阿里云
容器服务

深度学习解决方案

文档版本：20180808

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。
斜体	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *`[-all|-t]`*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *`{stand | slave}`*

目录

1 Swarm 集群

1.1 概述

基于阿里云强大计算能力的深度学习解决方案，为您提供一个低门槛、开放、端到端的深度学习服务平台。方便数据科学家和算法工程师快速开始利用阿里云的资源（包括 ECS 云服务器、GPU 云服务器、高性能计算 HPC、对象存储 OSS、Elastic MapReduce、负载均衡等服务）执行数据准备、模型开发、模型训练、评估和预测等任务。并能够方便地将深度学习能力转化为服务 API，加速与业务应用的集成。

具体而言，该深度学习解决方案具备以下特性：

- 简单：降低构建和管理深度学习平台的门槛。
- 高效：提升 CPU、GPU 等异构计算资源的使用效率，提供统一的用户体验。
- 开放：支持 TensorFlow、Keras、MXNet 等多种主流深度学习框架，同时用户也可使用定制的环境。
- 全周期：提供基于阿里云强大服务体系构建端到端深度学习任务流程的最佳实践。
- 服务化：支持深度学习能力服务化，与云上应用的轻松集成。

开始使用

1. 环境准备。

   创建容器集群。如果您需要使用 OSS 数据卷存储数据，您还需要 创建 *OSSFS* 数据卷。
2. 创建 *Jupyter* 环境，并 在 *Jupyter* 环境使用 *Git* 管理代码。
3. 运行 单机模型训练 或者 分布式模型训练，导出模型。
4. 利用导出的模型，执行 使用 *TensorFlow Serving*。

1.2 环境准备

1.2.1 创建数据卷

OSSFS 是阿里云官方提供的基于 FUSE 的文件系统（项目主页见 *https://github.com/aliyun/ossfs*）。OSSFS 数据卷可以将 OSS 的 Bucket 包装成数据卷。

由于数据需要经过网络同步到云端，OSSFS 在性能和功能上与本地文件系统有差距。请不要把数据库等重 IO 应用、日志等需要不断改写文件的应用运行在 OSSFS 上。OSSFS 比较适合多容器之间共享配置文件，或者附件上传等没有改写操作的场景。

OSSFS 和本地文件系统具体差异如下所示：

- 随机或者追加写文件会导致整个文件的重写。
- 因为需要远程访问 OSS 服务器，元数据操作（例如 list directory）性能较差。
- 文件/文件夹的重命名操作不是原子的。
- 多个客户端挂载同一个 OSS Bucket 时，需要您自行协调各个客户端的行为。例如，避免多个客户端写同一个文件等等。
- 不支持硬链接（hard link）。

前提条件

您的集群必须满足以下两个条件，才可以开通数据卷功能：

- 集群 Agent 的版本为 0.6 或更高。

  您可以在集群列表页面查看您的 Agent 的版本。选择所需的集群，单击右侧的 更多 > 升级**Agent**。

  如果您的 Agent 版本过低，请先升级您的 Agent。有关如何升级 Agent，参见 升级 *Agent*。

- 集群里部署了 acsvolumedriver 应用。建议您将 acsvolumedriver 升级到最新版本。

  您可以通过升级系统服务部署和升级 acsvolumedriver 应用。详细操作参见 升级系统服务。

  说明：

  升级或重启 acsvolumedriver 时，使用 OSSFS 数据卷的容器会重启，您的服务也会重启。

操作步骤

步骤 1 创建 OSS Bucket

登录 对象存储管理控制台，创建一个 Bucket，参见 创建存储空间。

本示例创建了一个位于华南 1 地域的 Bucket。

步骤 2 创建 OSSFS 数据卷

1. 登录 容器服务管理控制台。
2. 单击左侧导航栏中的 数据卷。
3. 选择要创建数据卷的集群（本示例中为 tfoss）并单击页面右上角的 创建。

4. 在弹出的对话框中，选择 数据卷类型 为 **OSS**，设置数据卷参数并单击 创建。容器服务会在集群的所有节点上创建名称相同的数据卷。

- **数据卷名**：数据卷的名称。数据卷名在集群内必须唯一。
- **AccessKey ID**、**AccessKey** Secret：访问 OSS 所需的 AccessKey。您可以从 阿里云账号 *AccessKey* 控制台 获取。
- **Bucket** ID：您要使用的 OSS bucket 的名称。单击 选择**Bucket**，在弹出的对话框中选择所需的 bucket（本示例中为 tensorflow-samples）并单击 选择。
- **访问域名**：选择 **vpc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（比如在机器 A 中修改文件，在机器 B 中读取修改后的内容），请关闭文件缓存。

> 说明：
> 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。因此，没有上述需求时，请打开文件缓存，提高 ls 的速度。

后续操作

创建数据卷之后，您可以在您的应用中使用创建的数据卷。有关如何在应用中使用数据卷，参见 使用第三方数据卷。

1.2.2 创建容器集群

深度学习解决方案支持使用云服务器 ECS 容器集群或者 GPU 服务器容器集群。本文档以 GPU 服务器容器集群为例进行说明。

> 说明：
> 有关如何创建 ECS 容器集群，参见 创建集群。

使用限制

- 目前，容器服务仅支持在华南 1 、华东 2、华北 2 和美西 1 地域创建 GN4 型 GPU 云服务器集群。
- 目前，GN4 型 GPU 云服务器只支持专有网络（VPC）。

提前准备

目前，按量付费的 GPU 计算型 GN4 云服务器需要申请工单开通。如果您需要使用按量付费的 GPU 计算型 GN4 云服务器，请按照如下内容 提交 *ECS* 工单。

我需要申请按量付费的GPU计算型gn4，请帮忙开通，谢谢。

操作流程

1. 登录 容器服务管理控制台。
2. 在 Swarm 菜单下，单击左侧导航中的 集群，单击右上角的 创建**Swarm**集群。

3. 设置集群的基本信息。本示例中创建一个位于华南 1 地域名为 EGS-cluster 的集群。

- **集群名称**：要创建的集群的名称。可以包含 1~64 个字符，包括数字、中文字符、英文字符和连字符（-）。

  > **说明：**
  > 集群名称在同一个用户和同一个地域下必须唯一。

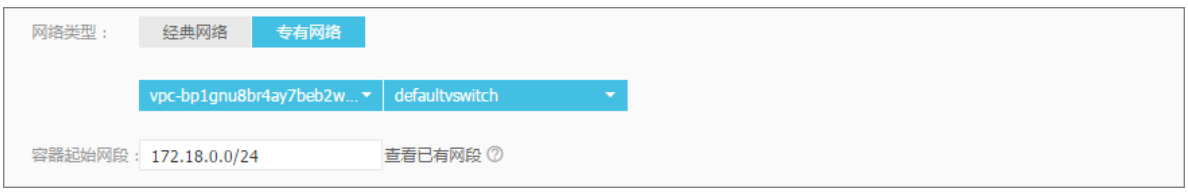
- **地域**：所创建集群将要部署到的地域。选择 **华南 1**、**华东 2**、**华北 2** 或 **美国西部 1**。

  > **说明：**
  > 目前，仅支持在华南 1、华东 2 、华北 2 和 美国西部 1 地域创建 GN4 型 GPU 云服务器集群。

- **可用区**：集群的可用区。

  > **说明：**
  > 您可以根据您的服务器分布情况，选择不同的地域和可用区。

4. 设置集群的网络类型为 **专有网络** 并配置相关信息。



专有网络 *VPC* 支持您基于阿里云构建一个隔离的网络环境，您可以完全掌控自己的虚拟网络，包括自由 IP 地址范围、划分网段、配置路由表和网关等。

专有网络需要您指定一个 VPC、一个 VSwitchId 和容器的起始网段（Docker 容器所属的子网网段，为了便于 IP 管理，每个虚拟机的容器属于不同网段，容器子网网段不能和虚拟机网段冲突）。

为了防止网络冲突等问题，建议您为容器集群建立属于自己的 VPC/VSwitchId。

5. 添加节点。

您可以在创建集群的同时创建若干个节点，或者创建一个零节点集群并添加已有云服务器。有关如何添加已有云服务器的详细信息，参见 添加已有云服务器。

- 创建节点

  1. 设置节点的操作系统。

     目前支持的操作系统包括 Ubuntu 14.04 64 位和 CentOS 7.4 64 位。

  2. 设置云服务器的实例规格。

     - 实例系列 选择 系列**III**。
     - 实例规格 选择 **32**核 48GB（ecs.gn4.8xlarge） 或 **56**核 96GB（ecs.gn4.14xlarge）。

       说明：

       如果您已经通过了 GN4 型 GPU 云服务器的使用申请，但是未找到这两种实例规格，说明目前这两种规格的实例没有资源，建议晚些时候或者次日再尝试购买。

您可选择实例的数量，并指定数据盘的容量（云服务器默认带有 20G 大小的系统盘）和登录密码。

> **说明：**
>
> - 如果您选择了数据盘，它会被挂载到 */var/lib/docker* 目录，用于 Docker 镜像和容器的存储。
> - 从性能和管理考虑，建议您在宿主机挂载独立的数据盘，并利用 Docker 的 volume 对容器的持久化数据进行管理。

- 添加已有节点

  您可以单击下边的 **选择已有实例** 将已有的云服务器添加到集群中，或者直接单击 **创建集群** 等集群创建完成后再通过集群列表页面添加已有云服务器。

6. 配置 EIP。

   当您将网络类型设置为 VPC 时，容器服务会默认给每一个专有网络下的云服务器配置一个 *EIP*。如果不需要，您可以勾选 **不配置公网EIP** 复选框，但是需要额外配置 SNAT 网关。

7. 创建一个负载均衡实例。

目前创建集群会默认创建一个负载均衡实例。您可以通过这个负载均衡实例访问集群内的容器应用。所创建的负载均衡实例为按量付费实例。

8. 单击 创建集群。

后续操作

您可以查看集群创建日志。在 集群列表 页面，选择所创建的集群并单击 查看日志。

1.3 模型开发

1.3.1 创建 Jupyter 环境

准备工作

在运行模型训练任务之前，请确认以下工作已经完成：

- 创建包含适当数量弹性计算资源（ECS 或 GPU 服务器）的容器集群。参见 创建容器集群。
- 如果您需要使用对象存储服务（OSS）保存用于模型训练的数据，您需要使用相同账号创建 OSS Bucket；然后在上面的容器集群中创建数据卷，用于将 OSS Bucket 作为本地目录挂载到执行训练任务的容器内。参见 创建数据卷。

约定

为了方便您的应用代码读取训练数据，输出训练日志，训练卷中的数据会存放在 `/input` 目录，用户代码需要从该目录中读取数据。

操作流程

1. 登录 容器服务管理控制台。
2. 在 Swarm 菜单下，单击左侧导航栏中的 镜像与方案 > 解决方案。
3. 在 模型开发 框中单击 创建。

4. 设置创建 Jupyter 环境的基本信息。

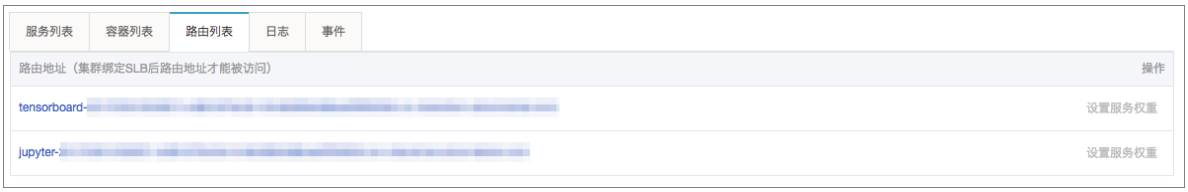
   - **集群**：所创建模型开发应用将要部署到的集群。本示例中为 EGS-cluster。
   - **应用名**：所创建应用的名称。名称可以包含 1~64 个字符，包括数字、英文字符和连字符（-），且不能以-开头。
   - **训练框架**：所支持的训练框架包括 TensorFlow，Keras 以及不同 Python 版本。
   - **GPU数量**：所使用的 GPU 数量，如果为 0 表示不使用 GPU。
   - **数据卷名**：指定为用于存储训练数据的数据来源，可以选择对象存储服务在该集群中创建的数据卷的名称；也可以选择本地目录，但要求填写绝对路径；或者选择 **不使用数据卷**。本示例中使用名为 tfoss 的数据卷。
   - **Jupyter密码**：登录 Jupyter 所用的密码。
   - **训练监控**：是否使用 TensorBoard 监控训练状态；一旦选择监控，请指定训练日志的路径，并保证与训练代码中日志输出的路径一致。
   - **启用SSH登录**：选择是否启用 SSH 方法访问服务。勾选此选项后，您需要填写您的 **SSH密码**。

   说明：

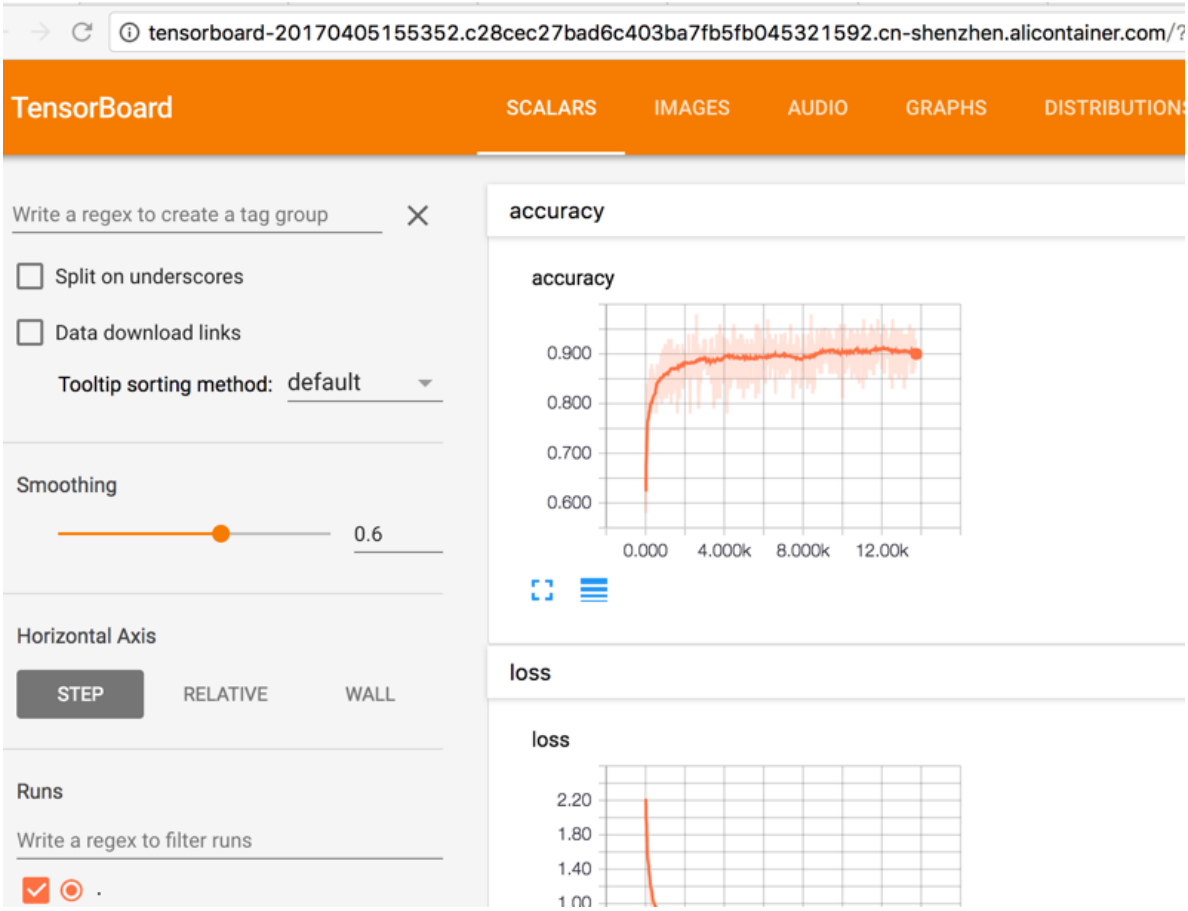
   有关如何通过 SSH 方法访问服务，参见 *通过 SSH 访问 Jupyter 服务*。

5. 设置完毕后，单击 确定。
6. 在应用列表页面，选择创建的应用， 单击应用名称进去。

7. 单击 路由列表，可以看到两个链接，分别是以 jupyter 和 tensorboard 开头的链接。



8. 单击 jupyter 开头的链接，并且输入 jupyter 的密码，就能进入 jupyter 环境。
9. 单击 tensorboard 开头的链接，查看训练结果。

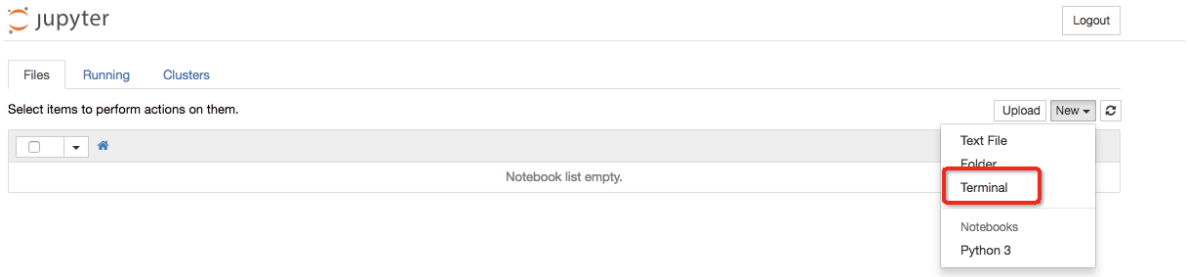


10.分布式存储中的训练数据都存储在本地的 /input 文件夹下，您可以从 /input 下读取数据。

1.3.2 在 Jupyter 环境使用 Git 管理代码

操作步骤

1. 在 Jupyter 主页面创建 Terminal。



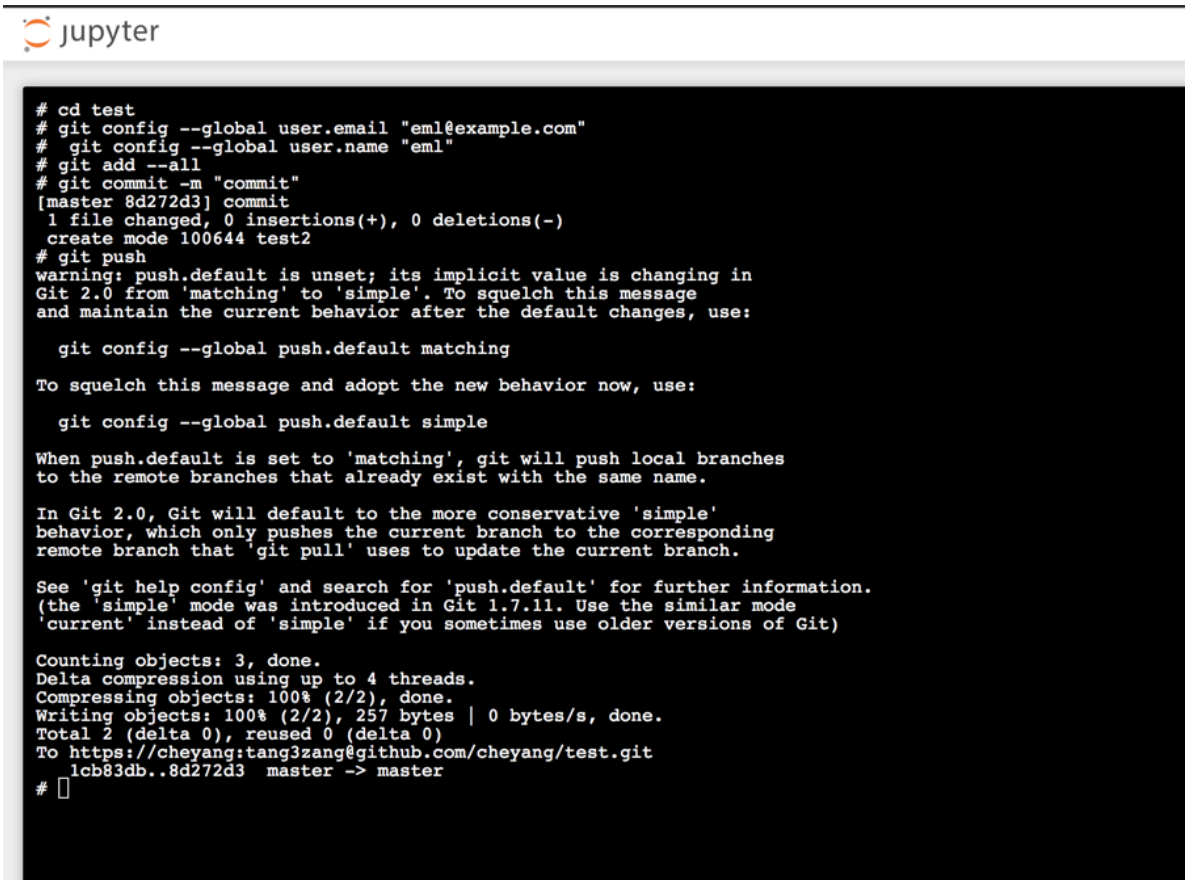
2. 在 Terminal 内执行利用 **`git clone`** 下载应用代码。

```
git clone https://{id}:{password}@github.com/{id}/test.git
Cloning into 'test'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
```

```
Checking connectivity... done.
```

其中，{id} 为 GitHub 用户名，{password} 为 GitHub 的密码。

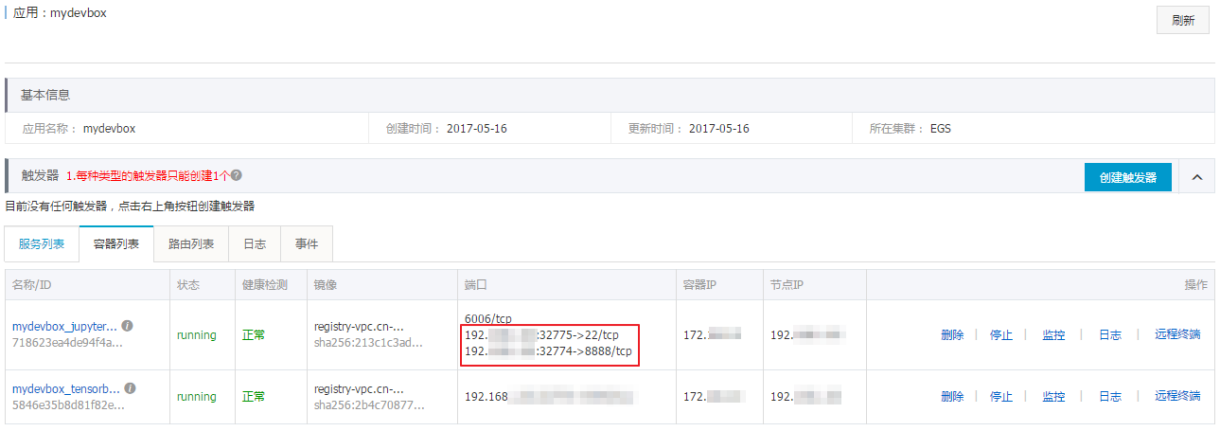
3. 切回到 Jupyter 主页面，可以看到应用代码已经在列，这样就可以利用 Jupyter 开发相应的代码。
4. 可以回到 Terminal，利用 Git 提交代码。



1.3.3 通过 SSH 访问 Jupyter 服务

如果您在 创建 *Jupyter* 环境 环境时选择了 启用**SSH**登录，您可以通过本文档中介绍的方法 SSH 访问您的 Jupyter 服务。

本示例中，SSH 访问的端口映射为 192.***.*.**.32775->22/tcp。其中，192.***.*.** 是 ECS 实例的私网 IP，您无法通过此 IP 访问您的 Jupyter 服务。您需要通过 ECS 实例的弹性公网 IP 从外部 SSH 访问 Jupyter 服务。



您可以登录到 *ECS* 管理控制台 查看该 ECS 实例绑定的弹性公网 IP。本示例中，弹性公网 IP 为 39.***.**.236。

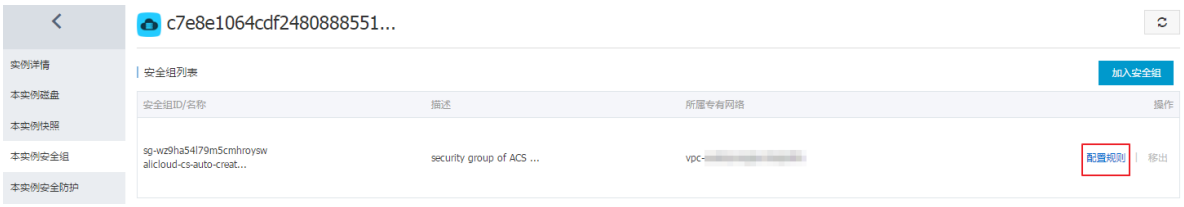
> **说明：**
>
> 如果 ECS 实例没有绑定弹性公网 IP，您需要先为其绑定弹性公网 IP，参见 管理弹性公网 *IP*。

此外，为了通过 ECS 实例的弹性公网 IP 从外部 SSH 访问您的 Jupyter 服务，您还需要配置安全组规则开放 32775 端口。

配置安全组规则

1. 登录 *ECS* 管理控制台。
2. 单击左侧导航栏中的 实例，选择地域（本示例中为 华南 1），选择容器服务应用 mydevbox 对应的 ECS 实例。

3. 单击右侧的 更多 > 安全组配置 。
4. 选择容器服务集群对应的安全组并单击 配置规则。



5. 单击 添加安全组规则，填写规则信息并单击 确定。

SSH 访问 Jupyter 服务

Linux 环境下访问

如果您使用的是 Linux 机器，你只需要运行以下命令即可 SSH 访问您的 Jupyter 服务。

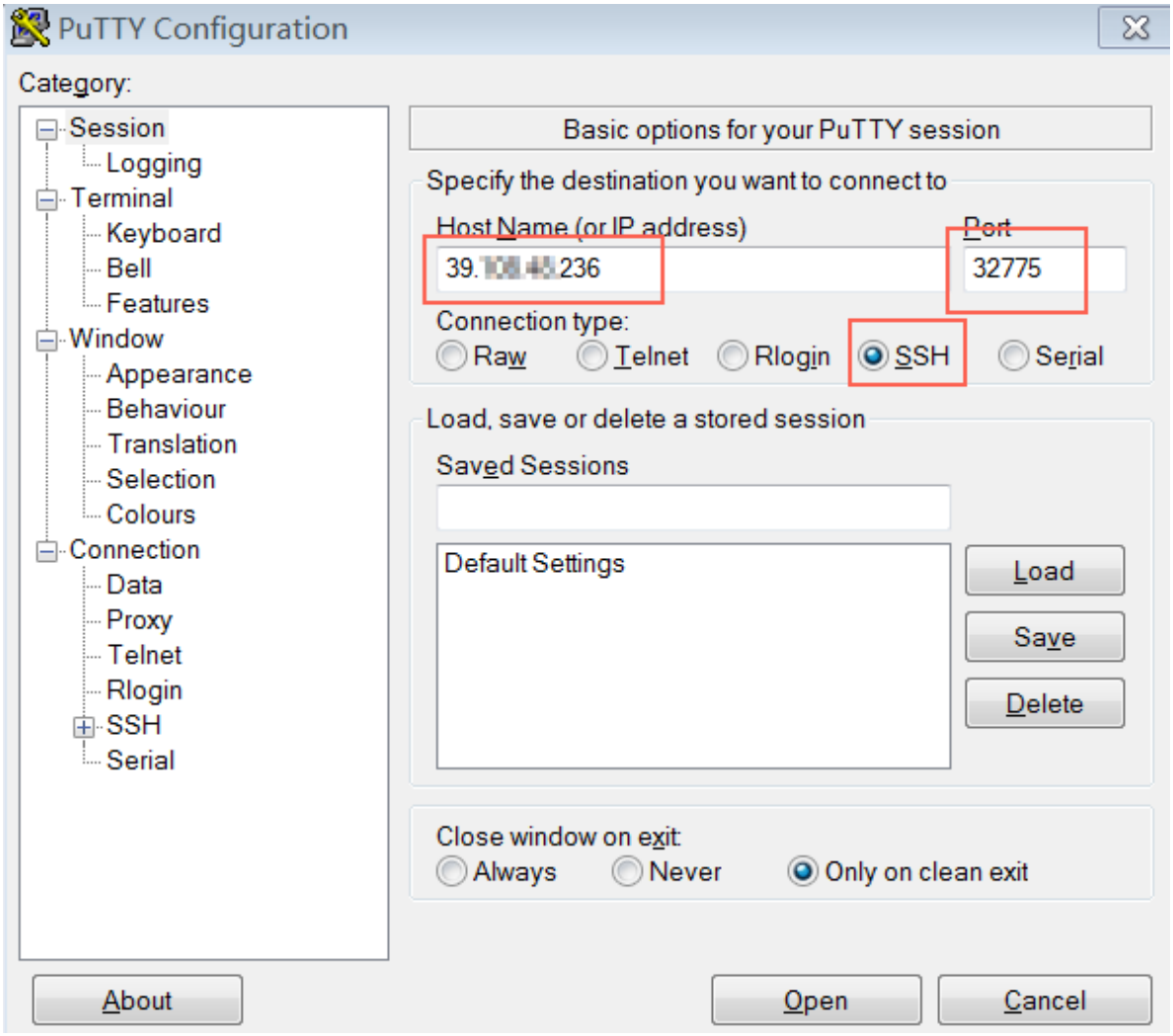
```
ssh -p 32775 root@39.***.**.236
```

其中，`32775` 为 SSH 访问的端口；`39.***.**.236` 是您的 ECS 实例绑定的弹性公网 IP。

Windows 环境下访问

1. 运行 PuTTY，配置 Session。

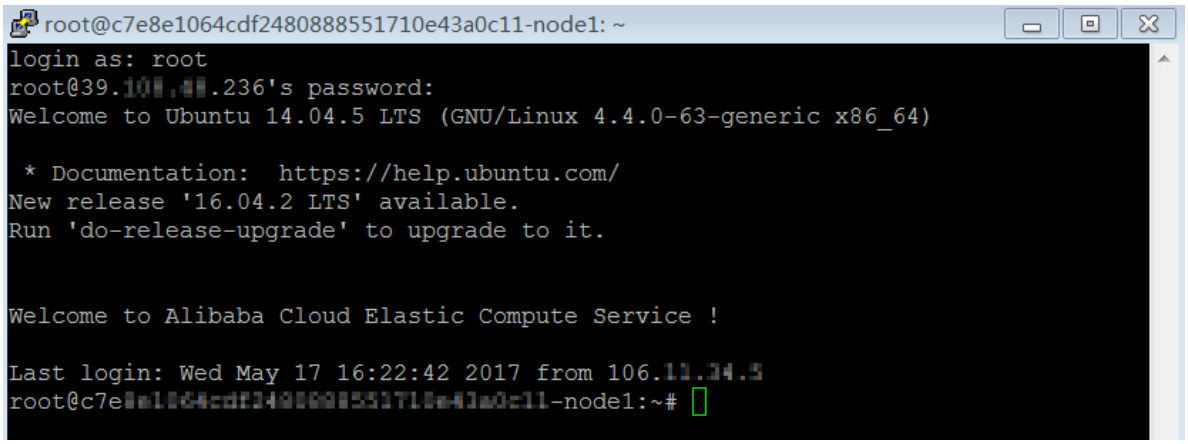
   设置 IP 地址（ECS 实例的弹性公网 IP）、端口（SSH 访问的端口，本示例为 32775）和连接方式（**SSH**），并单击 **Open**。



2. 在弹出的会话窗口登录并访问服务。

   输入登录账号 `root` 并输入您在 创建 *Jupyter* 环境 时 启用**SSH**登录 所输入的 **SSH**密码。

   可以看到您成功访问到 Jupyter 服务。

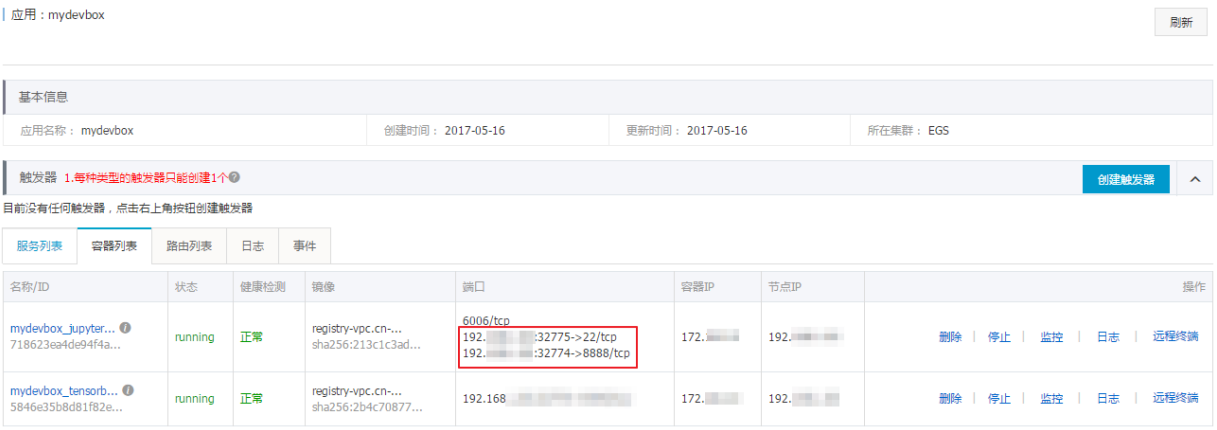


1.3.4 通过端口转发方式访问 Jupyter 服务

通过端口转发方式访问 Jupyter 服务具有以下优点，但是设置较复杂。

- 您不需要购买负载均衡实例，可以节省您的费用。
- 您不需要在官网上开放端口即可从外部访问 Jupyter 服务。

下面的示例假设通过本地的 12345 端口进行转发 SSH 访问您的 Jupyter 服务。



步骤 1 打通 SSH Tunnel

Mac OS X 和 Linux 环境下

运行以下命令打通本地端口和 ECS 实例的连接。

```
ssh -ND 12345 root@39.***.**.236
```

`12345` 是您要使用的本地端口，您可以自定义。`39.***.**.236` 是您的 ECS 实例绑定的弹性公网 IP。

说明：

您可以登录 *ECS* 管理控制台 查看 ECS 实例绑定的弹性公网 IP。如果您的实例未绑定弹性公网 IP，您需要先为其绑定弹性公网 IP，参见 管理弹性公网 *IP*。

Windows 环境下

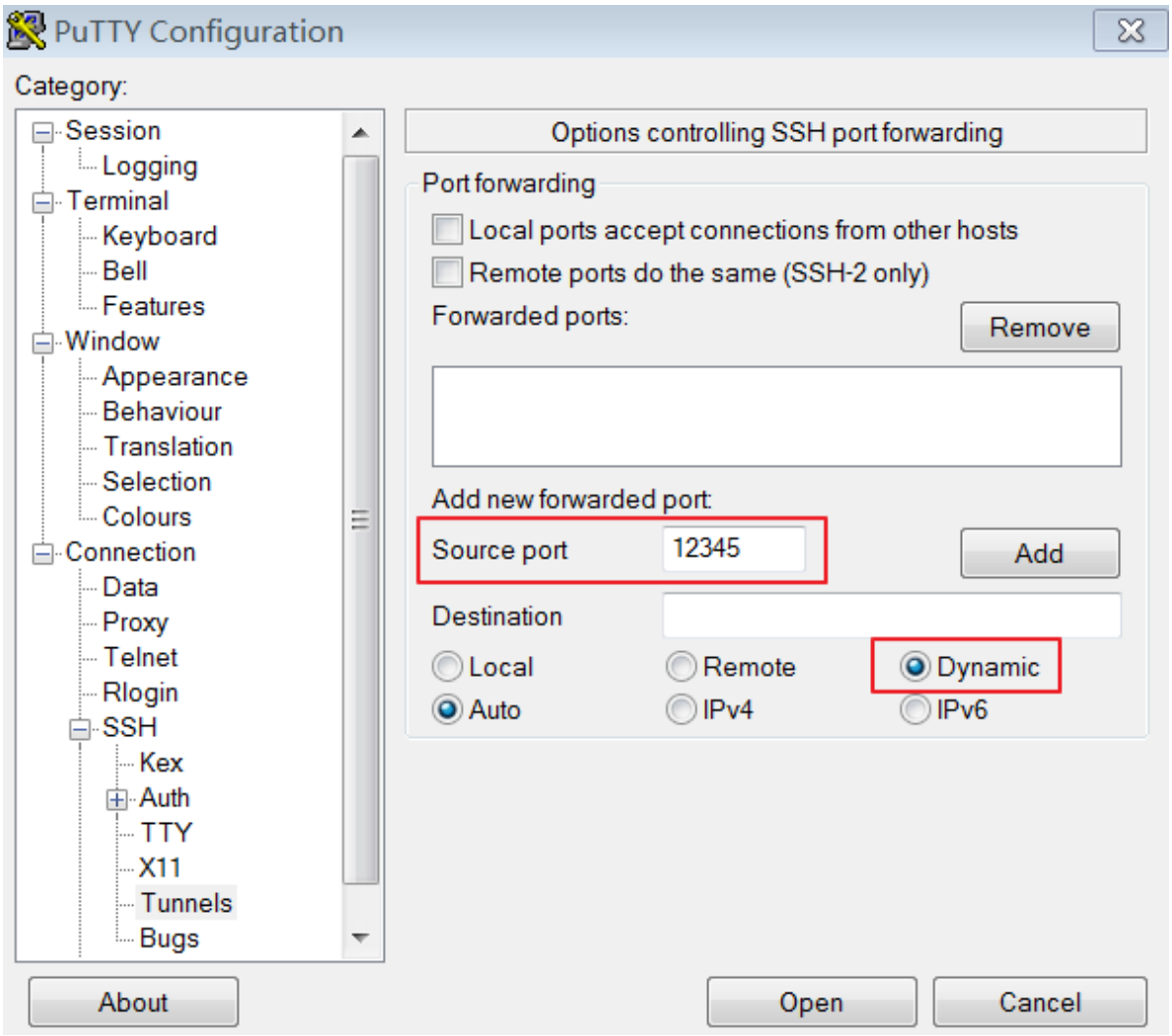
1. 配置本地端口跳转。

   运行 PuTTY 并配置 SSH **Tunnels**。

   a. 填写 **Source Port**。本示例中为 12345。

   b. 选择 **Dynamic**。

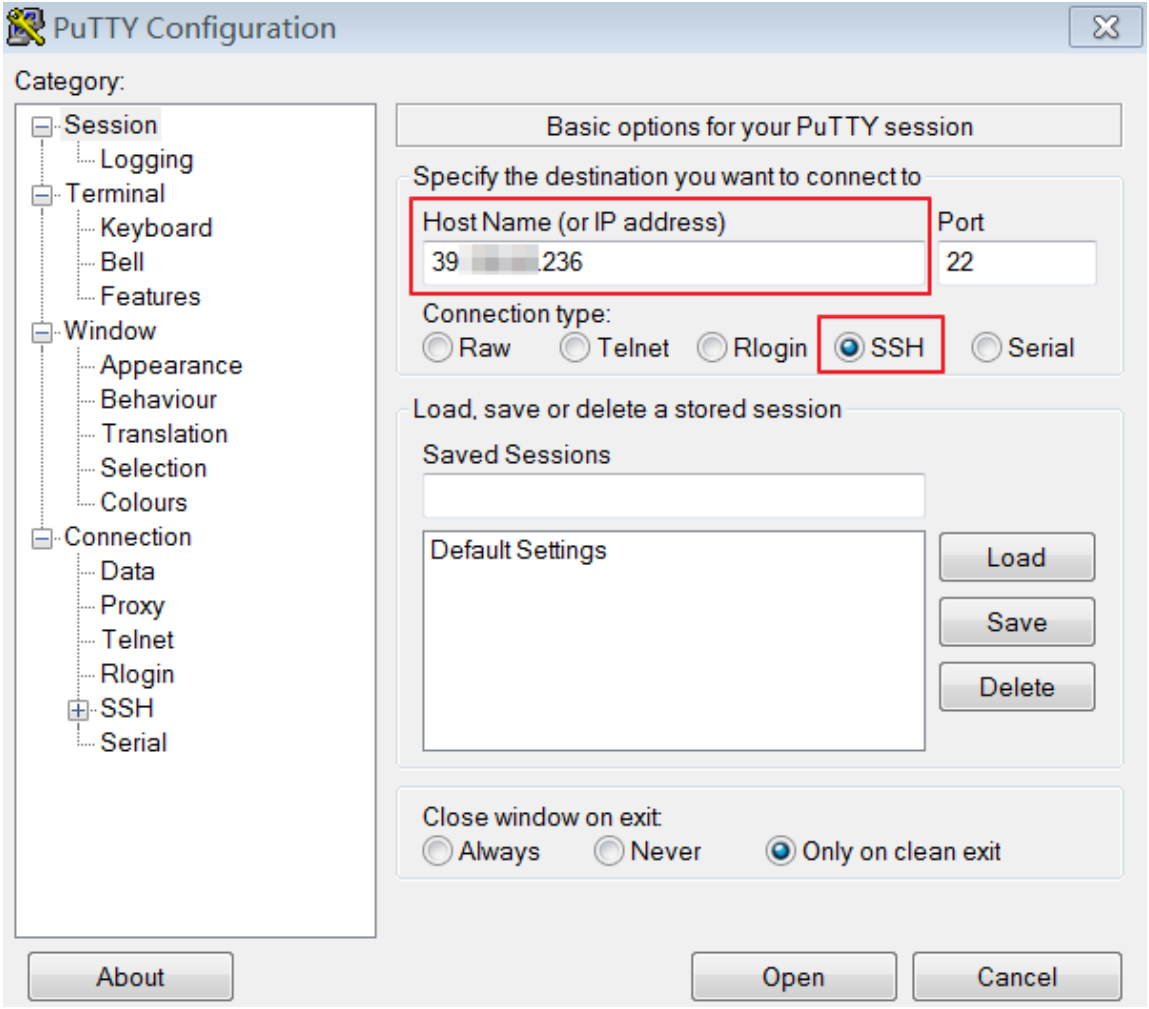
   c. 单击 **Add**。



2. 登录 ECS 实例。

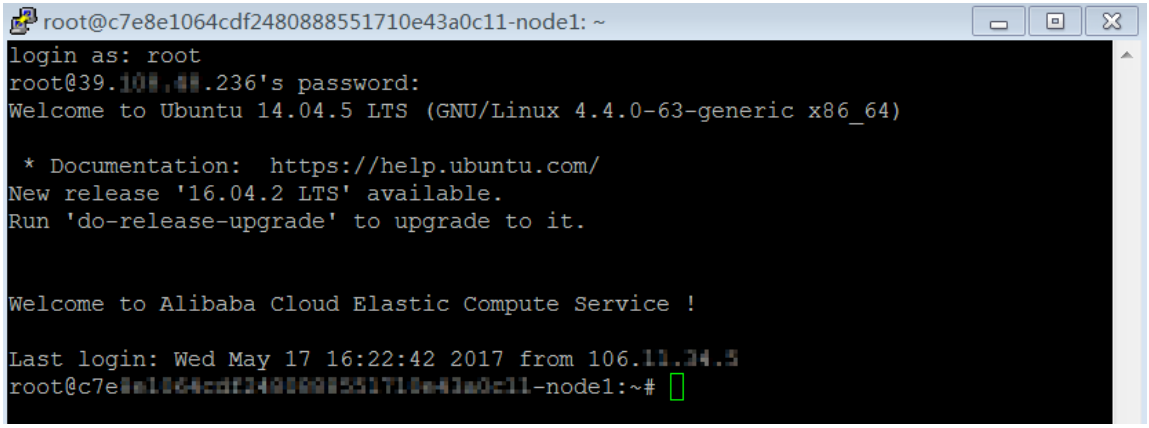
   a. 运行 PuTTY，配置 **Session** 并单击 **Open**。

      填写 IP 地址，即 ECS 实例的弹性公网 IP，本示例中为 39.***.**.236。



b. 在弹出的会话框里输入您的登录账号和 ECS 实例的登录密码。

您成功登录到 ECS 实例上。



步骤 2 配置浏览器的网络连接

Firefox 浏览器

如果您使用的是 Firefox 浏览器，打开浏览器，单击 工具 > 选项 > 高级 > 网络 选项卡 > 单击 连接 对应的 设置 按钮 > 设置 **SOCKS** 主机。

Chrome 浏览器

运行以下命令。

```
Chrome --proxy-server="socks5://localhost:12345" --host-resolver-rules
="MAP * 0.0.0.0 , EXCLUDE localhost" --user-data-dir=/tmppath/
```

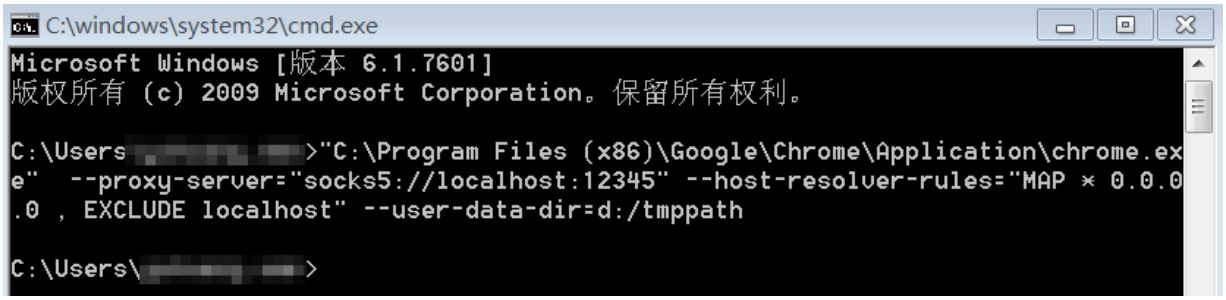
其中：

- `12345` 是您所使用的本地端口。
- 若是 Windows 系统，这里的 `/tmppath/` 可以写成类似 `d:/tmppath`；若是 Linux 或者 Mac OS X，可以直接写成 `/tmp/`。

在不同的操作系统中，Chrome 的位置不同，请参见下表：

操作系统	**Chrome** 的位置
Mac OS X	*`/Applications/Google\ Chromapp/Contents/MacOS/Google\ Chrome`*
Linux	*`/usr/bin/google-chrome`*
Windows	*`C:\Program Files (x86)\Google\Chrome\Application\chrome.exe`*

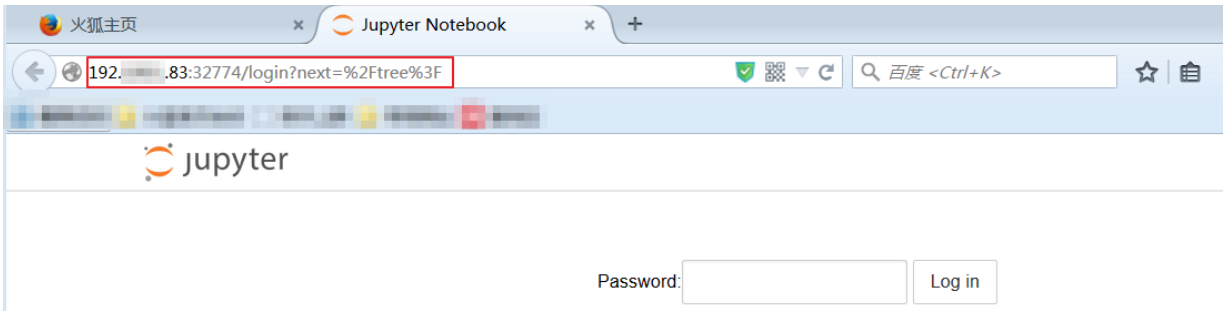
以 Windows 系统为例：



步骤 3 访问 **Jupyter** 服务

在您的浏览器中输入 Jupyter 服务的访问地址。本示例中为 192.***.*.83:32774。即可成功地从外部 SSH 访问到 Jupyter 服务。

Firefox：



Chrome：

1.4 模型训练

1.4.1 单机模型训练

利用阿里云提供的弹性计算资源和存储服务，执行您的模型训练代码，快速开始进行单机版训练迭代。训练过程中，您可以随时查看日志和监控训练状态。

准备工作

在运行模型训练任务之前，请确认以下工作已经完成：

- 创建包含适当数量弹性计算资源（ECS 或 EGS）的容器集群。创建步骤请参考 创建容器集群。
- 如果您需要使用对象存储服务（OSS）保存用于模型训练的数据，您需要使用相同账号创建 OSS Bucket；然后在上面的容器集群中创建数据卷，用于将 OSS Bucket 作为本地目录挂载到执行训练任务的容器内。参见 创建数据卷。
- 将模型训练代码同步到有效的 GitHub 代码仓库中。

约定

为了方便您的训练代码读取训练数据，输出训练日志，保存训练迭代状态数据（checkpoint），请注意以下约定。

- 训练数据将会自动存放在 `/input` 目录，且保持与 OSS 中路径一致。用户代码需要从该目录中读取训练数据。
- 代码输出的所有数据都应保存到 `/output` 目录，包括日志，checkpoint 文件等。所有保存在 `/output` 目录中的文件将以同样的目录结构被自动同步到用户的 OSS Bucket。
- 如果训练代码中需要使用特别的 Python 依赖库，请将所有依赖写入名为 `requirements.txt` 的配置文件，并保存在 GitHub 仓库中代码根目录下。

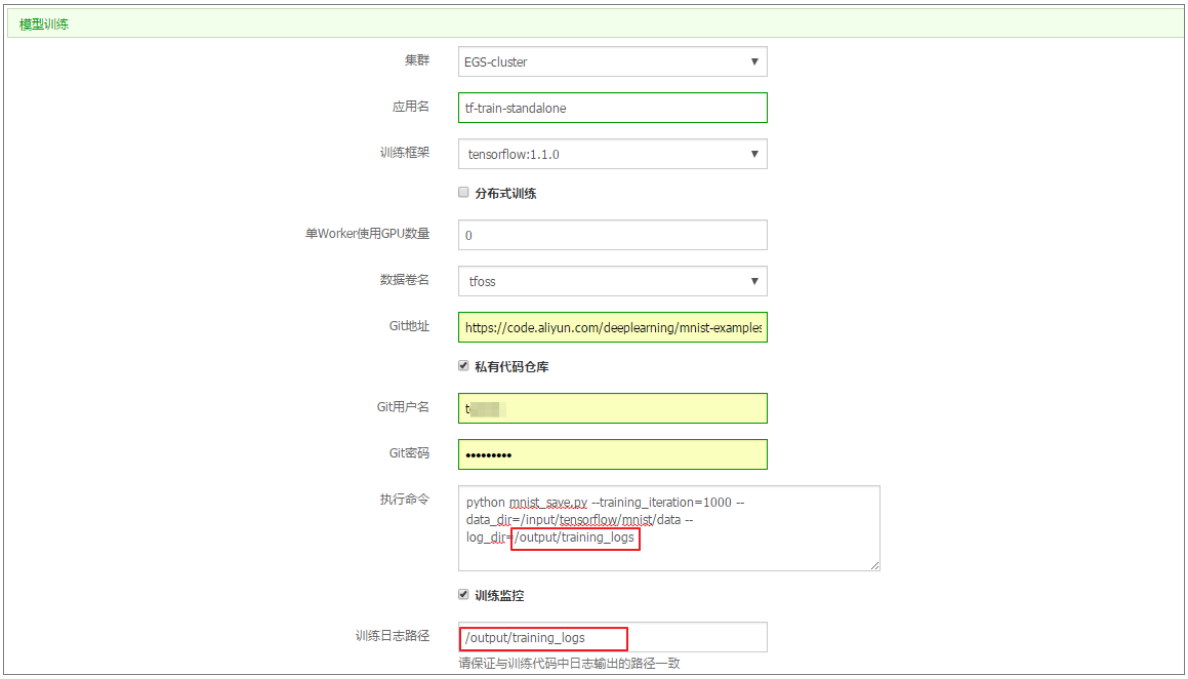
操作流程

1. 登录 容器服务管理控制台。
2. 在 Swarm 菜单下，单击左侧导航栏中的 **镜像与方案 > 解决方案**。

3. 在 模型训练 框中单击 创建。

4. 设置单机训练任务的基本信息。

进入模型训练页面，填写以下信息用于配置训练任务。



- 集群：选择将要运行单机模型训练任务的容器集群。
- 应用名：为运行单机模型训练任务所创建的应用名称，例如 tf-train-standalone。名称可以包含 1~64 个字符，包括数字、英文字母和连字符（-）。
- 训练框架：选择用来进行模型训练的框架。目前支持的框架包括 TensorFlow、Keras、Python 以及自定义镜像，选择自定义镜像要求您输入合法的镜像地址。

- **分布式训练**：选择是否执行分布式训练任务。这里不要勾选，则执行单机训练。
- **单Worker使用GPU数量**：指定所使用的 GPU 数量，如果为 0 表示不使用 GPU，那么将使用 CPU 训练。
- **数据卷名**： 指定为用于存储训练数据的数据来源，可以选择对象存储服务在该集群中创建的数据卷的名称；也可以选择本地目录，但要求填写绝对路径；或者选择 **不使用数据卷**。本示例中使用名为 tfoss 的数据卷。
- **Git地址**：指定训练代码所处的 GitHub 代码仓库地址。

> **说明：**
> 目前仅支持 HTTP 和 HTTPS 协议，不支持 SSH 协议。

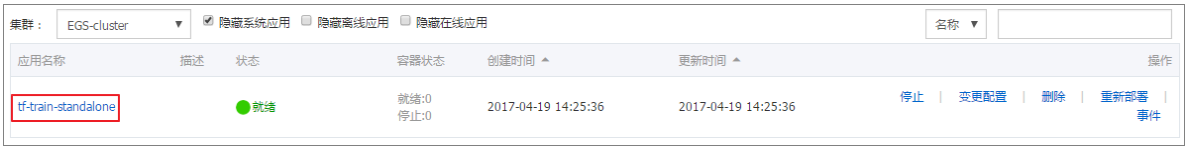
- **私有代码仓库**：如果您使用私有代码仓库，请勾选此选项并设置 **Git用户名**（您的 GitHub 账号）和 **Git密码**。
- **执行命令**：指定如何执行上述代码的命令，以进行模型训练。

> **说明：**
> 如果你选择的是支持 Python3 的框架，请在命令行中直接调用 `python3`，而不是 `python`。

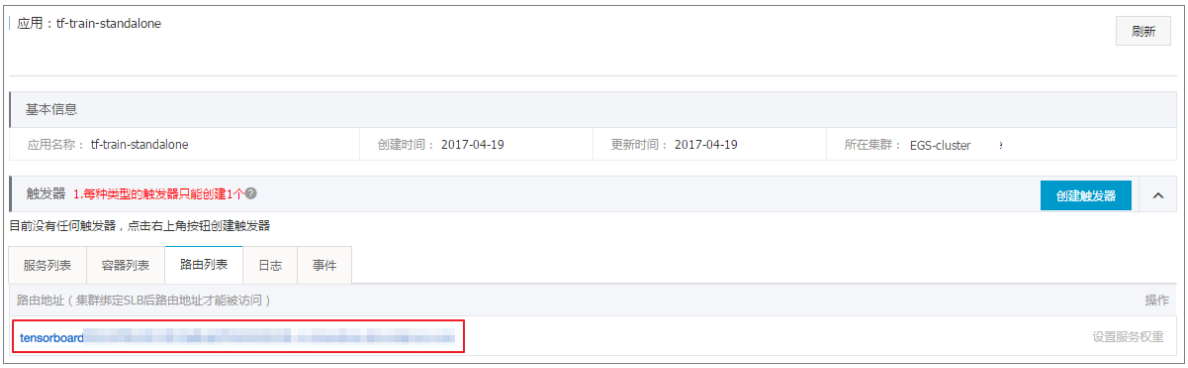
- **训练监控**：是否使用 TensorBoard 监控训练状态；一旦选择监控，请指定合法的训练日志路径，并确保与训练代码中日志输出的路径一致。例如，指定 **训练日志路径** 为 *`/output/training_logs`*，则用户代码也应该将日志输出到同样的路径下。图中示例是通过指定用户代码命令行参数 **`--log_dir`** 保证这一点的。

> **说明：**
> 这里的训练日志指的是使用 TensorFlow API 输出的供 Tensorboard 读取的事件文件，以及保存了模型状态的 checkpoint 文件。

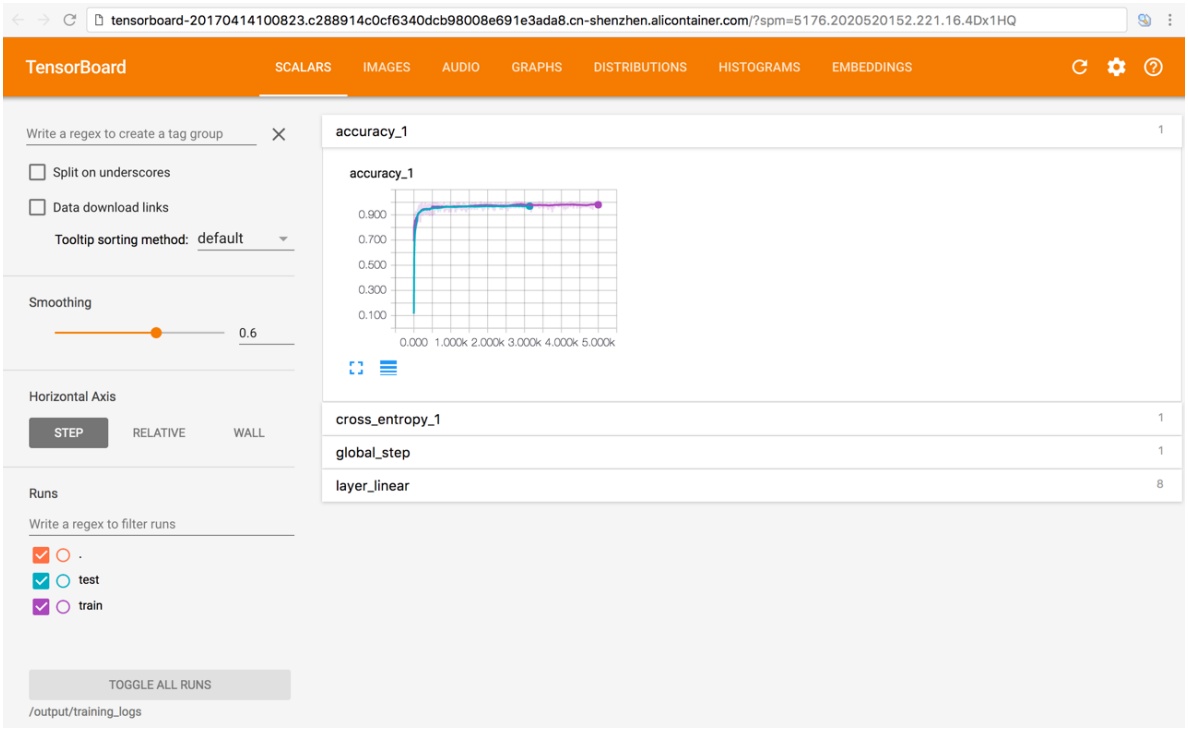
5. 设置完毕后，单击 **确定**，创建执行模型训练任务的应用。
6. 单击左侧导航栏中的 **应用**，单击上面创建的应用 **tf-train-standalone**。



7. 单击 **路由列表**，可以看到一个以 tensorboard 开头的链接，用来打开 TensorBoard 监控页面。



单击这个链接，可以查看训练监控数据。



8. 单击 日志，查看执行训练任务过程中，用户代码输出到标准输出/标准错误的日志内容。

> **说明：**
>
> 为了运行训练任务应用，一般会自动创建多个服务容器，分别运行不同的程序分支。比如 worker 容器一般用来运行用户的模型代码，tensorboard 容器用来运行 TensorBoard 训练监控。具体生成的服务 / 容器，可以单击 服务列表，或 容器列表 查看。

可以按服务容器名称筛选某个容器，查看该容器运行程序输出到 stdout/stderr 的详细日志，比如 `tf-train-standalone_worker_1`。也可以按时间和显示数量，过滤要查看的日志内容。同样，也可以选择下载日志文件到本地。

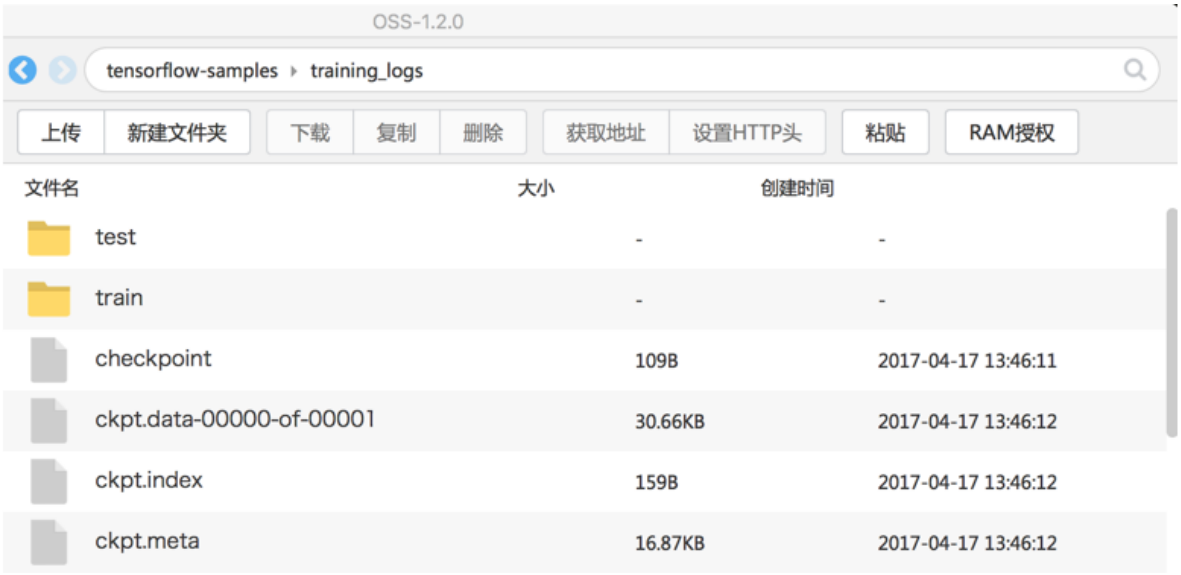
说明：

这里提到的标准输出/标准错误日志，请区别于上述 Tensorboard 事件文件和 checkpoint 文件。

9. 单击 服务列表，如果看到 worker 服务的状态为“已停止”，说明此次训练任务已经执行结束。

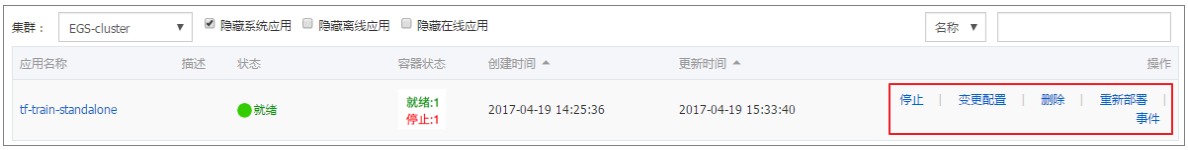
10. 通过 OSS 客户端可以看到训练结果自动保存在 OSS Bucket 中了。

训练结束后，系统自动将所有保存在本地 `/output` 路径中的文件，复制到用户给定数据卷所对应的对象存储服务（OSS）Bucket 中。通过 OSS 客户端，可以看到本示例中用户代码写入 `/output` 的事件文件和 checkpoint 文件都已经备份到 OSS Bucket 中了。



11.管理训练任务。

如果需要停止、重启、删除（比如为了释放计算资源给新的训练任务）某个训练任务，只需回到**应用 > 应用列表**页面。找到对应的应用，在右侧进行操作即可。



说明：

利用本节描述的模型训练服务，您不仅可以从零开始训练一个模型，同样也可以在一个已有模型的基础（checkpoint）之上，使用新的数据继续训练（比如 fine tuning)。利用已创建的应用，可以不断通过更新配置的方式调整超参数，进行迭代训练。

1.4.2 分布式模型训练

利用阿里云提供的弹性计算资源和存储服务，执行用户的模型训练代码，快速开始进行分布式训练。训练过程中，您可以控制如何分配计算资源（CPU， GPU），随时查看日志和监控训练状态，并将训练结果备份到存储服务中。

利用本文档描述的模型训练服务，您不仅可以从零开始训练一个模型，同样也可以在一个已有模型的基础（checkpoint）之上，使用新的数据继续训练（比如 fine tuning)。利用已创建的应用，可以不断通过更新配置的方式调整超参数，进行迭代训练。

准备工作

在运行模型训练任务之前，请确认以下工作已经完成：

- 创建包含适当数量弹性计算资源（ECS 或 EGS）的容器集群。创建步骤请参考 创建容器集群。
- 如果您需要使用对象存储服务（OSS）保存用于模型训练的数据，您需要使用相同账号创建 OSS Bucket；然后在上面的容器集群中创建数据卷，用于将 OSS Bucket 作为本地目录挂载到执行训练任务的容器内。参见 创建数据卷。
- 将模型训练代码同步到有效的 GitHub 代码仓库中。

约定

为了方便用户的训练代码读取训练数据，输出训练日志，保存训练迭代状态数据（checkpoint），请注意以下约定。

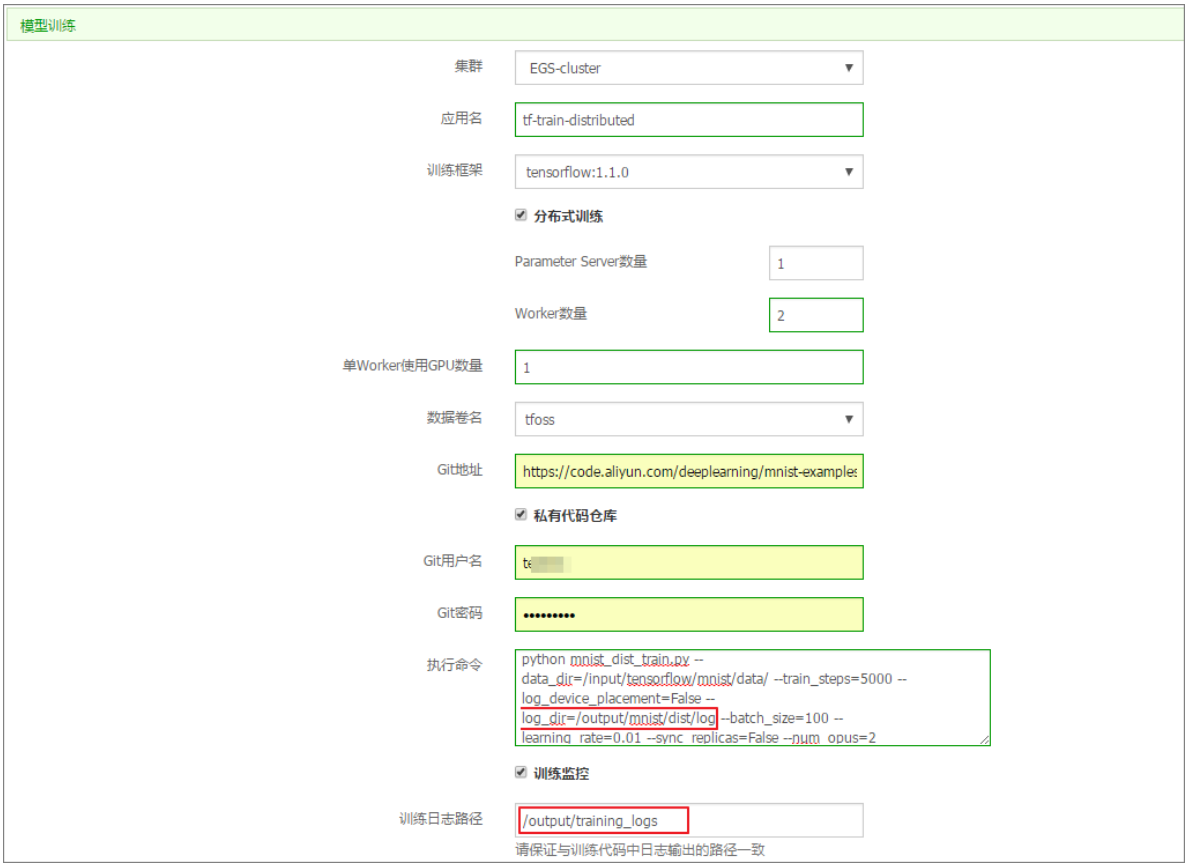
- 训练数据将会自动存放在 `/input` 目录，且保持与 OSS 中路径一致。用户代码需要从该目录中读取训练数据。
- 代码输出的所有数据都应保存到 `/output` 目录，包括日志，checkpoint 文件等。所有保存在 `/output` 目录中的文件将以同样的目录结构被自动同步到用户的 OSS Bucket。如果训练代码中需要使用特别的 Python 依赖库，请将所有依赖写入名为 `requirements.txt` 的配置文件，并保存在 GitHub 仓库中代码根目录下。

操作步骤

1. 登录 容器服务管理控制台。
2. 在 Swarm 菜单下，单击左侧导航栏中的 **镜像与方案 > 解决方案**。
3. 在 **模型训练** 框中单击 **创建**。

4. 设置分布式训练任务的基本信息。

   进入模型训练页面，填写以下信息用于配置训练任务。



- 集群：选择将要运行分布式模型训练任务的容器集群。
- 应用名：为运行训练任务所创建的应用名称，例如 tf-train-distributed。名称可以包含 1~64 个字符，包括数字、英文字母和连字符（-）。

- **训练框架**：选择用来进行模型训练的框架。目前支持的框架包括 TensorFlow、Keras、Python 以及自定义镜像，选择自定义镜像要求您输入合法的镜像地址。
- **分布式训练**：选择是否执行分布式训练任务。这里要勾选，则执行分布式训练。会出现具体的配置内容，包括 Parameter Server 数量和 Worker 数量，用于指定分布式训练时参数服务器节点的数量和进行具体计算的 Worker 节点数量。示例中设置了 1 个参数服务器和 2 个Worker。
- **单Worker使用GPU数量**：指定每个 Worker 在执行训练任务时所使用的 GPU 数量，示例中选择每个 Worker 使用 1 块 GPU 卡。如果为 0 表示不使用 GPU，那么将使用 CPU 训练。
- **数据卷名**：指定为用于存储训练数据的数据来源，可以选择对象存储服务在该集群中创建的数据卷的名称；也可以选择本地目录，但要求填写绝对路径；或者选择 **不使用数据卷**。本示例中使用名为 tfoss 的数据卷。
- **Git地址**：指定训练代码所处的 GitHub 代码仓库地址。

  说明：

  目前仅支持 HTTP 和 HTTPS 协议，不支持 SSH 协议。
- **私有代码仓库**：如果您使用私有代码仓库，请勾选此选项并设置 **Git用户名**（您的 GitHub 账号）和 **Git密码**。
- **执行命令**：指定如何执行上述代码的命令，以进行模型训练。

  说明：

  如果你选择的是支持 Python3 的框架，请在命令行中直接调用 `python3`，而不是 `python`。
- **训练监控**：是否使用 TensorBoard 监控训练状态；一旦选择监控，请指定合法的训练日志路径，并确保与训练代码中日志输出的路径一致。例如，指定 **训练日志路径** 为 *`/output/training_logs`*，则用户代码也应该将日志输出到同样的路径下。图中示例是通过指定用户代码命令行参数 **`--log_dir`** 保证这一点的。

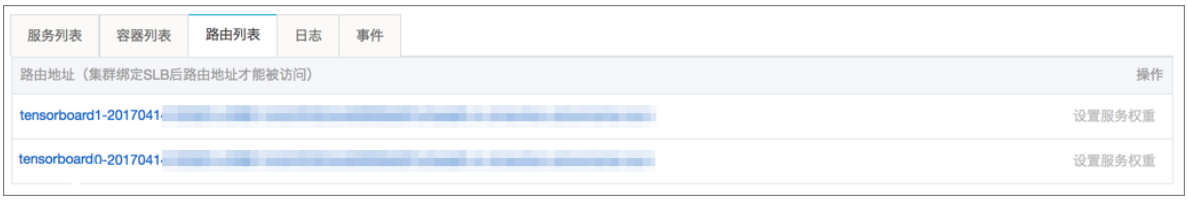
  

  说明：

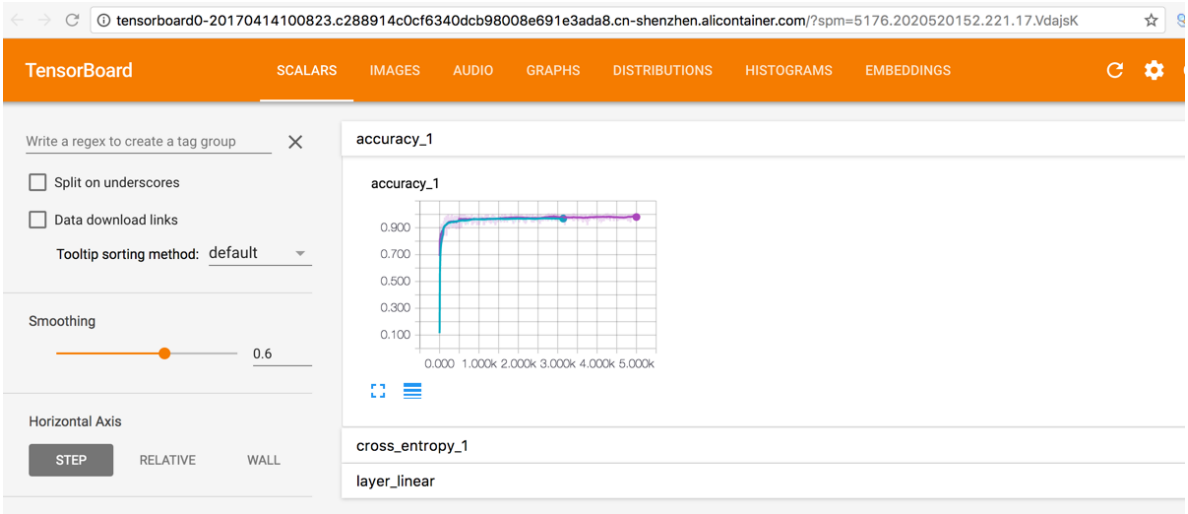
  这里的训练日志指的是使用 TensorFlow API 输出的供 Tensorboard 读取的事件文件，以及保存了模型状态的 checkpoint 文件（使用 Tensorflow 进行分布式训练时，一般由担任 Chief 角色的 Worker 负责保存 checkpoint）。

5. 设置完毕后，单击 确定，创建执行模型训练任务的应用。
6. 单击左侧导航栏中的 应用，单击上述创建的应用 **tf-train-distributed**。

7. 单击 路由列表，可以看到 2 个以 tensorboard 开头的链接，每个链接用于打开一个 Worker 节点对应的 TensorBoard 监控页面。tensorboard0 开头的链接代表 Worker 0， tensorboard1 开头的链接代表 Worker 1。



单击链接，可以查看训练监控数据。



8. 单击 日志，查看执行训练任务过程中，用户代码输出到标准输出/标准错误的日志内容。

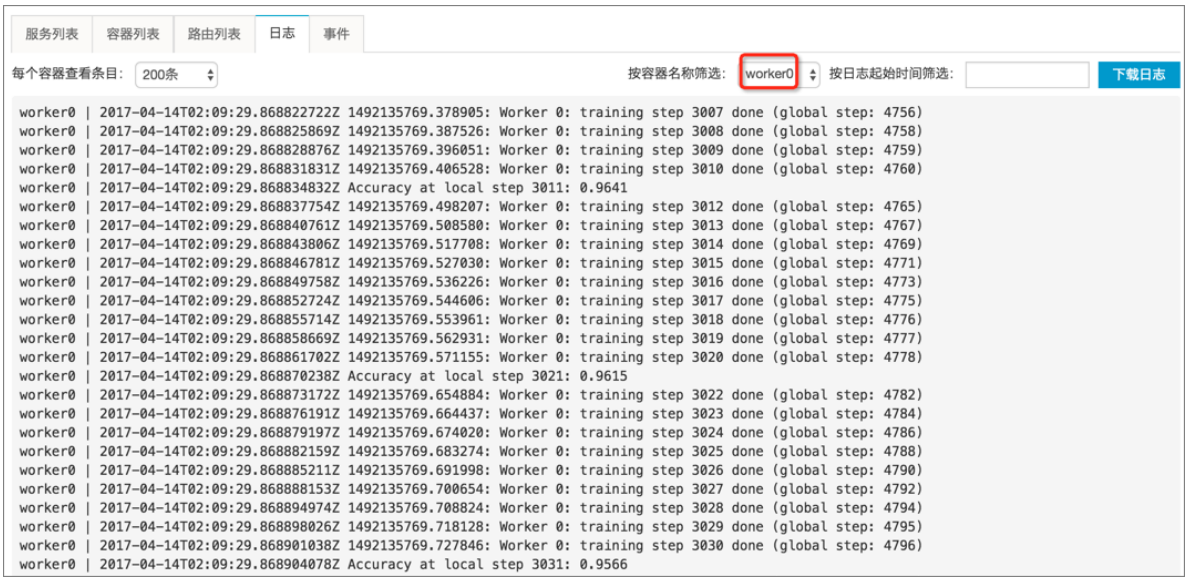
> **说明：**
> 为了运行训练任务应用，一般会自动创建多个服务容器，分别运行不同的程序分支。比如 ps 容器一般用来运行参数服务器代码，worker 容器一般用来运行模型计算代码，tensorboard 容器用来运行 TensorBoard 训练监控。具体生成的服务 / 容器，可以单击 服务列表 或 容器列表 查看。)

可以按服务容器名称筛选某个容器，查看该容器运行程序输出到 stdout/stderr 的详细日志，比如 worker0，如图所示。也可以按时间和显示数量，过滤要查看的日志内容。同样，也可以选择下载日志文件到本地。在本例中，如果查看 worker0 和 worker1 的日志，可以发现它们以数据并行的方式，针对部分训练数据分别计算同样的模型，并通过 ps0 同步梯度更新。

说明：

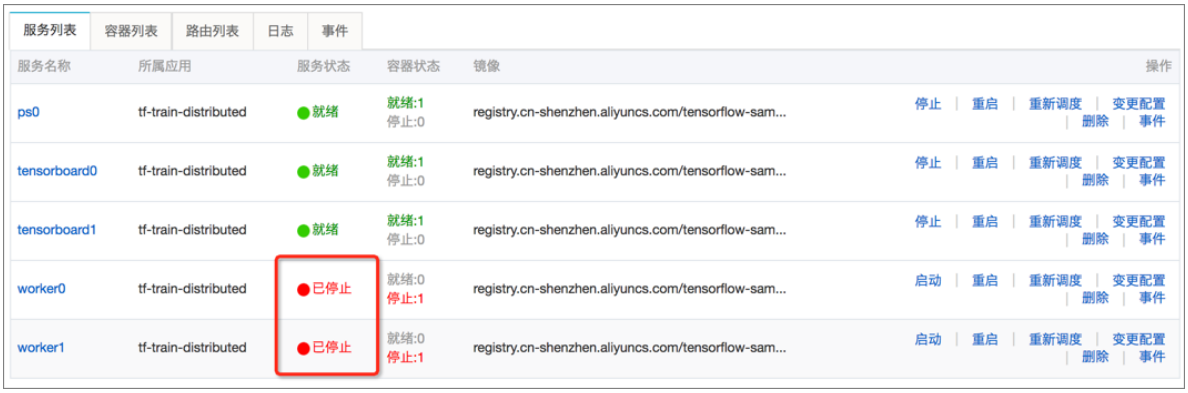
这里提到的标准输出/标准错误日志，请区别于上述 Tensorboard 事件文件和 checkpoint 文件。

worker0 日志



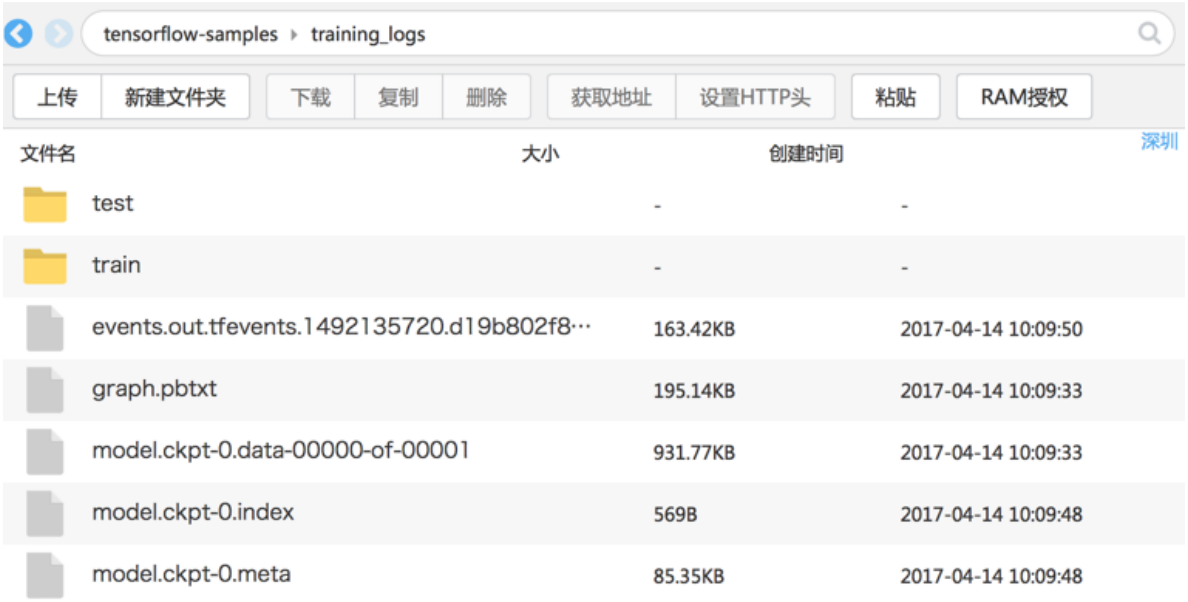
worker1 日志

9. 单击 服务列表，如果看到所有 worker 服务的状态为“已停止”，说明此次训练任务已经执行结束。



10. 通过 OSS 客户端您可以看到训练结果自动保存在了 OSS Bucket 中。

    训练结束后，系统自动将所有保存在 Worker 容器本地 `/output` 路径中的文件，复制到用户给定数据卷所对应的对象存储服务（OSS）Bucket 中。通过 OSS 客户端，可以看到本示例中用户代码写入 `/output` 的事件文件和 checkpoint 文件，都已经备份到用户的 OSS Bucket 中了。



11.管理训练任务。

如果需要停止、重启、删除（比如为了释放计算资源给新的训练任务）某个训练任务，只需回到应用 > 应用列表页面。找到对应的应用，在右侧进行操作即可。

1.5 预测服务

1.5.1 使用 TensorFlow Serving

利用阿里云弹性计算资源（ECS 或 EGS），负载均衡和对象存储服务，对于 TensorFlow 的导出模型，快速部署服务上线，用于其他应用的访问。

准备工作

在运行模型训练任务之前，请确认以下工作已经完成：

- 创建包含适当数量弹性计算资源（ECS 或 EGS）的容器集群。参见 创建容器集群。
- 使用相同账号创建对象存储服务（OSS）Bucket，使用它保存用于模型训练的数据。
- 为上面的容器集群创建数据卷，用于将 OSS Bucket 作为本地目录挂载到执行训练任务的容器内。创建方法请参考 创建数据卷。
- 了解 TensorFlow Serving 的基本概念和工作流程。参见 *Serving a TensorFlow Model*。

约定

为了简化您对于模型预测的使用，请注意以下约定。

- 在 OSS 的根目录以模型名创建文件夹。

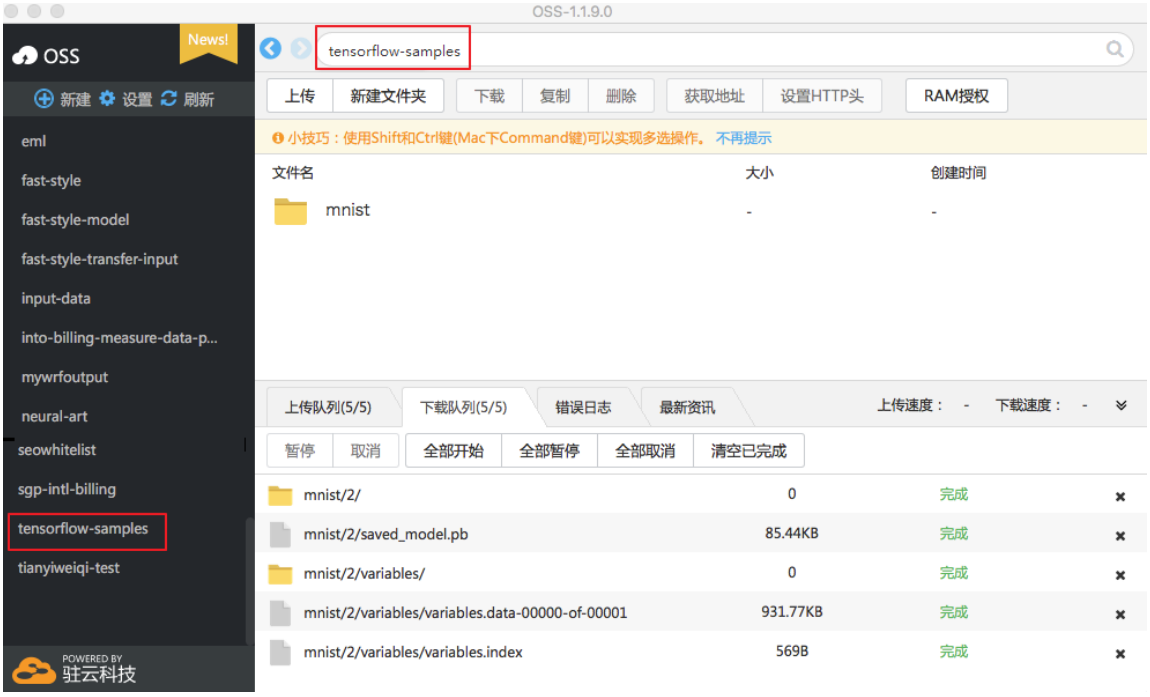
- 在模型预测中，负载均衡的前、后端端口和应用端口必须保持一致。

操作步骤

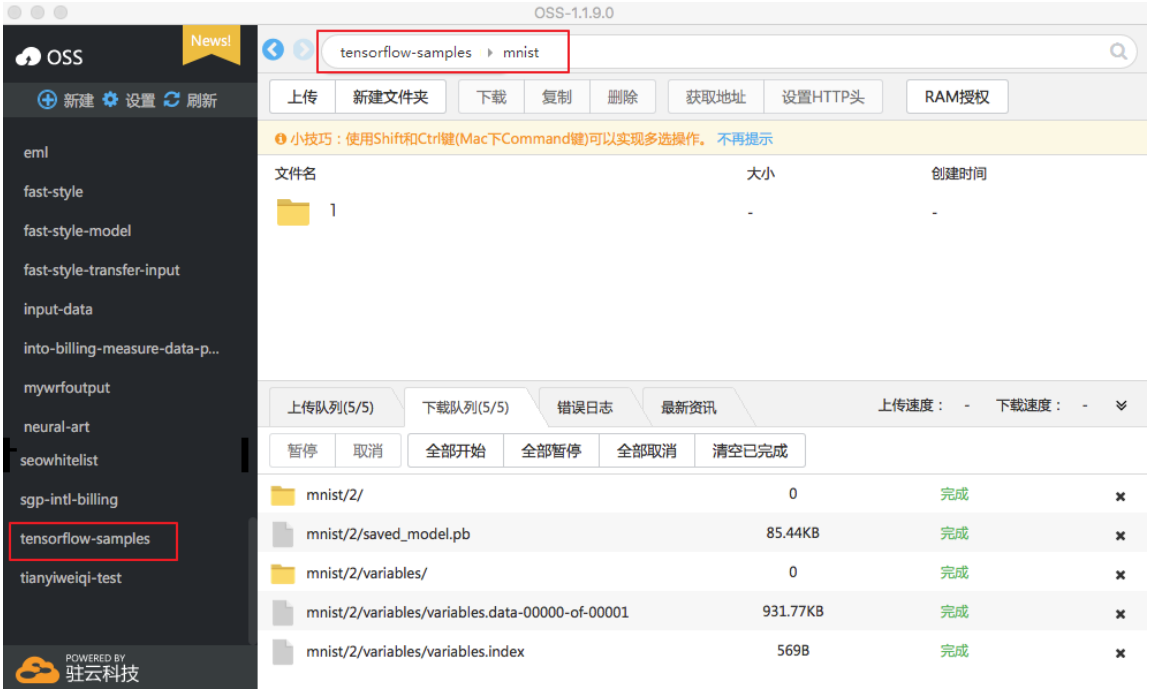
1. 将预测模型保存在共享存储中。

   可以通过在 OSS 客户端上传模型文件夹完成。

   a. 在 OSS Bucket 的根目录以模型名创建文件夹，在本例子中使用 mnist。



   b. 将带版本号的 Tensorflow Serving 模型文件夹上传到 mnist 下。



2. 配置负载均衡的监听端口。

   a. 在 负载均衡管理控制台 （单击页面右上角的 创建负载均衡）创建一个用于路由的负载均衡实例。

   本示例中选择的是公网实例，您可以根据自己的需要选择公网或者私网。

   > **说明：**
   > 由于负载均衡不支持跨地域（Region）部署，因此应选择与您所使用容器服务集群相同的地域。

   b. 返回 负载均衡管理控制台，将购买创建的负载均衡实例命名为 TensorFlow-serving。容器服务会通过该名称来引用这个负载均衡实例。

   单击左侧导航栏中的 实例管理 > 选择实例所在的地域 > 选择所需实例 > 编辑实例的名称并单击 确定。

c. 创建监听端口。

单击实例右侧的 管理 > 单击左侧导航栏中的 监听 > 单击 添加监听 > 设置监听配置。协议为 TCP，端口映射为 8000:8000。

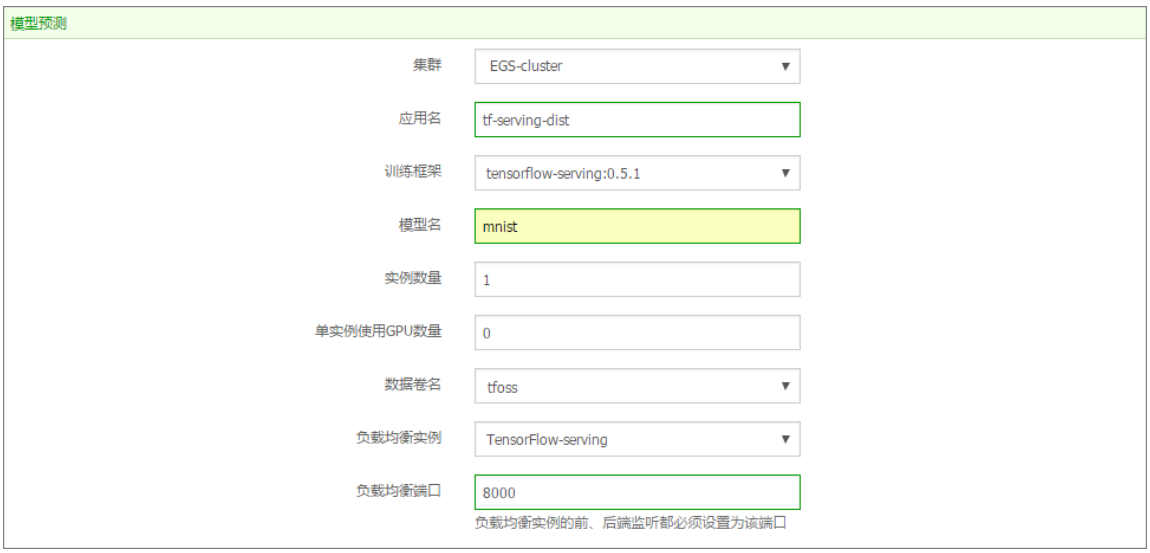
3. 启动模型预测。

a. 登录 容器服务管理控制台。

b. 单击左侧导航栏中的 镜像与方案 > 解决方案。

c. 在 模型预测 框中单击 创建。

d. 设置模型预测任务的基本信息。



- 集群：选择将要运行模型预测的集群。
- 应用名：为运行单机模型预测的应用名称。名称可以包含 1~64 个字符，包括数字、英文字母和连字符（-）。
- 训练框架：选择用来进行模型训练的框架，包括 TensorFlow Serving 和自定义镜像。这里选择 TensorFlow Serving。
- 模型名称：这里的模型名称要与步骤 1 中创建的模型文件夹一致。
- 实例数量：TensorFlow Serving 实例的数量，这里不能超过集群中节点数。

- **GPU数量**：指定所使用的 GPU 数量，如果为 0 表示不使用 GPU（将使用 CPU 预测）。
- **数据卷名**：指定为用于存储预测模型的对象存储服务（OSS）Bucket 在该集群中创建的数据卷的名称。数据卷的创建方法，请参考 创建数据卷。
- **负载均衡实例**：请选择步骤 2 负载均衡实例名称。
- **负载均衡端口**：请将步骤 2 设置的端口号填入。

4. 设置完毕后，单击 **确定**。
5. 在应用列表页面中，单击上述创建的应用。
6. 单击 **路由列表**，可以看到负载均衡提供的终端地址。这样您就可以利用 grpc 的客户端访问负载均衡地址:负载均衡端口，在本例子中为 101.37.193.36:8000。

1.6 在GPU 集群上实现Caffe模型训练

1.6.1 环境准备

目前，阿里云容器服务提供的深度学习解决方案内置了 Tensorflow、Keras、MXnet 框架的镜像，并支持基于它们的模型开发、训练和预测。同时，对于模型训练和预测，也可以通过指定自定义容器镜像，来支持用户使用其他框架。

本文将描述如何通过自定义镜像的方式，实现使用 Caffe 框架在 GPU 容器集群上进行模型训练。

使用容器服务的深度学习解决方案，主要的工作包括：

1. 准备计算集群。
   - 购买ECS计算资源，可以包括 CPU 和 GPU。
   - 创建容器集群管理上述 ECS 节点。
2. 准备数据存储，用于保存和共享训练数据集、训练日志和结果模型。
   - 创建阿里云存储服务实例。
   - 为上述数据存储创建数据卷，用于将共享存储挂载入容器内部，方便训练、预测代码本地读写。

3. 在解决方案页面配置任务参数，启动模型训练任务。

前提须知

1. 准备一个 GPU 类型的容器集群。详情参见 创建容器集群。
2. 不同 ECS 服务地域提供的 GPU 实例类型可能不同。您需要提前确认。
3. 请在与上述 ECS 节点的相同阿里云服务区域，创建 OSS 或 NAS 存储实例。否则，运行在ECS上的容器将无法访问它们。
4. 首次使用需要进行角色授权，开通容器服务。请参见 角色授权。

操作步骤

创建容器服务集群

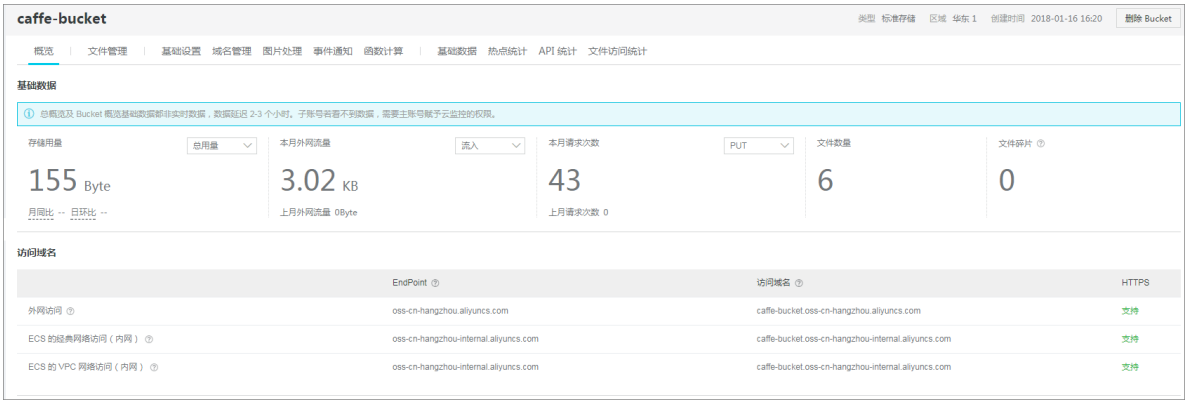
1. 登录 容器服务控制台。
2. 创建 GPU 型的容器集群，本示例创建 GN5 型的容器集群。详情请参见 创建容器集群。
3. 在容器服务控制台查看容器集群的详情，本例中创建的是位于华东 1 的容器集群 swarmcluster。

创建共享数据存储

容器服务可以通过数据卷挂载的方式支持阿里云 OSS 对象存储和 NAS 文件存储。本示例选择 OSS 数据存储类型。

1. 登录 对象存储 OSS 管理控制台。
2. 在左侧导航栏，单击 新建 **Bucket**。

3. 进行 Bucket 参数配置，注意选择和容器集群一样的地域。 本例中在华东 1 区创建 OSS bucket caffe-bucket，可以查看其内外网访问地址。



您也可以创建 NAS 文件存储实例，NAS 文件实例创建过程需要两个步骤，详见 创建文件系统 和 添加挂载点。

创建数据卷

创建好数据存储实例后，需要在容器集群中创建对应的数据卷。本例中使用 OSS 作为训练数据和日志存储，可以创建 OSS 数据卷，详情参见 创建 *OSSFS* 数据卷。

1. 登录 容器服务管理控制台。
2. 单击左侧导航栏中的数据卷，再单击右上角 **创建**，开始创建数据卷。
3. 本例中创建 OSS 数据卷 caffe-volume，用于连接前一步创建的 OSS bucket caffe-bucket。

创建NAS数据卷的过程与OSS类似。参见 *#unique_30*。

1.6.2 构建和推送自定义 caffe 镜像

目前，深度学习解决方案还未内置对 caffe 框架的支持，但是训练框架支持自定义镜像，提供了灵活性。您可以通过指定自定义镜像的方式，使用您自己的 caffe 框架来进行模型训练。

您可以使用指定的 caffe 框架制作成 Docker 镜像，推送到容器镜像仓库中，如阿里云容器镜像仓库。以后在该集群部署的训练任务就可以使用这个 caffe 镜像。在开通容器服务的同时，也会开通阿里云容器镜像仓库服务。

前提条件

1. 需要准备一个容器镜像镜像仓库，来推送生成的镜像。可以使用与容器集群相同地域的阿里云容器镜像仓库，或者私有的容器镜像仓库。本例中使用阿里云容器镜像仓库服务。
2. 需要一个可以构建 Docker 镜像的环境，且这个环境需要能 push 镜像到镜像仓库。本例中以前面创建的 swarmcluster 容器集群的一个 ECS 节点作为构建 Docker 镜像的环境。

构建容器镜像仓库

1. 登录 容器镜像服务管理控制台。
2. 单击右上角 **创建镜像仓库**。进行相关参数配置，详情请参见 *#unique_32* 中关于创建镜像仓库的内容。
3. 本示例里，在华东1区创建镜像仓库 **registry.cn-hangzhou.aliyuncs.com/dl-framework/acs-caffe**。

在节点上构建 dockerfile 和脚本文件

选择容器集群中的一台 ECS 实例，登录到该节点上。在该节点上创建一个目录，用于存放将要创建的 dockerfile 文件和脚本文件。

创建dockerfile文件

您可以在集群中的一个 ECS 节点上创建 custom_train_caffe.dockerfile 文件。

登录到 ECS 节点上，依次执行以下命令，开始创建 dockerfile。

```
mkdir caffe-file
cd  caffe-file
touch custom_train_caffe.dockerfile  ## 创建 dockerfile 空文件
vim custom_train_caffe.dockerfile    ##编辑 dockerfile 文件
```

然后输入如下的示例内容。

```
FROM bvlc/caffe:gpu
RUN  apt-get update && apt-get install -y git vim
RUN  mkdir /starter
COPY ./custom_train_helper.sh /starter
WORKDIR /starter
```

```
RUN  chmod +x custom_train_helper.sh
ENTRYPOINT ["./custom_train_helper.sh"]
```

该镜像基于 caffe 官方基础镜像 bvlc/caffe:gpu， 并使用一个自定义的脚本 custom_train_helper.sh 作为用镜像启动容器时的入口进程。

保存后，成功创建 dockerfile 文件。

创建脚本文件

在相同目录下创建 custom_train_helper.sh 文件供 dockerfile 文件里构建镜像时使用，创建脚本文件与上面的方式类似，脚本的内容如下：

```
#!/bin/bash
default_output_path='/output'
default_remote_volume_path=$DEFAULT_REMOTE_VOLUME_PATH
default_input_path='/input'
# default input dir
if [ -d "$default_remote_volume_path" ] && [ ! -z $default_re
mote_volume_path ]; then
  ln -s $default_remote_volume_path $default_input_path
fi
# default output dir
if [ ! -d "$default_output_path" ]; then
  mkdir -p $default_output_path
fi
# exec user's command
eval "$@"
echo "Done training."
```

脚本逻辑主要是在执行具体训练命令的前后期，设置工作目录，和训练日志、结果的备份工作。

创建完毕后，执行下面的命令查看是否创建成功。

```
[root@xxx]# ls -l
-rw-r--r-- 1 root root 396 1月  15 11:22 custom_train_caffe.dockerfile
-rw-r--r-- 1 root root 952 1月  15 11:16 custom_train_helper.sh
```

构建镜像和推送镜像到镜像仓库中

构建镜像

在同级目录下构建自定义镜像。

```
docker build -t registry-vpc.cn-hangzhou.aliyuncs.com/dl-framework/acs
-caffe:gpu -f custom_train_caffe.dockerfile .
```

执行命令 **docker images** 查看是否成功构建镜像。

```
[root@xxx]# docker images
REPOSITORY
  TAG                 IMAGE ID            CREATED             SIZE
```

```
registry-vpc.cn-hangzhou.aliyuncs.com/dl-framework/acs-caffe
  gpu                 7a2b58e3c5fb        39 seconds ago      3.49GB
```

将镜像推送到镜像仓库中

接下来将构建好的镜像 registry.cn-shanghai.aliyuncs.com/dl-framework/acs-caffe:gpu 推送到之前在华东1区创建的镜像仓库中去。可以参考镜像基本操作。

首先登录到镜像仓库，如下：

```
[root@xxx]# docker login --username=user@aliyun.com registry-vpc.cn-
hangzhou.aliyuncs.com  ##注意替换为用户登录账号
Password:       ##注意密码是镜像仓库的独立登录密码
Login Succeeded
```

然后将镜像 push 到镜像仓库。

```
docker push registry-vpc.cn-hangzhou.aliyuncs.com/dl-framework/acs-
caffe:gpu
```

您可以在 容器镜像服务控制台 查看该镜像仓库。可以查看到刚刚推送的 caffe 镜像的公网、内网地址。

您可以单击 镜像版本，查看镜像的基本信息。

1.7 利用 TFRecord 和 HDFS 准备 TensorFlow 训练数据

数据准备和预处理在一个深度学习训练过程中扮演着非常重要的角色，它影响着模型训练的速度和质量。

而 TensorFlow 对于 HDFS 的支持，将大数据与深度学习相集成，完善了从数据准备到模型训练的完整链条。在阿里云容器服务深度学习解决方案中，为 TensoFlow 提供了 OSS、NAS 和 HDFS 三种分布式存储后端的支持。

本文介绍如何将数据转化为 TFRecord 格式，并且将生成的 TFRecord 文件保存到 HDFS 中，本示例使用阿里云 Elastic MapReduce（E-MapReduce）的 HDFS 服务。

为什么要使用 TFRecord

TFRecord 是 TensorFlow 内定的统一标准数据格式，可以支持多线程数据读取，并且可以通过 batch size 和 epoch 参数来控制训练时单次 batch 的大小和样本文件迭次数，同时能更好的利用内存和方便数据的复制和移动，所以是利用 TensorFlow 进行大规模深度学习训练的首选。

步骤 1 创建 E-MapReduce 集群

E-MapReduce 是运行在阿里云平台上的一种大数据处理的系统解决方案。更多详细信息参见 *E-MapReduce 概述*。

登录 *E-MapReduce 管理控制台* 创建一个 E-MapReduce 集群。有关创建过程，参见 *创建 E-MapReduce 集群*。

本示例创建了一个位于华南 1 地域的集群，且 网络类型 为 **VPC**。

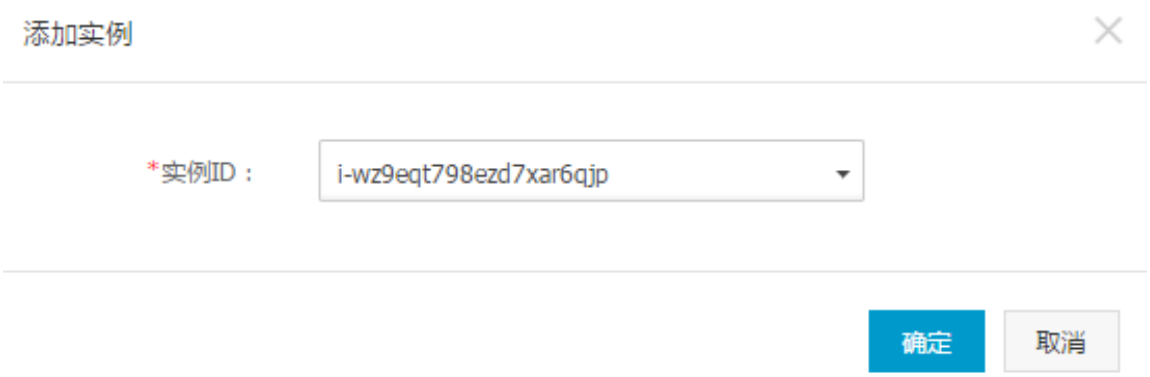
步骤 2 创建容器集群，并且打通两个集群间的网络

1. 登录 容器服务管理控制台，在同一个 VPC 下创建 GPU 容器集群。

2. 登录 *ECS* 管理控制台 将容器服务集群的节点加入到 E-MapReduce 集群对应的安全组。

   a. 选择安全组所在的地域（本示例为华南 1），选择安全组并单击右侧的 管理实例。

   b. 单击 添加实例，选择容器集群中的节点并单击 确定。



步骤 3 生成 TFRecord 数据

本示例利用模型训练服务提供运行环境执行 `convert_to_records.py`，生成 TFRecord 数据，保存到 HDFS 中。

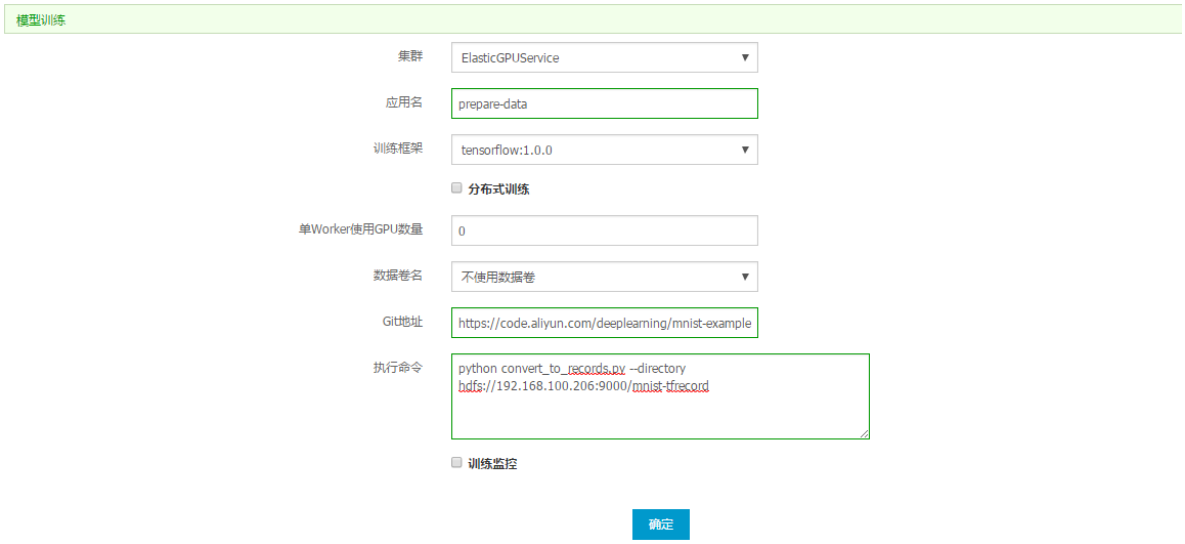
1. 登录 容器服务管理控制台。
2. 单击左侧导航栏中的 镜像与方案 > 解决方案。
3. 在 模型训练 中单击 创建。

4. 填写模型训练的配置信息并单击 确定。

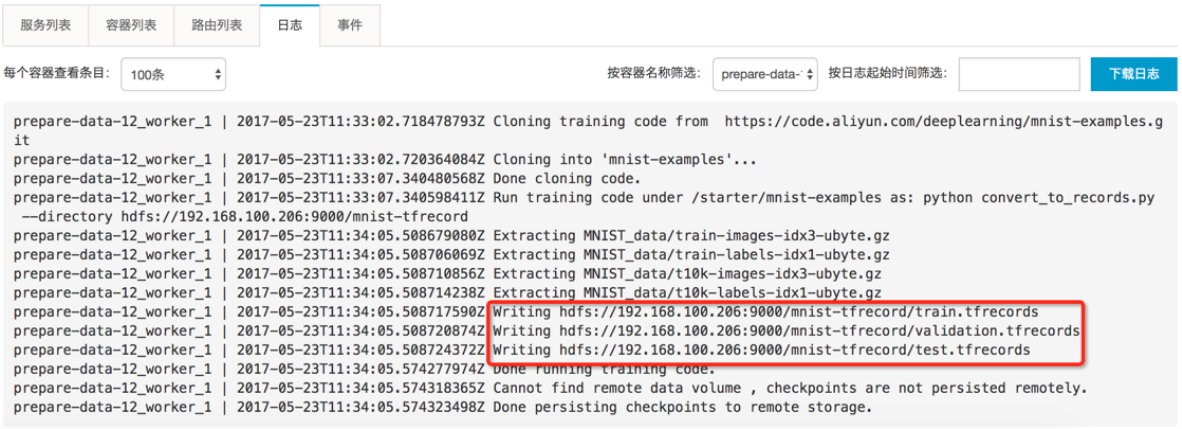
   本示例的具体配置如下所示：

   - 训练框架：**tensorflow:1.0.0**。
   - 单**Worker**使用**GPU**数量： 0。
   - 数据卷名：不使用数据卷。
   - **Git**地址：https://code.aliyun.com/deeplearning/mnist-examples.git。
   - 执行命令：

     ```
     python neural_style.py --iterations 50000 --content
                    /neural-style/examples/1-content.jpg --styles /
     neural-style/examples/1-style.jpg
                    --output /neural-style/output.jpg
     ```



   运行成功后，可以查看执行的日志，显示 TFRecord 文件已经保存到了 HDFS。



登录到 E-MapReduce 机器上查看产生的 TFRecord 文件。

```
# hdfs dfs -ls /mnist-tfrecord
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/apps/hadoop-2.7.2/share
/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/
StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/apps/tez-0.8.4/lib/slf4j-
log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an
explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 3 items
-rw-r--r--   3 root hadoop    8910000 2017-05-23 19:34 /mnist-
tfrecord/test.tfrecords
-rw-r--r--   3 root hadoop   49005000 2017-05-23 19:33 /mnist-
tfrecord/train.tfrecords
-rw-r--r--   3 root hadoop    4455000 2017-05-23 19:33 /mnist-
tfrecord/validation.tfrecords
```