运行一些简单的 Docker 容器

使用容器的方法有很多种。其中包括：

运行单个任务：这可能是 shell 脚本或自定义应用。

交互：这将使您连接到容器中，类似于 SSH 连接到远程服务器的方式。

在后台：用于网站和数据库等长时间运行的服务。

在此步骤中，我们将启动一个新的容器，并告诉它运行hostname命令。容器将启动，并在容器中执行命令，然后退出。

docker container run alpine hostname

下面的输出显示，无法在本地找到alpine:latest image。发生这种情况时，Docker 会自动将其从Docker Hub.拉取image

提取图像后，将显示容器的主机名（以下示例中为 888e888a3b36b）。

Unable to find image 'alpine:latest' locally

latest: Pulling from library/alpine

88286f41530e: Pull complete

Digest: sha256:f006ecbb824d87947d0b51ab8488634bf69fe4094959d935c0c103f4820a417d

Status: Downloaded newer image for alpine:latest

888e89a3b36b

只要容器在容器内启动的进程仍在运行，Docker 就保持容器运行。在这种情况下，主机名进程在写入输出后立即退出。这意味着容器停止。但是，Docker 默认情况下不会删除资源，因此容器仍以"退出"状态存在。

列出所有容器

docker container ls --all

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

888e89a3b36b alpine "hostname" 50 seconds ago Exited (0) 49 seconds ago awesome\_elion

执行一项任务然后退出的容器可能非常有用。您可以构建一个 Docker 映像，该映像执行脚本来配置某些内容。任何人都可以通过运行容器来执行该任务 - 他们不需要实际的脚本或配置信息。

运行交互式 Ubuntu 容器

您可以在 Docker 主机上运行一个与上例不同的基于另一个 Linux 版本的容器。

docker container run --interactive --tty --rm ubuntu bash

--interactive：需要一个交互会话

--tty：分配一个伪 tty

--rm：告诉 Docker 执行完成后删除容器。

前两个参数允许您与 Docker 容器进行交互。

我们还告诉容器运行 bash 作为其主进程（PID 1）。

当容器启动时，您将进入bash shell使用默认提示符root@<container id>:/#。Docker 已附加到容器中的 shell，在本地会话和容器中的 shell 会话之间中传递输入和输出。

在容器中运行以下命令。

ls / 将在容器中列出根控制器的内容，ps aux 将显示容器中的正在运行的进程，cat /etc/ issue将显示容器正在运行的 Linux 发行版，在本例中为 Ubuntu 18.04.1 LTS。

键入exit以离开 shell 会话。这将终止 bash 进程，导致容器退出。

让我们检查一下主机 VM 的版本。你将看到

Welcome to Alpine Linux 3.8

Kernel \r on an \m (\l)

请注意，我们的主机 VM 正在运行Alpine Linux，但我们能够运行 Ubuntu 容器。如前所述，容器内的 Linux 版本不需要与 Docker 主机上运行的 Linux 的发行版相匹配。

但是，Linux 容器要求 Docker 主机运行 Linux 内核。例如，Linux 容器不能直接在 Windows Docker 主机上运行。Windows 容器也是如此 - 它们需要在具有 Windows 内核的 Docker 主机上运行。

当您将自己的映像组合在一起时，交互式容器非常有用。您可以运行容器并验证部署应用所需的所有步骤，并在 Dockerfile 中捕获这些步骤。

后台运行 MySQL 容器

后台容器是大多数应用程序运行方式。下面是一个使用 MySQL 的简单示例。

使用以下命令运行新的 MySQL 容器。

docker container run \

--detach \

--name mydb \

-e MYSQL\_ROOT\_PASSWORD=my-secret-pw \

mysql:latest

--detach：将在后台运行容器。

--name：将容器命名为mydb。

-e：将使用环境变量指定root密码（注意：这绝不应在生产中完成）

由于 MySQL 映像在本地不可用，Docker 会自动将其从 Docker Hub拉取

Unable to find image 'mysql:latest' locallylatest: Pulling from library/mysql

aa18ad1a0d33: Pull complete

fdb8d83dece3: Pull complete

75b6ce7b50d3: Pull complete

ed1d0a3a64e4: Pull complete

8eb36a82c85b: Pull complete

41be6f1a1c40: Pull complete

0e1b414eac71: Pull complete

914c28654a91: Pull complete

587693eb988c: Pull complete

b183c3585729: Pull complete

315e21657aa4: Pull complete

Digest: sha256:0dc3dacb751ef46a6647234abdec2d47400f0dfbe77ab490b02bffdae57846ed

Status: Downloaded newer image for mysql:latest

41d6157c9f7d1529a6c922acb8167ca66f167119df0fe3d86964db6c0d7ba4e0

只要 MySQL 进程正在运行，Docker 就会保持容器在后台运行。

列出正在运行的容器。

docker container ls

请注意，容器正在运行。

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

3f4e8da0caf7 mysql:latest "docker-entrypoint..." 52 seconds ago Up 51 seconds 3306/tcp mydb

您可以使用几个内置的 Docker 命令来检查容器中发生的情况：docker container logs 和docker container top

docker container logs mydb

这将显示来自 MySQL Docker 容器的日志。

<output truncated>

2017-09-29T16:02:58.605004Z 0 [Note] Executing 'SELECT \* FROM INFORMATION\_SCHEMA.TABLES;' to get a list of tables using the deprecated partition engine. You may use the startup option '--disable-partition-engine-check' to skip this check.

2017-09-29T16:02:58.605026Z 0 [Note] Beginning of list of non-natively partitioned tables

2017-09-29T16:02:58.616575Z 0 [Note] End of list of non-natively partitioned tables

让我们看一下容器内运行的进程

docker container top mydb

您应该会看到 MySQL 守护进程 （mysqld） 正在容器中运行。

PID USER TIME COMMAND

2876 999 0:00 mysqld

尽管 MySQL 正在运行，但它在容器内是隔离的，因为没有向主机发布任何网络端口。除非显式发布端口，否则网络流量无法从主机到达容器。

使用 docker container exec列出 MySQL 版本。

docker container exec允许您在容器内运行命令。

docker exec -it mydb \

mysql --user=root --password=$MYSQL\_ROOT\_PASSWORD --version

您将看到 MySQL 版本号，以及一个警告。

mysql: [Warning] Using a password on the command line interface can be insecure.

mysql Ver 14.14 Distrib 5.7.19, for Linux (x86\_64) using EditLine wrapper

您还可以使用 docker container exec连接到已在运行的容器内的新 shell 进程。执行下面的命令将在 MySQL 容器内为您提供一个交互式shell （sh）。

docker exec -it mydb sh

请注意，您的 shell 提示已更改。这是因为您的 shell 现在连接到容器内运行的 sh 进程。

让我们通过再次运行相同的命令来检查版本号，只是这次从容器中的新 shell 会话中运行。

mysql --user=root --password=$MYSQL\_ROOT\_PASSWORD --version

键入exit以退出交互式 shell 会话。

任务 2：使用 Docker 打包并运行自定义应用

在此步骤中，您将学习如何使用 Docker 文件将您自己的应用打包为 Docker 映像。

Dockerfile 语法非常简单。在此任务中，我们将从 Dockerfile 创建一个简单的 NGINX 网站。

构建简单的网站映像

让我们来看看我们将使用的Dockerfile，它构建了一个简单的网站，允许您发送tweet。

使用git下载网站项目

git clone https://github.com/dockersamples/linux\_tweet\_app

确保您位于 linux\_tweet\_app 目录中。

cd /linux\_tweet\_app

显示 Docker 文件的内容。

cat Dockerfile

FROM nginx:latest

COPY index.html /usr/share/nginx/html

COPY linux.png /usr/share/nginx/html

EXPOSE 80 443

CMD ["nginx", "-g", "daemon off;"]

让我们看看 Dockerfile 中的每行都有什么用。

FROM 指定用作要创建的新映像的起点的基础映像。对于此示例，我们从 nginx：latest 开始。

COPY 将文件从 Docker 主机复制到映像中的指定目录中。在此示例中，COPY 用于将两个文件复制到映像中：index.html。和将在我们的网页上使用的图片。COPY指令的第一个参数是相对路径下的文件，它相对于执行创建映像命令时传入的上下文参数。

EXPOSE 记录应用程序使用的端口。

CMD 指定从映像启动容器时要运行的命令。请注意，我们可以指定命令以及运行时参数。

CMD有两种模式。

将<https://hub.docker.com/> 上面注册的DockerID设置为环境变量，方便后续操作

使用 docker image build命令使用 Dockerfile 中的指令创建新的 Docker 映像。

docker image build --tag $DOCKERID/linux\_tweet\_app:1.0 .

--tag：允许我们为图像指定自定义名称。在这种情况下，它由我们的 DockerID、应用程序名称和版本组成。将 Docker ID 附加到名称将使我们能够在后面的步骤中将其存储在Docker Hub上

请确保在命令的末尾包含句点 （.）。意思为将当前目录当作上下文路径，配合COPY指令使用

Docker 守护进程程序执行Dockerfile中的每行指令的输出显示如下

Sending build context to Docker daemon 32.77kB

Step 1/5 : FROM nginx:latest

latest: Pulling from library/nginx

afeb2bfd31c0: Pull complete

7ff5d10493db: Pull complete

d2562f1ae1d0: Pull complete

Digest: sha256:af32e714a9cc3157157374e68c818b05ebe9e0737aac06b55a09da374209a8f9

Status: Downloaded newer image for nginx:latest

---> da5939581ac8

Step 2/5 : COPY index.html /usr/share/nginx/html

---> eba2eec2bea9

Step 3/5 : COPY linux.png /usr/share/nginx/html

---> 4d080f499b53

Step 4/5 : EXPOSE 80 443

---> Running in 47232cb5699f

---> 74c968a9165f

Removing intermediate container 47232cb5699f

Step 5/5 : CMD nginx -g daemon off;

---> Running in 4623761274ac

---> 12045a0df899

Removing intermediate container 4623761274ac

Successfully built 12045a0df899

Successfully tagged <your docker ID>/linux\_tweet\_app:latest

使用 docker container run命令从创建的映像启动新容器。

由于此容器将运行 NGINX Web 服务器，我们将使用 --publish 标志将容器内的端口 80 发布到主机上的端口 80。这将允许进入端口 80 上的 Docker 主机的流量被定向到容器中的端口 80。--publish标志的格式是主机端口：容器端口。

docker container run \

--detach \

--publish 80:80 \

--name linux\_tweet\_app \

$DOCKERID/linux\_tweet\_app:1.0

在后面的步骤中，您将看到如何映射来自两个不同端口的流量 - 当两个容器使用同一个端口进行通信时，这是必需的，因为您只能在主机上公开端口一次。

关闭并删除容器

docker container rm --force linux\_tweet\_app

任务 3：修改正在运行的网站

当您积极处理应用程序时，每次更改源代码时，必须停止容器、重建映像并运行新版本是不方便的。

简化此过程的一种方法是将本地计算机上的源代码目录挂载到正在运行的容器中。这将允许对主机上的文件所做的任何更改立即反映在容器中。

我们使用称为bind mount来执行此操作。

使用bind mount时，主机上的文件或目录将装载到运行在同一主机上的容器中。

使用bind mount启动我们的 Web 应用

让我们启动 Web 应用，并将当前目录挂载到容器中。

在此示例中，我们将使用 --mount 标志将主机上的当前目录装载到容器内的 /usr/share/nginx/html 中。

请确保在 Docker 主机上的 linux\_tweet\_app 目录中运行此命令。

docker container run \

--detach \

--publish 80:80 \

--name linux\_tweet\_app \

--mount type=bind,source="$(pwd)",target=/usr/share/nginx/html \

$DOCKERID/linux\_tweet\_app:1.0

网站应该正在运行。

修改正在运行的网站

bind mount意味着对本地文件系统所做的任何更改都立即反映在正在运行的容器中。

将新 index.html 复制到容器中。

前面提取的 Git 存储库包含几个不同版本的 index.html 文件。可以从 [/linux\_tweet\_app]目录中手动运行 ls 命令以查看它们的列表。在此步骤中，我们将用index -new.html 替换 index.html。

cp index-new.html index.html

转到正在运行的网站并刷新页面。请注意，网站已更改。

尽管我们修改了 index.html 本地文件系统，并将其反映在正在运行的容器中，但我们实际上并未更改容器启动的 Docker 映像。

要显示这一点，请将当前容器停止并重新运行映像，而不进行bind mount

停止并删除当前正在运行的容器。

docker rm --force linux\_tweet\_app

在没有bind mount的情况下重新运行当前版本。

docker container run \

--detach \

--publish 80:80 \

--name linux\_tweet\_app \

$DOCKERID/linux\_tweet\_app:1.0

请注意，网站已回到原始版本。

停止并删除当前容器

docker rm --force linux\_tweet\_app

更新image

要将对 index.html 文件所做的更改保留到映像中，您需要构建映像的新版本。

生成新映像并将其标记为 2.0

请记住，您之前在 Docker 主机本地文件系统上修改了 index.html 文件。这意味着运行另一个 docker 映像生成命令将生成具有更新的 index.html 的新映像

docker image build --tag $DOCKERID/linux\_tweet\_app:2.0 .

注意它建立的速度有多快！这是因为 Docker 只修改了更改的image部分，而不是重新生成整个映像。

测试新版本

从映像的新版本运行新容器

docker container run \

--detach \

--publish 80:80 \

--name linux\_tweet\_app \

$DOCKERID/linux\_tweet\_app:2.0

检查网站的新版本

我们可以并行运行两个版本。我们唯一需要注意的是，不能在同一主机上使用端口 80 的两个容器。

由于我们已经为从映像的 2.0 版本运行的容器使用端口 80，我们将启动一个新容器并将其发布到端口 8080 上。此外，我们需要为我们的容器指定一个唯一的名称（old\_linux\_tweet\_app）

运行另一个新容器，这次是使用旧版本的映像。

请注意，此命令将新容器映射到主机上的端口 8080。这是因为两个容器无法映射到单个 Docker 主机上的同一端口。

docker container run \

--detach \

--publish 8080:80 \

--name old\_linux\_tweet\_app \

$DOCKERID/linux\_tweet\_app:1.0

查看网站的旧版本。

将image推送到 Docker Hub

列出 Docker 主机上的映像

docker image ls -f reference="$DOCKERID/\*"

REPOSITORY TAG IMAGE ID CREATED SIZE

<docker id>/linux\_tweet\_app 2.0 01612e05312b 3 minutes ago 108MB

<docker id>/linux\_tweet\_app 1.0 bb32b5783cd3 7 minutes ago 108MB

这些映像仅存储在 Docker 主机本地存储库中。在此步骤中，我们将映像推送到公共存储库，以便您可以使用 Docker 从任何 Linux 计算机运行它们。

分发功能内置于 Docker 平台中。您可以在本地生成映像，并将它们推送到公共或私有注册表，使其可供其他用户使用。具有访问权限的任何人都可以提取该映像并从中运行容器。容器中应用的行为对于每个人都是一样的，因为映像包含完全配置的应用 - 运行它的唯一要求是 Linux 和 Docker。

[Docker Hub](https://hub.docker.com) 是 Docker 映像的默认公共存储库

在推送映像之前，您需要登录到 Docker Hub

docker login

在出现提示时，您需要提供 Docker ID和 凭据。

使用 docker image push推送 Web 应用的 1.0 版本。

docker image push $DOCKERID/linux\_tweet\_app:1.0

当映像推送到 Docker Hub.时，您将看到进度。

The push refers to a repository [docker.io/<your docker id>/linux\_tweet\_app]

910e84bcef7a: Pushed

1dee161c8ba4: Pushed

110566462efa: Pushed

305e2b6ef454: Pushed

24e065a5f328: Pushed

1.0: digest: sha256:51e937ec18c7757879722f15fa1044cbfbf2f6b7eaeeb578c7c352baba9aa6dc size: 1363

现在推送版本 2.0。

docker image push $DOCKERID/linux\_tweet\_app:2.0

请注意，输出的几行表示层已存在。这是因为 Docker 将利用与以前上载的任何映像图层相同的只读图层。

The push refers to a repository [docker.io/<your docker id>/linux\_tweet\_app]

0b171f8fbe22: Pushed

70d38c767c00: Pushed

110566462efa: Layer already exists

305e2b6ef454: Layer already exists

24e065a5f328: Layer already exists

2.0: digest: sha256:7c51f77f90b81e5a598a13f129c95543172bae8f5850537225eae0c78e4f3add size: 1363

您可以浏览到https://hub.docker.com/r/<your docker id>/并查看新推送的 Docker 图像。这些是公共存储库，因此任何人都可以提取图像 - 您甚至不需要 Docker ID 来提取公共映像。Docker 中心还支持专用存储库。

<https://training.play-with-docker.com/beginner-linux/>