实验地址：<https://developer.aliyun.com/adc/scenario/373b1f412667447eaa841270a1fd429a?spm=a2c6h.13858378.0.0.19db39579PJP7l>

1. 创建资源

开始实验之前，您需要先创建实验相关资源。

1. 在实验室页面，单击**创建资源**。
2. （可选）在实验室页面左侧导航栏中，单击**云产品资源**列表，可查看本次实验资源相关信息（例如IP地址、子用户信息等）。

**说明：**资源创建过程需要3～5分钟（视资源不同开通时间有所差异，ACK等资源开通时间较长）。完成实验资源的创建后，您可以在**云产品资源**列表查看已创建的资源信息，例如：子用户名称、子用户密码、AK ID、AK Secret、资源中的项目名称等。

实验环境一旦开始创建则进入计时阶段，建议学员先基本了解实验具体的步骤、目的，真正开始做实验时再进行创建。

图形用户界面, 应用程序

描述已自动生成

资源创建成功，可在左侧的资源卡片中查看相关资源信息以及RAM子账号信息

图形用户界面, 文本, 应用程序

描述已自动生成

上一页下一页

2. 通过容器生成镜像

在前面的实验中中，我们学习了容器管理的基本命令。在使用容器的时候我们不但会使用已有的容器镜像，也经常需要制作新的容器镜像。

在Docker中，容器镜像的制作有两种方法，分别是commit方法和build方法。实际上不论是哪种容器镜像的制作方法，基本上我们都会选择基于一个已有的容器镜像为基础，在此之上进行镜像制作。在本实验中我们先来学习直接通过容器生成新的镜像的commit方法。

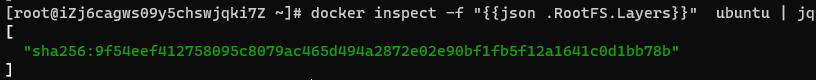
1. 实验环境准备

在本实验中我们会基于官方的ubuntu镜像来制作自定义镜像，在commit方法中，为了制作镜像，我们需要先利用基础镜像创建一个容器。接下来我们使用docker run创建容器ubuntu-commit。

除此之外，我们再使用docker inspect 命令查看基础容器镜像的层信息。

docker run -itd --name ubuntu-commit ubuntu

docker inspect -f "{{json .RootFS.Layers}}" ubuntu | jq



1. 查看容器修改内容

接下来我们对容器的内容进行修改，我们在容器中执行apt-get update命令修改容器中的文件。在文件修改之后，我们可以通过docker diff命令查看文件的修改详情，该命令格式为docker diff 容器名。

docker exec ubuntu-commit apt-get update

docker diff ubuntu-commit

图形用户界面, 文本

描述已自动生成

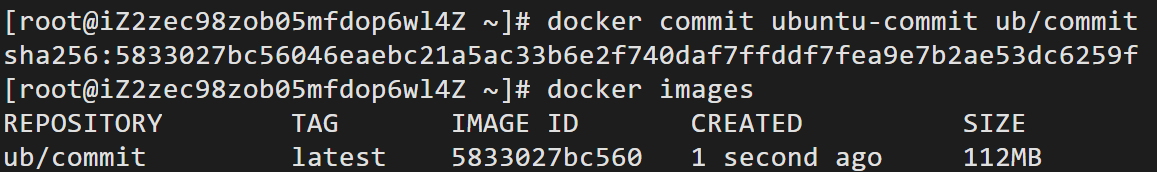
1. 生成新的镜像

在前面的实验中，我们讲到，容器镜像的是以只读分层的方式进行存储的。当我们通过镜像创建容器之后，每一个容器都会生成一个可以编写的存储层，所有用户对容器的文件修改，都会记录在容器存储层中。当我们把容器镜像的只读层和容器存储层叠加起来之后，就得到了容器的完整存储内容。而docker diff命令查看的，正是容器存储层所保存的内容。

在了解容器存储层的相关知识之后，我们就可以执行docker commit命令生成新的镜像，其命令的格式为docker commit 容器名 新镜像名。镜像生成之后我们使用docker images进行验证

docker commit ubuntu-commit ub/commit

docker images



1. 查看新镜像的层

接下来我们使用dcoker inspect来查看新镜像的层信息。会发现ub/commit在ubuntu的层之上构建了一个新的层。事实上这一层的内容就是将之前的ubuntu-commit容器存储层转化成镜像只读层。

docker inspect -f "{{json .RootFS.Layers}}" ub/commit | jq

文本

描述已自动生成

上一页下一页

3. 部署私人镜像仓库

在Docker 中推送镜像的命令是docker push，但是默认的docker push命令会将镜像推送到docker公共仓库。为了能将镜像推送到私人仓库，我们首先要使用docker tag命令为镜像设置需要推送的仓库信息。其命令的参数为docker tag [本地镜像名] [私有仓库的URL][私有仓库的镜像名称]。在标记镜像之后，我们使用docker images命令查看会发现，为本地镜像设置tag之后，会在本地镜像列表中生成一个新的镜像。

1. 部署私人镜像仓库

首先我们来学习如何启动docker私人镜像仓库。docker的默认私人镜像仓库registry应用也提供了镜像的部署方式。也就是说我们可以想启动其他容器一样的方式，本机用docker来启动默认的私有镜像仓库。当默认的私有镜像仓库启动之后，会使用http接口方式对外提供服务。

此处我们使用--network=host参数，指定容器在启动时使用Host网络模型。

docker run -d --network=host --name registry-1 registry



1. 验证网络服务

当使用Host网络模型创建容器后，容器会和宿主机共享网络设置，这也就意味着，容器中启动网络服务，同样可以在宿主机中使用127.0.0.1的网络地址访问。关于Host网络模型的详细用法，在后面的实验中还会有讲解。

当仓库容器启动后，会在5000端口启动服务。我们可以在宿主机中使用netstat命令验证。

netstat -tunple | grep 5000



1. 测试服务

接下来我们验证私人仓库的http服务接口是否可用，我们可以通过curl命令验证仓库的运行，常用的验证地址为[私人仓库IP]:5000/v2/\_catalog。通过验证我们发现私人镜像仓库服务已经启动

curl 127.0.0.1:5000/v2/\_catalog



上一页下一页

4. 向私有仓库中推送镜像

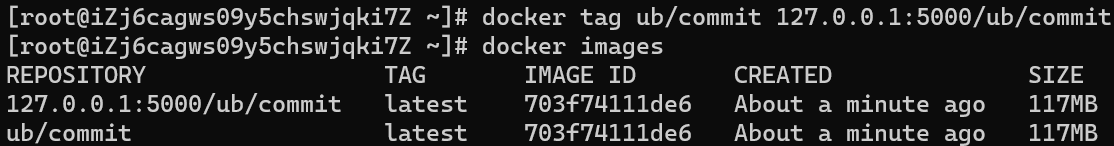
在前面的小节中，我们学习了如何启动本地的私人镜像仓库，在本小节中我们将学习如何将本地docker中的镜像推送到私有仓库中。

1. 为镜像设置仓库信息

默认的docker push命令会将容器推送到docker公共仓库，为了能将镜像推送到私人仓库，我们要使用docker tag命令为镜像设置需要推送的仓库信息。其命令的参数为docker tag [本地镜像名] [私有仓库的URL][私有仓库的镜像名称]。标记镜像之后，我们使用docker images命令查看会发现，为本地镜像设置tag之后，会在本地镜像列表中生成一个新的镜像。

docker tag ub/commit 127.0.0.1:5000/ub/commit

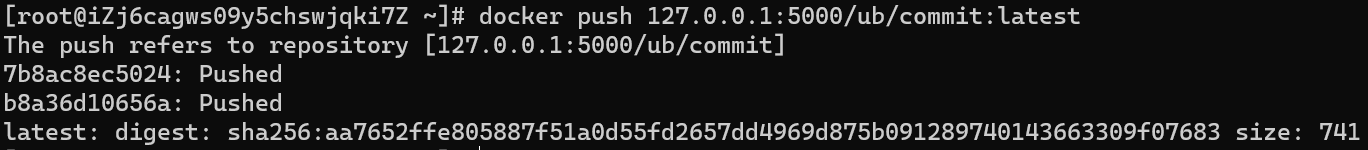
docker images



1. 向仓库推送镜像

通过docker tag命令生成新的镜像之后，我们就可以使用docker push将已经标记过的镜像，直接推送到私人仓库中。命令的参数为docker push tag后的镜像名:镜像版本。

docker push 127.0.0.1:5000/ub/commit:latest

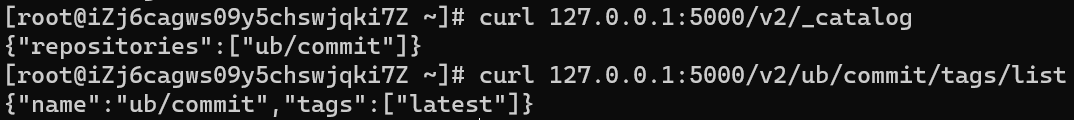


1. 通过私有仓库API验证推送

当镜像上传到私人仓库之后。我们可以通过访问curl [私人仓库IP]:5000/v2/\_catalog列出仓库中的镜像信息。同时还可以通过curl [私人仓库IP]:5000/v2/[镜像名称]/list来查看镜像的版本信息。

curl 127.0.0.1:5000/v2/\_catalog

curl 127.0.0.1:5000/v2/ub/commit/tags/list



上一页下一页

5. 从私有仓库中拉取镜像

1. 删除本地镜像

为了验证从私有仓库下载镜像，我们先通过docker rmi命令删除本地的容器镜像。

docker rmi ub/commit

docker rmi 127.0.0.1:5000/ub/commit

文本

中度可信度描述已自动生成

1. 从私有仓库中拉取镜像

当通过docker push将镜像推送到私人仓库之后。其他人就可以通过docker pull命令将其拉取到本地。拉取私有仓库镜像的命令格式为docker pull [私有仓库的URL][私有仓库的镜像名称]。

docker pull 127.0.0.1:5000/ub/commit

docker images

文本

描述已自动生成

1. 查看本地镜像信息

对于已经拉取到本地的容器镜像，我们可以通过docker inspcet来查看镜像的仓库信息和repo tag信息。在使用docker inspcet命令时，我们可以使用-f参数进行信息过滤，具体的过滤语法我们会在后面的实验中详细讲解。

docker inspect -f \

"{\"RepoTags\":{{json .RepoTags}}, \"RepoDigests\":{{json .RepoDigests}}}" \

127.0.0.1:5000/ub/commit | jq



上一页下一页