Docker

Docker链接说明

我找到了一个对docker解释与使用的很全面的网页，解释网页点击此处，还有一个docker命令的网页，命令网页点击此处

学习Docker，我们首先要知道虚拟机或者容器化是怎么一回事？

- 虚拟机：虚拟机（Virtual Machine，简称 VM）是一种由软件模拟的计算机系统，它能够运行与物理计算机类似的操作系统和应用程序。虚拟机通过虚拟化技术实现，将计算机的硬件资源（如 CPU、内存、硬盘、网络等）抽象成多个虚拟化的资源，为用户提供一个独立、隔离的运行环境。
- 容器化：容器化意味着封装或打包软件代码及其所有依赖项，以便它可以在任何基础架构上统一且一致地运行。

Docker就是目前一种非常火的实现容器化技术的工具，还有其他的工具感兴趣可以自己了解，目前Apollo的工程是使用了Docker

安装Docker

Docker可在 Mac、Windows 和 Linux 这三个主要平台上完美运行。 但是安装因使用的操作系统而异。我们在学习Apollo的过程中会在Linux的发行版（Ubuntu）中安装*Docker Engine*。

注意

这里主要说一下可能会存在的一个误区：我们在mac或者windows安装的*Docker Desktop*与Linux系统下安装的*Docker Engine*在学习的知识上没有什么太本质的区别。不要过多的去想为什么不一样的系统下载的东西不一样！！！

> Windows 或 Mac 上的*Docker Desktop*软件包是一系列工具的集合，例如`Docker Engine`、`Docker Compose`、`Docker Dashboard`、`Kubernetes`和其他一些好东西。Linux系统下也有*Docker Desktop*，只是Apollo的教程中只让我们下载了`Docker Engine`而已---有引擎就可以用

1. *mac上安装*
   mac上直接在官方下载页面单击*Download for Mac(stable)* 下载安装即可
2. *windows上安装*
   和mac一样，但是需要提前多安装一个WSL2。WSL2安装方法点击此处

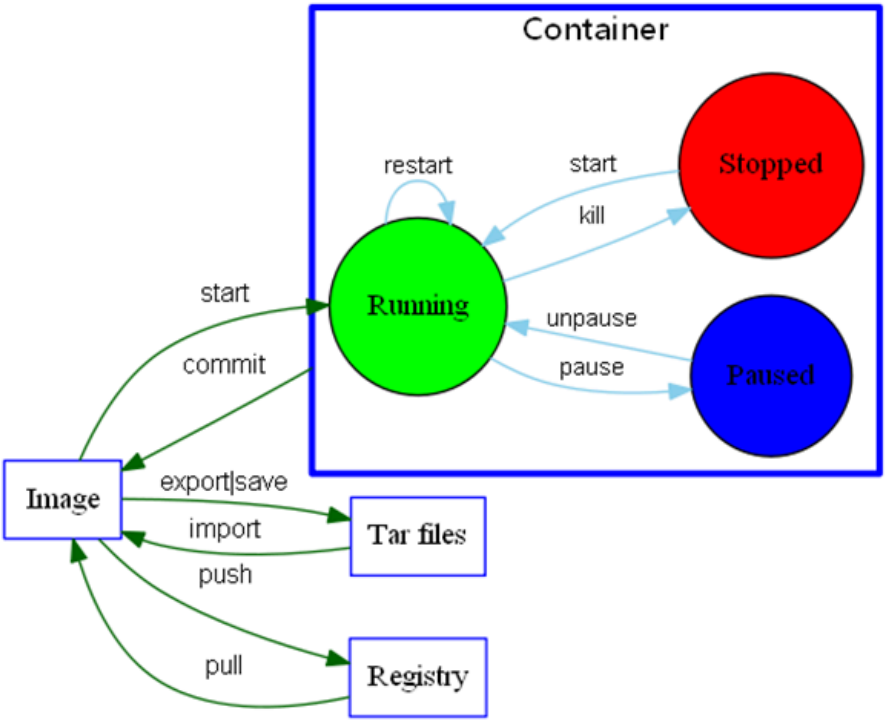
3. *Linux上安装*
   在Linux上没有捆绑包，可以手动安装所需的所有必要工具。 不同发行版的安装过程不同，Apollo的官方文档是让我们在Ubuntu下使用apt的方式去安装*Docker Engine*。他的安装方法还有别的方式，可以查看官网，具体安装方法点击此处

Docker三大基础部分

安装完成docker以后，接下来就是对docker的学习了。docker在系统学习如何部署以及使用之前一定要搞清楚以下三个部分：

1. 镜像(images):Docker 镜像是一个只读的、轻量级的文件系统模板，包含了运行某个应用程序所需的一切环境，包括代码、依赖、库、运行时、配置文件等。镜像类似于一个快照，是容器的运行基础。类似于虚拟机的镜像文件
2. 容器(container):Docker 容器是镜像的一个实例，是一个运行时环境。容器是基于镜像启动的动态、独立的应用实例，可以理解为镜像的一个运行版本。容器在运行时会加载镜像，并添加一个可写层。
3. 仓库(repository):Docker 仓库是用来存储和分发镜像的服务。它是镜像的存储库，用户可以从仓库中拉取镜像或将镜像推送到仓库。

> 我们以实际应用结合关系图来看一下流程。
>
> 1. 我们会在终端输入 docker run IMAGE命令，这个命令会先检查镜像是否存在。
> 2. 如果不存在，会先尝试从仓库中拉取镜像。
> 3. 镜像存在以后会创建一个新的容器，并运行。
>   注意！每次都会创建不一样的容器即便是同一个镜像



结合上面再简单的描述一下他们三者之间的关系。容器是由镜像创建的，我们平时上传和下载下来的都是镜像，想要运行程序，就需要使用镜像去创建容器后运行，而“运行镜像”实际上是包含了创建容器这一步。也就是说镜像就相当于一个只读的模板，容器是它的实例化。仓库则是存储镜像的，一般都是存放在官方的公有仓库docker hub上，它就类似于github。

这一我们只讲一下docker的三个基础部分，关于docker更加深入的知识，比如说我如何部署docker，怎么创建属于我自己的镜像等等因为目前还不是很需要，所以暂时还没学习到。有想了解的可以先自行学习想一下。

apollo中的Docker

apollo中关于docker大多是写好的脚本方便我们安装 Docker Engine 并且拉取镜像。

安装完成Docker Engine会让我们执行bash docker/scripts/dev_start.sh这个命令，他是让我们执行工程目录下的docker文件中的scripts文件夹下的dev_start.sh脚本，它会根据用户的输入参数和环境配置，使用 Docker 技术来拉取 Apollo 的相关镜像、配置容器环境、挂载所需资源，并最终启动一个适用于 Apollo 开发的容器。

我们拉取完成以后，关于docker他的镜像名字是`apolloauto/apollo:dev-x86-18.04-20210914_1336`，容器名字是`apollo_dev_${USER}`(这个USER就是电脑的用户名) 例如车辆工控机上就是apollo_dev_yhs。

> 1. 我们会在终端输入 docker start apollo_dev_yhs命令，这个命令会让我们启动已经存在的容器apollo_dev_yhs。
> 2. 执行bash docker/scripts/dev_into.sh命令进入容器进行开发测试

说明

这里的第一步和我们学习的第一步不太一样，简单说明一下原因。docker run IMAGE命令用来创建并启动一个新容器，每次执行它我们都会创建一个不同的容器，容器不会重复。docker start CONTAINER用来启动一个已经存在的容器而不是创建一个新的容器在启动。Apollo编写的dev_start.sh脚本会为我们创建一个固定的容器，以后我们都是利用docker start 启动同一个固定容器来开发和测试。

以下是脚本中具体创建固定容器的代码，就不再过多赘述了。

```
    ${DOCKER_RUN_CMD} -itd \

        --privileged \

        --name "${DEV_CONTAINER}" \

        -e DISPLAY="${display}" \

        -e DOCKER_USER="${user}" \

        -e USER="${user}" \

        -e DOCKER_USER_ID="${uid}" \

        -e DOCKER_GRP="${group}" \

        -e DOCKER_GRP_ID="${gid}" \

        -e DOCKER_IMG="${DEV_IMAGE}" \

        -e USE_GPU_HOST="${USE_GPU_HOST}" \

        -e NVIDIA_VISIBLE_DEVICES=all \

        -e NVIDIA_DRIVER_CAPABILITIES=compute,video,graphics,utility
\

        ${MAP_VOLUMES_CONF} \

        ${OTHER_VOLUMES_CONF} \

        ${local_volumes} \

        --net host \

        -w /apollo \

        --add-host "${DEV_INSIDE}:127.0.0.1" \

        --add-host "${local_host}:127.0.0.1" \
```

```
        --hostname "${DEV_INSIDE}" \

        --shm-size "${SHM_SIZE}" \

        --pid=host \

        -v /dev/null:/dev/raw1394 \

        "${DEV_IMAGE}" \

        /bin/bash
```