1. Docker
2. 概述

在项目的开发周期中，我们会接触到三种环境，分别是开发环境，测试环境和生产环境。开发环境就是开发人员在开发过程中配置的环境，当开发过程结束后，需要将程序打成war包给测试人员进行测试，测试人员的测试环境可能与开发环境大大不同，这时就会出现一些非业务的错误，这都是环境的不同造成的错误。而Docker就解决了这种软件跨环境迁移的问题，Docker本身是一个容器，每个容器中包含了程序和程序运行的环境，在迁移时我们直接迁移这个Docker容器即可。

1. Docker概念

Docker是一个开源的应用容器引擎，使用Go语言开发。Doker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。容器使用沙箱机制，每个容器之间是相互隔离的，互不影响。

1. 安装Docker

在Ubuntu16.04上面安装Docker，具体步骤如下所示：

1. 更新apt包索引

sudo pat-get update

1. 安装以下包使apt可以通过HTTPS使用repository

sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common

1. 添加Docker官方的GPG密匙

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

1. 设置stable存储库

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

1. 更新apt包索引

sudo apt-get update

1. 安装最新版本的Docker CE

sudo apt-get install -y docker-ce

1. 验证安装是否成功

docker -v

1. Docker架构

Docker的架构分为三部分，第一个为Docker的客户端，用来发送命令。第二部分为Docker的主机，分为本地主机和远程主机。当Docker在Ubuntu中安装成功后，就会以守护进程的形式存在，即后台进程。Docker主机中有两部分内容，一个叫做镜像，一个叫做容器。镜像是一个包含软件以及软件所需要的运行环境的文件系统，容器是根据镜像创建的，一个镜像可以创建多个容器，容器是镜像运行时的实体，可以被创建，启动，停止，删除，暂停。第三部分是Docker的仓库，Docker的仓库与Maven的仓库类似，分为中央仓库和私服，存储着镜像文件。

1. 配置docker国内镜像

这里配置的是阿里云的镜像，访问阿里云网站，<https://cr.console.aliyun.com/>，它会给每个账号设置一个加速器地址，复制该地址，然后修改Docker的配置文件 /etc/docker/daemon.json，设置成如下形式：

{ "registry-mirrors": ["https://nylxo38q.mirror.aliyuncs.com"] }

重启docker服务，镜像源就设置好了。

1. Docker命令

（一 ）Docker服务命令

1. 启动Docker服务

systemctl start docker

1. 停止docker服务

systemctl stop docker

1. 重启docker服务

systemctl restart docker

1. 查看docker服务状态

systemctl status docker

1. 设置开机启动docker服务

systemctl enable docker

（二）Docker镜像命令

1. 查看本地镜像

docker images

1. 查看所用镜像的id

docker images -q

1. 从互联网中搜索镜像

docker search 镜像名称

1. 拉取镜像

从Docker仓库中下载镜像到本地，镜像名称格式为 名称：版本号，如果不指定版本号则是最新版。

docker pull 镜像名称

1. 删除单个镜像

删除本地的镜像

docker rmi 镜像id

1. 删除所有镜像

即首先取出所有镜像的id，然后全部删除

docker rmi “docker images -q”

（三）容器相关命令

容器是镜像的运行实体，下面是与容器相关的命令。

1. 创建容器

容器是根据镜像创建的，因此在创建容器时需要指定镜像，-I 表示在没有客户端连接时容器也不关闭，-t 表示给容器分配一个终端来接受命令。 --name表示容器的名称，后面跟的是镜像的名称与版本号，最后还可以加一个进入容器的初始化指令。这种方式创建的容器在exit之后就会退出。这种容器称为交互式容器。

docker run -it --name=c1 centos:7 /bin/bash

这样，就会进入docker容器中，使用exit命令即可退出。

还有一种后台创建容器的方法，只是将-t改为了-d，表示在后台创建容器，输入exit命令也不会退出。这种容器称为守护式容器。如下所示：

docker run -id --name=c2 centos:7 /bin/bash

运行这条命令之后，它不会直接进入容器中，依然在linux用户之中，要使用docker exec 命令进入容器之中，如下所示：

docker exec -it c2 /bin/bash

这样，进入容器后再次使用exit退出容器，容器依旧能在后台运行着。

1. 查看容器

查看正在运行的容器：

docker ps

查看所有容器：

docker ps -a

1. 进入容器

docker exec 参数

1. 停止容器

docker stop 容器名称

1. 启动容器

docker start 容器名称

1. 删除容器

只能删除停止运行的容器

docker rm 容器名称

1. 查看容器信息

docker inspect 容器名称

1. Docker容器的数据卷
2. 概述

所谓的数据卷就是宿主机中的一个目录或者目录，数据卷解决了三个问题，一个是容器数据的持久化，一个是外部机器和容器间接通信，一个是容器之间的数据交换。

1. 容器数据持久化

在容器的目录和宿主机中数据卷绑定后，那么不论是哪一方进行修改，修改的内容都会立即同步到绑定的目录之中，这样就保证了Docker容器数据的持久化，就算容器被删除了，数据依旧存储在宿主机中。

1. 外部机器和容器间接通信

外部机器和容器是没办法直接通信的，然而外部机器与宿主机器是可以通信的，因此，可以借助宿主机器的数据卷与容器交互通信。

1. 容器之间数据交换

一个数据卷可以被多个容器同时挂载，一个容器也可以挂载多个数据卷。因此，容器之间就可以通过宿主机的数据卷通信交互。

1. 配置数据卷

在创建启动容器时，使用-v参数设置数据卷，就可以将宿主机的数据卷和容器内的目录映射起来了，注意，这里的目录必须是绝对路径，如果目录不存在则会自动创建，且一个容器可以使用多个-v参数挂载多个数据卷。如下所示：

docker run … -v 宿主机目录（文件）：容器内目录（文件）

当我们删除了该容器后，可以利用宿主机中的数据卷重新恢复该容器。

1. 数据卷容器

多个容器之间进行数据交换不仅可以通过多容器挂载同一个数据卷实现，还可以通过一个数据卷容器实现。所谓的数据卷容器也是一个容器，与其他容器无差别，该容器挂载了一个数据卷，然后其他的容器可以通过挂载在该容器中而挂载在宿主机的数据卷中。即该数据卷容器就是一个中介，即是该容器被删除，其他已经挂载成功的容器对于数据卷的挂载也不会受到影响。

使用如下的方式在创建数据卷容器， -v参数 中只有一个容器中的目录，而没有数据卷的目录，数据卷的目录会自动创建的:

docker run -it --name=c3 -v /volume centos:7 /bin/bash

使用如下的方式创建其他的容器挂载在数据卷容器中，--volumes-from是固定用法 ，后面跟着数据卷容器的名称：

docker run -it --name=c1 –volumes-from c3 centos:7 /bin/bash

docker run -it --name=c2 -volumes-from c3 centos:7 /bin/bash