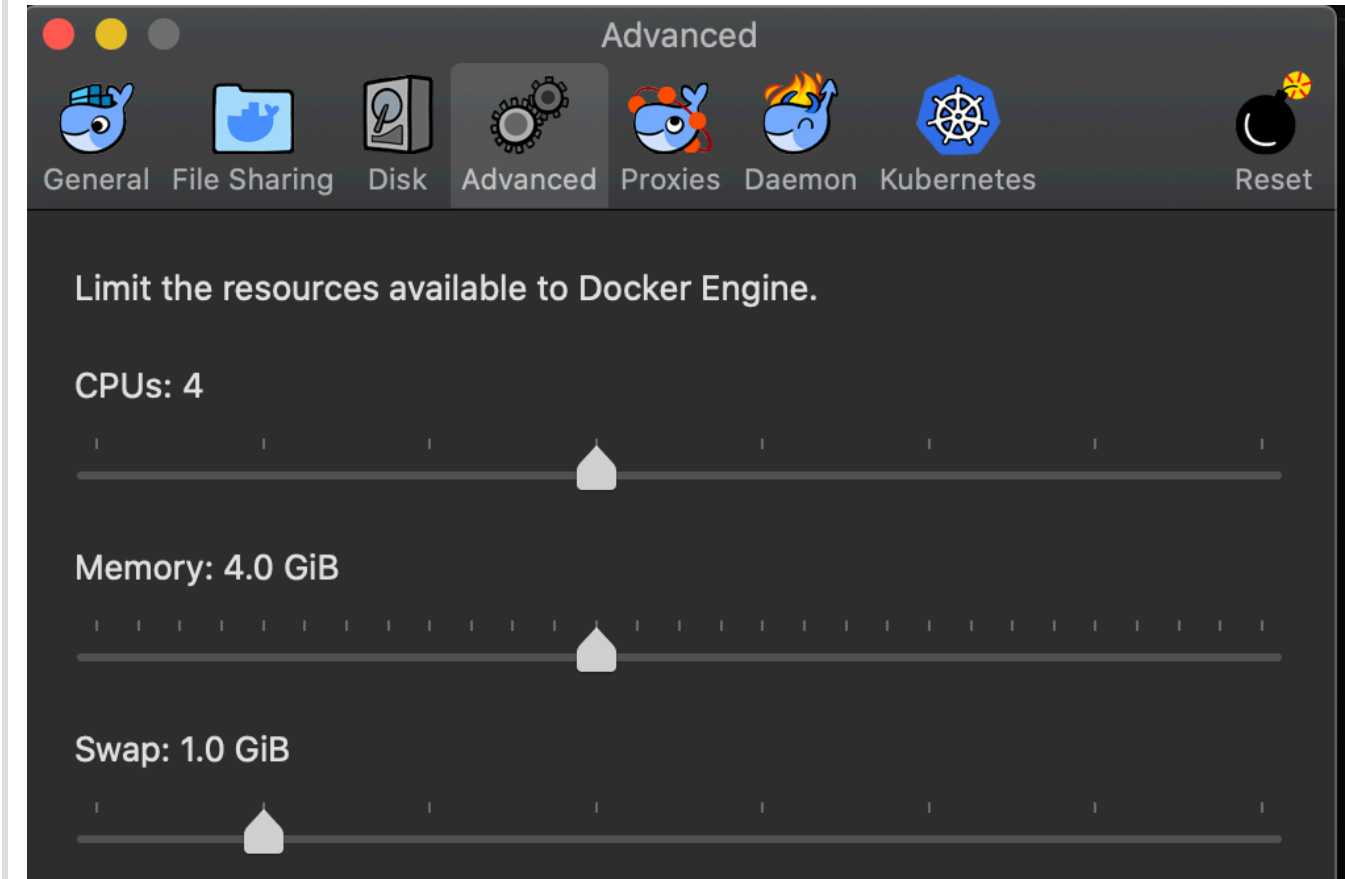


Chapter 13 环境搭建要点

以下内容所涉及的基础服务，默认你已经可以自主搭建，此处仅介绍关键细节。

本机启动devops服务

在执行docker-compose指令之前，请先确保本地的docker服务的资源分配至少 2c4g



Harbor

你有一台Harbor服务器，使用 Http 协议，默认 80 端口可以访问；

你有一个域名可以解析到此服务器上。

你的项目访问级别请勾选公开

新建项目

项目名称 *

访问级别

☒ 公开 

取消

确定

Gitlab

你有一台Gitlab服务器，使用admin账号登录，并添加 SSH Keys 。

SSH Key 是你的Jenkins服务器上的 ssh公钥 【位于 .ssh/id_rsa.pub 】。

Jenkins

你有一台Jenkins服务器；

这台机器上安装了 docker，并且使用过命令 `docker login` 登录了你的 Harbor 服务器；

这样你就可以将打包好的docker镜像推送到你的 Harbor服务器上。

配置你的Jenkins Job

关闭跨站伪造保护，参考文章：https://blog.csdn.net/ccc_bigdata/article/details/108080084

配置全局安全：

安全领域设置为 none；

Configure Global Security



Configure Global Security

Authentication

☐ Disable remember me

Security Realm

☐ Delegate to servlet container

☐ Jenkins' own user database

☒ None

新建：

Enter an item name

project-name_prod

» Required field **project-name:** 是你的真实应用名称

_prod: 是环境变量后缀, 这种写法是为了配合 devops 平台的环境名称, 是代码中的约定。



Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.



构建一个maven项目

构建一个maven项目.Jenkins利用你的POM文件,这样可以大大减轻构建配置.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



构建一个多配置项目

适用于多配置项目,例如多环境测试,平台指定构建,等等.



文件夹

创建一个可以嵌套存储的容器。利用它可以进行分组。视图仅仅是一个过滤器,而文件夹则是一个独立的命名空间,因此你可以有多个相同名称的内容,只要它们在不同的文件夹里即可。

配置:

配置参数化构建过程,此处是devops代码中的入参约定。

project-name_prod

General

Source Code Management

Build Triggers

Build Environment

Pre Steps

Build

Post Steps

构建设置

Post-build Actions

☐ Use alternative credential

☒ This project is parameterized

Add Parameter

☐ Boolean Parameter

☐ Choice Parameter

File Parameter

Multi-line String Parameter

Password Parameter

☐ Run Parameter

☒ String Parameter

凭据参数

Advanced...

需要添加 3 个参数, ENV、NAME、TAG

String Parameter

Name

ENV

Default Value

prod

Description

String Parameter

X ?

Name ?

TAG

Default Value ?

Description ?

General

Source Code Management

Build Triggers

Build Environment

Pre Steps

Build

Post Steps

构建设置

Post-build Actions

Source Code Management

☐ None

☒ Git

Repositories

Repository URL

github

https://github.com/any/any.git

git

Credentials

- 无 -

+ Add

Advanced...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

\${TAG}

Build

此处的 Build 配置内容是基于笔者自己的代码结构配置，读者可以根据自己代码结构的编排进行调整。

Root POM

pom.xml

Goals and options

clean package

Post Steps

☐ Run only if build succeeds ☐ Run only if build succeeds or is unstable ☒ Run regardless of build result

Should the post-build steps run only for successful builds, etc.

Execute shell

这里是你的 Harbor 服务器的域名

Command

```
docker build -t harbor.mirror.gcr.io/${NAME}/${NAME}${ENV}:${TAG} .
docker push harbor.mirror.gcr.io/${NAME}/${NAME}${ENV}:${TAG}
```

以上Job配置就结束了，build过程部分，需要读者按照自己的代码编排来修改。

你还需要一台服务器用来部署你在devops平台新建的应用

你有一台服务器，对外开放 TCP端口：4789 和 TCP端口：8080

这台机器上安装了 docker，配置 TCP端口：4789 监听 docker.socket；

举例：

如果这台是centos系统的服务器，并且你是使用 systemctl start docker 来启动；

那么你可以修改docker.service文件 /usr/lib/systemd/system/docker.service

```
ExecStart=/usr/bin/dockerd --insecure-registry harbor.xxxx.cn -H unix:///var/run/docker.sock -H tcp://0.0.0.0:4789 -H fd:// --con
```

上面的 --insecure-registry harbor.xxxx.cn 配置了你的harbor仓库地址

```
-H unix:///var/run/docker.sock -H tcp://0.0.0.0:4789 -H fd:// --containerd=/run/containerd/containerd.sock
```

配置了你的unix 套接字文件/tcp 监听端口和 fd 文件描述符

为了配合devops服务的健康检查，你的应用需要对外提供健康检查接口：

例如 springboot 应用：

application.properties

```
server.servlet.context-path=/project-name
spring.application.name=project-name
server.port=8080
```

controller

```
@GetMapping("/actuator/health")
public Object health() {
    Map<String, Object> rtn = new HashMap<>();
    rtn.put("status", "UP");
    return rtn;
}
```

例如 前端应用在 nginx.conf 中配置：

...

```
listen      80 default;  
listen      8080 default;
```

...

```
location = /project-name/actuator/health {  
    return 200 '{"status":"UP"}';  
}
```

...