docker 的使用

enzo liu

2016-06-30 Thu

前段时间，公司意外断电，导致内网服务器硬盘数据损坏, 之前纯手工打造的内网环境全跪了。欲哭无泪，于是就不哭了，借这个机会体验一把 `docker`, 这里记录一下使用中遇到的一些状况。

环境介绍

原来的内网环境中，包含了

jenkins 用于 `CI`

artifactory 用于内网 `maven` 仓库的托管

gitlab 内网 'git' 仓库

blackhole 亿华 出品，用于内网 dns 劫持

nginx 内网各种 web 服务的汇总代理, 偶尔用于劫持外部网页，做些自动化的小后门

以及内网的文件服务器，上传服务器等

去年 8 月份，公司刚成立，一个人瞎倒腾了 3-4 天才大致搞定了这些。

docker 搭建的过程

- 用 `docker pull docker run`

以 `jenkins` 为例，找了一个 `docker image`

```
docker pull jenkins
docker run -d -p 49001:8080 -v $PWD/jenkins:/var/jenkins_home -t jenkins
```

done!

- 用 ~docker-compose~

我使用的是这种方式 ~~因为搜索到的第一个介绍是这样用的~~ , 示例的 compose file 如下:

```
version: '2'

services:
  oss:
    restart: always
    image: jfrog-docker-reg2.bintray.io/jfrog/artifactory-oss:latest
    volumes:
      - /var/opt/jfrog/artifactory/backup:/var/opt/jfrog/artifactory/backup:Z
      - /var/opt/jfrog/artifactory/logs:/var/opt/jfrog/artifactory/logs:Z
      - /var/opt/jfrog/artifactory/data:/var/opt/jfrog/artifactory/data:Z
      - /var/opt/jfrog/artifactory/etc:/var/opt/jfrog/artifactory/etc:Z
    ports:
      - "10081:8081"
```

一共花了 **半天**时间，完成了 gitlab, gitlab-runner, spark, artifactory 的搭建, 主要的时间还是花在了下载 ~docker image~ 的过程中。

遇到的问题

- 没有 image 和 container 的概念

  第一次接触，对于其中的概念没有理解清楚就开始用，直接导致了后面的问题。

  - 容器的数据没有持久化

    部署的容器完全以搜索到的介绍为准，让我 mount 数据目录的，我就操作。没让我 mount 的，我就直接启动。于是，在一次重启之后，artifactory 内下载下来的所有 jar 包都丢了。

- 权限

  mount 进去的数据文件的权限默认是 root (其实和使用的 image 相关，看它是怎么设置目录权限的，如果没有管你 mount 进去的文件，就默认为 root), 如果想手工调整，看该镜像是否提供了 entry point 用于修改。或者自己可以基于该 image 重编一个符合自己需求的镜像。

- 环境变量

  在安装完成 gitlab-runner (类似于 jenkins 的一个 ci 工具), 我希望以我指定的 java 和 maven 来执行打包等操作。于是将这些文件 mount 进去之后，需要设定一下环境变量来使用指定路径下的软件。compose file 中有一个 directive 叫 environments 可以完成。

- 指定 host 的解析

  由于我们的 mvn, ci 都是在内网，并且手动绑定 host 来指定到某个 ip, 所以也需要手工设置容器内的 host。通过 compose file 中的 extra_hosts 即可制定。

题外话

遇到的磁盘坏了的坑，于是搭了 raid-1，脚本每天备份，备份数据文件夹以及对应的 docker compose file 。以后 ~~服务器跪子~~ 再重新搭建应该更快了.

总结

搭建，部署，启动 docker 真的都是 好快，好快，好快 ...

科技改变生活