阿里云
云服务器 ECS

最佳实践

文档版本：20190807

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
	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。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能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，恢复业务所需时间约10分钟。
	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。	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。	单击 **确定**。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	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。	`swich` *`{stand | slave}`*

目录

1 安全

1.1 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、Service、Database、Cache层使用不同的安全组，暴露不同的出入规则和权限。
- 没有必要为每台实例单独设置一个安全组，控制管理成本。
- 优先考虑VPC网络。
- 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5个安全组，一个安全组最多可以包括100个安全组规则，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。所以，应尽可能地保持单个安全组的规则简洁。
- 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，从而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。比如，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。比如，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。

1.2 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：

- 默认只有同一个安全组的ECS实例可以网络互通。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。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。

安全规则的属性

安全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*（接受）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：网络类型。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。

比如，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*，具体设置可以参考以下描述，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）：公网（internet）。如果是专有网络VPC类型的只需要填写intranet，通过EIP实现公网访问。
- 授权策略（Policy）：允许（accept）。
- 规则方向（NicType）：入网。
- 协议类型（IpProtocol）：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（NicType）：入网。
- 协议类型（IpProtocol）：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。

对于弹性计算来说，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（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（NicType）：入网。
- 协议类型（IpProtocol）：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成梦魇。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。

在控制台上，您可以根据文档加入安全组的描述将一台实例加入安全组。

如果您对阿里云的OpenAPI非常熟悉，您可以参见弹性管理ECS实例，通过OpenAPI进行批量操作。对应的Python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实例移除安全组

如果ECS实例加入不合适的安全组，将会暴露或者Block您的服务，这时您可以选择将ECS实例从这个安全组中移除。但是在移除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，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。

1.3 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。

合理规划和区分不同的安全组将使得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明晰和简单，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。

  1. 梳理当前提供的公网服务暴露的端口和协议，例如 80、443。
  2. 新创建一个安全组，例如 SG_WEB， 然后添加相应的端口和规则。

     说明:
     授权策略：允许，协议类型：ALL， 端口：80/80，授权对象：0.0.0.0/0， 授权策略：允许，协议类型：ALL，端口：443/443，授权对象：0.0.0.0/0。
  3. 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。

     说明:

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]。

4. 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. 在 ECS 控制台中，选择 安全组管理。
   b. 选择 SG_WEB > 管理实例 > 添加实例，选择实例 ECS_WEB_1 加入到新的安全组 SG_WEB 中，确认 ECS_WEB_1 实例的流量和网络工作正常。
5. 将 ECS_WEB_1 从原来的安全组中移出。
   a. 在 ECS 控制台中，选择 安全组管理。
   b. 选择 SG_WEB > 管理实例 > 添加实例，选择 ECS_WEB_1 ，从 SG_CURRENT 移除，测试网络连通性，确认流量和网络工作正常。
   c. 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。
6. 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的话，最好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，也应该同时合理的规划安全组。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。为了更合理的做网络隔离，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了线上影响线上的稳定性。

  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。

仅对需要公网访问子网或者云服务器分配公网 IP

不论是经典网络还是专有网络 (VPC) 中，合理的分配公网 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。

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，具体配置参见SNAT。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。

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，但是毕竟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 而不是密码登录。

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。

1.4 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使用自动快照策略。

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。具体步骤请参见管理专有网络。

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。

收集安全情报

在互联网安全领域，安全工程师和黑客比拼的就是时间。云盾态势感知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。详细信息请参见云盾态势感知。

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。

1.5 如何提高ECS实例的安全性

云服务器 ECS 实例是一个虚拟的计算环境，包含了 CPU、内存、操作系统、磁盘、带宽等最基础的服务器组件，是 ECS 提供给每个用户的操作实体。

我们基本可以理解为一个实例就等同于一台虚拟机，那么我们在本地维护的虚拟机一般会做虚拟机实例级别的安全防护，以防止虚拟机被攻击和入侵等。同样的，云上的ECS实例也需要做安全性防护。

ECS实例放置在云上，除了置身于阿里云自身的安全平台外，用户也需要根据实际的需求进一步定制化安全，所以说ECS的安全是阿里云和用户共同构建的。如果ECS实例没有安全的防护，可能会带来不少不良的影响，比如遭受到DDoS而导致业务中断，比如受到Web入侵而导致网页被篡改、挂马，比如被注入而导致信息和数据泄漏等，影响ECS的使用和无法正常提供服务。

一般可以通过设置安全组、AntiDDoS、态势感知、安装安骑士、接入Web应用防火墙等方式提高ECS实例的安全性。下面就从实例层面分别讲解一下如何提高ECS实例的安全性。

安全组是一个逻辑上的分组，这个分组是由同一个地域（Region）内具有相同安全保护需求并相互信任的实例组成。每个实例至少属于一个安全组，在创建的时候就需要指定。同一安全组内的实例之间网络互通，不同安全组的实例之间默认内网不通。可以授权两个安全组之间互访。

设置安全组

- 设置安全组的好处

  安全组是一种虚拟防火墙，具备状态检测包过滤功能。安全组用于设置单台或多台云服务器的网络访问控制，它是重要的网络安全隔离手段，用于在云端划分安全域。安全组规则可以允许或者禁止与安全组相关联的云服务器 ECS 实例的公网和内网的入出方向的访问。

  如果没有很好地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，在一定程度上为攻击者敞开了大门。

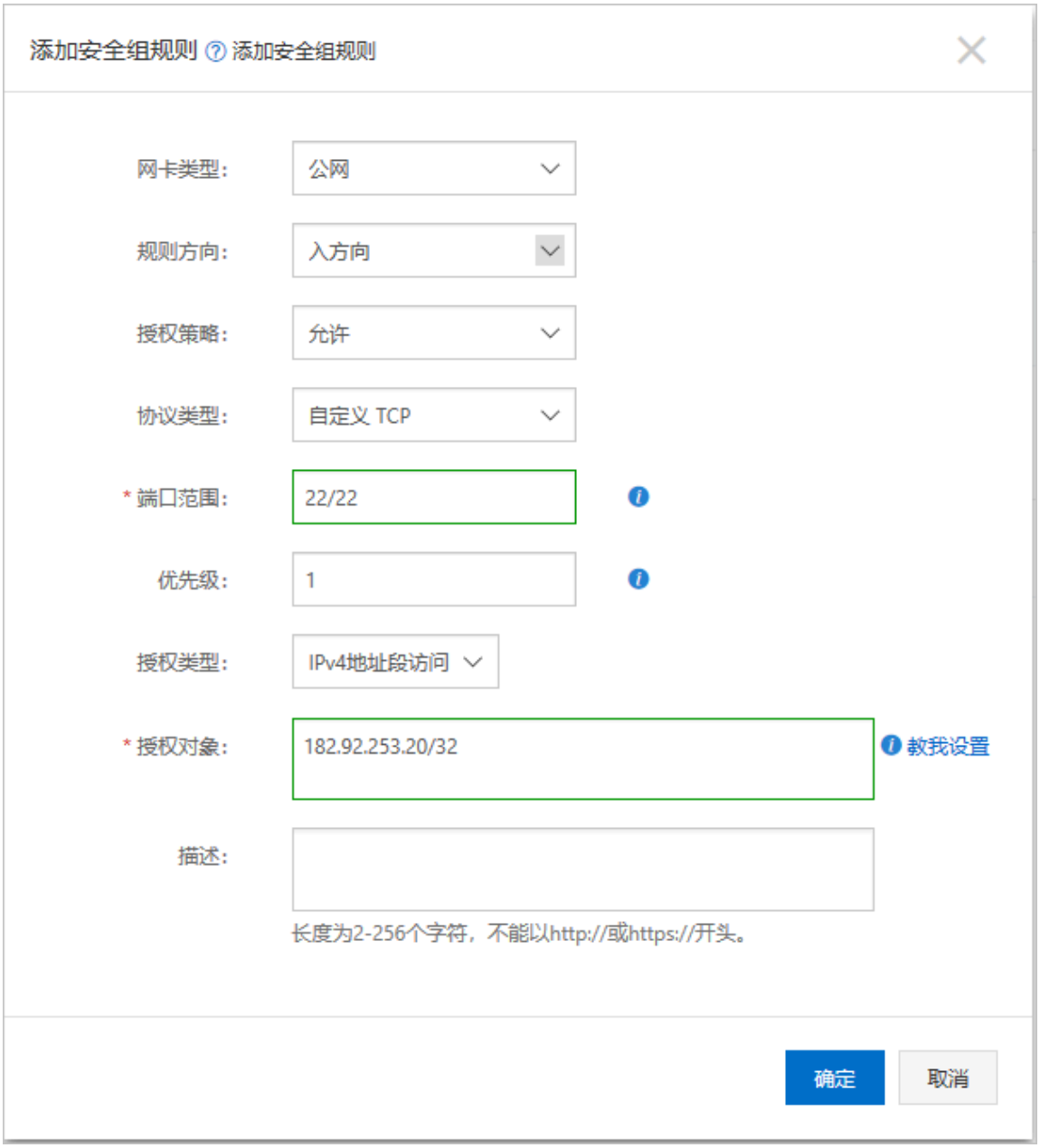
- 操作步骤

1. 登录 云服务器管理控制台。
2. 单击左侧导航中的 安全组。
3. 选择地域。
4. 单击添加安全组规则。
5. 在弹出的对话框中，分别设置网络类型、规则方向、授权策略、协议类型、端口范围、授权类型、授权对象和优先级。
6. 点击 确定，成功为该安全组授权一条安全组规则 。

下面结合一个案例来阐述一下，比如只允许特定IP远程登录到实例。

通过配置安全组规则可以设置只让特定 IP 远程登录到实例。只需要在公网入方向配置规则就可以了，以 Linux 服务器为例，设置只让特定 IP 访问 22 端口。

1. 添加一条公网入方向安全组规则，允许访问，协议类型选择 TCP，端口写 22/22，授权类型为地址段访问，授权对象填写允许远程连接的 IP 地址段，格式为 x.x.x.x/xx，即 IP地址/子网掩码，本例中的地址段为 182.92.253.20/32。优先级为 1。



2. 再添加一条规则，拒绝访问，协议类型选择 TCP，端口写 22/22，授权类型为地址段访问，授权对象写所有 0.0.0.0/0，优先级为 2。

最终的效果如下：

来自 IP 182.92.253.20 访问 22 端口优先执行优先级为 1 的规则允许。

来自其他 IP 访问 22 端口优先执行优先级为 2 的规则拒绝了。

AntiDDoS

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基础防护免费为阿里云用户提供最高5G的默认DDoS防护能力。

阿里云在此基础上，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，用户最高可获得100G以上的免费DDoS防护资源。

- 为什么需要AntiDDoS

  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，将多个计算机联合起来作为攻击平台，对一个或多个目标发动DDoS攻击，从而成倍地提高拒绝服务攻击的威力，影响业务和应用正常对用户提供服务。

  使用AntiDDoS，无需采购昂贵清洗设备，可以在受到DDoS攻击不会影响访问速度，带宽充足不会被其他用户连带影响，保证业务可用和稳定。

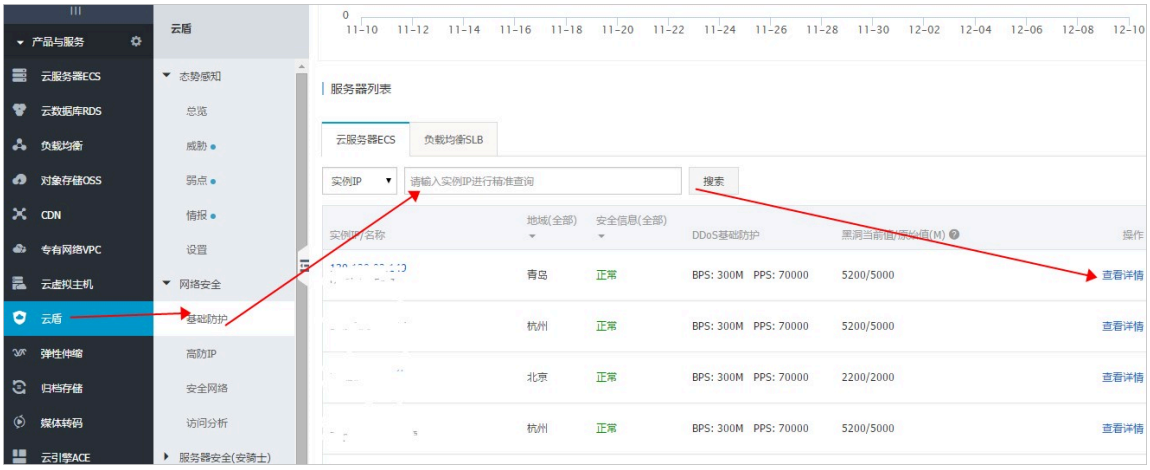
· 操作步骤

1. 进入阿里云官网，登录到 管理控制台。
2. 输入用户名密码。
3. 通过云盾 > DDOS防护 > 基础防护，查看基础防护配置。
4. 可以加入安全信誉防护联盟。勾选服务条款，点选加入安全信誉防护联盟加入联盟。如下图所示。

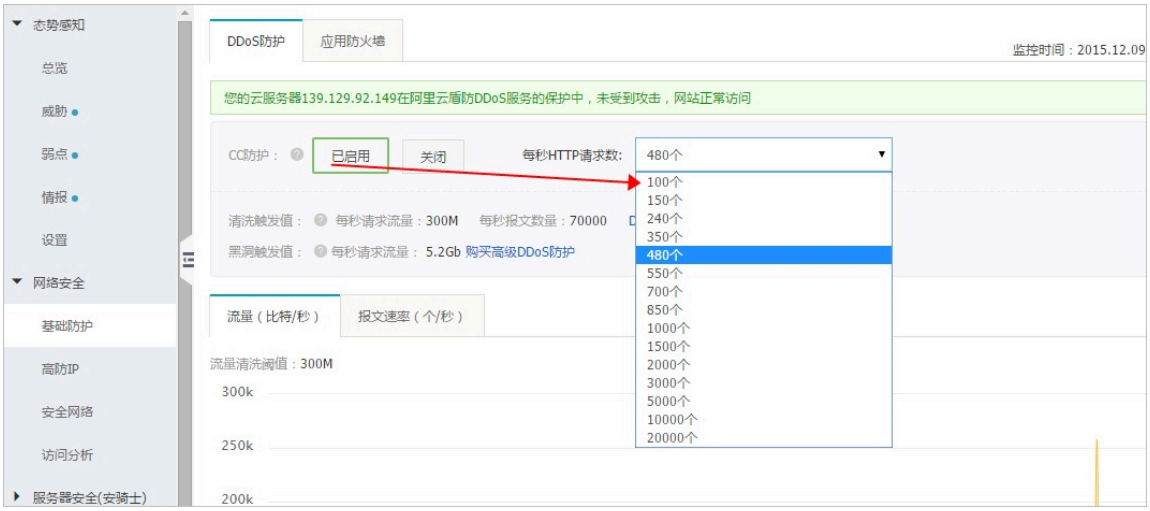
云盾DDoS基础版提供不大于5G的DDoS防护，在此基础上推出了安全信誉防护联盟计划，您可通过加入此联盟，在获得原默认防护能力基础上，会得到免费增量防护带宽机会。

加入联盟后，可查看自己的安全信誉分，并查看安全信誉组成，维护安全信誉，获得更大的防护能力。加盟成功后在基础防护界面显示如下信誉界面。

5. 在基础防护页面，点击对应ECS服务器的查看详情，如果服务器数量比较多，可以在云服务器ecs列表中通过实例IP和实例名称搜索服务器，再点击对应服务器的查看详情。

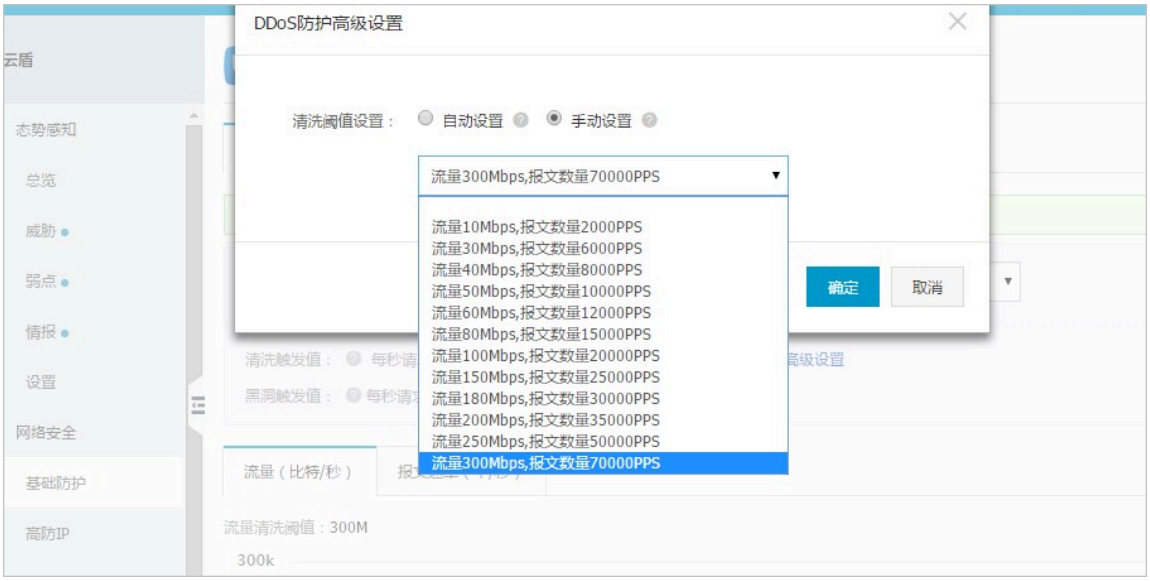


6. 进入页面后，可以在CC防护页面点击已启用开启CC防护，点击关闭则关闭CC防护功能，在每秒HTTP请求数可以对每秒http请求数设置清洗阈值，达到阈值后便会触发云盾的清洗。



7. 如果购买了高级DDoS防护，可以点击DDoS防护高级设置可以设置清洗阈值，选择自动设置后系统会根据云服务器的流量负载动态调整清洗阈值，选择手动设置可以手动对流量和报

文数量的阈值进行设置，当超过此阈值后云盾便会开启流量清洗(建议如果网站在做推广或者活动时适当调大)。



态势感知

态势感知态势感知提供的是一项SAAS服务，即在大规模云计算环境中，对那些能够引发网络安全态势发生变化的要素进行全面、快速和准确地捕获和分析。然后，把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，最终产出未来可能产生的安全事件的威胁风险，并提供一个体系化的安全解决方案。

- 态势感知的优势

  对“渗透攻击”有所感知，以云计算数据平台支撑，因此具有强大的安全数据分析能力，对各种常见类型的攻击可以实时分析和展示。

- 操作步骤

1. 在 管理控制台的态势感知中点击免费开启服务，即可使用态势感知。

2. 通过紧急时间、威胁、弱点、情报、日志等方面，辅以直观的可视化的分析，让安全一目了然。

安装安骑士

服务器安全(安骑士)是云盾推出的一款服务器安全运维管理产品。通过安装在服务器上的轻量级Agent插件与云端防护中心的规则联动，实时感知和防御入侵事件，保障服务器的安全。

- 安装安骑士的好处

  安骑士是很轻量的，服务器上运行的Agent插件，正常状态下只占用1%的CPU、10MB内存。安骑士可以自动识别服务器的Web目录，对服务器的Web目录进行后门文件扫描，支持通用Web软件漏洞扫描和Windows系统漏洞扫描，对服务器常见系统配置缺陷进行检测，包括可疑系统账户、弱口令、注册表等进行检测。

  我们可以将安骑士理解为ECS实例上的防病毒软件，如果没有安骑士，相当于少了一个可靠的卫士，我们ECS实例的健康性水平也会相应降低。

· 操作步骤

1. 服务器安全(安骑士)Agent插件目前集成于安全镜像中，在创建实例时选择安全加固后，您可以进入安骑士控制台-配置中心，查看每台服务器的在线状态。

2. 若不在线，请按照如下方式下载并安装。

a. 进入服务器安全(安骑士)控制台-设置-安装Agent页面，根据页面提示获取最新版本下载地址，以管理员权限在服务器上运行并安装。

b. 对于非阿里云服务器，在安装过程中会提示输入验证Key，这个验证Key用于关联阿里云账号，通过阿里云账号在安骑士控制台使用相关功能，验证key会显示在安装页面中。

c. 大约安装完成2分钟后在云盾·服务器安全(安骑士)控制台-配置中心里查看到在线数据，阿里云服务器将会从离线变成在线，非阿里云机器会新增在服务器列表中。

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

  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，淘宝天猫都在用。除了具有强大Web防御能力，还可以指定网站的专属防护，背后是大数据的安全能力。适用于在金融、电商、o2o、互联网+、游戏、政府、保险、政府等各类网站的Web应用安全防护上。

  如果缺少WAF，光靠前面提到的防护措施会存在短板，例如在面对如数据泄密、恶意CC、木马上传篡改网页等攻击的时候，就不能拿很好地防护了，可能会导致Web入侵。

· 操作步骤

1. 控制台配置。

a. 登录阿里云控制台，找到云盾 > Web应用防火墙 > 域名配置，点击添加域名按钮。

b. 弹出的对话框中输入相关信息：

c. 获取CNAME。配置好域名后，WAF会自动分配给当前域名一个CNAME，可点击域名信息来查看：

d. 上传HTTPS证书和私钥（仅针对HTTPS站点）。如果防护HTTPS站点，必须上传服务器的证书和私钥到WAF，否则访问HTTPS站点会有问题。勾选HTTPS后，会看到红色的“异常”字样，提示当前证书有问题，点击上传证书来上传：

e. 接入状态异常排查，刚添加完域名时，接入状态可能会提示异常。这是正常的，待修改DNS使用CNAME解析接入WAF后，或者是有正常流量经过WAF以后会变成正常的。

2. 放行回源IP段。

3. 本地验证。

a. 以前面步骤中添加的域名“www.aliyundemo.cn”为例，hosts文件应该添加如下内容，其中前面的IP地址为对应的WAFIP地址，WAF的IP可以通过ping提供的CNAME来获得。

```
# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost

    .255 www.aliyundemo.cn
```

b. 修改hosts文件后保存。然后本地ping一下被防护的域名，预期此时解析到的IP地址应该是刚才绑定的WAF IP地址。如果依然是源站地址，可尝试刷新本地的DNS缓存（Windows的cmd下可以使用ipconfig/flushdns命令）。

c. 确认hosts绑定已经生效（域名已经本地解析为WAF的IP）后，打开浏览器，输入该域名进行访问，如果WAF的配置正确，网站预期能够正常打开。

d. 尝试一下手动模拟一些简单的web攻击命令，如www.aliyundemo.cn/?alert(xss) 预期WAF能够弹出阻拦页面：

4. 通过DNS供应商或者其他域名解析系统，修改DNS解析。

阿里云给我们ECS实例的安全性提供了这么多的安全产品保驾护航，我们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实例接受到的侵害，使其稳定、持久地运行。

1.6 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 1. 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 100 条的限制，另外后期维护工作量比较大。

- **设置方法：**

  1. **选择需要互通的实例，进入本实例安全组。**
  2. **选择需要配置安全组，单击配置规则。**
  3. **单击内网入方向，并单击添加安全组规则。**
  4. **按以下描述添加安全组规则：**

     - **授权策略：允许。**
     - **协议类型：根据实际需要选择协议类型。**
     - **端口范围：根据您的实际需要设置端口范围，格式为*#####/#####*。**
     - **授权类型：地址段访问。**
     - **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*。**

方法 2. 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安全组，绑定同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 3. 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：

  1. 新建一个安全组并命名，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 4. 安全组互信授权

- 适用场景：如果您的网络架构比较复杂，各实例上部署的应用都有不同的业务角色，您就可以选择使用安全组互相授权方式。
- 优点：安全组规则结构清晰、阅读性强、可跨账户互通。
- 缺点：安全组规则配置工作量较大。

· 设置方法：

1. 选择需要建立互信的实例，进入本实例安全组。
2. 选择需要配置安全组，单击配置规则。
3. 单击内网入方向，并单击添加安全组规则。
4. 按以下描述添加安全组规则：

- 授权策略：允许。
- 协议类型：根据您的实际需要选择协议类型。
- 端口范围：根据实际需求设置。
- 授权类型：安全组访问。
- 授权对象：

■ 如果您选择本账号授权：按照您的组网要求，将有内网互通需求的对端实例的安全组 ID 填入授权对象即可。

■ 如果您选择跨账号授权：授权对象应填入对端实例的安全组 ID，账号 ID是对端账号 ID（可以在账号管理 > 安全设置里查到）。

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。

1.7 修改服务器默认远程端口

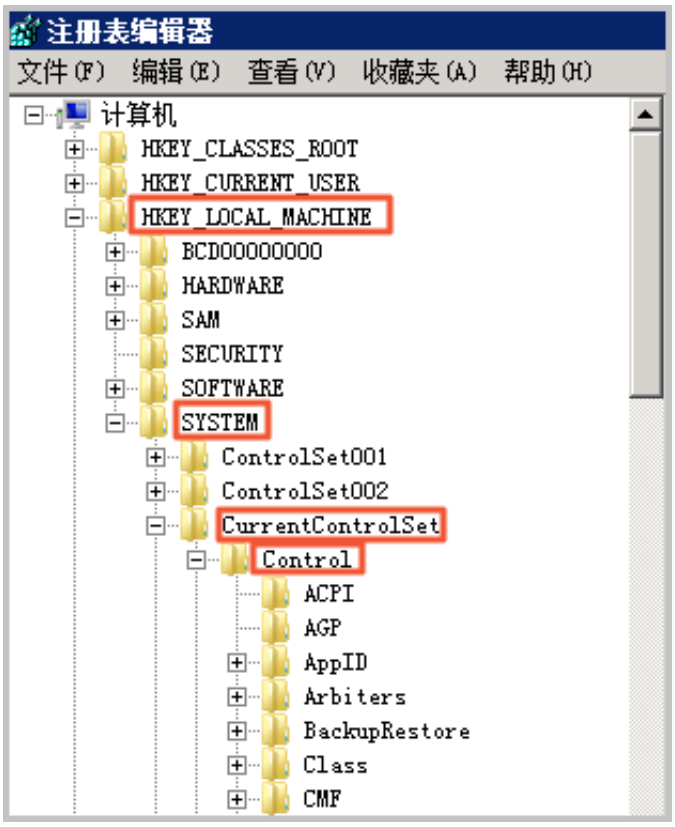
本文介绍如何修改 Windows 和 Linux 服务器的默认远程端口。

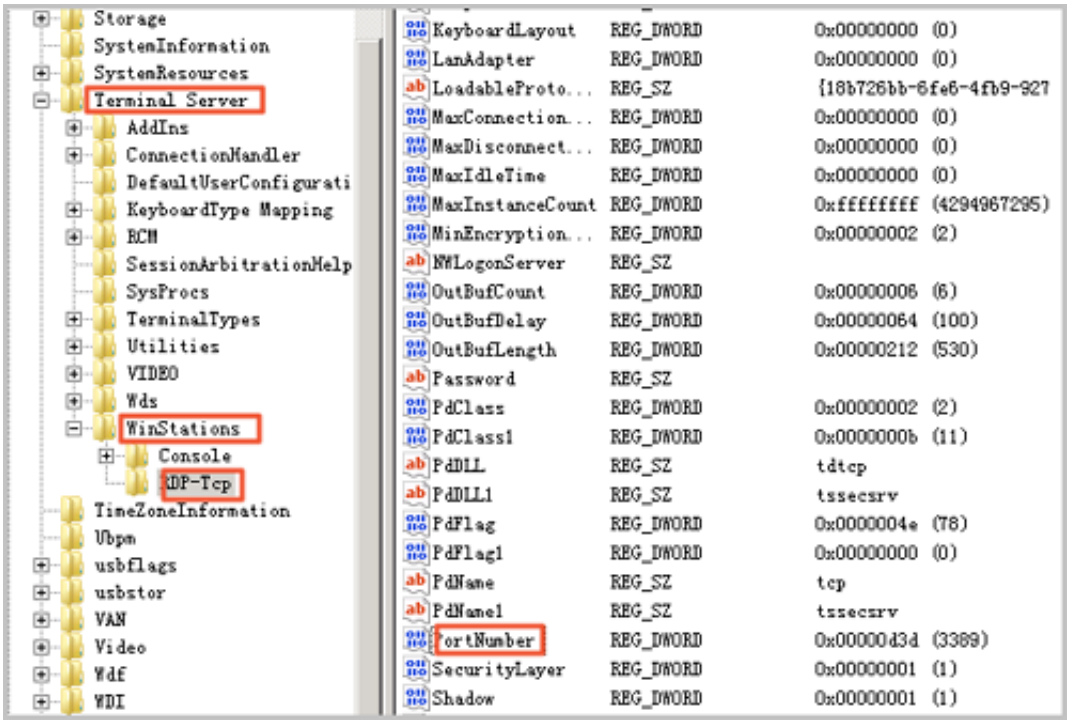
修改 Windows 服务器默认远程端口

本节以 Windows Server 2008 为例介绍如何修改 Windows 服务器默认远程端口。

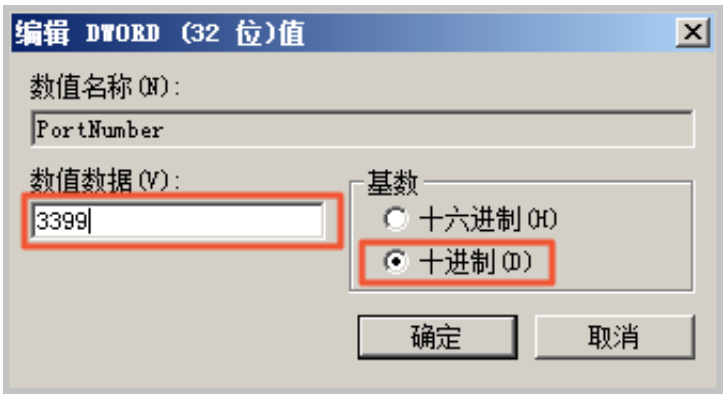
1. 远程连接并登录到 Windows 实例。
2. 运行*regedit.exe*打开注册表编辑器。

3. **找到如下注册表子项：***HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp\PortNumber*





4. 在弹出的对话框中，选择十进制，在数值数据中输入新的远程端口号，在本例中即 3399。单击确定。



5. （可选）如果您开启了防火墙，需要将新的端口号添加到防火墙并设置允许连接。

   具体方法参见设置 ECS 实例远程连接防火墙。

6. 登录 ECS管理控制台，找到该实例，选择更多 > 实例状态 > 重启。

7. 实例重新启动后，在实例的右侧单击管理，进入实例详情页面。选择本实例安全组。

8. 在安全组列表页面，找到相应的安全组，单击配置规则。

9. 在安全组规则页面，单击添加安全组规则。根据实际的使用场景来定义安全规则，允许新配置的远程端口进行连接。关于如何设置安全组参见添加安全组规则。

10.以上步骤完成后，远程访问服务器，在远程地址后面添加新远程端口号即可连接实例。例如：192.168.1.2:3399。

说明：

调整 3389 端口后，使用 Mac 的远程桌面连接客户仅支持默认的 3389 端口。

修改 Linux 服务器默认远程端口

本节以 CentOS 6.8 为例介绍如何修改 Linux 服务器默认远程端口。

说明：

不要直接修改 22 端口，先添加需要的默认远程端口。之所以先设置成两个端口，测试成功后再关闭一个端口，是为了防止在修改配置文件及网络调试过程中，万一出现新端口无法连接的情况下，还能通过 22 端口进行登录调试。

1. 远程连接并登录到 Linux 实例。
2. 运行 `vim /etc/ssh/sshd_config` 命令。
3. 在键盘上按“I”键，进入编辑状态。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*。
4. 在键盘上按“Esc”，输入：wq退出编辑状态。
5. 执行以下命令重启实例，之后您可以通过 22 端口和 1022 端口 SSH 登录到 Linux 实例。

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

6. （可选）配置防火墙。使用 CentOS 7 以前的版本并开启默认防火墙 iptables 时，应注意 iptables 默认不拦截访问，如果您配置了 iptables 规则，需要执行`iptables -A INPUT -p`

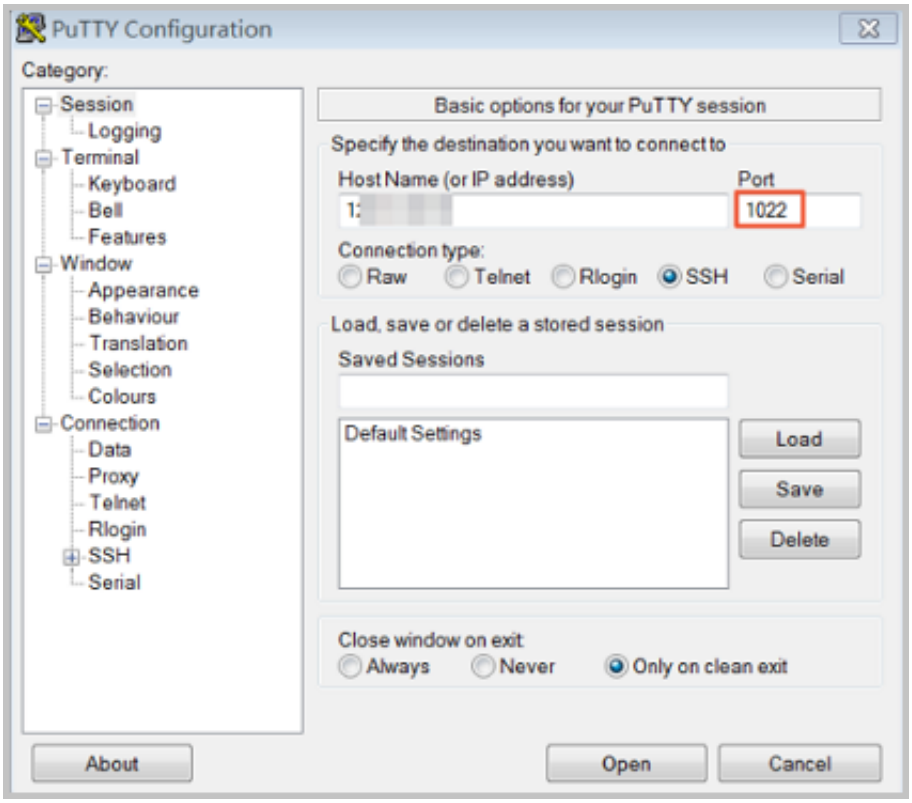
`tcp --dport 1022 -j ACCEPT`配置防火墙。然后执行`service iptables restart`重启防火墙。

> 说明:
> CentOS 7 以后版本默认安装 Firewalld。如果您已经启用 firewalld.service，需要放行 TCP 1022 端口：运行命令 firewall-cmd --add-port=1022/tcp --permanent。返回结果为 success 即表示已经放行 TCP 1022 端口。

7. 登录 ECS管理控制台，找到该实例，选择管理。
8. 进入实例详情页面。选择本实例安全组。

9. 在安全组列表页面，找到相应的安全组，单击配置规则。
10. 在安全组规则页面，单击添加安全组规则。根据实际的使用场景来定义安全规则，允许新配置的远程端口进行连接。关于如何设置安全组参见添加安全组规则。

11.使用 SSH 工具连接新端口，来测试是否成功。登录时在 Port 一栏输入新修改的端口号，在本例中即 1022。



12.使用 1022 端口连接成功后，再次运行`vim /etc/ssh/sshd_config`命令，将 Port 22 删除。

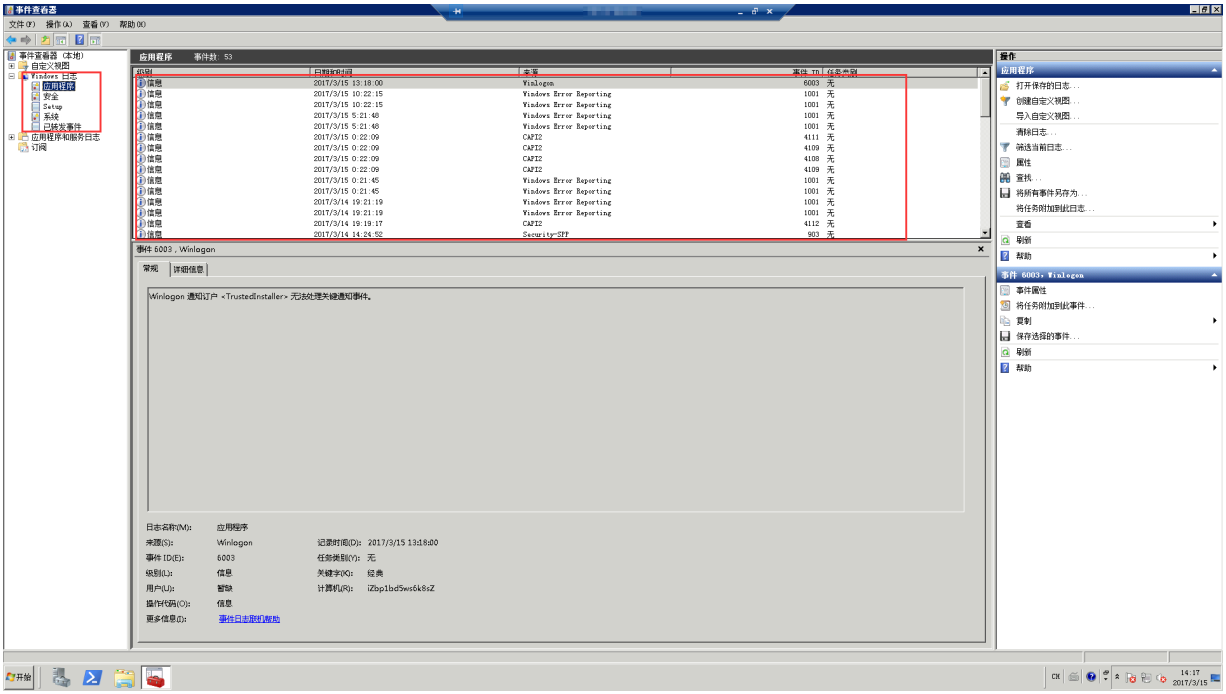
13.运行 `/etc/init.d/sshd restart` 命令重启实例，服务器默认远程端口修改完成。再次登录时使用新端口号登录即可。

1.8 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。当服务器被入侵或者系统（应用）出现问题时，管理员可以根据日志迅速定位问题的关键，再快速处理问题，从而极大地提高工作效率和服务器的安全性。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。本文以Windows Server 2008 R2为例，简单地介绍四种日志的使用和简要分析。

进入事件查看器

进入事件查看器：打开 运行 窗口，输入 `eventvwr`，打开 事件查看器。



之后，您可以在 事件查看器 里查看以下四种日志。

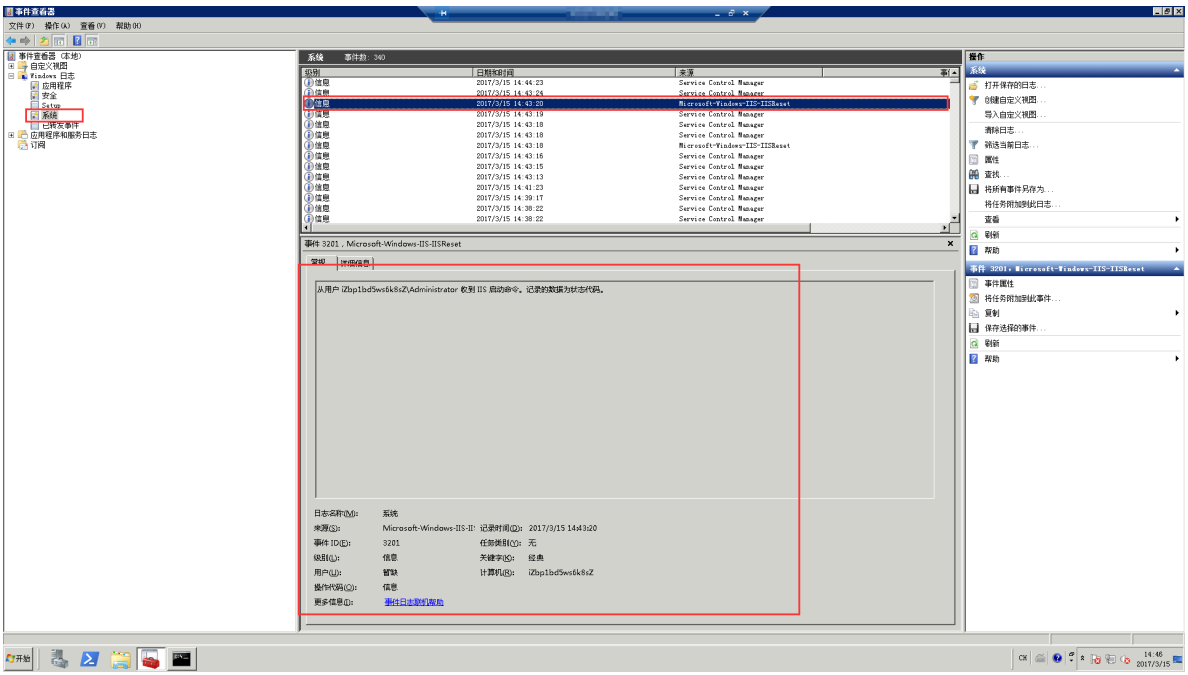
说明:

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**

  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。

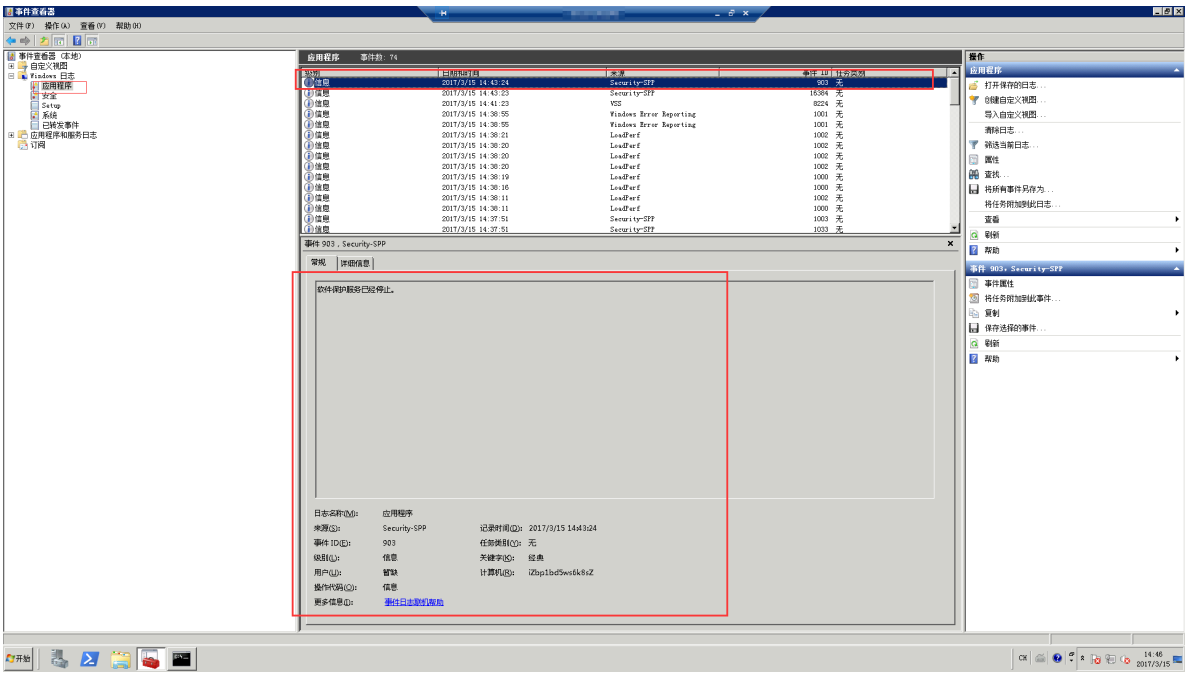
  系统组件所记录的事件类型由Windows预先确定。



- 应用程序日志

  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。

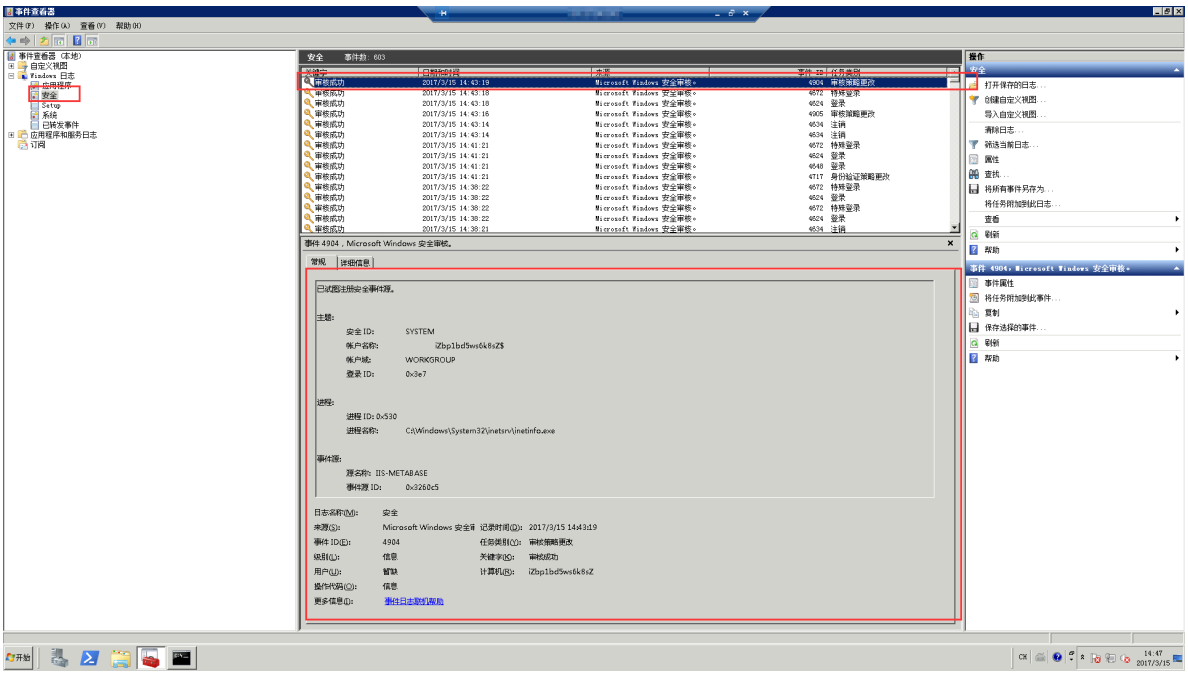
  程序开发人员决定记录哪些事件。

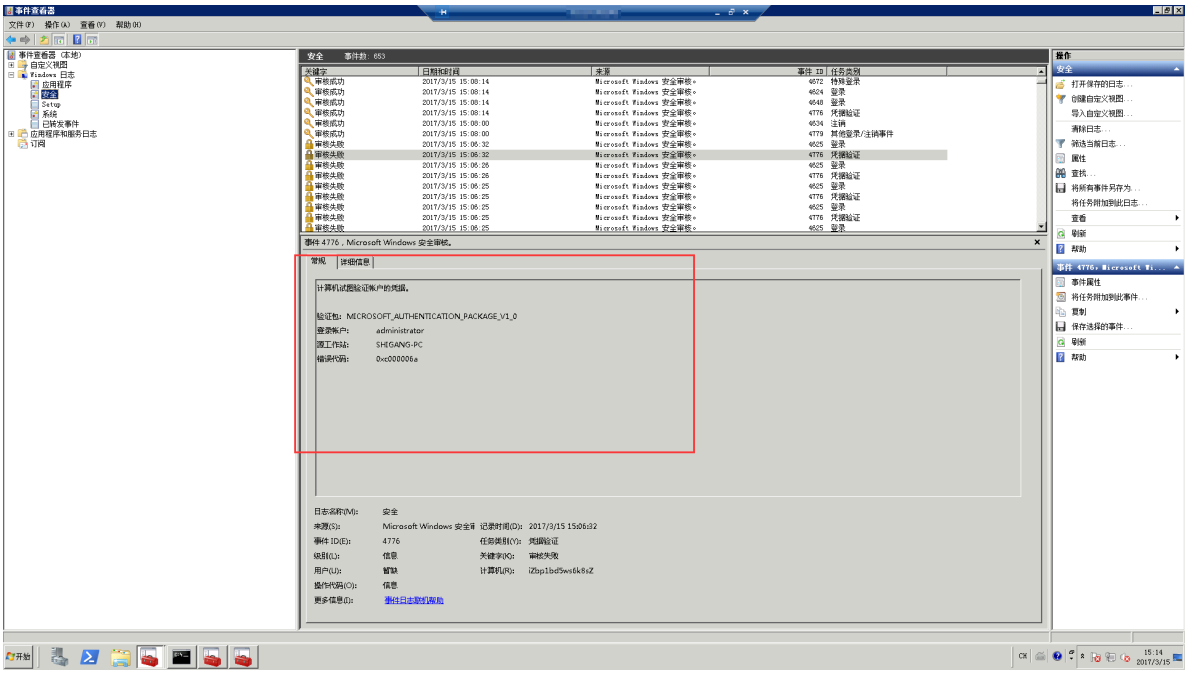


- 安全日志

  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。

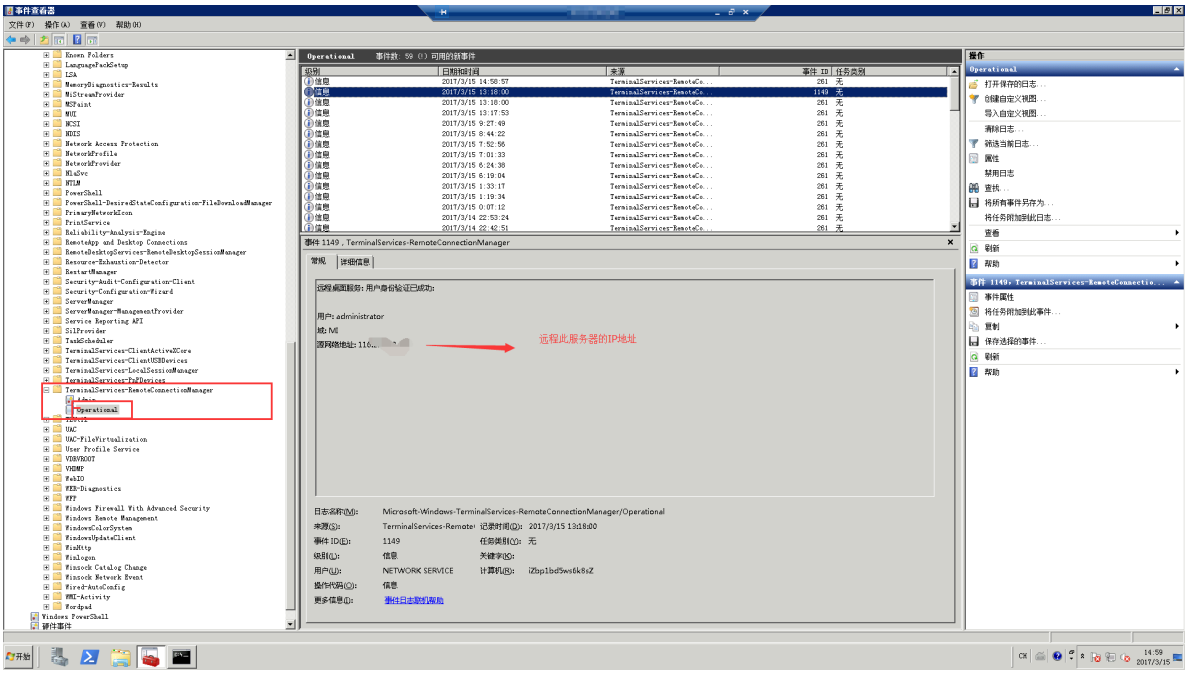
  管理员可以指定在安全日志中记录什么事件。例如，如果已启用登录审核，则安全日志将记录对系统的登录尝试。





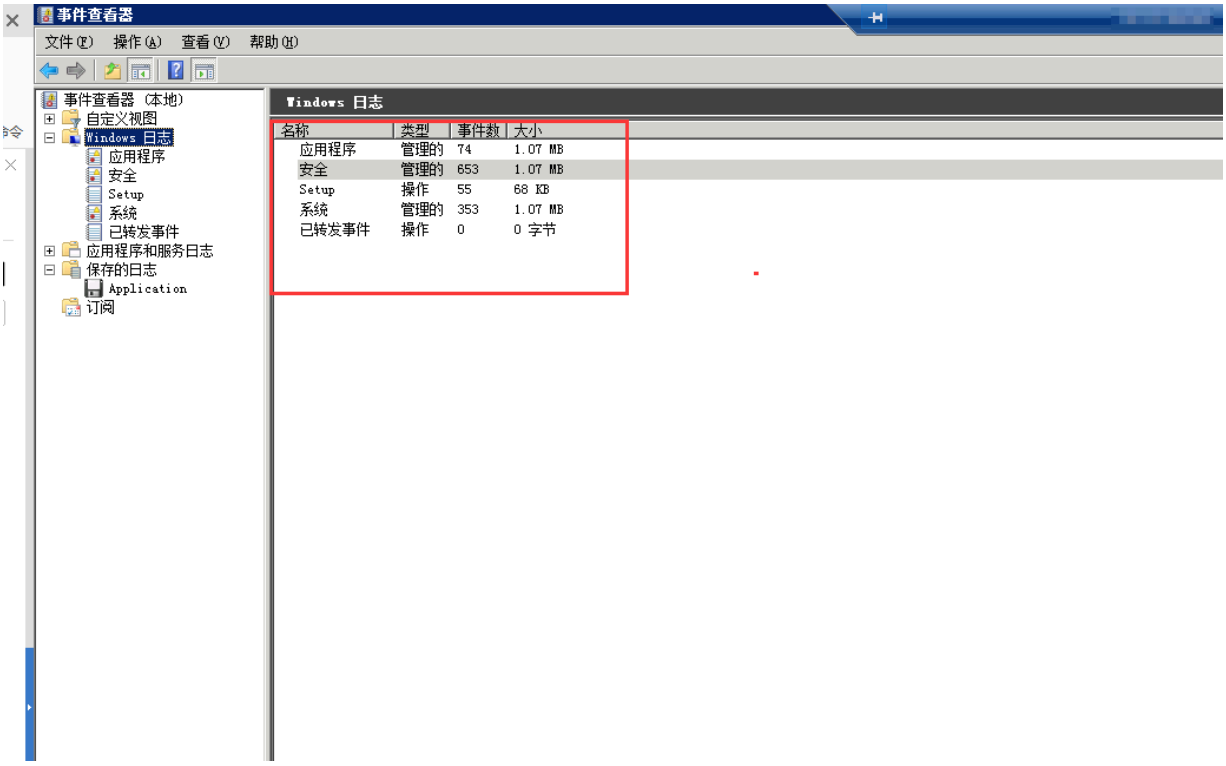
- 应用程序和服务日志

  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

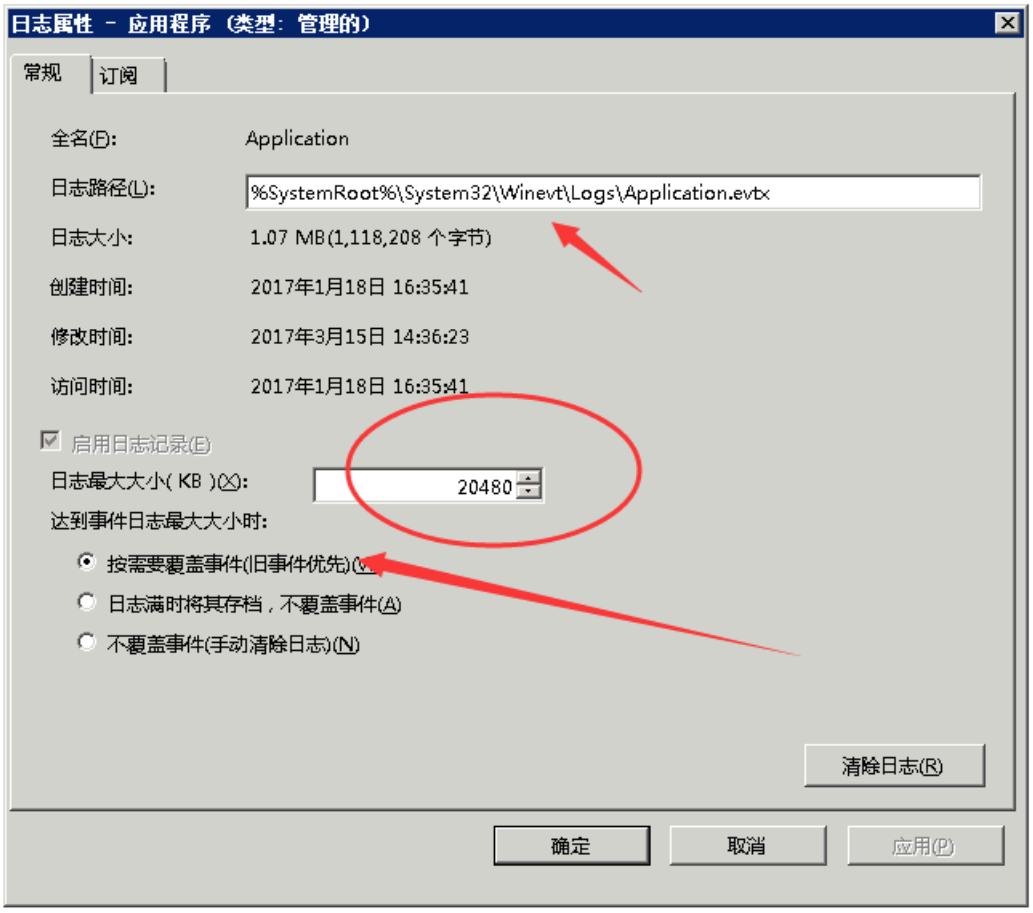
日志默认保存在系统盘里面。日志最大值默认是20 MB，超过20 MB时会覆盖之前的事件。您可以根据自己的需求修改。



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，选中一个日志目录，右键这一类日志，如截图所示的 应用程序。

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。



相关链接

云服务器 ECS Windows 安全审计日志简要说明

1.9 高级安全Windows防火墙概述以及最佳实践

本文简单介绍Windows防火墙的概念，给出使用场景并列出了常见的防火墙操作。

简介

在Windows NT6.0之后微软推出了高级安全Windows防火墙(简称WFAS)，高级安全Windows防火墙是分层安全模型的重要部分， 通过为计算机提供基于主机的双向网络通讯筛选， 高级安全Windows防火墙 阻止未授权的网络流量流向或流出本地计算机。高级安全 Windows 防火墙 还是用网络感知，以便可以将相应安全设置应用到计算机连接到的网络类型。Windows 防火墙和 Internet 协议保护 (sec) 配置设置集成到名为 高级安全 Windows 防火墙 的单个 Microsoft 管理控制台 (MMC)，高级安全Windows防火墙也成为网络隔离策略的重要部分。

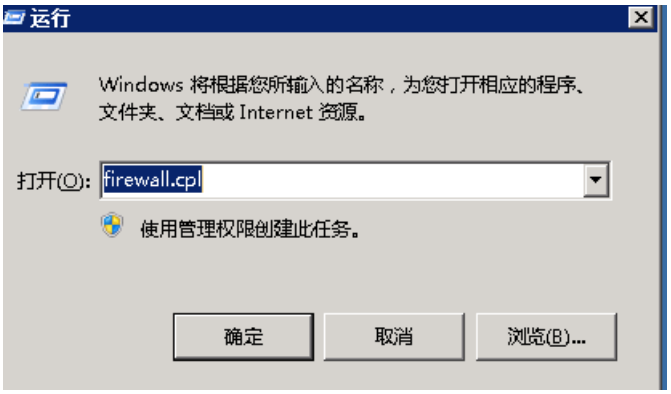
使用场景

作为一个运维人员，越来越多的用户反映服务器被恶意攻击，密码被暴力破解等等，其实大多数原因都是自己给那些“入侵者”留的“后门”导致的。入侵者通过扫描主机开放的端口，一旦发现可以利用的端口，就会进行下一步的入侵，例如Windows的远程端口（3389）和Linux的远程端口（22）。既然知道了问题的关键，那么我们也有相应的对策，我们可以通过修改默认的远程端口以及限制远程的访问来关闭所谓的“后门”。那么如何限制远程访问呢？接下来我们就以阿里云ECS实例Windows Server 2008 R2为例，来实现对远程桌面的限制。

操作步骤

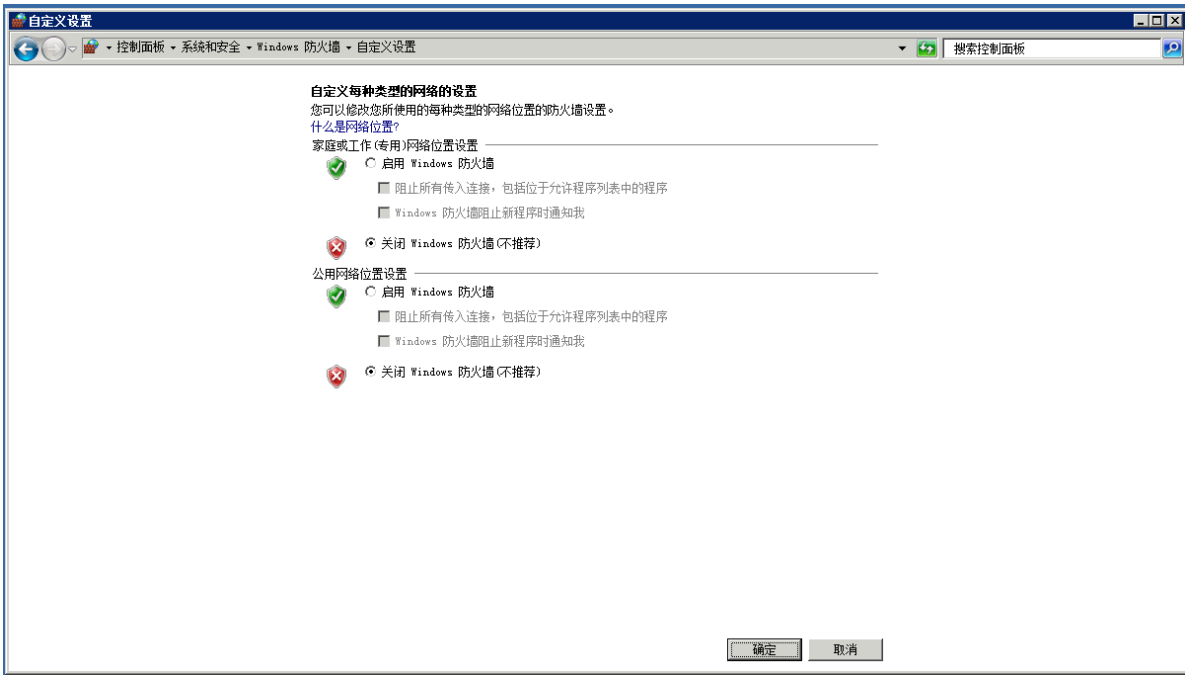
1. 查看防火墙状态

阿里云ECS实例Windows Server 2008 R2防火墙默认是关闭的，键盘输入Win+R打开运行输入*firewall.cpl* 回车来打开Windows防火墙控制台，见下图。



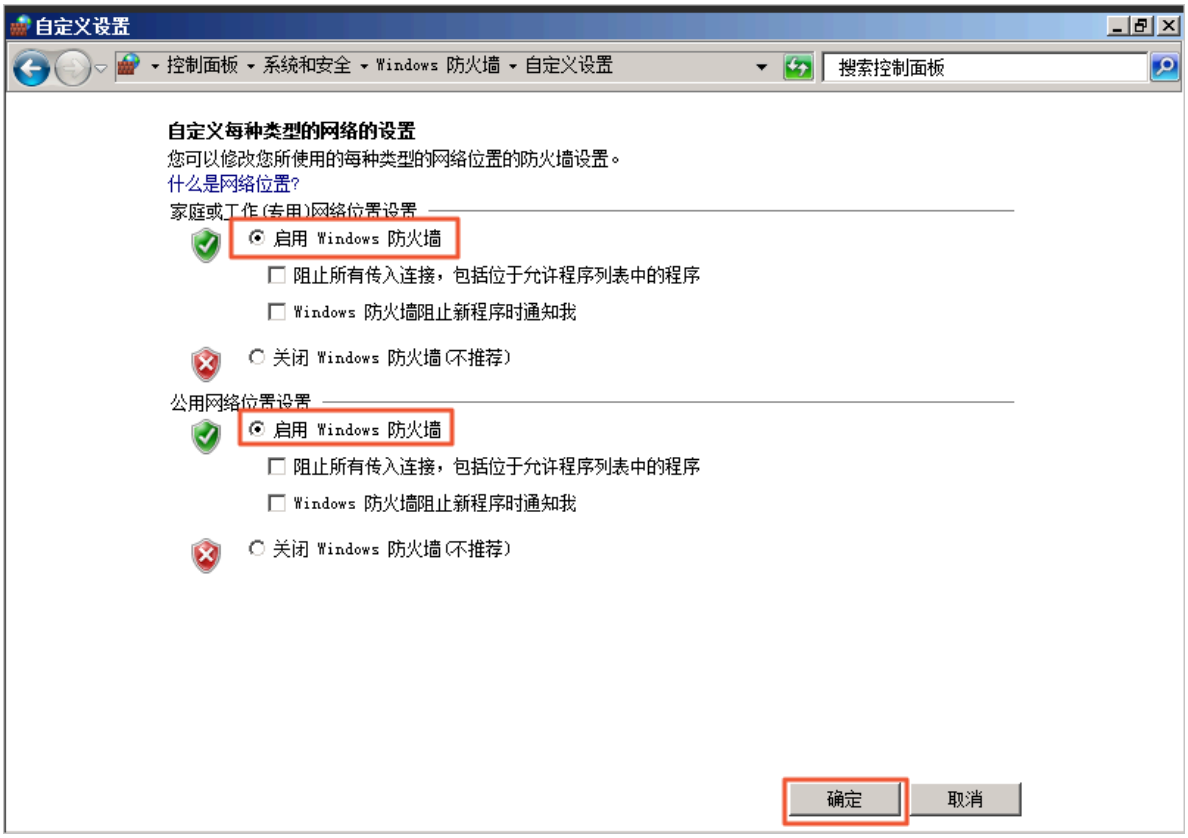
选择打开或关闭Windows防火墙。

如下图，我们看到防火墙是默认关闭的。


自定义设置
控制面板 ▸ 系统和安全 ▸ Windows 防火墙 ▸ 自定义设置
搜索控制面板
自定义每种类型的网络的设置
您可以修改您所使用的每种类型的网络位置的防火墙设置。
什么是网络位置?
家庭或工作(专用)网络位置设置
启用 Windows 防火墙
阻止所有传入连接，包括位于允许程序列表中的程序
Windows 防火墙阻止新程序时通知我
关闭 Windows 防火墙(不推荐)
公用网络位置设置
启用 Windows 防火墙
阻止所有传入连接，包括位于允许程序列表中的程序
Windows 防火墙阻止新程序时通知我
关闭 Windows 防火墙(不推荐)
确定
取消

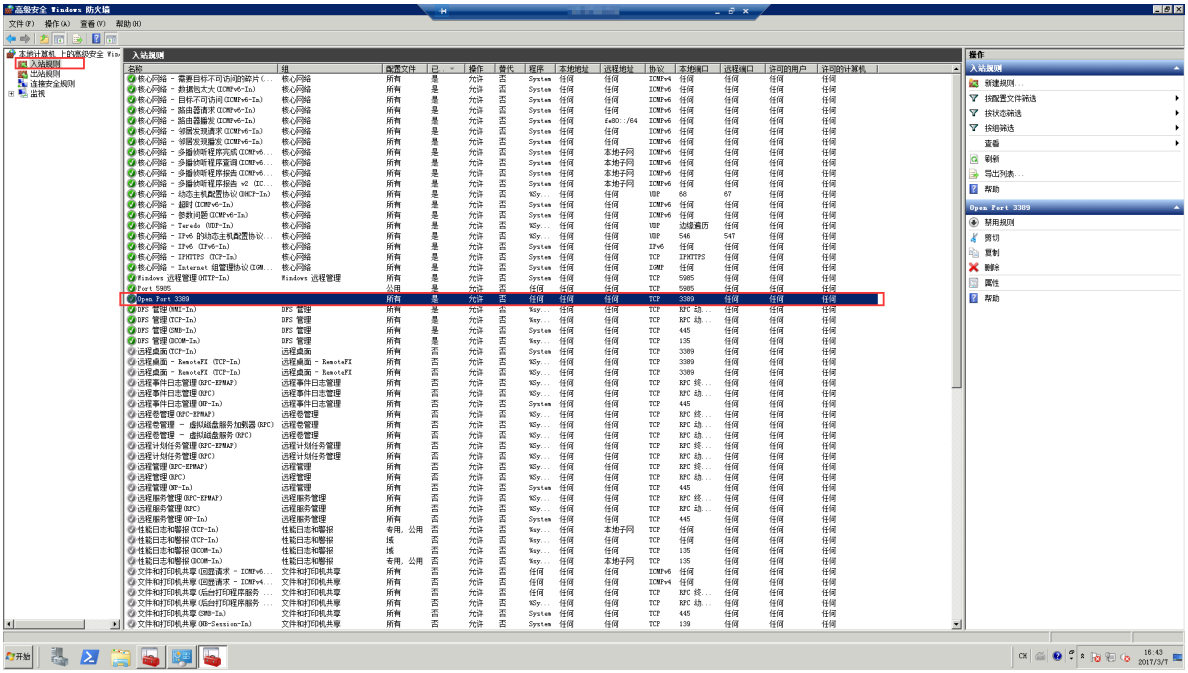
2. 启用防火墙

还是通过上面的步骤开启防火墙，见下图。



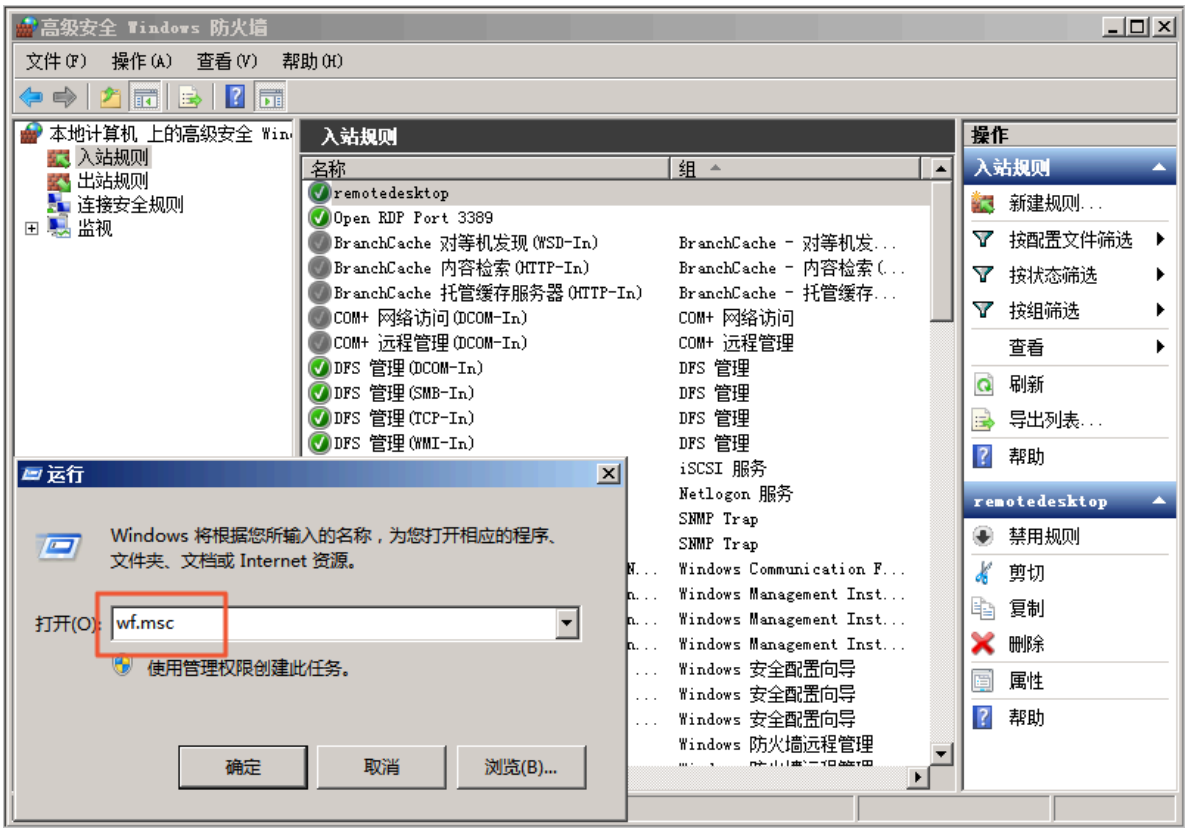
这里需要注意一点的是：启用之前请确认远程端口已经在里面，否则自己也将无法远程，不过高级安全Windows防护墙入站规则默认是放行3389端口的选择高级设置。

选择入站规则，我们看到open port 3389这条入站规则默认是放行3389端口的。

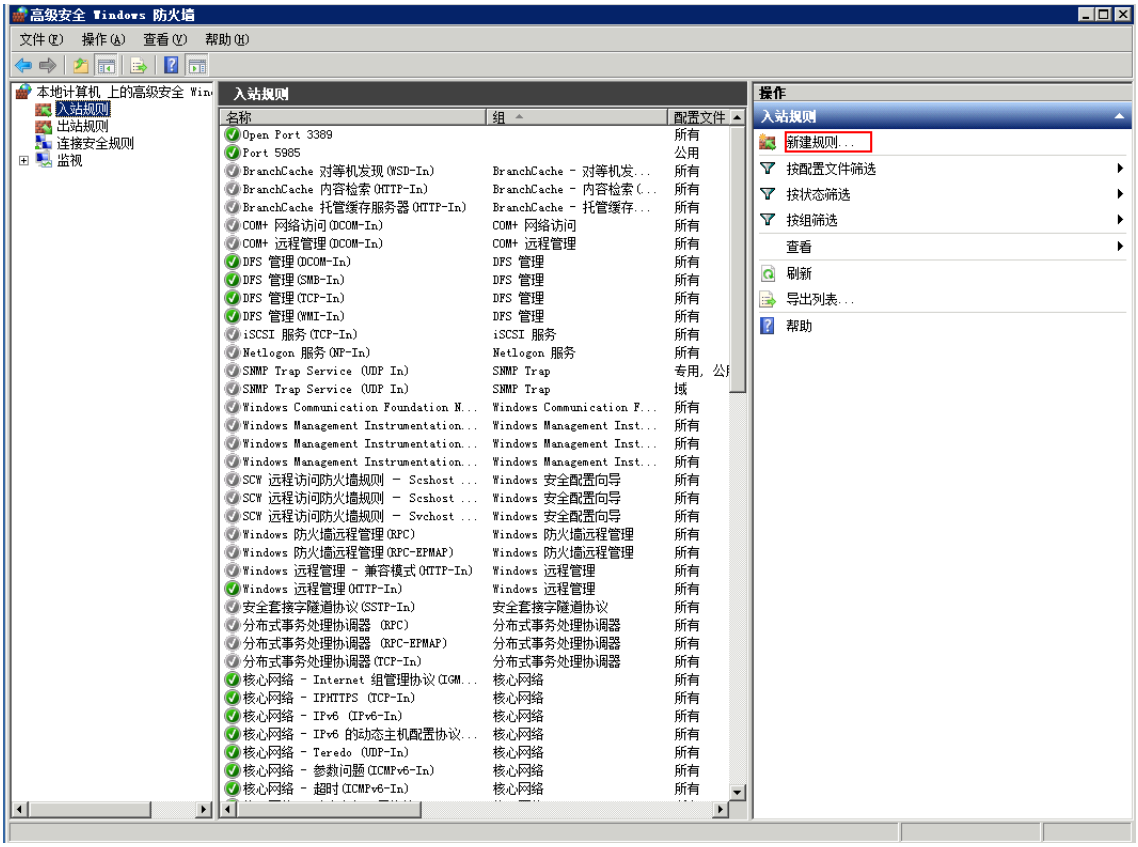


3. 配置高级安全Windows防火墙

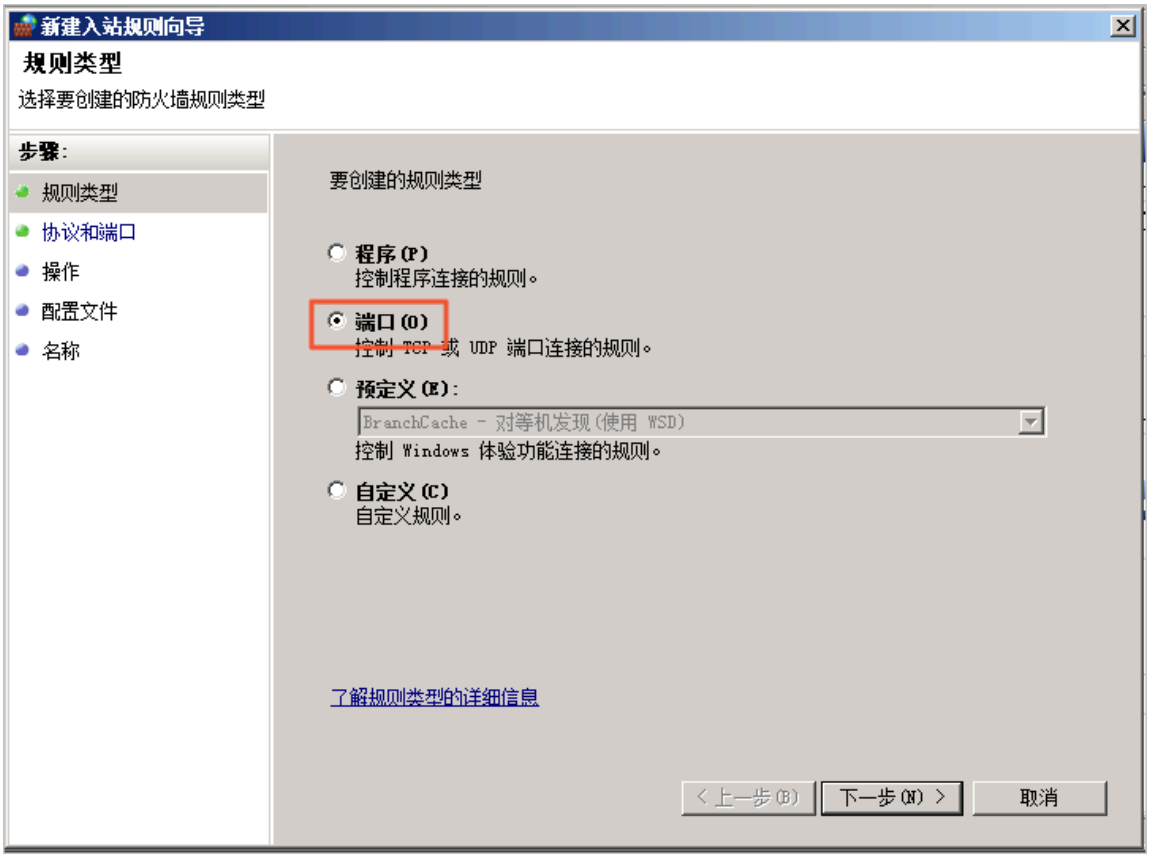
键盘输入Win+R打开运行输入*wf.msc* 回车来打开高级安全Windows防火墙，如下图。



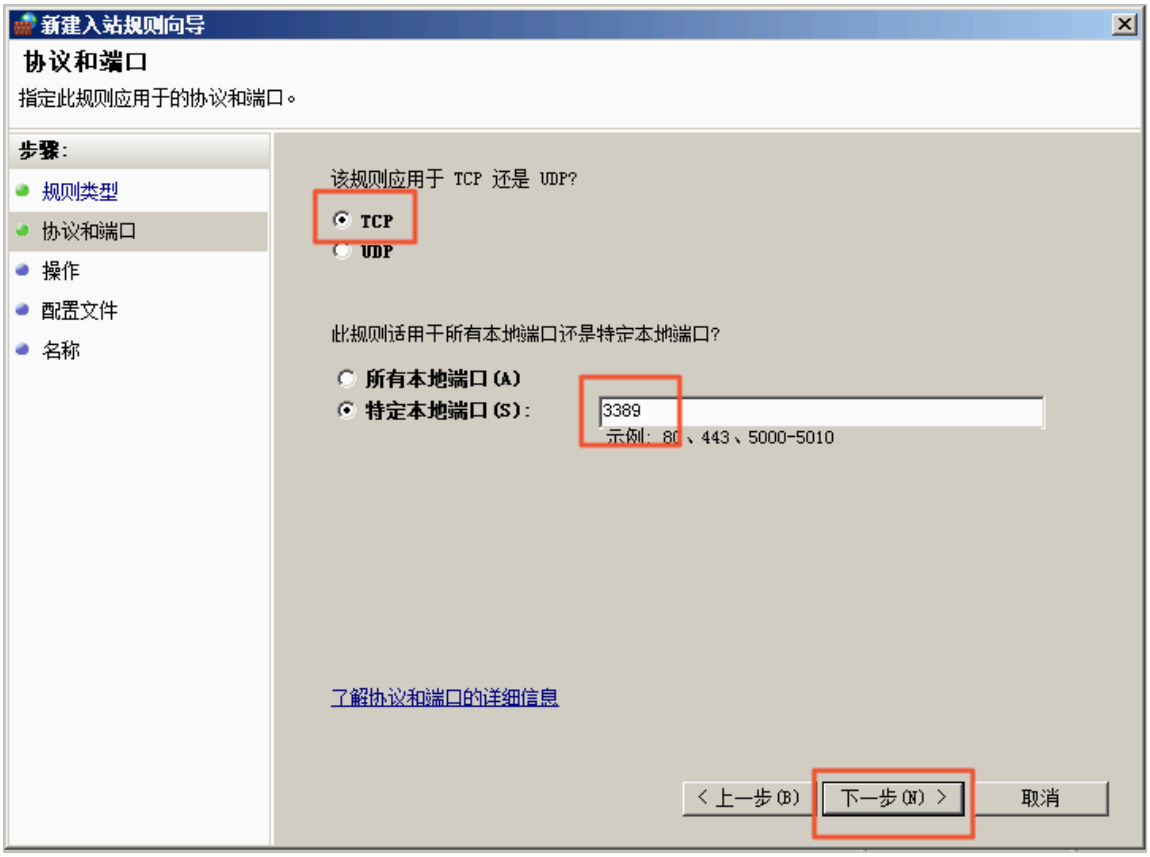
a. 通过手工新建入站规则



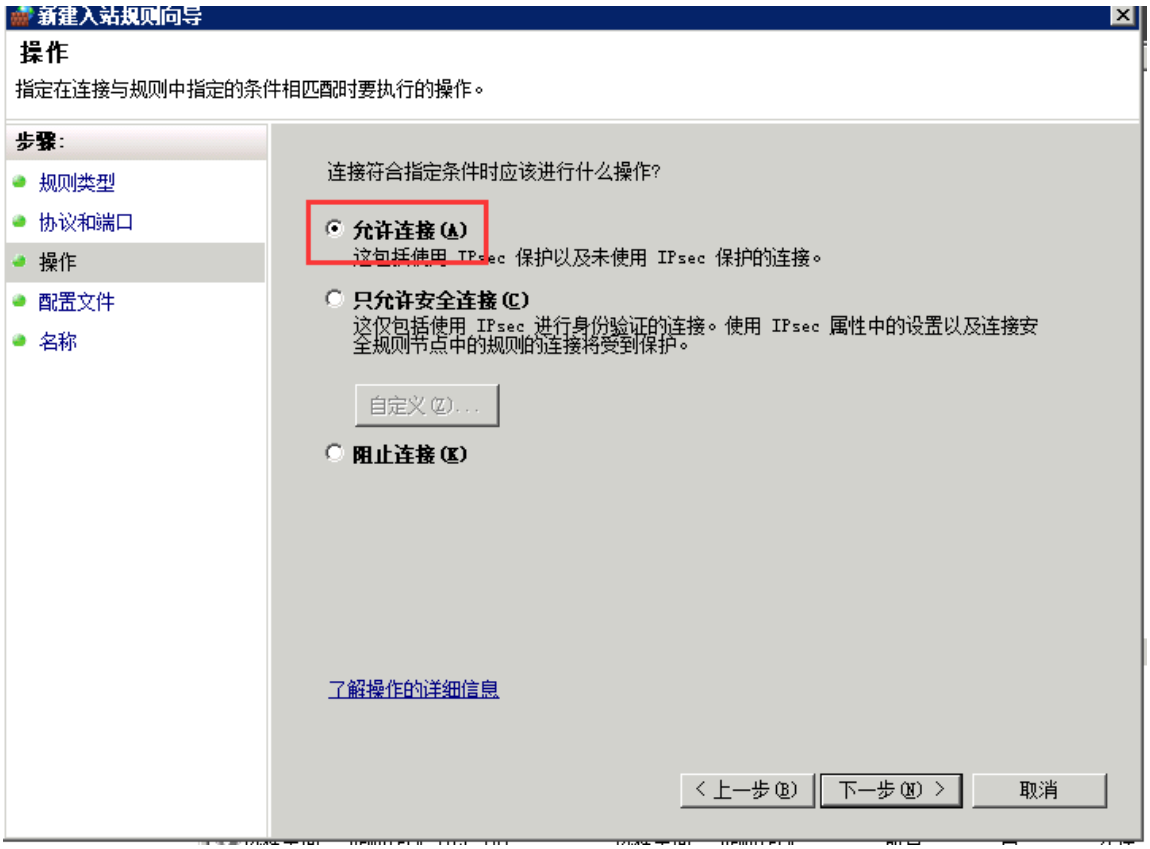
在弹出的新建入站规则向导窗口，选择 端口 然后鼠标左键单击下一步。



而后选择 TCP 并设置特定本地端口3389。

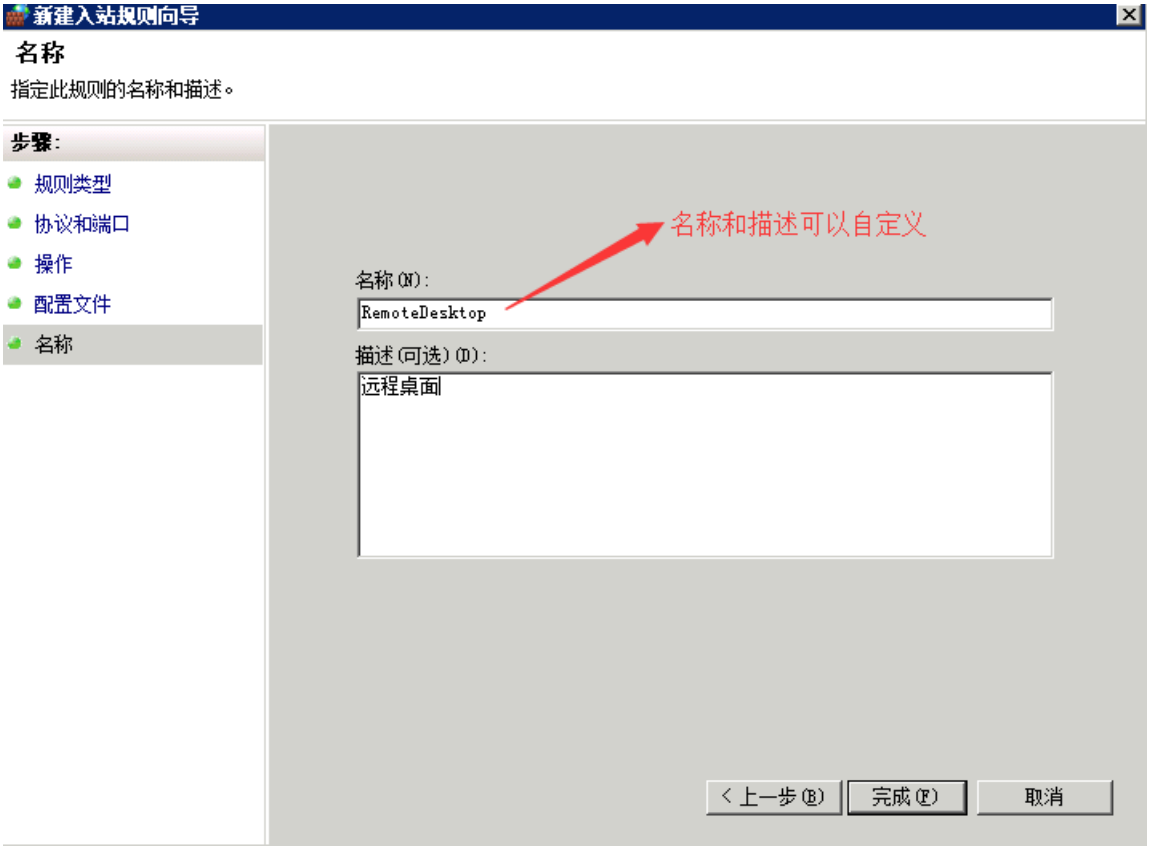

新建入站规则向导
协议和端口
指定此规则应用于的协议和端口。
步骤:
规则类型
协议和端口
操作
配置文件
名称
该规则应用于 TCP 还是 UDP?
TCP
UDP
此规则适用于所有本地端口还是特定本地端口?
所有本地端口(A)
特定本地端口(S):
3389
示例: 80、443、5000-5010
了解协议和端口的详细信息
< 上一步(B)
下一步(N) >
取消

下一步选择允许链接。

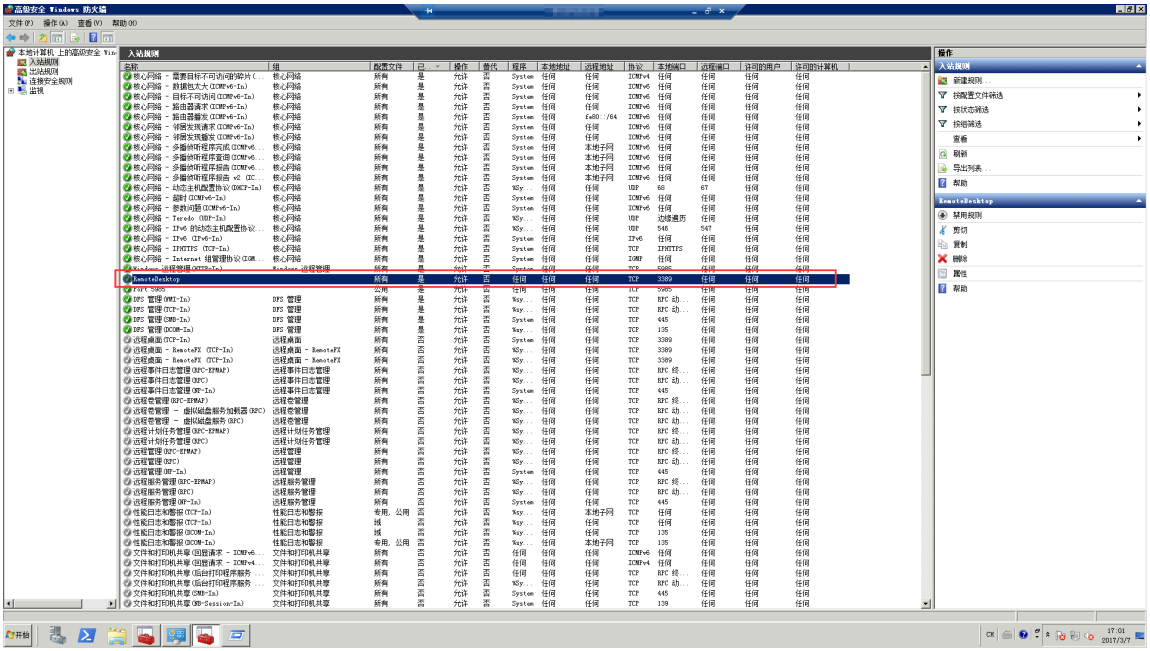

新建入站规则向导
操作
指定在连接与规则中指定的条件相匹配时要执行的操作。
步骤:
规则类型
协议和端口
操作
配置文件
名称
连接符合指定条件时应该进行什么操作?
允许连接(A)
这包括使用 IPsec 保护以及未使用 IPsec 保护的连接。
只允许安全连接(C)
这仅包括使用 IPsec 进行身份验证的连接。使用 IPsec 属性中的设置以及连接安全规则节点中的规则的连接将受到保护。
自定义(Z)...
阻止连接(K)
了解操作的详细信息
< 上一步(B)
下一步(N) >
取消

下一步 默认配置即可。

下一步 填写规则名称，例如 RemoteDesktop ,最后鼠标左键单击完成。



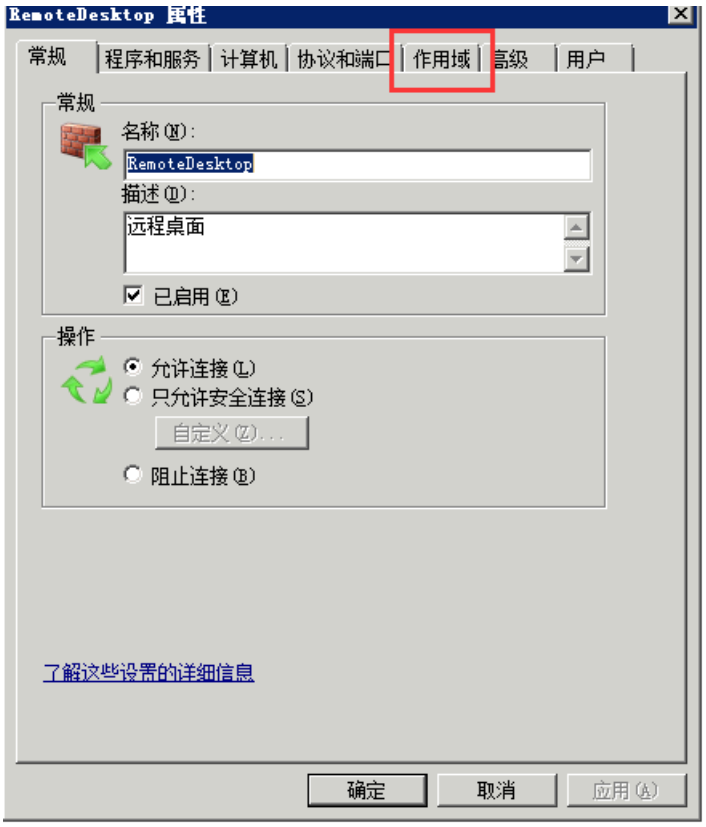
看到我们刚刚添加的规则。



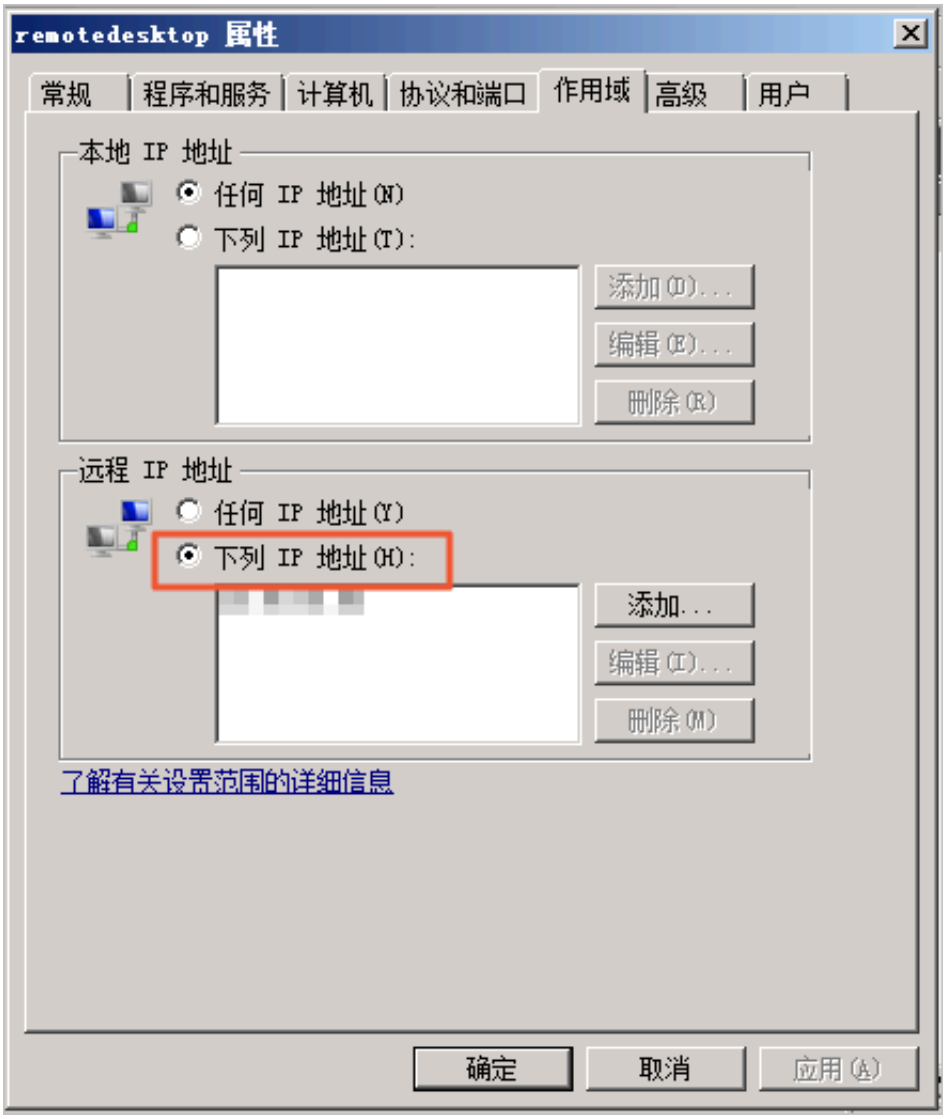
以上步骤就是把Windows远程端口加入到高级安全Windows防火墙了，但是依然没有实现我们的限制访问，接下来我们来实现访问限制。

b. 配置作用域

右键选中我们刚刚创建的入站规则，然后选择属性>作用域>远程IP地址>添加（将需要远程此服务器的IP地址填写进去，注意：一旦启用作用域，除了作用域里面的IP地址，别的地址将无法远程链接此服务器）。

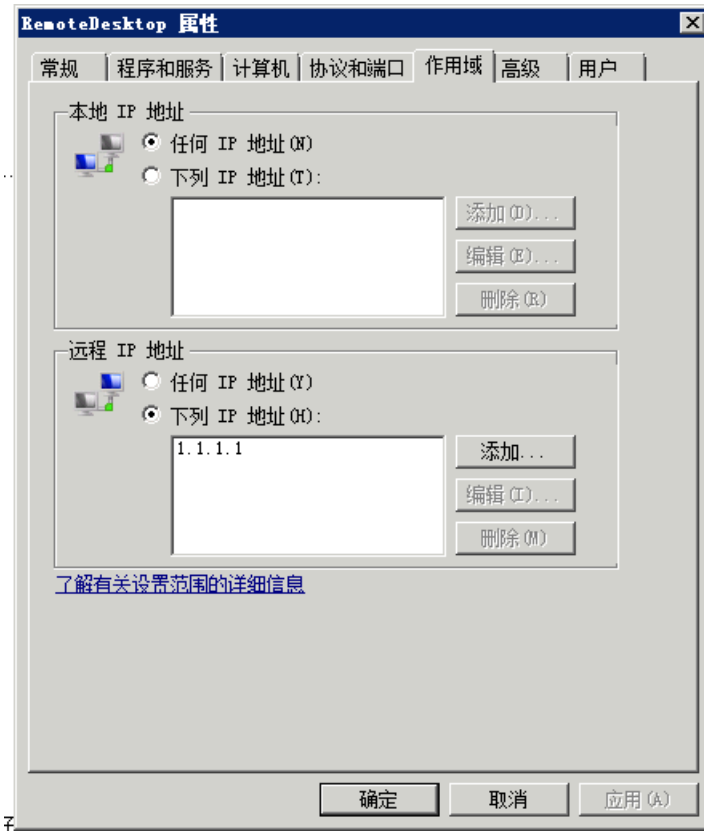


添加远程IP地址。

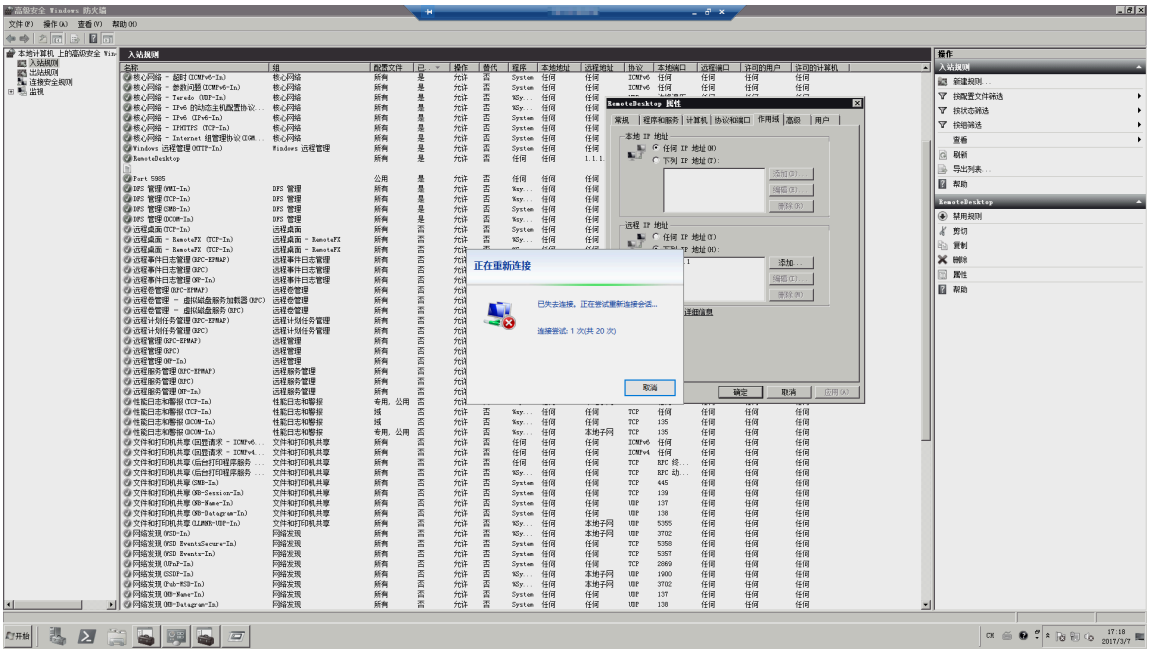


c. 验证作用域

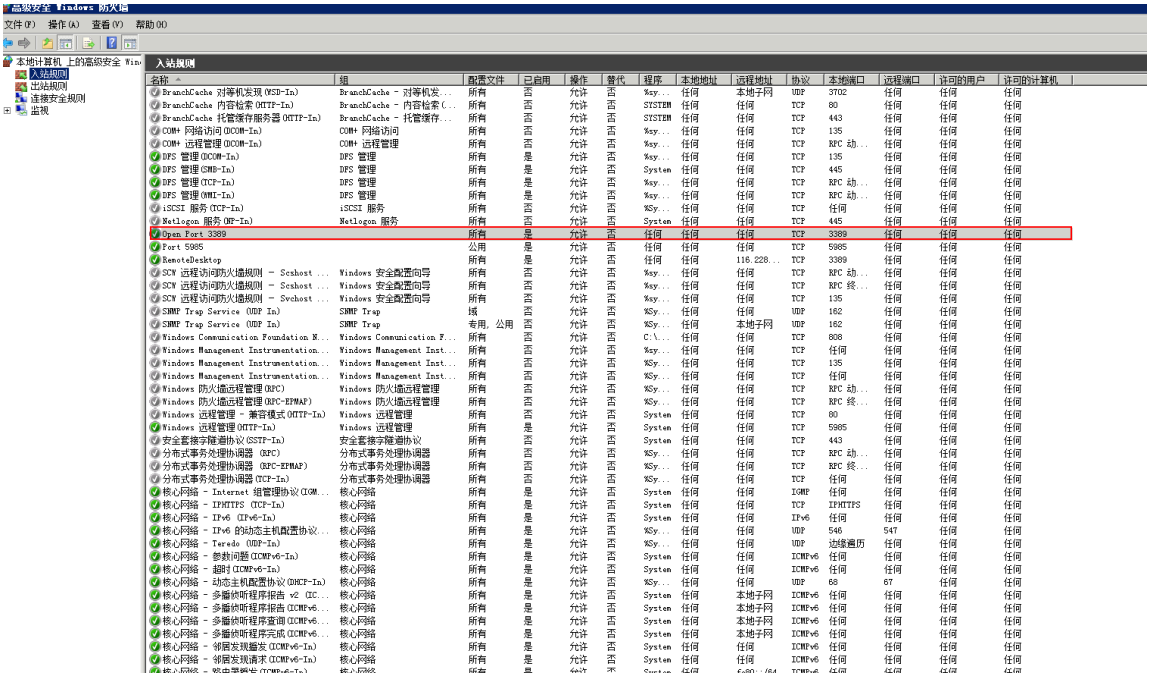
我们在作用域——远程IP地址里面随便写个地址，看看远程连接会发生什么。



远程连接断掉。



如果远程连接没有断开，让我们把下图中open port 3389这条入站规则禁用掉就可以了。



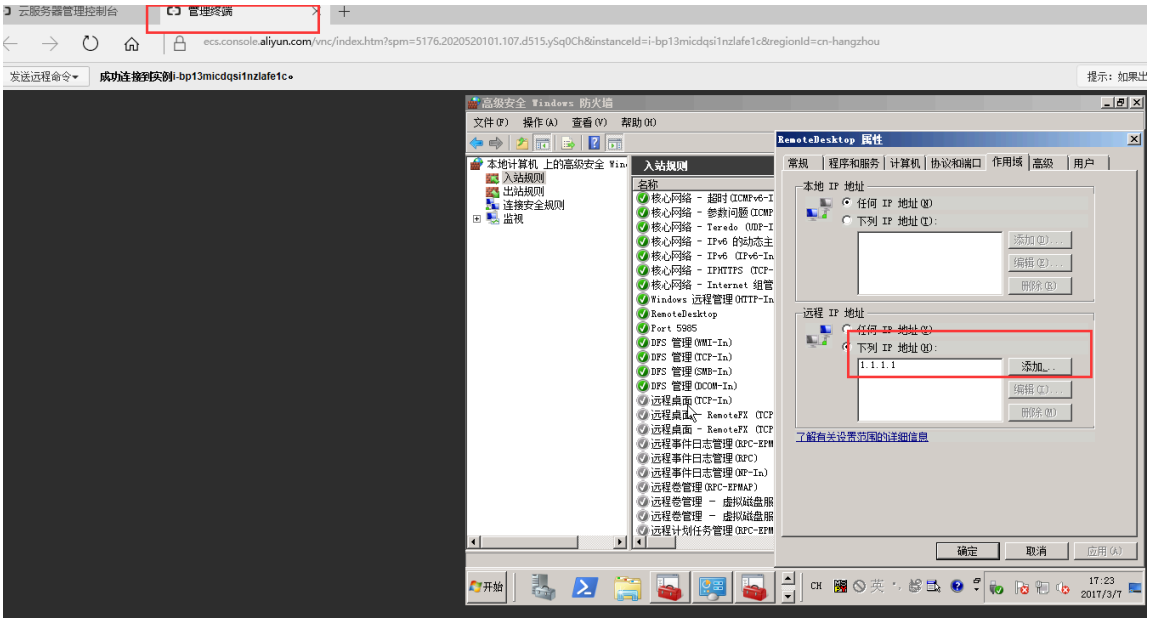
远程连接自己断开了，这就说明我们的作用域生效了，那现在自己都无法远程了，怎么办呢？别急，我们还有阿里云控制台，登录阿里云控制台，然后将上面的作用域地址换成自己

的地址（这里要写办公环境的公网地址，除非您的办公环境和阿里云线上的环境打通，）就可以正常远程了。

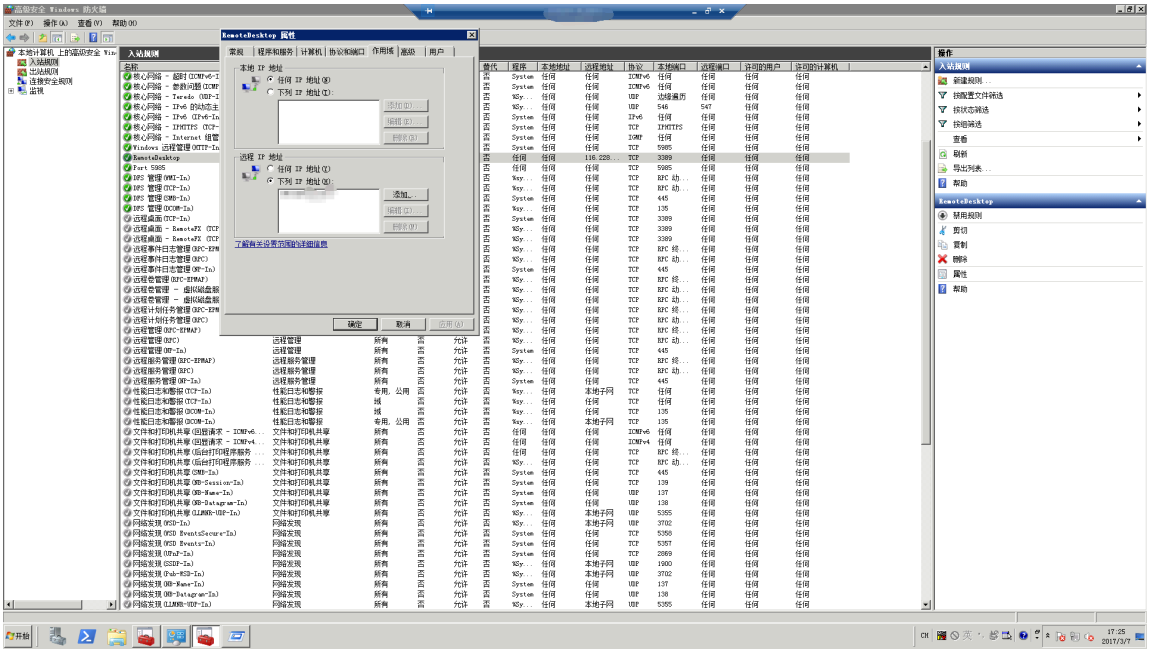
进入阿里云的控制台界面，找到相应实例打开远程连接。

登录系统。

与之前同样的方式，修改RemoteDesktop的作用域的远程IP地址，将之前测试设置的1.1.1.1换回自己的IP地址。



换回自己的IP地址后可以正常远程了，如果不知道自己的公网IP，可以点击此处查看。



以上就是使用高级安全Windows防火墙来实现对服务器远程访问的限制，其他的服务和端口都可以按照上面的方法来实现，例如，关闭不常用的135 137 138 445 端口，限制FTP和相关服务的访问等等，这样才能做到最大限度地保障服务器安全的运行。

命令行的方式

1. 导出防火墙配置到文件。

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

2. 导入防火墙配置文件到系统中。

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

3. 防火墙恢复默认设置。

```
Netsh advfirewall reset
```

4. 关闭防火墙。

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

5. 开启防火墙。

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

6. 在所有配置文件中设置默认阻挡入站并允许出站通信。

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

7. 删除名为 ftp 的规则。

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

8. 删除本地端口 80 的所有入则。

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

9. 添加远程桌面入站规则允许端口3389。

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

相关链接

用户可通过云中沙箱平台体验上述文档中的操作，点击此处

Windows防火墙限制端口/IP/应用访问的方法以及例外的配置

Windows 系统远程桌面端口查看和修改方法

Linux 修改默认远程端口方法

更多开源软件尽在云市场

1.10 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。安全组由同一个地域内具有相同安全保护需求并相互信任的实例组成。为了满足同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，支持安全组内实现网络隔离。

安全组内的网络隔离规则

- 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，需要在每个网卡上设置安全组隔离规则。
- 不会改变默认的网络连通策略。

  安全组内网络隔离是一种自定义的网络连通策略，对于默认安全组和新建的安全组无效。安全组默认的网络连通策略是：同一安全组内的实例之间私网互通，不同安全组的实例之间默认私网不通。
- 安全组内网络隔离的优先级最低。

  设置了组内网络隔离的安全组，仅在安全组内没有任何自定义规则的情况下保证安全组内实例之间网络隔离。以下情况设置了组内网络隔离但实例仍然互通：

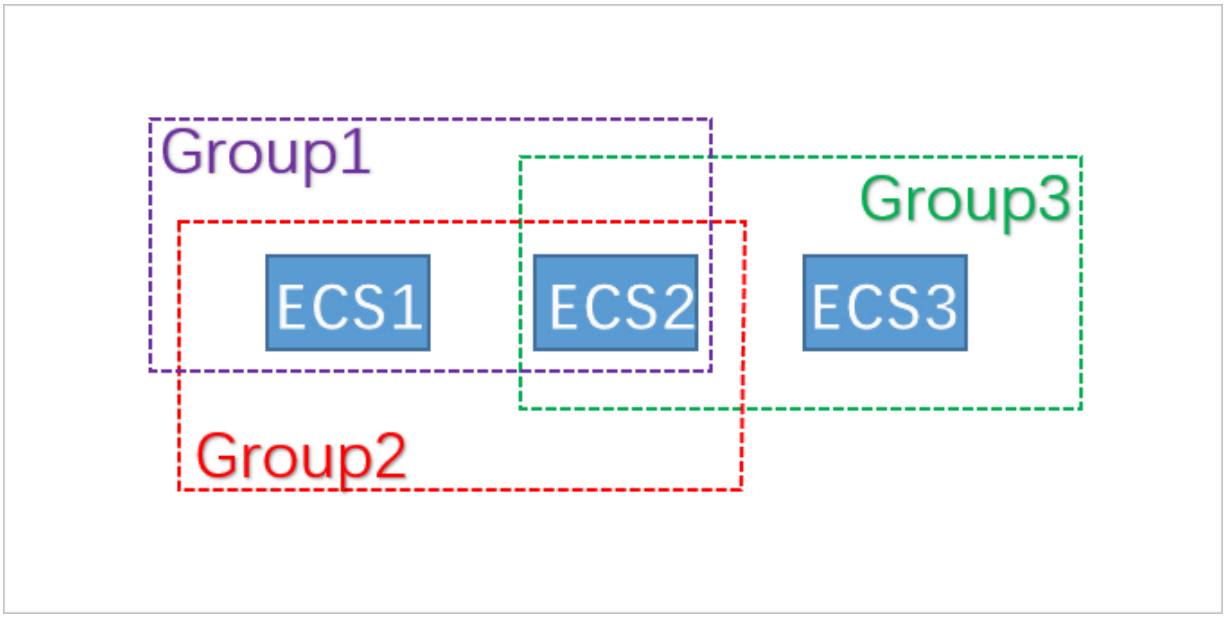
  - 安全组内既设置了组内隔离，又设置了让组内实例之间可以互相访问的ACL。
  - 安全组内既设置了组内隔离，又设置了组内互通。
- 网络隔离只对当前安全组内的实例有效。

修改策略

您可以使用ModifySecurityGroupPolicy接口来修改安全组内的网络连通策略。

案例分析

实例和实例所属的安全组的关系如下：



本示例中，Group1、Group2、Group3分别为3个不同的安全组，ECS1、ECS2、ECS3分别为3个不同的ECS实例。ECS1和ECS2同属安全组Group1和Group2，ECS2和ECS3同属安全组Group3。

3个安全组内的网络连通策略设置如下：

安全组	内网连通策略	包含的实例
Group1	隔离	ECS1、ECS2
Group2	互通	ECS1、ECS2
Group3	互通	ECS2、ECS3

各实例间的网络连通情况如下：

实例	网络互通／隔离	原因
ECS1和ECS2	互通	ECS1、ECS2同时属于Group1和Group2。Group1的策略是隔离，Group2的策略是互通，由于网络隔离的优先级最低，所以ECS1和ECS2互通。
ECS2和ECS3	互通	ECS2和ECS3同时属于Group3。Group3的策略是互通，所以ECS2和ECS3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。如果两个安全组之间需要互相访问，可以通过安全组规则授权。

1.11 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

在最初涉及安全组规则时，

- 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。

1.12 查看潜在高危安全组概览

安全组规则设置不当会造成严重的安全隐患。阿里云会定期检查您的安全组，如果安全组规则对特定端口的访问不做限制，就会产生预警。您可以使用 潜在高危安全组概览 发现不合理的安全组规则，通过修改安全组规则保证ECS实例的网络安全。

背景信息

专有网络和经典的安全保护不同：

- 专有网络的安全保护：
  - 不同专有网络之间内部网络完全隔离。
  - 专有网络内可以划分多个子网，子网之间互相通信需要经过子网ACL。
  - 专有网络内的ECS实例使用安全组进行三层网络访问控制。
- 经典网络的安全保护：仅依靠安全组设置网络访问控制策略。

所以，安全组对于专有网络和经典网络的安全保护非常重要。如果安全组规则设置了不受限制的访问规则，将会降低访问的限制级别，扩大攻击者执行恶意行为的攻击面。

功能限制

- 显示危险等级最高的50个安全组的风险数据。
- 风险数据为T+1的结果，非实时分析数据，即显示的是截至昨天的安全组状态。

操作步骤

1. 登录 云服务器 ECS 管理控制台。
2. 在 概览 页面中，单击 资源概览。

3. 单击 潜在高危安全组概览 查看检查结果。检查结果按照危险等级分为：

   - 严重：建议您及时修复安全组规则。
   - 警告：建议您尽量避免设置此安全组规则。

4. （可选）您可以执行以下操作：

   - 如果您需要修改安全组规则，单击 修改规则。修改安全组规则请参见 安全组实践的基本建议 和 安全组应用案例。

   - 如果您的实例需要 加入安全组 或 移出安全组，单击 管理实例。

相关操作

- 如果您想查看入方向和出方向的具体规则，您可以 查询安全组规则。
- 如果您需要对一个线上业务执行新的安全组规则，您可以先 克隆原来的安全组 作为备份，再 修改安全组规则。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。
- 如果您不再需要某个安全组规则，您可以 删除安全组规则。

1.13 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。本文介绍如何配置云防火墙并查看业务关系。

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、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。

使用场景

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。

前提条件

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，请参见云防火墙授权说明。
- 确保云防火墙为企业版或旗舰版，请参见云防火墙计费方式。

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，需要次日才能看到效果。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。

参考以下步骤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。

> 说明：
> 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，比如华东1。

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：
   a. 登录云服务器ECS控制台。
   b. 选择策略组所在地域，比如华东1。
   c. 在左侧导航栏，单击网络和安全 > 安全组。
   d. 搜索维度选择安全组名称，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，比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，比如所有部署了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。

参考以下步骤查看当前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 > 应用分组。
3. 创建业务区。
   a. 单击业务区。
   b. 单击新建业务区。
   c. 填写名称，比如DB业务、Web业务。
   d. 填写备注。
   e. 选择程度，比如非常重要。

4. 创建应用组。
   a. 单击应用组。
   b. 单击新建应用组。
   c. 填写名称，比如DB应用组、Web应用组。
   d. 填写备注。
   e. 选择程度，比如非常重要。
   f. 选择业务区类型，比如选择已有业务区。
   g. 选择业务区，比如DB业务、Web业务。
5. 分配应用。
   a. 选择VPC，比如华东 1 - vpc-xxx。
   b. 根据业务需要分配应用，比如将部署了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，比如华东 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

- 合理配置访问控制策略
- 通过限制端口控制访问流量
- 对互联网业务进行应用层的深度防御
- 防御数据库入侵

1.14 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，可以参考本文描述撤销安全组授权。

前提条件

本文使用阿里云CLI调用ECS API，请确保您已经安装了阿里云CLI。详情请参见阿里云CLI安装指南。

背景信息

本文通过RevokeSecurityGroup撤销已授权的安全组规则，涉及的操作步骤中，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：已授权的账号内网互通的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DescribeSecurityGroupReferences查询。
- ECS实例所在的地域名称：取值请参见地域和可用区。本文示例采用了cn-beijing，即华北2（北京）地域。

假设两个账号下的信息如下表所示。

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-bp1azkttqp
ldxgtedXXX --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-bp15ed6xe1
yxeycg7XXX --RegionId cn-beijing --IpProtocol all --PortRange -1/-1
 --SourceGroupId sg-bp1azkttqpldxgtedXXX --SourceGroupOwnerAccount a
@aliyun.com --NicType intranet
```

1.15 通过API允许不同账号下的ECS实例内网通信

若您需要实现同一地域下不同账号的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

本文调用API的工具为阿里云CLI，请确保您已安装并配置了阿里云CLI。具体操作，请参见安装CLI和配置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：授权同一账号两台ECS实例间的内网通信。

- 账号间内网通信：授权同一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：
  > 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。详情请参见高速通道、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。详情请参见高速通道、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-bp1azkttqpldxgtedXXX
实例B	10.0.0.2	sg2	sg-bp15ed6xe1yxeycg7XXX

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-bp1azkttqp
ldxgtedXXX --RegionId cn-qingdao --IpProtocol all  --PortRange=-1/-1
. --SourceCidrIp 10.0.0.2 --NicType intranet
```

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-bp15ed6xe1
yxeycg7XXX --RegionId cn-qingdao --IpProtocol all  --PortRange=-1/-1
. --SourceCidrIp 10.0.0.1 --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，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。

账号间内网通信

1. 查询两个账号的账号名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。假设两个账号的信息如下表所示。

账号	账号ID	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

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-bp1azkttqp
ldxgtedXXX --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-bp15ed6xe1
yxeycg7XXX --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。

2 灾备方案

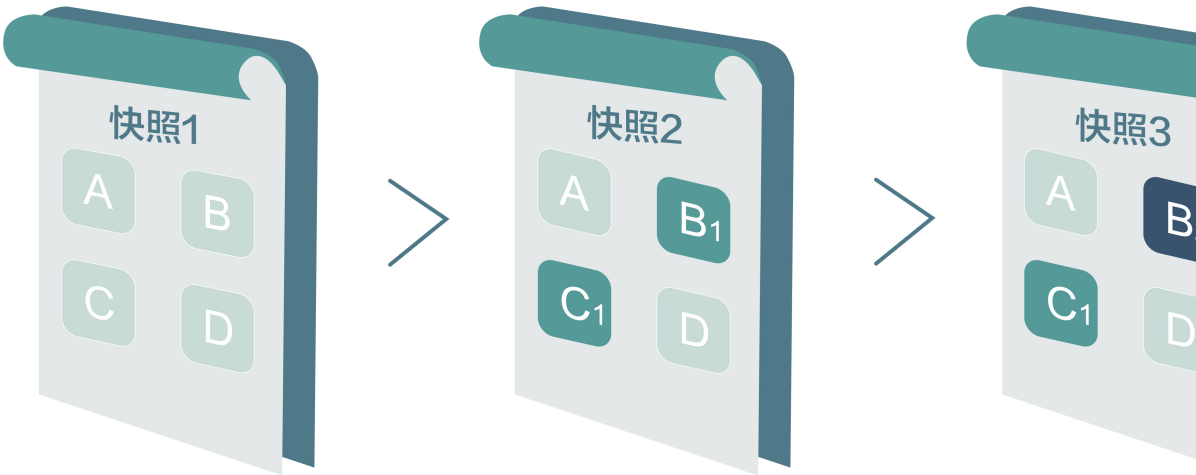
保障企业业务稳定、IT系统功能正常、数据安全十分重要，可以同时保障数据备份与系统、应用容灾的灾备解决方案应势而生，且发展迅速。ECS可使用快照、镜像进行备份。

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，建议您在创建完毕新快照后，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。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考快照与镜像。

  > 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

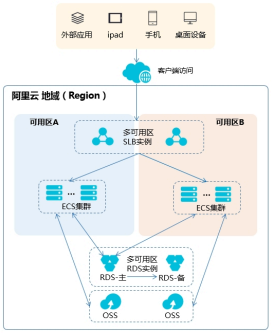
- 备份恢复

  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，比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。

3 数据恢复

3.1 误删文件后如何恢复数据

本文档主要以CentOS7操作系统为例，介绍如何使用开源工具Extundelete快速恢复被误删除掉的数据。

简介

在日常使用中有时难免会出现数据被误删除的情况，在这个时候该如何快速、有效地恢复数据呢？在阿里云上恢复数据有多种方式，例如：

- 通过阿里云控制台回滚备份好的快照，自定义镜像恢复等方式。
- 购买多台ECS，实现业务的负载均衡，高可用。
- 利用对象存储 OSS（Object Storage Service），存储静态网页和海量图片、视频等重要数据。

在Linux下，基于开源的数据恢复工具有很多，常见的有debugfs、R-Linux、ext3grep、extundelete等，比较常用的有ext3grep和extundelete，这两个工具的恢复原理基本一样，只是extundelete功能更加强大。

Extundelete是基于linux的开源数据恢复软件。在使用阿里云的云服务器时，如果您不小心误删除数据，并且Linux系统也没有与Windows系统下回收站类似的功能，您可以方便快速安装此工具。

Extundelete能够利用inode信息结合日志去查询该inode所在的block位置，以次来查找和恢复所需的数据，该工具最给力的一点就是支持ext3/ext4双格式分区恢复，基于整个磁盘的恢复功能较为强大。

在数据被误删除后，第一时间要做的是卸载被删除数据所在的磁盘或磁盘分区。因为将文件删除后，仅仅是将文件的inode结点中的扇区指针清零，实际文件还存储在磁盘上，如果磁盘以读写模式挂载，这些已删除的文件的数据块就可能被操作系统重新分配出去，在这些数据块被新的数据覆盖后，这些数据就真的丢失了，恢复工具也回力无天。所以，以只读模式挂载磁盘可以尽量降低数据块中数据被覆盖的风险，以提高恢复数据成功的几率。

说明:

在实际线上恢复过程中，切勿将extundelete安装到您误删的文件所在硬盘，这样会有一定几率将需要恢复的数据彻底覆盖，切记操作前做好快照备份。

适用对象

- 磁盘中文件误删除的用户，且未对磁盘进行过写入等操作
- 网站访问量小、少量 ECS 实例的用户

使用方法

需安装的软件及版本：e2fsprogs-devel e2fsprogs gcc-c++ make（编译器等）Extundelete-0.2.4。

> **说明:**
> extundelete需要libext2fs版本1.39或更高版本来运行，但是对于ext4支持，请确保您有e2fsprogs版本1.41或更新版本（可以通过运行命令“dumpe2fs”并记录其输出的版本）。

以上版本是写文档时的软件版本。您下载的版本可能与此不同。

- 部署extundelete工具

```
wget  http://zy-res.oss-cn-hangzhou.aliyuncs.com/server/extundelete-0.2.4.tar.bz2
yum -y install  bzip2  e2fsprogs-devel  e2fsprogs  gcc-c++  make
 #安装相关依赖和库
tar -xvjf extundelete-0.2.4.tar.bz2
cd extundelete-0.2.4                                    #进入程序目录
./configure                                             #如下图表示安装成功
```

```
extundelete-0.2.4/src/Makefile.am
extundelete-0.2.4/configure.ac
extundelete-0.2.4/depcomp
extundelete-0.2.4/Makefile.in
extundelete-0.2.4/Makefile.am
[root@iZy930wmhyutc2Z ~]# cd extundelete-0.2.4
[root@iZy930wmhyutc2Z extundelete-0.2.4]# ./configure
Configuring extundelete 0.2.4
Writing generated files to disk
[root@iZy930wmhyutc2Z extundelete-0.2.4]#
```

```
make && make install
```

这个时候会出现src目录，下面有个extundelete可执行文件以及相应路径，如下图，其实默认文件安装在*usr/local/bin*下面，下面演示就在*usr/local/bin*目录下。

- 使用extundelete，模拟数据误删除然后恢复的过程

1. 检查ECS现有的磁盘和可用分区，并对/dev/vdb进行分区，格式化，此处不在介绍磁盘分区格式化方式，如果不会的话可以点击此文档查看操作方式格式化和挂载数据盘。

```
fdisk -l
```

```
Disk identifier: 0x0000efd2

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

Disk /dev/vdb: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

2. 将分区后的磁盘挂载到/zhuyun目录下，然后在/zhuyun下面新建测试文件hello,写入test。

```
mkdir /zhuyun                                   #新建zhuyun目录
mount /dev/vdb1 /zhuyun                         #将磁盘挂载到zhuyun目录下
echo test > hello                               #写入测试文件
```

3. 记录文件MD5值，md5sum命令用于生成和校验删除前和恢复后俩个文件的md5值。

```
md5sum hello
```

```
[root@iZbp13micdqsi2364umm8aZ zhuyun]# md5sum hello
d8e8fca2dc0f896fd7cb4cb0031ba249  hello
```

4. 模拟删除hello文件。

```
rm -rf hello
cd ~
```

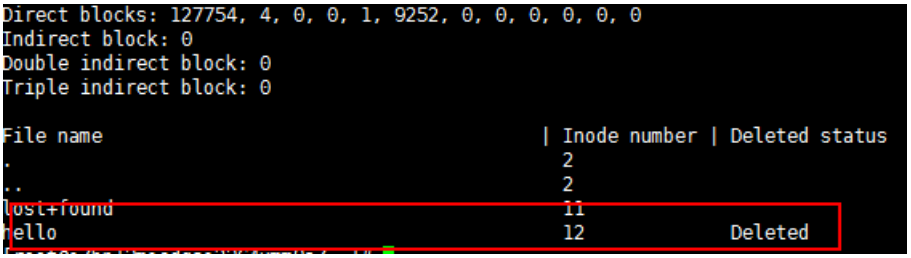
```
fuser -k /zhuyun                                    #结束使用某分区的进程树（确认没有资源占用的话，可以跳过此步）
```

5. 卸载数据盘。

```
umount /dev/vdb1                                    #任何的文件恢复工具，在使用前，均要将要恢复的分区卸载或挂载为只读，防止数据被覆盖使用
```

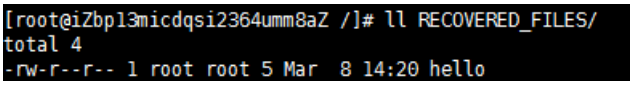
6. 使用Extundelete工具恢复文件。

```
extundelete --inode 2 /dev/vdb1          #为查找某i节点中的内容，使用2则说明为整个分区搜索，如果需要进入目录搜索，只须要指定目录I节点即可。这是可以看到删除的文件名和inode
```




```
/usr/local/bin/extundelete  --restore-inode 12  /dev/vdb1    #恢复删除的文件
```

这个时候会在执行命令的同级目录下出现RECOVERED_FILES目录，查看是否恢复。



通过md5值查看，前后俩个文件，一样说明恢复成功。

```
--restore-inode 12                  # --restore-inode 按指定的I节点恢复
--extundelete --restore-all         # --restore-all   全部恢复
```

相关链接

用户可通过云中沙箱平台体验上述文档中的操作，点击此处。

3.2 Windows 实例磁盘空间满的问题处理及最佳实践

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。

本文中的方法适用于 Windows Server 2003 以上系统，这里以 Windows Server 2008 R2 为例。

说明:

Linux 实例磁盘空间不足时对应的处理方法参考 ECS Linux 磁盘空间满排查处理。

解决方法

解决 Windows 磁盘空间满的问题，有以下两种处理方式：

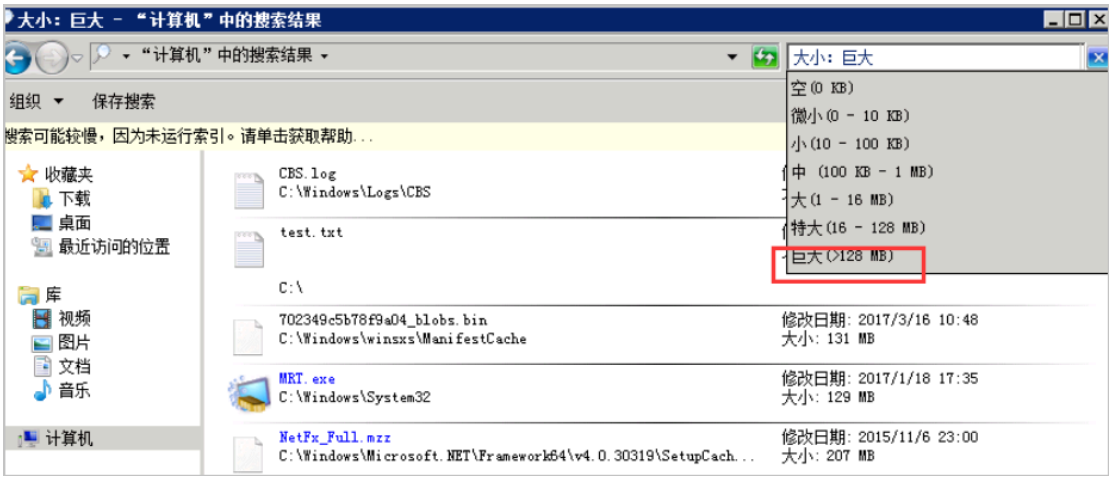
- 释放磁盘空间
- 扩容磁盘

- 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，首先找出占用磁盘空间过多的文件，然后删除不需要的文件，具体步骤如下：

  - 找出占用磁盘空间过多的文件

    1. 远程连接并登录到 Windows 实例。
    2. 双击计算机，单击要清理的磁盘，按下键盘的 Ctrl+F 键，定位到搜索框。
    3. 在搜索框中，选择大小，然后根据系统定义大小筛选指定磁盘的大文件。



 **说明：**

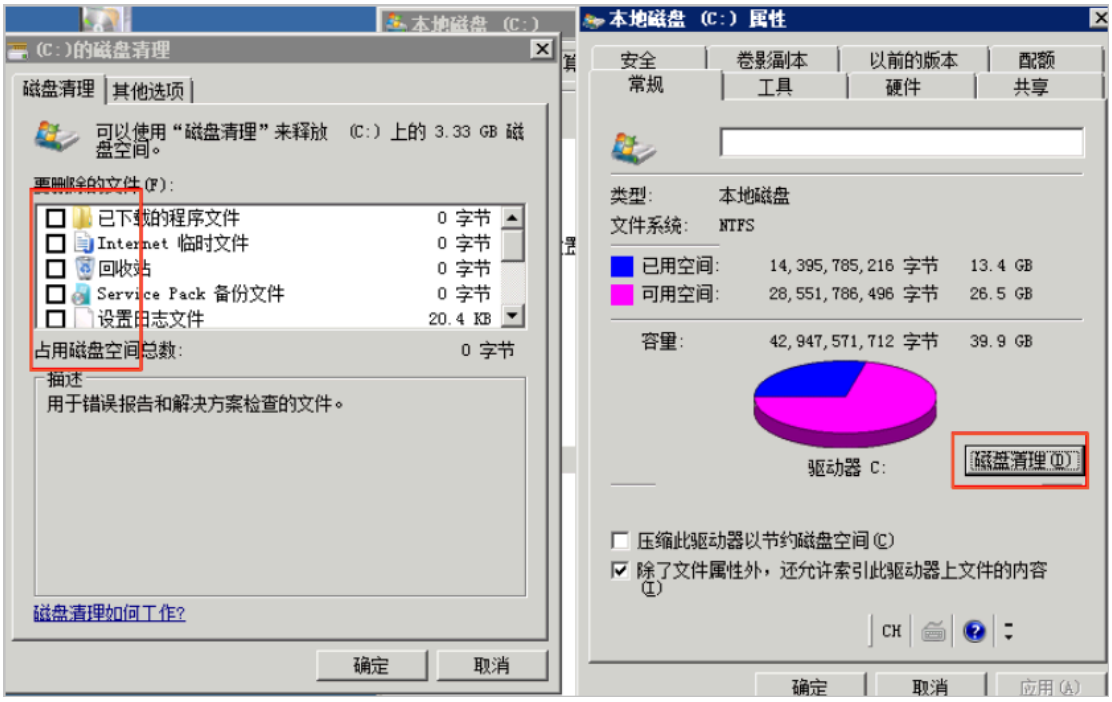
您也可以自定义文件大小范围进行检索，如输入`###>500M`，会检索该磁盘大于 500 M 的文件。如输入`###> 100M < 500M`，会检索大于 100 M 但小于 500 M 的文件。

- 删除不需要的文件

  找出占用了磁盘空间过多的文件后，如果文件不再需要，可以及时清理。

  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：

  1. 打开服务器管理器，单击功能，然后单击添加功能。
  2. 在添加功能向导窗口，勾选墨迹手写服务和桌面体验，然后单击下一步。
  3. 在弹出的窗口中，单击安装。
  4. 安装页面上，系统将提示您手动重新启动服务器，单击是重新启动服务器。重新启动服务器之后，确认已安装了桌面体验。
  5. 安装完成后，选择开始 > 所有程序 > 附件 > 系统工具 > 磁盘清理，选择要清理的选项，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，具体步骤参考扩容 Windows 系统盘，扩容 Windows 数据盘。

最佳实践

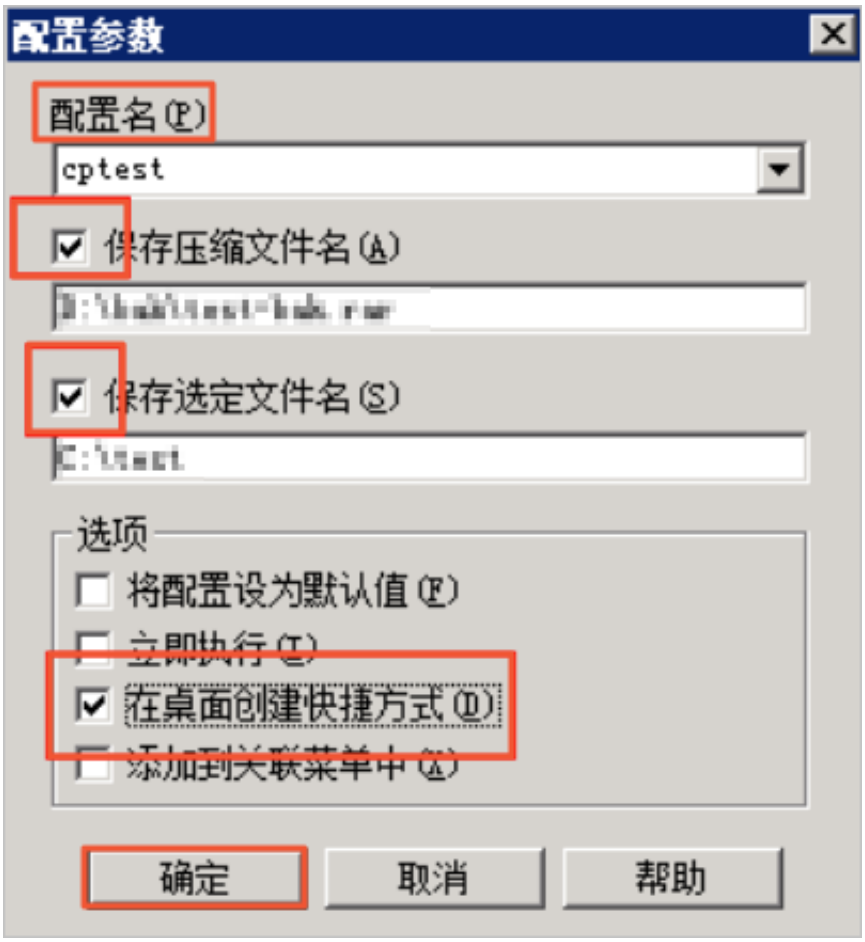
日常需要养成良好的磁盘使用习惯，这里推荐以下几个磁盘使用的最佳实践：

- 文件压缩保存
- 定期清理不必要的应用程序
- 设置磁盘监控

- 文件压缩保存

  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：

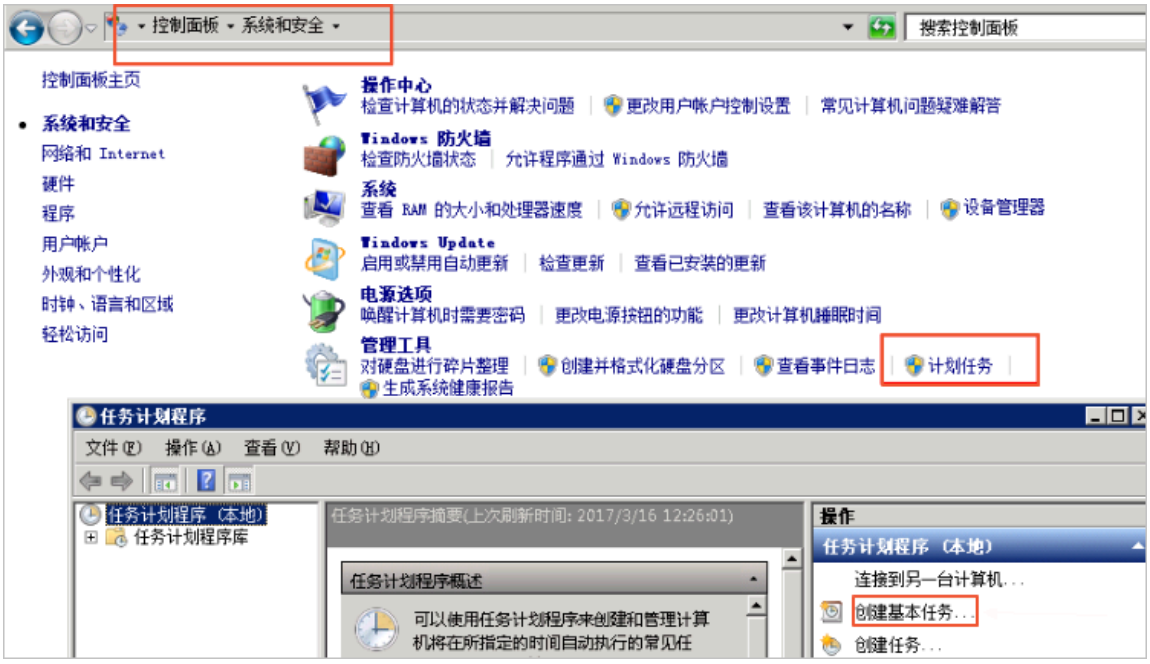
  1. 安装好软件后找到需要压缩的文件，右键该文件，选择添加到压缩文件。
  2. 在设置界面单击窗口上方备份选项卡，然后勾选按掩码产生文件名，注意此时不要单击确定。
  3. 单击窗口上方常规选项卡，单击浏览来定义压缩文件的路径。单击配置，选择保存当前配置为新配置。
  4. 在弹出的配置参数窗口中，输入配置名，勾选保存压缩文件名、保存选定文件名、桌面创建快捷方式，单击确定。

  5. 然后在压缩文件名和参数窗口，单击确定。桌面会生成一个此压缩包的快捷键。

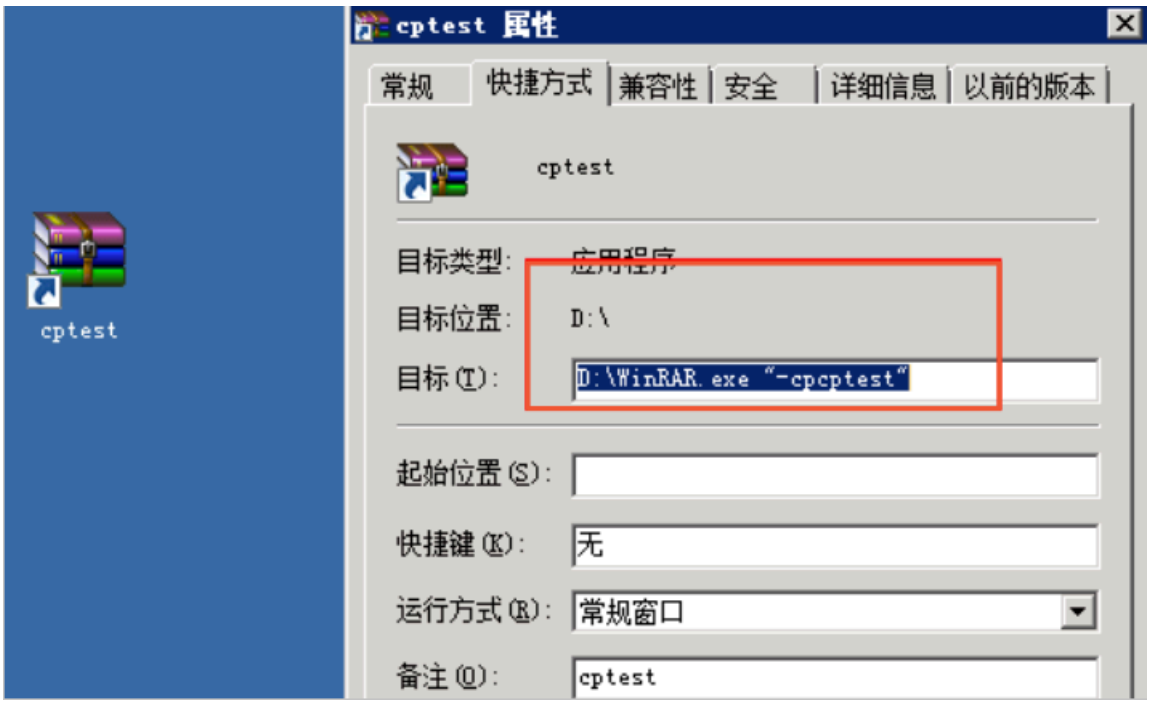
6. 选择开始 > 控制面板，单击系统和安全，单击计划任务，然后在任务计划程序窗口中，选择创建基本任务。



7. 在弹出的窗口中为新任务命名，单击下一步。

8. 选择触发周期，单击下一步。然后选择启动程序，单击下一步。

9. 此时会弹出窗口需要您输入程序或脚本。先找到刚才生成的压缩包快捷键，右键该快捷键，选择属性，复制目标内容。

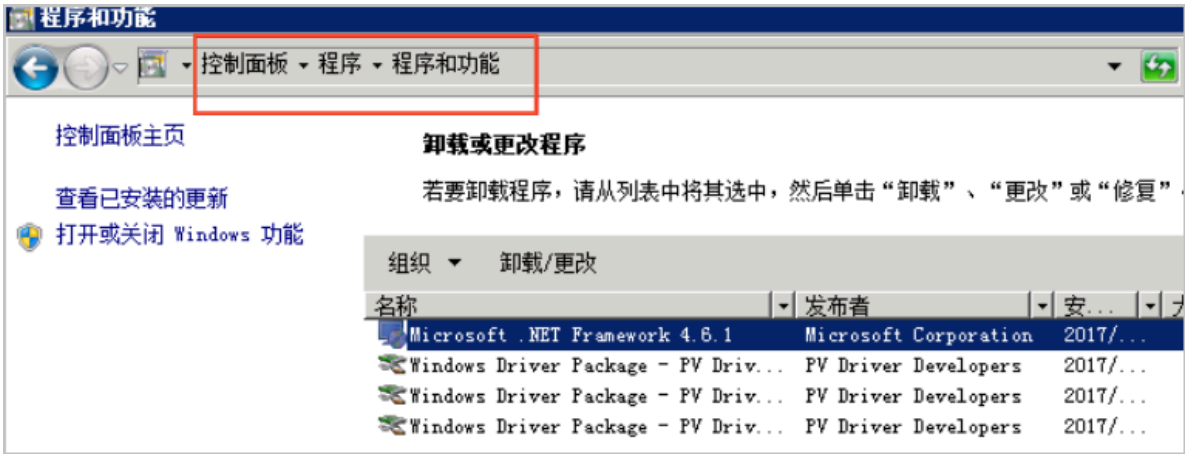


10.然后将复制内容粘贴到启动程序操作中的程序或脚本文本框中，单击确定完成创建。

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。

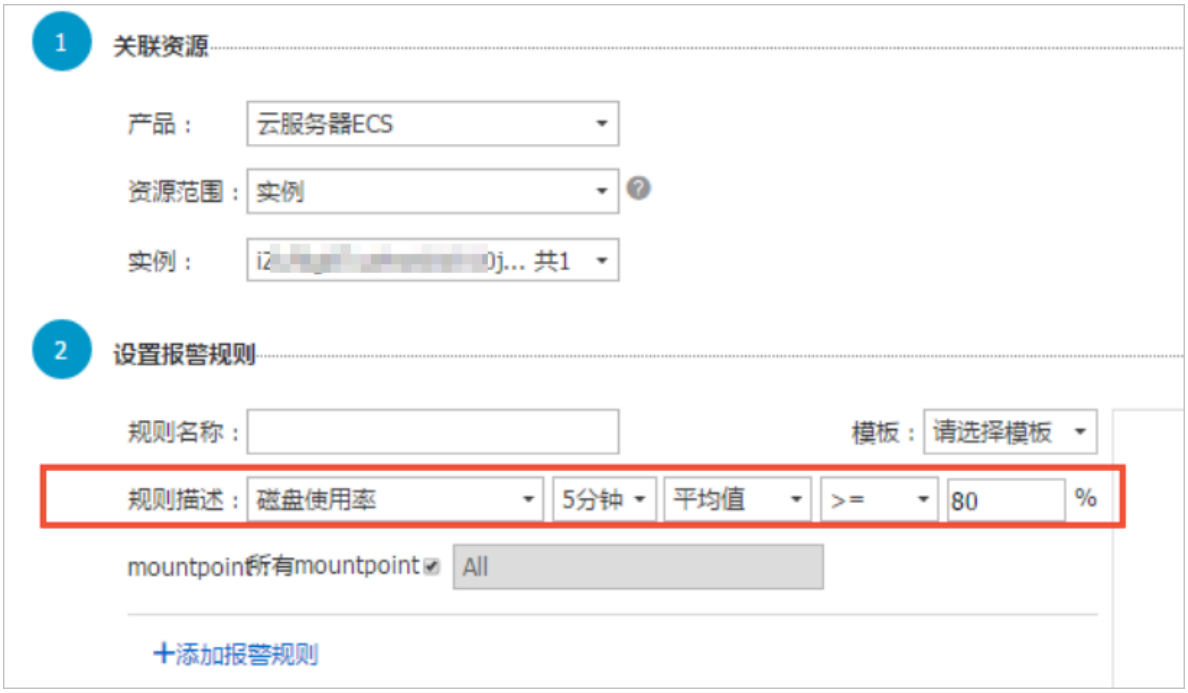
- 定期清理不必要的应用程序

  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窗口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。



3.3 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。

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。

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。

工具说明

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，比如，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。

Linux系统下数据盘分区丢失和数据恢复处理办法

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，您可以通过如下三种办法尝试进行处理：通过fdisk恢复分区、通过testdisk恢复分区 或者 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 新建分区进行恢复。具体操作，请参考 Linux 格式化和挂载数据盘。

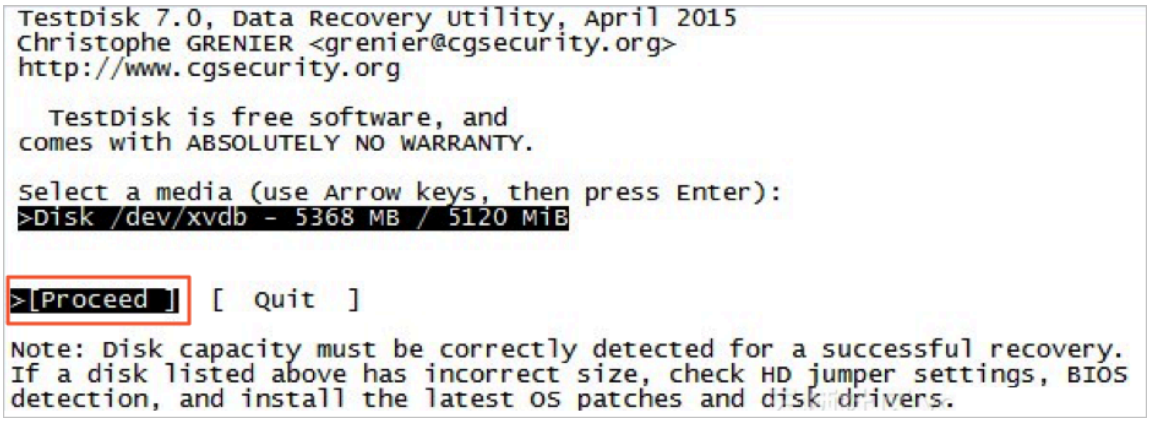
```
[root@Aliyun ~]# fdisk /dev/xvdb
Welcome to fdisk (util-linux 2.23.2).

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。

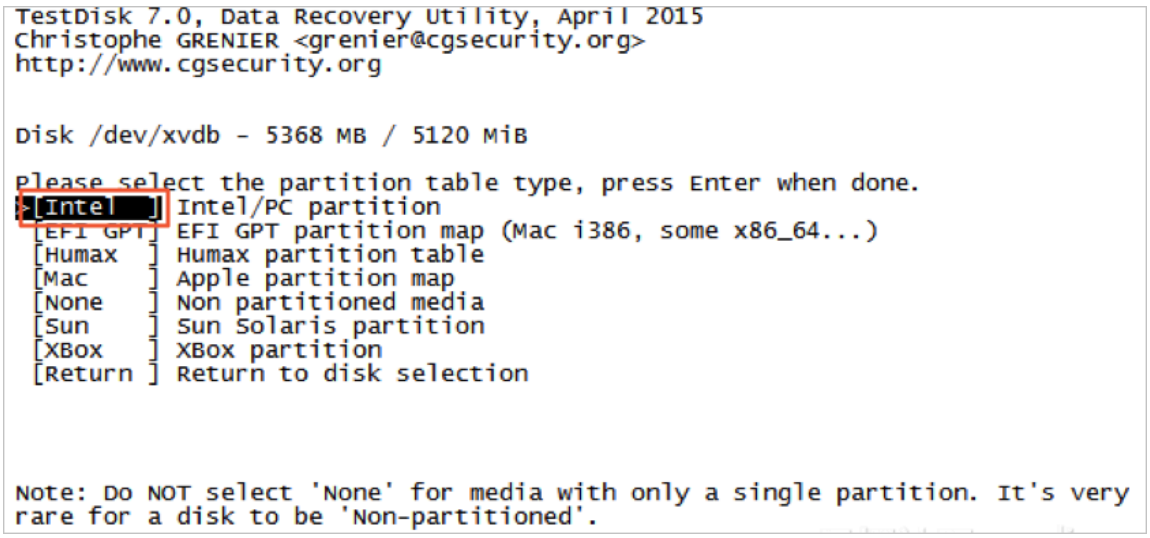
- **通过 testdisk 恢复分区**

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

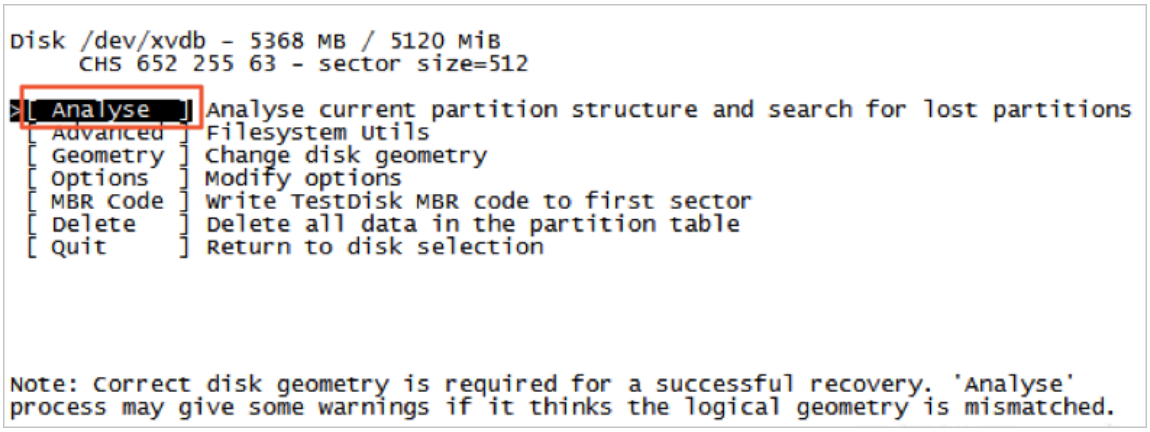
  1. **运行 `testdisk /dev/xvdb`（根据实际情况替换设备名），再选择 `Proceed`（默认值）后按回车键。**



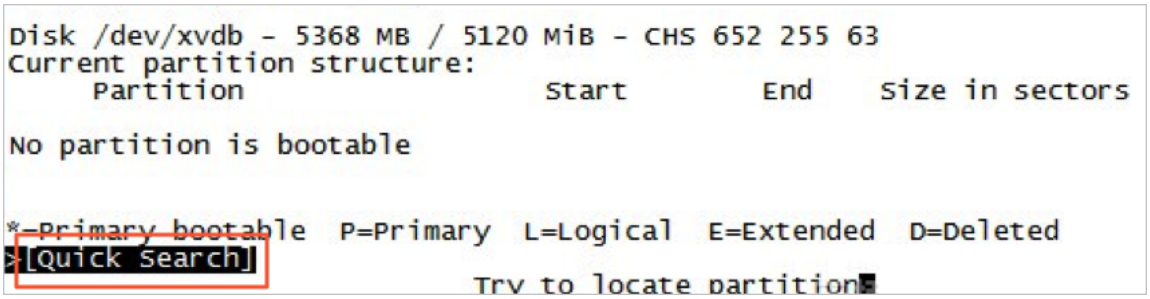
  2. **选择分区表类型进行扫描：一般选择 *`Intel`*（默认）。如果您的数据盘采用GPT分区，选择 *`EFI GPT`*。**



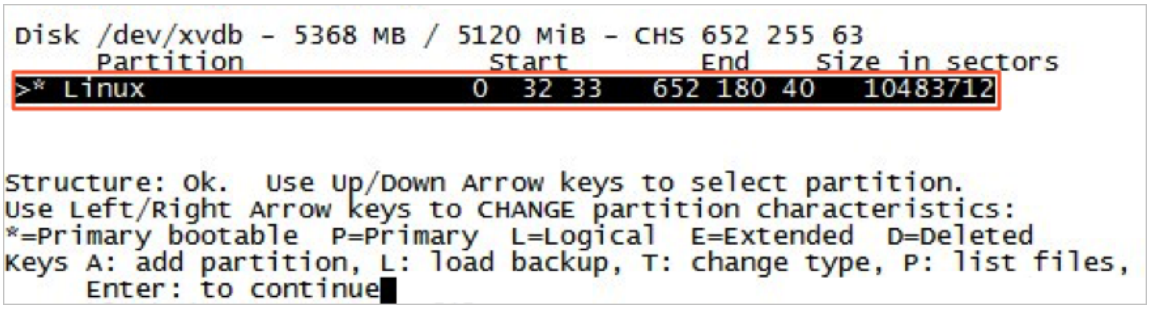
  3. **选择 *`Analyse`* 后按回车键。**



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



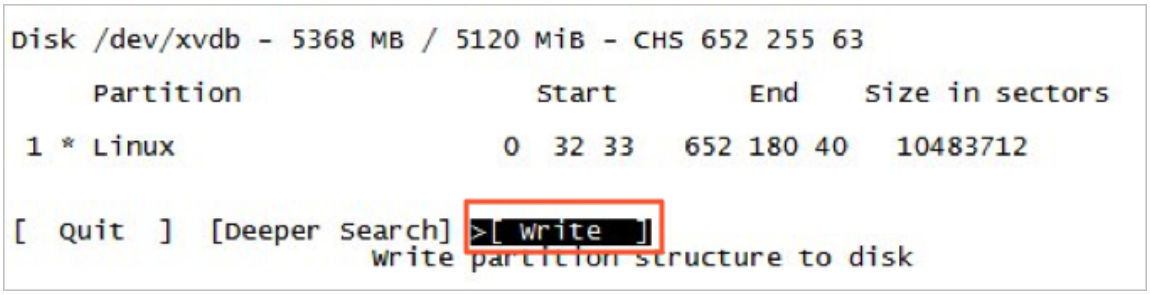
在返回结果中会显示分区信息，如下图所示。



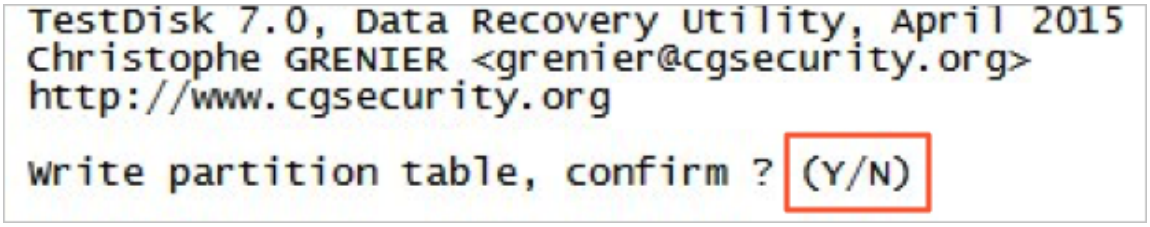
5. 选中分区后，按回车键。
6. 选择 *Write* 保存分区。

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。



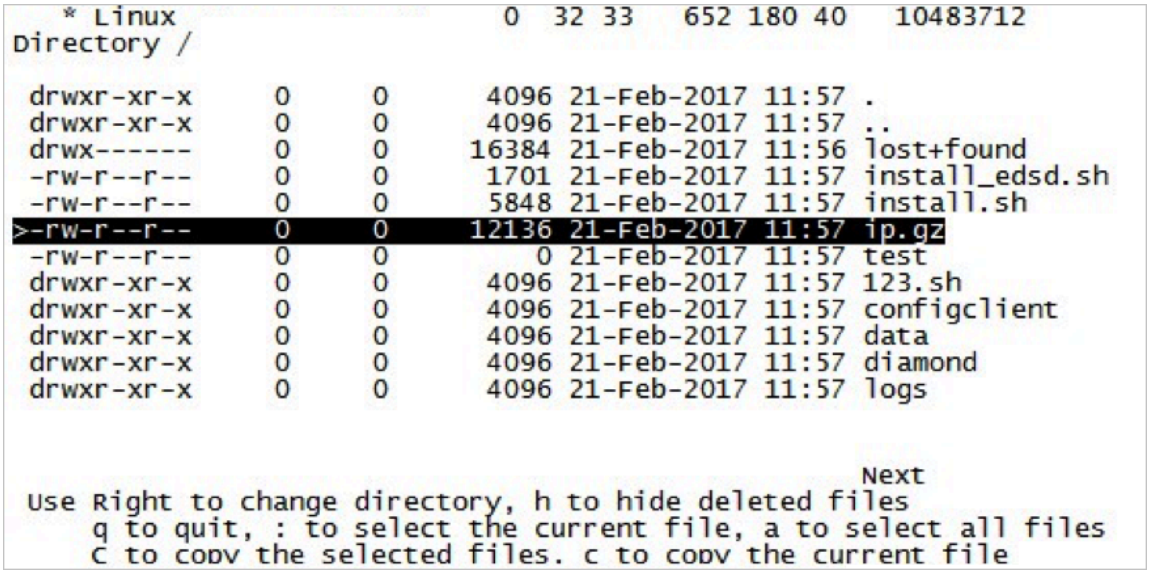
8. 运行 `partprobe /dev/xvdb`（根据实际情况替换设备名）手动刷新分区表。

9. 重新挂载分区，查看数据盘里的数据情况。

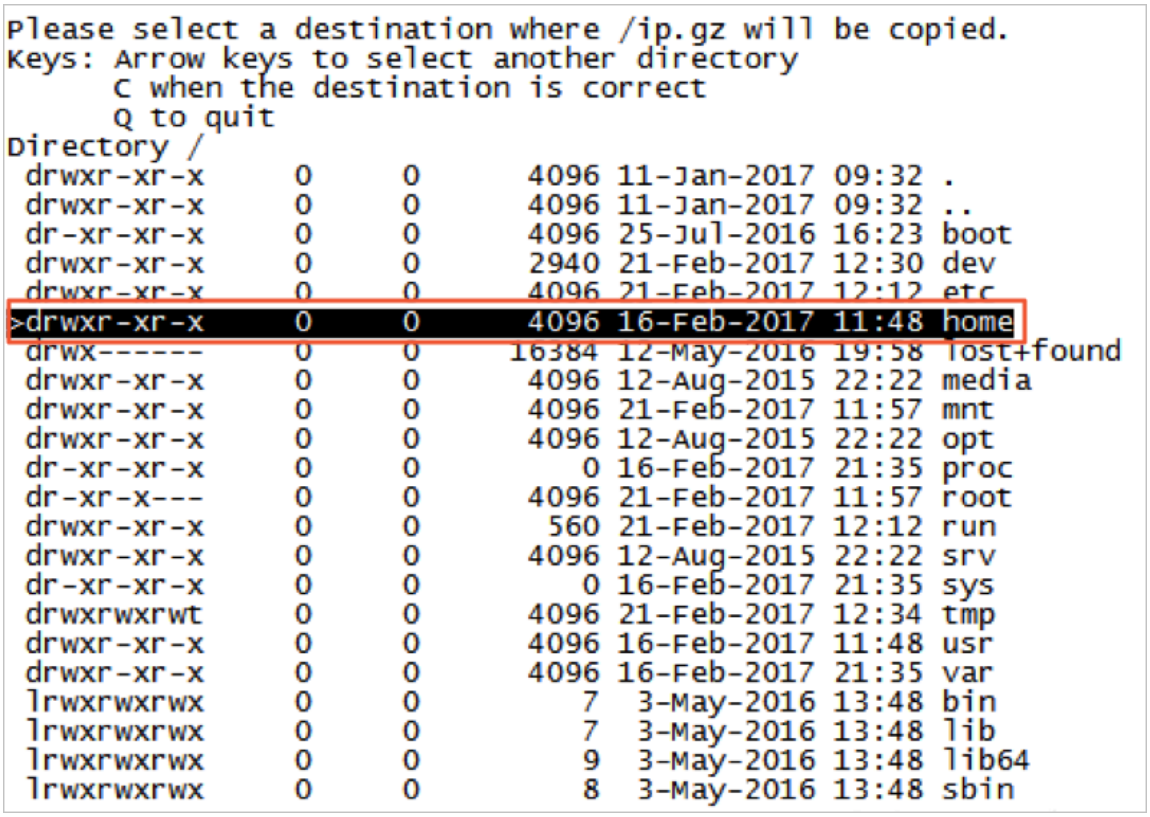
- 通过testdisk直接恢复数据

  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：

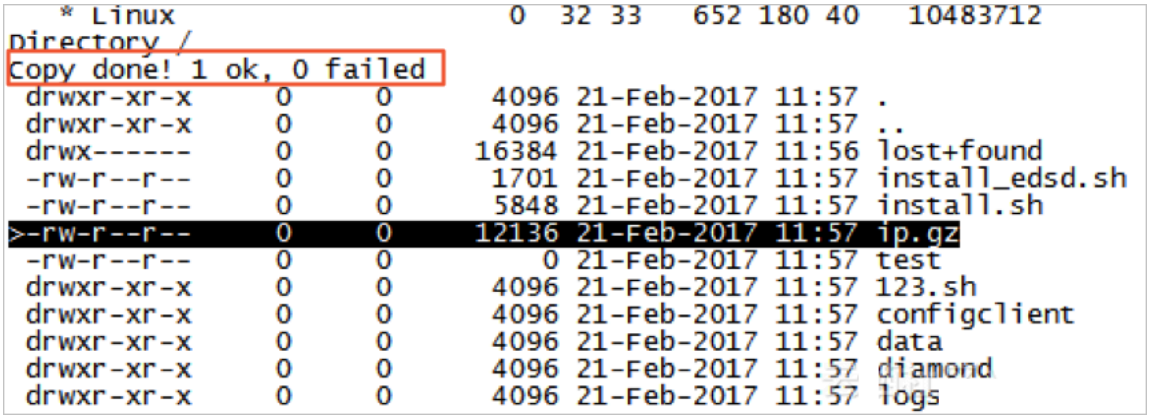
  1. 按 通过testdisk恢复分区 的第1步到第4步描述找到分区。
  2. 按 P 键列出文件。 返回结果如下图。



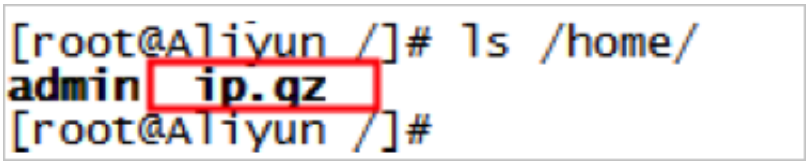
  3. 选中要恢复的文件，再按 C 键。
  4. 选择目标目录。本示例中以恢复到 */home* 为例。



如果您看到 `Copy done! 1 ok, 0 failed` 说明复制成功。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。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，比如中毒、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。

  详情请参考：ECS云服务器自动快照FAQ。

- 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。

3.4 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

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。

工具说明

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。

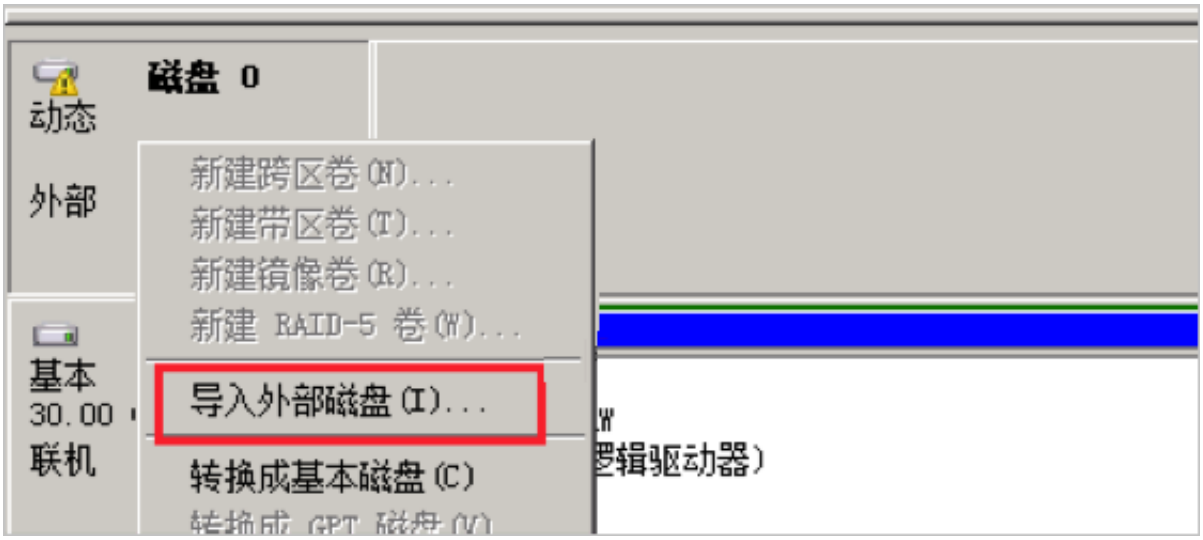
- 数据恢复软件：一般是商业软件，您可以去相应的官网下载使用。主要作用是文件系统异常恢复数据。

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。

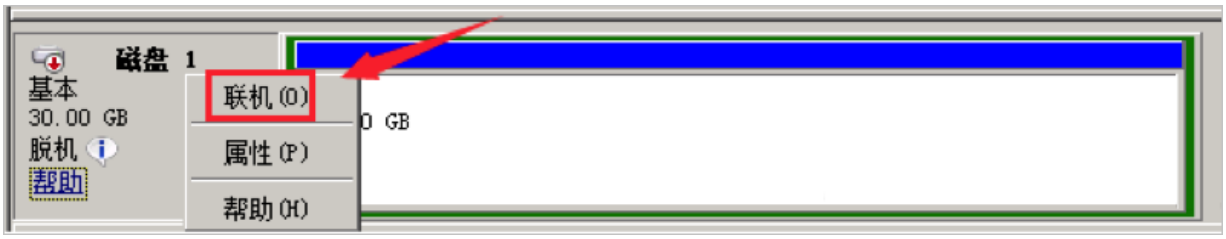


磁盘显示为“脱机”，无法显示分区

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。

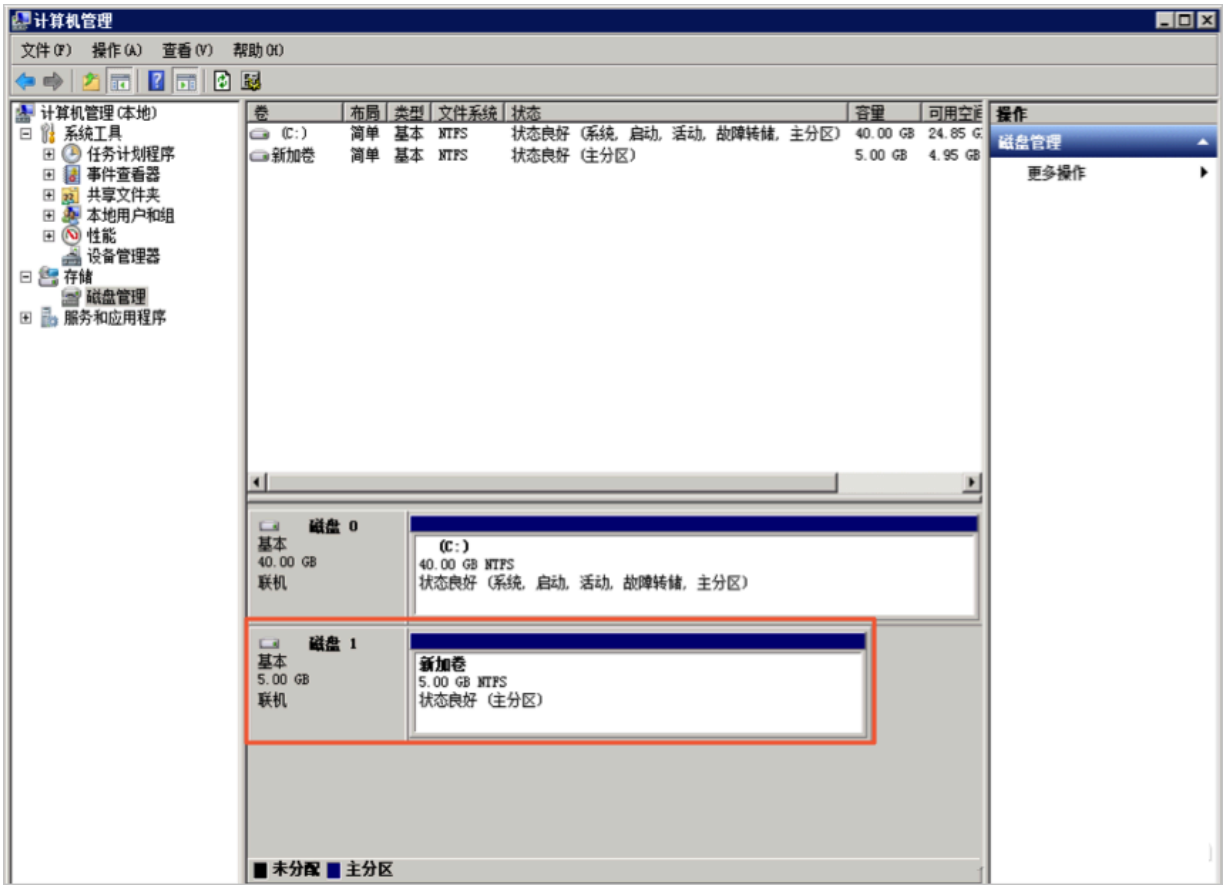
此时，按以下方式处理：

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。



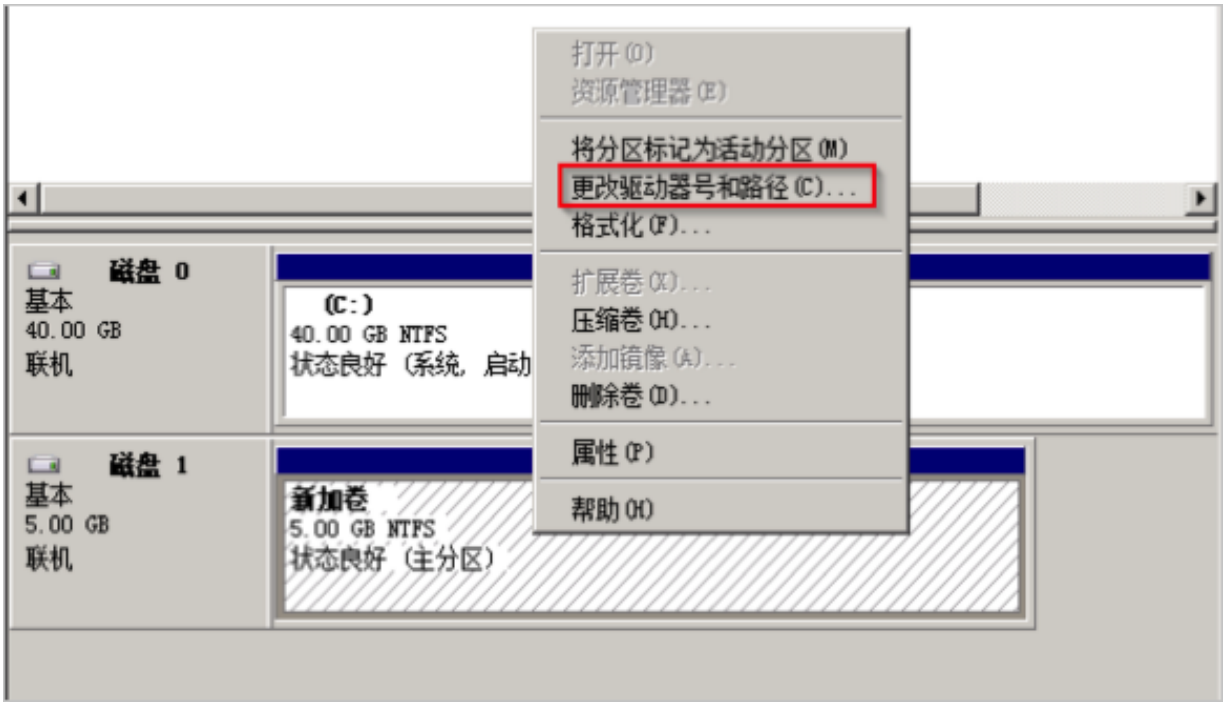
未分配盘符，无法显示分区

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”

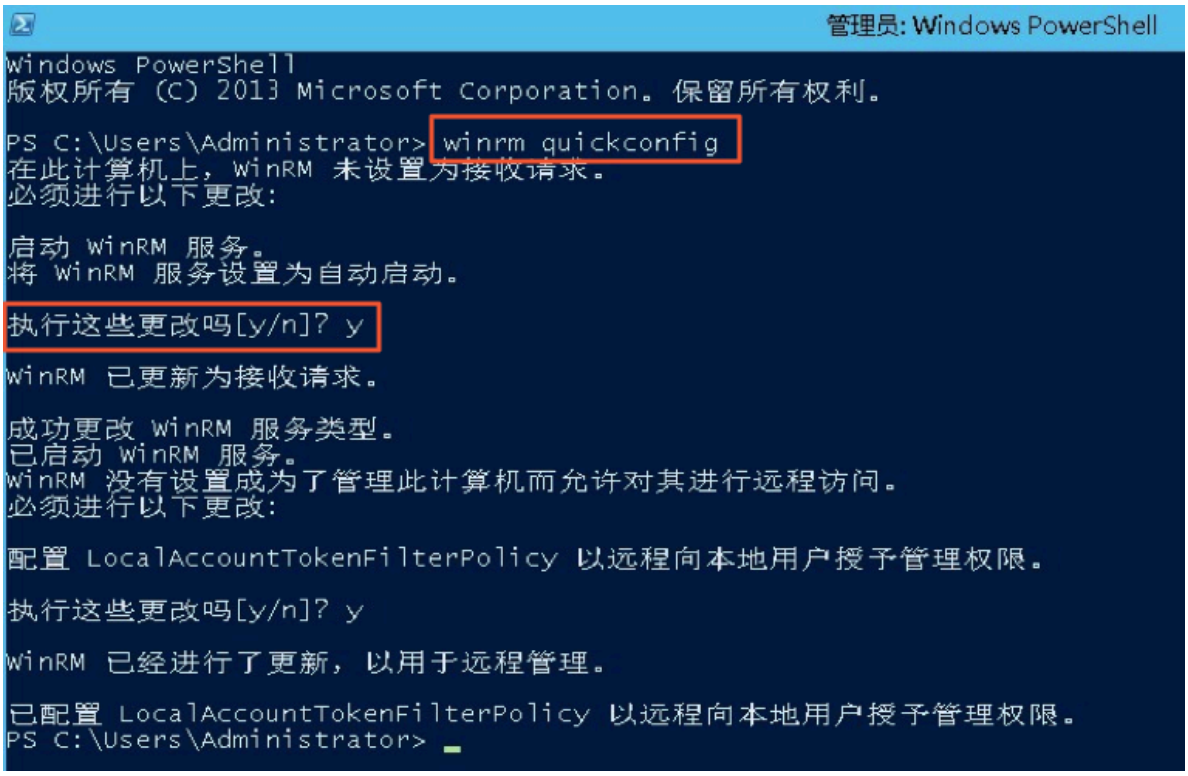
在Windows系统中，您在 磁盘管理器 里无法查看数据盘。系统日志里报错“枚举存储期间出错”，如下图所示。

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，比如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。

您可以参考微软官方的 Dskprobe Overview 文档修复磁盘。

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，比如中毒、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。

    详情请参考：ECS云服务器自动快照FAQ。

  - 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。

4 实例配置

4.1 ECS实例数据传输的实现方式

在信息化高速发展的今天，服务器每天都会与其它单机交换大量文件数据，文件传输对大家来说是家常便饭。因此，其重要性就不言而喻了。文件传输方式各有不同，选择一款合适自己的文件传输工具，在工作中能起到事半功倍的效果。节省资源、方便传输、提升工作效率、加密保护等等。因此，很多文件传输工具应运而生，例如：NC、FTP、SCP、NFS、SAMBA、RSYNC/SERVERSYNC等等，每种方式都有自己的特点。本文将首先简单介绍一下文件传输的基本原理，然后，详细介绍类Unix/Linux、Windows平台上常用文件传输方式，并针对它们各自的特点进行比较，让读者对文件传输方式有比较详尽地了解，从而能够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 NC、FTP、SCP、NFS 等等，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

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。

常用参数

参数	说明
-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

参数	说明
-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
-u	使用UDP传输协议
-v	显示指令执行过程
-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

用法举例

1.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
```

2. 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`

3.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 安全拷贝

SCP（Secure Copy）命令的用法和 RCP 命令格式非常类似，区别就是 SCP 提供更安全保障，SCP 在需要进行验证时会要求你输入密码或口令，一般推荐使用 SCP 命令，因为它比 RCP 更安全。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 是目前较可靠得，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

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它具备更好文件传输保密性。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。

常用示例

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：

生成 RSA 类型的密钥：

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

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/.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
```

从本地复制到远程（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：

第1,2个指定了用户名，命令执行后需要再输入密码，第1个仅指定了远程的目录，文件名字不变，第2个指定了文件名。

第3,4个没有指定用户名，命令执行后需要输入用户名和密码，第3个仅指定了远程的目录，文件名字不变，第4个指定了文件名。

从远程复制到本地：

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

说明：

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。

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。

安装方式

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
```

参数介绍:

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

rsync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信息即可。

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

rsync配置文件说明

```
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
```

4.2 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。本文分别从应用层和系统层来介绍读写分离的实现方法。

应用层实现方法

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，比如说以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。

系统层实现方法

系统层的实现方法包括以下两种：

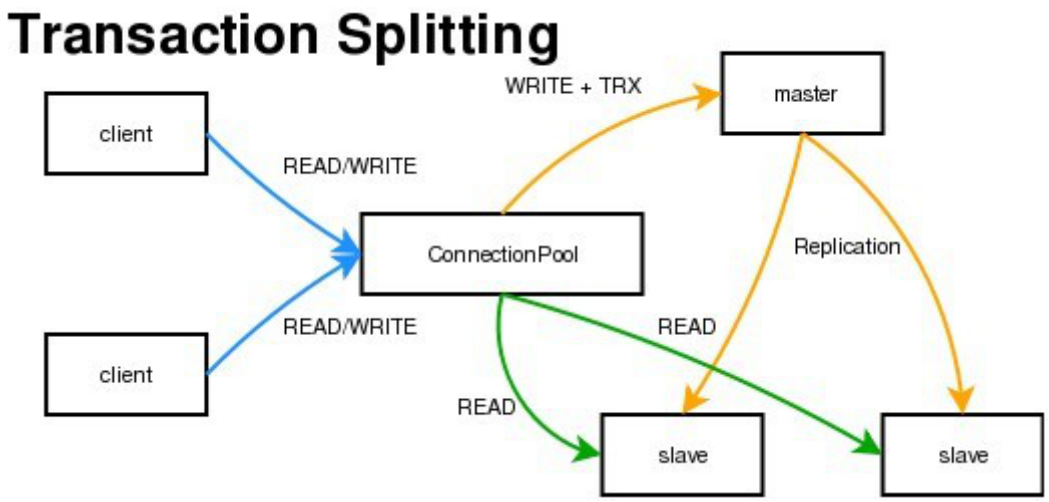
- 使用分布式关系型数据库DRDS实现读写分离。
- 使用中间件MySQL-proxy实现读写分离。

本教程介绍如何使用中间件MySQL-proxy实现读写分离。

MySQL proxy

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原理



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：

- 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

环境说明：

- 主库IP：121.40.18.26
- 从库IP：101.37.36.20
- MySQL-proxy代理IP：116.62.101.76

前期准备：

- 1、新建3台ECS，并安装mysql。
- 2、搭建主从，必须保证主从数据库数据一致。

主环境

1. 修改mysql配置文件。

```
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
```

2. 重启主从服务器中的MySQL服务。

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

3. 在主服务器上建立帐户并授权slave。

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

4. 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)
```

5. 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;
```

6. 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: [redacted]
                  Master_User: syncms
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000007
          Read_Master_Log_Pos: 154
               Relay_Log_File: i[redacted]Z-relay-bin.000003
                Relay_Log_Pos: 367
        Relay_Master_Log_File: mysql-bin.000007
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
```

7. 验证主从同步。

```
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)
```

从库操作

从库中查找testproxy.test1表的数据，与主库一致，主从同步成功

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

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

读写分离配置

1.安装MySQL-Proxy。

```
wget https://cdn.mysql.com/archives/mysql-proxy/mysql-proxy-0.8.5-
linux-glibc2.3-x86-64bit.tar.gz
mkdir /alidata
tar xvf mysql-proxy-0.8.5-linux-glibc2.3-x86-64bit.tar.gz
mv mysql-proxy-0.8.5-linux-glibc2.3-x86-64bit/  /alidata/mysql-proxy-0
.8.5
```

2.环境变量设置。

```
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.读写分离设置。

```
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/
```

授权

1.主库中操作授权，因主从同步的原因，从库也会执行。

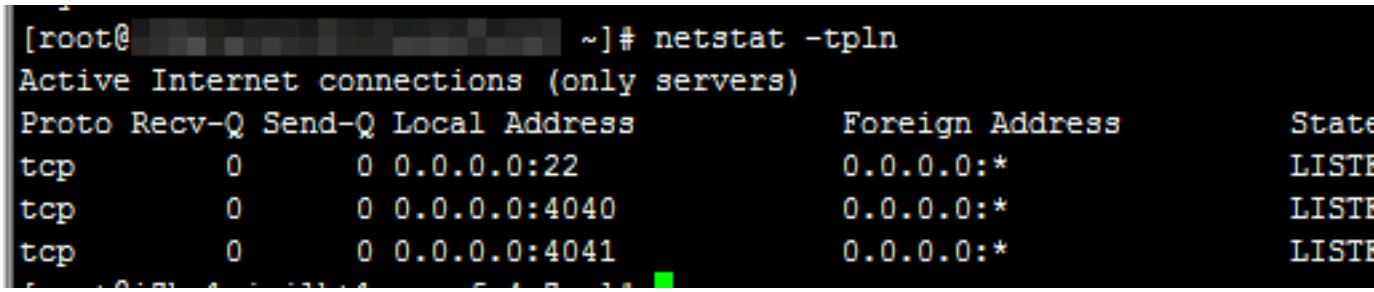
```
mysql -uroot -p95c7586783
grant all on *.* to 'mysql-proxy'@'填写MySQL Proxy IP' identified by '
123456';
flush privileges;
```

2.开启MySQL-Proxy。

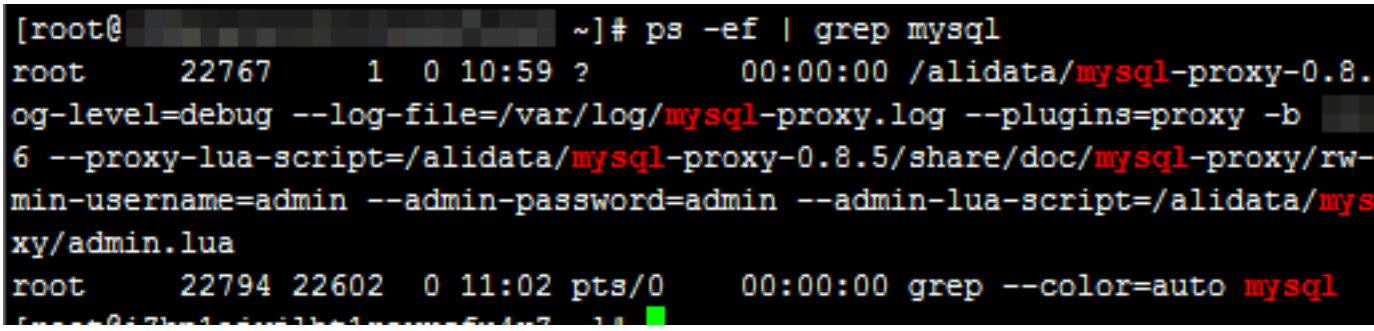
```
mysql-proxy --daemon --log-level=debug --log-file=/var/log/mysql-
proxy.log --plugins=proxy -b 填写master-IP:3306 -r 填写slave-IP:3306
 --proxy-lua-script="/alidata/mysql-proxy-0.8.5/share/doc/mysql-proxy
/rw-splitting.lua" --plugins=admin --admin-username="admin" --admin-
password="admin" --admin-lua-script="/alidata/mysql-proxy-0.8.5/share/
doc/mysql-proxy/admin.lua"
```

3.启动MySQL-Proxy之后，查看端口和相关进程。

```
netstat -tpln
```




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



测试读写分离

1.关闭从复制

```
stop slave;
```

2.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_cli
+-------------+--------------------+---------+------+------+-------------
|           1 | [redacted]:3306 | unknown | rw   | NULL |
|           2 | [redacted]:3306 | unknown | ro   | NULL |
+-------------+--------------------+---------+------+------+-------------
2 rows in set (0.00 sec)
```

第一次连接，会连接到主库上。

```
mysql -umysql-proxy -p123456 -h 116.62.101.76 -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@[redacted] ~]# mysql -umysql-proxy -p123456 -h [redacted]
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 state

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testproxy.test1表的数据显示是从库的数据时，读写分离成功。

```
mysql -umysql-proxy -p123456 -h 116.62.101.76 -P 4040
```

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

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

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

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

4.3 Windows Server 2012 搭建 AD 域

Active Directory（简称AD，即" 活动目录"的意思），是微软服务的核心组件，其主要优势是实现高效管理，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。本文通过 Windows Server 2012 实例示范如何搭建 AD 域。

名词解释

- Domain Controllers（DC）：域控制器
- Organizational Unit（OU）：组织单位
- Distinguished name（DN）：识别名
- Canonical Name（CN）：正式名称

安装指南

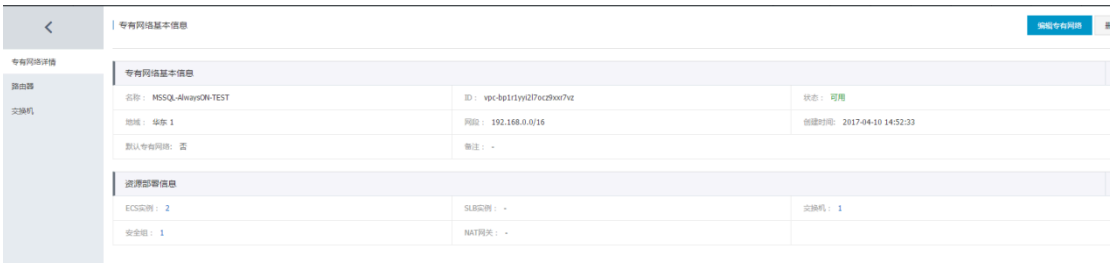
必要条件

- 安装者必须拥有管理员权限。
- 安装分区为NTFS分区。
- 需要支持DNS。

- 需要支持TCP/IP协议，并且需要有固定IP。任何服务器都应该使用固定IP，防止重启实例后IP地址发生变化。本文采用是阿里云VPC网络，手动修改IP会导致IP失效，如果想修改IP，您可以通过控制台修改。

环境

网络采用VPC， 虚拟交换机网段为 192.168.100.0/24，并使用网关。

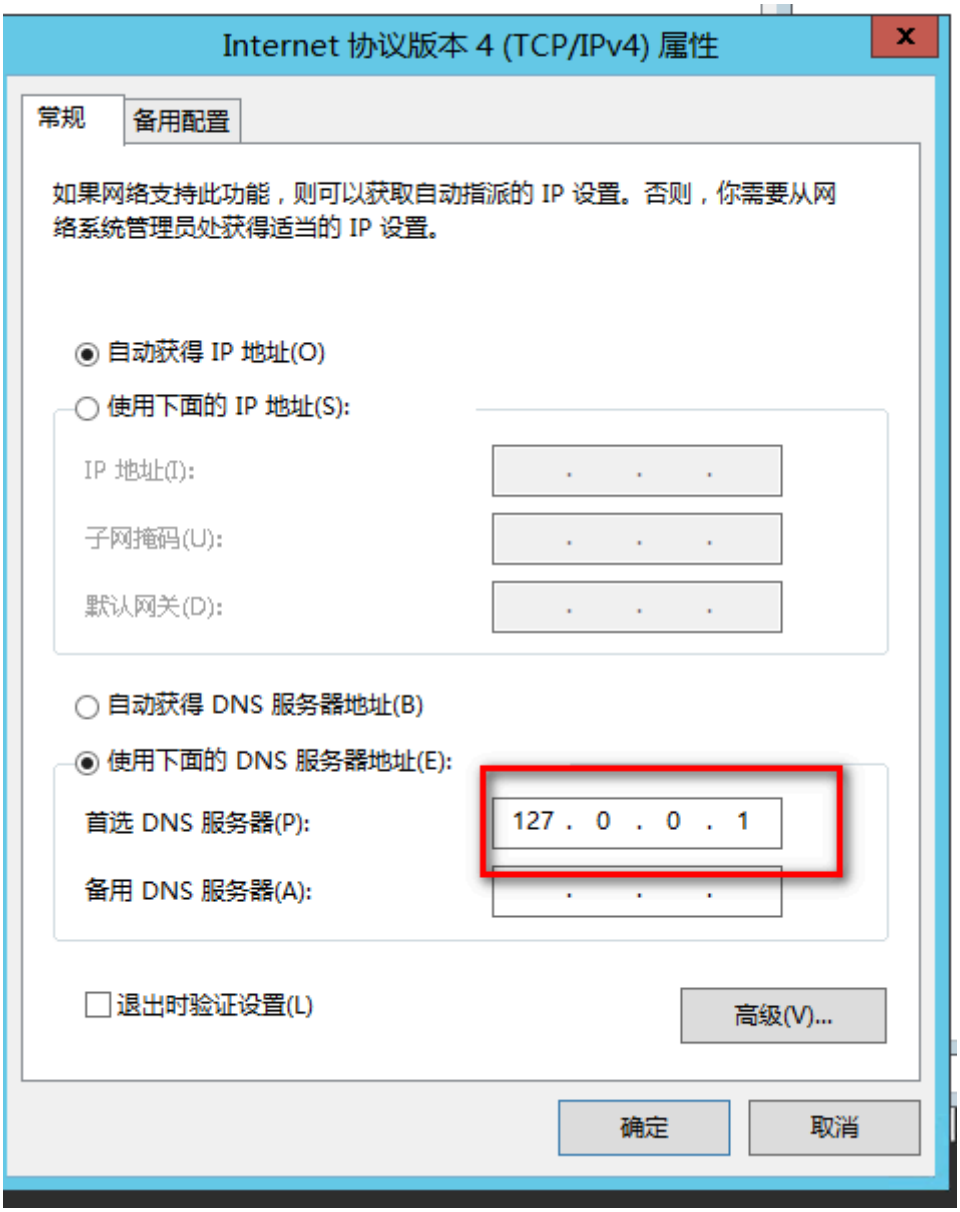


域名

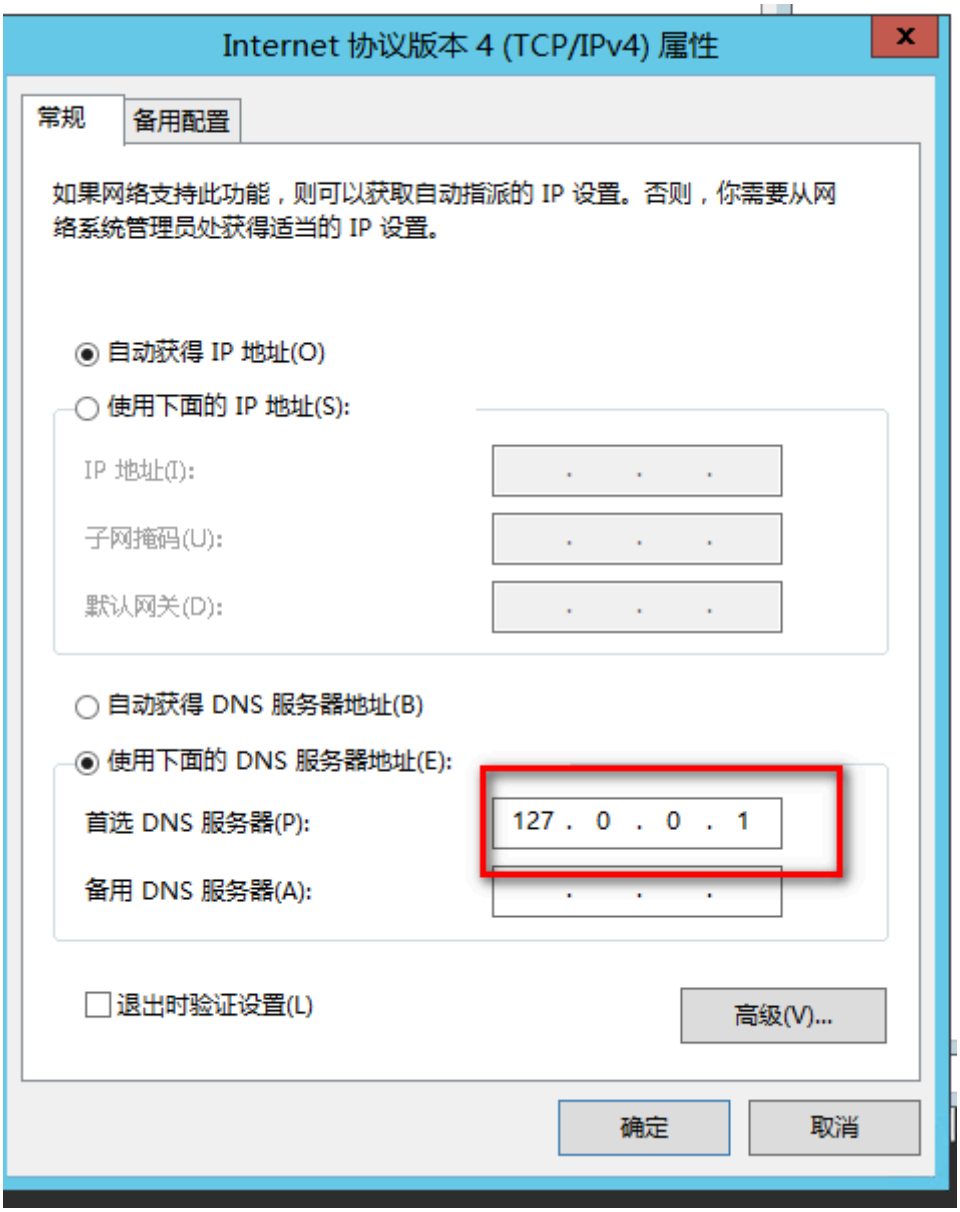
- lyonz.com
- DC：192.168.100.105
- 需要加入域的客户机（Client）IP：192.168.100.106

修改DC 的基本信息

修改DC主机名



修改DC 的DNS（将DNS地址指向自己的IP）



说明:

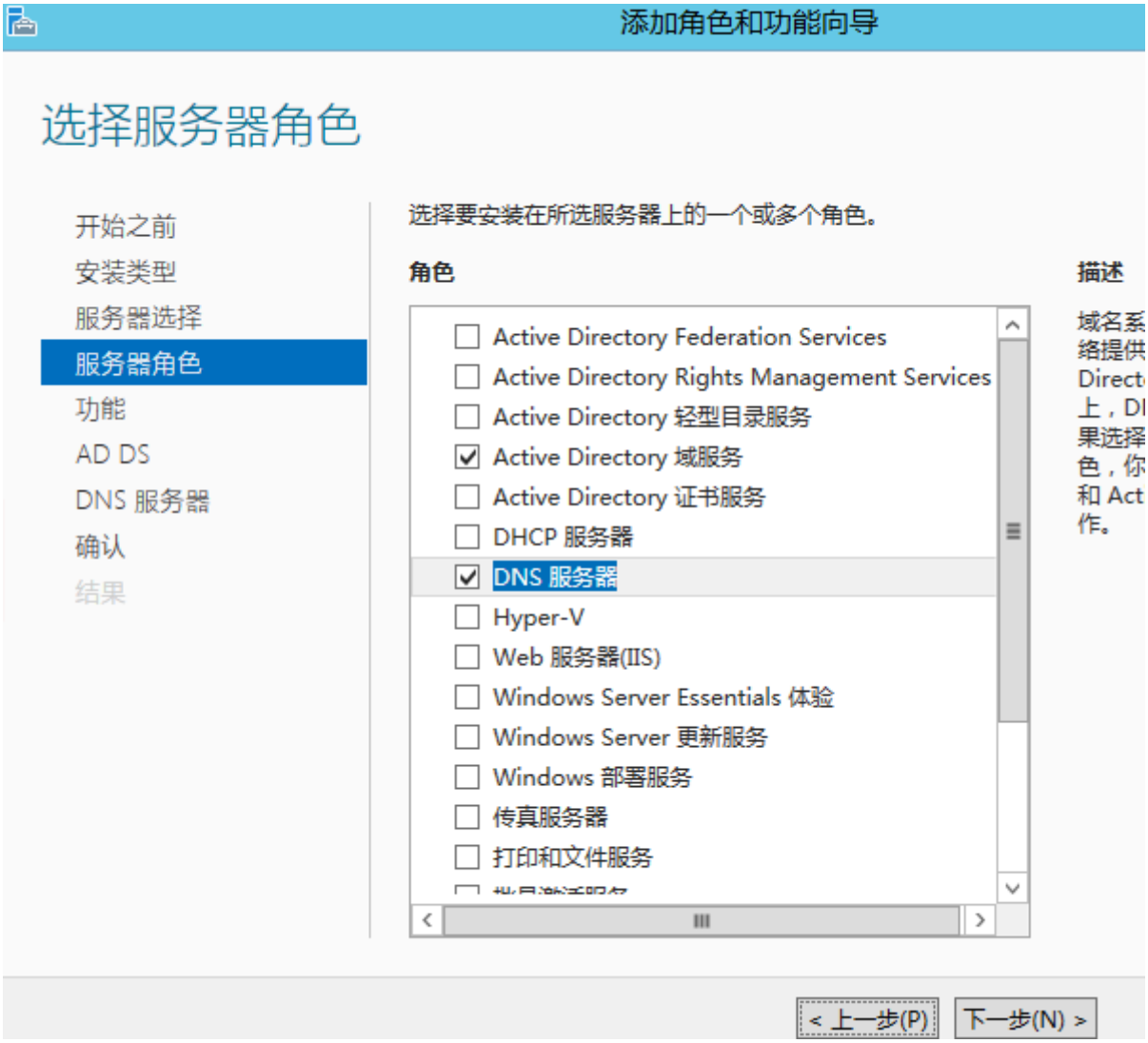
这里不要手动修改服务器的IP地址（手动修改服务器IP不会生效，也无需担心服务器IP会重启发生改变），如果要修改请在控制台操作。

开始安装

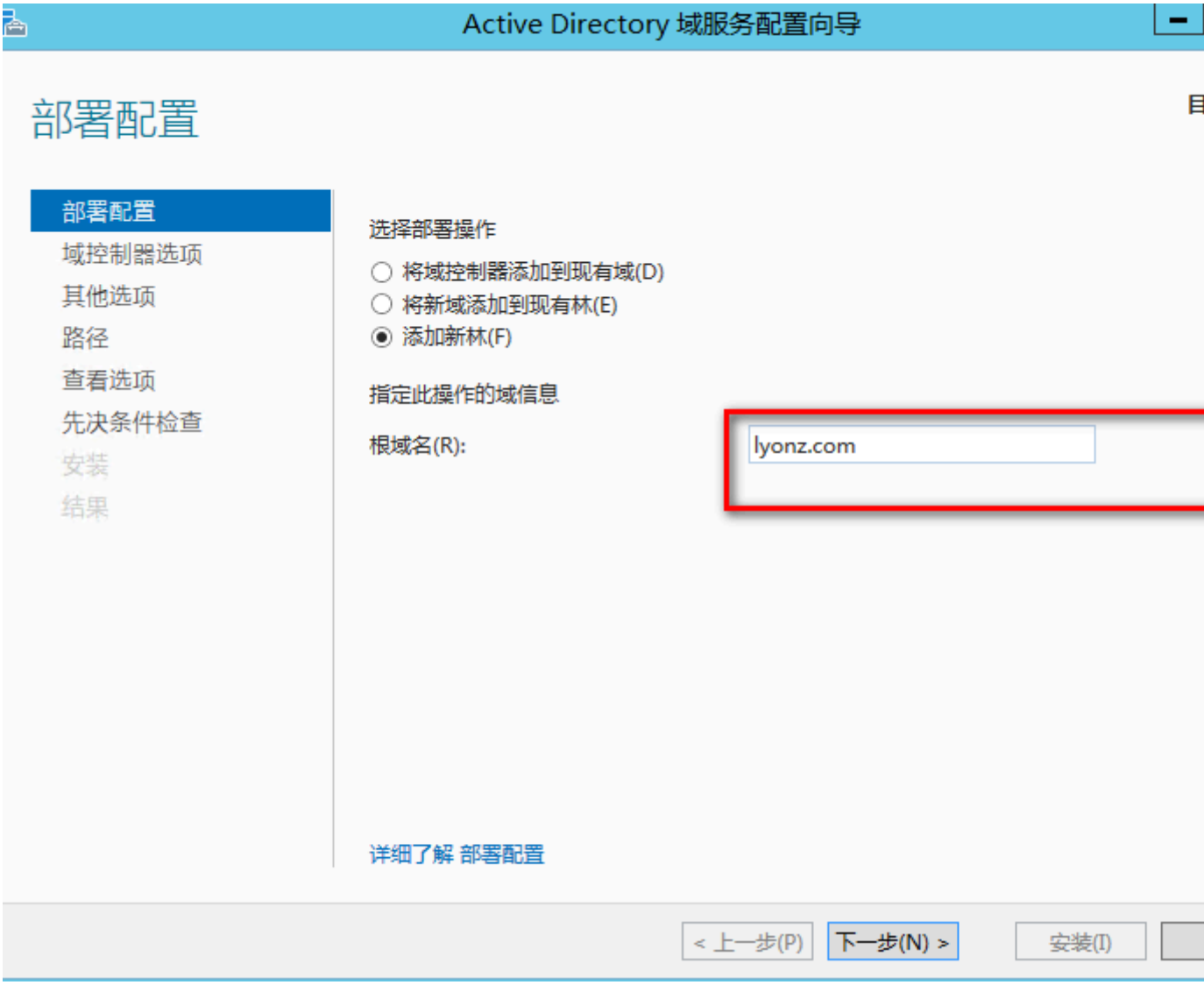

服务器管理器 ▸ 仪表板
仪表板
本地服务器
所有服务器
文件和存储服务
欢迎使用服务器管理器
快速启动(Q)
新增功能(W)
了解详细信息(L)
1 配置此本地服务器
2 添加角色和功能
3 添加要管理的其他服务器
4 创建服务器组
5 将此服务器连接到云服务
角色和服务器组
角色: 1 | 服务器组: 1 | 服务器总数: 1
文件和存储服务 1
本地服务器

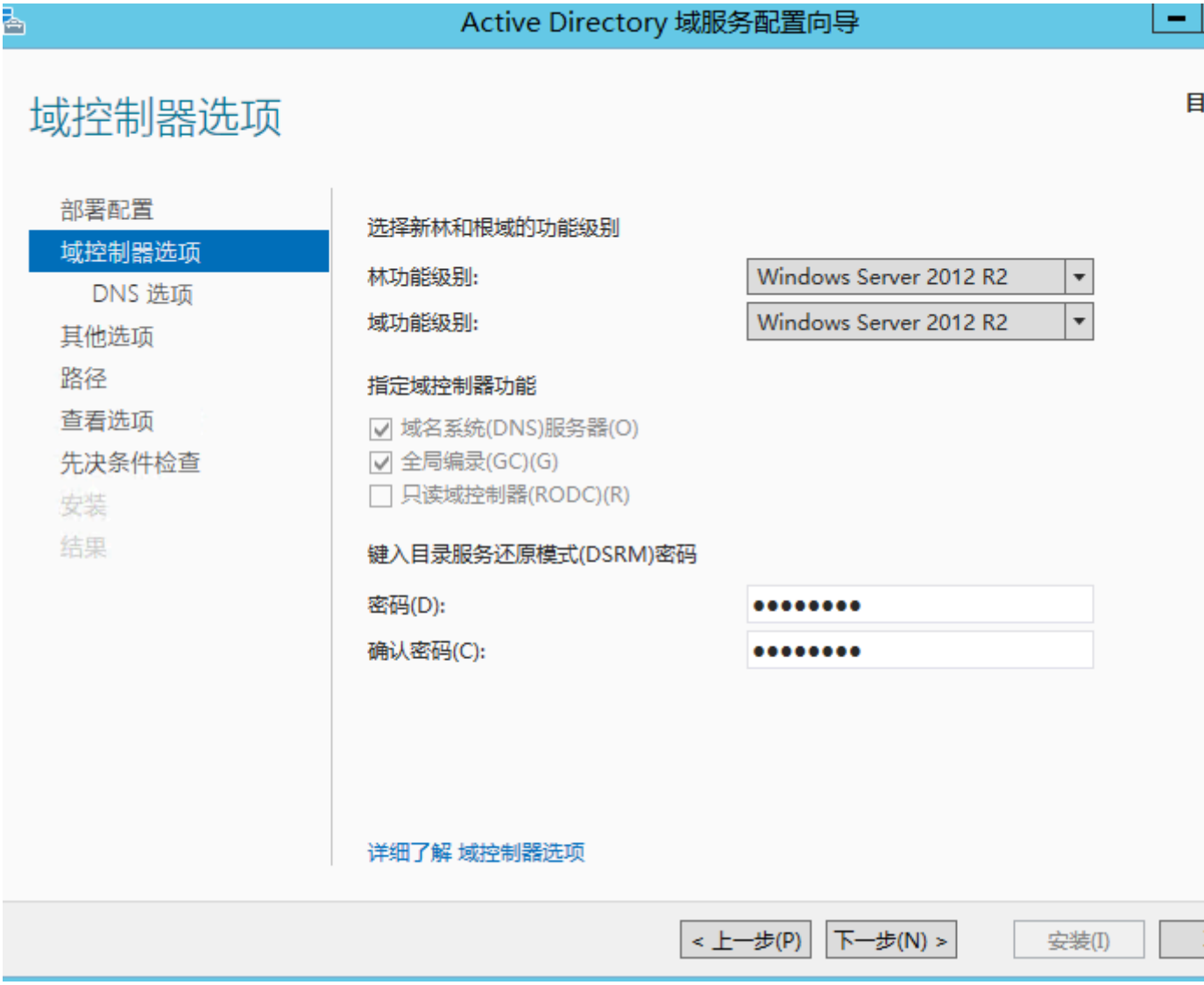

添加角色和功能向导
选择安装类型
开始之前
安装类型
服务器选择
服务器角色
功能
确认
结果
选择安装类型。你可以在正在运行的物理计算机、虚拟机或脱机虚拟硬盘(VHD)上安装角色
基于角色或基于功能的安装
通过添加角色、角色服务和功能来配置单个服务器。
远程桌面服务安装
为虚拟桌面基础结构(VDI)安装所需的角色服务以创建基于虚拟机或基于会话的桌面部署

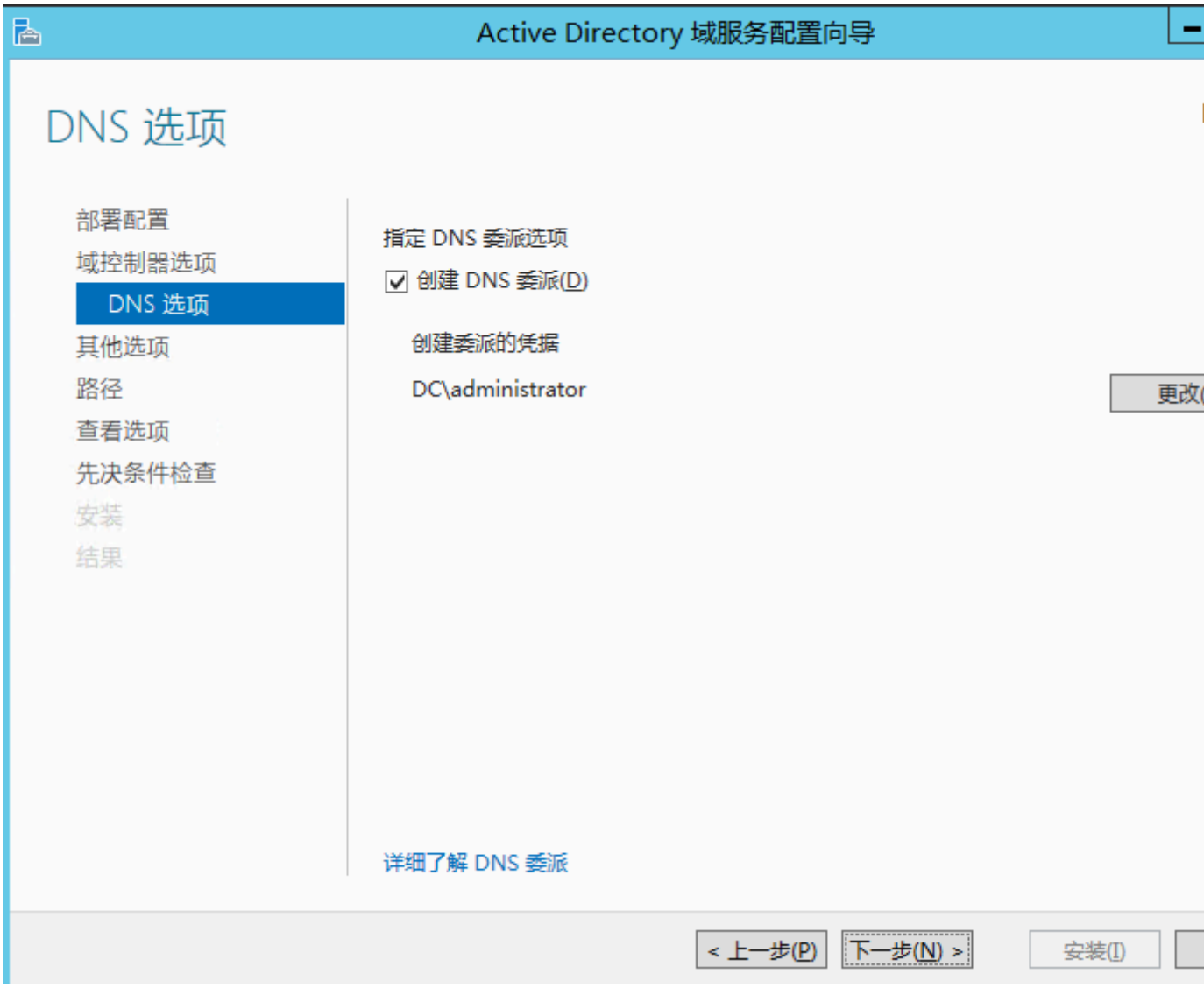

添加角色和功能向导
选择目标服务器
开始之前
安装类型
服务器选择
服务器角色
功能
确认
结果
选择要安装角色和功能的服务器或虚拟硬盘。
从服务器池中选择服务器
选择虚拟硬盘
服务器池
筛选器:
名称
IP 地址
操作系统
iZ3wny1h42ioiuZ
169.254.60.17...
Microsoft Windows Server 2012 R2


添加角色和功能向导
选择服务器角色
开始之前
安装类型
服务器选择
服务器角色
功能
AD DS
DNS 服务器
确认
结果
选择要安装在所选服务器上的一个或多个角色。
角色
描述
Active Directory Federation Services
Active Directory Rights Management Services
Active Directory 轻型目录服务
Active Directory 域服务
Active Directory 证书服务
DHCP 服务器
DNS 服务器
Hyper-V
Web 服务器(IIS)
Windows Server Essentials 体验
Windows Server 更新服务
Windows 部署服务
传真服务器
打印和文件服务
< 上一步(P)
下一步(N) >

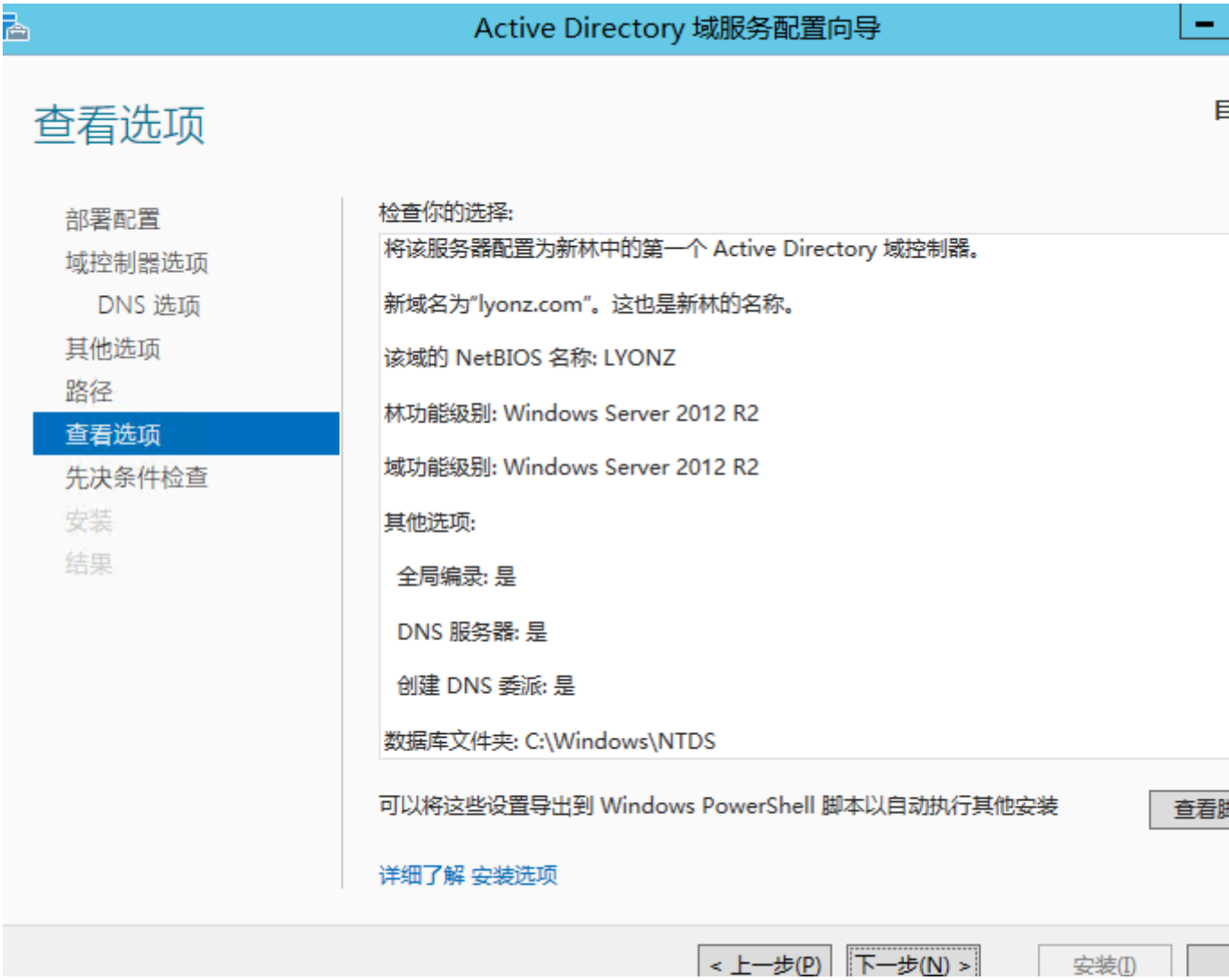

服务器管理器
仪表板
工具(T)
视图(
部署后配置
任务
DC 中的 Active Directory 域服务 所需的配置
将此服务器提升为域控制器
任务详细信息
仪表板
本地服务器
所有服务器
AD DS
DNS
文件和存储服务
地服务器
2 添加角色和功能
3 添加要管理的其他服务器
4 创建服务器组
5 将此服务器连接到云服务
新增功能(W)
了解详细信息(L)
角色和服务器组

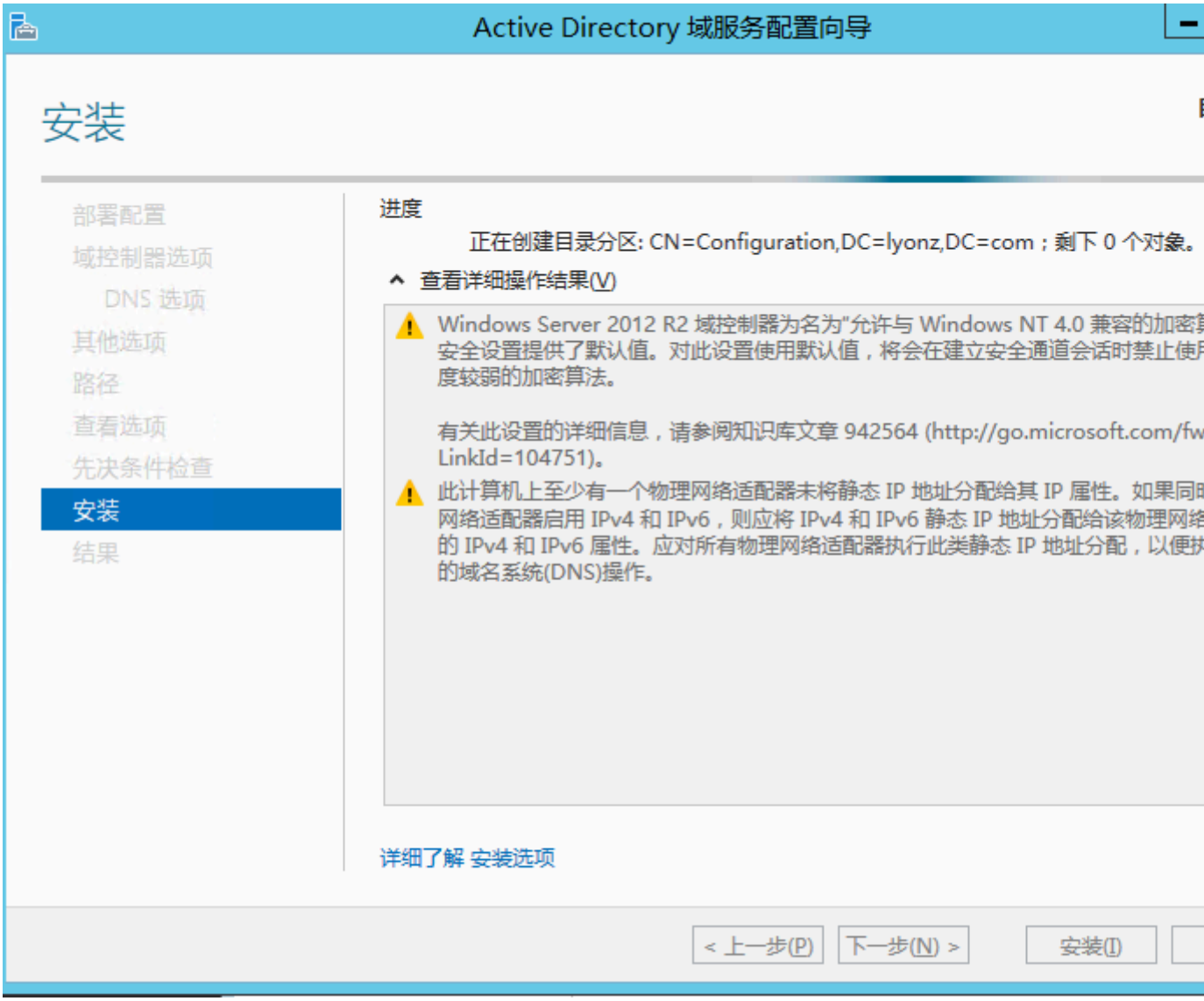

Active Directory 域服务配置向导
部署配置
部署配置
域控制器选项
其他选项
路径
查看选项
先决条件检查
安装
结果
选择部署操作
将域控制器添加到现有域(D)
将新域添加到现有林(E)
添加新林(F)
指定此操作的域信息
根域名(R):
lyonz.com
详细了解 部署配置
< 上一步(P)
下一步(N) >
安装(I)


Active Directory 域服务配置向导
域控制器选项
部署配置
域控制器选项
DNS 选项
其他选项
路径
查看选项
先决条件检查
安装
结果
选择新林和根域的功能级别
林功能级别:
Windows Server 2012 R2
域功能级别:
Windows Server 2012 R2
指定域控制器功能
域名系统(DNS)服务器(O)
全局编录(GC)(G)
只读域控制器(RODC)(R)
键入目录服务还原模式(DSRM)密码
密码(D):
确认密码(C):
详细了解 域控制器选项
< 上一步(P)
下一步(N) >
安装(I)


Active Directory 域服务配置向导
DNS 选项
部署配置
域控制器选项
DNS 选项
其他选项
路径
查看选项
先决条件检查
安装
结果
指定 DNS 委派选项
创建 DNS 委派(D)
创建委派的凭据
DC\administrator
更改(
详细了解 DNS 委派
< 上一步(P)
下一步(N) >
安装(I)


Active Directory 域服务配置向导
其他选项
部署配置
域控制器选项
DNS 选项
其他选项
路径
查看选项
先决条件检查
安装
结果
确保为域分配了 NetBIOS 名称，并在必要时更改该名称
NetBIOS 域名:
LYONZ
详细了解 其他选项


Active Directory 域服务配置向导
查看选项
部署配置
域控制器选项
DNS 选项
其他选项
路径
查看选项
先决条件检查
安装
结果
检查你的选择:
将该服务器配置为新林中的第一个 Active Directory 域控制器。
新域名为"lyonz.com"。这也是新林的名称。
该域的 NetBIOS 名称: LYONZ
林功能级别: Windows Server 2012 R2
域功能级别: Windows Server 2012 R2
其他选项:
全局编录: 是
DNS 服务器: 是
创建 DNS 委派: 是
数据库文件夹: C:\Windows\NTDS
可以将这些设置导出到 Windows PowerShell 脚本以自动执行其他安装
详细了解 安装选项
< 上一步(P)
下一步(N) >
安装(I)


Active Directory 域服务配置向导
安装
部署配置
域控制器选项
DNS 选项
其他选项
路径
查看选项
先决条件检查
安装
结果
进度
正在创建目录分区: CN=Configuration,DC=lyonz,DC=com；剩下 0 个对象。
查看详细操作结果(V)
Windows Server 2012 R2 域控制器为名为"允许与 Windows NT 4.0 兼容的加密算
安全设置提供了默认值。对此设置使用默认值，将会在建立安全通道会话时禁止使用
度较弱的加密算法。
有关此设置的详细信息，请参阅知识库文章 942564 (http://go.microsoft.com/fw
LinkId=104751)。
此计算机上至少有一个物理网络适配器未将静态 IP 地址分配给其 IP 属性。如果同时
网络适配器启用 IPv4 和 IPv6，则应将 IPv4 和 IPv6 静态 IP 地址分配给该物理网络
的 IPv4 和 IPv6 属性。应对所有物理网络适配器执行此类静态 IP 地址分配，以便找
的域名系统(DNS)操作。
详细了解 安装选项
< 上一步(P)
下一步(N) >
安装(I)

验证客户端的加入

在云上安装AD和我们线下安装AD步骤其实一样，但客户端加入域的步骤稍有不同，需要先修改客户端的SID，这是因为阿里云ECS Windows Server 2012系统采用的同一个镜像，所以SID是相同的，如果不修改，在加入域的时候会提示SID相同。

修改客户端的SID

Winodws Server 2012：

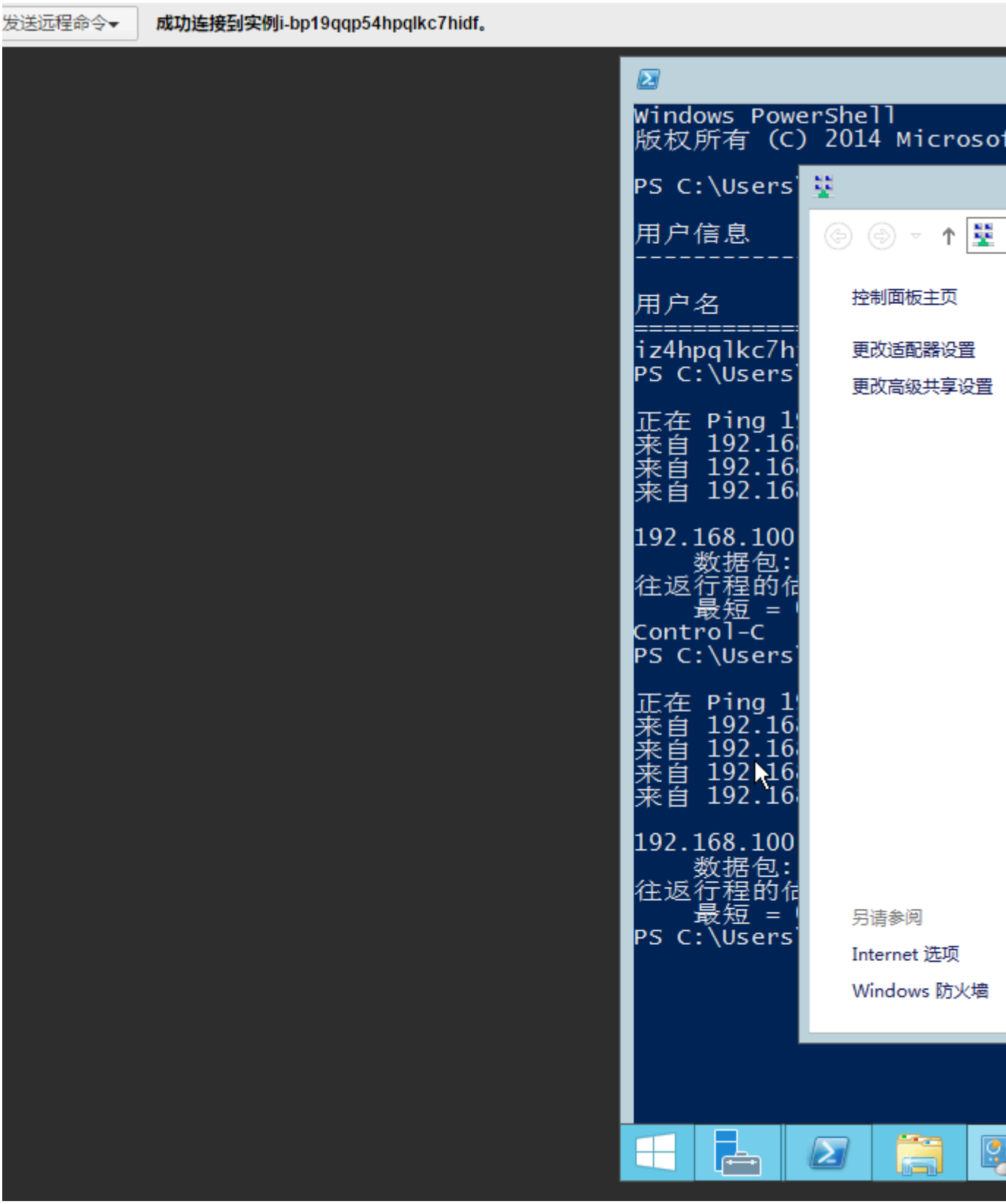
在 powershell 界面执行如下命令：

首先切换到脚本存放的路径，

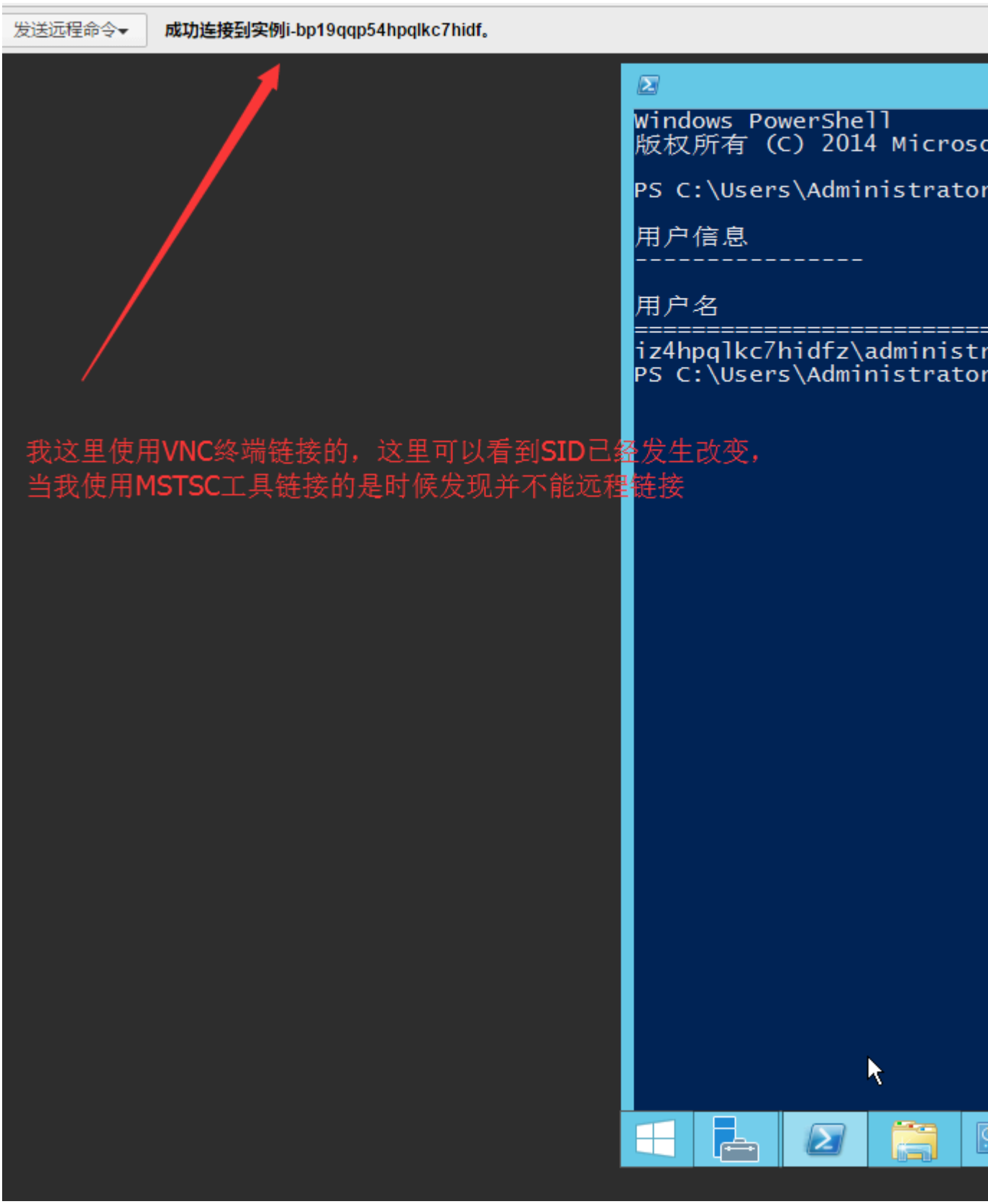
```
.\Sysprep.ps1 -ReserveHostname -ReserveNetwork -skiprearm -post_action
 "reboot"
```

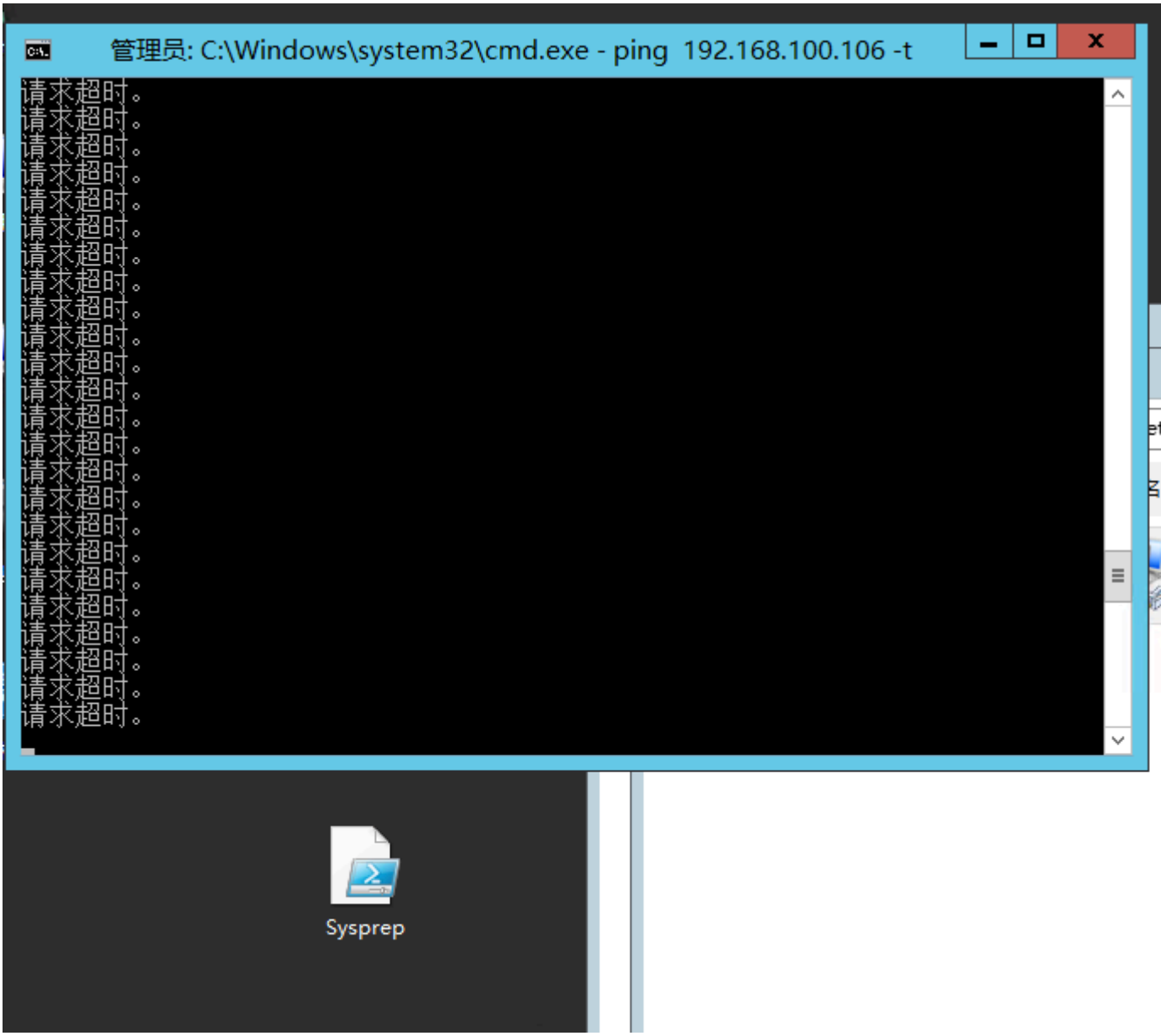
执行上面的命令后，服务器会重新初始化SID，初始化完成后，机器会重启，服务器启动后需要注意两点：

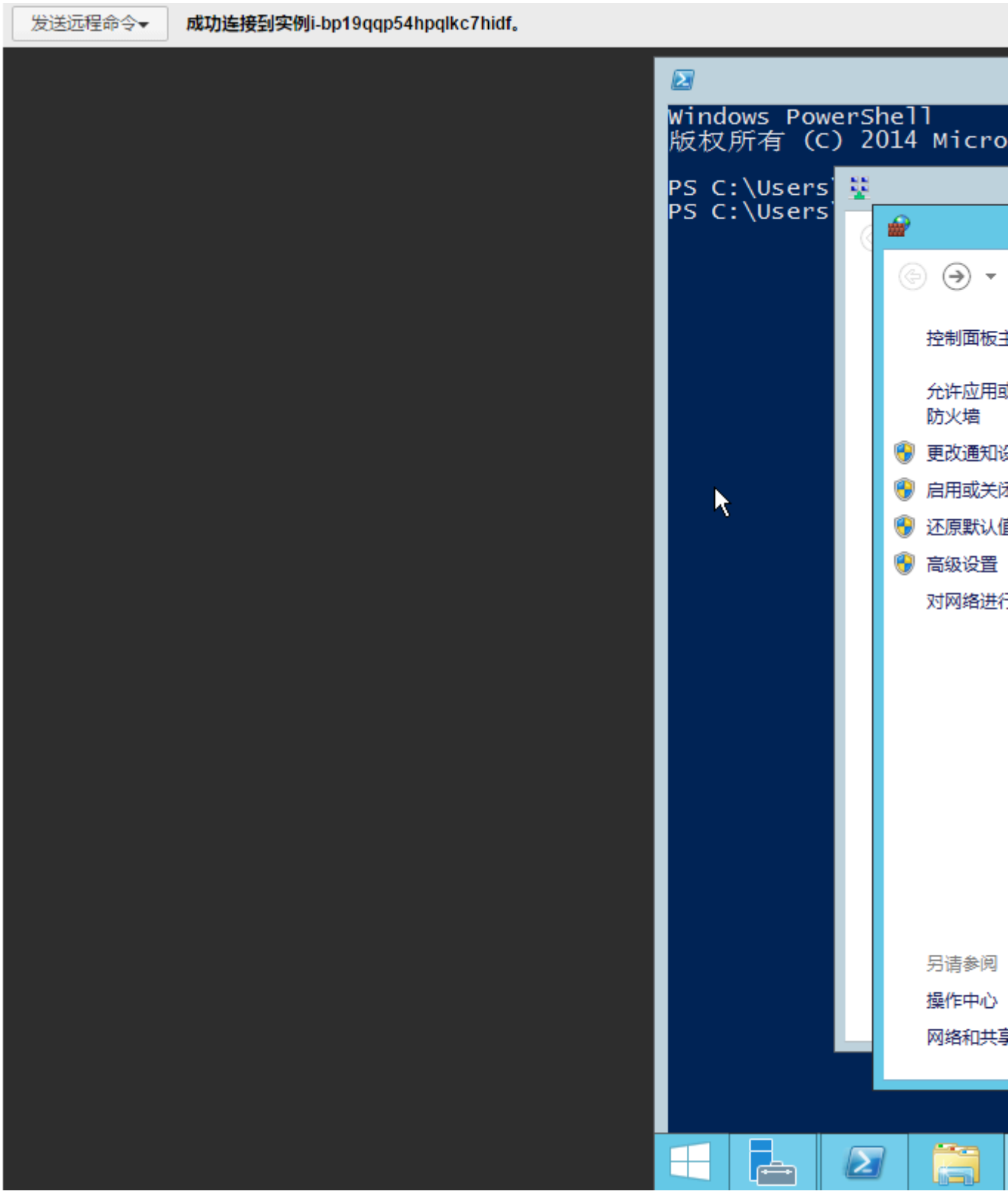
（1）服务器IP地址会从DHCP变成固定IP地址，这里你可以重新改成DHCP，我前面说过，如果想修改ECS 的地址最好从控制台操作。



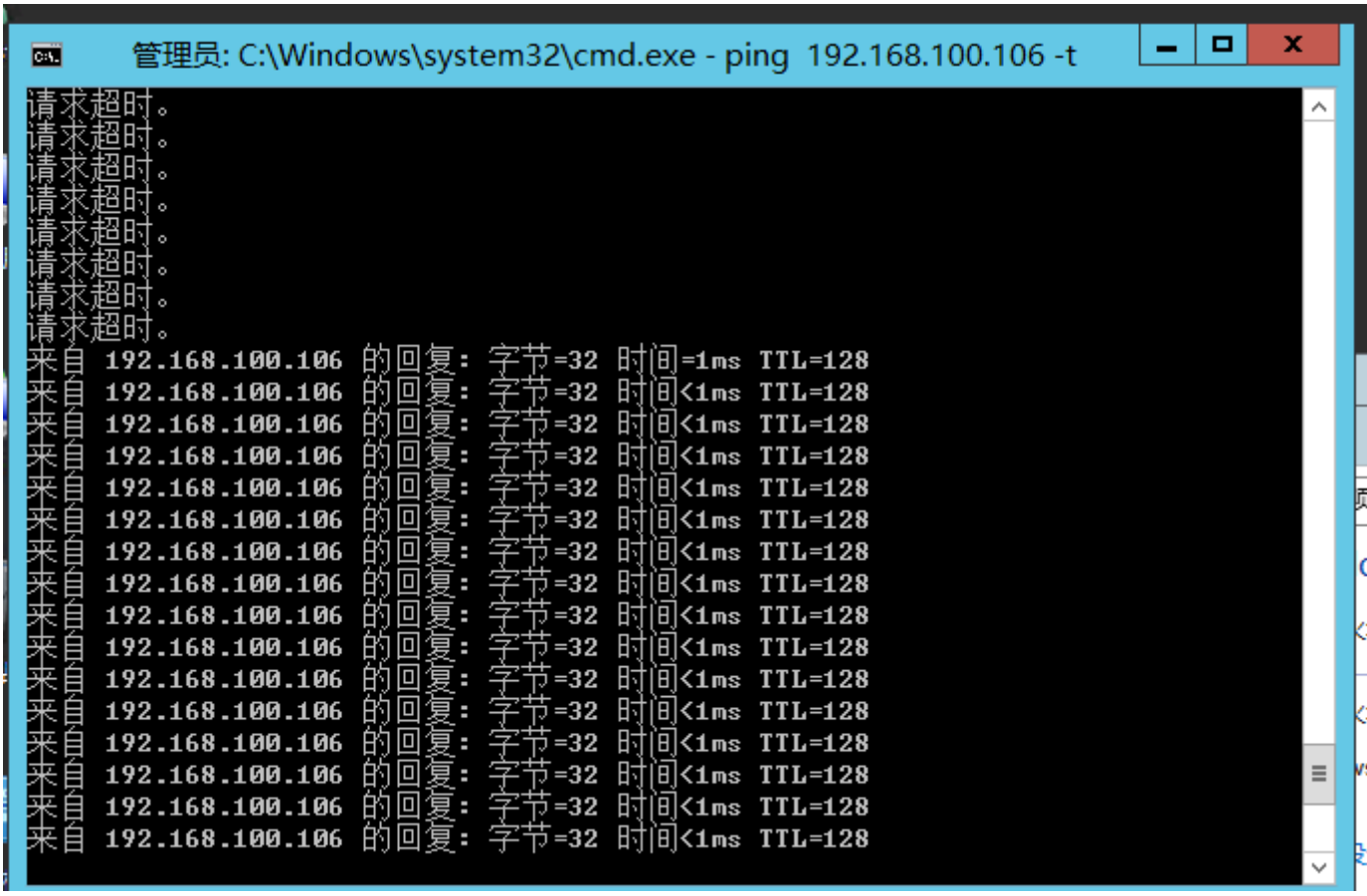
（2）服务器无法PING 通，这是因为服务器SID初始化完成后，也将服务器防火墙的配置修改成微软默认的配置，也就是将"来宾或公用网络"打开，导致无法ping 通服务器和远程。这个时候我们就需要在web console 界面将防火墙"来宾或公用网络"关闭，或者放行需要开放的端口。


发送远程命令
成功连接到实例i-bp19qqp54hpqlkc7hidf。
Windows PowerShell
版权所有 (C) 2014 Microsc
PS C:\Users\Administrator
用户信息
用户名
iz4hpqlkc7hidfz\administr
PS C:\Users\Administrator
我这里使用VNC终端链接的，这里可以看到SID已经发生改变，
当我使用MSTSC工具链接的是时候发现并不能远程链接


管理员: C:\Windows\system32\cmd.exe - ping 192.168.100.106 -t
请求超时。
Sysprep


发送远程命令
成功连接到实例i-bp19qqp54hpqlkc7hidf。
Windows PowerShell
版权所有 (C) 2014 Micro
PS C:\Users
PS C:\Users
控制面板主
允许应用或
防火墙
更改通知设
启用或关闭
还原默认值
高级设置
对网络进行
另请参阅
操作中心
网络和共享


Windows 防火墙
控制面板 ▸ 所有控制面板项 ▸ Windows 防火墙
控制面板主页
允许应用或功能通过 Windows 防火墙
更改通知设置
启用或关闭 Windows 防火墙
还原默认值
高级设置
对网络进行疑难解答
使用 Windows 防火墙来帮助保护你的电脑
Windows 防火墙有助于防止黑客或恶意软件通过 Internet 或网络访问你的电脑。
更新防火墙设置
Windows 防火墙未使用推荐的设置来保护计算机。
推荐的设置有哪些?
域网络(M)
工作区中连接到域的网络
Windows 防火墙状态:
关闭
传入连接:
阻止所有与未在允许应用列表中的
活动的域网络:
lyonz.com
通知状态:
Windows 防火墙阻止新应用时不
专用网络(R)
来宾或公用网络(P)
另请参阅

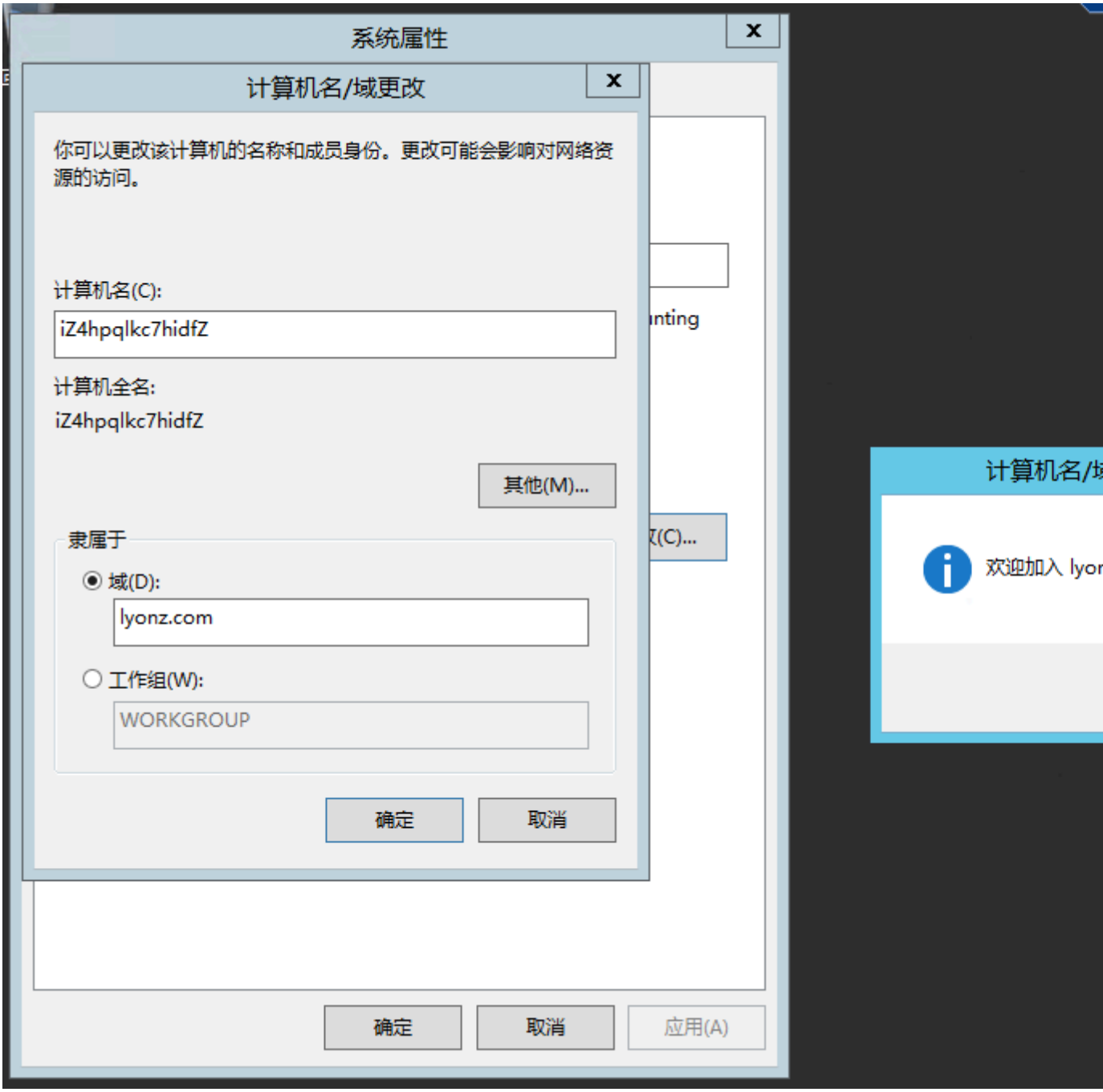


修改客户端的基本信息

DNS 指向 DC 的IP地址，您可以根据业务需求修改主机名。


Internet 协议版本 4 (TCP/IPv4) 属性
常规
备用配置
如果网络支持此功能，则可以获取自动指派的 IP 设置。否则，你需要从网络系统管理员处获得适当的 IP 设置。
自动获得 IP 地址(O)
使用下面的 IP 地址(S):
IP 地址(I):
子网掩码(U):
默认网关(D):
192 . 168 . 100 . 253
自动获得 DNS 服务器地址(B)
使用下面的 DNS 服务器地址(E):
首选 DNS 服务器(P):
192 . 168 . 100 . 105
备用 DNS 服务器(A):
退出时验证设置(L)
高级(V)...


版权所有 (C) 2014 Microsoft Corporation。保留所有权利。
PS C:\Users\Administrator> firewall.cpl
PS C:\Users\Administrator> nslookup
DNS request timed out.
timeout was 2 seconds.
默认服务器: UnKnown
Address: 192.168.100.105
> lyonz.com
服务器: UnKnown
Address: 192.168.100.105
名称: lyonz.com
Address: 192.168.100.105
> exit
PS C:\Users\Administrator> ping lyonz.com
正在 Ping lyonz.com [192.168.100.105] 具有 32 字节的数据:
来自 192.168.100.105 的回复: 字节=32 时间<1ms TTL=128
来自 192.168.100.105 的回复: 字节=32 时间<1ms TTL=128
192.168.100.105 的 Ping 统计信息:
数据包: 已发送 = 2, 已接收 = 2, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
最短 = 0ms, 最长 = 0ms, 平均 = 0ms
Control-C
PS C:\Users\Administrator>
PS C:\Users\Administrator>



以上就是阿里云ECS Windows Server 2012 搭建域以及客户端加入域的过程，如果您已经在线下或者虚拟机搭建了 AD 域，在阿里云上搭建 AD 域时需要注意修改客户端SID。

相关链接

- 域控常见问题配置
- 更多开源软件尽在云市场

4.4 设置Windows操作系统首选语言

本文使用公共镜像中的Windows Server 2016英语版操作系统为例，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。

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。本文为德语为示范步骤，适用于Windows Server 2012及其以上的版本操作系统。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。

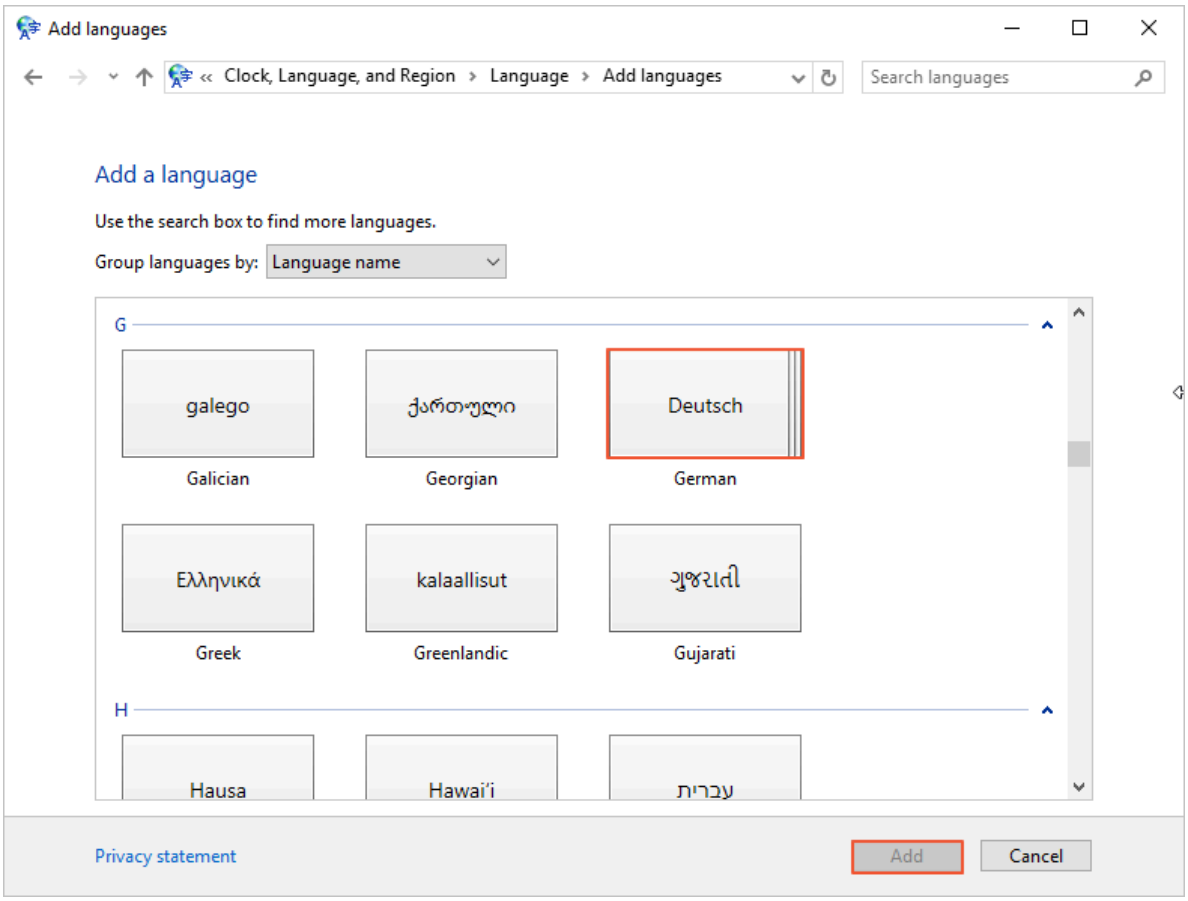
操作步骤

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。

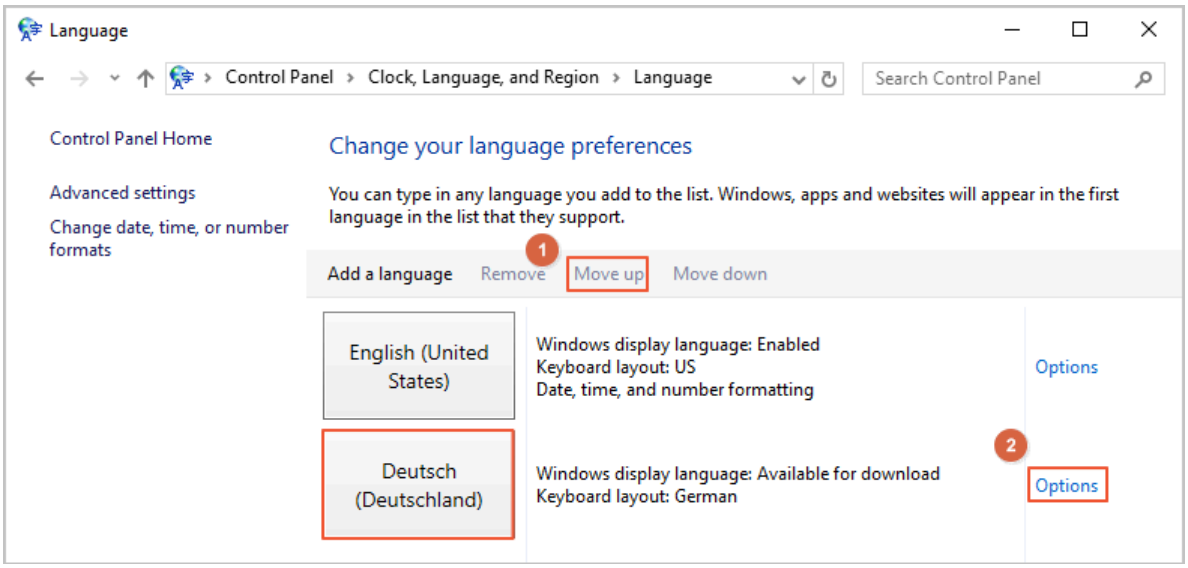
```
Set-ItemProperty -Path 'HKLM:\SOFTWARE\Policies\Microsoft\Windows\
WindowsUpdate\AU' -Name UseWUServer -Value 0
Restart-Service -Name wuauserv
```

4. 找到控制面板，单击Clock, Language, and Region > Language > Add a language。

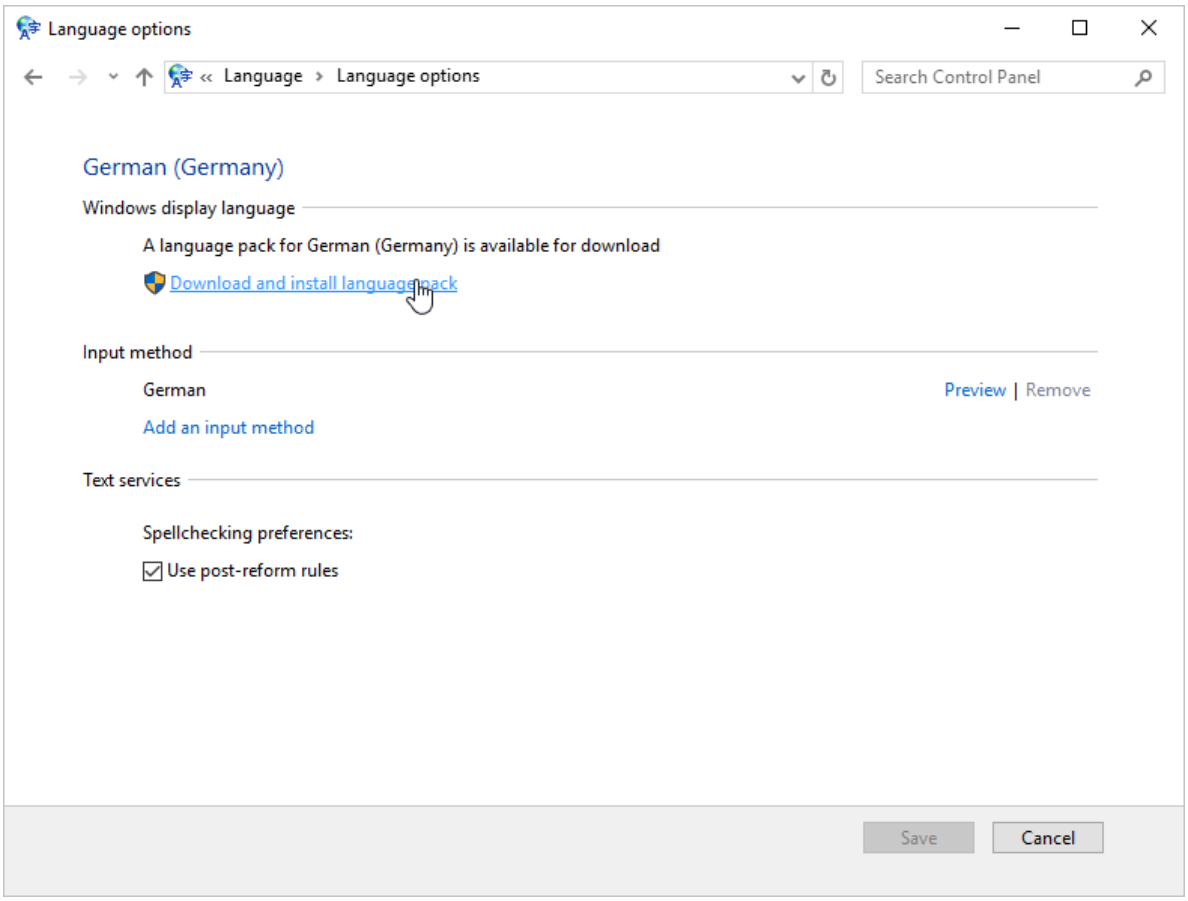
5. 在Add languages对话框中，选择一种语言，例如Deutsch (German) > Deutsch (Deutschland)，单击Add。



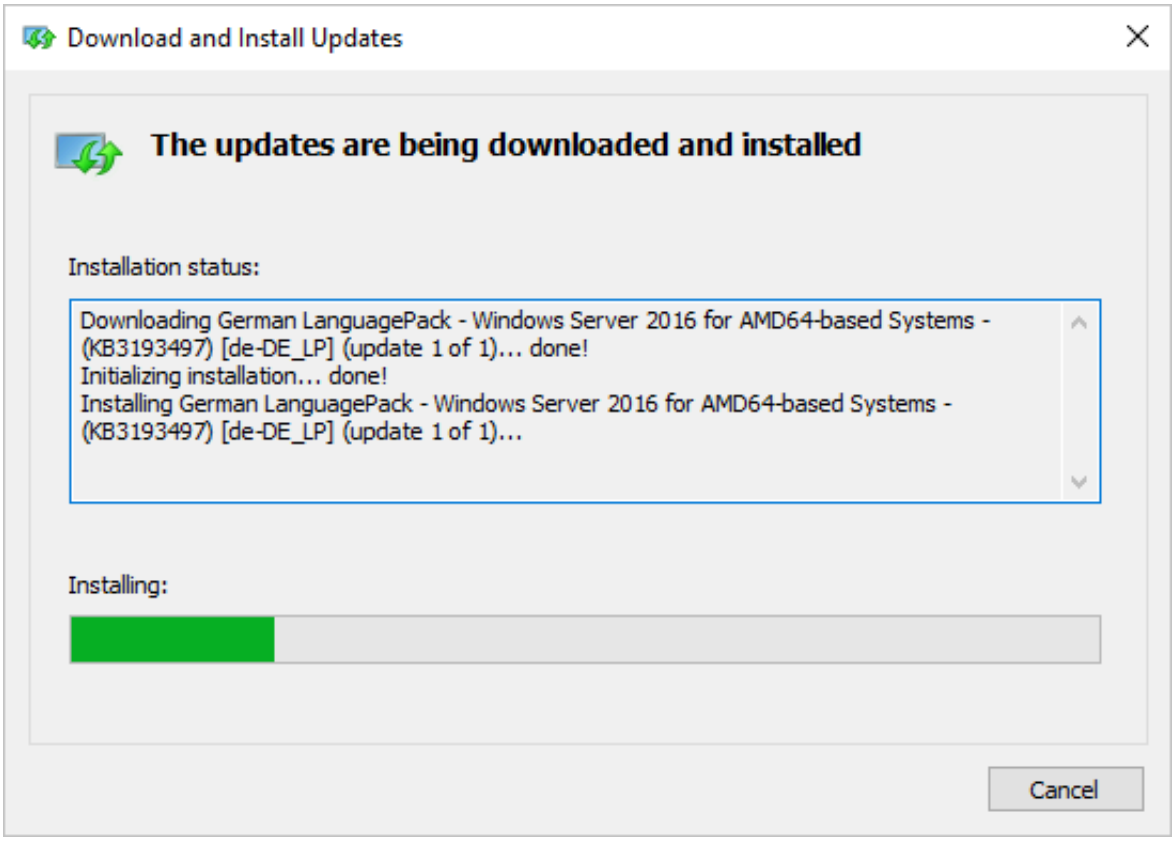
6. 选择语言，例如Deutsch (Deutschland)，单击Move up更改语言优先级。
7. 单击所选语言右侧的Options，在线检查语言更新。



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



9. 等待安装完成。



10.在ECS控制台重新启动实例。

11.再次连接Windows实例。

显示语言会在重启登录后更改为德语。

12.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' -Name UseWUServer -Value 1
Restart-Service -Name wuauserv
```

13.打开Windows Update，检查安全更新，重新安装配置语言设置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。
3. 通过自定义镜像创建指定数量的实例。

5 Packer实践之镜像即代码

5.1 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。

什么是Packer

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存在不一致。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。

镜像构建方式对比

目前，阿里云支持以下构建镜像的方式：

表 5-1: 构建镜像的方式对比

构建方式列举	工具与依赖	优点	缺点
使用快照创建自定义镜像	支持通过ECS控制台或者API创建，前提是您必须已经创建了一份系统盘快照。	· 易上手。 · 可多次利用计算资源。 · 可以根据实时生产环境创建镜像。 · 基于阿里云公共镜像创建，安全可靠。	· 随镜像内需预装的软件及其配置扩充变得复杂。 · 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。
使用实例创建自定义镜像	支持通过ECS控制台或者API创建，前提是您已经创建了一台实例。		

构建方式列举	工具与依赖	优点	缺点
使用Packer构建自定义镜像	支持识别Packer请求，通过AccessKey验证用户信息。	· 无需提前创建实例或快照。 · 可多次利用和修改JSON文件。 · 具有操作日志，步骤清晰可见易于回溯。 · 自动释放临时资源。 · 支持自动转换ISO文件并导入阿里云ECS。 · 可基于阿里云公共镜像和本地ISO文件构建，灵活方便。	具有少量学习成本。

操作条件

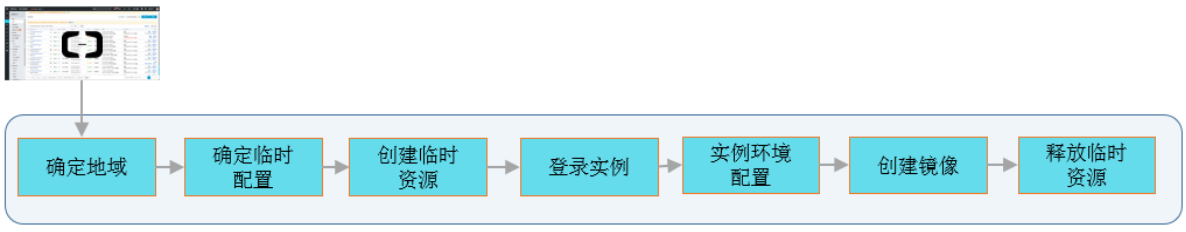
本文通过比较“使用实例创建自定义镜像”和“使用Packer构建自定义镜像”的操作流程，突出Packer在DevOps场景中的优势。以下为本次操作的假设场景和一致性条件：

- 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

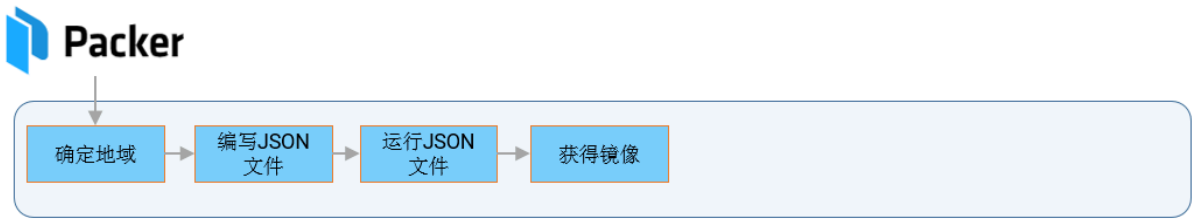
本示例介绍如何通过ECS管理控制台创建一份自定义镜像。以下为示例操作的流程示意图：



1. 登录云服务器ECS管理控制台。
2. 在左侧导航栏，单击实例。
3. 选择地域。
4. 参见使用向导创建实例完成实例购买。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，请参见实例规格族汇总。
   - 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。
5. 远程连接已创建的ECS实例。连接方式可参见连接方式导航。
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. 参见使用实例创建自定义镜像创建一份镜像。
9. 在快照和镜像 > 镜像页面查看镜像完成状态。
10. （可选）镜像制作成功后，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

本示例介绍如何通过Packer构建一份自定义镜像。以下为示例操作的流程示意图：



前提条件

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。

操作步骤

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"
    ]
  }]
}
```

表 5-2: Packer参数解释

参数	示例值	参数解释
variables{"varible1":"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。

表 5-3: 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AccessKeyID。更多详情，请参见创建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。	高
secret_key	String	CM1ycKrrCekQ0dhXXXXXXXXXl7yavUT	您的AccessKeySecret。	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，请参见实例规格族汇总。	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_201
70818.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_/
nv2r4drx3bs08l6tcx06ndb40000gn/T/packer-shell260049331
    alicloud-ecs: Loaded plugins: fastestmirror
    alicloud-ecs: Determining fastest mirrors
    alicloud-ecs: Resolving Dependencies
    alicloud-ecs: --> Running transaction check
    alicloud-ecs: ---> Package redis.x86_64 0:3.2.12-2.el7 will be
installed
    alicloud-ecs: --> Processing Dependency: libjemalloc.so.1()(
64bit) for package: redis-3.2.12-2.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
          Repository     Size
    alicloud-ecs:
 ================================================================================
    alicloud-ecs: Installing:
    alicloud-ecs:  redis             x86_64          3.2.12-2.el7
          epel          544 k
    alicloud-ecs: Installing for dependencies:
    alicloud-ecs:  jemalloc          x86_64          3.6.0-1.el7
          epel          105 k
    alicloud-ecs:
```

```
    alicloud-ecs: Transaction Summary
    alicloud-ecs:
 ================================================================================
    alicloud-ecs: Install  1 Package (+1 Dependent package)
    alicloud-ecs:
    alicloud-ecs: Total download size: 648 k
    alicloud-ecs: Installed size: 1.7 M
    alicloud-ecs: Downloading packages:
    alicloud-ecs:
 --------------------------------------------------------------------------------
    alicloud-ecs: Total
  2.2 MB/s | 648 kB  00:00
    alicloud-ecs: Running transaction check
    alicloud-ecs: Running transaction test
    alicloud-ecs: Transaction test succeeded
    alicloud-ecs: Running transaction
    alicloud-ecs:   Installing : jemalloc-3.6.0-1.el7.x86_64
                          1/2
    alicloud-ecs:   Installing : redis-3.2.12-2.el7.x86_64
                          2/2
    alicloud-ecs:   Verifying  : redis-3.2.12-2.el7.x86_64
                          1/2
    alicloud-ecs:   Verifying  : jemalloc-3.6.0-1.el7.x86_64
                          2/2
    alicloud-ecs:
    alicloud-ecs: Installed:
    alicloud-ecs:   redis.x86_64 0:3.2.12-2.el7
    alicloud-ecs:
    alicloud-ecs: Dependency Installed:
    alicloud-ecs:   jemalloc.x86_64 0:3.6.0-1.el7
    alicloud-ecs:
    alicloud-ecs: Complete!
==> alicloud-ecs: Stopping instance: i-xxx
==> alicloud-ecs: Waiting instance stopped: i-xxx
==> alicloud-ecs: Creating image: packer_basic
    alicloud-ecs: Detach keypair packer_xxx from instance: i-xxx
==> alicloud-ecs: Cleaning up 'EIP'
==> alicloud-ecs: Cleaning up 'instance'
==> alicloud-ecs: Cleaning up 'security group'
==> alicloud-ecs: Cleaning up 'vSwitch'
==> alicloud-ecs: Cleaning up 'VPC'
==> alicloud-ecs: Deleting temporary keypair...
Build 'alicloud-ecs' finished.

==> Builds finished. The artifacts of successful builds are:
--> alicloud-ecs: Alicloud images were created:

cn-beijing: m-xxx
```

相关链接

更多参数和样例，请参见：

- 《Packer官方文档》Alicloud Image Builder和Examples
- Packer的DevOps配置

5.2 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。本文推荐Alibaba Cloud DevOps tutorials系列教程，其中涉及Terraform和Packer的内容参见Continuous Delivery一节。
- 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的使用场景示例可参见修改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"communicator": "winrm"、"winrm_port": 5985、"winrm_username": "Administrator"和"winrm_password": "Test1234"分别表示通过WinRM连接实例、通信端口为5985、连接时使用Administrator账户、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，可以参见使用Packer创建并导入本地镜像。文档中包含两个重要的部分：

  1. 您需要使用本地虚拟化环境或软件对应的Builder，如Qemu Builder。
  2. 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。流程请参见导入镜像必读。

相关链接

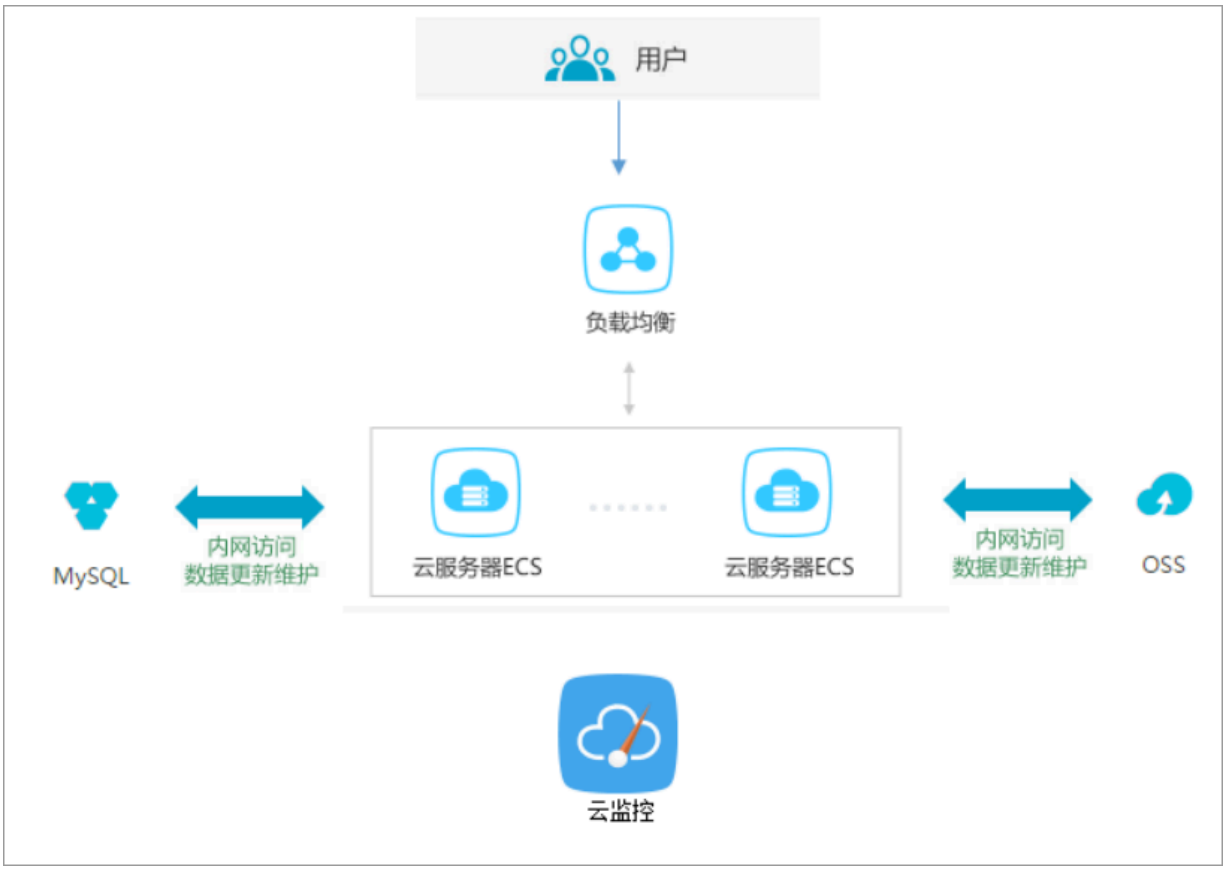
更多参数和样例，请参见Packer官方文档Alicloud Image Builder和Examples。

6 监控

6.1 使用云监控监控ECS实例

合理的监控设置能极大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。

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，使用了ECS、RDS、OSS和负载均衡。



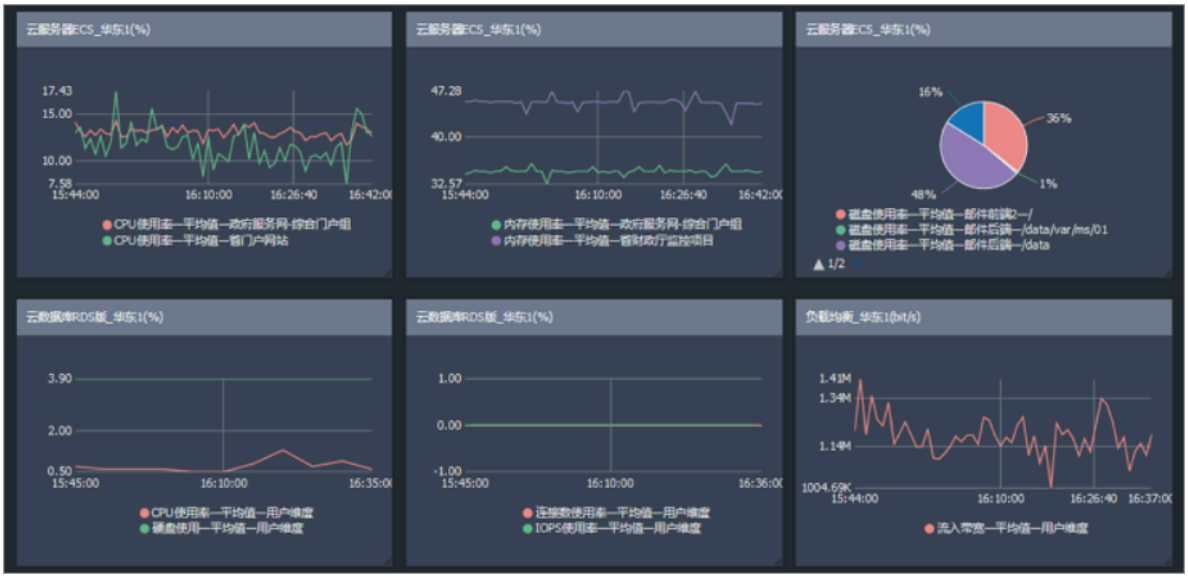
前提条件

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。如果安装失败需要手动安装，请参考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，具体可参考 云服务资源使用概览和报警概览。

- 利用云监控的Dashboard功能，给您业务系统的云资源设置一个全局监控总览，可随时检查整个业务系统资源的健康状态。

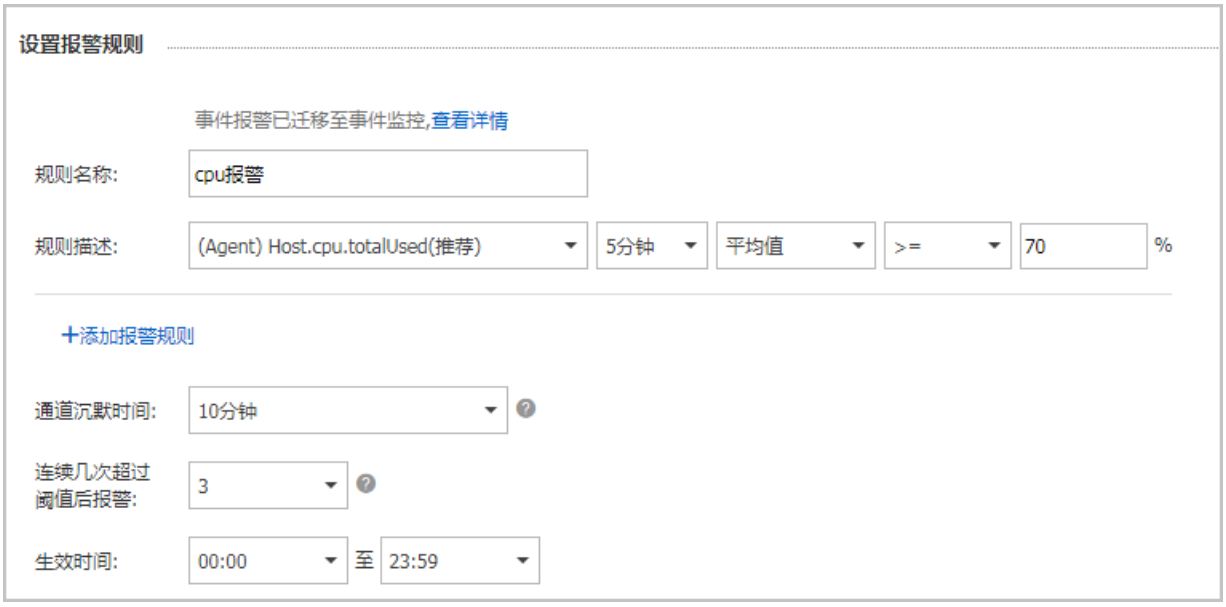
  为了更好地监控大屏展示效果，这里将ECS的CPU、内存、磁盘的使用率单独分组展示；将RDS的四项指标分两组展示。



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。

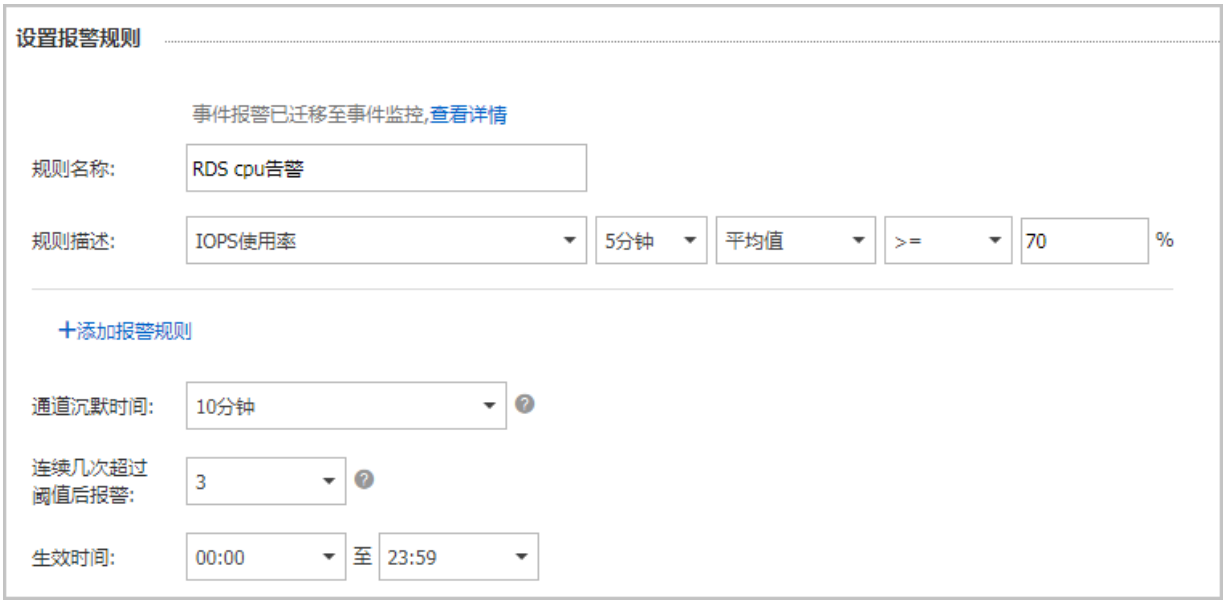
以CPU使用率为例，因为需要给服务器预留部分处理性能保障服务器正常运行，所以建议您将CPU告警阈值设置为70%，连续三次超过阈值后开始报警。



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，继续设置内存或磁盘的报警规则和报警通知人。示例：

设置RDS监控

建议将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。

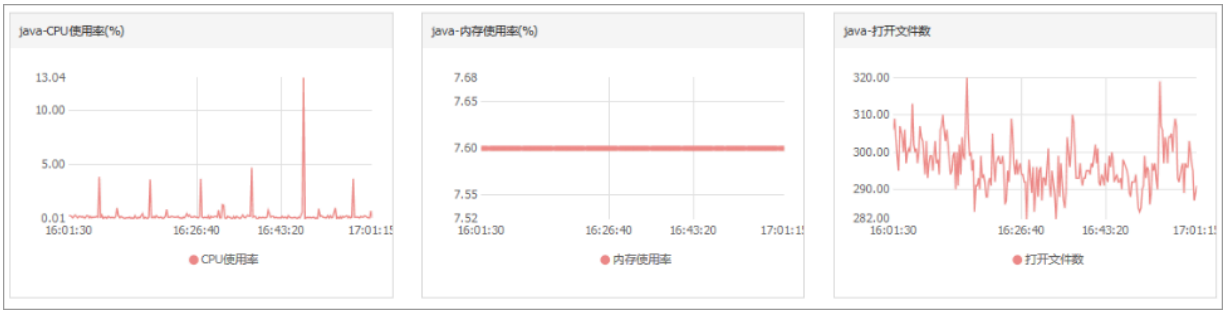


设置负载均衡监控

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，将负载均衡带宽值的70%作为告警阈值，如下图所示。

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，下图是Java进程的相关监控示例。具体操作请参考 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。

7 借助于实例RAM角色访问其他云产品

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。

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。

> 说明:
> 为了方便您随本文样例快速入门，文档里所有操作均在OpenAPI Explorer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：

步骤 1. 创建 RAM 角色并配置授权策略

步骤 2. 指定 RAM 角色创建并设置 ECS 实例

步骤 3. 在实例内部访问实例元数据 URL 获取 STS 临时凭证

步骤 4. 基于临时凭证，使用 Python SDK 访问 OSS

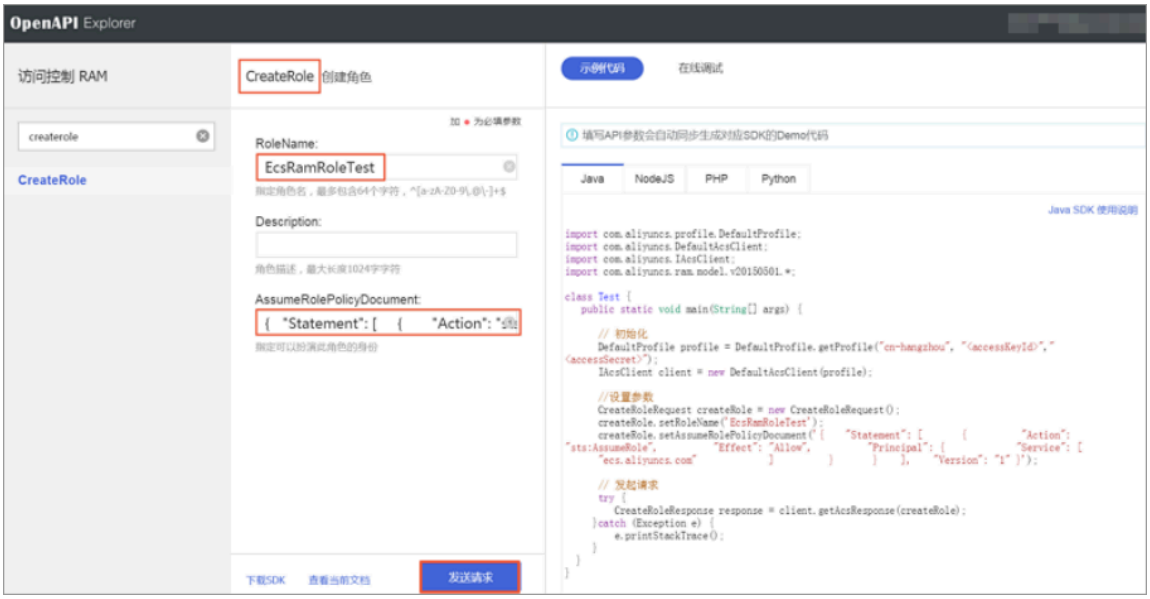
步骤 1. 创建 RAM 角色并配置授权策略

按以下步骤创建 RAM 角色并配置授权策略。

1. 创建 RAM 角色。找到 OpenAPI Explorer 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"
}
```

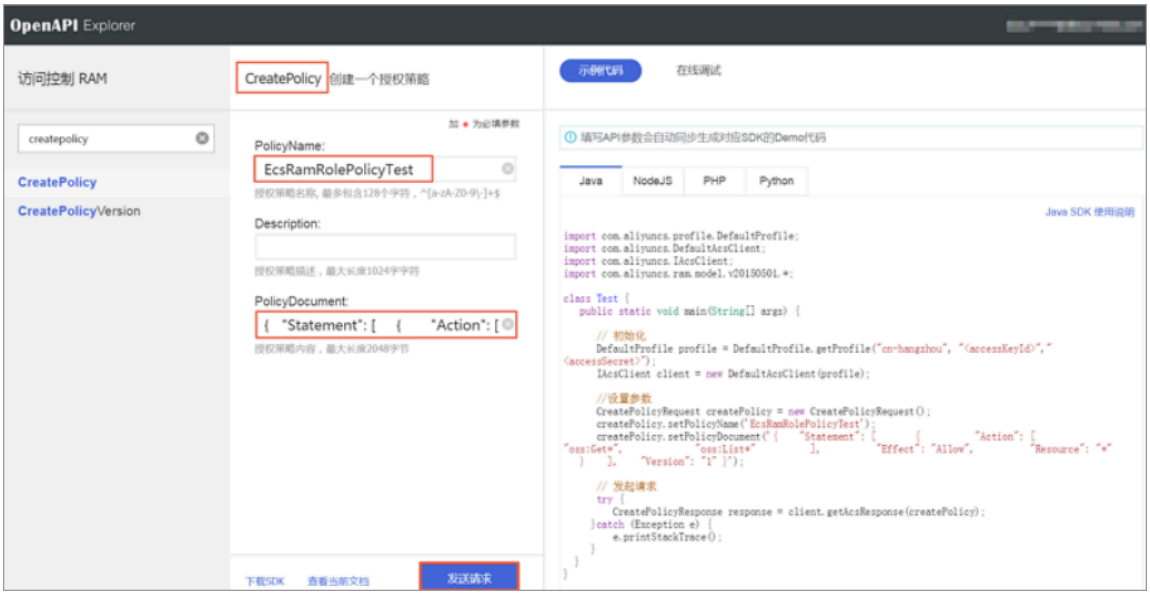


2. 创建授权策略。找到 OpenAPI Explorer RAM 产品下的 CreatePolicy API。其中：

   - 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。

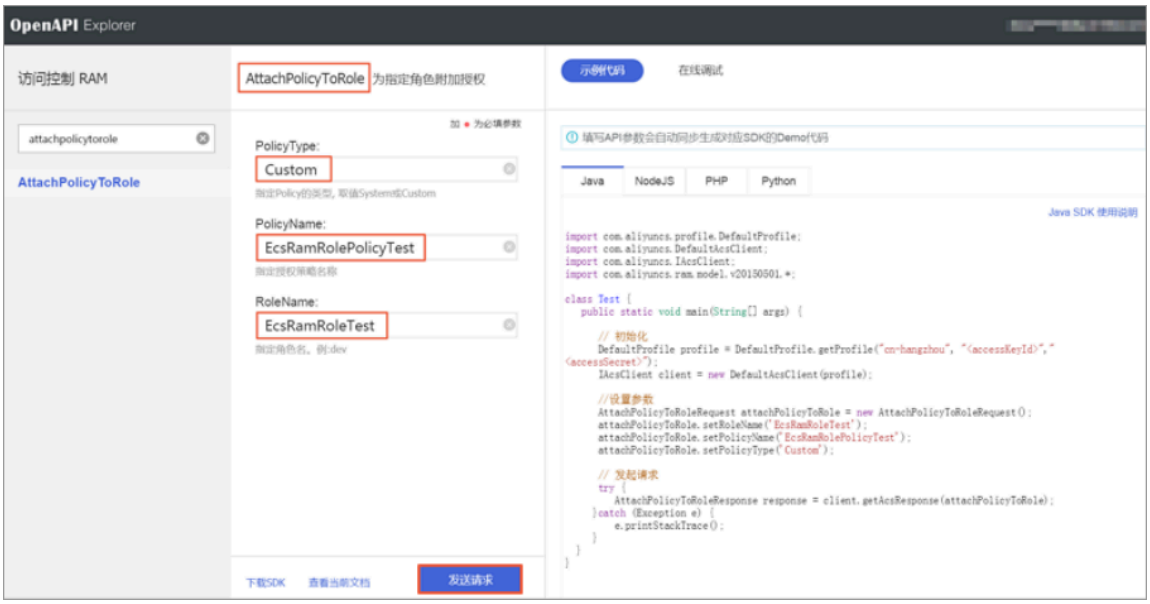
```
{
"Statement": [
{
"Action": [
  "oss:Get*",
  "oss:List*"
],
"Effect": "Allow",
"Resource": "*"
```

```
}
],
"Version": "1"
}
```



3. 为角色附加授权。找到 OpenAPI Explorer RAM 产品下 AttachPolicyToRole API。其中：

   - 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 2. 为 ECS 实例指定 RAM 角色

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 角色附加到一个已有的 VPC 类型ECS实例上**

- **指定 RAM 角色创建并设置 ECS 实例**

将实例 RAM 角色附加到一个已有的 VPC 类型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：

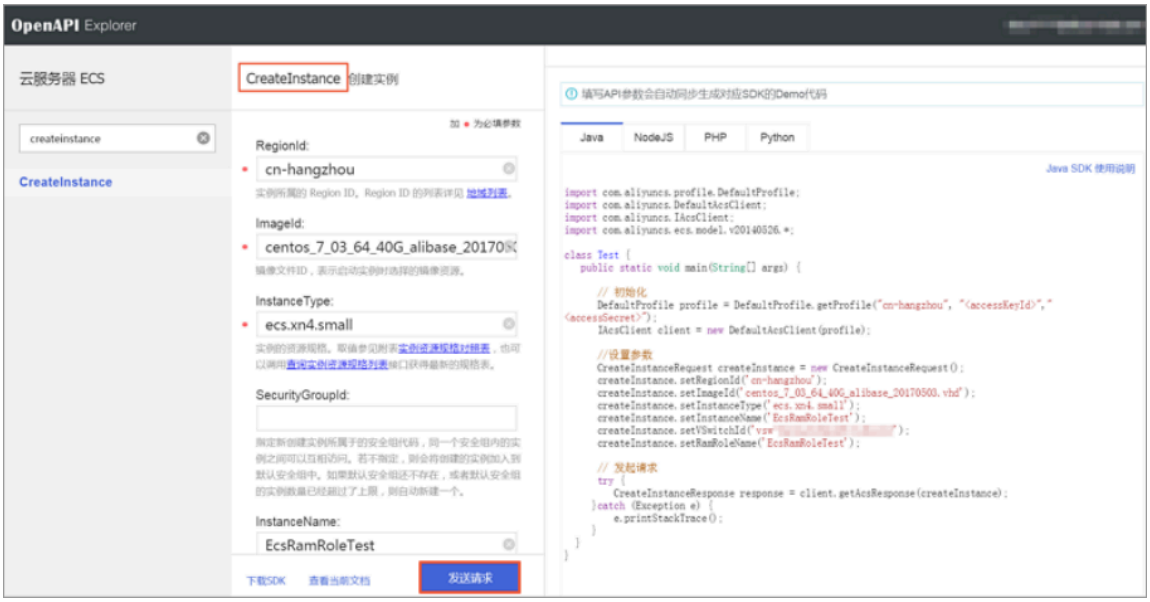
- `RegionId`：**为实例所在的地域 ID。**
- `RamRoleName`：**RAM 角色的名称。本示例中为** `EcsRamRoleTest`。
- `InstanceIds`：**需要附加实例 RAM 角色的 VPC 类型 ECS 实例 ID。本示例中为 ["i-bXXXXXXXX"]。**

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。

1. 创建实例。找到 OpenAPI Explorer 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.xn4.small。
   - 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:*`。

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

```
}
```

2. 设置密码并启动实例。
3. 使用 API 或在控制台设置 ECS 实例能访问公网。

步骤 3. 在实例内部访问实例元数据 URL 获取 STS 临时凭证

按以下步骤获取实例的 STS 临时凭证。

说明:

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. 远程连接实例。
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，您应替换为自己的创建的 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" : "9PjfXXXXXXXXBf2XAW",
"Expiration" : "2017-06-09T09:17:19Z",
"SecurityToken" : "CAIXXXXXXXXXXwmBkleCTkyI+",
"LastUpdated" : "2017-06-09T03:17:18Z",
"Code" : "Success"
}cess"
}
```

步骤 4. 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`。

操作步骤

按以下步骤使用 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，其中：

   - `oss2.StsAuth` 中的 3 个参数分别对应于上述 URL 返回的 AccessKeyId、AccessKeySecret 和 SecurityToken。
   - `oss2.Bucket` 中后 2 个参数是 Bucket 的名称和 Endpoint。

   ```
   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).key)
   ```

   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 informatio
   n.
   >>> import oss2
   >>> from itertools import islice
   >>> auth = oss2.StsAuth("STS.J8XXXXXXXXXX4", "9PjfXXXXXXXXXBf2XAW",
    "CAIXXXXXXXXXXwmBkleCTkyI+")
   >>> 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
   ```

8 GPU实例最佳实践

8.1 在gn5实例上部署NGC环境

本文以搭建TensorFlow深度学习框架为例详细介绍如何在gn5实例上搭建NGC环境。

前提条件

在开始搭建TensorFlow环境之前，您必须先完成以下工作：

- 注册阿里云账号，并完成实名认证。具体步骤，请参见注册阿里云账号和实名认证 。
- 登录NGC网站，注册NGC账号。
- 登录NGC网站，获取NGC API key并保存到本地。登录NGC容器环境时需要验证您的NGC API Key。

背景信息

NGC（NVIDIA GPU CLOUD）是NVIDIA开发的一套深度学习生态系统，可以使开发者免费访问深度学习软件堆栈，建立适合深度学习的开发环境。

目前NGC在阿里云gn5实例作了全面部署，并且在镜像市场提供了针对NVIDIA Pascal GPU优化的NGC容器镜像。通过部署镜像市场的NGC容器镜像，开发者能简单快速地搭建NGC容器环境，即时访问优化后的深度学习框架，大大缩减产品开发以及业务部署的时间，实现开发环境的预安装；同时支持调优后的算法框架，并且保持持续更新。

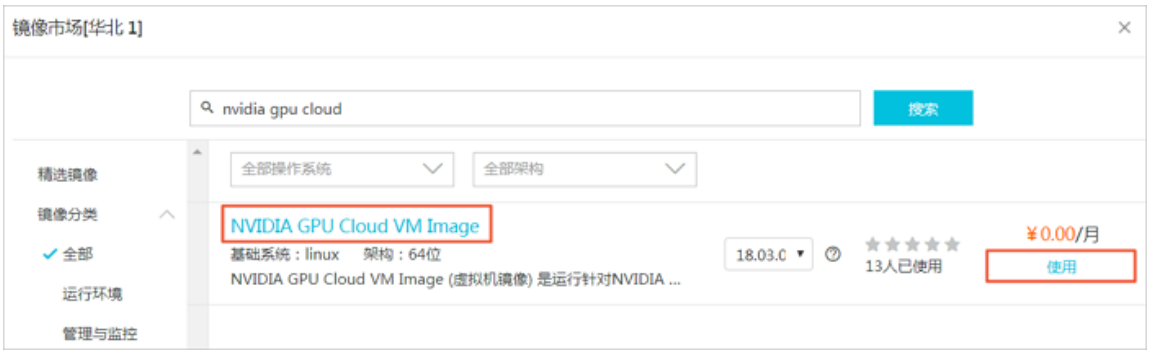
NGC网站提供了目前主流深度学习框架不同版本的镜像（比如Caffe、Caffe2、CNTK、MxNet、TensorFlow、Theano、Torch），您可以选择需要的镜像搭建环境。

操作步骤

1. 创建一台gn5实例。具体操作，请参见创建ECS实例。

   在配置参数时，您需要注意以下几点：

   - 地域：只能选择华北1（青岛）、华北2（北京）、华北3（张家口）、华北5（呼和浩特）、华东1（杭州）、华东2（上海）、华南1（深圳）。
   - 实例：选择gn5实例规格。
   - 镜像：单击镜像市场，在弹出对话框里，找到NVIDIA GPU Cloud VM Image后，单击使用。



   - 公网带宽：选择分配公网IP地址。

     说明：
     如果这里没有分配公网IP地址，则在实例创建成功后，需要绑定EIP地址。

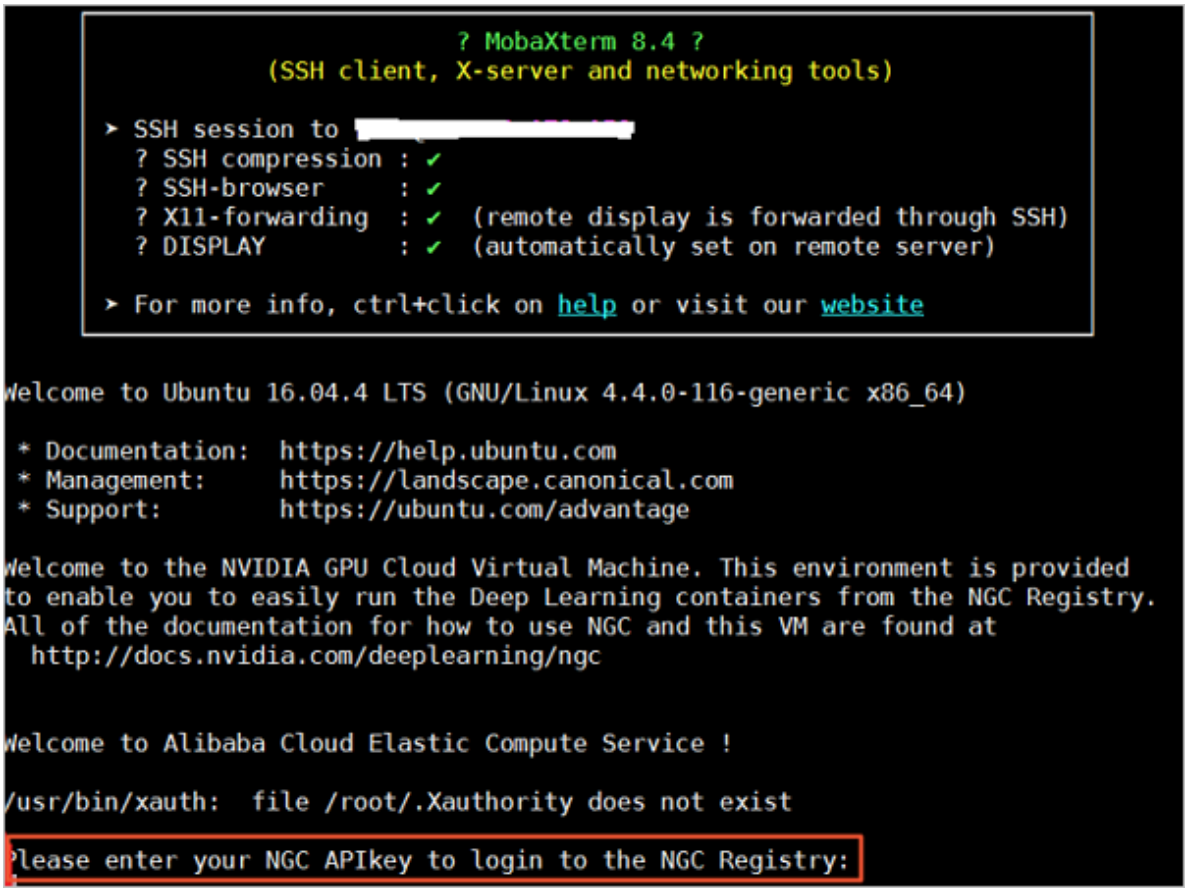
   - 安全组：选择一个安全组。安全组里必须开放 TCP 22 端口。如果您的实例需要支持HTTPS或 DIGIT 6 服务，必须开放TCP 443（用于HTTPS）或TCP 5000（用于DIGITS 6）端口。

2. 连接ECS实例。

   根据创建实例时选择的登录凭证选择以下任一方式连接ECS实例：

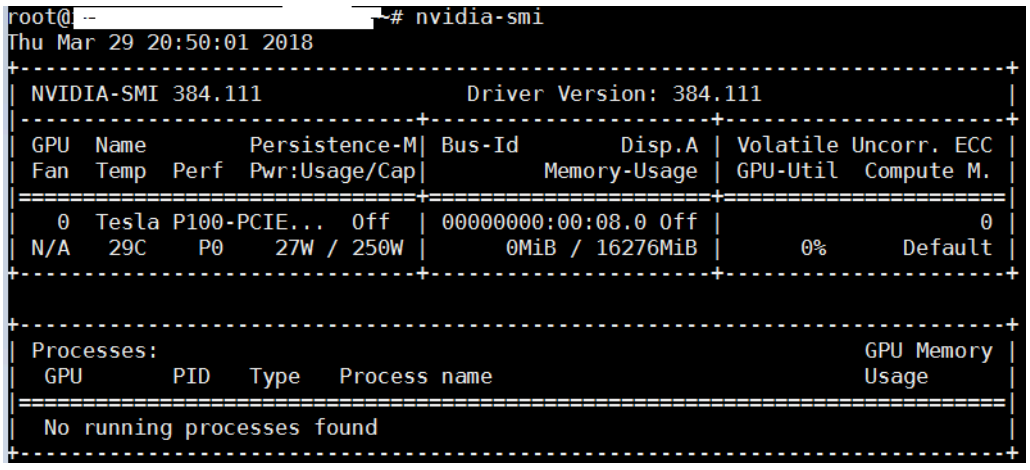
   - 使用密码验证连接ECS实例
   - 使用SSH密钥对验证连接ECS实例

3. 按界面提示输入NGC官网获取的NGC API Key后按回车键，即可登录NGC容器环境。



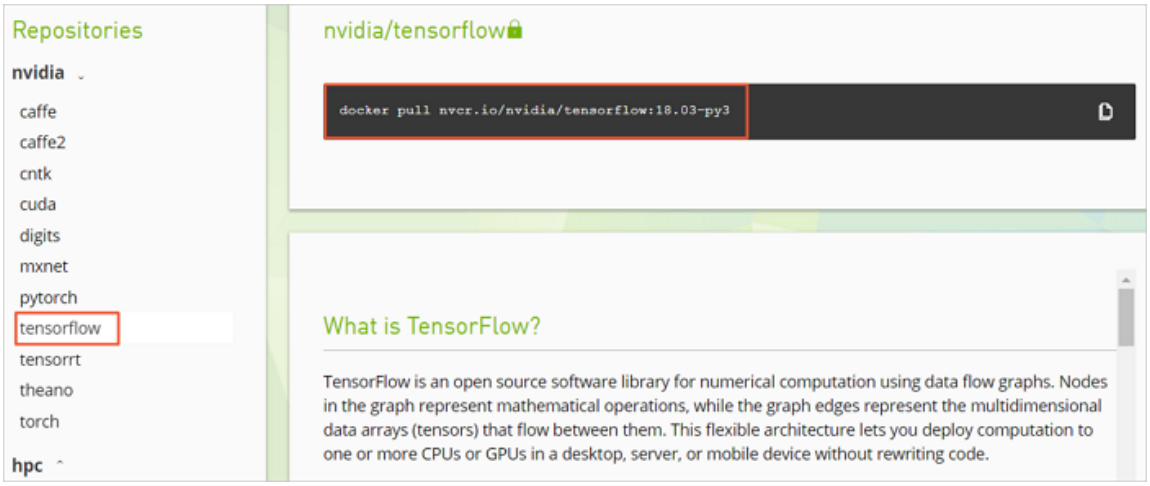
4. 运行`nvidia-smi`命令。

   您能查看当前GPU的信息，包括GPU型号、驱动版本等，如下图所示。



5. 按以下步骤搭建TensorFlow环境。

   a) 登录NGC网站，在TensorFlow镜像页面，获取`docker pull`命令。



   b) 下载TensorFlow镜像。

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

   c) 查看下载的镜像。

```
docker image ls
```

   d) 运行容器，完成TensorFlow开发环境的部署。

```
nvidia-docker run --rm -it nvcr.io/nvidia/tensorflow:18.03-py3
```

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

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

NVIDIA Release 18.03 (build 349854)

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

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

6. 选择以下任一种方式测试TensorFlow。

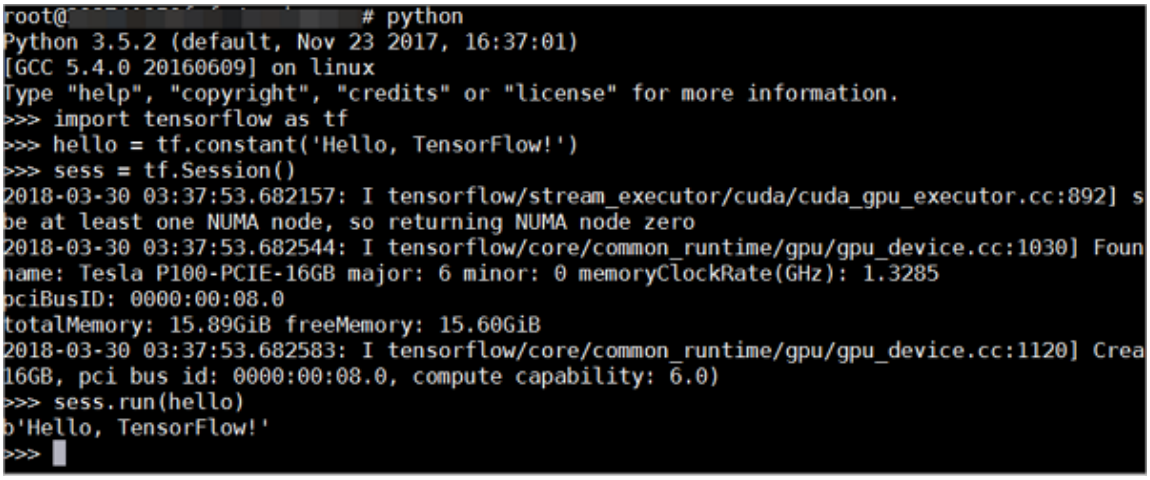
   · 简单测试TensorFlow。

```
$python
```

```
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
```

```
>>> sess.run(hello)
```

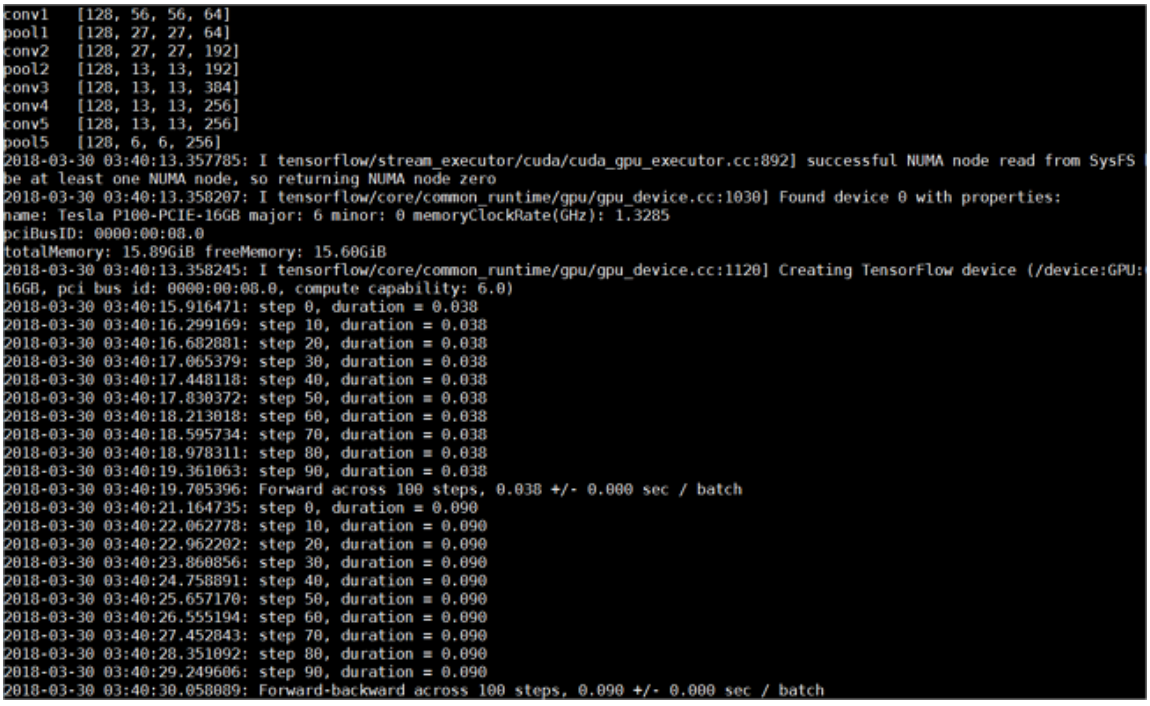
如果TensorFlow正确加载了GPU设备，返回结果如下图所示。



- 下载TensorFlow模型并测试TensorFlow。

```
git clone https://github.com/tensorflow/models.git
cd models/tutorials/image/alexnet
python alexnet_benchmark.py --batch_size 128 --num_batches 100
```

运行状态如下图所示。



7. 保存TensorFlow镜像的修改。否则，下次登录时配置会丢失。

8.2 在GPU实例上使用RAPIDS加速机器学习任务

本文介绍了如何在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。

背景信息

RAPIDS，全称Real-time Acceleration Platform for Integrated Data Science，是NVIDIA针对数据科学和机器学习推出的GPU加速库。更多RAPIDS信息请参见官方网站。

NGC，全称NVIDIA GPU CLOUD，是NVIDIA推出的一套深度学习生态系统，供开发者免费访问深度学习和机器学习软件堆栈，快速搭建相应的开发环境。NGC网站提供了RAPIDS的Docker镜像，预装了相关的开发环境。

JupyterLab是一套交互式的开发环境，帮助您高效地浏览、编辑和执行服务器上的代码文件。

Dask是一款轻量级大数据框架，可以提升并行计算效率。

本文提供了一套基于NVIDIA的RAPIDS Demo代码及数据集修改的示例代码，演示了在GPU实例上使用RAPIDS加速一个从ETL到ML Training端到端任务的过程。其中，ETL时使用RAPIDS的cuDF，ML Training时使用XGBoost。本文示例代码基于轻量级大数据框架Dask运行，为一套单机运行的代码。

> 说明:
> NVIDIA官方RAPIDS Demo代码请参见Mortgage Demo。

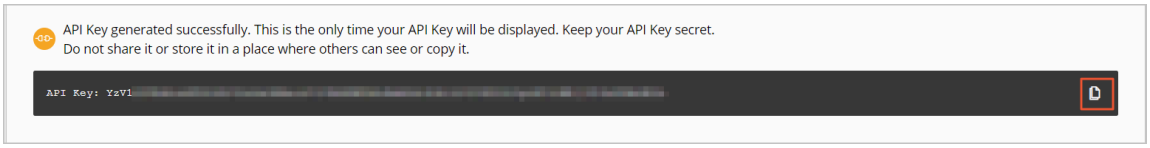
前提条件

- 注册阿里云账号并完成实名认证，请参见阿里云账号注册流程和个人实名认证 。
- 在NGC注册页面注册NGC账号。
- 获取NGC API Key。
  1. 登录NGC网站。
  2. 前往CONFIGURATION，单击Get API Key。
  3. 单击Generate API Key。
  4. 在Generate a New API Key中，单击Confirm。

> 说明:

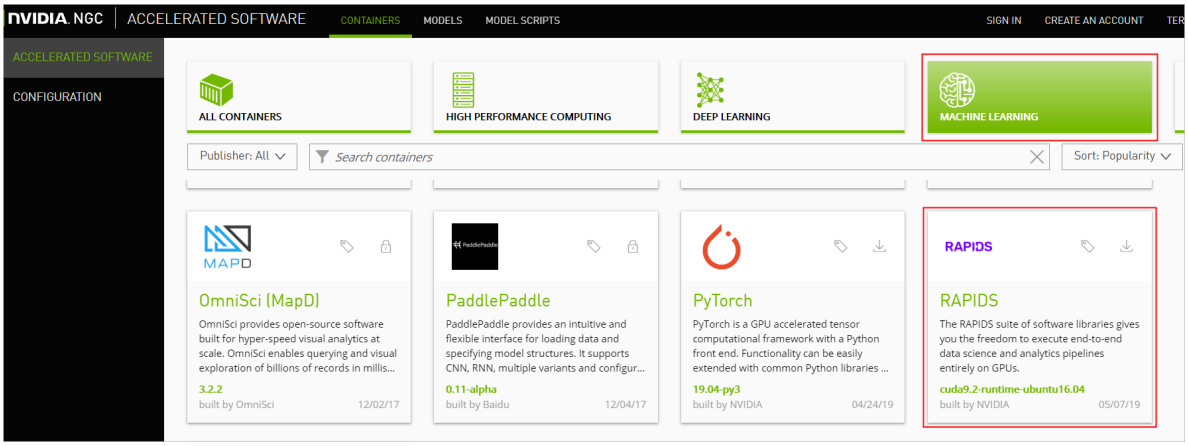
> 新的NGC API Key会覆盖旧的NGC API Key。如果您已持有NGC API Key，请确保不再需要旧的NGC API Key。

5. 复制API Key并保存到本地。



步骤一：获取RAPIDS镜像下载命令

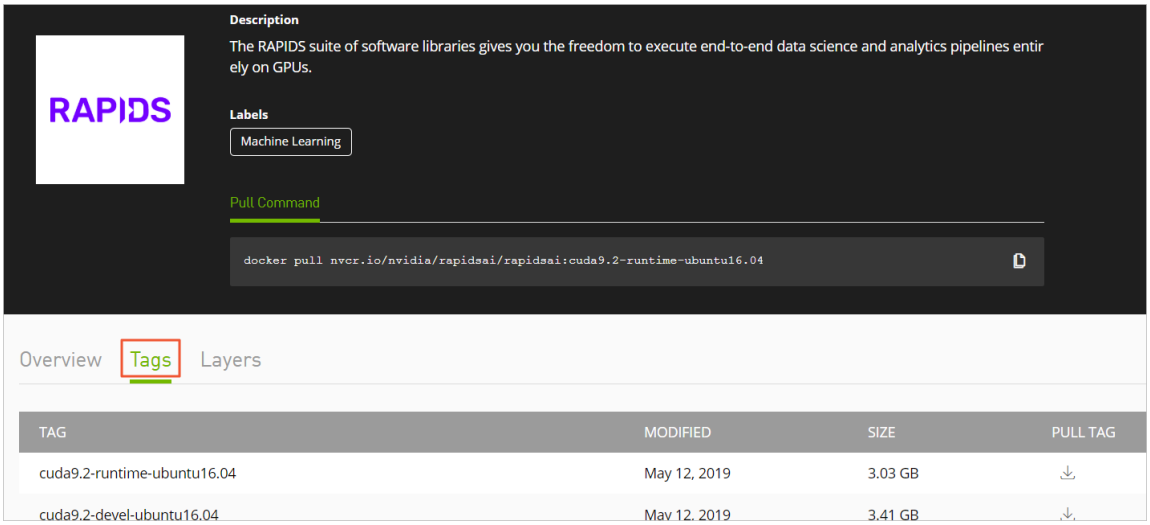
1. 登录NGC网站。
2. 打开MACHINE LEARNING页面，单击RAPIDS镜像。



3. 获取docker pull命令。

   本文示例代码基于RAPIDS 0.6版本镜像编写，因此在运行本示例代码时，使用Tag为0.6版本的镜像。实际操作时，请选择您匹配的版本。

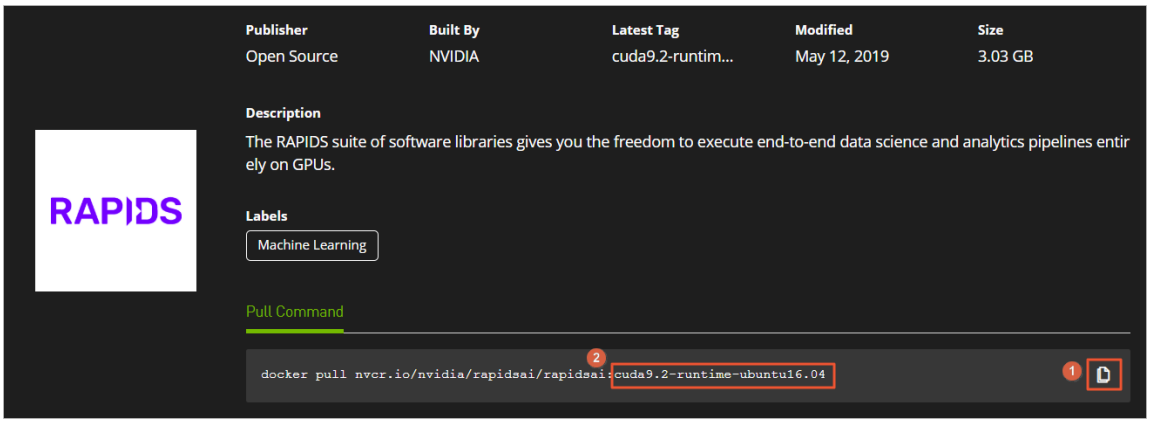
   a. 选择Tags页签。



   b. 找到并复制Tag信息。本示例中，选择`0.6-cuda10.0-runtime-ubuntu16.04-gcc5-py3.6`。

   c. 返回页面顶部，复制Pull Command中的命令到文本编辑器，将镜像版本替换为对应的Tag信息，并保存。 本示例中，将`cuda9.2-runtime-ubuntu16.04`替换为`0.6-cuda10.0-runtime-ubuntu16.04-gcc5-py3.6`。

   保存的docker pull命令用于在步骤二中下载RAPIDS镜像。



步骤二：部署RAPIDS环境

1. 创建一台GPU实例。

   详细步骤请参见使用向导创建实例。

   - 实例：RAPIDS仅适用于特定的GPU型号（采用NVIDIA Pascal及以上架构），因此您需要选择GPU型号符合要求的实例规格，目前有gn6i、gn6v、gn5和gn5i，详细的GPU型号请参见实例规格族。建议您选择显存更大的gn6i、gn6v或gn5实例。本示例中，选用了显存为16 GB的GPU实例。
   - 镜像：在镜像市场中搜索并使用`NVIDIA GPU Cloud VM Image`。

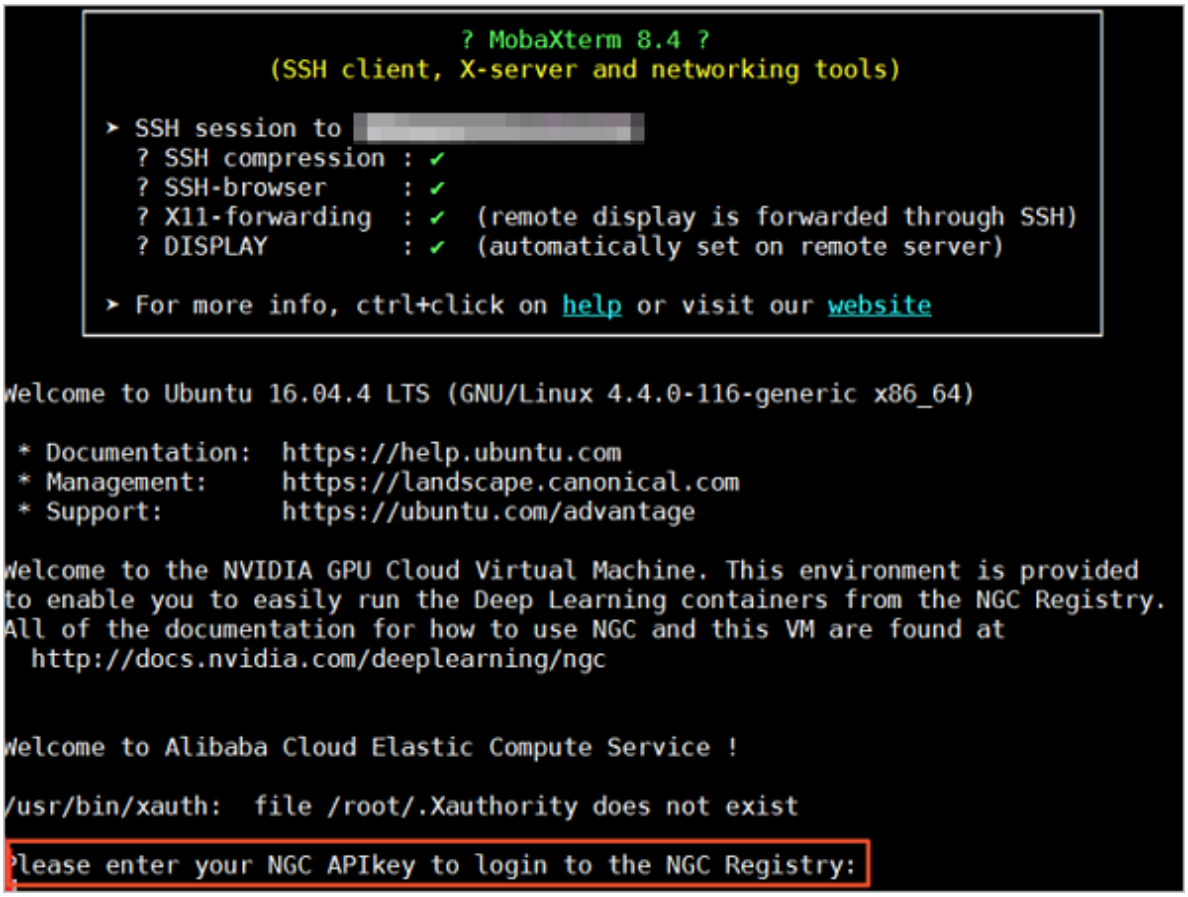
     

   - 公网带宽：选择分配公网IPv4地址或者在实例创建成功后绑定EIP地址。
   - 安全组：选择的安全组需要开放以下端口：
     - TCP 22 端口，用于SSH登录
     - TCP 8888端口，用于支持访问JupyterLab服务
     - TCP 8787端口、TCP 8786端口，用于支持访问Dask服务

2. 连接GPU实例。

   连接方式请参见连接Linux实例。

3. 输入NGC API Key后按回车键，登录NGC容器环境。



4. （可选）运行nvidia-smi查看GPU型号、GPU驱动版本等GPU信息。

   建议您了解GPU信息，预判规避潜在问题。例如，如果NGC的驱动版本太低，新Docker镜像版本可能会不支持。

5. 运行在步骤一中获取的docker pull命令下载RAPIDS镜像。

```
docker pull nvcr.io/nvidia/rapidsai/rapidsai:0.6-cuda10.0-runtime-ubuntu16.04-gcc5-py3.6
```

6. （可选）查看下载的镜像。

   建议您查看Docker镜像信息，确保下载了正确的镜像。

```
docker images
```

7. 运行容器部署RAPIDS环境。

```
docker run --runtime=nvidia \
        --rm -it \
        -p 8888:8888 \
        -p 8787:8787 \
        -p 8786:8786 \
```

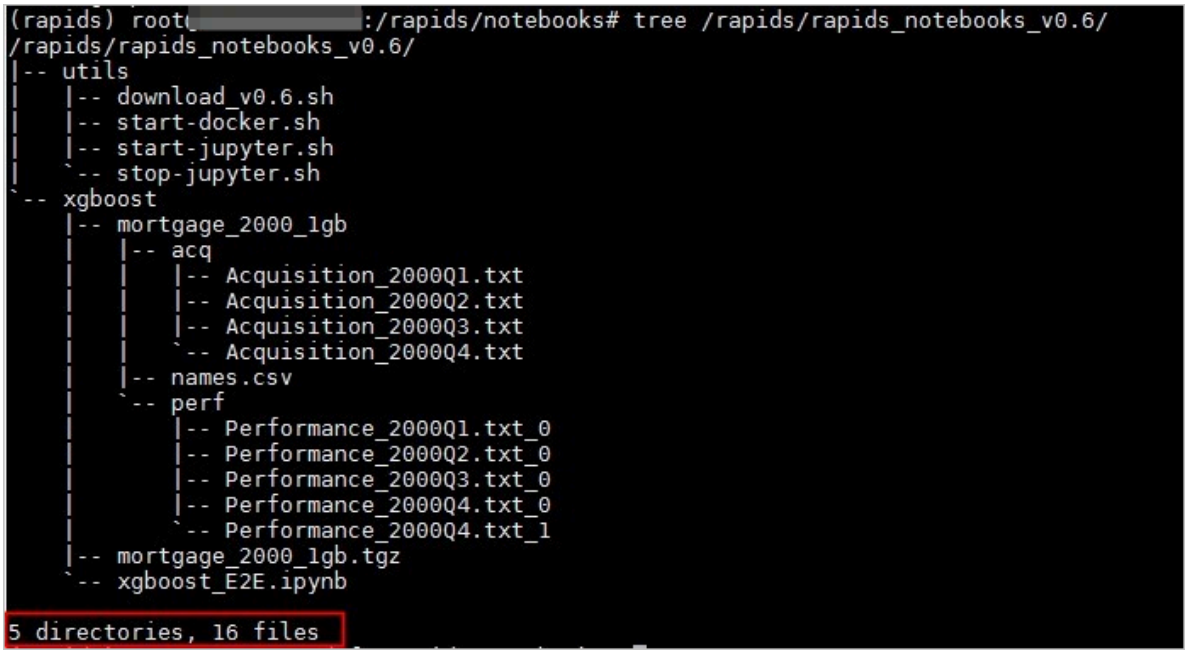
```
        nvcr.io/nvidia/rapidsai/rapidsai:0.6-cuda10.0-runtime-
ubuntu16.04-gcc5-py3.6
```

步骤三：运行RAPIDS Demo

1. **在GPU实例上下载数据集和Demo文件。**

```
# 获取apt源地址并下载脚本（脚本功能：下载训练数据、notebook、utils）
$ source_address=$(curl http://100.100.100.200/latest/meta-data/
source-address|head -n 1)
$ source_address="${source_address}/opsx/ecs/linux/binary/machine_le
arning/"
$ wget $source_address/rapids_notebooks_v0.6/utils/download_v0.6.sh
# 执行下载脚本
$ sh ./download_v0.6.sh
# 切换到下载目录查看下载文件
$ apt update
$ apt install tree
$ tree /rapids/rapids_notebooks_v0.6/
```

下载成功后的文件结构如下图，共5个文件夹、16个文件：



2. **在GPU实例上启动JupyterLab服务。**

   推荐直接使用命令启动。

```
# 切换到工作目录
$ cd /rapids/rapids_notebooks_v0.6/xgboost
# 启动jupyter-lab，直接使用命令启动，并设置登录密码
$ jupyter-lab --allow-root --ip=0.0.0.0 --no-browser --NotebookApp.
token='登录密码'
# 退出
```

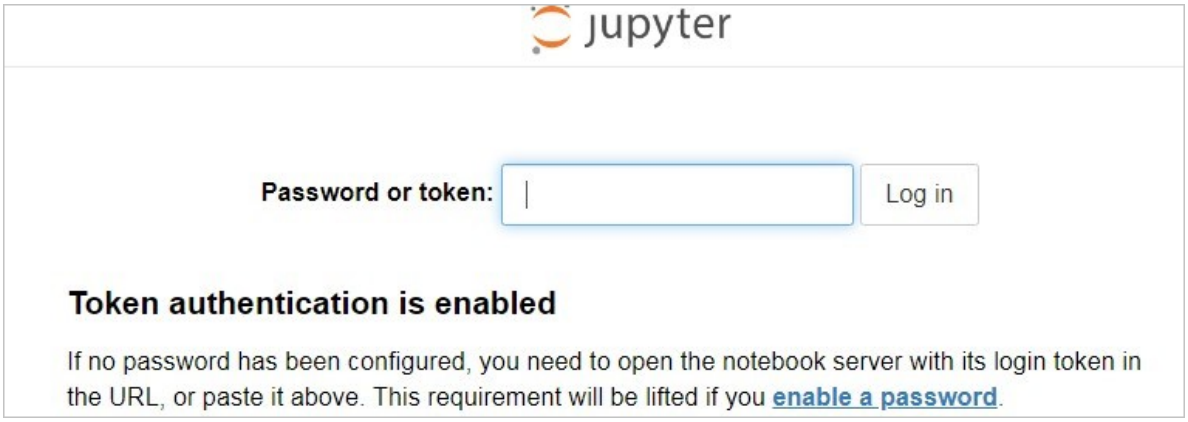
```
$ sh ../utils/stop-jupyter.sh
```

- 除使用命令外，您也可以执行脚本`$ sh ../utils/start-jupyter.sh`启动jupyter-lab，此时无法设置登录密码。
- 您也可以连续按两次`Ctrl+C`退出。

3. 打开浏览器，在地址栏输入`http://您的GPU实例IP地址:8888`远程访问JupyterLab 。

> **说明:**
> 推荐使用Chrome浏览器。

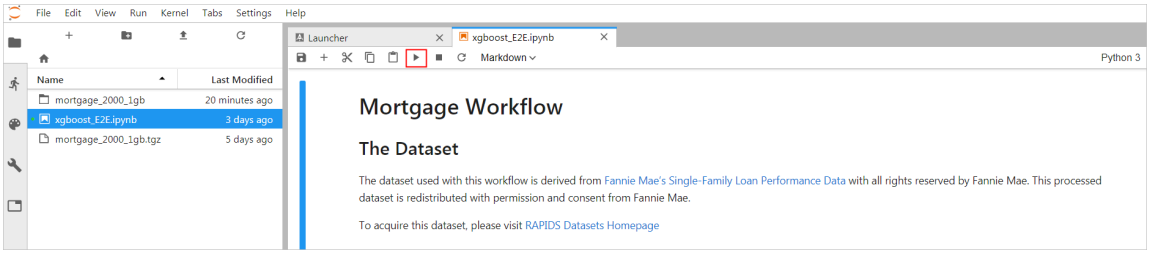
如果您在启动JupyterLab服务时设置了登录密码，会跳转到密码输入界面。



4. 运行NoteBook代码。

该案例是一个抵押贷款回归的任务，详细信息请参见代码执行过程。登录成功后，可以看到NoteBook代码的代码包括以下内容：

- *mortgage_2000_1gb*文件夹：存储解压后的训练数据。该文件夹下包含：acq文件夹、perf文件夹和names.csv文件。
- *xgboost_E2E.ipynb*文件： XGBoost Demo文件。双击文件可以查看文件详情，单击下图中的执行按钮可以逐步执行代码，每次执行一个Cell。



- *mortgage_2000_1gb.tgz*文件： 2000年的抵押贷款回归训练数据（1G分割的perf文件夹下的文件不会大于1G，使用1G分割的数据可以更有效的利用GPU显存）。

代码执行过程

该案例基于XGBoost演示了数据预处理到训练的端到端的过程，主要分为三个阶段：

- ETL（Extract-Transform-Load）：主要在GPU实例上进行。将业务系统的数据经过抽取、清洗转换之后加载到数据仓库。
- Data Conversion：在GPU实例上进行。将在ETL阶段处理过的数据转换为用于XGBoost训练的DMatrix格式。
- ML-Training：默认在GPU实例上进行。使用XGBoost训练梯度提升决策树 。

NoteBook代码的执行过程如下：

1. 准备数据集。

   本案例的Shell脚本会默认下载2000年的抵押贷款回归训练数据（*mortgage_2000_1gb.tgz*），并解压到*mortgage_2000_1gb*文件夹。

   如果您想获取更多数据用于XGBoost模型训练，可以设定参数download_url指定下载路径，具体下载地址请参见Mortgage Data。

   示例效果如下：

```
# 登录到数据下载页面：https://docs.rapids.ai/datasets/mortgage-data，官方提供了两种格式的数据集："Dataset"和"1GB Splits"，
# 其中"1GB Splits"适用于多GPU训练场景，这里建议下载"1GB Splits"格式数据集。只需将相应的下载链接地址赋值给 download_url 即可进行下载、解压。
# eg: download_url = 'http://rapidsai-data.s3-website.us-east-2.amazonaws.com/notebook-mortgage-data/mortgage_2000_1gb.tgz'

download_url = ''  # 如果 download_url = ''，则使用之前脚本已下载且解压好的数据集(数据已解压到文件夹:mortgage_2000-2001_1gb)。

if download_url != '':
    # 从url中截取要下载的文件名
    download_filename = download_url.split('/')[-1]
    # 数据文件解压目录。 默认使用文件名：如下载文件为 mortgage_2000-2001_1gb.tgz，则创建并解压到 mortgage_2000-2001_1gb 目录。
    mortgage_dir = '/rapids/rapids_notebooks_v0.6/xgboost/' + download_filename.split('.')[0]
    # 传入url并下载数据文件。如果 /rapids/rapids_notebooks_v0.6/xgboost/ 目录下已有下载文件，则不重新下载
    download_file_from_url(download_url, download_filename)
    # 将下载的文件解压到 mortgage_dir。如果文件夹mortgage_dir已经存在，则不重新解压。
    decompress_file(download_filename, mortgage_dir)
else:
    # 使用已下载的 mortgage_2000_1gb 数据集
    mortgage_dir = '/rapids/rapids_notebooks_v0.6/xgboost/mortgage_2000_1gb'
```

2. 设定相关参数。

参数名称	说明
start_year	指定选择训练数据的起始时间，ETL时会处理start_year到end_year之间的数据。
end_year	指定选择训练数据的结束时间，ETL时会处理start_year到end_year之间的数据。
train_with_gpu	是否使用GPU进行XGBoost模型训练，默认为True。
gpu_count	指定启动worker的数量，默认为1。您可以按需要设定参数值，但不能超出GPU实例的GPU数量。

参数名称	说明
part_count	指定用于模型训练的performance文件的数量，默认为 2 * gpu_count。如果参数值过大，在Data Conversion阶段会报错超出GPU内存限制，错误信息会在NoteBook后台输出。

示例效果如下：

3. 启动Dask服务。

   代码会启动Dask Scheduler，并根据gpu_count参数启动worker用于ETL和模型训练。

   示例效果如下：

4. 启动ETL。

ETL阶段会进行到表关联、分组、聚合、切片等操作，数据格式采用cuDF库的DataFrame格式（类似于pandas的DataFrame格式）。

示例效果如下：

ETL

Perform all of ETL with a single call to

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

```
%%time

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

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

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

5. 启动Data Conversion。

   将DataFrame格式的数据转换为用于XGBoost训练的DMatrix格式，每个worker处理一个DMatrix对象。

   示例效果如下：

Load the data from host memory, and convert to CSR

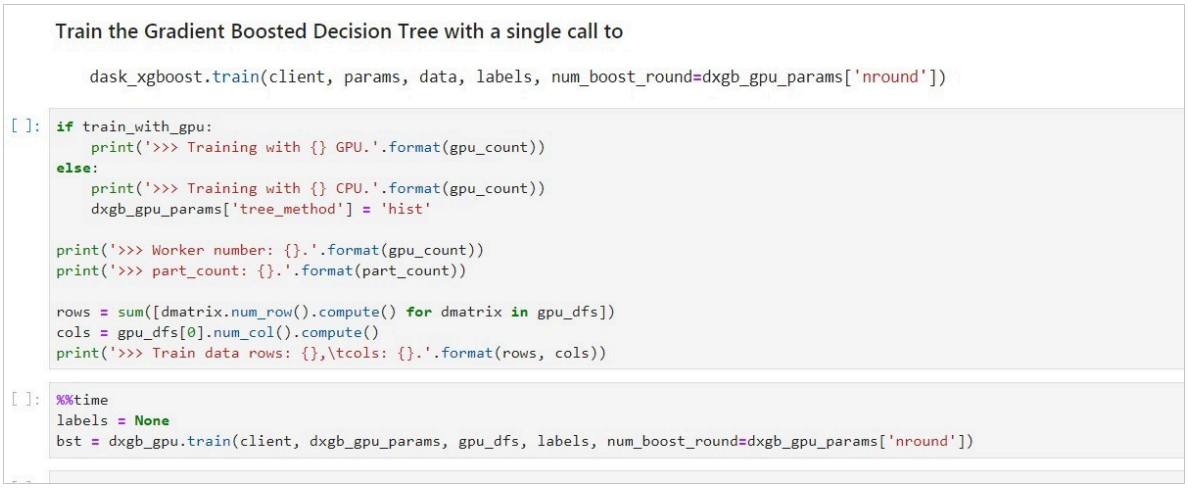
```
%%time

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

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

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

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

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

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

6. 启动ML Training。

   使用dask-xgboost启动模型训练，dask-xgboost负责多个dask worker间的通信协同工作，底层仍然调用xgboost执行模型训练。

   示例效果如下：



Train the Gradient Boosted Decision Tree with a single call to

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

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

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

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

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

相关函数

函数功能	函数名称
下载文件	def download_file_from_url(url, filename):
解压文件	def decompress_file(filename, path):
获取当前机器的GPU个数	def get_gpu_nums():
管理GPU内存	· def initialize_rmm_pool(): · def initialize_rmm_no_pool(): · def run_dask_task(func, **kwargs):
提交DASK任务	· def process_quarter_gpu(year=2000, quarter=1, perf_file=""): · def run_gpu_workflow(quarter=1, year=2000, perf_file="", **kwargs):
使用cuDF从CSV中加载数据	· def gpu_load_performance_csv(performance_path, **kwargs): · def gpu_load_acquisition_csv(acquisition_path, **kwargs): · def gpu_load_names(**kwargs):
处理和提取训练数据的特征	· def null_workaround(df, **kwargs): · def create_ever_features(gdf, **kwargs): · def join_ever_delinq_features(everdf_tmp, delinq_merge, **kwargs): · def create_joined_df(gdf, everdf, ** kwargs): · def create_12_mon_features(joined_df , **kwargs): · def combine_joined_12_mon(joined_df, testdf, **kwargs): · def final_performance_delinquency(gdf, joined_df, **kwargs): · def join_perf_acq_gdfs(perf, acq, ** kwargs): · def last_mile_cleaning(df, **kwargs):

8.3 在GPU实例上使用RAPIDS加速图像搜索任务

本文以使用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。

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

RAPIDS，全称Real-time Acceleration Platform for Integrated Data Science，是NVIDIA针对数据科学和机器学习推出的GPU加速库。更多RAPIDS信息请参见官方网站。

基于图像识别和搜索，图像搜索任务可以实现以图搜图，在不同行业应用和业务场景中帮助您搜索相同或相似的图片。

图像搜索任务背后的两项主要技术是特征提取及向量化、向量索引和检索。本文案例中，使用开源框架TensorFlow和Keras配置生产环境，然后使用ResNet50卷积神经网络完成图像的特征提取及向量化，最后使用RAPIDS cuML库的KNN算法实现BF方式的向量索引和检索。

> 说明:
> BF（Brute Force）检索方法是一种百分百准确的方法，对距离衡量算法不敏感，适用于所有的距离算法。

本文案例在阿里云gn6v（NVIDIA Tesla V100）实例上执行。执行案例后，对比了GPU加速的RAPIDS cuml KNN与CPU实现的scikit-learn KNN的性能，可以看到GPU加速的KNN向量检索速度为CPU的近600倍。

本文案例为单机单卡的版本，即一台GPU实例搭载一块GPU卡。

操作步骤

执行以下操作完成一次图像搜索任务：

1. 创建GPU实例
2. 启动和登录JupyterLab
3. 执行图像搜索案例

步骤一：创建GPU实例

具体步骤请参见使用向导创建实例。

- 实例：RAPIDS仅适用于特定的GPU型号（采用NVIDIA Pascal及以上架构），因此您需要选择GPU型号符合要求的实例规格，目前有gn6i、gn6v、gn5和gn5i。本文案例中，选用了ecs.gn6v-c8g1.2xlarge实例规格。
- 镜像：在镜像市场中使用关键字*RAPIDS*，搜索并使用预装了RAPIDS加速库的镜像。

- 安全组：选择的安全组需要开放TCP 8888端口，用于支持访问JupyterLab服务。

步骤二：启动和登录JupyterLab

1. 连接GPU实例，运行以下命令启动JupyterLab服务。

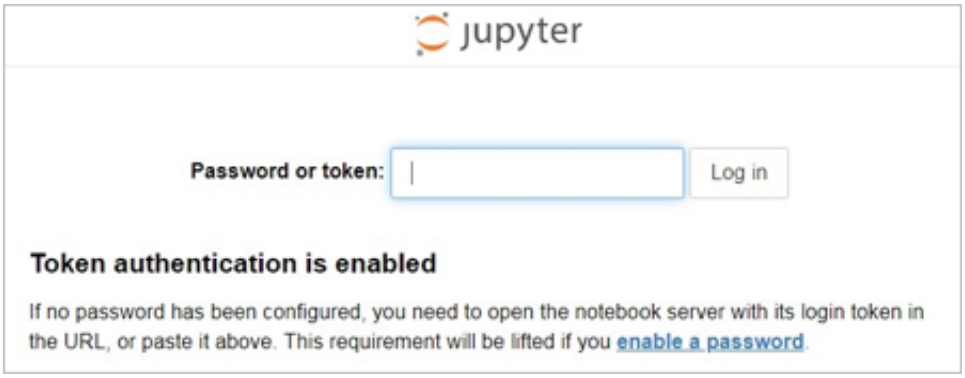
> **说明:**
> 连接GPU实例的步骤请参见连接方式导航。

```
# Go to the notebooks directory.
cd /rapids

# Run the following command to start JupyterLab and set the logon
password:
jupyter-lab --allow-root --ip=0.0.0.0 --no-browser --NotebookApp.
token='your logon password'

# Exit jupyterlab: press Ctrl+C twice.
```

2. 在您的本地机器上打开浏览器。输入`http://(IP address of your GPU instance):8888`远程访问JupyterLab。

> **说明:**
> 推荐使用Chrome浏览器。

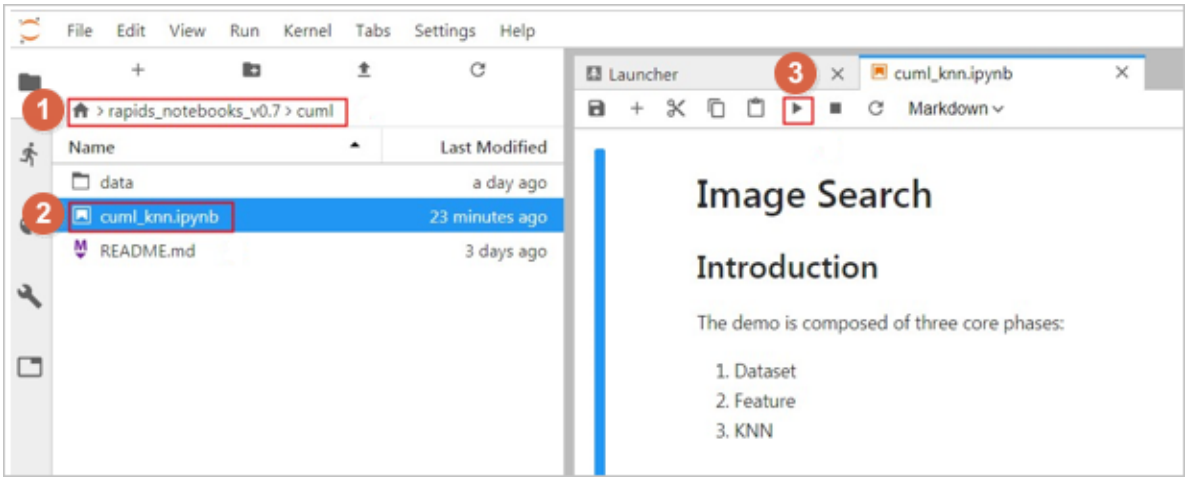
3. 输入启动命令中设置的密码，然后单击Log in。



步骤三：执行图像搜索案例

1. 进入案例所在目录*rapids_notebooks_v0.7/cuml*。
2. 双击*cuml_knn.ipynb*文件。
3. 单击  。

> **说明:**
> 单击一次执行一个cell，请单击至案例执行结束，详细说明请参见案例执行过程。



案例执行过程

图像搜索案例的执行过程分为三个步骤：处理数据集、提取图片特征和搜索相似图片。本文案例结果中对比了GPU加速的RAPIDS cuml KNN与CPU实现的scikit-learn KNN的性能。

1. 处理数据集。

   a) 下载和解压数据集。

   本文案例中使用了STL-10数据集，该数据集中包含10万张未打标的图片，图片的尺寸均为：96 x 96 x 3。您可以使用其他数据集，为便于提取图片特征，请确保数据集中图片的尺寸相同。

   本文案例提供了`download_and_extract(data_dir)`方法供您下载和解压STL-10数据集。RAPIDS镜像中已经将数据集下载到`./data`目录，您可以执行`download_and_extract()`方法直接解压数据集。

```
Download and Decompression

[2]: # the directory to save data
     data_dir = './data'
     # download and decompression
     download_and_extract(data_dir)

     >>> stl10_binary.tar.gz has exist in current directory.
     >>> Decompressing from ./data/stl10_binary.tar.gz....
     Successfully decompressed
```

   b) 读取图片。

   从数据集解压出的数据为二进制格式，执行`read_all_images(path_to_data)`方法加载数据并转换为NHWC（batch, height, width, channels）格式，以便用Tensorflow提取图片特征。

```
Read Data

[3]: # the path of unlabeled data
     path_unlabeled = os.path.join(data_dir, 'stl10_binary/unlabeled_X.bin')
     # get images from binary
     images = read_all_images(path_unlabeled)
     print('>>> images shape: ', images.shape)

     >>> images shape:  (100000, 96, 96, 3)
```

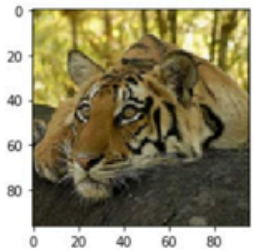
   c) 展示图片。

   执行`show_image(image)`方法随机展示一张数据集中的图片。

Show Image

```
[4]: import random
     import matplotlib.pyplot as plt
     %matplotlib inline

     def show_image(image):
         """show image"""
         fig = plt.figure(figsize=(3, 3))
         plt.imshow(image)
         plt.show()
         fig.clear()
```

```
[10]: # random show a image
      rand_image_index = random.randint(0, images.shape[0])
      show_image(images[rand_image_index])
```



d) 分割数据集。

按照9:1的比例把数据集分为两部分，分别用于创建图片索引库和搜索图片。

Split Dataset

```
from sklearn.model_selection import train_test_split

train_images, query_images = train_test_split(images, test_size=0.1, random_state=123)
print('train_images shape: ', train_images.shape)
print('query_images shape: ', query_images.shape)
```

```
train_images shape:  (90000, 96, 96, 3)
query_images shape:  (10000, 96, 96, 3)
```

2. 提取图片特征。

使用开源框架Tensorflow和Keras提取图片特征，其中模型为基于ImageNet数据集的ResNet50（notop）预训练模型。

a) 设定Tensorflow参数。

Tensorflow默认使用所有GPU显存，我们需要留出部分GPU显存供cuML使用。您可以选择一种方法设置GPU显存参数：

- 方法1：依据运行需求进行显存分配。

```
config.gpu_options.allow_growth = True
```

- 方法2：设定可以使用的GPU显存比例。本案例中使用方法2，并且GPU显存比例默认设置为0.3，即Tensorflow可以使用整块GPU显存的30%，您可以依据应用场景修改比例。

```
config.gpu_options.per_process_gpu_memory_fraction = 0.3
```

Image Features

```
# set tensorflow params to adjust GPU memory usage, if use default params, tensorflow would use
# nearly all of the gpu memory, we need reserve some gpu memory for cuml.
import os
# only use device 0
os.environ["CUDA_VISIBLE_DEVICES"] = "0"

import tensorflow as tf
from keras.backend.tensorflow_backend import set_session
config = tf.ConfigProto()
# method 1: allocate gpu memory base on runtime allocations
# config.gpu_options.allow_growth = True
# method 2: determines the fraction of the onerall amount of memory
# that each visibel GPU should be allocated.
config.gpu_options.per_process_gpu_memory_fraction = 0.3
set_session(tf.Session(config=config))
```

```
Using TensorFlow backend.
```

b) 下载ResNet50（notop）预训练模型。

连接公网下载模型（大小约91M），下载完成后默认保存到*/root/.keras/models/*目录。

参数名称	说明
weights	取值范围： · None：随机初始化权重值。 · imagenet：权重值的初始值设置为通过ImageNet预训练过的模型的权重值。 本案例中设置为imagenet。

参数名称	说明
include_top	取值范围： · True：包含整个ResNet50网络结构的最后一个全链接层。 · False：不包含整个ResNet50网络结构的最后一个全链接层。 本案例中，使用神经网络模型ResNet50的主要目的是提取图片特征而非分类图片，因此设置为False。
input_shape	可选参数，用于设置图片的输入shape，仅在include_top设置为False时生效。您必须为图片设置3个inputs channels，且宽和高不应低于32。此处设为(96，96，3)。

参数名称	说明
pooling	在include_top设置为False时，您需要设置池化层模式，取值范围： · None：输出为4D tensor。 · avg：平均池化，输出为2D tensor。 · max：最大池化，输出为2D tensor。 本案例中设置为max。

```
from keras.applications.resnet50 import ResNet50
from keras.preprocessing import image
from keras.applications.resnet50 import preprocess_input

# download resnet50(notop) model(first running) and load model
model = ResNet50(weights='imagenet', include_top=False, input_shape=(96, 96, 3), pooling='max')
```

```
WARNING:tensorflow:From /root/anaconda3/envs/rapids/lib/python3.6/site-packages/tensorflow/python/framework/op_def_library.py:263:
colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.2/resnet50_weights_tf_dim_ordering_tf_k
ernels_notop.h5
94658560/94653016 [==============================] - 9s 0us/step
```

您可以执行model.summary()方法查看模型的网络结构。

```
[10]: # network summary
      model.summary()
```

```
add_16 (Add)                    (None, 3, 3, 2048)   0           bn5c_branch2c[0][0]
                                                                 activation_46[0][0]
__________________________________________________________________________________________________
activation_49 (Activation)      (None, 3, 3, 2048)   0           add_16[0][0]
__________________________________________________________________________________________________
global_max_pooling2d_1 (GlobalM (None, 2048)         0           activation_49[0][0]
==================================================================================================
Total params: 23,587,712
Trainable params: 23,534,592
Non-trainable params: 53,120
```

c) 提取图片特征。

对分割得到的两个图片数据集执行model.predict()方法提取图片特征。

```
[10]: %%time
      train_features = model.predict(train_images)
      print('train features shape: ', train_features.shape)
```

```
train features shape:  (90000, 2048)
CPU times: user 33.6 s, sys: 7.94 s, total: 41.5 s
Wall time: 36.3 s
```

```
[11]: %%time
      query_features = model.predict(query_images)
      print('query features shape: ', query_features.shape)
```

```
query features shape:  (10000, 2048)
CPU times: user 3.64 s, sys: 704 ms, total: 4.34 s
Wall time: 3.76 s
```

3. 搜索相似图片。

a) 使用cuml KNN搜索相似图片。

通过`k=3`设置K值为3，即查找最相似的3张图片，您可以依据使用场景自定义K值。

其中，`knn_cuml.fit()`方法为创建索引阶段，`knn_cuml.kneighbors()`为搜索近邻阶段。

KNN向量检索耗时791 ms。

b) 使用scikit-learn KNN搜索相似图片。

通过`n_neighbors=3`设置K值为3，通过`n_jobs=-1`设置使用所有CPU进行近邻搜索。

说明:

ecs.gn6v-c8g1.2xlarge的配置为8 vCPU。

KNN向量检索耗时7分34秒。

c) 对比cuml KNN和scikit-learn KNN的搜索结果。

对比两种方式的KNN向量检索速度，使用GPU加速的cuml KNN耗时791 ms，使用CPU的scikit-learn KNN耗时7min 34s。前者为后者的近600倍。

验证两种方式的输出结果是否相同，输出结果为两个数组：

- distance：最小的K个距离值。本案例中搜索了10000张图片，K值为3，因此`distance.shape=(10000,3)`。
- indices：对应的图片索引。`indices.shape=(10000, 3)`。

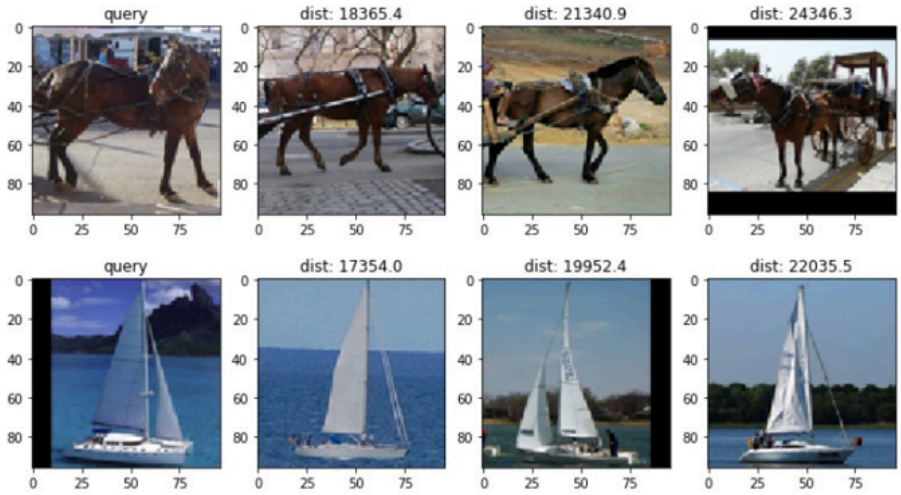
由于本案例所用数据集中存在重复图片，容易出现图片相同但索引不同的情况，因此使用`distances`，不使用`indices`对比结果。考虑到计算误差，如果两种方法得出的10000张图片中的3个最小距离值误差都小于1，则认为结果相同。

图片搜索结果

本案例从1万张搜索图片中随机选择5张图片并搜索相似图片，最终展示出5行4列图片。

第一列为搜索图片，第二列至第四列为图片索引库中的相似图片，且相似性依次递减。每张相似图片的标题为计算的距离，数值越大相似性越低。

```
# get random indices
random_show_index = np.random.randint(0, query_images.shape[0], size=5)
random_query = query_images[random_show_index]
random_indices = indices_cuml[random_show_index].astype(np.int)
random_distances = distances_cuml[random_show_index]

# show result images
for query_image, sim_indices, sim_dists in zip(random_query, random_indices, random_distances):
    sim_images = train_images[sim_indices]
    show_images(query_image, sim_images, sim_dists)
```



9 FaaS实例最佳实践

9.1 使用f1 RTL

本文描述如何使用f1 RTL（Register Transfer Level）。

前提条件

在开始本教程之前，请确认您已完成以下操作：

- 创建f1实例，确保实例能访问公网，并且实例所在安全组中已经添加规则放行SSH（22）端口的访问。

  > 说明：
  > f1实例只能使用镜像市场的FaaS F1基础镜像。详细信息，请参见创建f1实例。

- 您已经在云服务器ECS管理控制台f1实例的详情页上获取实例ID。
- 您必须先开通OSS服务，并创建一个OSS Bucket用于上传您的文件。Bucket与f1实例必须属于同一个账号、同一个地域。具体操作，请参见创建一个OSS Bucket。
- 如果需要加密服务，您还需要开通密钥管理服务。具体操作，请参见 开通密钥管理服务（KMS）。
- 使用RAM用户操作FPGA，必须完成以下操作：
  - 创建RAM用户并授权。
  - 创建RAM角色并授权。
  - 获取AccessKey ID和AccessKey Secret。

背景信息

> 说明：
> - 本文所述所有操作都必须由同一个账号在同一地域里执行。
> - 强烈建议您使用RAM用户操作FaaS实例。为了防止意外操作，您需要让RAM用户仅执行必要的操作。在操作FPGA镜像及下载时，因为您需要从指定的OSS Bucket下载原始DCP工程，所以您必须为FaaS管理账号创建一个角色，并授予临时权限，让FaaS管理账号访问指定的OSS Bucket。如果需要对IP加密，必须授予RAM用户KMS相关权限。如果需要做权限检查，必须授予查看用户资源的权限。

操作步骤

1. 远程连接f1实例。具体操作，请参见远程连接Linux实例。
2. 运行以下脚本配置基础环境。

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

3. 运行以下命令编译工程。

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

> **说明:**
> 编译时间较长，请耐心等待。

4. 制作镜像。
   a) 运行`faascmd`命令初始化。

```
#如果需要，添加环境变量及运行权限
export PATH=$PATH:/opt/dcp1_1/script/
chmod +x /opt/dcp1_1/script/faascmd
# 将hereIsYourSecretId替换为您的AccessKey ID，hereIsYourSecretKey替换为您的AccessKey Secret
faascmd config --id=hereIsYourSecretId --key=hereIsYourSecretKey
# 将hereIsYourBucket换为华东1地域里OSS Bucket名称
faascmd auth --bucket=hereIsYourBucket
```

   b) 在*/opt/dcp1_1/hw/samples/dma_afu*目录下，运行以下命令上传gbs文件。

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

   c) 运行以下命令制作镜像。

```
# 将hereIsYourImageName替换为您的镜像名称
faascmd create_image --object=dma_afu.gbs --fpgatype=intel --name=hereIsYourImageName --tags=hereIsYourImageTag --encrypted=false --shell=V1.1
```

5. 下载镜像。
   a) 运行`faascmd list_images`命令查看镜像是否制作成功。

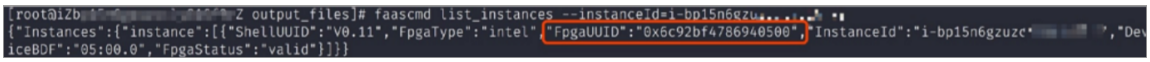
   返回结果里，如果出现`"State":"success"`，表示镜像制作成功。请记录返回结果里显示的`FpgaImageUUID`的值，稍后会用到。

   b) 运行命令获取FPGA ID。

```
# 将hereIsYourInstanceId替换为您的f1实例ID
```

```
faascmd list_instances --instanceId=hereIsYourInstanceId
```

以下为返回结果。请记录FpgaUUID的值。



c) 运行命令下载FPGA镜像到f1实例。

```
# 将hereIsYourInstanceID替换为刚刚保存的实例ID；将hereIsFpgaUUID替换为上一条命令中记下的FpgaUUID；将hereIsImageUUID替换为上一步记下FpgaImageUUID
faascmd download_image --instanceId=hereIsYourInstanceID --fpgauuid=hereIsFpgaUUID --fpgatype=intel --imageuuid=hereIsImageUUID --imagetype=afu --shell=V0.11
```

d) 运行命令检查是否下载成功。

```
# 将hereIsYourInstanceID替换为刚刚保存的实例ID；将hereIsFpgaUUID替换为上一条命令中记下的FpgaUUID;
faascmd fpga_status --instanceId=hereIsYourInstanceID --fpgauuid=hereIsFpgaUUID
```

如果返回结果里出现"TaskStatus":"operating"时，且FpgaImageUUID的值和下载镜像时FpgaImageUUID的值一致，说明下载成功。

6. 依次运行以下命令测试。

```
cd /opt/dcp1_1/hw/samples/dma_afu/sw
make
```

```
sudo LD_LIBRARY_PATH=/opt/dcp1_1/hw/samples/dma_afu/sw:$LD_LIBRARY
_PATH ./fpga_dma_test 0
```

如果您看到如图所示的输出结果，说明测试完成。

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

说明:

如果没有开启Huge pages，运行以下命令启用Huge pages。

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

9.2 f1实例OpenCL开发最佳实践

本文介绍如何在f1实例上使用OpenCL（Open Computing Language）制作镜像文件，并烧写到FPGA芯片中。

说明:

- 本文所述所有操作都必须由同一个账号在同一地域里执行。
- 强烈建议您使用RAM用户操作FaaS实例。为了防止意外操作，您需要让RAM用户仅执行必要的操作。在操作FPGA镜像及下载时，因为您需要从指定的OSS Bucket下载原始DCP工程，所以您必须为FaaS管理账号创建一个角色，并授予临时权限，让FaaS管理账号访问指定的OSS Bucket。如果需要对IP加密，必须授予RAM用户KMS相关权限。如果需要做权限检查，必须授予查看用户资源的权限。

前提条件

- 创建f1实例，确认实例能访问公网，并且实例所在安全组中已经添加规则放行SSH（22）端口的访问。

  说明:

  f1实例只能使用镜像市场的FaaS F1基础镜像。详细信息，请参见 创建f1实例。
- 您已经在 云服务器ECS管理控制台 f1实例的详情页上获取实例ID。
- 您必须先开通OSS服务，并 创建一个OSS Bucket 用于上传您的文件。Bucket与f1实例必须属于同一个账号、同一个地域。
- 如果需要加密文件，开通密钥管理服务（KMS）。
- 使用RAM用户操作FPGA，必须完成以下操作：
  - 创建RAM用户 并 授权。
  - 创建RAM角色 并 授权。
  - 获取AccessKey ID和AccessKey Secret。

操作步骤

按以下步骤在f1实例上使用OpenCL Example制作镜像文件，并烧写到FPGA芯片中。

第 1 步. 远程连接实例

远程连接Linux实例。

第 2 步. 安装基础环境

运行以下脚本安装基础环境。

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

第 3 步. 下载官方的OpenCL Example

按以下步骤下载官方的OpenCL Example。

1. 创建并切换到*/opt/tmp*目录。

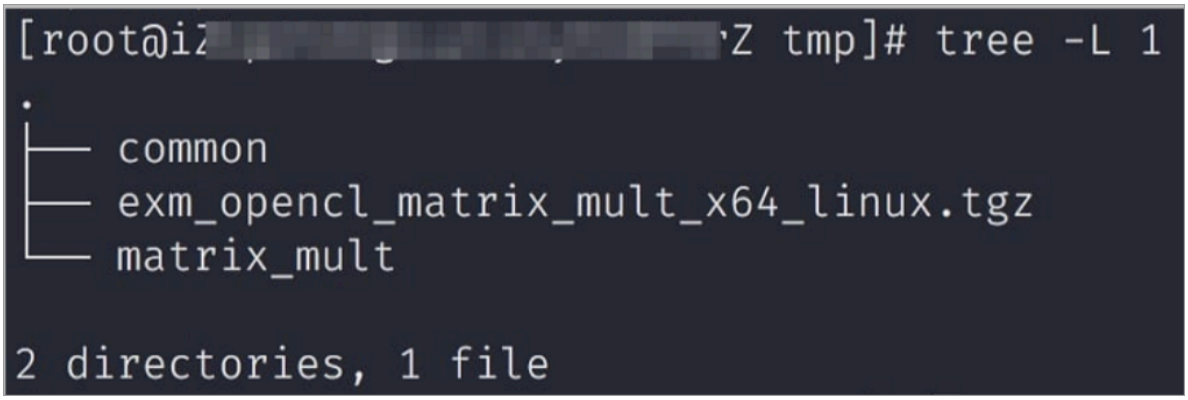
   ```
   mkdir -p /opt/tmp
   cd /opt/tmp
   ```

此时，您在*/opt/tmp*目录下。

2. 依次执行以下命令下载并解压Example文件。

```
wget https://www.altera.com/content/dam/altera-www/global/en_US/
others/support/examples/download/exm_opencl_matrix_mult_x64_linux.
tgz
tar -zxvf exm_opencl_matrix_mult_x64_linux.tgz
```

解压后的目录如下图所示。



3. 进入*matrix_mult*目录下，执行编译命令。

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

编译过程可能会持续数个小时，您可以再开一个会话，使用 `top` 命令监控系统占用，确定编译状态。

第 4 步. 上传配置文件

按以下步骤上传配置文件。

1. 运行以下命令初始化faascmd。

```
# 如果需要，要添加环境变量及运行权限
export PATH=$PATH:/opt/dcp1_1/script/
chmod +x /opt/dcp1_1/script/faascmd
# 将hereIsYourSecretId换为您的AccessKey ID，hereIsYourSecretKey替换为您
的AccessKey Secret
faascmd config --id=hereIsYourSecretId --key=hereIsYourSecretKey
# 将hereIsYourBucket换为华东1OSS的Bucket名称
faascmd auth --bucket=hereIsYourBucket
```

2. 进入matrix_mult/output_files，上传配置文件。

```
cd matrix_mult/output_files # 此时您应该在/opt/tmp/matrix_mult/
matrix_mult/output_files
faascmd upload_object --object=afu_fit.gbs --file=afu_fit.gbs
```

3. 使用gbs制作FPGA镜像。

```
# 将hereIsYourImageName换为您的镜象名，将hereIsYourImageTag替换为您的镜像
标签
faascmd create_image --object=dma_afu.gbs --fpgatype=intel --name=
hereIsYourImageName --tags=hereIsYourImageTag --encrypted=false --
shell=V1.1
```

4. 查看镜像是否制作成功：运行命令`faascmd list_images`。返回结果里，如果显示`"State":"success"`，表示镜像制作成功。请记录返回结果里显示的FpgaImageUUID，稍后会用到。

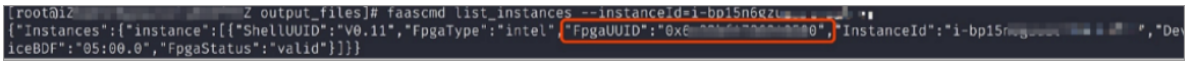
第 5 步. 下载镜像到f1实例

按以下步骤将镜像下载到f1实例。

1. 运行命令获取FPGA ID。

```
# 将hereIsYourInstanceId替换为您的FPGA实例ID
faascmd list_instances --instanceId=hereIsYourInstanceId
```

以下为返回结果。请记录FpgaUUID。



2. 运行命令下载镜像到f1实例。

```
# 将hereIsYourInstanceID替换为刚刚保存的实例ID；将hereIsFpgaUUID替换为上一
条命令中记下的FpgaUUID；将hereIsImageUUID替换为上一步记下的FpgaImageUUID
faascmd download_image  --instanceId=hereIsYourInstanceID --fpgauuid
=hereIsFpgaUUID --fpgatype=intel --imageuuid=hereIsImageUUID --
imagetype=afu --shell=V0.11
```

3. 运行命令检查是否下载成功。

```
# 将hereIsYourInstanceID替换为刚刚保存的实例ID；将hereIsFpgaUUID替换为上一
条命令中记下的FpgaUUID;
faascmd fpga_status --fpgauuid=hereIsFpgaUUID --instanceId=
hereIsYourInstanceID
```

如果返回结果里显示`"TaskStatus":"operating"`，说明下载成功。

第 6 步. 将FPGA镜像烧录到FPGA芯片

按以下步骤将FPGA镜像烧录到FPGA芯片。

1. 打开第2步环境的窗口。如果已关闭，重新执行第2步操作。
2. 运行命令配置OpenCL的运行环境。

```
sh /opt/dcp1_1/opencl/opencl_bsp/linux64/libexec/setup_permissions.
sh
```

3. 返回上级目录。

```
cd ../.. # 此时您在/opt/tmp/matrix_mult
```

4. 执行编译命令。

```
make
# 输出环境配置
export CL_CONTEXT_COMPILER_MODE_ALTERA=3
cp matrix_mult.aocx ./bin/matrix_mult.aocx
cd bin
host matrix_mult.aocx
```

当您看到如下输出时，说明配置完成。请注意，最后一行必须为`Verification: PASS`。

```
[root@iZbpXXXXXZ bin]# ./host matrix_mult.aocx
Matrix sizes:
  A: 2048 x 1024
  B: 1024 x 1024
  C: 2048 x 1024
Initializing OpenCL
Platform: Intel(R) FPGA SDK for OpenCL(TM)
Using 1 device(s)
  skx_fpga_dcp_ddr : SKX DCP FPGA OpenCL BSP (acl0)
Using AOCX: matrix_mult.aocx
Generating input matrices
Launching for device 0 (global size: 1024, 2048)
Time: 40.415 ms
Kernel time (device 0): 40.355 ms
Throughput: 106.27 GFLOPS
Computing reference output
Verifying
Verification: PASS
```

9.3 f3实例OpenCL开发最佳实践

本文介绍如何在f3实例上使用OpenCL（Open Computing Language）制作镜像文件，并烧录到FPGA芯片中。

前提条件

开始操作之前，您需要完成以下准备工作。

- 已创建f3实例。具体步骤，请参见创建f3实例。

  说明:

  - f3实例只能使用阿里云共享给您的镜像。
  - 创建实例时选择分配公网IP，确保实例能访问公网。
  - 实例所在安全组中已经添加规则放行SSH（22）端口的访问。
- 已在ECS控制台f3实例的详情页上，获取实例ID。
- 已使用同一个账号创建了与f3实例在同一地域的OSS Bucket。具体步骤，请参见开通OSS服务和创建一个OSS Bucket。
- 若使用RAM用户操作FPGA，确保您已经完成以下操作。
  - 创建RAM用户并授权。
  - 创建RAM角色并授权。
  - 获取AccessKey ID和AccessKey Secret。

操作须知

开始操作之前，您需要了解以下注意事项。

- 本文所述所有操作都必须由同一个账号在同一地域里执行。
- 建议您使用RAM用户操作FaaS实例。您需要为FaaS管理账号创建一个角色，并授予临时权限，让FaaS管理账号能访问指定的OSS Bucket。
- 本文的示例步骤和命令均基于2018.2版本Sdaccel开发环境。若您使用其他版本Sdaccel开发环境，步骤和命令可能会稍有差异。

基本流程

您需要按以下步骤在f3实例上使用OpenCL制作镜像文件，并烧写到FPGA芯片中。

- 步骤 1： 配置环境
- 步骤 2：编译二进制文件

- 步骤 3：检查打包脚本
- 步骤 4：制作镜像
- 步骤 5：下载镜像
- 步骤 6：运行Host程序

步骤 1： 配置环境

配置环境的操作步骤如下：

1. 远程连接f3实例。

   说明：

   后面步骤中的编译工程可能会持续数小时，建议您使用screen或者nohub等方式登录，防止ssh超时退出。

2. 运行以下命令安装Screen。

   ```
   yum install screen -y
   ```

3. 运行以下命令进入Screen。

   ```
   screen -S f3opencl
   ```

4. 运行以下命令配置环境。

   ```
   source /root/xbinst_oem/F3_env_setup.sh xocl  #每打开一个终端窗口就需要执行该命令一次
   ```

   说明：

   - 配置环境主要包括安装xocl驱动，设置vivado环境变量， 检查vivado license， 检测aliyun-f3 sdaccel平台，2018.2 runtime配置和faascmd版本检测。
   - 如果您要运行sdaccel的仿真，请勿运行以上命令配置环境。您只需要单独配置vivado的环境变量即可。
   - 推荐您使用Makefile方式仿真。

步骤 2：编译二进制文件

编译vadd二进制文件和kernel_global_bandwidth二进制文件的操作步骤如下：

- **示例一：编译vadd二进制文件**

  1. **复制*example*目录。**

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

  2. **进入*vadd*目录。**

     ```
     cd examples/vadd/
     ```

  3. **运行命令`cat sdaccel.mk | grep "XDEVICE="`查看XDEVICE的值，确保其配置为`XDEVICE=xilinx_aliyun-f3_dynamic_5_0`。**

  4. **按以下步骤修改*common.mk*文件。**

     a. **运行`vim ../common/common.mk`命令打开该文件。**

     b. **在第 61行代码（参数可能在 60~62 行，视您的文件而定）的末尾添加编译参数`--xp param:compiler.acceleratorBinaryContent=dcp`，修改后的代码如下：**

     ```
     CLCC_OPT += $(CLCC_OPT_LEVEL) ${DEVICE_REPO_OPT} --platform
      ${XDEVICE} ${KERNEL_DEFS} ${KERNEL_INCS} --xp param:compiler.
     acceleratorBinaryContent=dcp
     ```

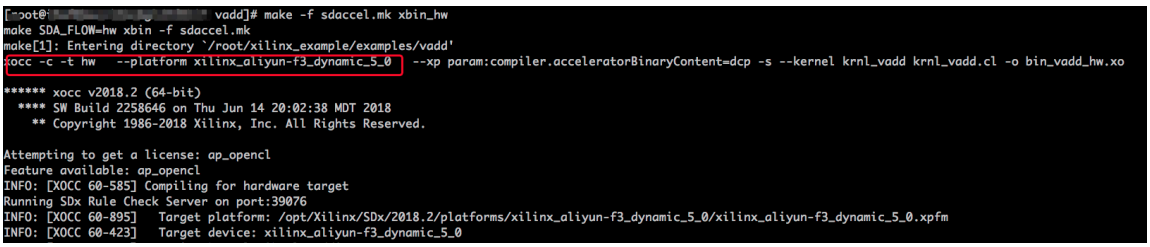
     说明：

     由于您必须向编译服务器提交DCP文件，所以需要添加`--xp param:compiler.acceleratorBinaryContent=dcp`编译参数，使得Xilinx® OpenCL™ Compiler（xocc）编译生成一个布局布线后的DCP文件，而不是bit文件。

  5. **运行以下命令编译程序。**

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

     如果您看到如下界面，说明二进制文件编译已经开始。编译过程可能会持续数个小时，请您耐心等待。

- 示例二：编译kernel_global_bandwidth二进制文件

1. 依次运行以下命令克隆`xilinx 2018.2 example`。

```
git clone https://github.com/Xilinx/SDAccel_Examples.git
```

```
cd SDAccel_Examples/
```

```
git checkout 2018.2
```

> **说明:**
> git分支必须为2018.2版本。

2. 运行`cd getting_started/kernel_to_gmem/kernel_global_bandwidth/`命令进入目录。
3. 按以下步骤修改*Makefile*文件。

   a. 运行`vim Makefile`命令打开该文件。

   b. 设置`DEVICES=xilinx_aliyun-f3_dynamic_5_0`。

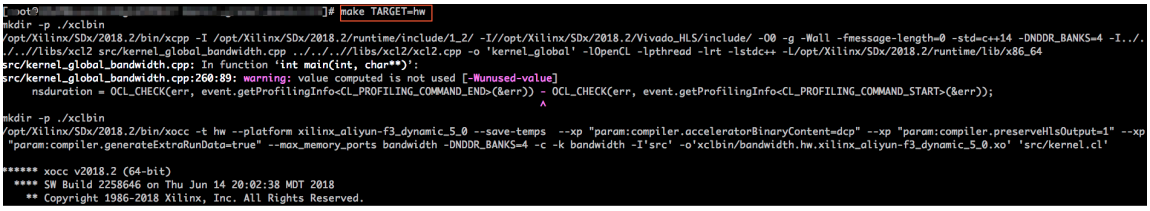
   c. 在第33行代码中添加编译参数`--xp param:compiler.acceleratorBinaryContent=dcp`，修改后的代码如下：

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

4. 运行以下命令编译程序。

```
make TARGET=hw
```

如果您看到该界面，说明二进制文件编译已经开始。编译工程可能会持续数小时，请您耐心等待。



步骤 3：检查打包脚本

运行以下命令检查打包脚本是否存在。

```
file /root/xbinst_oem/sdaccel_package.sh
```

如果返回结果中包含cannot open (No such file or directory)，说明不存在该文件，您需要运行以下命令手动下载打包脚本。

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

步骤 4：制作镜像

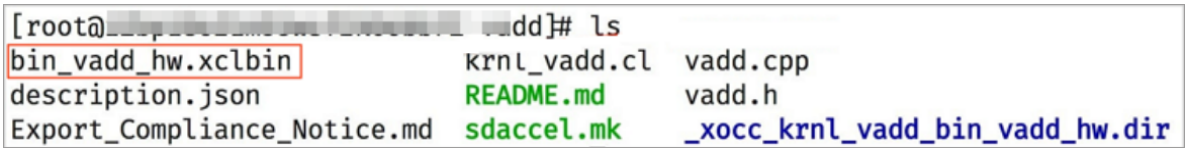
制作镜像文件的步骤如下：

1. 依次运行以下命令配置OSS环境。

```
faascmd config --id=hereIsYourSecretId --key=hereIsYourSecretKey #将hereIsYourSecretId和hereIsYourSecretKey替换为您的RAM用户AK信息
```

```
faascmd auth --bucket=hereIsYourBucket # 将hereIsYourBucket替换为您创建的OSS Bucket名称
```

2. 运行ls，获取后缀为.xclbin的文件名。



3. 运行以下命令打包二进制文件。

```
/root/xbinst_oem/sdaccel_package.sh -xclbin=/opt/Xilinx/SDx/2018.2/examples/vadd/bin_vadd_hw.xclbin
```

打包完成后，您会在同一目录下看到一个打包好的文件，如下图所示。

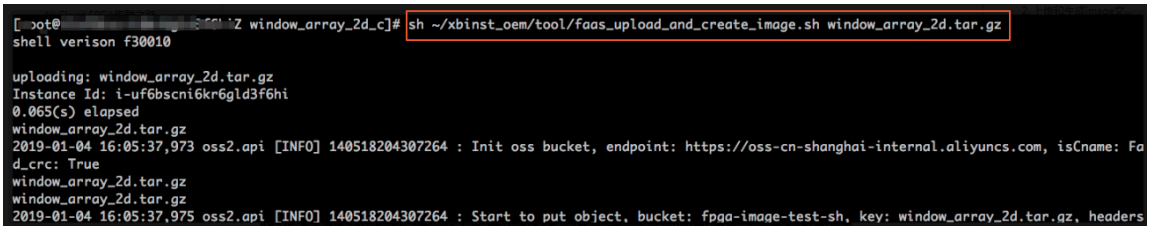
步骤 5：下载镜像

您可以采用脚本化流程或者单步操作流程来上传网表文件，并下载FPGA镜像。

- 脚本化流程：仅适用于配备单块FPGA卡的f3实例。

  1. 运行以下命令上传并生成镜像文件。

     ```
     sh /root/xbinst_oem/tool/faas_upload_and_create_image.sh <bit.tar.gz需要上传的压缩包文件名>
     ```

  2. 运行以下命令下载镜像文件。

     ```
     sh /root/xbinst_oem/tool/faas_download_image.sh <bit.tar.gz压缩包的文件名> <0/1>  # 最后的数字<0/1>为实例中fpga的序号
     ```

     说明:

0为FaaS实例中的第一个FPGA。单芯片实例序号一律为0；多芯片实例，例如，4芯片实例的序号为0，1，2，3。

如果需要对多个FPGA下载同一个镜像，可以在命令的末尾添加相应的序号。例如，对4芯片FPGA下载同一镜像的命令如下：

```
sh /root/xbinst_oem/tool/faas_download_image.sh <bit.tar.gz压缩包的文件名> 0
```

```
sh /root/xbinst_oem/tool/faas_download_image.sh <bit.tar.gz压缩包的文件名> 1
```

```
sh /root/xbinst_oem/tool/faas_download_image.sh <bit.tar.gz压缩包的文件名> 2
```

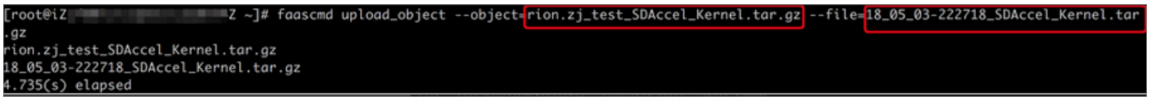
```
sh /root/xbinst_oem/tool/faas_download_image.sh <bit.tar.gz压缩包的文件名> 3
```

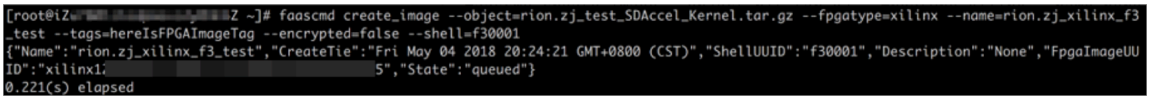
- 单步操作流程：使用faascmd工具操作。工具详情，请参见使用faascmd工具。

1. 依次运行以下命令，将压缩包上传到您个人的OSS Bucket，再将存放在您个人OSS Bucket中的gbs上传到FaaS管理单元的OSS Bucket中。

```
faascmd upload_object --object=bit.tar.gz --file=bit.tar.gz
```

```
faascmd create_image --object=bit.tar.gz --fpgatype=xilinx --name=hereIsFPGAImageName --tags=hereIsFPGAImageTag --encrypted=false --shell=hereIsShellVersionOfFPGA
```

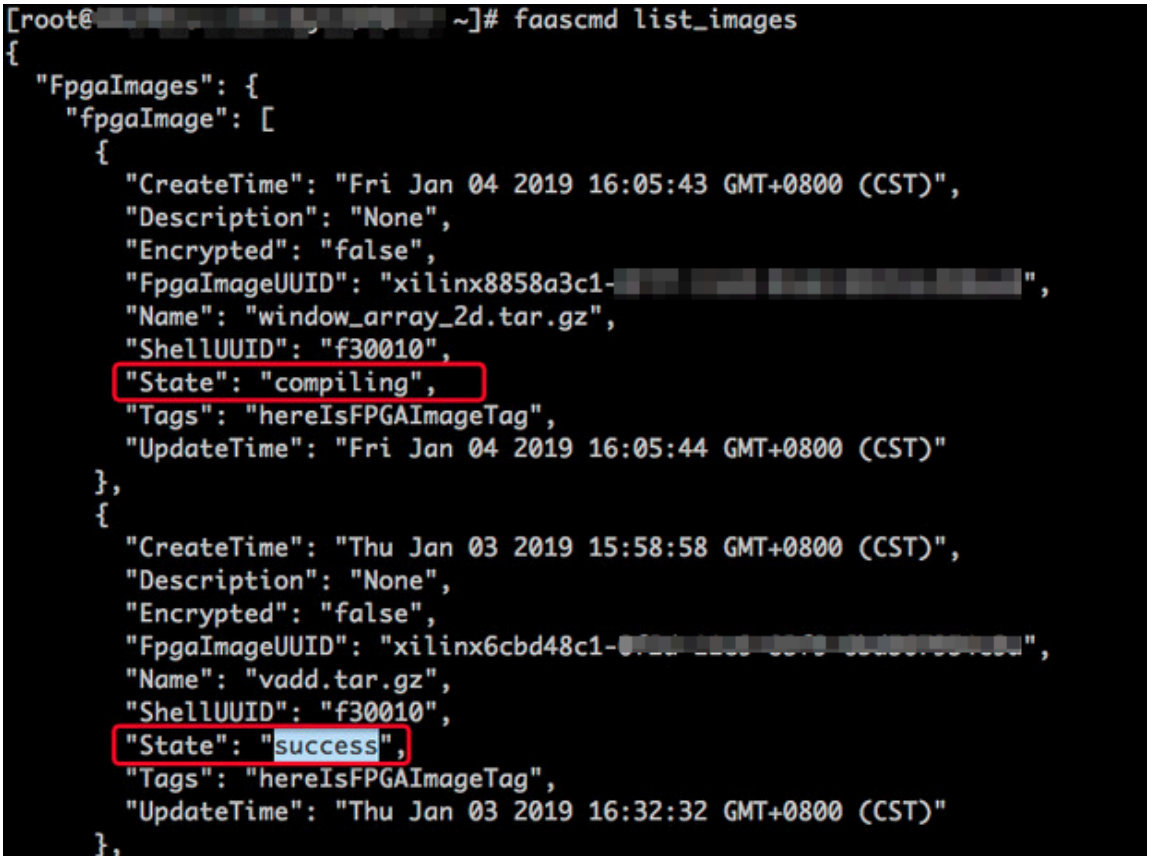




2. 运行以下命令查看FPGA镜像是否处于可下载状态。

```
faascmd list_images
```

返回结果中，如果FPGA镜像的"State"为"compiling"，表示FPGA镜像处于编译状态，您需要继续等待。如果FPGA镜像的"State"为"success"，表示FPGA镜像已可以下载，您需要找到并记录FpgaImageUUID。

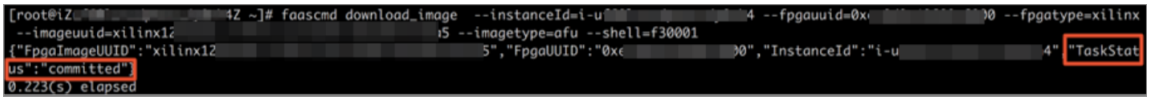


3. 运行以下命令在返回结果中，找到并记录FpgaUUID。

```
faascmd list_instances --instanceId=hereIsYourInstanceId # 将
hereIsYourInstanceId替换为f3实例ID
```

4. 运行以下命令下载FPGA镜像。

```
faascmd download_image --instanceId=hereIsYourInstanceId --
fpgauuid=hereIsFpgaUUID --fpgatype=xilinx --imageuuid=hereIsImag
eUUID --imagetype=afu --shell=hereIsShellVersionOfFpga
# hereIsYourInstanceId替换为f3的实例ID，hereIsFpgaUUID替换为您获取的
FpgaUUID，hereIsImageUUID替换为您获取的FpgaImageUUID
```



5. 运行以下命令查看镜像是否下载成功。

```
faascmd fpga_status --fpgauuid=hereIsFpgaUUID --instanceId=
hereIsYourInstanceId # hereIsFpgaUUID替换为您获取的FpgaUUID，
hereIsYourInstanceId替换为f3实例ID。
```

以下为返回结果示例。如果显示的FpgaImageUUID与您获取的FpgaImageUUID一致，并且显示"TaskStatus":"valid"，说明镜像下载成功。

步骤 6：运行Host程序

您需要按照下列步骤运行Host程序。

1. 运行以下命令配置环境。

```
source /root/xbinst_oem/F3_env_setup.sh xocl  # 每打开一个终端窗口就需要执行该命令一次
```

2. 配置*sdaccel.ini*文件。

   在Host二进制文件所在目录下，运行`vim sdaccel.ini`命令创建*sdaccel.ini*文件并输入下列内容。

```
[Debug]
profile=true
[Runtime]
runtime_log = "run.log"
hal_log = hal.log
ert=false
kds=false
```

3. 运行host。

   - vadd运行命令如下：

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

```
./vadd bin_vadd_hw.xclbin
```

   - kernel_global_bandwidth运行命令如下：

```
./kernel_global
```

如果返回结果中出现`Test Passed`，说明测试通过。

其他操作

本节介绍FPGA实例的部分常用操作。

任务	命令
查看帮助文档	`make -f ./sdaccel.mk help`
软件仿真	`make -f ./sdaccel.mk run_cpu_em`
硬件仿真	`make -f ./sdaccel.mk run_hw_em`
只编译 host 代码	`make -f ./sdaccel.mk host`
编译生成可以下载的文件	`make -f sdaccel.mk xbin_hw`
清理工作目录	`make -f sdaccel.mk clean`

任务	命令
强力清除工作目录	`make -f sdaccel.mk cleanall`

说明:

- 仿真时只需要按照Xilinx标准流程操作，不需要配置F3_env_setup环境。
- SDAccel runtime和SDAccel开发平台已在阿里云f3官方镜像中提供。 您也可以点击后面的链接直接下载SDAccel runtime和SDAccel开发平台。

9.4 f3实例RTL开发最佳实践

本文描述基于f3实例的RTL（Register Transfer Level）开发流程。

说明:

- 本文所述所有操作必须由同一个账号在同一个地域执行。
- 强烈建议您使用RAM用户操作FPGA实例。基于最小授权原则，建议您不要对RAM用户过度授权，而只授予RAM用户刚好满足其工作所需的权限，比如访问OSS bucket获取原始DCP/xclbin文件、上传Vivado编译log、操作指定的ECS实例等。您还需要指定RAM角色AliyunFAASDefaultRole，FaaS服务默认使用此角色来访问您在其他云产品中的资源，其权限策略AliyunFAASRolePolicy还包括KMS相关的权限，以便您使用KMS服务对IP进行加密。

前提条件

- 您已经 创建f3实例，实例能访问公网，并且实例所在安全组中已经添加对SSH（22）端口访问放行的规则。
- 登录 云服务器ECS管理控制台，在f3实例的详情页上，获取实例ID。
- 在华东2 创建一个OSS Bucket，专门用于FaaS服务。

  说明:
  这个Bucket会对FaaS管理账号开通读写权限，因此不建议您存储与FaaS无关的内容。
- 如果使用RAM用户操作FPGA，必须完成以下操作：
  - 新建RAM用户并授权。
  - 授权FaaS服务角色。
  - 获取AccessKey ID和AccessKey Secret。

操作步骤

1. 远程连接Linux实例。

   > **说明:**
   > 编译工程时需要 2 ~ 3 小时。建议您使用nohup或者VNC连接实例，以免编译时意外退出。

2. 下载并解压 RTL参考设计。
3. 配置环境。

   - 如果驱动为 `xdma`，需要运行以下命令来配置环境。

     ```
     source /root/xbinst_oem/F3_env_setup.sh xdma #每打开一个终端窗口就需要执行该命令一次
     ```

   - 如果驱动为 `xocl`，则需要运行以下命令来配置环境。

     ```
     source /root/xbinst_oem/F3_env_setup.sh xocl #每打开一个终端窗口就需要执行该命令一次
     ```

   > **说明:**
   > 配置环境主要包括安装xdma驱动或xocl驱动，设置vivado环境变量，检查vivado license，检测aliyun-f3 sdaccel平台，2018.2 runtime配置和faascmd版本检测 。

4. 指定OSS存储空间。

   ```
   faascmd config --id=hereIsYourSecretId --key=hereIsYourSecretKey #将hereIsYourSecretId和hereIsYourSecretKey替换为您的RAM用户AK信息
   faascmd auth --bucket=hereIsYourBucket # 将hereIsYourBucket替换为您创建的OSS Bucket名称
   ```

5. 运行以下命令编译RTL工程。

   ```
   cd <您之前解压的路径>/hw/ # 进入解压后的hw路径
   sh compiling.sh
   ```

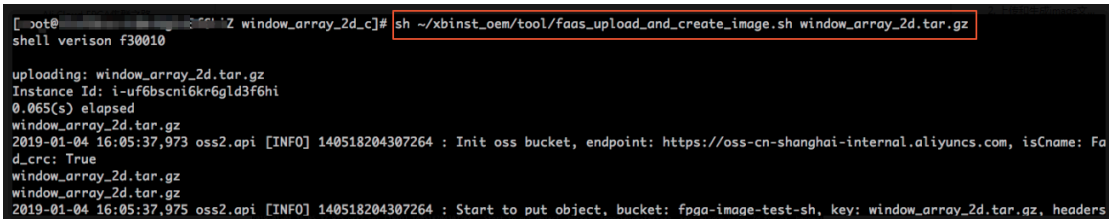
   > **说明:**
   > 编译工程需要2 ~ 3小时。

6. 上传网表文件，并下载FPGA镜像。您可以采用脚本化流程或者单步操作流程完成该步骤。

   · 脚本化流程：仅适用于配备单块FPGA卡的f3实例。

   a. 运行以下命令上传并生成镜像文件。

   ```
   sh /root/xbinst_oem/tool/faas_upload_and_create_image.sh <bit.
   tar.gz需要上传的压缩包文件名>
   ```

   b. 下载镜像文件。

   ```
   sh /root/xbinst_oem/tool/faas_download_image.sh <bit.tar.gz压缩
   包的文件名> <0/1> # 最后的数字<0/1>为实例中fpga的序号
   ```

   0为FaaS实例中的第一个FPGA，单芯片实例序号一律为0，对多芯片实例，例如4芯片的序号为0,1,2,3。

   如果需要对多个FPGA下载同一个镜像，可以在命令的末尾添加相应的序号。例如，对4芯片FPGA下载同一镜像的命令为：

   ```
   sh /root/xbinst_oem/tool/faas_download_image.sh <bit.tar.gz压缩
   包的文件名> 0
   sh /root/xbinst_oem/tool/faas_download_image.sh <bit.tar.gz压缩
   包的文件名> 1
   sh /root/xbinst_oem/tool/faas_download_image.sh <bit.tar.gz压缩
   包的文件名> 2
   sh /root/xbinst_oem/tool/faas_download_image.sh <bit.tar.gz压缩
   包的文件名> 3
   ```

   · 单步操作流程：使用faascmd工具 进行操作。

   a. 运行以下命令，将压缩包上传到您个人的OSS Bucket，再将存放在您个人OSS Bucket中的gbs上传到FaaS管理单元的OSS Bucket中。

   ```
   faascmd upload_object --object=bit.tar.gz --file=bit.tar.gz
   faascmd create_image --object=bit.tar.gz --fpgatype=xilinx --
   name=hereIsFPGAImageName --tags=hereIsFPGAImageTag --encrypted=
   false --shell=hereIsShellVersionOfFPGA
   ```

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

```
[root@iZ                Z ~]# faascmd create_image --object=rion.zj_test_SDAccel_Kernel.tar.gz --fpgatype=xilinx --name=rion.zj_xilinx_f3
_test --tags=hereIsFPGAImageTag --encrypted=false --shell=f30001
{"Name":"rion.zj_xilinx_f3_test","CreateTie":"Fri May 04 2018 20:24:21 GMT+0800 (CST)","ShellUUID":"f30001","Description":"None","FpgaImageUU
ID":"xilinx1                                5","State":"queued"}
0.221(s) elapsed
```

b. 运行命令查看FPGA镜像是否处于可下载状态。

```
faascmd list_images
```

在返回结果中，如果State为 compiling，表示FPGA镜像处于编译状态，您需要继续等待。如果 State 为 success，表示FPGA镜像已经可以下载。您需要找到并记录FpgaImageUUID。

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

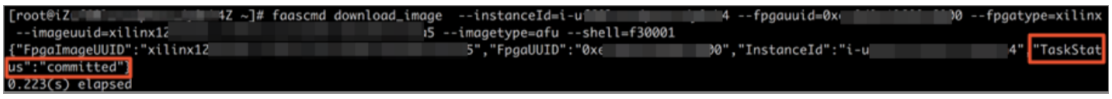
c. 运行以下命令。在命令返回结果中，您需要找到并记录FpgaUUID。

```
faascmd list_instances --instanceId=hereIsYourInstanceId # 将
hereIsYourInstanceId替换为f3实例ID
```

d. 运行以下命令下载FPGA镜像。

```
faascmd download_image --instanceId=hereIsYourInstanceId
 --fpgauuid=hereIsFpgaUUID --fpgatype=xilinx --imageuuid=
hereIsImageUUID --imagetype=afu --shell=hereIsShellVersionOf
Fpga
```

```
# hereIsYourInstanceId替换为f3的实例ID，hereIsFpgaUUID替换为您获取的FpgaUUID，hereIsImageUUID替换为您获取的FpgaImageUUID
```



e. 运行以下命令查看镜像是否下载成功。

```
faascmd fpga_status --fpgauuid=hereIsFpgaUUID --instanceId=hereIsYourInstanceId # hereIsFpgaUUID替换为您获取的FpgaUUID，hereIsYourInstanceId替换为f3实例ID。
```

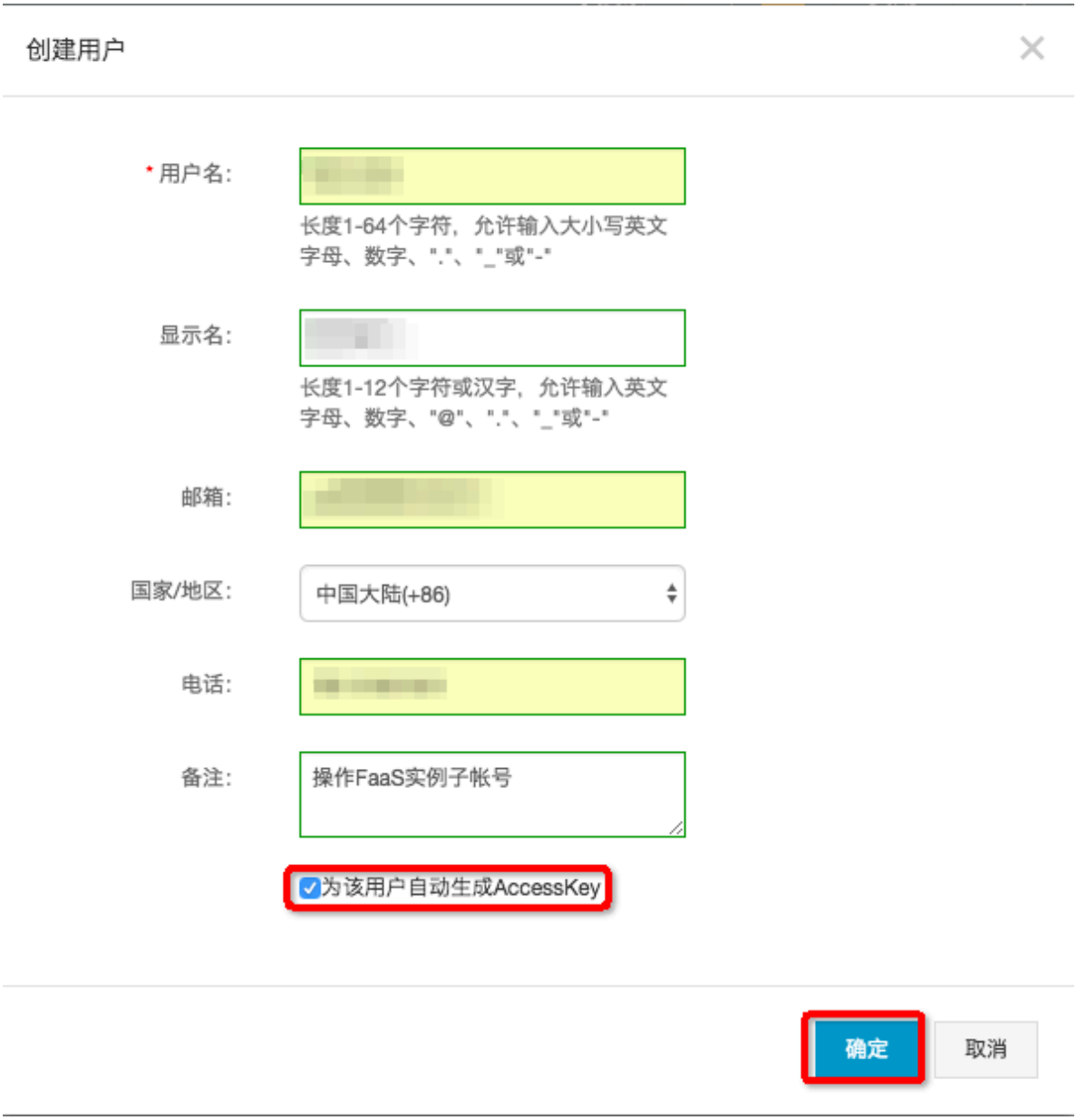
以下为返回结果示例。如果显示的FpgaImageUUID与您获取的FpgaImageUUID一致，并且显示 `"TaskStatus":"valid"`，说明镜像下载成功。

新建RAM用户并授权

请按照以下步骤新建一个RAM用户并授予必要的权限。

1. 登录RAM控制台。
2. 在用户管理页面，单击新建用户。

3. 填写用户名、显示名、邮箱等信息，并勾选为该用户自动生成AccessKey，单击确定。



4. 单击保存AK信息。

注意:

每对AK仅一次下载机会，请妥善保管AccessKeyID和AccessKeySecret。如果AK丢失，您只能重新创建AK。更多信息，请参见RAM用户AK说明。

5. 在用户管理页面，在新建的RAM用户操作列下，单击授权。

6. 为该RAM用户授予权限，至少包括：AliyunOSSFullAccess、AliyunECSFullAccess、AliyunRAMFullAccess和AliyunSTSAssumeRoleAccess，然后单击确定。

FAQ

上传镜像时出现异常，如何查看异常详情?

如果您的工程在上传生成镜像的过程中出现异常，例如云上编译服务器编译报错，你可以通过以下两种方式来查看异常详情：

- 查看faas_compiling.log。使用上传脚本faas_upload_and_create_image.sh时，如果编译失败会自动下载并打印faas_compiling.log到terminal中。
- 手动执行命令查看编译log文件：`sh /root/xbinst_oem/tool/faas_checklog.sh <bit.tar.gz之前上传的压缩包文件名>`

如何重新加载镜像?

您可以参考以下步骤重新加载镜像：

1. 卸载驱动。
   - 如果您安装了`xdma`驱动，需要在实例中运行`sudo rmmod xdma`命令卸载驱动。
   - 如果您安装了`xocl`驱动，则需要在实例中运行`sudo rmmod xocl`命令卸载驱动。

2. 下载镜像。您可以使用以下两种方式之一：

   · 使用脚本：

   ```
   sh faas_download_image.sh bit.tar.gz <0/1> #最后的数字为实例中FPGA的序号
   ```

   · 使用faascmd：

   ```
   faascmd download_image --instanceId=hereIsYourInstanceId --fpgauuid=hereIsFpgaUUID --fpgatype=xilinx --imageuuid=hereIsImageUUID --imagetype=afu --shell=hereIsShellVersionOfFpga
   ```

3. 安装驱动。

   · 如果您需要安装 `xdma` 驱动，运行以下命令。

   ```
   sudo depmod
   sudo modprobe xdma
   ```

   · 如果您需要安装 `xocl` 驱动，则需要运行以下命令。

   ```
   sudo depmod
   sudo modprobe xocl
   ```

9.5 faascmd工具

9.5.1 faascmd工具概述

faascmd是阿里云FPGA云服务器（FaaS）提供的一个命令行工具，是基于python SDK开发的脚本。

您可以使用faascmd工具：

· 进行授权及相关操作
· 管理和操作FPGA镜像
· 查看和上传objects
· 获取FPGA实例信息

9.5.2 安装faascmd

本文为您介绍如何下载安装faascmd工具。

前提条件

在安装faascmd工具之前，请确认已完成以下操作：

- 您需要在运行fasscmd的实例上完成以下准备工作：

  1. 运行`python -V`命令检查Python版本，需为2.7.x。

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

  2. 运行以下命令安装python模块。

     ```
     pip -q install oss2
     pip -q install aliyun-python-sdk-core
     pip -q install aliyun-python-sdk-faas
     pip -q install aliyun-python-sdk-ram
     ```

  3. 运行以下命令检查aliyun-python-sdk-core的版本号，需为2.11.0或以上版本。

     ```
     cat /usr/lib/python2.7/site-packages/aliyunsdkcore/__init__.py
     ```

     ```
     [root@testhost python2.7]# cat /usr/lib/python2.7/site-packages/aliyunsdkcore/__init__.py
     __version__ = "2.11.0"[root@testhost python2.7]#
     ```

     说明：
     如果版本号低于2.11.0，运行`pip install --upgrade aliyun-python-sdk-core`命令升级至最新版本。

- 获取RAM用户的AccessKey ID和AccessKey Secret

操作步骤

1. 登录实例后，在当前目录或任意目录下运行`wget http://fpga-tools.oss-cn-shanghai.aliyuncs.com/faascmd`命令下载faascmd。

   说明：
   请记录此目录。在配置faascmd时，您需要把faascmd所在目录的绝对路径添加到PATH变量中。

2. 运行以下命令为faascmd添加可执行权限。

   ```
   chmod +x faascmd
   ```

9.5.3 配置faascmd

在使用faascmd之前，您需要配置相关环境变量和RAM用户的AccessKey。

操作步骤

1. 登录您的实例后，运行以下命令配置PATH环境变量。

```
export PATH=$PATH:<faascmd工具所在路径>
```

2. 运行下列命令配置AccessKey ID和AccessKey Secret。

```
faascmd config --id=<yourAccessKeyID> --key=<yourAccessKeySecret>
```

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

9.5.4 使用faascmd

您可以通过本主题了解faascmd命令的用法。

前提条件

使用faascmd工具之前，您需要先 配置faascmd 。

语法说明

- faascmd工具提供的所有命令和参数都严格区分大小写。
- faascmd命令中各参数“=”前后不能有多余空格。

授权

`faascmd auth` 命令用于授权faas admin访问用户的OSS bucket。

前提条件

1. 为FaaS新建一个OSSbucket，用于上传原始编译的DCP文件。
2. 在该FaaSOSSbucket中，新建一个名为compiling_logs的文件夹。

命令格式

```
faascmd auth --bucket=<yourFaasOSSBucketName>
```

示例代码

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

> **说明:**
> 如果同一主账户下有多个子账户，建议子账户间共享一个OSS bucket，以避免重复修改或覆盖授权策略。

查看授权策略

`faascmd list_policy` 命令用来查看指定的OSS bucket是否已添加到相应的授权策略（faasPolicy）里。

命令格式

```
faascmd list_policy
```

示例代码

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

> **说明:**
> 请关注您的OSS Bucket和OSS Bucket/compiling_logs是否出现在列出的策略信息中。

删除授权策略

`faascmd delete_policy` 命令用于删除授权策略（faasPolicy）。

命令格式

```
faascmd delete_policy
```

示例代码

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

> **说明:**
> 如果同一主账户下有多个子账户，建议您去RAM控制台操作，以避免误删授权策略。

查看OSS Bucket下所有的objects

`faascmd list_objects` 命令用于查看用户OSS Bucket下所有的objects。

命令格式

```
faascmd list_objects
```

示例代码

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

说明:

您可以配合grep命令筛选出您想要的文件。例如: `faascmd list_objects | grep "xxx"` 。

上传原始编译文件

`faascmd upload_object` 命令用于将本地编译的原始文件上传到用户指定的OSS bucket中。

命令格式

```
faascmd upload_object --object=<newFileNameinOSSBucket> --file= <
your_file_path>/fileNameYouWantToUpload
```

示例代码

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

说明:

- 如果需上传的文件在当前目录下，则无需提供路径。
- intel fpga的本地编译原始文件为.gbs格式；xilinx fpga的本地编译原始文件为脚本处理后得到的tar包。

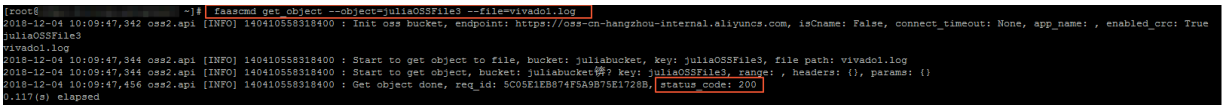
下载OSS Bucket中的object

`faascmd get_object` 命令用来下载OSS Bucket中指定的object。

命令格式

```
faascmd get_object --obejct=<yourObjectName> --file=<your_local_path>/
<yourFileName>
```

示例代码



说明:

如果您不提供路径，则默认下载到当前文件夹。

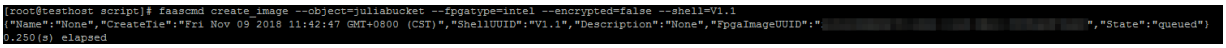
新建fpga镜像

`faascmd create_image`**命令用来提交制作fpga镜像的请求。请求成功时，返回fpga imageuuid。**

命令格式

```
faascmd create_image --object=<yourObjectName>
--fpgatype=<intel/xilinx>  --encrypted=<true/false>
--kmskey=<key/如果encrypted为true，必须；否则可选>
--shell=<Shell Version/必选> --name=<name/可选>
--description=<description/可选> --tags=<tags/可选>
```

示例代码



查看fpga镜像

`faascmd list_images`**命令用于查看用户制作的所有fpga镜像的信息。**

命令格式

```
faascmd list_images
```

示例代码

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

说明:

每个子账户最多允许保留10个fpga镜像。

删除fpga镜像

`faascmd delete_image`命令用于删除fpga镜像。

命令格式

```
faascmd delete_image --imageuuid=<yourImageuuid>
```

示例代码

```
[root@testhost script]# faascmd delete_image --imageuuid=
{"Status":200,"FpgaImageUUID":"i                                ","Message":"delete succeed!"}
0.143(s) elapsed
```

下载fpga镜像

`faascmd download_image`命令用于提交下载fpga镜像的请求。

命令格式

```
faascmd download_image  --instanceId=<yourInstanceId>
--fpgauuid=<yourfpgauuid> --fpgatype=<intel/xilinx>
--imageuuid=<yourImageuuid> --imagetype=<afu>
```

```
--shell=<yourImageShellVersion>
```

示例代码

```
faascmd download_image --instanceId=XXXXX --fpgauuid=XXXX --fpgatype=
intel --imageuuid=XXXX
```

查看fpga镜像下载状态

`faascmd fpga_status`命令用于查看当前fpga板卡状态或fpga镜像的下载进度。

命令格式

```
faascmd fpga_status --fpgauuid=<fpgauuid> --instanceId=<instanceId>
```

示例代码

```
[root@testhost script]# faascmd fpga_status --fpgauuid=                    --instanceId=
{"shellUUID":"V1.0","FpgaImageUUID":"                                      ","FpgaUUID":"                    ",
askStatus":"invalid","Encrypted":"false"}
0.310(s) elapsed
```

发布fpga镜像

`faascmd publish_image` 命令用来提交发布fpga镜像的请求。

命令格式

```
faascmd publish_image --imageuuid=<yourImageuuid> --imageid=<
yourFPGAImageid>
```

说明:

- `imageuuid` 是您要发布到云市场的镜像id。您可以通过 `faascmd list_images` 命令查看。
- `imageid` 是fpga镜像id。您可以通过ECS控制台的实例详情页查看。

查看fpga实例的信息

`faascmd list_instances`命令用于获取fpga实例的基本信息，包括实例id、fpga板卡信息和shell版本。

命令格式

```
faascmd list_instances --instanceId=<yourInstanceId>
```

示例代码

```
[root@testhost script]# faascmd list_instances --instanceId=
{
  "Instances": {
    "instance": [
      {
        "DeviceBDF": "05:00.0",
        "FpgaStatus": "invalid",
        "FpgaType": "intel",
        "FpgaUUID": "",
        "InstanceId": "",
        "ShellUUID": "V1.1"
      }
    ]
  }
}
0.275(s) elapsed
```

9.5.5 faascmd工具FAQ

本文介绍使用faascmd工具时常见的问题与解决办法。

常见问题

- Name Error:global name'ID' is not defined.

  原因：faascmd没有获取到您的AccessKeyID或AccessKeySecret信息。

  解决办法：执行`faascmd config`命令，此命令执行后，会将您输入的AccessKeyID和AccessKeySecret信息保存在文件`/root/.faascredentials`中。

- HTTP Status:403 Error:RoleAccessError. You have no right to assume this role.

  原因：faascmd没有获取到roleArn信息，或者roleArn信息与当前的AccessKeyID和AccessKeySecret信息不属于同一个账户。

  解决办法：检查`/root/.faascredentials`文件是否包含以下信息。

  ```
  [FaaSCredentials]
  accessid=xxxxxxxxxx
  accesskey=xxxxxxxxxxxxxxxxxxxxxxx
  [Role]
  role=acs:ram::1234567890123456:role/xxxxxx
  [OSS]
  bucket=xxxx
  ```

  说明:

  - 如果上述信息存在，确认该role信息与AccessKeyID/AccessKeySecret的云ID是否一致。
  - 如果上述信息不存在，执行 `faascmd auth bucket=xxxx` 命令授权。

- HTTP Status: 404 Error: EntityNotExist. Role Error. The specified Role not exists .

  原因：您的云账户下的faasrole角色不存在。

  解决办法： 登陆RAM控制台查看faasrole角色是否存在。

  - 如果faasrole角色不存在，您需要执行 `faascmd config` 和 `faascmd auth` 命令创建该角色并为其授权。
  - 如果faasrole角色存在，请提交工单处理。
- SDK.InvalidRegionId. Can not find endpoint to access.

  原因：获取不到faas服务的endpoint地址。

  解决办法：您需要逐项检查是否满足以下配置。

  - 运行`python -V`命令检查python版本是否为2.7.x。
  - 运行`which python`命令检查python的默认安装路径是否为 `/usr/bin/python` 。
  - 运行`cat /usr/lib/python2.7/site-packages/aliyunsdkcore/__init__.py`命令检查aliyunsdkcore版本是否为2.11.0及以上。

    说明:

    如果aliyunsdkcore版本号低于2.11.0，您需要运行`pip install --upgrade aliyun-python-sdk-core`命令升级至最新版本。
- 下载镜像时返回 HTTP Status:404 Error:SHELL NOT MATCH. The image Shell is not match with fpga Shell!Request ID:D7D1AB1E-8682-4091-8129-C17D54FD10D4

  原因：要下载的fpgaImage和指定fpga上的shell版本不匹配。

  解决办法：您需要按下列步骤逐项检查。

  - 运行`faascmd list_instances --instance=xxx`命令检查当前fpga的shell版本号。
  - 运行`faascmd list_images`命令检查指定的fpgaImage的shell版本号。

    说明:

    ■ 如果以上两个shell版本号不同，您需要重新制作一个与fpga的shell版本号相同的fpgaImage，然后下载。

    ■ 如果确定两个shell版本一致，请提交工单。

- 下载镜像时返回HTTP Status:503 Error:ANOTHER TASK RUNNING . Another task is running,user is allowed to take this task half an hour Request ID: 5FCB6F75-8572-4840-9BDC-87C57174F26D

  原因：您之前提交的下载请求异常失败或中断导致fpga的状态还停留在operating状态。

  解决办法：建议您等待10分钟，直至下载任务自动结束，然后再次提交下载镜像请求。

   说明:
  如果问题仍旧没有解决，请提交工单。

- 运行faascmd list_images命令时，发现镜像状态是failed。

  解决方法：您可以通过以下方式获取编译日志，以定位相关错误。

```
faascmd list_objects|grep vivado
faascmd get_object --obejct=<yourObjectName> --file=<your_local_path
>/vivado.log  #路径选填，默认下载到当前文件夹。
```

常见错误码

faascmd命令	API名字	错误信息	错误描述	错误码
适用所有命令	适用所有API	PARAMETER INVALIDATE	输入参数有误。	400
适用所有命令	适用所有API	InternalError	未知错误，提交工单。	500
auth	auth	NoPermisson	没有访问某个openAPI的权限。	403
create_image	CreateFpgaImage	IMAGE NUMBER EXCEED	镜像列表不能超过10个镜像，删除不需要的镜像即可。	401
		FREQUENCY ERROR	目前提交镜像请求的时间间隔为30min一次。	503
		SHELL NOT SUPPORT	输入的shell版本不支持，请检查shell版本是否正确。	404
		EntityNotExist.RoleError	用户账户没有创建faasRole。	404
		RoleAccessError	用户输入的roleArn为空，或者roleArn信息与AccessKey ID/AccessKey Secret不属于同一个云账号。	403

faascmd命令	API名字	错误信息	错误描述	错误码
		InvalidAccessKeyIdError	AccessKey ID/AccessKey Secret不合法。	401
		Forbidden.KeyNotFoundError	找不到指定的KMS key，请登陆KMS控制台检查输入的keyId是否存在。	503
		AccessDeniedError	faas admin 账户没有访问当前bucket的权限。	
		OSS OBJECT NOT FOUND	指定的oss bucket/object不存在，或者不具备访问权限。	404
delete_image	DeleteFpgaImage	IMAGE NOT FOUND	指定的fpgaImage找不到。	400
list_instances	DescribeFpgaInstances	NOT AUTHORIZED	指定的instance不存在或者不属于当前的云账户。	401
		RoleAccessError	用户输入的roleArn为空，或者roleArn信息与AccessKey ID/AccessKey Secret不属于同一个云账号。	403
		INSTANCE INVALIDATE	指定的instance不属于fpga实例。如果确定是fpga实例，请提交工单。	404
fpga_status	DescribeLoadTaskStatus	NOT AUTHORIZED	找不到指定的instanceId，请检查输入参数。	401
		FPGA NOT FOUND	找不到指定fpgauuid，请检查输入参数。	404
download_image	LoadFpgaImage	ANOTHER TASK RUNNING	之前提交的下载镜像任务还在operating状态。	503
		IMAGE ACCESS ERROR	指定的image不属于当前云账户。	401
		YOU HAVE NO ACCESS TO THIS INSTANCE	指定的instance不属于当前的云账户。	401
		IMAGE NOT FOUND	指定的fpgaImage找不到。	404
		FPGA NOT FOUND	指定的fpga找不到。	404

faascmd命令	API名字	错误信息	错误描述	错误码
		SHELL NOT MATCH	镜像的shell版本和指定的fpga上的shell版本不匹配。	404
		RoleAccess Error	用户输入的roleArn为空，或者roleArn信息与AccessKey ID/AccessKey Secret不属于同一个云账号。	403
		Image not in success state	指定的image不是success状态，只有状态为success的image才可以下载。	404
publish_image	PublishFpgaImage	FPGA IMAGE STATE ERROR	指定的image不是success状态。	404
		FPGA IMAGE NOT FOUND	指定的image没有找到或者不属于当前用户。	404

10 磁盘缩容

由于目前云服务器 ECS 不支持系统盘或者数据盘缩容，如果您有磁盘缩容的需求，可用通过阿里云迁云工具达成目的。

前提条件

在开始本教程前，请确认您已完成以下操作：

- 当磁盘挂载的是 Linux 实例时，您需要预先在实例内安装远程数据同步工具 rsync。
  - CentOS 实例：运行`yum install rsync -y`
  - Ubuntu 实例：运行`apt-get install rsync -y`
  - Debian 实例：运行`apt-get install rsync -y`
  - 其他发行版：参考发行版官网安装相关的文档
- 您需要预先在控制台创建 AccessKey， 用于输出到配置文件*user_config.json*里。具体步骤，请参见创建 AccessKey。

  说明:
  由于 AccessKey 权限过大，为防止数据泄露，建议您创建RAM用户子账号，并使用 RAM 用户子账号创建 AccessKey。具体操作，请参见创建 RAM 用户子账号和创建 AccessKey。
- 其他更多前提条件和限制条件，请参见使用迁云工具迁移服务器至阿里云。

背景信息

迁云工具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的磁盘缩容。

迁云工具可以根据您的 ECS 实例重新创建一份自定义镜像，在创建过程中通过重新指定磁盘大小，以达到缩容的目的。除了将目标对象换成了 ECS 实例之外，磁盘缩容和迁云这两种场景的工具的使用方法和使用限制完全一致。由于使用对象为虚拟化的 ECS 实例，还可以降低报错机率，更加高效。

然而，这种缩容方式，会引起原有 ECS 实例的部分属性发生变化，例如，实例 ID（`InstanceId`）和 公网 IP。如果您的实例为专有网络（VPC）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，建议使用弹性公网IP（EIP）或者对公网 IP 依赖程度较轻的用户使用该方式缩容。

操作步骤

1. 使用管理员/root 账号远程连接到目标 ECS 实例。具体步骤，请参见远程连接。

2. 单击此处下载阿里云迁云工具 ZIP 压缩包。
3. 解压迁云工具ZIP压缩包，并进入对应操作系统及版本的客户端文件目录找到配置文件 *user_config.json*。
4. 完成配置。详情请参见步骤2：配置迁移源和迁移目标。

   该配置文件 Linux Shell 显示效果如下图所示。

   ```
   "access_id": "",
   "secret_key": "",
   "region_id": "",
   "image_name": "",
   "system_disk_size":    ,
   "platform": "",
   "architecture": "",
   "data_disks": [],
   "bandwidth_limit": 0
   ```

   在磁盘缩容的场景中，您需要重点关注以下参数：

   - `system_disk_size`：该参数可以置为缩容系统盘的预期数值，单位为 GB，该值不能小于系统盘实际使用空间大小。
   - `data_disks`：该参数可以置为缩容数据盘的预期数值，单位为 GB，该值不能小于数据盘实际使用空间大小。

   > 说明：
   >
   > - 当 Linux 实例自带数据盘时，即使您不考虑缩容数据盘，也需要配置参数`data_disks`。
   > - 当 Windows 实例自带数据盘时，如果没有缩容数据盘的需求，可以不配置参数 `data_disks`。

5. 执行客户端主程序*go2aliyun_client.exe*。
   - Windows 实例：右击*go2aliyun_client.exe*，选择以管理员身份运行。
   - Linux 实例：

     a. 运行`chmod +x go2aliyun_client`赋予客户端可执行权限。

     b. 运行`./ go2aliyun_client`运行客户端。

6. 等待运行结果。

   - 当出现`Goto Aliyun Finished!`提示时，前往ECS 控制台镜像详情页查看经过缩容后的自定义镜像。如果自定义镜像已生成，您可以释放原实例，然后使用生成的自定义镜像创建ECS实例，创建完成后，磁盘缩容工作已完成。如何创建，请参见创建 ECS 实例。
   - 当出现`Goto Aliyun Not Finished!`提示时，检查同一目录下*Logs*文件夹下的日志文件排查故障，详情请参见排查故障。

     修复问题后，重新运行迁云工具即可恢复缩容工作，迁云工具会从上一次执行的进度中继续迁云，无需重头开始。

相关文档

什么是迁云工具

使用迁云工具迁移服务器至阿里云

11 ECS状态变化事件的自动化运维最佳实践

本文通过实践案例为您介绍云监控如何利用MNS消息队列实现自动化处理ECS状态变化事件。

背景信息

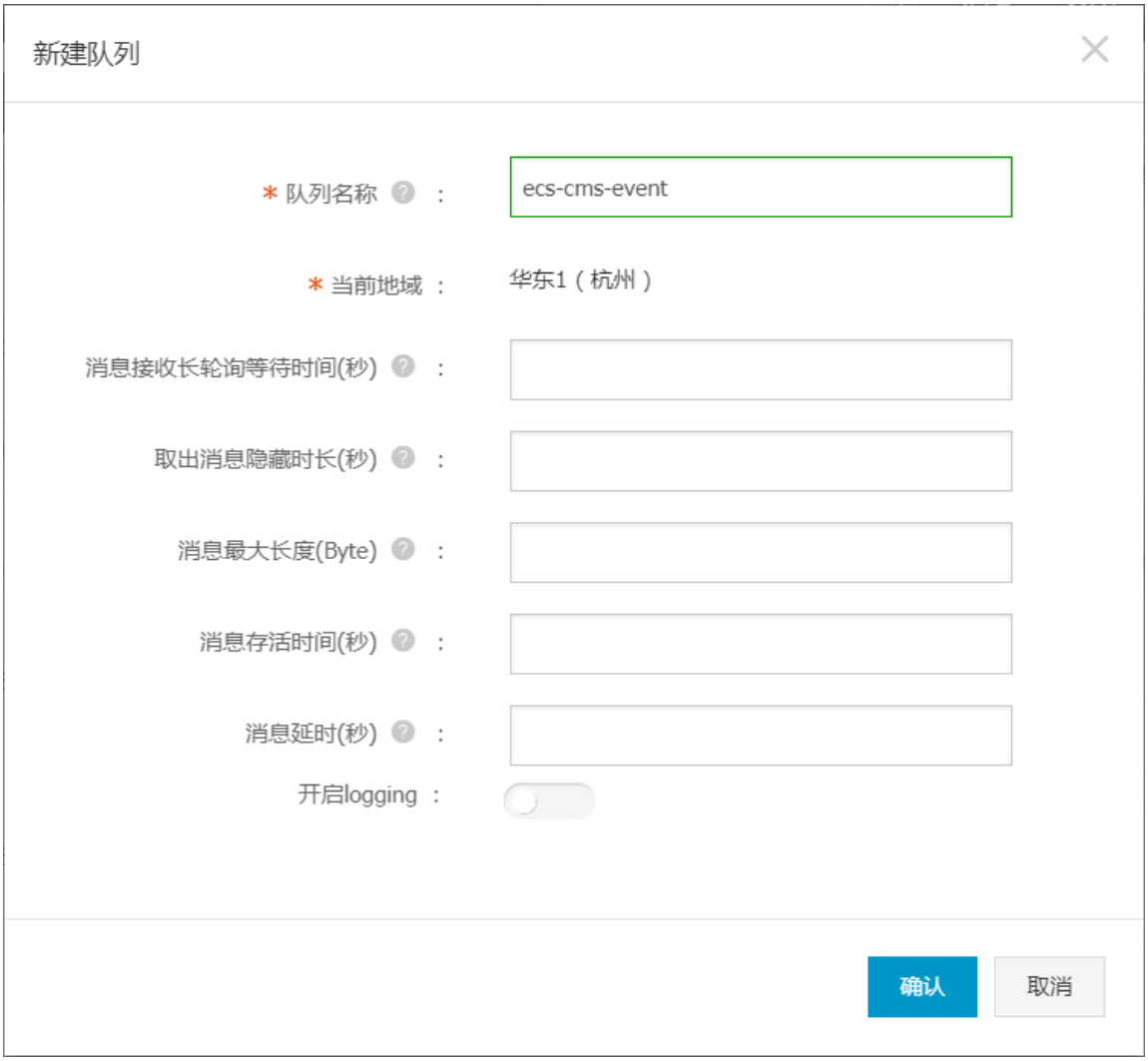
阿里云ECS在已有的系统事件的基础上，通过云监控新发布了状态变化类事件和抢占型实例的中断通知事件。每当ECS实例的状态发生变化的时候，都会触发一条ECS实例状态变化事件。这种变化包括您在控制台/OpenAPI/SDK操作导致的变化，也包括弹性伸缩或欠费等原因而自动触发的变化，还包括因为系统异常而触发的变化。

云监控以前发布的系统事件，主要针对告警后人工介入的场景，而这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等场景。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。本文将为您介绍利用MNS消息队列自动化处理ECS事件的三种最佳实践。

自动化处理ECS状态变化事件的准备工作

- 创建消息队列

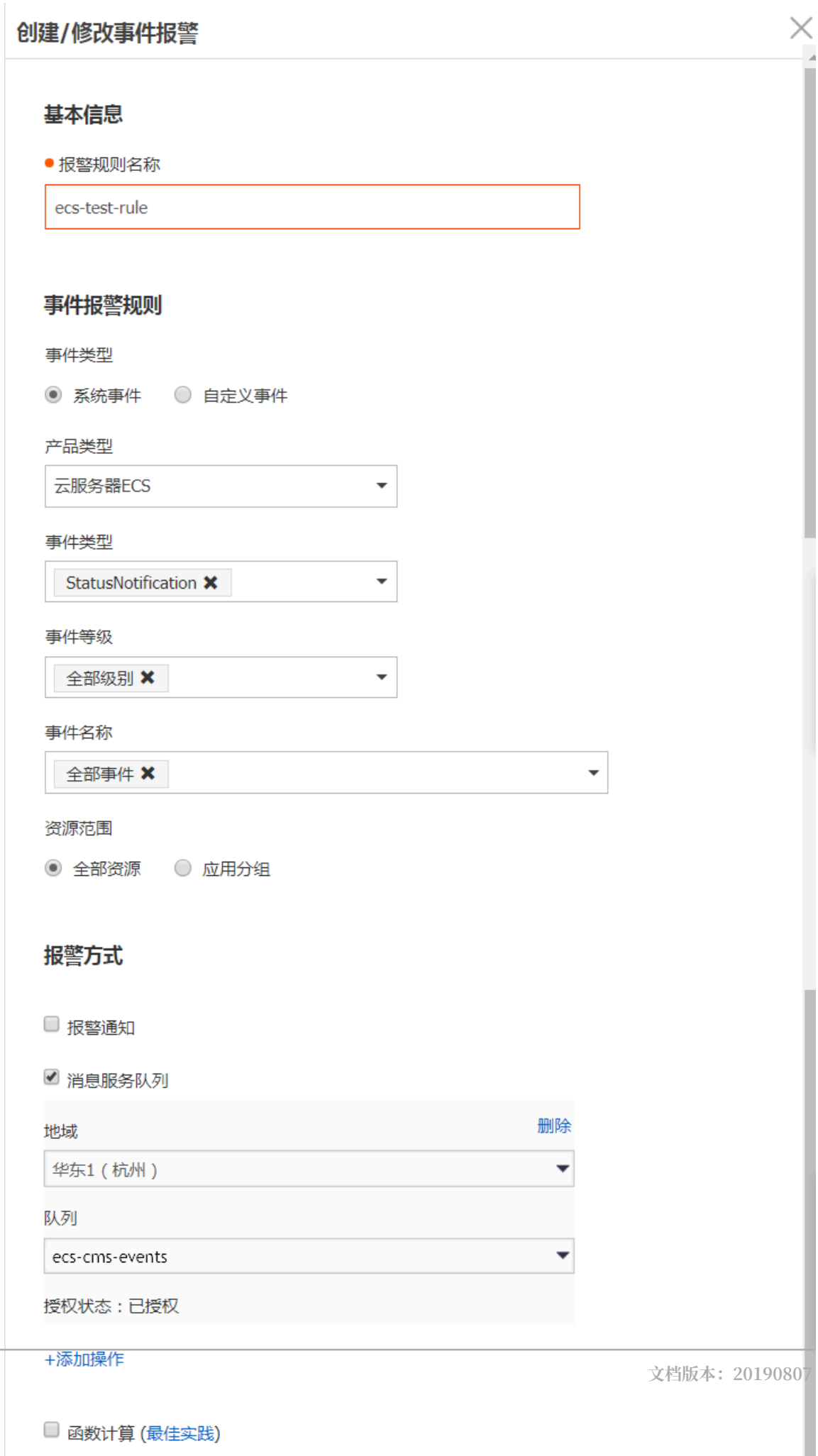
1. 登录MNS控制台。
2. 在队列列表页面，选择地域，单击右上角的创建队列，进入新建队列页面。



3. 输入队列的名称（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

  1. 登录云监控控制台。
  2. 单击左侧导航栏中的事件监控，进入事件查询页面
  3. 单击报警规则页签，然后单击右上角的创建事件报警，弹出创建/修改事件报警对话框。


创建/修改事件报警
基本信息
报警规则名称
ecs-test-rule
事件报警规则
事件类型
系统事件
自定义事件
产品类型
云服务器ECS
事件类型
StatusNotification
事件等级
全部级别
事件名称
全部事件
资源范围
全部资源
应用分组
报警方式
报警通知
消息服务队列
地域
删除
华东1（杭州）
队列
ecs-cms-events
授权状态：已授权
+添加操作
函数计算 (最佳实践)

4. 在基本信息区域，填写报警规则名称，例如如“ecs-test-rule”。
5. 设置事件报警规则：选择事件类型为系统事件。
   - 产品类型、事件等级、事件名称：产品类型选择云服务器ECS，事件类型选择StatusNotification，其余按照实际情况填写。
   - 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；选择应用分组时，只有指定分组内的资源发生相关事件时，才会发送通知。
6. 在报警方式中，选择消息队列，然后选择地域和队列（例如ecs-cms-event）。
7. 完成以上设置后，单击确定按钮即可完成创建事件报警规则。

· 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

云监控会把云服务器ECS所有的状态变化事件都投递到MNS里面，接下来我们需要通过编写代码从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状态变化事件把所有ECS的生命周期记录在自己的数据库或者日志里。每当创建ECS时，会首先发送一个Pending事件，每当释放ECS时，会最后发送一个Deleted事件。我们需要对这两种事件进行记录。

1. 编辑一个Conf文件。需包含mns的endpoint（可以登录MNS的控制台，在队列列表页，单击获取Endpoint得到）、阿里云的access key和secrect、region id（例如cn-beijing）以及mns queue的名字。

```
class Conf:
    endpoint = 'http://<id>.mns.<region>.aliyuncs.com/'
    access_key = '<access_key>'
    access_key_secret = '<access_key_secrect>'
    region_id = 'cn-beijing'
    queue_name = 'test'
    vsever_group_id = '<your_vserver_group_id>'
```

2. 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_secret)
        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 receive message.', self.queue_name)
                else:
                    logging.error('No Message, continue waiting')


class BasicListener(object):
    def process(self, event):
        pass
```

上述代码只是对MNS消息进行拉取，调用Listener消费消息之后删除消息，后面的实践也会用到。

3. **注册一个Listener进消费指定事件。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 == 'Panding':
```

```
            logging.info(f'The instance {resource_id} state is {
state}')
        elif state == 'Deleted':
            logging.info(f'The instance {resource_id} state is {
state}')
```

Main函数可以这么写：

```
mns_client = MNSClient()

mns_client.regist_listener(ListenerLog())

mns_client.run()
```

实际生产环境下，可能需要把事件存储在数据库里，或者利用SLS日志服务，方便后期的搜索和审计。

实践二：ECS的关机自动重启

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。

为了实现这一目的，我们复用实践一里面的MNS Client，添加一个新的Listener。当收到Stopped事件的时候，对该ECS执行一个Start命令。

```
# -*- 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/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。

我们还是复用实践一的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_gr
oup_id)
        self.request.add_query_param('BackendServers',
                                     "[{'ServerId':'" + instance_id +
 "','Port':'80','Weight':'100'}]")
        response = self.client.do_action_with_exception(self.request)
        return str(response, encoding='utf-8')


class ListenerSLB(BasicListener):
    def __init__(self, vsever_group_id):
        self.slb_caller = SLBClient()
        self.vsever_group_id = Conf.vsever_group_id

    def process(self, event):
        detail = event['content']
        instance_id = detail['instanceId']
        if detail['action'] == 'delete':
```

```
            self.slb_caller.remove_vserver_group_backend_servers(self.
vsever_group_id, instance_id)
```

注意:

抢占型实例释放告警的event name与前面不同，应该是“Instance:PreemptibleInstanceInterruption”， mns_client.regist_listener(ListenerSLB(Conf.vsever_group_id), 'Instance:PreemptibleInstanceInterruption')

在实际生产环境下，您可能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。