高性能计算

用户指南

为了无法计算的价值 | 阿里云

用户指南

购买须知

ECS设置

安全组是一个逻辑上的分组，这个分组是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。

安全组，类似防火墙，用于设置单台或多台云服务器的网络访问控制，它是重要的安全隔离手段。

每个实例至少属于一个安全组，在创建的时候就需要指定。

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。更多详情

ECS安全组配置

目前，ECS安全组有三种方式

- 配置方式1：阿里云管理控制台配置 目前，非VPC用户无法在阿里云控制台看到安全组配置，可提交工单申请开通，注明“开通控制台显示‘本实例安全组’”。

在开通安全组之后，用户可以在ECS实例控制台看到安全组，如图:


云服务器 ECS
概览
实例
磁盘
快照
镜像
标签管理
安全组
安全组列表
亚太（
专有网络ID
安全组ID/名称
sg-28shvdck8
sg-28shvdck8
删除

点击右上角 “创建安全组” ：

填写安全组名称和备注，选择网络类型:

安全组创建成功后，需要配置规则，点击安全组的“配置规则”链接:

添加安全组配置规则时，规则方向需要选“入方向”，网卡类型选“内网”：

规则创建成功后，可以看到此时的安全组还没有应用到任何ECS实例，如图显示，相关实例是0:

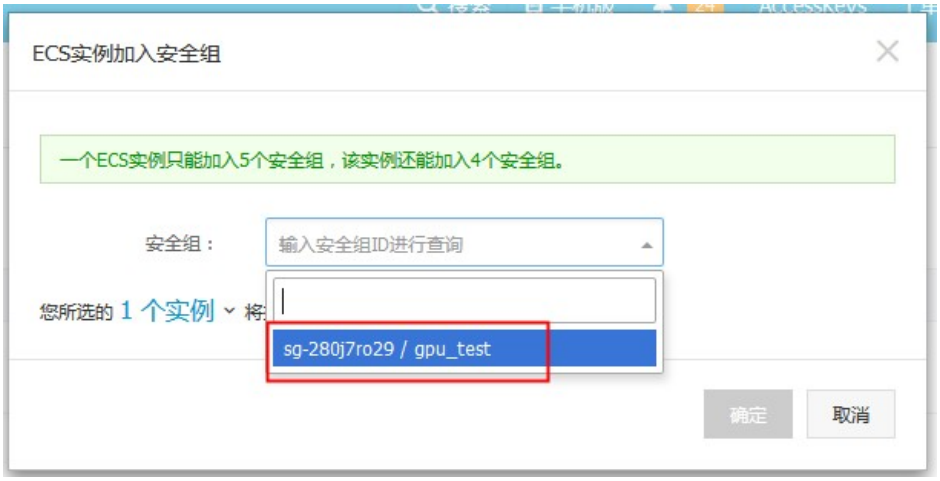
下面，还需要将安全组绑定到ecs实例，才能使ecs按此安全组规则生效。

选择ecs实例的“安全组配置”选项


付费方式(全部)
操作
IB
包年包月
15-11-25 00:00到期
管理
续费
变配
更多
每页显示：20条
启动
停止
重启
重置密码
修改信息
连接管理终端...
连接帮助
重新初始化磁盘
更换系统盘
购买相同配置
编辑标签
安全组配置
修改私网IP

点击右上角“加入安全组”：

在弹出对话框中，选择刚刚创建的安全组:



此时返回安全组列表，可以看到相关实例的值已经变为1:

此时测试从配置的网段访问此ecs的特定端口，即可访问。

- 配置方式2：命令行jar包配置 首先下载jar包到一个有java环境的linux上。执行如下命令，进入交互模式：

$ java -jar ~/aliyuncs.jar -id <您的Access Key ID> -secret <您的Access KeySecret> -service ECS:2014-05-26

其中，aliyuncs.jar为上面下载的文件解压得到，2014-05-26为ECS api版本，此处可以指定为2014-05-26

然后创建安全组实例，如下命令，RegionId为ecs所在城市，Description为安全组名称

@ECS:/>CreateSecurityGroup RegionId=cn-qingdao,Description=gpu_test

创建成功后，会返回安全组实例名，当前操作是sg-28fcdqtel，接着可以给安全组赋予更多的属性。下例是实现10.239.23.101/28网段的所有机器可以访问当前ecs的80端口。

@ECS:/>AuthorizeSecurityGroup SecurityGroupId=sg-28fcdqtel,IpProtocol=TCP,PortRange=80/80,SourceCidrIp=10.239.23.101/28,NicType=intranet,RegionId=cn-qingdao

完善安全组实例属性后，就可以将安全组实例和ecs实例进行绑定。

@ECS:/>JoinSecurityGroup SecurityGroupId=sg-28fcdqtel,InstanceId=i-28weu9ksr

以上步骤操作后，即可实现10.239.23.101/28网段的所有主机访问当前ecs的80端口。

除以上操作外，还有以下一些相关操作。

列出所有安全组

@ECS:/>DescribeSecurityGroups RegionId=cn-qingdao

列出当前安全组的所有属性

@ECS:/>DescribeSecurityGroupAttribute SecurityGroupId=sg-28fcdqtel,NicType=intranet,RegionId=cn-qingdao

解除当前安全组和当前ecs实例的绑定

@ECS:/>LeaveSecurityGroup SecurityGroupId=sg-28fcdqtel,InstanceId=i-28weu9ksr

删除安全组

@ECS:/>DeleteSecurityGroup SecurityGroupId=sg-28fcdqtel,RegionId=cn-qingdao

授权从当前ecs向外访问的安全组规则

@ECS:/>AuthorizeSecurityGroupEgress SecurityGroupId=sg-28fcdqtel,IpProtocol=TCP,PortRange=1/65535,DestCidrIp=0.0.0.0/0,NicType=intranet,RegionId=cn-qingdao

更多信息可以参考Jar包实践

- 配置方式3：API方式配置 详见ECS API 使用文档，使用方式较复杂，不推荐使用。

注：

当前的RegionId，可参考：

可选区域	RegionId

华东1	cn-hangzhou
华东2	cn-shanghai
华北1	cn-qingdao
华北2	cn-beijing
香港	cn-hongkong
华南1	cn-shenzhen
美国硅谷	us-west-1

阿里云HPC物理机本身不能访问外网，只能通过ECS正向代理访问。

本文档将指导用户如何设置代理服务器。

1. 确定IP地址

用户应首先确认这几个IP地址：

ECS外网IP（不便于透露，本文用XXX.XXX.XXX.XXX表示）和内网IP（实验用10.10.10.10）

HPC物理机内网IP（实验用10.239.23.4）

2. 登录ECS跳板机

用户可以用PUTTY工具（Windows环境）或SSH命令（Linux环境）登录ECS，注意应使用ECS外网IP登入。

```
ssh -l login_name XXX.XXX.XXX.XXX（ECS外网IP）
```

登录成功后，可以在ECS跳板机上用SSH命令登录HPC物理机：

```
ssh -l root 10.239.23.4（HPC物理机内网IP）
```

3. ECS跳板机上部署代理服务器squid

这里选择squid，因为它不仅支持访问HTTP的服务还支持访问HTTPS的服务。

3.1 安装squid

重新开一个终端，登录到ECS跳板机。

直接用yum安装：

```
```

```
yum install squid
```

默认情况下安装位置在 /usr/sbin/squid

3.2 编辑squid配置文件

用root权限打开 /etc/squid/squid.conf 文件，首先把不需要的内网地址全部注释上，在上面增加一行，添加自己的HPC物理机IP地址，然后把不需要开放的端口注释上，只留下80和443端口，然后增加一行：access_log /var/log/squid/access.log用来记录访问情况。

修改后如下：

```
......

acl localnet src 10.239.23.4     # 这里增加一行，添加自己的HPC物理机IP地址
#acl localnet src 10.0.0.0/8     # RFC1918 possible internal network，注释上
#acl localnet src 172.16.0.0/12  # RFC1918 possible internal network，注释上
#acl localnet src 192.168.0.0/16 # RFC1918 possible internal network，注释上
#acl localnet src fc00::/7       # RFC 4193 local private network range，注释上
#acl localnet src fe80::/10      # RFC 4291 link-local (directly plugged) machines，注释上

acl SSL_ports port 443
acl Safe_ports port 80          # http服务端口打开
#acl Safe_ports port 21         # ftp可以根据情况是否打开
acl Safe_ports port 443         # https服务端口打开
#acl Safe_ports port 70         # gopher，注释上
#acl Safe_ports port 210        # wais，注释上
#acl Safe_ports port 1025-65535  # unregistered ports，注释上
#acl Safe_ports port 280        # http-mgmt，注释上
#acl Safe_ports port 488        # gss-http，注释上
#acl Safe_ports port 591        # filemaker，注释上
#acl Safe_ports port 777        # multiling http，注释上
acl CONNECT method CONNECT

......

# And finally deny all other access to this proxy
http_access deny all                    # 除上面允许的外，其他一律禁止访问

# Squid normally listens to port 3128
http_port 3128                          # squid默认监听端口号，可以修改成别的端口号

# Added
access_log /var/log/squid/access.log          # 这里增加一行，用于监控访问记录
```

保存该文件。

3.3 启动squid

用root权限运行：sudo service squid start

查看squid运行状态：

```
sudo service squid status
squid (pid 31659) is running...
```

3.4 设置ECS防火墙

出于节省流量和安全考虑，需要在ECS上设置防火墙规则，将除了HPC物理机之外的所有访问3128端口的请求都挡在防火墙外。

步骤如下：

3.4.1 开启防火墙

```
CentOS 6: service iptables start
CentOS 7: systemctl start firewalld
```

3.4.2 添加防火墙规则

首先允许HPC物理机IP地址（本文用10.239.23.4，请根据实际情况修改）访问3128端口:

```
iptables -I INPUT -s 10.239.23.4  -p TCP --dport 3128 -j ACCEPT
```

端口3128要和3.2节squid配置文件中的端口设置相同。

然后禁止所有访问3128端口的tcp连接：

```
iptables -A INPUT -p TCP --dport 3128 -j DROP
```

然后保存iptables设置：

```
service iptables save
```

查看规则是否生效：

```
iptables -L -n
```

可以看到新增了两条规则：

```
target     prot opt source               destination
ACCEPT     tcp  --  10.239.23.4          0.0.0.0/0           tcp dpt:3128
DROP       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:3128
```

4. 在HPC物理机上设置代理

回到HPC物理机终端，进行代理设置。

最简单的方式是使用环境变量，假设ECS内网IP为10.10.10.10（用户需要自行替换为真实ECS内网IP），则可以执行：

export http_proxy=http://10.10.10.10:3128

export https_proxy=http://10.10.10.10:3128

注意: 这里的代理服务器端口设置应该和ECS跳板机上squid.conf中监听端口一致，另外必须是ECS内网地址。

也可以将上述语句放入/etc/profile或~/.bashrc实现登录HPC物理机时自动配置代理服务器。

5. 测试

在HPC物理机上使用wget、git clone和yum install测试结果如下：

5.1 测试http的访问

```
wget http://www.cmake.org/files/v3.3/cmake-3.3.1.tar.gz
--2015-09-29 18:12:52--  http://www.cmake.org/files/v3.3/cmake-3.3.1.tar.gz
Connecting to 120.26.218.226:3128... connected.
Proxy request sent, awaiting response... 200 OK
Length: 6577869 (6.3M) [application/x-gzip]
Saving to:  'cmake-3.3.1.tar.gz'

100%[==========================================================================
==========================================================>] 6,577,869
28.7KB/s   in 3m 55s

2015-09-29 18:16:47 (27.4 KB/s) -  'cmake-3.3.1.tar.gz'  saved [6577869/6577869]
```

可以查看/var/log/squid/access.log文件，找到上述下载的log:

```
1443530801.311   7144 10.239.23.4 TCP_MISS/200 132972 GET http://www.cmake.org/files/v3.3/cmake-3.3.1.tar.gz -
DIRECT/66.194.253.19 application/x-gzip
```

5.2 测试https的访问

```
wget https://codeload.github.com/gflags/gflags/tar.gz/v2.1.2
--2015-09-29 18:18:13--  https://codeload.github.com/gflags/gflags/tar.gz/v2.1.2
Connecting to 120.26.218.226:3128... connected.
Proxy request sent, awaiting response... 200 OK
Length: 95716 (93K) [application/x-gzip]
Saving to:  'v2.1.2'
```

```
100%[=======================================================================
=============================================================>] 95,716
78.7KB/s   in 1.2s

2015-09-29 18:18:17 (78.7 KB/s) -  ‘v2.1.2’  saved [95716/95716]
```

可以查看/var/log/squid/access.log文件，找到上述下载的log:

```
1443531073.042  11238 10.239.23.4 TCP_MISS/200 101394 CONNECT codeload.github.com:443 -
DIRECT/192.30.252.145 -
```

5.3 测试git clone

```
git clone https://github.com/gflags/gflags.git
Cloning into 'gflags'...
remote: Counting objects: 1772, done.
remote: Total 1772 (delta 0), reused 0 (delta 0), pack-reused 1772
Receiving objects: 100% (1772/1772), 1.27 MiB | 493.00 KiB/s, done.
Resolving deltas: 100% (1013/1013), done.
```

可以查看/var/log/squid/access.log文件，找到上述下载的log:

```
1443531119.620  29512 10.239.23.4 TCP_MISS/200 1356342 CONNECT github.com:443 - DIRECT/192.30.252.129 -
```

5.4 测试yum

```
# yum install openssl
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirrors.aliyuncs.com
 * extras: mirrors.aliyuncs.com
 * updates: mirrors.aliyuncs.com
Resolving Dependencies
--> Running transaction check
---> Package openssl.x86_64 1:1.0.1e-42.el7.9 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
================================================================================
==============================
 Package                 Arch                 Version                      Repository
Size
================================================================================
================================================================================
==============================
```

```
Installing:
 openssl                  x86_64                1:1.0.1e-42.el7.9             updates
711 k

Transaction Summary
================================================================================
================================================================================
==============================
Install  1 Package

Total download size: 711 k
Installed size: 1.5 M
Is this ok [y/d/N]: y
Downloading packages:
openssl-1.0.1e-42.el7.9.x86_64.rpm                                          | 711
kB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Warning: RPMDB altered outside of yum.
** Found 1 pre-existing rpmdb problem(s), 'yum check' output follows:
authconfig-6.2.8-9.el7.x86_64 has missing requires of /usr/bin/openssl
  Installing : 1:openssl-1.0.1e-42.el7.9.x86_64
1/1
  Verifying  : 1:openssl-1.0.1e-42.el7.9.x86_64
1/1

Installed:
  openssl.x86_64 1:1.0.1e-42.el7.9

Complete!
```

可以查看/var/log/squid/access.log文件，找到上述yum安装的log：

```
1443522032.861    129 10.239.23.4 TCP_MISS/200 728129 GET
http://mirrors.aliyuncs.com/centos/7/updates/x86_64/Packages/openssl-1.0.1e-42.el7.9.x86_64.rpm -
DIRECT/10.143.34.200 application/x-redhat-package-manager
```

通过以上测试，验证了HPC物理机已经可以通过ECS正向代理访问外网。

阿里云GPU物理机本身不能访问外网，只能通过ECS正向代理访问。本文档将指导用户如何设置代理服务器。

1. 确定IP地址

用户应首先确认这几个IP地址： ECS外网IP（不便于透露，本文用XXX.XXX.XXX.XXX表示）和内网IP（实验用10.10.10.10）； GPU物理机内网IP（实验用10.239.23.4）；

2. 登录ECS跳板机

用户可以用PUTTY工具（Windows环境）或SSH命令（Linux环境）登录ECS，注意应使用ECS外网IP登入。

```
ssh -l login_name XXX.XXX.XXX.XXX（ECS外网IP）
```

登录成功后，可以在ECS跳板机上用SSH命令登录GPU物理机：

```
ssh -l root 10.239.23.4（GPU物理机内网IP）
```

3. ECS跳板机上部署代理服务器

这里选择Tengine，它是在NGINX的基础上由淘宝网发起的开源Web服务器项目。用户应注意，NGINX做正向代理服务器是不支持HTTPS连接的，所以客户端只能访问HTTP服务。如果用户需要在物理机上访问HTTPS服务可以选择其他Web服务器做代理。

3.1 安装Tengine

重新开一个终端，登录到ECS跳板机。 获取Tengine源码：

```
wget http://tengine.taobao.org/download/tengine-2.1.1.tar.gz
```

解压：

```
tar zxvf tengine-2.1.1.tar.gz
cd tengine-2.1.1/
```

配置和编译：

```
./configure
make
sudo make install
```

默认情况下安装位置在 /usr/local/nginx/

3.2 编辑Tengine配置文件

用root权限打开 /usr/local/nginx/conf/nginx.conf 文件，在http{}语句块内增加如下内容：("//”后面为注释，真正的conf文件中应删除）

```
server {
resolver 8.8.8.8;
//  设置DNS的IP，可以根据实际情况修改
resolver_timeout 5s;
//  DNS连接超时设置
```

```
listen 0.0.0.0:8080;
// 用于连接客户端的监听端口，也可改为其他端口
access_log  /root/logs/proxy.access.log;
//  连接日志，用于记录所有连接建立的情况
error_log   /root/logs/proxy.error.log;
//  错误日志，用于记录所有错误情况

    location / {
        allow 10.239.23.4;
//  允许接入的物理机内网IP，根据需要设置
        deny all;
//  拒绝为除了上一句指定物理机内网IP之外的所有主机服务
                                    //  以下不需要用户修改，保持默认即可
        proxy_pass $scheme://$host$request_uri;
        proxy_set_header Host $http_host;

        proxy_buffers 256 4k;
        proxy_max_temp_file_size 0;

        proxy_connect_timeout 30;

        proxy_cache_valid 200 302 10m;
        proxy_cache_valid 301 1h;
        proxy_cache_valid any 1m;
    }
}
```

保存该文件。

3.3 启动Tengine

用root权限运行：sudo /usr/local/nginx/sbin/nginx 如果报错，请根据报错信息对3.2节中的nginx.conf配置文件做必要的修改。

3.4 设置ECS防火墙

出于节省流量和安全考虑，需要在ECS上设置防火墙规则，将除了GPU物理机之外的所有访问8080端口的请求都挡在防火墙外。步骤如下：

3.4.1 开启防火墙

```
CentOS6: service iptables start
CentOS7: systemctl start firewalld
```

3.4.2 添加防火墙规则

首先允许GPU物理机IP地址（本文用10.239.23.4，请根据实际情况修改）访问8080端口:

```
iptables -I INPUT -s 10.239.23.4  -p TCP --dport 8080 -j ACCEPT
```

端口8080要和3.2节Tengine配置文件中的端口设置相同。 然后禁止所有访问8080端口的tcp连接：

```
iptables -A INPUT -p TCP --dport 8080 -j DROP
```

查看规则是否生效：

```
iptables -L -n
```

可以看到新增了两条规则：

```
target     prot opt source               destination
ACCEPT     tcp  --  10.239.23.4          0.0.0.0/0           tcp dpt:8080
DROP       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:8080
```

4. 在GPU物理机上设置代理

回到GPU物理机终端，进行代理设置。 最简单的方式是使用环境变量，假设ECS内网IP为10.10.10.10（用户需要自行替换为真实ECS内网IP），则可以执行：

export http_proxy=http://10.10.10.10:8080

注意这里的代理服务器端口设置应该和ECS跳板机上nginx.conf中监听端口（listen 0.0.0.0:8080）相互对应。 也可以将上述语句放入/etc/profile或~/.bashrc实现登录GPU物理机时自动配置代理服务器。

5. 测试

在GPU物理机上使用不同的工具测试外网连接情况，结果如下：

5.1 wget和curl测试

```
$ wget http://tengine.taobao.org/download/tengine-2.1.1.tar.gz
--2015-09-21 16:15:50--  http://tengine.taobao.org/download/tengine-2.1.1.tar.gz
Connecting to 10.10.10.10:8080... connected.
Proxy request sent, awaiting response... 200 OK
Length: 2062650 (2.0M) [application/octet-stream]
Saving to:  'tengine-2.1.1.tar.gz'

100%[======================================>] 2,062,650   1014KB/s   in 2.0s

2015-09-21 16:15:52 (1014 KB/s) -  'tengine-2.1.1.tar.gz'  saved [2062650/2062650]
```

通过以上测试，验证了GPU物理机已经可以通过ECS正向代理访问外网。

5.2 测试yum

```
$ sudo yum install openssl
Loaded plugins: fastestmirror, langpacks
base                                                     | 3.6 kB     00:00
Loading mirror speeds from cached hostfile
 * base: mirrors.aliyuncs.com
 * extras: mirrors.aliyuncs.com
 * updates: mirrors.aliyuncs.com
Resolving Dependencies
--> Running transaction check
---> Package openssl.x86_64 1:1.0.1e-42.el7 will be updated
---> Package openssl.x86_64 1:1.0.1e-42.el7.9 will be an update
--> Processing Dependency: openssl-libs(x86-64) = 1:1.0.1e-42.el7.9 for package: 1:openssl-1.0.1e-42.el7.9.x86_64
--> Running transaction check
---> Package openssl-libs.x86_64 1:1.0.1e-42.el7 will be updated
---> Package openssl-libs.x86_64 1:1.0.1e-42.el7.9 will be an update
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
======
 Package           Arch        Version                  Repository     Size
================================================================================
======
Updating:
 openssl           x86_64      1:1.0.1e-42.el7.9        updates       711 k
Updating for dependencies:
 openssl-libs      x86_64      1:1.0.1e-42.el7.9        updates       949 k

Transaction Summary
================================================================================
======
Upgrade  1 Package (+1 Dependent package)

Total download size: 1.6 M
Is this ok [y/d/N]:y
Downloading packages:
Delta RPMs disabled because /usr/bin/applydeltarpm not installed.
(1/2): openssl-1.0.1e-42.el7.9.x86_64.rpm                  | 711 kB   00:00
(2/2): openssl-libs-1.0.1e-42.el7.9.x86_64.rpm             | 949 kB   00:00
--------------------------------------------------------------------------------
Total                                       5.7 MB/s | 1.6 MB  00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Updating   : 1:openssl-libs-1.0.1e-42.el7.9.x86_64                    1/4
  Updating   : 1:openssl-1.0.1e-42.el7.9.x86_64                         2/4
  Cleanup    : 1:openssl-1.0.1e-42.el7.x86_64                           3/4
  Cleanup    : 1:openssl-libs-1.0.1e-42.el7.x86_64                      4/4
  Verifying  : 1:openssl-libs-1.0.1e-42.el7.9.x86_64                    1/4
  Verifying  : 1:openssl-1.0.1e-42.el7.9.x86_64                         2/4
  Verifying  : 1:openssl-1.0.1e-42.el7.x86_64                           3/4
  Verifying  : 1:openssl-libs-1.0.1e-42.el7.x86_64                      4/4
```

```
Updated:
  openssl.x86_64 1:1.0.1e-42.el7.9

Dependency Updated:
  openssl-libs.x86_64 1:1.0.1e-42.el7.9

Complete!
```

在ECS跳板机上查看Tengine access log文件（/root/logs/proxy.access.log），找到上述与yum安装相关的log如图所示。

```
10.239.23.4 - - [21/Sep/2015:17:26:11 +0800] "GET
http://mirrors.aliyuncs.com/centos/7/os/x86_64/Packages/libcom_err-devel-1.42.9-7.el7.x86_64.rpm HTTP/1.1" 502
690 "-" "urlgrabber/3.10 yum/3.4.3"
10.239.23.4 - - [21/Sep/2015:17:26:11 +0800] "GET
http://mirrors.aliyuncs.com/centos/7/updates/x86_64/Packages/krb5-libs-1.12.2-15.el7_1.x86_64.rpm HTTP/1.1" 502
691 "-" "urlgrabber/3.10 yum/3.4.3"
10.239.23.4 - - [21/Sep/2015:17:26:11 +0800] "GET
http://mirrors.aliyuncs.com/centos/7/updates/x86_64/Packages/krb5-devel-1.12.2-15.el7_1.x86_64.rpm HTTP/1.1"
502 692 "-" "urlgrabber/3.10 yum/3.4.3"
10.239.23.4 - - [21/Sep/2015:17:26:11 +0800] "GET
http://mirrors.aliyuncs.com/centos/7/os/x86_64/Packages/keyutils-libs-devel-1.5.8-3.el7.x86_64.rpm HTTP/1.1" 502
692 "-" "urlgrabber/3.10 yum/3.4.3"
10.239.23.4 - - [21/Sep/2015:17:26:11 +0800] "GET
http://mirrors.aliyuncs.com/centos/7/os/x86_64/Packages/libselinux-devel-2.2.2-6.el7.x86_64.rpm HTTP/1.1" 200
178532 "-" "urlgrabber/3.10 yum/3.4.3"
10.239.23.4 - - [21/Sep/2015:17:26:11 +0800] "GET
http://mirrors.aliyuncs.com/centos/7/os/x86_64/Packages/libsepol-devel-2.1.9-3.el7.x86_64.rpm HTTP/1.1" 200
72300 "-" "urlgrabber/3.10 yum/3.4.3"
10.239.23.4 - - [21/Sep/2015:17:26:11 +0800] "GET
http://mirrors.aliyuncs.com/centos/7/os/x86_64/Packages/libverto-devel-0.2.5-4.el7.x86_64.rpm HTTP/1.1" 200
11776 "-" "urlgrabber/3.10 yum/3.4.3"
10.239.23.4 - - [21/Sep/2015:17:26:11 +0800] "GET http://mirrors.aliyuncs.com/centos/7/os/x86_64/Packages/pcre-
devel-8.32-14.el7.x86_64.rpm HTTP/1.1" 200 488780 "-" "urlgrabber/3.10 yum/3.4.3"
10.239.23.4 - - [21/Sep/2015:17:26:11 +0800] "GET http://mirrors.aliyuncs.com/centos/7/os/x86_64/Packages/zlib-
devel-1.2.7-13.el7.x86_64.rpm HTTP/1.1" 200 50592 "-" "urlgrabber/3.10 yum/3.4.3"
10.239.23.4 - - [21/Sep/2015:17:26:11 +0800] "GET
http://mirrors.aliyun.com/centos/7/os/x86_64/Packages/keyutils-libs-devel-1.5.8-3.el7.x86_64.rpm HTTP/1.1" 200
38232 "-" "urlgrabber/3.10 yum/3.4.3"
10.239.23.4 - - [21/Sep/2015:17:26:11 +0800] "GET
http://mirrors.aliyun.com/centos/7/os/x86_64/Packages/libcom_err-devel-1.42.9-7.el7.x86_64.rpm HTTP/1.1" 200
30804 "-" "urlgrabber/3.10 yum/3.4.3"
10.239.23.4 - - [21/Sep/2015:17:26:12 +0800] "GET
http://mirrors.aliyuncs.com/centos/7/updates/x86_64/Packages/openssl-devel-1.0.1e-42.el7.9.x86_64.rpm
HTTP/1.1" 200 1235792 "-" "urlgrabber/3.10 yum/3.4.3"
10.239.23.4 - - [21/Sep/2015:17:26:12 +0800] "GET
http://mirrors.aliyun.com/centos/7/updates/x86_64/Packages/krb5-devel-1.12.2-15.el7_1.x86_64.rpm HTTP/1.1" 200
655972 "-" "urlgrabber/3.10 yum/3.4.3"
10.239.23.4 - - [21/Sep/2015:17:26:12 +0800] "GET
http://mirrors.aliyun.com/centos/7/updates/x86_64/Packages/krb5-libs-1.12.2-15.el7_1.x86_64.rpm HTTP/1.1" 200
845708 "-" "urlgrabber/3.10 yum/3.4.3"
```

上述安装包所在的yum源为阿里云内网，故ECS代理会通过内网流量获取，这样节省了用户流量费用。

阿里云GPU物理机本身不能被外网直接访问，需要通过ECS反向代理。本文档将指导用户如何设置代理服务器。

1. 确定IP地址

用户应首先确认这几个IP地址： ECS外网IP（不便于透露，本文用XXX.XXX.XXX.XXX表示）和内网IP（实验用10.10.10.10）； GPU物理机内网IP（实验用10.239.23.4）；

2. 登录ECS跳板机

用户可以用PUTTY工具（Windows环境）或SSH命令（Linux环境）登录ECS，注意应使用ECS外网IP登入。

```
ssh -l login_name XXX.XXX.XXX.XXX（ECS外网IP）
```

登录成功后，可以在ECS跳板机上用SSH命令登录GPU物理机：

```
ssh -l root 10.239.23.4（GPU物理机内网IP）
```

3. ECS跳板机上部署代理服务器

这里选择Tengine，它是在NGINX的基础上由淘宝网发起的开源Web服务器项目。用户应注意，NGINX做前向代理服务器是不支持HTTPS连接的，所以客户端只能访问HTTP服务。

3.1 安装Tengine

重新开一个终端，登录到ECS跳板机。 获取Tengine源码：

```
wget http://tengine.taobao.org/download/tengine-2.1.1.tar.gz
```

解压：

```
tar zxvf tengine-2.1.1.tar.gz
cd tengine-2.1.1/
```

配置和编译：

```
./configure
make
sudo make install
```

默认情况下安装位置在 /usr/local/nginx/ 出于测试目的，我们需要在GPU物理机上也安装Tengine，步骤与ECS上安装基本一致，安装路径也在/usr/local/nginx/。

3.2 编辑ECS Tengine配置文件

登录ECS跳板机，用root权限打开 /usr/local/nginx/conf/nginx.conf 文件，增加一个server块，作用为监听本机的10000端口，将所有请求转发给GPU物理机（10.239.23.4:10000），配置内容如下：("//”后面为注释，真正的conf文件中应删除）

```
server {
        listen       10000;        // 监听本机的10000端口
        server_name  localhost;

        #charset koi8-r;

        #access_log  logs/host.access.log  main;

        location / {
            proxy_pass http://10.239.23.4:10000;
                // 10.239.23.4为本实验物理机内网IP，请根据需要修改
                // 这里实现了将来自外网的请求转发至物理机
        }

        #error_page  404              /404.html;

        # redirect server error pages to the static page /50x.html
        #
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }
    }
```

保存该文件。

3.3 编辑GPU物理机Tengine配置文件和主页

登录GPU物理机，用root权限编辑 /usr/local/nginx/conf/nginx.conf：

```
server {
        listen       10000;
                // 本文实验将默认的80改为10000，用户可根据需要修改
        server_name  localhost;

        #charset koi8-r;

        #access_log  logs/host.access.log  main;

        location / {
            root   html;
            index  index.html index.htm;
```

```
        }
        ..........略.............
    }
```

为了区分GPU物理机和ECS跳板机的主页内容，我们修改物理机的主页（/usr/local/nginx/html/index.html）代码如下：

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to GPU tengine!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to GPU tengine!</h1>
<p>If you see this page, the tengine web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://tengine.taobao.org/">tengine.taobao.org</a>.</p>

<p><em>Thank you for using tengine.</em></p>
</body>
</html>
```

ECS上主页保持不变即可。

3.4 启动GPU物理机上的Tengine

登录到GPU物理机，用root权限运行：sudo /usr/local/nginx/sbin/nginx 如果报错，请根据报错信息对3.2节中的nginx.conf配置文件做必要的修改。 查看NGINX进程：

```
# ps -ef | grep nginx
root     17205     1  0 10:58 ?        00:00:00 nginx: master process /usr/local/nginx/sbin/nginx
nobody   17206 17205  0 10:58 ?        00:00:00 nginx: worker process
root     20108 17042  0 11:54 pts/1    00:00:00 grep --color=auto nginx
```

可以看到NGINX主进程的pid为17205，worker进程的pid为17206，在需要关闭web server时可以直接以root权限运行：kill master_pid

3.5 设置GPU物理机防火墙

3.5.1 开启GPU物理机防火墙

```
CentOS6: service iptables start
CentOS7: systemctl start firewalld
```

3.5.2 添加防火墙规则

GPU物理机服务器需要使用10000端口，故添加防火墙例外：

```
iptables -I INPUT -p TCP --dport 10000 -j ACCEPT
```

4. 启动ECS反向代理服务

回到ECS跳板机终端，启动代理服务。 在启动ECS跳板机上的Tengine之前，我们需要确定ECS跳板机能访问物理机上的web server。在ECS跳板机上运行： curl 物理机内网IP:端口号 这里物理机内网IP为10.239.23.4，端口号在3.2节配置为10000，因此命令如下：

```
# curl 10.239.23.4:10000
<!DOCTYPE html>
<html>
<head>
<title>Welcome to GPU tengine!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to GPU tengine!</h1>
<p>If you see this page, the tengine web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://tengine.taobao.org/">tengine.taobao.org</a>.</p>

<p><em>Thank you for using tengine.</em></p>
</body>
</html>
```

根据返回内容，对照3.3节的主页修改情况可以确认返回内容来自GPU物理机。 确认ECS跳板机可以访问HPC物理机上的web server之后，开启ECS跳板机上的Tengine：（root权限执行）/usr/local/nginx/sbin/nginx 如果启动时报错，请根据报错信息修改。 查看ECS跳板机上的Tengine进程：

```
# ps -ef | grep nginx
root     16487     1  0 11:33 ?        00:00:00 nginx: master process /usr/local/nginx/sbin/nginx
nobody   16488 16487  0 11:33 ?        00:00:12 nginx: worker process
root     16565 16419  0 12:06 pts/8    00:00:00 grep nginx
```

可以看到相应主进程和worker进程的pid。注意不要与物理机上的pid混淆。

5. 测试

5.1 ECS上本地回环测试

可以在ECS跳板机上直接用curl进行本地回环测试：

```
# curl http://localhost:10000
<!DOCTYPE html>
<html>
<head>
<title>Welcome to GPU tengine!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to GPU tengine!</h1>
<p>If you see this page, the tengine web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://tengine.taobao.org/">tengine.taobao.org</a>.</p>

<p><em>Thank you for using tengine.</em></p>
</body>
</html>
```

对比3.3节的主页内容，可见返回的确实是GPU物理机上web server主页。

5.2 远程web页面测试

在浏览器的地址栏输入：http://XXX.XXX.XXX.XXX:10000/（这里XXX.XXX.XXX.XXX是ECS跳板机的公网IP），显示结果如下图所示：

可见，实现了远程访问HPC物理机上的web server，也验证了通过ECS进行反向代理的有效性。

使用流程

您在购买了高性能计算产品后，可以通过 **管理控制台** 找到 **高性能计算** 进行实例的管理。快速入口 https://hpc.console.aliyun.com/ 。

购买HPC实例后

用户购买高性能计算实例后，对于每个实例，用户将获得以下登录信息：

ECS公网IP。

ECS内网IP。

GPU物理机内网IP。

ECS登录账号（默认为root）和密码。

GPU物理机登录账号（默认为root）和密码。

首次登录之前

对于新购高性能计算实例，请修改初始化登录密码。

登录密码将作为您登录高性能计算实例的唯一凭证。

阿里云将不会以任何形式储存，因而无法提供密码找回功能。

跳转机密码需要进行重置才能登录。

初始物理机登录密码将通过站内消息发送给您，可在 **消息中心** 中进行查看。

重置跳板机密码

跳板机是您访问物理机前，需要首先登陆的虚拟机，随物理机所赠送。

在 **高性能计算** 控制台中，点击 **实例** 标签。

选择对应的高性能计算实例的 **操作** 选择 **跳板机**， 点击 **重置跳板机密码**。

在弹出的对话框中输入新密码，需要符合规定的密码强度。

需要重启跳板机后登录密码才能生效。

重置物理机密码

物理机仅能通过对应的跳板机进行访问，与其他用户之间网络完全隔离。

由于您拥有完全的访问权限，您可以在登录后输入 passwd 根据命令行提示进行修改。

物理机登录密码修改后将无法由控制台进行重置。

登录GPU物理机

登录方式如下：

在控制台中查看跳板机的公网 IP、内网 IP 以及物理机的内网 IP。

登录跳板机，打开终端输入ssh root@跳板机公网IP输入跳板机密码后登录到跳板机。

登录到跳板机后，在终端输入ssh root@物理机内网IP输入物理机密码后登录到物理机。

登录到物理机后，您将可以操作具有极致性能的 HPC 产品。

您在购买了高性能计算实例后，可以通过**管理控制台**找到**高性能计算**进行实例的管理。或者选择快速入口进入HPC控制台

购买HPC实例后

用户购买高性能计算实例后，对于每个实例，用户将获得以下登录信息

1. GPU物理机内网IP
2. GPU物理机登录账号（默认为root）和密码（通过控制台站内信告知）

华北2 区域HPC不再提供ECS跳板机，需要使用用户自己的任意一台华北2 区域经典网络ECS作为跳板机

首次登录之前

对于新购高性能计算实例，请修改HPC实例初始化登录密码。

1. 登录密码将作为您登录高性能计算实例的唯一凭证。
2. 初始登录密码将通过站内消息发送给您，可在 **消息中心** 中进行查看。
3. 若忘记密码，可以通过控制台重置HPC实例密码。

重置物理机密码

加入安全组的ECS实例以及同一账户下的HPC实例之间可以相互访问，但与其他用户之间网络完全隔离。

1. 由于您拥有完全的访问权限，您可以在登录后输入 passwd 根据命令行提示进行修改。
2. 可以通过控制台重置HPC实例密码。

登录GPU物理机

登录方式如下:

1. 在控制台中查看加入 HPC 安全组规则的 ECS 实例信息。公共云HPC如何使用安全组
2. 为保障 HPC 能够内网访问 ECS，请设置您接入 HPC 的 ECS 所在的安全组规则 内网入方向已经打开。ECS安全组默认规则
3. 登录跳板机，打开终端输入 ssh root@跳板机公网IP 输入跳板机密码后登录到跳板机
4. 登录到跳板机后，在终端输入ssh root@物理机内网IP输入物理机密码后登录到物理机
5. 登录到物理机后，您将可以操作具有极致性能的 HPC 产品

控制台使用

进入控制台后，单击左侧的**实例**。

查看实例

购买HPC后，您可以进入控制台查看购买的实例，包括物理机和ECS的状态、物理机IP、ECS的公网和内网IP以及实例生命周期等等。

重启物理机

您可以在控制台对物理机进行重启。首先找到想要重启的物理机，单击该实例右侧的**物理机**选项，然后在下拉菜单中选择**重启物理机**。

操作ECS跳板机

您可以在控制台对ECS跳板机进行启动、关闭、重启和重置密码等操作。首先找到想要重启ECS的实例，点击该实例右侧的**跳板机**选项，然后在下拉菜单中选择需要的操作。

默认安全组

1. 目前仅支持经典网络
2. 目前仅支持**华北2**

   目前暂不支持用户增加新安全组，当用户开通HPC服务后，系统会自动为客户创建一个名为default的默认安全组

目前最多允许用户增加**32条安全组规则**

默认安全组的默认网络访问控制规则为:

i. 内网入方向 denay all, 内网出方向 accept all;
ii. HPC暂不支持直接访问公网
iii. 同一账号的多个HPC实例可以相互访问

查询安全组内HPC实例

1. 登录HPC管理控制台
2. 单击左侧导航中的**安全组**
3. 选择地域为:**华北2**

选择default安全组，单击操作中的**管理实例**

在新窗口中可以看到您账号下所有的HPC实例，这些实例之间可以内网相互访问

查询安全组内安全组规则

1. 登录HPC管理控制台
2. 单击左侧导航中的**安全组**
3. 选择地域为:**华北2**

选择default安全组，单击操作中的**配置规则**

新页面中将显示您目前所有的安全组规则

授权安全组规则

授权安全组规则可以允许或者禁止与安全组相关的HPC实例的内网入出方向的访问，您可以随时授权和取消安全组规则。您的变更安全组规则会自动应用于与安全组相关联的HPC实例上。用户自己的ECS要访问HPC时，必须先给HPC增加一条安全组规则，允许该ECS访问HPC。**安全组规则的优先级不能重复**

操作步骤

1. 登录HPC管理控制台
2. 单击左侧导航中的**安全组**
3. 选择地域为:**华北2**

选择default安全组，单击操作中的**配置规则**

在新窗口中单击右上角的**添加规则**

在弹出的对话框中，设置下面参数：

i. 需要允许或拒绝访问的ECS IP地址

ii. 授权策略：目前仅支持允许操作，因此下拉菜单中选择 **允许**
iii. 网络类型：目前仅支持内网，因此下拉菜单中选择 **私有**

优先级：填写该规则的优先级，优先级为1-2999内整数，数字越大，优先级约低

1. 点击 **确定**，成功为该安全组授权一条安全组规则

删除安全组规则

操作步骤

1. 登录HPC管理控制台
2. 单击左侧导航中的**安全组**
3. 选择地域为:**华北2**

选择default安全组，单击操作中的**配置规则**

选择您要删除的规则，点击**删除规则**

或者您也可以批量删除安全组规则，选择需要删除的安全组规则前面选择框，单击**批量删除**

桌面环境设置方法

在HPC物理机上，如果需要使用桌面环境，可以参考如下步骤：

安装X Window运行 yum groups install "X Window System"

安装桌面这里以KDE桌面为例。运行 yum groups install "KDE Plasma Workspaces"

安装VNCServer运行 yum install tigervnc-server

启动VNCServer运行 vncserver –geometry 1920x1080后面窗口尺寸可以根据你的本地显示器分辨率进行修改。此时会提示需要创建VNC登录密码，根据需要设置。

VNC端口代理设置在ECS跳板机上，运行命令： ssh -4 -f -g -L 5901:GPU私网IP:5901 -N localhost 开启端口代理。这样可以通过ECS公网IP访问物理机上的VNCServer。

使用VNCClient登录Windows下可以使用Linux下可以使用Mac下可以使用Chicken of the VNC工

具

如果用户需要将本地文件传输到HPC上，除了使用OSS，还可以将HPC磁盘通过NFS方式挂载到ECS跳板机，利用ECS跳板机的公网IP可以直接从用户本地scp拷贝到挂载的物理机磁盘上。注意，这里ECS跳板机为NFS Client，而HPC物理机为NFS Server。假设物理机和ECS跳板机的操作系统均为Cent OS 7，其他系统可根据实际情况做调整。

1. ECS跳板机上安装nfs

```
yum upgrade lvm2
yum install nfs-utils
```

注意：如果没有执行yum upgrade lvm2，在安装nfs-utils会报错：

```
Transaction check error:
  file /usr/lib/systemd/system/blk-availability.service from install of device-mapper-7:1.02.107-5.el7_2.1.x86_64 conflicts with file from package lvm2-7:2.02.105-14.el7.x86_64
  file /usr/sbin/blkdeactivate from install of device-mapper-7:1.02.107-5.el7_2.1.x86_64 conflicts with file from package lvm2-7:2.02.105-14.el7.x86_64
  file /usr/share/man/man8/blkdeactivate.8.gz from install of device-mapper-7:1.02.107-5.el7_2.1.x86_64 conflicts with file from package lvm2-7:2.02.105-14.el7.x86_64
```

HPC物理机上安装nfs
yum install nfs-utils
vi /etc/exports

/disk1 ECS跳板机内网ip(rw,secure,no_root_squash,sync)
例如：
/disk1 10.117.12.27 (rw,secure,no_root_squash,sync)

vi /etc/hosts
GPU内网ip GPU的hostname
例如：
10.172.66.99 AliHPC-GPU017

HPC物理机上启动NFS服务

```
service rpcbind start
service nfs start
```

查看是否生效：
showmount –e

显示：

```
Export list for AliHPC-GPU017:
/disk1 10.117.12.27
```

1. 在ECS跳板机上挂载物理机磁盘

查看GPU上export的地址：
showmount -e GPU内网ip
例如：
showmount -e 10.172.66.99
显示：

```
Export list for 10.172.66.99:
/disk1 10.117.12.27
```

mount GPU上的nfs目录：
mount -t nfs GPU内网ip:/disk1 /disk1
例如：

```
mkdir /disk1
mount -t nfs 10.172.66.99:/disk1 /disk1
```

运行后，物理机的/disk1将被挂载至ECS跳板机的/disk1下。此时用户可以在本地执行scp命令，将文件拷贝至ECS跳板机的/disk1/下，即可自动同步到物理机。
scp my_local_file username@ECS公网IP:/disk1/
输入登录信息即可启动文件传输。

访问其他云产品

应用场景

当用户需要将大量数据（几十GB到上TB）传输到HPC机器上时，使用scp拷贝有两个缺陷：（1）需要通过ECS中转数据，scp须执行两次；（2）数据太大时ECS自带磁盘容量很可能不够用；此时，使用OSS服务是比较理想的方式，无需ECS中转，直接实现端对端的传输，而且在HPC上通过内网访问OSS，速度更快、更稳定。本节介绍如何在HPC机器上使用OSS服务。

物理机访问OSS方法

首先登录阿里云官网，进入OSS控制台https://oss.console.aliyun.com/index#/

点击Access Keys，“显示”Access Key，用手机短信验证。

拿到Access Key ID和Access Key Secret，用于鉴别身份。

创建一个Bucket，位于华东1区。其host为oss-cn-**-internal.aliyuncs.com，具体可以在控制台查到。

注意：目前GPU物理机通过内网只能访问华东1区的OSS服务

在HPC机器上运行：

```
osscmd config --host=oss-cn-********-internal.aliyuncs.com \
--id=用户的OSS Access Key ID  \
--key=用户的OSS Access Key Secret
```

之后运行

osscmd ls

可以看到你创建的Bucket，说明配置成功，可以通过osscmd进行文件上传/下载操作。

先浏览有哪些bucket：

```
# osscmd ls
CreateTime          BucketLocation       BucketName
2015-09-17 17:17:08  oss-cn-********      hpc-data-release
2015-09-15 11:29:58  oss-cn-********      hpc100users

Bucket Number is: 2
0.047(s) elapsed
```

可见有两个bucket，都位于华东1区。

进一步浏览其中一个bucket详细内容：

```
# osscmd ls oss://hpc100users
prefix list is:
object list is:
2015-09-18 10:38:49 1066.97MB Standard oss://hpc100users/cuda_7.0.28_linux.run
(oss://hpc100users/cuda_7.0.28_linux.run)

prefix list number is: 0
object list number is: 1
0.022(s) elapsed
```

注意OSS路径都以"oss://"开头，后面紧跟bucket名称，再向后为文件名或目录名。

下载命令：

```
# osscmd get oss://Bucket名称/文件名称 本地文件名称
```

测试下载：

```
# osscmd get  oss://hpc-data-release/cuda_7.0.28_linux.run cuda_7.0.28_linux.run
100%  The object cuda_7.0.28_linux.run is downloaded to cuda_7.0.28_linux.run, please check.
61.593(s) elapsed
```

下载cuda_7.0.28_linux.run到本地用时61.593s，该文件大小为1066.97MB，下载速度约为17.3 MB/s
上传命令：

```
# osscmd put  本地文件名称 oss://Bucket名称/文件名称
```

测试上传：

```
# osscmd put cuda_7.0.28_linux.run oss://hpc-data-release/cuda_7.0.28_linux.run
100%
Object URL is: http://hpc-data-release.oss-cn-********-internal.aliyuncs.com/cuda_7.0.28_linux.run
Object abstract path is: oss://hpc-data-release/cuda_7.0.28_linux.run
ETag is "312AEDE1C3D1D3425C8CAA67BBB7A55E"
61.287(s) elapsed
```

常见问题

运行osscmd ls后报错如下：

```
Error Headers:

[('content-length', '467'), ('server', 'AliyunOSS'), ('connection', 'keep-alive'), ('x-oss-request-id',
'56921E152530B7D05CA805DD'), ('date', 'Sun, 10 Jan 2016 09:02:13 GMT'), ('content-type', 'application/xml')]
Error Body:

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>RequestTimeTooSkewed</Code>
  <Message>The difference between the request time and the current time is too large.</Message>
  <RequestId>56921E152530B7D05CA805DD</RequestId>
  <HostId>oss-cn-********-internal.aliyuncs.com</HostId>
  <MaxAllowedSkewMilliseconds>900000</MaxAllowedSkewMilliseconds>
  <RequestTime>2015-10-09T16:00:04.000Z</RequestTime>
  <ServerTime>2016-01-10T09:02:13.000Z</ServerTime>
</Error>

Error Status:

403
ls Failed!
```

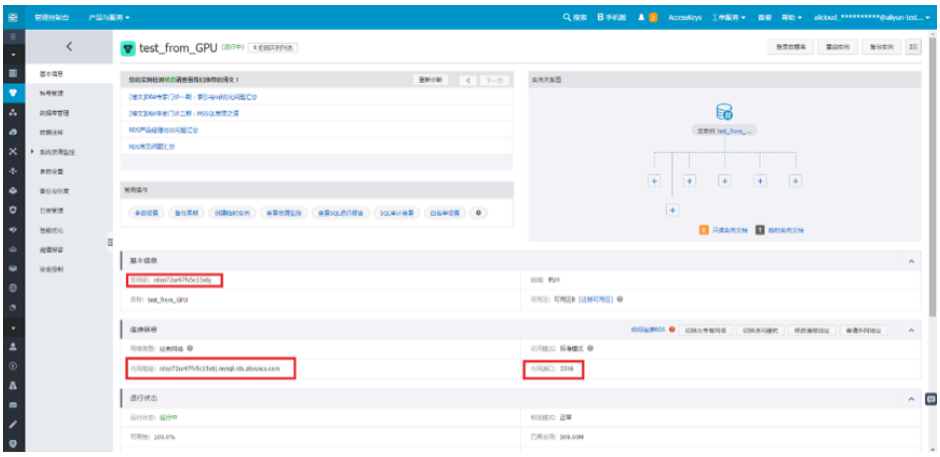
这是由于HPC机器日期与OSS服务器日期差别太大造成的。通过date命令修改日期时间为正确的时间即可。

HPC访问云数据库RDS

购买和配置RDS

用户可以登陆阿里云官网购买RDS实例。

购买成功后，在RDS管理控制台可以看到当前实例：



注意将内网地址、端口号和访问账号记录下来，访问密码在创建实例时已经设置，如果忘记，则需要通过“账号管理”——“修改密码”进行修改。

重要：在“常用操作”中点击“白名单设置”，将GPU物理机的内网IP加入白名单，这里实验用10.239.23.4。设置完成后如下图所示：

GPU物理机访问RDS

目前GPU物理无法直接访问RDS，需要通过ECS中转，方法如下。

如果物理机上没有安装mysql，则运行以下命令安装：

```
sudo yum install mysql
```

建立ssh tunnel，在GPU物理上运行命令

```
ssh -fN -v -N root@ECS内网IP -L 中转端口号/RDS内网地址/RDS内网端口
```

其中中转端口号由用户指定，RDS内网端口默认为3306，可以在控制台查到，RDS内网地址用户可以在控制台查到。

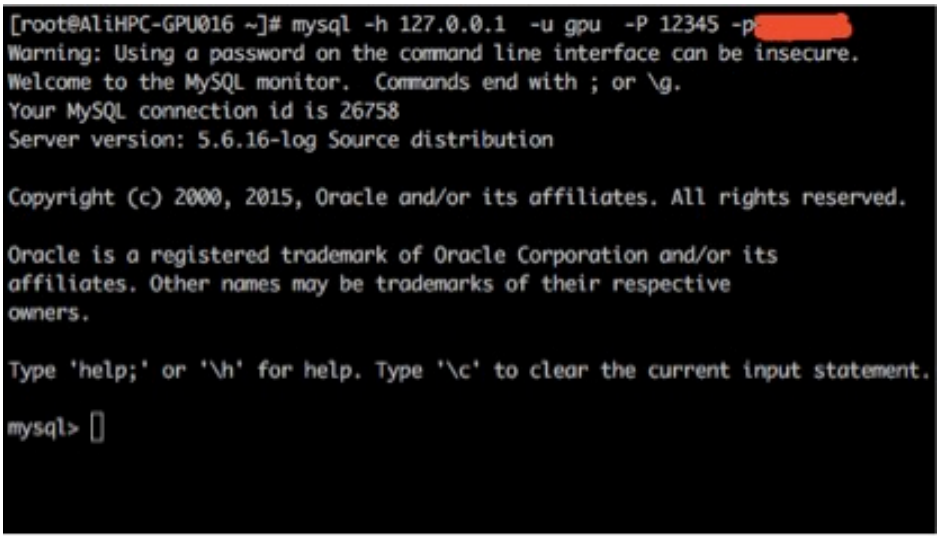
运行上述命令后，输入ECS跳板机的密码。

之后，执行下面命令来登录RDS：

```
mysql -h 127.0.0.1  -u dbuser -P 12345 -ppassword
```

其中–h 指定本地IP-u 是RDS的访问账号，在控制台创建，本例中用户名为dbuser-P 是上一步中指定的中转端口号-p 后面是dbuser对应的密码，-p与密码之间无空格

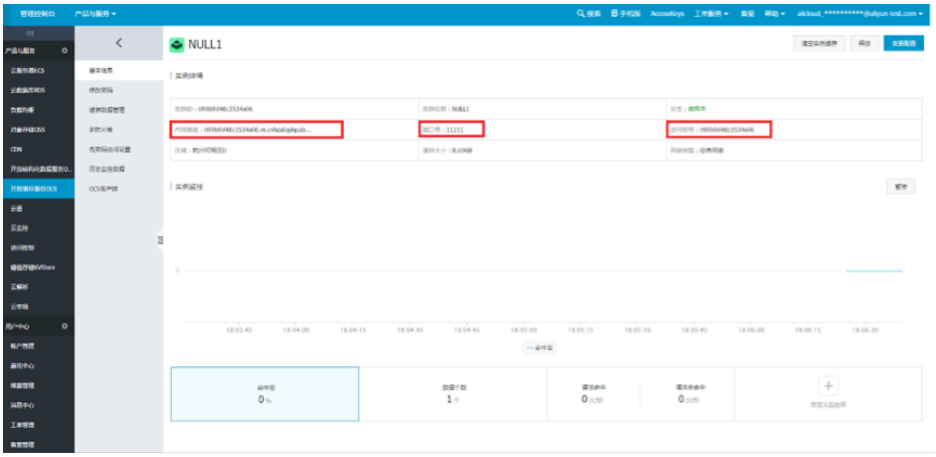
连接成功后出现如下信息



上图中用户名为gpu，使用的中转端口为12345

用户可以在[在阿里云官网购买]云数据库memcache版实例(http://www.aliyun.com/product/ocs)

购买成功后，在云数据库memcache版管理控制台可以看到当前实例。



注意将内网地址、端口号和访问账号记录下来，访问密码在创建实例时已经设置，如果忘记，则需要点击左侧面板“修改密码”进行修改。

通过ECS跳板机登录GPU物理机，从github上下载文件python-binary-memcached-0.24.2.tar.gz并拷贝到物理机上。

解压：

tar zxvf python-binary-memcached-0.24.2.tar.gz

安装：

```
cd python-binary-memcached-0.24.2/
sudo python setup.py install
```

目前物理机无法直接访问云数据库memcache版，需要通过ECS中转，方法如下。

建立ssh tunnel，在GPU物理上运行命令

```
ssh -fN -v -N root@ECS内网IP -L 中转端口号/云数据库memcache版内网地址/云数据库memcache版内网端口
```

其中中转端口号由用户指定，云数据库memcache版端口号、云数据库memcache版内网地址用户可以在控制台查到。

运行上述命令后，输入ECS跳板机的密码。

之后，物理机可以访问云数据库memcache版，但是需要用localhost来替代云数据库memcache版内网地址，用中转端口号来替换云数据库memcache版端口号。

测试： 在物理机上编写test_ocs.py代码如下：

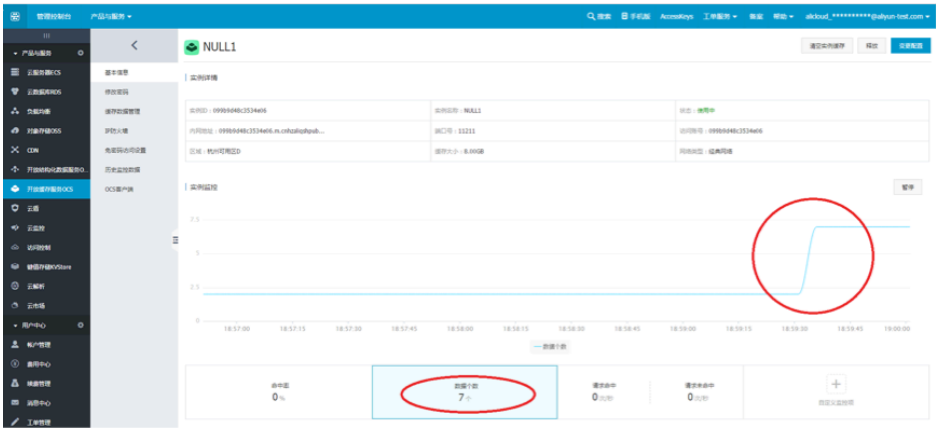
```
#!/usr/bin/env python
import bmemcached
client = bmemcached.Client(('localhost:中转端口号'), 'OCS访问账号', 'OCS密码')
print client.set('city1', 'Beijing')
print client.set('city2', 'Shanghai')
print client.set('city3', 'Guangzhou')
```

```
print client.set('city4', 'Shenzhen')
print client.set('city5', 'Hangzhou')
print client.get('city5')
print client.get('city4')
print client.get('city3')
print client.get('city2')
print client.get('city1')
```

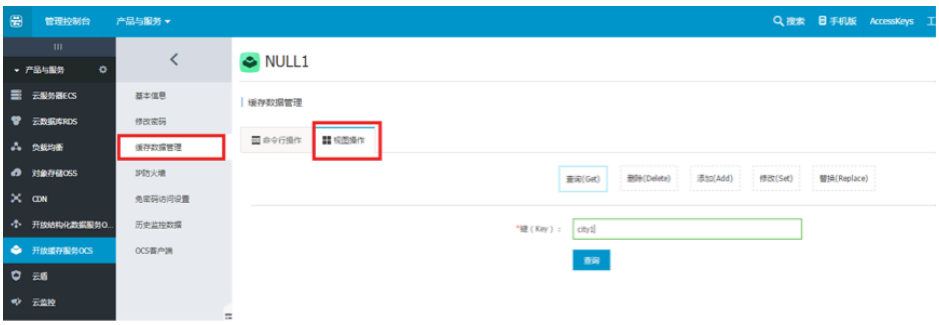
运行：

```
python test_ocs.py
True
True
True
True
True
Hangzhou
Shenzhen
Guangzhou
Shanghai
Beijing
```

回到OCS管理控制台查看，如下图所示



可以看到OCS的数据个数增加了5个，这是由GPU物理机写入的。查看数据可以通过控制台左侧面板中的“缓存数据管理”实现。

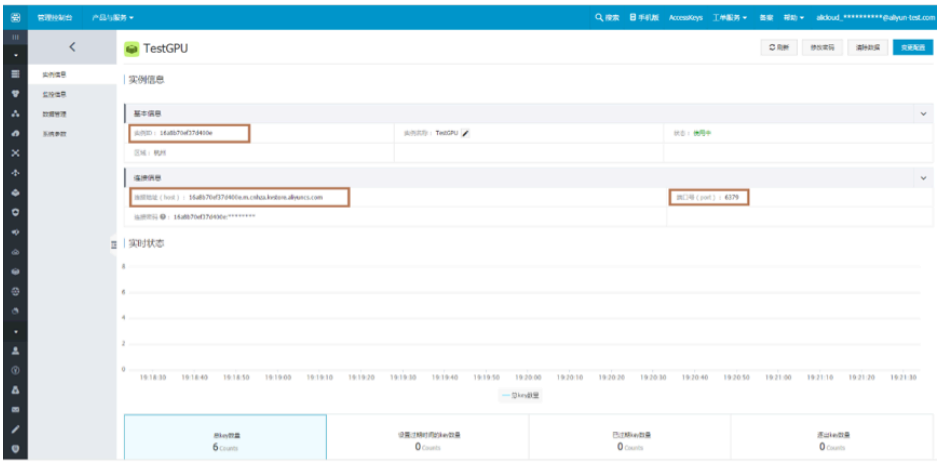


依次输入“city1” ~ “city5” ，结果应该与test_ocs.py中预设值一致。

关于OCS更多文档参考OCS文档中心。

用户可以[在阿里云官网购买]云数据库Redis版(http://www.aliyun.com/product/kvstore)

购买成功后，在云数据库Redis版可以看到当前实例：



注意将内网地址、端口号和访问账号记录下来，访问密码在创建实例时已经设置，如果忘记，则需要点击右上方“修改密码”进行修改。

通过ECS跳板机登录GPU物理机。 获取redis-py：
git clone https://github.com/andymccurdy/redis-py.git
（如果不能访问外网，需要参考如何通过ECS设置代理）
安装：

```
cd redis-py/
sudo python setup.py install
```

物理机目前无法直接访问云数据库Redis版，需要通过ECS中转，方法如下。

建立ssh tunnel，在GPU物理上运行命令

```
ssh -fN -v -N root@ECS内网IP -L 中转端口号/云数据库Redis版内网地址/云数据库Redis版内网端口
```

其中中转端口号由用户指定，云数据库Redis版端口号、云数据库Redis版内网地址用户可以在控制台查到。

运行上述命令后，输入ECS跳板机的密码。

测试： 在物理机上编写test_kvstore.py代码如下：

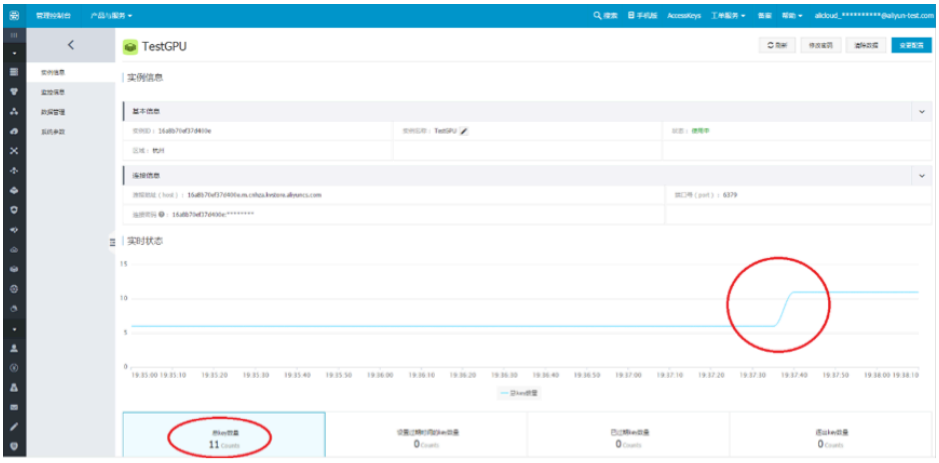
```
#!/usr/bin/env python
#-*- coding: utf-8 -*-
import redis

#这里替换为localhost的内网IP和中转端口号
host = 'localhost'
port = 中转端口号

#这里替换为实例id和实例password
```

```
user = '16a8b70ef37d400e'
pwd = '************'

#连接时通过password参数指定AUTH信息，由user,pwd通过":"拼接而成
r = redis.StrictRedis(host=host, port=port, password=user+':'+pwd)

#连接建立后就可以进行数据库操作，详情文档参考https://github.com/andymccurdy/redis-py
r.set('book1', 'Algorithms');
r.set('book2', 'Assembly');
r.set('book3', 'Mathmatics');
r.set('book4', 'Artists');
r.set('book5', 'Calculors');
print r.get('book1')
print r.get('book2')
print r.get('book3')
print r.get('book4')
print r.get('book5')
```

运行：

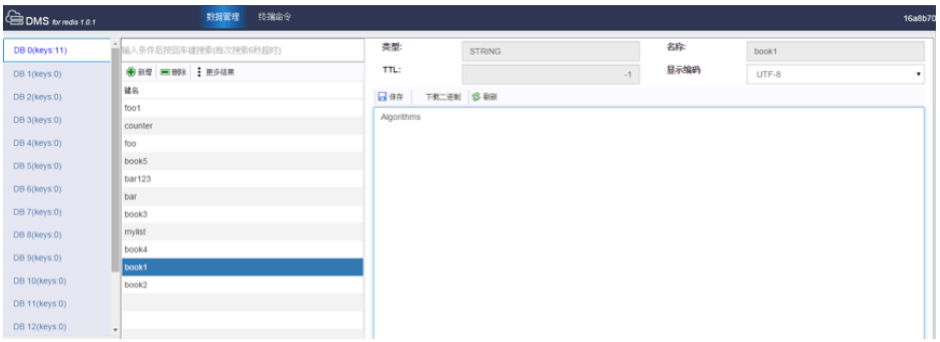
```
python test_kvstore.py
Algorithms
Assembly
Mathmatics
Artists
Calculors
```

回到KVStore管理控制台查看，如下图所示



可以看到Key个数增加了5个，这是由GPU物理机写入的。查看数据可以通过控制台左侧面板中的“数据管理”实现

点击“立即体验”进入阿里云数据管理系统（Aliyun DMS），界面如下：

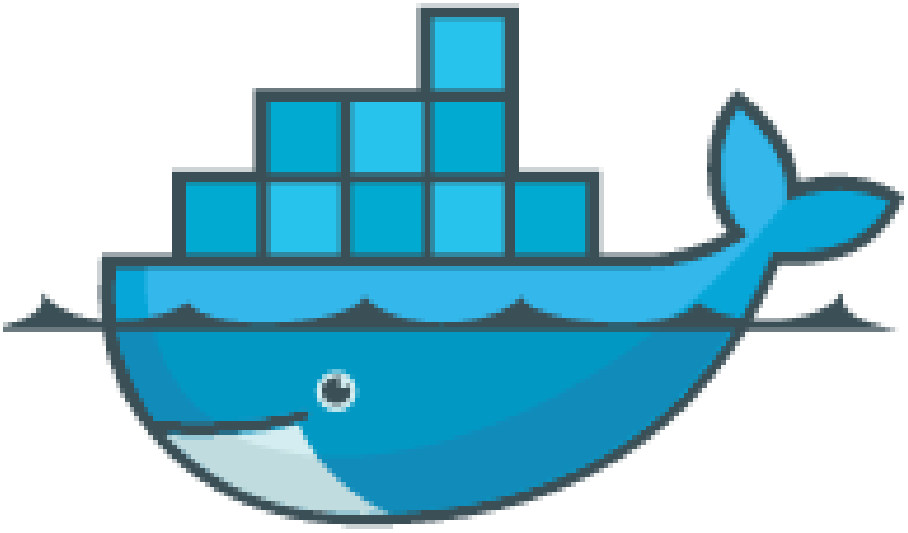


依次点击“book1”～“book5”，结果应该与test_kvstore.py中预设值一致。

关于KVStore更多文档参考KVStore文档中心。

软件镜像

什么是docker



- docker 是一个开源的**应用容器引擎**，基于 Go 语言并遵从 Apache2.0 协议开源。Docker 可以让开发者打包他们的应用以及依赖包到一个轻量级、可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。容器完全使用沙箱机制，相互之间不会有任何接口（类似 iPhone 的 app）,更重要的是容器性能开销极低。

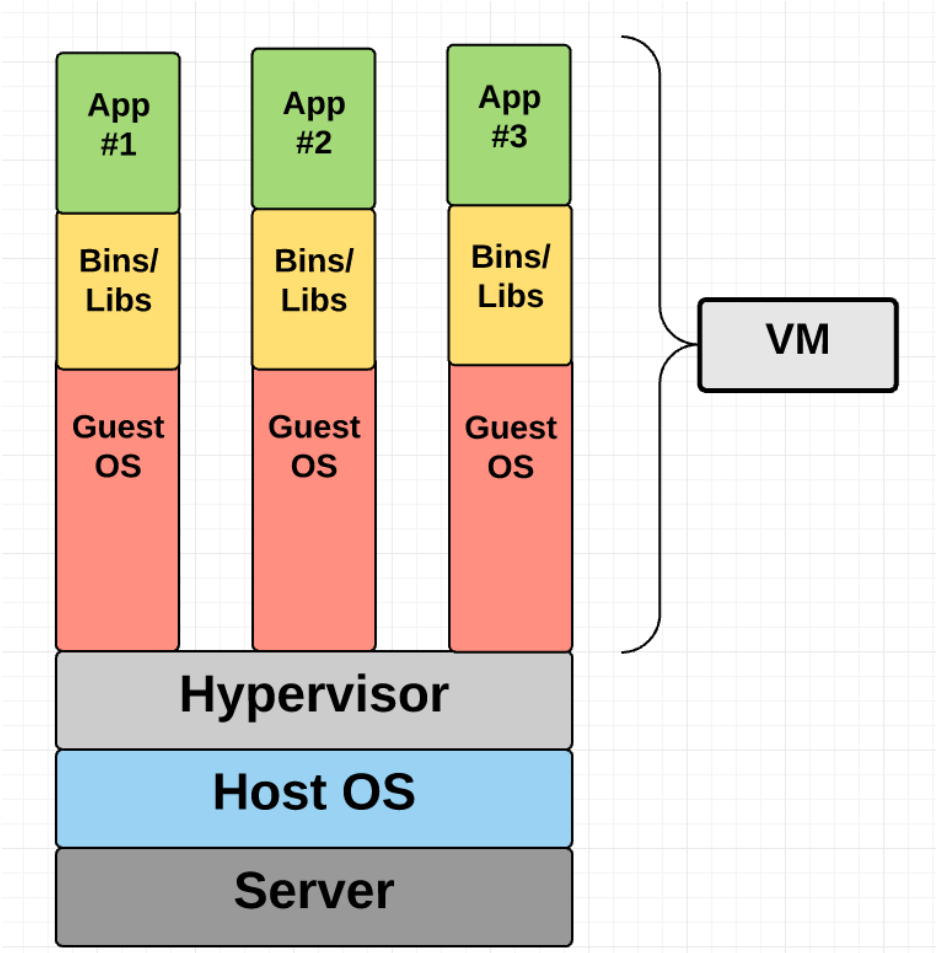
为什么用 docker

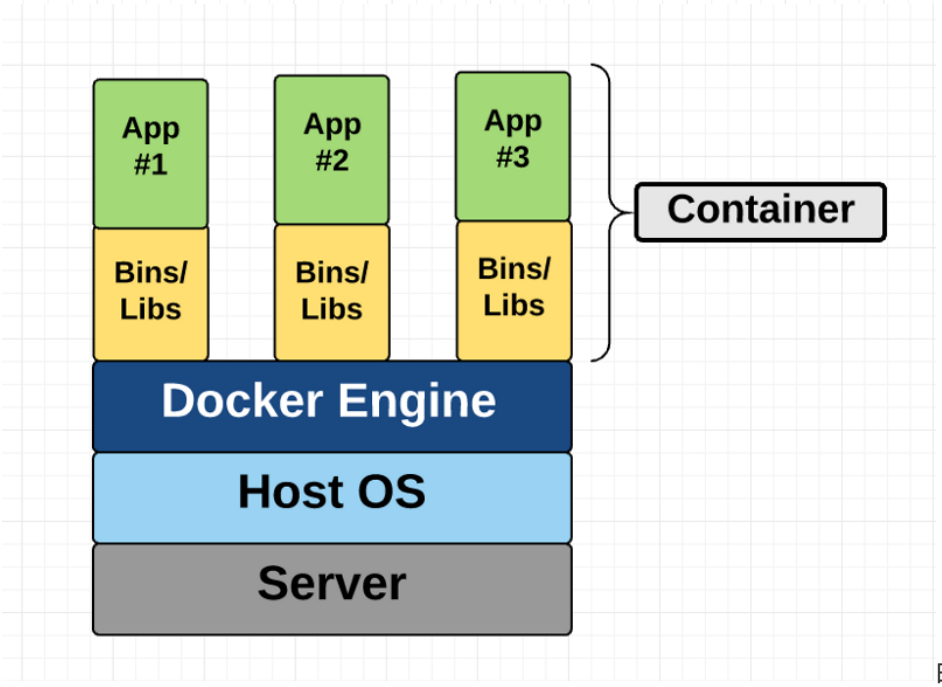
- 当前最火热的技术之一，为您的部署运维带来颠覆性变化。
- 简洁优雅，既为每个应用提供干净的环境，又免于虚拟机带来的性能损失。
- 通过阿里云开发者平台，立即获取您喜爱的 CUDA / Caffe / tensorflow 等框架。

如何使用Docker

基本概念

初次接触 docker 的用户可能觉得 docker 与虚拟机有许多功能上的相似之处，但两者其实完全不同。我们知道**操作系统 = 内核 + 服务组件**，centos、redhat、ubuntu 等各个发行版差异正是主要在服务组件上。虚拟机提供了完整的硬件和操作系统虚拟化，而 docker 仅仅对服务组件（用户空间）的部分进行抽象化，并不会虚拟宿主机的内核与硬件层，两者架构如图所示：





由于docker容器共用宿主机内核，因此其在满足用户对各个应用“环境”隔离需求下，能获得几乎等于裸机的性能。

nvidia - docker

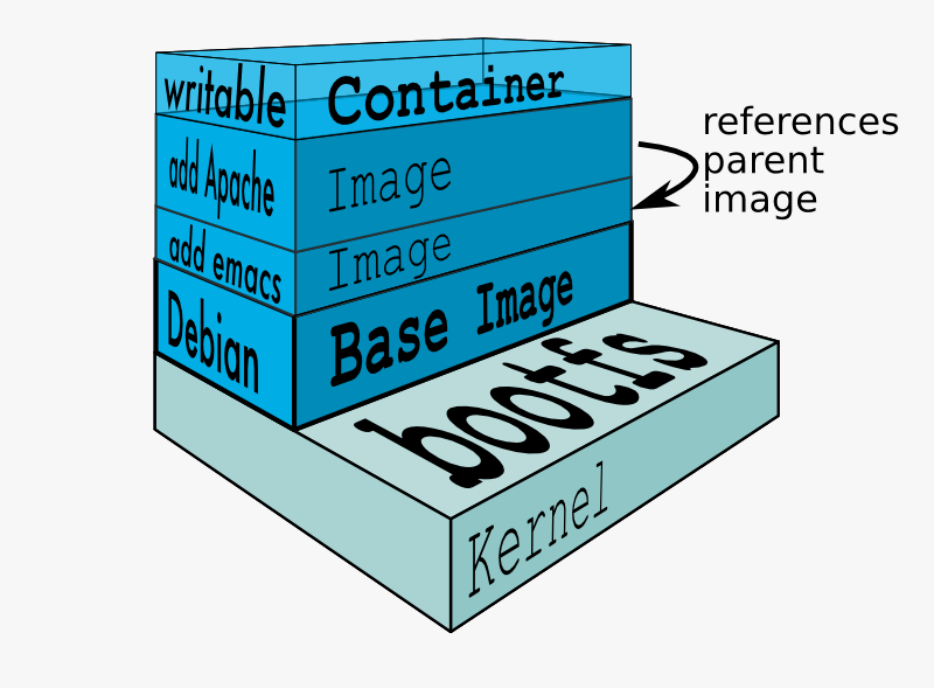
nvidia - docker 是 Nvidia 公司为 docker 所做的封装，开源代码可以在 Github 找到。nvidia - docker 可以对GPU做抽象，只要容器中的GPU驱动版本不高于宿主机的GPU驱动版本，即可在容器中使用GPU资源。

镜像与容器

- image：镜像。镜像可以理解为是对系统某一时刻的打包或快照。
- container：容器。容器是指从镜像生成的一个系统**实例**，类似于虚拟机实例，是实际运行应用的地方。

对镜像与容器的关系做一个类比的话，就类似于面向对象语言中“类”与“实例”的关系。镜像是只读的“蓝图”，容器则是由镜像生成的实体。同一个镜像可以在一台机器上生成多个容器，每个容器都运行着自己的环境和应用。不过容器也可以通过 commit 操作，自己当前的状态打包保存成一个新的镜像。

类似于 git，镜像也采用“分层”的机制，每层仅保存在上一层镜像基础上的增量改动：



基本组件与术语解释

- docker client：用户运行 docker 的客户端。
- docker daemon：docker 服务。在使用 docker 的机器上必须有 docker 服务进程运行。
- registry：托管镜像的服务，完全类似于托管代码的 GitHub，实际上，最著名的镜像托管服务就叫做 DockerHub。其他可用的 registry 服务还包括**阿里云开发者平台**，在阿里云上使用可以享受阿里云加速。
- repostory：镜像仓库。与 Github 完全类似，用户在托管服务上可以创建多个镜像仓库，每个仓库用来保存某一应用镜像的历次 push 版本。
- dockerfile：docker 镜像描述文件。docker 除了使用 registry 进行镜像分发外，还可以使用一个文本文件来描述镜像的内容，便于用户在自己的机器上构建镜像。

安装docker

> 目前HPC北京区域接入了阿里云容器服务，请您直接按照这里进行安装。请注意，安装容器服务后将会自动在您的机器上运行两个容器，这两个容器运行着容器服务的管理程序，请勿停止它们或关闭 docker 后台进程。
>
> HPC杭州区域没有接入阿里云容器服务，请您先按照文档设置跳板机公网访问，再参考这里进行安装。

基本用法

docker的工作流程非常类似git版本控制系统：**从托管仓库（如DockerHub）拉取（pull）镜像到本地 -> 从镜像生成容器 -> 在容器里运行应用 -> 将容器提交（commit）为新的镜像 -> 将新镜像推送（push）到托管仓库。**

接下来我们以运行带 cuda 的容器为例来展示 docker 的用法。由于我们需要在镜像中使用GPU，所以需要使用 Nvidia 包装的 nvidia-docker。如果不需使用GPU，则下文中的 nvidia-docker 命令均可直接替换为 docker。

启动 docker daemon

首先您必须确保 docker 服务已经在后台运行。启动 docker daemon 请使用命令：

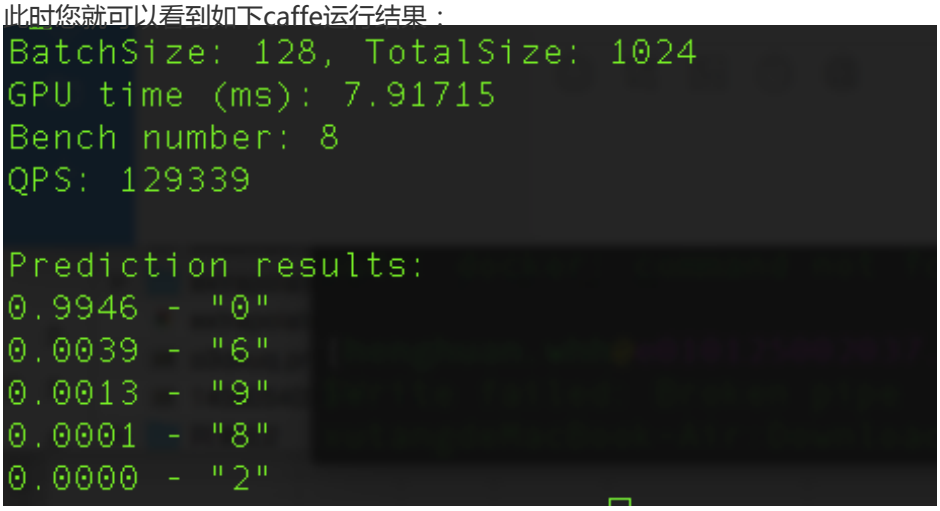
```
sudo systemctl start docker
sudo systemctl start nvidia-docker
```

一条命令获取并运行caffe

让我们先来感受一下docker一键部署的魔法。执行以下命令即可获取并运行一套包含 caffe 和其运行环境的容器并输出mnist预测结果：

```
nvidia-docker run registry-internal.cn-beijing.aliyuncs.com/hpc_beijing/caffe:batch-googlenet8 ./run_mnist.sh
```

此时您就可以看到如下caffe运行结果：



是不是很方便呢？现在让我们仔细看一下run命令用法。首先需要明白，docker的工作流程是”拉取镜像->从镜像创建容器->启动容器”，这些操作可以分步进行，而run命令则直接一步到位全部执行完毕。

命令中的registry-internal.cn-beijing.aliyuncs.com/hpc_beijing/caffe:batch-googlenet8指定了需要使用的镜像，一般格式为registry地址/namespace/repo:tag，其中registry地址是您使用的托管服务器地址，比如dockerhub，或者使用阿里云开发者平台。run 命令会优先使用本地已有的镜像，如果本地找不到指定镜像，则会从托管服务器拉取镜像。您也可以使用镜像的 image ID 来指定镜像，但是这样只能使用本地镜像。namespace是您在托管服务上注册的身份ID。repo是项目仓库，类似于 Github 中的代码 repo。Docker 建议一个镜像仅用来运行一个应用，这个应用的所有镜像版本都保存在同一仓库中。tag是用于标记同一仓库中的不同镜像版本，如果省略，将默认使用 latest 标签。

请注意，如果您使用的托管服务器是Dockerhub，则需要保证您的hpc机器可以访问公网；如果您使用阿里云开发者平台，网站提供的地址一般是公网地址 registry.cn-hangzhou.aliyuncs.com，请您自行修改为内网地址 registry-internal.cn-hangzhou.aliyuncs.com，这样可以享受更快的内网速度并且不占用您的公网带宽和流量。

命令中末尾部分的./run_mnist.sh是指定容器运行后将要运行的命令，本例中即是运行“run_mnist.sh”脚本。请注意，容器内的**前台命令**执行完毕后容器就会自动退出（stop），不再占用系统运行的资源。如果您希望保持容器长期运行，您需要让容器执行一个长期的前台命令，比如 top 或 tail 。以运行 nginx 为例，您应该指定这样的命令：

```
nvidia-docker run -d your/nginx/image nginx -g "daemon off;"
```

将nginx挂到前台；或者借助tail命令:

```
nvidia-docker run -d your/nginx/image service nginx start && tail -f /var/log/nginx/error.log
```

这里的-d是将容器本身挂到宿主机的后台运行，避免把容器的输出直接打印到您的屏幕上。

“进入”容器中（交互式运行容器）

很多时候您更希望直接在容器中进行开发，而不是让容器执行一条命令就退出。还是以 caffe 为例，您可以一条命令轻松进入一个干净的 caffe 环境：

```
nvidia-docker run -ti registry-internal.cn-beijing.aliyuncs.com/hpc_beijing/caffe:batch-googlenet8 /bin/bash
```

命令中-t参数表示需要开启tty，-i参数表示交互式运行，/bin/bash表示进入 bash shell 。运行后您会看到这样的提示符：

这表示您已经在容器中，以root身份位于/workspace目录下。您可以通过ls /caffe看到caffe目录。当您完成开发后，可以ctrl + D退出容器，此时容器没有前台进程，会自动停止运行。

通常您可能需要从本地挂载文件到容器中以便使用。您可以使用-v命令挂载本地目录到镜像中。一下为将本地的 /root/share 目录挂载到容器中 /share 目录的命令，请注意目录必须使用绝对路径：

```
nvidia-docker run -ti -v /root/share:/share registry-internal.cn-beijing.aliyuncs.com/hpc_beijing/caffe:batch-googlenet8 /bin/bash
```

版本控制

docker可以如同git管理代码版本一样管理您的镜像版本。首先再强调一下概念，**镜像**是保存在仓库中的模板，**容器**是由这个模板生成并可实际运行的实例。您可以使用nvidia-docker images查看您本地的所有镜像：

使用nvidia-docker ps可以查看所有运行中的容器：

注意直接使用ps命令只会显示出正在运行的容器，加上-a选项后能看到所有的容器（包括已经停止的）。

下面我们来对容器进行一些开发修改，然后将修改提交为新的镜像以保存修改。首先，从一个基本镜像开始，创建、运行并进入容器中：

```
nvidia-docker run -ti registry-internal.cn-beijing.aliyuncs.com/hpc_beijing/caffe:batch-googlenet8 /bin/bash
```

然后，我们在容器中创建一个”hello”文件：

```
touch hello
```

然后，ctrl+D退出容器。请注意，此时您的容器已经停止。如果您此时再重复使用上面的run命令，将会从基础镜像创建一个**新的**容器，并没有包括您的修改。此时，请使用以下命令查看所有容器：

```
nvidia-docker ps -a
```

从列表中找到刚才退出的容器，记下其 container ID，本例中是5080ed3c40a6。然后提交这个容器成为新的镜像：

```
nvidia-docker commit 5080ed3c40a6 caffe:v0.1
```

这个命令将会使用ID为5080ed3c40a6的容器创建一个新的镜像，其仓库名为caffe，标签为v0.1。此时您使用nvidia-docker images可以看到您刚提交的新镜像。现在您可以从新镜像创建容器并运行：

```
nvdia-docker run -ti caffe:v0.1 /bin/bash
```

此时您可以看到，新容器里已经包括了您刚刚添加的 hello 文件了。如果您在任一托管服务上注册了仓库，那么您可以按照托管服务提供的方式注册后，将您的新镜像 push 到远端仓库中。假定您在阿里云开发者平台上注册了一个名为 tester 的 namespace，然后在此空间下创建了一个名为 caffe 的仓库。您首先需要将您要 push 的镜像打标签指明目的地：

```
nvidia-docker tag caffe:v0.1 registry-internal.cn-beijing.aliyuncs.com/tester/caffe:v0.1
```

然后可以推送镜像到远端保存：

```
nvidia-docker push registry-internal.cn-beijing.aliyuncs.com/tester/caffe:v0.1
```

分解 run 命令

上文中我们使用run命令一步完成了“拉取”、“创建”、“运行”的动作，但是有时您可能想要自行管理镜像和容器。您可以使用pull命令从远端仓库拉取镜像：

```
nvidia-docker pull registry-internal.cn-beijing.aliyuncs.com/hpc_beijing/cuda:latest
```

然后，您可以使用create命令从某一镜像中创建容器：

```
nvidia-docker create -ti -v /root/share:/share registry-internal.cn-beijing.aliyuncs.com/hpc_beijing/cuda:latest
/bin/bash
```

注意create命令的格式与run命令基本一致，如果您想要交互式运行、挂载文件夹、指定容器运行的命令等，请在此设置。create命令仅会创建一个容器实例但是不会运行它，您可以在任意时间手动运行：

```
nvidia-docker start CONTAINERID
```

这里的CONTAINERID可以通过docker ps -a命令查到。容器运行后，您并不会自动进入容器中，您可以使用以下命令进入容器：

```
nvidia-docker attach CONTAINERID
```

镜像与容器管理

如果您想停止某个运行中的容器，可以使用命令：

```
nvidia-docker stop CONTAINERID
```

无论是您手动停止还是因为没有前台程序运行而自动停止的容器，都不会自动从您的机器中删除。您随时可以使用docker start命令重新启动它们，之前在容器中做的修改也会保留。当您不需要某些容器时，可以用以下命令移除他们，以节省资源：

```
nvidia-docker rm CONTAINERID
```

同样，对于不需要的镜像，您也可以用以下命令移除：

```
nvidia-docker rmi IMAGEID
```

更多资源

深度学习相关镜像

我们制作了一些与深度学习相关的镜像，帮助您快速开始开发：

软件	Registry
CUDA	开发者平台
Caffe	开发者平台
tensorflow	开发者平台
theano	开发者平台
torch	开发者平台

快速部署HPC监控

出于用户隐私的考虑，我们HPC机器并没有部署监控。如果您购买了北京区域HPC并且有监控相关需求，可以参考文档使用容器服务快速部署监控应用。

更多阿里云容器服务

您可以从这里获取更多阿里云容器服务资源，包括使用 docker 快速部署集群等。

操作流程

1. 登录HPC容器管理界面，如果页面提示您还未开通容器服务，请单击开通。

   单击左侧导航的**集群**页，选中右上角的**创建集群**旁边的小三角，单击**创建本地集群**，跳转到下面的页面：

设置集群的名称和网络类型,单击**创建集群**

集群名称可以设置一个名称用于标示集群。名称要求在同一个用户和同一个region下唯一。目前使用的region是**cn-beijing-hpc**。

地域选项请一定选择HPC地域。如果您的界面上没有HPC地域，请提工单将您的阿里云账号告诉我们，为您开通相关权限。

网络类型请一定选择Overlay网络。**并且请一定注意不要使用192.168.0.0/16，建议使用172.80.0.0/16。**

是否新增节点默认不创建节点。

添加节点

4.1 回到**集群**页面，在已经创建的集群右侧单击**更多**下拉框，选择**添加已有实例**

4.2 查看需要在HPC机器上执行的脚本,特别注意其中的长串

4.3 登录到HPC机器

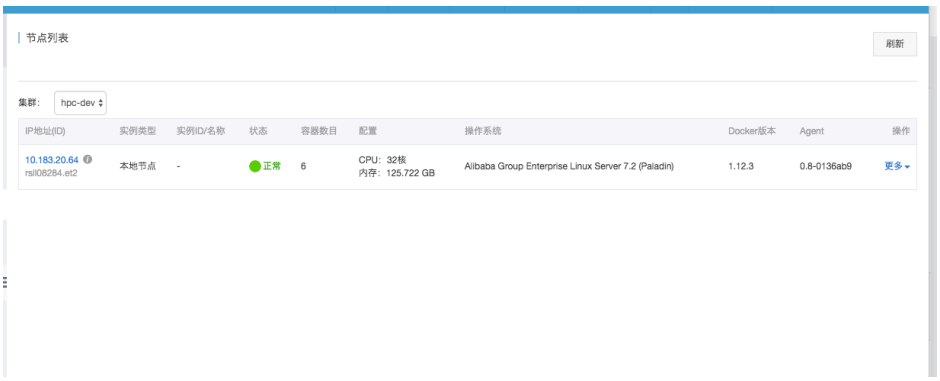
(1) 清理已有的nvidia docker和docker。**如果这是您新购买的北京HPC，请略过此步。**如果这不是您第一次注册HPC容器集群，未避免兼容问题请在机器上运行以下脚本：

```
docker volume rm $(docker volume ls -q)
docker rm -f $(sudo docker ps -aq)
service nvidia-docker stop
service docker stop
rpm -qa|grep nvidia-docker|xargs yum remove -y
rpm -qa|grep docker|xargs yum remove -y
rm -rf /etc/docker
rm -rf /disk2/docker
ifconfig docker0 down
brctl delbr docker0
ifconfig docker_gwbridge down
brctl delbr docker_gwbridge
```

(2) 在HPC机器中下载并运行注册脚本。注意c46e89653c69d0fcedc156d19b6e2156f5668001 是步骤**3.2**中的长串，请根据您在**3.2**步中的实际情况设置。 --advertise-interface 是与外界通信的网络接口，北京HPC一律是bond0.700

```
curl -Ls http://aliyuncontainerservice.oss-cn-hangzhou-internal.aliyuncs.com/hpc/1.12.3/attachNodeScript | sudo -H bash -s c46e89653c69d0fcedc156d19b6e2156f5668001 --advertise-interface bond0.700 --override-kernel-check
```

5. 访问https://cs.console.aliyun.com/#/node查看节点列表



6. 现在您的机器上已经成功安装并运行了docker和nvidia-docker服务，可以开始利用容器服务部署应用了。

前提

本文档仅适用于使用了阿里云容器服务的北京HPC实例。

由于脚本仅支持 Linux 64 位系统，且需要公网访问，因此建议您在您的 Linux ECS 跳板机上运行部署脚本。

获取访问地址

在阿里云容器服务控制台中，单击左侧**集群**，单击您需要部署监控的集群实例右侧的**管理**,进入集群管理页面:

在这个截图里，访问地址是tcp://112.74.142.6:16639，但是我们只需要地址112.74.142.6:16639,下面的例子会使用这个地址。

下载和放置证书。

在上一步的页面中单击**下载证书**按钮，开始下载证书。下载到的文件是certFile.zip，请将该文件上传到跳板机上。如果您本地电脑是 Linux 系统，您可以在本地执行以下命令以上传文件：

```
scp /您的下载目录 / certFile.zip 您的账户名@{您的跳板机}:~/.
```

登录到跳板机上，创建证书文件夹，解压证书：

```
mkdir /certs
mv certFile.zip /certs
cd /certs
unzip certFile.zip
```

下载脚本 setup-monitoring脚本：

```
curl -o setup-monitoring.sh http://aliyuncontainerservice.oss-cn-hangzhou.aliyuncs.com/toolings/setup-monitoring.sh
```

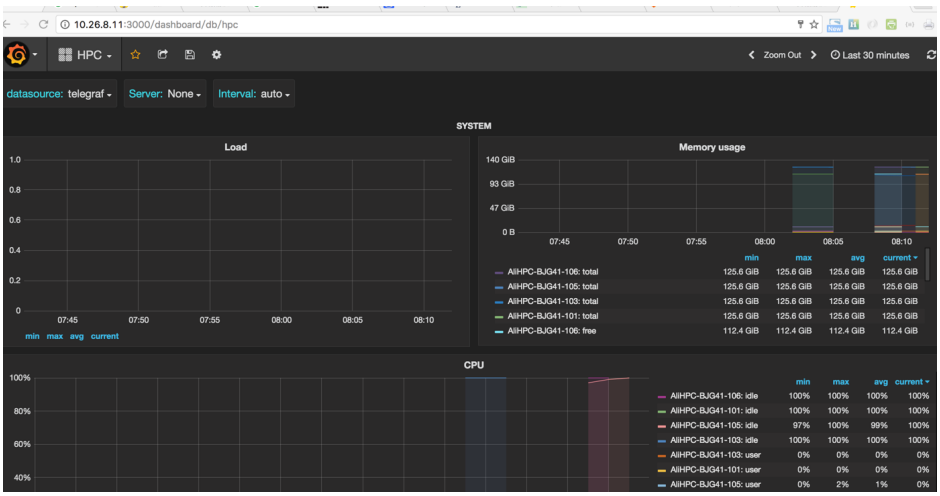
运行 setup-monitoring 脚本，需要指定您在第2步中获取的地址、证书解压目录、您自行设定的监控密码：

```
chmod u+x setup-monitoring.sh
./setup-monitoring.sh <SWARM_MASTER> <cert-dir> <admin-password>
#比如
./setup-monitoring.sh 112.74.142.6:16639 /certs hpc12345
Not Found
Not Found
access http://10.26.8.11:3000 with username admin and password hpc12345
```

脚本最后的输出即是访问监控面板的方式。注意此处的 IP 是您 HPC 的内网地址，您可能需要在跳板机上做一下代理设置才能访问到。我们推荐简单的设置方式是在跳板机上执行以下命令即可：

```
ssh -4 -g -f -L 3000:您的HPC地址:3000 -N localhost
```

此时您就可以在本地浏览器以 http:跳板机IP:3000 作为地址来访问监控了，账户名为 admin，密码为运行脚本时设定的密码。



在集群环境中，物理机上的数据卷有很大的局限性：

- 容器在机器间迁移时，数据无法迁移
- 不同机器之间不能共享本地数据卷

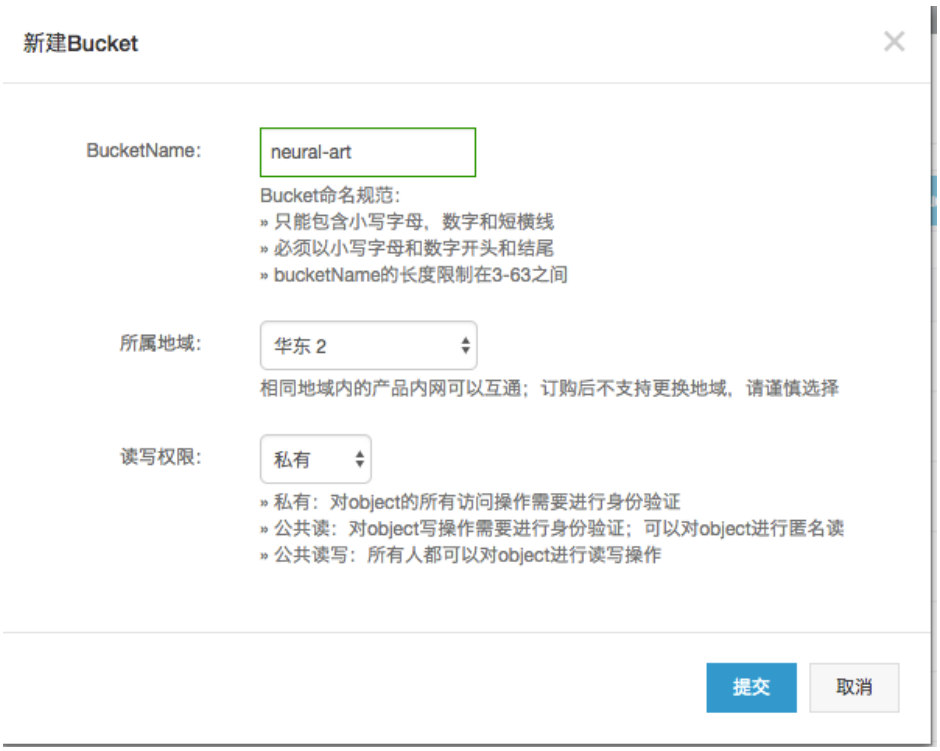
另一方面，在高性能计算和深度学习的场景里，需要使用大量的数据。上传、共享和下载数据也需要借助OSSFS和NAS这类数据服务。

为了解决这一矛盾，阿里云容器服务提供第三方数据卷。将各种云存储包装成数据卷，可以直接挂载在容器上，并在容器重启、迁移时自动重新挂载。目前支持 OSSFS 数据卷，您可以通过容器服务快速挂载数据卷。

注意：目前北京HPC仅支持OSS数据卷，后续会提供NAS的支持。

创建OSS bucket

首先您需要创建一个和容器服务处于同一区域中的bucket，并且将权限设置为私有。这样容器应用可以通过内网地址来访问bucket中存储的文件数据，提升访问速度并节省公网带宽：



注：更多的细节可以查看oss新手入门。

利用OSS客户端上传和下载数据

建议您在本地电脑上下载云市场中的OSS图形客户端, 您在本地电脑上只需要通过图形用户界面上简单的拖拽实现文件的上传和下载。

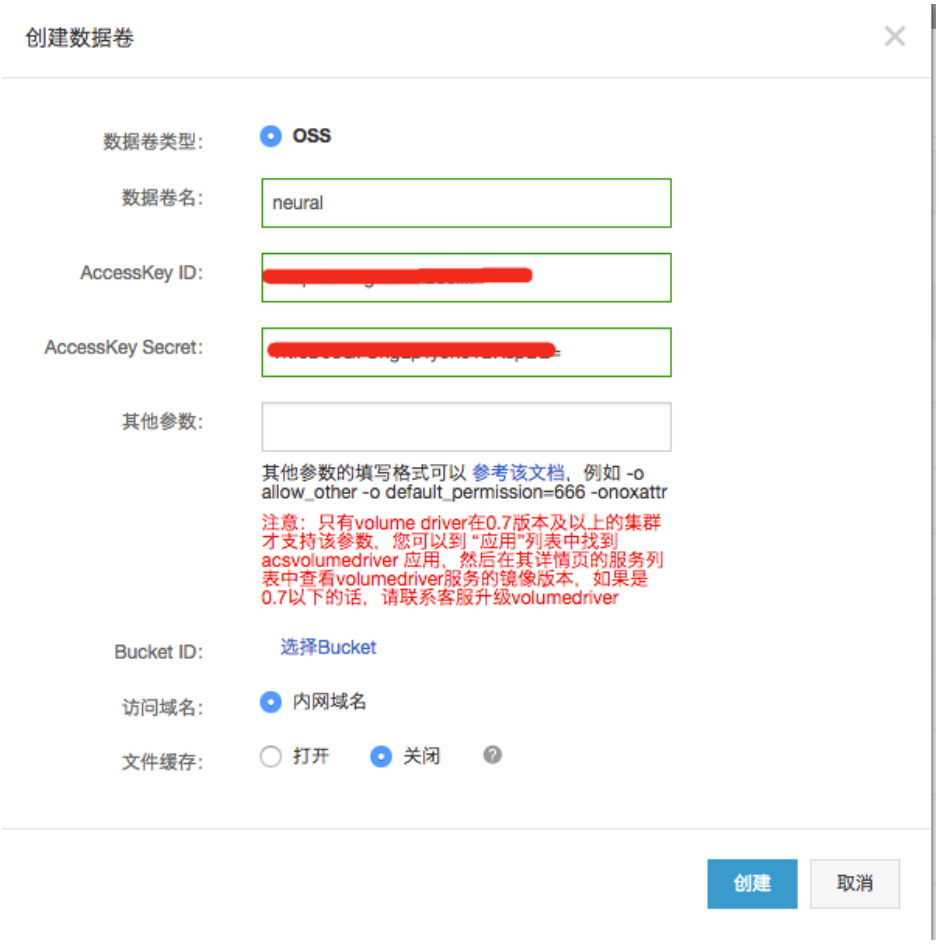
在容器服务中创建OSS数据卷

数据卷是Docker提供的容器储存模型，可以实现容器和数据生命周期的解耦，当容器被删除或重建之后数据依然存在；提供了可扩展的插件机制，支持不同的存储实现。

阿里云容器服务内置了针对阿里云的数据卷驱动，支持不同类型的云存储服务：包括NAS(文件存储服务NFS)，OSS(对象存储服务，OSSFS)和云盘（即将推出）。关于数据卷的详细信息可以参阅**帮助文档**。目前阿里云北京HPC容器服务支持OSS数据卷。

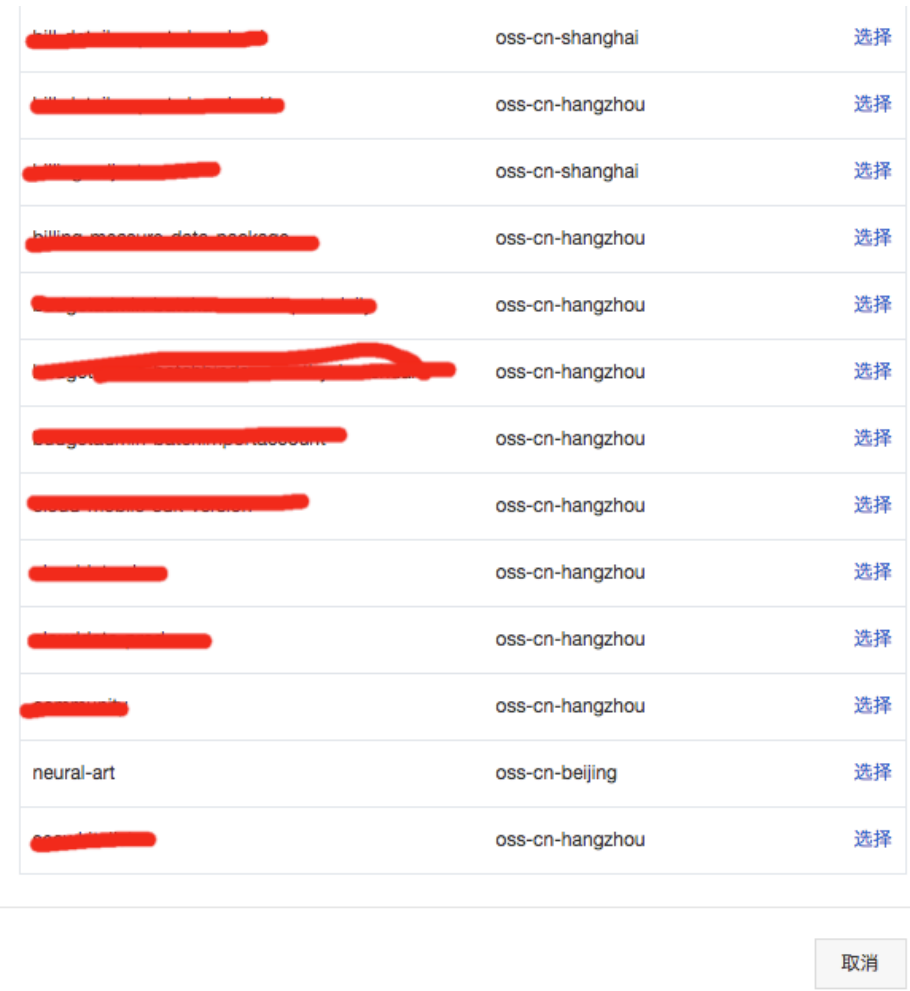
点击容器服务控制台左侧**数据卷**，展开数据卷功能。目前仅有OSS数据卷是开放的。

OSS数据卷将oss的bucket包装成数据卷，创建界面如下：



- **数据卷名**：数据卷的id，在集群内唯一。
- **AccessKeyId**、**AccessKeySecret**：访问OSS所需的AK，可以从**AK控制台**获取。
- **访问域名**：如果bucket跟ECS在同一个区（Region），选内网域名；否则选外网域名。
- **文件缓存**：如果需要在不同机器间同步同一个文件的修改（比如在A机器中修改文件，在B机器中读取修改后的内容），请关闭文件缓存。但请注意，关闭文件缓存将导致ls文件夹变得很缓慢，尤其是同一个文件夹下文件比较多时。没有上述需求时，请打开文件缓存，提高ls的速度。

3. 点击**选择Bucket**按钮，选择您之前创建neural-art数据卷, 再点击**创建**：



4. 这样就能看到您创建的OSS数据卷已经成功的出现在了数据卷列表中：

5. 此时您集群中的每台HPC机器（包括未来新加入集群的HPC机器）就能在/mnt目录下看到您挂载的数据卷了。您也可以在使用容器服务的时候指定Volumes了。

```
version: '2'
labels:
aliyun.project_type: "batch"
services:
neural:
image: registry-internal.cn-beijing.aliyuncs.com/cheyang/neural-style:latest
volumes:
- neural-art:/neural
```