ALIBABA CLOUD

阿里云

云服务器ECS

最佳实践

文档版本：20220712

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.常用操作导航

ECS高频操作ECS学习路径

在使用云服务器ECS时，您可能会遇到各种问题，例如远程连接、更换操作系统、扩容云盘、升高或降低实例配置、使用快照或镜像等。本文介绍了云服务器ECS的常用操作，供您参考。

使用限制

- 使用云服务器ECS的注意事项，请参见使用须知。
- 使用云服务器ECS的资源规格限制，请参见使用限制和查看和提升实例配额。
-

创建并管理ECS实例

- 您可以按以下步骤操作ECS实例的生命周期：
  i. 使用向导创建实例
  ii. 远程连接ECS实例
  iii. 停止实例
  iv. 释放实例
- 如果当前的实例规格或网络配置无法满足业务需求，您可以变更实例规格、IP地址和公网带宽峰值：
  - 包年包月实例：
    - 包年包月实例升配规格
    - 续费降配
    - 临时升级带宽（连续时间段）
  - 按量付费实例：
    - 按量付费实例变配规格
    - 按量付费实例修改带宽
  - ECS实例IP地址操作：
    - 更换公网IP地址
    - 专有网络类型ECS公网IP转为弹性公网IP
- 如果当前的操作系统无法满足需求，您可以更换操作系统。具体操作，请参见更换操作系统。
- 您可以使用以下功能精细化控制和管理ECS实例：
  - 实例自定义数据
  - 实例元数据
  - 实例标识
  - 实例RAM角色

管理计费

- 包年包月实例：

  您可以使用不同的方式续费包年包月实例：
  - 手动续费实例
  - 自动续费实例

- 续费降配
- 续费变配
- 统一包年包月实例的到期日

- 按量付费实例：

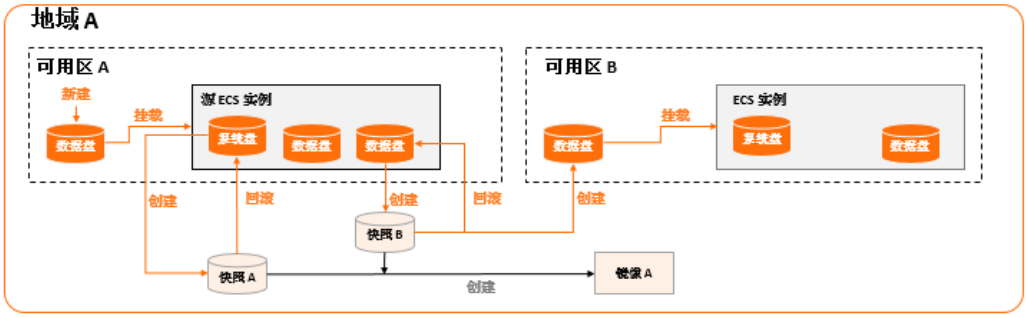
  您可以为按量付费实例启用节省停机模式，更多信息，请参见按量付费实例节省停机模式。

- 转换实例计费方式：
  - 按量付费转包年包月
  - 包年包月转按量付费

提高计费性价比

- 您可以购买抢占式实例，降低部分场景下的使用成本，搭配弹性供应实现自动化交付。具体操作，请参见创建弹性供应组和创建抢占式实例。
- 您可以购买预留实例券，提高实例抵扣方式灵活性和降低成本。具体操作，请参见购买预留实例券。
- 您可以购买存储容量单位包，抵扣同一地域下按量付费云盘的计费账单。具体操作，请参见创建存储容量单位包。

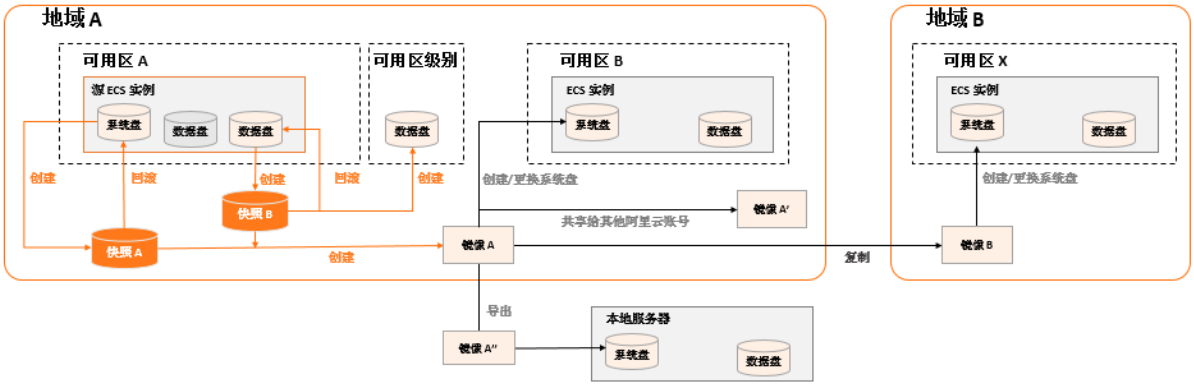
创建并管理云盘



当云盘作数据盘用时，您可以按以下步骤使用云盘：

1. 创建云盘。
2. 挂载数据盘。
3. 分区格式化数据盘（Linux）或分区格式化数据盘（Windows）。
4. 创建快照备份数据。具体操作，请参见创建一个云盘快照。
5. 如果已有的云盘容量无法满足需求，您可以扩容云盘。具体操作，请参见以下文档：
   - 在线扩容云盘（Linux系统）
   - 离线扩容云盘（Linux系统）
   - 在线扩容云盘（Windows系统）
   - 离线扩容云盘（Windows系统）
6. 如果云盘数据出错，您可以使用某个时刻的云盘快照回滚云盘。具体操作，请参见使用快照回滚云盘。
7. 如果要将云盘恢复到初始状态，您可以重新初始化云盘。具体操作，请参见重新初始化数据盘。
8. 卸载数据盘。
9. 释放云盘。

创建和管理快照



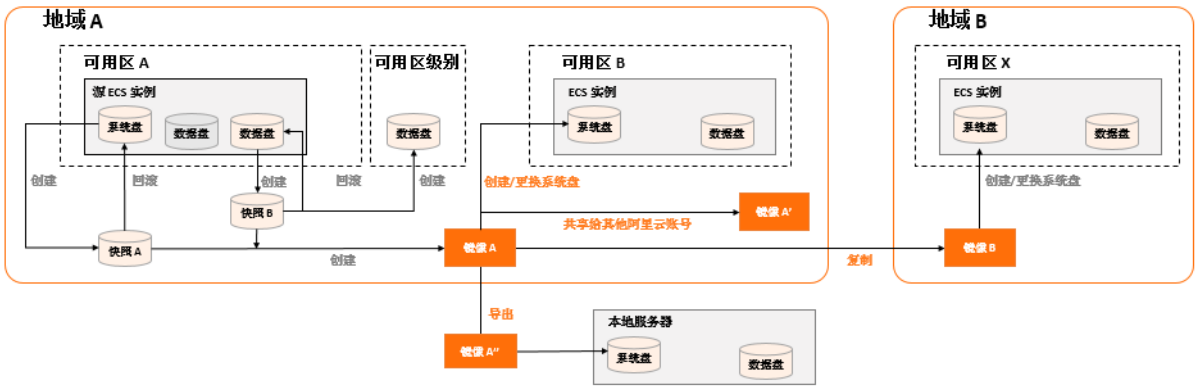
您可以按以下步骤使用快照：

1. 创建快照，支持手动创建快照和自动创建快照：
   - 创建一个云盘快照。
   - 使用自动快照策略，定期自动创建快照。具体操作，请参见执行或取消自动快照策略。
2. 查看快照容量。
3. 为了节省快照存储空间，删除不必要的快照。具体操作，请参见优化快照使用成本。

快照的常见应用场景如下所示：

- 用于拷贝或恢复数据：您可以使用快照创建云盘或者回滚云盘。具体操作，请参见使用快照创建云盘和使用快照回滚云盘。
- 用于快速部署环境：您可以使用系统盘快照创建自定义镜像，并使用自定义镜像创建实例。具体操作，请参见使用快照创建自定义镜像和使用自定义镜像创建实例。

创建并管理自定义镜像



控制台上操作的主要都是自定义镜像。使用自定义镜像，您可以快速部署业务环境。您可以通过以下方式获取自定义镜像。

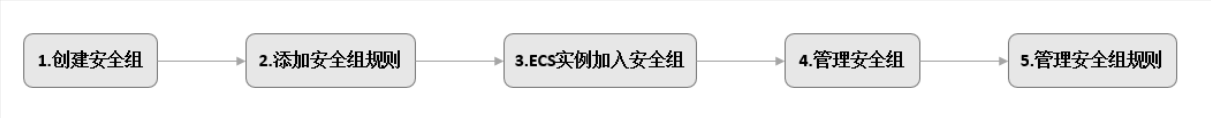
- 使用快照创建自定义镜像。
- 使用实例创建自定义镜像。
- 使用Packer创建自定义镜像。
- 不同地域之间复制镜像。具体操作，请参见复制镜像。
- 不同账号之间共享镜像。具体操作，请参见共享镜像。

- 导入自定义镜像。
- 使用Packer创建并导入本地镜像。

您可以导出镜像备份环境。具体操作，请参见导出镜像。

创建并管理安全组

您可以按以下步骤使用安全组：



1. 创建安全组。
2. 添加安全组规则。
3. ECS实例加入安全组。
4. 删除安全组规则。
5. 删除安全组。

为了方便部署业务，您可以跨地域、跨网络类型克隆安全组。具体操作，请参见克隆安全组。

如果新的安全组规则对线上业务产生了不利影响，您可以全部或部分还原安全组规则。具体操作，请参见还原安全组规则。

创建并授予实例RAM角色

您可以按以下步骤使用密钥对：

1. （可选）为RAM用户授予操作实例RAM角色的权限策略。具体操作，请参见授权RAM用户使用实例RAM角色。
2. 创建并授予实例RAM角色。具体操作，请参见授予实例RAM角色。
3. 在使用过程中，您可以随时更换实例RAM角色。具体操作，请参见更换实例RAM角色。

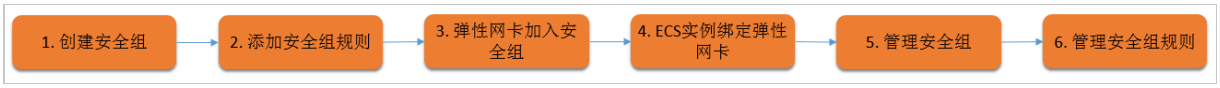
创建并使用密钥对

您可以按以下步骤使用密钥对：

1. 创建SSH密钥对或者导入SSH密钥对。
2. 绑定SSH密钥对。
3. 通过密钥认证登录Linux实例。
4. 解绑SSH密钥对。
5. 删除SSH密钥对。

创建并使用弹性网卡

您可以按以下步骤使用弹性网卡：



1. 创建弹性网卡。
2. 将弹性网卡附加到实例或者在创建实例时附加弹性网卡。
3. （可选）配置辅助弹性网卡。

4. 分配辅助私网IP地址。
5. 解绑弹性网卡。
6. 删除弹性网卡。

搭建IPv6专有网络

- 运行Windows Server操作系统的ECS实例的具体操作，请参见Windows实例使用IPv6导航。
- 运行Linux操作系统的ECS实例的具体操作，请参见Linux实例使用IPv6导航。

使用标签

您可以使用标签管理各种资源，提高效率。您可以按以下步骤使用标签：

1. 新建并绑定标签。
2. 使用标签检索资源。
3. 删除或解绑标签。

使用实例启动模板

实例启动模板帮助您快速创建相同配置的ECS实例，您可以按以下步骤使用实例启动模板：

1. 创建实例启动模板。
2. 创建实例启动模板的新版本。
3. 删除实例启动模板和版本。

使用部署集

部署集帮助您提供底层应用的高可用性，您可以按以下步骤使用部署集：

1. 创建部署集。
2. 在部署集内创建ECS实例。
3. 调整实例所属部署集。
4. 删除部署集。

使用云助手

云助手可以发送远程命令，免去了运维过程中的使用跳板机的不便。您可以按以下步骤使用云助手：

1. （可选）部分ECS实例需要您手动安装和配置云助手客户端。具体操作，请参见安装云助手客户端。
2. 创建命令。
3. 执行命令。
4. 查看执行结果及修复常见问题。

2.选型最佳实践

本文通过一些常见的选型场景推荐，便于您了解实例规格的关键特点，在库存不足、产品下线、使用抢占式实例等场景中，您可以有多种备选实例规格，充分利用阿里云云服务器ECS弹性灵活的特点。

本文主要介绍如何选择企业级实例规格族，不包括入门级（共享型）规格族。有关入门级实例选型，请参见共享型或突发性能实例概述。

最新活动，可关注ECS产品详情页。

了解实例规格族

启动一台ECS实例前，您需要结合性能、价格、工作负载等因素，做出性价比与稳定性最优的决策。根据业务场景和vCPU、内存、网络性能、存储吞吐等配置划分，阿里云云服务器ECS提供了多种实例规格族，一种实例规格族又包括多个实例规格。实例规格族名称格式为ecs.<规格族>，实例规格名称为ecs.<规格族>.<nx>large。

- ecs：云服务器ECS的产品代号。
- <规格族>：由小写字母加数字组成。
  - 小写字母为某个单词的缩写，并标志着规格族的性能领域。部分小写字母的含义如下所示：
    - c：一般表示计算型（computational）
    - g：一般表示通用型（general）
    - r：一般表示内存型（ram）
    - ne：一般表示网络增强型（network enhanced）
  - 数字一般用于区别同类型规格族间的发布时间。更大的数字代表新一代规格族，拥有更高的性价比，价格低性能好。
- <nx>large：n越大，vCPU核数越多。

例如，ecs.g6.2xlarge表示通用型g6规格族中的一个实例规格，拥有8个vCPU核。相比于g5规格族，g6为新一代通用型实例规格族。

开始选型

为方便您在选型时对比实例性能，您可以从以下途径了解实例详情。

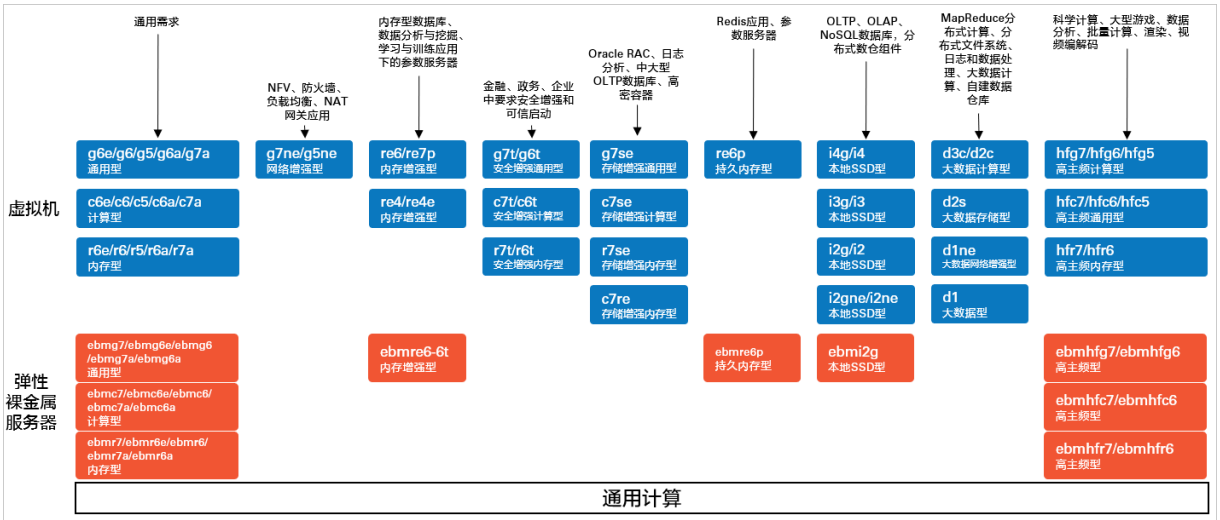
- 实例规格族：查阅文档了解实例规格族的产品详情，无需账号登录。
- DescribeInstanceTypes：调用ECS API，获取最新的性能规格参数，但需要您已注册账号。

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

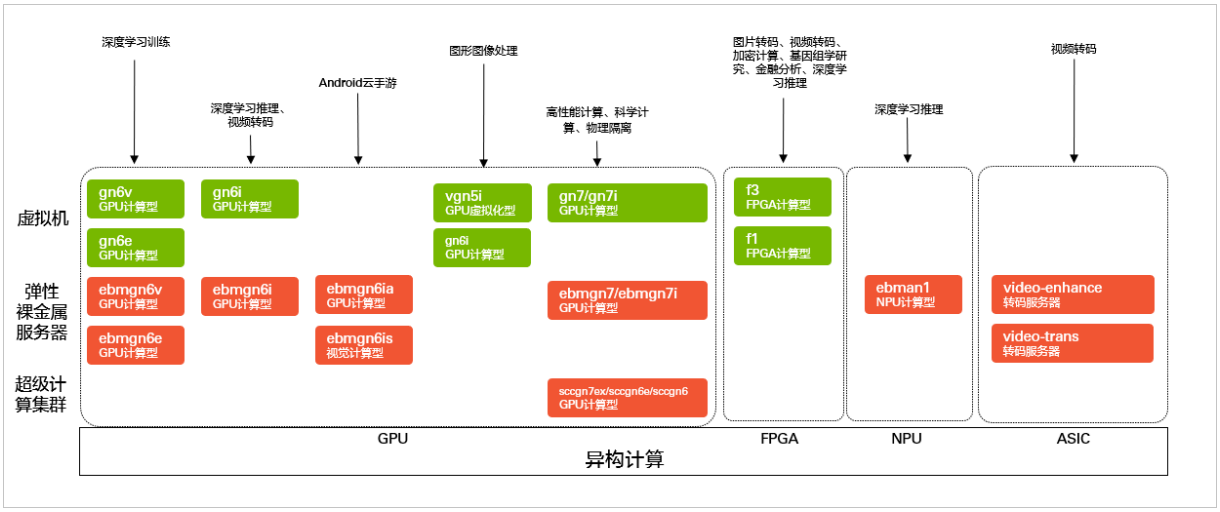
- 云产品定价页：了解ECS资源的定价信息、最新优惠活动、以及初步估算使用成本。
- ECS自定义购买页面：在**基础配置**页面的**实例**配置处，了解更多选购指导。
- 活动页面：根据您是否购买过云服务器ECS选择以下页面了解适用的优惠活动：
  - 老用户活动页
  - 新用户活动页

根据使用场景挑选

下图列举了云服务器ECS部分通用计算实例规格族及其对应的业务场景。



下图列举了云服务器ECS部分异构计算实例规格族及其对应的业务场景。



根据典型应用推荐

如果您使用的是类似于下图中的软件或应用，可以挑选右侧对应的实例规格族。

自建服务的选型推荐

如果您是自建服务，可以根据您使用的应用，并参考选型原则，选择对应的实例规格族。

应用类型	常用应用	选型原则	推荐实例规格族
负载均衡	Nginx	应用特点：需要支持高频率的新建连接操作。 • CPU计算能力：要求较高。 • 内存：要求不高。	c6e、hfc7、g5ne系列
RPC产品	• SOFA • Dubbo	应用特点：网络链接密集型；进程运行时需要消耗较高的内存。	g6e、g6、g6a、g7a系列
缓存	• Redis • Memcache • Solo	• CPU计算能力：要求不高。 • 内存：要求较高。	• 实例规格族：r6e、re6、re6p、re7p、r7p系列 • 块存储：SSD云盘或ESSD云盘
配置中心	ZooKeeper	在应用启动协商时会有大量I/O读写操作。 • CPU计算能力：要求不高。 • 内存：要求不高。	• 实例规格族：c6e、c6、c6a、c7a系列 • 块存储：SSD云盘或ESSD云盘

应用类型	常用应用	选型原则	推荐实例规格族
消息队列	• Kafka • RabbitMQ	从消息完整性方面考虑，存储优先选用云盘。 • CPU计算能力：要求不高。 • 内存和vCPU配比通常为1:1。 • 存储：要求不高。	• 实例规格族：c6e、c6、c6a、c7a系列 • 块存储：SSD云盘或ESSD云盘
容器编排	Kubernetes	通过弹性裸金属服务器和容器组合，可以最大限度挖掘计算潜能。	ebmc6e、ebmg6e、ebmc6、ebmg6、ebmc6a、ebmc7a、ebmg6a、ebmg7a系列
大表存储	HBase	• 一般可以选择d系列。 • 如果业务存在超高IOPS（Input/Output Operations Per Second）需求，可以选择i系列。	• d2c、d2s系列 • i3系列
数据库	• MySQL • NoSQL	• 对于存储有弹性扩展的需求，可以选择ECS和ESSD。 • 对于I/O敏感型业务的需求，优先选择i系列。	• 实例规格：c6e、g6e、r6e系列 块存储：ESSD云盘 • i3、i4p系列
	SQLServer	• 由于Windows的I/O单通道特性，对I/O读写能力要求较高，优先选择ESSD。 • ECS的逻辑和物理扇区设置为4K。	• 实例规格族：c6e、g6e、r6e系列 • 块存储：ESSD云盘
文本搜索	Elasticsearch	• 选用内存与vCPU配比较大的ECS规格。 • 日常需要将数据库数据导出成ES文件，对I/O读写有要求。	• 实例规格：g6e、g6、g6a、g7a系列 块存储：ESSD云盘 • d2c、d2s系列
实时计算	• Flink • Blink	基于存储量可以选择ECS通用规格和云盘，也可以选择d系列。	d2c、d2s系列

应用类型	常用应用	选型原则	推荐实例规格族
离线计算	• Hadoop • HDFS • CDH	优先选择d系列。	d2c、d2s系列

通用场景、游戏服、视频直播场景推荐

在该类场景中，性能需求表现为CPU计算密集型，您需要相对均衡的处理器与内存资源配比，通常选用CPU与内存配比1:2、系统盘选用高效云盘、数据盘选用SSD云盘或者ESSD云盘。如果业务需要更强的网络性能，如视频弹幕等，您可以选用同系列中更高规格的实例规格，提高网络收发包能力（PPS）。

场景分类	场景细分	推荐规格族	性能需求	处理器与内存比
通用应用	均衡性能应用，后台应用	g系列，如g6e	中主频，计算密集型	1:4
	高网络收发包应用	g系列，如g6e	高网络PPS，计算密集型	1:4
	高性能计算	hfc系列，如hfc7	高主频，计算密集型	1:2
游戏应用	高性能端游	hfc系列，如hfc7	高主频	1:2
	手游、页游	g系列，如g6e	中主频	1:4
视频直播	视频转发	g系列，如g6e	中主频，计算密集型	1:4
	直播弹幕	g系列，如g6e	高网络PPS，计算密集型	1:4

Hadoop、Spark、Kafka大数据场景推荐

在该类场景中，由于涉及不同的节点，性能需求表现较为复杂，您需要均衡各个节点的性能表现，包括计算、存储吞吐、网络性能等。

- 管理节点：当作通用场景处理，请参见通用场景、游戏服、视频直播场景推荐。
- 计算节点：当作通用场景处理，请参见通用场景、游戏服、视频直播场景推荐。根据集群规模的不同，需要选择的实例规格不同。例如100个节点以下可以选用ecs.g6e.4xlage，100个节点以上可以选用ecs.g6e.8xlage。

  > **说明** 计算节点在计费模式上可以采用抢占式实例，实现性价比最优化。更多详情，请参见抢占式实例概述。

- 数据节点：需要高存储吞吐、高网络吞吐、均衡的处理器与内存配比，推荐您使用大数据型d系列规格族。例如MapReduce/Hive可选择ecs.d2s.5xlarge，Spark/Mlib可选择ecs.d2s.10xlarge。

数据库、缓存、搜索场景推荐

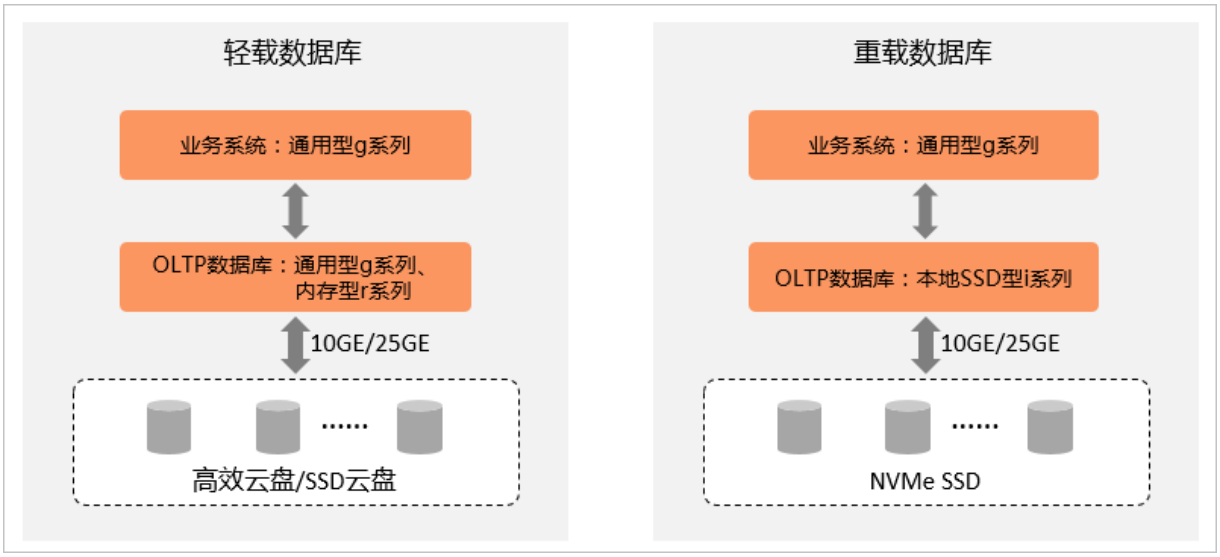
在该类场景中，实例规格的处理器与内存配比一般要求高于1:4，部分软件对存储I/O读写能力及时延性能较为敏感，建议您选用单位内存性价比较高的规格族。

场景分类	场景细分	推荐规格族	处理器与内存比	数据盘
关系型数据库	高性能，依赖应用层高可用	i系列	1:4	本地SSD存储、高效云盘、SSD云盘
	中小型数据库	g系列，或其他内存占比为1:4的规格族	1:4	高效云盘、SSD云盘
	高性能数据库	r系列	1:8	高效云盘、SSD云盘
分布式缓存	中内存消耗场景	g系列，或其他内存占比为1:4的规格族	1:4	高效云盘、SSD云盘
	高内存消耗场景	r系列	1:8	高效云盘、SSD云盘
NoSQL数据库	高性能，应用层高可用	i 系列	1:4	本地SSD存储、高效云盘、SSD云盘
	中小型数据库	g系列，或其他内存占比为1:4的规格族	1:4	高效云盘、SSD云盘
	高性能数据库	r系列	1:8	高效云盘、SSD云盘
ElasticSearch	小集群，靠云盘保证数据高可用	g系列，或其他内存占比为1:4的规格族	1:4	高效云盘、SSD云盘

ElasticSearch 场景分类	场景细分	推荐规格族	处理器与内存比	数据盘
	大集群，高可用	d系列	1:4	本地SSD存储、高效云盘、SSD云盘

以数据库为例，在传统方式中，业务系统直接对接OLTP数据库，数据冗余大多通过RAID磁盘阵列实现。选择云服务器ECS，您的轻载、重载数据库都能实现灵活部署。

- 轻载数据库：采用企业级实例规格搭配云盘使用，性价比更高。
- 重载数据库：需要高存储IOPS和低读写延时，推荐您使用本地SSD型i系列实例规格族（搭配了高I/O型本地NVMeSSD本地盘），满足大型重载数据库要求。

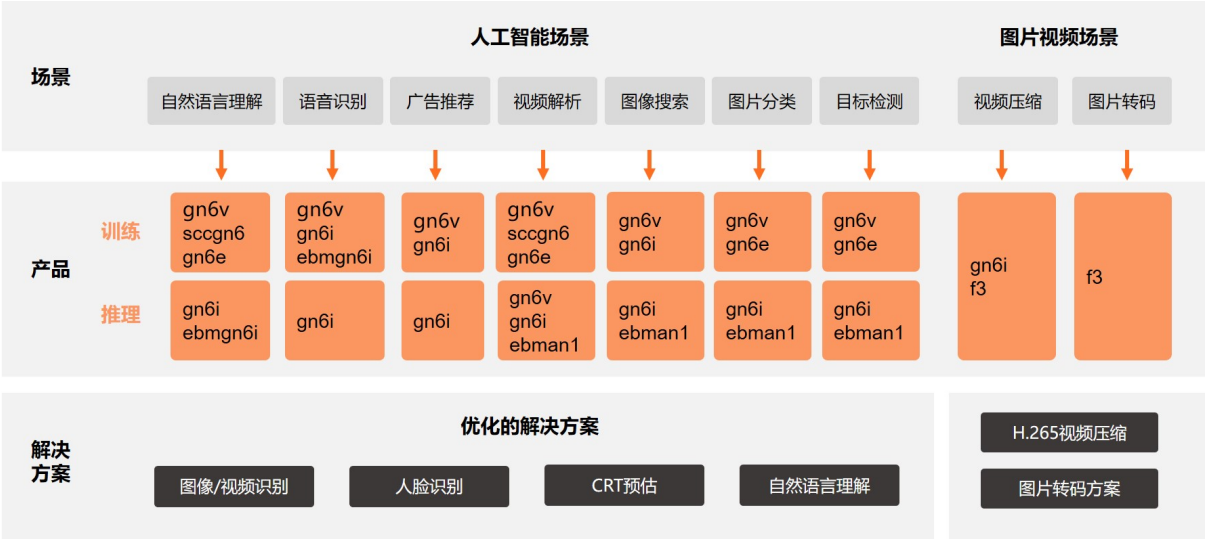


深度学习、图像处理场景推荐

在该类场景中，应用需要高性能的GPU加速器，在GPU和CPU配比方面有如下建议。

- 深度学习训练：GPU与CPU比例推荐为1:8到1:12之间。
- 通用深度学习：GPU与CPU比例推荐为1:4到1:48之间。
- 图像识别推理：GPU与CPU比例推荐为1:4到1:12之间。
- 语音识别与合成推理：GPU与CPU比例推荐为1:16到1:48之间。

常见场景的GPU选型推荐如下图所示。



验证与调整

当您完成选型并开始使用云服务器ECS实例后，建议您根据一段时间的性能监控信息，验证所选实例规格是否合适。

假设您选择了ecs.g6e.xlarge，通过监控发现实例CPU使用率一直较低，建议您登录实例检查内存占用率是否较高，如果内存占用较高，您可以调整为处理器与内存资源配比更合适的规格族。更多详情，请参见以下文档：

- ECS自带监控服务
- 查看云盘监控信息
- 概览

使用云服务器ECS的过程中，如果发生地域中库存不足、实例规格族停售、修改为更高性价比规格族、升级配置等情况，您可以根据实例规格族的特点进行变配。更多信息，请参见升降配方式概述与支持变配的实例规格。

相关视频

3.ECS实例交付（创建）方式

云服务器ECS提供单台交付、批量交付、高可用部署、自动化创建集群等多种ECS实例交付（创建）方式，支持控制台操作和API调用，满足您在不同场景下的ECS实例创建需求。

手动创建单台或多台实例

适用场景：批量创建具有相同实例规格、可用区、付费模式等配置的ECS实例。

创建方式：

- 使用控制台：
  - 使用向导创建实例

    在向导页面选择配置，可视化界面，操作简单。
  - 使用自定义镜像创建实例

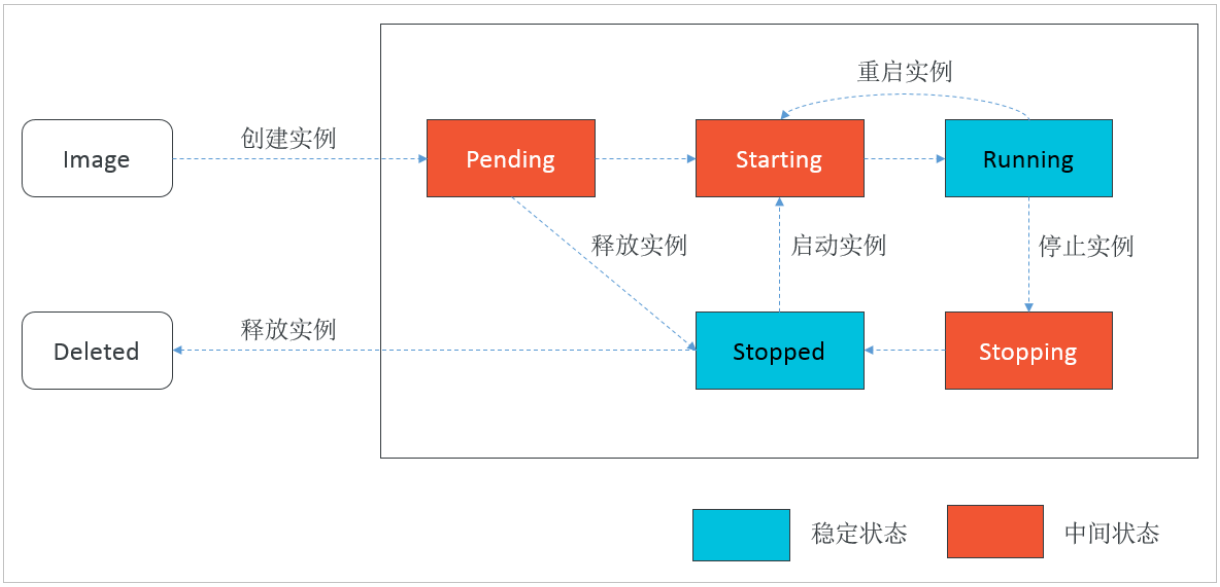
    使用账号中的自定义镜像创建实例，在向导页面中选择其它配置。
  - 购买相同配置的实例

    使用已有实例的配置创建实例，在向导页面中确认配置。
  - 使用实例启动模板创建实例

    使用启动模板创建实例，在向导页面中确认配置。
- 使用API RunInstances：
  - RunInstances
  - 批量创建ECS实例

创建数量：控制台根据您的云服务器使用情况而定，RunInstances单次1~100台。

使用控制台和RunInstances创建ECS实例时，实例生命周期如下：



您也可以使用CreateInstance创建一台ECS实例，但创建完成后进入已停止（Stopped）状态，您必须手动启动ECS实例。

高可用打散部署实例（部署集）

适用场景：将ECS实例分散部署到不同的物理机上，适合为具有高可用和底层容灾要求的应用提供算力。

创建方式：先创建部署集，然后在创建ECS实例时指定部署集。创建ECS实例时可通过控制台、RunInstances或CreateInstance。

创建数量：视创建方式而定，控制台和RunInstances单次1～20台，CreateInstance单次1台。

使用限制：

- 每个部署集在单个可用区下最多创建20台ECS实例。
- 仅支持特定的ECS实例规格，具体说明请参见部署集概述。
- 付费模式支持包年包月和按量付费，不支持抢占式实例。

详细操作：

- 使用控制台：
  i. 创建部署集
  ii. 在部署集内创建ECS实例
- 使用API：
  i. CreateDeploymentSet
  ii. RunInstances或者CreateInstance

自动化低成本弹性创建实例集群（弹性供应）

适用场景：一键部署跨付费模式、跨可用区和跨实例规格的实例集群。适合需要快速交付稳定算力，同时使用抢占式实例降低成本的场景。

创建方式：创建弹性供应组，由弹性供应组自动批量创建ECS实例。

创建数量：单个弹性供应组1~1000台ECS实例。

使用限制：付费模式支持按量付费和抢占式实例，不支持包年包月。

详细操作：

- 使用控制台：创建弹性供应组
- 使用API：CreateAutoProvisioningGroup
- 使用API批量创建的最佳实践：使用弹性供应组API批量创建ECS实例

自动化弹性创建和释放实例（弹性伸缩）

适用场景：持续维护跨付费模式、跨可用区、跨实例规格的实例集群。适合业务负载存在峰谷波动的场景。

创建方式：创建伸缩组和触发任务，由伸缩组自动批量创建或释放ECS实例。

创建数量：

- 单次伸缩活动最多创建1000台ECS实例。
- 单个伸缩组最多支持1000台ECS实例。

使用限制：自动创建ECS实例的付费模式支持按量付费和抢占式实例。支持将已有包年包月实例手动添加至伸缩组，但不支持在伸缩组内自动创建包年包月实例。

详细操作：

- 使用控制台：
  i. 快速扩缩容ECS实例
  ii. 实现自动扩张或者实现自动收缩

- 使用API:
  i. CreateScalingGroup
  ii. CreateScalingConfiguration
  iii. CreateScalingRule
  iv. CreateScheduledTask

弹性伸缩还支持更多便捷功能，提高交付效率，缩短算力需求出现和算力投入使用之间的流程。例如为ECS实例自动关联SLB实例和RDS实例，配置生命周期挂钩用于对ECS实例进行自定义操作等。您可以基于弹性伸缩实现贴合您业务需求的极致弹性，最佳实践示例请参见：

- 搭建可自动伸缩的Web应用
- 利用弹性伸缩降低成本
- 部署高可用计算集群

4.批量设置有序的实例名称或主机名称

您可以通过ECS控制台或者调用API RunInstances创建多台ECS实例。在创建多台ECS实例时，自定义设置实例名称或者主机名称可以帮助您更好地管理实例。本文介绍如何批量设置有序的实例名称或主机名称。

背景信息

批量配置有序名称，支持指定排序和自动排序两种方式。

本文通过四个场景示例，分别介绍通过ECS控制台和API如何配置三台实例的有序实例名称和主机名称。

- ECS控制台：
  - 场景一：设置三台实例名称或主机名称按指定排序（ECS控制台）
  - 场景二：设置三台实例名称或主机名称自动排序（ECS控制台）
- API:
  - 场景三：设置三台实例名称或主机名称按指定排序（API RunInstances）
  - 场景四：设置三台实例名称或主机名称自动排序（API RunInstances）

如果您想直接查看具体的配置规则，请参见名称规则、指定排序和自动排序。

场景一：设置三台实例名称或主机名称按指定排序（ECS控制台）

本场景主要描述在ECS控制台，配置按照指定数值排序的实例名称或者主机名称。其他配置信息，请参见使用向导创建实例。

1. 前往实例创建页。
2. 完成**基础配置**和**网络和安全组**配置。

   本示例在**基础配置**页签创建的实例数量为3台。
3. 在**系统配置**，完成系统配置项。

   在实例名称和主机名处，输入格式为*name_prefix[begin_number,bits]name_suffix*的指定排序。指定排序的具体规则，请参见指定排序。

   本示例指定名称以k8s-node-开头，从0006开始排序，主机名以-ecshost结尾。将实例名称配置为k8s-node-[6,4]，将主机名配置为k8s-node-[6,4]-ecshost。

   > **说明** 本示例仅用于指定排序，此处不勾选有序后缀。

   指定排序示例图

4. 完成**分组配置**，并确认订单。

   您可以在**实例列表**查看新增的实例。按照本文示例，生成的实例名分别为k8s-node-0006、k8s-node-0007、k8s-node-0008，生成的主机名分别为k8s-node-0006-ecshost、k8s-node-0007-ecshost、k8s-node-0008-ecshost。

场景二：设置三台实例名称或主机名称自动排序（ECS控制台）

本场景主要描述在ECS控制台，配置自动排序的实例名称或者主机名称。其他配置信息，请参见使用向导创建实例。

1. 前往实例创建页。
2. 完成**基础配置**和**网络和安全组**。

   本示例在**基础配置**页签创建的实例数量为3台。

3. 在**系统配置**，完成系统配置项。

   勾选有序后缀，系统会对实例名称和主机名自动排序，增加的后缀从001开始，按实例数量依次递增。自动排序的具体规则，请参见自动排序。

   本示例将实例名称配置为ecs，将主机名配置为ecshost。

   自动排序示例图

4. 完成**分组配置**，并确认订单。

   您可以在**实例列表**查看新增的实例，按照本文示例，生成的实例名分别为ecs001、ecs002、ecs003，生成的主机名分别为ecshost001、ecshost002、ecshost003。

场景三：设置三台实例名称或主机名称按指定排序（API RunInstances）

以下内容主要描述指定排序名称的参数配置，其他参数信息，请参见RunInstances。

InstanceName和HostName指定排序的配置格式为*name_prefix[begin_number,bits]name_suffix*。指定排序的具体规则，请参见指定排序。

本示例创建三台实例，实例名称和主机名称以k8s-node-开头，从0006开始排序，主机名以-ecshost结尾。具体参数配置如下：

- Amount：*3*
- InstanceName：*k8s-node-[6,4]*
- HostName：*k8s-node-[6,4]-ecshost*

> **说明** 本示例仅用于指定排序，此处UniqueSuffix保持默认不开启。

按照本文示例，生成的实例名分别为k8s-node-0006、k8s-node-0007、k8s-node-0008，生成的主机名分别为k8s-node-0006-ecshost、k8s-node-0007-ecshost、k8s-node-0008-ecshost。

场景四：设置三台实例名称或主机名称自动排序（API RunInstances）

以下内容主要描述自动排序名称的参数配置，其他参数信息，请参见RunInstances。

UniqueSuffix配置为*true*，系统会对InstanceName和HostName自动排序，增加的后缀从001开始，按实例数量依次递增。自动排序的具体规则，请参见自动排序。

本示例创建三台自动排序实例，具体参数配置如下：

- Amount：*3*
- InstanceName：*ecs*
- HostName：*ecshost*
- UniqueSuffix：*true*

按照本文示例，生成的实例名分别为ecs001、ecs002、ecs003，生成的主机名分别为ecshost001、ecshost002、ecshost003。

名称规则

- 实例名称：长度为2～128个字符，以大小写字母或中文开头，可包含数字、点号（.）、下划线（_）、半角冒号（:）或连字符（-）。
- 主机名：
  - Windows系统：长度为2～15个字符，允许使用大小写字母、数字或连字符（-）。不能以连字符（-）开头或结尾，不能连续使用连字符（-），也不能仅使用数字。
  - 其他操作系统（Linux等）：长度为2～64个字符，允许使用点号（.）分隔字符成多段，每段允许使用大小写字母、数字或连字符（-），但不能连续使用点号（.）或连字符（-）。不能以点号（.）或连字符（-）开头或结尾。

指定排序

参数格式为*name_prefix[begin_number,bits]name_suffix*。

参数说明表

字段名称	配置说明	示例

字段名称	配置说明	示例
name_prefix	指定实例名称或者主机名称的前缀。 **说明** 在有序命名规则中，前缀是必选项，否则当作普通名称处理。	k8s-node-
[begin_number,bits]	指定实例名称或者主机名称的有序数值。设置后，实例的名称数值会依次递增。 • begin_number：有序数值的起始值，取值支持[0,999999]，默认值为*0*。 • bits：有序数值所占的位数，取值支持[1,6]，默认值为*6*。 **注意** • *[begin_number,bits]*字段中不能有空格。 • 当指定的*begin_number*位数大于*bits*的取值时，*bits*默认为*6*。 • 相同前后缀的实例名称或主机名称最大支持999999台ECS实例。超过部分的ECS实例都使用*999999*。	[0,6]
name_suffix	指定实例名称或者主机名称的后缀。	-ecshost

参数示例表

输入参数示例	生成名称（以3台ECS实例为例）
k8s-node-[]-ecshost或k8s-node-[,]-ecshost	k8s-node-000000-ecshost、k8s-node-000001-ecshost、k8s-node-000002-ecshost
k8s-node-[99]-ecshost或k8s-node-[99,]-ecshost	k8s-node-000099-ecshost、k8s-node-000100-ecshost、k8s-node-000101-ecshost
k8s-node-[99,1]-ecshost	k8s-node-000099-ecshost、k8s-node-000100-ecshost、k8s-node-000101-ecshost
k8s-node-[999998]-ecshost	k8s-node-999998-ecshost、k8s-node-999999-ecshost、k8s-node-999999-ecshost
k8s-node-[0,4]	k8s-node-0000、k8s-node-0001、k8s-node-0002

自动排序

在创建多台实例时，您可以选择开启自动排序功能，为实例名称和主机名称自动添加有序后缀。有序数字后缀从001开始递增，最大不能超过999。

说明 自动排序功能默认关闭。

参数示例表

命名格式（实例名称或主机名）	输入参数示例	生成名称（以3台ECS实例为例）
普通名称	ecs	ecs001、ecs002、ecs003
指定排序*name_prefix[begin_number,bits]name_suffix*	k8s-node-[]-ecshost或k8s-node-[,]-ecshost	k8s-node-000000-ecshost001、k8s-node-000001-ecshost002、k8s-node-000002-ecshost003 **说明** 指定排序和自动排序同时生效。
指定排序*name_prefix[begin_number,bits]*	k8s-node-[0,4]	k8s-node-0000、k8s-node-0001、k8s-node-0002 **说明** 指定排序格式未设置命名后缀*name_suffix*，自动排序不生效。

5.使用弹性供应组API批量创建ECS实例

在需要大批量创建按量付费实例的场景中，通过API完成创建操作更加高效。其中，使用RunInstances完成该需求较为复杂，本文将推荐您使用交付过程更加方便稳定的CreateAutoProvisioningGroup。

背景信息

在业务需要使用按量付费ECS实例的场景下，RunInstances是使用最频繁的API。RunInstances拥有一次调用能够最多创建100台ECS实例的能力，但是在实际的生产环境中，如果需要超过100台的大批量创建ECS实例场景，直接使用RunInstances会存在一定的技术瓶颈。更多信息，请参见RunInstances创建实例时存在的问题。

> **说明** 如果您已了解RunInstances批量创建实例过程中存在的技术瓶颈，可以跳过该章节。

为了解决大批量创建ECS实例的需求场景，阿里云提供了弹性供应组，您可以通过CreateAutoProvisioningGroup创建弹性供应组，一键式的部署跨计费方式、跨可用区、跨实例规格族的实例集群。相较于RunInstances，CreateAutoProvisioningGroup更适合大批量创建ECS实例的业务场景。两者的功能对比与优势分析，请参见RunInstances与CreateAutoProvisioningGroup功能对比以及弹性供应组的优势。

RunInstances与CreateAutoProvisioningGroup功能对比

本章节对比RunInstances与CreateAutoProvisioningGroup两接口的部分功能，使您可以快速了解两者的差异，选择合适的创建实例方式。

对比项	RunInstances	CreateAutoProvisioningGroup
单次批量创建实例的数量上限	100台	1000台（vCPU上限为10000）
容量交付方式	实例数量	实例数量、vCPU核数、实例规格的权重等
是否支持多可用区	否	是
是否支持多个实例规格	否	是
是否支持多种磁盘规格	否	是

对比项	RunInstances	CreateAutoProvisioningGroup

是否提供了创建实例的策略	否	是。提供了如下策略：
		• 按量付费实例
		  ○ 成本优化策略：从备选实例规格中选取成本最低的实例规格，创建实例。
		  ○ 优先级策略：按照备选实例规格设置的优先级，依次尝试创建实例。
		• 抢占式实例
		  ○ 成本优化策略：从备选实例规格中选取成本最低的实例规格，创建实例。
		  ○ 可用区均衡分布策略：在备选的可用区之间，数量均匀的创建实例。
		  ○ 容量优化分布策略：根据抢占式实例的库存情况，选择最优的实例规格及可用区进行创建实例。

交付稳定性	受资源库存影响较大	多可用区、多实例规格的配置组合有效降低了资源库存造成的影响
API响应格式	同步返回创建结果	同步返回创建结果

创建实例的方式由RunInstances更换为CreateAutoProvisioningGroup的部分示例场景：

- 如果您之前使用RunInstances在单可用区、单实例规格的配置下批量创建实例，更换为CreateAutoProvisioningGroup后，您只需配置一组实例规格与可用区的组合，即可实现批量创建实例。
- 如果您之前使用RunInstances时手动设置了业务部署方案，更换为CreateAutoProvisioningGroup后，将由系统为您提供一键式的多可用区、多实例规格、多磁盘配置的部署能力，并且系统提供了多种创建实例的策略供您选择。

  例如：您之前手动设置了遍历多个实例规格及可用区的方案进行RunInstances调用，以提高实例创建的成功率。更换为CreateAutoProvisioningGroup后，您只需要通过参数配置多个实例规格及可用区的组合，选择合适的创建策略，系统将自动完成批量创建实例的操作。

> **注意** 弹性供应组的创建策略存在使用限制，单次最大可创建1000台实例，如果指定了实例规格的权重（`WeightedCapacity`），则单次创建的最大加权容量为10000。

RunInstances创建实例时存在的问题

基于RunInstances功能的限制，您在大批量创建实例时，可能遇到下表所示的问题。

问题	说明	解决方案
批量创建的能力有限	调用一次RunInstances最多可以创建100台ECS实例。	当您需要创建大于100台ECS实例时，需要通过循环或并发的方式多次调用该接口，以完成业务需求。
批量创建的稳定性不足	调用RunInstances只支持设置单可用区、单实例规格。因此您在批量创建ECS实例的过程中，可能因为实例规格的库存不足、停止售卖或使用限制等问题。引发以下情况： • 在某一时间段，实例规格的库存不足导致批量创建失败。 • 在某一时间段，实例规格停止售卖导致无法再创建指定的实例规格。 • 指定的实例规格只在部分可用区售卖。 • 指定的实例规格只能搭配指定的磁盘类型。	• 库存问题是导致批量创建ECS实例失败的主要原因。因此阿里云会推荐您在批量创建ECS实例之前，先调用DescribeAvailableResource查询实例规格与可用区下资源的库存情况，手动确认多个库存充足的可用区与实例规格的组合后，再批量创建ECS实例。通过复杂的创建方式，换来了较高的业务交付稳定性。 示例场景：当您确认多个库存充足的可用区与实例规格的组合后，您还需要构建合适的创建ECS实例的策略。例如，您可以根据手动确认的多个组合顺序，依次创建100台ECS实例，如果第一个组合的资源库存只支持创建50台ECS实例，那么您需要使用第二个组合尝试创建其余50台ECS实例。 • 实例规格存在使用限制。您可以通过DescribeAvailableResource查询限制，并自行建立容错方案，避免因使用限制变更带来的影响。 **说明** 您也可以根据文档提供的实例规格特点确定相关限制。更多信息，请参见实例规格族。 示例场景：`ecs.g6e.large`实例规格只支持ESSD云盘类型、`cn-beijing-x`可用区下不支持选择ESSD云盘类型等。

问题	说明	解决方案
创建策略过于单一	RunInstances仅支持设置单可用区、单实例规格。如果您的业务需要多可用区部署实现异地容灾、需要按照最低成本创建ECS实例等，则需要您自行构建业务部署方案，以保障实例的成功部署。自行构建的业务部署方案存在以下问题： • 开发成本高。自行构建的业务部署方案需要处理一系列的问题。例如，库存不足时如何顺利的创建ECS实例、扩容服务器时如何在获取抢占式实例最低成本的同时保证计算能力等 • 稳定性与专业性不足。对于阿里云提供的资源，您难以用专业的方式自行构建业务部署方案，并无法对方案进行测试，进而将对生产环境造成一定的风险。	自行解决或联系阿里云提供帮助。

弹性供应组的优势

针对RunInstances批量创建ECS实例存在的问题，阿里云提供了弹性供应组，解决了大批量创建ECS实例的场景下存在的问题。弹性供应组支持一键部署跨计费方式、跨可用区、跨实例规格族的实例集群。您可以通过弹性供应组稳定提供计算力，缓解抢占式实例的回收机制带来的不稳定因素，免去重复手动创建实例的繁琐操作。本章节主要介绍弹性供应组的优势。

优势	说明
批量创建ECS实例的数量上限更高	弹性供应组支持单次创建最多1000台ECS实例。
支持设置多可用区、多实例规格、多种磁盘类型	弹性供应组支持您配置最多10种实例规格或可用区的组合、最多5种磁盘类型的选择，帮助您实现高可用的批量创建ECS实例。 示例场景： 当您通过弹性供应组提供的均衡可用区分布策略创建ECS实例时，可以配置多个可用区和多个实例规格。按照策略的要求，多个可用区下，创建实例的数量应相对平均，但如果其中某个可用区无法完成创建，弹性供应组会尝试将该可用区待创建的实例数量，转移到其他可用区进行创建。 如果您指定了多种磁盘规格，弹性供应组将按照指定顺序作为各磁盘类型的优先级顺序，当某一种磁盘不可用时，自动更换磁盘类型。 **说明** 当所有磁盘类型都不可用时，系统将会自动更换其它创建方式，不再尝试该种创建方式。

优势	说明

支持多种创建实例的策略	针对按量付费实例和抢占式实例，分别提供了以下创建策略：
	• 按量付费实例
	  ○ 成本优化策略：从备选实例规格中选取成本最低的实例规格，创建实例。
	  ○ 优先级策略：按照备选实例规格设置的优先级，依次尝试创建实例。
	• 抢占式实例
	  ○ 成本优化策略：从备选实例规格中选取成本最低的实例规格，创建实例。
	  ○ 可用区均衡分布策略：在备选的可用区之间，数量均匀的创建实例。
	  ○ 容量优化分布策略：根据抢占式实例的库存情况，选择最优的实例规格及可用区进行创建实例。

可提高抢占式实例的可用性	抢占式实例因其价格优势使用量越来越高，但是其价格的不稳定性与系统回收的特性，造成管理抢占式实例存在一定的难度。您可以通过弹性供应组，实现在低成本的前提下，提高抢占式实例的可用性。具体方式如下： • 创建策略选择默认的成本优化策略，每次的扩容策略将按照实例规格价格从低到高的顺序尝试创建。 • 抢占式实例对应的不同实例规格与可用区的资源库存情况互相隔离。多个实例规格与多个可用区的配置组合，可以有效降低所有组合都无库存的概率。 • 创建弹性供应组时，配置多种备选的磁盘类型，保证创建实例的过程中，系统能够自动选取合适的磁盘类型。 • 配置 `SpotInstancePoolsToUseCount` 参数，指定抢占式实例在多个最低价格的实例规格及可用区的组合中创建。避免某一种实例规格对应的实例回收，造成计算能力产生雪崩效应。

CreateAutoProvisioningGroup最佳实践

本章节提供CreateAutoProvisioningGroup接口对应的Java代码示例，使您快速了解该接口的使用方式。

1. 安装ECS Java SDK以及阿里云核心库。

   具体操作，请参见安装Java SDK。

2. 编写调用CreateAutoProvisioningGroup接口的Java代码。

   代码示例如下：

```
CreateAutoProvisioningGroupRequest request = new CreateAutoProvisioningGroupRequest();
request.setRegionId(regionId);
request.setLaunchConfigurationImageId(RequestHelper.IMAGE_ID);
request.setLaunchConfigurationSecurityGroupId(securityGroupId);
request.setTotalTargetCapacity(totalTargetCapacity);
request.setPayAsYouGoTargetCapacity(payAsYouGoTargetCapacity);
request.setSpotTargetCapacity(spotTargetCapacity);
request.setLaunchConfigurationSystemDiskCategory("cloud_ssd");
request.setLaunchConfigurationSystemDiskSize(40);
request.setAutoProvisioningGroupType("instant");
// 设置抢占式实例的创建策略
request.setSpotAllocationStrategy("lowest-price");
request.setSpotInstancePoolsToUseCount(spotInstancePoolsToUseCount);
// 设置按量付费实例的创建策略
request.setPayAsYouGoAllocationStrategy("prioritized");
request.setMaxSpotPrice(maxSpotPrice);
// 多实例规格，多可用区配置信息，最大支持10种
request.setLaunchTemplateConfigs(launchTemplateConfigs);
request.setClientToken(clientToken);
CreateAutoProvisioningGroupResponse response = client.getAcsResponse(request);
```

JSON返回值示例如下：

```
{
    "autoProvisioningGroupId":"apg-****",
    "launchResults":[
        {
            "instanceIds":[
                "i-****"
            ],
            "instanceType":"ecs.c5.large",
            "spotStrategy":"NoSpot",
            "zoneId":"cn-shanghai-b"
        },
       {
            "instanceIds":[],
            "instanceType":"ecs.c5.large",
            "spotStrategy":"NoSpot",
            "zoneId":"cn-shanghai-b",
            "errorCode" : "Invalid.Parameter",
            "errorMsg" : "Specific Parameter 'imageId' is not valid"
        }
    ],
    "requestId":"20DA1E9F-BF7F-4BE7-8204-E4DE58E4FC7B"
}
```

通过CreateAutoProvisioningGroup创建弹性供应组时，您只需要设置批量创建实例的相关配置项，无需关心创建过程，弹性供应组将以尽力交付的方式，完成创建。

> **说明** 尽力交付的方式是指，当您配置的某些资源组合无法创建实例时，将自动切换到其他可用的资源组合继续进行创建。该方式创建实例需要一定的时间，并且可能导致实际创建结果与创建策略存在一定的偏差。

相关文档

- 弹性供应概述
- 弹性供应组设置示例

6.异构计算产品最佳实践概览

本文为您汇总了异构计算产品的最佳实践，您可以根据自身业务场景选择查看。

GPU云服务器

- 在GPU实例上部署NGC环境

  以搭建TensorFlow深度学习框架为例，介绍如何在GPU实例上部署NGC环境。

- GPU AI模型训练最佳实践

  适用于AI图片训练场景，使用CPFS/NAS作为共享存储，利用容器服务Kubernetes版管理GPU云服务器集群进行AI图片训练。

- 在GPU实例上使用RAPIDS加速机器学习任务

  在GPU实例上基于NGC环境使用RAPIDS加速库，加速数据科学和机器学习任务，提高计算资源的使用效率。

- RAPIDS加速机器学习最佳实践

  适用于使用RAPIDS加速库和GPU云服务器来对机器学习任务或者数据科学任务进行加速的场景。相比CPU，利用GPU和RAPIDS在某些场景下可以取得非常明显的加速效果。

- 在GPU实例上使用RAPIDS加速图像搜索任务

  使用RAPIDS加速图像搜索任务为例，介绍如何在预装镜像的GPU实例上使用RAPIDS加速库。

- RAPIDS加速图像搜索最佳实践

  适用于使用RAPIDS加速平台和GPU云服务器来对图像搜索任务进行加速的场景。相比CPU，利用GPU和RAPIDS在图像搜索场景下可以取得非常明显的加速效果。

神龙AI加速引擎AIACC

- 使用AIACC-Training（AIACC训练加速）加速BERT Finetune模型

  适用于自然语言训练场景，使用GPU云服务器和极速型NAS进行BERT Finetune模型训练，使用AIACC-Training（AIACC训练加速）可以有效提升多机多卡的训练效率。

集群极速部署工具FastGPU

- 使用FastGPU进行极速AI训练

  本教程利用FastGPU工具一键构建阿里云上的AI训练环境，并使用AIACC加速工具进行加速。

- 使用FastGPU一键部署并训练应用

  在开发者实验室中，阿里云为您提供了FastGPU训练场景的相关实验教程，您可以通过教程提供的真实环境，体验并完成所需教程的学习和实验。

GPU容器共享技术cGPU

- 使用ACK服务实现GPU成本优化

  适用于在利用阿里云容器服务ACK部署GPU集群后，出于成本优化的考虑，对于集群中GPU利用率不高的应用，使用GPU容器共享技术cGPU让一定数量的应用共享一张GPU卡，从而提高利用率。对于GPU利用率较高的应用，则不做改动。实现了灵活管理的同时降低整体成本。

FPGA云服务器

- FPGA RTL开发流程最佳实践
  - 使用f1 RTL

    介绍基于f1实例的RTL（Register Transfer Level）开发流程。
  - RTL工程目录介绍

    介绍RTL（Register Transfer Level）开发平台所使用的工程模式及目录，并提供示例框架帮助您理解并使用RTL。
  - f3实例RTL开发最佳实践

    介绍基于f3实例的RTL（Register Transfer Level）开发流程。
- FPGA OpenCL开发流程最佳实践
  - f1实例OpenCL开发最佳实践

    在f1实例上使用OpenCL（Open Computing Language）制作镜像文件，并烧录到FPGA芯片中。
  - f3 SDAccel开发环境介绍

    FaaS f3 SDAccel开发环境以Xilinx SDAccel dynamic 5.0版本为原型，您可以基于OpenCL进行开发以及应用。本文为您简要介绍f3实例的SDAccel开发环境。
  - f3实例OpenCL开发最佳实践

    在f3实例上使用OpenCL（Open Computing Language）制作镜像文件，并烧录到FPGA芯片中。

7.抢占式实例最佳实践

7.1. 查询抢占式实例当前价格

如果您是开发者，可参考本文提供的Java代码示例查询抢占式实例当前最新的价格。

前提条件

- 已准备阿里云账号以及对应的访问密钥（AccessKey）。

  使用Alibaba Cloud SDK for Java时需要设置阿里云账号的AccessKey信息。AccessKey的获取方式，请参见获取AccessKey。

- 已在开发环境中安装Java SDK。

  您需要在Maven项目中添加以下依赖。具体操作，请参见安装Java SDK。

```
<dependencies>
        <dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-java-sdk-ecs</artifactId>
            <version>4.23.10</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-java-sdk-core</artifactId>
            <version>4.0.8</version>
        </dependency>
        <dependency>
            <groupId>commons-lang</groupId>
            <artifactId>commons-lang</artifactId>
            <version>2.6</version>
        </dependency>
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.2.68</version>
        </dependency>
</dependencies>
```

代码示例

本文提供名为 `QuerySpotLatestPrice` 的示例类，代码中主要通过ECS的DescribePrice接口实现查询抢占式实例当前最新的价格功能。

```
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.ecs.model.v20140526.DescribePriceRequest;
import com.aliyuncs.ecs.model.v20140526.DescribePriceResponse;
import com.aliyuncs.profile.DefaultProfile;
/**
 * 通过DescribePrice查询最新价格。
 */
public class QuerySpotLatestPrice {
    private static IAcsClient client;
    // 请将regionId修改为您需要查询的地域ID。
    static String regionId = "cn-hangzhou";
    // 查询专有网络VPC类型的ECS实例。
    static String resourceType = "instance";
    static String instanceNetworkType = "vpc";
    // 请将instanceType修改为您需要查询的实例规格。
    static String instanceType = "ecs.g6.8xlarge";
    // 设置抢占策略为系统自动出价。
    static String spotStrategy = "SpotAsPriceGo";
    // 请将spotDuration修改为您需要保留抢占式实例的时长。不能确定保留时长时，请设置为0。
    static Integer spotDuration = 1;
    // 请将zoneId修改为您需要查询的可用区ID。
    static String zoneId = "cn-hangzhou-i";
    public static void main(String[] args) throws Exception {
        client = Initialization();
        describePrice(client);
    }
    public static void describePrice(IAcsClient client) throws Exception {
        // 设置DescribePrice参数，并向DescribePrice发送请求。
        DescribePriceRequest request = new DescribePriceRequest();
        request.setRegionId(regionId);
        request.setResourceType(resourceType);
        request.setInstanceType(instanceType);
        request.setInstanceNetworkType(instanceNetworkType);
        request.putQueryParameter("spotStrategy",spotStrategy);
        request.putQueryParameter("spotDuration",spotDuration);
        request.putQueryParameter("zoneId",zoneId);
        // 接收调用的返回结果，并输出查询到的抢占式实例当前最新价格。
        DescribePriceResponse describePriceResponse = client.getAcsResponse(request);
        System.out.println("抢占式实例价格："+describePriceResponse.getPriceInfo().getPrice()
.getTradePrice()+"元");
    }
    private static IAcsClient Initialization() {
        /**
         * 初始化请求参数。
         * 其中变量<your-access-key-id>需要设置为您的阿里云账号的AccessKey ID。
         * <your-access-key-secret>需要设置为您的阿里云账号的AccessKey Secret。
         */
        DefaultProfile profile = DefaultProfile.getProfile(regionId, "<your-access-key-id>"
, "<your-access-key-secret>");
        return new DefaultAcsClient(profile);
    }
}
```

查询结果的返回示例，如下图所示：

```
"C:\Program Files\Java\jdk1.8.0_211\bin\java.exe" ...
抢占式实例价格：7.77元

Process finished with exit code 0
```

7.2. 选择抢占式实例出价模式

抢占式实例存在多种出价模式，您可以根据实际情况进行选择。本文将对抢占式实例不同出价模式进行对比分析，并提供出价模式最佳实践。

背景信息

抢占式实例是一种按需实例，相对于按量付费实例价格有一定的折扣，旨在为您降低部分场景下使用ECS实例的成本。抢占式实例自身支持以下出价模式：

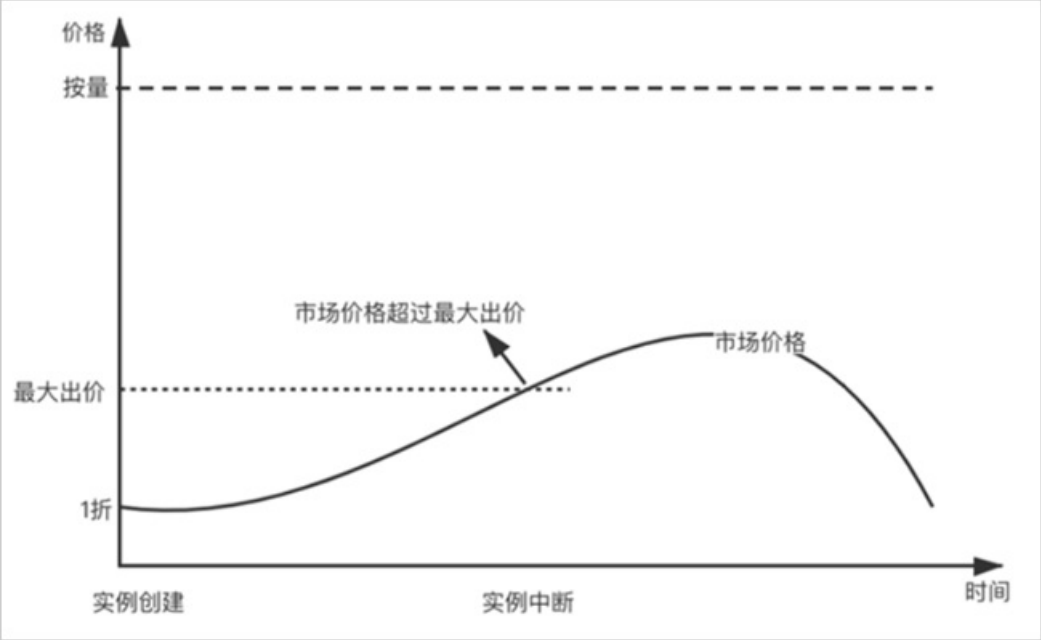
- 设定您的最高价（SpotWithPriceLimit）
- 使用自动出价（SpotAsPriceGo）

本教程还提供了使用自动出价（SpotAsPriceGo）结合运维编排服务OOS的`ACS-ECS-AlarmWhenDiscountAndPriceExceedsThresholdInMultiZoneAndInstanceType`公共模板的组合方案，为您提供成本保障的同时，降低抢占式实例的中断概率。

出价模式对比

- 模式一：设定您的最高价（SpotWithPriceLimit）

  该模式下需要您设置一个价格上限，即您愿意为这个实例规格支付的最高价格。如果实例规格在价格波动时超出您设置的价格上限，则对应的ECS实例将会触发中断事件。

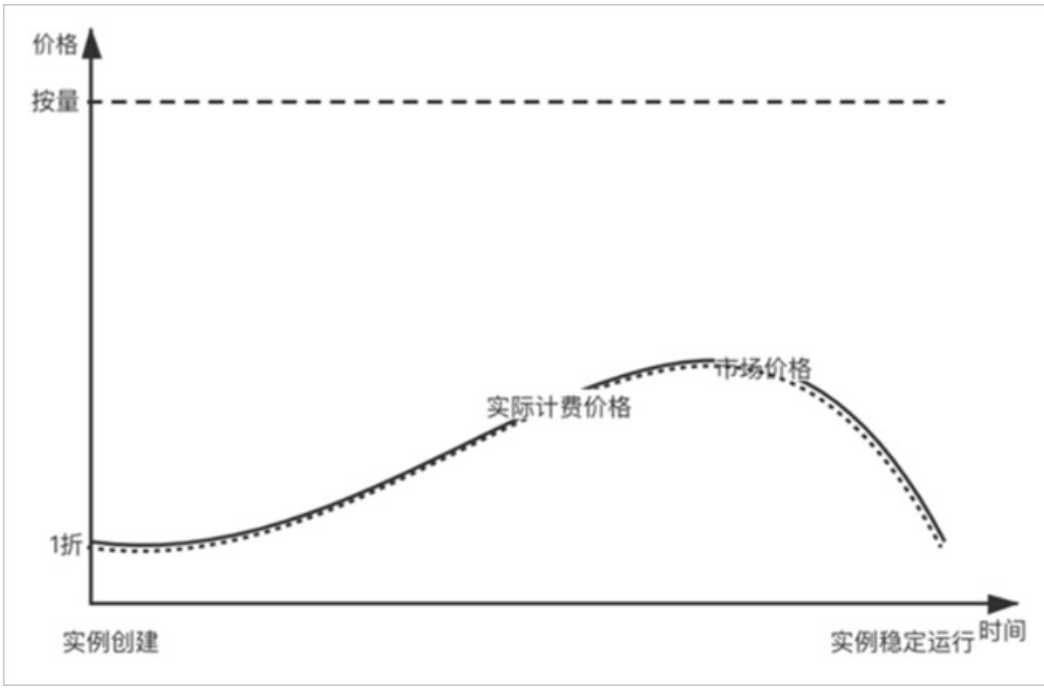


  对该模式的分析如下：

  - 优点：严格控制成本，能够保证实例规格的费用不会超出既定的成本。
  - 缺点：如果资源价格波动剧烈，则会提升触发实例中断事件的概率，降低实例的稳定性。
  - 适用场景：业务对ECS实例的预算以及价格要求极其严格，完全不允许超出预算。

- 模式二：使用自动出价（SpotAsPriceGo）

  该模式为跟随当前市场价格的模式，即表示您始终接受实时的市场价格作为实例规格的计费价格。

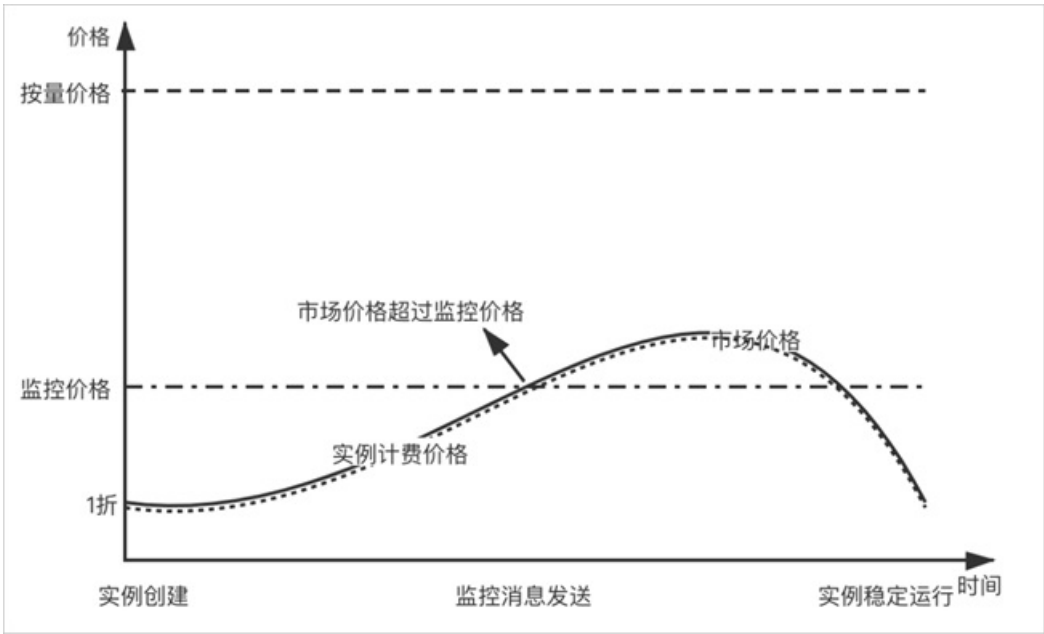


  对该模式的分析如下：

  - 优点：即使资源价格波动剧烈，也仍能保证实例不会被中断，降低了实例中断的概率，增加了实例的稳定性。
  - 缺点：较难控制成本，当资源价格上升时也无法感知到该信息，可能导致成本超支。
  - 适用场景：业务成本要求不严格，要求在提升实例稳定性的同时，尽可能的节省成本。

- 模式三：使用自动出价（SpotAsPriceGo）结合运维编排OOS

抢占式实例的使用自动出价（SpotAsPriceGo）模式无法直接感知到资源价格的上升，当结合运维编排OOS的 `ACS-ECS-AlarmWhenDiscountAndPriceExceedsThresholdInMultiZoneAndInstanceType` 公共模板后，OOS可以根据您自行设置的资源价格阈值，在抢占式实例价格超过阈值时向您推送通知，以提示您进行资源管理。



对该模式的分析如下：

- 优点：同时具备了资源稳定性以及资源价格上升的感知能力。
- 缺点：需要接入运维编排OOS，存在一定的接入成本。
- 适用场景：业务要求在提升实例稳定性的同时，具备感知成本增加的能力。

三种模式的比较如下表所示：

模式	实例被中断概率	实例稳定性	成本优化程度	成本可控性
设定您的最高价（SpotWithPriceLimit）	高	低	高	高
使用自动出价（SpotAsPriceGo）	低	高	较高	较低
使用自动出价（SpotAsPriceGo）结合运维编排OOS	低	高	较高	高

出价模式最佳实践

抢占式实例自身直接支持以下两种模式，您可以根据业务实际需求选择：

- 如果您需要严格控制预算，实例的稳定性要求不严格。可以选择设定您的最高价（SpotWithPriceLimit）的出价模式。
- 如果您对实例稳定性有较高要求，成本要求不严格。可以选择使用自动出价（SpotAsPriceGo）的出价模式。

如果您希望保障实例稳定性的同时，兼顾业务成本。可以参考以下操作步骤，使用 `SpotAsPriceGo+OOS` 的组合模式，即通过 **使用自动出价（SpotAsPriceGo）** 的出价模式提升实例的稳定性；通过运维编排OOS的 `ACS-ECS-AlarmWhenDiscountAndPriceExceedsThresholdInMultiZoneAndInstanceType` 公共模板监控抢占式实例的价格，当价格超过设置的阈值时，系统会向您发送提示消息。

1. 准备工作。
   i. （可选）创建出价模式为使用自动出价（SpotAsPriceGo）的抢占式实例。

   具体操作，请参见创建抢占式实例。如果您已经创建了抢占式实例，请跳过本步骤。

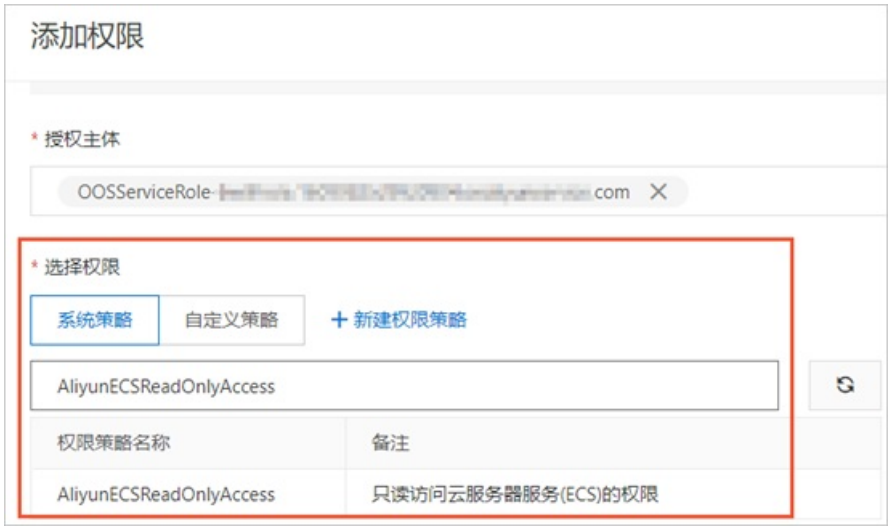
   ii. 查看抢占式实例的信息。

   具体操作，请参见查看实例信息。本教程中创建了两台抢占式实例，信息如下表所示，后续步骤将会基于实例的信息设置价格监控的运维脚本。

实例名称	地域和可用区	实例规格
抢占式实例1	华东1（杭州）可用区I	ecs.c5.xlarge
抢占式实例2	华东1（杭州）可用区K	ecs.r6.xlarge

   iii. 基于RAM访问控制创建OOS所需的 `OOSServiceRole` 角色。

   具体操作，请参见为OOS服务设置RAM权限。您需要注意，在**选择权限**时仅授予 `AliyunECSReadOnlyAccess` 权限即可。



iv. 接入钉钉自定义机器人。

具体操作，请参见自定义机器人接入。您需要注意在接入机器人时，**安全设置**选择**自定义关键词**，并设置 `监控` 关键词。后续步骤中，OOS将通过钉钉自定义机器人的Webhook地址发送消息。

2. 登录运维编排管理控制台。
3. 在左侧导航栏，单击**定时运维**。
4. 在**定时运维**页面，单击**创建**。
5. 在**创建定时运维**页面，完成以下配置，然后单击**立即执行**。

配置项	说明
定时设置	选择**周期性重复执行**，然后进行以下配置： ○ **重复频率**：保持默认配置，即1小时执行一次。 **说明** 抢占式实例短时间内价格变化不频繁，建议您以1小时为周期进行价格监控即可。 ○ **重复频率的时区**：根据您所在的地区时区自行设置。本教程中保持默认配置。 ○ **规则结束时间**：根据您实际的业务需求自行设置。本教程中保持默认配置。
选择模板	通过搜索框搜索并选中 `ACS-ECS-AlarmWhenDiscountAndPriceExceedsThresholdInMultiZoneAndInstanceType` 公共模板。

配置项	说明
设置参数	本步骤的参数设置需要参照已获取的抢占式实例信息进行设置。参数说明如下： ○ **region**：华东1（杭州）。 ○ **zoneId**：cn-hangzhou-i、cn-hangzhou-k。 ○ **instanceType**：ecs.c5.xlarge、ecs.r6.xlarge。 ○ **taskType**：您可以根据自身需求选择监控类型。具体说明如下： ■ Discount：基于实时折扣的监控。例如，某实例规格对应的按量付费价格为0.400，抢占式实例价格为0.080，则抢占式实例的折扣为2折。如果您期望抢占式实例在折扣大于2.5折时收到消息通知，则可以设置Discount，并设置threshold参数（阈值）为25（25表示2.5折）。 ■ Price：基于实时价格的监控。例如，某实例规格对应的按量付费价格为0.400，抢占式实例价格为0.080。如果您期望抢占式实例价格大于0.090时收到消息通知，则可以设置Price，并设置threshold参数（阈值）为0.090。 ○ **threshold**：设置阈值。需要结合taskType使用。 ○ **webhook**：设置钉钉自定义机器人的Webhook地址。 ○ **rateControl**：保持默认配置。 ○ **执行使用到的权限的来源**：选择已创建的 `OOSServiceRole` 角色。
高级选项	保持默认配置。您可以根据实际情况自行配置。

执行定时任务后，您可以在**定时运维**页面查看定时任务执行的状态。

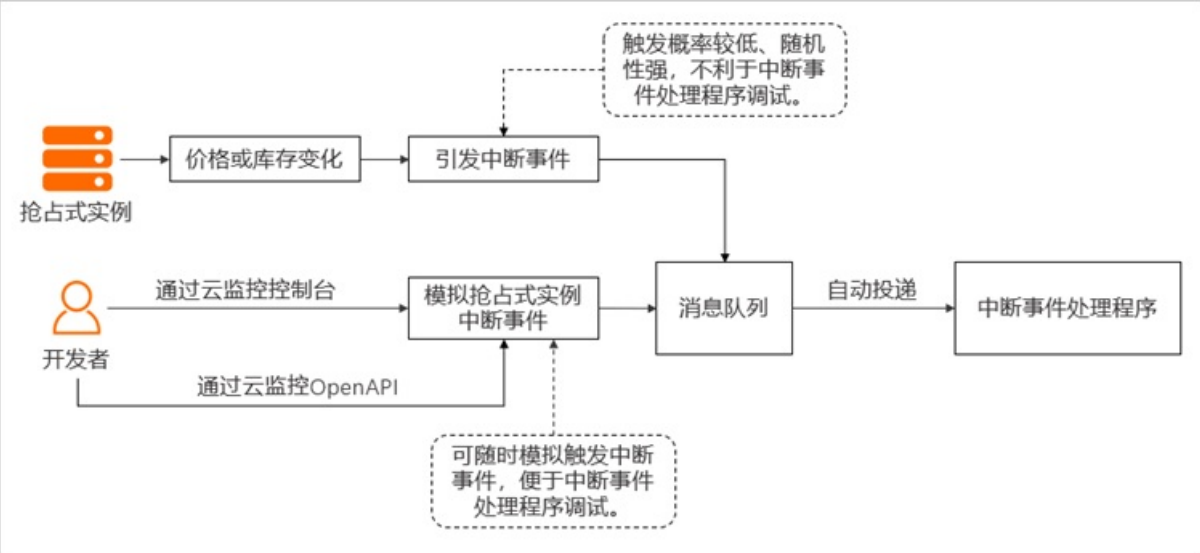
6. 等待定时任务执行或者测试定时任务。
   - 成功执行定时任务后，系统会实时监控抢占式实例的出价情况，如果超过您设置的阈值，则会通过钉钉自定义机器人发送提示消息。
   - 实际场景下，抢占式实例价格波动不频繁，不易验证定时任务的触发结果。您可以新建一个定时任务，并将阈值设置为必定触发的值。例如，抢占式实例每小时价格平均为0.080，阈值可以设置为0.040。因为抢占式实例价格始终高于阈值，所以后续定时任务执行时必定会触发报警并通过钉钉自定义机器人发送提示消息。

7.3. 模拟抢占式实例中断事件

抢占式实例的中断事件为被动触发事件，当您在开发抢占式实例中断事件处理程序过程中，无法有效地进行代码调试。因此阿里云提供了模拟抢占式实例中断事件的方式，便于您调试运维程序。

背景信息

模拟抢占式实例中断事件需要依赖阿里云云监控，您可以通过云监控控制台或者云监控API模拟中断事件。中断事件的触发机制如下图所示：



模拟抢占式实例中断事件的方式主要分为以下两种：

- 方式一：通过云监控控制台模拟中断事件
- 方式二：通过云监控OpenAPI模拟中断事件

方式一：通过云监控控制台模拟中断事件

本方式中，以云监控设置中断事件报警规则，并将事件投递到消息服务队列为例，介绍如何通过云监控控制台模拟中断事件。

1.
2.
3.
4.
5. 在**创建/修改事件报警**界面，完成以下配置，然后单击**确定**。
   - 报警规则名称：自定义名称。例如：抢占式实例中断事件报警。
   - 事件类型：选择**系统事件**。
   - 产品类型：选择**云服务器ECS**。
   - 事件类型：选择**状态通知**。
   - 事件等级：选择**警告**。
   - 事件名称：选择**抢占式实例中断通知**。
   - 资源范围：保持默认配置。
   - 报警方式：您需要根据实际情况，选择适用于您业务的报警方式。本示例中，选择消息服务队列。

   成功创建后，您可以在**报警规则**页签，查看已创建的规则。
6. 在抢占式实例中断事件的报警规则的**操作**列，单击**调试**。
7. 在**创建事件调试**界面，修改JSON文件，然后单击**确定**。

   您需要将JSON文件中资源相关的信息替换为待模拟中断事件的抢占式实例的信息。JSON文件内容如下所示。其中：

- *阿里云账号UID*变量需要替换为当前登录的阿里云账号UID。
- *<resource-id>*以及*i-abcdef*两个变量需要替换为抢占式实例的实例ID。
- *<地域ID>*变量需要替换为抢占式实例所属的地域ID。

```
{
    "product": "ECS",
    "resourceId": "acs:ecs:cn-shanghai:阿里云账号UID:instance/<resource-id>",
    "level": "WARN",
    "instanceName": "instanceName",
    "regionId": "<地域ID>",
    "groupId": "0",
    "name": "Instance:PreemptibleInstanceInterruption",
    "content": {
        "instanceId": "i-abcdef",
        "action": "delete"
    },
    "status": "Normal"
}
```

成功发起调试后，系统将根据您设置的报警方式推送报警信息。

方式二：通过云监控OpenAPI模拟中断事件

本方式中，使用Alibaba Cloud SDK for Java，以云监控设置中断事件报警规则，并将事件投递到消息服务队列为例，介绍如何通过云监控OpenAPI模拟中断事件。

1. 完成准备工作。
   i. 获取阿里云账号对应的AccessKey。

      具体操作，请参见获取AccessKey。

   ii. 在开发环境中安装Java SDK。

      您需要在Maven项目中添加以下依赖。具体操作，请参见Java SDK使用手册。

      ```
      <dependency>
        <groupId>com.aliyun</groupId>
        <artifactId>tea-openapi</artifactId>
        <version>0.0.13</version>
      </dependency>
      <dependency>
        <groupId>com.aliyun</groupId>
        <artifactId>cms20190101</artifactId>
        <version>1.0.1</version>
      </dependency>
      ```

2. 调用云监控的PutEventRule接口，创建抢占式实例中断事件报警规则。

   Java代码样例如下所示：

```
import com.aliyun.cms20190101.models.*;
import com.aliyun.teaopenapi.models.*;
/**
 * 调用PutEventRule创建或修改事件的报警规则。
 */
public class Sample {
    // 初始化请求参数。
    public static com.aliyun.cms20190101.Client createClient(String accessKeyId, String
accessKeySecret) throws Exception {
        //您的阿里云账号的AccessKey ID以及AccessKey Secret。
        Config config = new Config().setAccessKeyId(accessKeyId).setAccessKeySecret(acc
essKeySecret);
        // OpenAPI的接入点（Endpoint）。
        config.endpoint = "metrics.cn-hangzhou.aliyuncs.com";
        return new com.aliyun.cms20190101.Client(config);
    }
    public static void main(String[] args_) throws Exception {
        java.util.List<String> args = java.util.Arrays.asList(args_);
        /**
         * 变量<your-access-key-id>需要设置为您的阿里云账号的AccessKey ID。
         * <your-access-key-secret>需要设置为您的阿里云账号的AccessKey Secret。
         */
        com.aliyun.cms20190101.Client client = Sample.createClient("<your-access-key-id
>", "<your-access-key-secret>");
        PutEventRuleRequest.PutEventRuleRequestEventPattern eventPattern0 = new PutEven
tRuleRequest.PutEventRuleRequestEventPattern()
                // 设置事件报警规则的类型。
                .setEventTypeList(java.util.Arrays.asList("*"))
                // 事件报警规则的等级。
                .setLevelList(java.util.Arrays.asList("*"))
                // 事件报警规则的名称
                .setNameList(java.util.Arrays.asList("Instance:PreemptibleInstanceInter
ruption"))
                // 云服务类型
                .setProduct("ECS");
        PutEventRuleRequest putEventRuleRequest = new PutEventRuleRequest()
                // 自定义规则名称。
                .setRuleName("spot_release_event_test")
                .setEventPattern(java.util.Arrays.asList(eventPattern0))
                // 事件报警规则的类型。
                .setEventType("SYSTEM")
                // 事件报警规则的状态。
                .setState("ENABLED");
        //未打印返回结果，如有需要您可以自行打印。
        client.putEventRule(putEventRuleRequest);
    }
}
```

3. 通过阿里云消息服务MNS，创建消息队列。

   创建消息队列的代码样例，请参见步骤二：创建队列。

4. 调用云监控的PutEventRuleTargets接口，为已创建的报警规则设置报警方式，将消息投递至已创建的消息队列中。

Java代码样例如下所示：

```
import com.aliyun.cms20190101.models.*;
import com.aliyun.teaopenapi.models.*;
/**
 * 调用PutEventRuleTargets添加或修改报警规则的发送目标。
 */
public class Sample {
    // 初始化请求参数。
    public static com.aliyun.cms20190101.Client createClient(String accessKeyId, String
accessKeySecret) throws Exception {
        //您的阿里云账号的AccessKey ID以及AccessKey Secret。
        Config config = new Config().setAccessKeyId(accessKeyId).setAccessKeySecret(acc
essKeySecret);
        // OpenAPI的接入点（Endpoint）。
        config.endpoint = "metrics.cn-hangzhou.aliyuncs.com";
        return new com.aliyun.cms20190101.Client(config);
    }
    public static void main(String[] args_) throws Exception {
        /**
         * 变量<your-access-key-id>需要设置为您的阿里云账号的AccessKey ID。
         * <your-access-key-secret>需要设置为您的阿里云账号的AccessKey Secret。
         */
        com.aliyun.cms20190101.Client client = Sample.createClient("<your-access-key-id
>", "<your-access-key-secret>");
        PutEventRuleTargetsRequest.PutEventRuleTargetsRequestMnsParameters mnsParameter
s0 = new PutEventRuleTargetsRequest.PutEventRuleTargetsRequestMnsParameters()
                // 消息队列所属的地域。
                .setRegion("cn-hangzhou")
                // 规则发送目标的唯一标识。
                .setId("1")
                // 队列名称。
                .setQueue("mq-test");
        PutEventRuleTargetsRequest putEventRuleTargetsRequest = new PutEventRuleTargets
Request()
                // 指定报警规则的名称。
                .setRuleName("spot_release_event_test")
                .setMnsParameters(java.util.Arrays.asList(
                        mnsParameters0
                ));
        //未打印返回结果，如有需要您可以自行打印。
        client.putEventRuleTargets(putEventRuleTargetsRequest);
    }
}
```

5. 调用云监控的SendDryRunSystemEvent接口，发送模拟中断事件。

   Java代码样例如下所示：

```
import com.aliyun.cms20190101.models.*;
import com.aliyun.teaopenapi.models.*;
/**
 *调用SendDryRunSystemEvent调试云资源的系统事件。
 */
public class Sample {
    // 初始化请求参数。
    public static com.aliyun.cms20190101.Client createClient(String accessKeyId, String
accessKeySecret) throws Exception {
        //您的阿里云账号的AccessKey ID以及AccessKey Secret。
        Config config = new Config().setAccessKeyId(accessKeyId).setAccessKeySecret(acc
essKeySecret);
        // OpenAPI的接入点（Endpoint）。
        config.endpoint = "metrics.cn-hangzhou.aliyuncs.com";
        return new com.aliyun.cms20190101.Client(config);
    }
    public static void main(String[] args_) throws Exception {
        /**
         * 变量<your-access-key-id>需要设置为您的阿里云账号的AccessKey ID。
         * <your-access-key-secret>需要设置为您的阿里云账号的AccessKey Secret。
         */
        com.aliyun.cms20190101.Client client = Sample.createClient("<your-access-key-id
>", "<your-access-key-secret>");
        SendDryRunSystemEventRequest sendDryRunSystemEventRequest = new SendDryRunSyste
mEventRequest()
                // 云服务名称。
                .setProduct("ecs")
                // 报警事件名称。
                .setEventName("Instance:PreemptibleInstanceInterruption")
                // 设置模拟事件的内容。
                .setEventContent("{\"product\":\"ECS\",\"resourceId\":\"acs:ecs:cn-shan
ghai:133160284996****:instance/i-abcdef\",\"level\":\"WARN\",\"instanceName\":\"instanc
eName\",\"regionId\":\"cn-beijing\",\"name\":\"Instance:PreemptibleInstanceInterruption
\",\"content\": {\"instanceId\":\"i-abcdef****\",\"action\":\"delete\"},\"status\":\"No
rmal\"}");
        //未打印返回结果，如有需要您可以自行打印。
        client.sendDryRunSystemEvent(sendDryRunSystemEventRequest);
    }
}
```

6. 在队列中接收模拟事件的消息。

   接收消息的代码样例，请参见步骤四：接收和删除消息。

7.4. 接收抢占式实例中断事件

阿里云云监控提供强大的资源监控功能，您可以通过云监控实时监控抢占式实例的中断事件，并在发生事件报警时通过您指定的报警方式接收到抢占式实例中断事件。

背景信息

抢占式实例具有被中断的风险，实例在完全中断的前5分钟会触发中断事件。如果您的业务对实例中断敏感，则需要注意及时接收抢占式实例的中断事件，并对实例进行处理。本文主要提供了基于消息服务MNS或函数计算接收中断事件的最佳实践。

- 方式一：通过消息服务MNS接收中断事件
- 方式二：通过函数计算接收中断事件

除本文提供的接收抢占式实例中断事件最佳实践外，您也可以通过阿里云OpenAPI接收抢占式实例的中断事件。具体操作，请参见查询抢占式实例中断事件。

方式一：通过消息服务MNS接收中断事件

1. 通过消息服务MNS创建一个队列。

   具体操作，请参见创建队列。

2. 通过云监控创建事件报警。

   具体操作，请参见创建系统事件报警规则。本教程中，需要完成以下配置：

   - 报警规则名称：自定义名称。例如：抢占式实例中断事件报警。
   - 事件类型：选择**系统事件**。
   - 产品类型：选择**云服务器ECS**。
   - 事件类型：选择**状态通知**。
   - 事件等级：选择**警告**。
   - 事件名称：选择**抢占式实例中断通知**。
   - 资源范围：保持默认配置。
   - 报警方式：您需要根据实际情况，选择适用于您业务的报警方式。例如，选择**消息服务队列**并配置已创建的队列信息。

   成功创建后，您可以在**报警规则**页签，查看已创建的规则。

3. 在抢占式实例中断事件的报警规则的**操作**列，单击**调试**。

4. 在**创建事件调试**界面，修改JSON文件，然后单击**确定**。

   您需要将JSON文件中资源相关的信息替换为待模拟中断事件的抢占式实例的信息。JSON文件内容如下所示。其中：

   - *阿里云账号UID*变量需要替换为当前登录的阿里云账号UID。
   - *<resource-id>*以及*i-abcdef*两个变量需要替换为抢占式实例的实例ID。
   - *<地域ID>*变量需要替换为抢占式实例所属的地域ID。

   ```
   {
       "product": "ECS",
       "resourceId": "acs:ecs:cn-shanghai:阿里云账号UID:instance/<resource-id>",
       "level": "WARN",
       "instanceName": "instanceName",
       "regionId": "<地域ID>",
       "groupId": "0",
       "name": "Instance:PreemptibleInstanceInterruption",
       "content": {
           "instanceId": "i-abcdef",
           "action": "delete"
       },
       "status": "Normal"
   }
   ```

   成功发起调试后，系统将根据您设置的报警方式推送报警信息。

5. 通过消息服务MNS接收中断事件。

   具体操作，请参见接收消息。

   您也可以使用消息服务MNS SDK关联到实际的业务中。更多信息，请参见Java SDK版本说明。

方式二：通过函数计算接收中断事件

1. 创建函数计算服务。
   i. 登录函数计算控制台。
   ii. 在左侧导航栏，单击**服务及函数**。
   iii. 在顶部菜单栏，选择地域。
   iv. 在**服务列表**页面，单击**创建服务**。
   v. 在**创建服务**面板，填写服务名称和描述，然后单击**确定**。

      您也可以在**创建服务**面板，设置是否启用阿里云日志服务和是否启用阿里云链路追踪功能。详细信息，请参见管理服务。

   当您成功创建服务后，页面会跳转至该服务页面的**函数管理**页面。
2. 创建函数。
   i. 在**管理函数**页面，单击**创建函数**。
   ii. 在**创建函数**页面，选择**从零开始创建**。
   iii. 在**基本设置**区域，设置相关参数，然后单击**创建**。

      参数配置说明如下：

      - （可选）**名称**：自定义函数的名称。例如：testSpotInstance。
      - **运行环境**：选择**Python 2.7**。
      - **函数触发方式**：选择**通过事件触发**。
      - **实例类型**：选择**弹性实例**。
      - **内存规格**：设置函数执行内存为**512 MB**。

      当您成功创建函数后，页面会跳转至该函数详情页面的**函数代码**页签。
3. 配置函数代码。
   i. 在IDE开发环境中，找到并单击默认存在的**index.py**文件。

ii. 将**index.py**文件内容替换为以下代码，然后单击**部署代码**。

代码中的变量说明如下：

- *阿里云账号UID*变量需要替换为当前登录的阿里云账号UID。
- *<resource-id>*以及*i-abcdef*两个变量需要替换为抢占式实例的实例ID。
- *<地域ID>*变量需要替换为抢占式实例所属的地域ID。

```
# -*- coding: utf-8 -*-
import logging
import json, random, string, time
LOGGER = logging.getLogger()
clt = None
def handler(event, context):
  '''
{
    "product": "ECS",
    "resourceId": "acs:ecs:cn-shanghai:阿里云账号UID:instance/<resource-id>",
    "level": "WARN",
    "instanceName": "instanceName",
    "regionId": "<地域ID>",
    "groupId": "0",
    "name": "Instance:PreemptibleInstanceInterruption",
    "content": {
        "instanceId": "i-abcdef",
        "action": "delete"
    },
    "status": "Normal"
}
  '''
  evt = json.loads(event)
  content = evt.get("content");
  regionId = evt.get("regionId");
  instanceId = content.get("instanceId");
  LOGGER.info( regionId + " " + instanceId + " termination ongoing");
```

iii. 单击**测试函数**右侧的下拉列表，然后单击**配置测试参数**。

iv. 在**配置测试参数**对话框，完成以下配置并单击**确定**。

- 事件模板：保持默认配置。
- 事件名称：自定义名称。例如：抢占式实例中断事件。
- 文本框：将默认信息替换为以下代码，其中的变量信息需要与已配置的index.py文件内的变量信息保持一致。

```
{
    "product": "ECS",
    "resourceId": "acs:ecs:cn-shanghai:阿里云账号UID:instance/<resource-id>",
    "level": "WARN",
    "instanceName": "instanceName",
    "regionId": "<地域ID>",
    "groupId": "0",
    "name": "Instance:PreemptibleInstanceInterruption",
    "content": {
        "instanceId": "i-abcdef",
        "action": "delete"
    },
    "status": "Normal"
}
```

4. 在**函数代码**页签，单击**测试函数**。

   您可以通过测试函数功能，查看函数代码的示例输出值。例如，本示例中，输出结果如下图所示，其中输出了触发中断事件的抢占式实例所属地域以及实例ID信息。

5. 测试无问题后，通过云监控创建事件报警。

   具体操作，请参见创建系统事件报警规则。本教程中，需要完成以下配置：

   - 报警规则名称：自定义名称。例如：抢占式实例中断事件报警。
   - 事件类型：选择**系统事件**。
   - 产品类型：选择**云服务器ECS**。
   - 事件类型：选择**状态通知**。
   - 事件等级：选择**警告**。
   - 事件名称：选择**抢占式实例中断通知**。
   - 资源范围：保持默认配置。
   - 报警方式：您需要根据实际情况，选择适用于您业务的报警方式。例如，选择**函数计算**并配置已创建的函数计算服务信息。

   成功创建后，您可以在**报警规则**页签，查看已创建的规则。

6. 在抢占式实例中断事件的报警规则的**操作**列，单击**调试**。
7. 在**创建事件调试**界面，修改JSON文件，然后单击**确定**。

   您需要将JSON文件中资源相关的信息替换为待模拟中断事件的抢占式实例的信息。JSON文件内容如下所示。其中：

   - *阿里云账号UID*变量需要替换为当前登录的阿里云账号UID。
   - *<resource-id>*以及*i-abcdef*两个变量需要替换为抢占式实例的实例ID。

- *<地域ID>*变量需要替换为抢占式实例所属的地域ID。

```
{
    "product": "ECS",
    "resourceId": "acs:ecs:cn-shanghai:阿里云账号UID:instance/<resource-id>",
    "level": "WARN",
    "instanceName": "instanceName",
    "regionId": "<地域ID>",
    "groupId": "0",
    "name": "Instance:PreemptibleInstanceInterruption",
    "content": {
        "instanceId": "i-abcdef",
        "action": "delete"
    },
    "status": "Normal"
}
```

成功发起调试后，系统将根据您设置的报警方式推送报警信息。

7.5. 抢占式实例被回收时数据恢复最佳实践

7.5.1. 使用实例创建自定义镜像

抢占式实例可能会因为价格因素或者市场供需变化而被强制回收。本文将以Alibaba Cloud SDK for Java为例，介绍如何通过Java代码监控到抢占式实例被回收的中断事件后，系统创建整个实例的自定义镜像，您可以使用该镜像新建抢占式实例完成实例内的数据恢复。

前提条件

- 已准备阿里云账号以及对应的访问密钥（AccessKey）。

  使用Alibaba Cloud SDK for Java时需要设置阿里云账号的AccessKey信息。AccessKey的获取方式，请参见获取AccessKey。

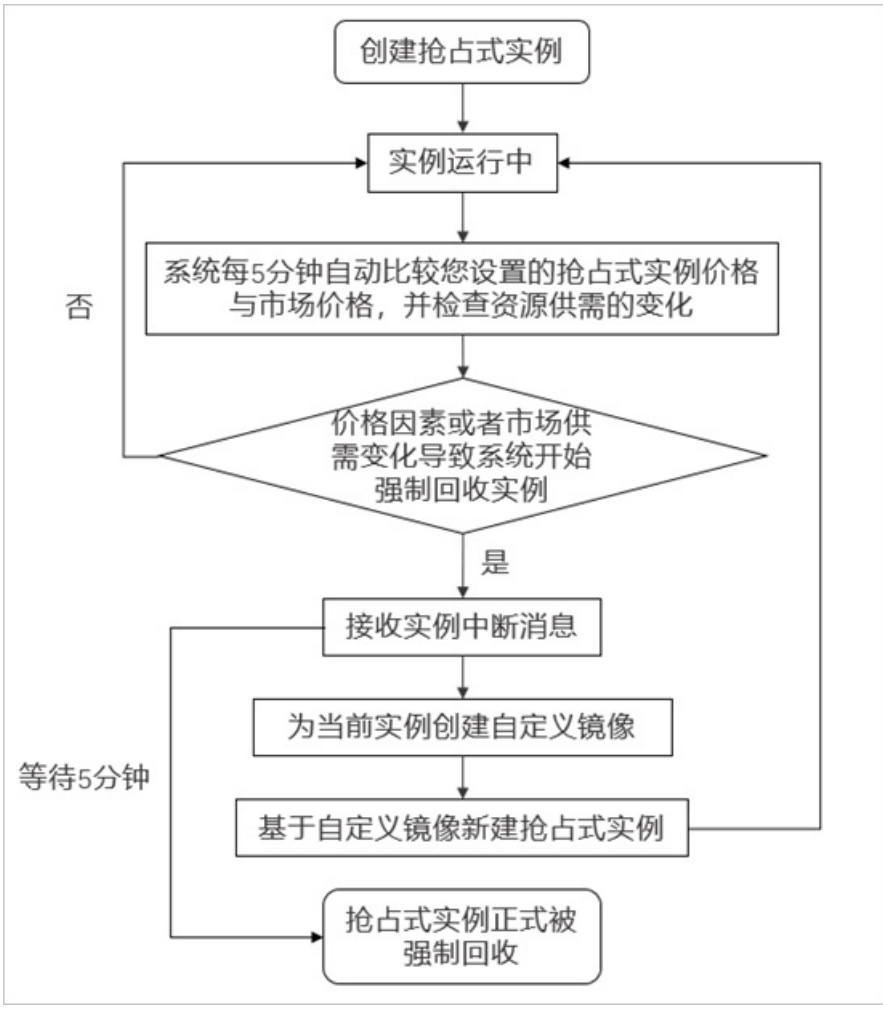
- 已在开发环境中安装Java SDK。

  您需要在Maven项目中添加以下依赖。具体操作，请参见安装Java SDK。

```
<dependencies>
       <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.2.68</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-java-sdk-ecs</artifactId>
            <version>4.23.10</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-java-sdk-core</artifactId>
            <version>4.0.8</version>
        </dependency>
</dependencies>
```

背景信息

您在使用抢占式实例时，实例可能会因为价格因素或者市场供需变化而被强制回收，在被完全回收前，实例会进入锁定状态，并触发抢占式实例的中断事件。您可以基于该事件设置监控机制，当接收到抢占式实例的中断事件后，通过Java代码自动为实例创建自定义镜像，并基于创建好的自定义镜像新建抢占式实例，以实现实例内的数据恢复。本文提供的示例场景中，运维工作流程图如下所示：



注意事项

> **注意** 本文提供的示例代码仅供参考，并不能保证您的实例一定会在5分钟内完成镜像的创建与数据的恢复。

抢占式实例会提前至少5分钟发送实例中断消息，但数据恢复的具体耗时取决于您实例的镜像类型与系统盘文件大小等因素。例如，系统盘文件越大，恢复时间越久。请您使用示例代码前务必自行进行评估与验证。

步骤一：创建抢占式实例

本步骤提供名为 `CreateSpotInstance` 的示例类，代码中主要通过ECS的RunInstances接口创建抢占式实例。

```
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.ecs.model.v20140526.RunInstancesRequest;
import com.aliyuncs.ecs.model.v20140526.RunInstancesResponse;
```

```
import com.aliyuncs.profile.DefaultProfile;
/**
 * 通过RunInstances创建抢占式实例。
 */
public class CreateSpotInstance {
    static IAcsClient client;
    // 指定地域ID。指定后您创建的ECS实例属于该地域内。
    static String regionId = "cn-hangzhou";
    // 指定可用区ID。指定后您创建的ECS实例属于该可用区内。
    static String zoneId = "cn-hangzhou-i";
    // 指定创建的ECS实例所使用的实例规格。
    static String instanceType = "ecs.s6-c1m1.small";
    // 指定创建的ECS实例所使用的镜像ID。
    static String imagesId = "centos_7_6_x64_20G_alibase_20211130.vhd";
    // 指定创建的ECS实例所属的交换机ID。
    static String vSwitchId = "<your-vSwitchId>";
    // 指定创建的ECS实例所属的安全组ID。
    static String securityGroupId = "<your-securityGroupId>";
    // 指定抢占策略。
    static String spotStrategy = "SpotAsPriceGo";
    // 修改您需要保留抢占式实例的时长。不能确定保留时长时，请设置为0。
    static Integer spotDuration = 0;
    // 指定ECS实例的登录密码。
    static String password = "<your-password>";
    public static void main(String[] args) {
        client = Initialization();
        createInstance();
    }
    private static IAcsClient Initialization() {
        /**
         * 初始化请求参数。
         * 其中变量<your-access-key-id>需要设置为您的阿里云账号的AccessKey ID。
         * <your-access-key-secret>需要设置为您的阿里云账号的AccessKey Secret。
         */
        DefaultProfile profile = DefaultProfile.getProfile(regionId,"<your-access-key-id>",
"<your-access-key-secret>");
        return new DefaultAcsClient(profile);
    }
    //创建实例。
    public static String createInstance() {
        try {
            // 设置RunInstances参数，发送请求。
            RunInstancesRequest request = new RunInstancesRequest();
            request.setRegionId(regionId);
            request.setZoneId(zoneId);
            request.setInstanceType(instanceType);
            request.setSpotDuration(spotDuration);
            request.setSpotStrategy(spotStrategy);
            request.setImageId(imagesId);
            request.setVSwitchId(vSwitchId);
            request.setSecurityGroupId(securityGroupId);
            // InstanceChargeType取值为PostPaid时才会生效抢占策略。
            request.setInstanceChargeType("PostPaid");
            request.setPassword(password);
```

```
            request.setInternetMaxBandwidthOut(1);
            // 接收调用的返回结果，并输出已创建的ECS实例ID。
            RunInstancesResponse response = client.getAcsResponse(request);
            if (null == response.getInstanceIdSets() || response.getInstanceIdSets().isEmpt
y()) {
                return null;
            }
            String instanceId = response.getInstanceIdSets().get(0);
            System.out.println("创建的实例ID：" + instanceId);
            return instanceId;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

步骤二：监控到中断事件后自动创建自定义镜像

本步骤提供名为 `CreateSpotImage` 的示例类，代码中依次调用了以下接口分别实现功能：

- 调用DescribeInstances监控抢占式实例的状态。
- 当监控到抢占式实例产生中断事件后，调用CreateImage为指定的抢占式实例创建自定义镜像。
- 创建自定义镜像后，调用DescribeImages监控自定义镜像的状态，当镜像变为可用状态时，返回提示信息。

```
import com.alibaba.fastjson.JSON;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.ecs.model.v20140526.*;
import com.aliyuncs.profile.DefaultProfile;
import java.util.ArrayList;
import java.util.List;
/**
 * 监控抢占式实例的中断事件。当中断事件发生时自动为实例创建自定义镜像。
 * 代码中将会调用以下ECS API：
 * DescribeInstances：查询实例信息
 * CreateImage：创建自定义镜像
 * DescribeImages：查询自定义镜像的状态
 */
public class CreateSpotImage {
    static IAcsClient client;
    // 请将regionId修改为您的抢占式实例所属的地域ID。
    static String regionId = "cn-hangzhou";
    // 抢占式实例的实例ID。
    static String instanceId = "<your-instanceId>";
    public static void main(String[] args) {
        client = Initialization();
        // 步骤一：等待抢占式实例到待回收状态，并产生中断事件。
        waitForInstanceMarked();
        System.out.println("spot instance will be recycled immediately, instance id:" + ins
tanceId);
        // 步骤二：当抢占式实例产生中断事件时，自动为实例创建自定义镜像。
```

```
        String image1 = createImage();
        // 步骤三：等待自定义镜像创建成功。
        waitCreateImageSuccess(image1);
    }
    private static IAcsClient Initialization() {
        /**
         * 初始化请求参数。
         * 其中变量<your-access-key-id>需要设置为您的阿里云账号的AccessKey ID。
         * <your-access-key-secret>需要设置为您的阿里云账号的AccessKey Secret。
         */
        DefaultProfile profile = DefaultProfile.getProfile(regionId,"<your-access-key-id>",
"<your-access-key-secret>");
        return new DefaultAcsClient(profile);
    }
    // 监控抢占式实例的状态，当产生中断事件时，输出实例相关信息。
    public static void waitForInstanceMarked() {
        // 将对象转化为JSON字符串。
        ArrayList<String> instanceIds = new ArrayList();
        instanceIds.add(instanceId);
        String instanceIdStr = JSON.toJSONString(instanceIds);
        boolean isMarked = false;
        // 判断抢占式实例是否产生中断事件。
        while (!isMarked) {
            try {
                // 设置DescribeInstances参数，发送请求。
                DescribeInstancesRequest request = new DescribeInstancesRequest();
                // 指定抢占式实例所在的地域。
                request.setRegionId(regionId);
                // 指定抢占式实例ID查询。
                request.setInstanceIds(instanceIdStr);
                // 接收调用的返回结果。
                DescribeInstancesResponse response = client.getAcsResponse(request);
                // 获取抢占式实例相关的返回结果。
                List<DescribeInstancesResponse.Instance> instanceList = response.getInstanc
es();
                // 如果未查询到实例信息，则跳出循环。
                if (instanceList == null || instanceList.isEmpty()) {
                    break;
                }
                DescribeInstancesResponse.Instance instance = instanceList.get(0);
                // 如果查询到的实例没有被中断，则重新开始循环。
                if (instance.getOperationLocks() == null || instance.getOperationLocks().si
ze() == 0) {
                    continue;
                }
                for (DescribeInstancesResponse.Instance.LockReason lockReason : instance.ge
tOperationLocks()) {
                    // 如果查询到的实例被中断，则输出指定实例ID以及造成中断的原因。
                    System.out.println("instance:" + instance.getInstanceId() + "-->lockRea
son:" + lockReason.getLockReason() + ",vmStatus:" + instance.getStatus());
                    if ("Recycling".equals(lockReason.getLockReason())) {
                        isMarked = true;
                    }
                }
                Thread.sleep(2 * 1000);
```

```
                Thread.sleep(2 * 1000);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
    // 创建自定义镜像。
    public static String createImage() {
        try {
            // 设置CreateImage参数，发送请求。
            CreateImageRequest request = new CreateImageRequest();
            request.setRegionId(regionId);
            request.setInstanceId(instanceId);
            // 接收调用的返回结果，并输出已创建的自定义镜像ID。
            CreateImageResponse response = client.getAcsResponse(request);
            System.out.println("imageID:" + response.getImageId());
            return response.getImageId();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
    // 查询镜像创建是否成功。
    public static void waitCreateImageSuccess(String imageId) {
        boolean isSuccess = false;
        while (!isSuccess) {
            DescribeImagesResponse.Image image = describeImage(imageId);
            if (null == image) {
                System.err.println("image not exist. imageId: " + imageId);
                break;
            }
            if ("Available".equals(image.getStatus())) {
                System.out.println("Image created successfully.");
                isSuccess = true;
            }
        }
    }
    // 调用DescribeImages监控镜像状态。
    public static DescribeImagesResponse.Image describeImage(String imageId) {
        try {
            Thread.sleep(6 * 60 * 1000);
            DescribeImagesRequest imagesRequest = new DescribeImagesRequest();
            imagesRequest.setRegionId(regionId);
            imagesRequest.setImageId(imageId);
            imagesRequest.setPageSize(100);
            DescribeImagesResponse imagesResponse = client.getAcsResponse(imagesRequest);
            if (null == imagesResponse.getImages() || imagesResponse.getImages().isEmpty())
{
                return null;
            }
            return imagesResponse.getImages().get(0);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
```

```
    }
}
```

步骤三：使用自定义镜像新建抢占式实例实现数据恢复

本步骤提供名为 `CreateSpotInstanceFromImage` 的示例类，代码中调用ECS的RunInstances接口，指定已创建的自定义镜像新建抢占式实例。

```
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.ecs.model.v20140526.RunInstancesRequest;
import com.aliyuncs.ecs.model.v20140526.RunInstancesResponse;
import com.aliyuncs.profile.DefaultProfile;
/**
 *通过RunInstances创建抢占式实例。
 */
public class CreateSpotInstanceFromImage {
    static IAcsClient client;
    // 指定实例所属的地域ID。建议与源抢占式实例所属地域保持一致。
    static String regionId = "cn-hangzhou";
    // 指定实例所属的可用区ID。建议与源抢占式实例所属可用区保持一致。
    static String zoneId = "cn-shanghai-l";
    // 指定创建的ECS实例所使用的实例规格。
    static String instanceType = "ecs.s6-c1m1.small";
    // 指定已创建的自定义镜像ID。
    static String imagesId = "<your-imagesId>";
    // 指定创建的ECS实例所属的交换机ID。
    static String vSwitchId = "<your-vSwitchId>";
    // 指定创建的ECS实例所属的安全组ID。
    static String securityGroupId = "<your-securityGroupId>";
    // 指定抢占策略。
    static String spotStrategy = "SpotAsPriceGo";
    // 修改您需要保留抢占式实例的时长。不能确定保留时长时，请设置为0。
    static Integer spotDuration = 0;
    // 指定ECS实例的登录密码。
    static String password = "<your-passwd>";
    public static void main(String[] args) {
        client = Initialization();
        createInstance();
    }
    private static IAcsClient Initialization() {
        /**
         * 初始化请求参数。
         * 其中变量<your-access-key-id>需要设置为您的阿里云账号的AccessKey ID。
         * <your-access-key-secret>需要设置为您的阿里云账号的AccessKey Secret。
         */
        DefaultProfile profile = DefaultProfile.getProfile(regionId,"<your-access-key-id>",
"<your-access-key-secret>");
        return new DefaultAcsClient(profile);
    }
    //调用RunInstances创建实例。
    public static String createInstance() {
        try {
            RunInstancesRequest request = new RunInstancesRequest();
```

```
            RunInstancesRequest request = new RunInstancesRequest();
            request.setRegionId(regionId);
            request.setZoneId(zoneId);
            request.setInstanceType(instanceType);
            request.setSpotDuration(spotDuration);
            request.setSpotStrategy(spotStrategy);
            request.setImageId(imagesId);
            request.setVSwitchId(vSwitchId);
            request.setSecurityGroupId(securityGroupId);
            request.setInstanceChargeType("PostPaid");
            request.setPassword(password);
            request.setInternetMaxBandwidthOut(1);
            RunInstancesResponse response = client.getAcsResponse(request);
            if (null == response.getInstanceIdSets() || response.getInstanceIdSets().isEmpt
y()) {
                return null;
            }
            String instanceId = response.getInstanceIdSets().get(0);
            System.out.println("创建的实例ID:  " + instanceId);
            return instanceId;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

7.5.2. 使用系统盘快照创建自定义镜像

抢占式实例可能会因为价格因素或者市场供需变化而被强制回收。本文将以Alibaba Cloud SDK for Java为例，介绍如何通过Java代码监控到抢占式实例被回收的中断事件后，系统自动创建实例的系统盘快照并使用快照创建自定义镜像，您可以使用该镜像新建抢占式实例完成实例内的数据恢复。

前提条件

- 已准备阿里云账号以及对应的访问密钥（AccessKey）。

  使用Alibaba Cloud SDK for Java时需要设置阿里云账号的AccessKey信息。AccessKey的获取方式，请参见获取AccessKey。

- 已在开发环境中安装Java SDK。

  您需要在Maven项目中添加以下依赖。具体操作，请参见安装Java SDK。

```
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>java.demo</groupId>
    <artifactId>test</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>
       <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.2.68</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-java-sdk-ecs</artifactId>
            <version>4.23.10</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-java-sdk-core</artifactId>
            <version>4.0.8</version>
        </dependency>
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-collections4</artifactId>
            <version>4.4</version>
        </dependency>
    </dependencies>
</project>
```

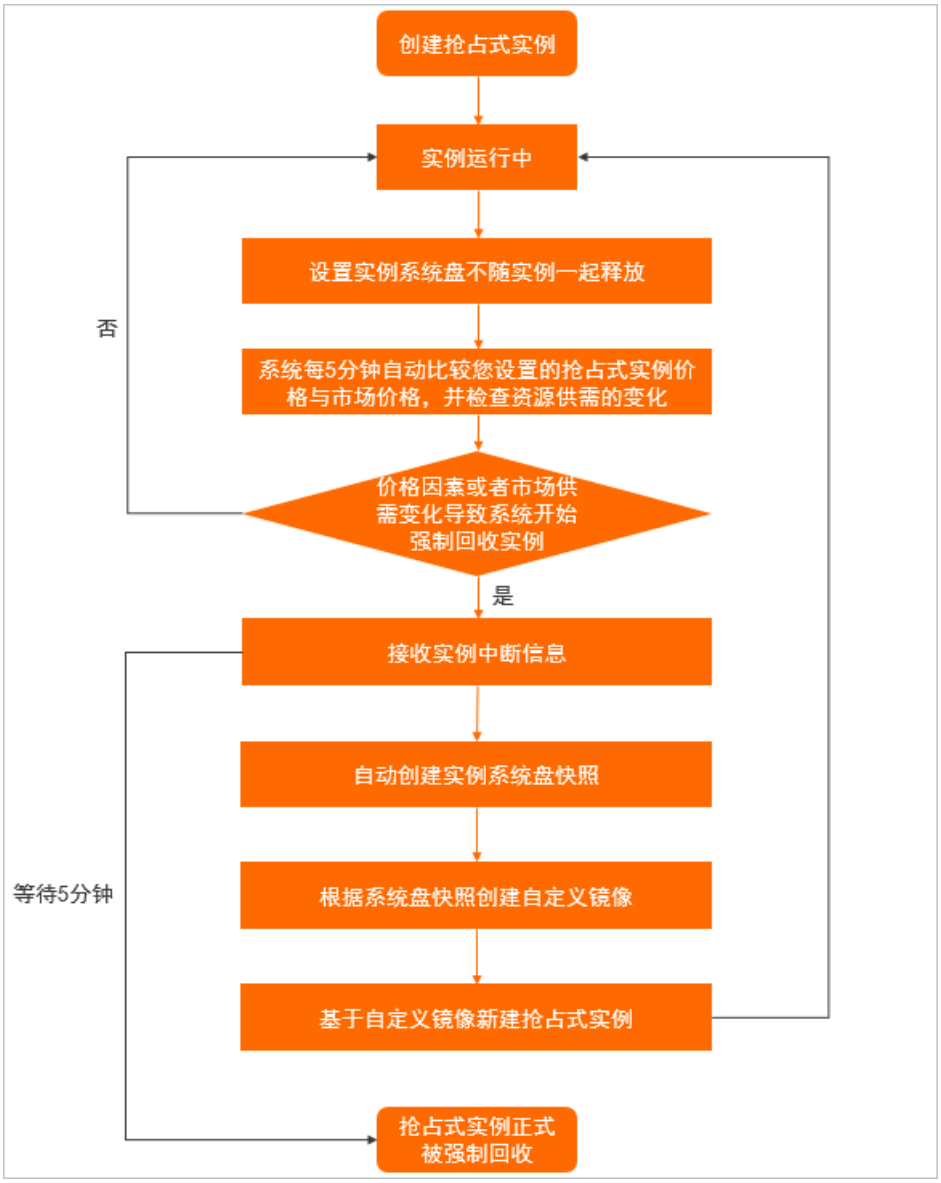
背景信息

您在使用抢占式实例时，实例可能会因为价格因素或者市场供需变化而被强制回收，在被完全回收前，实例会进入锁定状态，并触发抢占式实例的中断事件。

您可以基于该事件设置监控机制，并在实例正常运行过程中设置系统盘不随实例一起释放，当接收到抢占式实例的中断事件后，系统通过Java代码自动为系统盘创建快照，再根据系统盘快照自动创建自定义镜像，您可以使用创建好的自定义镜像新建抢占式实例，实现实例内的数据恢复。

> **说明** 设置系统盘不随实例一起释放后，即使抢占式实例被释放，创建系统盘快照、创建自定义镜像等工作不受影响。

本文提供的示例场景中，运维工作流程图如下所示：



注意事项

> **注意** 本文提供的示例代码仅供参考，并不能保证您的实例一定会在5分钟内完成镜像的创建与数据的恢复。

抢占式实例会提前至少5分钟发送实例中断消息，但数据恢复的具体耗时取决于您实例的镜像类型与系统盘文件大小等因素。例如，系统盘文件越大，恢复时间越久。请您使用示例代码前务必自行进行评估与验证。

使用限制

抢占式实例被回收时的数据恢复过程中，存在如下限制：

- 为了保证磁盘快照不丢失，建议关闭快照随磁盘一起释放的功能。
- 如果抢占式实例中含有数据盘，且数据盘中有重要数据，建议数据盘设置为不随实例释放。具体操作，请参见步骤二：设置系统盘不随实例一起释放中的*方式二：通过ECS控制台实现*。
- 当您调用ModifyDiskAttribute接口时设置了不随实例释放（DeleteWithInstance=false）属性，一旦磁盘

挂载的ECS实例被安全锁定且OperationLocks中标记了 `"LockReason" : "security"` 的锁定状态，释放实例时会忽略磁盘的DeleteWithInstance属性而被同时释放。

> **说明** 您可以设置 `DiskIds.N` 参数批量修改多个块存储的名称、描述、是否随实例释放等属性。

步骤一：创建抢占式实例

本步骤提供名为 `CreateSpotInstance` 的示例类，代码中主要通过ECS的RunInstances接口创建抢占式实例。

```
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.ecs.model.v20140526.RunInstancesRequest;
import com.aliyuncs.ecs.model.v20140526.RunInstancesResponse;
import com.aliyuncs.profile.DefaultProfile;
/**
 * 通过RunInstances创建抢占式实例。
 */
public class CreateSpotInstance {
    static IAcsClient client;
    // 指定地域ID。指定后您创建的ECS实例属于该地域内。
    static String regionId = "cn-hangzhou";
    // 指定可用区ID。指定后您创建的ECS实例属于该可用区内。
    static String zoneId = "cn-hangzhou-i";
    // 指定创建的ECS实例所使用的实例规格。
    static String instanceType = "ecs.s6-c1m1.small";
    // 指定创建的ECS实例所使用的镜像ID。
    static String imagesId = "centos_7_6_x64_20G_alibase_20211130.vhd";
    // 指定创建的ECS实例所属的交换机ID。
    static String vSwitchId = "<your-vSwitchId>";
    // 指定创建的ECS实例所属的安全组ID。
    static String securityGroupId = "<your-securityGroupId>";
    // 指定抢占策略。
    static String spotStrategy = "SpotAsPriceGo";
    // 修改您需要保留抢占式实例的时长。不能确定保留时长时，请设置为0。
    static Integer spotDuration = 0;
    // 指定ECS实例的登录密码。
    static String password = "<your-password>";
    public static void main(String[] args) {
        client = Initialization();
        createInstance();
    }
    private static IAcsClient Initialization() {
        /**
         * 初始化请求参数。
         * 其中变量<your-access-key-id>需要设置为您的阿里云账号的AccessKey ID。
         * <your-access-key-secret>需要设置为您的阿里云账号的AccessKey Secret。
         */
        DefaultProfile profile = DefaultProfile.getProfile(regionId,"<your-access-key-id>",
"<your-access-key-secret>");
        return new DefaultAcsClient(profile);
    }
    //创建实例
```

```
    //创建实例。
    public static String createInstance() {
        try {
            // 设置RunInstances参数，发送请求。
            RunInstancesRequest request = new RunInstancesRequest();
            request.setRegionId(regionId);
            request.setZoneId(zoneId);
            request.setInstanceType(instanceType);
            request.setSpotDuration(spotDuration);
            request.setSpotStrategy(spotStrategy);
            request.setImageId(imagesId);
            request.setVSwitchId(vSwitchId);
            request.setSecurityGroupId(securityGroupId);
            // InstanceChargeType取值为PostPaid时才会生效抢占策略。
            request.setInstanceChargeType("PostPaid");
            request.setPassword(password);
            request.setInternetMaxBandwidthOut(1);
            // 接收调用的返回结果，并输出已创建的ECS实例ID。
            RunInstancesResponse response = client.getAcsResponse(request);
            if (null == response.getInstanceIdSets() || response.getInstanceIdSets().isEmpt
y()) {
                return null;
            }
            String instanceId = response.getInstanceIdSets().get(0);
            System.out.println("创建的实例ID：" + instanceId);
            return instanceId;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

步骤二：设置系统盘不随实例一起释放

方式一：通过Java代码实现

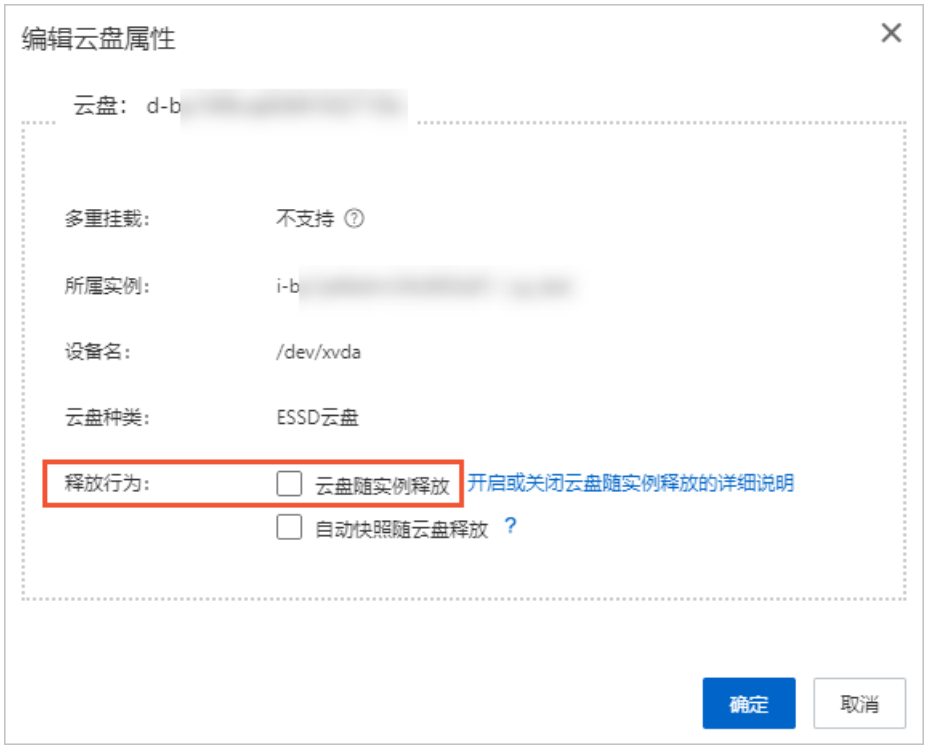
```
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.ecs.model.v20140526.DescribeDisksRequest;
import com.aliyuncs.ecs.model.v20140526.DescribeDisksResponse;
import com.aliyuncs.ecs.model.v20140526.DescribeDisksResponse.Disk;
import com.aliyuncs.ecs.model.v20140526.ModifyDiskAttributeRequest;
import com.aliyuncs.ecs.model.v20140526.ModifyDiskAttributeResponse;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.profile.DefaultProfile;
import org.apache.commons.collections4.CollectionUtils;
public class DiskRelated {
    static IAcsClient client;
    // 请将regionId修改为您的抢占式实例所属的地域ID。
    static String regionId = "cn-hangzhou";
   // 抢占式实例的实例ID。
    static String instanceId = "<your-instance-id>";
    public static void main(String[] args) {
```

```
        client = initialization();
        Disk disk = getDisks();
          if(null == disk){
            System.out.println("disk not exist");
            return;
        }
        String diskId = disk.getDiskId();
        modifyDiskAttribute(diskId);
        Boolean b = diskNotDeleteWithInstance();
        if(b){
            //如果第一次设置系统盘不随实例一起释放失败，则重新设置一次。
            modifyDiskAttribute(diskId);
        }
    //查询系统盘详情。
    public static Disk getDisks(){
        DescribeDisksRequest request = new DescribeDisksRequest();
        request.setSysRegionId(regionId);
        request.setInstanceId(instanceId);
        request.setDiskType("system");
        try {
            DescribeDisksResponse response = client.getAcsResponse(request);
            if(CollectionUtils.isEmpty(response.getDisks())){
                System.out.println(("disk not exist. instanceId: " + instanceId));
                return null;
            }
            Disk disk = response.getDisks().get(0);
            return disk;
        } catch (ClientException e) {
            e.printStackTrace();
        }
        return null;
    }
    //设置系统盘不随实例一起释放。
    public static void modifyDiskAttribute(String diskId){
        ModifyDiskAttributeRequest request = new ModifyDiskAttributeRequest();
        request.setDeleteWithInstance(false);
        request.setSysRegionId(regionId);
        request.setDiskId(diskId);
        try {
            ModifyDiskAttributeResponse response = client.getAcsResponse(request);
            System.out.println(response.getRequestId());
        } catch (ClientException e) {
            e.printStackTrace();
        }
    }
    //查询系统盘是否随实例一起释放。
    public static Boolean diskNotDeleteWithInstance(){
        Disk disks = getDisks();
        if (disks.getDeleteWithInstance()){
            System.out.println(("disk is delete with instance"));
        }else {
            System.out.println(("disk not delete with instance"));
        }
        return disks.getDeleteWithInstance();
```

```
    }
    private static IAcsClient initialization() {
        /**
         * 初始化请求参数。
         * 其中变量<your-access-key-id>需要设置为您的阿里云账号的AccessKey ID。
         * <your-access-key-secret>需要设置为您的阿里云账号的AccessKey Secret。
         */
        DefaultProfile profile = DefaultProfile.getProfile(regionId,"<your-access-key-id>",
"<your-access-key-secret>");
        return new DefaultAcsClient(profile);
    }
}
```

方式二：通过ECS控制台实现

1. 登录ECS管理控制台。
2. 在左侧导航栏，选择**实例与镜像 > 实例**。
3. 在顶部菜单栏左上角处，选择地域。
4. 找到目标抢占式实例，单击实例ID。
5. 在**实例详情**页，单击**云盘**页签。
6. 在目标云盘的**操作**列中，单击 ⋮ 图标，选择**编辑属性**。
7. 取消勾选**云盘随实例释放**，然后单击**确定**。



步骤三：监控到中断事件后自动创建自定义镜像

当监控到抢占式实例产生中断事件后，系统通过Java代码自动为系统盘创建快照，再根据系统盘快照自动创建自定义镜像。

本步骤提供名为 `CreateSpotImage` 的示例类，代码中依次调用了以下接口分别实现功能：

- 调用DescribeInstances监控抢占式实例的状态。
- 当监控到抢占式实例产生中断事件后，先调用CreateSnapshot创建系统盘快照，再调用DescribeSnapshots查询快照状态。
- 创建系统盘快照后，调用CreateImage，根据已创建的系统盘快照创建自定义镜像。
- 创建自定义镜像后，调用DescribeImages监控自定义镜像的状态，当镜像变为可用状态时，返回提示信息。

```
import java.util.ArrayList;
import java.util.List;
import com.alibaba.fastjson.JSON;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.ecs.model.v20140526.CreateImageRequest;
import com.aliyuncs.ecs.model.v20140526.CreateImageResponse;
import com.aliyuncs.ecs.model.v20140526.CreateSnapshotRequest;
import com.aliyuncs.ecs.model.v20140526.CreateSnapshotResponse;
import com.aliyuncs.ecs.model.v20140526.DescribeDisksRequest;
import com.aliyuncs.ecs.model.v20140526.DescribeDisksResponse;
import com.aliyuncs.ecs.model.v20140526.DescribeDisksResponse.Disk;
import com.aliyuncs.ecs.model.v20140526.DescribeImagesRequest;
import com.aliyuncs.ecs.model.v20140526.DescribeImagesResponse;
import com.aliyuncs.ecs.model.v20140526.DescribeInstancesRequest;
import com.aliyuncs.ecs.model.v20140526.DescribeInstancesResponse;
import com.aliyuncs.ecs.model.v20140526.DescribeInstancesResponse.Instance;
import com.aliyuncs.ecs.model.v20140526.DescribeSnapshotsRequest;
import com.aliyuncs.ecs.model.v20140526.DescribeSnapshotsResponse;
import com.aliyuncs.ecs.model.v20140526.DescribeSnapshotsResponse.Snapshot;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.profile.DefaultProfile;
import com.google.common.collect.Lists;
/**
 * 监控抢占式实例的中断事件。当中断事件发生时自动创建系统盘快照，再根据系统盘快照创建自定义镜像。
 * 代码中将会调用以下ECS API：
 * DescribeInstances：查询实例信息。
 * CreateSnapshot：创建系统盘快照。
 * DescribeSnapshots：查询系统盘快照状态。
 * CreateImage：创建自定义镜像。
 * DescribeImages：查询自定义镜像的状态。
 */
public class CreateSpotImage {
    static IAcsClient client;
    // 请将regionId修改为您的抢占式实例所属的地域ID。
    static String regionId = "cn-hangzhou";
    // 抢占式实例的实例ID。
    static String instanceId = "<your-instance-id>";
    public void main(String[] args) {
        client = initialization();
        // 步骤一：等待抢占式实例到待回收状态，并产生中断事件。
        waitForInstanceMarked();
        String diskId = getDiskId();
        // 步骤二：当抢占式实例产生中断事件时，自动创建系统盘快照。
        String snapshotId = createSnapshot(diskId);
```

```
        String snapshotId = createSnapshot(diskId);
        // 步骤三：等待系统盘快照创建成功。
        waitCreateSnapshotSuccess(snapshotId);
        // 步骤四：根据系统盘快照创建自定义镜像。
        String imageId = createImage(snapshotId);
        // 步骤五：等待自定义镜像创建成功。
        waitCreateImageSuccess(imageId);
    }
    // 监控抢占式实例的状态，当产生中断事件时，输出实例相关信息。
    public void waitForInstanceMarked() {
        // 将对象转化为JSON字符串。
        ArrayList<String> instanceIds = new ArrayList();
        DescribeInstancesResponse.Instance instance = null;
        instanceIds.add(instanceId);
        String instanceIdStr = JSON.toJSONString(instanceIds);
        boolean isMarked = false;
        // 判断抢占式实例是否产生中断事件。
        while (!isMarked) {
            try {
                // 设置DescribeInstances参数，发送请求。
                DescribeInstancesRequest request = new DescribeInstancesRequest();
                // 指定抢占式实例所在的地域。
                request.setRegionId(regionId);
                // 指定抢占式实例ID查询。
                request.setInstanceIds(instanceIdStr);
                // 接收调用的返回结果。
                DescribeInstancesResponse response = client.getAcsResponse(request);
                // 获取抢占式实例相关的返回结果。
                List<Instance> instanceList = response.getInstances();
                // 如果未查询到实例信息，则跳出循环。
                if (instanceList == null || instanceList.isEmpty()) {
                    break;
                }
                instance = instanceList.get(0);
                // 如果查询到的实例没有被中断，则重新开始循环。
                if (instance.getOperationLocks() == null || instance.getOperationLocks().si
ze() == 0) {
                    continue;
                }
                for (DescribeInstancesResponse.Instance.LockReason lockReason : instance.ge
tOperationLocks()) {
                    // 如果查询到的实例被中断，则输出指定实例ID以及造成中断的原因。
                    System.out.println("instance:" + instance.getInstanceId() + "-->lockRea
son:" + lockReason.getLockReason() + ",vmStatus:" + instance.getStatus());
                    if ("Recycling".equals(lockReason.getLockReason())) {
                        isMarked = true;
                    }
                }
                Thread.sleep(2 * 1000);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
    //查询系统盘信息。
```

```
    public static String getDiskId(){
        String diskId = null;
        DescribeDisksRequest request = new DescribeDisksRequest();
        request.setSysRegionId(regionId);
        request.setInstanceId(instanceId);
        request.setDiskType("system");
        try {
            DescribeDisksResponse response = client.getAcsResponse(request);
            List<Disk> disks = response.getDisks();
            if (null == disks || 0 == disks.size() ){
                System.out.println("disk not exist. instance: " + instanceId);
                return null;
            }
            Disk disk = disks.get(0);
            diskId = disk.getDiskId();
        } catch (ClientException e) {
            e.printStackTrace();
        }
        return diskId;
    }
    //创建系统盘快照。
    public static String createSnapshot(String diskId){
        CreateSnapshotRequest request = new CreateSnapshotRequest();
        request.setDiskId(diskId);
        request.setSnapshotName("disk_test");
        CreateSnapshotResponse response = null;
        try {
            response = client.getAcsResponse(request);
            System.out.println(JSON.toJSONString(response));
            System.out.println(response.getSnapshotId());
            return response.getSnapshotId();
        } catch (ClientException e) {
            e.printStackTrace();
        }
        return response.getSnapshotId();
    }
    //查询系统盘快照创建是否成功。
    public static void waitCreateSnapshotSuccess(String snapshotId){
        boolean isSuccess = false;
        while (!isSuccess) {
            Snapshot snapshot = describeSnapshots(snapshotId);
            if (null == snapshot) {
                System.err.println("image not exist. imageId: " + snapshotId);
                break;
            }
            if("accomplished".equals(snapshot.getStatus())){
                System.out.println("snapshot created successfully.");
                isSuccess = true;
            }
        }
    }
    //调用DescribeSnapshots查询系统盘快照状态。
    public static Snapshot describeSnapshots(String snapshotId){
        DescribeSnapshotsRequest request = new DescribeSnapshotsRequest();
```

```
        request.setSysRegionId(regionId);
        List<String> snapshotIds = Lists.newArrayList(snapshotId);
        String s = JSON.toJSONString(snapshotIds);
        request.setSnapshotIds(s);
        try {
            DescribeSnapshotsResponse response = client.getAcsResponse(request);
            if (null == response.getSnapshots() || response.getSnapshots().isEmpty()) {
                return null;
            }
            return response.getSnapshots().get(0);
        } catch (ClientException e) {
            e.printStackTrace();
        }
        return null;
    }
    // 创建自定义镜像。
    public static String createImage(String snapshotId) {
        try {
            // 设置CreateImage参数，发送请求。
            CreateImageRequest request = new CreateImageRequest();
            request.setRegionId(regionId);
            request.setSnapshotId(snapshotId);
            request.setImageName("image_test");
            // 接收调用的返回结果，并输出已创建的自定义镜像ID。
            CreateImageResponse response = client.getAcsResponse(request);
            System.out.println("imageID:" + response.getImageId());
            return response.getImageId();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
    // 查询镜像创建是否成功。
    public static void waitCreateImageSuccess(String imageId) {
        boolean isSuccess = false;
        while (!isSuccess) {
            DescribeImagesResponse.Image image = describeImage(imageId);
            if (null == image) {
                System.err.println("image not exist. imageId: " + imageId);
                break;
            }
            if ("Available".equals(image.getStatus())) {
                System.out.println("Image created successfully.");
                isSuccess = true;
            }
        }
    }
    // 调用DescribeImages监控镜像状态。
    public static DescribeImagesResponse.Image describeImage(String imageId) {
        try {
            Thread.sleep(6 * 60 * 1000);
            DescribeImagesRequest imagesRequest = new DescribeImagesRequest();
            imagesRequest.setRegionId(regionId);
            imagesRequest.setImageId(imageId);
```

```
            imagesRequest.setPageSize(100);
            DescribeImagesResponse imagesResponse = client.getAcsResponse(imagesRequest);
            if (null == imagesResponse.getImages() || imagesResponse.getImages().isEmpty())
{
                return null;
            }
            return imagesResponse.getImages().get(0);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
    private static IAcsClient initialization() {
        /**
         * 初始化请求参数。
         * 其中变量<your-access-key-id>需要设置为您的阿里云账号的AccessKey ID。
         * <your-access-key-secret>需要设置为您的阿里云账号的AccessKey Secret。
         */
        DefaultProfile profile = DefaultProfile.getProfile(regionId,"<your-access-key-id>",
"<your-access-key-secret>");
        return new DefaultAcsClient(profile);
    }
}
```

步骤四：使用自定义镜像新建抢占式实例实现数据恢复

本步骤提供名为 `CreateSpotInstanceFromImage` 的示例类，代码中调用ECS的RunInstances接口，指定已创建的自定义镜像新建抢占式实例。

```
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.ecs.model.v20140526.RunInstancesRequest;
import com.aliyuncs.ecs.model.v20140526.RunInstancesResponse;
import com.aliyuncs.profile.DefaultProfile;
/**
 *通过RunInstances创建抢占式实例。
 */
public class CreateSpotInstanceFromImage {
    static IAcsClient client;
    // 指定实例所属的地域ID。建议与源抢占式实例所属地域保持一致。
    static String regionId = "cn-hangzhou";
    // 指定实例所属的可用区ID。建议与源抢占式实例所属可用区保持一致。
    static String zoneId = "cn-hangzhou-i";
    // 指定创建的ECS实例所使用的实例规格。
    static String instanceType = "ecs.s6-c1m1.small";
    // 指定步骤三：监控到中断事件后自动创建自定义镜像中创建的自定义镜像ID。
    static String imagesId = "<your-image-id>";
    // 指定创建的ECS实例所属的交换机ID。
    static String vSwitchId = "<your-vsw-id>";
    // 指定创建的ECS实例所属的安全组ID。
    static String securityGroupId = "<your-sg-id>";
    // 指定抢占策略。
    static String spotStrategy = "SpotAsPriceGo";
```

```
    // 修改您需要保留抢占式实例的时长。不能确定保留时长时，请设置为0。
    static Integer spotDuration = 0;
    // 指定ECS实例的登录密码。
    static String password = "<your-passwd>";
    public static void main(String[] args) {
        client = Initialization();
        createInstance();
    }
    private static IAcsClient Initialization() {
        /**
         * 初始化请求参数。
         * 其中变量<your-access-key-id>需要设置为您的阿里云账号的AccessKey ID。
         * <your-access-key-secret>需要设置为您的阿里云账号的AccessKey Secret。
         */
        DefaultProfile profile = DefaultProfile.getProfile(regionId,"<your-access-key-id>",
"<your-access-key-secret>");
        return new DefaultAcsClient(profile);
    }
    //调用RunInstances创建实例。
    public static String createInstance() {
        try {
            RunInstancesRequest request = new RunInstancesRequest();
            request.setRegionId(regionId);
            request.setZoneId(zoneId);
            request.setInstanceType(instanceType);
            request.setSpotDuration(spotDuration);
            request.setSpotStrategy(spotStrategy);
            request.setImageId(imagesId);
            request.setVSwitchId(vSwitchId);
            request.setSecurityGroupId(securityGroupId);
            request.setInstanceChargeType("PostPaid");
            request.setPassword(password);
            request.setInternetMaxBandwidthOut(1);
            RunInstancesResponse response = client.getAcsResponse(request);
            if (null == response.getInstanceIdSets() || response.getInstanceIdSets().isEmpt
y()) {
                return null;
            }
            String instanceId = response.getInstanceIdSets().get(0);
            System.out.println("创建的实例ID： " + instanceId);
            return instanceId;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

相关链接

- RunInstances
- DescribeInstances

- CreateImage
- DescribeImages
- DescribeDisks
- ModifyDiskAttribute
- CreateSnapshot
- DescribeSnapshots

8.安全
8.1. ECS安全组实践（一）

本文介绍配置安全组的入方向规则的最佳实践。您可以通过配置安全组规则，允许或禁止安全组内的ECS实例对公网或私网的访问。

安全组实践建议

您在云端安全组提供类似虚拟防火墙功能，用于设置单台或多台ECS实例的网络访问控制，是重要的安全隔离手段。创建ECS实例时，您必须选择一个安全组。您还可以添加安全组规则，对某个安全组下的所有ECS实例的出方向和入方向进行网络控制。

在使用安全组前，您应先了解以下实践建议：

- 最重要的规则：安全组应作为白名单使用。
- 开放应用出入规则时应遵循最小授权原则。例如，您可以选择开放具体的端口，如80端口。
- 不应使用一个安全组管理所有应用，因为不同的分层一定有不同的需求。
- 对于分布式应用来说，不同的应用类型应该使用不同的安全组，例如，您应对Web层、Service层、Database层、Cache层使用不同的安全组，暴露不同的出入规则和权限。
- 避免为每台实例单独设置一个安全组，控制管理成本。
- 优先考虑专有网络VPC。
- 不需要公网访问的资源不应提供公网IP。
- 尽可能保持单个安全组的规则简洁。因为一台实例最多可以加入五个安全组，一个安全组最多可以包括200条安全组规则，所以一台ECS实例可能同时应用数百条安全组规则。您可以聚合所有分配的安全规则以判断是否允许流入或流出，但是，如果单个安全组规则很复杂，就会增加管理的复杂度。
- 阿里云的控制台提供了克隆安全组和安全组规则的功能。如果您想要修改线上的安全组和规则，您应先克隆一个安全组，再在克隆的安全组上进行调试，避免直接影响线上应用。

> **说明** 调整线上的安全组的出入规则是比较危险的动作。如果您无法确定，不应随意更新安全组出入规则的设置。

避免设置0.0.0.0/0授权对象

允许全部入网访问是经常犯的错误。使用0.0.0.0/0意味着所有的端口都对外暴露了访问权限。这是非常不安全的。正确的做法是，先拒绝所有的端口对外开放。安全组应该是白名单访问。例如，如果您需要暴露Web服务，默认情况下可以只开放80、8080和443之类的常用TCP端口，其它的端口都应关闭。

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

关闭不需要的入网规则

如果您当前使用的入规则已经包含了0.0.0.0/0，您需要重新审视自己的应用需要对外暴露的端口和服务。如果确定不想让某些端口直接对外提供服务，您可以加一条拒绝的规则。例如，如果您的服务器上安装了MySQL数据库服务，默认情况下您不应该将3306端口暴露到公网，此时，您可以添加一条拒绝规则，如下所示，并将其优先级设为100，即优先级最低。

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

上面的调整会导致所有的端口都不能访问3306端口，极有可能会阻止您正常的业务需求。此时，您可以通过授权另外一个安全组的资源进行入规则访问。

以安全组为授权对象添加规则

不同的安全组按照最小原则开放相应的出入规则。对于不同的应用分层应该使用不同的安全组，不同的安全组应有相应的出入规则。

例如，如果是分布式应用，您会区分不同的安全组，但是，不同的安全组可能网络不通，此时您不应该直接授权IP或者CIDR网段，而是直接授权另外一个安全组ID的所有的资源都可以直接访问。例如，您的应用对Web、Database分别创建了不同的安全组：sg-web和sg-database。在sg-database中，您可以添加如下规则，授权所有的sg-web安全组的资源访问您的3306端口。

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

以IP地址段为授权对象添加规则

经典网络中，因为网段不太可控，建议您使用安全组ID来授信入网规则。

VPC网络中，您可以自己通过不同的vSwitch设置不同的IP域，规划IP地址。所以，在VPC网络中，您可以默认拒绝所有的访问，再授信自己的专有网络的网段访问，直接授信可以相信的CIDR网段。

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

变更安全组规则步骤

变更安全组规则可能会影响您的实例间的网络通信。为了保证必要的网络通信不受影响，您应先尝试以下方法放行必要的实例，再执行安全组策略收紧变更。

> **说明** 执行收紧变更后，应观察一段时间，确认业务应用无异常后再执行其它必要的变更。

- 新建一个安全组，将需要互通访问的实例加入这个安全组，再执行变更操作。
- 如果授权类型为**安全组访问**，则将需要互通访问的对端实例所绑定的安全组ID添加为授权对象。
- 如果授权类型为**地址段访问**，则将需要互通访问的对端实例内网IP添加为授权对象。

具体操作指引请参见添加安全组规则。

8.2. ECS安全组实践（二）

本文从授权和撤销安全组规则、加入和移出安全组讲解云服务器ECS的安全组最佳实践。

网络类型

阿里云的网络类型分为经典网络和专有网络VPC，对安全组支持不同的设置规则：

- 如果是经典网络，您可以设置内网入方向、内网出方向、公网入方向和公网出方向的安全组规则。
- 如果是专有网络VPC，您可以设置内网入方向和内网出方向的安全组规则。

安全组是区分网络类型的，一台经典网络类型的ECS实例只能加入经典网络的安全组。一台专有网络VPC类型的ECS实例只能加入本VPC的安全组。

安全组内网通讯的概念

本文开始之前，您应知道以下几个安全组内网通讯的概念：

- 即使是同一个账户下的ECS实例，如果分属不同安全组，内网网络也是不通的。这个对于经典网络和专有网络VPC都适用。所以，经典网络类型的ECS实例也是内网安全的。
- 如果您有两台ECS实例，不在同一个安全组，您希望它们内网不互通，但实际上它们却内网互通，那么，您需要检查您的安全组内网规则设置。如果内网协议存在下面的协议，建议您重新设置。
  - 允许所有端口。
  - 授权对象为CIDR网段（SourceCidrIp）：*0.0.0.0/0*或者*10.0.0.0/8*的规则。

  > **说明** 如果是经典网络，上述协议会造成您的内网暴露给其它的访问。

- 如果您想实现在不同安全组的资源之间的网络互通，您应使用安全组方式授权。对于内网访问，您应使用源安全组授权，而不是CIDR网段授权。

安全组规则的属性

安全组规则主要是描述不同的访问权限，包括如下属性：

- Policy：授权策略，参数值可以是*accept*（允许）或*drop*（拒绝）。
- Priority：优先级，根据安全组规则的创建时间降序排序匹配。规则优先级可选范围为1~100，默认值为1，即最高优先级。数字越大，代表优先级越低。
- NicType：网卡类型。如果只指定了SourceGroupId而没有指定SourceCidrIp，表示通过安全组方式授权，此时，NicType必须指定为*intranet*。
- 规则描述：
  - IpProtocol：IP协议，取值：*tcp*、*udp*、*icmp*、*gre*或*all*。*all*表示所有的协议。
  - PortRange：IP协议相关的端口号范围：
    - IpProtocol取值为*tcp*或*udp*时，端口号取值范围为1~65535，格式必须是“起始端口号/终止端口号”，如“1/200”表示端口号范围为1~200。如果输入值为“200/1”，接口调用将报错。
    - IpProtocol取值为*icmp*、*gre*或*all*时，端口号范围值为-1/-1，表示不限制端口。
  - 如果通过安全组授权，应指定SourceGroupId，即源安全组ID。此时，根据是否跨账号授权，您可以选择设置源安全组所属的账号SourceGroupOwnerAccount。
  - 如果通过CIDR授权，应指定SourceCidrIp，即源IP地址段，必须使用CIDR格式。

授权一条入网请求规则

在控制台或者通过API创建一个安全组时，入网方向默认*deny all*，即默认情况下您拒绝所有入网请求。这并不适用于所有的情况，所以您要适度地配置您的入网规则。

例如，如果您需要开启公网的80端口对外提供HTTP服务，因为是公网访问，您希望入网尽可能多访问，所以在IP网段上不应做限制，可以设置为*0.0.0.0/0*，具体设置可以参见以下描述，其中，括号外为控制台参数，括号内为OpenAPI参数，两者相同就不做区分。

- 网卡类型（NicType）：如果是经典网络，填写公网（internet）。如果是专有网络VPC，只需要填写内网（intranet），通过EIP实现公网访问。
- 授权策略（Policy）：允许（accept）。
- 规则方向：入方向。
- 协议类型（IpProtocol）：自定义TCP（tcp）。
- 端口范围（PortRange）：80/80。
- 授权对象（SourceCidrIp）：0.0.0.0/0。
- 优先级（Priority）：1。

> **说明** 上面的建议仅对公网有效。内网请求不建议使用CIDR网段，请参见经典网络的内网安全组规则不要使用CIDR或者IP授权。

禁止一个入网请求规则

禁止一条规则时，您只需要配置一条拒绝策略，并设置较低的优先级即可。这样，当有需要时，您可以配置其它高优先级的规则覆盖这条规则。例如，您可以采用以下设置拒绝6379端口被访问。

- 网卡类型（NicType）：内网（intranet）。
- 授权策略（Policy）：拒绝（drop）。
- 规则方向：入方向。
- 协议类型（IpProtocol）：自定义TCP（tcp）。
- 端口范围（PortRange）：6379/6379。
- 授权对象（SourceCidrIp）：0.0.0.0/0。
- 优先级（Priority）：100。

经典网络的内网安全组规则不要使用CIDR或者IP授权

对于经典网络类型的ECS实例，阿里云默认不开启任何内网的入规则。内网的授权一定要谨慎。

> **说明** 为了安全考虑，不建议开启任何基于CIDR网段的授权。

对于ECS实例来说，内网的IP经常变化，另外，这个IP的网段是没有规律的，所以，建议您通过安全组授权对经典网络内网的访问。

例如，您在安全组sg-redis上构建了一个Redis的集群，为了只允许特定的机器（例如sg-web）访问这个Redis的服务器编组，您不需要配置任何CIDR，只需要添加一条入规则：指定相关的安全组ID即可。

- 网卡类型（NicType）：内网（intranet）。
- 授权策略（Policy）：允许（accept）。
- 规则方向：入方向。
- 协议类型（IpProtocol）：自定义TCP（tcp）。
- 端口范围（PortRange）：6379/6379。
- 授权对象（SourceGroupId）：sg-web。
- 优先级（Priority）：1。

对于专有网络VPC类型的实例，如果您已经通过多个vSwitch规划好自己的IP范围，您可以使用CIDR设置作为安全组入规则。但是，如果您的专有网络VPC网段不够清晰，建议您优先考虑使用安全组作为入规则。

将需要互相通信的ECS实例加入同一个安全组

一个ECS实例最多可以加入5个安全组，而同一安全组内的ECS实例之间是网络互通的。如果您在规划时已经有多个安全组，而且，直接设置多个安全组规则过于复杂的话，您可以新建一个安全组，然后将需要内网通讯的ECS实例加入这个新的安全组。

这里也不建议您将所有的ECS实例都加入一个安全组，这将会使得您的安全组规则设置变的复杂。对于一个中大型应用来说，每个服务器编组的角色不同，合理地规划每个服务器的入方向请求和出方向请求是非常有必要的。

您可以在控制台上将一台实例加入安全组，具体操作，请参见ECS实例加入安全组。

如果您对阿里云的OpenAPI非常熟悉，您可以通过OpenAPI进行批量操作，具体操作，请参见查询ECS实例。对应的Python代码片段如下：

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

将ECS实例移出安全组

如果ECS实例加入不合适的安全组，将会允许本该拒绝的访问或者拒绝本该允许的访问，这时您可以选择将ECS实例从这个安全组中移出。但是在移出安全组之前必须保证您的ECS实例已经加入其它安全组。

> **说明** 将ECS实例从安全组移出，将会导致这台ECS实例和当前安全组内的网络不通，建议您在移出之前做好充分的测试。

对应的Python代码片段如下：

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

定义合理的安全组名称和标签

合理的安全组名称和描述有助于您快速识别当前复杂的规则组合。您可以通过修改名称和描述来帮助自己识别安全组。

您也可以通过为安全组设置标签分组管理自己的安全组。您可以在控制台直接设置标签，具体操作请参见设置标签，也可以通过API设置标签。

删除不需要的安全组

安全组中的安全组规则类似于一条条白名单和黑名单。所以，请不要保留不需要的安全组，以免因为错误加入某台ECS实例而造成不必要的麻烦。

8.3. ECS安全组实践（三）

在安全组的使用过程中，通常会将所有的云服务器放置在同一个安全组中，从而可以减少初期配置的工作量。但从长远来看，业务系统网络的交互将变得复杂和不可控。在执行安全组变更时，您将无法明确添加和删除规则的影响范围。

背景信息

合理规划和区分不同的安全组将使您的系统更加便于调整，梳理应用提供的服务并对不同应用进行分层。这里推荐您对不同的业务规划不同的安全组，并设置不同的安全组规则。

区分不同的安全组

- 公网服务的云服务器和内网服务器尽量属于不同的安全组

  是否对外提供公网服务，包括主动暴露某些端口对外访问（例如80、443 等），被动地提供端口转发规则（例如云服务器具有公网IP、EIP、NAT端口转发规则等），都会导致自己的应用可能被公网访问到。

  2种场景的云服务器所属的安全组规则要采用最严格的规则，建议拒绝优先，默认情况下应当关闭所有的端口和协议，仅仅暴露对外提供需要服务的端口，例如80、443。由于仅对属于对外公网访问的服务器编组，调整安全组规则时也比较容易控制。

  对于对外提供服务器编组的职责应该比较清晰和简单，避免在同样的服务器上对外提供其它的服务。例如MySQL、Redis等，建议将这些服务安装在没有公网访问权限的云服务器上，然后通过安全组的组组授权来访问。

如果当前有公网云服务器已经和其它的应用在同一个安全组SG_CURRENT。您可以通过下面的方法来进行变更。

i. 梳理当前提供的公网服务暴露的端口和协议，例如80、443。

ii. 创建一个安全组，例如SG_WEB， 然后添加相应的端口和规则。具体操作，请参见创建安全组。

- 授权策略：允许
- 协议类型：ALL
- 端口：80/80和443/443
- 授权对象：0.0.0.0/0

iii. 选择安全组SG_CURRENT， 然后添加一条安全组规则，组组授权，允许SG_WEB中的资源访问SG_CURRENT。具体操作，请参见添加安全组规则。

- 授权策略：允许
- 协议类型：ALL
- 端口：-1/-1
- 授权对象：SG_WEB
- 优先级：按照实际情况自定义[1-100]

iv. 将一台需要切换安全组的实例ECS_WEB_1添加到新的安全组中。

a. 登录ECS管理控制台。

b. 在左侧导航栏，选择**网络与安全 > 安全组**。

c. 找到安全组SG_WEB，在**操作**列下单击**管理实例**。

d. 单击**添加实例**。

e. 在弹出的对话框中，选择实例ECS_WEB_1加入到新的安全组SG_WEB中，并单击**确定**。确认ECS_WEB_1实例的流量和网络工作正常。

v. 将ECS_WEB_1从原来的安全组中移出。

a. 登录ECS管理控制台。

b. 在左侧导航栏，选择**网络与安全 > 安全组**。

c. 找到安全组SG_CURRENT，在**操作**列下单击**管理实例**。

d. 选中需要移出安全组的实例**ECS_WEB_1**，单击**移出安全组**。

e. 在弹出的对话框中，单击**确定**。

f. 测试网络连通性，确认流量和网络工作正常。

如果工作不正常，将ECS_WEB_1仍然加回到安全组SG_CURRENT中，检查设置的SG_WEB暴露的端口是否符合预期，然后继续变更。

vi. 执行其它的服务器安全组变更。

- 不同的应用使用不同的安全组

在生产环境中，不同的操作系统大多情况下不会属于同一个应用分组来提供负载均衡服务。提供不同的服务意味着需要暴露的端口和拒绝的端口是不同的，建议不同的操作系统尽量归属于不同的安全组。

例如，对于Linux操作系统，可能需要暴露TCP（22）端口来实现SSH，对Windows可能需要开通TCP（3389）远程桌面连接。

除了不同的操作系统归属不同的安全组，即便同一个镜像类型，提供不同的服务，如果之间不需要通过内网进行访问，建议也划归不同的安全组。这样方便解耦，并对未来的安全组规则进行变更，做到职责单一。

在规划和新增应用时，除了考虑划分不同的虚拟交换机配置子网，同时也应该合理地规划安全组。使用网段+安全组约束自己作为服务提供者和消费者的边界。

具体的变更流程请参见上面的操作步骤。

- 生产环境和测试环境使用不同的安全组

  为了更好的做系统的隔离，在实际开发过程中，您可能会构建多套的测试环境和一套线上环境。为了更合理地做网络隔离，您需要对不同的环境配置使用不同的安全策略，避免因为测试环境的变更刷新到线上，从而影响线上的稳定性。

  通过创建不同的安全组，限制应用的访问域，避免生产环境和测试环境联通。同时也可以对不同的测试环境分配不同的安全组，避免多套测试环境之间互相干扰，提升开发效率。

仅对需要公网访问的云服务器分配公网IP

不论是经典网络还是专有网络（VPC）中，合理地分配公网IP可以让系统更加方便地进行公网管理，同时减少系统受攻击的风险。在专有网络的场景下，创建虚拟交换机时，建议您尽量将需要公网访问的服务区的IP区间放在固定的几个交换机（子网 CIDR）中，方便审计和区分，避免不小心暴露公网访问。

在分布式应用中，大多数应用都有不同的分层和分组，对于不提供公网访问的云服务器尽量不提供公网IP，如果是有多台服务器提供公网访问，建议您配置公网流量分发的负载均衡服务来公网服务，提升系统的可用性，避免单点。详情请参见负载均衡服务。

对于不需要公网访问的云服务器尽量不要分配公网IP。专有网络中当您的云服务器需要访问公网的时候，优先建议您使用NAT网关，用于为VPC内无公网IP的ECS实例提供访问互联网的代理服务，您只需要配置相应的SNAT规则即可为具体的CIDR网段或者子网提供公网访问能力，避免因为只需要访问公网的能力而在分配了公网IP（EIP）之后也向公网暴露了服务。具体配置，请参见创建和管理SNAT条目。

最小原则

安全组应该是白名单性质的，所以需尽量开放和暴露最少的端口，同时尽可能少地分配公网IP。若想访问线上机器进行任务日志或错误排查的时候直接分配公网IP，挂载EIP虽然简便，但是会将整个机器暴露在公网之上，更安全的策略是通过跳板机来管理。

使用跳板机

跳板机由于其自身的权限巨大，除了通过工具做好审计记录。在专有网络中，建议将跳板机分配在专有的虚拟交换机之中，对其提供相应的EIP或者NAT端口转发表。

首先创建专有的安全组SG_BRIDGE，例如开放相应的端口，例如 Linux TCP（22）或者Windows RDP（3389）。为了限制安全组的入网规则，可以限制能登录的授权对象为企业的公网出口范围，减少被登录和扫描的概率。

然后将作为跳板机的云服务器加入到该安全组中。为了让该机器能访问相应的云服务器，可以配置相应的组授权。例如在SG_CURRENT添加一条规则允许SG_BRIDGE访问某些端口和协议。

使用跳板机SSH时，建议您优先使用SSH密钥对登录。详情请参见SSH密钥对。

总之，合理的安全组规划使您在扩容应用时更加游刃有余，同时让您的系统更加安全。

8.4. ECS数据安全最佳实践

本文档从使用云服务器ECS的角度出发，结合相关产品和运维架构经验，介绍如何保障云端的数据安全。

适用对象

本文档适用于刚开始接触阿里云的个人或者中小企业用户。

定期备份数据

数据备份是容灾的基础，可以降低因系统故障、操作失误以及安全问题而导致数据丢失的风险。ECS自带的快照功能可满足大部分用户数据备份的需求。您可根据自身业务需求选择创建快照的方式。具体步骤请参见手动创建快照和执行或取消自动快照策略。

建议您每日创建一次自动快照，每次快照至少保留7天。养成良好的备份习惯，在故障发生时可以迅速恢复重要数据，减少损失。

合理设计安全域

您可以基于VPC专有网络，构建自定义专属网络，隔离企业内部不同安全级别的服务器，避免互通网络环境下受其他服务器影响。

建议您创建一个专有网络，选择自有 IP 地址范围、划分网段、配置路由表和网关等。然后将重要的数据存储在一个跟互联网网络完全隔离的内网环境，日常可以用弹性IP（EIP）或者跳板机的方式对数据进行管理。具体步骤请参见创建专有网络。

设置安全组规则

安全组是重要的网络安全隔离手段，用于设置单台或多台云服务器的网络访问控制。通过设置安全组规则，可以在网络层过滤服务器的主动/被动访问行为，限定服务器对外/对内的端口访问，授权访问地址，从而减少攻击面，保护服务器的安全。

例如：Linux系统默认远程管理端口22，不建议直接向外网开放，可以通过配置ECS公网访问控制，只授权本地固定IP对服务器进行访问。如果您对访问控制有更高要求，可以使用第三方VPN产品对登录行为进行数据加密。

增加口令复杂度

弱口令容易导致数据泄露，因为弱口令是最容易出现和最容易被利用的漏洞之一。因此建议服务器的登录口令至少设置8位以上，从字符种类上增加口令复杂度，如包含大小写字母、数字和特殊字符等，并且要不定时更新口令，养成良好的安全运维习惯。

保护服务器端口安全

服务器给互联网提供服务的同时会暴露对应的服务端口。从安全管理的角度来说，开启的服务端口越多，越不安全。建议只对外提供必要的服务端口，并修改常见端口为高端口（30000以后），再对提供服务的端口做访问控制。

例如：数据库服务尽量在内网环境使用，避免暴露在公网。如果必须要在公网访问，则需要修改默认连接端口3306为高端口，并根据业务授权可访问的客户端地址。

防护系统漏洞

系统漏洞问题是长期存在的安全风险，可以通过系统补丁程序，或者安骑士补丁修复。Windows系统需要一直开启补丁更新，Linux系统要设置定期任务，通过执行 `yum update -y` 来更新系统软件包及内核。安骑士如何修复漏洞，请参见安骑士补丁管理。

云盾旗下的安骑士产品具有识别并防御非法破解密码行为的功能，避免被黑客入侵，批量维护服务器安全。安骑士能针对服务器应用软件安全方面提供配置检测和修复方案，提高服务器安全强度。详细功能介绍请参见安骑士产品功能列表。

防护应用漏洞

应用漏洞是指针对Web应用、缓存、数据库、存储等服务，通过利用渗透攻击而非法获取数据的一种安全缺陷。常见应用漏洞包括：SQL注入、XSS跨站、Webshell上传、后门隔离保护、命令注入、非法HTTP协议请求、常见Web服务器漏洞攻击、核心文件非授权访问、路径穿越等。应用漏洞不同于系统漏洞，修复难度很大，需要在设计应用前就充分考虑应用安全基线问题。因此建议通过接入Web应用防火墙（Web Application Firewall，简称 WAF），来轻松应对各类Web应用攻击，确保网站的Web安全与可用性。如何部署并使用WAF，请参见Web应用防火墙。

收集安全风险信息

在互联网安全领域，安全工程师和黑客比拼的就是时间。云安全中心是一种基于大数据的安全服务，即在大规模云计算环境中，对可能引发网络安全威胁的要素进行全面、快速和准确地捕获和分析，然后将客户当前遇到的安全威胁与过去的威胁进行关联、回溯和大数据分析，最终预测未来可能发生的威胁安全的风险事件，并提供一个体系化的安全解决方案。详细信息请参见快速掌握ECS安全态势。

所以，技术人员除了在做好日常安全运维的同时，还要掌握全面的信息，提升预警能力，在发现安全问题后可以及时修复和处理，真正保证云服务器ECS的数据安全闭环。

8.5. 提高ECS实例的安全性

您可以把一个ECS实例等同于一台虚拟机，本地维护的虚拟机一般会做虚拟机实例级别的安全防护，防止攻击和入侵等。同样的，ECS实例也需要安全性防护。除了置身于阿里云自身的安全平台外，您需要根据实际的需求进一步强化安全方案。

前提条件

使用本教程进行操作前，请确保您已经注册了阿里云账号。如还未注册，请先完成账号注册。

背景信息

如果ECS实例没有设置安全防护，可能会带来许多不良的影响。例如：

- 受到DDoS攻击而导致业务中断。
- 受到Web入侵而导致网页被篡改、挂马。
- 被注入而导致信息和数据泄漏等，影响ECS的使用，使其无法正常提供服务。

您可以通过以下方式提高ECS实例的安全性：

- 设置安全组
- 开启DDoS基础防护服务
- 接入云安全中心
- 接入Web应用防火墙

设置安全组

安全组是一种虚拟防火墙，具备状态检测包过滤功能。设置安全组的作用如下：

- 设置单台或多台云服务器的网络访问控制。安全组规则可以允许或者禁止与安全组相关联的ECS实例的公网和内网的入出方向的访问。
- 如果没有正确设置安全组或者安全组规则过于开放，则降低了访问的限制级别，存在安全隐患。

完成以下操作，为ECS实例所在安全组添加安全组规则：

1. 登录ECS管理控制台。

2. 在左侧导航栏，选择**网络与安全 > 安全组**。
3. 在顶部菜单栏左上角处，选择地域。
4. 找到要配置授权规则的安全组，在**操作**列中，单击**配置规则**。
5. 选择**入方向**页签。

   此处以专有云网络的安全组为例，如果是经典网络的安全组请选择**公网入方向**。
6. 单击**手动添加**。
7. 在弹出的对话框中，设置参数。

   例如：只允许特定IP远程登录到ECS实例，只需要在公网入方向配置规则即可。以Linux服务器为例，设置只让特定IP访问22端口。

   i. 单击**手动添加**。

   ii. 添加一条公网入方向安全组规则。

   规则内容如下：

   - **授权策略**选择**允许**。
   - **协议类型**选择**自定义 TCP**。
   - **端口范围**设置为*22/22*。
   - **授权对象**设置为允许远程连接的IP地址段，格式为x.x.x.x/xx，即IP地址/子网掩码。本示例中的地址段为*10.x.x.x/32*。优先级为*1*。

   iii. 单击**保存**。

   iv. 重复以上步骤，再添加一条公网入方向安全组规则。

   规则内容如下：

   - **授权策略**选择**拒绝**。
   - **协议类型**选择**自定义 TCP**。
   - **端口范围**设置为*22/22*。
   - **授权对象**设置为*0.0.0.0/0*。优先级为*2*。

设置安全组规则后，可以实现以下效果：

- 来自IP地址10.x.x.x访问22端口优先执行优先级为1的允许规则。
- 来自其他IP访问22端口优先执行优先级为2的拒绝规则。

开启DDoS基础防护服务

DDoS（Distributed Denial of Service，即分布式拒绝服务）攻击指借助于客户/服务器技术，联合多个计算机作为攻击平台，对一个或多个目标发动攻击，成倍地提高拒绝服务攻击的威力，影响业务和应用对用户提供服务。

阿里云云盾可以防护SYN Flood、UDP Flood、ACK Flood、ICMP Flood、DNS Flood、CC攻击等3到7层DDoS的攻击。DDoS基础防护免费提供高达5GB的默认DDoS防护能力。ECS实例默认开启DDoS基础防护服务。使用DDoS基础防护服务，无需采购昂贵清洗设备，受到DDoS攻击时不会影响访问速度，带宽充足不会被其他用户连带影响，保证业务可用和稳定。ECS实例创建后，您可以设置清洗阈值，具体步骤请参见设置流量清洗阈值。

在此基础上，阿里云推出了安全信誉防护联盟计划，将基于安全信誉分进一步提升DDoS防护能力，您可获得高达100GB以上的免费DDoS防护资源。您可以在云盾DDoS基础防护控制台中查看您账号当前的安全信誉分以及安全信誉详情和评分依据。详情请参见安全信誉防护联盟。

接入云安全中心

云安全中心是一个实时识别、分析、预警安全威胁的统一安全管理系统，通过防勒索、防病毒、防篡改、合规检查等安全能力，实现威胁检测、响应、溯源的自动化安全运营闭环，保护云上资产和本地主机并满足监管合规要求。

Agent插件是云安全中心提供的本地安全插件，您必须在要防护的服务器上安装该插件才能使用云安全中心的服务。如何安装Agent插件，请参见安装Agent。

> **说明** 在购买ECS实例时，选择**安全加固**即可自动安装Agent并开通云安全中心基础版，无需您手动安装Agent。

云安全中心自动为您开通基础版功能。基础版仅提供主机异常登录检测、漏洞检测、云产品安全配置项检测，如需更多高级威胁检测、漏洞修复、病毒查杀等功能，请登录云安全中心控制台。

接入Web应用防火墙

云盾Web应用防火墙（Web Application Firewall，简称WAF）基于云安全大数据能力实现，通过防御SQL注入、XSS跨站脚本、常见Web服务器插件漏洞、木马上传、非授权核心资源访问等OWASP常见攻击，过滤海量恶意CC攻击，避免您的网站资产数据泄露，保障网站的安全与可用性。

接入Web应用防火墙的好处如下：

- 无需安装任何软、硬件，无需更改网站配置、代码，它可以轻松应对各类Web应用攻击，确保网站的Web安全与可用性。除了具有强大的Web防御能力，还可以为指定网站做专属防护。适用于在金融、电商、o2o、互联网+、游戏、政府、保险等各类网站的Web应用安全防护上。
- 如果缺少WAF，只有前面介绍的防护措施，会存在短板，例如在面对数据泄密、恶意CC、木马上传篡改网页等攻击的时候，不能全面地防护，可能会导致Web入侵。

接入Web应用防火墙的具体步骤，请参见部署WAF防护。

阿里云为ECS实例的安全性提供了这么多的安全产品保驾护航，您可以根据实际需要选择相应的产品，加强对系统和数据的防护，减少ECS实例受到侵害，使其稳定、持久地运行。

8.6. 经典网络内网实例互通设置方法

安全组是实例级别防火墙，为保障实例安全，设置安全组规则时要遵循最小授权原则，下面介绍四种安全的内网实例互通设置方法。

方法一：使用单IP地址授权

- 适用场景：适用于小规模实例间内网互通场景。
- 优点：以IP地址方式授权，安全组规则清晰，容易理解。
- 缺点：内网互通实例数量较多时，会受到安全组规则条数200条的限制，并且后期维护工作量比较大。

设置方法如下：

1. 找到需要互通的实例，单击实例ID。
2. 在**实例详情**页，单击**安全组**页签。
3. 找到需要配置的安全组，单击对应的**配置规则**。
4. 单击**入方向**页签。
5. 单击**手动添加**。
6. 按以下描述添加安全组规则。
   - **授权策略**：允许。
   - **优先级**：按需设置，默认为1。
   - **协议类型**：按需选择协议类型。
   - **端口范围**：按需设置端口范围。
   - **授权对象**：输入想要内网互通的实例的内网 IP 地址，格式必须是*a.b.c.d/32*。其中，子网掩码必须是*/32*。

7. 单击**保存**完成规则添加。

方法二：加入同一安全组

- 适用场景：如果您的应用架构比较简单，可以为所有的实例选择相同的普通安全组，绑定同普通一安全组的实例之间不用设置特殊规则，默认网络互通。
- 优点：安全组规则清晰。
- 缺点：仅适用于简单的应用网络架构，网络架构调整时授权方法要随之进行修改。

设置方法，请参见ECS实例加入安全组。

方法三：绑定互通安全组

- 适用场景：为需要互通的实例增加绑定一个专门用于互通的普通安全组，适用于多层应用网络架构场景。
- 优点：操作简单，可以迅速建立实例间互通，可应用于复杂网络架构。
- 缺点：实例需绑定多个安全组，安全组规则阅读性较差。

设置方法如下：

1. 新建一个普通安全组并命名，例如：互通安全组，不需要给新建的安全组添加任何规则。
2. 将需要互通的实例都添加绑定新建的互通安全组。利用同一普通安全组的实例之间默认互通的特性，达到内网实例互通的效果。

方法四： 安全组互信授权

- 适用场景：为需要互通的实例增加绑定一个专门用于互通的安全组，适用于多层应用网络架构场景。
- 优点：操作简单，可以迅速建立实例间互通，可应用于复杂网络架构。
- 缺点：实例需绑定多个安全组，安全组规则阅读性较差。

设置方法如下：

1. 找到需要互通的实例，单击实例ID。
2. 在**实例详情**页，单击**安全组**页签。
3. 找到需要配置的安全组，单击对应的**配置规则**。
4. 单击**入方向**页签。
5. 单击**手动添加**。
6. 按以下描述添加安全组规则。
   - **授权策略**：允许。
   - **优先级**：按需设置，默认为1。
   - **协议类型**：按需选择协议类型。
   - **端口范围**：按需设置端口范围。
   - **授权对象**：
     - 本账号授权：输入安全组ID。
     - 跨账号授权：输入账号ID和安全组ID，格式为*账号ID/安全组ID*。
7. 单击**保存**完成规则添加。

建议

如果前期安全组授权过大，建议采用以下流程收紧授权范围。

图中的**删除0.0.0.0**是指删除原来的允许*0.0.0.0/0*地址段的安全组规则。

如果安全组规则变更操作不当，可能会导致您的实例间通信受到影响，请在修改设置前备份您要操作的安全组规则，以便出现互通问题时及时恢复。

安全组映射了实例在整个应用架构中的角色，推荐按照应用架构规划防火墙规则。例如：常见的三层Web应用架构就可以规划三个安全组，将部署了相应应用或数据库的实例绑定对应的安全组：

- Web层安全组：开放80端口。
- APP层安全组：开放8080端口。
- DB层安全组：开放3306端口。

8.7. 修改服务器默认远程端口

本文介绍如何修改ECS实例的默认远程端口。

修改Windows系统实例默认远程端口

本节以Windows Server 2012为例介绍如何修改Windows系统实例默认远程端口。

1. 远程连接并登录到Windows实例。

   具体操作，请参见连接Windows实例。

2. 修改注册表子项Port Number的值。

   i. 按快捷键 *Win（Windows徽标键）+R*，启动**运行**窗口

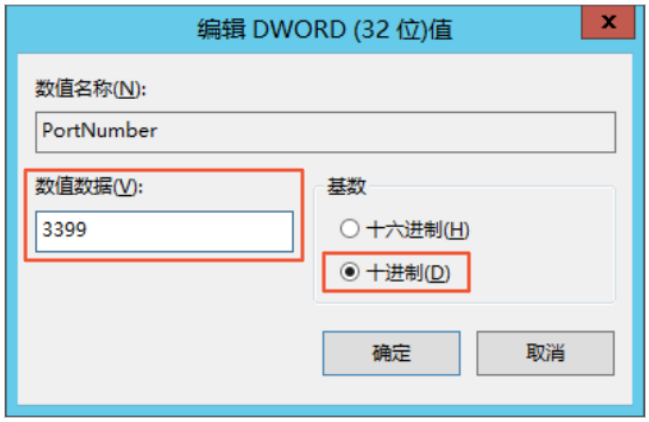
i. 按快捷键 Win+R（Windows徽标键+R），启动运行窗口。

ii. 输入*regedit.exe*后按回车键，打开注册表编辑器。

iii. 在左侧导航栏，选择**HKEY_LOCAL_MACHINE > System > CurrentControlSet > Control > Terminal Server > Wds > rdpwd > Tds > tcp**。

iv. 在右侧列表中找到注册表子项**PortNumber**并右键单击，选择**修改**。

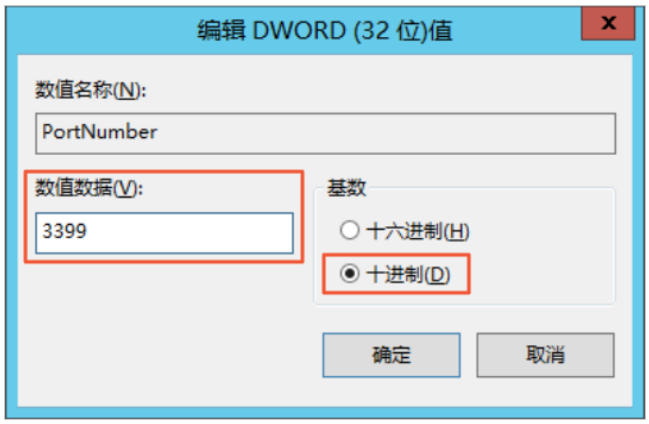
v. 在弹出的对话框中，在**数值数据**的文本框中输入新的远程端口号，在本示例中即3399。在**基数**区域单击**十进制**，然后单击**确定**。



vi. 在左侧导航栏，选择**HKEY_LOCAL_MACHINE > System > CurrentControlSet > Control > Terminal Server > WinStations > RDP-Tcp**。

vii. 在右侧列表中找到注册表子项**PortNumber**并右键单击，选择**修改**。

viii. 在弹出的对话框中，在**数值数据**的文本框中输入新的远程端口号，在本示例中即3399。在**基数**区域单击**十进制**，然后单击**确定**。



3. 在ECS管理控制台重启ECS实例。

   具体操作，请参见重启实例。

4. 为该实例添加安全组规则，允许新配置的远程端口进行连接。

   具体操作，请参见添加安全组规则。

5. 远程访问实例，在远程地址后面添加新远程端口号即可连接实例。

> **说明** 使用Mac远程桌面连接ECS实例仅支持默认的3389端口。

修改Linux系统实例默认远程端口

本节以CentOS 6.8和CentOS 7.7为例介绍如何修改Linux系统实例默认远程端口。

1. 远程连接并登录到Linux实例。

   具体操作，请参见通过密码或密钥认证登录Linux实例。

2. 运行以下命令备份sshd服务配置文件。

   ```
   cp /etc/ssh/sshd_config /etc/ssh/sshd_config_bak
   ```

3. 修改sshd服务的端口号。

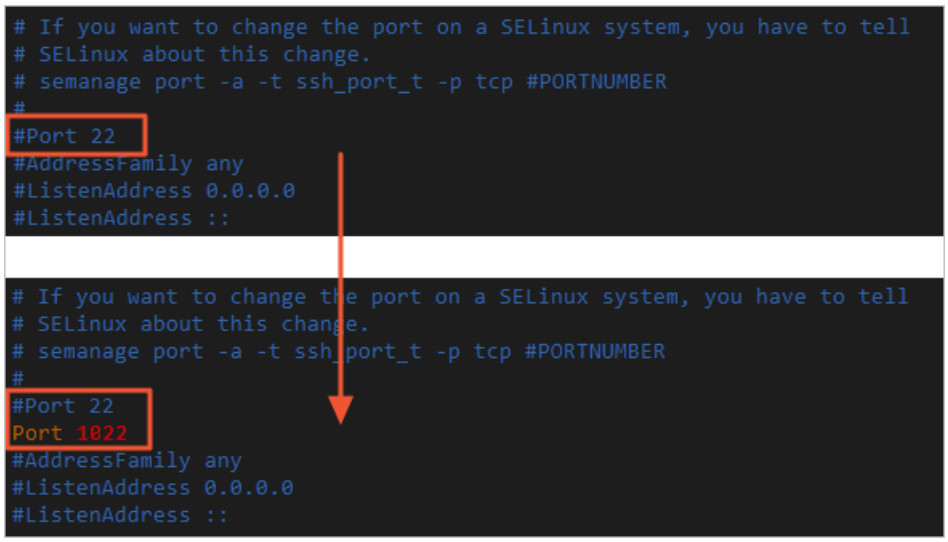
   i. 运行以下命令编辑*sshd_config*配置文件。

   ```
   vim /etc/ssh/sshd_config
   ```

   ii. 在键盘上按 `i` 键，进入编辑状态。

   iii. 添加新的远程服务端口。

   本节以1022端口为例。在 `Port 22` 下输入 `Port 1022` 。

   iv. 在键盘上按 `Esc` 键，输入**:wq**后保存并退出编辑状态。

4. 运行以下命令重启sshd服务。重启sshd服务后您可以通过1022端口SSH登录到Linux实例。

- CentOS 7及以上版本、Alibaba Cloud Linux 2：

```
systemctl restart sshd
```

- CentOS 6版本：

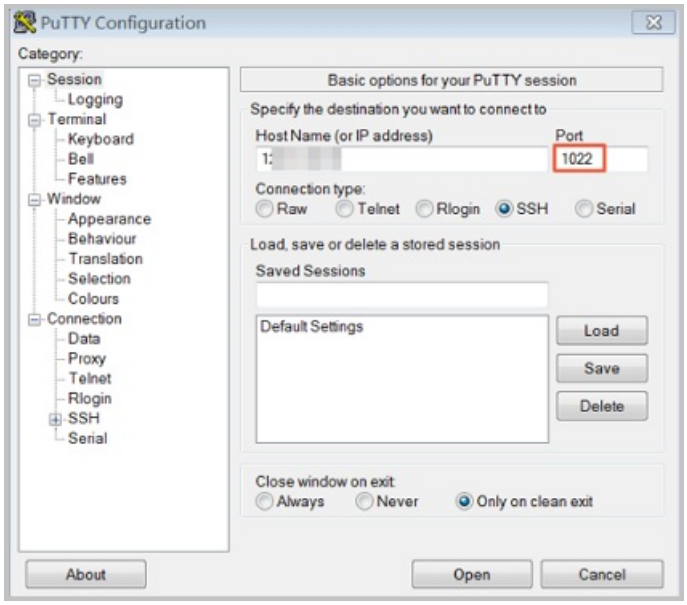
```
/etc/init.d/sshd restart
```

5. 配置实例的安全组放行TCP协议1022端口。

   具体操作，请参见添加安全组规则。

6. 使用SSH工具连接新端口，测试是否成功。

   登录时在**Port**文本框中输入修改后的端口号，在本示例中即1022。



> **说明**
>
> 修改完成后，您将无法使用默认的22端口远程访问ECS实例。

8.8. 使用Windows实例的日志

日志记录了系统中硬件、软件和系统问题的信息，同时还监视着系统中发生的事件。当ECS实例被入侵或者系统（应用）出现问题时，您可以根据日志迅速定位问题的关键，从而极大地提高工作效率和服务器的安全性。本文以Windows Server 2012 R2为例，介绍四种日志的使用和简要分析。

前提条件

使用本教程进行操作前，请确保您已经注册了阿里云账号。如还未注册，请先完成账号注册。

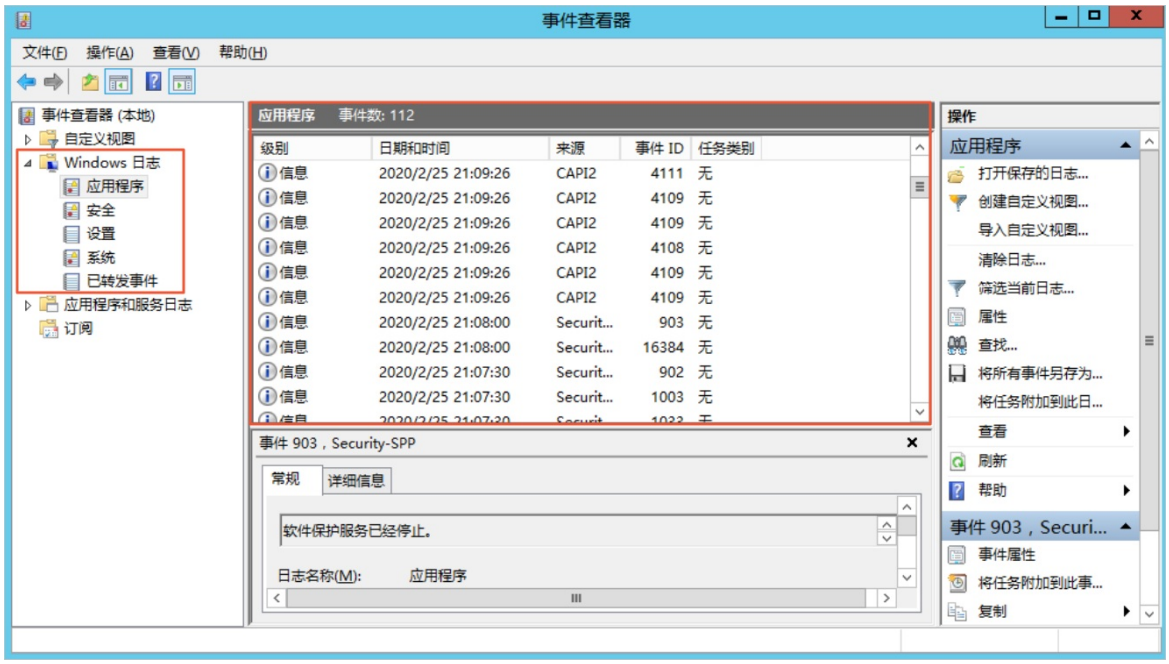
背景信息

Windows系统日志主要分为：系统日志、应用程序日志、安全日志以及应用程序和服务日志。

进入事件查看器查看日志

完成以下操作，进入事件查看器查看日志：

1. 单击**开始**，在底部单击下拉按钮，然后单击**运行**。
2. 在运行对话框中执行命令 `eventvwr` ，打开**事件查看器**。



3. 您可以在**事件查看器**里查看以下四种日志。

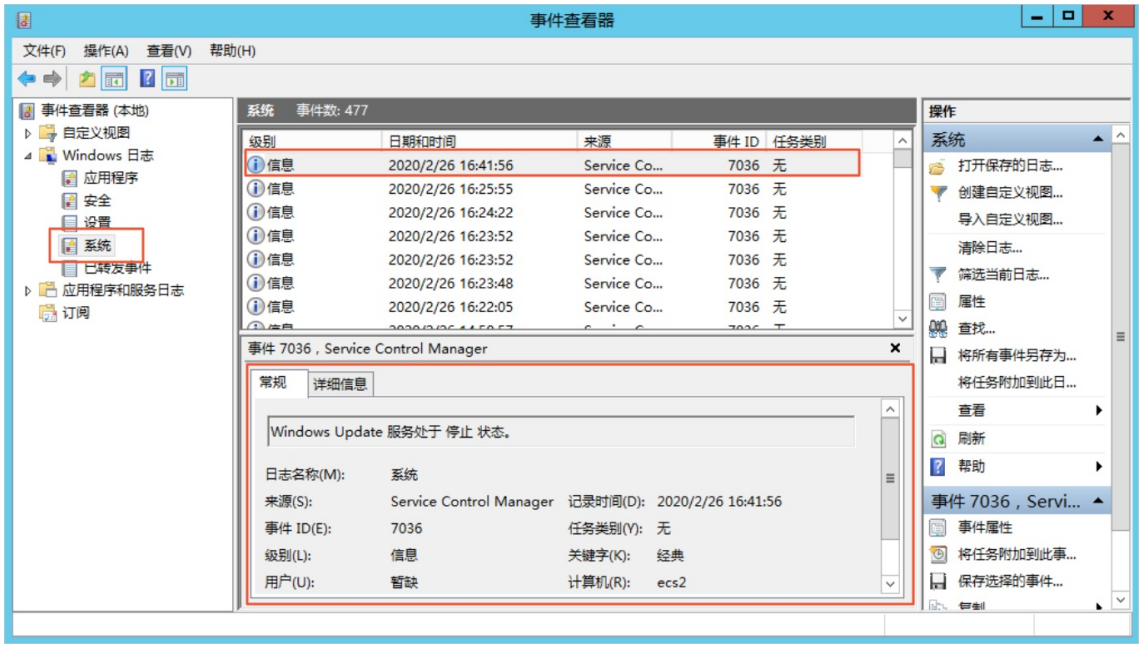
> **说明** 通过本文所述四种日志的查看方法找到的所有错误日志事件ID，您可以用于在微软知识库找到解决方法。

- 系统日志

  在左侧导航栏，单击**Windows 日志 > 系统**，查看系统日志。

  系统日志包含Windows系统组件记录的事件。例如，系统日志中会记录在启动过程中加载驱动程序或其他系统组件失败。

  系统组件所记录的事件类型由Windows预先确定。

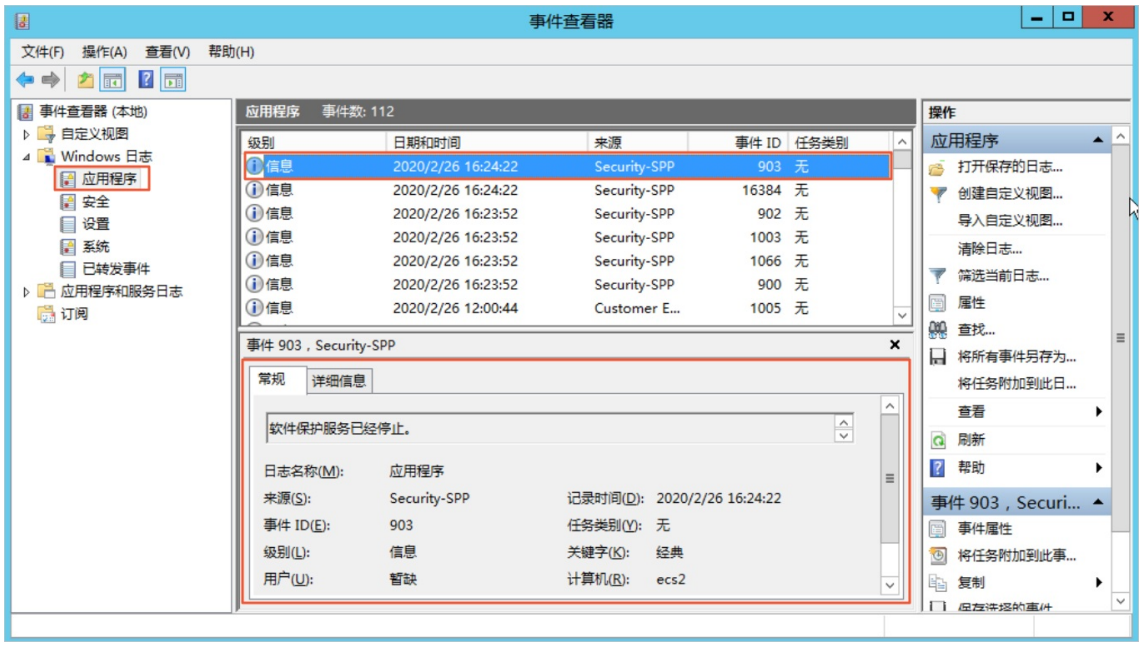


- 应用程序日志

  在左侧导航栏，单击**Windows 日志 > 应用程序**，查看应用程序日志。

  应用程序日志包含由应用程序或程序记录的事件。例如，数据库程序可在应用程序日志中记录文件错误。

  程序开发人员决定记录哪些事件。

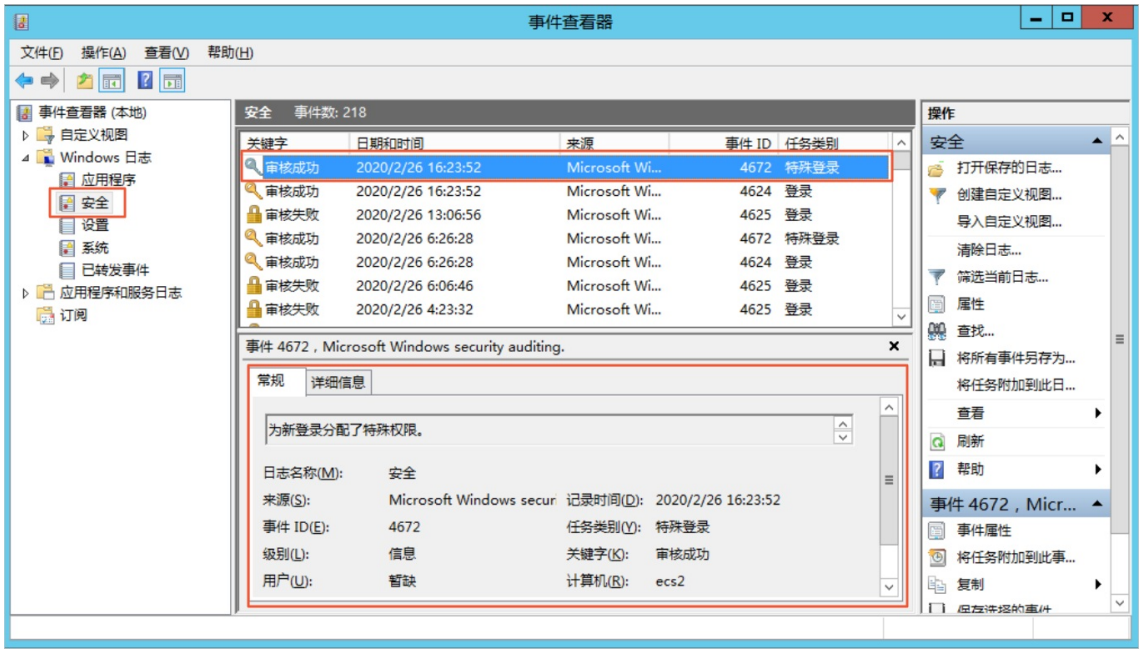


- 安全日志

  在左侧导航栏，单击**Windows 日志 > 安全**，查看安全日志。

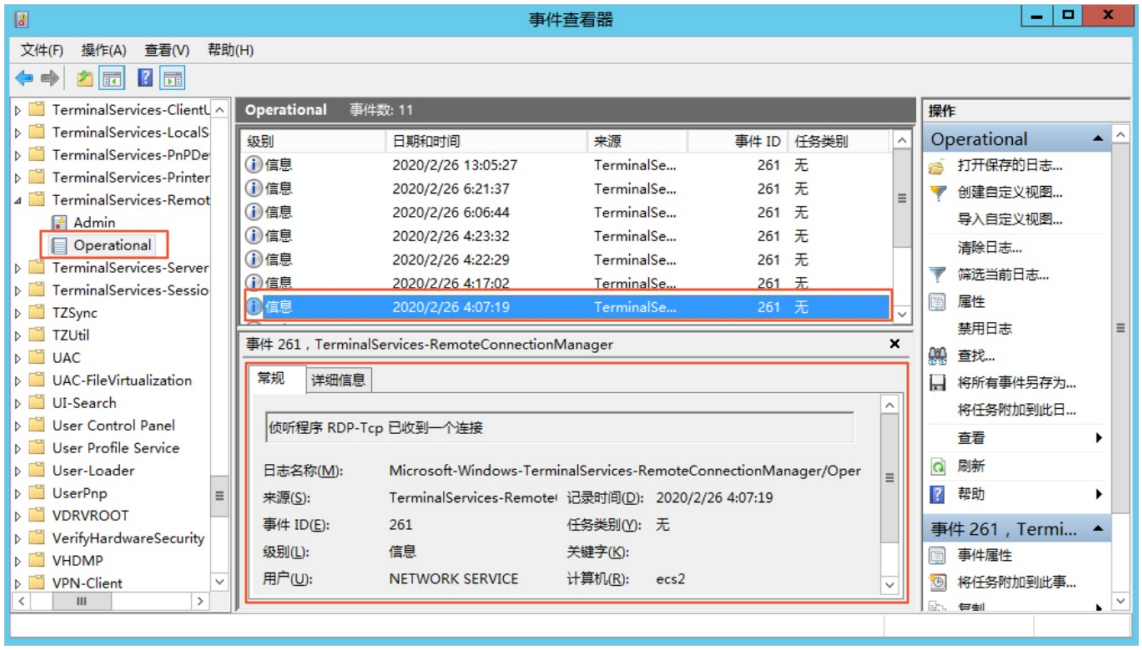
  安全日志包含诸如有效和无效的登录尝试等事件，以及与资源使用相关的事件，如创建、打开或删除文件或其他对象。

  管理员可以指定在安全日志中记录什么事件。例如，如果已启用登录审核，则安全日志将记录对系统的登录尝试。



- 应用程序和服务日志

  应用程序和服务日志是一种新类别的事件日志。这些日志存储来自单个应用程序或组件的事件，而非可能影响整个系统的事件。



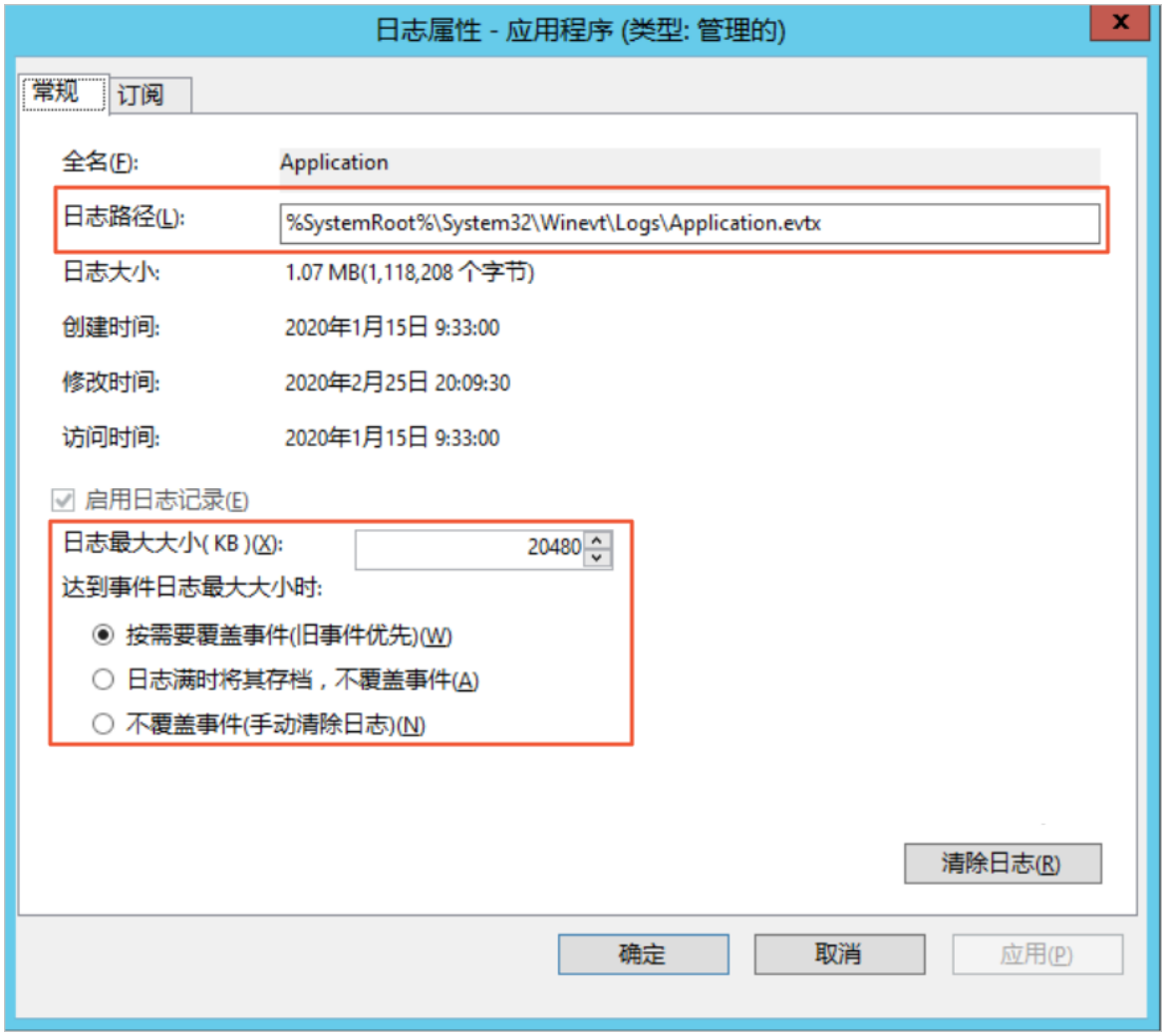
修改日志路径并备份日志

日志默认保存在系统盘里面。日志最大值默认是20 MB，超过20 MB时会覆盖之前的事件。您可以根据自己的需求修改。

按以下步骤修改日志路径并备份日志。

1. 在**事件查看器**窗口，在左侧导航栏里，单击**Windows 日志**。
2. 在右边列表中，右键单击一个日志名称，选择**属性**。

3. 在**日志属性** 窗口，按界面显示修改以下信息：
   - 日志路径。
   - 日志最大大小。
   - 达到事件日志最大大小时系统应采取的操作。



相关文档

- 云服务器ECS Windows安全审计日志简要说明

8.9. 普通安全组内网络隔离

安全组是一种虚拟防火墙，具备状态检测和包过滤功能。加入同一个普通安全组内的实例之间默认允许所有协议、端口的互相访问。为了满足普通安全组内实例之间网络隔离的需求，阿里云丰富了安全组网络连通策略，实现普通安全组组内网络隔离。

安全组内网络隔离规则

安全组内默认的网络连通策略如下：

- 普通安全组内的实例之间允许所有协议、端口的互相访问，企业安全组内的实例之间网络隔离。
- 不同安全组的实例之间默认网络隔离。

> 说明 如果分属于不同安全组的实例之间需要互相访问，可以通过安全组规则授权。

针对普通安全组内的实例之间默认网络互通的情况，您可以修改普通安全组内的网络连通策略，实现组内隔离。具体步骤可参见修改策略。

设置安全组内网络隔离时，需注意以下事项：

- 仅设置指定的普通安全组内的网络隔离，不改变默认的网络连通策略， 即其他已有和新建的普通安全组，以及企业安全组仍采用默认策略。
- 安全组内网络隔离是网卡之间的隔离，而不是ECS实例之间的隔离。若实例上绑定了多张弹性网卡，需设置每个网卡所属安全组的组内网络隔离。
- 安全组内网络隔离的优先级最低，即设置组内网络隔离后，仅在安全组内没有任何自定义规则的情况下保证组内实例之间网络隔离。

  以下情况，安全组内实例之间仍然可以互相访问：
  - 实例同时归属于多个安全组时，有一个及以上的安全组未设置组内隔离。
  - 既设置了安全组内隔离，又设置了让组内实例之间可以互相访问的ACL。

  > **说明** ACL相关内容请参见网络ACL概述。

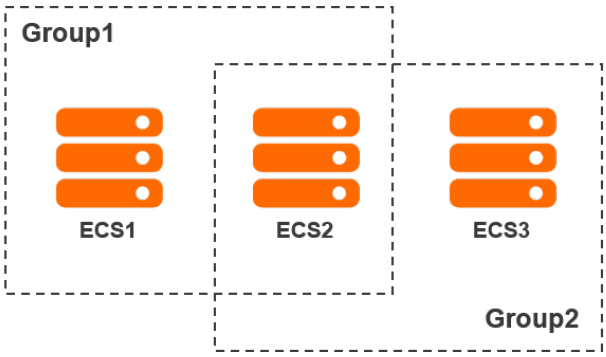
修改策略

您可以使用ModifySecurityGroupPolicy接口来修改普通安全组内的网络连通策略，实现组内隔离。

> **说明** 企业安全组默认组内隔离，不支持修改组内网络连通策略。

案例分析

本示例中，Group1、Group2分别为2个不同的普通安全组，ECS1、ECS2、ECS3分别为3个不同的ECS实例。实例和实例所属的安全组的关系如下：



- Group1：包含ECS1和ECS2，设置组内网络隔离。
- Group2：包含ECS2和ECS3，保持默认，即网络互通。

各实例间的网络连通情况如下：

实例	网络连通情况	说明
ECS1和ECS2	隔离	ECS1和ECS2同时属于Group1。Group1的策略是组内隔离，所以ECS1和ECS2之间网络隔离。
ECS2和ECS3	互通	ECS2和ECS3同时属于Group2。Group2的策略是默认互通，所以ECS2和ECS3之间网络互通。
ECS1和ECS3	隔离	ECS1和ECS3分属不同的安全组，不同安全组的实例之间默认网络不通，所以ECS1和ECS3之间网络隔离。

8.10. 安全组五元组规则

安全组用于设置单台或多台ECS实例的网络访问控制，它是重要的网络安全隔离手段，用于在云端划分安全域。安全组五元组规则能精确控制源IP、源端口、目的IP、目的端口以及传输层协议。

背景信息

在最初设计安全组规则时：

- 安全组入规则只支持：源IP地址、目的端口、传输层协议。
- 安全组出规则只支持：目的IP地址、目的端口、传输层协议。

在多数应用场景下，该安全组规则简化了设置，但存在如下弊端：

- 无法限定入规则的源端口范围，默认放行所有源端口。
- 无法限定入规则的目的IP地址，默认放行安全组下的所有IP地址。
- 无法限定出规则的源端口范围，默认放行所有源端口。
- 无法限定出规则的源IP地址，默认放行安全组下的所有IP地址。

五元组规则定义

五元组规则包含：源IP地址、源端口、目的IP地址、目的端口、传输层协议。

五元组规则完全兼容原有的安全组规则，能更精确的控制源IP地址、源端口、目的IP地址、目的端口以及传输层协议。

五元组出规则示例如下：

```
源IP地址：172.16.1.0/32
源端口：22
目的IP：10.0.0.1/32
目的端口：不限制
传输层协议：TCP
授权策略：Drop
```

示例中的出规则表示禁止 `172.16.1.0/32` 通过22端口对 `10.0.0.1/32` 发起TCP访问。

应用场景

- 某些平台类网络产品接入第三方厂商的解决方案为用户提供网络服务，为了防范这些产品对用户的ECS实例发起非法访问，则需要在安全组内设置五元组规则，更精确的控制出流量和入流量。
- 设置了组内网络隔离的安全组，如果您想精确控制组内若干ECS实例之间可以互相访问，则需要在安全组内设置五元组规则。

配置五元组规则

您可以使用OpenAPI设置五元组规则。

- 增加安全组入规则，请参见AuthorizeSecurityGroup。
- 增加安全组出规则，请参见AuthorizeSecurityGroupEgress。
- 删除安全组入规则，请参见RevokeSecurityGroup。
- 删除安全组出规则，请参见RevokeSecurityGroupEgress。

参数说明

在授权或解除授权时，各参数的含义如下表所示。

参数	入规则中各参数含义	出规则中各参数含义
SecurityGroupId	当前入规则所属的安全组ID，即目的安全组ID。	当前出规则所属的安全组ID，即源安全组ID。
DestCidrIp	目的IP范围，可选参数。 • 如果指定DestCidrIp，则可以更精细地控制入规则生效的目的IP范围； • 如果不指定DestCidrIp，则入规则生效的IP范围就是SecurityGroupId这个安全组下的所有IP。	目的IP，DestGroupId与DestCidrIp二者必选其一，如果二者都指定，则DestCidrIp优先级高。
PortRange	目的端口范围，必选参数	目的端口范围，必选参数。
DestGroupId	不允许输入。目的安全组ID一定是SecurityGroupId。	目的安全组ID。DestGroupId与DestCidrIp二者必选其一，如果二者都指定，则DestCidrIp优先级高。
SourceGroupId	源安全组ID，SourceGroupId与SourceCidrIp二者必选其一，如果二者都指定，则SourceCidrIp优先级高。	不允许输入，出规则的源安全组ID一定是SecurityGroupId。
SourceCidrIp	源IP范围，SourceGroupId与SourceCidrIp二者必选其一，如果二者都指定，则SourceCidrIp优先级高。	源IP范围，可选参数。 • 如果指定SourceCidrIp，则会更精细地限定出规则生效的源IP。 • 如果不指定SourceCidrIp，则生效的源IP就是SecurityGroupId这个安全组下的所有IP。
SourcePortRange	源端口范围，可选参数，不填则不限制源端口。	源端口范围，可选参数，不填则不限制源端口。

8.11. 通过云防火墙控制ECS实例间访问

云防火墙可以统一管理ECS实例之间（东西向）、互联网和ECS实例之间（南北向）的流量。本文介绍如何配置主机边界防火墙并查看业务关系。

前提条件

- 已注册阿里云账号。如还未注册，请先完成账号注册。
- 在使用主机边界防火墙前，您需要授权云防火墙访问云资源。具体操作，请参见授权云防火墙访问云资源。
- 在使用主机边界防火墙前，您需要确保云防火墙为企业版或旗舰版。具体操作，请参见云防火墙计费方式。

背景信息

云防火墙提供防火墙一键开关、入侵检测、主动外联阻断、流量分析、日志等功能，包括主机边界防火墙、互联网边界防火墙和VPC边界防火墙。更多云防火墙概念介绍，请参见云防火墙和云防火墙词汇表。

主机边界防火墙作用于东西向流量，底层使用了ECS安全组的能力。您可以在云防火墙控制台为主机边界防火墙设置内对内策略组，也可以在ECS控制台的安全组中设置规则，来控制东西向（即，ECS实例之间）的访问。云防火墙和ECS安全组的配置自动保持同步。您还可以设置应用组，直观查看ECS实例间的访问关系，从而根据访问情况优化内对内策略。

互联网边界防火墙作用于南北向流量，在互联网和ECS实例间进行访问控制。您可以按需设置外对内和内对外策略，在入侵防御的基础上进行策略加固，请参见网络流量活动概览和访问控制策略概览。

以下场景建议您使用云防火墙：

- 基于域名的访问控制。
- 基于应用的访问控制。
- 对失陷主机的主动外联进行自动阻断。
- 因等保需求，需要近6个月的访问日志。

配置主机边界防火墙

在云防火墙控制台发布策略组后，数据立即同步到安全组，但是在ECS控制台配置安全组后，数据每天在固定时间同步到策略组，次日在策略组生效。购买企业版或旗舰版云防火墙后，您可以在云防火墙控制台统一维护东西向的访问控制策略。

完成以下操作，配置主机边界防火墙：

1. 登录云防火墙控制台。
2. 在左侧导航栏，单击**访问控制**。
3. 单击**主机边界防火墙**。
   - **策略组来源**表示了策略组的来源。
   - **自定义**表示在云防火墙中创建。
   - **同步安全组**表示同步自ECS安全组。
   - **同步应用组**表示同步自应用组。
4. 单击**新增策略组**。
5. 配置**策略组类型**、**策略组名称**、**所属VPC**、**实例ID**、**描述**和**模板**，然后单击**提交**。

   > **说明** 配置**所属VPC**后，地域也确定为VPC所属的地域，例如**华东1（杭州）**。

6. （可选）在策略组**操作**列下单击**配置策略**，按照业务需求新建策略。
7. 在策略组**操作**列下单击**发布**，发布成功后即同步到ECS安全组。按照以下步骤查看同步效果：
   i. 登录ECS管理控制台。
   ii. 选择策略组所在地域，例如**华东1（杭州）**。
   iii. 在左侧导航栏，单击**网络与安全 > 安全组**。
   iv. 搜索维度选择**安全组名称**，在文本框中输入策略组名称，然后单击**搜索**，出现同名安全组即表示同步成功。

主机边界防火墙配置完成后，即开始控制ECS实例间的访问。在云防火墙中，您还可以设置应用组，可视化呈现业务关系。

查看业务关系

在云防火墙中，业务区是东西向业务中构成用户某个业务的各个应用组的集合，例如门户网站业务区可能包含Web应用组、DB应用组等。应用组是东西向业务中提供的相同/相似服务的应用集合，例如所有部署了MySQL的ECS实例归属到同一个DB应用组，部署了Apache服务的ECS实例归属到同一个Web应用组。

完成以下操作，查看当前ECS实例之间的关系：

1. 登录云防火墙控制台。
2. 在左侧导航栏，单击**业务可视 > 自定义分组**。
3. 创建业务区。
   i. 单击**业务区**。
   ii. 单击**新建业务区**。
   iii. 完成业务区配置，然后单击**确定**。
   业务区配置项如下表所示。

配置项	示例
名称	DB业务、Web业务
备注	无
程度	**非常重要**

4. 创建应用组。
   i. 单击**应用组**。
   ii. 单击**新建应用组**。
   iii. 完成应用组配置，然后单击**确定**。
   应用组配置项如下表所示。

配置项	示例
名称	DB应用组、Web应用组
备注	无
程度	**非常重要**
业务区	**选择已有业务区**
名称	DB业务、Web业务

5. 分配应用。
   i. 选择VPC，例如华东 1 - vpc-xxx。
   ii. 根据业务需要分配应用，例如将部署了MySQL的ECS实例分配至DB应用组，将部署了Apache服务的ECS实例分配至Web应用组。
6. 在左侧导航栏，单击**业务关系**。
7. 选择VPC，例如华东 1 - vpc-xxx，即可查看不同业务区的访问关系。您也可以进入应用组和应用层级查看访问关系。

相关文档

相关文档

- 配置外到内流量只允许访问某个端口的访问控制策略
- 云防火墙中控蠕虫防御最佳实践
- 云防火墙数据库防御最佳实践

8.12. 开启或关闭SELinux

安全增强型Linux（SELinux）是一个Linux内核的功能，它提供支持访问控制的安全政策保护机制。本文介绍如何开启或关闭SELinux，并且避免系统无法启动的问题。

前提条件

已使用阿里云公共镜像或者自定义镜像创建了ECS实例。

> **说明** 如果自定义镜像是通过服务器迁移中心SMC，由源服务器迁移产生的镜像，或者是您导入的本地镜像文件，请确保迁移前源服务器中的SELinux是禁用状态。

背景信息

一般情况下，开启SELinux会提高系统的安全性，但是会破坏操作系统的文件，造成系统无法启动的问题。如果您所在的企业或团队具有十分严格的安全策略，要求在Linux操作系统中开启SELinux，您可以参考本文的步骤开启，避免系统无法启动的问题。本教程使用的操作系统为：CentOS 7.2 64位。

开启SELinux

1. 以root权限远程连接ECS实例。

   关于连接方式的介绍，请参见连接方式概述ECS远程连接操作指南。

2. 在ECS实例上运行以下命令，编辑SELinux的 `config` 文件。

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

3. 找到 `SELINUX=disabled` ，按 `i` 进入编辑模式，通过修改该参数开启SELinux。

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

您可以根据需求修改参数，开启SELinux有以下两种模式：

- 强制模式 `SELINUX=enforcing` ：表示所有违反安全策略的行为都将被禁止。
- 宽容模式 `SELINUX=permissive` ：表示所有违反安全策略的行为不被禁止，但是会在日志中作记录。

4. 修改完成后，按下键盘 `Esc` 键，执行命令 `:wq` ，保存并退出文件。

> **说明** 修改 `config` 文件后，需要重启实例，但直接重启实例将会出现系统无法启动的错误。因此在重启之前需要在根目录下新建 `autorelabel` 文件。

5. 在根目录下新建隐藏文件 `autorelabel` ，实例重启后，SELinux会自动重新标记所有系统文件。

```
touch /.autorelabel
```

6. 重启ECS实例。

```
shutdown -r now
```

验证SELinux状态

1. 以root权限远程连接ECS实例。

   关于连接方式的介绍，请参见连接方式概述ECS远程连接操作指南。

2. 运行命令 `getenforce` ，验证SELinux状态。

   返回状态应为 `enforcing` 或者 `permissive` ，本教程当前状态为 `enforcing` 。

```
[root@ecs01 conf]# getenforce
Enforcing
```

3. 运行命令 `sestatus` ，获取更多SELinux信息。

```
[root@ecs01 conf]# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      28
```

参数信息`SELinux status`显示为`enabled`，表示SELinux已启动。

关闭SELinux

1. 以root权限远程连接ECS实例。

   关于连接方式的介绍，请参见连接方式概述ECS远程连接操作指南。

2. 运行命令`getenforce`，验证SELinux状态。

   返回状态如果是`enforcing`，表明SELinux已开启。

3. 选择临时关闭或者永久关闭SELinux。
   - 执行命令`setenforce 0`临时关闭SELinux。
   - 永久关闭SElinux。
     a. 运行以下命令，编辑SELinux的`config`文件。

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

     b. 找到`SELINUX=enforcing`，按`i`进入编辑模式，将参数修改为`SELINUX=disabled`。

     ```
     # This file controls the state of SELinux on the system.
     # SELINUX= can take one of these three values:
     #     enforcing - SELinux security policy is enforced.
     #     permissive - SELinux prints warnings instead of enforcing.
     #     disabled - No SELinux policy is loaded.
     SELINUX=enforcing
     # SELINUXTYPE= can take one of three two values:
     #     targeted - Targeted processes are protected,
     #     minimum - Modification of targeted policy. Only selected processes are protected.
     #     mls - Multi Level Security protection.
     SELINUXTYPE=targeted
     ```

     c. 修改完成后，按下键盘`Esc`键，执行命令`:wq`，保存并退出文件。

     d. 重启ECS实例。

     ```
     shutdown -r now
     ```

     e. 重启后，运行命令`getenforce`，验证SELinux状态为`disabled`，表明SELinux已关闭。

后续步骤

您可以通过已启动SELinux的ECS实例创建自定义镜像。在您需要的时候，直接通过该镜像创建已启动SELinux的ECS实例。

8.13. 通过API撤销不同账号下的ECS实例内网通信

若您在同一地域下授权过不同账号的ECS实例内网通信，可以通过API接口撤销安全组授权。

前提条件

- 已注册阿里云账号。如还未注册，请先完成账号注册。
- 请确保您已经为ECS实例安装了阿里云CLI，在不同操作系统中安装CLI的方式请参见：
  - 在Windows上安装阿里云CLI
  - 在Linux上安装阿里云CLI
  - 在macOS上安装阿里云CLI

背景信息

本文通过调用RevokeSecurityGroup接口撤销已授权的安全组规则。在操作之前，您需要准备以下信息：

- 账号名：您登录ECS管理控制台的账号名称。
- ECS实例所在的安全组ID：已授权账号内网互通的ECS实例所在的安全组。

  您可以在ECS管理控制台查看，也可以通过调用DescribeSecurityGroupReferences接口查询。
- ECS实例所在的地域名称：取值请参见。本文示例设置为*cn-beijing*，即华北 2（北京）地域。

假设两个账号的信息如下表所示。

账号	账号名	安全组	安全组ID
账号A	a@aliyun.com	sg1	sg-bp1azkttqpldxgtedXXX
账号B	b@aliyun.com	sg2	sg-bp15ed6xe1yxeycg7XXX

除了撤销授权不同账号下的ECS实例内网通信，您也可以重新授权。详情请参见通过API允许不同账号下的ECS实例内网通信。

操作步骤

1. 账号A运行以下命令。

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

2. 账号B运行以下命令。

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

相关文档

- RevokeSecurityGroup
- DescribeSecurityGroupReferences

8.14. 通过API允许不同阿里云账号下的ECS实例内网通信

若您需要实现同一地域下不同阿里云账号的ECS实例内网通信，可以参考本文描述授权安全组间互访。

前提条件

请确保您已经为ECS实例安装了阿里云CLI，在不同操作系统中安装CLI的方式请参见：

- 在Windows上安装阿里云CLI
- 在Linux上安装阿里云CLI
- 在macOS上安装阿里云CLI

背景信息

目前授权安全组内网通信有以下两种，请根据您的实际需求选择方式。

- ECS实例间通信：授权同一阿里云账号两台ECS实例间的内网通信。
- 阿里云账号间内网通信：授权不同阿里云账号同一地域下两个安全组内所有的ECS实例的内网通信，包括授权以后购买的同一安全组内的ECS实例。

> **说明** 阿里云账号间内网通信实际上是安全组间授权，即授权处于这两个安全组内的ECS实例后就可以实现内网通信。修改安全组配置会影响到安全组内所有的ECS实例，请根据实际需要进行操作，避免影响到ECS实例网络下运行的业务。

安全组是ECS实例的虚拟防火墙，安全组本身不提供通信能力和组网能力。授权不同安全组内的实例内网通信后，请同时确保实例可以建立内网互通的能力。

- 若实例均是经典网络类型，必须位于同一地域下。
- 若实例均是VPC类型，不同VPC间默认内网不通。建议通过公网访问的方式通信，或者通过高速通道、VPN网关和云企业网等方式提供访问能力。更多信息，请参见高速通道、VPN网关和云企业网。
- 若实例网络类型不同，请设置ClassicLink允许实例通信。具体操作，请参见经典网络和专有网络互通。
- 若实例位于不同地域，建议通过公网访问的方式通信，或者通过高速通道、VPN网关和云企业网等方式提供访问能力。更多信息，请参见高速通道、VPN网关和云企业网。

ECS实例间通信

1. 查询两台ECS实例的内网IP地址和两台ECS实例所处的安全组ID。

   您可以通过控制台或调用DescribeInstances接口获得ECS实例所属的安全组ID。假设两台ECS实例的信息如下表所示。

实例	IP地址	所属安全组	安全组ID
实例A	10.0.0.1	sg1	sg-bp1azkttqpldxgte****
实例B	10.0.0.2	sg2	sg-bp15ed6xe1yxeycg****

2. 在sg1安全组中添加放行10.0.0.2的入方向的规则。

```
aliyun ecs AuthorizeSecurityGroup --SecurityGroupId sg-bp1azkttqpldxgte**** --RegionId
cn-qingdao --IpProtocol all  --PortRange=-1/-1 --SourceCidrIp 10.0.0.2 --NicType intran
et
```

3. 在sg2安全组中添加放行10.0.0.1的入方向的规则。

```
aliyun ecs AuthorizeSecurityGroup --SecurityGroupId sg-bp15ed6xe1yxeycg**** --RegionId
cn-qingdao --IpProtocol all  --PortRange=-1/-1 --SourceCidrIp 10.0.0.1 --NicType intran
et
```

> 说明
> - 以上命令中，地域取值为华北 1（青岛）*cn-qingdao*，请您根据实际情况修改。
> - 以上命令中，调用AuthorizeSecurityGroup接口添加安全组入方向的放行规则，主要关注的参数为SecurityGroupId和SourceCidrIp。

4. 等待一分钟后，使用**ping**命令测试两台ECS实例之间是否内网互通。

阿里云账号间内网通信

1. 查询两个阿里云账号的账号名称和两个账号下对应的安全组ID。

   您可以通过控制台或调用DescribeInstances接口获得ECS实例所属的安全组ID。假设两个阿里云账号的信息如下表所示。

阿里云账号	阿里云账号ID	安全组	安全组ID
账号A	testA****@aliyun.com	sg1	sg-bp1azkttqpldxgte****
账号B	testB****@aliyun.com	sg2	sg-bp15ed6xe1yxeycg****

2. 在sg1安全组中添加放行sg2安全组入方向的规则。

```
aliyun ecs AuthorizeSecurityGroup --SecurityGroupId sg-bp1azkttqpldxgte**** --RegionId
cn-qingdao --IpProtocol all  --PortRange=-1/-1 --SourceGroupId sg-bp15ed6xe1yxeycg7XXX
--SourceGroupOwnerAccount b@aliyun.com --NicType intranet
```

3. 在sg2安全组中添加放行sg1安全组入方向的规则。

```
aliyun ecs AuthorizeSecurityGroup --SecurityGroupId sg-bp15ed6xe1yxeycg**** --RegionId
cn-qingdao --IpProtocol all  --PortRange=-1/-1 --SourceGroupId sg-bp1azkttqpldxgtedXXX
--SourceGroupOwnerAccount a@aliyun.com --NicType intranet
```

> 说明
> - 以上命令中，地域取值为华北 1（青岛）*cn-qingdao*，请您根据实际情况修改。
> - 以上命令中，调用AuthorizeSecurityGroup接口添加安全组入方向的放行规则时，主要关注的参数为SecurityGroupId、SourceGroupId和SourceGroupOwnerAccount。

4. 等待一分钟后，使用ping命令测试查看两台ECS实例之间是否内网互通。

8.15. 对安全组规则的合规性进行审计和预警

您可以在配置审计（Config）中创建审计规则，对安全组规则的合规性进行审计，并将审计结果投递到日志服务SLS中。当账号下的安全组规则有不符合审计规则的变更时，您能够通过日志服务SLS收到告警通知。本文以安全组不允许全网段开启22端口和不允许入方向全通两个审计规则为例进行介绍。

效果速览

通过本文的配置，您可以对账号下已有安全组规则进行合规性审计，也能够在安全组规则发生不合规变更时，实时监控和告警。

- 账号下对已有安全组规则的审计不合规结果，可以在配置审计的控制台查看，例如查看“安全组不允许全网段开启22端口”的审计结果如下：

- 账号下安全组规则的不合规变更，可以通过配置钉钉、短信、电话等不同方式进行不同等级的告警。例如“安全组不允许全网段开启22端口”审计的钉钉告警通知效果如下：

操作步骤

通过配置审计和日志服务对安全组规则进行审计和告警通知的操作步骤如下：

1. 步骤一：开通配置审计并添加审计规则
2. 步骤二：将资源不合规事件投递到日志服务
3. 步骤三：配置告警
4. （可选）步骤四：升级告警并优化内容模板

5. 步骤五：完成配置与告警测试

步骤一：开通配置审计并添加审计规则

开启配置审计和日志服务，并为安全组规则的合规性添加审计规则。

1. 开通配置审计和日志服务。
   - 开通配置审计。具体操作，请参见配置审计服务授权。
   - 开通日志服务。具体操作，请参见开通日志服务。
2. 选择配置审计的监控范围。
   i. 登录配置审计控制台。
   ii. 在左侧导航栏，选择**设置 > 监控范围**。
   iii. 检查确保已选资源类型中包含**Ecs 安全组**。

3. 配置安全组不允许全网段开启22端口的审计规则。
   i. 在左侧导航栏，单击**规则**。
   ii. 在**规则**页面，单击**新建规则**。
   iii. 在**新建规则**页面的筛选输入框中输入**安全组**，筛选出安全组相关的审计规则。
   iv. 找到规则名称为**安全组不允许对全部网段开启风险端口**的托管审计规则，单击**应用规则**。

v. 在**基本属性**页面，设置规则名称、风险等级和备注，单击**下一步**。

托管规则的名称、风险等级和备注信息均为系统默认，您可以根据实际情况进行修改。

vi. 在**评估资源范围**页面，**已选资源类型**默认为**Ecs 安全组**，单击**下一步**。

您也可以根据资源ID、资源组ID、地域和Tag等信息，对审计规则生效的资源做筛选。

vii. 在**参数设置**页面的**期望值**文本框中输入22，单击**下一步**。

期望值为22表示：如果在您的账号下某个安全组对全部网段开启22端口，会审计为不合规。

viii. 在**修正设置**页面，单击**下一步**。

对于支持修正设置的托管规则，您可以选中**修正设置**前面的复选框。更多信息，请参见修正设置。

ix. 在**预览并保存**页面，确认规则设置后，单击**提交**。

x. 单击**返回规则列表**，查看新建的规则。

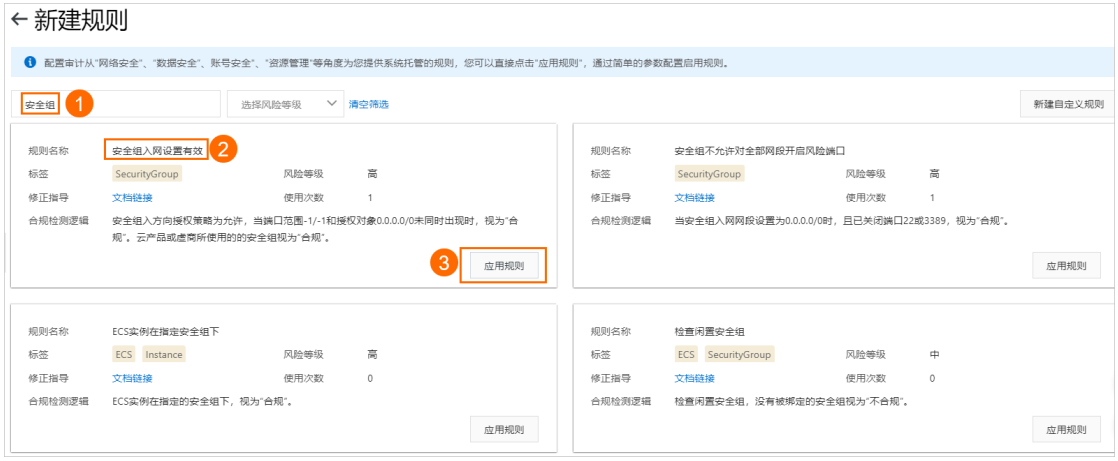
您可以在规则列表中查看新建的**安全组不允许全网段开启22端口**规则，其运行状态为**应用中**。

4. 配置安全组不允许入方向全通的审计规则。

i. 在**规则**页面，单击**新建规则**。

ii. 在**新建规则**页面的筛选输入框中输入**安全组**，筛选出安全组相关的审计规则。

iii. 找到规则名称为**安全组入网设置有效**的托管审计规则，单击**应用规则**。



iv. 在**基本属性**页面，设置规则名称、风险等级和备注，单击**下一步**。

托管规则的名称、风险等级和备注信息均为系统默认，您可以根据实际情况进行修改。

v. 在**评估资源范围**页面，**已选资源类型**默认为**Ecs 安全组**，单击**下一步**。

您也可以根据资源ID、资源组ID、地域和Tag等信息，对审计规则生效的资源做筛选。

vi. 在**参数设置**页面，单击**下一步**。

vii. 在**修正设置**页面，单击**下一步**。

对于支持修正设置的托管规则，您可以选中**修正设置**前面的复选框。更多信息，请参见修正设置。

viii. 在**预览并保存**页面，确认规则设置后，单击**提交**。

ix. 单击**返回规则列表**，查看新建的规则。

您可以在规则列表中查看新建的**安全组不允许入方向全通**规则，其运行状态为**应用中**。

经过以上步骤，您已经建立了两个关于安全组规则的配置审计规则。 您可以基于配置审计的托管规则，或者自定义审计规则来建立更多的规则。

步骤二：将资源不合规事件投递到日志服务

配置审计规则配置后，您可以在配置审计控制台随时查看每条审计规则的检查结果。如果您需要及时收到安全组规则配置不符合审计规则的告警信息，可以基于日志服务SLS来实现。

1. 将资源不合规事件投递到日志服务。
   i. 登录配置审计控制台。
   ii. 在左侧导航栏，选择**投递服务 > 投递到日志服务SLS**。
   iii. 在**投递到日志服务SLS**页面，打开**设置日志服务SLS**开关。
   iv. 设置资源投递数据的相关参数，单击**确定**。
   - **选择接收内容**：选中**资源不合规事件**复选框。当资源的审计结果不合规时，配置审计向日志服务SLS投递资源不合规事件。
   - 选择**本账号中新建日志项目**。
   - **日志项目地域**：选择日志项目所在地域。
   - **日志项目名称**：输入日志服务SLS中日志项目的名称，例如config-alarm2。
   - **日志库名称**：输入日志服务SLS中日志库的名称，例如config-alarm2。

   更多信息，请参见设置投递数据到日志服务SLS。

   如下图所示，您可以单击日志库名称config-alarm2，进入日志服务控制台。

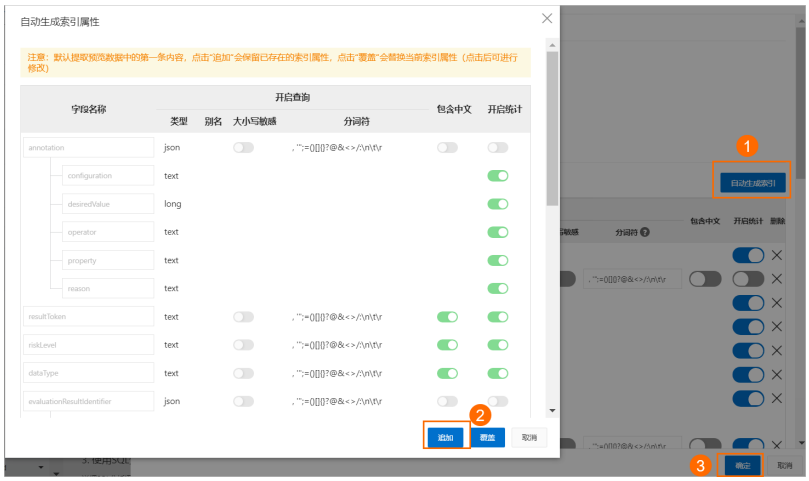
2. 配置日志服务日志库的索引。
   i. 登录日志服务控制台。
   ii. 找到新建的日志项目config-alarm2，单击其日志库config-alarm2。
   iii. 进入索引配置页面，单击**开启索引**。
   iv. 单击**自动生成索引**。

   如果提示“当前暂无数据，请确保有数据后再进行操作！”，请先创建安全组并添加不符合审计规则的安全组规则。具体操作，请参见创建安全组和修改安全组规则。

   修改不符合审计规则的安全组规则，例如配置安全组规则为允许全网段开启22端口或者允许入方向全通，以此来触发一条SLS日志投递。修改安全组规则后，等待约3分钟时间，系统会触发已配置的审计规则，并将不合规事件投递到日志服务中。

   v. 再次单击**自动生成索引**，日志服务会自动检测日志格式，并生成索引。

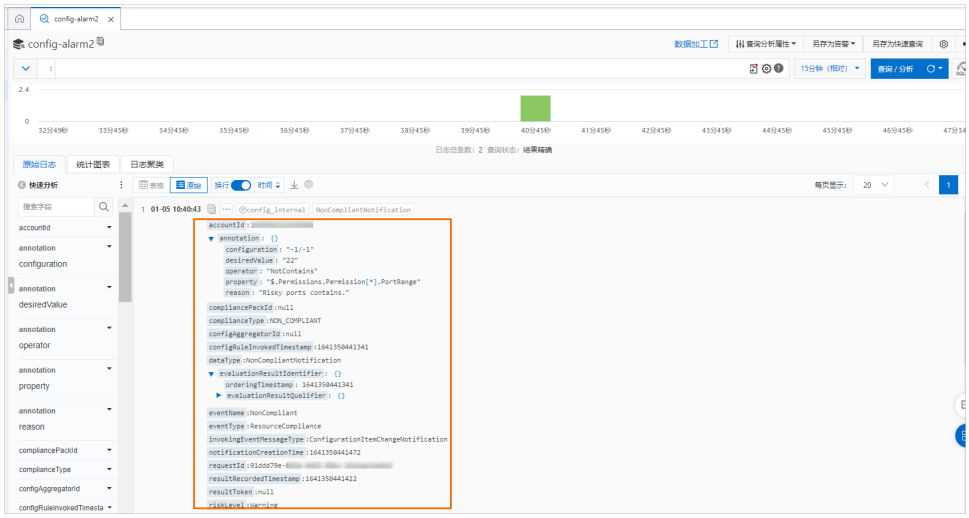
vi. 依次单击**追加**和**确定**，保存索引配置。



等待约1分钟，日志服务的索引生效。

> **说明** 因为索引配置仅对新产生的日志生效，您可以再次修改安全组规则，来触发一个不合规事件，从而在日志服务中查看日志投递。

vii. 单击**查询/分析**，从config-alarm2日志库中，可以搜索到一条不合规事件的日志。



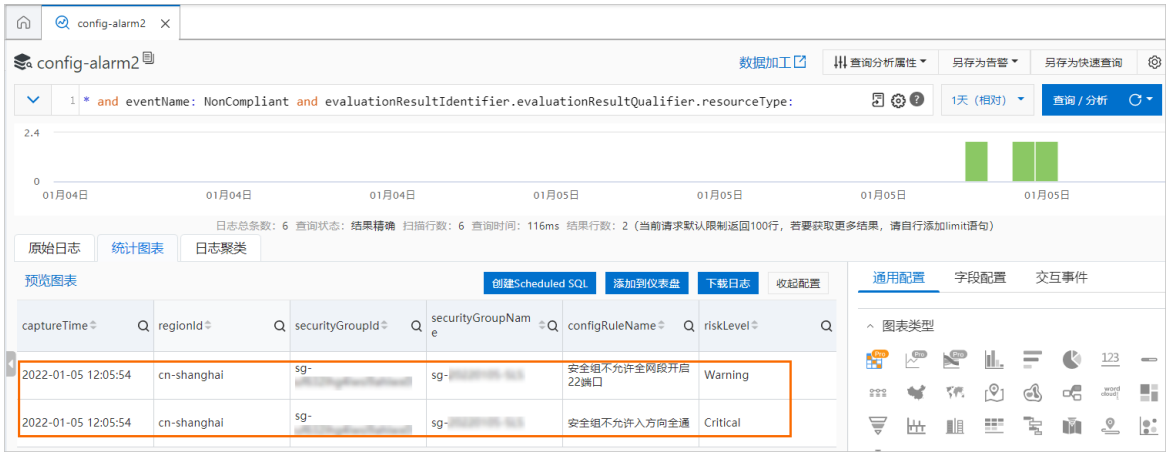
步骤三：配置告警

在日志服务SLS中配置告警，定期检查查询或分析结果，当检查结果满足预设条件时发送告警通知，实现实时的服务状态监控。

1. 登录日志服务控制台。
2. 找到新建的日志项目config-alarm2，单击其日志库config-alarm2。
3. 在搜索框中输入如下筛选条件，单击**查询/分析**。

```
* and eventName: NonCompliant and evaluationResultIdentifier.evaluationResultQualifier.
resourceType: SecurityGroup |
SELECT
DATE_FORMAT(MAX("evaluationResultIdentifier.evaluationResultQualifier.captureTime")/100
0, '%Y-%m-%d %H:%i:%s') captureTime,
"evaluationResultIdentifier.evaluationResultQualifier.regionId" regionId,
"evaluationResultIdentifier.evaluationResultQualifier.resourceId" securityGroupId,
"evaluationResultIdentifier.evaluationResultQualifier.resourceName" securityGroupName,
"evaluationResultIdentifier.evaluationResultQualifier.configRuleName" configRuleName,
riskLevel riskLevel
GROUP BY regionId, securityGroupId, securityGroupName, configRuleName, riskLevel
```

您能看到如下所示的筛选结果。

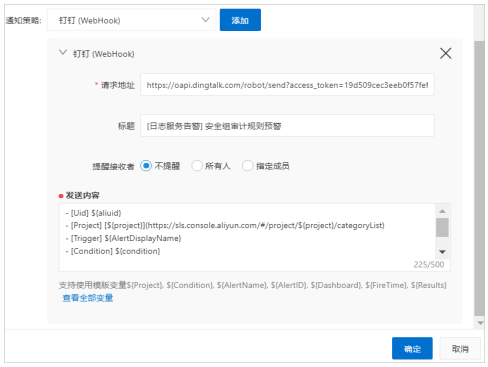


4. 在页面右上角，选择**另存为告警 > 旧版告警**，为查询结果设置告警。
5. 配置告警监控规则。
   - **触发条件**设置为 `securityGroupId!='null'` 。
   - **通知策略**选择**钉钉 (WebHook)**，并单击**添加**。
   - 其他参数根据实际情况配置。

   更多信息，请参见设置告警。

6. 配置钉钉通知方式，单击**确定**。

   更多信息，请参见通知方式。

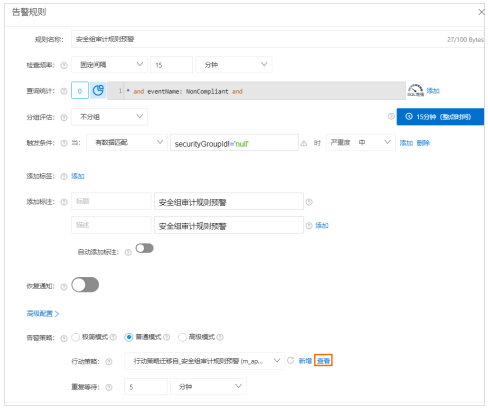


（可选）步骤四：升级告警并优化内容模板

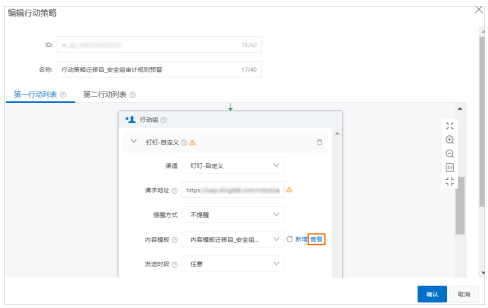
新版本告警相对于旧版告警，在升级原有功能的基础上扩展了告警监控、告警管理、通知（行动）管理的能力。将日志服务告警从旧版升级为新版，可以支持更加丰富的告警配置功能。更多信息，请参见旧版与新版区别。

1. 登录日志服务控制台。
2. 找到新建的日志项目config-alarm2，单击其日志库config-alarm2。
3. 在左侧导航栏中，单击**告警**。
4. 在**规则/事务**页签中，找到目标监控规则，在**操作**列中单击**升级**。

   如果您是首次使用新版告警，请根据页面提示配置存储中心，然后单击**确认**。
5. 配置内容模板。
   i. 在**规则/事务**页签，找到目标监控规则，在**操作**列中单击**编辑**。
   ii. 在**告警规则**对话框，单击**行动策略**后的**查看**。



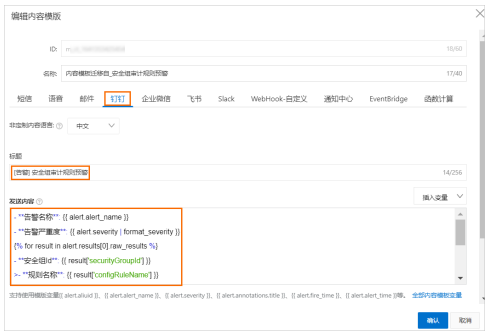
   iii. 在**编辑行动策略**对话框，单击**内容模板**后的**查看**，找到对应的内容模板。



iv. 在**编辑内容模板**对话框，修改以下内容，并单击**确认**保存内容模板。

- 将内容模板的标题修改为**[告警] 安全组规则审计预警**。
- 将发送内容修改为以下格式，提升告警通知的可读性。

```
- **告警名称**: {{ alert.alert_name }}
- **告警严重度**: {{ alert.severity | format_severity }}
{% for result in alert.results[0].raw_results %}
- **安全组Id**: {{ result['securityGroupId'] }}
 >- **规则名称**: {{ result['configRuleName'] }}
 >- **触发时间**: {{ result['captureTime'] }}
 >- **严重程度**: {{ result['riskLevel'] }}
 >- **地域**: {{ result['regionId'] }}
 >- **安全组名称**: {{ result['securityGroupName'] }}
{% endfor %}
```



步骤五：完成配置与告警测试

- 您可以在账号下创建一个安全组，并设置不符合审计规则的安全组规则，对告警的可用性进行测试。
- 完成以上配置后，在安全组规则不符合审计规则时，您就可以收到钉钉机器人的告警通知，如下图所示。从而提升您对于安全组规则变更的感知能力，提升您云上资产的安全性。

9.数据恢复

9.1. 解决Windows实例磁盘空间不足

本文主要介绍Windows实例磁盘空间不足时对应的解决方法以及磁盘日常维护的最佳实践。

背景信息

本文中的方法适用于Windows Server 2003以上系统，这里以Windows Server 2012 R2为例。

> **说明** Linux 实例磁盘空间不足时对应的处理方法请参见ECS Linux 磁盘空间满排查处理。

解决方法及最佳实践

解决Windows实例磁盘空间不足的问题，有以下两种处理方式：

- 释放磁盘空间
- 扩容磁盘

日常需要养成良好的磁盘使用习惯，本文推荐以下几个磁盘使用的最佳实践：

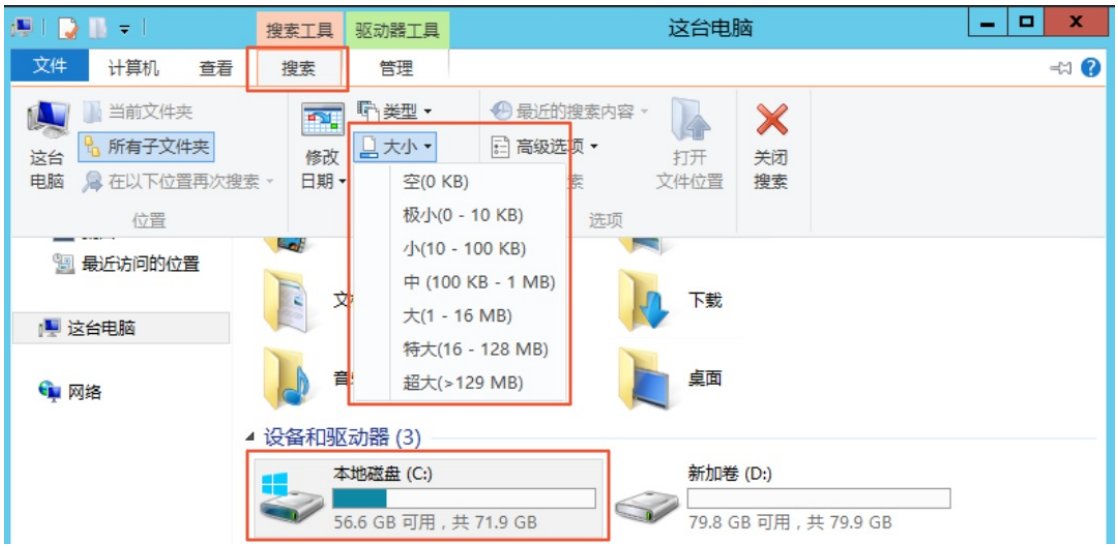
- 压缩保存文件
- 定期清理不必要的应用程序
- 设置磁盘监控

释放磁盘空间

您可以通过清理磁盘中不需要的文件来解决磁盘空间满的问题，具体操作步骤如下：

1. 找出占用磁盘空间过多的文件。
   i. 远程连接并登录到Windows实例。具体操作，请参见远程连接。
   ii. 单击**开始 > 这台电脑**。
   iii. 单击要清理的磁盘，按下键盘的**Ctrl+F**键。

iv. 单击顶部导航栏**搜索 > 大小**，然后根据系统定义大小筛选指定磁盘的大文件。



> **说明** 您也可以自定义文件大小范围进行检索，例如：
>
> - 在搜索框输入*大小：>500M*，会检索该磁盘大于500 M的文件。
> - 在搜索框输入*大小：> 100M < 500M*，会检索大于100 M但小于500 M的文件。

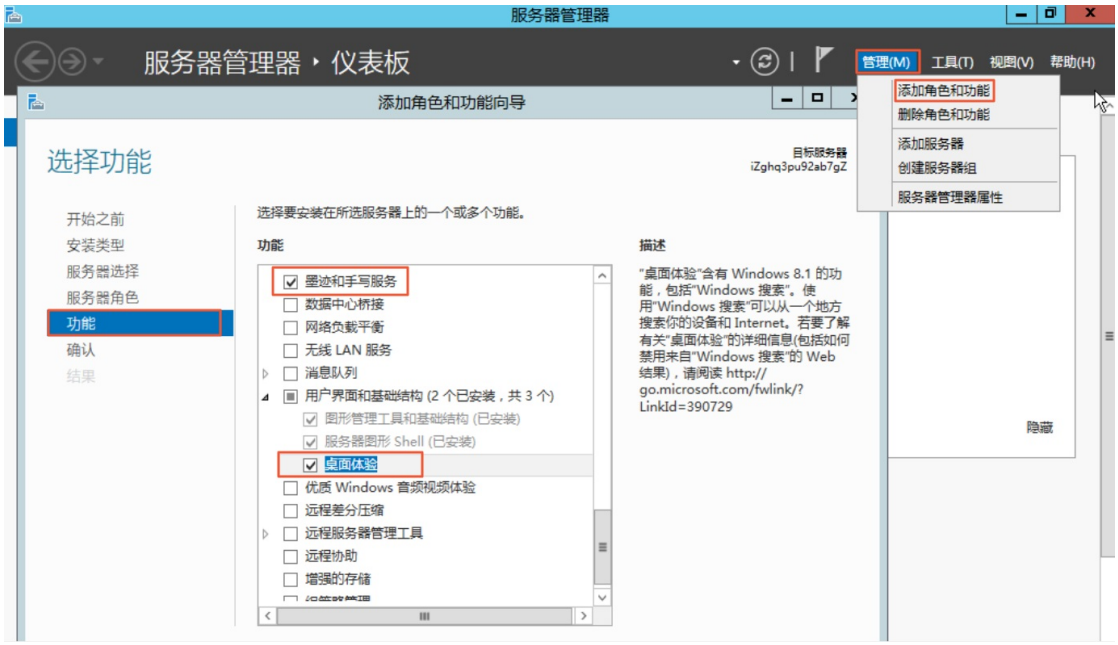
2. 删除不需要的文件。

   推荐您使用系统自带的磁盘清理工具，删除日志文件及系统上其他不需要文件，并清空回收站。磁盘清理工具服务器默认没有安装，需要手动安装，具体安装以及删除文件的步骤如下：

   i. 在底部任务栏单击**服务器管理器**图标，打开服务器管理器。

   ii. 单击右上方**管理 > 添加角色和功能**。

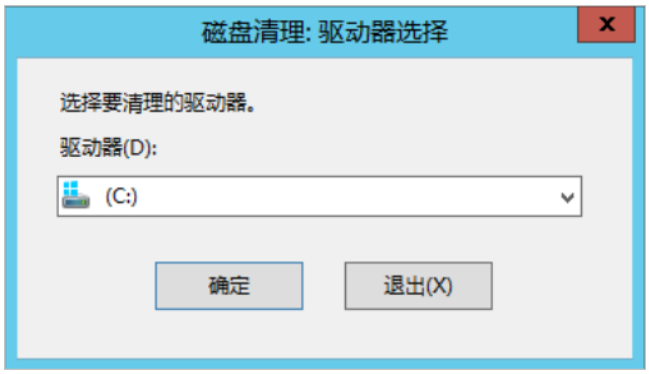
   iii. 在添加角色和功能向导页面，按默认设置单击**下一步**至**功能**模块，勾选**墨迹手写服务**和**桌面体验**，然后单击**下一步**。



   iv. 单击**安装**。

v. 安装成功后，系统将提示您手动重启服务器。您需要手动重启服务器。重新启动服务器之后，确认已安装了桌面体验。

vi. 单击**开始**，在顶部搜索栏搜索**磁盘清理**，选择要清理的磁盘，单击**确定**。



扩容磁盘

您可以通过扩容磁盘的方式解决磁盘空间满的问题。具体操作，请参见在线扩容云盘（Windows系统）或离线扩容云盘（Windows系统）。

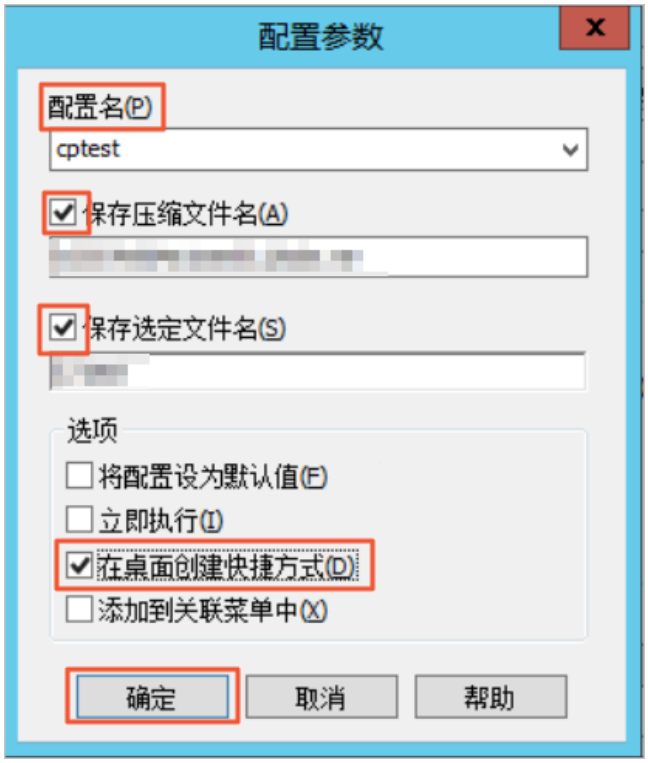
压缩保存文件

磁盘中一些定期生成的文件可以进行归档压缩后保存，以提高磁盘使用率。压缩工具推荐使用WinRAR，配置压缩策略过程如下。

1. 下载并安装WinRAR工具，官方下载地址请参见WinRAR。

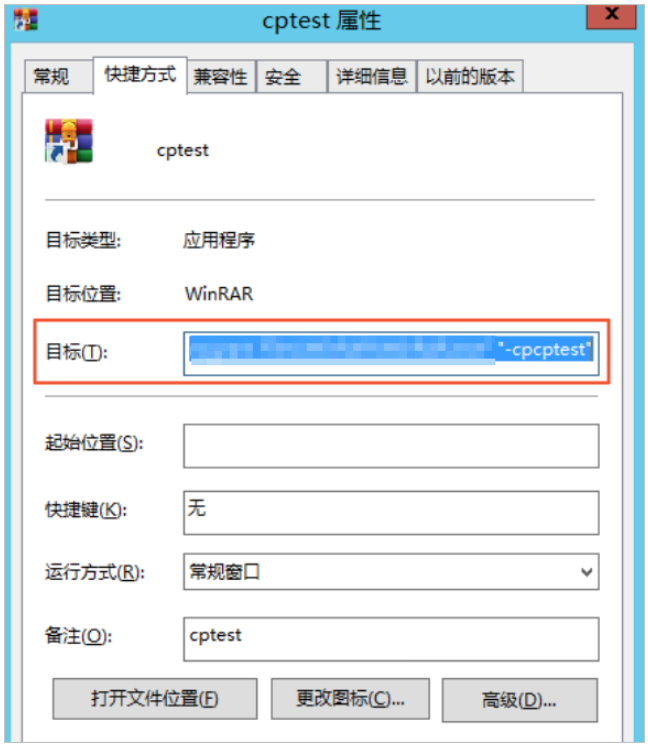
   本示例使用WinRAR中文版。

2. 安装WinRAR工具后，找到需要压缩的文件，右键单击该文件，选择**添加到压缩文件**。
3. 在**设置**界面单击窗口上方的**备份**选项卡，然后勾选**按掩码产生文件名**，注意此时不要单击**确定**。
4. 单击窗口上方的**常规**选项卡，单击**浏览**来定义压缩文件的路径。单击**配置**，选择**保存当前配置为新配置**。
5. 在弹出的**配置参数**对话框中，输入**配置名**，选中**保存压缩文件名**、**保存选定文件名**、**桌面创建快捷方式**，单击**确定**。

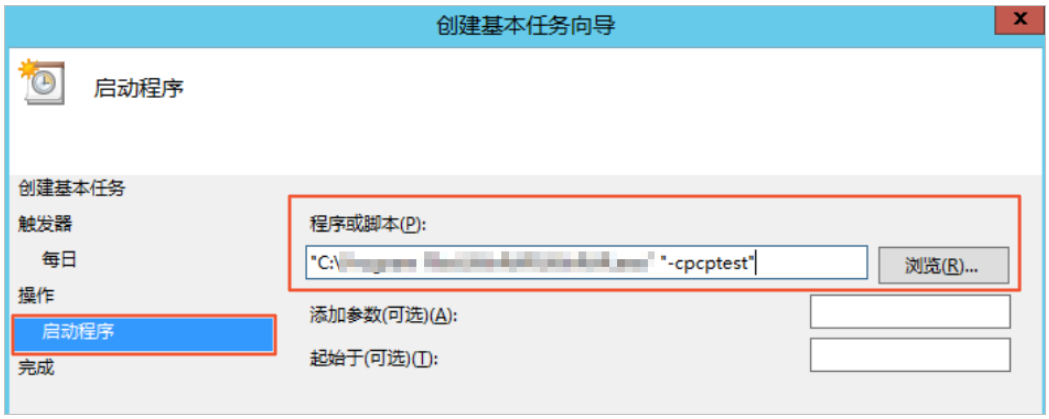


6. 然后在**压缩文件名和参数**对话框，单击**确定**。
桌面会生成一个此压缩包的快捷键。
7. 在桌面按下键盘**Win+R**键打开**运行**窗口，执行命令 `control` 打开**控制面板**。在**控制面板**页面单击**系统和安全**，单击**计划任务**，然后在**任务计划程序**对话框中，单击**创建基本任务**。

8. 在弹出的对话框中为新任务命名，单击**下一步**。
9. 选择触发周期，单击**下一步**。然后选择**启动程序**，单击**下一步**。
10. 此时会弹出对话框需要您输入**程序或脚本**。先找到刚才生成的压缩包快捷键，右键单击该快捷键，选择**属性**，复制**目标**内容。



11. 然后将复制内容粘贴到**启动程序**操作中的**程序或脚本**文本框中，单击**确定**完成创建。



设置好备份策略以后，可以定期的去清理过期的备份文件，避免占用过大的空间。

定期清理不必要的应用程序

定期清理不必要的应用程序，您可以通过**控制面板**中的**程序和功能**界面清理不再使用的程序软件。

设置磁盘监控

阿里云的ECS服务器默认安装了监控插件，您可以在云监控控制台中创建磁盘的报警规则。这样可以实时了解磁盘空间使用率是否到达一个高位值，以便及时清理。详情请参见报警规则。

9.2. Linux实例中数据恢复

在处理磁盘相关问题时，您可能会碰到操作系统中数据盘分区丢失的情况。本文介绍了Linux系统下常见的数据盘分区丢失的问题以及对应的处理方法，同时提供了使用云盘的常见误区以及最佳实践，避免可能的数据丢失风险。

前提条件

- 在修复数据前，您必须先对分区丢失的数据盘创建快照，在快照创建完成后再尝试修复。如果在修复过程中出现问题，您可以通过快照回滚将数据盘还原到修复之前的状态。具体操作，请参见创建一个云盘快照和使用快照回滚云盘。
- 已注册阿里云账号。如还未注册，请先完成账号注册。

背景信息

在Linux实例里，您可以选择以下任一种工具修复磁盘分区并恢复数据：

- fdisk：Linux系统默认安装的分区工具。
- testdisk：主要用于恢复Linux系统的磁盘分区或者数据。Linux系统默认不安装，您需要自行安装这个软件。例如，在CentOS系统里，您可以运行**yum install -y testdisk**在线安装。
- partprobe：Linux系统默认安装的工具。主要用于不重启系统时让kernel重新读取分区。

数据恢复方法

在Linux实例里，您重启系统后，可能会出现数据盘分区丢失或者数据丢失的问题。这可能是因为您未在*etc/fstab*文件里设置自动挂载。此时，您可以先手动挂载数据盘分区。如果手动挂载时报分区表丢失，您可以尝试如下三种办法进行处理：

- 通过fdisk恢复分区
- 通过testdisk恢复分区
- 通过testdisk直接恢复数据

通过fdisk恢复分区

对数据盘分区时，分区磁盘的起止扇区一般使用默认的值，所以可以先尝试直接使用fdisk命令新建分区进行恢复。具体操作，请参见分区格式化数据盘（Linux）。

```
[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
```

如果上述操作无效，您可以使用testdisk工具尝试修复。

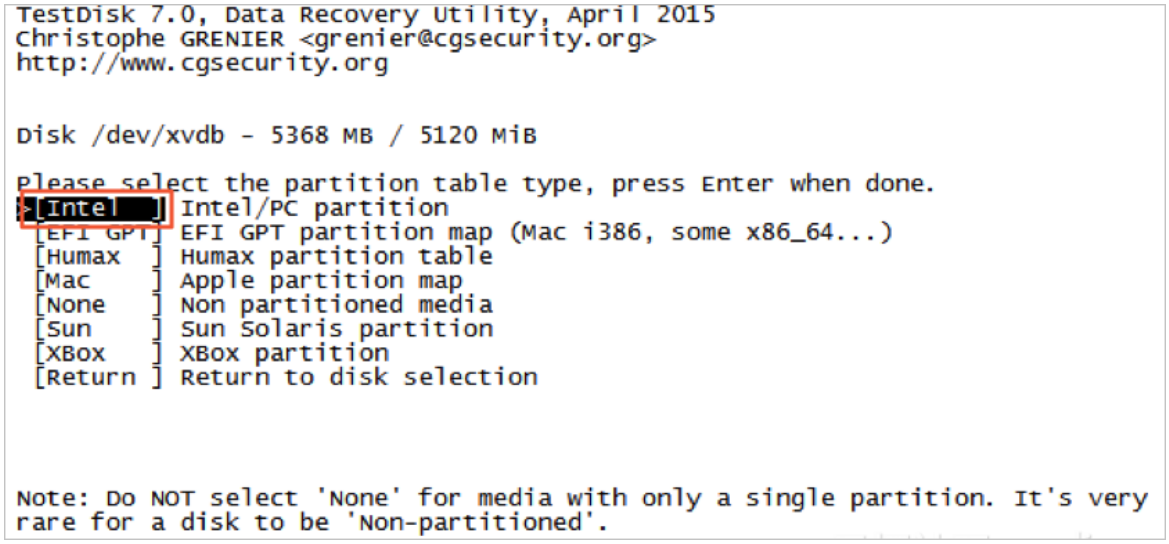
通过testdisk恢复分区

这里假设云盘的设备名为*/dev/xvdb*。按以下步骤使用testdisk工具恢复分区：

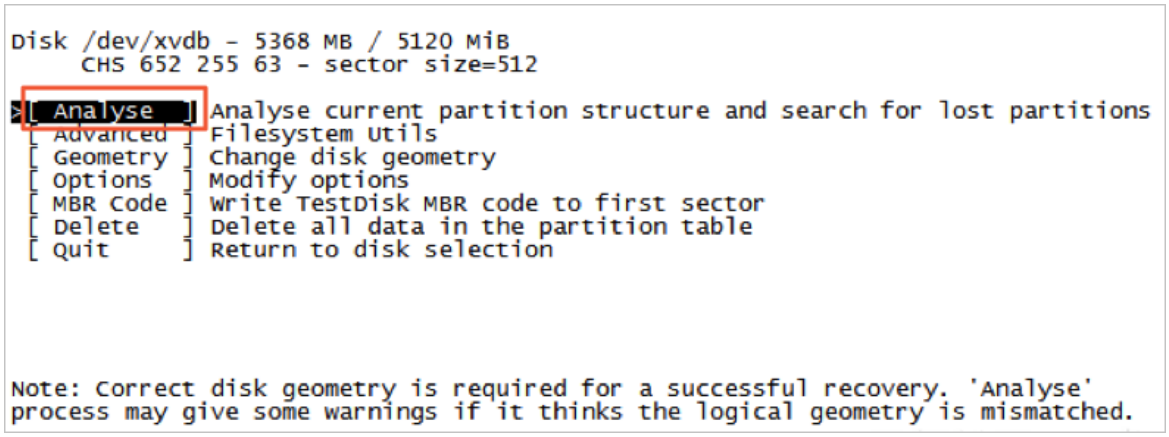
1. 运行**testdisk /dev/xvdb**（根据实际情况替换设备名），再选择**Proceed**（默认值）后按回车键。

```
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.
```

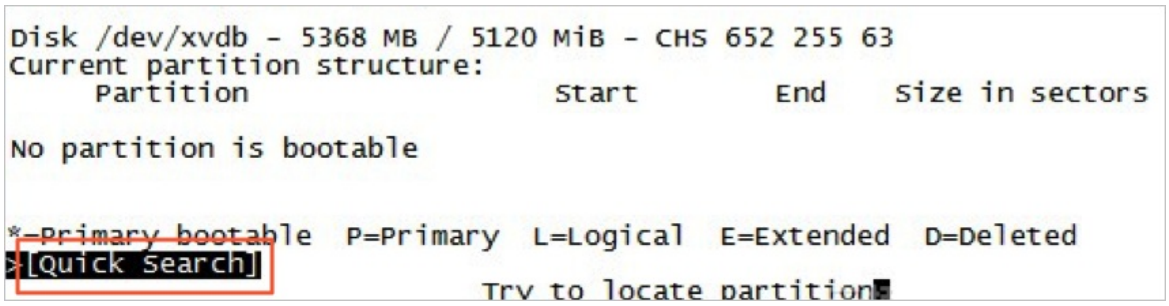
2. 选择分区表类型进行扫描：一般选择*Intel*（默认）。如果您的数据盘采用GPT分区，选择*EFI GPT*。



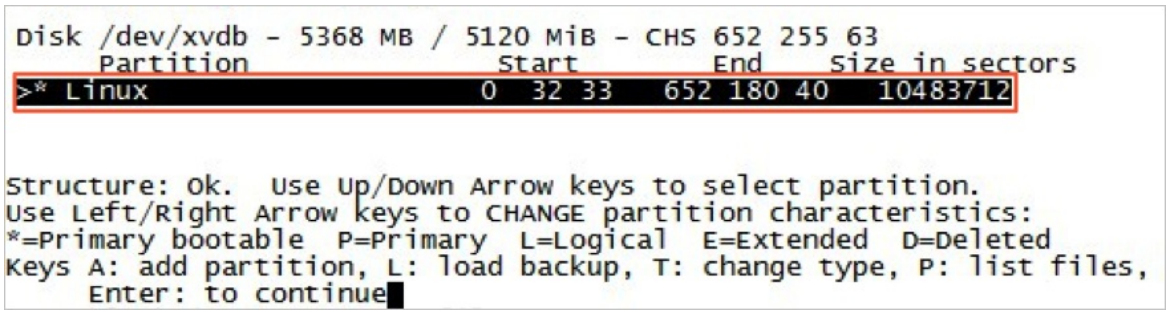
3. 选择*Analyse*后按回车键。



4. 如果您没有看到任何分区信息，选择*Quick Search*后按回车键快速搜索。



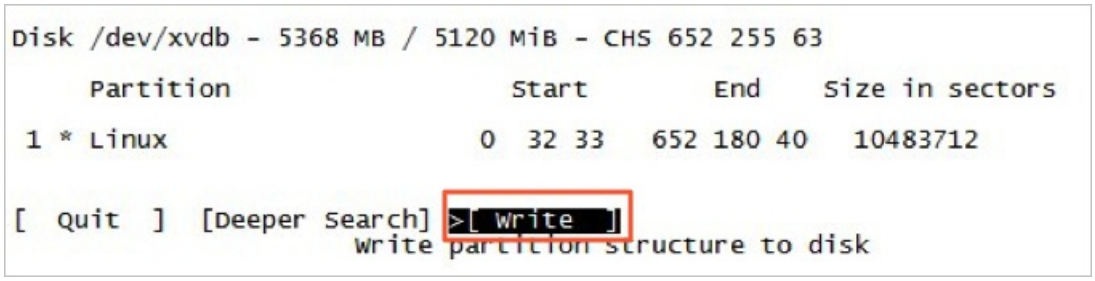
在返回结果中会显示分区信息，如下图所示。



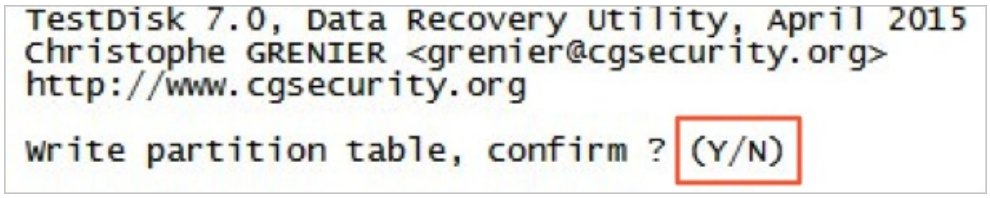
5. 选中分区后，按回车键。

6. 选择*Write*保存分区。

   > 说明 如果不是您需要的分区，可以选择*Deeper Search*继续搜索。



7. 按*Y*键确认保存分区。

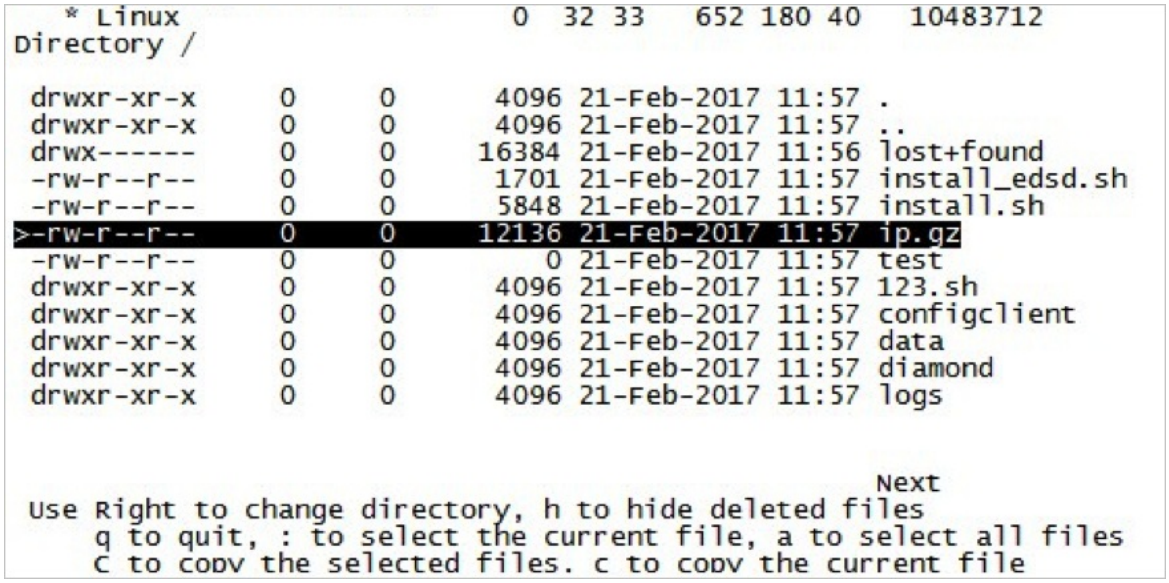


8. 运行**partprobe /dev/xvdb**（根据实际情况替换设备名）手动刷新分区表。
9. 重新挂载分区，查看数据盘里的数据情况。

通过testdisk直接恢复数据

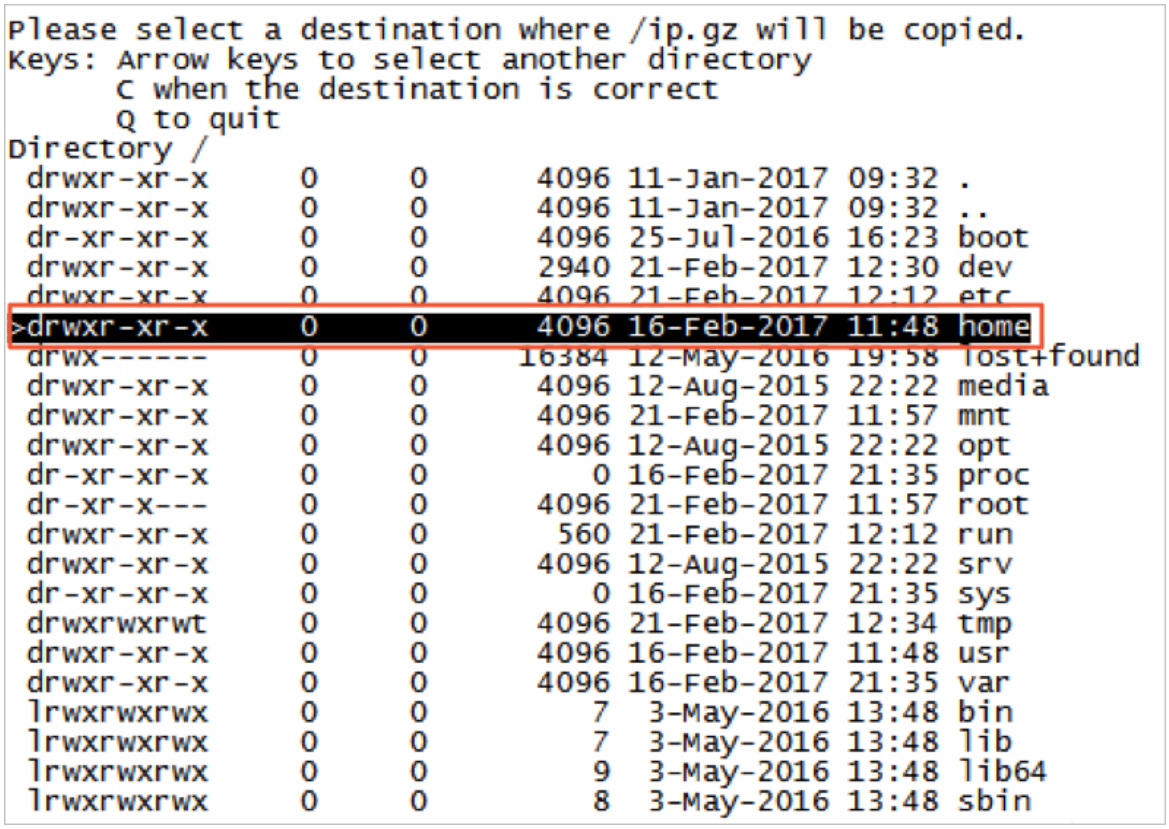
在某些情况下，您可以用testdisk扫描出磁盘分区，但是无法保存分区，此时，您可以尝试直接恢复文件。具体操作步骤如下所示：

1. 用testdisk扫描出磁盘分区。具体操作，请参见通过testdisk恢复分区的第1步到第4步。
2. 按*P*键列出文件。
   返回结果如下图。

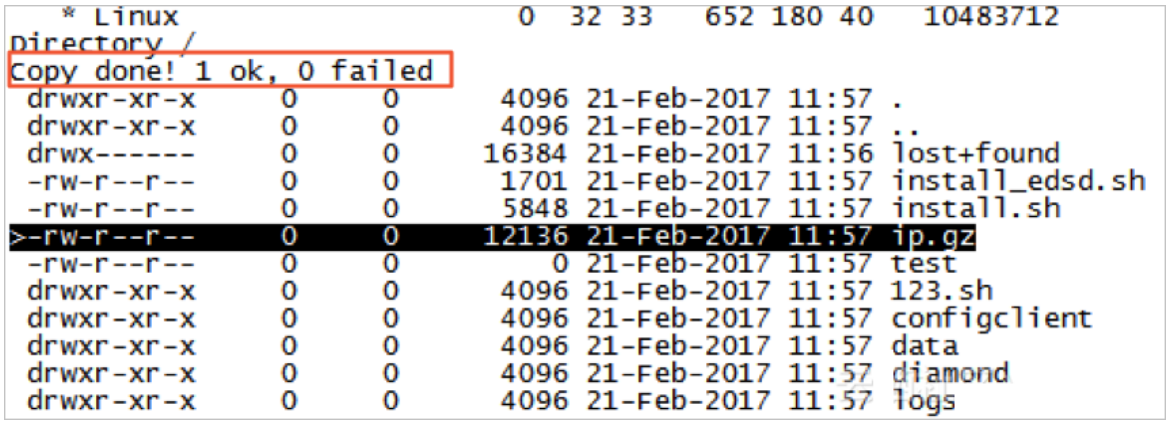


3. 选中要恢复的文件，再按*C*键。

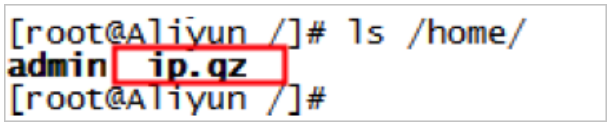
4. 选择目标目录。本示例中以恢复到*/home*为例。



如果您看到 `Copy done! 1 ok, 0 failed` ，表示复制成功，如下图所示。



5. 切换到*/home*目录查看。
如果您能看到文件，说明文件恢复成功。



常见误区与最佳实践

数据是用户的核心资产，很多用户在ECS实例上构建网站、自建数据库（MYSQL/MongoDB/Redis）。数据丢失会给用户的业务带来巨大的风险。本节介绍下数据安全方面的常见误区和最佳实践。

- 常见误区

阿里云的底层存储基于三副本，因此有些用户认为操作系统内数据没有任何丢失风险。实际上这是误解。底层存储的三副本提供对数据磁盘的物理层保护，但是，如果系统内部使用云盘逻辑上出现问题，例如中毒、误删数据、文件系统损坏等情况，还是可能出现数据丢失。此时，您需要通过快照、异地备份等相关技术最大保证数据的安全性。关于三副本的介绍，请参见云盘三副本技术。

- 最佳实践

  数据盘分区恢复以及数据恢复是处理数据丢失问题最后的一道防线，但未必一定能够恢复数据。强烈建议您参考如下最佳实践，通过对数据创建快照（自动或手动）以及各类备份方案，最大程度地保证数据的安全性。

  - 启用自动快照

    根据实际业务，对系统盘、数据盘创建自动快照。注意，在更换系统盘、实例到期后或手动释放磁盘时，自动快照可能会被释放。

    您可以在ECS控制台上通过**修改磁盘属性**选择**自动快照随磁盘释放**。如果想保留自动快照，您可以手动去掉该选项。

    详情请参见快照FAQ和创建自动快照策略。

  - 创建手动快照

    在做下列重要或有风险的操作前，请手动为磁盘创建快照。例如：

    - 系统升级内核
    - 应用升级变更
    - 磁盘数据恢复

    在恢复磁盘时，一定要先对磁盘创建快照，快照完成后做相应的操作。

  - OSS、线下、异地备份

    您可酌情使用OSS、线下、异地等方式备份重要数据。

9.3. Windows实例中数据恢复

在处理磁盘相关问题时，您可能会碰到操作系统中数据盘分区丢失的情况。本文介绍了Windows系统下常见的数据盘分区丢失的问题以及对应的处理方法，同时提供了使用云盘的常见误区以及最佳实践，避免可能的数据丢失风险。

前提条件

- 已注册阿里云账号。如还未注册，请先完成账号注册。
- 在修复数据前，您必须先对丢失分区的数据盘创建快照，在快照创建完成后再尝试修复。如果在修复过程中出现问题，您可以通过快照回滚将数据盘还原到修复之前的状态。具体操作，请参见创建一个云盘快照和使用快照回滚云盘。

背景信息

在Windows实例里，您可以选择以下任一种工具恢复数据盘数据：

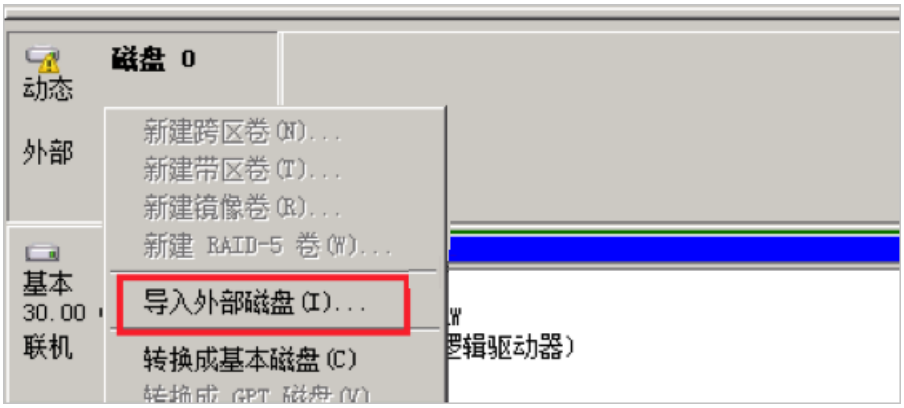
- 磁盘管理：Windows系统自带工具，主要用于分区格式化数据盘等。
- 数据恢复软件：一般是商业软件，您可以去相应的官网下载使用。主要作用是文件系统异常恢复数据。

磁盘显示为外部，无法显示分区

在Windows系统中，您在**磁盘管理器**中看到磁盘显示为**外部**，而且不显示分区情况，如下图所示。

此时，按以下方式处理：

在**外部**磁盘处，右键单击右边的空白处，选择**导入外部磁盘**，再单击**确定**。

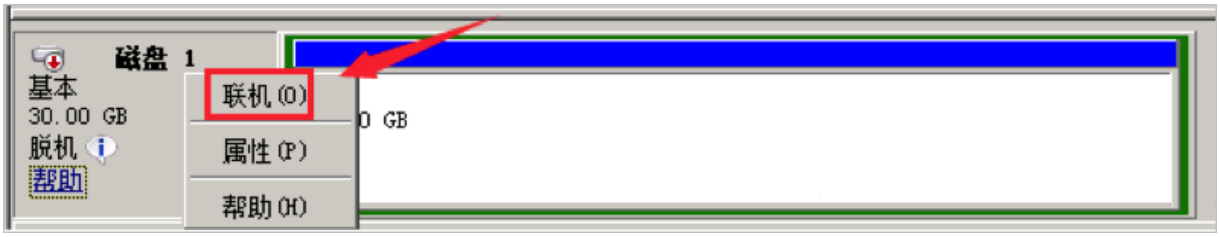


磁盘显示为脱机，无法显示分区

在Windows系统中，您在**磁盘管理器**中看到磁盘显示为**脱机**，而且不显示分区情况，如下图所示。

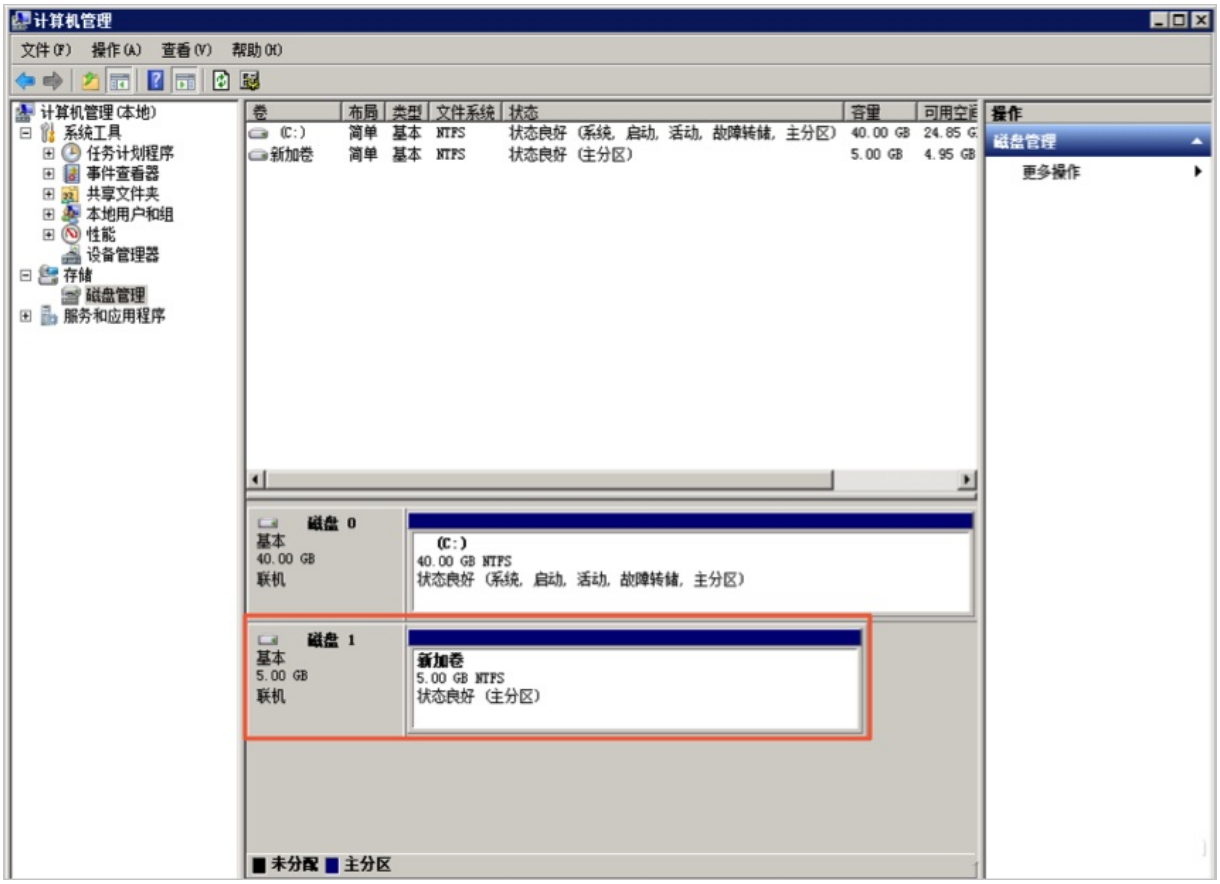
此时，按以下方式处理：

在**脱机**磁盘处，右键单击磁盘名称（如上图中的**磁盘1**）周边的空白区，在弹出的菜单中，选择**联机**，再单击**确定**。



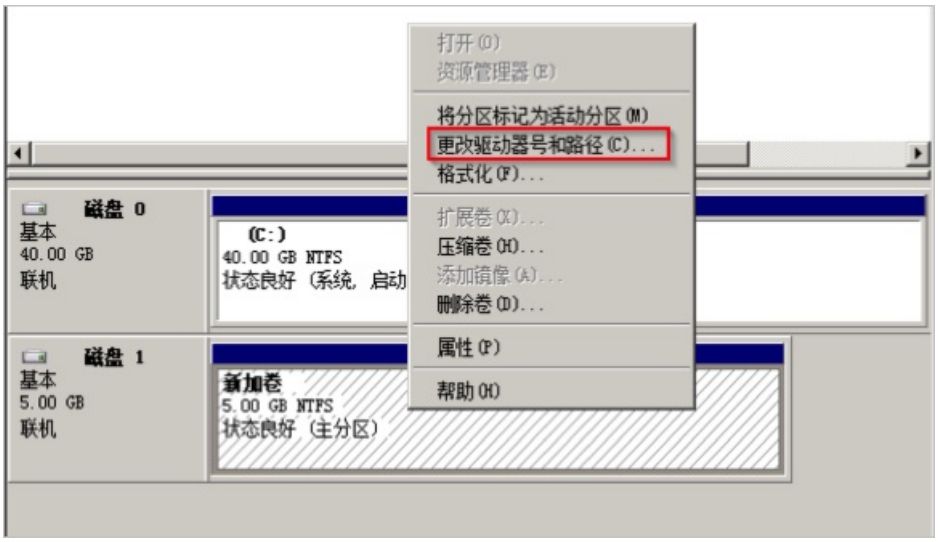
未分配盘符，无法显示分区

在Windows系统中，您在**磁盘管理器**中能看到数据盘的信息，但数据盘未分配盘符，如下图所示。



此时，按以下方式处理：

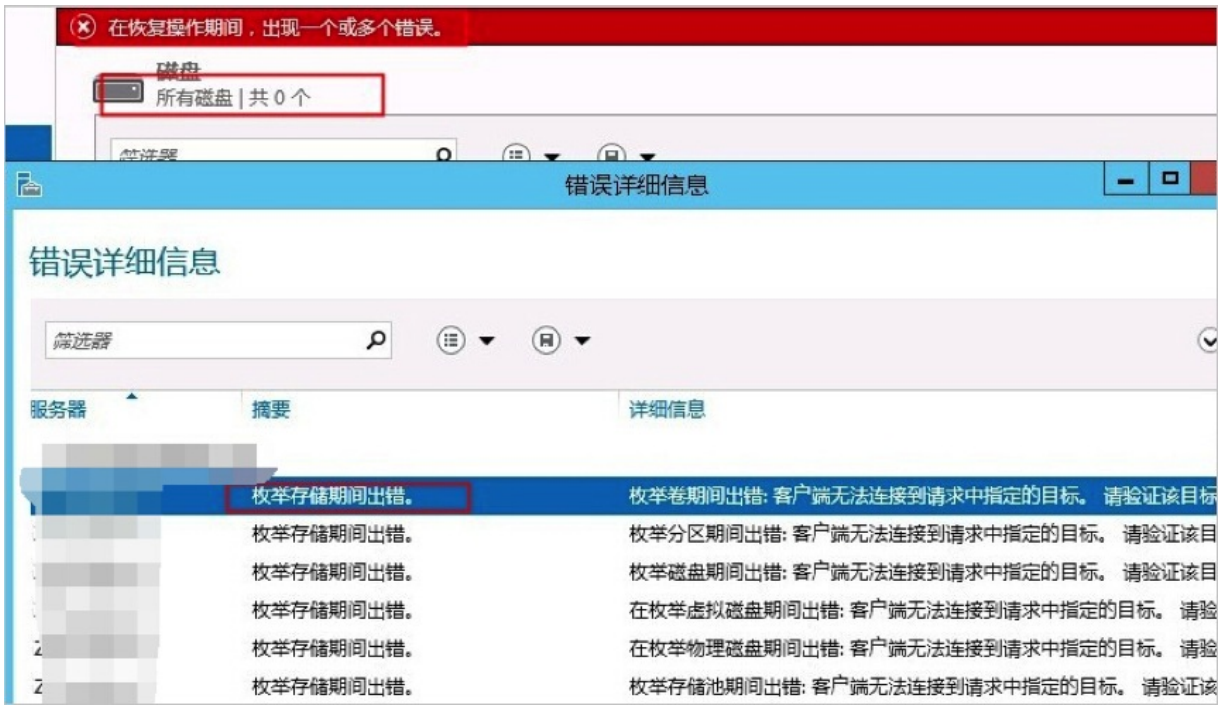
右键单击磁盘（如上图所示的**磁盘1**）的主分区，在弹出菜单中，选择**更改驱动器号和路径**，并按提示完成操作。



在磁盘管理器无法查看数据盘，报错枚举存储期间出错

在Windows系统中，您在**磁盘管理器**里无法查看数据盘。系统日志里报错**枚举存储期间出错**，如下图所示。

> **说明** 操作系统的版本不同，报错内容也可能是**枚举卷期间出错**。



此时，按以下步骤处理：

1. 启动Windows PowerShell。
2. 运行命令**winrm quickconfig**进行修复。

   当界面上询问**执行这些更改吗[y/n]?**时，输入*y*确认执行。

修复完成后，再打开**磁盘管理器**，一般数据盘已经能正常显示。

数据盘变成RAW格式

在某些特殊情况下，您可能会发现Windows下磁盘变为RAW格式。

磁盘显示为RAW格式是因为Windows无法识别磁盘上的文件系统。一般是因为记录文件系统类型或者位置的信息丢失或者损坏，例如partition table或者boot sector。以下列出了一些比较常见的原因：

- 外接硬盘发生这种问题通常是因为没有使用**Safely remove hardware**选项断开磁盘。
- 意外断电导致的磁盘问题。

- 硬件层故障也可能导致磁盘分区信息丢失。
- 底层与磁盘相关的驱动或应用，例如您使用的diskprobe工具就可以直接修改磁盘的表结构。
- 计算机病毒。

如何修复磁盘，请参见微软官方文档Dskprobe Overview。

此外，Windows下有大量免费或商业的数据恢复软件可用于找回丢失的数据。例如，您可以尝试使用Disk Genius工具扫描，来尝试恢复相应的文件。

常见误区和最佳实践

数据是用户的核心资产，很多用户在ECS上构建网站、自建数据库（MYSQL/MongoDB/Redis）。如果出现数据丢失，会给用户的业务带来巨大的风险。如下是在数据安全方面的常见误区和最佳实践。

- 常见误区

  阿里云的底层存储基于三副本，因此有些用户认为操作系统内数据没有任何丢失风险。实际上这是误解。底层存储的三副本提供对数据磁盘的物理层保护，但是，如果系统内部使用云盘逻辑上出现问题，例如中毒、误删数据、文件系统损坏等情况，还是可能出现数据丢失。此时，您需要通过快照、异地备份等相关技术最大保证数据的安全性。

- 最佳实践

  数据盘分区恢复以及数据恢复是处理数据丢失问题最后的一道防线，但未必一定能够恢复数据。强烈建议您参考如下最佳实践，通过对数据创建快照（自动或手动）以及各类备份方案，最大程度地保证数据的安全性。

  - 启用自动快照

    根据实际业务，对系统盘、数据盘创建自动快照。注意，在更换系统盘、实例到期后或手动释放磁盘时，自动快照可能会被释放。

    您可以在ECS控制台上通过**修改磁盘属性**选择**自动快照随磁盘释放**。如果想保留自动快照，您可以手动去掉该选项。

    详情请参见快照FAQ和设置自动快照随云盘释放。

  - 创建手动快照

    在做下列重要或有风险的操作前，请手动为磁盘创建快照。例如：

    - 系统升级内核
    - 应用升级变更
    - 磁盘数据恢复

    在恢复磁盘时，一定要先对磁盘创建快照，快照完成后再做相应的操作。

  - OSS、线下、异地备份

    您可酌情使用OSS、线下、异地等方式备份重要数据。

10.实例配置

10.1. ECS实例数据传输的实现方式

本文通过介绍文件传输的基本原理，以及类Unix/Linux平台上常用的文件传输方式，可以让您根据不同的需要选择合适的文件传输方式。

文件传输原理

文件传输是信息传输的一种形式，它是在数据源和数据宿之间传送文件数据的过程，也称文件数据通信。操作系统把文件数据提取到内存中做暂存，再复制到目的地，加密就是在文件外加了一个壳，文件本身还是一个整体，复制只是把这个整体转移到其它地方，不需要解密，只有打开压缩包时才需解密。一个大文件作为一个数据整体，是不可能瞬间从一台主机转移到其它的主机，传输是一个持续的过程，但不是把文件分割了，因此，如果在传输的过程中意外中断，目标路径中是不会有传输的文件。另外，如果传输的是多个文件，那么这些文件是按顺序分别传输，如果中间中断，则正在传输的文件会传输失败，但是，之前已经传完的文件传输成功（如果传输的是文件压缩包，那么不管里面有几个文件，它本身被视为一个文件）。

通常我们看到的诸如NETCAT、SCP、Rsync等都是可以用来传输文件数据的工具，下面我们将详细介绍主要文件传输工具的特点以及用法。

NETCAT

NETCAT在网络工具中有“瑞士军刀”的美誉，它功能强大，作为网络工具的同时，它传输文件的能力也不容小觑。它可以建立TCP连接、发送UDP数据包、对TCP和UDP端口进行扫描、处理IPv4和IPv6数据包。

参数说明

NETCAT常用参数说明如下表所示：

参数	说明
-C	一直不断连接
-d	后台执行
-g <网关>	设置路由器跃程通信网关，最多可设置8个
-G <指向器数目>	设置来源路由指向器，其数值为4的倍数
-i <延迟秒数>	设置时间间隔，以便传送信息及扫描通信端口
-l	使用监听模式，管控传入的资料
-o <输出文件>	指定文件名称，把往来传输的数据以16进制字码倾倒成该文件保存
-p <通信端口>	设置本地主机使用的通信端口
-r	指定本地与远端主机的通信端口
-s <IP地址>	本地源地址
-u	使用UDP传输协议
-v	显示指令执行过程

参数	说明
-w <超时秒数>	设置等待连线的时间
-z	使用0输入/输出模式，只在扫描通信端口时使用
-n	直接使用IP地址，而不通过域名服务器

常用示例

NC是NETCAT的简写，NC的常用示例如下：

- 端口扫描21~24（以IP192.168.2.34为例）。

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

  返回示例：

  ```
  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
  ```

- 从192.168.2.33拷贝文件到192.168.2.34。
  - 在192.168.2.34上：`nc-l 1234 > test.txt`
  - 在192.168.2.33上：`nc192.168.2.34 < test.txt`
- 用NC命令操作memcached。
  - 存储数据：`printf "set key 0 10 6rnresultrn" |nc 192.168.2.34 11211`
  - 获取数据：`printf "get keyrn" |nc 192.168.2.34 11211`
  - 删除数据：`printf "delete keyrn" |nc 192.168.2.34 11211`
  - 查看状态：`printf "statsrn" |nc 192.168.2.34 11211`
  - 模拟top命令查看状态：`watch "echo stats" |nc 192.168.2.34 11211`
  - 清空缓存：

    ```
    printf "flush_allrn" |nc 192.168.2.34 11211        #谨慎操作，清空了缓存就没了
    ```

SCP

SCP（Secure Copy）即安全拷贝，是一种在两个服务器（本地与本地、本地与远程、远程与远程）间安全的进行文件传输的方法，它以SSH协议为基础。SCP命令的用法和RCP命令格式非常类似，一般推荐使用SCP命令，因为它比RCP更安全。

- SCP在需要进行验证时会要求您输入密码或口令。
- SCP命令使用SSH来传输数据，并使用与SSH相同的认证模式，提供同样的安全保障。

  SSH是目前较可靠的、为远程登录会话和其他网络服务提供安全性的协议，利用SSH协议可以有效防止远程管理过程中的信息泄露问题。SCP是基于SSH的应用，所以进行数据传输的机器上必须支持SSH服务。

特点说明

SCP的特点如下：

- SCP类似于RCP，它能够保留一个特定文件系统上的文件属性，能够保留文件属性或者需要递归的拷贝子目录。
- SCP具备更好的文件传输保密性。与此同时，付出的代价就是文件传输时需要输入密码而且涉及到SSH的一些配置问题，这些都影响其使用的方便性，对于有特定需求的用户，是比较合适的传输工具。

参数说明

SCP常用参数说明如下表所示：

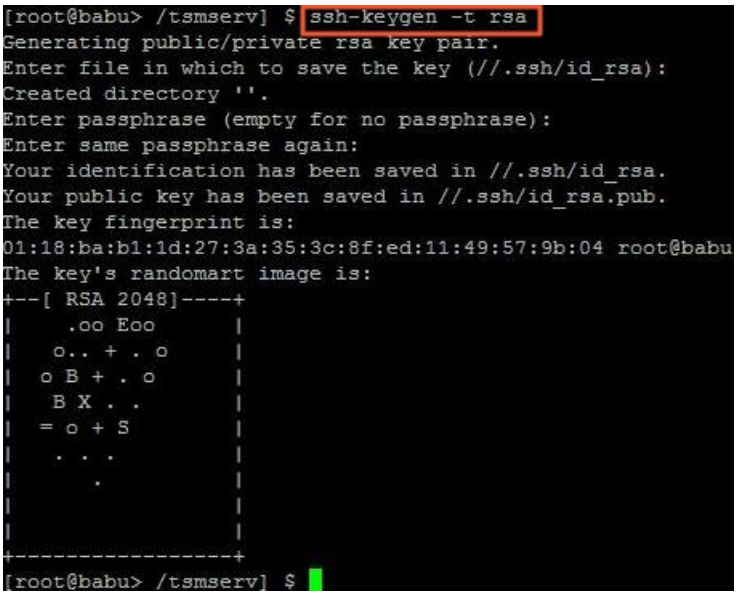
参数	说明
-v	详细方式显示输出，可以用来调试连接、验证或者配置问题
-B	使用批处理模式（传输过程中不询问传输口令或短语）
-C	在复制过程中压缩文件或目录
-P	如果默认SSH端口不是22，则使用此选项指定SSH端口
-r	递归复制整个目录
-4	强制SCP命令只使用IPv4地址
-6	强制SCP命令只使用IPv6地址

常用示例

SCP的常用示例如下：

- 生成RSA类型的密钥

  使用SCP命令，需要输入密码，如果不想每次都输入，可以通过配置SSH，这样在两台机器间拷贝文件时不需要每次都输入用户名和密码。



上述命令生成RSA类型的密钥。在提示密钥的保存路径和密码时，可以直接回车使用默认路径和空密码。这样，生成的公共密钥保存在*/.ssh/id_rsa.pub*，私有密钥保存在 */.ssh/id_rsa*。然后把这个密钥对中的公共密钥的内容复制到要访问的机器上的*/.ssh/authorized_keys*文件中。这样，下次再访问那台机器时，就不用输入密码了。

- 在两台Linux主机间复制文件

  命令基本格式：

  ```
  scp [可选参数] file_source file_target
  ```

  - 从本地复制文件到远程服务器（如下四种方式）

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

    > **说明** 命令说明如下：
    >
    > - 第1、2条命令指定了用户名，命令执行后需要再输入密码；第3、4条命令没有指定用户名，命令执行后需要输入用户名和密码。
    > - 第1、3条命令指定了远程的目录，命令执行后会将本地文件复制到远程指定的目录下。
    > - 第2、4条命令指定了具体的文件名，命令执行后会本地文件复制到远程主机上，且命名为指定的文件名。

  - 从远程服务器复制文件到本地

    从远程复制到本地，只要将从本地复制到远程的命令的后2个参数调换顺序即可。

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

- 在两台Linux主机间复制目录

  命令基本格式：

  ```
  scp -r file_source file_target
  ```

  - 从本地复制目录到远程服务器（如下两种方式）

    ```
    scp -r local_file remote_username@remote_ip:remote_folder
    scp -r local_file remote_ip:remote_folder
    ```

    > **说明** 第1条命令指定了用户名，命令执行后需要再输入密码；第2条命令没有指定用户名，命令执行后需要输入用户名和密码。

  - 从远程服务器复制目录到本地

    从远程复制到本地，只要将从本地复制到远程的命令的后2个参数调换顺序即可。

    ```
    scp -r remote_username@remote_ip:remote_folder local_file
    ```

Rsync

Rsync是Linux/Unix文件同步和传送工具。用于替代RCP的一个工具，Rsync可以通过rsh或ssh使用，也能以daemon模式去运行，在以daemon方式运行时Rsync server会开一个873端口，等待客户端去连接。连接时Rsync server会检查口令是否相符，若通过口令查核，则可以通过进行文件传输，第一次连通完成时，会把整份文件传输一次，以后则就只需进行增量备份。

安装说明

Rsync的安装方式如下：

> **说明** 可以使用每个发行版本自带的安装包管理器安装。

```
sudo apt-get  install  rsync      #在debian、ubuntu 等在线安装方法；
slackpkg  install  rsync          #Slackware 软件包在线安装；
yum install rsync                 #Fedora、Redhat 等系统安装方法；
```

源码编译安装：

```
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
```

参数说明

Rsync常用参数说明如下表所示：

参数	说明
-v	详细模式输出
-a	归档模式，表示以递归的方式传输文件，并保持所有文件属性不变，相当于使用了组合参数-rlptgoD
-r	对子目录以递归模式处理
-l	保留软链接
-p	保持文件权限
-t	保持文件时间信息
-g	保持文件属组信息
-o	保持文件属主信息
-D	保持设备文件信息
-H	保留硬链结
-S	对稀疏文件进行特殊处理以节省DST的空间
-z	对备份的文件在传输时进行压缩处理

工作模式

Rsync有以下六种不同的工作模式：

- 拷贝本地文件，将*/home/coremail*目录下的文件拷贝到*/cmbak*目录下。

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

- 拷贝本地机器的内容到远程机器。

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

- 拷贝远程机器的内容到本地机器。

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

- 拷贝远程Rsync服务器（daemon形式运行Rsync）的文件到本地机。

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

- 拷贝本地机器文件到远程Rsync服务器（daemon形式运行rsync）中。当DST路径信息包含 `::` 分隔符时启动该模式。

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

- 显示远程机器的文件列表。这类似于Rsync传输，不过只要在命令中省略掉本地机器信息即可。

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

配置文件说明

Rsync配置文件说明如下：

```
cat/etc/rsyncd.conf                #内容如下
port = 873                         #端口号
uid = nobody                       #指定当模块传输文件的守护进程UID
gid = nobody                       #指定当模块传输文件的守护进程GID
use chroot = no                    #使用chroot到文件系统中的目录中
max connections = 10               #最大并发连接数
strict modes = yes                 #指定是否检查口令文件的权限
pid file = /usr/local/rsyncd/rsyncd.pid        #指定PID文件
lock file = /usr/local/rsyncd/rsyncd.lock      #指定支持max connection的锁文件，默认为/var/run/
rsyncd.lock
motd file = /usr/local/rsyncd/rsyncd.motd      #定义服务器信息的，自己写 rsyncd.motd 文件内容
log file = /usr/local/rsyncd/rsync.log         #rsync 服务器的日志
log format = %t %a %m %f %b
syslog facility = local3
timeout = 300
[conf]                                         #自定义模块
path = /usr/local/nginx/conf                   #用来指定要备份的目录
comment = Nginx conf
ignore errors                                  #可以忽略一些IO错误
read only = no                                 #设置no，客户端可以上传文件，yes是只读
write only = no                                #no为客户端可以下载，yes不能下载
hosts allow = 192.168.2.0/24                   #可以连接的IP
hosts deny = *                                 #禁止连接的IP
list = false                                   #客户请求时，使用模块列表
uid = root
gid = root
auth users = backup                            #连接用户名，和linux系统用户名无关系
secrets file = /etc/rsyncd.pass                #验证密码文件
```

10.2. 通过读写分离提升数据吞吐性能

一般情况下，对数据库的读和写都在同一个数据库服务器中操作时，业务系统性能会降低。为了提升业务系统性能，优化用户体验，可以通过读写分离来减轻主数据库的负载。本教程主要介绍如何使用中间件MySQL-proxy实现读写分离。

前提条件

使用本教程进行操作前，请确保您已经注册了阿里云账号。如还未注册，请先完成账号注册。

背景信息

应用层中直接使用代码实现，在进入Service之前，使用AOP来做出判断，是使用写库还是读库，判断依据可以根据方法名判断，例如以query、find、get等开头的就走读库，其他的走写库。

优点：

- 多数据源切换方便，由程序自动完成。
- 不需要引入中间件。
- 理论上支持任何数据库。

缺点：

- 由程序员完成，运维参与不到。

- 不能做到动态增加数据源。

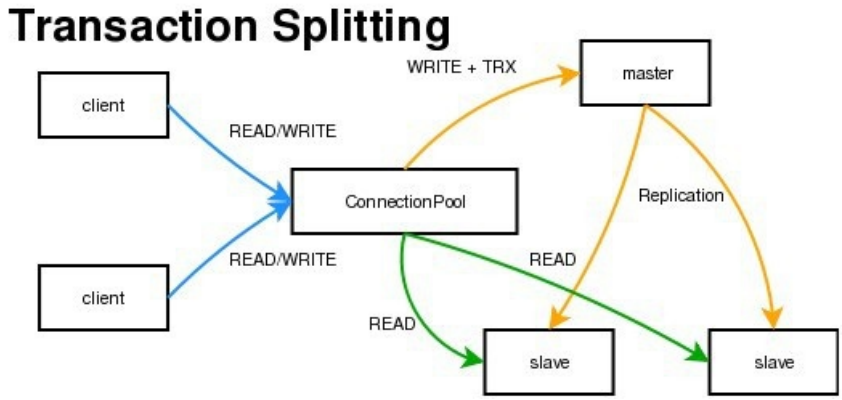
系统层的实现方法包括以下两种：

- 使用分布式关系型数据库DRDS实现读写分离。
- 使用中间件MySQL-proxy实现读写分离。

本教程介绍如何使用中间件MySQL-proxy实现读写分离。

MySQL Proxy是一个处于Client端和MySQL server端之间的简单程序，它可以监测、分析或改变它们的通信。它使用灵活，没有限制。常见的用途包括：负载平衡，故障查询分析，查询过滤和修改等等。

MySQL-proxy的原理如下图：



MySQL Proxy是一个中间层代理，简单的说，MySQL Proxy就是一个连接池，负责将前台应用的连接请求转发给后台的数据库，并且通过使用lua脚本，可以实现复杂的连接控制和过滤，从而实现读写分离和负载平衡。对于应用来说，MySQL Proxy是完全透明的，应用则只需要连接到MySQL Proxy的监听端口即可。当然，这样proxy机器可能成为单点失效，但完全可以使用多个proxy机器做为冗余，在应用服务器的连接池配置中配置到多个proxy的连接参数即可。

优点：

- 源程序不需要做任何改动就可以实现读写分离。
- 动态添加数据源不需要重启程序。

缺点：

- 源程序依赖于中间件，会导致切换数据库变得困难。
- 由中间件做了中转代理，性能有所下降。

操作步骤

使用中间件MySQL-proxy实现读写分离的操作步骤如下：

1. 步骤一：完成准备工作
2. 步骤二：配置读写分离
3. 步骤三：授权
4. 步骤四：验证读写分离

步骤一：完成准备工作

环境说明如下：

- 主库IP：121.40.xx.xx
- 从库IP：101.37.xx.xx

- MySQL-proxy代理IP：116.62.xx.xx

完成以下操作，做好准备工作：

1. 新建3台ECS，并安装MySQL。
2. 搭建主从环境，必须保证主从数据库数据一致。
3. 修改主从环境的MySQL配置文件。
   - 主环境：

```
vim /etc/my.cnf
[mysqld]
server-id=202                  #设置服务器唯一的id，默认是1
log-bin=mysql-bin              # 启用二进制日志
```

   - 从环境：

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

4. 重启主从服务器中的MySQL服务。

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

5. 在主服务器上建立用户并授权slave。

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

6. 查看主数据库状态。

```
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)
```

7. 配置从数据库。

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

8. 启动slave同步进程并查看状态。

```
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:
```

9. 验证主从同步。

   i. 在主数据库的表*testproxy.test1*中写入数据。

```
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)
```

ii. 在从数据库中运行以下命令，查找表*testproxy.test1*的数据。

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

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

如果表*testproxy.test1*的内容与主数据库的一致，则主从同步成功。

步骤二：配置读写分离

完成以下操作，配置读写分离：

1. 安装MySQL-Proxy。

```
wget https://cdn.mysql.com/archives/mysql-proxy/mysql-proxy-0.8.5-linux-glibc2.3-x86-64
bit.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. 设置环境变量。

```
vim /etc/profile                        #加入以下内容
PATH=$PATH:/alidata/mysql-proxy-0.8.5/bin
export $PATH
source /etc/profile                     #使变量立即生效
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. 设置读写分离。

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

MySQL Proxy会检测客户端连接，当连接没有超过min_idle_connections预设值时，不会进行读写分离。默认最小4个（最大8个）以上的客户端连接才会实现读写分离。现改为最小1个（最大2个），便于读写分离的测试。生产环境中，可以根据实际情况进行调整。

调整前：

```
-- 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
```

调整后：

```
-- 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. 将lua管理脚本*admin.lua*复制到读写分离脚本*rw-splitting.lua*所在目录。

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

步骤三：授权

完成以下操作，进行授权：

1. 在主库中操作授权。因主从同步的原因，从库也会执行。

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

2. 开启MySQL-Proxy。

```
mysql-proxy --daemon --log-level=debug --log-file=/var/log/mysql-proxy.log --plugins=pr
oxy -b 填写master-IP:3306 -r 填写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. 查看端口和相关进程。

```
netstat -tpln
```

```
[root@ ~]# netstat -tpln
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 --log-level=debug --log-file=/var/log/mysql-proxy.log --plugins=proxy -b :3306 -r :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
root     22794 22602  0 11:02 pts/0    00:00:00 grep --color=auto mysql
```

步骤四：验证读写分离

完成以下操作，验证读写分离：

1. 关闭从复制。

```
stop slave;
```

2. 在MySQL-Proxy上操作，登录MySQL-Proxy后台管理。

```
mysql  -u  admin -padmin -P 4041 -h MySQL-Proxy-IP
select * from backends;                #查看状态
```

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

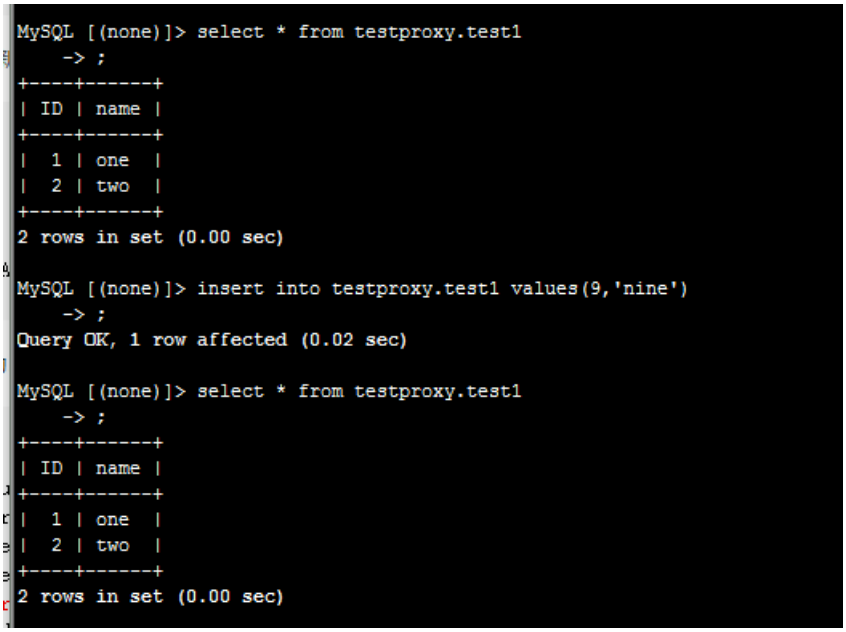
第一次连接，会连接到主库上。

```
mysql -umysql-proxy -p123456 -h 116.62.xx.xx -P 4040
insert into testproxy.test1 values(3,'three');              #新增一条数据，由于测试需要，关闭了从复制，因此该数据在主库中存在，在从库中不存在
```

```
[root@ ~]# mysql -umysql-proxy -p123456 -h  -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)]>
```

多开几个连接进行测试，当查询表*testproxy.test1*的数据显示是从库的数据时，读写分离成功。

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



10.3. ECS实例搭建Windows系统AD域

ECS实例搭建Windows系统AD域

活动目录AD（Active Directory）是微软服务的核心组件。AD能实现高效管理，例如批量管理用户、部署应用和更新补丁等。许多微软组件（例如Exchange）和故障转移群集也需要AD域环境。本文以Windows Server 2012 R2 Datacenter操作系统为例，介绍如何搭建AD域。

前提条件

已创建两台ECS实例，分别作为域控制器（DC）和客户端（Client）。创建的ECS实例需满足以下条件：

- 分区为NTFS分区。
- 实例支持DNS服务。
- 实例支持TCP/IP协议。

背景信息

活动目录AD（Active Directory）是微软服务的核心组件，相关名词概念如下：

- DC：Domain Controllers，域控制器
- DN：Distinguished Name，识别名
- OU：Organizational Unit，组织单位
- CN：Canonical Name，正式名称
- SID：Security Identifier，安全标识符

本文以Windows Server 2012为例介绍如何搭建AD域，搭建过程中采用的环境示例如下：

- 组网信息：网络类型采用专有网络VPC，交换机的私有网段为192.168.100.0/24。
- 域名信息：示例域名为example.com，作为DC的ECS实例IP地址为192.168.100.105，作为客户端的ECS实例IP地址为192.168.100.106。

> **说明** 如果进行了搭建AD域的操作，请保证相关的ECS实例始终使用相同的IP地址，否则IP地址变化会导致访问异常。

步骤一：部署AD域控制器

> **说明** 阿里云不推荐您使用已有的域控制器创建自定义镜像来部署新的域控。如果必须使用，请注意新建实例的主机名（hostname）和创建自定义镜像之前实例的主机名必须保持一致，否则可能会报错 `服务器上的安全数据库没有此工作站信任关系` ；您也可以在创建实例后修改成相同的主机名，解决此问题。

1. 远程连接作为DC的ECS实例。
2. 打开服务器管理器，添加角色和功能。

> **说明** 除额外说明的配置外，部分配置步骤已省略，配置时保持默认配置，单击**下一步**即可。

i. 选择安装类型。

ii. 选择要安装角色和功能的服务器。

iii. 选择要安装在服务器上的角色。

此处以将AD域服务和DNS服务部署在同一台服务器上为例，选择安装AD域服务和DNS服务。

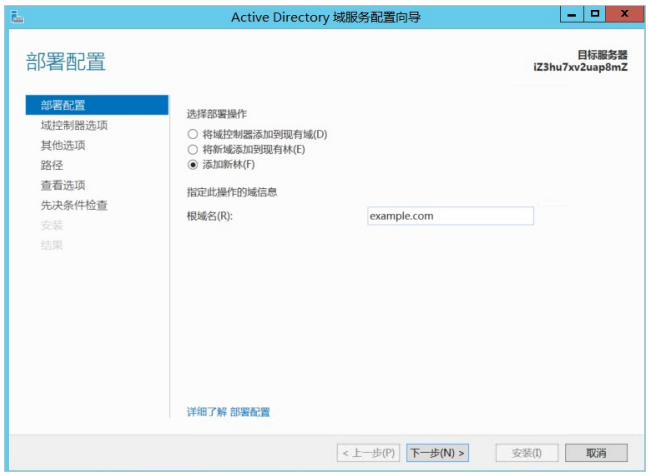
iv. 安装完成后，关闭对话框。

3. 在**服务器管理器**页面的右上角单击三角图标，将此服务器提升为域服务器。

> **说明** 除额外说明的配置外，部分配置步骤已省略，配置时保持默认配置，单击**下一步**即可。

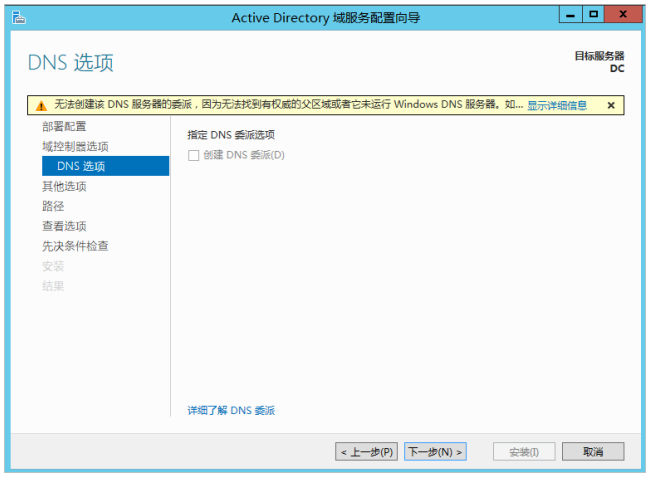
i. 添加新林，设置域名。

AD域的域名示例为example.com。

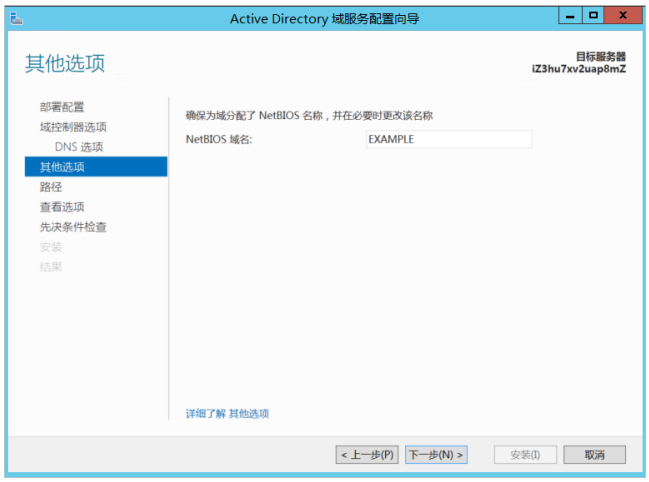


ii. 配置域服务器参数。

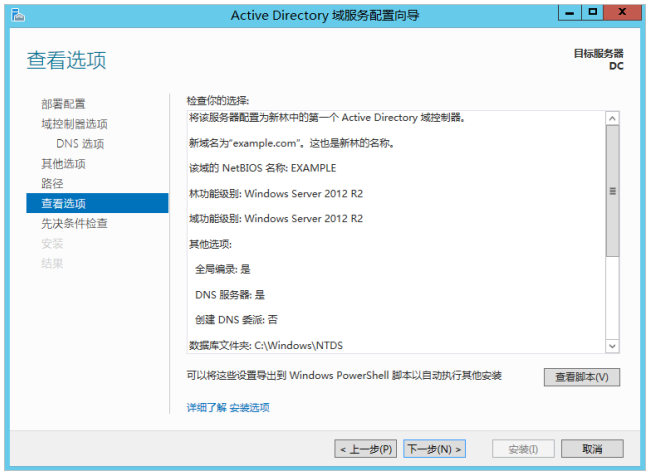
iii. 配置DNS选项。



iv. 配置NetBIOS域名。

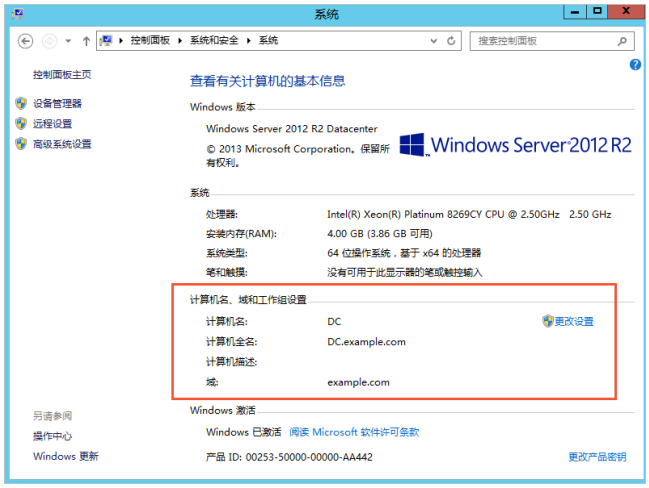


v. 检查并确认您的选择，单击**下一步**。



vi. 单击**安装**，开始安装AD域服务器。

安装完成后将自动重启服务器，重新连接该服务器后可以查看安装结果。



步骤二：修改客户端的SID

如果您使用自定义镜像创建的ECS实例部署域控制器，需按照本文中的步骤修改SID。如果已经修改了SID，可跳过该步骤。

1. 远程连接作为客户端的ECS实例。
2. 下载修改客户端SID的PowerShell脚本。
   - 下载地址：AutoSysprep.ps1
   - 脚本来源：阿里云官方
3. 打开CMD，输入*powershell*切换至**Windows PowerShell**界面。

   > **说明** 如果您的实例操作系统是64位，则不能使用32位的PowerShell（即**Windows PowerShell (x86)**），否则会报错。

4. 切换至脚本存储的路径，执行如下命令，查看脚本工具说明。

   ```
   .\AutoSysprep.ps1 -help
   ```

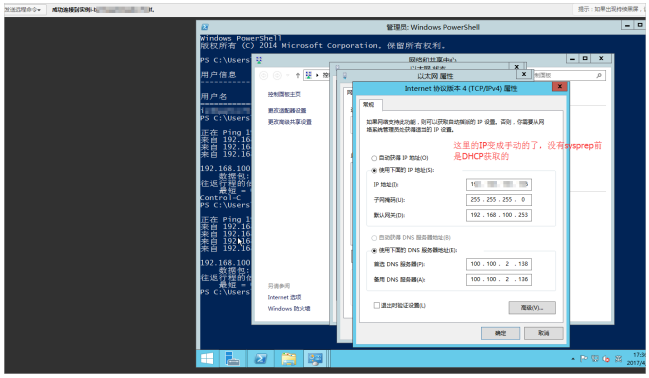
5. 执行如下命令，重新初始化服务器的SID。

```
.\AutoSysprep.ps1 -ReserveHostname -ReserveNetwork -SkipRearm -PostAction "reboot"
```

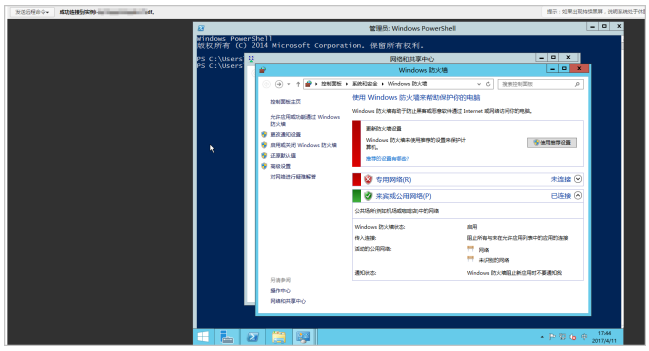
初始化完成后，会重启实例，您需要注意以下事项。

- IP地址的获取方式会从DHCP变成固定IP地址，请确保该固定IP地址和开始设置前ECS实例的IP地址一致。您也可以将获取方式改回DHCP，以自动获取控制台中为ECS实例分配的主私有IP地址。

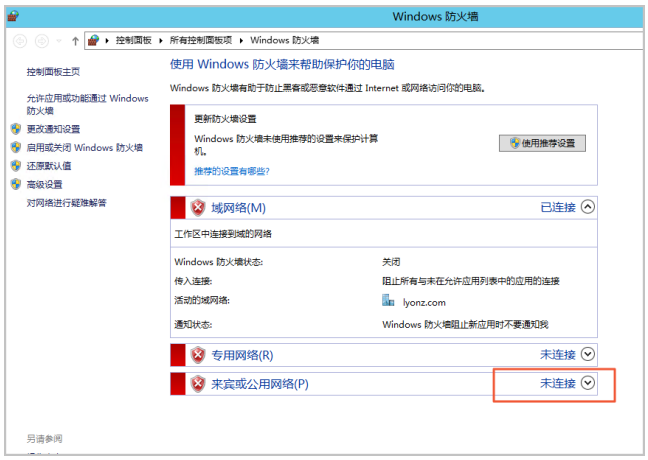
  > **说明** 请不要在控制台修改ECS实例的主私有IP地址，否则IP地址变化会导致访问异常。



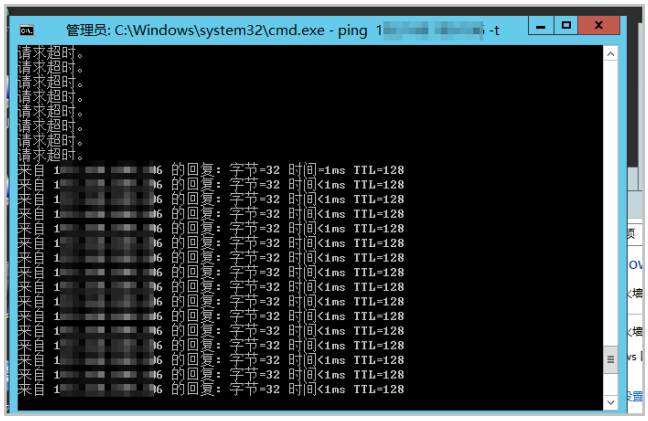
- 初始化SID后，云服务器防火墙的配置被修改成微软的默认配置，导致云服务器无法Ping通。您需要关闭防火墙**来宾或公用网络**，或者放行需要开放的端口。下图表示防火墙**来宾或公用网络**的状态是已连接。



6. 打开控制面板修改防火墙设置，关闭来宾或公用网络防火墙。



关闭后，可以Ping通服务器。

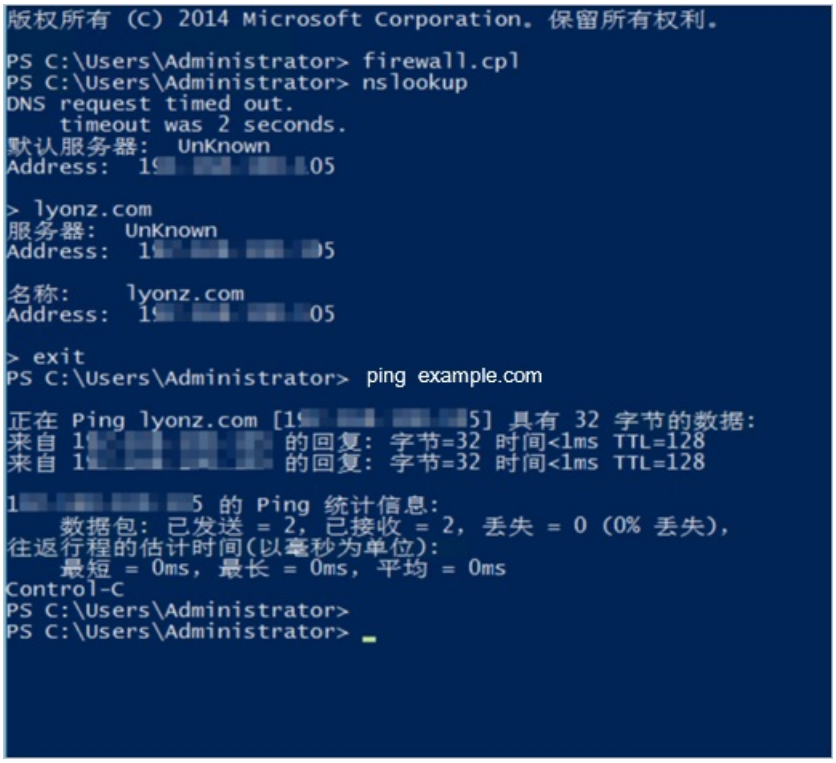


步骤三：将客户端加入AD域

1. 远程连接作为客户端的ECS实例。
2. 修改DNS服务器地址。

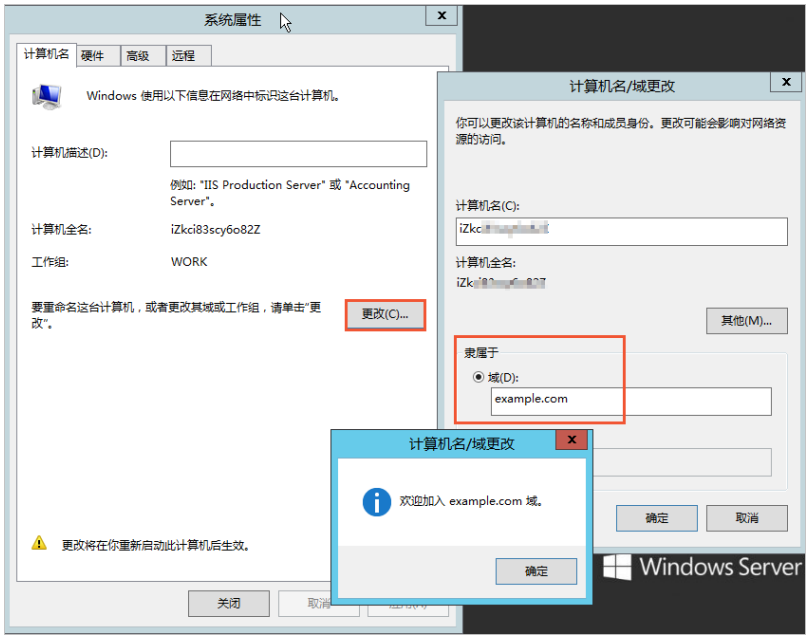
   在步骤一中已经将AD域服务和DNS服务部署在同一台ECS实例上（IP地址为192.168.100.105），此处指定DNS服务器的地址为192.168.100.105。

3. 检查是否能Ping通DNS服务器IP地址。



4. 修改主机名并加入AD域。

   i. 打开控制面板修改系统属性，将该客户端加入到AD域中。



   ii. 重新启动服务器，使修改生效。

> **说明** 对于作为客户端的ECS实例，阿里云不推荐您使用已加入域的客户端实例来创建自定义镜像，否则新镜像创建的实例会报错 `服务器上的安全数据库没有此工作站信任关系` 。如果确实需要，建议您在创建新的自定义镜像前先退出域。

相关文档

- 域控常见问题配置
- 云市场

10.4. 设置Windows操作系统首选语言

本文以公共镜像中的Windows Server 2016英语版操作系统为例，介绍如何从Windows更新下载语言资源包，为一台ECS实例重新设置首选语言。

前提条件

使用本教程进行操作前，请确保您已经注册了阿里云账号。如还未注册，请先完成账号注册。

背景信息

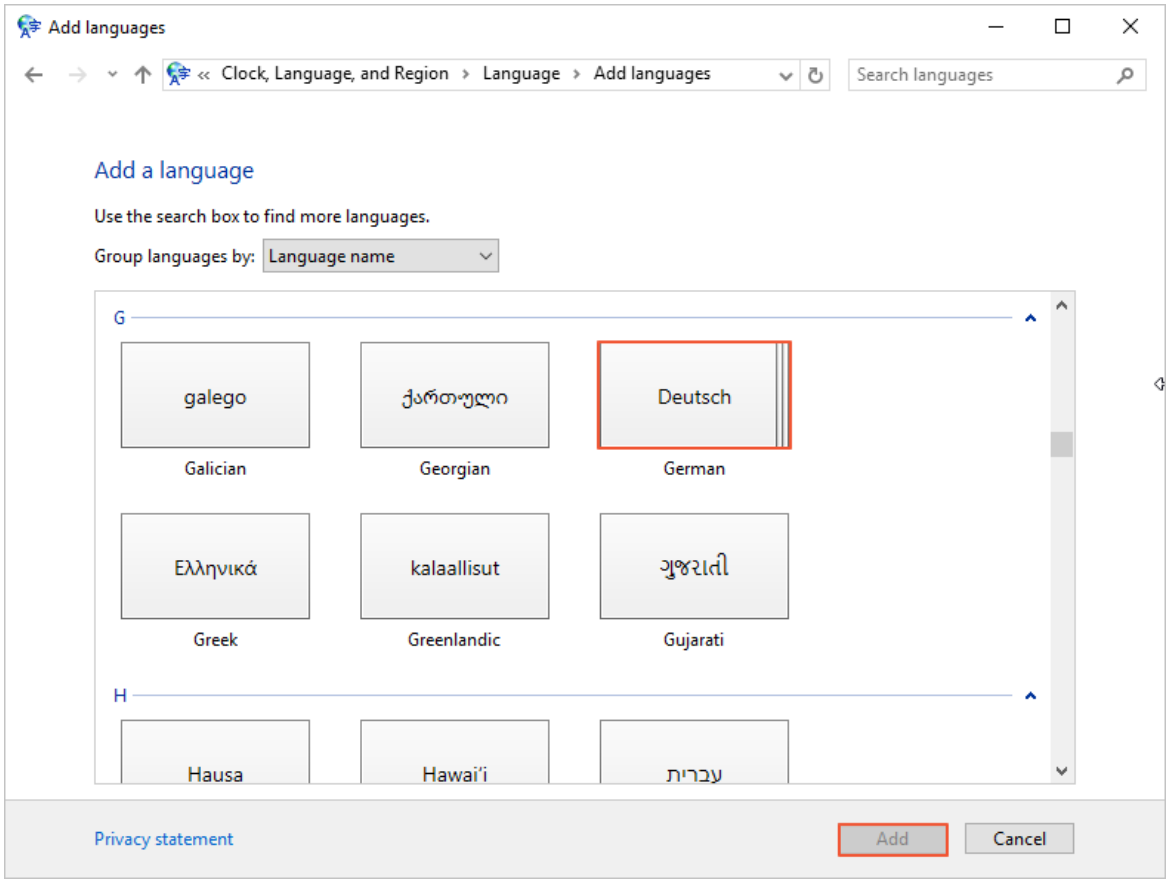
云服务器ECS仅提供中文版和英文版的Windows Server公共镜像。如果您需要使用其他语言版本，如阿拉伯语、德语、俄语或日语等，可以根据本文设置ECS实例的首选语言。本文以德语为示范步骤，适用于Windows Server 2012及其以上版本的操作系统。创建使用德语和德语键盘设置的自定义镜像后，您可以使用该自定义镜像根据自身需求创建任意数量的实例。

操作步骤

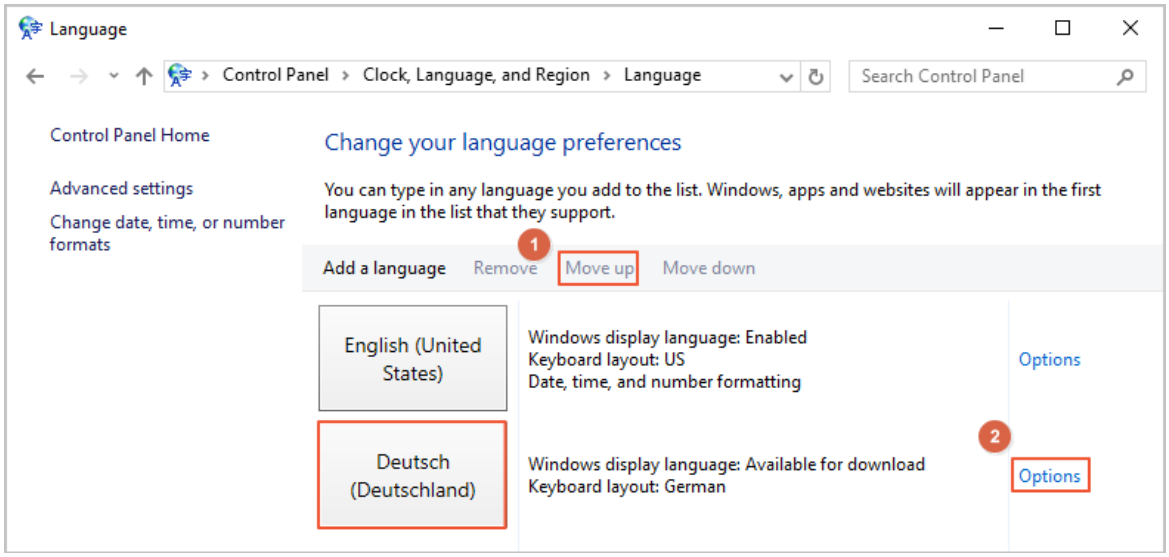
1. 连接Windows实例。连接方式请参见连接方式导航。
2. 打开PowerShell模块。
3. 运行以下命令临时禁用WSUS（Windows Server Update Services）更新源。

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

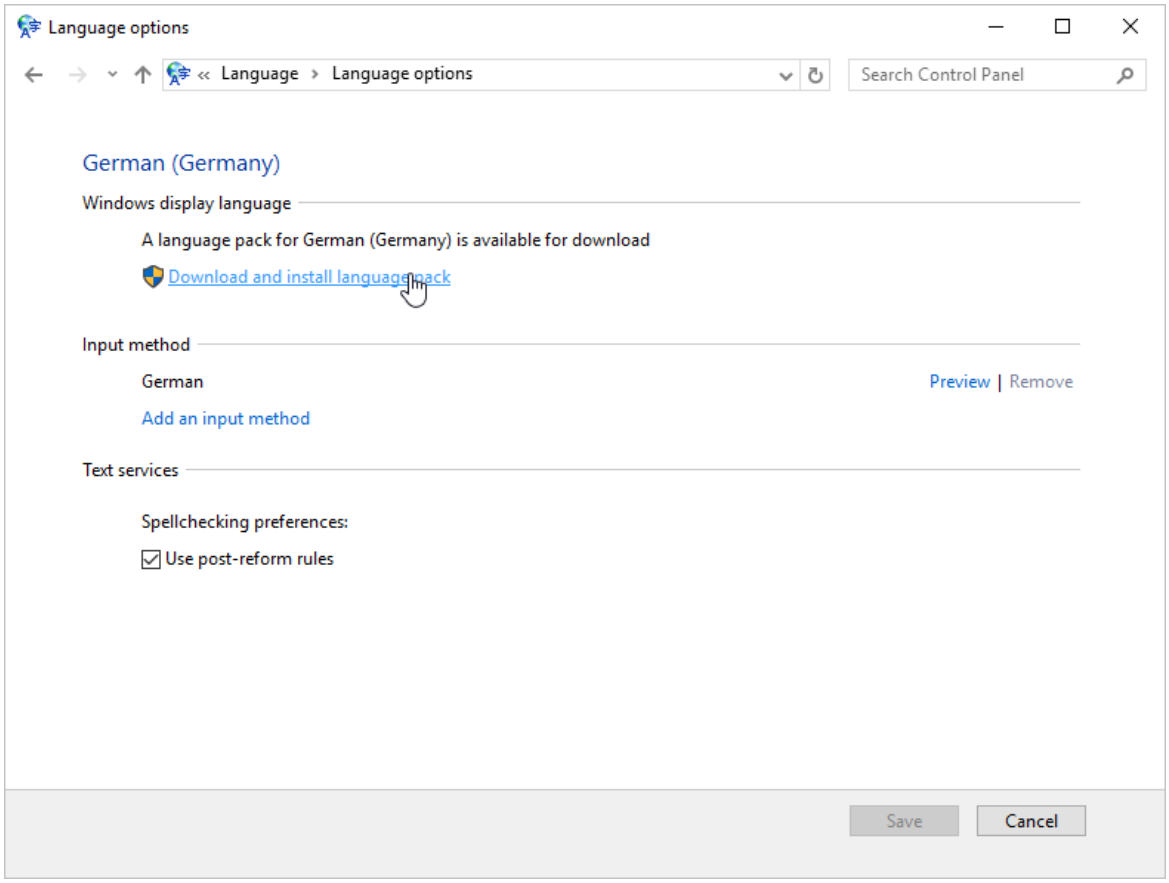
4. 找到控制面板，单击Clock, Language, and Region > Language > Add a language。
5. 在Add languages对话框中，选择一种语言，例如Deutsch (German) > Deutsch (Deutschland)，单击Add。



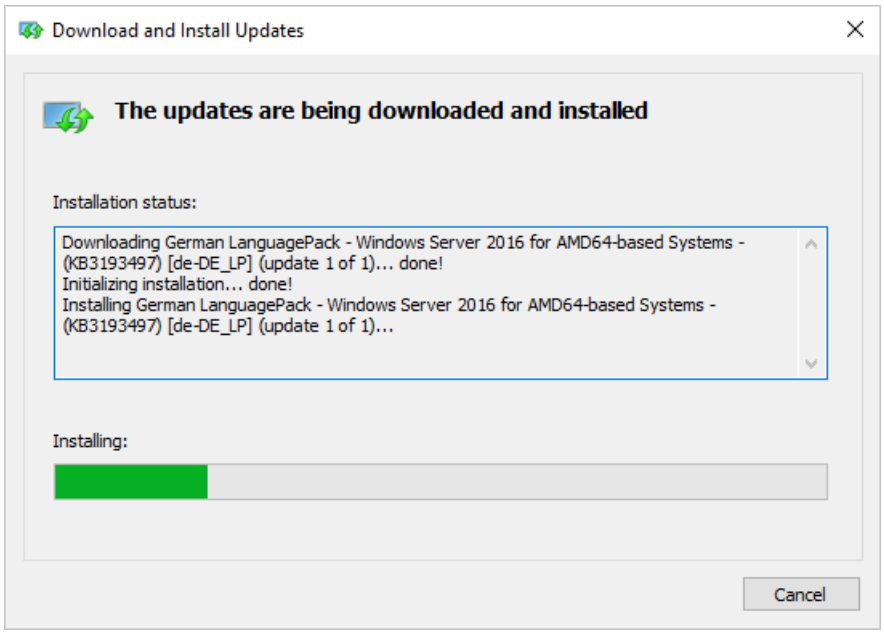
6. 选择语言，例如Deutsch (Deutschland)，单击Move up更改语言优先级。
7. 单击所选语言右侧的Options，在线检查语言更新。



8. 等待实例检查更新，大约三分钟后更新完成，会提示可供下载，单击Download and install language pack。



等待安装完成。



9. 安装完成后，在ECS管理控制台重启实例。具体操作，请参见重启实例。
10. 再次连接Windows实例。
    显示语言更改为德语。
11. 打开PowerShell ISE模块，运行以下命令重新启用WSUS。

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

12. 打开Windows Update，检查安全更新，重新安装配置在设置语言之前已完成的所有安全更新。

后续步骤

您可以使用相同语言设置创建多台实例：

1. 登录ECS管理控制台。
2. 根据该Windows实例创建自定义镜像。具体操作，请参见创建自定义镜像。
3. 通过自定义镜像创建指定数量的实例。具体操作，请参见使用自定义镜像创建实例。

10.5. Linux系统进入单用户模式

本文将分别介绍使用CentOS、Debian、SLES和Ubuntu操作系统镜像的ECS实例如何进入单用户模式。

前提条件

- 已注册阿里云账号。如还未注册，请先完成账号注册。
- 已经创建了一台ECS实例，具体操作步骤请参见创建方式导航。本文中创建ecs.g6.large实例规格的ECS实例。

背景信息

Linux系统的单用户模式是系统启动方式之一，您可以通过Linux系统的系统引导器（GRUB）进入单用户模式。进入单用户模式后，操作者拥有系统管理员权限并能修改全部系统配置信息。该模式常用于以下场景：

- 修改系统密码
- 排查启动故障
- 修复系统异常
- 维护硬盘分区

> **注意** 在单用户模式下，您能修改系统的关键配置，因此建议您在必要场景中设置该模式，并谨慎操作。

您也可以通过卸载系统盘功能来排查启动故障问题，详情请参见卸载或挂载系统盘。

示例导航

- 示例一：CentOS操作系统进入单用户模式
- 示例二：Debian操作系统进入单用户模式
- 示例三：SLES操作系统进入单用户模式

- 示例四：Ubuntu操作系统进入单用户模式

示例一：CentOS操作系统进入单用户模式

本示例中连接CentOS 8.0 64位操作系统的ECS实例。

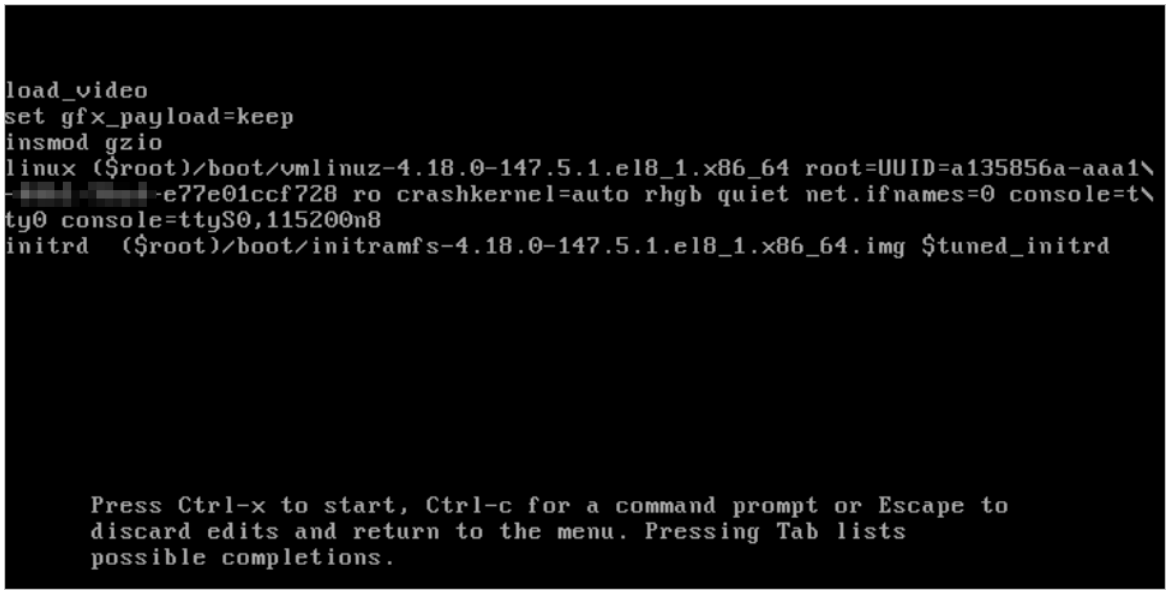
1. 远程连接ECS实例。

   连接方式请参见通过密码认证登录Linux实例。

   > **说明** 使用Workbench和SSH命令远程连接的实例，在通过命令重启时不能直接进入启动系统页面，因此不建议使用这两种连接方式。

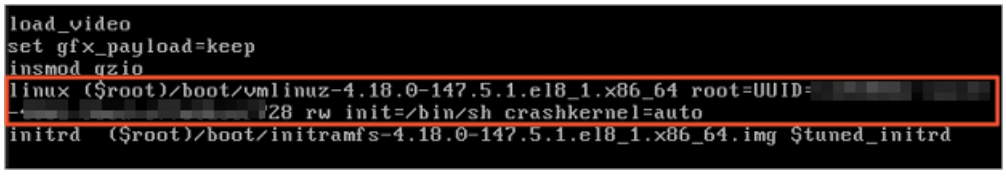
2. 运行 `reboot` 重启ECS实例，并在重启过程中出现选择启动系统界面时按下键盘*e*键，跳转至启动项配置界面。

   跳转界面如下。

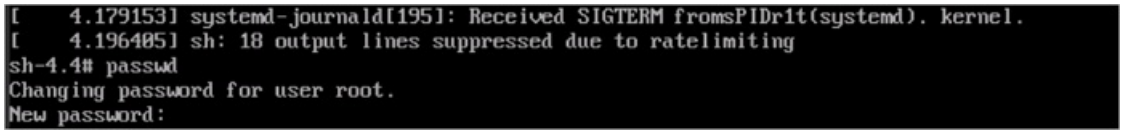
3. 使用键盘的方向键，移动光标至 `linux` 开头的一行，并在本行中将 `ro` 至末尾的内容替换为 `rw init=/bin/sh crashkernel=auto` 。

   修改后的信息如图所示。

4. 按下键盘的*ctrl+x*组合键或按*F10*键。

   系统会直接进入单用户模式。重置系统密码示例如图所示。

示例二：Debian操作系统进入单用户模式

本示例中连接Debian 10.2 64位操作系统的ECS实例。

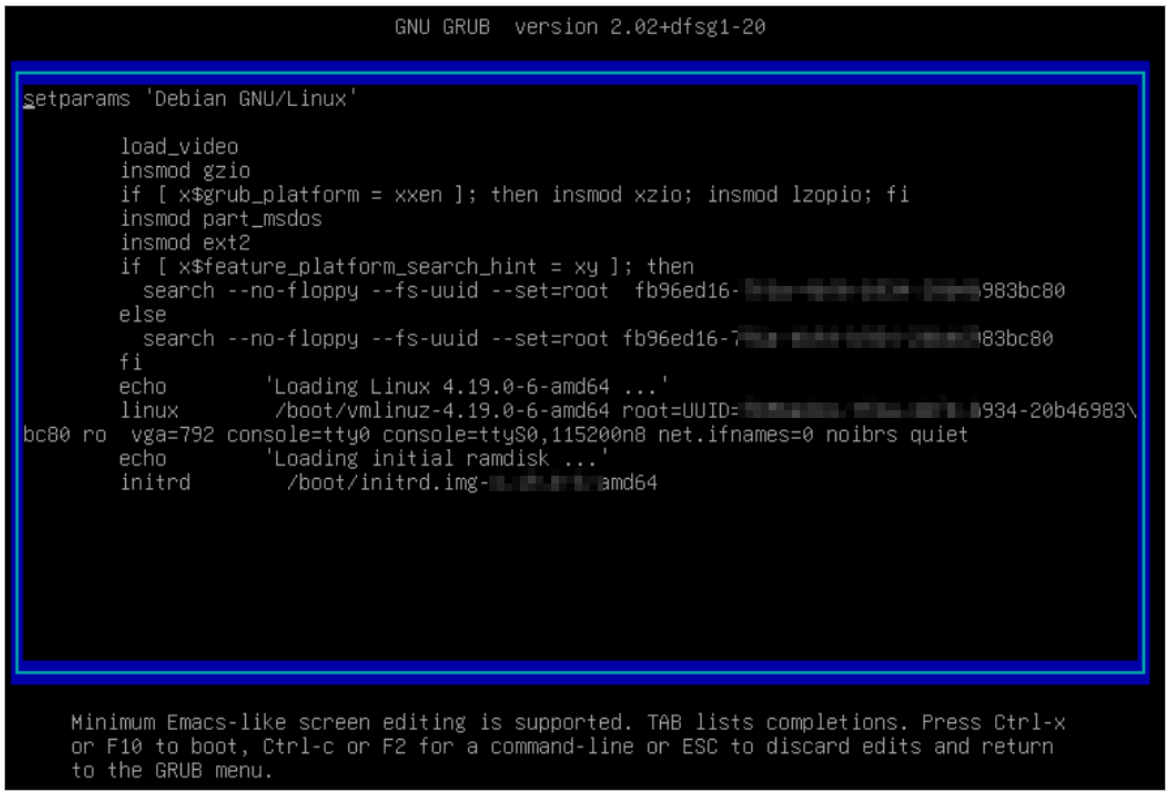
1. 远程连接ECS实例。

   连接方式请参见通过密码认证登录Linux实例。

   > **说明** 使用Workbench和SSH命令远程连接的实例，在通过命令重启时不能直接进入启动系统页面，因此不建议使用这两种连接方式。

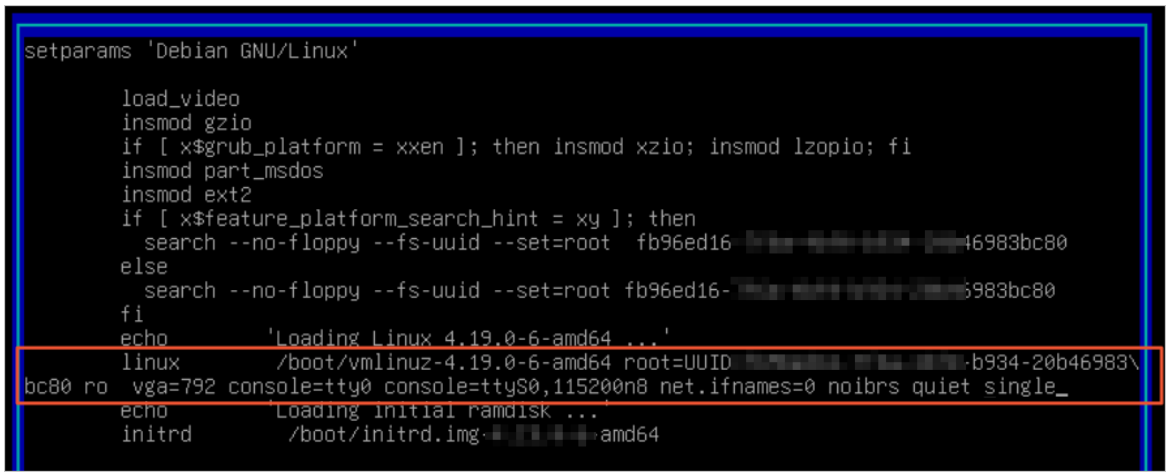
2. 运行 `reboot` 重启ECS实例，并在重启过程中出现内核项界面时按下键盘*e*键，进入GRUB界面。

   GRUB界面如下。



3. 使用键盘的方向键，移动光标至 `linux` 开头的一行，并在本行末尾添加 `single` 。

   修改后的信息如图所示。



4. 按下键盘的*ctrl+x*组合键或按*F10*键启动系统，并输入root用户的密码。

   系统会进入单用户模式。

示例三：SLES操作系统进入单用户模式

本示例中连接SUSE Linux Enterprise Server 15 SP1 64位操作系统的ECS实例。

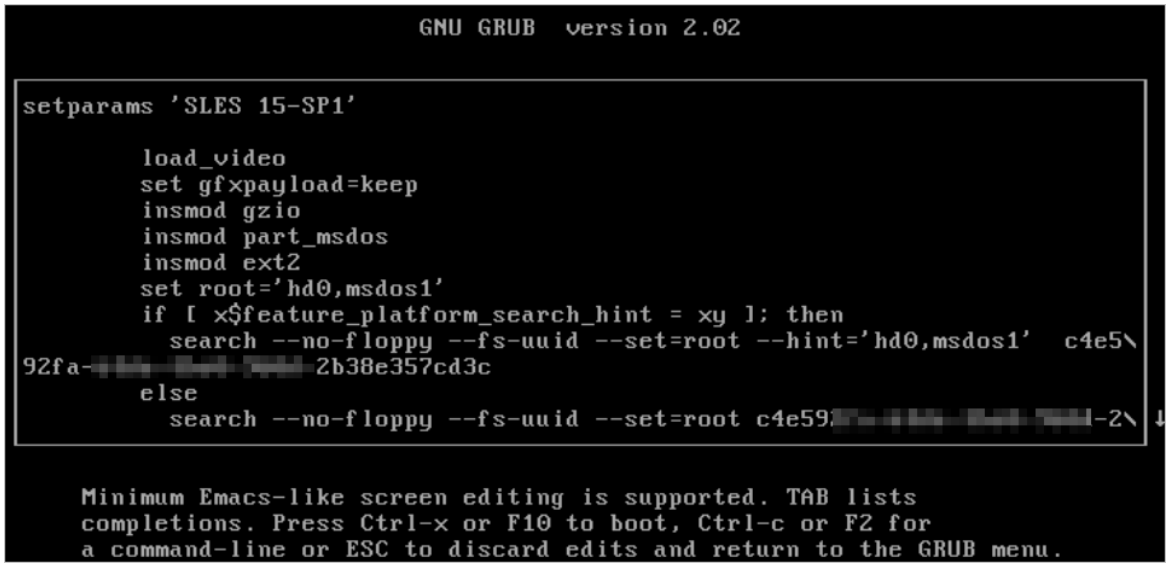
1. 远程连接ECS实例。

   连接方式请参见通过密码认证登录Linux实例。

   > **说明** 使用Workbench和SSH命令远程连接的实例，在通过命令重启时不能直接进入启动系统页面，因此不建议使用这两种连接方式。

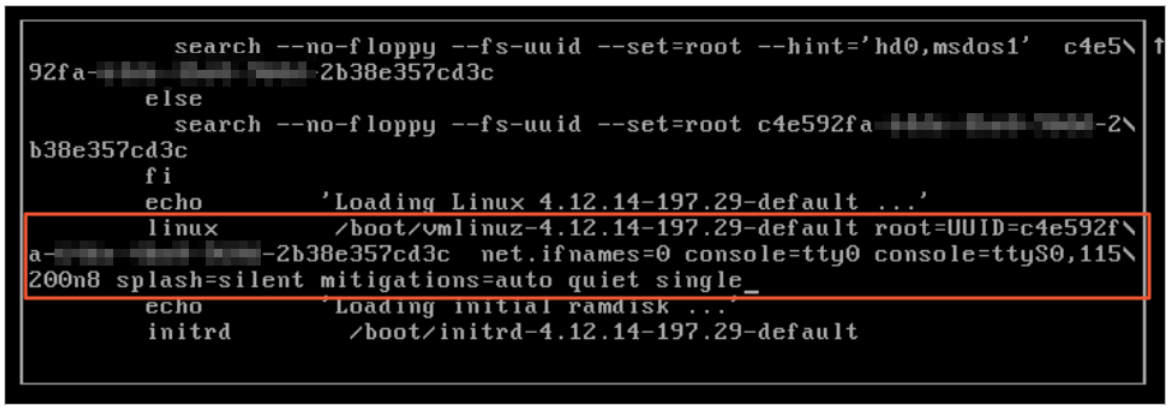
2. 运行 `reboot` 重启ECS实例，并在重启过程中出现内核项界面时按下键盘*e*键，进入GRUB界面。

   GRUB界面如下。



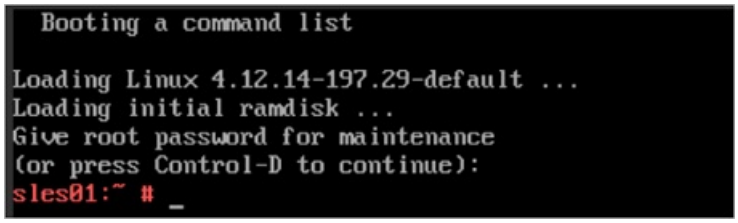
3. 使用键盘的方向键，移动光标向下至 `linux` 开头的一行，并在本行末尾添加 `single` 。

   修改后的信息如图所示。



4. 按下键盘的*ctrl+x*组合键或按*F10*键启动系统，并输入root用户的密码。

   系统会进入单用户模式。



示例四：Ubuntu操作系统进入单用户模式

本示例中连接Ubuntu 18.04 64位操作系统的ECS实例。

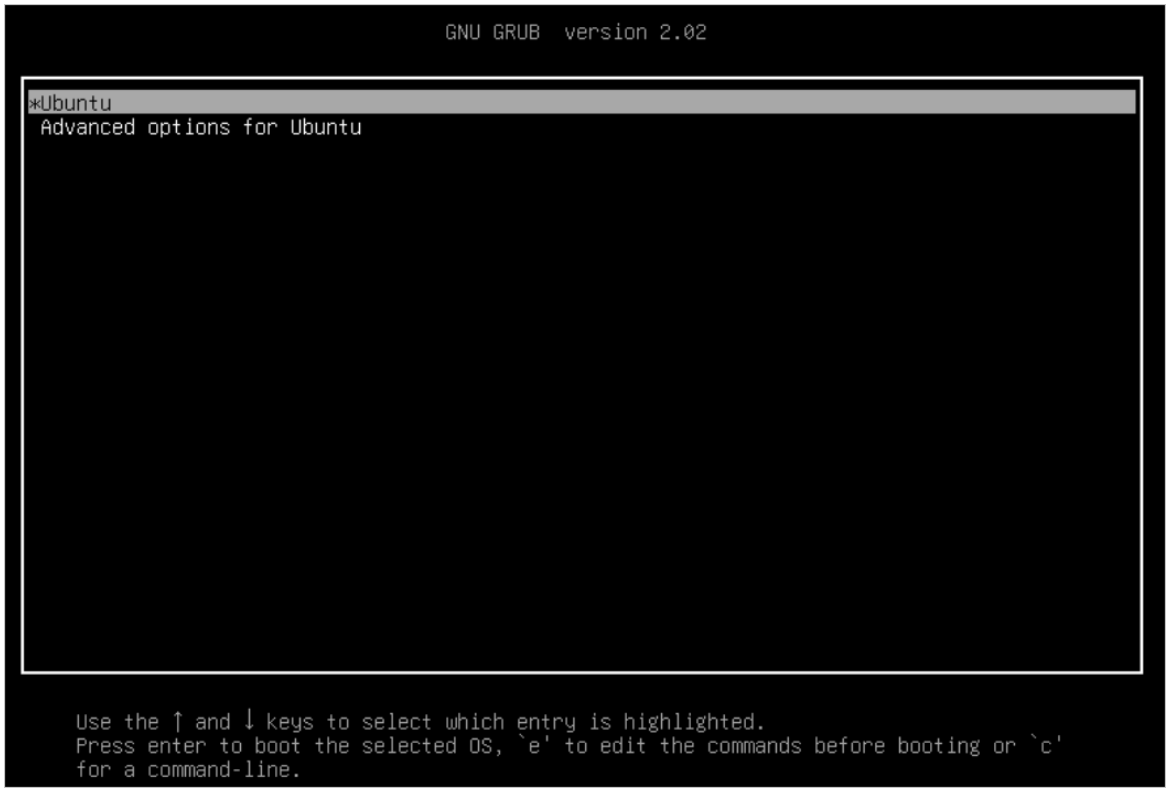
1. 远程连接ECS实例。

   连接方式请参见通过密码认证登录Linux实例。

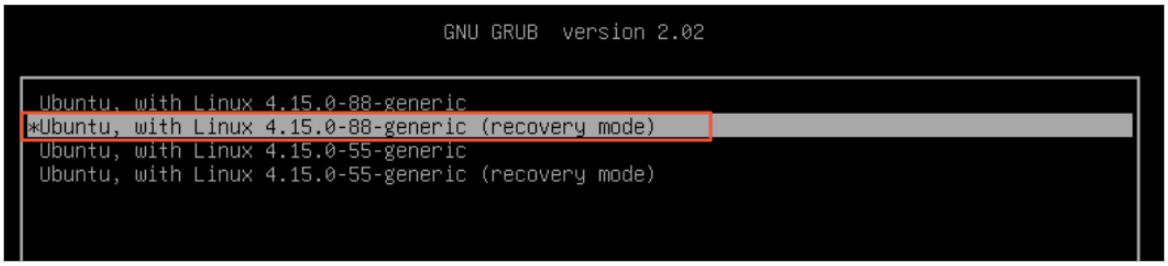
   > **说明** 使用Workbench和SSH命令远程连接的实例，在通过命令重启时不能直接进入启动系统页面，因此不建议使用这两种连接方式。

2. 运行 `reboot` 重启ECS实例，并在重启过程中按下键盘*shift*键，进入GRUB界面。

   GRUB界面示例如下。

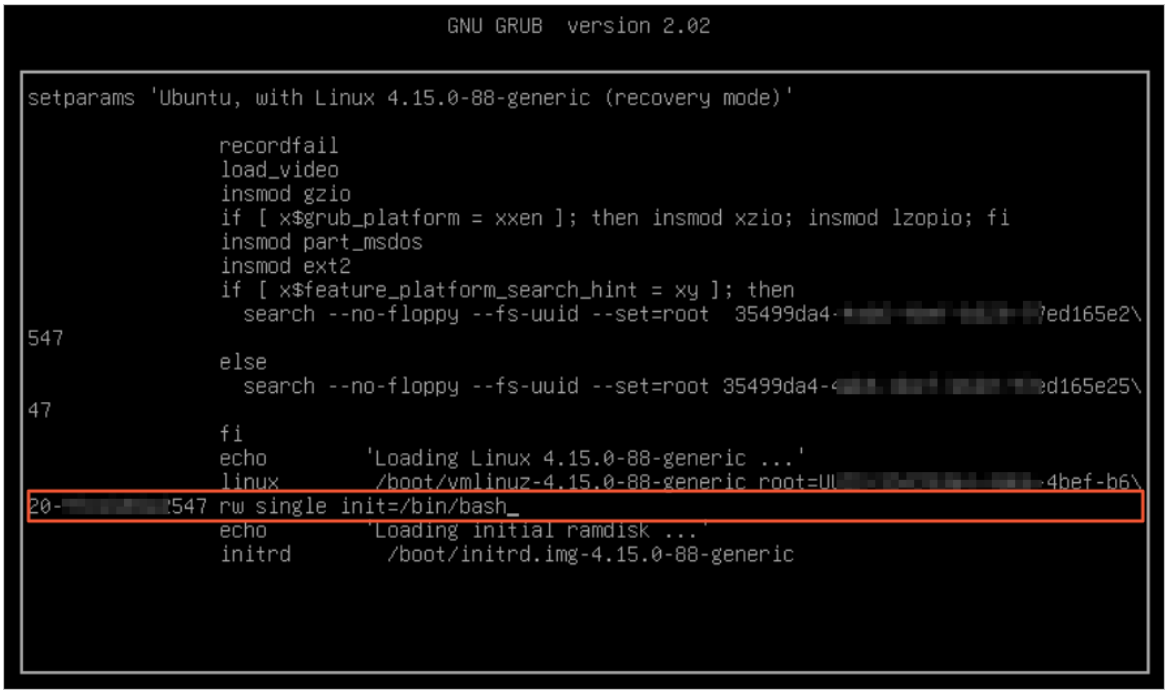


3. 选择GRUB页面第二行的高级选项，并按下键盘的*enter*键。
4. 在跳转页面选择第二行的恢复模式，并按下键盘的*e*键编辑启动项。



5. 在编辑页面，使用键盘的方向键，移动光标向下至 `linux` 开头的一行，并在本行中将 `ro` 至末尾的内容替换为 `rw single init=/bin/bash` 。

   修改结果如下图所示。



6. 按下键盘的*ctrl+x*组合键或按*F10*键。

   系统会直接进入单用户模式。重置系统密码示例如图所示。

11.块存储

11.1. 扩展分区和文件系统_Linux系统盘

本文提供了如何使用growpart或者xfsprogs等工具完成Linux系统盘的扩展分区和文件系统的操作指导。

前提条件

在扩展系统盘分区和文件系统前，请提前完成以下工作。

1. 已创建快照备份数据。

   为防止操作失误导致数据丢失，建议您操作前使用快照备份数据。具体操作请参见创建一个云盘快照。

2. 已在控制台上扩容云盘。

   若尚未扩容，请参见步骤二：在控制台扩容云盘容量。

3. 远程连接ECS实例。

   关于连接方式的介绍，请参见连接方式概述ECS远程连接操作指南。

4. 根据操作系统安装growpart或者xfsprogs扩容格式化工具。

   - Alibaba Cloud Linux 2、CentOS 7

     运行 `yum install <package_name>` 命令安装工具，示例如下。

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

   - Ubuntu 14、Ubuntu 16、Ubuntu 18、Debian 9

     运行 `apt install <package_name>` 命令安装工具，示例如下。

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

   - Debian 8、OpenSUSE 42.3、OpenSUSE 13.1、SUSE Linux Enterprise Server 12 SP2

     请使用上游版本（upstream）的growpart或者xfsprogs工具。

     > **说明** 当出现因扩容格式化工具问题导致的扩容失败时，建议您卸载工具后重新安装。

5. 运行 `uname -a` 命令查看实例的内核版本。

   - 如果内核版本大于等于3.6.0，请参见高内核版本的操作步骤。
   - 如果内核版本小于3.6.0，如CentOS 6、Debian 7和SUSE Linux Enterprise Server 11 SP4等发行版，需要经过一次控制台重启或者API重启才能完成分区扩容。请参见低内核版本的操作步骤。

背景信息

本文的操作步骤适用于以下分区和文件系统格式的云盘。

- 分区格式支持MBR、GPT
- 文件系统支持ext*、xfs、btrfs

扩展高内核版本实例的系统盘分区和文件系统

本节以Alibaba Cloud Linux 2.1903 LTS 64位操作系统为例，说明扩展分区和文件系统的步骤。

> **说明** 本示例操作命令同样适用于CentOS 7系统。

1. 运行以下命令查看现有云盘大小。

```
fdisk -l
```

以下示例返回云盘（*/dev/vda*）容量是100 GiB。

```
[root@ecshost ~]# fdisk -l
Disk /dev/vda: 107.4 GB, 107374182400 bytes, 209715200 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000bad2b
   Device Boot      Start         End      Blocks   Id  System
/dev/vda1   *        2048    83886046    41941999   83  Linux
```

2. 运行以下命令查看云盘分区大小和文件系统类型。

```
df -Th
```

以下示例返回分区（*/dev/vda1*）容量是40 GiB，文件系统类型为ext4。

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

3. 运行以下命令扩容分区。

```
growpart <DeviceName> <PartionNumber>
```

其中，*<DeviceName>*是系统盘的设备名称，*<PartionNumber>*是分区编号，且设备名称和分区编号之间需要空格分隔。

以下示例命令表示扩容系统盘的第一个分区。

```
[root@ecshost ~]# growpart /dev/vda 1
CHANGED: partition=1 start=2048 old: size=83883999 end=83886047 new: size=209713119 end
=209715167
```

> **说明**
>
> - 如果单盘有多个连续分区的情况，例如系统盘 /dev/vda有三个分区 `/dev/vda1` 、 `/dev/vda2` 和 `/dev/vda3` 。扩容时，只需要扩容最后一个分区即可，即执行 `growpart /dev/vda 3` ，即可完成系统盘 /dev/vda的分区扩容。
> - 如果您在运行 `growpart /dev/vda 1` 时，系统提示 `unexpected output in sfdisk --version [sfdisk，来自 util-linux 2.23.2]` ，可以尝试修改字符编码解决问题。具体操作，请参见常见问题。

4. 扩展文件系统。

   请先使用 `df -Th` 命令查看文件系统类型，然后根据不同的文件系统类型运行以下命令扩展文件系统。

   - ext*文件系统（例如ext3和ext4）：运行以下命令扩展文件系统。

     ```
     resize2fs <PartitionName>
     ```

     示例命令表示为扩容系统盘的*/dev/vda1*分区的文件系统。

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

   - xfs文件系统：运行以下命令扩展文件系统。

     ```
     xfs_growfs <mountpoint>
     ```

     示例命令表示为扩容系统盘的*/dev/vda1*分区的文件系统。其中根目录（/）为*/dev/vda1*的挂载点。

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

     > **说明** 不同版本的xfs_growfs命令可能存在差异，您可以运行 `xfs_growfs --help` 查看对应的命令。

   - btrfs文件系统：运行以下命令扩展文件系统。

     ```
     btrfs filesystem resize max <mountpoint>
     ```

示例命令表示为扩容系统盘的*/dev/vda1*分区的文件系统。其中根目录（/）为*/dev/vda1*的挂载点。

```
[root@ecshost ~]# btrfs filesystem resize max /
```

5. 运行以下命令检查云盘扩容结果。

```
df -h
```

以下示例返回分区（*/dev/vda1*）容量是100 GiB，表示已经成功扩容。

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

扩展低内核版本实例的系统盘分区和文件系统

本节以CentOS 6操作系统为例，说明扩展分区和文件系统的步骤。

1. 切换CentOS 6的yum软件源。

   CentOS 6操作系统版本结束了生命周期（EOL），如果您需要在CentOS 6上通过yum安装软件，需要先切换yum软件源。具体操作，请参见CentOS 6 EOL如何切换源？。

2. 运行以下命令安装dracut-modules-growroot工具。

```
yum install -y dracut-modules-growroot
```

> **说明** 如果您使用的是其他软件包管理器，请将yum修改为对应的命令。

3. 运行以下命令覆盖已有的initramfs文件。

```
dracut -f
```

4. 运行以下命令查看云盘和分区信息。
   - 查看云盘大小：

```
fdisk -l
```

以下示例返回云盘（*/dev/vda*）容量是100 GiB。

```
[root@ecshost ~]# fdisk -l
Disk /dev/vda: 107.4 GB, 107374182400 bytes
255 heads, 63 sectors/track, 13054 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0003a7b4
   Device Boot      Start         End      Blocks   Id  System
/dev/vda1   *           1        2611    20970496   83  Linux
```

- 查看云盘分区和文件系统类型：

```
df -Th
```

以下示例返回分区（*/dev/vda1*）容量是20 GiB，文件系统类型为ext4。

```
[root@ecshost ~]# df -Th
Filesystem     Type      Size  Used Avail Use% Mounted on
/dev/vda1      ext4       20G  1.1G   18G   6% /
tmpfs          tmpfs     7.8G     0  7.8G   0% /dev/shm
```

5. 运行以下命令扩容分区。

```
growpart <DeviceName> <PartionNumber>
```

其中，*<DeviceName>*是系统盘的设备名称，*<PartionNumber>*是分区编号，且设备名称和分区编号之间需要空格分隔。

以下示例命令表示扩容系统盘的第一个分区。

```
[root@ecshost ~]# growpart /dev/vda 1
CHANGED: partition=1 start=2048 old: size=41940992 end=41943040 new: size=209710462,end
=209712510
```

6. 在控制台重启实例。

> **注意** 只能通过控制台重启实例或者调用API RebootInstance重启实例，扩容操作才能生效。具体操作，请参见重启实例和RebootInstance。

7. 再次远程连接实例。
8. 扩展文件系统。

请先使用 `df -Th` 命令查看文件系统类型，然后根据不同的文件系统类型运行以下命令扩展文件系统。

- ext*文件系统（例如ext3和ext4）：运行以下命令扩展文件系统。

```
resize2fs <PartitionName>
```

示例命令表示为扩容系统盘的*/dev/vda1*分区的文件系统。

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

- xfs文件系统：运行以下命令扩展文件系统。

```
xfs_growfs <mountpoint>
```

示例命令表示为扩容系统盘的*/dev/vda1*分区的文件系统。其中根目录（/）为*/dev/vda1*的挂载点。

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

> **说明** 不同版本的xfs_growfs命令可能存在差异，请运行 `xfs_growfs --help` 查看对应的命令。

9. 运行以下命令查看云盘分区大小。

```
df -h
```

以下示例返回分区（*/dev/vda1*）容量是100 GiB，表示已经成功扩容。

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

相关文档

- 在线扩容云盘（Linux系统）
- 离线扩容云盘（Linux系统）
- 扩展分区和文件系统_Linux数据盘

11.2. 扩展分区和文件系统_Linux数据盘

扩容云盘（ResizeDisk）只是扩大云盘的存储容量，不会扩容ECS实例的文件系统，您需要按照本文步骤扩容文件系统，实现ECS实例存储空间的扩展。

前提条件

1. 已创建快照备份数据。

   为防止操作失误导致数据丢失，建议您操作前使用快照备份数据。具体操作请参见创建一个云盘快照。

2. 已在控制台上扩容云盘。

   若尚未扩容，请参见步骤二：在控制台扩容云盘容量。

3. 远程连接ECS实例。

   关于连接方式的介绍，请参见连接方式概述ECS远程连接操作指南。

背景信息

本文示例中使用的配置如下：

- ECS实例的操作系统：公共镜像Alibaba Cloud Linux 2.1903 LTS 64位
- 数据盘：高效云盘
- 数据盘设备名：*/dev/vdb*

若您使用的操作系统和数据盘设备名与本文示例不同，请根据实际情况调整命令或参数配置。

确认分区表格式和文件系统

1. 运行以下命令确认数据盘的分区表格式。

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

本示例中，原有的数据盘空间已做分区*/dev/vdb1*。

- 如果 `System` 为 `Linux`，说明数据盘使用的是MBR分区表格式。
- 如果 `System` 为 `GPT`，说明数据盘使用的是GPT分区表格式。

```
[root@ecshost ~]# fdisk -lu /dev/vdb
Disk /dev/vdb: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x9277b47b
Device Boot Start End Blocks Id System
/dev/vdb1 2048 41943039 20970496 83 Linux
```

2. 运行以下命令确认已有分区的文件系统类型。

```
blkid /dev/vdb1
```

本示例中，*/dev/vdb1*的文件系统类型为ext4。

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

> **说明** 未分区并且未创建文件系统的数据盘，以及已分区但未创建文件系统的数据盘，不会返回结果。

3. 运行以下命令确认文件系统的状态。
   - ext*文件系统：

     ```
     e2fsck -n /dev/vdb1
     ```

   - xfs文件系统：

     ```
     xfs_repair -n /dev/vdb1
     ```

   - btrfs文件系统：

     ```
     btrfsck /dev/vdb1
     ```

   不同文件系统的查询示例如下：

   - ext*和xfs文件系统的示例结果如下所示，当文件系统状态为clean，表示文件系统状态正常。如果状态不是clean，请排查并修复。

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

   - btrfs文件系统的示例结果如下所示，例如 `found 114688 bytes used err is 0` 表示文件系统状态正常。如果查询结果中存在报错提示，请排查并修复。

```
[root@ecshost ~]# btrfsck /dev/vdb1
Checking filesystem on /dev/vdb1
UUID: 1234b7a7-68ff-4f48-a88c-8943f27f1234
checking extents
checking free space cache
checking fs roots
checking csums
checking root refs
found 114688 bytes used err is 0
total csum bytes: 0
total tree bytes: 114688
total fs tree bytes: 32768
total extent tree bytes: 16384
btree space waste bytes: 109471
file data blocks allocated: 0
referenced 0
```

选择扩容分区或文件系统的方式

根据您查询到的分区格式和文件系统情况确定操作选项。

扩容场景	相关操作
数据盘已分区并创建文件系统	• 如果您需要扩展数据盘已有的MBR分区，请参见选项一：扩展已有MBR分区。 • 如果新增空间用于增加新的MBR分区，请参见选项二：新增并格式化MBR分区。 • 如果您需要扩展数据盘已有的GPT分区，请参见选项三：扩展已有GPT分区。 • 如果新增空间用于增加新的GPT分区，请参见选项四：新增并格式化GPT分区。
全新数据盘，未分区，未创建文件系统	在控制台扩容数据盘空间后，请参见分区格式化数据盘（Linux）或者分区格式化大于2 TiB数据盘。
数据盘是裸设备，已创建文件系统，未分区	在控制台扩容数据盘空间后，请参见选项五：扩容裸设备文件系统。
数据盘未挂载到实例上	挂载数据盘到实例后，参见本文档的操作步骤完成扩容。

说明

- 如果一个已有分区采用了MBR分区格式，则不支持扩容到2 TiB及以上。为避免造成数据丢失，建议您创建一块大于2TiB的云盘，格式化一个GPT分区，再将MBR分区中的数据拷贝到GPT分区中。具体操作，请参见分区格式化大于2 TiB数据盘。
- 当出现因扩容格式化工具问题导致的扩容失败时，您可以提前升级工具版本，或者卸载工具后重新安装。

选项一：扩展已有MBR分区

> **说明** 为了防止数据丢失，不建议扩容已挂载的分区和文件系统。请先取消挂载（umount）分区，完成扩容并正常使用后，重新挂载（mount）。针对不同的Linux内核版本，推荐以下操作方式：
>
> - 实例内核版本小于3.6：先取消挂载该分区，再修改分区表，最后扩容文件系统。
> - 实例内核版本大于等于3.6：先修改对应分区表，再通知内核更新分区表，最后扩容文件系统。

如果新增空间用于扩容已有的MBR分区，按照以下步骤在实例中完成扩容：

1. 修改分区表。
   i. 运行以下命令查看分区信息，并记录旧分区的起始和结束的扇区位置。

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

   本示例中，分区*/dev/vdb1*的起始扇区是2048，结束扇区是41943039。

   ```
   [root@ecshost ~]# fdisk -lu /dev/vdb
   Disk /dev/vdb: 42.9 GB, 42949672960 bytes, 83886080 sectors
   Units = sectors of 1 * 512 = 512 bytes
   Sector size (logical/physical): 512 bytes / 512 bytes
   I/O size (minimum/optimal): 512 bytes / 512 bytes
   Disk label type: dos
   Disk identifier: 0x9277b47b
   Device Boot Start End Blocks Id System
   /dev/vdb1 2048 41943039 20970496 83 Linux
   ```

   ii. 查看数据盘的挂载路径，根据返回的文件路径卸载分区，直至完全卸载已挂载的分区。

   查看挂载（mount）信息。

   ```
   mount | grep "/dev/vdb"
   ```

   取消挂载（umount）数据盘。

   ```
   umount /dev/vdb1
   ```

   查看操作结果。

   ```
   mount | grep "/dev/vdb"
   ```

   示例结果如下所示。

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

iii. 使用fdisk工具删除旧分区。

> **警告** 删除旧分区如果出错，可能会删除分区内的数据。如有重要数据（例如数据库中的用户数据），请在操作前进行备份，避免因删除旧分区而造成数据丢失。

a. 运行 `fdisk -u /dev/vdb` ：分区数据盘。

b. 输入*p*：打印分区表。

c. 输入*d*：删除分区。

d. 输入*p*：确认分区已删除。

e. 输入*w*：保存修改并退出。

以下为删除旧分区的命令行交互示例。

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

iv. 使用fdisk命令新建分区。

a. 运行 `fdisk -u /dev/vdb` ：分区数据盘。

b. 输入*p*：打印分区表。

c. 输入*n*：新建分区。

d. 输入*p*：选择分区类型为主分区。

e. 输入*<分区号>*：选择分区号。本示例选取了1。

f. 设置新分区的起始位置和结束位置。

> **警告** 新分区的起始位置必须和旧分区的起始位置相同，结束位置必须大于旧分区的结束位置，否则会导致扩容失败。具体问题与解决方案请参见使用fdisk扩容新分区起始位置无法与扩容前保持一致。

g. 输入*w*：保存修改并退出。

以下为扩容分区的命令行交互示例。本示例中，将*/dev/vdb*1由20GiB扩容到40GiB。

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

v. 运行以下任一命令通知内核更新分区表。

```
partprobe /dev/vdb
```

> **说明** 如果您在Centos 6操作系统上，发现报错信息 `-bash: partprobe: command not found` ，可先切换yum源，具体操作，请参见CentOS 6 EOL如何切换源？；再运行命令 `yum install -y parted` 安装parted；最后重新运行命令。

```
partx -u /dev/vdb1
```

vi. 运行以下命令确保分区表已经增加。

```
lsblk /dev/vdb
```

vii. 运行以下命令再次检查文件系统，确认扩容分区后的文件系统状态为clean。

```
e2fsck -f /dev/vdb1
```

> **说明** 如果运行命令后未显示文件系统状态为clean，您可以尝试用 `e2fsck -n /dev/vdb1` 检查。

2. 扩容文件系统。
   - ext*文件系统（例如ext3和ext4）：依次运行以下命令调整ext*文件系统大小并重新挂载分区。

     调整ext*文件系统大小。

     ```
     resize2fs /dev/vdb1
     ```

     分区挂载到*/mnt*。

     ```
     mount /dev/vdb1 /mnt
     ```

   - xfs文件系统：依次运行以下命令先重新挂载分区，再调整xfs文件系统大小。

     分区挂载到*/mnt*。

     ```
     mount /dev/vdb1 /mnt
     ```

     调整xfs文件系统大小。

     ```
     xfs_growfs /mnt
     ```

     > **说明** 新版xfs_growfs根据挂载点识别待扩容设备，例如 `xfs_growfs /mnt` 。您可以运行 `xfs_growfs --help` 查看不同版本xfs_growfs的使用方法。

   - btrfs文件系统：依次运行以下命令先重新挂载分区，再调整btrfs文件系统大小。

     分区挂载到*/mnt*。

     ```
     mount /dev/vdb1 /mnt
     ```

     调整btrfs文件系统大小。

```
btrfs filesystem resize max /mnt
```

选项二：新增并格式化MBR分区

如果新增空间用于增加新的MBR分区，按照以下步骤在实例中完成扩容：

1. 运行以下命令新建分区。

   ```
   fdisk -u /dev/vdb
   ```

   以下为新建分区的命令行交互示例。本示例中，为新增的20GiB新建分区，作为*/dev/vdb2*使用。

   ```
   [root@ecshost ~]# fdisk -u /dev/vdb
   Welcome to fdisk (util-linux 2.23.2).
   Changes will remain in memory only, until you decide to write them.
   Be careful before using the write commad.
   Command (m for help): p
   Disk /dev/vdb: 42.9 GB, 42949672960 bytes, 83886080 sectors
   Units = sectors of 1 * 512 = 512 bytes
   Sector size (logical/physical): 512 bytes / 512 bytes
   I/O size (minimum/optimal): 512 bytes / 512 bytes
   Disk label type: dos
   Disk identifier: 0x2b31a2a3
      Device Boot      Start         End      Blocks   Id  System
   /dev/vdb1            2048    41943039    20970496   83  Linux
   Command (m for help): n
   Partition type:
      p   primary (1 primary, 0 extended, 3 free)
      e   extended
   Select (default p): p
   Partition number (2-4, default 2): 2
   First sector (41943040-83886079, default 41943040):
   Using default value 41943040
   Last sector, +sectors or +size{K,M,G} (41943040-83886079, default 83886079):
   Using default value 83886079
   Partition 2 of type Linux and of size 20 GiB is set
   Command (m for help): w
   The partition table has been altered!
   Calling ioctl() to re-read partition table.
   Syncing disks.
   ```

2. 运行以下命令查看分区。

   ```
   lsblk /dev/vdb
   ```

   示例结果如下所示。

   ```
   [root@ecshost ~]# lsblk /dev/vdb
   NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
   vdb    253:16   0  40G  0 disk
   ├─vdb1 253:17   0  20G  0 part
   └─vdb2 253:18   0  20G  0 part
   ```

3. 运行以下命令创建文件系统。
   - 创建ext4文件系统：

```
mkfs.ext4 /dev/vdb2
```

- 创建xfs文件系统：

```
mkfs.xfs -f /dev/vdb2
```

- 创建btrfs文件系统：

```
mkfs.btrfs /dev/vdb2
```

4. 运行以下命令查看文件系统信息。

```
blkid /dev/vdb2
```

示例结果如下所示。

```
[root@ecshost ~]# blkid /dev/vdb2
/dev/vdb2: UUID="e3f336dc-d534-4fdd-****-b6ff1a55bdbb" TYPE="ext4"
```

5. 运行以下命令挂载分区。

```
mount /dev/vdb2 /mnt
```

6. 运行以下命令查看目前数据盘空间和使用情况。

```
df -h
```

显示新建文件系统的信息，表示挂载成功。

```
[root@ecshost ~]# df -h
Filesystem Size Used Avail Use% Mounted on
/dev/vda1 40G 1.6G 36G 5% /
devtmpfs 3.9G 0 3.9G 0% /dev
tmpfs 3.9G 0 3.9G 0% /dev/shm
tmpfs 3.9G 460K 3.9G 1% /run
tmpfs 3.9G 0 3.9G 0% /sys/fs/cgroup
/dev/vdb2 9.8G 37M 9.2G 1% /mnt
tmpfs 783M 0 783M 0% /run/user/0
```

选项三：扩展已有GPT分区

如果新增空间用于扩容已有的GPT分区，按照以下步骤在实例中完成扩容：

1. 查看数据盘的挂载路径，根据返回的文件路径卸载分区，直至完全卸载已挂载的分区。

查看挂载（mount）信息。

```
mount | grep "/dev/vdb"
```

取消挂载（umount）数据盘。

```
umount /dev/vdb1
```

查看操作结果。

```
mount | grep "/dev/vdb"
```

示例结果如下所示。

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

2. 使用Parted工具为现有GPT分区分配容量。
   i. 运行以下命令进入Parted分区工具。

   ```
   parted /dev/vdb
   ```

   如需查看Parted工具使用说明，运行 `help` 命令。

   ii. 运行以下命令查看分区信息，并记录现有分区的分区号和起始扇区的值。

   ```
   print
   ```

   若界面提示 `Fix/Ignore/Cancel?` 和 `Fix/Ignore?` ，均输入*Fix*即可。

   本示例中，现有分区大小为1TiB，分区号（即 `Number` 的值）为 `1` ，起始扇区（即 `Start` ）的值为 `1049kB` 。

   ```
   (parted) print
   Model: Virtio Block Device (virtblk)
   Disk /dev/vdb: 3299GB
   Sector size (logical/physical): 512B/512B
   Partition Table: gpt
   Disk Flags:

   Number  Start   End     Size    File system  Name     Flags
    1      1049kB  1100GB  1100GB  ext4         primary
   ```

   iii. 运行以下命令删除现有分区。

   ```
   rm <分区号>
   ```

   本示例中，现有分区的分区号为 `1` ，因此命令为：

   ```
   rm 1
   ```

   iv. 运行以下命令重新创建主分区。

   ```
   mkpart primary <原分区的起始扇区> <容量分配百分比>
   ```

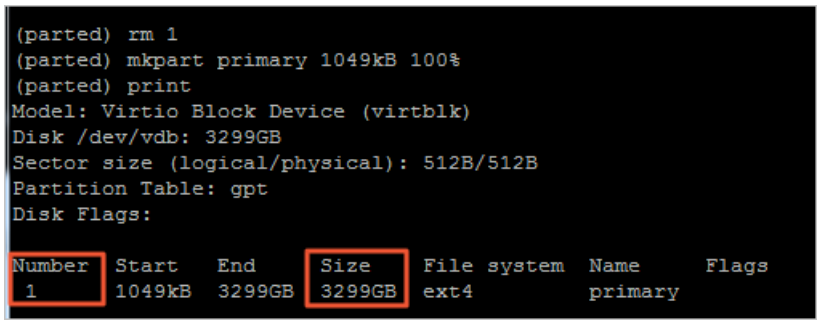
   本示例中，原分区的起始扇区为 `1049kB` ，且要将扩容后的总容量（即3TiB）全部分配给该分区，因此命令为：

   ```
   mkpart primary 1049kB 100%
   ```

v. 运行以下命令查看新分区是否创建成功。

```
print
```

如下图所示，新的GPT分区仍为1号分区，容量已变更为3TiB。



vi. 运行以下命令退出Parted分区工具。

```
quit
```

以下为命令行交互示例。

```
[root@ecshost ~]# parted /dev/vdb
GNU Parted 3.1
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Error: The backup GPT table is not at the end of the disk, as it should be.
This might mean that another operating system believes the disk is smaller.
Fix, by moving the backup to the end (and removing the old backup)?
Fix/Ignore/Cancel? Fix
Warning: Not all of the space available to /dev/vdb appears to be used, you can
fix the GPT to use all of the space (an extra 4294967296 blocks) or continue
with the current setting?
Fix/Ignore? Fix
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 3299GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:
Number  Start   End     Size    File system  Name     Flags
 1      1049kB  1100GB  1100GB  ext4         primary
(parted) rm 1
(parted) mkpart primary 1049kB 100%
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 3299GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:
Number  Start   End     Size    File system  Name     Flags
 1      1049kB  3299GB  3299GB  ext4         primary
(parted) quit
Information: You may need to update /etc/fstab.
```

3. 运行以下命令确认文件系统一致性。

```
fsck -f /dev/vdb1
```

示例结果如下所示。

```
[root@ecshost ~]# fsck -f /dev/vdb1
fsck from util-linux 2.23.2
e2fsck 1.43.5 (04-Aug-2017)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/vdb1: 11/67108864 files (0.0% non-contiguous), 4265369/268434944 blocks
```

4. 扩展分区对应的文件系统并重新挂载分区。
   - ext*文件系统（例如ext3和ext4）：

     运行以下命令调整新分区的ext*文件系统大小。

     ```
     resize2fs /dev/vdb1
     ```

     运行以下命令重新挂载分区。

     ```
     mount /dev/vdb1 /mnt
     ```

   - xfs文件系统：

     运行以下命令重新挂载分区。

     ```
     mount /dev/vdb1 /mnt
     ```

     运行以下命令调整xfs文件系统大小。

     ```
     xfs_growfs /mnt
     ```

     > **说明** 新版xfs_growfs根据挂载点识别待扩容设备，例如 `xfs_growfs /mnt` 。您可以运行 `xfs_growfs --help` 查看不同版本xfs_growfs的使用方法。

   - btrfs文件系统：

     运行以下命令重新挂载分区。

     ```
     mount /dev/vdb1 /mnt
     ```

     运行以下命令调整btrfs文件系统大小。

     ```
     btrfs filesystem resize max /mnt
     ```

选项四：新增并格式化GPT分区

如果新增空间用于增加新的分区并希望使用GPT分区格式，按照以下步骤在实例中完成扩容。示例采用一块32 TiB的数据盘，已有一个4.8TiB的分区*/dev/vdb1*，此次新建了一个*/dev/vdb2*分区。

1. 运行以下命令查看数据盘中已有分区的信息。

```
fdisk -l
```

示例结果如下所示。

```
[root@ecshost ~]# fdisk -l
Disk /dev/vda: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000b1b45
   Device Boot      Start         End      Blocks   Id  System
/dev/vda1   *        2048    83875364    41936658+  83  Linux
WARNING: fdisk GPT support is currently new, and therefore in an experimental phase. Us
e at your own discretion.
Disk /dev/vdb: 35184.4 GB, 35184372088832 bytes, 68719476736 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: gpt
Disk identifier: BCE92401-F427-45CC-8B0D-B30EDF279C2F
#         Start          End    Size  Type            Name
 1         2048  10307921919    4.8T  Microsoft basic mnt
```

2. 使用Parted工具创建新分区并分配容量。
   i. 运行以下命令进入Parted工具。

   ```
   parted /dev/vdb
   ```

   ii. 运行以下命令查看数据盘待分配的容量，记录已有分区的扇区位置和容量。

   ```
   print free
   ```

   示例中*/dev/vdb1*的起始位置为1049KB，结束扇区为5278GB，容量为5278GiB。

   ```
   (parted) print free
   Model: Virtio Block Device (virtblk)
   Disk /dev/vdb: 35.2TB
   Sector size (logical/physical): 512B/512B
   Partition Table: gpt
   Disk Flags:
   Number  Start   End     Size    File system  Name  Flags
           17.4kB  1049kB  1031kB  Free Space
    1      1049kB  5278GB  5278GB  ext4         mnt
           5278GB  35.2TB  29.9TB  Free Space
   ```

   iii. 运行以下命令设置起始扇区和分配容量。

   ```
   mkpart <分区名称> <起始扇区> <容量分配百分比>
   ```

   以下示例新建了一个名为test的*/dev/vdb2*分区，起始扇区为上一个分区的结束扇区，并将所有新增空间分配给该分区。

   ```
   mkpart test 5278GB 100%
   ```

iv. 运行以下命令查看容量（Size）是否发生变化。

```
print
```

示例结果如下所示。

```
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 35.2TB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:
Number  Start   End     Size    File system  Name  Flags
 1      1049kB  5278GB  5278GB  ext4         mnt
 2      5278GB  35.2TB  29.9TB               test
```

v. 运行以下命令退出Parted工具。

```
quit
```

3. 为新分区创建文件系统。
   - 创建ext4文件系统：

     ```
     mkfs.ext4 /dev/vdb2
     ```

   - 创建ext3文件系统：

     ```
     mkfs.ext3 /dev/vdb2
     ```

   - 创建xfs文件系统：

     ```
     mkfs.xfs -f /dev/vdb2
     ```

   - 创建btrfs文件系统：

     ```
     mkfs.btrfs /dev/vdb2
     ```

   示例中创建了一个xfs文件系统，如下所示。

   ```
   [root@ecshost ~]# mkfs -t xfs /dev/vdb2
   meta-data=/dev/vdb2              isize=512    agcount=28, agsize=268435455 blks
            =                       sectsz=512   attr=2, projid32bit=1
            =                       crc=1        finobt=0, sparse=0
   data     =                       bsize=4096   blocks=7301444096, imaxpct=5
            =                       sunit=0      swidth=0 blks
   naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
   log      =internal log           bsize=4096   blocks=521728, version=2
            =                       sectsz=512   sunit=0 blks, lazy-count=1
   realtime =none                   extsz=4096   blocks=0, rtextents=0
   ```

4. 运行以下命令查看分区容量变化。

   ```
   fdisk -l
   ```

   示例结果如下所示。

```
[root@ecshost ~]# fdisk -l
Disk /dev/vda: 42.9 GB, 42949672960 bytes, 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000b1b45
   Device Boot      Start         End      Blocks   Id  System
/dev/vda1   *        2048    83875364    41936658+  83  Linux
WARNING: fdisk GPT support is currently new, and therefore in an experimental phase. Us
e at your own discretion.
Disk /dev/vdb: 35184.4 GB, 35184372088832 bytes, 68719476736 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: gpt
Disk identifier: BCE92401-F427-45CC-8B0D-B30EDF279C2F
#         Start          End    Size  Type            Name
 1         2048  10307921919    4.8T  Microsoft basic mnt
 2  10307921920  68719474687   27.2T  Microsoft basic test
```

5. 运行以下命令查看存储设备的文件系统类型。

```
blkid
```

示例结果如下所示。

```
[root@ecshost ~]# blkid
/dev/vda1: UUID="ed95c595-4813-480e-****-85b1347842e8" TYPE="ext4"
/dev/vdb1: UUID="21e91bbc-7bca-4c08-****-88d5b3a2303d" TYPE="ext4" PARTLABEL="mnt" PART
UUID="576235e0-5e04-4b76-****-741cbc7e98cb"
/dev/vdb2: UUID="a7dcde59-8f0f-4193-****-362a27192fb1" TYPE="xfs" PARTLABEL="test" PART
UUID="464a9fa9-3933-4365-****-c42de62d2864"
```

6. 挂载新分区。

```
mount /dev/vdb2 /mnt
```

选项五：扩容裸设备文件系统

当数据盘没有创建分区，并且在裸设备上创建了文件系统时，您可以参见以下步骤直接扩容文件系统。

1. 运行以下命令查看存储设备的文件系统类型。

```
df -Th
```

以下示例返回分区（/dev/vdb）文件系统类型为xfs。

```
[root@ecshost ~]# df -Th
Filesystem     Type      Size  Used Avail Use% Mounted on
devtmpfs       devtmpfs  434M     0  434M   0% /dev
tmpfs          tmpfs     446M     0  446M   0% /dev/shm
tmpfs          tmpfs     446M  524K  446M   1% /run
tmpfs          tmpfs     446M     0  446M   0% /sys/fs/cgroup
/dev/vda1      ext4       20G  2.5G   17G  14% /
tmpfs          tmpfs      90M     0   90M   0% /run/user/0
/dev/vdb       xfs        20G   53M   20G   1% /mnt
```

2. 根据不同的文件系统类型，执行以下不同的操作来扩容文件系统。
   - xfs文件系统

     直接使用root权限执行xfs_growfs命令扩容文件系统。

     ```
     xfs_growfs /mnt
     ```

     其中，`/mnt` 为文件系统的挂载点。

     > **说明** 新版xfs_growfs根据挂载点识别待扩容设备，例如 `xfs_growfs /mnt` 。您可以运行 `xfs_growfs --help` 查看不同版本xfs_growfs的使用方法。

   - ext*和btrfs文件系统

     a. 查看挂载（mount）信息。

     ```
     mount | grep "/dev/vdb"
     ```

     b. 取消挂载（umount）数据盘。

     ```
     umount /dev/vdb
     ```

     c. 查看操作结果。

     ```
     mount | grep "/dev/vdb"
     ```

     示例结果如下所示。

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

     d. 扩容文件系统。

     - ext*文件系统：使用root权限执行resize2fs命令扩容文件系统。

       ```
       resize2fs /dev/vdb
       ```

     - btrfs文件系统：使用root权限执行btrfs命令扩容文件系统。

       ```
       btrfs filesystem resize max /mnt
       ```

       其中，`/mnt` 为文件系统的挂载点。

e. 将云盘挂载至挂载点。

```
mount /dev/vdb /mnt
```

3. 运行 `df -Th` 查看数据盘扩容结果。

```
df -Th
```

以下示例显示文件系统容量完成扩充，表示扩容成功。

```
[root@ecshost ~]# df -Th
Filesystem     Type      Size  Used Avail Use% Mounted on
devtmpfs       devtmpfs  434M     0  434M   0% /dev
tmpfs          tmpfs     446M     0  446M   0% /dev/shm
tmpfs          tmpfs     446M  524K  446M   1% /run
tmpfs          tmpfs     446M     0  446M   0% /sys/fs/cgroup
/dev/vda1      ext4       20G  2.5G   17G  14% /
tmpfs          tmpfs      90M     0   90M   0% /run/user/0
/dev/vdb       xfs        30G   63M   30G   1% /mnt
```

相关文档

- 在线扩容云盘（Linux系统）
- 在线扩容云盘（Windows系统）
- 扩展分区和文件系统_Linux系统盘

11.3. 使用逻辑卷（Linux）

11.3.1. 通过LVM创建逻辑卷

逻辑卷管理LVM（Logical Volume Manager）是Linux系统的一种管理硬盘分区机制，具有动态管理硬盘的能力。本文介绍了如何通过LVM在多块云盘上创建一个逻辑卷，适用于Linux实例。

前提条件

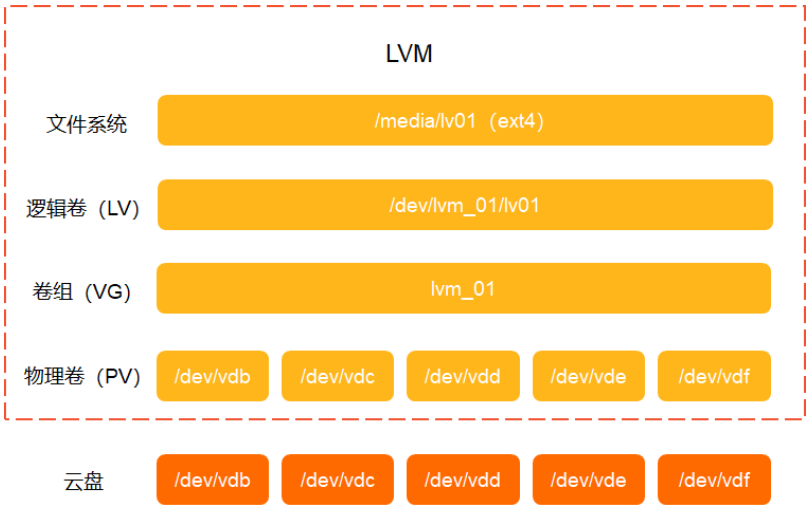
- 您已经创建并挂载了多块云盘。具体操作，请参见创建云盘和挂载数据盘。
- 为防止操作失误导致数据丢失，建议您操作前使用快照一致性组备份数据。具体操作，请参见创建快照一致性组。

背景信息

LVM在硬盘和分区之上建立一个逻辑层，提高了硬盘分区管理的灵活性。 逻辑卷的大小可以动态调整，而且不会丢失现有数据。即使您新增了数据盘，也不会改变现有的逻辑卷。

> **注意**
>
> - 为了防止数据丢失，不能在已有数据的云盘上创建逻辑卷。
> - 由于云盘快照只能备份单块云盘数据，使用LVM后，回滚单个云盘时会造成数据差异。因此，建议您通过快照一致性组进行备份数据。更多信息，请参见创建快照一致性组。

本文中，LVM配置示意图如下所示。



步骤一：创建物理卷

1. 以root权限远程连接ECS实例。

   关于连接方式的介绍，请参见连接方式概述ECS远程连接操作指南。

2. 使用以下命令查看ECS实例上所有云盘信息。

   ```
   lsblk
   ```

   结果如下所示，表示您有五块云盘可以通过LVM创建弹性可扩展的逻辑卷。

   ```
   [root@ecs ~]# lsblk
   NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
   vda    253:0    0  40G  0 disk
   └─vda1 253:1    0  40G  0 part /
   vdb    253:16   0  40G  0 disk
   vdc    253:32   0  40G  0 disk
   vdd    253:48   0  40G  0 disk
   vde    253:64   0  40G  0 disk
   vdf    253:80   0  40G  0 disk
   ```

3. 如果您的ECS实例未安装LVM2工具，使用以下命令安装LVM2。

   ```
   yum install -y lvm2
   ```

4. 使用以下命令创建物理卷PV（Physical Volume）。

   ```
   pvcreate <数据盘设备名称1> ... <数据盘设备名称N>
   ```

   以下示例表示添加/dev/vdb、/dev/vdc、/dev/vdd、/dev/vde、/dev/vdf这5块新创建的数据盘，多个数据盘设备名称之间以空格间隔。您需要根据您的实际情况修改设备名称。

   ```
   pvcreate /dev/vdb /dev/vdc /dev/vdd /dev/vde /dev/vdf
   ```

   结果如下所示。

   ```
   [root@ecs ~]# pvcreate /dev/vdb /dev/vdc /dev/vdd /dev/vde /dev/vdf
     Physical volume "/dev/vdb" successfully created.
     Physical volume "/dev/vdc" successfully created.
     Physical volume "/dev/vdd" successfully created.
     Physical volume "/dev/vde" successfully created.
     Physical volume "/dev/vdf" successfully created.
   ```

5. 使用以下命令查看ECS实例已经创建的物理卷（PV）信息。

```
lvmdiskscan | grep LVM
```

结果如下所示。

```
[root@ecs ~]# lvmdiskscan | grep LVM
  /dev/vdb  [       40.00 GiB] LVM physical volume
  /dev/vdc  [       40.00 GiB] LVM physical volume
  /dev/vdd  [       40.00 GiB] LVM physical volume
  /dev/vde  [       40.00 GiB] LVM physical volume
  /dev/vdf  [       40.00 GiB] LVM physical volume
  5 LVM physical volume whole disks
  0 LVM physical volumes
```

步骤二：创建卷组

1. 使用以下命令创建卷组VG（Volume Group）。

```
vgcreate <卷组名> <物理卷名称1>……<物理卷名称N>
```

以下示例表示创建lvm_01卷组，并添加/dev/vdb、/dev/vdc、/dev/vdd、/dev/vde、/dev/vdf这5块物理卷，多个物理卷名称之间以空格间隔。您需要根据您的实际情况修改卷组名称和物理卷名称。

```
vgcreate lvm_01 /dev/vdb /dev/vdc /dev/vdd /dev/vde /dev/vdf
```

结果如下所示。

```
[root@ecs ~]# vgcreate lvm_01 /dev/vdb /dev/vdc /dev/vdd /dev/vde /dev/vdf
  Volume group "lvm_01" successfully created
```

2. （可选）如果您需要在卷组中添加新的物理卷，使用以下命令添加新的物理卷。

```
vgextend 卷组名称  <物理卷名称1>……<物理卷名称N>
```

以下示例表示在卷组lvm_01中添加新的物理卷/dev/vdg，如果添加多个物理卷，则物理卷名称之间以空格间隔。

```
vgextend lvm_01 /dev/vdg
```

结果如下所示。

```
[root@ecs ~]# vgextend lvm_01 /dev/vdg
  Volume group "lvm_01" successfully extended
```

3. 使用以下命令查看卷组信息。

```
vgs
```

结果如下所示。

```
[root@ecs ~]# vgs
  VG     #PV #LV #SN Attr   VSize   VFree
  lvm_01   5   0   0 wz--n- 199.98g 199.98g
```

步骤三：创建逻辑卷

1. 使用以下命令创建逻辑卷LV（Logical Volume）。

```
lvcreate [-L <逻辑卷大小>][ -n <逻辑卷名称>] <卷组名称>
```

> 说明
> - 逻辑卷大小：逻辑卷的大小应小于卷组（VG）剩余可用空间，容量单位支持M、G或者T。
> - 逻辑卷名称：由您自定义。
> - 卷组名称：已经创建的卷组的名称。

以下示例创建一个150 GiB的逻辑卷（LV）。

```
lvcreate -L 150g -n lv01 lvm_01
```

结果如下所示。

```
[root@ecs ~]# lvcreate -L 150g -n lv01 lvm_01
  Logical volume "lv01" created.
```

2. 使用以下命令查看逻辑卷详细信息。

```
lvdisplay
```

结果如下所示。

```
[root@ecs ~]# lvdisplay
  --- Logical volume ---
  LV Path                /dev/lvm_01/lv01
  LV Name                lv01
  VG Name                lvm_01
  LV UUID                3dP9u0-6htd-PPYW-qlfZ-[redacted]
  LV Write Access        read/write
  LV Creation host, time ecs, 2021-06-03 11:37:55 +0800
  LV Status              available
  # open                 0
  LV Size                150.00 GiB
  Current LE             38400
  Segments               4
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
  Block device           252:0
```

步骤四：创建并挂载文件系统

1. 使用以下命令在逻辑卷（LV）上创建文件系统。

```
mkfs.<文件系统格式> <逻辑卷路径>
```

您可以根据需要创建文件系统类型，以下以ext4和xfs文件系统为例：

- 创建一个ext4文件系统

```
mkfs.ext4 /dev/lvm_01/lv01
```

结果如下所示。

```
[root@ecs ~]# mkfs.ext4 /dev/lvm_01/lv01
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
9830400 inodes, 39321600 blocks
1966080 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2187329536
1200 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

- 创建一个xfs文件系统

```
mkfs.xfs /dev/lvm_01/lv01
```

结果如下所示。

```
[root@ecs ~]# mkfs.xfs /dev/lvm_01/lv01
meta-data=/dev/lvm_01/lv01       isize=512    agcount=4, agsize=9830400 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=0, sparse=0
data     =                       bsize=4096   blocks=39321600, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal log           bsize=4096   blocks=19200, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

2. 创建挂载点，例如/media/lv01。

   如果您使用已有的挂载点，可以跳过此步骤。

```
mkdir /media/lv01
```

3. 使用以下命令挂载文件系统。

   本示例中，逻辑卷路径为/dev/lvm_01/lv01，挂载点为/media/lv01，您需要根据实际情况修改。

```
mount /dev/lvm_01/lv01 /media/lv01
```

4. 使用以下命令查看逻辑卷的挂载信息。

```
df -h
```

结果如下所示。

```
[root@ecs ~]# df -h
Filesystem               Size  Used Avail Use% Mounted on
/dev/vda1                 40G  2.0G   36G   6% /
devtmpfs                 3.8G     0  3.8G   0% /dev
tmpfs                    3.8G     0  3.8G   0% /dev/shm
tmpfs                    3.8G  488K  3.8G   1% /run
tmpfs                    3.8G     0  3.8G   0% /sys/fs/cgroup
tmpfs                    768M     0  768M   0% /run/user/0
/dev/mapper/lvm_01-lv01  150G   33M  150G   1% /media/lv01
```

相关文档

- 通过LVM扩容逻辑卷

11.3.2. 通过LVM扩容逻辑卷

本文介绍了如何通过LVM（Logical Volume Manager）扩容一个逻辑卷LV（Logical Volume），适用于Linux系统ECS实例。

前提条件

- 您已经创建了一个逻辑卷。具体操作，请参见通过LVM创建逻辑卷。
- 云盘已经在控制台完成扩容。具体操作，请参见步骤二：在控制台扩容云盘容量。本文示例为/dev/vdf扩容了40 GiB。
- 为防止操作失误导致数据丢失，建议您操作前使用快照一致性组备份数据。具体操作，请参见创建快照一致性组。

操作步骤

1. 以root权限远程连接ECS实例。

   关于连接方式的介绍，请参见连接方式概述ECS远程连接操作指南。

2. 使用以下命令查看ECS实例中已经创建的逻辑卷LV信息。

   ```
   lvdisplay
   ```

   结果如下所示，表示已经创建了/dev/lvm_01/lv01逻辑卷，拥有150 GiB空间。

   ```
   [root@ecs ~]# lvdisplay
     --- Logical volume ---
     LV Path                /dev/lvm_01/lv01
     LV Name                lv01
     VG Name                lvm_01
     LV UUID                3dP9u0-6htd-PPYW-qlfZ-p8K[redacted]
     LV Write Access        read/write
     LV Creation host, time ecs, 2021-06-03 11:37:55 +0800
     LV Status              available
     # open                 1
     LV Size                150.00 GiB
     Current LE             38400
     Segments               4
     Allocation             inherit
     Read ahead sectors     auto
     - currently set to     8192
     Block device           252:0
   ```

3. 使用以下命令扩容物理卷PV（Physical Volume）。

```
pvresize <物理卷名称>
```

以下示例为扩容物理卷（/dev/vdf），您需要根据实际情况修改物理卷名称。

```
pvresize /dev/vdf
```

执行结果如下。

```
[root@ecs ~]# pvresize /dev/vdf
  Physical volume "/dev/vdf" changed
  1 physical volume(s) resized or updated / 0 physical volume(s) not resized
```

4. 使用以下命令查看物理卷（PV）使用情况。

```
pvs
```

结果如下所示，表示物理卷/dev/vdf已有80 GiB（原有40 GiB空间，云盘扩容40 GiB）待分配空间。

```
[root@ecs ~]# pvs
  PV         VG     Fmt  Attr PSize   PFree
  /dev/vdb   lvm_01 lvm2 a--  <40.00g      0
  /dev/vdc   lvm_01 lvm2 a--  <40.00g      0
  /dev/vdd   lvm_01 lvm2 a--  <40.00g      0
  /dev/vde   lvm_01 lvm2 a--  <40.00g   9.98g
  /dev/vdf   lvm_01 lvm2 a--  <80.00g <80.00g
```

5. 使用以下命令扩容逻辑卷。

```
lvextend [-L <逻辑卷大小>] <逻辑卷名称>
```

以下示例为扩容逻辑卷容量。

```
lvextend -L +80G /dev/lvm_01/lv01
```

本示例中变量说明如下，您需要根据实际情况修改。

- `+80G`：增减容量，卷组VG（Volume Group）必须有剩余容量时才可以执行扩容逻辑卷操作。
- `/dev/lvm_01/lv01`：逻辑卷名称。

结果如下所示，表示您为逻辑卷/dev/lvm_01/lv01扩容了80 GiB物理空间。

```
[root@ecs ~]# lvextend -L +80G /dev/lvm_01/lv01
  Size of logical volume lvm_01/lv01 changed from 150.00 GiB (38400 extents) to 230.00 GiB (58880 extents).
  Logical volume lvm_01/lv01 successfully resized.
```

6. 使用以下命令扩容逻辑卷文件系统。

您需要根据逻辑卷的文件系统类型执行不同的扩容命令，以下以ext4和xfs文件系统为例：

> **说明** 如果您不清楚逻辑卷的文件系统类型，可以通过 `df -Th` 命令查询。

- 如果是ext4文件系统，使用以下命令扩容。

```
resize2fs /dev/lvm_01/lv01
```

- 如果是xfs文件系统，使用以下命令扩容。

```
xfs_growfs /dev/lvm_01/lv01
```

7. 使用以下命令查看文件系统扩容结果。

```
df -h
```

结果如下所示，显示逻辑卷的总容量为230 GiB，表示扩容成功。

```
[root@ecs ~]# df -h
Filesystem               Size  Used Avail Use% Mounted on
/dev/vda1                 40G  2.0G   36G   6% /
devtmpfs                 3.8G     0  3.8G   0% /dev
tmpfs                    3.8G     0  3.8G   0% /dev/shm
tmpfs                    3.8G  488K  3.8G   1% /run
tmpfs                    3.8G     0  3.8G   0% /sys/fs/cgroup
tmpfs                    768M     0  768M   0% /run/user/0
/dev/mapper/lvm_01-lv01  230G   33M  230G   1% /media/lv01
```

相关文档

- 通过LVM创建逻辑卷

11.4. 创建RAID阵列（Linux）

本文以Ubuntu系统ECS实例为例，介绍了如何使用Linux系统内置的mdadm命令为多块数据盘创建一个200 GiB的RAID阵列。

前提条件

您已经创建并挂载了多块云盘。建议您创建具有相同容量和相同类型的云盘。创建并挂载云盘的具体操作，请参见创建云盘和挂载数据盘。

背景信息

独立冗余磁盘阵列（Redundant Array of Independent Disks，简称RAID）是将多块云盘按一定的方式组成一个磁盘阵列组。相比单块云盘，RAID能够有效的提高磁盘的容量、读写带宽、可靠性和可用性。

建议您使用RAID0或者RAID1模式，每块云盘采用相同大小的分区，从而减少云盘空间浪费。由于RAID5或者RAID6模式的奇偶校验数据会占用云盘IOPS，带来性能阻碍，因此不推荐使用RAID5或者RAID6模式。

下表对比了RAID0和RAID1模式的优缺点以及适用场景。

模式	优势	劣势	适用场景
RAID0	I/O在存储卷内以条带化的方式分布在各云盘上。增加云盘空间会直接增加吞吐量，阵列中的容量和带宽等于各个云盘容量和带宽之和。	单块云盘的损坏有可能造成整个虚拟盘数据丢失，缺乏数据冗余能力。	对I/O性能要求很高，并且已通过其他方式备份数据，或者不需要备份数据的应用。
RAID1	数据以镜像的方式存储在各云盘上，可以获取更高的数据冗余性。阵列中的容量和带宽等于阵列中容量和带宽最小的云盘。	因为要同时向多块云盘写入数据，写性能较差。	容错能力比 I/O 性能更重要，例如在关键应用程序中。

操作步骤

1. 以root权限远程连接ECS实例。

   关于连接方式的介绍，请参见连接方式概述ECS远程连接操作指南。

2. 运行以下命令查看ECS实例上所有云盘信息。

   ```
   lsblk
   ```

结果如下所示。

```
root@ecs:~# lsblk
NAME    MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
vda     253:0    0  40G  0 disk
└─vda1 253:1    0  40G  0 part /
vdb     253:16   0  40G  0 disk
vdc     253:32   0  40G  0 disk
vdd     253:48   0  40G  0 disk
vde     253:64   0  40G  0 disk
vdf     253:80   0  40G  0 disk
```

3. 使用**mdadm**命令创建RAID阵列*/dev/md0*。

   请根据您的实际情况，创建RAID0或RAID1模式。

   > **说明**
   > - 以下示例中 `/dev/vd[bcdef]` 表示为/dev/vdb、/dev/vdc、/dev/vdd、/dev/vde和/dev/vdf五块云盘组成RAID阵列。如果您使用其他云盘，需要修改成对应的云盘。
   > - 如果提示未安装mdadm，请先运行 `apt-get install mdadm` 命令安装mdadm工具。

   - 创建RAID0模式，运行以下命令。

     ```
     mdadm --create /dev/md0 --level=0 --raid-devices=5 /dev/vd[bcdef]
     ```

     - `--level=0` ：表示用于将阵列条带化的RAID0模式。
     - `--raid-devices=5` ：表示RAID阵列由五块云盘组成。
     - `/dev/vd[bcdef]` ：表示使用/dev/vdb、/dev/vdc、/dev/vdd、/dev/vde和/dev/vdf五块云盘组成一个RAID阵列。

     结果如下所示。

     ```
     root@ecs:~# mdadm --create /dev/md0 --level=0 --raid-devices=5 /dev/vd[bcdef]
     mdadm: Defaulting to version 1.2 metadata
     mdadm: array /dev/md0 started.
     ```

   - 创建RAID1模式，运行以下命令。

     ```
     mdadm --create /dev/md0 --level=1 --raid-devices=5 /dev/vd[bcdef]
     ```

     - `--level=1` 表示用于将阵列镜像化的RAID1模式。
     - `--raid-devices=5` ：表示RAID阵列由五块云盘组成。
     - `/dev/vd[bcdef]` ：表示使用/dev/vdb、/dev/vdc、/dev/vdd、/dev/vde和/dev/vdf五块云盘组成一个RAID阵列。

4. 运行以下命令查看创建的RAID阵列/dev/md0信息。

   ```
   mdadm --detail /dev/md0
   ```

结果如下所示。

```
root@ecs:~# mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Sat May  8 15:10:52 2021
     Raid Level : raid0
     Array Size : 209551360 (199.84 GiB 214.58 GB)
   Raid Devices : 5
  Total Devices : 5
    Persistence : Superblock is persistent

    Update Time : Sat May  8 15:10:52 2021
          State : clean
 Active Devices : 5
Working Devices : 5
 Failed Devices : 0
  Spare Devices : 0

     Chunk Size : 512K

           Name : ecs:0  (local to host ecs)
           UUID : 09873fbc:5172dd8[redacted]
         Events : 0

    Number   Major   Minor   RaidDevice State
       0     253       16        0      active sync   /dev/vdb
       1     253       32        1      active sync   /dev/vdc
       2     253       48        2      active sync   /dev/vdd
       3     253       64        3      active sync   /dev/vde
       4     253       80        4      active sync   /dev/vdf
```

5. 运行以下命令在RAID阵列上创建一个文件系统，例如，ext4文件系统。

   您也可以创建其他类型的文件系统。

   ```
   mkfs.ext4 /dev/md0
   ```

   结果如下所示。

   ```
   root@ecs:~# mkfs.ext4 /dev/md0
   mke2fs 1.42.13 (17-May-2015)
   Creating filesystem with 52387840 4k blocks and 13099008 inodes
   Filesystem UUID: 9d924ab0-480a-4482-a[redacted]
   Superblock backups stored on blocks:
           32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
           4096000, 7962624, 11239424, 20480000, 23887872

   Allocating group tables: done
   Writing inode tables: done
   Creating journal (32768 blocks): done
   Writing superblocks and filesystem accounting information: done
   ```

6. 运行以下命令，创建一份包含RAID信息的配置文件，设置RAID阵列在启动ECS实例时自动重组。

   ```
   sudo mdadm --detail --scan | sudo tee -a /etc/mdadm/mdadm.conf
   ```

7. 挂载RAID阵列的文件系统。

   i. 运行以下命令，创建挂载点，例如*/media/raid0*。

   ```
   mkdir /media/raid0
   ```

> **说明** 您也可以将云盘挂载到已有目录下，例如*/mnt*。

ii. 运行以下命令挂载文件系统，例如将/dev/md0挂载至/media/rad0。

```
mount /dev/md0 /media/raid0
```

8. 运行以下命令查看RAID阵列的挂载信息。

```
df -h
```

结果如下所示，返回信息中，文件系统已经挂载到指定的挂载点。

```
root@ecs:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            1.9G     0  1.9G   0% /dev
tmpfs           381M  2.9M  378M   1% /run
/dev/vda1        40G  2.4G   36G   7% /
tmpfs           1.9G     0  1.9G   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           1.9G     0  1.9G   0% /sys/fs/cgroup
tmpfs           381M     0  381M   0% /run/user/0
/dev/md0        197G   60M  187G   1% /media/raid0
```

后续步骤

如果您需要在每次启动ECS实例时设置自动加载RAID阵列，可以在*/etc/fstab*配置文件中添加如下信息。

1. 运行以下命令，向*/etc/fstab*配置文件写入自启动设置。

```
echo `blkid /dev/md0 | awk '{print $2}' | sed 's/\"//g'` /media/raid0 ext4 defaults 0 0
>> /etc/fstab
```

- `/dev/md0`：磁盘阵列名称。
- `/media/raid0`：挂载点信息，如果需要挂载到其他路径，您需要修改成对应路径。

> **说明** 如果您需要在未挂载RAID阵列的情况下启动ECS实例，可以添加nofail配置。即使在安装云盘时出现错误，nofail配置也允许启动ECS实例。如果您使用的是Ubuntu系统，还需要额外添加nobootwait配置。

2. 运行以下命令挂载*/etc/fstab*配置文件中的所有文件系统。

```
mount -a
```

11.5. 修改云盘的UUID

如果您使用快照创建云盘后挂载到原Linux实例，新创建云盘的UUID会和原云盘冲突。本文介绍如何修改新云盘的UUID。

背景信息

使用快照创建云盘后，新创建的云盘的UUID和原云盘是一样的。如果您将新创建的云盘挂载到原来的Linux实例，此时会导致UUID冲突，存在以下问题：

- 如果您使用系统盘快照创建一个新云盘，挂载到原Linux实例。Linux可能不是从系统盘启动，而是从新挂载的数据盘启动。

- 如果您的云盘使用xfs文件系统，会因为UUID冲突禁止挂载（mount），提示 `“mount: wrong fs type, bad option, bad superblock on /dev/vdd1,”` 。

因此，您在使用快照创建新云盘并在控制台挂载到原Linux实例后，需要登录实例修改新云盘的UUID，再执行挂载（mount）操作。关于如何修改云盘的UUID，您可以通过 `blkid` 命令查询文件系统类型，并根据查询结果选择合适的操作：

- 如果查询结果为 `TYPE="ext4"` 、 `TYPE="ext3"` 或 `TYPE="ext2"` ，具体操作，请参见修改ext2、ext3或ext4文件系统的UUID。
- 如果查询结果为 `TYPE="xfs"` ，具体操作，请参见修改xfs文件系统的UUID。

修改ext2、ext3或ext4文件系统的UUID

> **说明** 本示例以/dev/vdb1为例，您需要根据自己的设备名修改相关命令。

1. 远程连接ECS实例。

   具体操作，请参见通过密码认证登录Linux实例。

2. 运行以下命令查询云盘的UUID。

   ```
   blkid
   ```

   查询结果如下所示，此时通过快照新创建云盘的UUID和原云盘一样。

   ```
   [root@ecs ~]# blkid
   /dev/vda1: UUID="dcbdbcd3-f78c-4739-8cc7-      50da3b" TYPE="ext4"
   /dev/vdb1: UUID="dcbdbcd3-f78c-4739-8cc7-      50da3b" TYPE="ext4"
   [root@ecs ~]#
   ```

3. 运行以下命令检查文件系统。

   ```
   e2fsck -f /dev/vdb1
   ```

4. 运行以下命令为云盘生成新的UUID。

   ```
   uuidgen | xargs tune2fs /dev/vdb1 -U
   ```

5. 运行以下命令查看是否已经修改UUID。

   ```
   blkid
   ```

   查询结果如下，表示已经修改/dev/vdb1的UUID。

   ```
   [root@ecs ~]# blkid
   /dev/vda1: UUID="dcbdbcd3-f78c-4739-8cc7-      50da3b" TYPE="ext4"
   /dev/vdb1: UUID="56570712-2c72-42c0-9b13-      7cae0b" TYPE="ext4"
   ```

6. 运行以下命令挂载（mount）云盘。

   ```
   mount /dev/vdb1 /mnt
   ```

7. 配置*/etc/fstab*文件，开机自动挂载新云盘。

   关于如何配置*/etc/fstab*文件，请参见在fstab文件中配置UUID方式自动挂载数据盘。

修改xfs文件系统的UUID

> **说明** 本示例以/dev/vdd1为例，您需要根据自己的设备名修改相关命令。

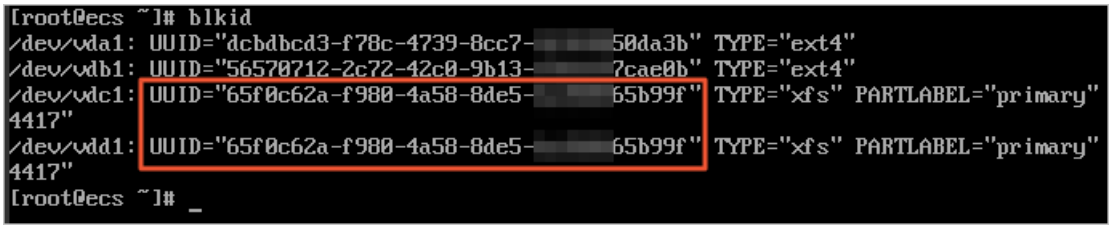
1. 远程连接ECS实例。

   具体操作，请参见通过密码认证登录Linux实例。

2. 运行以下命令查询云盘的UUID。

   ```
   blkid
   ```

   查询结果如下所示，此时通过快照新创建云盘的UUID和原云盘一样。

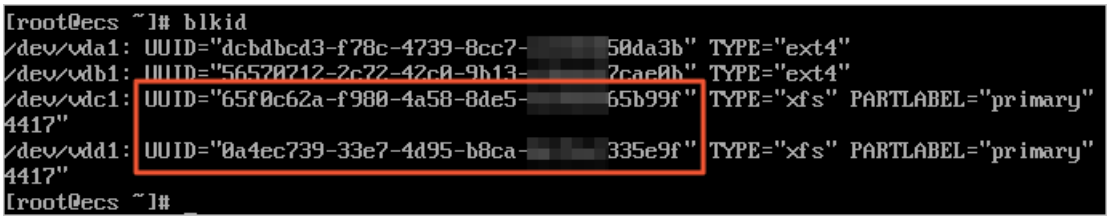
3. 运行以下命令为云盘生成新的UUID。

   ```
   xfs_admin -U generate /dev/vdd1
   ```

4. 运行以下命令查看是否已经修改UUID。

   ```
   blkid
   ```

   查询结果如下，表示已经修改/dev/vdd1的UUID。

5. 运行以下命令挂载（mount）云盘。

   ```
   mount /dev/vdd1 /mnt
   ```

6. 配置*/etc/fstab*文件，开机自动挂载新云盘。

   关于如何配置*/etc/fstab*文件，请参见在fstab文件中配置UUID方式自动挂载数据盘。

相关文档

- 使用快照创建云盘
- 在fstab文件中配置UUID方式自动挂载数据盘
- Linux系统中xfs类型分区在挂在时提示“mount: wrong fs type, bad option, bad superblock on /dev/vdd1,”

11.6. 在fstab文件中配置UUID方式自动挂载数据盘

在Linux系统中，您可以通过配置fstab文件让ECS启动时会自动挂载数据盘的文件系统。但是，如果fstab文件配置不当，那么您的云盘的挂载顺序变更后，可能会导致ECS重启后不能正常运行。本文介绍如何在fstab文件中配置UUID方式自动挂载数据盘的文件系统，可以解决此类重启异常问题。

前提条件

挂载到实例的云盘已经进行分区格式化。具体操作，请参见分区格式化数据盘（Linux）。

背景信息

fstab支持使用云盘分区名（例如/dev/vdb1）或UUID标识文件系统，两者的差异如下所示：

- 在fstab中使用云盘分区名标识文件系统，如果云盘的挂载顺序变更，云盘分区可能不会被正确的挂载（mount）到原来的挂载点。这种情况下可能会影响您ECS上运行的应用。
- 在fstab中使用UUID标识文件系统，如果云盘的挂载顺序变更，云盘分区仍然可以正确的挂载（mount）到原来的挂载点。因此，本文建议使用UUID标识文件系统。

操作步骤

1. 远程连接ECS实例。

   关于如何远程连接ECS实例，请参见通过密码或密钥认证登录Linux实例。

2. 运行以下命令查看实例的云盘信息。

   ```
   fdisk -lu
   ```

   运行结果如下所示。

   ```
   [root@ecs ~]# fdisk -lu

   Disk /dev/vda: 42.9 GB, 42949672960 bytes, 83886080 sectors
   Units = sectors of 1 * 512 = 512 bytes
   Sector size (logical/physical): 512 bytes / 512 bytes
   I/O size (minimum/optimal): 512 bytes / 512 bytes
   Disk label type: dos
   Disk identifier: 0x000c2bef

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

   Disk /dev/vdb: 42.9 GB, 42949672960 bytes, 83886080 sectors
   Units = sectors of 1 * 512 = 512 bytes
   Sector size (logical/physical): 512 bytes / 512 bytes
   I/O size (minimum/optimal): 512 bytes / 512 bytes
   Disk label type: dos
   Disk identifier: 0x41a5a16d

      Device Boot      Start         End      Blocks   Id  System
   /dev/vdb1            2048    83886079    41942016   83  Linux

   Disk /dev/vdc: 42.9 GB, 42949672960 bytes, 83886080 sectors
   Units = sectors of 1 * 512 = 512 bytes
   Sector size (logical/physical): 512 bytes / 512 bytes
   I/O size (minimum/optimal): 512 bytes / 512 bytes
   Disk label type: dos
   Disk identifier: 0x93f147d9

      Device Boot      Start         End      Blocks   Id  System
   /dev/vdc1            2048    83886079    41942016   83  Linux
   [root@ecs ~]#
   ```

3. 运行以下命令查询云盘的UUID信息。

   ```
   blkid
   ```

   运行结果如下所示。

   ```
   [root@ecs ~]# blkid
   /dev/vda1: UUID="dcbdbcd3-f78c-4739-8cc7-fd9498[redacted]" TYPE="ext4"
   /dev/vdb1: UUID="59f23670-94c1-42d1-8bb0-209d78[redacted]" TYPE="ext4"
   /dev/vdc1: UUID="88619b1a-d971-41c2-91d0-3a440f[redacted]" TYPE="xfs"
   [root@ecs ~]#
   ```

4. 运行以下命令分别创建数据盘的挂载点。
   - 创建/dev/vdb1的挂载点*/test01*：

     ```
     mkdir /test01
     ```

   - 创建/dev/vdc1的挂载点*/test02*：

     ```
     mkdir /test02
     ```

5. 在fstab文件中添加挂载信息。
   i. 运行以下命令编辑fstab。

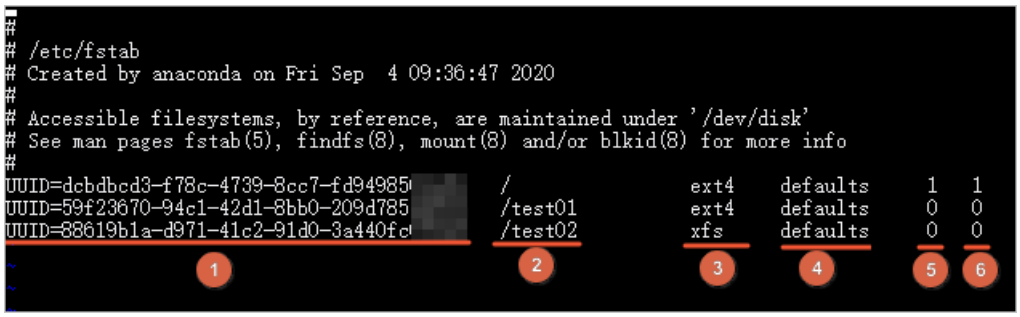
   ```
   vi /etc/fstab
   ```

   ii. 按 `i` 键进入编辑模式。

   iii. 新增以下挂载信息。

   ```
   UUID=59f23670-94c1-42d1-8bb0-209d7854****   /test01     ext4    defaults     0   0
   UUID=88619b1a-d971-41c2-91d0-3a440fc0****   /test02     xfs     defaults     0   0
   ```

   结果如下所示。



序号	字段	说明
①	<file system>	要挂载分区的文件系统。 此处建议使用UUID，可以使用 `blkid` 命令查询分区文件系统的UUID。
②	<dir>	文件系统的挂载位置。 您可以自己创建新的挂载位置，例如本文中的*/test01*和*/test02*。
③	<type>	要挂载分区的文件系统类型。 您可以使用 `blkid` 命令查询分区的文件系统类型。
④	<options>	挂载时使用的参数，一般情况下使用defaults参数。如果需要使用多个参数，通过英文逗号（,）分隔，例如 `defaults,noatime` 。 对于<options>参数的更多信息，请参见fstab说明。

序号	字段	说明
⑤	<dump>	dump工具是否对这个文件系统进行备份。 ■ 0：表示忽略。 ■ 1：表示进行备份。 一般情况下没有使用dump工具，可以设置为0。
⑥	<pass>	fsck检查文件系统的优先级。 ■ 0：表示不检查文件系统。 ■ 1：如果需要检查，根目录（/）对应的文件系统设置为1。 ■ 2：如果需要检查，非根目录对应的其它文件系统设置为2。 一般情况下，可以设置为0。

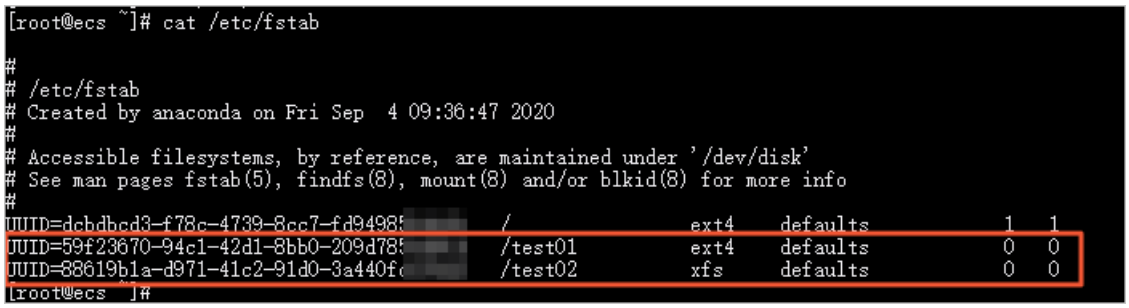
iv. 修改完成后，按 `Esc` 键退出编辑模式。

v. 输入 `:wq` 后，按 `Enter` 键保存并退出。

6. 运行以下命令查看fstab文件。

```
cat /etc/fstab
```

执行结果如下所示。



7. 运行以下命令挂载数据盘分区的文件系统。
   - 挂载/dev/vdb1：

     ```
     mount /dev/vdb1 /test01
     ```

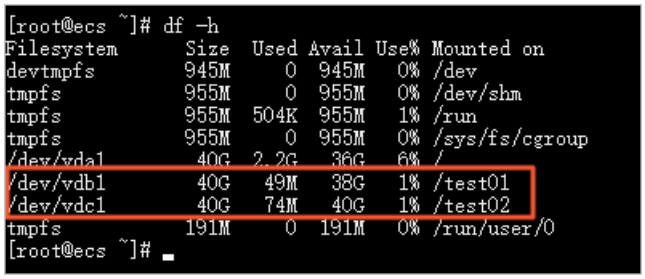
   - 挂载/dev/vdc1：

     ```
     mount /dev/vdc1 /test02
     ```

8. 运行以下命令检查挂载结果。

```
df -h
```

执行结果如下所示。



配置完成后，您后续如果重启ECS实例，系统将自动挂载数据盘。

常见问题

Linux实例的/etc/fstab文件配置错误导致系统启动异常

11.7. 云盘缩容

由于目前云服务器ECS不支持系统盘或者数据盘缩容，如果您有云盘缩容的需求，可以通过阿里云服务器迁移中心SMC达成目的。

前提条件

请确保您已完成迁移前的准备工作，更多信息，请参见准备工作（迁移前必读）。

背景信息

SMC的研发初衷是为了平衡阿里云用户的云上及线下业务负载，但是您也可以利用其工作原理，实现云服务器ECS的云盘缩容。

SMC可以根据您的ECS实例重新创建一份自定义镜像或者直接迁移至目标实例，在导入迁移源时，通过重新指定云盘大小进行缩容。开始迁移前，您必须注意：

- 如果您使用迁移源迁移至目标实例的方式完成云盘缩容，请确保迁移源与目标实例不是同一台实例。您必须迁移至除迁移源的其它可用ECS实例，并且该目标实例内无数据或已将所有数据备份至镜像、快照或该实例以外的设备中。

  > **警告** 创建迁移任务后，目标实例中的原数据将会清除。因此，如果作为目标实例的ECS实例中存在重要数据，SMC不推荐您使用该方式将迁移源直接迁移至目标实例。推荐您在迁移时，目标类型选择云服务器镜像，再通过镜像创建ECS实例。

- 由于使用SMC进行云盘缩容将更换ECS实例，因此会引起ECS实例的部分属性发生变化。例如：实例ID（`InstanceId`）和公网IP。

  如果您的迁移源实例为专有网络VPC类型的实例，可以将公网IP转换为弹性公网IP以保留该公网IP。因此，如果您已使用弹性公网IP（EIP）或者对公网IP依赖程度较轻，建议您使用SMC完成云盘缩容。

操作步骤

1. 导入迁移源。

   您需要先在待缩容的ECS实例内下载SMC客户端，并通过SMC客户端将该实例作为迁移源导入SMC。具体操作，请参见步骤一：导入迁移源。

2. 创建并启动迁移任务。

   迁移至指定目标实例完成云盘缩容的具体操作，请参见源服务器迁移至目标实例。创建迁移任务时您需要注意：

- **目标磁盘**中的**系统盘**和**数据盘**大小，设置为您需要缩容的预期值，该值不能小于系统盘或数据盘实际使用空间的大小。
- 如果您需要通过迁移源创建自定义镜像，再通过自定义镜像创建新的ECS实例，请在创建迁移任务时，将**目标类型**设置为**云服务器镜像**。

3. 等待迁移任务完成。
   - 当迁移任务状态为**已完成**（Finished），表示任务完成并能够查看目标实例。
     -
     -
   - 当任务状态为**出错**（InError），表示任务失败。您需要查看日志修复问题后，再次重启迁移任务。常见错误及修复方案，请参见SMC FAQ。

11.8. ECS数据加密的应用

数据加密适用于数据安全或法规合规等场景，帮助您加密保护存储在阿里云ECS上的数据，您可以选择对系统盘、数据盘或者镜像进行加密，然后基于加密后的云盘和镜像去创建ECS实例，以保护数据的隐私性和安全性。本文主要为您介绍加密云盘、快照和镜像的一些限制条件和相关操作。

前提条件

您已开通当前地域的密钥管理服务KMS（Key Management Service）。具体操作，请参见开通密钥管理服务。

背景信息

加密功能默认使用服务密钥（Default Service CMK）为用户数据进行加密，也支持使用自定义密钥BYOK（Bring Your Own Key）为用户的数据进行加密。云盘的加密机制中，每一块云盘会有相对应的用户主密钥CMK（Customer Master Key）和数据密钥DK（Data Key），并通过信封加密（Envelope Encryption）机制对用户数据进行加密。更多信息，请参见加密概述。

使用密钥加密时，请仔细阅读以下注意事项：

限制项	说明
服务密钥	每个地域每个用户的服务密钥（Default Service CMK）唯一，不支持删除和禁用操作。

限制项	说明
自定义密钥BYOK	• 在ECS控制台上首次选择自定义密钥（BYOK）加密云盘时，需单击**去授权**，根据页面引导为ECS授权 `AliyunECSDiskEncryptDefaultRole` 角色，允许ECS访问您的KMS资源。 关于角色的更多信息，请参见访问控制RAM介绍。 • 在KMS控制台创建密钥时，需选择Aliyun_AES_256或Aliyun_SM4密钥类型，ECS创建加密云盘时暂不支持其他密钥类型。 • 用户删除、禁用BYOK密钥前，需要确认卸载或更换该密钥关联的云盘，避免出现云盘数据丢失、实例启动失败等问题。查询密钥关联的云盘信息，请参见API DescribeDisks。 因BYOK密钥一旦删除将无法恢复，使用该密钥加密的内容及产生的数据密钥也将无法解密。在密钥失效前，推荐您使用禁用密钥功能，或者自行排查该密钥是否存在关联使用的云资源，避免密钥丢失后数据不可恢复。 **注意** 因用户手动创建的BYOK密钥可被用户进行删除、禁用等操作导致密钥失效，当密钥失效后会存在已创建的加密云盘、加密镜像、加密快照数据不可恢复的风险。 声明：由用户自行操作密钥失效后导致关联的云盘资源相关数据丢失后不可恢复的风险，由用户自行承担责任。

加密系统盘

系统盘是装有操作系统的云盘，只能随实例创建，生命周期与挂载的ECS实例相同，您可以在创建实例时加密系统盘。

说明 创建实例时加密系统盘功能正在公测中，公测地域和可用区仅支持中国（香港）的B、C可用区、新加坡的B、C可用区。

使用须知

限制项	说明
实例规格族	不包括ecs.ebmg5、ecs.ebmgn5t、ecs.ebmi3、ecs.sccg5、ecs.scch5、ecs.ebmc4和ecs.ebmhfg5。更多信息，请参见实例规格族。
镜像	仅支持公共镜像和自定义镜像，不支持共享镜像和市场镜像。
云盘类型	仅支持ESSD云盘类型。

操作步骤

加密系统盘的具体操作，请参见加密系统盘。

加密数据盘

加密数据盘后，数据盘上的动态数据传输以及静态数据都会被加密。您可以在创建实例时加密数据盘，也可以在创建云盘时加密数据盘。

使用须知

加密数据盘时，如果选择**用快照创建磁盘**，必须满足以下条件才能选择**加密**选项为数据盘加密。

> **说明** 该功能限制正在公测中，公测地域和可用区仅支持中国（香港）的B、C可用区、新加坡的B、C可用区。

限制项	说明
实例规格族	不包括ecs.ebmg5、ecs.ebmgn5t、ecs.ebmi3、ecs.sccg5、ecs.scch5、ecs.ebmc4和ecs.ebmhfg5。更多信息，请参见实例规格族。
镜像	仅支持公共镜像和自定义镜像，不支持共享镜像和市场镜像。
云盘类型	仅支持ESSD云盘类型。

操作步骤

加密数据盘的具体操作，请参见加密数据盘。

加密快照

如果云盘是加密云盘，使用该云盘创建的快照也是加密快照。

操作步骤

创建快照的具体操作，请参见创建一个云盘快照。

共享加密镜像

共享镜像可用于跨账号部署ECS实例。如果ECS实例挂载的云盘开启了加密功能，则通过该ECS实例所创建的自定义镜像为加密镜像，此时您可以将加密后的自定义镜像共享给其他阿里云账号使用。该账号可以使用您共享的加密自定义镜像，快速创建并运行同一镜像环境的ECS实例。

> **说明** 共享加密自定义镜像的功能目前支持华北2（北京）、华东2（上海）、中国（香港）、新加坡地域。

使用须知

限制项	说明

<table>
<tr><th>限制项</th><th>说明</th></tr>
<tr><td>角色授权</td><td>加密自定义镜像在被共享之前，您需要通过阿里云访问控制创建 <code>AliyunECSShareEncryptImageDefaultRole</code> 角色并进行授权。授权的具体操作，请参见如何共享加密自定义镜像？。
为 <code>AliyunECSShareEncryptImageDefaultRole</code> 角色授权时建议遵循最小权限的安全原则，只授予镜像相关的密钥权限。
角色授权代码示例：
<pre>{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "kms:List*"
      ],
      "Resource": [
        "acs:kms:cn-beijing:xxx:key",
        "acs:kms:cn-shanghai:xxx:key"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "kms:DescribeKey",
        "kms:TagResource",
        "kms:UntagResource"
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "acs:kms:cn-beijing:xxx:key/yyy",
        "acs:kms:cn-shanghai:xxx:key/yyy"
      ],
      "Effect": "Allow"
    }
  ]
}</pre></td></tr>
<tr><td>密钥使用建议</td><td>建议用户为共享加密镜像创建独立的BYOK密钥，避免KMS密钥泄露导致安全风险。</td></tr>
<tr><td>取消共享加密镜像</td><td>• 被共享者无法通过ECS管理控制台或ECS API查询到该镜像。
• 被共享者无法使用该镜像创建实例和更换系统盘。
• 被共享者使用共享镜像创建的实例无法重新初始化云盘。</td></tr>
</table>

限制项	说明
禁用加密镜像	• 被共享者无法使用该镜像创建实例和更换系统盘。 • 被共享者使用共享镜像创建的实例无法重新初始化云盘。

操作步骤

共享加密镜像的具体操作，请参见：

1. 如何共享加密自定义镜像？
2. 共享或取消共享镜像

12.本地盘最佳实践

本地盘是ECS实例所在物理机上的本地硬盘设备。相比云盘，本地盘具有较高的存储I/O性能，但同时也有更大的数据风险。本文介绍如何正确选择本地盘，以及如何降低本地盘数据风险。

什么是本地盘

本地盘是ECS实例所在物理机上的本地硬盘设备，能够为ECS实例提供本地存储访问能力，具有低时延、高随机IOPS、高吞吐量和高性价比的优势。对存储I/O性能有极高的要求，并且具备应用层高可用架构的业务，适合选择本地盘实例。了解更多本地盘信息，请参见本地盘。

本地盘的风险

本地盘只挂载在单台物理机上，而不具备分布式的多副本机制，其数据可靠性取决于物理机的可靠性。如果本地盘发生故障、物理服务器发生宕机，或者人为误操作，本地盘会丢失数据。请勿在本地盘上存储需要长期保存的业务数据。

但云盘采用分布式三副本机制，能防止意外硬件故障导致的数据不可用。如果应用没有多节点数据冗余架构，强烈建议您选择云盘。

最佳实践

- 选型
- 备份本地盘
- 使用部署集提高可用性
- 本地盘数据迁移到云盘
- 本地盘发生损坏后的处理

选型

对于大数据、重型数据库应用，带本地盘的实例（例如i2、d1等）在成本、存储访问时延上有着较大的优势。如果您业务场景对存储I/O性能有极高要求，并且应用层具备高可用架构，可以购买本地盘实例。此外，如果您有大数据集群，建议使用部署集将实例分散部署在不同的物理服务器上，从而降低某一本地盘损坏带来的影响。

如果您的应用没有高可用架构，建议您使用其他实例。实例的详细参数请参见实例规格族。最佳实践请参见选型最佳实践。

备份本地盘

如果已经使用了本地盘，并且应用层没有数据可靠性的架构设计，强烈建议您做好数据备份。您可以按照以下方式备份本地盘数据：

- 方式一：使用混合云备份服务HBR（Hybrid Backup Recovery）

  HBR是一种高效、安全、低成本的全托管式云备份存储服务。您可以使用HBR将企业数据中心的数据、分支机构数据，或云上资源备份到HBR的云上备份仓库。详情请参见什么是混合云备份。

- 方式二：搭建冗余架构

  搭建多节点冗余数据架构，降低本盘故障带来的影响。您可以购买云盘，并将本盘数据实时拷贝到云盘；或者，在其他可用区或地域购买ECS实例并部署应用作为灾备。

使用部署集提高可用性

为保证数据的可用性，建议您在应用层做数据冗余。

您可以使用部署集将业务涉及到的几台ECS实例分散部署在不同的物理服务器上，保证业务的高可用性和底层容灾能力。详情请参见创建部署集。

本地盘数据迁移到云盘

如果您已经购买了带本地盘的实例，可以将该实例变更为带云盘的实例。云盘采用分布式三副本机制，能防止意外硬件故障导致的数据不可用。云盘还能够随时创建快照来备份数据。

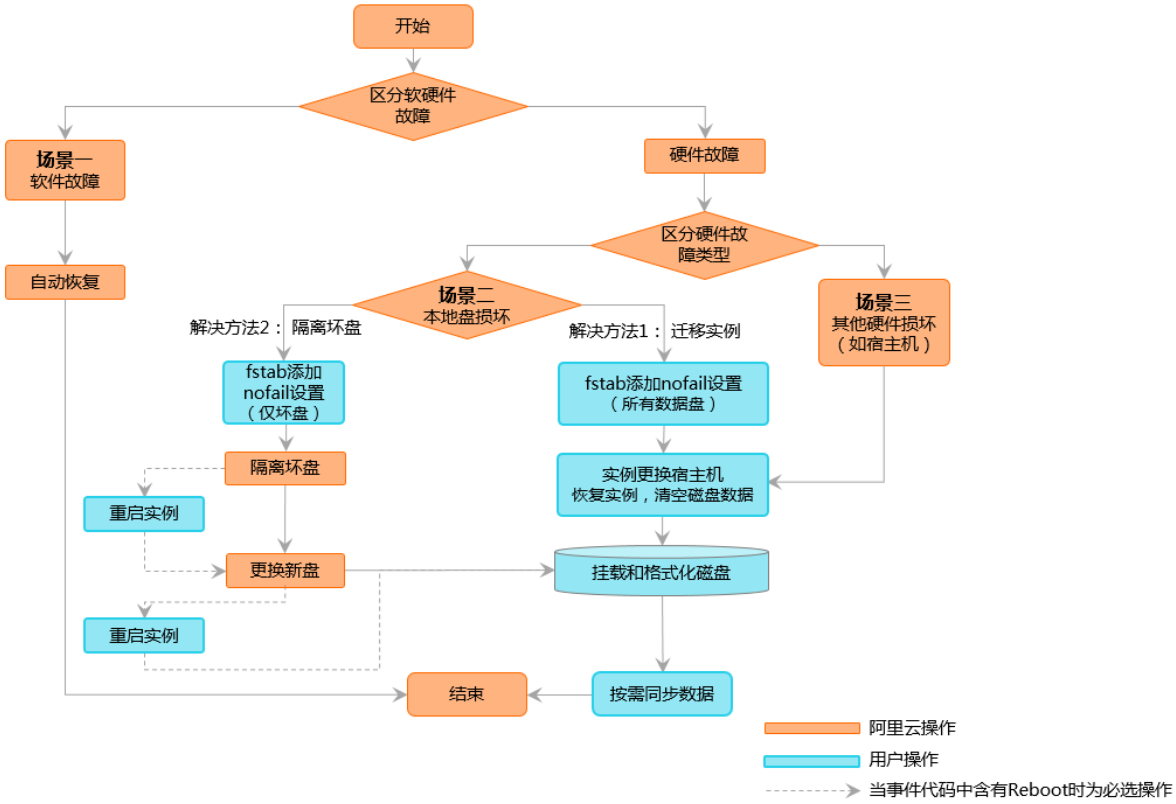
使用服务器迁移中心SMC（Server Migration Center），能将本地盘实例的数据完整备份。SMC可将单台或多台本地盘实例整体数据一键迁移到阿里云，生成镜像备份。详情请参见什么是服务器迁移中心。

在迁移过程中，请注意：

- SMC是免费服务，但迁移过程中使用ECS资源会产生少量费用。
- 如果您想保留原公网IP，可以将源实例公网IP转换为弹性公网IP（即EIP），然后将EIP与实例解绑。在创建新的目标实例后，将EIP绑定到目标实例即可。
- 如果您想保留原私有IP，在创建新的目标实例时，必须指定和源实例相同网段的VPC与VSwitch。通过ECS控制台创建新实例后，在控制台将私有IP修改成和源实例一致；或者，通过API创建实例，直接指定内网IP，同时必须指定和源实例相同网段的VPC与VSwitch。
- 建议选择**内网传输**的网络模式，并启用块复制，最大化提高迁移备份效率。
- 建议开启**自动增量同步**，定期使用SMC迁移生成备份镜像。
- SMC迁移时不会干涉原系统，不会修改原系统配置或文件，除了会占用一定的CPU/内存、带宽资源，其他不影响原系统业务。如果对数据库之类的应用要保证最后数据一致性，建议暂停服务后再进行SMC迁移备份。

本地盘发生损坏后的处理

如果本地盘发生损坏，阿里云会触发系统事件，并及时给您发送通知、应对措施和事件周期等信息。您可以根据场景来运维，如下图所示。详情请参见本地盘实例系统事件。



13.标签设计最佳实践

随着您云上资源的增加，管理难度也随之变化，您可以通过标签实现批量管理资源。标签是人员、财务、物品管理的重要分组工具，帮助您横向管理多款云产品。

应用场景

标签的常见场景包括资源管理、访问控制和自动化运维及分账等，如下所示：

- 管理应用发布流程
- 资源溯源，基于标签分组检索和管理资源
- 搭配运维编排服务、资源编排、弹性伸缩和云助手等实现基于标签自动化分组运维
- 基于标签管理成本和分账
- 设计资源或角色访问控制

原则概述

您在创建标签时，可以根据以下设计原则实现标签最佳实践：

- 互斥原则
- 集体详尽原则
- 有限值原则
- 考虑未来变化后果原则
- 简化设计原则

互斥原则

互斥是指尽量避免对同一个属性含义使用两个或以上的标签键。例如标记归属者用 `key="owner"` 表示时，就不能使用其他相同含义的标签键，如*own*、*belonger*或*归属者*等。

集体详尽原则

集体详尽是指规划资源时，您需要同时规划标签，并优先规划标签键。所有资源对象都必须绑定已规划的标签键及其对应的标签键。

- 标签键值对需要采用标准化命名格式。
- 集体详尽原则是后续通过标签维度在访问控制、成本跟踪、自动化运维以及分组搜索的必要条件。

有限值原则

有限值是指为资源剔除多余的标签值，只保留核心标签值。

有限值原则简化了资源管理、访问控制、自动化运维及分账等流程。您还可以结合标签及自动化工具管理资源，云服务器ECS支持通过API编程控制标签，方便您自动管理、检索和筛选资源。

考虑未来变化后果原则

您需要在满足有限值的前提下，在规划标签时同时考虑后续工作中增加或者减少标签值的影响，提高标签修改的灵活性。

当您修改标签时，可能会引起基于标签的访问控制、自动化运维或相关账单报表的变化。无论是公司或个人层面的业务，最佳实践是创建与业务相关的标签组，以便从技术、业务和安全维度管理资源。使用自动化运维来管理其资源及服务时，还设计额外的自动化专用的标签，帮助您完成自动化运维工作。

简化设计原则

简化设计原则是指简化标签键的使用，在规划标签时使用固定维度的标签键。简化设计原则可减少由于过多的标签键导致的操作报错。

- 您可以创建与业务相关的标签组，方便您从技术、业务或安全等维度管理资源。
- 使用自动化运维工具管理资源及服务时，您可以设计自动化运维专用的标签。

标签键设计示例

下表列举了常见业务维度的标签命名示例。涉及英文标签命名时，建议使用小写英文字母。

业务维度	标签键（key）	标签值（value）
组织架构	• company • department • organization • team • group	相关名称
业务架构	• product • business • module • service	相关名称
角色架构	• role • user	• 网络管理员 • 应用管理员 • 系统管理员 • 运维管理员或OpsUser • 研发或DevUser • 测试或TestUser
用途类标签	• purpose • use	用途值

业务维度	标签键（key）	标签值（value）

项目类标签	• 项目维度：
	  ◦ project
	  ◦ risk
	  ◦ schedule
	  ◦ subtask
	  ◦ environment
	• 人员维度：
	  ◦ sponsor
	  ◦ member
	  ◦ decisionMaker或owner
	  ◦ creator
		项目相关值

业务部门（实现成本分配和业务跟踪）	• costcenter • businessunit • biz • financecontact	部门相关值
财务维度责任人（确定资源负责人）	owner	人名或邮箱等
财务维度客户（识别资源组服务的客户）	自定义或真实值	客户名称
财务维度项目（确定资源支持的项目）	project	项目名称
财务维度订单	order	订单分类ID

相关链接

- 使用标签检索资源
- 通过OOS批量启动ECS实例实践
- 全局标签实践
- 使用OOS批量修改标签值
- 基于标签的自动化分组监控
- 创建带特定标签的资源

相关API

- TagResources
- ListTagResources
- UntagResources

14.使用标签控制云助手命令的执行

云服务器ECS资源绑定标签后，您需要对RAM用户授予某种要求的标签鉴权策略，RAM用户才可以通过执行云助手命令控制带有某种标签的ECS实例，并对该ECS实例进行控制访问。本文介绍如何通过标签控制云助手命令的执行。

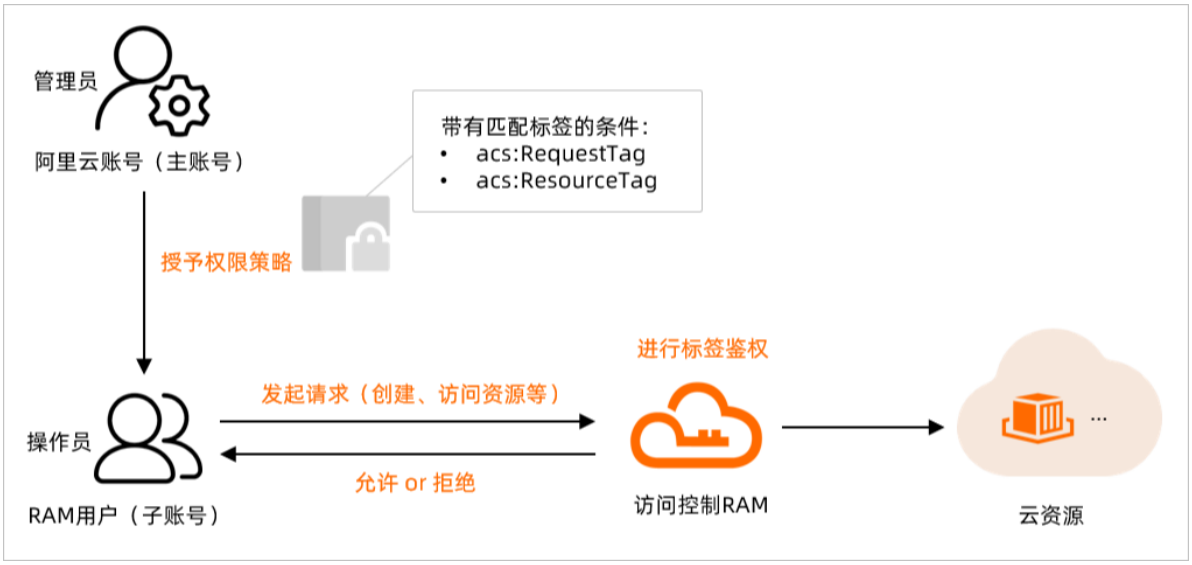
前提条件

- 已创建RAM用户，详情请参见创建RAM用户。
- 已创建云助手命令，详情请参见创建命令。

背景信息

- 标签由一组键值对组成，可以用于标记ECS实例，实现资源的分类管理。更多信息，请参见标签概述。
- 访问控制RAM可基于权限策略，管理用户身份，控制云资源的访问和操作权限。您可以从地域、ECS实例、云助手命令等维度设计自定义策略，并授权给RAM用户使用，从而灵活控制RAM用户使用云助手命令的权限。更多信息，请参见RAM用户概览和权限策略概览。
- 标签和RAM结合，将标签作为权限策略的匹配条件，可以实现对云资源的精细化管理。

  基于标签控制RAM用户权限（即标签鉴权）的逻辑如下：



应用场景说明

本文以如下应用场景为示例，说明如何使用标签鉴权控制云助手命令。

- RAM用户只能将命令执行到带有标签（例如test:tony）的ECS实例上。
- RAM用户只能将文件传输到带有标签（例如test:tony）的ECS实例上。
- RAM用户可以查询标签、实例或云助手命令和执行结果。

操作步骤

本步骤将使用阿里云账号（主账号）新建自定义策略 `UseTagAccessResoures` 为例，将自定义策略 `UseTagAccessResoures` 授权给RAM用户后，RAM用户只能将命令执行或者文件上传到带有标签 `test:tony` 的ECS实例上。

1. 创建带有标签的ECS实例。

本步骤以创建标签为 `test:tony` 的ECS实例为例。具体操作，请参见创建带特定标签的资源。

2. 使用阿里云账号（主账号）登录RAM控制台。
3. 创建自定义策略 `UseTagAccessResoures` 。

   具体操作，请参见创建自定义权限策略。

   您可以在权限策略（Condition）中为ECS资源设置多个标签条件来限制操作权限，支持的标签鉴权条件如下所示：

标签鉴权条件	说明
`acs:RequestTag`	限制在请求中必须传入特定的标签。 如果API请求中没有标签参数，则不能使用 `acs:RequestTag` ，否则会导致鉴权失败。
`acs:ResourceTag`	限制指定的资源必须包含特定的标签。 如果API请求中没有资源ID参数，则不能使用 `acs:ResourceTag` ，否则会导致鉴权失败。

```
{
    "Version": "1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:InvokeCommand",
                "ecs:RunCommand",
                "ecs:StopInvocation",
                "ecs:SendFile"
            ],
            "Resource": "acs:ecs:*:*:instance/*",
            "Condition": {
                "StringEquals": {
                    "acs:ResourceTag/test": "tony"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecs:InvokeCommand",
                "ecs:RunCommand",
                "ecs:StopInvocation",
                "ecs:SendFile"
            ],
            "Resource": "acs:ecs:*:*:command/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecs:DescribeTag*",
                "ecs:DescribeInstance*"
```

```
                "ecs:DescribeInstance",
                "ecs:DescribeCommands",
                "ecs:CreateCommand",
                "ecs:DeleteCommand",
                "ecs:ModifyCommand",
                "ecs:DescribeInvocationResults",
                "ecs:DescribeSendFileResults",
                "ecs:DescribeInstances",
                "ecs:DescribeCloudAssistantStatus",
                "ecs:DescribeInvocations",
                "ecs:DescribeResourceByTags",
                "ecs:DescribeTagKeys",
                "ecs:DescribeTags",
                "ecs:ListTagResources",
                "ecs:DescribeManagedInstances"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "oos:ListSecretParameters",
            "Resource": "*"
        }
    ]
}
```

以上策略可以实现如下权限控制：

- 允许在绑定 `test:tony` 标签的ECS实例上执行云助手命令或者发送远程文件，具体权限策略如下所示：

```
{
      "Effect": "Allow",
      "Action": [
         "ecs:InvokeCommand",
         "ecs:RunCommand",
         "ecs:StopInvocation",
         "ecs:SendFile"
      ],
      "Resource": "acs:ecs:*:*:instance/*",
      "Condition": {
          "StringEquals": {
              "acs:ResourceTag/test": "tony"
          }
      }
},
{
     "Effect": "Allow",
     "Action": [
         "ecs:InvokeCommand",
         "ecs:RunCommand",
         "ecs:StopInvocation",
         "ecs:SendFile"
     ],
     "Resource": "acs:ecs:*:*:command/*"
}
```

- 允许查询标签、实例以及云助手等资源，具体权限策略如下所示：

```
{
     "Effect": "Allow",
     "Action": [
           "ecs:DescribeTag*",
           "ecs:DescribeInstance*",
           "ecs:DescribeCommands",
           "ecs:CreateCommand",
           "ecs:DeleteCommand",
           "ecs:ModifyCommand",
           "ecs:DescribeInvocationResults",
           "ecs:DescribeSendFileResults",
           "ecs:DescribeInstances",
           "ecs:DescribeCloudAssistantStatus",
           "ecs:DescribeInvocations",
           "ecs:DescribeResourceByTags",
           "ecs:DescribeTagKeys",
           "ecs:DescribeTags",
           "ecs:ListTagResources",
           "ecs:DescribeManagedInstances"
        ],
        "Resource": "*"
 },
{
     "Effect": "Allow",
     "Action": "oos:ListSecretParameters",
     "Resource": "*"
}
```

4. 将自定义权限策略 `UseTagAccessResoures` 授予您希望控制访问的RAM用户。

   具体操作，请参见为RAM角色授权。

5. 验证权限策略是否生效。

   登录ECS管理控制台后，对待验证的实例执行以下操作：

   > **说明** 对实例执行云助手命令或者发送远程文件的同时，已经同步进行了查询标签、实例以及云助手资源等操作。

   - 对不同标签的ECS实例执行云助手命令

     在**ECS云助手**页面中的**命令列表**页签下，选择已创建好的云助手命令（以命令ID为 `c-hz02jt1ncrf***` 为例），对不同标签的ECS实例执行云助手命令。具体操作，请参见执行命令。

     > **说明** 如果**命令列表**页签下还没有创建好的云助手命令，您也可以创建云助手命令后再进行后续操作。具体操作，请参见创建命令。

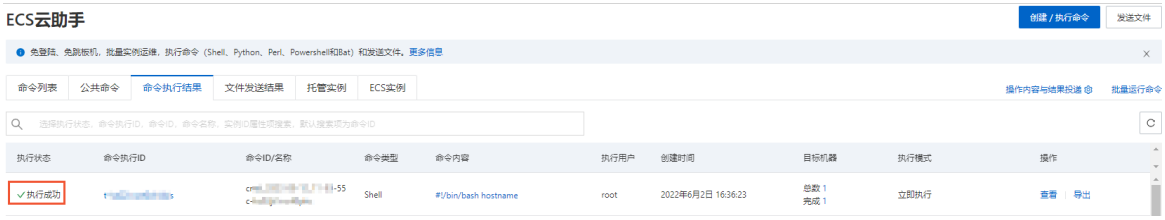
   - 对不同标签的ECS实例发送远程文件

     在**ECS云助手**页面中右上角位置，单击**发送文件**，对不同标签的ECS实例发送远程文件。具体操作，请参见上传本地文件到ECS实例。

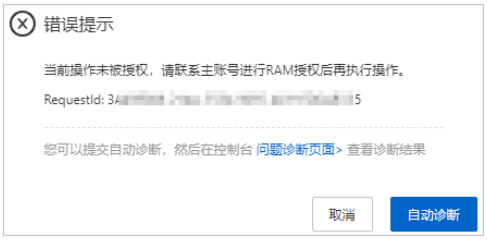
   如果权限策略已生效，上述操作的执行结果详情，请参见执行结果。如果权限策略未生效，请参见常见问题进行排查。

执行结果

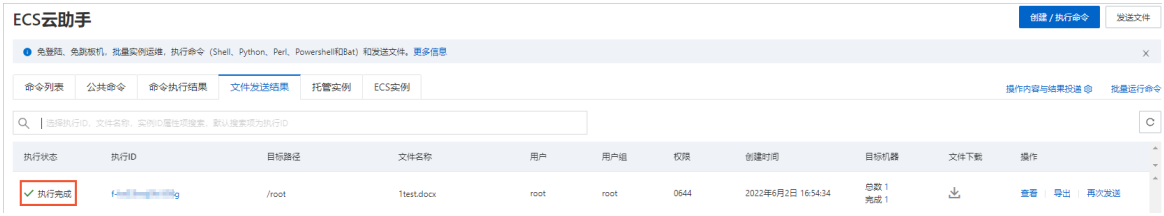
- 在ECS实例中执行云助手命令
  - ECS实例已绑定 `test:tony` 标签，则执行云助手命令时显示执行成功。



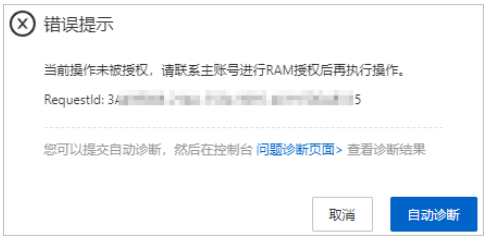
  - ECS实例未绑定 `test:tony` 标签，则执行云助手命令时显示执行失败。



- 发送远程文件到ECS实例中
  - ECS实例已绑定 `test:tony` 标签，则发送远程文件时显示执行成功。



  - ECS实例未绑定 `test:tony` 标签，则发送远程文件时显示执行失败。



常见问题

如果权限策略控制功能未生效怎么办？

如果权限控制功能未生效，请检查授权的RAM用户权限是否已经对Action中的以下几个参数设置了Allow操作。如果该参数已设置了Allow操作，请您将该权限策略从RAM用户权限中移除。

- ecs:InvokeCommand
- ecs:RunCommand
- ecs:StopInvocation
- ecs:SendFile

例如，如果自定义权限策略中存在如下所示的策略，请您将该权限策略从RAM用户权限中移除。

```
{
    "Version": "1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:InvokeCommand",
                "ecs:RunCommand",
                "ecs:StopInvocation",
                "ecs:SendFile"
            ],
            "Resource": "*"
        }
    ]
}
```

15.通过API设置自定义镜像的启动模式为UEFI模式

如果您需要使用支持UEFI启动模式的自定义镜像，需要参考本文设置该自定义镜像的启动模式为UEFI模式。

背景信息

阿里云支持通过API的方式，设置自定义镜像的启动模式为UEFI模式。当您的业务需要使用支持UEFI启动的自定义镜像时（例如，阿里云的部分实例规格族ebmg6a、ebmc6a、ebmr6a等，要求自定义镜像的启动模式为UEFI模式），可以通过ImportImage或者ModifyImageAttribute接口，为自定义镜像设置UEFI启动模式。

使用限制

设置自定义镜像启动模式时，存在以下限制：

- 仅支持通过ECS API的方式为自定义镜像设置UEFI启动模式。
- 镜像的启动模式暂不支持通过ECS控制台或查询类的API（例如DescribeImages）进行查询。

注意事项

您需要注意：

- 通过ECS API设置自定义镜像为UEFI启动模式后，仅表示通过该自定义镜像的ECS实例会以UEFI模式启动。因此，您需要自行确保该自定义镜像内的配置已经支持了UEFI启动模式，否则ECS实例将会启动失败。
- 要求自定义镜像启动模式为UEFI模式的实例规格族中，部分处于邀测或公测阶段，如果您在使用UEFI相关的功能或资源时出现异常，建议提交工单反馈。

操作说明

根据实际业务场景不同，您可以通过以下任一方式，为自定义镜像设置UEFI启动模式。

方式	说明
方式一：通过ImportImage导入镜像时设置UEFI启动模式	当您需要把已支持UEFI启动的镜像导入到阿里云作为自定义镜像时，可以调用ImportImage接口导入镜像，并在调用接口时设置 `BootMode` 参数的值为 `UEFI` 。关于接口的详细说明与注意事项，请参见ImportImage。
方式二：通过ModifyImageAttribute修改自定义镜像的启动模式为UEFI模式	当您的阿里云账号下存在已支持UEFI启动模式的自定义镜像时，可以调用ModifyImageAttribute修改自定义镜像的属性，并在调用接口时设置 `BootMode` 参数为 `UEFI` 。关于接口的详细说明与注意事项，请参见ModifyImageAttribute。

调用ECS API的方式如下，您可以选择任一方式调用API为自定义镜像设置UEFI启动模式。

- 通过阿里云OpenAPI开发者门户，可以快速对ECS API进行调试。更多信息，请参见OpenAPI开发者门户。
- 通过云服务器ECS提供的软件开发工具包（SDK）调用ECS相关的API。更多信息，请参见SDK。
- 通过阿里云命令行工具（CLI），您可以在命令行Shell中，使用aliyun命令与阿里云服务进行交互，管理您的阿里云资源。更多信息，请参见阿里云CLI。

16.基于SCC实例规格族的RDMA驱动安装说明

如果您是基于CentOS 7.9或CentOS 8.4版本的自定义镜像创建的ecs.sccc7或ecs.sccg7规格族的超级计算集群SCC（Super Computing Cluster）实例，且需要实现多台实例之间使用RoCE RDMA通信，则需要参考本文在实例内手动安装RDMA驱动，来保证您可以正常使用RDMA功能。

前提条件

已通过CentOS 7.9或CentOS 8.4版本的自定义镜像创建了ecs.sccc7或ecs.sccg7规格族的SCC实例。具体操作，请参见使用自定义镜像创建实例。

背景信息

SCC实例规格族ecs.sccc7和ecs.sccg7配置了RDMA网卡，处于同一可用区同一高可用集群内的多台实例之间可以使用RoCE RDMA网卡通信。阿里云的SCC实例默认仅支持特定的SCC版操作系统镜像，如果是基于CentOS 7.9或CentOS 8.4版本的自定义镜像创建的ecs.sccc7或ecs.sccg7规格的实例，默认没有安装RDMA驱动，需手动安装。

> **说明** 除以上情况，SCC实例默认已安装了RDMA驱动，无需再手动安装。

- 有关超级计算集群SCC的更多信息，请参见超级计算集群概述。
- 有关RDMA网卡的更多信息，请参见使用ERI。

操作步骤

1. 远程连接已创建的SCC实例。

   具体操作，请参见连接方式概述ECS远程连接操作指南。

2. 依次运行以下命令，安装相关依赖包。

   i. 安装DKMS（Dynamic Kernel Module Support）。

   ```
   yum install dkms -y
   ```

   ii. 运行以下命令，下载MFT工具安装包。

   ```
   wget https://www.mellanox.com/downloads/MFT/mft-4.17.0-106-x86_64-rpm.tgz
   ```

   iii. 运行以下命令，解压MFT工具安装包。

   ```
   tar zvxf mft-4.17.0-106-x86_64-rpm.tgz
   ```

   iv. 依次运行以下命令，进入MFT工具并运行该工具。

   ```
   cd mft-4.17.0-106-x86_64-rpm
   ./install.sh
   ```

3. 依次运行以下命令，分别下载SCC实例规格族定制的驱动包。

```
wget https://scc7-pkg.oss-cn-shanghai.aliyuncs.com/nic-drivers-mellanox-rdma-4.0.0-8.noarch.rpm
wget https://scc7-pkg.oss-cn-shanghai.aliyuncs.com/nic-libs-mellanox-rdma-4.0.0-1.x86_64.rpm
wget https://scc7-pkg.oss-cn-shanghai.aliyuncs.com/rdma-service-eflops-3.1.1u4-lossyv4.noarch.rpm
wget https://scc7-pkg.oss-cn-shanghai.aliyuncs.com/ali-bonding-1.0.0-4.noarch.rpm
```

4. 依次运行以下命令，分别安装SCC实例规格族定制的驱动包。

```
rpm -ivh nic-drivers-mellanox-rdma-4.0.0-8.noarch.rpm
rpm -ivh nic-libs-mellanox-rdma-4.0.0-1.x86_64.rpm
rpm -ivh rdma-service-eflops-3.1.1u4-lossyv4.noarch.rpm
rpm -ivh ali-bonding-1.0.0-4.noarch.rpm
```

5. 在*/etc/dhcp/dhclient.conf*配置文件中添加 `bootp-broadcast-always;` 配置。
   i. 打开*/etc/dhcp/dhclient.conf*文件。

   ```
   vim /etc/dhcp/dhclient.conf
   ```

   ii. 按*i*键进入编辑模式，在配置文件中添加一行 `bootp-broadcast-always;` 配置信息。
   iii. 按*Esc*键退出编辑模式，输入 `:wq` 并按下*Enter*键，保存并退出文件。
6. 将RDMA网卡（eth1、eth2）绑定到bond0网卡。
   - 新建*ifcfg-bond0*文件并添加内容。
     a. 运行以下命令，新建*ifcfg-bond0*文件。

     ```
     vim /etc/sysconfig/network-scripts/ifcfg-bond0
     ```

     b. 按*i*键进入编辑模式，添加以下内容到文件中。

     ```
     DEVICE=bond0
     BOOTPROTO=dhcp
     TYPE="ethernet"
     ONBOOT=yes
     USERCTL=no
     PEERDNS=no
     BONDING_OPTS="miimon=100 mode=4 xmit_hash_policy=layer3+4"
     DEFROUTE=no
     ```

     c. 按*Esc*键退出编辑模式，输入 `:wq` 并按下*Enter*键，保存并退出文件。
   - 新建*ifcfg-eth1*文件并添加内容。
     a. 运行以下命令，查看eth1网卡的ether地址。

     ```
     ifconfig -a
     ```

     b. 运行以下命令，新建*ifcfg-eth1*文件。

     ```
     vim /etc/sysconfig/network-scripts/ifcfg-eth1
     ```

c. 按键进入编辑模式，添加以下内容到文件中。

```
DEVICE=eth1
TYPE="Ethernet"
HWADDR=xx:xx:xx:xx:xx:xx
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
PEERDNS=no
ETHTOOL_OPTS="autoneg on"
RX_MAX=`ethtool -g "$DEVICE" | grep 'Pre-set' -A1 | awk '/RX/{print $2}'`
RX_CURRENT=`ethtool -g "$DEVICE" | grep "Current" -A1 | awk '/RX/{print $2}'`
[[ "$RX_CURRENT" -lt "$RX_MAX" ]] && ethtool -G "$DEVICE" rx "$RX_MAX"
```

其中，`HWADDR`是eth1网卡的ether地址，请您替换为实际值。

d. 按*Esc*键退出编辑模式，输入`:wq`并按下*Enter*键，保存并退出文件。

- 新建*ifcfg-eth2*文件并添加内容。

a. 运行以下命令，查看eth2网卡的ether地址。

```
ifconfig -a
```

b. 运行以下命令，新建*ifcfg-eth2*文件。

```
vim /etc/sysconfig/network-scripts/ifcfg-eth2
```

c. 按键进入编辑模式，添加以下内容到文件中。

```
DEVICE=eth2
TYPE="Ethernet"
HWADDR=xx:xx:xx:xx:xx:xx
BOOTPROTO=none
ONBOOT=yes
MASTER=bond0
SLAVE=yes
PEERDNS=no
ETHTOOL_OPTS="autoneg on"
RX_MAX=`ethtool -g "$DEVICE" | grep 'Pre-set' -A1 | awk '/RX/{print $2}'`
RX_CURRENT=`ethtool -g "$DEVICE" | grep "Current" -A1 | awk '/RX/{print $2}'`
[[ "$RX_CURRENT" -lt "$RX_MAX" ]] && ethtool -G "$DEVICE" rx "$RX_MAX"
```

其中，`HWADDR`是eth2网卡的ether地址，请您替换为实际值。

d. 按*Esc*键退出编辑模式，输入`:wq`并按下*Enter*键，保存并退出文件。

7. 运行以下命令，激活bond0网卡。

```
ifup bond0
```

当出现如下结果时，表示bond0网卡激活成功。

```
Determining IP information for bond0... done.
```

8. 运行以下命令，确认bond0已获取RDMA IP地址。

```
ifconfig bond0
```

当查询到bond0网卡与eth1、eth2网卡的ether地址相同时，表示RDMA驱动安装完成。

17.自定义镜像构建实践

17.1. 自定义镜像构建概述

阿里云支持通过运维编排服务OOS、快照、ECS实例以及Packer构建自定义镜像，还支持导入本地的自定义镜像。出于安全考虑，如果您频繁地使用自定义镜像，建议您定期更新镜像。例如，定期升级操作系统补丁、更新中间件软件、更新证书或者安装最新的第三方软件。

构建方式对比

阿里云支持的构建自定义镜像的方式对比见下表所示：

构建方式	优点	缺点	计费
使用OOS更新自定义镜像	• 官方模板，无需编码 • 在线使用，无需安装 • 无需登录密钥，安全可靠 • 可视化执行过程	无	可能涉及实例规格、云盘、快照等资源消费。详情请参见计费概述。
• 使用快照创建自定义镜像 • 使用实例创建自定义镜像	通过ECS控制台操作，简单易用	• 随预装软件扩充变得复杂 • 难以确保人工操作是否准确无误和前后一致 • 后期维护成本高	可能涉及快照使用费用，详情请参见快照计费。
使用Packer构建自定义镜像	工具开源并支持众多云服务提供商	• 需要安装和维护 • 需要自己编写脚本	可能涉及实例规格、云盘、快照等资源消费。详情请参见计费概述。
创建并导入自定义镜像	可实现应用迁移上云	需操作图形化界面和配置虚拟驱动，有一定难度	可能涉及快照使用费用和对象存储OSS存储费用，详情请参见快照计费和 *对象存储OSS文档* 计费概述。

构建流程

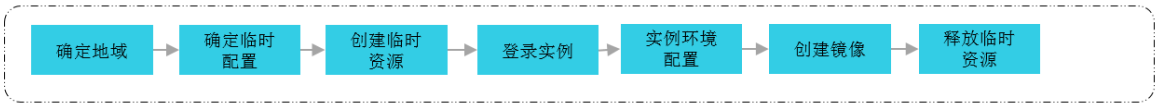
除导入自定义镜像外，构建自定义镜像均依赖于ECS实例某一时刻的系统状态和应用数据。其中，运维编排服务OOS与Packer都是通过自动创建并释放临时ECS实例实现自定义镜像构建，更适合敏捷的开发流程。

不同的自定义镜像构建流程如下所示：

- 使用OOS更新自定义镜像，需要您使用公共模板（例如ACS-ECS-UpdateImage）或者创建自定义运维模板。通过创建运维任务构建自定义镜像，构建流程可以通过YAML、JSON或者可视化预览呈现。

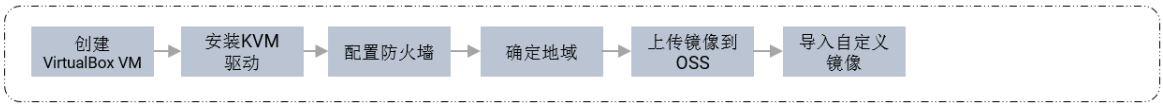
- 使用快照或ECS实例构建自定义镜像，需要您使用已有的快照或者运行中的ECS实例在ECS管理控制台创建

自定义镜像。如果您只需要获取自定义镜像，还需要自行释放临时创建的ECS实例。



- 使用Packer构建自定义镜像，需要您根据Packer的生成器等JSON模板自行编写脚本。

- 创建并导入本地的自定义镜像，需要您根据阿里云自定义镜像规范配置虚拟机（例如VirtualBox VM），再通过对象存储OSS将自定义镜像导入到ECS。自定义镜像规范请参见导入镜像必读。



17.2. 使用OOS更新自定义镜像

运维编排服务OOS为更新自定义镜像的场景提供了公共模版。您只需选择一个源镜像，输入更新镜像所需的云助手脚本等必要参数，就可以创建立即执行或定时执行的运维任务，一键更新自定义镜像。

前提条件

背景信息

在更新自定义镜像的完整流程中，运维编排服务OOS的ACS-ECS-UpdateImage公共模板按顺序执行以下任务，并生成新的自定义镜像：

1. 检查新自定义镜像的名称是否已存在，以及是否符合规则。
2. 根据您配置的实例规格、源镜像ID、安全组ID等参数创建并运行一台临时ECS实例。
3. 检查临时ECS实例是否安装了云助手客户端，若缺失则安装云助手客户端。
4. 在临时ECS实例上，通过云助手执行脚本更新实例系统环境。

> **说明** 运维编排服务OOS通过调用云助手API执行Shell、Bat或者PowerShell等脚本，更新ECS实例的系统应用环境。更多详情，请参见云助手概述。

5. 停止临时ECS实例。
6. 根据临时ECS实例创建自定义镜像。
7. 释放临时ECS实例。

操作步骤

1. 登录OOS管理控制台。
2. （可选）如果您是第一次使用运维编排服务OOS，单击**立即开通**。
3. 在左侧导航栏，单击**公共模板**。
4.

5. 选择**更新ECS镜像**，单击**创建执行**。
6. 在**创建执行**页签中，完成以下操作：
   i. 保持**基本信息**的默认设置，单击**下一步：设置参数**。
   ii. 完成**参数设置**填写，用以自动化创建或更新自定义镜像运维任务，参数含义如下表所示。

参数	说明	示例
targetImageName	更新的自定义镜像的名称，必须满足正则表达式*/^[A-Za-z0-9\-_]*$/*要求，且不能和已有镜像名称重名。	*add_testtxt_20191010*
sourceImageId	待更新的源镜像ID。 **说明** 如果您还未创建过自定义镜像，可以使用公共镜像ID，例如*centos_7_06_64_20G_alibase_20190711.vhd*。	*m-bp13y4of6mdoqw*******
instanceType	用以创建临时ECS实例的实例规格，取值请参见实例规格族。	*ecs.g5.xlarge*
securityGroupId	用以创建临时ECS实例的安全组ID。	*sg-bp1azkttqpldxg*******
vSwitchId	用以创建临时ECS实例的交换机ID。指定的虚拟交换机必须和安全组在同一个专有网络VPC中。	*vsw-bp1s5fnvk4gn2tw*******
ramRoleName	实例的RAM角色。	*TestRAMRole*
commandType	云助手脚本类型： ■ RunShellScript：Linux实例适用的Shell脚本。 ■ RunBatScript：Windows实例适用的Bat脚本。 ■ RunPowerShellScript：Windows实例适用的PowerShell脚本。	*RunShellScript*
tags	镜像标签。 ■ **标签键（必选）**：镜像的标签键。 ■ **标签值（可选）**：镜像的标签值。 ■ （默认设置）**将资源的标签同时打到该执行上**：将资源的标签绑定到该OOS模板。执行后，可以方便您在列表页通过标签过滤标签资源对应的执行结果。	■ 标签键：ECS ■ 标签值：Image
commandContent	在临时ECS实例中执行的脚本内容。	`echo "hello world" >/root/test.txt .`

参数	说明	示例
执行使用到的权限的来源	可选参数。 ■ （默认设置）**当前账号的已有权限**：执行您使用的账号的权限动作。请确保您拥有创建自定义镜像涉及的所有ECS API调用权限。 ■ **指定RAM角色，使用该角色的权限**：如果指定了RAM角色名称，OOS扮演该RAM角色执行运维任务。	**当前账号的已有权限**

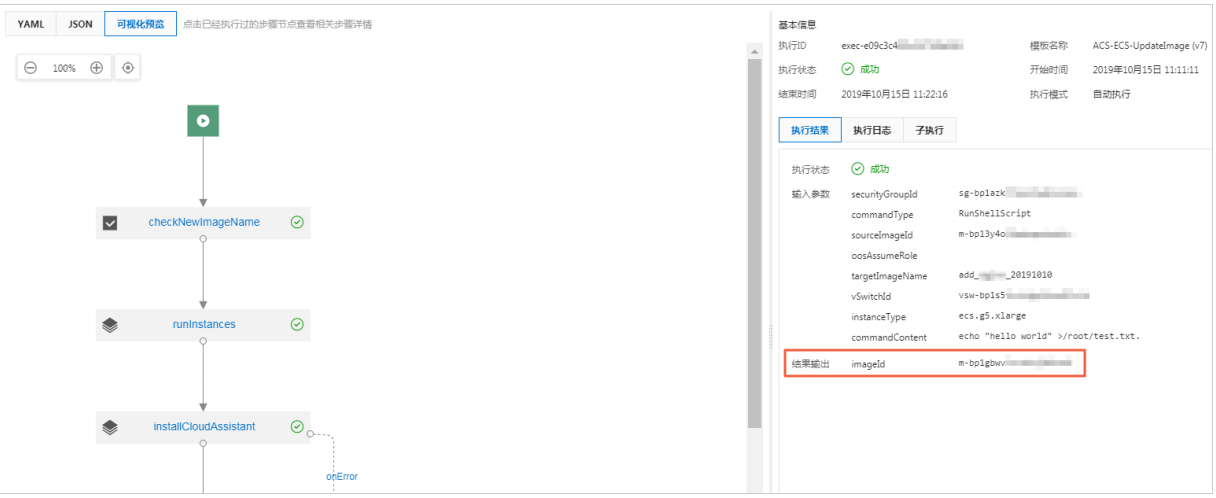
iii. 单击**下一步：确定**。

iv. 查看运维任务详情以及风险操作，确认无误后单击**确认风险并执行**。

7. 在**执行管理**中查看创建的运维任务。

执行结果

若成功创建运维任务，且执行状态处于**运行中**，则表示更新自定义镜像正在进行中。当**执行状态**转换为**成功**时，则表示镜像更新成功，您可以在执行详情中查看新镜像ID。



说明 如需了解更新自定义镜像的详细过程，您可单击该执行的**详情**，查看**执行日志**，了解运维任务的实时进度和状态。

相关文档

相关文档

- 什么是运维编排服务
- 使用运维编排创建和更新自定义镜像
-

17.3. Packer实践之镜像即代码

17.3.1. Packer构建镜像的优势

通过Packer，您只需在JSON配置文件中指明构建镜像所需的基本信息、以及需要安装到镜像中的软件及配置，即可自动化构建ECS镜像。

前提条件

使用本教程进行操作前，请确保您已经注册了阿里云账号。如还未注册，请先完成账号注册。

背景信息

Packer是HashiCorp推出的一款镜像工具，旨在通过简易的方式自动化构建镜像。由于构建镜像的过程以一份JSON配置文件为准，您无需担心多次构建的镜像不一致。Packer还能为测试和更新镜像带来使用便利，降低运维和管理镜像的时间成本。更多详情，请访问Packer官网。

操作条件

本文通过比较“使用实例创建自定义镜像”和“使用Packer构建自定义镜像”的操作流程，突出Packer在DevOps场景中的优势。以下为本次操作的假设场景和一致性条件：

- 目标地域：阿里云华北2（北京）地域，更多详情，请参见地域和可用区。
- 操作系统：CentOS 7.3 64位。本文两种方式均采用公共镜像centos_7_03_64_20G_alibase_20170818.vhd，您可以在ECS管理控制台或调用DescribeImages查询其他操作系统的镜像ID列表。
- 自定义服务：Redis。
- 是否保留临时资源：否。

> **说明** 本文操作会创建计费资源，请注意释放和清理。如实例、公网IP、快照等。

使用实例创建自定义镜像

本示例介绍如何通过ECS管理控制台创建一份自定义镜像。

1. 登录ECS管理控制台。
2. 在左侧导航栏，选择**实例与镜像 > 实例**。
3. 在顶部菜单栏左上角处，选择地域。
4. 创建实例。具体操作请参见使用向导创建实例。

   为较少费用消耗和简化操作流程，您可以选择以下配置：
   - 计费方式：按量付费。更多详情，请参见按量付费。
   - 实例规格：ecs.t5-lc1m1.small。更多详情，请参见实例规格族。
   - 公共镜像：CentOS 7.3 64位。
   - 专有网络：默认VPC。
   - 安全组：默认安全组。
   - 公网带宽：如果不需要公网访问，可以选择不开通公网带宽，并通过管理终端远程连接实例。更多详情，请参见通过密码认证登录Linux实例。
5. 远程连接ECS实例。

   关于连接方式的介绍，请参见连接方式概述ECS远程连接操作指南。
6. 运行 `yum install redis.x86_64 -y` 安装Redis服务。
7. 返回ECS控制台，选择华北2（北京）地域。

8. 创建一份镜像。具体操作，请参见使用实例创建自定义镜像。
9. 在左侧导航栏，选择**实例与镜像 > 镜像**。
10. 在**镜像列表**页面查看镜像完成状态。
11. （可选）镜像制作成功后，请释放临时资源，包括实例等。若您使用的是弹性公网IP，也可以选择释放。

使用Packer构建自定义镜像

您已经安装了Packer。关于如何安装Packer，请参见Packer官方文档或者阿里云文档使用Packer构建自定义镜像。完成以下操作，使用Packer构建自定义镜像：

1. 本地新建一份*alicloud.json*文件，文件内容如下：

```
{
  "variables": {
    "access_key": "{{env `ALICLOUD_ACCESS_KEY`}}",
    "secret_key": "{{env `ALICLOUD_SECRET_KEY`}}"
  },
  "builders": [{
    "type":"alicloud-ecs",
    "access_key":"{{user `access_key`}}",
    "secret_key":"{{user `secret_key`}}",
    "region":"cn-beijing",
    "image_name":"packer_basic",
    "source_image":"centos_7_03_64_20G_alibase_20170818.vhd",
    "ssh_username":"root",
    "instance_type":"ecs.t5-lc1m1.small",
    "internet_charge_type":"PayByTraffic",
    "io_optimized":"true"
  }],
  "provisioners": [{
    "type": "shell",
    "inline": [
      "sleep 30",
      "yum install redis.x86_64 -y"
    ]
  }]
}
```

Packer参数解释

参数	示例值	参数解释
variables{"variable1":"value"}	variables{"access_key":"{{env `ALICLOUD_ACCESS_KEY`}}"}	定义了builders中会用到的变量（variables）。将AccessKey（access_key和secret_key）信息写入配置文件有信息泄露的风险，设置成变量后可防止意外，变量的值取自运行时的输入值。
builders{"type":"value"}	builders{"type":"alicloud-ecs"}	Packer定义的镜像生成器（builders）。阿里云支持alicloud-ecs，又称Alicloud Image Builder，用于在阿里云ECS创建自定义镜像。

参数	示例值	参数解释
provisioners{"type":"value"}	provisioners{"type":"shell"}	Packer定义的镜像配置器（provisioners），用以定义需要在临时实例内执行的操作。本文使用的是Shell Provisioner，表示在连接Linux实例后自动执行一段Shell命令（如 `yum install redis.x86_64 -y` ）安装Redis服务。

阿里云参数解释

参数	数据类型	示例值	参数解释	重要度
access_key	String	LTAInPyXXXXQXXXX	您的AccessKey ID。更多详情，请参见获取AccessKey。 **说明** 由于AccessKey权限过大，为防止错误操作，建议您创建RAM用户，并使用RAM子账号创建AccessKey。更多详情，请参见创建RAM用户。	高
secret_key	String	CM1ycKrrCekQ0dhXXXXXXXXXl7yavUT	您的AccessKey Secret。	高
region	String	cn-beijing	目标自定义镜像的所属地域。更多详情，请参见地域和可用区。	高
image_name	String	packer_basic	目标自定义镜像的名称。不允许与已有镜像重名。	低
source_image	String	centos_7_03_64_20G_alibase_20170818.vhd	具有相同操作系统的阿里云公共镜像ID。	高
instance_type	String	ecs.t5-lc1m1.small	创建自定义镜像时使用的临时实例的实例规格。更多详情，请参见实例规格族。	低
internet_charge_type	String	PayByTraffic	临时实例的公网带宽付费类型。建议设置为按流量付费（PayByTraffic）。	低
io_optimized	Boolean	true	临时实例的I/O优化属性。建议设置为I/O优化（true）。	低

2. 执行以下命令构建一份镜像：

```
packer build alicloud.json
```

说明 构建镜像是相对耗时的任务，请您耐心等待。镜像构建成功后，会出现在相应阿里云地域中，您可以通过ECS控制台或DescribeImages查看。

构建镜像时会产生的操作日志。日志给出了构建过程中执行的每一个步骤，包括校验参数、创建临时资源、预安装软件、创建目标资源和释放临时资源等。

```
alicloud-ecs output will be in this color.
==> alicloud-ecs: Prevalidating image name...
    alicloud-ecs: Found image ID: centos_7_03_64_20G_alibase_20170818.vhd
==> alicloud-ecs: Creating temporary keypair: packer_xxx
==> alicloud-ecs: Creating vpc
==> alicloud-ecs: Creating vswitch...
==> alicloud-ecs: Creating security groups...
==> alicloud-ecs: Creating instance.
==> alicloud-ecs: Allocating eip
==> alicloud-ecs: Allocated eip xxx
    alicloud-ecs: Attach keypair packer_xxx to instance: i-xxx
==> alicloud-ecs: Starting instance: i-xxx
==> alicloud-ecs: Using ssh communicator to connect: ***
==> alicloud-ecs: Waiting for SSH to become available...
==> alicloud-ecs: Connected to SSH!
==> alicloud-ecs: Provisioning with shell script: /var/folders/k_/nv2r4drx3xxxxxxxxxxxn
db40000gn/T/packer-shell260049331
    alicloud-ecs: Loaded plugins: fastestmirror
    alicloud-ecs: Determining fastest mirrors
    alicloud-ecs: Resolving Dependencies
    alicloud-ecs: --> Running transaction check
    alicloud-ecs: ---> Package redis.x86_64 0:3.2.12-2.el7 will be installed
    alicloud-ecs: --> Processing Dependency: libjemalloc.so.1()(64bit) for package: red
is-3.2.12-2.el7.x86_64
    alicloud-ecs: --> Running transaction check
    alicloud-ecs: ---> Package jemalloc.x86_64 0:3.6.0-1.el7 will be installed
    alicloud-ecs: --> Finished Dependency Resolution
    alicloud-ecs:
    alicloud-ecs: Dependencies Resolved
    alicloud-ecs:
    alicloud-ecs: ================================================================
===========
    alicloud-ecs:  Package           Arch            Version                  Repositor
y     Size
    alicloud-ecs: ================================================================
===========
    alicloud-ecs: Installing:
    alicloud-ecs:  redis             x86_64          3.2.12-2.el7             epel
544 k
    alicloud-ecs: Installing for dependencies:
    alicloud-ecs:  jemalloc          x86_64          3.6.0-1.el7              epel
105 k
    alicloud-ecs:
    alicloud-ecs: Transaction Summary
    alicloud-ecs: ================================================================
===========
    alicloud-ecs: Install  1 Package (+1 Dependent package)
    alicloud-ecs:
    alicloud-ecs: Total download size: 648 k
    alicloud-ecs: Installed size: 1.7 M
    alicloud-ecs: Downloading packages:
```

```
    alicloud-ecs: ------------------------------------------------------------------
-----------
    alicloud-ecs: Total                                               2.2 MB/s | 648 kB
00:00
    alicloud-ecs: Running transaction check
    alicloud-ecs: Running transaction test
    alicloud-ecs: Transaction test succeeded
    alicloud-ecs: Running transaction
    alicloud-ecs:   Installing : jemalloc-3.6.0-1.el7.x86_64
1/2
    alicloud-ecs:   Installing : redis-3.2.12-2.el7.x86_64
2/2
    alicloud-ecs:   Verifying  : redis-3.2.12-2.el7.x86_64
1/2
    alicloud-ecs:   Verifying  : jemalloc-3.6.0-1.el7.x86_64
2/2
    alicloud-ecs:
    alicloud-ecs: Installed:
    alicloud-ecs:   redis.x86_64 0:3.2.12-2.el7
    alicloud-ecs:
    alicloud-ecs: Dependency Installed:
    alicloud-ecs:   jemalloc.x86_64 0:3.6.0-1.el7
    alicloud-ecs:
    alicloud-ecs: Complete!
==> alicloud-ecs: Stopping instance: i-xxx
==> alicloud-ecs: Waiting instance stopped: i-xxx
==> alicloud-ecs: Creating image: packer_basic
    alicloud-ecs: Detach keypair packer_xxx from instance: i-xxx
==> alicloud-ecs: Cleaning up 'EIP'
==> alicloud-ecs: Cleaning up 'instance'
==> alicloud-ecs: Cleaning up 'security group'
==> alicloud-ecs: Cleaning up 'vSwitch'
==> alicloud-ecs: Cleaning up 'VPC'
==> alicloud-ecs: Deleting temporary keypair...
Build 'alicloud-ecs' finished.
==> Builds finished. The artifacts of successful builds are:
--> alicloud-ecs: Alicloud images were created:
cn-beijing: m-bp67acfmxazb4ph***
```

相关文档

相关文档

- DescribeImages
- Alicloud Image Builder
- Examples
- Packer的DevOps配置

17.3.2. Packer的DevOps配置

本文提供了在阿里云ECS使用Packer创建自定义镜像的DevOps（开发运维一体化）常用配置，适用于使用Packer创建ECS自定义镜像的场景。

镜像标签

- 字段名称：tags{"key":"value"}。
- 适用场景：当您的自定义镜像达到一定的数量时，适当的标记镜像有利于镜像管理和检索。例如记录镜像版本号和镜像包含的应用类型等。阿里云Builder提供了tags参数，支持为镜像绑定标签。生成的镜像自动包含阿里云ECS标签，更多有关标签的详情，请参见标签概述。
- 配置作用：ECS管理控制台镜像列表页面和API DescribeImages均支持查询镜像时返回标签以及根据标签过滤镜像。为镜像绑定标签能够和Terraform一起为企业级标准化DevOps流程提供支持。
- 配置示例：以下配置文件为最终生成的镜像和对应的快照绑定 `version=v1.0.0` 和 `app=web` 两个标签。

```
{
  "variables": {
    "access_key": "{{env `ALICLOUD_ACCESS_KEY`}}",
    "secret_key": "{{env `ALICLOUD_SECRET_KEY`}}"
  },
  "builders": [{
    "type":"alicloud-ecs",
    "access_key":"{{user `access_key`}}",
    "secret_key":"{{user `secret_key`}}",
    "region":"cn-beijing",
    "image_name":"packer_basic",
    "source_image":"centos_7_03_64_20G_alibase_20170818.vhd",
    "ssh_username":"root",
    "instance_type":"ecs.t5-lc1m1.small",
    "internet_charge_type":"PayByTraffic",
    "io_optimized":"true",
    "tags": {
      "version": "v1.0.0",
      "app": "web"
    }
  }]
}
```

只包含系统盘快照

- 字段名称：image_ignore_data_disks，数据类型为Boolean。
- 适用场景：默认情况下Packer直接从ECS实例创建镜像，从实例创建镜像时如果包含数据盘，则镜像会同时包含数据盘快照。创建包含数据盘的实例通常有两种方式：
  - 方式一：通过image_disk_mappings设置数据盘相关参数。更多详情，请参见 *Packer文档*Alicloud Image Builder。
  - 方式二：选择默认带有数据盘的实例规格。该类实例规格包含的数据盘大多为本地盘，如ecs.d1ne.2xlarge。本地盘当前并不支持创建快照，所以无法直接通过此类实例创建镜像。
- 配置作用：如果您需要选择默认带有数据盘的实例规格，但实际上数据盘部分并不是必须的，可以在配置文件中加上 `"image_ignore_data_disks": "true"` 实现只基于系统盘创建镜像。

设置快照超时时间

- 字段名称：wait_snapshot_ready_timeout，数据类型为Interger，默认值为3600（秒s）。
- 适用场景：创建镜像依赖于快照，快照的创建时间依赖于磁盘大小。当磁盘较大时，创建快照所需时间会

相应增加。

- 配置作用：当磁盘太大导致超时错误时，可以通过 `wait_snapshot_ready_timeout` 调大超时时间。

通过私网IP连接实例

- 字段名称：ssh_private_ip，数据类型为Boolean。
- 适用场景：默认情况下，Packer创建EIP并绑定实例，再通过EIP对应的公网IP连接实例安装软件或执行命令。如果您能通过私网IP直接连接实例，可以免除公网IP。
- 配置作用：通过设置 `"ssh_private_ip": "true"` ，Packer不会分配EIP或者公网IP，而是通过私网IP连接实例。

设置停止实例选项

- 字段名称：disable_stop_instance，数据类型为Boolean。
- 适用场景：默认情况下，Packer执行完provisioners后，会先停止实例再创建镜像。某些特殊场景，如在Windows实例中运行Sysprep，需要实例处于运行中状态。

  Sysprep的使用场景示例请参见修改Windows实例SID以搭建域环境。
- 配置作用：通过设置 `"disable_stop_instance": "true"` ，Packer不会主动停止实例，而是假设配置（provisioners）中提供的命令会自行停止实例。

通过UserData启用WinRM

- 字段名称：user_data_file。
- 适用场景：出于安全考虑，Windows镜像默认关闭了WinRM（Windows Remote Management）。但连接Windows实例及之后在实例内部执行命令都依赖于WinRM。在实例创建时，您可以通过UserData启用WinRM。
- 配置作用：通过配置 `"user_data_file":"examples.ps1"` 指定UserData文件路径。
- 配置示例：本示例假定UserData文件在给定的相对路径*examples/alicloud/basic/winrm_enable_userdata.ps1*下。

```
{
  "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:\\"]
  }]
}
```

说明

- 示例中与WinRM相关的参数含义如下：
  - `"communicator": "winrm"` 表示通过WinRM连接实例。
  - `"winrm_port": 5985` 表示通信端口为5985。
  - `"winrm_username": "Administrator"` 表示连接时使用Administrator账户。
  - `"winrm_password": "Test1234"` 表示密码采用Test1234。
- `image_force_delete` 表示如果存在同名镜像，则先删除已有镜像。

基于本地ISO文件制作镜像

- 字段名称：builders{"type":"qemu"}，post-processors{"type":"alicloud-import"}。
- 适用场景：如果线下ISO文件环境为其他虚拟化环境，也可以通过Packer完成操作。
- 配置示例：如果线下环境使用的是qemu，请参见使用Packer创建并导入本地镜像。文档中包含两个重要的部分：
  i. 您需要使用本地虚拟化环境或软件对应的Builder，如Qemu Builder。
  ii. 请通过定义Alicloud Import Post-Processor将生成的本地镜像文件导入阿里云ECS。

如果您采用导入自定义镜像流程，请在本地安装虚拟化环境，将ISO文件制作成阿里云支持的镜像文件格式后再导入，如QCOW2、VHD和RAW。导入流程请参见导入镜像必读。

相关链接

更多参数和样例，请参见Packer官方文档Alicloud Image Builder和Examples。

17.4. 创建并导入自定义镜像

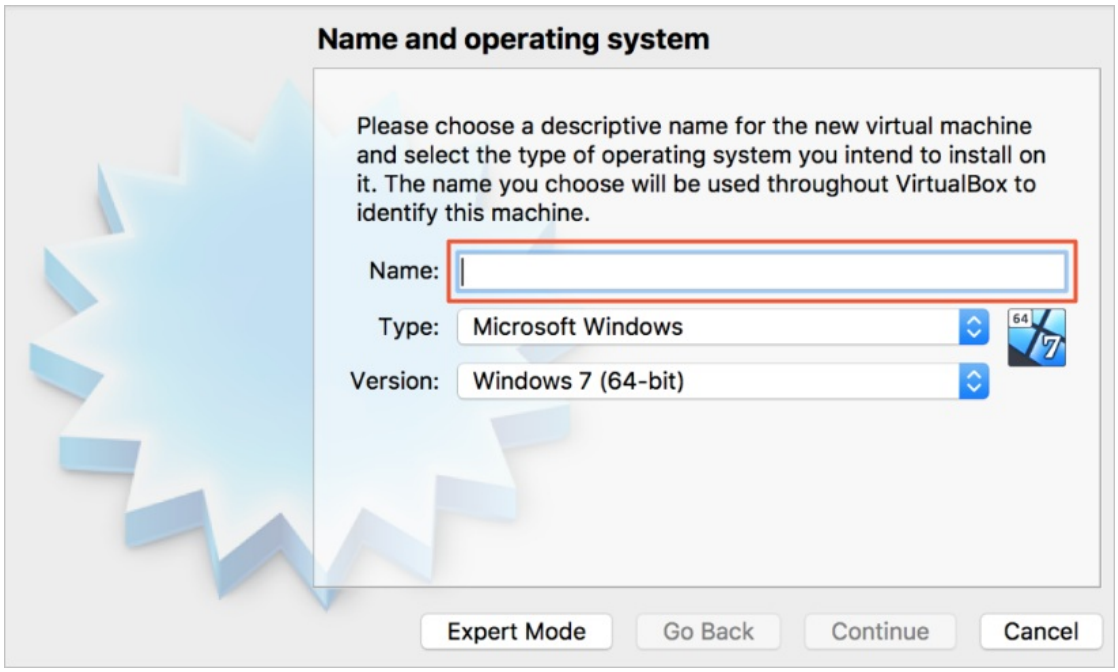
如果您在创建实例时，在阿里云未找到需要使用的操作系统，可以创建自定义镜像并将其导入ECS控制台，然后在创建实例时使用该自定义镜像。本文介绍如何创建自定义镜像并将自定义镜像导入阿里云。

前提条件

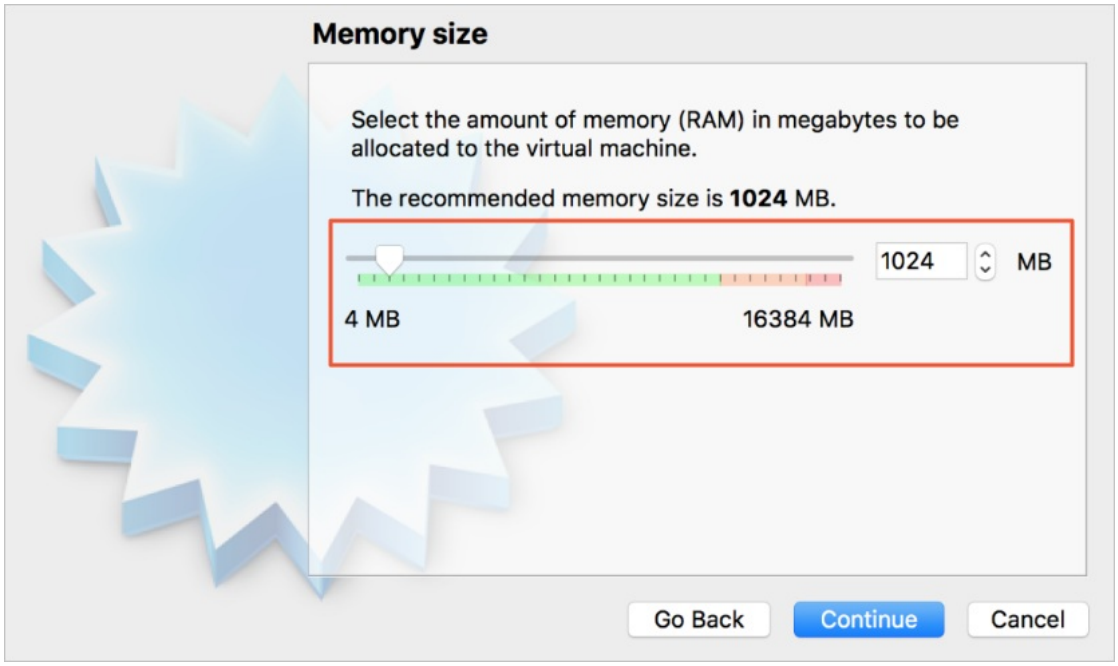
- 已安装VirtualBox，下载地址，请参见VirtualBox官网。
- 网络连接稳定。
-
- 已安装操作系统ISO或二进制文件，例如Red Hat Enterprise Linux。
- 已安装阿里云OSS浏览器，具体操作请参见快速使用ossbrowser。

操作步骤

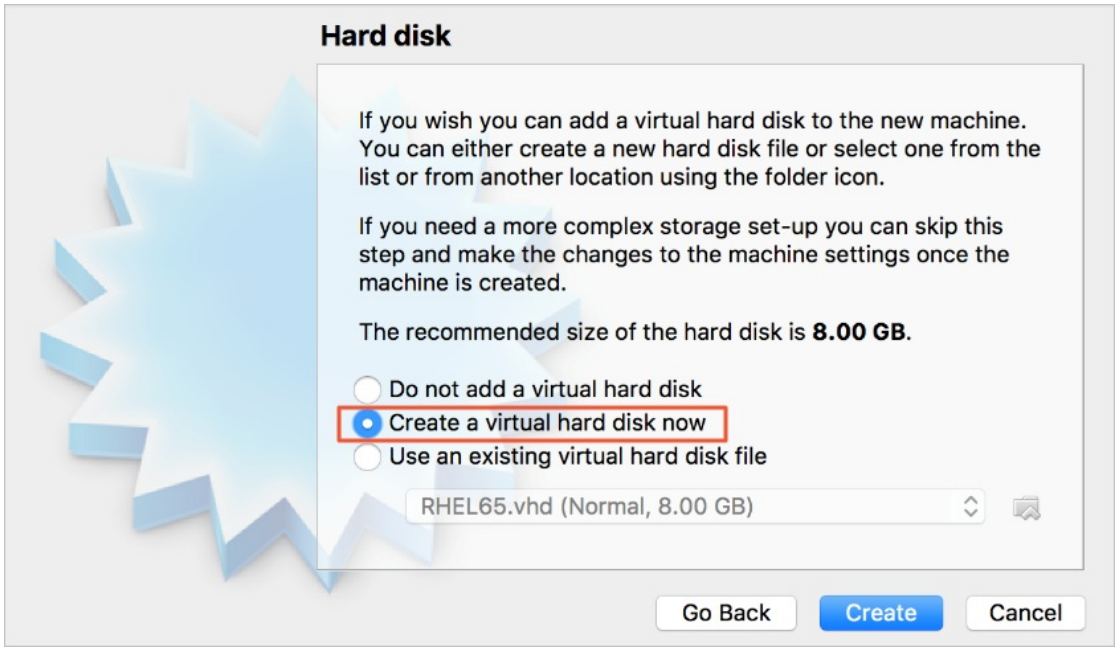
1. 在VirtualBox上创建一个新虚拟机（VM）。
   i. 启动VirtualBox并单击**New**。
   ii. 输入虚拟机的名称并选择正确的操作系统类型（例如**Microsoft Windows**）和操作系统版本（例如**Windows 7 (64-bit)**）。



iii. 设置内存大小。



iv. 创建一个虚拟硬盘。

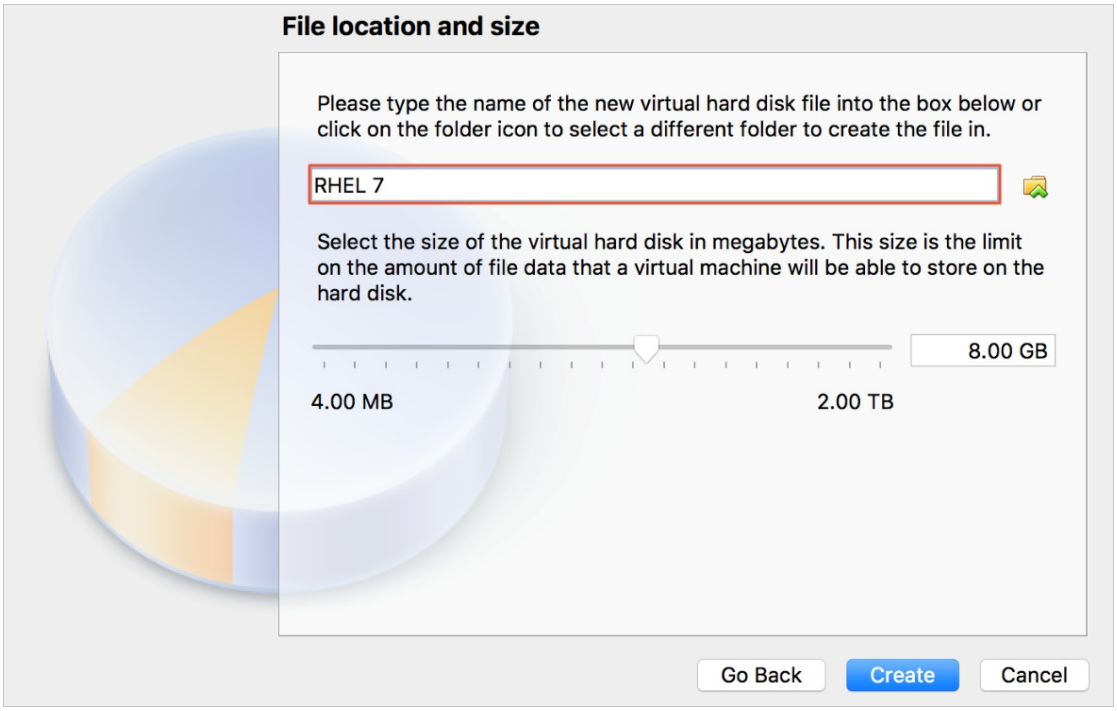


v. 硬盘文件类型选择VHD (Virtual Hard Disk)。

阿里云支持RAW、VHD和qcow2格式。

vi. 存储类型选择Dynamically allocated。

vii. 输入虚拟硬盘的名称，然后单击Create。



viii. 等待VHD文件创建完毕，然后双击启动新创建的虚拟机。

2. 运行以下命令，安装KVM虚拟化平台驱动程序。

```
yum install qemu-kvm qemu-img libvirt
```

3. 运行以下命令，禁用虚拟机的防火墙。

```
service firewalld stop
```

4. 将VHD文件上传到OSS，然后使用OSS浏览器将VHD文件上传到需要创建实例的地域。

   更多信息，请参见快速使用ossbrowser。

5. 导入自定义镜像。
   i. 登录ECS管理控制台。
   ii. 在左侧导航栏，单击**实例与镜像 > 镜像**。
   iii. 选择将VHD文件上传到OSS时所选的地域，然后单击**导入镜像**。

   有关如何允许ECS访问OSS资源的信息，请参见导入自定义镜像。

iv. 设置相关参数，然后单击**确定**。

> **说明** 您可以登录OSS控制台以获取OSS Object地址，更多信息请参见下载文件。

导入的自定义镜像将显示在镜像列表中。

18.监控

18.1. 使用云监控功能监控网站环境（部署于ECS实例）

合理的监控设置能减轻云上业务的运维成本和压力。设置合理的监控可以让您实时了解系统业务的运行情况，并能帮助您提前发现问题，避免可能会出现的业务故障。同时，告警机制能让您在故障发生后第一时间发现问题，缩短故障处理时间，以便尽快恢复业务。

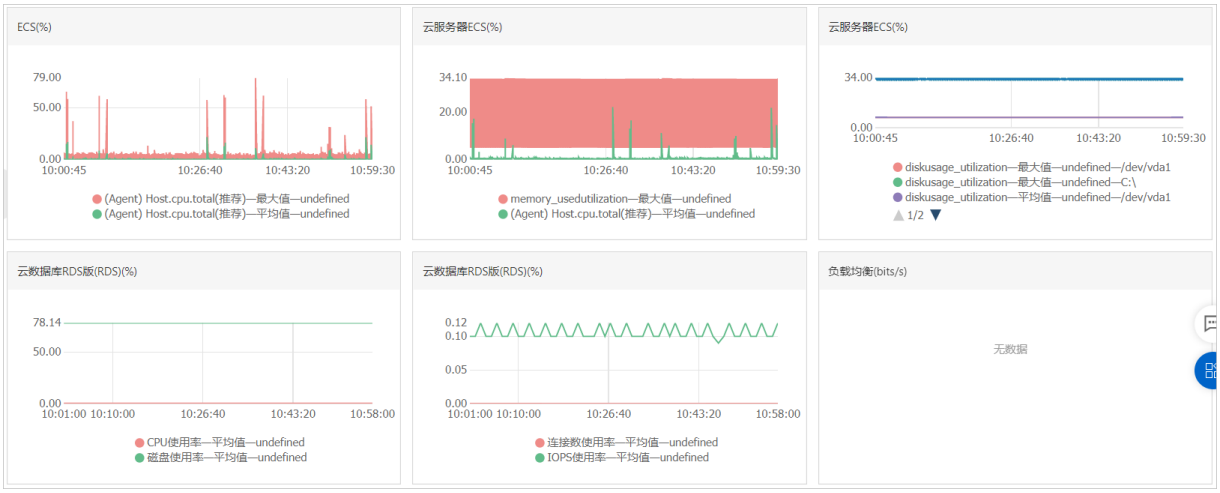
前提条件

在开始设置云监控前，您需要完成以下操作：

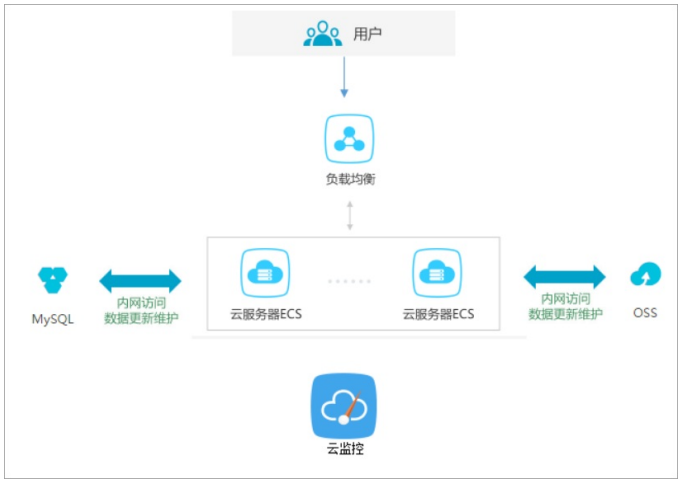
- 检查ECS监控插件运行情况，确保监控信息能够正常采集。如果安装失败则需要手动安装插件，具体操作，请参见云监控插件安装指南。
- 提前添加报警联系人和联系组，建议设置至少2人以上的联系人，互为主备，以便及时响应监控告警。监控选项的设定说明，请参见报警服务和云服务资源使用概览和报警概览。

背景信息

利用云监控的Dashboard功能为业务系统的云资源设置监控总览，方便随时检查整个业务系统资源的健康状态。为了更好地展示监控信息，本文中将ECS实例的CPU、内存和磁盘的使用率单独分组展示，将RDS实例的四项指标分两组展示。



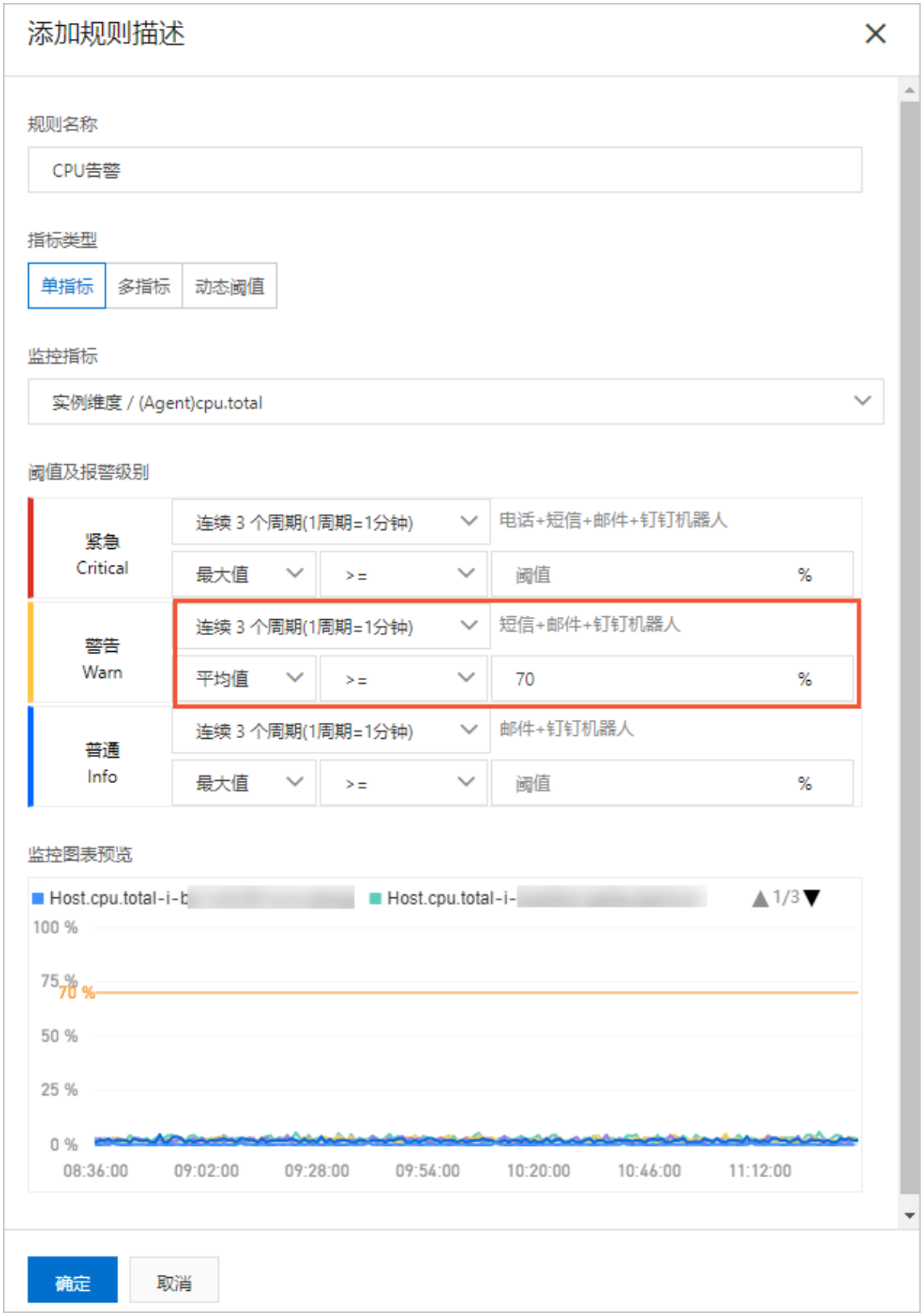
本文中以一个网站为示例，介绍如何配置使用云监控。本示例中，使用的云产品包括ECS、RDS、OSS和负载均衡。



设置报警阈值和报警规则

建议您根据实际业务情况设置各项监控指标的报警阈值。阈值太低会频繁触发报警，影响监控服务体验。阈值太高，在触发阈值后没有足够的预留时间来响应和处理告警。

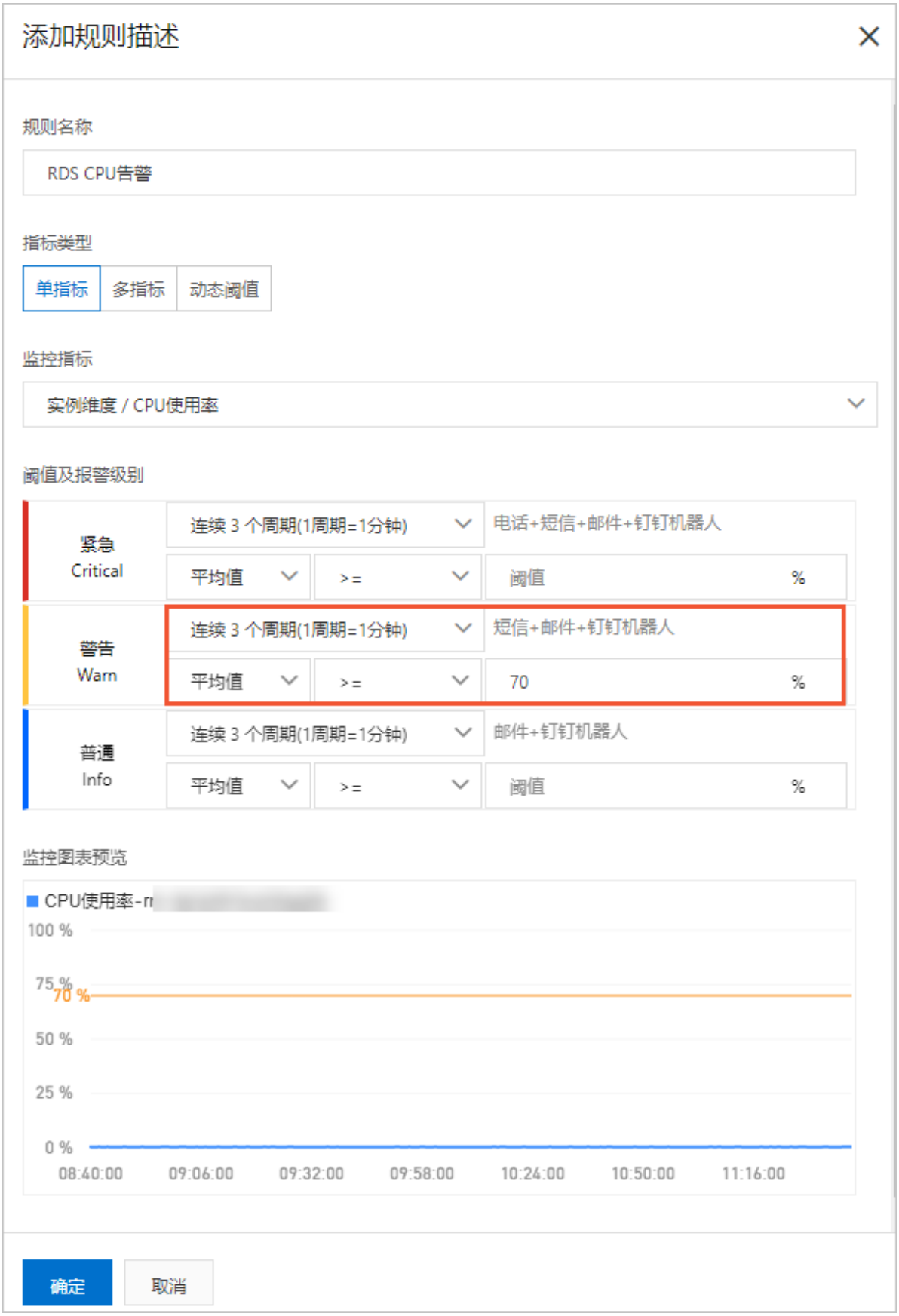
以CPU使用率为例，需要给服务器预留部分处理性能保障服务器正常运行，建议根据实际业务情况为您需要监控的ECS实例设置报警规则，例如CPU告警阈值为70%，连续三次超过阈值后开始报警。



如果您还需要设置其他资源的报警规则，单击**添加报警规则**，继续设置内存、磁盘的报警规则和报警通知人。示例如下：

- 设置RDS监控

  建议根据实际情况为需要监控的RDS实例设置报警规则，例如将RDS的CPU使用率报警阈值设置为70%，连续三次超过阈值后开始报警。您可以根据实际情况设置硬盘使用率、IOPS使用率、连接数等其他监控项。查看更多监控项信息的方法，请参见云产品监控。

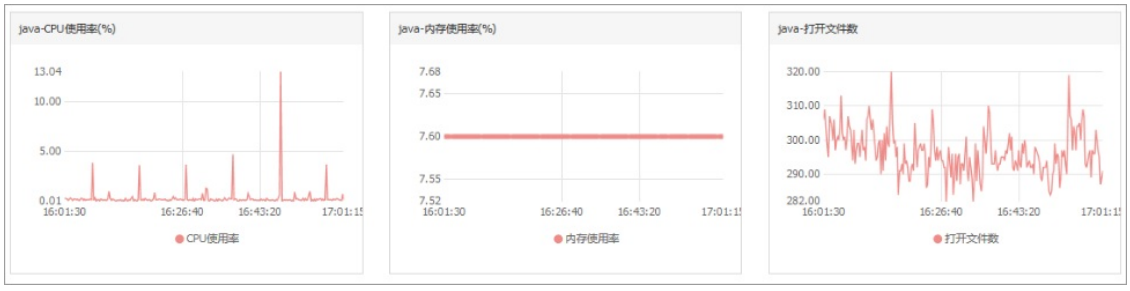

添加规则描述
规则名称
RDS CPU告警
指标类型
单指标
多指标
动态阈值
监控指标
实例维度 / CPU使用率
阈值及报警级别
紧急
Critical
连续 3 个周期(1周期=1分钟)
电话+短信+邮件+钉钉机器人
平均值
>=
阈值
%
警告
Warn
连续 3 个周期(1周期=1分钟)
短信+邮件+钉钉机器人
平均值
>=
70
%
普通
Info
连续 3 个周期(1周期=1分钟)
邮件+钉钉机器人
平均值
>=
阈值
%
监控图表预览
CPU使用率-
100 %
75 %
70 %
50 %
25 %
0 %
08:40:00
09:06:00
09:32:00
09:58:00
10:24:00
10:50:00
11:16:00
确定
取消

- 设置负载均衡监控

为了更好使用负载均衡的云监控服务，您需要先开启负载均衡的健康检查，并根据实际情况设置报警规则，例如将负载均衡带宽值的告警阈值设置为7 Mbit/s。

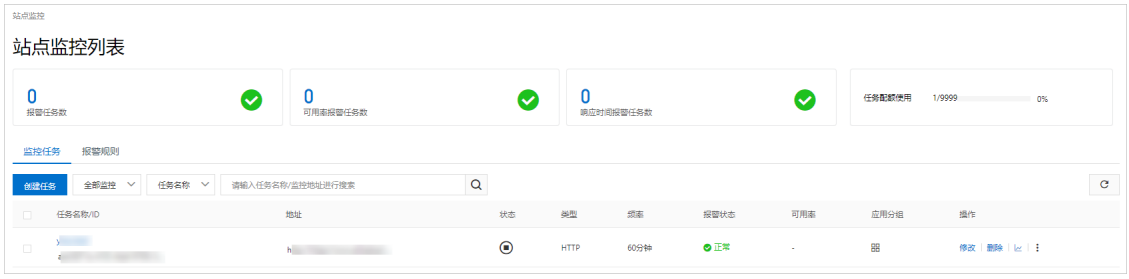
设置进程监控

对于常见的Web应用，设置进程监控，不仅可以实时监控应用进程的运行情况，还有助于排查处理故障。具体操作，请参见添加进程监控。



设置站点监控

在云服务器外层的监控服务，站点监控主要用于模拟真实用户访问情况，实时测试业务可用性，有助于排查处理故障。



如果以上监控选项不能满足您的实际业务监控需求，您可以使用自定义监控。更多信息，请参见概览。

18.2. 通过Prometheus监控自发现云服务器ECS

Prometheus监控提供了自发现功能，可以自动感知资源的创建和删除操作，并自动加入或者剔除监控列表。除了支持原生自带的自动发现（例如Consul、DNS等），还支持自动发现云服务器ECS，方便您更快捷地自动部署和配置ECS。本文介绍如何通过Prometheus监控自发现云服务器ECS。

背景信息

概念

阿里云Prometheus监控全面对接开源Prometheus生态，支持类型丰富的组件监控，提供多种开箱即用的预置监控大盘，且提供全面托管的Prometheus服务。更多信息，请参见什么是Prometheus监控。

自发现功能

Prometheus监控是云原生领域流行的监控组件，但将各资源加入Prometheus监控项却并非易事，尤其在大规模云环境下，资源的创建和删除越来越频繁，手动维护成本也越来越高。为了能及时地将资源配置到Prometheus监控中，避免运维人员手动变更引起的出错问题或者由于人为操作的延迟导致监控数据不全的缺陷，Prometheus监控提供了自动发现机制。

自发现功能会周期性从自动发现组件中获取最新的监控对象，并启动对应的监控数据采集任务。您可以在Prometheus上配置自动发现云服务器ECS功能，Prometheus会通过阿里云的API接口周期性获取最新的ECS列表，当您手动或者通过弹性伸缩（Auto Scaling）新建ECS实例后，无需运维人员手动操作，便可以自动加入Prometheus监控，从而极大地节省了运维成本。

注意事项

在跨VPC场景中，默认情况下VPC之间的网络是不连通的，建议开启云企业网来打通VPC，更多信息，请参见什么是云企业网。或者您也可以通过将ECS实例绑定公网IP方式，从公网获取监控指标，即通过修改 `Prometheus.yaml` 文件中的 `relabel_configs` 参数（将 `__address__` 修改为 `__meta_ecs_public_ip` ）来实现。

步骤一：安装阿里云Prometheus监控

阿里云Prometheus监控的安装非常方便，您可以通过以下三种常见方式进行安装。

方式一：通过二进制方式安装

命令代码如下所示：

```
# 下载二进制
wget 'http://arms-public.oss-cn-hangzhou.aliyuncs.com/prometheus-community/ecs_sd/prometheus' -O prometheus
# 添加执行权限
chmod 755 ./prometheus
# 启动服务
./prometheus --config.file="/root/test/prometheus.yaml"
```

方式二：通过Docker容器方式安装

使用该安装方式时，您需要提前在云服务器ECS上安装Docker，更多信息，请参见Install Docker Engine on CentOS。

命令代码如下所示：

```
# 创建配置文件目录
mkdir /root/aliprometheus
# 启动容器
docker run -d -p9090:9090 -v /root/aliprometheus:/etc/prometheus/ registry.cn-hangzhou.aliyuncs.com/public-community/prometheus-alibaba:v0.2
```

主要参数说明：

参数	说明
`-d`	表示参考后台进程来启动。
`-p`	表示端口映射。
`-v`	表示挂载主机目录，这里的目录挂载并非必须，主要是为了便于后期维护配置文件。

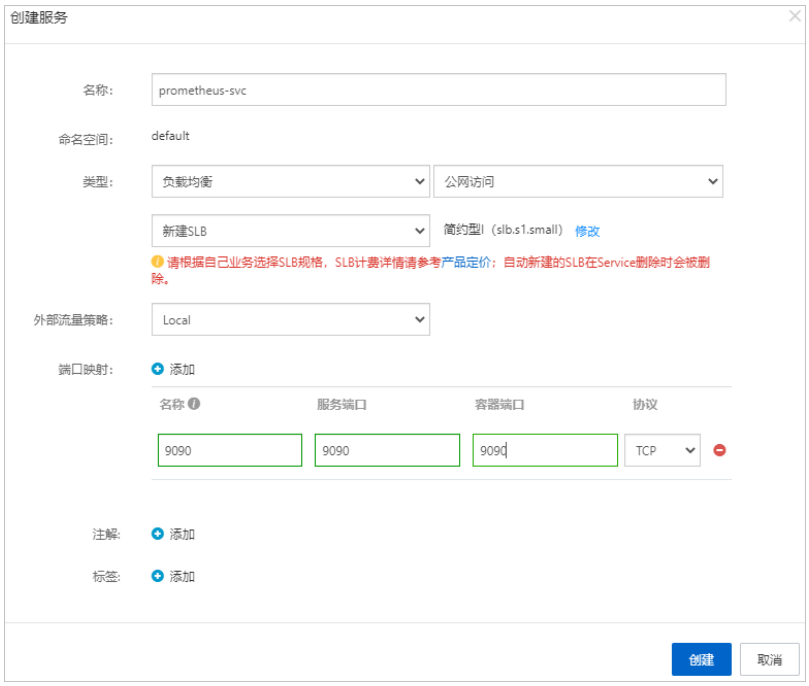
方式三：通过阿里云kubernetes集群方式安装

1. 登录容器服务管理控制台。
2. 在**集群列表**页面，找到已创建的kubernetes托管版集群，单击对应**操作**列的**详情**。
3. 在左侧导航栏，选择**工作负载 > 无状态**。
4. 在无状态（Deployment）页面，单击**使用镜像创建**。
5. 在**容器配置**页面下**基本配置**区域，单击**选择镜像**。
6. 在弹出的**镜像选择**面板，单击**搜索**页签，并选择阿里云Prometheus镜像。

   选择镜像所属的地域，例如华东1（杭州），然后选择**Aliyun镜像**并在输入框中输入**prometheus**，单击**搜索**。选择搜索到的Prometheus镜像后，单击**确定**。

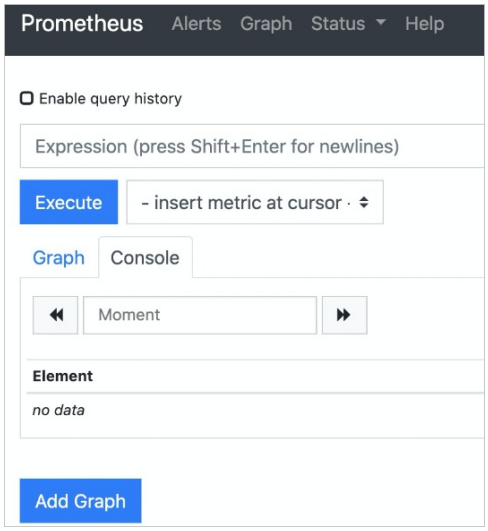
7. 在**高级配置**页面下的**访问设置**区域，单击**服务(Service)**后的**创建**。

   在弹出的**创建服务**面板中，创建外部访问负载均衡，便于后期从公网访问Prometheus监控。具体参数配置如下图所示：



8. 在无状态（Deployment）页面，单击已创建的prometheus监控进入prometheus详情页，然后单击**访问方式**页签。
9. 在**服务(Service)**区域，单击应用服务对应**外部端点**列下的访问地址，通过公网直接访问Prometheus服务。

访问结果如下所示：



步骤二：配置Prometheus监控动态自发现功能

对于上述三种安装方式，Prometheus都需要加载 `prometheus.yaml` 配置文件才可以正常启动，Prometheus动态自发现配置文件如下所示。您可以直接复制该内容并保存为 `prometheus.yaml` 文件。关于Prometheus的更多配置，请参见Prometheus Introduction。

```
global:
  scrape_interval: 15s
  scrape_timeout: 10s
  evaluation_interval: 30s
scrape_configs:
- job_name: _aliyun-prom/ecs-sd
  honor_timestamps: true
  scrape_interval: 30s
  scrape_timeout: 10s
  metrics_path: /metrics
  scheme: https
  aliyun_sd_configs:
    - port:  9100                          # 服务发现后的prometheus抓取采集点port
      user_id: <aliyun userId>             # Aliyun用户身份表示userId，填写会为discovery target
带上__meta_ecs_user_id的label，可不填写
      refresh_interval: 30s                # 列表刷新时间
      region_id: cn-hangzhou               # 设置获取ECS的regionId
      access_key: ****                     # Aliyun鉴权字段AK
      access_key_secret: ****              # Aliyun鉴权字段SK
#      tag_filters:                        # Aliyun ECS tag filter，按tagKey tagValue匹配筛选实
例
#       - key: 'testK'
#         values: ['*', 'test1*']
#       - key: 'testM'
#         values: ['test2*']
#      limit: 40                                    # 从接口取到的最大实例个数限制，不填为获取所有e
cs实例
# relabel为可选配置，用于手动筛选实例
  relabel_configs:
#   1. 手动设置使用ECS的哪种IP
#   默认ECS会按经典网络公网IP>经典网络内网IP>VPC网络公网IP>VPC网络内网IP的顺序查找并赋予此ECS的采集I
P，此时的采集点port为aliyun_sd_configs.port设置
```

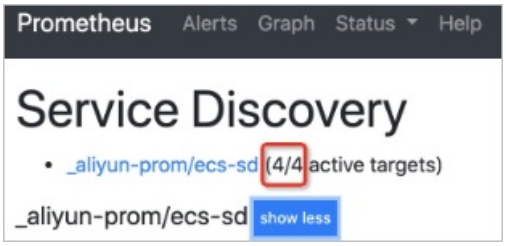
```
#   用户可用过一下relabel设置，手动设置ECS的采集IP
#    - source_labels: [__meta_ecs_public_ip]       # 经典网络公网ip __meta_ecs_public_ip
#    - source_labels: [__meta_ecs_inner_ip]        # 经典网络内网ip __meta_ecs_inner_ip
#    - source_labels: [__meta_ecs_eip]             # VPC网络公网ip __meta_ecs_eip
#    - source_labels: [__meta_ecs_private_ip]      # VPC网络内网ip __meta_ecs_private_ip
#      regex: (.*)
#      target_label: __address__
#      replacement: $1:<port>                      # 注意此处为手动设置relabel时的采集port
#   2. 按ECS属性过滤 keep为只保留此条件筛选到的target，drop为过滤掉此条件筛选到的target
#   __meta_ecs_instance_id          实例id
#   __meta_ecs_region_id            实例regionId，注意配置中aliyun_sd_configs.region_id决定了
获取的ECS的regionId
#   __meta_ecs_status               实例状态Running：运行中、实例状态Starting：启动中、实例状态St
opping：停止中、实例状态Stopped：已停止
#   __meta_ecs_zone_id              实例区域id
#   __meta_ecs_network_type         实例网络类型classic：经典网络、实例网络类型vpc：专有网络
#   __meta_ecs_tag_<TagKey>         实例tag TagKey为tag的名
    - source_labels:  ["__meta_ecs_instance_id"]
      regex: ".+"       # or other value regex
      action: keep      # keep / drop
    - source_labels: ["__scheme__"]
      target_label: "__scheme__"
      replacement: "http"
```

注意事项如下所示：

- 如果使用方式三：阿里云kubernetes集群安装方式，您需要先通过 `promethues.yaml` 创建 `Configmap` ，然后将 `Configmap` 挂载到Pometheus的*/etc/prometheus*路径下。
- 上述配置文件中的 `tag_filters` 支持根据指定的标签筛选特定的ECS实例加入Pometheus监控。
- 如果是通过RAM方式，则不需要配置AK/SK，但为了能够动态获取ECS实例列表以及Tag，需要在RAM配置访问权限，配置权限策略如下所示：

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                  "ecs:DescribeInstances",
                  "ecs:ListTagResources"
          ],
            "Resource": "*"
        }
    ],
    "Version": "1"
}
```

更新 `prometheus.yaml` 配置文件后，请您重新加载Prometheus，例如通过给Prometheus发送SIGHUP信号或者发送HTTP POST请求（即Prometheus IP/-/reload）。配置重新加载后，您可以在Prometheus的**Service Discovery**页面看到，Prometheus已经将4台ECS实例加入监控列表。



步骤三：为ECS安装采集客户端

虽然您已成功配置了Prometheus动态自发现功能，如果由于目前ECS上还未安装采集客户端，则所有的监控对象（Targets）都处于**DOWN**状态，如下所示：

1. 安装Node Exporter采集客户端。

   本步骤以安装Node Exporter采集客户端为例，该采集客户端可以收集主机的性能指标，例如CPU使用率、内存用量、磁盘读写性能等，通过在ECS上安装Node Exporter来端到端地测试自发现功能。Node Exporter安装流程如下所示：

   - 通过二进制方式安装Node Exporter

     示例代码如下所示：

```
# 下载二进制
cd /tmp
curl -LO https://github.com/prometheus/node_exporter/releases/download/v0.18.1/node_exporter-0.18.1.linux-amd64.tar.gz
# 解压
tar -xvf node_exporter-0.18.1.linux-amd64.tar.gz
mv node_exporter-0.18.1.linux-amd64/node_exporter /usr/local/bin/
# 添加启动用户
useradd -rs /bin/false node_exporter
# 配置/etc/systemd/system/node_exporter.service，内容如下：
[Unit]
Description=Node Exporter
After=network.target
[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter
[Install]
WantedBy=multi-user.target
# 启动服务并加入开机自启动
systemctl daemon-reload
systemctl start node_exporter
systemctl enable node_exporter
```

- 通过Docker方式安装Node Exporter

  示例代码如下所示：

```
docker run -d \
  --net="host" \
  --pid="host" \
  -v "/:/host:ro,rslave" \
  quay.io/prometheus/node-exporter \
  --path.rootfs=/host
```

2. 开放安全组规则，允许**入方向**的 `9100` 端口访问。

   关于安全组规则的相关内容，请参见安全组概述。

   操作完成后，查看Prometheus target列表，如果监控对象已处于**UP**状态，并且可以查看到ECS的CPU load指标如下所示。表示后续无论是新创建ECS还是之前ECS需要移除时，在无需运维人员操作的情况下，Prometheus监控都可以自动完成监控对象的管理。

相关文档

- 什么是Prometheus监控
- 什么是云企业网
- 安全组概述

19.使用实例RAM角色访问其他云产品

本文以部署在ECS实例上的Python访问OSS为例，详细介绍了如何借助ECS实例RAM角色，使实例内部的应用程序可以使用STS临时凭证访问其他云产品。

前提条件

已创建一个ECS实例，详情请参见使用向导创建实例。

背景信息

以往部署在ECS实例中的应用程序如果需要访问阿里云其他云产品，您通常需要借助AccessKeyID和AccessKeySecret（下文简称AK）来实现。AK是您访问阿里云API的密钥，具有相应账号的完整权限。为了方便应用程序对AK的管理，您通常需要将AK保存在应用程序的配置文件中或以其他方式保存在ECS实例中，这在一定程度上增加了AK管理的复杂性，并且降低了AK的保密性。甚至，如果您需要实现多地域一致性部署，AK会随着镜像以及使用镜像创建的实例扩散出去。这种情况下，当您需要更换AK时，您就需要逐台更新和重新部署实例和镜像。

现在借助于ECS实例RAM角色，您可以将RAM角色和ECS实例关联起来，实例内部的应用程序可以通过STS临时凭证访问其他云产品。其中STS临时凭证由系统自动生成和更新，应用程序可以使用指定的实例元数据URL获取STS临时凭证，无需特别管理。同时借助于RAM，通过对角色和授权策略的管理，您可以达到不同实例对不同云产品或相同云产品具有各自访问权限的目的。

> **注意** 为了方便您随本文样例快速入门，文档里所有操作均在OpenAPI开发者门户完成。OpenAPI Explorer通过已登录用户信息获取当前账号临时AK，对当前账号发起线上资源操作，请谨慎操作。创建实例操作会产生费用。操作完成后请及时释放实例。

操作步骤

为了使ECS借助实例RAM角色，实现内部Python可以使用STS临时凭证访问OSS，您需要完成以下步骤：

1. 步骤一：创建RAM角色并配置授权策略
2. 步骤二：指定RAM角色创建并设置ECS实例
3. 步骤三：在实例内部访问实例元数据URL获取STS临时凭证
4. 步骤四：基于临时凭证，使用Python SDK访问OSS

步骤一：创建RAM角色并配置授权策略

完成以下步骤，创建RAM角色并配置授权策略：

1. 创建RAM角色。

   找到OpenAPI开发者门户RAM产品下CreateRole API。其中：

   - RoleName：设置角色的名称。根据自己的需要填写，本示例中为 *EcsRamRoleTest*。
   - AssumeRolePolicyDocument： 填写如下内容，表示该角色为一个服务角色，受信云服务（本示例中为ECS）可以扮演该角色。

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

2. 创建授权策略。

   找到OpenAPI开发者门户RAM产品下的CreatePolicy API。其中：

   - PolicyName：设置授权策略的名称。本示例中为*EcsRamRolePolicyTest*。
   - PolicyDocument：输入授权策略内容。本示例中填写如下内容，表示该角色具有OSS只读权限。

```
{
    "Statement": [
        {
            "Action": [
                "oss:Get*",
                "oss:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ],
    "Version": "1"
}
```

3. 为角色附加授权。找到OpenAPI开发者门户RAM产品下的AttachPolicyToRole API。
   - PolicyType：填写*Custom*。
   - PolicyName：填写第2步创建的策略名称，如本示例中的*EcsRamRolePolicyTest*。
   - RoleName：填写第1步创建的角色名称，如本示例中的*EcsRamRoleTest*。

步骤二：指定RAM角色创建并设置ECS实例

您可以通过以下任一种方式为ECS实例指定RAM角色：

- 将实例RAM角色附加到一个已创建的ECS实例上

  您可以使用ECS的AttachInstanceRamRole API附加实例RAM角色到已有的VPC类型ECS实例授权访问，设置信息如下：

  - RegionId：为实例所在的地域ID。
  - RamRoleName：RAM角色的名称。本示例中为*EcsRamRoleTest*。
  - InstanceIds：需要附加实例RAM角色的VPC类型ECS实例ID。本示例中为*["i-bXXXXXXXX"]*。

- 指定RAM角色创建并设置ECS实例

按以下步骤指定RAM角色创建并设置ECS实例。

i. 创建实例。

找到OpenAPI开发者门户ECS产品下的CreateInstance API，根据实际情况填写请求参数。必须填写的参数包括：

- RegionId：实例所在地域。本示例中为*cn-hangzhou*。
- ImageId：实例的镜像。本示例中为*centos_7_03_64_40G_alibase_20170503.vhd*。
- InstanceType：实例的规格。本示例中为*ecs.g6.large*。
- VSwitchId：实例所在的VPC虚拟交换机。因为ECS实例RAM角色目前只支持VPC类型ECS实例，所以VSwitchId是必需的。
- RamRoleName：RAM角色的名称。本示例中为*EcsRamRoleTest*。

如果您希望授权子账号创建指定RAM角色的ECS实例，那么子账号除了拥有创建ECS实例的权限之外，还需要增加PassRole权限。所以，您需要创建一个如下所示的自定义授权策略并绑定到子账号上。

- 如果是创建ECS实例，[ECS RAM Action]可以是`ecs:CreateInstance`，您也可以根据实际情况添加更多的权限。
- 如果您需要为子账号授予所有ECS操作权限，[ECS RAM Action]应该替换为`ecs:*`。

> **说明** [ECS RAM Action]的取值详情请参见鉴权规则。

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

ii. 设置密码并启动实例。

iii. 使用API或在控制台设置ECS实例能访问公网。

步骤三：在实例内部访问实例元数据URL获取STS临时凭证

完成以下步骤，获取实例的STS临时凭证：

> **注意** STS临时凭证失效前半小时会生成新的STS临时凭证，在这半小时内，新旧STS临时凭证均可使用。

1. 远程连接ECS实例。

关于连接方式的介绍，请参见连接方式概述ECS远程连接操作指南。

2. 访问 `http://100.100.100.200/latest/meta-data/ram/security-credentials/EcsRamRoleTest` 获取STS临时凭证。路径最后一部分是RAM角色名称，您应当替换为自己创建的RAM角色名称。

   > **说明** 本示例中使用 `curl` 命令访问上述URL。如果您使用的是Windows ECS实例，请参见实例元数据。

   示例输出结果如下。

```
[root@local ~]# curl http://100.100.100.200/latest/meta-data/ram/security-credentials/EcsRamRoleTest
{
"AccessKeyId" : "STS.J8XXXXXXXXX4",
"AccessKeySecret" : "9PjfXXXXXXXXXBf2XAW",
"Expiration" : "2017-06-09T09:17:19Z",
"SecurityToken" : "CAIXXXXXXXXXXXwmBkleCTkyI+",
"LastUpdated" : "2017-06-09T03:17:18Z",
"Code" : "Success"
}
```

步骤四：基于临时凭证，使用Python SDK访问OSS

本示例中，我们基于STS临时凭证使用Python SDK列举实例所在地域的某个OSS存储空间（Bucket）里的10个文件。

前提条件：

- 您已经远程连接到ECS实例。
- 您的ECS实例已经安装了Python。如果您用的是Linux ECS实例，必须安装pip。
- 您在实例所在的地域已经创建了存储空间（Bucket），并已经获取Bucket的名称和Endpoint。本示例中，Bucket名称为 `ramroletest`，Endpoint为 `oss-cn-hangzhou.aliyuncs.com`。

完成以下步骤，使用Python SDK访问OSS：

1. 运行命令 `pip install oss2`，安装OSS Python SDK。

   > **说明** 如果您用的是Windows ECS实例，请参见 *对象存储 OSS SDK 参考*的安装 Python SDK。

2. 执行下述命令进行测试。

```
import oss2
from itertools import islice
auth = oss2.StsAuth(<AccessKeyId>, <AccessKeySecret>, <SecurityToken>)
bucket = oss2.Bucket(auth, <您的 Endpoint>, <您的 Bucket 名称>)
for b in islice(oss2.ObjectIterator(bucket), 10):
  print(b.key)
```

   其中：

   - `oss2.StsAuth` 中的3个参数分别对应于步骤三中返回的AccessKeyId、AccessKeySecret和SecurityToken。
   - `oss2.Bucket` 中后2个参数是Bucket的名称和Endpoint。

   示例输出结果如下。

```
[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.J8XXXXXXXXX4", "9PjfXXXXXXXXXBf2XAW", "CAIXXXXXXXXXXXwmBk
leCTkyI+")
>>> bucket = oss2.Bucket(auth, "oss-cn-hangzhou.aliyuncs.com", "ramroletest")
>>> for b in islice(oss2.ObjectIterator(bucket), 10):
...     print(b.key)
...
ramroletest.txt
test.shh
```

20.网络

20.1. 配置公网带宽最佳实践

公网带宽的计费方式包括按固定带宽和按使用流量两种。如果您的业务需求变化较大，使用按固定带宽计费的模式不能满足业务需求时，您可以通过转换带宽计费方式来限制最大网络带宽，以满足您的业务需求，同时节省费用成本。本文为您介绍公网带宽在不同业务场景下的计费模式和配置方式，以及通过转换计费模式提升带宽限制的方法。

前提条件

如果您需要将包年包月实例的公网带宽从固定带宽转为使用流量，请先确认实例所在的账号具有实时降配的功能特权。

> **说明** 您可以在ECS管理控制台的**概览**页面，单击**权益配额**，然后查看账号是否具有实时降配的功能特权。

背景信息

根据不同的业务场景，公网带宽的计费模式包括按固定带宽和按使用流量两种。详细说明如下表所示：

业务场景	带宽计费模式	带宽指定情况	计费情况
适用于对网络带宽要求比较稳定且费用较低可节省成本的业务场景。	按固定带宽	需指定公网出方向的带宽的大小，例如10 Mbps。使用过程中，实际的出网带宽不会高于指定的带宽值。	按固定带宽值（单位为Mbps）采用阶梯计费，带宽费用合并在ECS实例中收取。公网带宽的详细计费信息，请参见公网带宽计费。
适用于对网络带宽需求变化较大的业务场景，例如平时带宽使用较低但会间歇性出现网络访问高峰的场景。	按使用流量	需指定公网出方向的带宽峰值，最大可设置为100 Mbps。为了防止突然爆发的流量产生较高的费用，您可以指定容许的最大网络带宽进行限制。使用过程中，实际的出网流量不会高于指定的带宽峰值。 **说明** 公网带宽限制由ECS实例和公网带宽的计费方式决定，更多信息，请参见公网带宽。	按公网出方向的实际发生的网络流量（单位为GB）进行收费，是一种后付费模式，每小时整点结算。公网带宽的详细计费信息，请参见公网带宽计费。

根据不同的业务场景，公网带宽涉及的配置方式如下表所示：

业务场景	配置方式	操作方法

业务场景	配置方式	操作方法
当您需要通过公网IP地址访问公网时，您可以配置公网及公网带宽。	配置公网及公网带宽	• 您可以直接在创建ECS实例时，配置公网IP地址和公网带宽。更多信息，请参见步骤一：创建ECS实例时配置公网IP带宽。 • 如果创建ECS实例没有分配公网IP地址，您可以变更实例公网带宽，从而分配一个固定公网IP地址。更多信息，请参见升降配方式概述。
如果当前带宽计费方式不满足需求，您可以转换带宽计费方式。	转换带宽计费模式	具体操作，请参见转换公网带宽计费方式。
如果您已配置的公网带宽无法满足或者超出业务需求时，您可以修改公网带宽。	修改公网带宽	• 包年包月的ECS实例：具体操作，请参见包年包月实例修改带宽。 • 按量付费的ECS实例：具体操作，请参见按量付费实例修改带宽。
如果需要在某个连续的时间段内或者周期性地提升公网带宽，您可以临时升级带宽。	升级带宽	仅限于包年包月的实例。具体操作，请参见： • 临时升级带宽（连续时间段） • 临时升级带宽（按日周期性）
当弹性公网带宽无法满足或者超出业务需求时，您可以调整EIP带宽峰值和计费方式。	修改EIP带宽	具体操作，请参见变更EIP带宽。

公网带宽其他的一些常见问题，请参见网络FAQ。

本教程为您介绍如何在创建ECS实例时配置公网带宽，以及在需要限制最大网络带宽时，如何转换带宽的计费方式并设置带宽峰值。

步骤一：创建ECS实例时配置公网IP带宽

创建ECS实例，并分配按固定带宽计费方式的公网IP地址。

1. 登录ECS管理控制台。
2. 在左侧导航栏，选择**实例与镜像 > 实例**。
3. 在顶部菜单栏左上角处，选择地域。
4. 在**实例列表**页面，单击**创建实例**。
5. 按照界面向导完成各项配置。

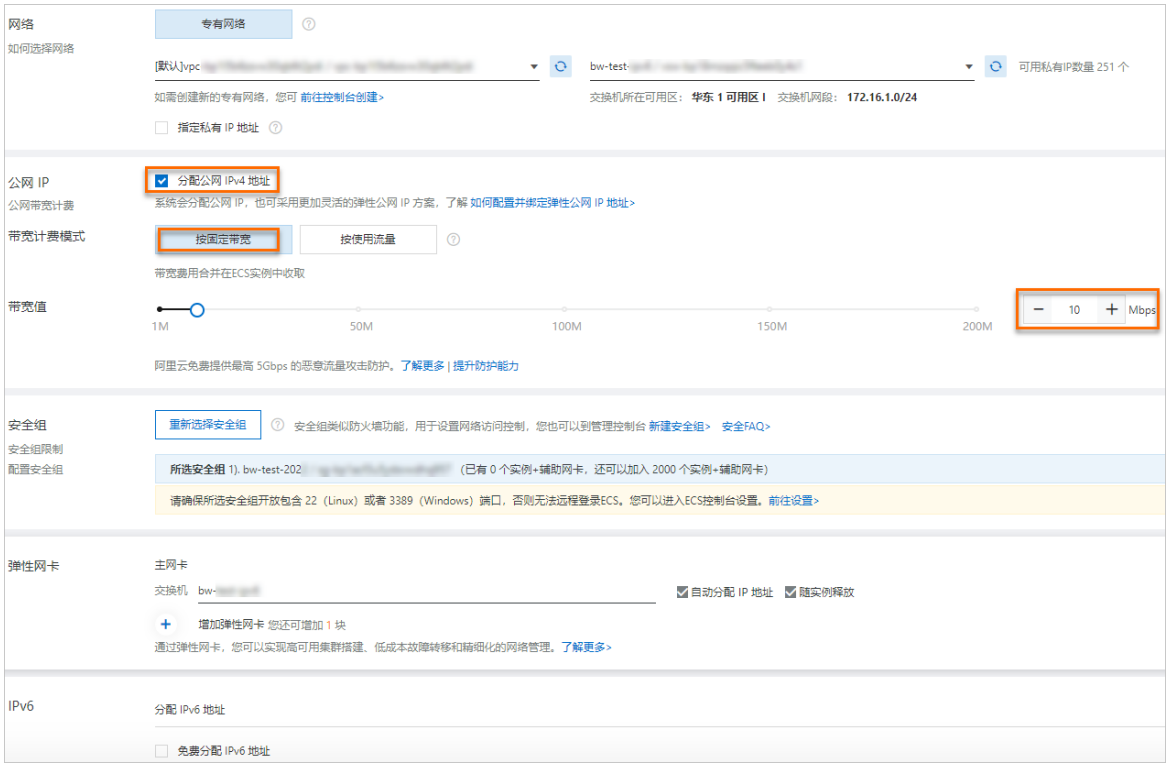
   详细的操作步骤及参数配置，请参见使用向导创建实例。

   在本教程中，关于网络部分主要的配置参数说明如下：

   - 网络：选择专有网络类型。
     - 选择已有的专有网络和交换机。
     - 选中**指定私有 IP 地址**，可以指定具体的私有IP地址。
   - 公网IP：选中**分配公网 IPv4 地址**。
   - 带宽计费模式：选择**按固定带宽**。

- 带宽值：配置该公网的带宽大小，例如10 Mbps。您在使用过程中，系统按指定的10 Mbps收费，且实际的出网带宽流量不会高于10 Mbps。
- 安全组：为该实例选择一个安全组，用于控制安全组内实例的入流量和出流量。
- 弹性网卡：主网卡不支持从实例解绑，只能随实例一起创建和释放。如需随实例一起创建辅助网卡，请单击+图标，然后选择辅助网卡所属的交换机。
- （可选）IPv6：根据需要配置IPv6。

详细的参数说明，请参见网络和安全组配置。



实例创建成功后，在ECS实例页面的**配置**列会显示已设置的固定带宽值。

步骤二：计费模式按固定带宽转为按使用流量

当业务需求变化较大需要限制最大网络带宽时，您可以转换带宽的计费方式并设置带宽峰值，以避免突然爆发的流量产生较高的费用。

1. 登录ECS管理控制台。
2. 在左侧导航栏，选择**实例与镜像 > 实例**。
3. 在顶部菜单栏左上角处，选择地域。
4. 找到待转换的实例，根据实例计费方式选择相应的方式进入配置页面。
   - 包年包月实例
     a. 找到待转换的实例，在**操作**列单击**升降配**。
     b. 在弹出的对话框中，单击**降低配置 > 降低带宽配置（支持修改带宽计费方式为按量付费）**。
     c. 单击**继续**。

> **注意** 如果您操作过带宽临时升级，确认按固定带宽转按使用流量会作废所有已生效和未生效的带宽临时升级订单并退款。

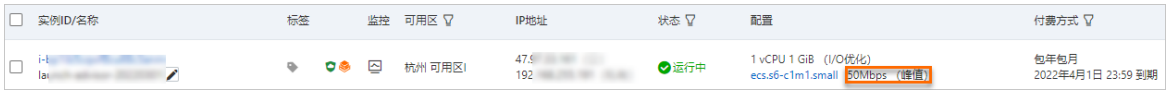
- 按量付费实例

  找到待转换的实例，在**操作**列单击**更多 > 资源变配 > 按量付费实例更改带宽**。

5. 找到公网带宽对应的配置项，选择**按使用流量**，并设置流量带宽峰值，例如50 Mbps。

   您在使用过程中，系统按实际使用的流量计费，且实际的出网带宽流量不会高于50 Mbps，以防突然爆发的流量产生较高费用。

6. 阅读下方注意事项和服务条款，如无问题，选中*云服务器ECS服务条款*。
7. 确认费用，单击右下方按钮，按页面提示完成后续操作。
   转换完成后，新配置立即生效。在ECS实例页面的**配置**列会显示已设置的带宽峰值，例如：50Mbps（峰值）。



> **说明** 如果您使用的是弹性公网IP，转换计费模式操作请参见变更EIP带宽。

20.2. 网络性能测试最佳实践

本文主要以ECS实例使用Netperf和sockperf工具测试网络性能的方法为例，为您介绍如何测试ECS实例的网络PPS、网络带宽和网络时延。

前提条件

- 已创建符合测试场景规格要求的ECS实例。具体操作，请参见使用向导创建实例。
- 在同一测试场景下，所有ECS实例必须所属同一个VPC、同一台交换机和同一个安全组。

> **注意**
> - 强烈建议您在新购买的无数据的ECS实例上使用工具测试网络性能，避免造成数据丢失。
> - 实例规格指标均在测试数据环境下验证获得。在真实场景中，受实例负载、组网模型等其他因素的影响，实例的性能表现可能存在差异，请您以实际情况为准。

准备环境

做网络性能测试（网络PPS、网络带宽和网络时延）的ECS实例作为本次的测试机和辅助测试机。测试机可作为Netperf或sockperf工具测试中的Client端或Server端。辅助测试机也可作为Netperf或sockperf工具测试中的Client端或Server端，与测试机之间建立控制连接，传递测试配置相关的信息。

不同测试场景下，实例的示例规格及推荐数量如下表所示。

> **说明** 测试ECS实例的网络带宽和网络时延对网络PPS大小没有要求，可任意选择实例规格进行测试。

- 测试ECS实例的网络PPS（小于600万）、网络带宽、网络时延

测试示例	测试机	辅助测试机
实例规格	ecs.g7.large	ecs.g7.large
镜像	Alibaba Cloud Linux 2.1903 LTS 64位	Alibaba Cloud Linux 2.1903 LTS 64位
数量	1	1

  - 测试网络PPS（小于600万）的具体操作，请参见测试网络PPS（小于600万）。
  - 测试网络带宽的具体操作，请参见测试网络带宽。
  - 测试网络时延的具体操作，请参见测试网络时延。

- 测试ECS实例的网络PPS（大于600万小于2000万）

测试示例	测试机	辅助测试机
实例规格	ecs.g7.16xlarge	ecs.g7.16xlarge
镜像	Alibaba Cloud Linux 2.1903 LTS 64位	Alibaba Cloud Linux 2.1903 LTS 64位
数量	1	3

  测试网络PPS（大于600万小于2000万）的具体操作，请参见测试网络PPS（大于600万小于2000万）。

- 测试ECS实例的网络PPS（大于2000万）

测试示例	测试机	辅助测试机
实例规格	ecs.g7.32xlarge	ecs.g7.32xlarge
镜像	Alibaba Cloud Linux 2.1903 LTS 64位	Alibaba Cloud Linux 2.1903 LTS 64位
数量	1	3

  测试网络PPS（大于2000万）的具体操作，请参见测试网络PPS（大于2000万）。

测试网络PPS（小于600万）

1. 分别远程连接测试机和辅助测试机。

   关于连接方法的介绍，请参见实例连接概述。

2. 分别在测试机和辅助测试机上执行以下命令，下载Netperf。

```
wget https://benchmark-packages.oss-cn-qingdao.aliyuncs.com/netperf-2.7.0.tar.gz
```

3. 分别在测试机和辅助测试机上执行以下命令，安装Netperf和sar监控工具。

   i. 执行以下命令，解压Netperf包。

   ```
   yum install -y gcc autoconf automake libtool sysstat
   tar -zxvf netperf-2.7.0.tar.gz
   ```

   ii. 执行以下命令，查询gcc版本号。

   ```
   gcc -v 2>&1
   ```

   iii. （可选）如果测试机和辅助测试机的gcc版本高于10，需先执行以下命令，打开并手动删除*nettest_omni.c*文件中的声明变量内容。

   ```
   cd netperf
   vim src/nettest_omni.c
   ```

   手动删除*nettest_omni.c*文件中如下声明变量内容。

   ```
   /* different options for the sockets        */
   int
     loc_nodelay,    /* don't/do use NODELAY  locally    */
     rem_nodelay,    /* don't/do use NODELAY remotely  */
     loc_sndavoid,    /* avoid send copies locally    */
     loc_rcvavoid,    /* avoid recv copies locally    */
     rem_sndavoid,    /* avoid send copies remotely    */
     rem_rcvavoid;   /* avoid recv_copies remotely    */
   ```

   修改完成后按*Esc*键，并输入 `:wq` 后按下回车键，保存并退出。

   iv. 执行以下命令，安装Netperf和sar监控工具。

   ```
   cd netperf
   ./configure
   make && make install
   ```

4. 在测试机上执行以下命令，启动64个netserver服务。

```
#!/bin/bash
for j in `seq 64`; do
    netserver -p $[16000+j] > server_$[16000+j].netperf > /dev/null 2>&1  &
done
```

5. 在测试机上执行以下命令，查询测试机的私网IP地址。

```
ifconfig || ip addr
```

```
[root@launch-0322 netperf]# ifconfig || ip addr
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.[redacted]3  netmask 255.255.240.0  broadcast 172.16.[redacted]
        inet6 [redacted]la  prefixlen 64  scopeid 0x20<link>
        ether [redacted]ueuelen 1000  (Ethernet)
        RX packets 327551824  bytes 14206009941 (13.2 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 44604  bytes 6561192 (6.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

6. 在辅助测试机上执行以下命令，向测试机输入流量。

```
#!/bin/bash
server_ip=<测试机私网IP地址>
for j in `seq 64`; do
    port=$[16000+j]
    netperf -H ${server_ip} -l ${run_time:-300} -t UDP_STREAM  -p $port  -- -m 1 -D > /
dev/null 2>&1 &
done
```

`<测试机私网IP地址>` 需替换为上一步查询的实际测试机的私网IP地址，示例如下图所示。

```
[root@i[redacted] netperf]# #!/bin/bash
[root@i[redacted] netperf]# server_ip=172.1[redacted]3
[root@i[redacted] netperf]#
[root@i[redacted] netperf]# for j in `seq 64`; do
>     port=$[16000+j]
>     netperf -H ${server_ip} -l ${run_time:-300} -t UDP_STREAM  -p $port  -- -m 1 -D > /dev/null 2>&1 &
> done
[65] 5819
[66] 5820
[67] 5821
[68] 5822
[69] 5823
```

7. 在测试机上执行以下命令，测试网络流量。

```
sar -n DEV 1
```

在测试结果中查看 `rxpck/s` 列的数据值，`rxpck/s` 表示该测试机每秒钟接收的数据包总数。如下图所示，示例中测试机每秒钟接收到的数据包数约为90万。

```
[root@launch-0322 netperf]# sar -n DEV 1
[redacted]_64 (launch-0322)     03/24/2022      _x86_64_        (2 CPU)

10:51:19 AM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
10:51:20 AM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00
10:51:20 AM      eth0 898970.00      7.00  37750.35      2.86      0.00      0.00      0.00

10:51:20 AM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
10:51:21 AM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00
10:51:21 AM      eth0 899982.00      9.00  37792.87      1.29      0.00      0.00      0.00

10:51:21 AM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
10:51:22 AM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00
10:51:22 AM      eth0 899955.00      8.00  37791.75      1.22      0.00      0.00      0.00
```

测试网络PPS（大于600万小于2000万）

1. 分别远程连接测试机和辅助测试机。

   关于连接方法的介绍，请参见实例连接概述。

2. 分别在3台辅助测试机和1台测试机上执行以下命令，安装sockperf。

   ```
   yum install -y sockperf
   ```

   如果 `yum` 不能安装，则执行以下命令，使用编译安装。

   ```
   yum install -y autoconf automake libtool g++ gcc-c++
   cd /opt
   wget https://github.com/Mellanox/sockperf/archive/3.6.tar.gz
   tar -zxf 3.6.tar.gz
   cd sockperf-3.6/
   ./autogen.sh
   ./configure
   make -j `cat /proc/cpuinfo| grep process | wc -l`
   make install
   ```

3. 分别在3台辅助测试机上执行以下命令，向测试机输入流量。

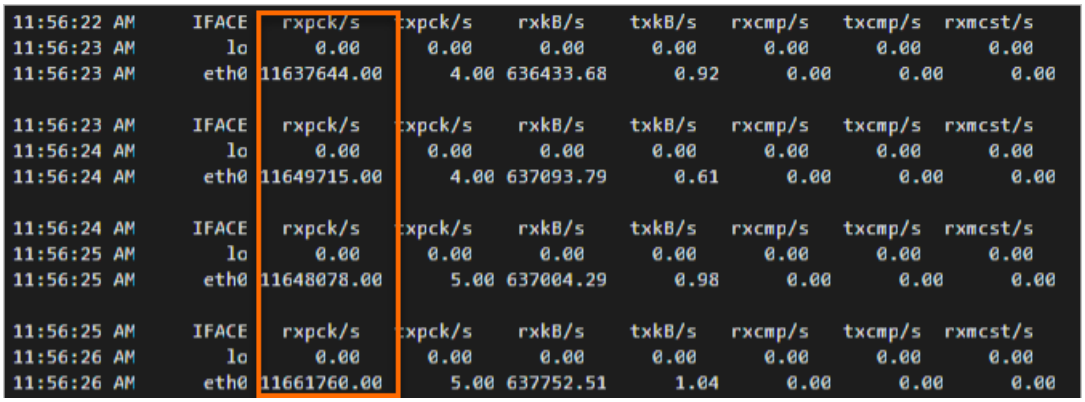
   ```
   server_ip="<测试机私网IP地址>"
   threads=64
   msg_size=14
   run_time=60
   basePort=6666
   for((i=0;i<$threads;++i));do
       nohup sockperf tp -i $server_ip --client_port $[${basePort}+${i}] --pps max -m ${ms
   g_size} -t ${run_time} --port $[${basePort}+${i}] 2>&1 &
   done
   ```

   其中 `<测试机私网IP地址>` 是实际测试机的私网IP地址， `run_time` 是输入流量的时间，请您根据实际情况修改。

4. 分别在3台辅助测试机和1台测试机上执行以下命令，测试网络流量。

   ```
   sar -n DEV 1
   ```

   在测试机上查看 `rxpck/s` 列的数据值， `rxpck/s` 表示该测试机每秒钟接收到的数据包。如下图所示，示例中测试机每秒钟接收到的数据包数约为1200万。

   ```
   11:56:22 AM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
   11:56:23 AM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00
   11:56:23 AM      eth0 11637644.00      4.00 636433.68      0.92      0.00      0.00      0.00

   11:56:23 AM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
   11:56:24 AM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00
   11:56:24 AM      eth0 11649715.00      4.00 637093.79      0.61      0.00      0.00      0.00

   11:56:24 AM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
   11:56:25 AM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00
   11:56:25 AM      eth0 11648078.00      5.00 637004.29      0.98      0.00      0.00      0.00

   11:56:25 AM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
   11:56:26 AM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00
   11:56:26 AM      eth0 11661760.00      5.00 637752.51      1.04      0.00      0.00      0.00
   ```

测试网络PPS（大于2000万）

1. 分别远程连接测试机和辅助测试机。

   关于连接方法的介绍，请参见实例连接概述。

2. 分别在3台辅助测试机和1台测试机上执行以下命令，安装sockperf。

   ```
   yum install -y sockperf
   ```

   如果 `yum` 不能安装，则执行以下命令，使用编译安装。

   ```
   yum install -y autoconf automake libtool g++ gcc-c++
   cd /opt
   wget https://github.com/Mellanox/sockperf/archive/3.6.tar.gz
   tar -zxf 3.6.tar.gz
   cd sockperf-3.6/
   ./autogen.sh
   ./configure
   make -j `cat /proc/cpuinfo| grep process | wc -l`
   make install
   ```

3. 在测试机上执行以下命令，绑定中断。

   ```
   a=$(cat /proc/interrupts | grep virtio2-input | awk -F ':' '{print $1}')
   cpu=0
   for irq in $a; do
       echo $cpu >/proc/irq/$irq/smp_affinity_list
       let cpu+=2
   done
   ```

4. 分别在3台辅助测试机上执行以下命令，向测试机输入流量。

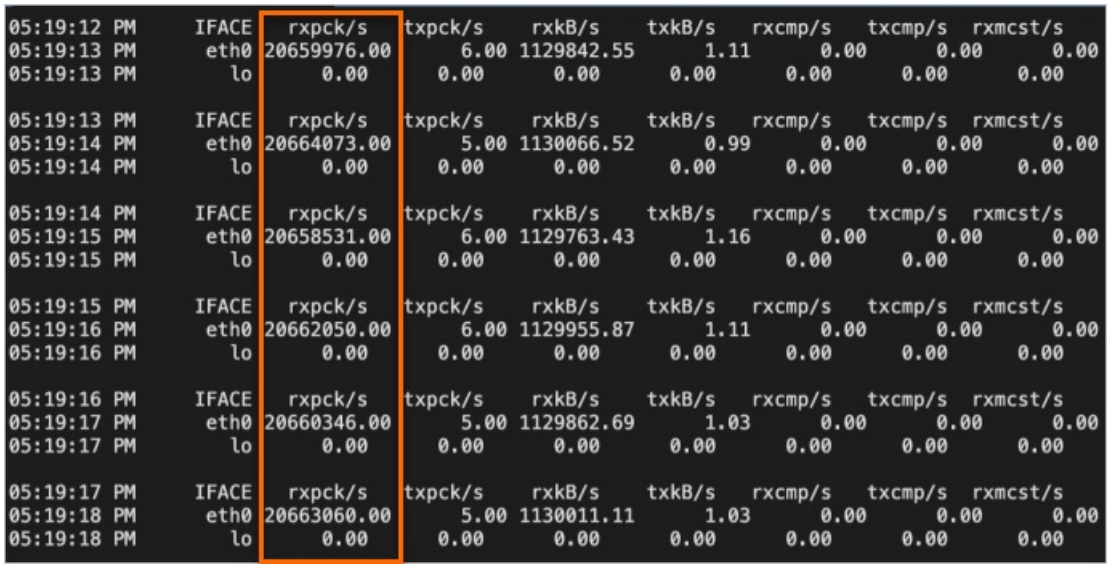
   ```
   server_ip="<测试机私网IP地址>"
   threads=64
   msg_size=14
   run_time=60
   basePort=6666
   for((i=0;i<$threads;++i));do
       nohup sockperf tp -i $server_ip --client_port $[${basePort}+${i}] --pps max -m ${ms
   g_size} -t ${run_time} --port $[${basePort}+${i}] 2>&1 &
   done
   ```

   其中 `<测试机私网IP地址>` 是实际测试机的私网IP地址， `run_time` 是输入流量的时间，请您根据实际情况修改。

5. 分别在3台辅助测试机和1台测试机上执行以下命令，测试网络流量。

   ```
   sar -n DEV 1
   ```

   在测试机上查看 `rxpck/s` 列的数据值， `rxpck/s` 表示该测试机每秒钟接收到的数据包。如下图所示，示例中测试机每秒钟接收到的数据包数约为2000万。



```
05:19:12 PM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
05:19:13 PM      eth0 20659976.00      6.00 1129842.55      1.11      0.00      0.00      0.00
05:19:13 PM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00

05:19:13 PM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
05:19:14 PM      eth0 20664073.00      5.00 1130066.52      0.99      0.00      0.00      0.00
05:19:14 PM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00

05:19:14 PM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
05:19:15 PM      eth0 20658531.00      6.00 1129763.43      1.16      0.00      0.00      0.00
05:19:15 PM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00

05:19:15 PM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
05:19:16 PM      eth0 20662050.00      6.00 1129955.87      1.11      0.00      0.00      0.00
05:19:16 PM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00

05:19:16 PM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
05:19:17 PM      eth0 20660346.00      5.00 1129862.69      1.03      0.00      0.00      0.00
05:19:17 PM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00

05:19:17 PM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
05:19:18 PM      eth0 20663060.00      5.00 1130011.11      1.03      0.00      0.00      0.00
05:19:18 PM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00
```

测试网络带宽

1. 分别远程连接测试机和辅助测试机。

   关于连接方法的介绍，请参见实例连接概述。

2. 分别在测试机和辅助测试机上执行以下命令，下载Netperf。

   ```
   wget https://benchmark-packages.oss-cn-qingdao.aliyuncs.com/netperf-2.7.0.tar.gz
   ```

3. 分别在测试机和辅助测试机上执行以下命令，安装Netperf和sar监控工具。

   ```
   yum install -y gcc autoconf automake libtool sysstat
   tar -zxvf netperf-2.7.0.tar.gz
   cd netperf
   ./configure
   make && make install
   ```

4. 在测试机上执行以下命令，启动64个netserver服务。

   ```
   #!/bin/bash
   for j in `seq 64`; do
       netserver -p $[16000+j] > server_$[16000+j].netperf > /dev/null 2>&1  &
   done
   ```

5. 在测试机上执行以下命令，查询测试机的私网IP地址。

   ```
   ifconfig || ip addr
   ```

```
[root@launch-0322 netperf]# ifconfig || ip addr
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.[redacted]3  netmask 255.255.240.0  broadcast 172.16.[redacted]
        inet6 [redacted]la  prefixlen 64  scopeid 0x20<link>
        ether [redacted]ueuelen 1000  (Ethernet)
        RX packets 327551824  bytes 14206009941 (13.2 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 44604  bytes 6561192 (6.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

6. 在辅助测试机上执行以下命令，向测试机输入流量。

```
#!/bin/bash
server_ip=<测试机私网IP地址>
for j in `seq 64`; do
    port=$[16000+j]
    netperf -H ${server_ip} -l ${run_time:-300} -t UDP_STREAM  -p $port  -- -m 1 -D > /
dev/null 2>&1 &
done
```

`<测试机私网IP地址>` 需替换为上一步查询的实际测试机的私网IP地址，示例如下图所示。

```
[root@i[redacted] netperf]# #!/bin/bash
[root@i[redacted] netperf]# server_ip=172.1[redacted]3
[root@i[redacted] netperf]#
[root@i[redacted] netperf]# for j in `seq 64`; do
>     port=$[16000+j]
>     netperf -H ${server_ip} -l ${run_time:-300} -t UDP_STREAM  -p $port  -- -m 1 -D > /dev/null 2>&1 &
> done
[65] 5819
[66] 5820
[67] 5821
[68] 5822
[69] 5823
```

7. 在测试机上执行以下命令，测试网络带宽。

```
sar -n DEV 1
```

在测试结果中查看 `rxkB/s` 列的数据值，`rxkB/s` 表示该测试机每秒钟接收的字节数。所以转换为带宽（kbps）=字节数（rxkB/s）*8，示例如下图所示。

```
[root@launch-0322 netperf]# sar -n DEV 1
[redacted]_64 (launch-0322)     03/24/2022      _x86_64_        (2 CPU)

10:51:19 AM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
10:51:20 AM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00
10:51:20 AM      eth0 898970.00      7.00  37750.35      2.86      0.00      0.00      0.00

10:51:20 AM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
10:51:21 AM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00
10:51:21 AM      eth0 899982.00      9.00  37792.87      1.29      0.00      0.00      0.00

10:51:21 AM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
10:51:22 AM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00
10:51:22 AM      eth0 899955.00      8.00  37791.75      1.22      0.00      0.00      0.00
```

测试网络时延

1. 分别远程连接测试机和辅助测试机。

   关于连接方法的介绍，请参见实例连接概述。

2. 分别在测试机和辅助测试机上执行以下命令，安装sockperf。

   ```
   yum install -y sockperf
   ```

   如果 `yum` 不能安装，则执行以下命令，使用编译安装。

   ```
   cd /opt
   wget https://github.com/Mellanox/sockperf/archive/3.6.tar.gz
   tar -zxf 3.6.tar.gz
   cd sockperf-3.6/
   ./autogen.sh
   ./configure
   make -j `cat /proc/cpuinfo| grep process | wc -l`
   make install
   ```

3. 在测试机上执行以下命令，启动服务。

   ```
   sockperf sr --daemonize > /dev/null 2>&1
   ```

4. 在测试机上执行以下命令，查询测试机的私网IP地址。

   ```
   ifconfig || ip addr
   ```

   ```
   [root@launch-0322 netperf]# ifconfig || ip addr
   eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
           inet 172.16.xxx.xxx  netmask 255.255.240.0  broadcast 172.16.xxx.xxx
           inet6 xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx  prefixlen 64  scopeid 0x20<link>
           ether xx:xx:xx:xx:xx:xx  txqueuelen 1000  (Ethernet)
           RX packets 327551824  bytes 14206009941 (13.2 GiB)
           RX errors 0  dropped 0  overruns 0  frame 0
           TX packets 44604  bytes 6561192 (6.2 MiB)
           TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

   lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
           inet 127.0.0.1  netmask 255.0.0.0
           inet6 ::1  prefixlen 128  scopeid 0x10<host>
           loop  txqueuelen 1000  (Local Loopback)
           RX packets 0  bytes 0 (0.0 B)
           RX errors 0  dropped 0  overruns 0  frame 0
           TX packets 0  bytes 0 (0.0 B)
           TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
   ```

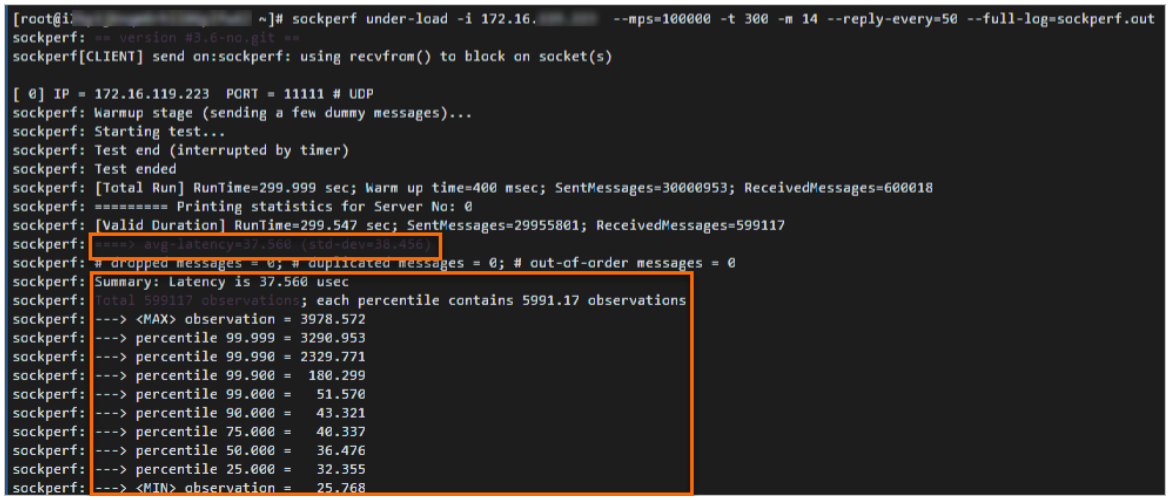
5. 在辅助测试机上执行以下命令，向测试机输入流量。

   ```
   sockperf under-load -i <测试机私网IP地址>  --mps=100000 -t 300 -m 14 --reply-every=50 --full-log=sockperf.out
   ```

   `<测试机私网IP地址>` 需替换为上一步查询的实际测试机的私网IP地址。

   查看结果数据，示例如下图所示。

   - 以 `avg-latency` 开头的结果数据表示平均时延，单位为us。
   - 以 `percentile 99.000` 开头的结果数据表示99分位时延，单位为us。


[root@i ~]# sockperf under-load -i 172.16. --mps=100000 -t 300 -m 14 --reply-every=50 --full-log=sockperf.out
sockperf: == version #3.6-no.git ==
sockperf[CLIENT] send on:sockperf: using recvfrom() to block on socket(s)
[0] IP = 172.16.119.223 PORT = 11111 # UDP
sockperf: Warmup stage (sending a few dummy messages)...
sockperf: Starting test...
sockperf: Test end (interrupted by timer)
sockperf: Test ended
sockperf: [Total Run] RunTime=299.999 sec; Warm up time=400 msec; SentMessages=30000953; ReceivedMessages=600018
sockperf: ========= Printing statistics for Server No: 0
sockperf: [Valid Duration] RunTime=299.547 sec; SentMessages=29955801; ReceivedMessages=599117
sockperf: ====> avg-latency=37.560 (std-dev=38.456)
sockperf: # dropped messages = 0; # duplicated messages = 0; # out-of-order messages = 0
sockperf: Summary: Latency is 37.560 usec
sockperf: Total 599117 observations; each percentile contains 5991.17 observations
sockperf: ---> <MAX> observation = 3978.572
sockperf: ---> percentile 99.999 = 3290.953
sockperf: ---> percentile 99.990 = 2329.771
sockperf: ---> percentile 99.900 = 180.299
sockperf: ---> percentile 99.000 = 51.570
sockperf: ---> percentile 90.000 = 43.321
sockperf: ---> percentile 75.000 = 40.337
sockperf: ---> percentile 50.000 = 36.476
sockperf: ---> percentile 25.000 = 32.355
sockperf: ---> <MIN> observation = 25.768

21.经典网络和专有网络互通最佳实践

经典网络和专有网络互通可以通过建立ClassicLink连接实现网络互通，也可以将经典网络迁移至专有网络后，通过云企业网实现专有网络的互通。本教程介绍如何将多台经典网络类型的ECS实例迁移至专有网络，并和新创建的专有网络类型的ECS实例实现互通。

前提条件

确保经典网络类型的ECS实例符合以下条件：

- 不能是挂载了本地盘的实例规格族。
- ECS实例不在华东1（杭州）地域的可用区C内，该可用区的部分ECS实例不支持迁移网络类型。
- 您业务需求的应用服务均已设置为开机自启动。
- 检查业务需求的软件，是否采用绑定网卡MAC地址注册合法环境。由于迁移专有网络VPC后公网网卡会被移除，因此网卡MAC地址会丢失。此时，您可以联系软件供应商确认所使用的软件是否通过绑定网卡MAC地址注册了您的ECS实例，如果是则需要重新绑定到ECS实例主网卡。

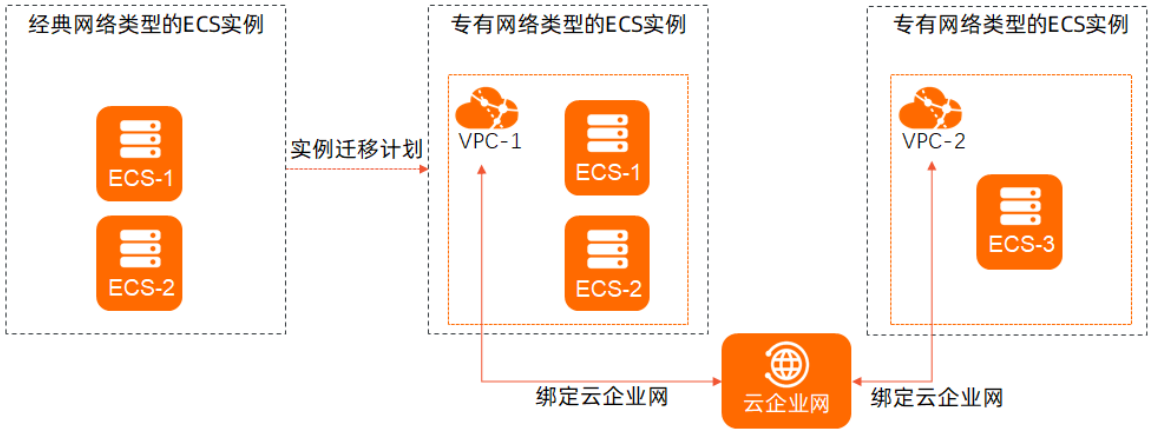
背景信息

在正式操作前，请确保您已了解以下注意事项及使用限制：

- ECS实例从经典网络迁移至专有网络的注意事项，请参见ECS实例从经典网络迁移到专有网络中背景信息章节。
- 云企业网的使用限制，请参见使用限制。

本教程中使用如下资源作为示例：

同账号同地域



- 两台经典网络类型的ECS实例：`ECS-1` 和 `ECS-2`
- 一台专有网络类型的ECS实例：`ECS-3`
- 实例迁移计划中新创建的VPC：`VPC-1`
- 一台专有网络类型的ECS实例对应的VPC：`VPC-2`

通过以下步骤实现网络互通：

1. 通过实例迁移计划，将ECS实例从经典网络迁移至专有网络。

2. 通过绑定云企业网的方式实现 `VPC-1` 与 `VPC-2` 网络互通。

本教程中的示例场景为同账号同地域，因此不需要配置云企业网的互通带宽等功能。如果您的互通需求需要跨账号跨地域，也可以通过云企业网实现。详情请参见云企业网。

如果您的云资源均在同一个账号的同一个地域下，并且业务需求较少，不需要将经典网络迁移至专有网络，您可以使用ClassicLink功能实现经典网络和专有网络互通，详情请参见经典网络和专有网络互通。

步骤一：经典网络迁移到专有网络

通过实例迁移计划将ECS实例（ECS-1和ECS-2）从经典网络迁移到专有网络（VPC-1）。您可以直接在实例迁移计划中自动创建VPC，即**选择迁移的目标专有网络**设置为**（默认）系统自动创建专有网络，网段：10.0.0.0/8**。

具体操作，请参见ECS实例从经典网络迁移到专有网络。

步骤二：创建并绑定云企业网

等待经典网络类型的ECS实例迁移网络完成后，需要完成以下操作实现 `VPC-1` 和 `VPC-2` 的网络互通。

1. 登录云企业网控制台。
2. 单击**创建云企业网实例**。
3. 在**加载网络实例**区域，完成以下配置项。
   - **实例类型**：选择**专有网络（VPC）**
   - **地域**：选择 `VPC-1` 所在的地域。例如：**华东2（上海）**
   - **网络实例**：选择步骤一中创建的 `VPC-1`
4. 完成其它基础配置项后，单击**确定**。

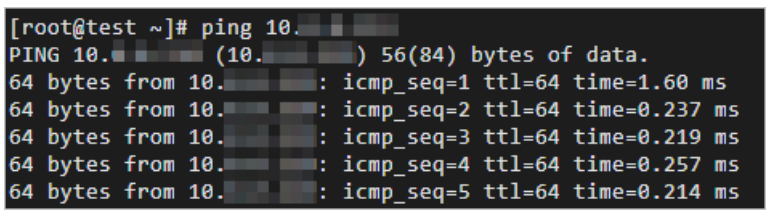
   详情请参见创建CEN实例。
5. 登录专有网络管理控制台。
6. 在顶部菜单栏处，选择ECS实例 `ECS-3` 的专有网络 `VPC-2` 所在的地域。
7. 找到 `VPC-2` ，并单击名称进入**专有网络详情**页面。
8. 单击**加入云企业网**，并选择已创建好的云企业网。

步骤三：验证互通性

通过以上步骤，即可实现 `ECS-1` 、 `ECS-2` 、 `ECS-3` 三台ECS实例的网络互通。由于 `ECS-1` 和 `ECS-2` 迁移至了同一个VPC下，所以两台ECS实例是内网互通的。您只需要远程连接使用 `VPC-2` 的ECS实例 `ECS-3` ，并运行以下命令，检查与其它两台ECS实例是否互通即可。

```
ping <ECS实例的私网IP>
```

例如，运行命令 `ping 10.0.**.**` 。正确的返回结果如下图所示。



22.ECS状态变化事件的自动化运维最佳实践

阿里云ECS在已有的系统事件的基础上，通过云监控新发布了状态变化类事件和抢占型实例的中断通知事件。当ECS实例的状态发生变化时，会触发一条ECS实例状态变化事件。这种变化包括您在控制台、OpenAPI和SDK操作导致的变化，也包括弹性伸缩或欠费等原因而自动触发的变化，还包括因系统异常而触发的变化。

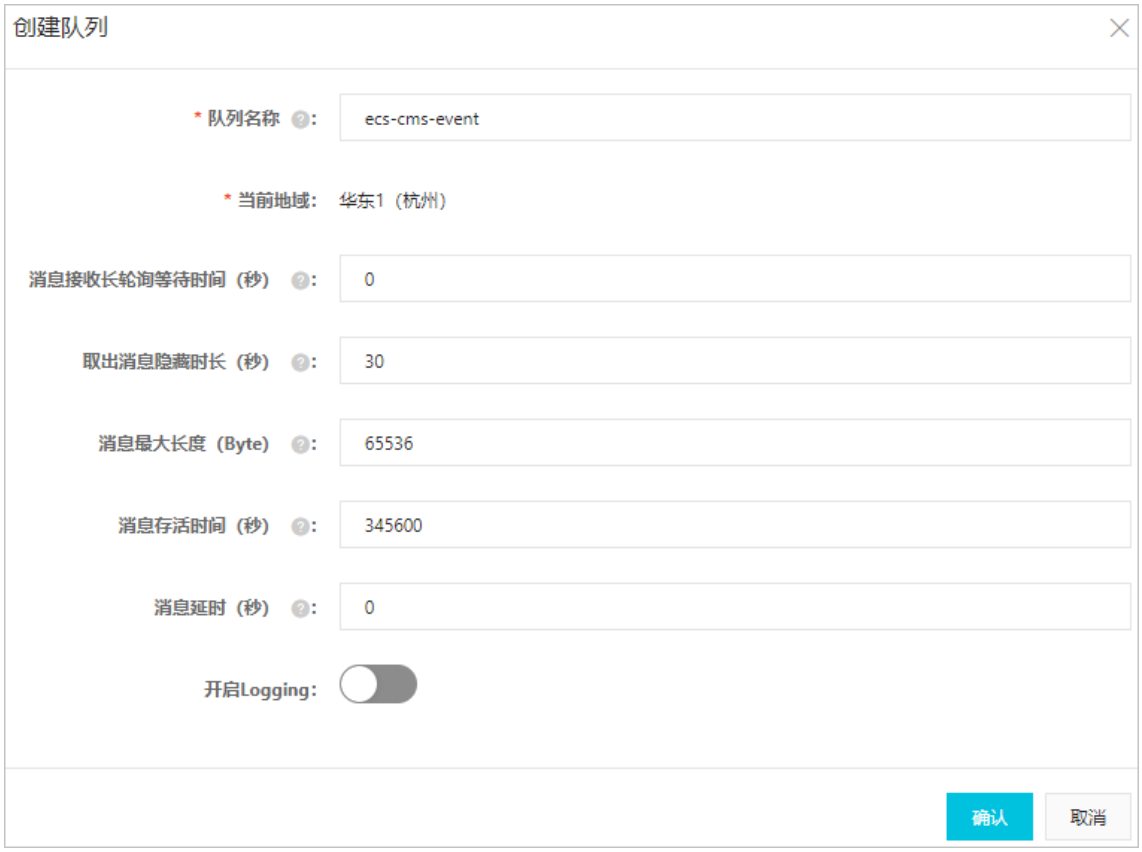
背景信息

云监控之前发布的系统事件，主要针对告警后人工介入的场景，而本次新发布的事件属于正常类的信息通知，适合自动化的审计运维场景。为了自动化处理ECS状态变化事件，云监控提供了两种主要途径：一种是通过函数计算，另一种是通过MNS消息队列。

自动化处理ECS状态变化事件的准备工作

创建消息队列

- 创建消息队列
  i. 登录MNS控制台。
  ii. 在**队列**页面，选择地域，单击右上角的**创建队列**，进入**新建队列**页面。



  iii. 输入队列名称（例如：ecs-cms-event）等信息，单击**确认**，即可完成创建消息队列。
- 创建事件报警规则
  i. 登录云监控控制台。

ii. 在左侧导航栏，单击**事件监控**。

iii. 在**事件监控**页面，单击**报警规则**页签，单击右上角的**创建事件报警**。

iv. 在**基本信息**区域，填写报警规则名称，例如：ecs-test-rule。

v. 在**事件报警规则**区域，相关参数设置如下：

- **事件类型**选择**系统事件**。
- **产品类型**选择**云服务器ECS**。
- **事件类型**选择**状态通知**。
- **事件名称**按照实际情况选择。

- 当**资源范围**选择**全部资源**时，任何资源发生相关事件，都会按照配置发送通知；当**资源范围**选择**应用分组**时，只有指定应用分组内的资源发生相关事件，才会按照配置发送通知。

vi. 在**报警方式**区域，相关参数设置如下：

- **联系人组**和**通知方式**按照实际情况选择。
- 报警数据写入方式选择**消息服务队列**，**地域**和**队列**按照实际情况选择（例如：ecs-cms-event）。

vii. 单击**确定**。

- 安装Python依赖

  本文所有的代码均使用Python 3.6测试通过，您也可以使用Java等其他编程语言。

  请使用PyPI（Python Package Index）安装以下Python依赖：

  - aliyun-python-sdk-core-v3>=2.12.1
  - aliyun-python-sdk-ecs>=4.16.0
  - aliyun-mns>=1.1.5

自动化处理ECS状态变化事件的实施步骤

云监控会将云服务器ECS所有的状态变化事件投递到MNS中，再通过编写代码从MNS获取消息并进行消息处理。

- 实践一：对所有ECS的创建和释放事件进行记录

  目前ECS控制台无法查询已经释放的实例。如果您有查询需求，可以通过ECS状态变化事件将所有ECS的生命周期记录在自己的数据库或日志中。当您创建ECS时，会发送一个Pending事件，当您释放ECS时，会发送一个Deleted事件，云监控对这两种事件进行记录。

  i. 编辑一个Conf文件。

  Conf文件需包含MNS的如下信息：

  - `endpoint`：在MNS控制台的**队列**页面，单击**获取Endpoint**。
  - `access_key` 和 `access_key_secret`：在用户信息管理控制台中获取。
  - `region_id` 和 `queue_name`：在MNS控制台的**队列**页面，查看队列名称和所属地域。

```
class Conf:
    endpoint = 'http://<id>.mns.<region>.aliyuncs.com/'
    access_key = '<access_key>'
    access_key_secret = '<access_key_secrect>'
     = 'cn-beijing'
    queue_name = 'test'
    vsever_group_id = '<your_vserver_group_id>'
```

  ii. 使用MNS的SDK编写一个MNS Client用来获取MNS消息。

```
# -*- 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_secre
t)
        self.queue_name = Conf.queue_name
        self.listeners = dict()
    def regist_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 rec
eive message.', self.queue_name)
                else:
                    logging.error('No Message, continue waiting')
class BasicListener(object):
    def process(self, event):
        pass
```

上述代码只对MNS消息获取的数据，调用Listener消费消息之后删除消息。

iii. 注册一个指定Listener消费事件。这个简单的Listener判断收到Pending和Deleted事件时，打印一行日志。

```
 # -*- 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}')
```

Main函数写法如下：

```
mns_client = MNSClient()
mns_client.regist_listener(ListenerLog())
mns_client.run()
```

实际生产环境下，可能需要将事件存储在数据库里，或者使用日志服务（SLS），方便后期的搜索和审计。

- 实践二：ECS关机自动重启

  在某些场景下，ECS会非预期的关机，您可能需要自动重启已经关机的ECS。

  为了实现ECS关机自动重启，您可以复用实践一里面的MNS Client，添加一个新的Listener。当收到Stopped事件时，对该ECS执行命令start。

```
# -*- coding: utf-8 -*-
import logging
from aliyunsdkecs.request.v20140526 import StartInstanceRequest
from aliyunsdkcore.client import AcsClient
from .mns_client import BasicListener
from .config import Conf
class ECSClient(object):
    def __init__(self, acs_client):
        self.client = acs_client
    # 启动ECS实例
    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)
```

在实际生产环境下，执行完start命令后，可能还需要继续接收后续的Starting、Running或Stopped等事件，再配合计时器和计数器，进行start成功或失败之后的处理。

- 实践三：抢占型实例释放前，自动从负载均衡（SLB）移除

  抢占型实例在释放之前五分钟左右，会发出释放告警事件，您可以利用这短暂的时间运行一些业务不中断的逻辑。例如，主动从SLB的后端服务器中去掉这台即将被释放的抢占型实例，而不是被动等待实例释放后SLB的自动处理。

  您复用实践一的MNS Client，添加一个新的Listener，当收到抢占型实例的释放告警时，调用SLB的SDK。

```
# -*- coding: utf-8 -*-
from aliyunsdkcore.client import AcsClient
from aliyunsdkcore.request import CommonRequest
from .mns_client import BasicListener
from .config import Conf
class SLBClient(object):
    def __init__(self):
        self.client = AcsClient(Conf.access_key, Conf.access_key_secret, Conf.region_id)
        self.request = CommonRequest()
        self.request.set_method('POST')
        self.request.set_accept_format('json')
        self.request.set_version('2014-05-15')
        self.request.set_domain('slb.aliyuncs.com')
        self.request.add_query_param('RegionId', Conf.region_id)
    def remove_vserver_group_backend_servers(self, vserver_group_id, instance_id):
        self.request.set_action_name('RemoveVServerGroupBackendServers')
        self.request.add_query_param('VServerGroupId', vserver_group_id)
        self.request.add_query_param('BackendServers',
                                     "[{'ServerId':'" + instance_id + "','Port':'80','Wei
ght':'100'}]")
        response = self.client.do_action_with_exception(self.request)
        return str(response, encoding='utf-8')
class ListenerSLB(BasicListener):
    def __init__(self, vsever_group_id):
        self.slb_caller = SLBClient()
        self.vsever_group_id = Conf.vsever_group_id
    def process(self, event):
        detail = event['content']
        instance_id = detail['instanceId']
        if detail['action'] == 'delete':
            self.slb_caller.remove_vserver_group_backend_servers(self.vsever_group_id, in
stance_id)
```

> **注意**
>
> 抢占型实例释放告警的 `event name` 与前面不同，应该是 `mns_client.regist_listener(ListenerSLB(Conf.vsever_group_id), 'Instance:PreemptibleInstanceInterruption')` 。

在实际生产环境下，您需要再申请一台新的抢占型实例，挂载到SLB上，来保证服务能力。

23.基于快照与镜像功能迁移实例数据

随着ECS实例的不断迭代，较早创建的ECS实例可能出现无法新增资源补给等问题，进而影响您对云上业务的运维。因此，阿里云建议您通过快照与镜像功能，将源ECS实例数据迁移至新创建的目标ECS实例上，以保障您云上业务的运维效率。

背景信息

阿里云快照服务是一种无代理（Agentless）的数据备份方式，用于备份或者恢复整个云盘数据。关于快照的更多信息，请参见快照概述。自定义镜像是基于ECS实例的快照生成的一种镜像，您可以通过自定义镜像快速完成相同配置ECS实例的创建。关于自定义镜像的更多信息，请参见自定义镜像概述。

通过阿里云提供的快照与自定义镜像，完成ECS实例间数据迁移的操作流程说明如下：

- 数据迁移前，请您仔细阅读注意事项。更多信息，请参见注意事项。
- 步骤一：为源ECS实例创建自定义镜像。
- 步骤二：（可选）跨地域复制镜像。
- 步骤三：使用自定义镜像新建目标ECS实例。
- 步骤四：检查新创建的目标ECS实例内的数据。
- 步骤五：（可选）释放或删除源ECS实例及相关资源。

注意事项

您需要了解以下注意事项，确认无误后再进行ECS实例的数据迁移操作。

- 部分包含本地盘的实例无法创建快照，因此该部分实例不支持通过本文的操作完成实例的数据迁移。
- 源ECS实例的网络类型可以是经典网络或专有网络VPC。
- 新建目标ECS实例时，仅支持创建VPC网络类型的ECS实例。
- 新建目标ECS实例时，仅支持选择当前可用区下有库存的实例规格。

  > **说明** 如果您需要跨地域和可用区迁移实例数据，建议您提前自行做好资源所属地域和可用区的规划工作。

- 由于是通过快照与镜像功能完成的实例数据迁移操作，因此数据迁移后，新创建的目标ECS实例中云盘数据与源ECS实例中的云盘数据保持一致，但新创建的目标ECS实例的实例元数据会重新生成，与源ECS实例中的实例元数据相比较会发生变化。关于实例元数据的更多信息，请参见ECS实例元数据概述。

  由于实例元数据会发生变化，在实例数据迁移之前，建议您手动排查资源关联关系，并在数据迁移后及时更新资源的关联关系。例如：

  - 集群内部通过私网IP地址互联互通，在进行实例数据迁移后，您需要替换为最新的私网IP地址。
  - 某些应用的许可证（License）与ECS实例的MAC地址绑定，在进行实例数据迁移后，这些许可证将因为ECS实例的MAC地址改变而失效，您需要重新绑定最新的MAC地址。

步骤一：为源ECS实例创建自定义镜像

通过实例创建自定义镜像前，您需要了解相关注意事项，更多信息，请参见使用实例创建自定义镜像。在创建自定义镜像期间，系统会对ECS实例的各个云盘自动创建快照，快照将产生一定的费用，快照费用的详细信息，请参见快照计费。

1. 登录ECS管理控制台。
2. 在左侧导航栏，选择**实例与镜像 > 实例**。
3. 在顶部菜单栏左上角处，选择地域。
4. 找到源ECS实例，在**操作**列，选择**更多 > 云盘和镜像 > 创建自定义镜像**。
5. 在**创建自定义镜像**对话框，完成配置，然后单击**创建**。

   您必须配置自定义镜像的名称与描述信息。其他相关配置项非必须配置，您可以根据自身需求选择设置。
6. 在左侧导航栏，选择**实例与镜像 > 镜像**。
7. 在**自定义镜像**页签，找到并查看已创建的自定义镜像状态。

   如果您存在多个自定义镜像，可以通过您配置的自定义镜像名称搜索目标镜像。当目标自定义镜像的状态变为**可用**时，再进行下一步操作。

步骤二：（可选）跨地域复制镜像

如果您需要将源ECS实例的数据跨地域迁移至新创建的目标ECS实例，需要先通过复制镜像功能将自定义镜像复制到其他地域。具体操作，请参见复制镜像。

镜像复制完成后，后续的操作步骤您需要在新的地域下进行。

步骤三：使用自定义镜像新建目标ECS实例

1. 在左侧导航栏，选择**实例与镜像 > 镜像**。
2. 在顶部菜单栏左上角处，选择地域。

   如果是跨地域复制的镜像，需要先将地域切换至目标地域。
3. 在**自定义镜像**页签，找到基于源ECS实例创建的自定义镜像。

   如果是跨地域复制的镜像，需要找到复制后生成的自定义镜像。
4. 在**操作**列，单击**创建实例**。
5. 在云服务器ECS的购买页面，完成资源配置，然后新建目标ECS实例。

   创建ECS实例的具体操作，请参见使用向导创建实例。其中，您需要注意以下配置项：
   - 在**基础配置**中的**镜像**区域以及**存储**区域，已经默认指定了自定义镜像与云盘的信息，您无需更改。

   - 在**系统配置**的**登录凭证**区域，选择**使用镜像预设密码**。

> **说明** 使用镜像预设密码后，新创建的目标ECS实例登录密码与源ECS实例的登录密码一致。

步骤四：检查新创建的目标ECS实例内的数据

您需要检查新创建的目标ECS实例的相关数据情况，确保实例数据迁移后，业务功能仍可流畅运行。例如：

- 结合源ECS实例内数据存储的实际情况，自行检查新创建的目标ECS实例内数据的完整性。
- 对比源ECS实例与新创建的目标ECS实例相关的资源信息变化，并自行修改已配置的资源关联关系。查看ECS实例信息的具体操作，请参见查看实例信息或查看实例元数据。

步骤五：（可选）释放或删除源ECS实例及相关资源

在您仔细检查新创建的目标ECS实例与源ECS实例数据没有差异，且完成了资源关联关系的更新，确保新创建的目标ECS实例内业务可以流畅运行后，结合自身的实际需求，可以选择释放或删除源ECS实例的相关资源，避免资源持续产生费用。相关操作说明如下：

> **注意** 释放实例、删除快照以及删除镜像的操作为单向操作，一旦操作完成，资源内的数据不可恢复。请确保您已完成所有业务数据的迁移再执行释放或删除资源的操作。

- 释放实例的具体操作，请参见释放实例。
- 删除快照的具体操作，请参见删除快照。
- 删除自定义镜像的具体操作，请参见删除自定义镜像。

> **说明** 删除自定义镜像后，已使用该镜像创建的ECS实例将无法初始化系统盘。如果您的自定义镜像为免费镜像，建议不删除该自定义镜像。镜像计费的详细信息，请参见镜像计费。

24.灾备方案

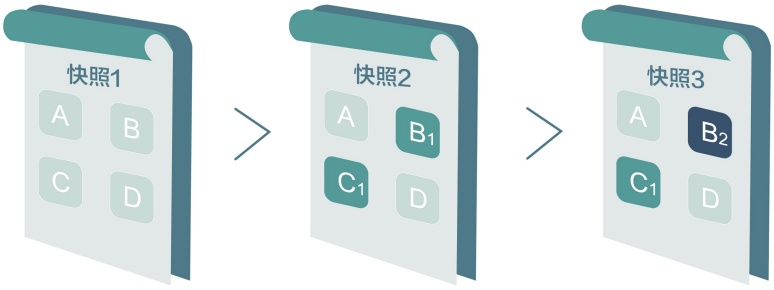
保障企业业务稳定、IT系统功能正常、数据安全十分重要，可以同时保障数据备份与系统、应用容灾的灾备解决方案应势而生，且发展迅速。ECS可使用快照、镜像进行备份。

灾备设计

- 快照备份

  阿里云ECS可使用快照进行系统盘、数据盘的备份。目前，阿里云提供快照2.0服务，提供了更高的快照额度、更灵活的自动任务策略，并进一步降低了对业务I/O的影响。快照备份实行增量原理，第一次备份为全量备份，后续为增量备份。增量快照具有快速创建以及存储容量小的优点。备份所需时间与待备份的增量数据体积有关。

  > **说明** 快照创建遵循增量原理，为了提高您的备份速度，建议您在创建完新快照后，再删除最新的历史快照。

  例如，快照1、快照2和快照3分别是磁盘的第一份、第二份和第三份快照。文件系统对磁盘的数据进行分块检查，当创建快照时，只有变化了的数据块，才会被复制到快照中。阿里云ECS的快照备份可配置为手动备份，也可配置为自动备份。配置为自动备份后可以指定磁盘自动创建快照的时间（24个整点）、重复日期（周一到周日）和保留时间（可自定义，范围是1-65536天，或选择永久保留）。

- 快照回滚

  当系统出现问题，需要将一块磁盘的数据回滚到之前的某一时刻，可以通过快照回滚实现，前提是该磁盘已经创建了快照。具体操作，请参见使用快照回滚云盘。请注意：

  - 回滚磁盘是不可逆操作，一旦回滚完成，原有的数据将无法恢复，请谨慎操作。
  - 回滚磁盘后，从所使用的快照的创建日期到当前时间这段时间内的数据都会丢失。

- 镜像备份

  镜像文件相当于副本文件，该副本文件包含了一块或多块磁盘中的所有数据，对于ECS而言，这些磁盘可以是单个系统盘，也可以是系统盘加数据盘的组合。使用镜像备份时，均是全量备份，且只能手动触发。

- 镜像恢复

  阿里云ECS支持使用快照创建自定义镜像，将快照的操作系统、数据环境信息完整的包含在镜像中。然后使用自定义镜像创建多台具有相同操作系统和数据环境信息的实例。ECS的快照与镜像配置请参见创建一个云盘快照和使用快照创建自定义镜像。

  > **说明** 创建的自定义镜像不能跨地域使用。

技术指标

RTO和RPO：与数据量大小有关，通常而言是小时级别。

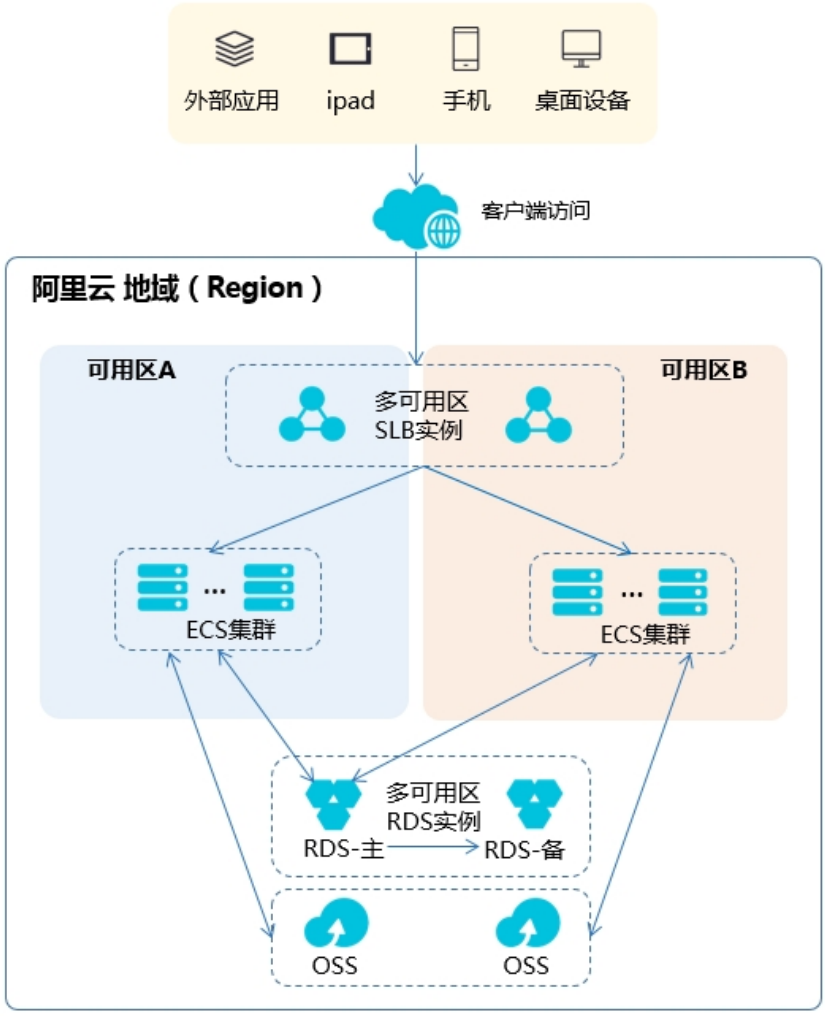
应用场景

- 备份恢复

  阿里云ECS可通过快照与镜像对系统盘、数据盘进行备份。如果存储在磁盘上的数据本身就是错误的数据，例如由于应用错误导致的数据错误，或者黑客利用应用漏洞进行恶意读写，此时就可以使用快照服务将磁盘上的数据恢复到期望的状态。另外ECS可通过镜像重新初始化磁盘或使用自定义镜像新购ECS实例。

- 容灾应用

  ECS可以从架构上实现容灾场景下的应用。例如，在应用前端购买SLB产品，后端相同应用部署至少两台ECS服务器，或者是使用阿里云的弹性伸缩技术，根据自定义ECS自身资源的使用规则进行弹性扩容。这样即便其中一台ECS服务器故障或者资源利用超负荷，也不会使服务对外终止，从而实现容灾场景下的应用。下图以同城两可用区机房部署ECS集群为例，所有通信均在阿里云千兆内网中完成，响应快速并减少了公网流量费用：



  - 负载均衡SLB：设备侧通过多可用区级别SLB做首层流量接入，用户流量被分发至两个及以上的可用区机房，机房内均部署ECS集群。
  - ECS集群：可用区机房部署的ECS节点是对等的，单节点故障不影响数据层应用和服务器管控功能。发生故障后系统会自动热迁移，另外的ECS节点可以持续提供业务访问，防止可能的单点故障或者热迁移失败导致的业务访问中断。热迁移失败后通过系统事件获知故障信息，您可以及时部署新节点。

- 数据层：在地域级别部署对象存储，不同可用区机房的ECS节点可以直接读取文件信息。若是数据库应用，使用多可用区ApsaraDB for RDS服务做承载，主节点支持多可用区读写，与应用层流量来源无冲突关系。同时，备节点支持多可用区读能力，防止主节点故障时，ECS无法读取数据。

25.部署高可用架构

25.1. 高可用架构部署方案

高可用架构提供业务分发、弹性扩展、多可用区部署等功能。相较于使用单台ECS实例部署数据库与应用，高可用架构只需简单部署，并且拥有更高的稳定性和可扩展性。

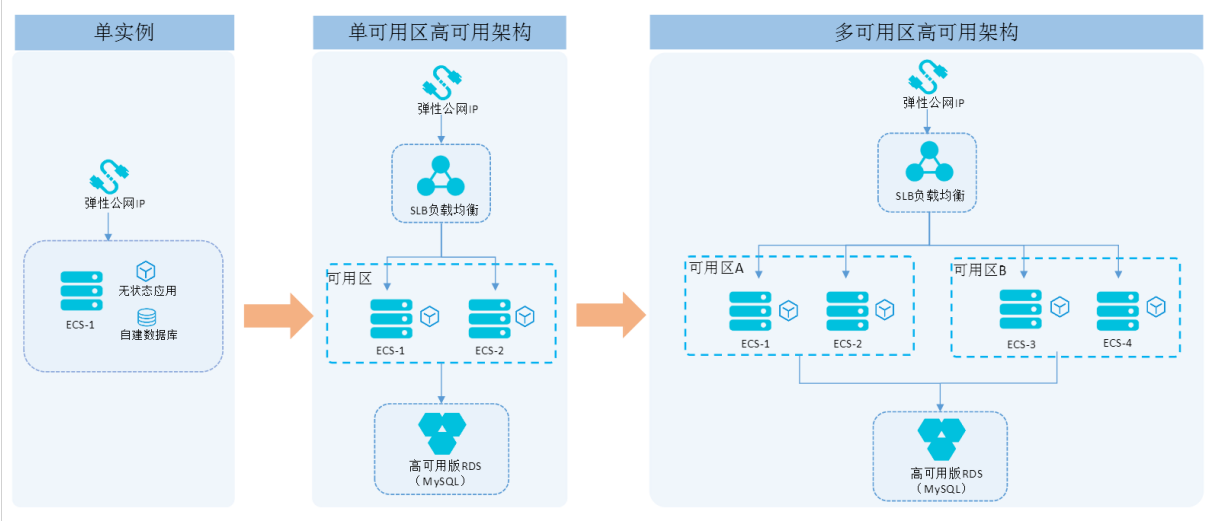
高可用架构特点

高可用架构具有如下特点：

- 使用多可用区高可用版的负载均衡SLB（Server Load Balancer）对多台云服务器ECS进行流量分发，可扩展应用系统对外服务能力、消除单点故障，提升应用系统的可用性。使用SLB自动跨可用区部署，可加强业务容灾能力。
- 通过自定义镜像，可以迅速复制出相同应用部署的云服务器ECS实例，之后将实例添加到SLB后端服务器组中，实现业务高可用。SLB可以同时配置四层和七层监听，及轮循、加权轮循、加权最小连接数等多种算法，合理分配后端ECS计算资源。
- 使用云数据库RDS（Relational Database Service），针对高并发场景进行特殊优化，同时引入线程池、并行复制、隐含主键等功能保证系统持续稳定和高吞吐。云数据库CloudDBA具有完备的性能监控数据，实时监控实例硬件使用指标、慢SQL，并给出各种优化建议，帮您快速定位并解决问题。

部署流程

假设您已拥有一台ECS实例，并且在该实例上部署了数据库与应用，您可以将单实例部署方式转变为单可用区或多可用区高可用架构。本教程指导您如何使用ECS、EIP、SLB和RDS产品来部署多可用区高可用架构。



1. 使用自定义镜像，部署多台相同配置的ECS实例。详情请参见复制ECS实例。
2. 创建负载均衡SLB实例，将实例添加到SLB后端服务器组中，用于跨可用区挂载ECS实例，实现业务的高可用性。详情请参见配置SLB实例。
3. 使用DTS将ECS实例上的自建数据库迁移至RDS实例，保障业务数据库不中断，自动备份保障数据不丢失。详情请参见迁移自建数据库至高可用版RDS实例。

25.2. 复制ECS实例

为了支持跨可用区容灾部署，本教程使用源实例的自定义镜像复制出三台ECS实例。一台与源实例位于同一可用区，两台与源实例位于同一地域下的不同可用区。

前提条件

- 已注册阿里云账号。如还未注册，请先完成账号注册。
- 已拥有待复制的源ECS实例。

操作步骤

1. 为ECS实例创建自定义镜像。
   i. 登录ECS管理控制台。
   ii. 在左侧导航栏，单击**实例与镜像 > 实例**。
   iii. 在顶部菜单栏处，选择地域。
   iv. 找到目标实例。在**操作**列中，单击**更多 > 磁盘和镜像 > 创建自定义镜像**。
   v. 输入镜像名称和描述信息。
   vi. 单击**创建**。

   > **说明** 创建镜像需要一段时间，请您耐心等待。

   在左侧导航栏，单击**实例与镜像 > 镜像**。当目标镜像的**进度**为100%、状态为**可用**时，表示镜像创建成功。

2. 使用自定义镜像创建3台ECS实例。
   i. 在左侧导航栏，单击**实例与镜像 > 镜像**。
   ii. 在**自定义镜像**页面，找到上一步创建的自定义镜像，在**操作**列，单击**创建实例**。
   iii. 在**自定义购买**页面，**镜像**区域已设置为您选择的自定义镜像。根据页面提示，完成其他配置项并购买1台ECS实例。

   其中：

   - **地域**：选择与源实例相同的地域。
   - **可用区**：选择与源实例相同的可用区。
   - **公网带宽**：取消勾选**分配公网IPv4地址**。

   更多配置详情，请参见使用向导创建实例。

   iv. 重复第i步和第ii步。在**自定义购买**页，**镜像**区域已设置为您选择的自定义镜像。根据页面提示，完成其他配置项并购买2台实例。

   其中：

   - **地域**：选择与源实例相同的地域。
   - **可用区**：选择与源实例不同的可用区。
   - **实例**区域：设置**购买实例数量**为2。
   - **公网带宽**区域：取消勾选**分配公网IPv4地址**。

   更多配置详情，请参见使用向导创建实例。

执行结果

在左侧导航栏，单击**实例与镜像 > 实例**。在**实例列表**页面，四台ECS实例的状态均为**运行中**，可用区两两相同。

后续步骤

配置SLB实例

25.3. 配置SLB实例

ECS实例复制完成后，在支持多可用区的地域创建负载均衡SLB实例，用于跨可用区挂载ECS实例，扩展应用系统对外服务能力、消除单点故障，提升应用系统的可用性。本文介绍SLB实例的部署方法。

前提条件

- 已复制三台ECS实例。更多信息，请参见复制ECS实例。
- 四台ECS实例的Web服务均已启动并正常运行。

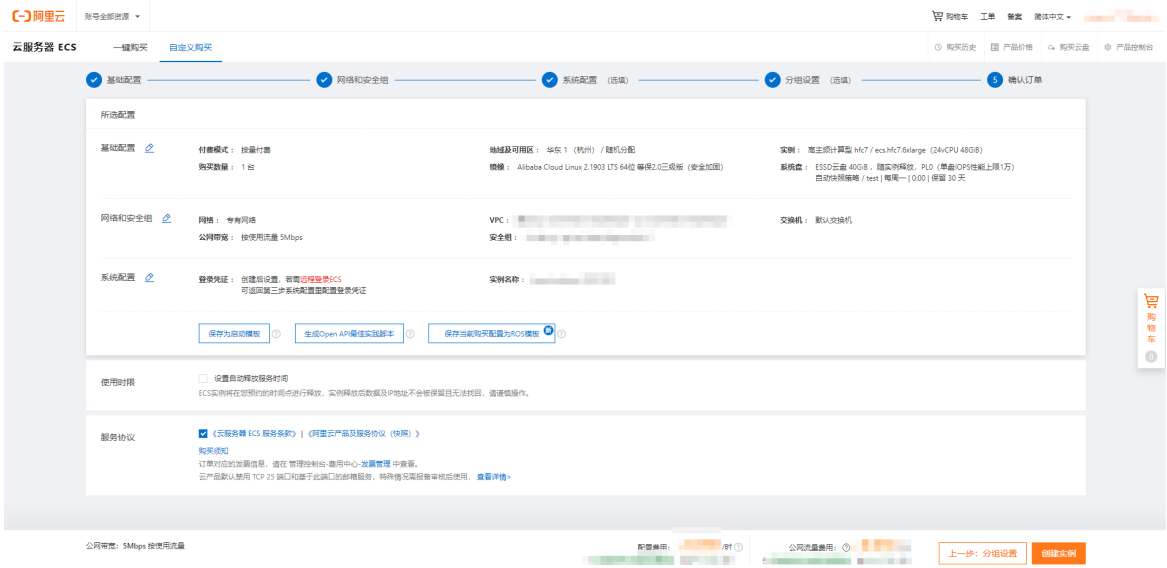
> 注意 若Web服务未运行，则SLB实例与ECS实例之间无法正常通信。

操作步骤

1. 创建SLB实例。具体操作，请参见创建实例。

   本教程使用的配置如下：

   - **地域**：必须与ECS实例位于同一地域。
   - **可用区类型**：选择**多可用区**。
   - **实例类型**：选择**私网**。
   - **网络类型**：选择**专有网络**。
   - **主可用区**和**备可用区**：按需配置。



2. 将源实例的公网IP转换为弹性公网IP。具体操作，请参见专有网络类型ECS公网IP转为弹性公网IP。

> **说明** 为避免影响业务，需保证源实例IP地址不变。因此，需要先将源实例的公网IP转换为弹性公网IP，与源实例解绑后，再将其绑定至高可用版SLB实例上。

3. 解绑源实例与弹性公网IP。

i. 在源实例的**IP地址**列，单击弹性IP地址链接。

ii. 在**弹性公网IP**页面，单击**解绑**。

iii. 单击**确定**。更多信息，请参见绑定ECS实例。

4. 绑定弹性公网IP至SLB实例。

i. 在**弹性公网IP**页面，找到与源实例解绑后的弹性公网IP。

ii. 在**操作**列，单击**绑定**。

iii. **实例类型**选择**SLB实例**，**SLB实例**选择刚创建的SLB实例，单击**确定**。更多信息，请参见绑定ECS实例。

5. 配置SLB实例。具体操作，请参见配置实例。

基本配置如下：

i. 在**协议&监听**页签，完成以下配置。

- **负载均衡协议**：选择**TCP**。
- **监听端口**：输入 `80` 。
- **调度算法**：按需选择。本教程选择**轮询**。
- 其他配置使用默认值。

ii. 单击**下一步**。在**后端服务器**页签，选择**默认服务器组**，单击**继续添加**添加ECS实例。

iii. 勾选源实例和已复制的三台ECS实例，单击**下一步：配置权重和端口号**。**端口**配置为80，其他值保持默认，单击**下一步**。

iv. 在**健康检查**页签，使用默认值，单击**下一步**。

v. 在**配置审核**页签，核对信息后，单击**提交**。

vi. 单击**确定**，返回**实例管理**页面，单击 。

当健康检查状态为**正常**时，表示后端ECS实例可以正常处理负载均衡转发的请求了。

> **说明** 健康检查需要几分钟时间，请您耐心等待并单击刷新图标查看状态。

执行结果

为方便测试，本教程分别在四台ECS实例上搭建了静态网页，以标识每台ECS实例。在浏览器中输入负载均衡实例的服务地址，测试负载均衡服务。由于**调度算法**为**轮询**，请求会轮流发往每台ECS实例。

后续步骤

迁移自建数据库至高可用版RDS实例

25.4. 迁移自建数据库至高可用版RDS实例

将源ECS实例上的数据库迁移至高可用版云数据库RDS，可实现数据库服务的高可用性、高可靠性、高安全性和高易用性。本教程以MySQL数据库为例，介绍如何使用DTS将ECS实例上的自建数据库迁移至高可用版RDS实例。

前提条件

- 已配置SLB实例，详情请参见配置SLB实例。
- 已创建高可用版RDS实例，并且部署方案为多可用区部署。如未创建，请参见创建RDS MySQL实例。
- 已为RDS实例创建账号。如未创建，请参见创建数据库和账号。
- 已为ECS实例上的自建数据库创建非root账号，用于DTS迁移。

  例如，您可以运行以下命令为MySQL数据库创建名为dts、密码为123456的账号。

```
grant all on *.*  to 'dts'@'%'  IDENTIFIED BY '123456';
```

背景信息

DTS提供的数据迁移功能能够支持同异构数据源之间的数据迁移，同时提供了库表列三级映射、数据过滤多种ETL特性。您可以使用DTS进行零停机迁移，在迁移过程中，源数据库正常持续提供服务，最大程度降低迁移对业务的影响。DTS支持的数据库类型请参见数据迁移。

操作步骤

1. 登录数据传输DTS控制台。
2. 在左侧导航栏，单击**数据迁移**。
3. 选择目标RDS实例所在地域，并单击**创建迁移任务**。
4. 配置迁移任务。

i. 配置任务名称。

您可以使用默认的名称或者自定义名称。

ii. 配置源库信息。

DTS支持通过公网、VPN网关、专线及智能网关访问的自建数据库。本教程使用的源数据库为ECS实例上的自建数据库。其他类型数据库的迁移方案，请参见DTS用户手册。

参数名称	描述
实例类型	ECS上的自建数据库。
实例地区	源ECS实例所在地域。
ECS实例ID	源ECS实例的实例ID。DTS支持经典网络及专有网络的ECS实例。
数据库类型	源ECS实例上自建数据库的类型。本示例中，数据库类型为MySQL。
端口	MySQL数据库监听的端口号。
数据库账号	源ECS实例上MySQL数据库的非root账号。 **说明** 数据库账号必须填写非root账号，否则测试连接时会报错。
数据库密码	非root账号对应的密码。

iii. 单击**源库信息**右下角的**测试连接**。

当返回的结果为**测试通过**时，表示源库连接正常。

iv. 配置目标库信息。

参数名称	参数值
实例类型	RDS实例。
实例地区	RDS实例所在地域。
RDS实例ID	RDS实例的实例ID。
数据库账号	RDS实例的账号。 为RDS实例创建账号，请参见创建数据库和账号。 **说明** 数据库账号必须填写非root账号，否则测试连接时会报错。
数据库密码	账号对应的密码。

v. 单击**目标库信息**右下角的**测试连接**。

当返回的结果为**测试通过**时，表示目标库连接正常。

vi. 单击**授权白名单并进入下一步**。

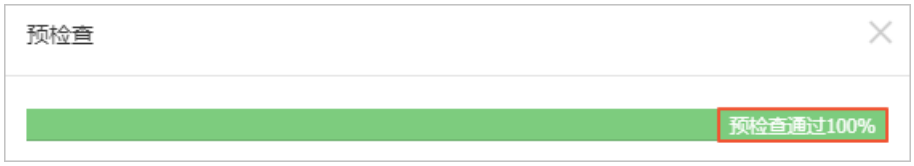
5. 配置迁移类型及迁移对象。
   i. 配置迁移类型。
      - 业务零停机迁移，请选择：结构迁移 + 全量数据迁移 + 增量数据迁移。
      - 全量迁移，请选择：结构迁移 + 全量数据迁移。
   ii. 配置迁移对象。

      在**迁移对象**框中单击要迁移的数据库对象，如数据库、表或列，然后单击>添加到**已选择对象**框中。

      > **说明**
      >
      > 默认情况下，数据库对象迁移到ECS自建MySQL实例后，对象名跟本地MySQL实例一致。如果迁移的数据库对象在源实例跟目标实例上名称不同，您需要使用DTS提供的对象名映射功能，详情请参见库表列映射。

6. 单击**预检查并启动**。

   在迁移任务正式启动之前，会预检查连通性、权限及日志格式等。下图表示预检查成功通过。



   预检查通过后，您可以在**迁移任务列表**中查看迁移任务的迁移状态及进度。

后续步骤

在应用程序中配置RDS实例的连接地址和账号密码，以连接到RDS实例。您还可以使用数据管理服务DMS（Data Management Service）或客户端管理RDS实例。具体操作，请参见通过客户端、命令行连接RDS MySQL。

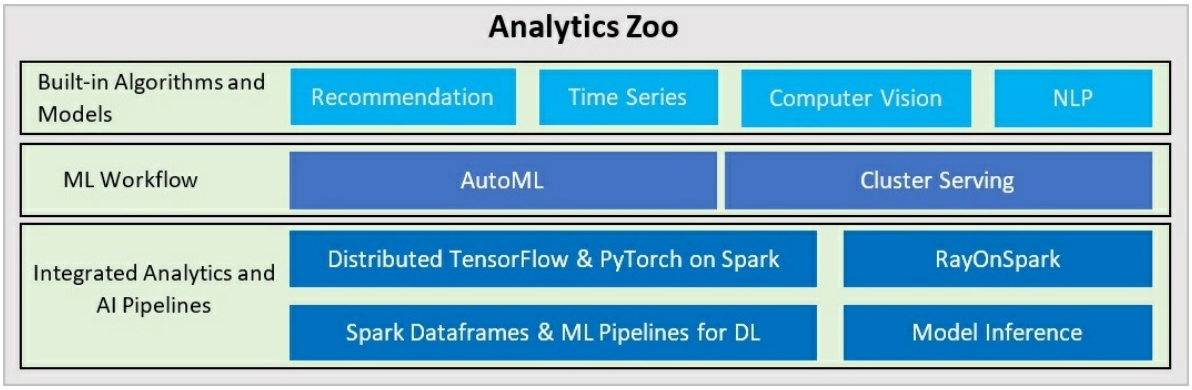
26.在ECS上使用Analytics Zoo对人工智能应用进行bfloat16加速

本章节将介绍在第七代高主频ECS实例上，利用Analytics Zoo和第三代智能英特尔®至强®可扩展处理器提供的bfloat16特性提高人工智能应用的性能。

背景信息

- 阿里云第七代高主频ECS实例构建于第三代神龙平台之上，基于第三代智能英特尔®至强®可扩展处理器创建。相对于上一代，阿里云ECS云服务器第七代高主频实例计算性能最大可以提升260%。在ECS上使用Analytics Zoo，可以利用Analytics Zoo的高级流水线特性，比如使用英特尔优化的深度学习框架（例如TensorFlow、PyTorch等）开发深度学习应用。
- 第三代智能英特尔®至强®可扩展处理器提供了业界领先、经工作负载优化的平台，并内置了AI加速功能--增强型英特尔®Deep Learning Boost（英特尔®DL Boost）。增强型英特尔®DL Boost通过业界首次对bfloat16的x86支持，增强了人工智能推理和训练性能。

  第三代智能英特尔®至强®可扩展处理器可运行复杂的人工智能工作负载。增强型英特尔®DL Boost将人工智能训练最高提升1.93倍，图像分类性能最高提升1.87倍，自然语言处理的训练性能提升1.7倍，推理提升1.9倍。新的bfloat16处理支持使医疗保健、金融服务和零售业的人工智能训练工作负载受益匪浅。
- Analytics Zoo是英特尔开源的统一的大数据和AI平台，它可以无缝的将TensorFlow、Keras、PyTorch等AI程序扩展到分布式Spark、Flink、Ray等大数据平台上运行。Analytics Zoo提供了以下特性：
  - 为基于TensorFlow、PyTorch、OpenVINO等的AI模型提供运行在大数据平台之上的端到端的流水线。例如开发者可以在Spark代码中嵌入TensorFlow或者PyTorch代码，进行分布式的训练和推理。开发者可以在Spark ML流水线中使用原生的深度学习支持如TensorFlow、Keras、PyTorch、BigDL等。
  - 为自动化的机器学习任务提供了高级ML工作流支持，例如自动的TensorFlow、PyTorch、OpenVINO等模型的分布式推理Cluster Serving以及可扩展的时序数据预测的AutoML功能。
  - 内置提供了Recommendation、Time Series、CV、NLP等应用常用的模型。



- bfloat16是一种业界广泛用于神经网络的数字格式。
- Resnet50是一个50层的残差网络（Residual Network），该神经网络广泛用于目标分类等领域。

操作步骤

如果您想在ECS上使用Analytics Zoo对人工智能应用进行bfloat16加速，按照以下步骤在ECS上加速人工智能应用：

1. 步骤一：创建高主频ECS实例

2. 步骤二：在ECS上准备带有bfloat16优化支持的Analytics Zoo环境
3. 步骤三：在ECS实例上训练Resnet50模型和bfloat16的性能提升

步骤一：创建高主频ECS实例

完成以下操作，创建一台ECS实例。

1. 前往实例创建页。
2. 创建一台hfc7实例。具体操作，请参见使用向导创建实例。

   在配置参数时，您需要注意当前场景支持的实例规格族包括hfc7和hfg7。具体规格，请参见高主频型。
3. 在实例列表中，找到创建的实例，单击实例ID。查看并确认实例规格。

步骤二：在ECS上准备带有bfloat16优化支持的Analytics Zoo环境

Analytics Zoo提供了预先创建的支持bfloat16的docker image，按照方法一可以轻松在阿里云ECS上获取Analytics Zoo的docker image。您也可以按照方法二使用Analytics Zoo nightly build来支持bfloat16。相关代码说明请参见代码示例：Analytics Zoo如何利用bfloat16加速深度模型训练。

- 方法一：在ECS上获取Analytics Zoo预先创建的docker image创建。
  i. 连接ECS实例。具体步骤，请参见连接ECS实例。
  ii. 运行以下命令安装并运行Docker。

  ```
  yum install docker-io -y
  systemctl start docker
  ```

  iii. 运行以下命令获取支持bfloat16的Analytics Zoo docker image。

  ```
  docker pull intelanalytics/analytics-zoo:0.8.1-bigdl_0.10.0-spark_2.4.3-bf16
  ```

  iv. 运行以下命令运行docker container。

  ```
  docker run -itd --name az1 --net=host  --privileged intelanalytics/analytics-zoo:0.8.
  1-bigdl_0.10.0-spark_2.4.3-bf16
  ```

  v. 运行以下命令进入container。

```
docker exec -it az1 bash
```

- 方法二：用户使用Analytics Zoo nightly build来支持bfloat16手动创建。
  i. 连接ECS实例。具体步骤，请参见连接ECS实例。
  ii. 运行以下命令下载并解压最新的Analytics Zoo nightly build pre-build package。

```
wget https://oss.sonatype.org/content/repositories/snapshots/com/intel/analytics/zoo/analytics-zoo-bigdl_0.11.1-spark_2.4.3/0.9.0-SNAPSHOT/analytics-zoo-bigdl_0.11.1-spark_2.4.3-0.9.0-20201026.210040-51-dist-all.zip
unzip analytics-zoo-bigdl_0.11.1-spark_2.4.3-0.9.0-{datetime}-dist-all.zip -d analytics-zoo
```

  iii. 运行以下命令安装git。

```
yum -y install git
```

  iv. 运行以下命令下载TensorFlow源代码。

```
git clone https://github.com/Intel-tensorflow/tensorflow.git
git checkout v1.15.0up1
```

  v. 运行以下命令编译TensorFlow。

```
bazel build --cxxopt=-D_GLIBCXX_USE_CXX11_ABI=0 --copt=-O3 --copt=-Wformat
--copt=-Wformat-security --copt=-fstack-protector --copt=-fPIC
--copt=-fpic --linkopt=-znoexecstack --linkopt=-zrelro
--linkopt=-znow --linkopt=-fstack-protector --config=mkl --define
build_with_mkl_dnn_v1_only=true --copt=-DENABLE_INTEL_MKL_BFLOAT16
--copt=-march=native
//tensorflow/tools/lib_package:libtensorflow_jni.tar.gz
//tensorflow/java:libtensorflow.jar
//tensorflow/java:libtensorflow-src.jar
//tensorflow/tools/lib_package:libtensorflow_proto.zip
```

  vi. 运行以下命令整理Analytics Zoo需要的库文件。

```
cd bazel-bin/tensorflow/tools/lib_package
mkdir linux-x86_64
tar -xzvf libtensorflow_jni.tar.gz -C linux_x86-64
rm libtensorflow_framework.so
rm libtensorflow_framework.so.1
mv libtensorflow_framework.so.1.15.0 libtensorflow_framework-zoo.so
cp ../../../../_solib_k8/_U@mkl_Ulinux_S_S_Cmkl_Ulibs_Ulinux___Uexternal_Smkl_Ulinux_Slib/* ./
```

  vii. 运行以下命令更新Analytics Zoo Jar。

```
cd ~/analytics-zoo/lib/
cp ~/tensorflow/bazel-bin/tensorflow/tools/lib_package/linux-x86_64 ./
jar -ufanalytics-zoo-bigdl_0.11.1-spark_2.4.3-0.9.0-SNAPSHOT-jar-with-dependencies.jar linux-x86_64/*
```

步骤三：在ECS实例上训练Resnet50模型和bfloat16的性能提升

1. 运行以下命令进入Analytic Zoo docker容器。

```
docker exec -it az1 bash
```

2. 运行以下命令配置spark，对*/opt/work/spark-2.4.3/conf/spark-defaults.conf*进行修改。

```
spark.authenticate=false
spark.ui.killEnabled=true
spark.eventLog.enabled=true
spark.history.ui.port=18080
spark.eventLog.dir=file:///var/log/spark/spark-events
spark.history.fs.logDirectory=file:///var/log/spark/spark-events
spark.shuffle.service.port=7337
spark.master=spark://$(hostname):7077
```

3. 运行以下命令启动spark master。

```
cd /opt/work/spark-2.4.3
./sbin/start-master.sh
```

4. 用numactl命令启动8个spark workers，每个worker绑定到12个vcpu。在*/opt/work/spark-2.4.3/bin*目录下创建如下脚本。

```
numactl -C 0-11 ./spark-class org.apache.spark.deploy.worker.Worker spark://$(hostname)
:7077 &
numactl -C 12-23 ./spark-class org.apache.spark.deploy.worker.Worker spark://$(hostname
):7077 &
numactl -C 24-35 ./spark-class org.apache.spark.deploy.worker.Worker spark://$(hostname
):7077 &
numactl -C 36-47 ./spark-class org.apache.spark.deploy.worker.Worker spark://$(hostname
):7077 &
numactl -C 48-59 ./spark-class org.apache.spark.deploy.worker.Worker spark://$(hostname
):7077 &
numactl -C 60-71 ./spark-class org.apache.spark.deploy.worker.Worker spark://$(hostname
):7077 &
numactl -C 72-83 ./spark-class org.apache.spark.deploy.worker.Worker spark://$(hostname
):7077 &
numactl -C 84-95 ./spark-class org.apache.spark.deploy.worker.Worker spark://$(hostname
):7077 &
```

5. 运行以下命令检查，返回 `8` 表示已启动了8个Worker。

```
jps | grep Worker | wc -l
```

6. 运行以下命令从github上下载resnet50示例代码。

```
git clone https://github.com/yangw1234/models-1.git
git checkout branch-1.6.1-zoo
```

7. 在*models-1/models/image_recognition/tensorflow/resnet50v1_5/training/mlperf_resnet*目录下运行run.sh脚本，用--use_bfloat16选项来开启bfloat16训练，不加此选项则默认为FP32训练。

```
# Register the model as a source root
export PYTHONPATH="$(pwd):${PYTHONPATH}"
export KMP_BLOCKTIME=0
# 8 instances
export OMP_NUM_THREADS=6
export KMP_AFFINITY=granularity=fine,compact,1,0
export KMP_SETTINGS=1
export ANALYTICS_ZOO_HOME=/opt/work/analytics-zoo/dist
export SPARK_HOME=/opt/work/spark-2.4.3
bash $ANALYTICS_ZOO_HOME/bin/spark-submit-python-with-zoo.sh --master
spark://$(hostname):7077 \
--executor-cores 1 --total-executor-cores 8 --driver-memory 20g --executor-memory 18g \
--conf spark.network.timeout=10000000 --conf spark.executor.heartbeatInterval=100000 \
imagenet_main.py 1 --model_dir ./logs --batch_size 128 --version 1 \
--resnet_size 50 --train_epochs 90 --data_dir /opt/ILSVRC2012/ --use_bfloat16
```

本次训练测试结果如下所示。

Resnet50模型训练	FP32	BF16	BF16相对FP32的性能提升
Throughput(images/sec)	119.636	212.315	1.775

代码示例：Analytics Zoo如何利用bfloat16加速深度模型训练

下面的代码用于说明Analytics Zoo如何利用bfloat16来加速深度学习模型的训练（例如Resnet50等）。在Analytics Zoo的docker image中已经包含，您不需要任何操作，仅作示例参考。

1. 通过以下代码将输入图片转换成bfloat16格式。

```
if use_bfloat16 == True:
dtype = tf.bfloat16
features = tf.cast(features, dtype)
```

2. 通过以下代码编写custom_dtype_getter。

```
DEFAULT_DTYPE = tf.float32
CASTABLE_TYPES = (tf.float16,tf.bfloat16)
def _custom_dtype_getter(getter, name, shape=None, dtype=DEFAULT_DTYPE,
      *args, **kwargs):
    if dtype in CASTABLE_TYPES:
      var = getter(name, shape, tf.float32, *args, **kwargs)
      return tf.cast(var, dtype=dtype, name=name + '_cast')
    else:
      return getter(name, shape, dtype, *args, **kwargs)
```

3. 通过以下代码创建variable_scope，并在该scope下构建模型。

```
def _model_variable_scope():
    return tf.compat.v1.variable_scope('resnet_model',
                    custom_getter=_custom_dtype_getter)
with _model_variable_scope():
    logits = _resnet_50_model(features)
```

4. 通过以下代码将logits装换成float32计算loss以保证数值稳定性（numerical stability）。

```
logits = tf.cast(logits, tf.float32)
```

5. 使用Analytics TFPark 进行分布式训练。具体操作，请参见Analytics Zoo分布式TensorFlow。