >
```

Code example

```
[root@testhost script]# faascmd auth --bucket=juliabucket
faasRole has existed!
RAMSECTION has existed!
OSSSECTION has existed!
RoleArn: acs:ram::[REDACTED]:role/faasrole
Create role success
faasPolicy has not existed! Create it Now!
Create policy success
Attach policy to role success
0.459(s) elapsed
```



Note:

If an Alibaba Cloud account has multiple RAM user accounts, we recommend that the RAM user accounts share an OSS bucket to prevent authorization policies from being repeatedly modified or overwritten.

View authorization policies

The `faascmd list_policy` command is used to view whether the specified OSS bucket has been added to the corresponding authorization policy (faasPolicy).

Command format

```
faascmd list_policy
```

Code example

```
[root@testhost script]# faascmd list_policy
VersionId : v1   CreateTime : 2018-11-09T03:22:01Z   IsDefaultVersion : True
{
  "Statement": [
    {
      "Action": "ecs:DescribeInstances",
      "Effect": "Allow",
      "Resource": "acs:ecs:*:*:*"
    }
  ],
}
```



Note:

You need to check whether your OSS bucket and OSS bucket/compiling_logs appear in the policy information.

Delete authorization policies

The `faascmd delete_policy` command is used to delete authorization policies (faasPolicy).

Command format

```
faascmd delete_policy
```

Code example

```
[root@testhost script]# faascmd delete_policy
Detach faasPolicy from faasRole successfully!!!
Delete the faasPolicy successfully!!!
0.306(s) elapsed
```



Note:

If an Alibaba Cloud account has multiple RAM user accounts, we recommend that you delete the target policy in the RAM console to prevent incorrect authorization policy deletion.

View all objects under an OSS bucket

The `faascmd list_objects` command is used to view all objects under your OSS bucket.

Command format

```
faascmd list_objects
```

Code example

```
[root@testhost script]# faascmd list_objects
compiling_logs/
juliabucket
juliafile
0.081(s) elapsed
[root@testhost script]# faascmd list_objects |grep "julia"
0.082(s) elapsed
juliabucket
juliafile
```



Note:

You can use this command with the `grep` command to filter for the files you want, for example, `faascmd list_objec ts | grep " xxx "`.

Upload original compilation files

The `faascmd upload_obj ect` command is used to upload the original files that are compiled on your local PC to a specified OSS bucket.

Command format

```
faascmd upload_obj ect -- object =< newFileNam einOSSBuck et >
-- file = < your_file_ path >/ fileNameYo uWantToUpl oad
```

Code example

```
[root@testhost script]# faascmd upload_object --object=juliaOSSFile1 --file=julia_test.tar
juliaOSSFile1
julia_test.tar
0.091(s) elapsed
[root@testhost script]# faascmd upload_object --object=juliaOSSFile2 --file=/opt/dcp1_0/testfile.tar
juliaOSSFile2
/opt/dcp1_0/testfile.tar
0.089(s) elapsed
```



Note:

- No path is needed if the target files are stored in the current directory.
- Locally compiled original files provided by Intel FPGA are in .gbs format and those provided by Xilinx FPGA are compressed as packages in .tar format after script processing.

Download objects from an OSS bucket

The `faascmd get_object` command is used to download a specified object from an OSS bucket.

Command format

```
faascmd get_object -- obejct =< yourObject Name > -- file =<
your_local _path >/< yourFileNa me >
```

Code example

```
[root@ ~]# faascmd get_object --object=juliaOSSFile3 --file=vivadol.log
2018-12-04 10:09:47.342 oss2.api [INFO] 140410558318400 : Init oss bucket, endpoint: https://oss-cn-hangzhou-internal.aliyuncs.com, isCName: False, connect_timeout: None, app_name: , enabled_crc: True
juliaOSSFile3
vivadol.log
2018-12-04 10:09:47.344 oss2.api [INFO] 140410558318400 : Start to get object to file, bucket: juliabucket, key: juliaOSSFile3, file path: vivadol.log
2018-12-04 10:09:47.344 oss2.api [INFO] 140410558318400 : Start to get object, bucket: juliabucket, key: juliaOSSFile3, range: , headers: {}, params: {}
2018-12-04 10:09:47.456 oss2.api [INFO] 140410558318400 : Get object done, req_id: 5c08e12b74f2a9b7521728b, status_code: 200
0.117(s) elapsed
```



Note:

If no path is provided, the objects are downloaded to the current folder by default.

Create FPGA images

The `faascmd create_image` command is used to submit FPGA image creation requests. If the request succeeds, `fpga_imageuuid` is returned.

Command format

```
faascmd create_image --object=< yourObject Name >
-- fpgatype=< intel / xilinx > -- encrypted=< true / false >
-- kmskey=< key / mandatory if encrypted is true >
-- shell=< Shell Version / mandatory > -- name=< name / optional
>
-- description=< description / optional > -- tags=< tags /
optional >
```

Code example

```
[root@testhost script]# faascmd create_image --object=juliasbucket --fpgatype=intel --encrypted=false --shell=V1.1
{"Name":"None","CreateTime":"Fri Nov 09 2018 11:42:47 GMT+0800 (CST)","ShellUUID":"V1.1","Description":"None","FpgaImageUUID":"", "State":"queued"}
0.255(s) elapsed
```

View FPGA images

The `faascmd list_images` command is used to view information about all the FPGA images you have created.

Command format

```
faascmd list_images
```

Code example

```
[root@testhost script]# faascmd list_images
{
  "FpgaImages": {
    "fpgaImage": [
      {
        "CreateTime": "Fri Nov 09 2018 11:42:47 GMT+0800 (CST)",
        "Description": "None",
        "Encrypted": "false",
        "FpgaImageUUID": " ",
        "Name": "None",
        "ShellUUID": "V1.1",
        "State": "success",
        "Tags": "None",
        "UpdateTime": "Fri Nov 09 2018 11:43:53 GMT+0800 (CST)"
      }
    ]
  }
}
0.076(s) elapsed
```



Note:

A maximum of 10 FPGA images can be reserved for each RAM user account.

Delete FPGA images

The `faascmd delete_image` command is used to delete FPGA images.

Command format

```
faascmd delete_image --imageuuid =< yourImageu uid >
```

Code example

```
[root@testhost script]# faascmd delete_image --imageuuid=
{"Status":200,"FpgaImageUUID":"j","Message":"delete succeed!"}
0.143(s) elapsed
```

Download FPGA images

The `faascmd download_image` command is used to submit FPGA image download requests.

Command format

```
faascmd download_image --instanceId =< yourInstan ceId >
-- fpgauid =< yourfpgauu id > -- fpgatype =< intel / xilinx >
-- imageuuid =< yourImageu uid > -- imagetype =< afu >
-- shell =< yourImageS hellVersio n >
```

Code example

```
faascmd download_image --instanceId = XXXXX -- fpgauid = XXXX
-- fpgatype = intel -- imageuuid = XXXX
```

View the FPGA image download status

The `faascmd fpga_status` command is used to view the status of the current FPGA board card and the FPGA image download status.

Command format

```
faascmd fpga_status -- fpgauid =< fpgauid > -- instanceId =<
instanceId >
```

Code example

```
[root@testhost script]# faascmd fpga_status --fpgauid=
--instanceId=
{"shellUUID":"V1.0","FpgaImageUUID":"","FpgaUUID":"","askStatus":"invalid","Encrypted":"false"}
0.310(s) elapsed
```

Publish FPGA images

The `faascmd publish_image` command is used to submit FPGA image publishing requests.

Command format

```
faascmd publish_image -- imageuuid =< yourImageu  uid > --  
imageid =< yourFPGAIm  ageid >
```



Note:

- `imageuuid` is the ID of the image you are going to publish to the cloud marketplace. You can view the image ID by running the `faascmd list_images` command.
- `imageid` is the FPGA image ID. You can view the ID on the instance details page in the ECS console.

View FPGA instance information

The `faascmd list_instances` command is used to obtain basic information about an FPGA instance, including the instance ID, FPGA board card information, and shell version.

Command format

```
faascmd list_instances -- instanceId =< yourInstan  ceId >
```

Code example

```
[root@testhost script]# faascmd list_instances --instanceId=  
{  
  "Instances": {  
    "instance": [  
      {  
        "DeviceBDF": "05:00.0",  
        "FpgaStatus": "invalid",  
        "FpgaType": "intel",  
        "FpgaUUID": "  
        "InstanceId": "  
        "ShellUUID": "V1.1"  
      }  
    ]  
  }  
}  
0.275(s) elapsed
```

10.5.5 FAQ

This topic lists common FAQs relating to the faascmd tool and provides corresponding solutions.

FAQ

- What do I do if an error indicating "Name Error:global name 'ID' is not defined." is reported?

Cause: faascmd cannot obtain your AccessKeyId or AccessKeySecret.

Solution: Run the `faascmd config` command. Then, the information about the AccessKeyId and AccessKeySecret you have entered will be saved in the `/root/.faascreden tials` file.

- What do I do if an error indicating "HTTP Status:403 Error:RoleAccessError. You have no right to assume this role." is reported?

Cause: faascmd cannot obtain information about the role ARN or the obtained ARN does not belong to the same account as the existing AccessKeyId and AccessKeySecret.

Solution: Check whether the following information is contained in the `/root/.faascreden tials` file:

```
[ FaaScreden tials ]
accessid = xxxxxxxxxx
accesskey = xxxxxxxxxx xxxxxxxxxx xxx
[ Role ]
role = acs : ram :: 1234567890 123456 : role / xxxxxx
[ OSS ]
bucket = xxxx
```



Note:

- If the preceding information already exists, check whether the role ARN and the AccessKeyId/AccessKeySecret belong to the same account.
- If the preceding information does not exist, run `faascmd auth bucket = xxxx` to grant permissions.

- What do I do if an error indicating "HTTP Status: 404 Error: EntityNotExist. Role Error. The specified Role not exists." is reported?

Cause: There is no faasrole role in your account.

Solution: Log on to the RAM console to check whether a faasrole role exists.

- If no faasrole role exists, run the `faascmd config` and `faascmd auth` commands to create such a role and grant permissions to it.
- If a faasrole role already exists, open a ticket.
- What do I do if an error indicating "SDK.InvalidRegionId. Can not find endpoint to access." is reported?

Cause: faascmd cannot obtain the endpoint address of FaaS.

Solution: Perform the following steps check whether faascmd configurations meet the specified requirements:

- Run the `python -V` command to check whether the Python version 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/aliyun_sdk_core/__init__.py` command to check whether the aliyunsdkcore version is 2.11.0 or later.



Note:

If the aliyunsdkcore version is earlier than 2.11.0, you need to run the `pip install --upgrade aliyun-python-sdk-core` command to upgrade the aliyunsdkcore to the latest version.

- What do I do if an error indicating "HTTP Status:404 Error:SHELL NOT MATCH The image Shell is not match with fpga Shell! Request ID:D7D1AB1E-8682-4091-8129-C17D54FD10D4" is returned when I download an image?

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 need to create an FPGA image whose shell version is the same as that of the FPGA, and then download the image.
- If the two shell versions are consistent, open a ticket.

- What do I do if an error indicating "HTTP Status:503 Error:ANOTHER TASK RUNNING. Another task is running,user is allowed to take this task half an hour Request ID: 5FCB6F75-8572-4840-9BDC-87C57174F26D" is returned when I download an image?

Cause: The FPGA is stuck in operating state due to unexpected failure or interruption of the download request you have submitted.

Solution: Wait for 10 minutes until the download task ends, and then resubmit an image download request.



Note:

If the problem persists, open a ticket.

- What do I do if the image status is failed when I run the `faascmd list_images` command?

Solution: Obtain the compilation logs for troubleshooting by running the following command:

```
faascmd list_objects | grep vivado
faascmd get_object -- object =< yourObject Name > -- file =<
your_local_path >/ vivado . log # The path is optional .
```

The compilation logs are downloaded to the current folder by default.

Common error codes

faascmd command	API name	Error message	Error description	Error code
Applicable to all commands	Applicable to all APIs	PARAMETER INVALIDATE	The input parameter is incorrect.	400
Applicable to all commands	Applicable to all APIs	InternalError	There is an internal error. Please open a ticket.	500
auth	auth	NoPermisson	You do not have the permission to access a specific open API.	403
create_image	CreateFpga Image	IMAGE NUMBER EXCEED	There cannot be more than 10 images in the image list . Please delete unnecessary images and try again.	401
		FREQUENCY ERROR	The interval for submitting image requests is 30 minutes.	503
		SHELL NOT SUPPORT	The input shell version is not supported. Please verify that the shell version is correct.	404
		EntityNotExist. RoleError	The current account has no faasrole role.	404
		RoleAccess Error	The role ARN is empty, or the role ARN and the AccessKeyId/AccessKeySecret do not belong to the same account.	403
		InvalidAccessKeyIdError	The AccessKeyId/AccessKeySecret is invalid.	401
		Forbidden. KeyNotFoundError	The specified KMS key cannot be found. Please log on to the KMS console and check whether the input KeyId exists .	503

faascmd command	API name	Error message	Error description	Error code
		AccessDeniedError	The faas admin account is not authorized to access the current bucket.	
		OSS OBJECT NOT FOUND	The specified OSS bucket/object does not exist or is inaccessible.	404
delete_image	DeleteFpgaImage	IMAGE NOT FOUND	The specified FPGA image cannot be found.	400
list_instances	DescribeFpgaInstances	NOT AUTHORIZED	The specified instance does not exist or does not belong to the current account.	401
		RoleAccess Error	The role ARN is empty, or the role ARN and the AccessKeyId/AccessKeySecret do not belong to the same account.	403
		INSTANCE INVALIDATE	The specified instance is not an FPGA instance. If the specified instance is an FPGA instance, please open a ticket.	404
fpga_statuses	DescribeLoadTaskStatus	NOT AUTHORIZED	The specified instanceId cannot be found. Please check the input parameter.	401
		FPGA NOT FOUND	The specified fpgauid cannot be found. Please check the input parameter.	404
download_image	LoadFpgaImage	ANOTHER TASK RUNNING	The image download task you submitted is still in operating state.	503
		IMAGE ACCESS ERROR	The specified image does not belong to the current account.	401
		YOU HAVE NO ACCESS TO THIS INSTANCE	The specified instance does not belong to the current account.	401
		IMAGE NOT FOUND	The specified FPGA image cannot be found.	404

faascmd command	API name	Error message	Error description	Error code
		FPGA NOT FOUND	The specified FPGA cannot be found.	404
		SHELL NOT MATCH	The image and the specified FPGA do not match in shell version.	404
		RoleAccess Error	The role ARN is empty, or the role ARN and the AccessKeyI d/AccessKeySecret do not belong to the same cloud account.	403
		Image not in success state	The specified image is not in success state. Only images in success state can be downloaded.	404
publish_image	PublishFpgaImage	FPGA IMAGE STATE ERROR	The specified image is not in success state.	404
		FPGA IMAGE NOT FOUND	The specified image cannot be found or does not belong to the current account.	404

11 Shrink disk volume

Currently, Elastic Compute Service (ECS) does not support system disk or data disk volume shrink. If you want to shrink your disk volumes, try [Alibaba Cloud Migration Tool](#) instead.

Though Cloud Migration Tool is designed to balance the cloud-based and offline workloads of Alibaba Cloud users, you can use it to shrink ECS disk volumes.

The tool creates a custom image based on your ECS instance. During this process, it re-specifies the size of the disk to shrink it. Apart from replacing the target object with an ECS instance, the tools for cloud migration and disk volume shrinking are identical, in terms of [both operation and limitations](#). Because the ECS instance is already virtual, it is more convenient to use and the chances of reporting errors is reduced.

However, using this tool may change some attributes of the ECS instance. For example, instance ID (`InstanceId`) and public IP. If your instance is a [VPC-Connected](#) instance, you can reserve the public IP address by [converting public IP address to EIP address](#). We recommend that users using [Alibaba Cloud Elastic IP \(EIP\)](#) and users with less dependency on public IP use this approach to shrink the disk volume.

Prerequisites

- When the disk is mounted on a Linux instance, you must first install rsync, a remote data synchronization tool.
 - CentOS Instance: Run `yum install rsync -y`.
 - Ubuntu Instance: Run `apt-get install rsync -y`.
 - Debian Instance: Run `apt-get install rsync -y`.
 - Other distributions: Please visit the official website to find the relevant installation documents.
- You must [create an AccessKey](#) in the console first, which is used to output it into the configuration file `user_config.json`.



Note:

To prevent data leakage due to excessive permissions for AccessKey, we recommend that you [create a RAM sub-account](#) and use this account to [create an AccessKey](#).

- For other prerequisites and limitations, see [migrate to Alibaba Cloud by using Cloud Migration Tool](#).

Procedure

1. [Connect](#) to the target ECS instance by using the administrator/root account.
2. [Download](#) the Alibaba Cloud Migration Tool zip file.
3. Unzip the Cloud Migration Tool. Enter the corresponding operating system and version of the client file directory to find the configuration file `user_config.json`.
4. See customize [user_config.json](#) to complete the configuration.

See the following figure for the configuration file in a Linux instance.

```
"access_id": "",
"secret_key": "",
"region_id": "",
"image_name": "",
"system_disk_size": ,
"platform": "",
"architecture": "",
"data_disks": [],
"bandwidth_limit": 0
```

The most important parameters to configure for shrinking the disk volume are as follows:

- `system_disk_k_size` : Set this parameter to the expected system disk size in GB. The value cannot be less than the actual size of the system disk.
- `data_disks` : Set this parameter to the expected data disk size in GB. The value cannot be less than the actual size of the data disk.



Note:

- When a Linux instance comes with a data disk, the `data_disks` parameter is required even if you do not want to shrink the data disk volume. If it is not configured, Cloud Migration Tool copies data from the data disk to the system disk by default.
- When a Windows instance comes with a data disk, the `data_disks` parameter is optional if you do not want to shrink the size of the data disk.

5. Run the program go2aliyun_client.exe:

- Windows instance: Right-click go2aliyun_client.exe and select Run as administrator.
- Linux instance:
 - a. Run `chmod +x go2aliyun_client` to give the client executable permissions.
 - b. Run `./go2aliyun_client` to run the client.

6. Wait for the running results:

- If `Goto Aliyun Finished !` is displayed, go to the [ECS console](#) and check the custom image after shrinking. If the custom image has been generated, you can release the original instance and use the custom image to [create an ECS instance](#). After you create a new instance, the disk volume shrinking process is complete.
- If `Goto Aliyun Not Finished !` is displayed, check the log files in the same directory for [troubleshooting](#). After fixing any problems, run Cloud Migration Tool again to resume volume shrinking. The tool continues the most recent migration progress and does not start over.

References

- For a detailed introduction to Cloud Migration Tool, see [what is Alibaba Cloud Migration Tool](#).
- For instructions on how to use Cloud Migration Tool, see [migrate to Alibaba Cloud by using Cloud Migration Tool](#).

12 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.

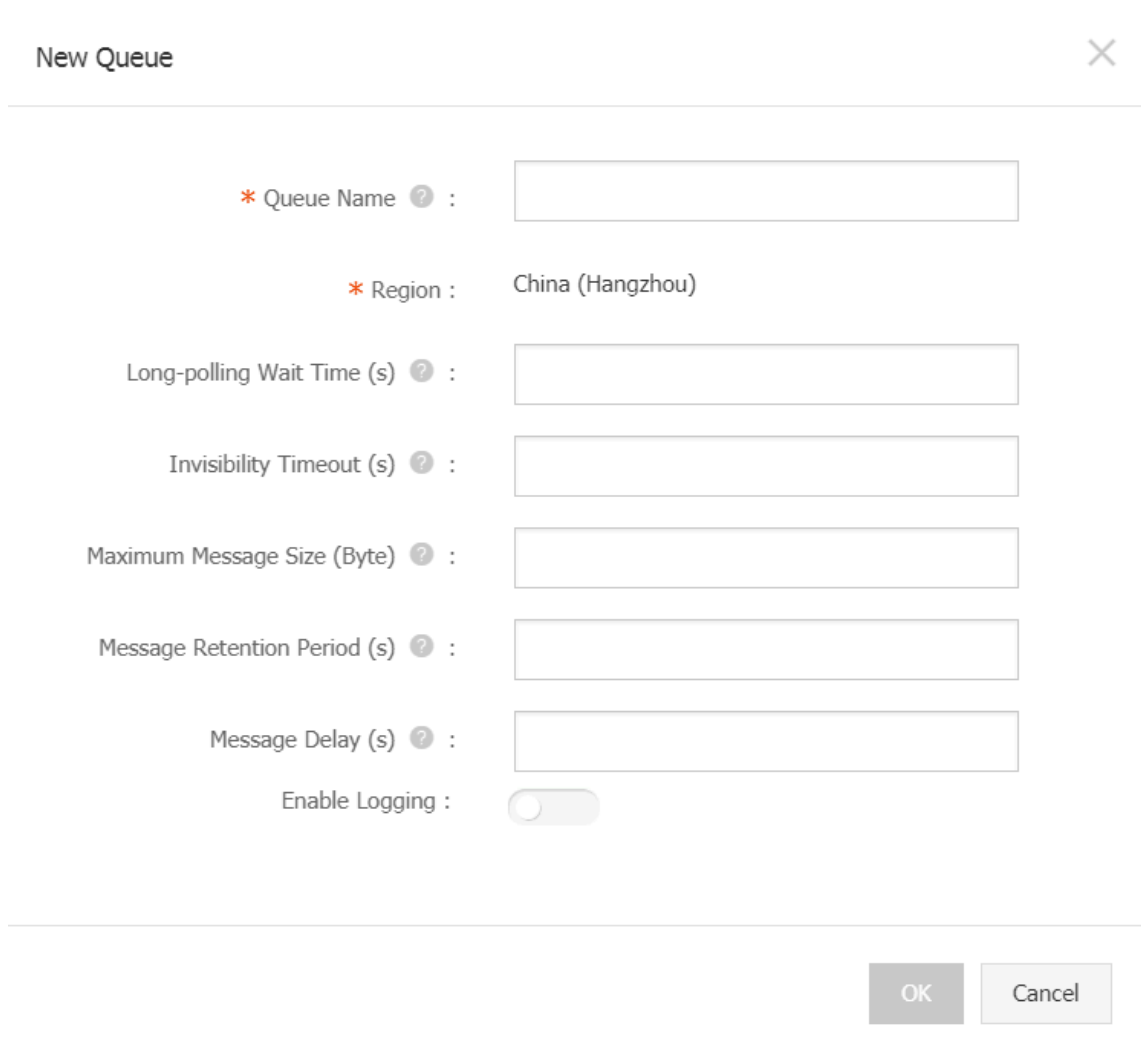
To automate the processing of ECS status change events, CloudMonitor provides two methods: function calculation formulas and MNS message queues. This topic describes three best practice cases that use MNS message queues.

Preparations

- Create a message queue.

1. Log on to the [MNS Console](#).

2. On the Queue List page, select the target region, and click Create Queue in the upper-right corner.



New Queue

* Queue Name ? :

* Region : China (Hangzhou)

Long-polling Wait Time (s) ? :

Invisibility Timeout (s) ? :

Maximum Message Size (Byte) ? :

Message Retention Period (s) ? :

Message Delay (s) ? :

Enable Logging :

OK Cancel

3. 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.
 1. Log on to the [CloudMonitor Console](#).
 2. In the left-side navigation pane, click Event Monitoring.
 3. Switch to the Alarm Rules tab page, and then click Create Event Alerts.

Create / Modify Event Alerts**Basic Information**

Alarm Rule Name

Combination of alphabets, numbers and underscores

Event alert

Event Type

☒ System Event ☐ Custom Event

Product Type

ECS

Event Type

StatusNotification ✕

Event Level

All Levels ✕

Event Name

All Events ✕

Resource Range

☒ All Resources ☐ Application Groups**Alarm Type**☒ Alarm Notification

Contact Group

[Delete](#)

GPU监控

Notification Method

Warning (Message+Email ID+Ali WandWai)

[+Add](#)☐ MNS queue☐ Function service ([Best Practises](#))☐ URL callback

OK

Cancel

4. In the Basic Information area, enter a name for the alarm rule, for example, ecs-test-rule.

5. 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 StatusNotification, 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.

6. In the Alarm Type area, select MNS queue, and then specify Region and Queue (for example, ecs-cms-event).

7. 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_secret>'
    region_id = 'cn-beijing'
    queue_name = 'test'
    vserver_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 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 == ' Pending ' :
            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 aliyunsdkcsore.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 aliyunsdkcsore.client import AcsClient
from aliyunsdkcsore.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 = CommonRequ est ()
        self . request . set_method ( ' POST ' )
        self . request . set_accept _format ( ' json ' )
        self . request . set_versio n ( ' 2014 - 05 - 15 ' )
        self . request . set_domain ( ' slb . aliyuncs . com ' )
        self . request . add_query_ param ( ' RegionId ', Conf .
region_id )

    def remove_vse rver_group _backend_s ervers ( self ,
vserver_gr oup_id , instance_i d ):
        self . request . set_action _name ( ' RemoveVSe rverGroupBa
ckendServe rs ' )
        self . request . add_query_ param ( ' VServerGro upId ',
vserver_gr oup_id )
        self . request . add_query_ param ( ' BackendSer vers ',
d + " ',' Port ':' 80 ',' Weight ':' 100 '}]")
        response = self . client . do_action_ with_excep tion (
self . request )
        return str ( response , encoding =' utf - 8 ' )

class ListenerSL B ( BasicListe ner ):
    def __init__ ( self , vserver_gro up_id ):
        self . slb_caller = SLBClient ()
        self . vserver_gro up_id = Conf . vserver_gro up_id

    def process ( self , event ):
        detail = event [ ' content ' ]
        instance_i d = detail [ ' instanceId ' ]
        if detail [ ' action ' ] == ' delete ':
            self . slb_caller . remove_vse rver_group _backend_s
ervers ( self . vserver_gro up_id , instance_i d )

```

**Notice:**

The event name of the preemptible instance release alarm is `Instance:PreemptibleInstanceInterruption`, `mns_client.register_listener(ListenerSLB(Conf.vserver_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.

13 DevOps tutorials

14 DevOps for small and medium web apps

14.1 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 [#unique_142](#).

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.

14.2 Install and configure GitLab

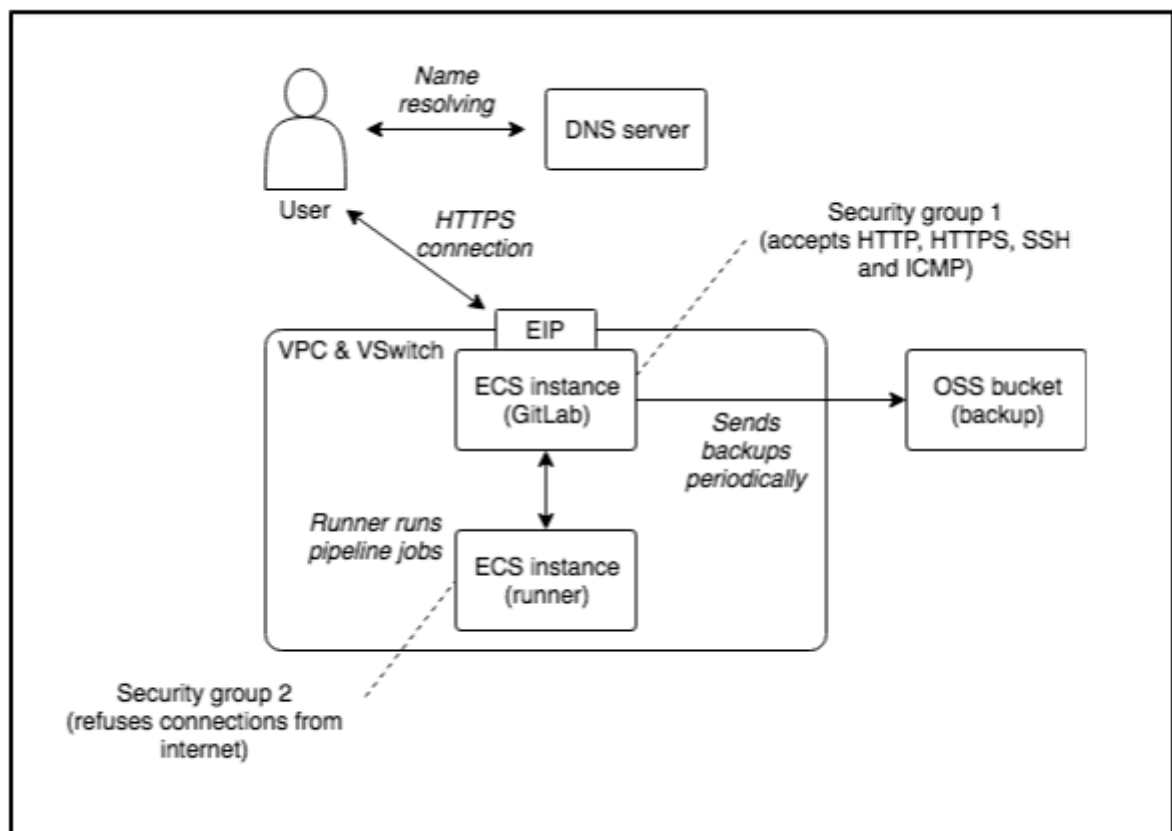
This topic describes how to install and configure GitLab.

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
instance

# 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
instance

# Install dependencies
apt-get install ca-certificates openssh-server
apt-get install postfix # During the installation
, select "Internet Site" and set your domain (for
example, 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 t [' enable ' ] = true
letsencrypt t [' contact_emails ' ] = [ " john . doe @ your -
company . com " ] # Your email address
letsencrypt t [' auto_renew ' ] = true
letsencrypt t [' auto_renew _hour ' ] = 11
letsencrypt t [' auto_renew _minute ' ] = 42
letsencrypt t [' 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:spfmail-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, mxmail-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.

DNS Settings my-sample-domain.xyz

✓ DNS Server:ns7.alidns.com, ns8.alidns.com

For fuzzy search, use keyword:

<input type="checkbox"/>	Type	Host	Line(ISP)	Value	MX Priority	TTL	Status	Actions
<input type="checkbox"/>	MX	mail	Default	mxdm-ap-southeast-1.aliyun.com	10	10 minute(s)	Normal	Edit Disable Delete Remark
<input type="checkbox"/>	MX	mail	Outside mainland China	mxdm-ap-southeast-1.aliyun.com	10	10 minute(s)	Normal	Edit Disable Delete Remark
<input type="checkbox"/>	CNAME	dmtrace.mail	Default	tracedm-ap-southeast-1.aliyuncs.com	--	10 minute(s)	Normal	Edit Disable Delete Remark
<input type="checkbox"/>	CNAME	dmtrace.mail	Outside mainland China	tracedm-ap-southeast-1.aliyuncs.com	--	10 minute(s)	Normal	Edit Disable Delete Remark
<input type="checkbox"/>	TXT	mail	Default	v=spf1 include:spfdom-ap-southeast-1.aliyun.com -all	--	10 minute(s)	Normal	Edit Disable Delete Remark
<input type="checkbox"/>	TXT	mail	Outside mainland China	v=spf1 include:spfdom-ap-southeast-1.aliyun.com -all	--	10 minute(s)	Normal	Edit Disable Delete Remark
<input type="checkbox"/>	TXT	aliyundm.mail	Default	3cdb41a3351449c2af6f	--	10 minute(s)	Normal	Edit Disable Delete Remark
<input type="checkbox"/>	TXT	aliyundm.mail	Outside mainland China	3cdb41a3351449c2af6f	--	10 minute(s)	Normal	Edit Disable Delete Remark
<input type="checkbox"/>	A	gitlab	Default	47.100.100.70	--	10 minute(s)	Normal	Edit Disable Delete Remark
<input type="checkbox"/>	A	gitlab	Outside mainland China	47.100.100.70	--	10 minute(s)	Normal	Edit Disable Delete Remark

☐ Total 10

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
instance

# 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 just
created
```

```
gitlab_rai ls [' gitlab_ema il_display _name '] = ' GitLab '
gitlab_rai ls [' gitlab_ema il_reply_t o '] = ' gitlab @ mail . my
- sample - domain . xyz '
```

Scroll down to **### GitLab email server settings** and insert the following lines:

```
gitlab_rai ls [' smtp_enabl e '] = true
gitlab_rai ls [' smtp_addre ss '] = " smtpdm - ap - southeast - 1
. aliyun . com " # SMTP address written in the Direct
Mail console
gitlab_rai ls [' smtp_port '] = 465
# SMTP port written in the Direct Mail console
gitlab_rai ls [' smtp_user_ name '] = " gitlab @ mail . my - sample
- domain . xyz " # Sender address
gitlab_rai ls [' smtp_passw ord '] = " HangzhouMa il2018 "
# SMTP password for the sender address
gitlab_rai ls [' smtp_domai n '] = " mail . my - sample - domain .
xyz " # Your email domain
gitlab_rai ls [' smtp_auth e ntication '] = " login "
gitlab_rai ls [' smtp_enabl e_starttls _auto '] = false
gitlab_rai ls [' smtp_tls '] = true
```

Apply the configuration change and restart GitLab:

```
gitlab - ctl reconfigur e
```

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 :
rc15yggaCX 08A ***** 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 configurat ion
nano / etc / gitlab / gitlab . rb
```

Scroll to **### Backup Settings** and insert the following line:

```
gitlab_rai ls [' backup_pat h ' ] = "/ mnt / gitlab - bucket /
backup /"
```

Quit and save by pressing CTRL+X, and then check if it works:

```
# Apply GitLab configurat ion
gitlab - ctl reconfigur e

# 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 /$(/ bin / date '+\% s_ \% Y_ \% m_ \% d ' )
_gitlab . rb "
0 2 * * * / bin / cp / etc / gitlab / gitlab - secrets . json "/
mnt / gitlab - bucket / backup /$(/ 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 bash

# 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 -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

# 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:

Setup a shared Runner manually

1. [Install GitLab Runner](#)
2. Specify the following URL during the Runner setup: `https://gitlab.my-sample-domain.xyz/`
3. Use the following registration token during setup: `gXppo8ZyDgqdFb1vPG-w`
4. Start the Runner!

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.

Runner description or token		Search		Runners currently online: 1				
Type	Runner token	Description	Version	IP Address	Projects	Jobs	Tags	Last contact
<div>shared</div> <div>locked</div>	d12e17e8	devops-simple-app-security-group-runner	11.4.0	47.172	n/a	0		11 minutes ago

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 / 50unattend ed - upgrades

# Enable automatic upgrades
dpkg - reconfigur e -- 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 - Upgradeabl e - Packages " 1 ";
APT :: Periodic :: AutocleanI nterval " 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.

14.3 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:

SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

Add an SSH key

To add an SSH key you need to [generate one](#) or use an [existing key](#).

Key

Paste your public SSH key, which is usually contained in the file `~/.ssh/id_rsa.pub` and begins with `'ssh-rsa'`. Don't use your private SSH key.

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDAQDV/G9tSniNCwhxGrxa8XrM08HctdYkAgSEprQt+oZ
UH464RHo6BIDWA/7+K69mKfe/L0dWRsiQY+RSVftnYDZtbnj0sfR1L99xpXTTup1F+OyYnVi4pb
MF3Uf6ccvewmKK7dHbOMQlRn6HZqEAsgROfrf8ttLfd3CPRmGwQ9hvlZbxmjj57HiLkTEcrpKL
sZ7OcDsi9jT45r4OBG2a7IUXJUrhsl50oGE6r8h+B4Q4GuY9dZQ7oGqhmMM5SPiiv8t8purwC
8Yb9CXKlIn+aCnNM80S5NC1wWNj3GKTaZkKJCHGgTcNEI6V7tLX7mLDMcUA4+xNij8/AN
YLIUQtr john.doe@your-company.com
```

Title

Name your individual key via a title

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 / todoist` . 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 / todoist / 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_profi le

# Reload . bash_profi le
. ~/. bash_profi le
```

3. Launch MySQL on your computer and connect to it with your terminal:

```
# Connect to the database ( use the password you
received during the installati on )
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 '
YouNewRoot Password ';

-- 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`.

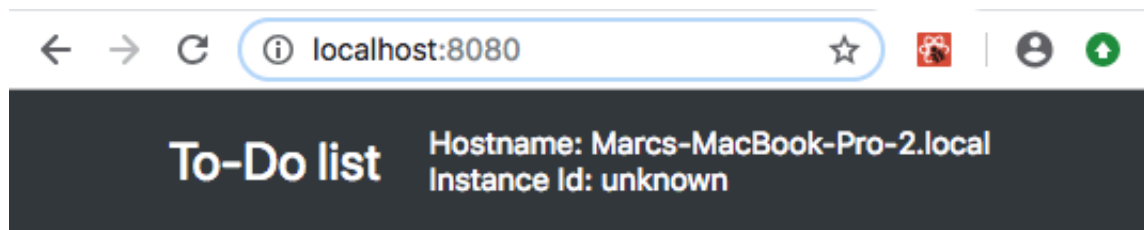
You can now launch the application locally with the following command:

```
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:



Tasks

Add a new task

Description:

Existing tasks

Description	
Write a new tutorial	<input type="button" value="Delete"/>
Buy a battery for the car	<input type="button" value="Delete"/>



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_sch ema_histor y |
| 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_sch ema_histor y`:

```
-- Look at the content of the flyway_sch ema_histor y
table
SELECT * FROM flyway_sch ema_histor y ;
```

The result should look like this:

```
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| installed_ rank | version | descriptio n | type |
| script | checksum | installed_ by |
| installed_ on | execution_ time | success |
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
| 1 | 001 | Create task table | SQL |
| V001__Crea te_task_ta ble . 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 / to do list

# 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 ."

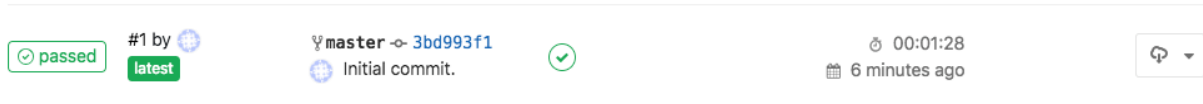
# Push the commit to the GitLab server
git push origin master
```

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 todomlist 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.

14.4 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
instance

# 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 - gs5wm687b2 e3uc770 . pgsql . singapore .
rds . aliyuncs . com : 3433 / postgres - U sonarqube

```

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=YourSonarP@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 reboots
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:

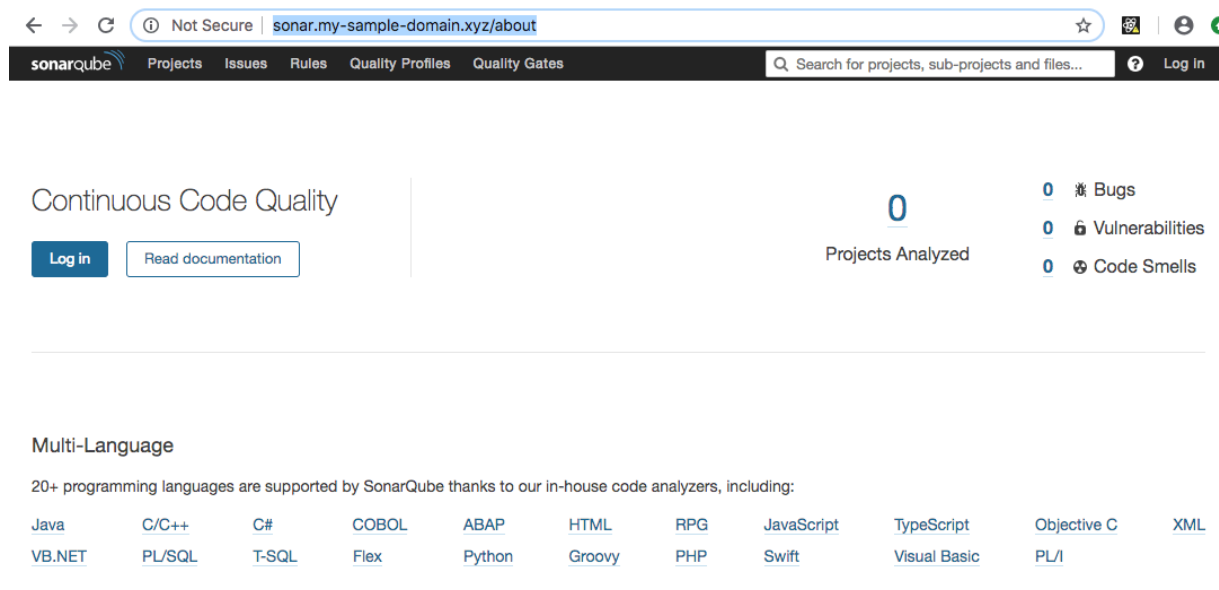
```
# Enable the new configuration file
ln -s /etc/nginx/sites-available/sonarqube /etc/nginx/sites-enabled/sonarqube

# 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 redirectio n

# The certificat e will be automatica lly renewed . If
you want , you can check the Cron configurat ion :
nano / etc / cron . d / certbot

# Check the renewal process with the following command
certbot renew -- dry - run
# The logs should contain " Congratula tions , all
renewals succeeded " with your domain name ( for example
, sonar . my - sample - domain . xyz )

# Restart Nginx
systemctl restart nginx

# Configure Nginx to automatica lly 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:

Conditions

Only project measures are checked against thresholds. Sub-projects, directories and files are ignored. [More](#)

Metric	Over Leak Period	Operator	Warning	Error		
Blocker Issues	<input type="checkbox"/>	is not		0	Update	Delete
Coverage	<input type="checkbox"/>	is less than		70	Update	Delete
Coverage on New Code	Always	is less than		80	Update	Delete
Critical Issues	<input type="checkbox"/>	is not		0	Update	Delete
Duplicated Lines on New Code (%)	Always	is greater than		3	Update	Delete
Maintainability Rating on New Code	Always	is worse than		A ×	Update	Delete
Major Issues	<input type="checkbox"/>	is not		0	Update	Delete
Reliability Rating on New Code	Always	is worse than		A ×	Update	Delete
Security Rating on New Code	Always	is worse than		A ×	Update	Delete
Unit Test Errors	<input type="checkbox"/>	is not		0	Update	Delete
Unit Test Failures	<input type="checkbox"/>	is not		0	Update	Delete

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 >
        < configurat ion >
          < append > true </ append >
        </ configurat ion >
        < 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 . sonarsourc e . 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 = cfe2e3d7d7 a15df20e3e cb7de53b6a 23b3757474
  \
  - 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_sonar_qube . url "
  SONAR_LOGI N : " token_gene rated_by_s onarqube "

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_LOGI N - Dsonar . 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 - gate -
breaker - all - 1 . 0 . 1 . jar "
- " java - jar sonar - gate - breaker - all - 1 . 0 . 1 . jar
- u $ SONAR_LOGI N "
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

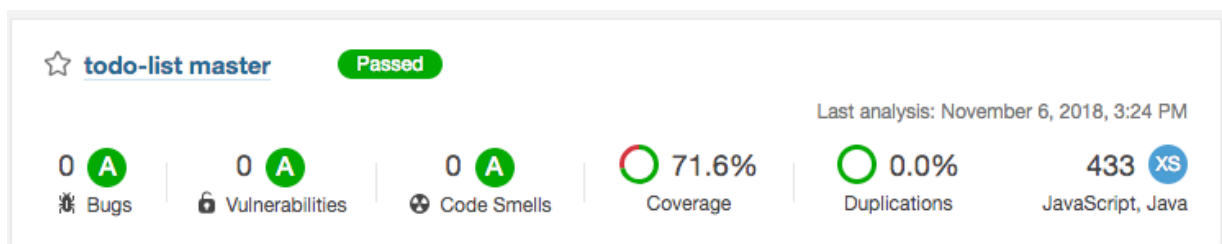
# Check the files to commit
git status

# 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 ."

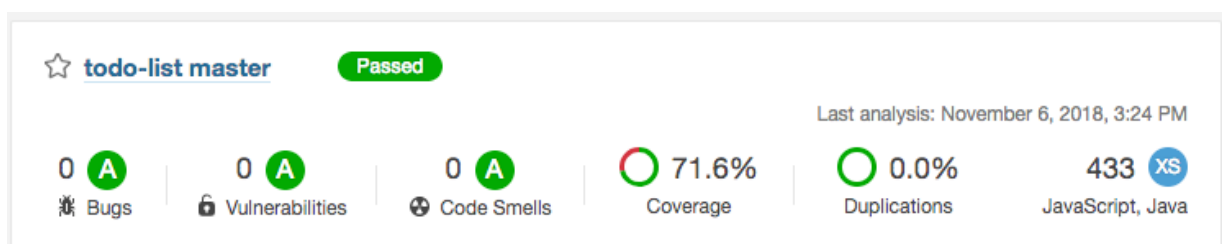
# Push the commit to the GitLab server
git push origin master
```

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:

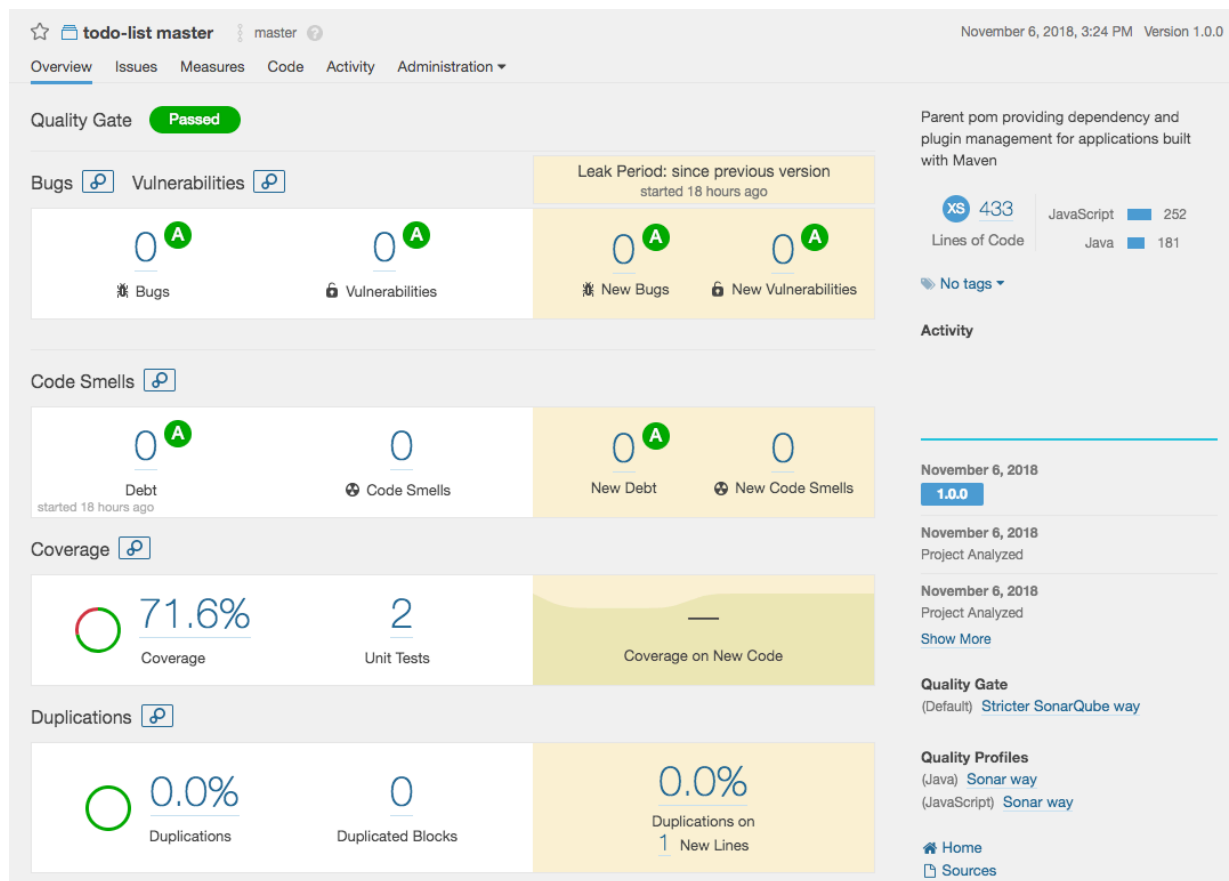


As you can see, there is now two stages in the pipeline. You can click them to check detailed logs.

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 /
controller s / TaskContro llerTest . java
```

At the line 87 of this file, change:

```
assertEqual (" Task 2 ", createdTask2 . getDescription ());
```

Into:

```
assertEqual (" Task 2222222222 ", createdTask2 . getDescription ());
```

Save by pressing CTRL + X, then commit the change:

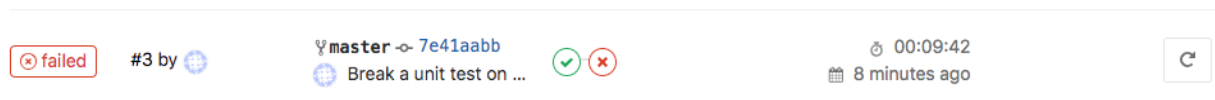
```
# Check the files to commit
git status

# Add the file
git add src / test / java / com / alibaba / intl / todolist /
controller s / TaskContro llerTest . java

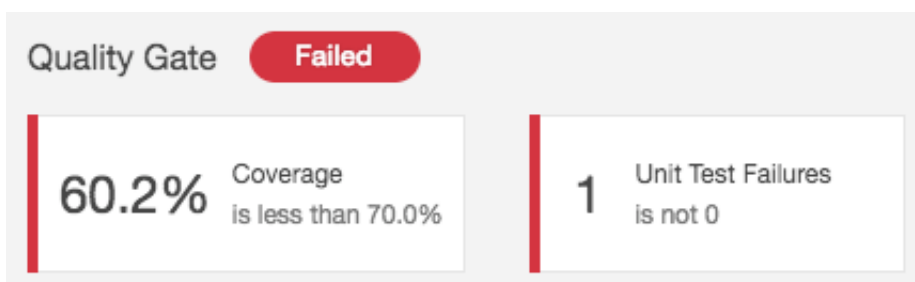
# Commit the file and write a comment
git commit -m "Break a unit test on purpose ."

# 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 /
controller s / TaskContro llerTest . 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 /
controller s / TaskContro llerTest . 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 /
controller s / MachineCon troller . java
```

Insert the following lines at the end of the class (line 71, before the last brace):

```
private String dummy = " example ";

    public synchroniz ed 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 /
controller s / MachineCon troller . 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:

Quality Gate **Failed**

Some Quality Gate conditions on New Code were ignored because of the small number of New Lines ?

 Reliability Rating on New Code is worse than A	69.6% Coverage is less than 70.0%	1 Major Issues is not 0
--	--	--------------------------------

```

71 marc...
72 marc... private String dummy = "example";
73 marc...
74 marc... public synchronized String getDummy() {
75 marc...     return dummy;
76 marc... }
77 marc...
78 marc... public void setDummy(String dummy) {
79 marc...     this.dummy = dummy;
80 marc... }
81 marc... }

```

Synchronize this method to match the synchronization on "getDummy". 2 hours ago L78

Bug Major Open Marc Plouhinec 5min effort Comment cert, multi-threading

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 /
controller s / MachineCon troller . 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 /
controller s / MachineCon troller . 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.

14.5 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.

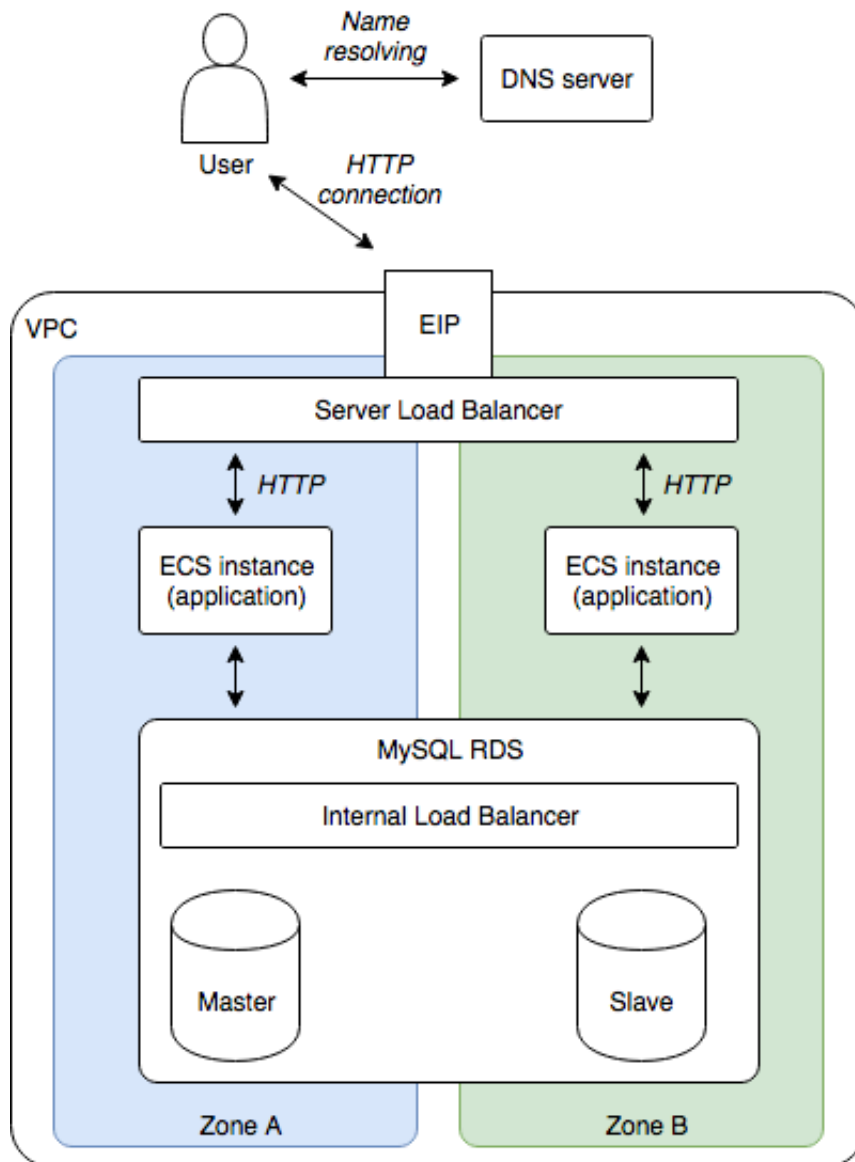
The next two sections introduces [Terraform](#) and [Packer](#): as you can see in the previous parts of this tutorial, creating our environment for GitLab and SonarQube with the [web console](#) is quite slow. Since we will have to create 3 nearly identical environments, we will use Terraform and Packer 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 independent, 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:

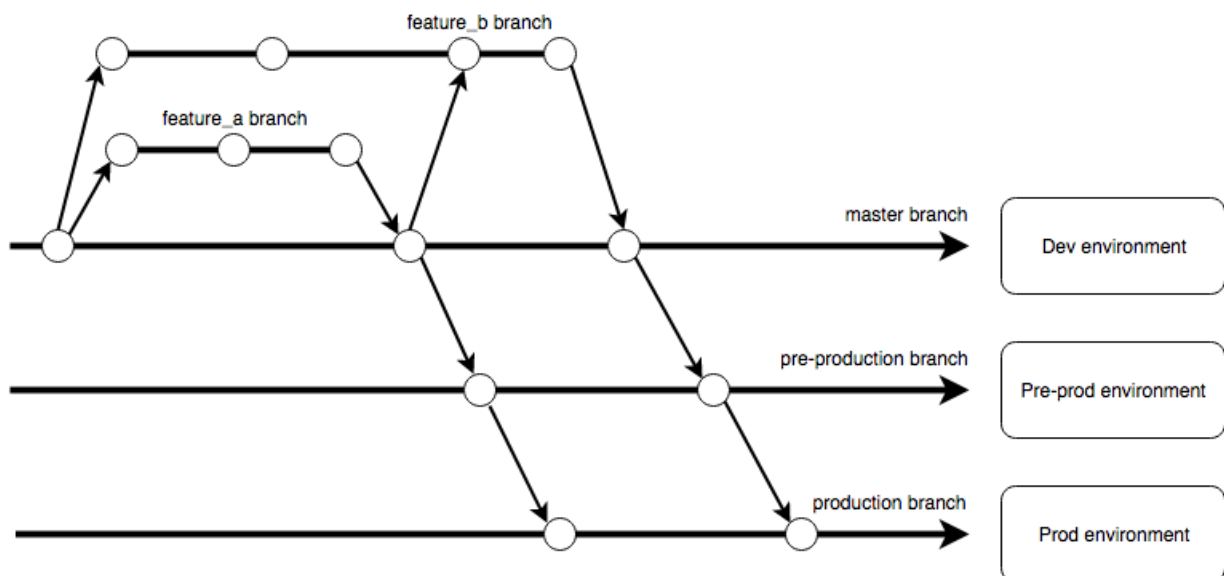
With this diagram you can understand why a stateless application is advantageous: the only place where data is shared is the database, we do not need to establish a direct link between the application servers. Moreover, if two users are modifying

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_v pc " " 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_A CCESS_KEY =" your - accesskey - id "
export ALICLOUD_S ECRET_KEY =" your - accesskey - secret "
export ALICLOUD_R EGION =" 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 [#unique_147](#). For `ALICLOUD_REGION`, the available values can be found in [#unique_41](#).

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-t4nhi7y0wp_zkfr2auxc0_p)

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:

Instance ID/Name	Destination CIDR Block	Status	Default VPC	Route Table	VSwitch	Resource Group	Actions
vpc-t4nhi7y0wpzkfr2auxc0p sample-vpc	192.168.0.0/16	● Available	No	1	0	rg-acfnypgj-cnsejva	Manage Delete

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:

```
alicloud_vpc.sample_vpc : Refreshing state ... ( ID : vpc -
t4nhi7y0wp_zkfr2auxc0_p )
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-t4nhi7y0wp_zkfr2auxc0_p"

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 - t4nhi7y0wp_zkfr2auxc0_p "
alicloud_vswitch.sample_vswitch : Creation complete after
7s ( ID : vsw - t4nvtqld0k_tk4kddxq70_9 )

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:

Instance ID/Name	VPC	Status	Destination CIDR Block	Default VSwitch	Zone	Number of Available Private IPs	Route Table	Route Table Type	Actions
vsw-t4nvtqld0ktk4kddxq709 sample-vswitch	vpc-t4nhi7y0wpzkfr2auxc0p sample-vpc	● Available	192.168.1.0/24	No	Singapore Zone A	252	vtb-t4ncu7v9w7igzwo dfoqv	System	Manage Delete Purchase ✓

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 -
t4nhi7y0wp_zkfr2auxc0_p )
data.alicloud_zones.az : Refreshing state ...
alicloud_vswitch.sample_vswitch : Refreshing state ... ( ID
: vsw - t4nvtqld0k_tk4kddxq70_9 )

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 infrastruc
  ture , 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_v switch . sample_vsw itch : Destroying ... ( ID : vsw -
t4nvtqld0k tk4kddxq70 9 )
alicloud_v switch . sample_vsw itch : Destructio n complete
after 1s
alicloud_v pc . sample_vpc : Destroying ... ( ID : vpc -
t4nhi7y0wp zkfr2auxc0 p )
alicloud_v pc . sample_vpc : Destructio n 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_v switch . sample_vsw itch : Destroying ... ( ID : vsw -
t4nvtqlld0k tk4kddxq70 9 )
alicloud_v switch . sample_vsw itch : Destructio n complete
after 1s
alicloud_v pc . sample_vpc : Destroying ... ( ID : vpc -
t4nhi7y0wp zkfr2auxc0 p )
alicloud_v pc . sample_vpc : Destructio n complete after 3s

Destroy complete ! Resources : 2 destroyed .
```

Copy the following content into your script:

```
{
  " variables ": {
    " access_key ": "{{ env ` ALICLOUD_A CCESS_KEY ` }}",
    " secret_key ": "{{ env ` ALICLOUD_S ECRET_KEY ` }}",
    " region_id ": "{{ env ` ALICLOUD_R EGION ` }}",
    " source_ima ge ": "{{ env ` SOURCE_IMA GE ` }}",
    " instance_t ype ": "{{ env ` INSTANCE_T YPE ` }}"
  },
  " builders ": [
    {
      " type ": " alicloud - ecs ",
      " access_key ": "{{ user ` access_key ` }}",
      " secret_key ": "{{ user ` secret_key ` }}",
      " region ": "{{ user ` region_id ` }}",
      " image_name ": " sample - image ",
      " image_desc ription ": " Sample image for testing
Packer .",
      " image_vers ion ": " 1 . 0 ",
      " source_ima ge ": "{{ user ` source_ima ge ` }}",
      " ssh_userna me ": " root ",
      " instance_t ype ": "{{ user ` instance_t ype ` }}",
      " io_optimiz ed ": " true ",
      " internet_c harge_type ": " PayByTraff ic ",
      " image_forc e_delete ": " true ",
      " system_dis k_mapping ": {
        " disk_categ ory ": " cloud_ssd ",
        " disk_size ": 20
      }
    }
  ],
  " provisione rs ": [
    {
      " type ": " shell ",
      " inline ": [
        " export DEBIAN_FRO NTEND = noninterac tive ",
        " 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_befo re ": " 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_5be
a5aa2 - e524 - 1af8 - 80d1 - 1db78347ed 15
==> 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_5be a5aa2 - e524
- 1af8 - 80d1 - 1db78347ed 15 to instance : i - t4nhcv8qx0
69trkfgye6
==> alicloud - ecs : Starting instance : i - t4nhcv8qx0
69trkfgye6
==> 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 / jvjz3zmn64 q0j34yc9m9 n4w00000gn / T / packer -
shell04740 4213
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 Dload
alicloud - ecs :
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 successful ly installed and
alicloud - ecs : working . Further configurat ion is
required .</ p >
alicloud - ecs :
alicloud - ecs : < p > For online documentat ion 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 - t4nhcv8qx0
69trkfgye6
==> alicloud - ecs : Waiting instance stopped : i - t4nhcv8qx0
69trkfgye6
==> alicloud - ecs : Creating image : sample - image
alicloud - ecs : Detach keypair packer_5be a5aa2 - e524
- 1af8 - 80d1 - 1db78347ed 15 from instance : i - t4nhcv8qx0
69trkfgye6
==> 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 - t4n938t1pl plyl7akeor

```

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. [#unique_148](#) 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:

<input type="checkbox"/>	ID/Name	Tags	Type	Platform	System Bit	Created At	Status	Progress	Actions
<input type="checkbox"/>	m-14n938t1plplyl7akeor sample-image		Custom Images	Ubuntu	64Bit	November 13, 2018, 13:06	Available	100%	Create Instance Modify Description Related Instances Copy Image Share Image Export 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 /
controller s / HealthCont roller . java
```

Copy the following content into the new file:

```
package com . alibaba . intl . todolist . controller s ;

import org . springfram ework . web . bind . annotation .
RequestMap ping ;
import org . springfram ework . web . bind . annotation .
RestContro ller ;
```

```
/**
 * 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 /
controller s / HealthControllerTest . java
```

Copy the following content into the new file:

```
package com . alibaba . intl . todolist . controller s ;

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 . WebApplication
Context ;

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 (). getContent AsString
        ();
        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 /
controller s / HealthCont roller . java
git add src / test / java / com / alibaba / intl / todolist /
controller s / HealthCont rollerTest . 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 / to do list

# Copy the scripts from this tutorial ( adapt the
path to where you copied this tutorial )
cp - R path / to / sample - app / version3 / infrastruc ture .

# Check the content of this folder
ls - l infrastruc ture
ls - l infrastruc ture / 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_zone0.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 [#unique_149](#).
- 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 @ tabasePass w0rd '

# 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 " alibabacloud_images " " ubuntu_images " {
  owners = " system "
  name_regex = " ubuntu_18 [ a - zA - Z0 - 9_ ]+ 64 "
  most_recent = true
}

data " alibabacloud_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"
    }
  ]
}

```

[illegible]

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/todo-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="YourDatabasePassword"
packer build app_image.json
```

You can check the newly created image using the console:

1. Select Images from the left-side navigation pane.
2. If necessary, select your region on the top of the page.
3. 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).

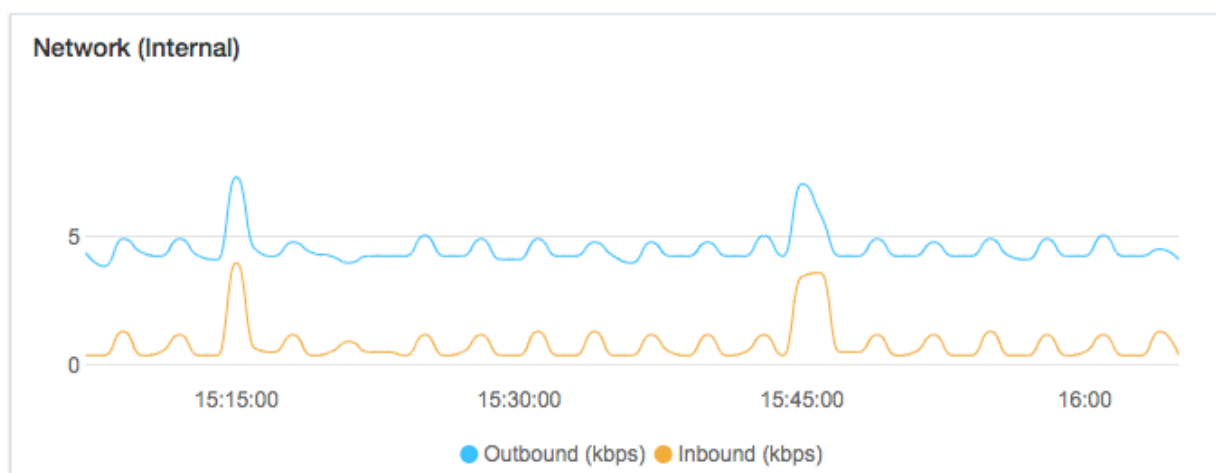
Let's check the deployment of our application:

1. Select your region if necessary.

Your new SLB looks like this:

<input type="checkbox"/>	Instance Name/ID	IP Address [?]	Status [?]	Monitoring	Port/Health Check/Backend Server [?]	Actions
<input type="checkbox"/>	sample-app-slb-dev lb-gs5cu183xqhcvm0r8etns The tag is not set.	192.168.0.230(VPC) 161.117.2.245(Elastic IP Address) vpc-t4np92sn4wqnzeom31juz vsw-t4n855hfk5szg22rha260	Active		HTTP: 80 ● Normal Default Server Group 2	Configure Listener Add Backend Servers More [?]

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:



Tasks

Add a new task

Description: Add

Existing tasks

Description	
Backup photos from smartphone	<button>Delete</button>
Buy ink cartridges	<button>Delete</button>

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:

Hostname: sample-app-ecs-zone-0-dev
Instance Id: i-t4n62r2d07oe0x4t7nte

Hostname: sample-app-ecs-zone-1-dev
Instance Id: i-t4nicuzkuqxqtj5b9wd7

As you can see, your HTTP requests are distributed among your two ECS instances.



Note:

If you wish, you can enable [#unique_150](#) 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 / infrastruc ture / 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_domai n
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. Click Instance from the left-side navigation pane.
2. Select your region if necessary.
3. Search for your instance named devops-simple-app-gitlab-runner.
4. Click Connect on the right side of your instance.
5. The VNC console should appear: copy the VNC password displayed in the popup and paste it to the next one.
6. 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. The configuration file should look like this:

```
concurrent = 1
check_intel_rapl = 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]
```

9. Change `privileged = false` to `privileged = true`. The file should now look like this:

```
concurrent = 1
check_intel_rapl = 0

[[ session_server ]]
  session_timeout = 1800

[[ runners ]]
```

```

name = " devops - simple - app - gitlab - runner "
url = " https :// gitlab . my - sample - domain . xyz /"
token = " 8943412dd8 5a41002f9f 803f21bdbf "
executor = " docker "
[ runners . docker ]
  tls_verify = false
  image = " alpine : latest "
  privileged = true
  disable_en trypoint_o verwrite = false
  oom_kill_d isable = false
  disable_ca che = false
  volumes = ["/ cache "]
  shm_size = 0
[ runners . cache ]
[ runners . cache . s3 ]
[ runners . cache . gcs ]

```

10. Save and quit by pressing CTRL + X.

11. Restart the GitLab Runner using the following command:

```
gitlab - runner restart
```

12. 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_A CCESS_KEY : " your - accesskey - id "
  ALICLOUD_S ECRET_KEY : " your - accesskey - secret "
  ALICLOUD_R EGION : " your - region - id "
  GITLAB_BUC KET_NAME : " gitlab - my - sample - domain - xyz "
  GITLAB_BUC KET_ENDPOI NT : " http :// oss - ap - southeast - 1 -
internal . aliyuncs . com "
  ECS_ENDPOI NT : " ecs . aliyuncs . com "
  DOMAIN_NAM E : " my - sample - domain . xyz "
  DB_ACCOUNT _PASSWORD : " your - db - password "
  ECS_ROOT_P ASSWORD : " your - ecs - root - password "
  OSSFS_VERS ION : " 1 . 80 . 5 "
  TERRAFORM_ VERSION : " 0 . 11 . 11 "
  PACKER_VER SION : " 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_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 / mount_ossfs . sh "
- "./ gitlab - ci - scripts / deploy / build_basiss_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_PREPRODUCTION="pre - prod"
ENV_NAME_PRODUCTION="prod"

if [[ ${BRANCH_NAME} == "production" ]]; then
    echo ${ENV_NAME_PRODUCTION};
elif [[ ${BRANCH_NAME} == "pre - production" ]]; then
    echo ${ENV_NAME_PREPRODUCTION};
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_PREPRODUCTION="pre-prod"
ENV_NAME_PRODUCTION="prod"

if [[ ${BRANCH_NAME} == "production" ]]; then
    echo ${ENV_NAME_PRODUCTION};
elif [[ ${BRANCH_NAME} == "pre-production" ]]; then
    echo ${ENV_NAME_PREPRODUCTION};
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 Packer
version ${PACKER_VERSION}..."

# Create a temporary folder
mkdir -p installation_tmp
cd installation_tmp

# Install OSSFS
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 installati on_tmp

echo "Installati on 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}) into ${BUCKET_LOCAL_PATH}..."

# Configure OSSFS
echo "$GITLAB_BUCKET_NAME:$ALICLOUD_ACCESS_KEY:$ALICLOUD_SECRET_KEY" > /etc/passwd - ossfs
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_A CCESS_KEY
# - ALICLOUD_S ECRET_KEY
# - ALICLOUD_R EGION
# - ENV_NAME
# - DOMAIN_NAME
# - SUB_DOMAIN _NAME
# - BUCKET_LOC AL_PATH
#

echo " Building the basis infrastructure ( environmen t
: ${ ENV_NAME },\
  region : ${ ALICLOUD_R EGION },\
  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_LOC AL_PATH / infrastruc
ture /$ ENV_NAME / 05_vpc_slb _eip_domain "
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 successful ly built (
environmen t : ${ ENV_NAME }, region : ${ ALICLOUD_R EGION })."
```

**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_A CCESS_KEY
```

```

# - ALICLOUD_S ECRET_KEY
# - ALICLOUD_R EGION
# - ENV_NAME
# - DB_ACCOUNT _PASSWORD
# - ECS_ROOT_P ASSWORD
# - BUCKET_LOC AL_PATH
# - CI_PIPELIN E_IID
#

echo " Building the applicatio n infrastruc ture (
environmen t : ${ ENV_NAME }, region : ${ ALICLOUD_R EGION
})..."

# Set values for Terraform and Packer variables
export TF_VAR_env=${ ENV_NAME }
export TF_VAR_db_account_pa ssword=${ DB_ACCOUNT _PASSWORD }
export TF_VAR_ecs _root_pass word=${ ECS_ROOT_P ASSWORD }

export APPLICATIO N_PATH=$( pwd )/$( ls target/*.jar )
export PROPERTIES _PATH=$( pwd )/ src / main / resources /
applicatio n . properties
export IMAGE_VERS ION=${ CI_PIPELIN E_IID }
export ENVIRONMEN T=${ ENV_NAME }
export RDS_DATABA SE = todolist
export RDS_ACCOUN T = todolist
export RDS_PASSWO RD=${ DB_ACCOUNT _PASSWORD }

# Create / update the applicatio n database
cd infrastruc ture / 10_webapp / 05_rds
export BUCKET_DIR _PATH="$ BUCKET_LOC AL_PATH / infrastruc
ture /$ 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 * ${ BUCKET_DIR _PATH }
export RDS_CONNEC TION_STRIN G=$( terraform output
app_rds_co nnection_s tring )

# Extract Alibaba Cloud informatio n for building the
applicatio n image
cd ../ 10_image
export BUCKET_DIR _PATH="$ BUCKET_LOC AL_PATH / infrastruc
ture /$ 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 * ${ BUCKET_DIR _PATH }
export SOURCE_IMA GE=$( terraform output image_id )
export INSTANCE_T YPE=$( terraform output instance_t ype )

# Build the applicatio n image
packer build app_image . json

# Create / update the ECS instances
cd ../ 15_ecs
export BUCKET_DIR _PATH="$ BUCKET_LOC AL_PATH / infrastruc
ture /$ 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_ID` 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 todoist 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 [#unique_151](#). 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 todoist 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 infrastruc ture / 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 left-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:

Marc Plouhinec > [todolist](#) > Merge Requests > [New](#)

New Merge Request






Source branch	Target branch
<div>marcplouhinec/todolist ▼</div> <div>master ▼</div> <div> Minor fixes. Marc Plouhinec authored 1 hour ago ✓ 5316cac7 </div>	
<div>marcplouhinec/todolist ▼</div> <div>pre-production ▼</div> <div> Merge branch 'master' into 'pre-production' ... ✓ af740016 </div>	

Compare branches and continue

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:

Marc Plouhinec > [todolist](#) > Merge Requests > [New](#)

New Merge Request

Source branch	Target branch
<div>marcplouhinec/todolist ▼</div> <div>pre-production ▼</div> <div> Merge branch 'master' into 'pre-production' ...  573c3045 </div>	
<div>marcplouhinec/todolist ▼</div> <div>production ▼</div> <div> Improve the delivery stage. Marc Plouhinec authored 1 day ago ✓ 4f47d88c </div>	

Compare branches and continue

6. Let another person to check and accept the merge request to start the deployment into production.

14.6 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 the certificate](#). 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

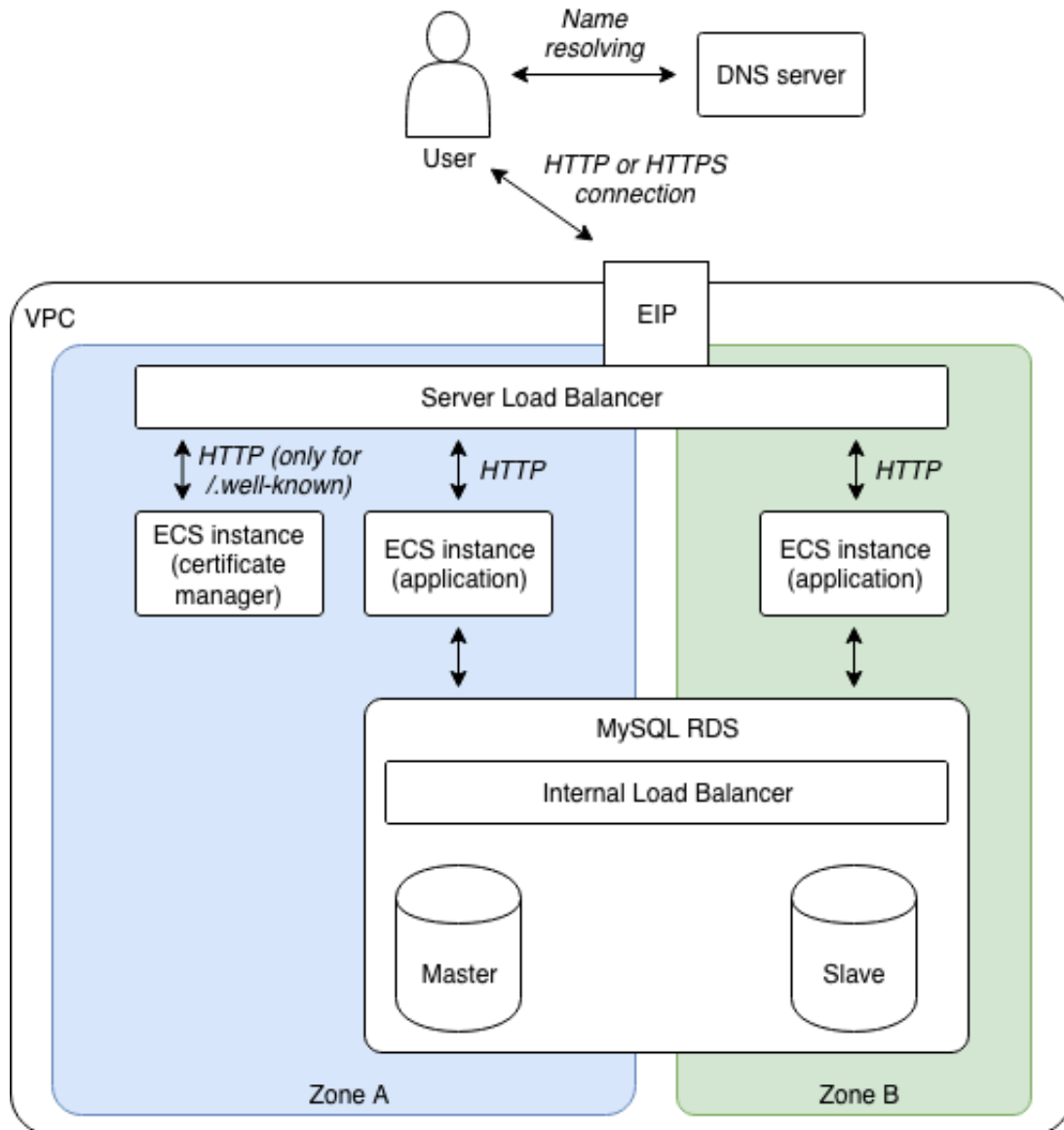
HTTPS works by encrypting HTTP traffic through the [TLS protocol](#). To configure HTTPS, we first need to obtain an [SSL/TLS certificate](#) and configure it on our SLB (by adding [an HTTPS listener](#)).

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 can 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:

- Certbot creates a file with a unique name in the folder `/var/www/html/certman/.well-known/acme-challenge/`, so that Nginx can serve this file when the URL path is `/well-known/acme-challenge/unique-name`.
- Certbot contacts a Let's Encrypt server and asks it to make an HTTP request to this file with the URL `http://dev.my-sample-domain.xyz/.well-known/acme-challenge/unique-name`.
- 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_http" "app_slb_listener_http" {
  // ...
}
resource "alicloud_slb_listener_https" "app_slb_listener_https" {
  load_balancer_id = "${alicloud_slb.id}"

  backend_port = 8080
  frontend_port = 443
  bandwidth = -1
  protocol = "https"
  ssl_certificate_id = "${alicloud_slb_server_certificate.id}"
  app_slb_certificate_id = "${alicloud_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  infrastruc ture / 15_certman / 05_image
cd  infrastruc ture / 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:

[illegible]

```
}
  ]
}
```

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 `sed` in 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 certmanager . 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_bot_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 / certificat e - updater ",
    " pip install pipenv -- upgrade ",
    " pipenv install aliyun - python - sdk - core ==${
ALIYUN_PYT_HON_SDK_CO_RE_VERSION }",
    " pipenv install aliyun - python - sdk - slb ==${
ALIYUN_PYT_HON_SDK_SLB_VERSION }",
    " pipenv install pyopenssl ",
    " pipenv install pytz ",
    " export ESCAPED_AC_CESS_KEY=$( echo $ ALICLOUD_A
CESS_KEY | sed - e ' s /\ \ \ \ \ \ \ \ \ \ / g ; s /\ \ \ \ \ \ \ \ \ \ / g ;
s /\& /\ \ \ \ \ \ \ \ \ \ & / g ' )",
    " export ESCAPED_SE_CRET_KEY=$( echo $ ALICLOUD_S
ECRET_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 $ ENVIRONMEN T |
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_ADDR
ESS | sed - e ' s /\ \ \ \ \ \ \ \ \ \ / g ; s /\ \ \ \ \ \ \ \ \ \ / g ; s /\& /\ \ \ \ \ \ \ \ \ \ & /
\ \ \ \ \ \ \ \ \ \ & / g ' )",
    " sed - i \" s /% access - key - id %/${ ESCAPED_AC
CESS_KEY }/\ " / etc / certificat e - updater / config . ini ",
    " sed - i \" s /% access - key - secret %/${ ESCAPED_SE
CRET_KEY }/\ " / etc / certificat e - updater / config . ini ",
    " sed - i \" s /% region - id %/${ ESCAPED_REGION }/\ " /
etc / certificat e - updater / config . ini ",
    " sed - i \" s /% environmen t %/${ ESCAPED_ENVIRONMENT
} /\ " / etc / certificat e - updater / config . ini ",
    " sed - i \" s /% domain %/${ ESCAPED_DOMAIN }/\ " / etc /
certificat e - updater / config . ini ",
    " sed - i \" s /% sub - domain %/${ ESCAPED_SUB_DOMAIN }/
\" / etc / certificat e - updater / config . ini ",
    " sed - i \" s /% email - address %/${ ESCAPED_EMAIL
ADDRESS } /\ " / etc / certificat e - updater / config . ini ",
    " chmod + x / opt / certificat e - updater / certificat e
- updater . sh ",
    " systemctl enable certificat e - 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 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 publicKeyE exists :
    publicKey = open ( publicKeyP ath , " rt "). read ()
    x509 = OpenSSL . crypto . load_certificate ( OpenSSL . crypto
. FILETYPE_P EM , publicKey )
    expiration Date = datetime . strptime ( x509 . get_notAfter
(), "% Y % m % d % H % M % SZ "). replace ( tzinfo = pytz . UTC )
    now = datetime . now ( pytz . utc )
    certExpire Soon = now + timedelta ( weeks = 1 ) >
expiration Date

runCertBot = not publicKeyE exists or not privateKey
Exists or certExpire Soon

certbotCronConfigure d = not os . path . exists ("/ etc / cron
. d / certbot ")
print (" Let ' s Encrypt certificat e status :")
print ("     publicKeyP ath = % s " % publicKeyP ath )
print ("     privateKey Path = % s " % privateKey Path )
print ("     publicKeyE exists = % s " % publicKeyE exists
)
print ("     privateKey Exists = % s " % privateKey Exists
)
print ("     certExpire Soon = % s " % certExpire Soon )
print ("     certbotCronConfigure d = % s " % certbotCronConfigure d )
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
, emailAddre ss ), shell = True )
    if returnCode != 0 :
        print (" Unable to run certbot , quitting ...")
        quit ( 1 )

    print (" Replace the certificat e on the OSS bucket
...")
    if not os . path . exists ( certFolder Path ):
        os . makedirs ( certFolder Path )
    for f in glob . glob ( certFolder Path + "/*"):
        os . remove ( f )
    for f in glob . glob ( certbotCer tFolderPat h + "/*"):
        shutil . copy2 ( f , certFolder Path + "/" )

# Check if the SLB certificat e needs to be updated
print (" Getting informatio n about the SLB sample - app -
slb -" + environmen t + "...")
client = AcsClient ( accessKeyId , accessKeyS ecret , regionId
)
request = CommonRequ est ()
request . set_accept _format (" json ")
request . set_domain (" slb . aliyuncs . com ")
request . set_method (" POST ")
request . set_versio n (" 2014 - 05 - 15 ")
request . set_action _name (" DescribeLoadBalancer s ")
request . add_query_ param (" LoadBalanc erName ", " sample - app -
slb -" + environmen t )
request . add_query_ param (" RegionId ", regionId )
jsonRespon se = client . do_action_ with_excep tion ( 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
informatio n ..." % 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 informatio n found . Loading
informatio n 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 informatio n :")
print (" slbCertFingerprint = % s " % slbCertFingerprint )

```

```

print ("      certFinger print      = % s " % certFinger print )
print ("      updateList enerCert = % s " % updateList enerCert )

if not updateList enerCert :
    print (" SLB listener certificat e is up to date .")
    quit ( 0 )

# Upload the SLB listener certificat e
now = datetime . now ()
certName = " sample - app - slb - certificat e -" + environmen t
+ "-" + now . strftime ("% Y % m % d % H % M % S ")
print (" Upload the Let ' s Encrypt certificat e " +
certName + "...")
request = CommonRequ est ()
request . set_accept _format (" json ")
request . set_domain (" slb . aliyuncs . com ")
request . set_method (" POST ")
request . set_versio n (" 2014 - 05 - 15 ")
request . set_action _name (" UploadServ erCertific ate ")
privateKey = open ( privateKey Path , " rt "). read ()
privateKey = privateKey . replace (" BEGIN PRIVATE ", " BEGIN
RSA PRIVATE ")
privateKey = privateKey . replace (" END PRIVATE ", " END RSA
PRIVATE ")
request . add_query_ param (" ServerCert ificate ", publicKey )
request . add_query_ param (" PrivateKey ", privateKey )
request . add_query_ param (" ServerCert ificateNam e ", certName
)
jsonRespon se = client . do_action_ with_excep tion ( request )
response = json . loads ( jsonRespon se )
if not response [" ServerCert ificateId "]:
    print (" Unable to upload the certificat e " +
certName + ". Response :")
    print ( response )
    quit ( 1 )
certId = response [" ServerCert ificateId "]

# Update the HTTPS listener with the new certificat e
print (" Certificat e " + certName + " ( id : " + certId + ")
uploaded with success . Updating the HTTP listener ...")
request = CommonRequ est ()
request . set_accept _format (" json ")
request . set_domain (" slb . aliyuncs . com ")
request . set_method (" POST ")
request . set_versio n (" 2014 - 05 - 15 ")
request . set_action _name (" SetLoadBal ancerHTTPS ListenerAt
tribute ")
request . add_query_ param (" ListenerPo rt ", " 443 ")
request . add_query_ param (" LoadBalanc erId ", slbId )
request . add_query_ param (" ServerCert ificateId ", certId )
jsonRespon se = client . do_action_ with_excep tion ( request )
response = json . loads ( jsonRespon se )
if " Code " in response :
    print (" Unable to update the SLB HTTPS certificat e
. Response :")
    print ( response )
    quit ( 1 )
print (" SLB listener certificat e 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 start after we have mounted our OSS bucket. The

`ExecStart` property runs 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 " certificat e -
updater " / opt / certificat e - updater / certificat e - 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 certificat e
updater
nano resources / certificat e - manager . sh
```

Enter the following content:

```
#!/usr/bin/env bash
cd /opt/certificat e - updater
/usr/local/bin/pipenv run python2 /opt/certificat e -
updater / certificat e - 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 infrastruc ture / 15_certman / 10_ecs_slb _rule
cd infrastruc ture / 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 = "${data.alicloud_images.certman_images.0.id}"
  instance_type = "${data.alicloud_instance_types.0.id}"
  instance_types_zone_0 = "${data.alicloud_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 [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_cert man_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_ID
# - OSSFS_VERSION
#

echo "Building the certificate manager infrastructure (environment: ${ENV_NAME}, region: ${ALICLOUD_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=${CI_PIPELINE_ID}
export ENVIRONMENT=${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 : ${ ALICLOUD_R
EGION })."
```

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_cert man_infra .
sh
git add . gitlab - ci . yml

# Commit and push to GitLab
git commit - m " Add the Certificat e 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 ", " certificat e " and " infrastruc ture "
ls / mnt / oss_bucket
```

10. Check the logs of the certificate updater:

```
# Check the logs of the certificat e updater
journalctl -- unit = certificat e - updater
```

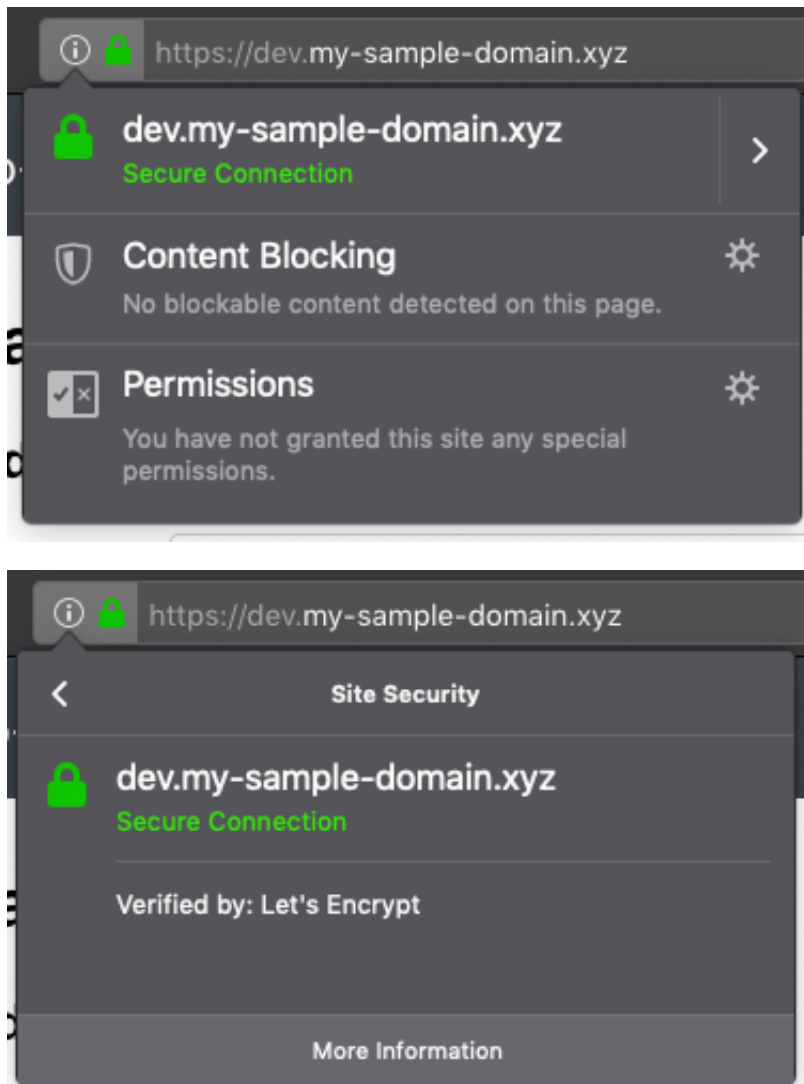
You can then test the web application from your computer with the following command:

```
# Check that the SLB is well configured with the new certificat e
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 todoclist 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 environmen t 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 environmen t 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 >
```

14.7 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.

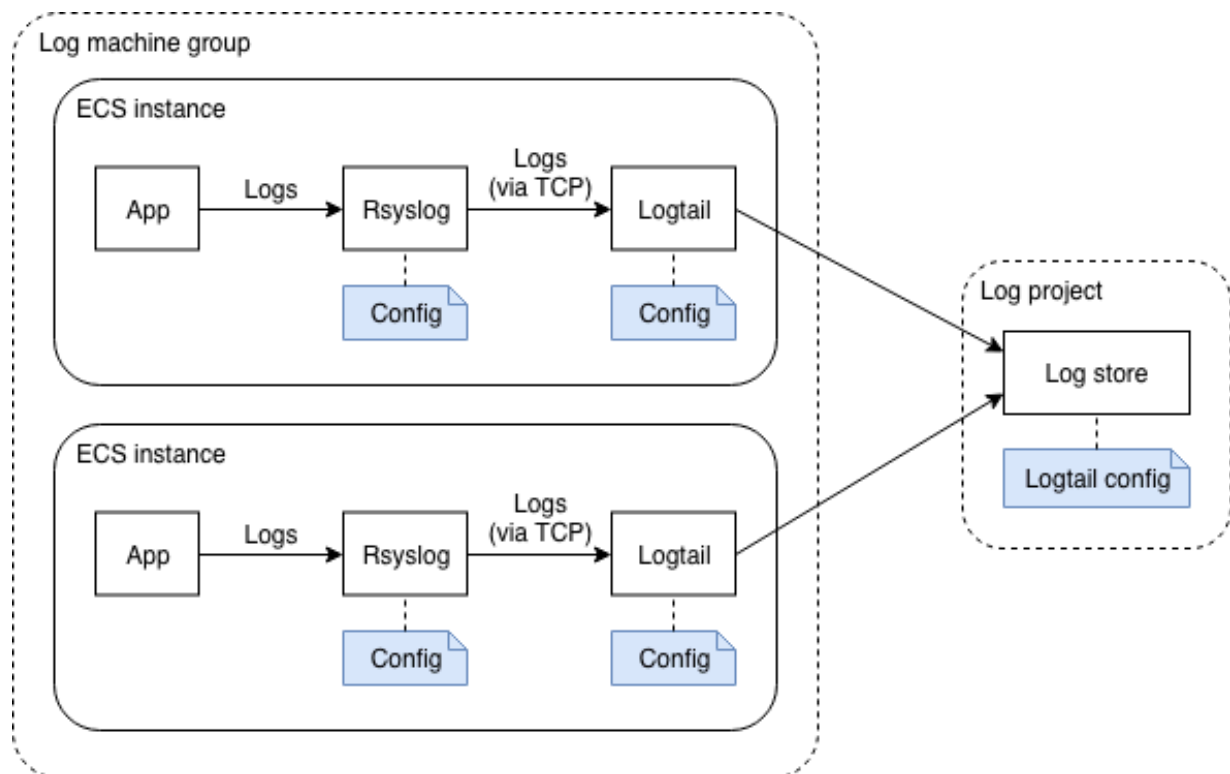


Note:

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 through **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 [#unique_161](#).

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 [#unique_163](#) 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_log_tail_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.logtailconfigdetail 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 +
      ", logProject Name = " + logProject Name + ", logtailConfi gName = " + logtailConfi gName + ")...")

client = LogClient(endpoint, accessKeyId, accessKeySecret)
existingConfig = None
try:
    response = client.get_logtail_config(logProject Name, logtailConfi gName)
    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 = logtailConfi gName, tag = "sys_tag")
if existingConfig is None:
    print("Create the logtail configuration:", configDetail.to_json())
    client.create_logtail_config(logProject Name, configDetail)
else:
    print("Update the logtail configuration:", configDetail.to_json())
    client.update_logtail_config(logProject Name, configDetail)

# Apply the configuration to the machine group
print("Apply the logtail configuration to the machine group", logMachineGroupName)
client.apply_config_to_machine_group(logProject Name, logtailConfi gName, 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 -${ ENVIRONMEN T }\" > / etc / ilogtail /
user_defin ed_id ",
" wget \" http :// logtail - release -${ REGION }. oss -${ REGION
}. aliyuncs . com / linux64 / logtail . sh \" - O logtail . sh ",
" 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_c onfig . json ",
" sed - i \" s /\(\(\(\\" streamlog_ formats \(\\" : \(\).*\\$\\\" 1
${ STREAMLOG_ FORMATS },/\\" / usr / local / ilogtail / ilogtail_c
onfig . 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 = cloudmonit.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
cp 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:

```
$ ActionQueue eFileName fwdRule1 # unique name prefix for
  spool files
$ ActionQueue eMaxDiskSpace 1g # 1gb space limit ( use as
  much as possible )
$ ActionQueue eSaveOnShutdown on # save messages to disk
  on shutdown
$ ActionQueue eType 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 % % hostname % % pri - text
  % % protocol - version % % app - name % % procid % % msgid % % msg
  :: drop - last - lf % \n "
*. * @@ 127 . 0 . 0 . 1 : 11111 ; ALI_LOG_FMT T
```

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_image / 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 /
deploy / 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_base_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

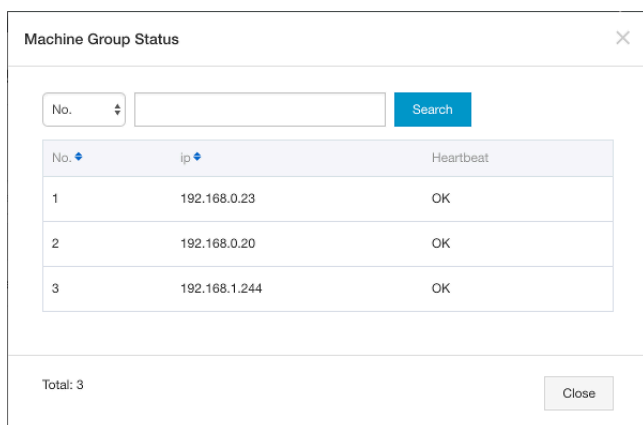
# Add the modified and new files
git add . gitlab - ci . yml
git add gitlab - ci - scripts / deploy / install_python_packages . sh
git add gitlab - ci - scripts / deploy / update_logtail_config . py
git add infrastructure / 05_vpc_slb_eip_domain / main . tf
git add infrastructure / 10_webapp / 10_image / 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 /  
certificat e - updat er . 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):



The image shows a 'Machine Group Status' popup window. It contains a search bar at the top with a dropdown menu labeled 'No.' and a 'Search' button. Below the search bar is a table with three columns: 'No.', 'ip', and 'Heartbeat'. The table has three rows of data. At the bottom of the popup, it says 'Total: 3' and has a 'Close' button.

No.	ip	Heartbeat
1	192.168.0.23	OK
2	192.168.0.20	OK
3	192.168.1.244	OK

Total: 3

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:

sample-app-log-store-de... [Back to Logstores](#)

Shard: 0 15 Mi... Preview

Log preview is only used to check whether log data is uploaded successfully. If you want to search logs through keywords, enable log index.

Time/Source	Content
Dec 13, 2018, 1:25:00 PM 127.0.0.1	app-name:CRON hostname:sample-app-ecs-zone-1-dev msg: pam_unix(cron:session): session opened for user root by (uid=0) msgid:- pri-text:authpriv.info procid:10783 protocol-version:0
Dec 13, 2018, 1:25:00 PM 127.0.0.1	app-name:CRON hostname:sample-app-ecs-zone-1-dev msg: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1) msgid:- pri-text:cron.info procid:10784 protocol-version:0
Dec 13, 2018, 1:25:00 PM 127.0.0.1	app-name:CRON hostname:sample-app-ecs-zone-1-dev msg: pam_unix(cron:session): session closed for user root msgid:- pri-text:authpriv.info procid:10783 protocol-version:0
Dec 13, 2018, 1:35:00 PM 127.0.0.1	app-name:CRON hostname:sample-app-certman-ecs-dev msg: pam_unix(cron:session): session opened for user root by (uid=0) msgid:- pri-text:authpriv.info procid:2403 protocol-version:0
Dec 13, 2018, 1:35:00 PM 127.0.0.1	app-name:CRON hostname:sample-app-certman-ecs-dev msg: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1) msgid:- pri-text:cron.info procid:2404 protocol-version:0
Dec 13, 2018, 1:35:00 PM 127.0.0.1	app-name:CRON hostname:sample-app-certman-ecs-dev msg: pam_unix(cron:session): session closed for user root msgid:- pri-text:authpriv.info procid:2403 protocol-version:0
Dec 13, 2018, 1:35:00 PM 127.0.0.1	app-name:CRON hostname:sample-app-ecs-zone-1-dev msg: pam_unix(cron:session): session opened for user root by (uid=0) msgid:- pri-text:authpriv.info procid:10846 protocol-version:0
Dec 13, 2018, 1:35:00 PM 127.0.0.1	app-name:CRON hostname:sample-app-ecs-zone-1-dev msg: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1) msgid:- pri-text:cron.info procid:10847 protocol-version:0
Dec 13, 2018, 1:35:00 PM 127.0.0.1	app-name:CRON hostname:sample-app-ecs-zone-1-dev msg: pam_unix(cron:session): session closed for user root msgid:- pri-text:authpriv.info procid:10846 protocol-version:0



Note:

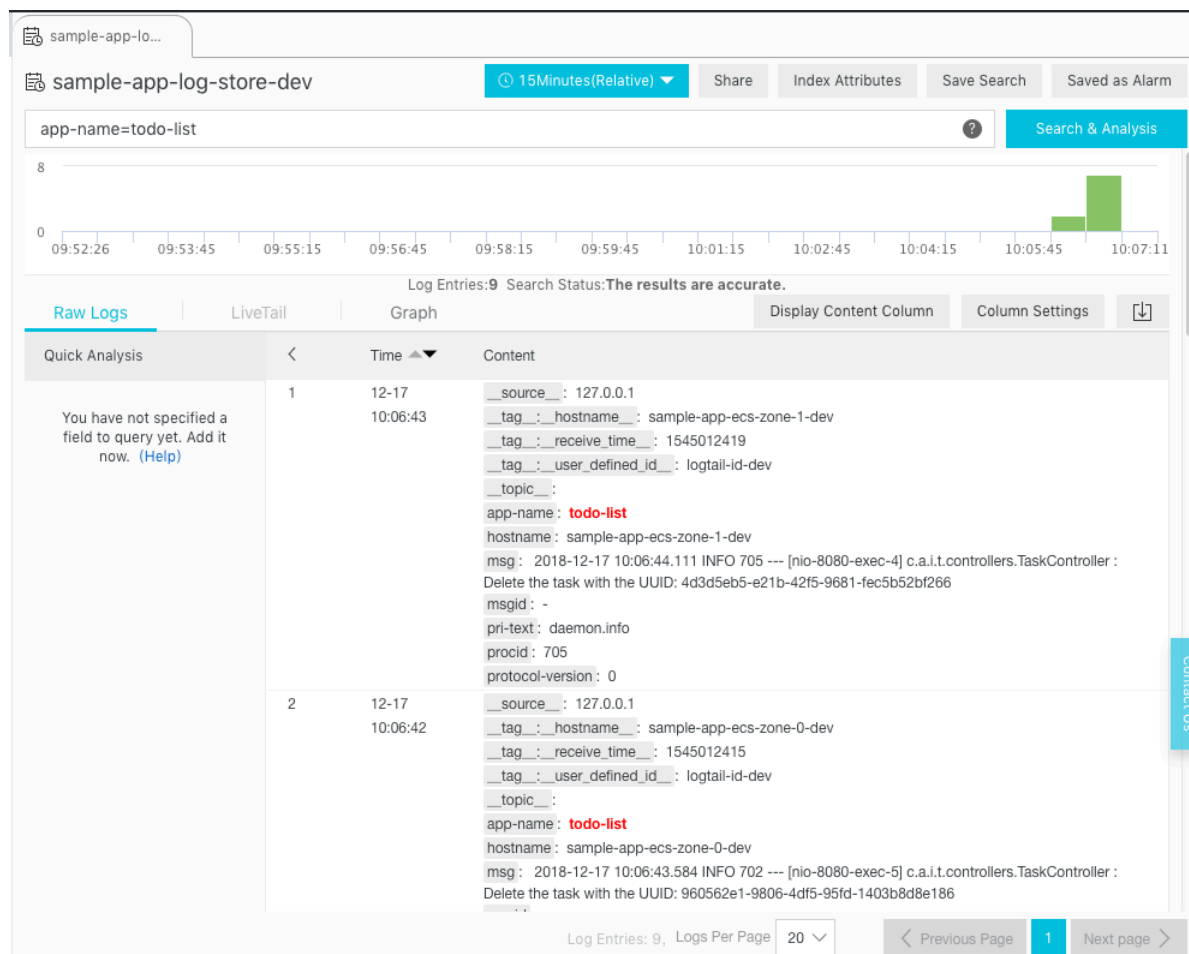
If there is no log, try to switch to the Shard: 1 and click Preview. You can generate logs by yourself by opening your web application (<http://dev.my-sample-domain.xyz/>) and by creating and deleting tasks. Sometime it may be necessary to wait for few minutes for the logs to appear.

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 [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.controller s.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.controller s.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.controller.s.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.controller.s.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.controller.s.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.controller.s.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.controller.s.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 = certificat e - 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`.

14.8 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. Log on to the [RAM console](#).

| RAM Overview

Welcome to Resource Access Management (RAM)

RAM User Logon Link: <http://signin-intl.aliyun.com/5939306421830868/login.htm>

2. Log on to the [RAM console](#).

| RAM Overview

Welcome to Resource Access Management (RAM)

RAM User Logon Link: <http://signin-intl.aliyun.com/5939306421830868/login.htm>

3. Copy the URL next to RAM User Logon Link. We will use it later.

4. Click Users from the left-side navigation pane.

5. Click Create User.

6. In the pop-up window form set sample-app-gitlab in the User Name field and click OK.

7. The page should refresh itself and display our sample-app-gitlab user. Click Authorize on the right.

8. In the new pop-up window, select the policy name AliyunContainerRegistryFullAccess and click the button with an arrow pointing to the right.

9. Click OK to close the pop-up window.

10. Click Manage on the right of the user sample-app-gitlab.

11. Click Enable Console Logon.

12. 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.

- 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:

Repositories

Reset Docker Login Password Create Repository

All Namespaces Repository Name

Container Registry will suspend the function of console authorization at 24:00, Dec. 31, 2018. Please change to use RAM to manage sub-accounts permissions, may refer to [RAM Access Management](#)

Repository Name	Namespace	Status	Repository Type	Permi...	Repository Address	Created On	Actions
simple-rest-service	cicd-k8s-tutorial	Normal	Public	Manage	↓	10/15/2018, 15:28:35	Manage Delete
crrepo1	crname	Normal			Internet registry-intl.ap-southeast-1.aliyuncs.com/sample-app/d VPC registry-intl-vpc.ap-southeast-1.aliyuncs.com/sample-a Intranet registry-intl-internal.ap-southeast-1.aliyuncs.com/samp		Manage Delete
deployment-toolbox	sample-app	Normal	Private	Manage	↓	12/19/2018, 10:44:02	Manage Delete

- 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`).
- Open a terminal and type:

```
# Test your repository configuration
docker login --username = sample - app - gitlab @ 5939306421
83 **** 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](https://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_VERS ION = 1 . 80 . 5
ENV TERRAFORM_ VERSION = 0 . 11 . 11
ENV PACKER_VER SION = 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_VERS ION }/ ossfs_ ${ OSSFS_VERS ION }
_ubuntu16 . 04_amd64 . deb "
RUN gdebi - n " ossfs_ ${ OSSFS_VERS ION } _ubuntu16 . 04_amd64
. deb "

# Install Terraform
```

```

RUN   wget "https://releases.hashicorp.com/terraform/${TERRAFORM_VERSION}/terraform_${TERRAFORM_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}_linux_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_U  SERNAME :  sample - app - gitlab @ your - user - id -
or - enterprise - alias
    REGISTRY_P  ASSWORD :  your - docker - login - password
    REGISTRY_U  RL :  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_U  SERNAME -p $ REGISTRY_P
ASSWORD $ REGISTRY_U  RL
  script :
    - docker pull $ IMAGE_URL : latest || true
    - docker build -- cache - from $ IMAGE_URL : latest -- tag
$ IMAGE_URL :$ CI_PIPELIN E_IID -- tag $ IMAGE_URL : latest .
    - docker push $ IMAGE_URL :$ CI_PIPELIN E_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](#) (with your normal account).
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:

deployment-toolbox
Singapore | Private | Local | ● Normal

Deploy Application

Tags

Refresh

Version	Image ID ⓘ	Status	Digest ⓘ	Image Size ⓘ	Last Updated	Actions
latest	d482b055579b...	● Normal	1f5235e1f38773d 3c2b9aeda32b8e 857630ac8f3414 1d0a09c80ce74b ff6700a	312.721 MB	12/19/2018, 17:54:33	Security Scan Layers Delete

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_to ols . sh` and - `./ gitlab - ci - scripts / deploy / install_py thon_packa ges . 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 ": " 3dFtcGxLLX FwcC1naXR5 YWJAMTkzOT MwNjQyMTgz
DMg2ODpIYW 5nemhvdTEw " } } }
```



Note:

- The URL `registry - intl . ap - southeast - 1 . aliyuncs . com` must be adapted to your registry domain name.
- The `auth` value `3dFtcGxLLX FwcC1naXR5 YWJAMTkzOT MwNjQyMTgz DMg2ODpIYW 5nemhvdTEw` 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_to ols . sh
```



```
git add gitlab - ci - scripts / deploy / install_py thon_packa
ges . 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 applicatio n project folder
cd ~/ projects / todolist

# Edit the pipeline definition file
nano . gitlab - ci . yml
```

Let' s start by replacing the `deploy` stage by `deploy_bas is` and `deploy_app s`:

```
stages :
- build
- quality
- deploy_bas is
- deploy_app s
```

Then split the `deploy` job into 3 blocks:

```
deploy_bas is :
  stage : deploy_bas is
  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_na me_by_bran ch_name . sh $ CI_COMMIT_ REF_NAME )"
    - " export SUB_DOMAIN _NAME=$(./ gitlab - ci - scripts /
deploy / get_sub_do main_name_ by_branch_ name . sh $ CI_COMMIT_
REF_NAME )"
    - " export BUCKET_LOC AL_PATH=/ mnt / oss_bucket "
    - " ./ gitlab - ci - scripts / deploy / mount_ossf s . sh "
    - " ./ gitlab - ci - scripts / deploy / build_basi s_infra . sh "
    - " python3 ./ gitlab - ci - scripts / deploy / update_log
tail_confi g . py $ ALICLOUD_A CCESS_KEY $ ALICLOUD_S ECRET_KEY
$ ALICLOUD_R EGION $ ENV_NAME "
    - " umount $ BUCKET_LOC AL_PATH "
    - " sleep 10 "
  only :
    - master
    - pre - production
```

```

- production

deploy_web app :
  stage : deploy_app s
  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 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_app s
  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_subdomain_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_certman_infra . sh "
    - " umount $ BUCKET_LOCAL_PATH "
    - " sleep 10 "
  only :
    - master
    - pre - production
    - production

```

As you can see the `deploy_app s` stage has 2 jobs: `deploy_web app` 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 parallel jobs for the last one:

The screenshot displays the GitLab CI/CD pipeline interface. It shows two pipeline runs, both of which have passed. The top run is for the 'master' branch (commit b9754529) and the bottom run is for the 'production' branch (commit 8c54776b). Both runs show a 'passed' status. The top run has a duration of 00:36:01 and the bottom run has a duration of 00:48:45. A tooltip for the 'production' run shows two parallel jobs: 'deploy_certman' and 'deploy_webapp', both of which also passed.

As usual, you can now merge the master branch to pre-production, and then pre-production to production.

15 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 main 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 : alibabaclo ud - howto / devops .
git

# Navigate to the folder of this tutorial
cd devops / tutorials / getting_st arted_with _devops_wi
th_kuberne tes /
```

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 =
YourSecretRootPassword'
```



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_instant_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=YourSecretRootPassword'
```



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 ckd-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 your 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_pub lic_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 deployment s
```

The result should be something like:

NAME	DESIRED	CURRENT	UP - TO - DATE
AVAILABLE			
AGE			
simple - rest	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	LoadBalanc er	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:

[illegible]

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 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 environmen t folder
cd environmen t / rancher

# Download the latest stable version of the Alibaba
Cloud provider
terraform init

# Configure the Alibaba Cloud provider
export ALICLOUD_A CCESS_KEY =" your - accesskey - id "
export ALICLOUD_S ECRET_KEY =" your - accesskey - secret "
export ALICLOUD_R EGION =" your - region - id "

# Configure variables for the Terraform scripts
export TF_VAR_ecs _root_pass word =" 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_ei_p_ip_address = 161 . 117 . 4 . 26
```

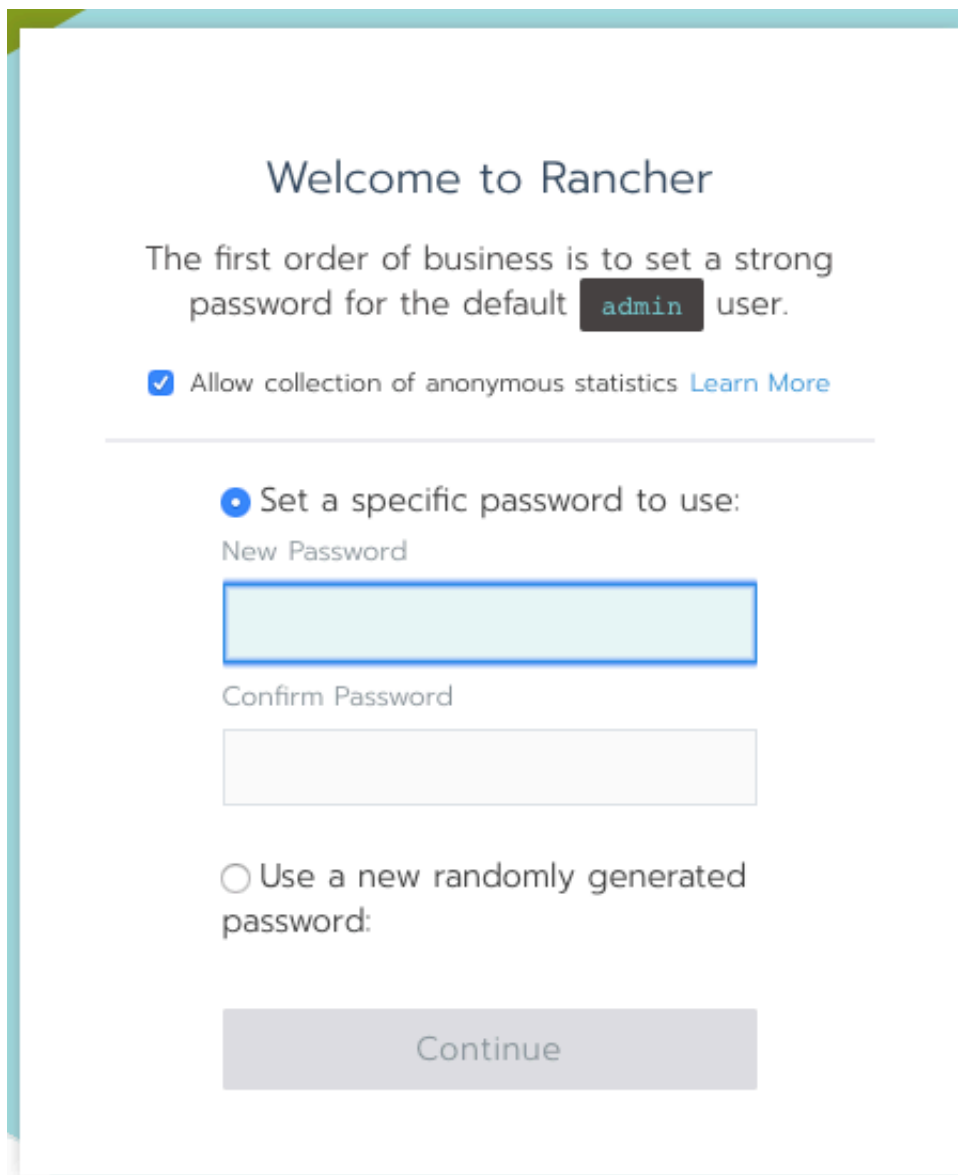
Open a web browser tab and enter the URL corresponding to `https://rancher_ei_p_ip_address` (for example, `https://161.117.4.26/`). Your web browser will complain that the connection is unsecured (which is normal because we did not configure any SSL/TLS certificate). Make an exception and continue browsing.



Note:

If using an invalid certificate bothers you, follow [this documentation](#) to set up HTTPS.

The following welcome page is displayed.

The image shows the Rancher welcome screen. At the top, it says "Welcome to Rancher". Below that, it says "The first order of business is to set a strong password for the default `admin` user." There is a checkbox labeled "Allow collection of anonymous statistics" with a link "Learn More". Below this, there are two radio button options. The first option is "Set a specific password to use:", which is selected. It has two input fields: "New Password" and "Confirm Password". The second option is "Use a new randomly generated password:", which is not selected. At the bottom, there is a "Continue" button.

Welcome to Rancher

The first order of business is to set a strong password for the default `admin` user.

☒ Allow collection of anonymous statistics [Learn More](#)

☒ Set a specific password to use:

New Password

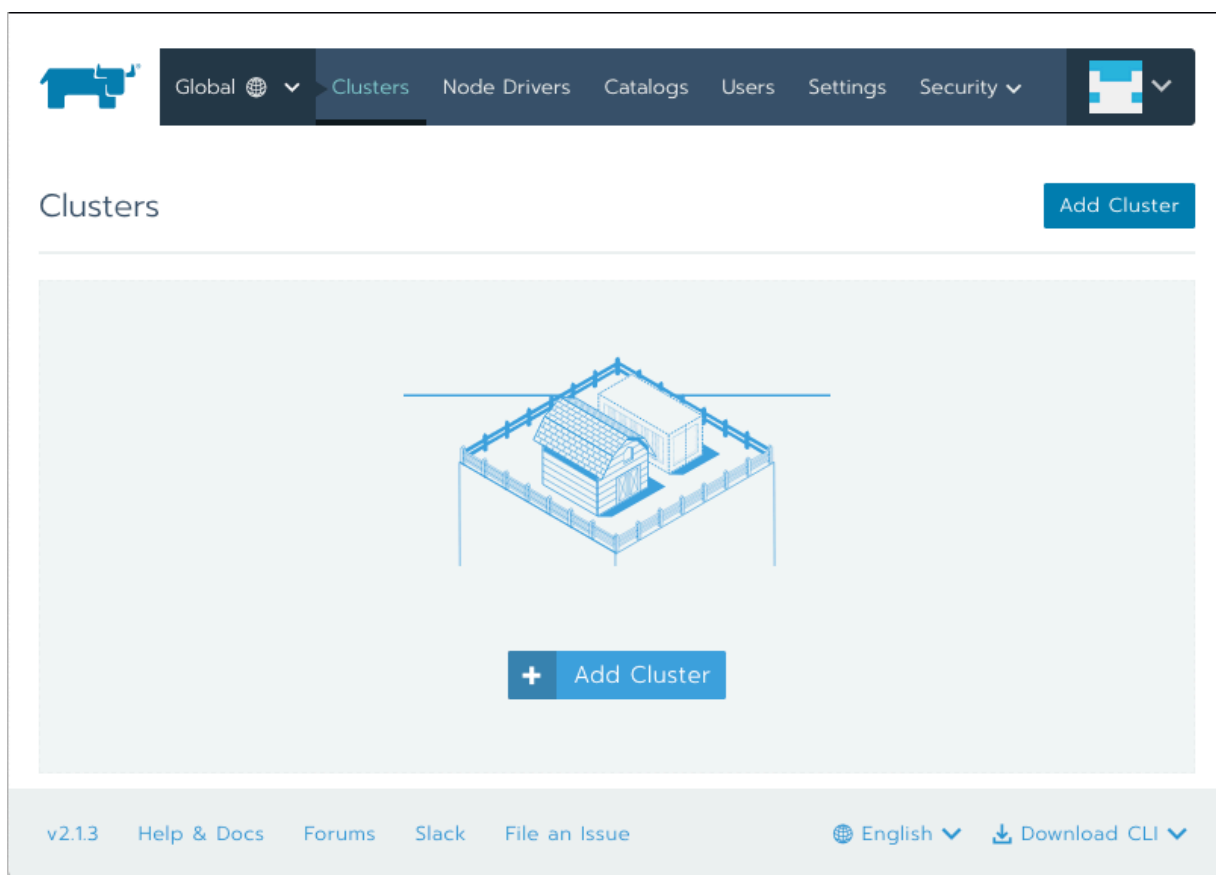
Confirm Password

☐ Use a new randomly generated password:

Continue

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 environmen t folder
cd environmen t / kubernetes - cluster

# Download the latest stable version of the Alibaba
Cloud provider
terraform init

# Configure the Alibaba Cloud provider
export ALICLOUD_A CCESS_KEY =" your - accesskey - id "
export ALICLOUD_S ECRET_KEY =" your - accesskey - secret "
export ALICLOUD_R EGION =" your - region - id "

# Configure variables for the Terraform scripts
export TF_VAR_ecs _root_pass word =" YourR00tP @ ssword "
export TF_VAR_mas ter_instan ce_cpu_cou nt = 4
export TF_VAR_mas ter_instan ce_ram_amo unt = 8 # in GB
export TF_VAR_mas ter_instan ce_disk_si ze = 40 # in GB
export TF_VAR_wor ker_instan ce_count = 3 # Must be >= 3
export TF_VAR_wor ker_instan ce_cpu_cou nt = 4
export TF_VAR_wor ker_instan ce_ram_amo unt = 32 # in GB
export TF_VAR_wor ker_instan ce_disk_si ze = 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_k8 s_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_k8 s_cluster_ ip_address `.
# Note 1 : the password is the one that was set in
` TF_VAR_ecs _root_pass word `.

# 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:

Global Clusters Node Drivers Catalogs Users Settings Security

Add Cluster: alibabacloud-cluster

Note: If you want to import a Google Kubernetes Engine (GKE) cluster (or any cluster that does not supply you with a kubectl configuration file with the ClusterRole **cluster-admin** bound to it), you need to bind the ClusterRole **cluster-admin** using the command below.

Replace **[USER_ACCOUNT]** with your Google account address (you can retrieve this using **gcloud config get-value account**). If you are not importing a Google Kubernetes Engine cluster, replace **[USER_ACCOUNT]** with the executing user configured in your kubectl configuration file.

```
kubectl create clusterrolebinding cluster-admin-binding --clusterrole cluster-admin --user [USER_ACCOUNT]
```

Run the kubectl command below on an existing Kubernetes cluster running a supported Kubernetes version to import it into Rancher:

```
kubectl apply -f https://161.117.4.26/v3/import/nlz588gctmkkkpc8jsnrht8ff65gbp4d629smqzbcjpvxzltfdmph.yaml
```

If you get an error about 'certificate signed by unknown authority' because your Rancher installation is running with an untrusted/self-signed SSL certificate, run the command below instead to bypass the certificate check:

```
curl --insecure -sfL https://161.117.4.26/v3/import/nlz588gctmkkkpc8jsnrht8ff65gbp4d629smqzbcjpvxzltfdmph.yaml | kubectl apply -f -
```

Done

v2.13 Help & Docs Forums Slack File an Issue English Download CLI

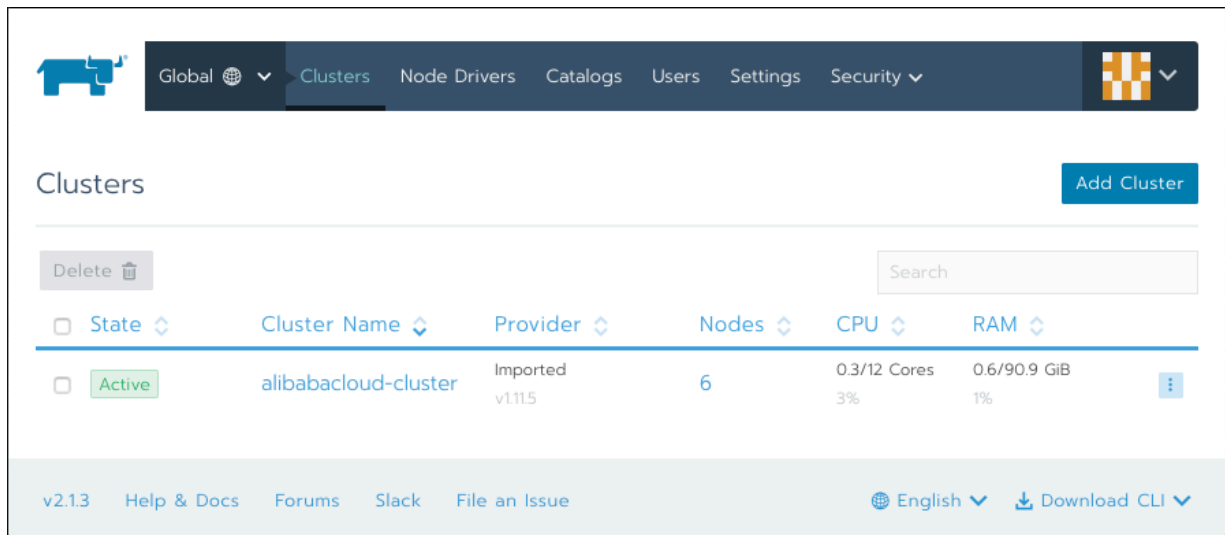
Let's execute the last command. Copy it and paste it in your terminal:

```
# Command from Rancher in order to import the
cluster
curl --insecure -sfL https://161.117.4.26/v3/import
/nlz588gctmkkkpc8jsnrht8ff65gbp4d629smqzbcjpvxzltfdmph.
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 - credential - 1d26caa created
clusterrole .rbac.authorization.k8s.io / cattle - admin
created
deployment.extensions / cattle - cluster-agent created
daemonset.extensions / cattle - node-agent created
```

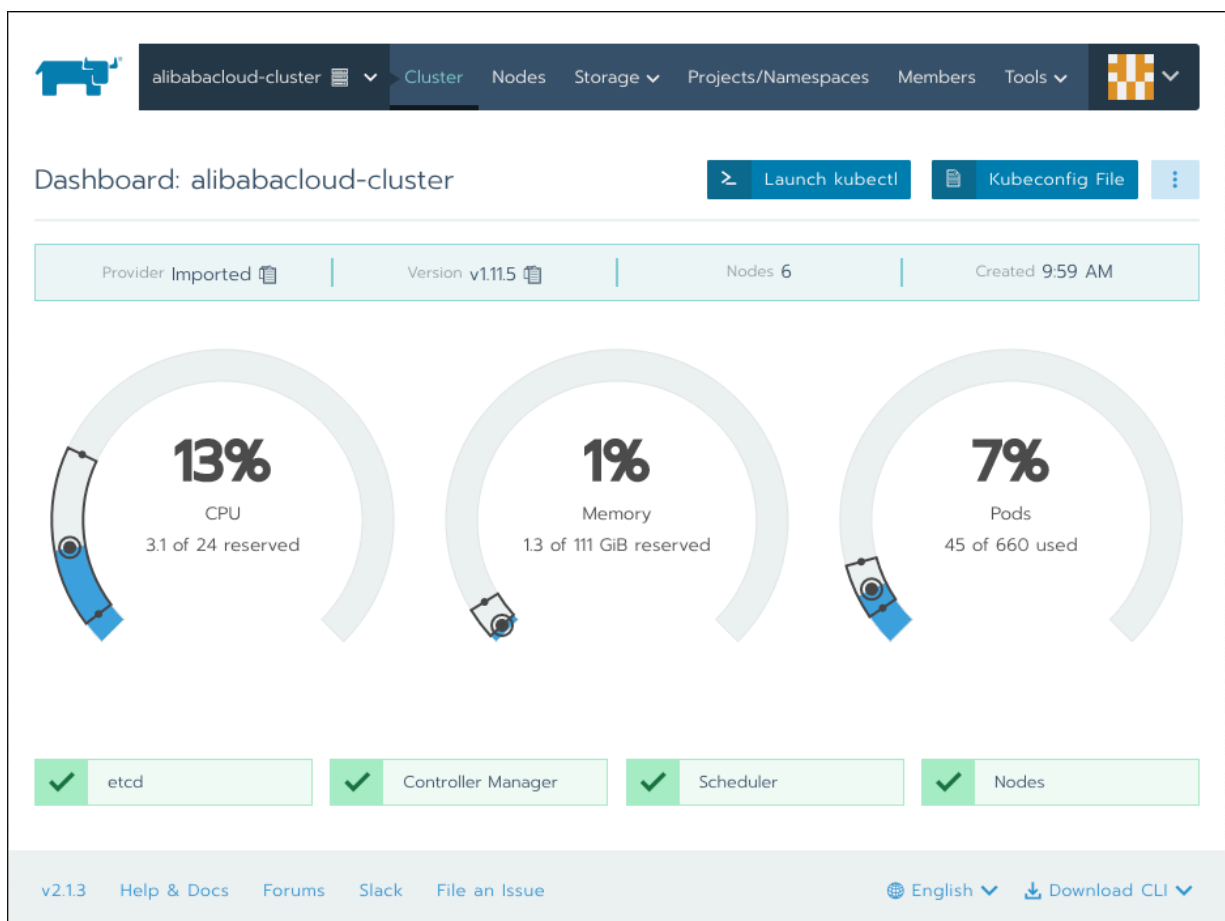
Go back to the web browser tab and click Done. You should now see your cluster:



The screenshot shows the Rancher 'Clusters' page. At the top, there's a navigation bar with 'Global', 'Clusters', 'Node Drivers', 'Catalogs', 'Users', 'Settings', and 'Security'. Below this, the 'Clusters' section has a 'Delete' button and a search bar. A table lists the clusters with columns: State, Cluster Name, Provider, Nodes, CPU, and RAM. One cluster is listed: 'alibabacloud-cluster' with status 'Active', provider 'Imported v1.11.5', 6 nodes, 0.3/12 Cores (3% CPU), and 0.6/90.9 GiB (1% RAM). At the bottom, there are links for 'v2.1.3', 'Help & Docs', 'Forums', 'Slack', 'File an Issue', 'English', and 'Download CLI'.

State	Cluster Name	Provider	Nodes	CPU	RAM
Active	alibabacloud-cluster	Imported v1.11.5	6	0.3/12 Cores 3%	0.6/90.9 GiB 1%

On this page, click the cluster name (alibabacloud-cluster). You should obtain a dashboard similar to this one:



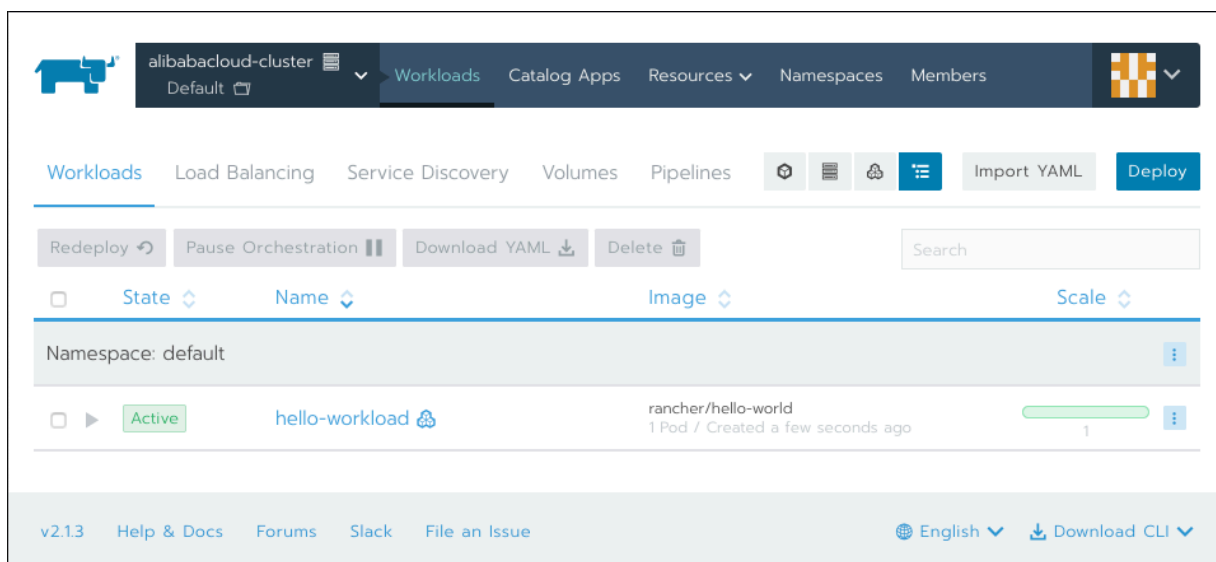
The screenshot shows the 'Dashboard: alibabacloud-cluster' page. The top navigation bar includes 'alibabacloud-cluster', 'Cluster', 'Nodes', 'Storage', 'Projects/Namespaces', 'Members', and 'Tools'. Below the navigation bar, there are buttons for 'Launch kubectl' and 'Kubeconfig File'. A summary bar shows 'Provider Imported', 'Version v1.11.5', 'Nodes 6', and 'Created 9:59 AM'. The main section features three large circular gauges: CPU at 13% (3.1 of 24 reserved), Memory at 1% (1.3 of 111 GiB reserved), and Pods at 7% (45 of 660 used). Below these, there are four status boxes with green checkmarks: 'etcd', 'Controller Manager', 'Scheduler', and 'Nodes'. At the bottom, there are links for 'v2.1.3', 'Help & Docs', 'Forums', 'Slack', 'File an Issue', 'English', and 'Download CLI'.

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:

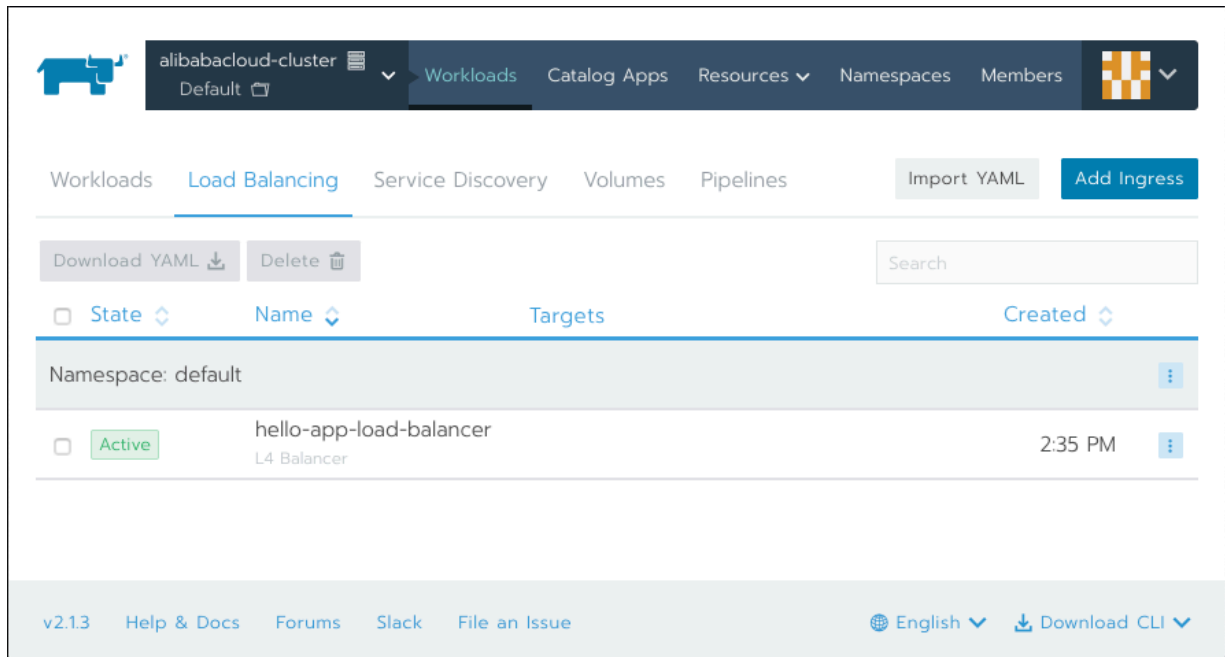
```
apiVersion : v1
kind : Service
metadata :
  name : hello - app - load - balancer
  labels :
    app : hello - app
spec :
  type : LoadBalancer
  ports :
  - port : 80
    protocol : TCP
    targetPort : 80
```

```
selector :
  app : hello - app
```

3. Click Import.

4. Click Load Balancing.

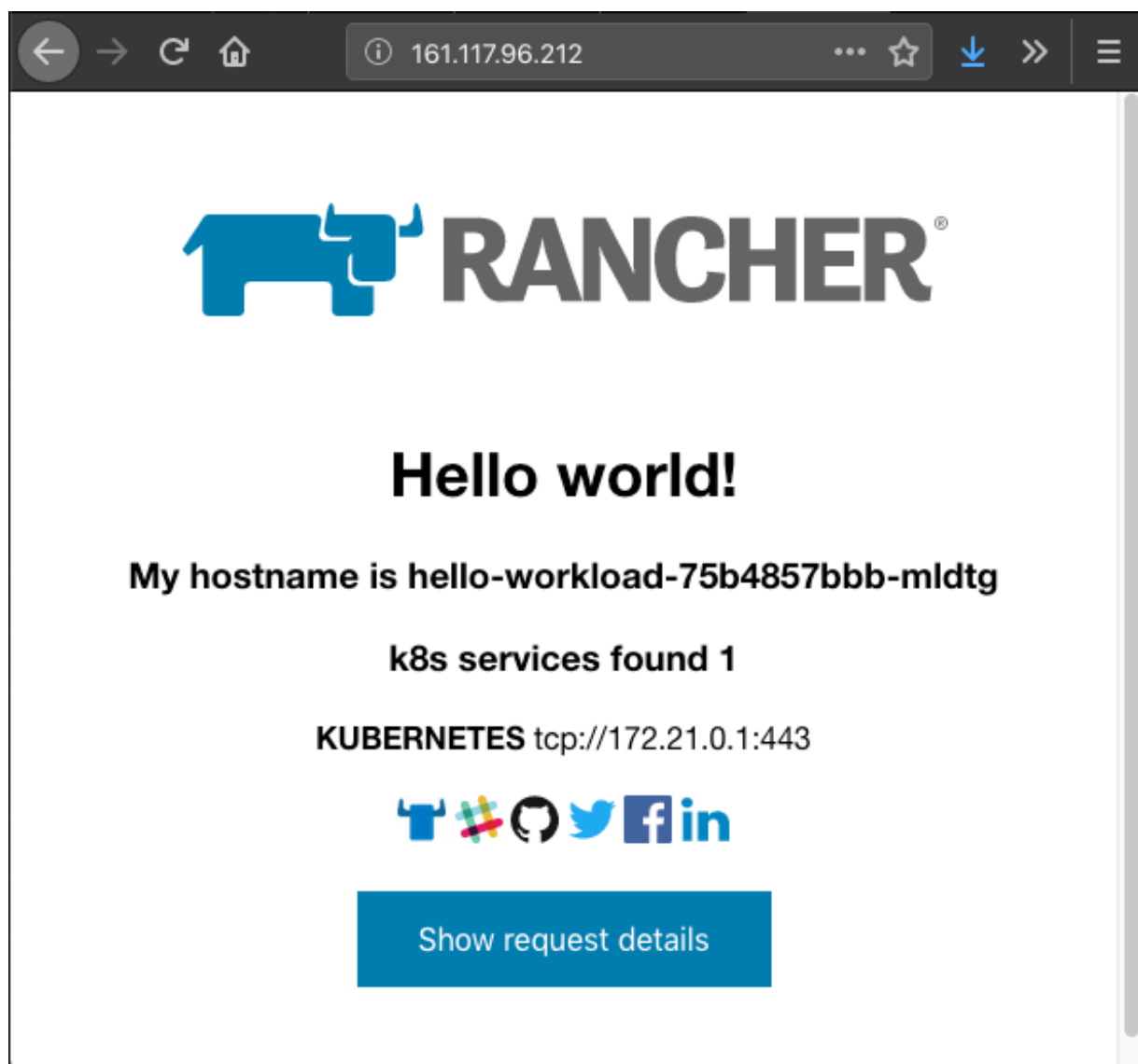
You can see your load balancer:



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](#).