

CPT202 WebApp Deployment & Docker Tutorial



Tianxin Zhao

Tianxin.Zhao21@student.xjtlu.edu.cn

Group25, ICS, AY2024, XJTLU

CPT202 WebApp Deployment & Docker Tutorial

A. 简介:

1. Docker是什么
2. Docker和虚拟机有什么不同
3. 为什么CPT202推荐用Docker部署APP

B. Docker 镜像/容器/网络

1. Docker镜像
2. Docker容器
3. Docker仓库
4. Docker网络

C. 准备工作

1. 登录阿里云服务器
2. 命令行连接服务器(可选)
3. 放行服务器端口
4. 安装Docker社区版
5. 安装Portainer
6. 安装Mysql(可选)

D. 部署APP

1. 调试代码
2. 打包为jar
- 2.5 本地Jar测试(可选)
3. 上传Jar文件(可选)
4. 分析组件
5. 使用Portainer构建Docker镜像
6. (), 启动!
7. 访问&测试
8. 容器异常退出

F. Docker生命周期

1. 下线/重启
2. 升级
3. 数据持久化(可选)
4. 快速部署(可选)

A. 简介:

本文档是应CPT202教学组邀请提供的 Docker与CI/CD 部分(非官方)教程，旨在花更短的时间，敲更少的代码，用更少的精力上线自己小组的App。从部署服务器开始，讲解如何使用Docker以及web图形化界面快速调试，部署，上线，更新APP，但仅包含Docker生命周期中CPT202会用到的部分，不是正经的Docker教程，不细讲Docker的语法(因为这门课的主要内容不是Docker)，如果您对这方面有兴趣，可以参考相关链接：

[Docker Docs](#)

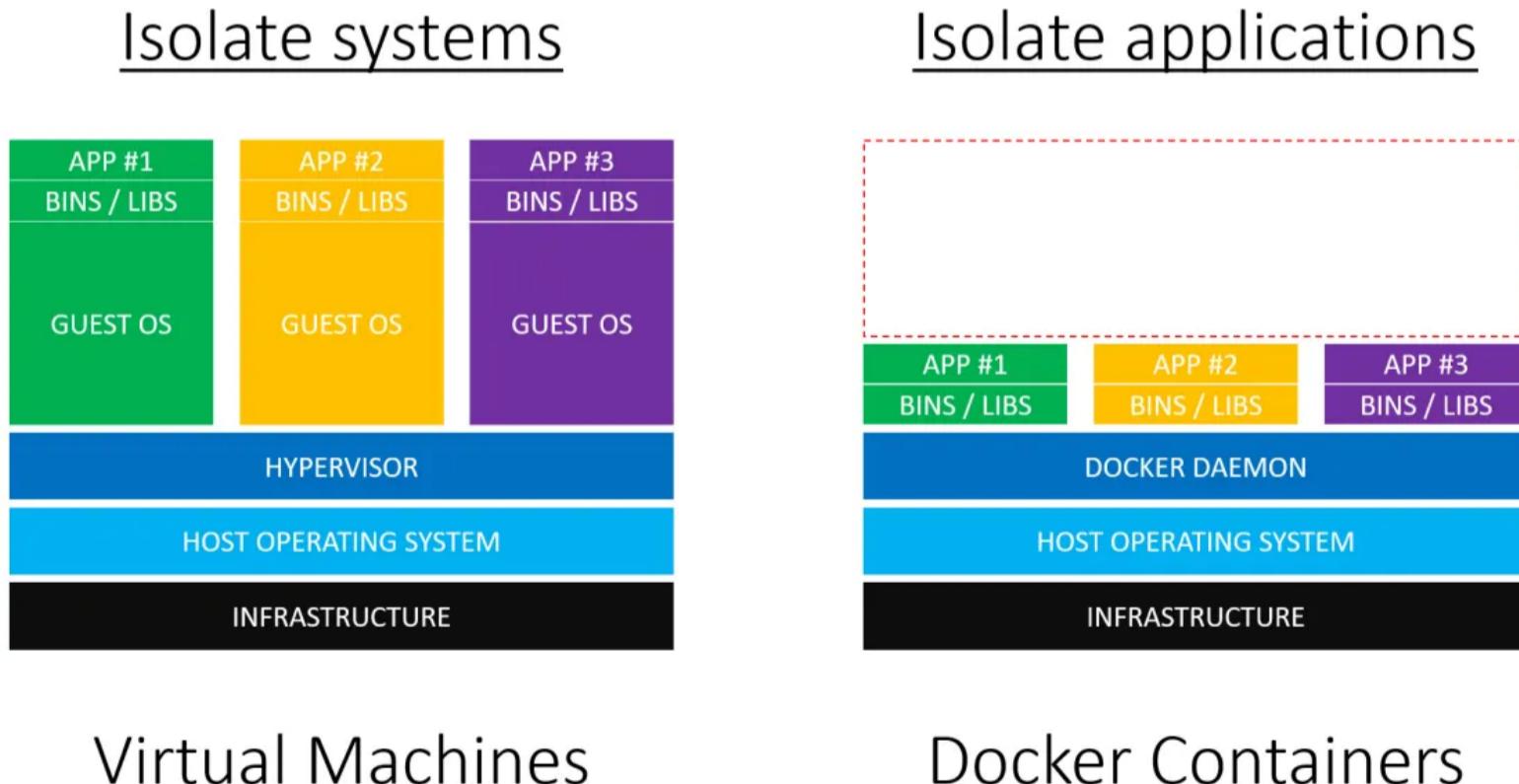
[Kubernetes](#)

1. Docker是什么

Docker像**虚拟机**一样创建了一个隔离的环境，在该环境内的修改与宿主系统隔离，与宿主系统无关。官方将Docker的特性称作：Build, Ship and Run Any App, Anywhere，即：**构建应用及其依赖环境的容器镜像；分发这个镜像到任何服务器；在任意Docker环境中运行镜像。**只要有镜像(模板)，便可在任何安装了Docker的机器上运行，每个机器的运行时的行为都是一样的。

2. Docker和虚拟机有什么不同

Docker**更轻量**，虚拟机需要安装从操作系统，在从操作系统中安装部署各种应用。资源占用多，启动速度慢。Docker利用Linux容器技术（LXC），即Linux提供的**内核级进程隔离**功能，容器直接运行于宿主机的操作系统之上，并**共享宿主机的内核**，减少了不必要的操作系统负担，(模拟一个完整的操作系统有很大的开销且无必要)。由于Docker不需要Hypervisor实现硬件资源虚拟化，有更少的抽象层，因此在CPU与内存等利用率上有明显优势。此外，虚拟机一般不可以嵌套虚拟化(宿主机上安装虚拟机，虚拟机内一般不能再安装虚拟机)，Docker没有这个限制。



3. 为什么CPT202推荐用Docker部署APP

- 一致性：**开发APP会同时用到Java, MySQL等组件，进入发布阶段时需确保生产环境中(即你的服务器内)安装了所有依赖，且每个依赖组件要与开发时的版本一致，若有不同可能会导致各种各样的问题。例如，开发时使用的是Java21+新版Mysql(8.0.36)，但服务器上的环境是Java1.8以及旧版MySQL(或者根本没安装)，代码无Error但是项目无法启动。

-> Docker 容器包含应用本体及其所有依赖项，避免了“**在我的机器上能运行，但换个环境就是不能运行**”。

- **避免繁琐配置**: 大多数同学开发时使用Windows，但部署时可能会学校分配的Linux，由于部分同学不熟悉Linux命令行，配置环境可能比较复杂。但如果使用Docker部署，不需要手动安装依赖，直接使用镜像启动即可。“**如果在自己的Windows/MAC内的Docker能正常运行，则该镜像在学校分配的服务器上肯定能正常运行**”。
- **快速部署**: 对于CPT202的App，拿到服务器后仅需一个Docker镜像以及一行指令就能在几秒钟上线，如果更换了服务器或者服务器重装了系统，可以不需要重复配置环境(如安装Java等)，快速上线
-> 如果使用Docker Compose，则会复杂些，但这门课里一般用不到
- **环境隔离**: Docker容器内数据是与系统隔离的*，传统方法下更新App版本需要停止进程，删除旧版配置以及数据库中文件，但使用Docker部署时直接停止容器即可。
*这里不考虑容器数据卷
- **安全**: 传统部署模式下修改系统配置会影响系统运行，但在Docker容器的修改一般仅限于**容器自身**，例如Docker部署时误删重要文件只会影响该容器，不影响整个系统，重新下线再上线Docker容器即可；但是传统部署模式下可能需要重装整个服务器。
- **Report**: 用了Docker应该能在CW2 Report里多扯点东西

B. Docker 镜像/容器/网络

1. Docker镜像

Docker镜像可简单理解为一个只读的**模板**文件，类似与上学期CAN201Lab给的的虚拟机镜像(CAN201.iso)。我们用这个模板来启动Docker**容器**，一个镜像可以创建**多个**容器。我们在之后使用**DockerFile**来构建镜像。

2. Docker容器

容器就像运行时的虚拟机，小组开发的**App**运行在这个容器里面。(可理解为Docker Logo 鲸鱼上的集装箱，Docker这个软件是下面的鲸鱼，承载容器服务)。一个镜像可以创建多个容器，但是CPT202每个版本我们只需要使用一个镜像启动一个容器(因为我们只需要同时运行一个软件)。

3. Docker仓库

Docker仓库是集中存放**镜像**文件的场所。可以拉取别人定义好的镜像。Docker官方的仓库为Docker Hub(<https://hub.docker.com/>)，暂时不会用到

4. Docker网络

容器中的App在服务器内运行时，需要通过网络(即**ip:端口**的形式)对外提供服务，而前面提到了每个运行时的容器就像一个虚拟机，每个容器会有一套自己的端口，我们需要将容器的端口以某种方式**映射**到服务器上。Docker一共提供四种网络模式，我们只会用到下面的两种。

模式	原理	示例	优缺点
Host	容器绕过Docker 的虚拟网络层，直接绑定到宿主机的网络接口，无需端口映射或NAT转换，直接使用宿主机的网络地址(ip:端口)访问容器。	访问服务器ip:端口即访问容器:端口	简单，性能好；但可能出现端口冲突，例如两个容器使用了同一个端口

模式	原理	示例	优缺点
Bridge	每个容器有一个内部的ip，使用NAT来处理转发，容器的某个端口绑定至宿主机的某个端口	某个容器的a端口绑定到宿主机的b端口时，在外部访问宿主机 ip:b，数据进入宿主机会被自动转发至容器:a	容器与宿主机，与外部环境有选择性的隔离；但是配置时需要手动指定端口映射规则

如何选择：如果你清楚的知道你的程序使用了哪些端口(找项目内的配置文件)，就用Bridge，否则用Host。

Tips: 这里是为了快速上线，实际生产环境会更复杂一些

C. 准备工作

我们假设你的小组使用IDEA开发，项目控制为Maven，版本控制为Git，OS为Linux-Ubuntu，服务器为阿里云，为了简化代码数量和部署难度，在接下来的教程中将尽可能使用图形化界面。

1. 登录阿里云服务器

学校会为每一组分配阿里云服务器，用账号密码登录阿里云(也可能是用RAM登录，具体是给access key，还是密码，还是直接给服务器我也不知道，但是登录大同小异)如果还没领就发邮件给老师，注明Operating system为Ubuntu(Linux)，组号为自己的组号，之后会下发账号密码。

Tips1: Windows服务器有GUI，使用简单，但Windows下使用Docker要使用WSL技术(底层模拟了Linux内核的行为，再在模拟出的内核上启动Docker)，2vcpu 4g实例下性能开销占比很大，因此首选Ubuntu，如果实在不想用Linux，再选择Windows。

Tips2: 如果已经发送了邮件，但是想用另一个操作系统(例如申请的是Windows，但想换成Linux，反之亦然)，在登陆后阿里云控制台内停止运行该机器，点击全部操作，搜索系统，选择合适的进行更换即可。

Tips3: 如果不想使用Docker部署可以参考最后一个章节

以下为登录服务器的步骤：



概览 资源管理 运维管理 安全中心 成本管理 最近访问

资源概览

我的资源 最近创建 最近操作 云上架构

地域 全部	资源类型 全部	地域	创建时间
d-f8zb1k5pyi59kuvysw0	云服务器 ECS-磁盘	华南2 (河源)	2024-03-27 20:00:42
i-f8z7snfrqthprwbmbg4w	云服务器 ECS-实例	华南2 (河源)	2024-03-27 20:00:00

查看更多 < >

云产品可观测 CloudLens for OSS CloudLens for SLS CloudLens for ALB

CloudLens for OSS

您当前暂未开通Cloud Lens for OSS

针对对象存储 OSS，实现用户资源用量监控、操作审计、访问统计、健康度分析、异常事件回溯和问题定位等工作。

了解OSS Lens详情

统一云产品运维数据采集
更全面的云产品运维辅助分析
可灵活订阅的数据平台

当前功能依赖于SLS，未开通无法使用
前往开通 开通成功，请点刷新

用量分析 访问分析 安全分析 性能监控 异常检测 数据保护

135.13 TiB

登陆阿里云后点击控制台，找到自己的服务器，一般名字叫ECS_XXXX，点击进入控制台，选择远程连接(模式WorkBench)，如果忘记密码或者未设置密码点击重置密码，按要求重置，使用用户名-密码登录(在以后实际生产环境中建议用ssh密钥/Session，更安全，202为了方便就用密码登录)

云服务器 ECS

实例ID: i-f8z7snfrqthprwbmbg4wZ 运行中

实例详情 监控 安全组 云盘 快照一致性组 快照 弹性网卡 定时与自动化任务 操作记录 健康诊断 事件

基本信息

实例ID: i-f8z7snfrqthprwbmbg4wZ 远程连接 重置密码 实例名称: iF8z7snfrqthprwbmbg4wZ 运行中

所在可用区: 河源 可用区A 付费类型: 包年包月/按量计费

健康状态: 正常 实例问题排查: 实例问题排查历史

配置信息

实例规格: ecs-e8m1large 更换 购买相同配置 主私网IP: 专有网络: vpc-f8zh8zsrfj9642c3galun 更换 公网带宽: 1 Mbps 更改带宽

CPU&内存: 2核(vCPU) 2 GiB 镜像ID: ubuntu_22_04_uefi_x64_20G_alibase_20230515.vhd 创建自定义镜像 虚拟交换机: vsw-f8zb15pyi59kuvysw5 更换 网络类型: 专有网络 带宽计费方式: 按需带宽 支持自动续费: 自动续费 支持手动续费

绑定资源

安全组: sg-f8zb1k5pyi59kuvysw 配置规则 云盘: 1个云盘 基于初始化化云盘 快照: 快照一致性组 弹性网卡: eni-f8zb15pyi59kuvysw5 绑定辅助弹性网卡 弹性公网IP实例ID: - 插件中私网IP: -

文件 编辑 视图 实例 会话 功能 帮助 数据库

登录实例

实例: i-f8z7snfrqthprwbmbg4wZ 华南2(河源) 简体中文

认证方式: 密码认证 (selected) SSH密钥认证 凭据认证 临时SSH密钥认证

用户名: root

密码: ****

【运维安全中心（堡垒机）】专业、安全的运维管控审计平台，[最高30元/每月](#)

立即购买 查看产品介绍

重置密码 完整选项 取消 确定

2. 命令行连接服务器(可选)

如果想更便捷的登录，或学校只给你了服务器的账号密码可以用下面的方法：

首先打开系统的终端，windows下是powershell，mac是Terminal，命令是：

```
ssh 用户名@服务器地址  
ssh root@192.168.1.1  
将这里的用户名和ip换成实际的值
```

```
Windows PowerShell  
PS C:\Users\18362> ssh root@x.x.x.x  
The authenticity of host 'x.x.x.x (x.x.x.x)' can't be established.  
ED25519 key fingerprint is SHA256:MnkeyxnYX/EAvZNUU180cn5QWG0XYq5GT5fsVZZNvV0.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])?  
出现这个提示是验证ED25519指纹，防中间人攻击，输入yes
```

```
root@x.x.x.x's password:
```

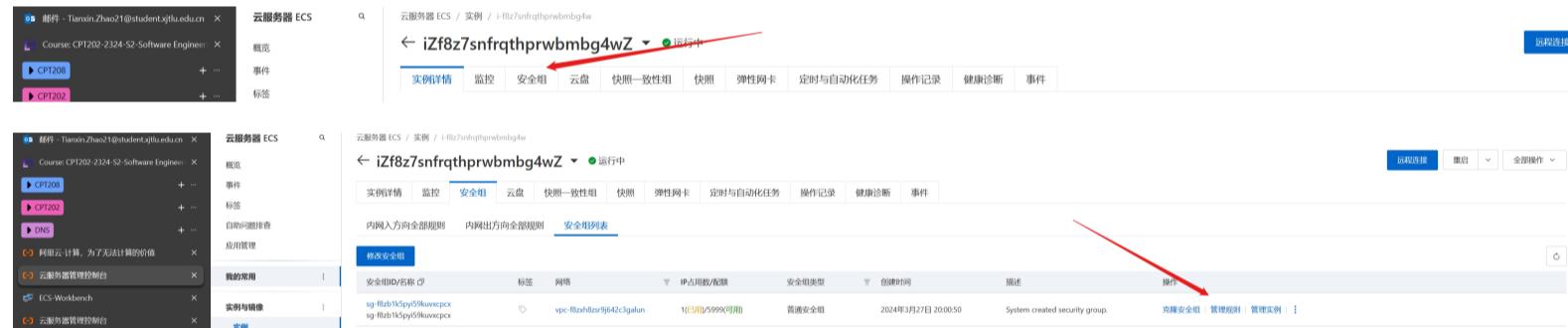
出现这个时输入密码，注意输入内容不回显，用键盘盲打，大小写要一致

Tips: 也可使用脚本自动填充指令，用快捷键登录(这里不演示)

3. 放行服务器端口

上线服务后，自己的服务可能会使用除了80端口以外其他的端口，为避免无法访问(例如app没有问题，但是数据库就是无法远程连接)，需要放行所有端口，这部分由阿里云管理：

打开安全组，点击管理规则



删除所有原有规则，添加如下规则并保存(每个字段都要填写正确)



Tips:上面的规则放行所有协议的入站连接，方便调试，实际生产环境不要这么做，很不安全

4. 安装Docker社区版

登录服务器，在命令行输入：

```
apt update  
# 更新软件源
```

```

sudo apt-get update
sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs)
stable"
sudo apt-get update
sudo apt-get install docker-ce
# 下载Docker社区版,上面的的command一个一个复制粘贴过去
# 遇到Do you want to continue? [Y/n] 一律选Y
# 遇到Press [ENTER] to continue or Ctrl-c to cancel. 按Enter

```

安装完成验证是否安装成功:

```
docker --version
```

输出类似信息则安装成功

```

root@iZf8z7snfrqthprwmbg4wZ:~# docker --version
Docker version 26.0.0, build 2ae903e

```

5. 安装Portainer

Portainer是Docker的轻量级图形化监视程序(就像Navicat与Mysql的关系), Docker本身是使用命令行操作的, 新手较难使用, 例如报错信息, 监控, 部署等都需要指令完成, 但是Portainer提供了基于web的图形化控制, 点相应的开关就可以执行部分原本需要在命令行手动输入指令的操作, 建议使用, 以下为安装方法:

```

# 先不用理解这个Command的意义
docker volume create portainer_data

```

```

# 先不用理解这个Command的意义
docker run -d -p 8000:8000 -p 9000:9000 --name portainer --restart=always -v
/var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:latest

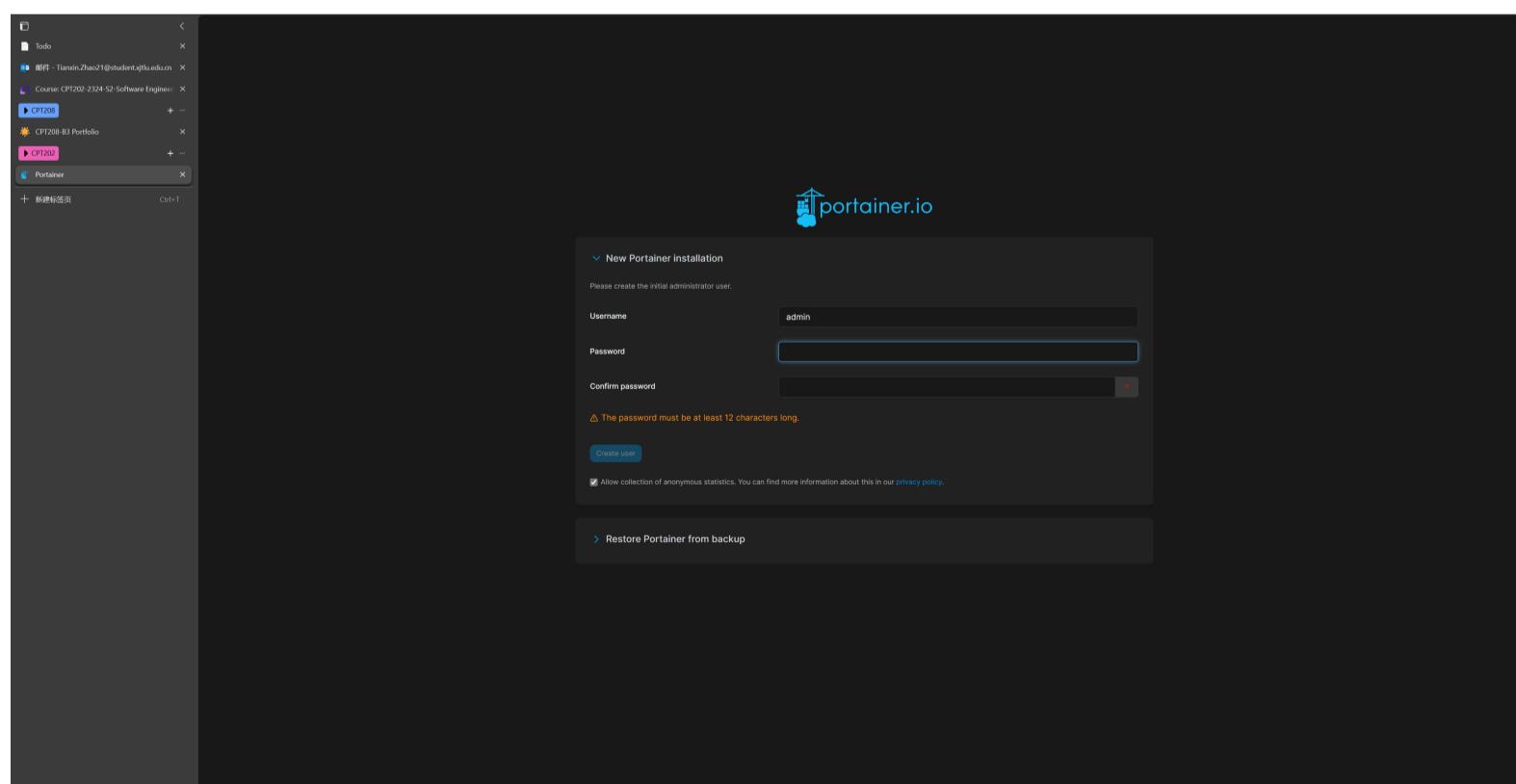
```

```

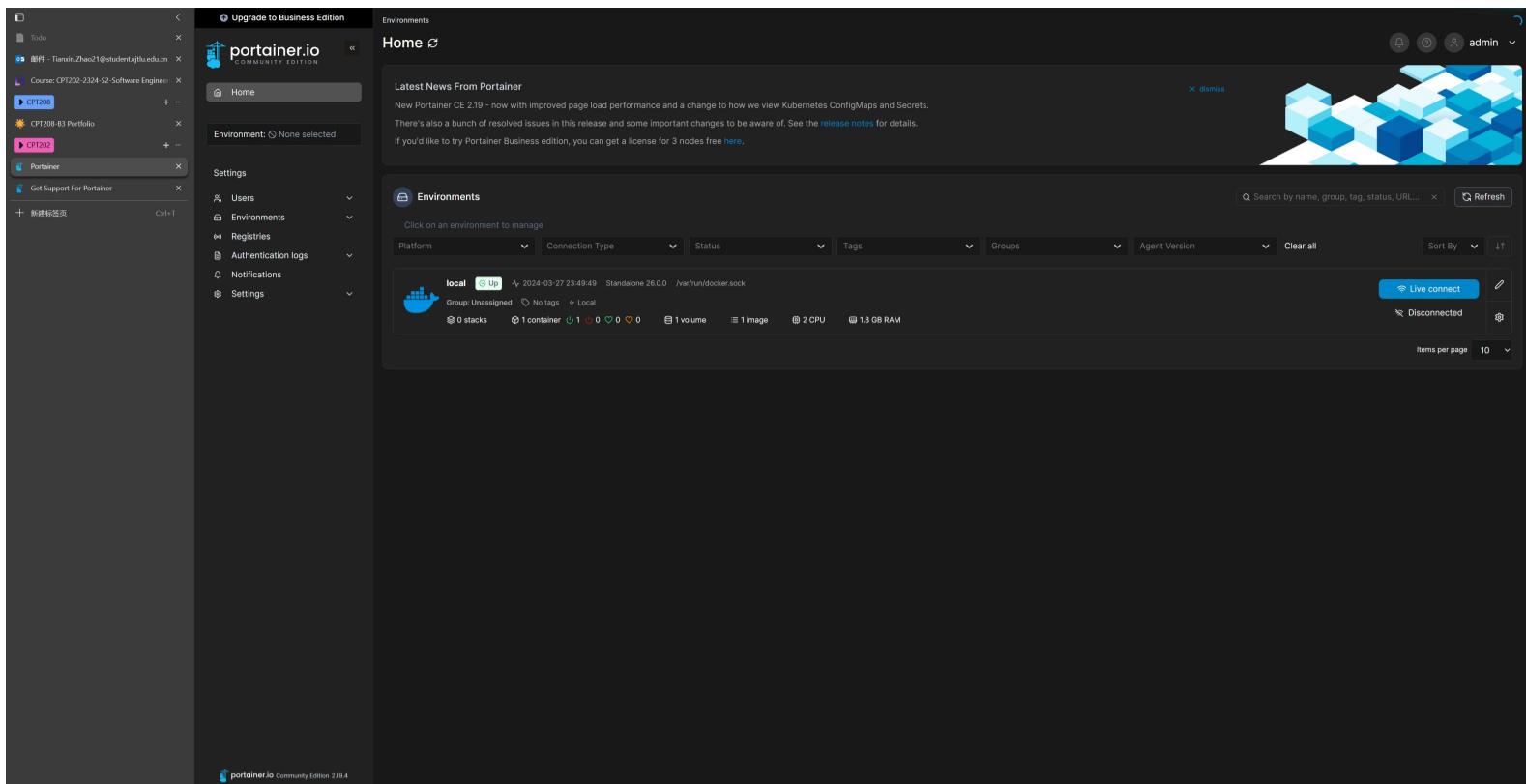
# 打印类似信息是正常的, 如下载速度一直很慢可换源为阿里云Docker仓库
Unable to find image 'portainer/portainer-ce:latest' locally
latest: Pulling from portainer/portainer-ce
379538b6d68e: Pull complete
4ea3e2c3a39b: Pull complete
5171176db7f2: Downloading [=>                                         ] 445.6kB/13.98MB
52e9438966a5: Downloading [>                                         ] 180.8kB/17.99MB
43d4775415ac: Waiting
.....

```

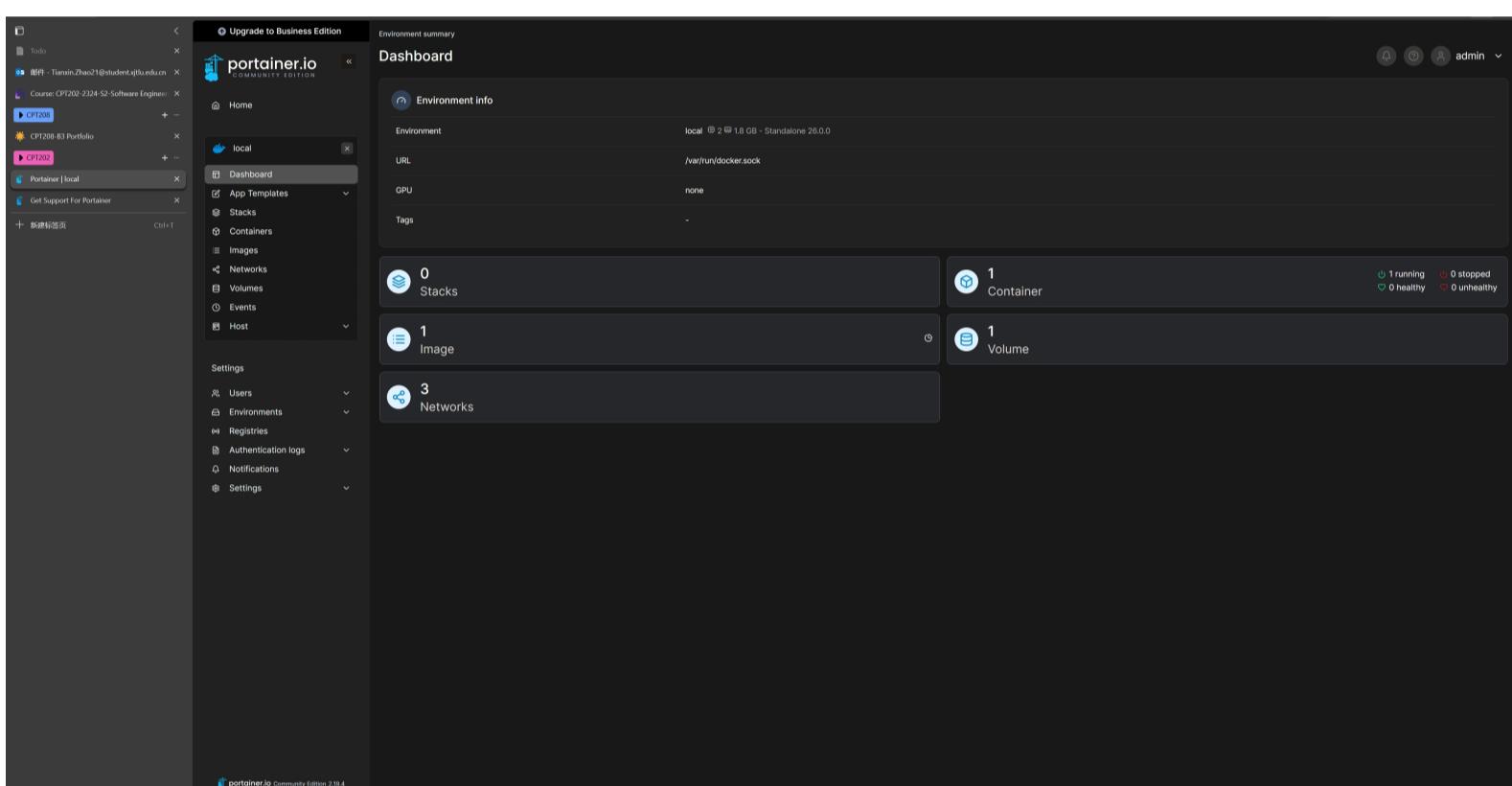
打开浏览器, 输入 **你的服务器ip:9000**



显示上述界面, 说明Portainer安装成功, 自行设置账号密码, 以后大部分时间用都是这个来控制Docker



点击local-LiveConnect, 这里选本地主机(127.0.0.1), 也就是你的阿里云服务器, 其他的几个选项是给集群(例如k8s)用的, 我们不会用到



这里显示的Image&Container 就是前面提到的镜像与容器, 后续不熟悉命令行可以直接用这里提供的选项来控制, 简化运维难度

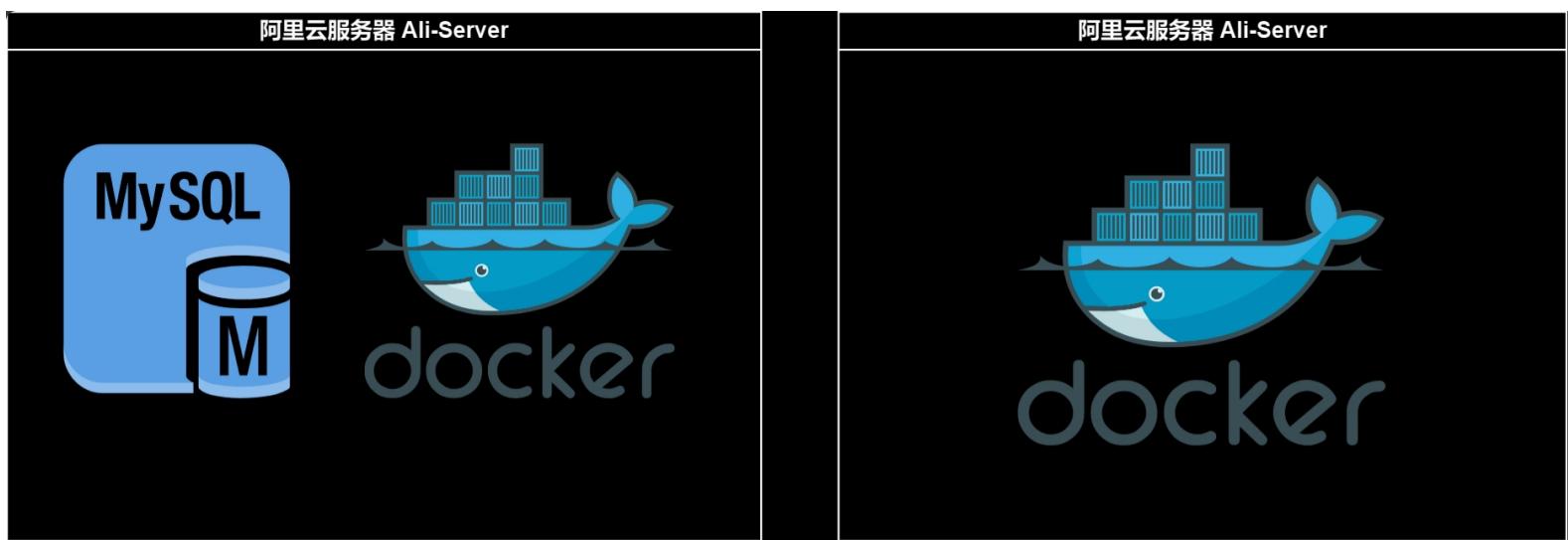
Tips: Portainer启动时使用了9000端口, 这是为了方便连接, 实际生产环境要用https(9443端口), 否则密码等是明文传输的

6. 安装Mysql(可选)

如果你的服务用到了数据库, 可以使用该指令安装数据库, 数据库的SQL语句(增删改查等)这里不演示

```
sudo apt install mysql-server -y
```

Tips: 也可不直接安装MySql, 而是在Docker内将其容器化, 如右图, 但**为方便调试建议按左图的方式直接安装**



左侧: 系统内安装Mysql和Docker，小组的App是Docker容器(一个鲸鱼logo上的集装箱)，这个容器会用到docker外的Mysql

右侧: 系统内安装Docker，小组的App和Mysql都是Docker容器(相当于有两个鲸鱼logo上的集装箱，一个是自己的App，一个是Mysql；当然你也可以把Docker&Mysql放到同一个容器中)

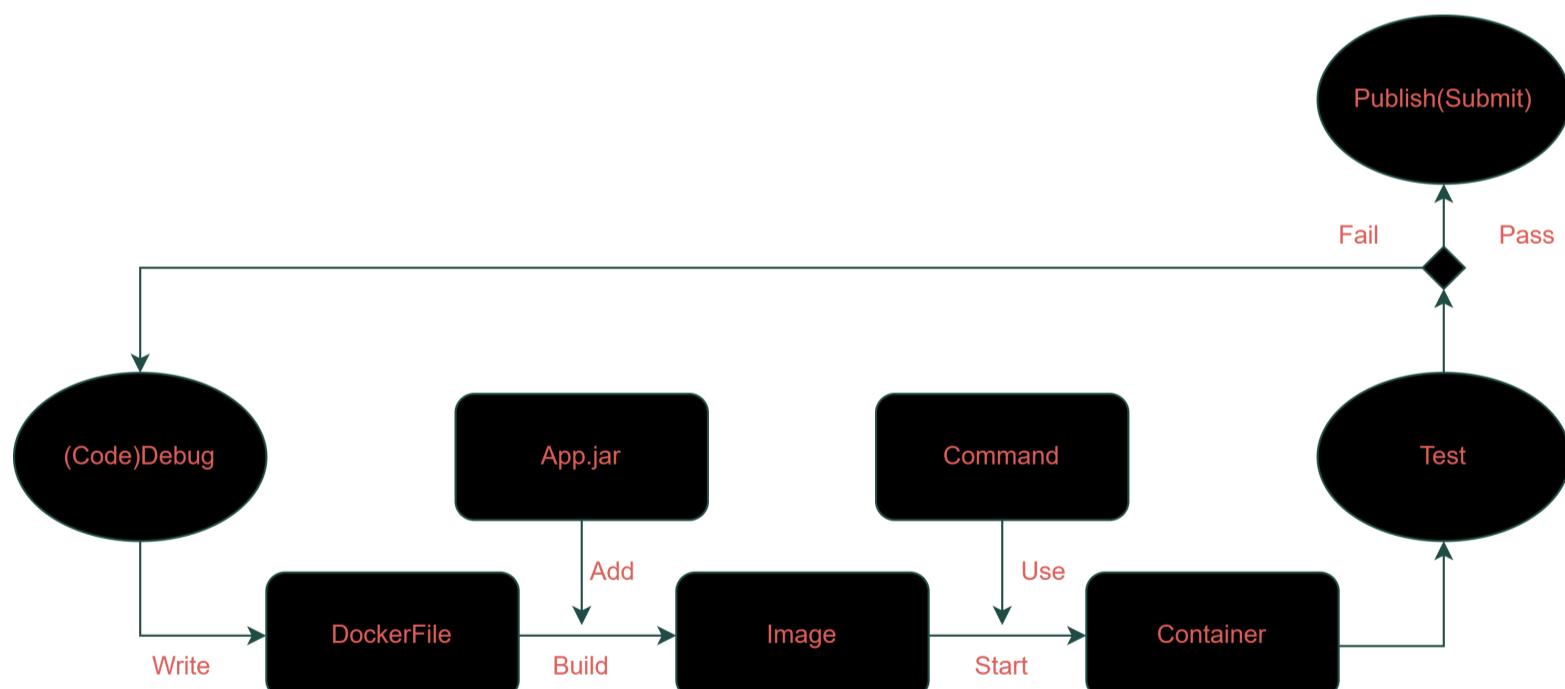
后续教程是基于左图的方式

D. 部署APP

在CPT202小组作业中，使用Docker部署APP需要:

1. 分析本小组使用的运行时组件，以及App会用到的所有端口(仅一次，1min)
2. 小组内负责运维的同学学习如何使用Docker(运维及Debug的同学，仅一次，30min)
3. 服务器安装Docker(运维及Debug的同学，5min)
4. 生成Docker镜像(每个版本仅一次，15min)
5. 使用该镜像来启动容器(每个版本仅一次，1min)

流程:



1. 调试代码

使用Docker部署前需要先确保能在IDEA中正常运行，假设你已经完成初始版本的开发，点击右上角绿色箭头(我们的名称可能和你的不一样)



```

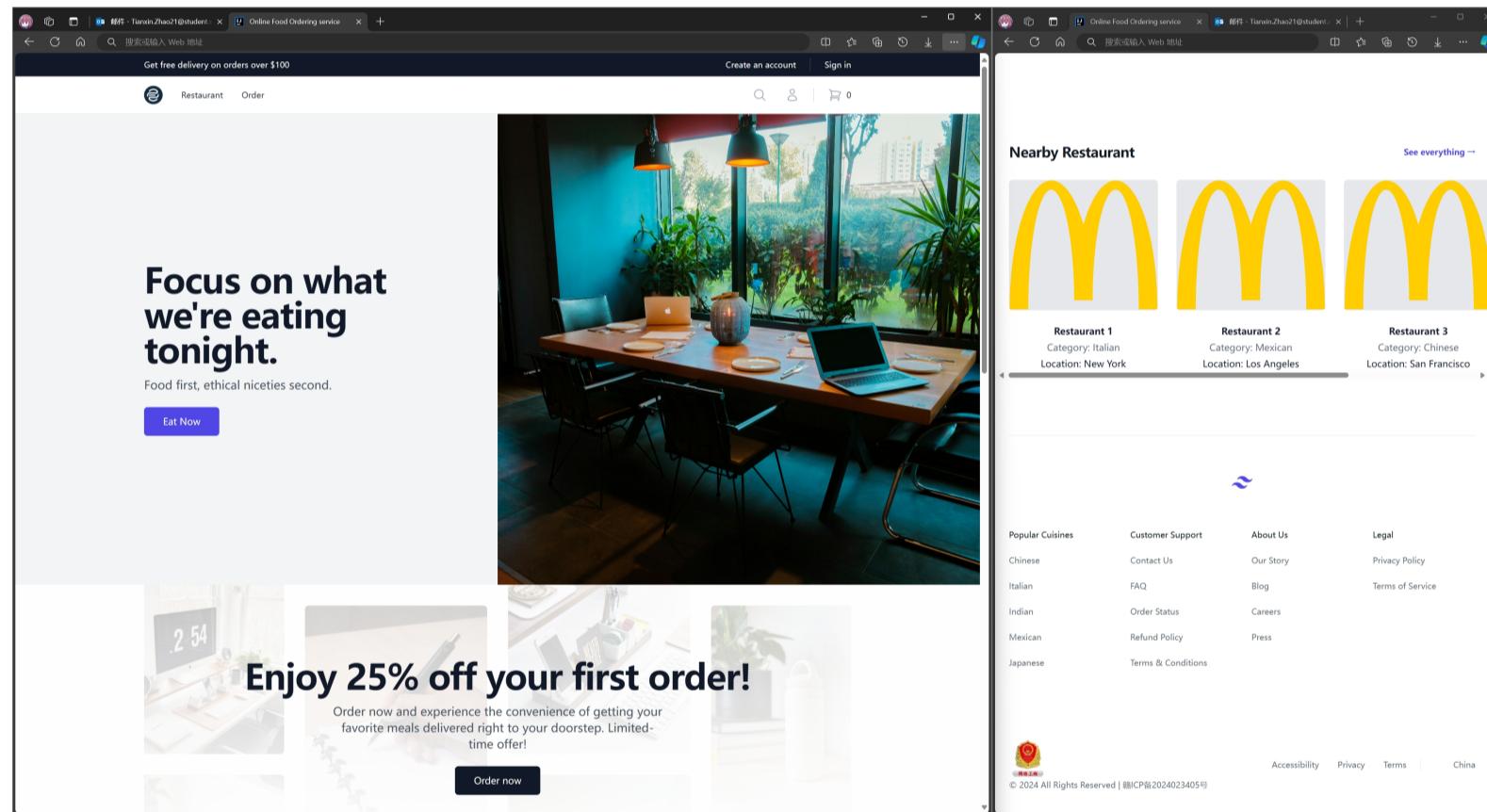
D:\JAVA\JDK21\bin\java.exe ...
```
:: Spring Boot :: (v3.2.3)

2024-03-28T18:59:07.205+08:00 INFO 12728 --- [restartedMain] w.cpt202.backend.BackendApplication : Starting BackendApplication using Java 21.0.2 with PID 12728 (F:\PROJECTS\CPT202\Backend\target\classes started by 18362)
2024-03-28T18:59:07.206+08:00 INFO 12728 --- [restartedMain] w.cpt202.backend.BackendApplication : No active profile set, falling back to 1 default profile: "default"
2024-03-28T18:59:07.235+08:00 INFO 12728 --- [restartedMain] o.s.b.devtools.restart.ChangeableUrls : The Class-Path manifest attribute in F:\FILES\MAVEN\com\mchange\c3p0\0.9.5.2\c3p0-0.9.5.2.jar referenced one or more files that could not be found: file:/F:/FILES/MAVEN/com/mchange/c3p0/0.9.5.2/c3p0-0.9.5.2.jar!/org/springframework/beans/factory/xml/SpringResource.class
2024-03-28T18:59:07.235+08:00 INFO 12728 --- [restartedMain] o.e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults activated. Set 'spring.devtools.add-properties' to 'false' to disable
2024-03-28T18:59:07.235+08:00 INFO 12728 --- [restartedMain] o.e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.web' property to 'DEBUG'
2024-03-28T18:59:07.656+08:00 WARN 12728 --- [restartedMain] trationDelegate$BeanPostProcessorChecker : Bean 'org.springframework.ws.config.annotation.DelegatingWsConfiguration' of type [org.springframework.ws.config.annotation.DelegatingWsConfiguration] is not eligible for getting processed by all BeanPostProcessors (for example: not eligible for auto-configuration). This may result in unexpected behavior when processing this bean. To ignore this warning, re-enable processing of this bean using @EnableBeanPostProcessors or a BeanPostProcessor Advisor
2024-03-28T18:59:07.664+08:00 INFO 12728 --- [restartedMain] .w.s.a.s.AnnotationActionEndpointMapping : Supporting [WS-Addressing August 2004, WS-Addressing 1.0]
2024-03-28T18:59:07.830+08:00 INFO 12728 --- [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2024-03-28T18:59:07.838+08:00 INFO 12728 --- [restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-03-28T18:59:07.868+08:00 INFO 12728 --- [restartedMain] o.a.c.c.C.[Tomcat].[localhost].[] : Starting Servlet engine: [Apache Tomcat/10.1.19]
2024-03-28T18:59:07.868+08:00 INFO 12728 --- [restartedMain] w.s.c.ServletWebServerApplicationContext : Initializing Spring embedded WebApplicationContext: Root WebApplicationContext: initialization completed in 632 ms
2024-03-28T18:59:08.177+08:00 INFO 12728 --- [restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2024-03-28T18:59:08.197+08:00 INFO 12728 --- [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path ''
2024-03-28T18:59:08.203+08:00 INFO 12728 --- [restartedMain] w.cpt202.backend.BackendApplication : Started BackendApplication in 1.212 seconds (process running for 1.643)
2024-03-28T18:59:15.357+08:00 INFO 12728 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing DispatcherServlet 'dispatcherServlet'
2024-03-28T18:59:15.358+08:00 INFO 12728 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 0 ms

```

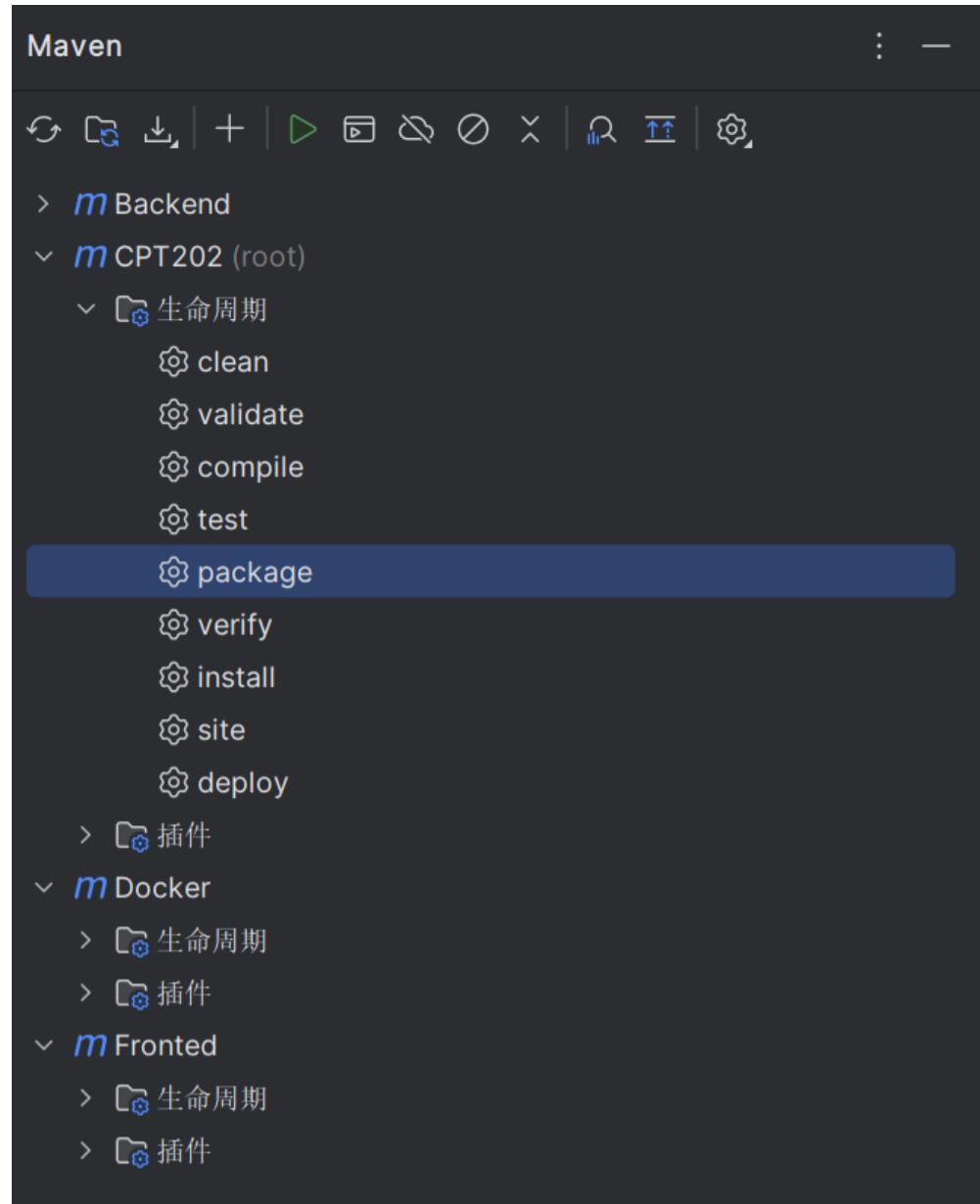
查看控制台输出是否有报错，用浏览器打开对应的url，检查网页的功能和UI是正常的，这里一定要仔细检查每个功能，不然上线后再修改会很浪费时间

`http://localhost:端口/`  
例如本服务使用8000端口，就用`http://localhost:8000/`



## 2. 打包为jar

测试后，打开IDEA右侧的Maven，选择自己的项目，点击生命周期-package.将整个项目打包为.jar文件。然后在文件夹内找到这个jar文件(一般在target文件夹)。



```
[INFO] Reactor Summary:
[INFO]
[INFO] CPT202 1.0-SNAPSHOT SUCCESS [0.001 s]
[INFO] Fronted 1.0-SNAPSHOT SUCCESS [17.088 s]
[INFO] Backend 0.0.1-SNAPSHOT SUCCESS [01:04 min]
[INFO] Docker 1.0-SNAPSHOT SUCCESS [0.014 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:39 min
[INFO] Finished at: 2024-03-28T19:29:07+08:00
[INFO] -----
```

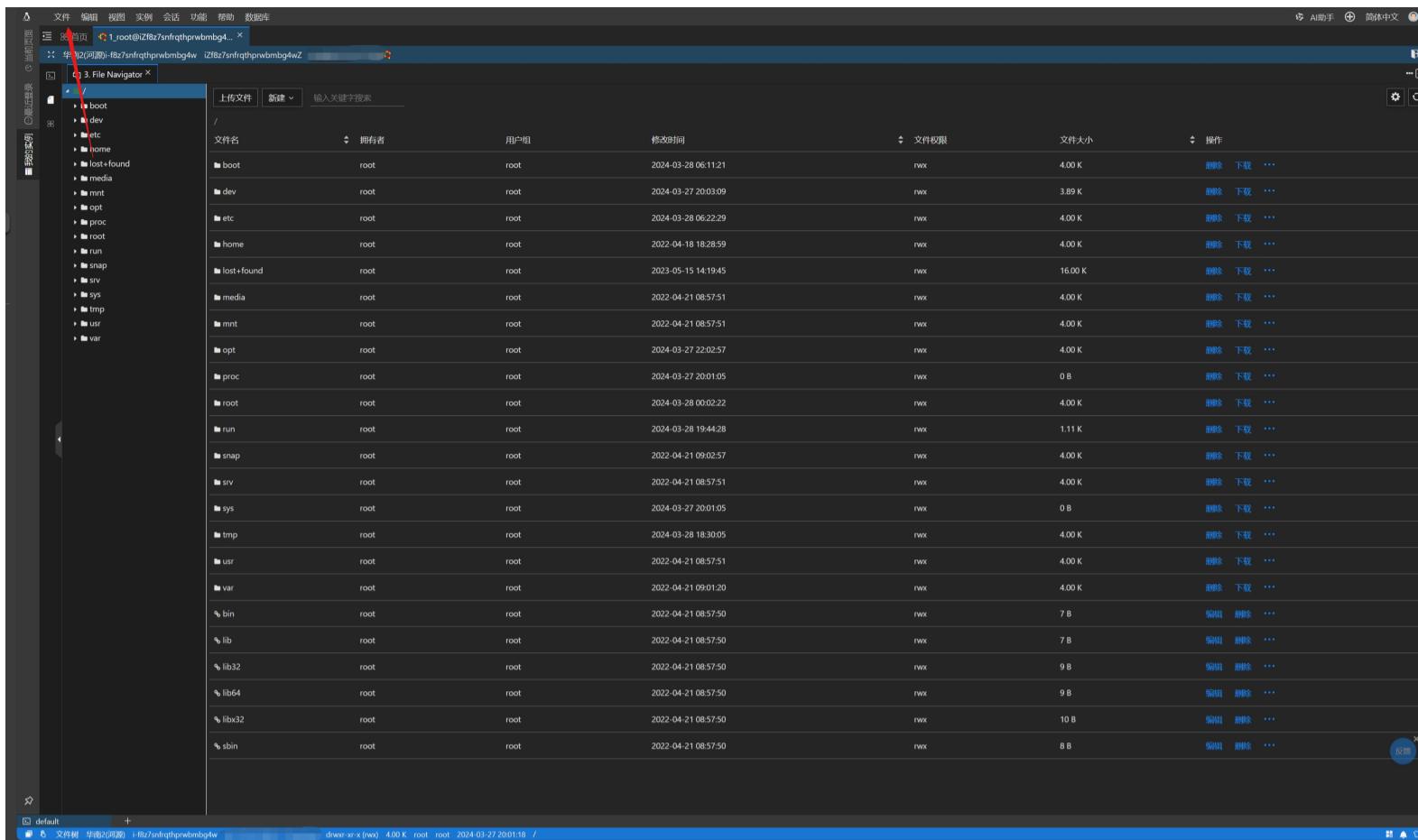
## 2.5 本地Jar测试(可选)

找到刚刚生成的jar文件，在自己电脑的终端里输入下面的命令，再使用浏览器重复刚刚的测试(这是避免在少数情况下打包策略导致IDEA中能运行，而无依赖的环境下无法运行，如果出现这个情况，修改maven配置即可)

```
如果jar文件就在当前目录
java -jar app.jar
或者手动指定目录
java -jar C:\apps\myapp.jar
```

## 3. 上传Jar文件(可选)

打开阿里云控制台，把生成的jar文件上传到tmp文件夹。后续如果用Portainer构建镜像，这步可以跳过

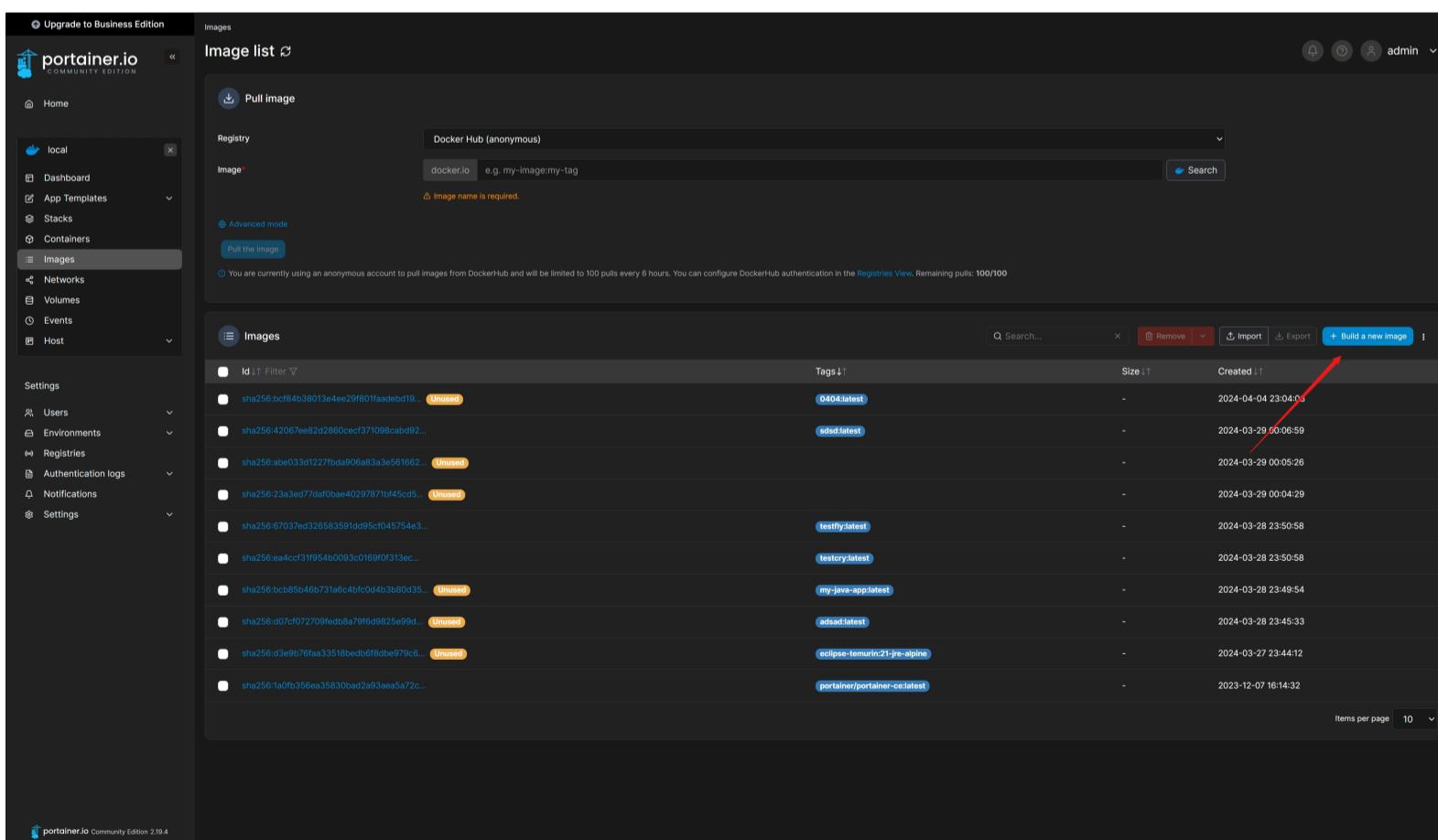


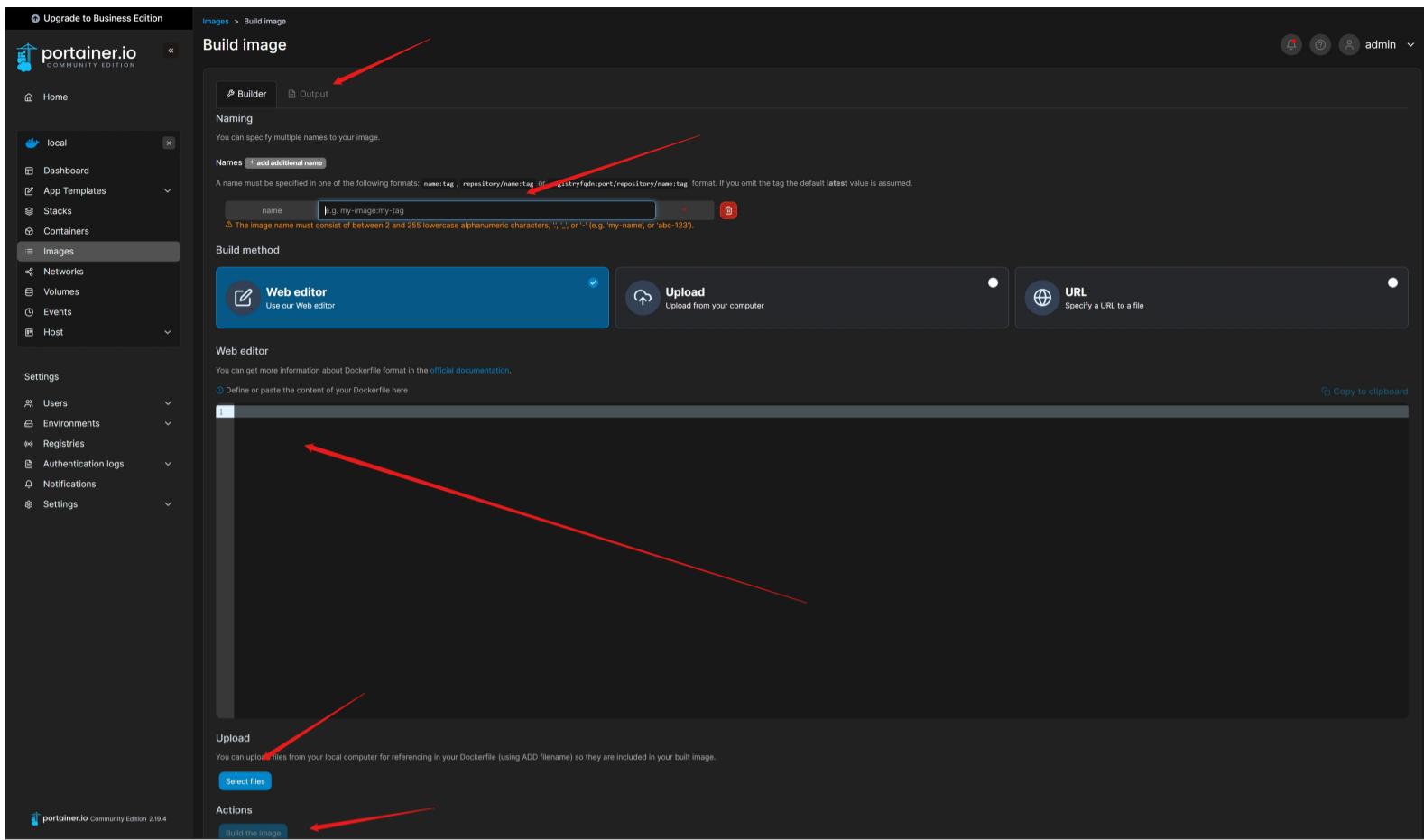
## 4. 分析组件

前面提到了Docker容器是一个独立的环境，因此容器内部应该要有所有的依赖组件，一般SpringBoot项目只会用到Java环境与MySQL，MySQL已经在上面的步骤中安装在系统里了(不在Docker里)，所以容器内只需要有Java(JRE)就可以了，因为我们组还用了Arthas，Kafka所以多几个组件，记住自己组要用到的组件，构建镜像的时候要用

## 5. 使用Portainer构建Docker镜像

**DockerFile**告诉Docker构建这个镜像的具体流程，有了DockerFile我们就可以构建镜像，正常情况下应该使用命令行来手写DockerFile。但是为了方便这里用Portainer。首先打开Portainer控制面板，点击images-Build a new image





1. 设置镜像的**名字**(以及版本号), 例如GroupXX-1.1, 这个名字要记住, 等会要用
2. 点击Select Files, 上传你的**jar文件**, 如果还有其他文件, 一并上传(如果服务器带宽低有可能失败, 多试几次)
3. 将模板粘贴到编辑器中(这里复制的东西就是DockerFile)
4. Build
5. Output里不显示异常就成功

以下是DockerFile的模板, Java环境根据自己的版本**四选一**即可, 如果还有其他组件自己添加

```
DockerFile Template

1. 在基础的Java环境下构建镜像, 注意以下四条指令只能选一个, 用哪个jdk开发的就用选哪个版本
第一个: jdk21 第二个: jdk17 第三个: jdk11 第四个: jdk8 一般是11或17
FROM eclipse-temurin:21-jre-alpine
FROM openjdk:17-oracle
FROM openjdk:11-jdk-slim
FROM openjdk:8-jdk-alpine

2. 将jar复制到容器中 这里的第一个app.jar替换为你实际上传的文件名, 第二个/opt/app.jar不改
ADD app.jar /opt/app.jar

2.5(可选) 如果还有其他要添加的文件取消下面的注释
ADD fileName /opt/file

3. 设置工作目录, 不用修改
WORKDIR /opt

4. 设置容器启动时执行的命令, 相当与容器一启动就会执行 java -jar /opt/myApp.jar, 不用修改
ENTRYPOINT ["java", "-jar", "myApp.jar"]

这个DockerFile相当于 先配置了Java环境, 再添加了其他文件(可选) 然后在容器启动时使用java指令立即启动自己的App
```

## 6. ( ), 启动!

登录Portainer控制面板，点击Containers-Add container，Name里填想设置的容器名称，Image里是你刚构建的镜像名字，如果不知道镜像叫什么，就在Portainer-images里找刚刚构建的镜像的名称

The screenshot shows the Portainer Community Edition 2.19.4 interface. On the left, there's a sidebar with options like Home, Dashboard, App Templates, Stacks, Containers (which is selected and highlighted with a red arrow), Images, Networks, Volumes, Events, and Host. The main area is titled 'Container list' and shows a table with one row: portainer1 (running, portainer/portainer-celestest, created 2024-03-28 23:36:08, IP 172.17.0.2, ports 8000:8001, 9000:9000). At the top right, there are buttons for Start, Stop, Kill, Restart, Pause, Resume, Remove, and Add container.

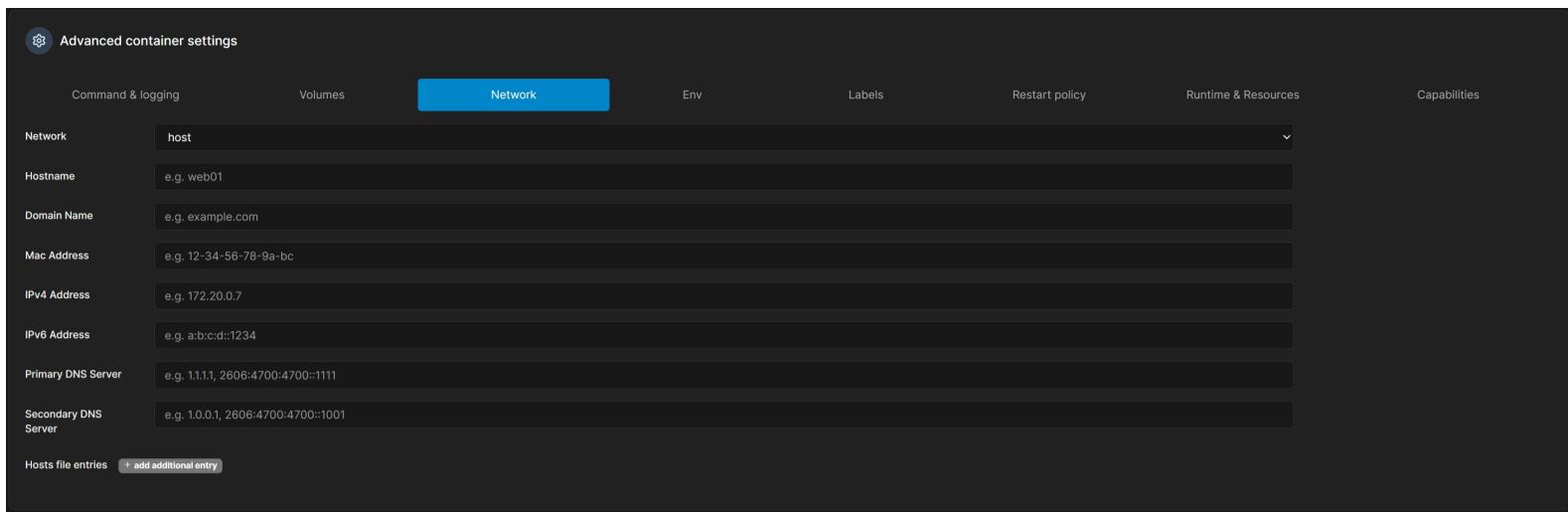
This screenshot shows the 'Add container' configuration dialog. It includes fields for Name (e.g. myContainer), Image configuration (Registry: Docker Hub (anonymous), Image: docker.io e.g. my-image:my-tag, with a note 'Image name is required.'), Advanced mode (disabled), Always pull the image (disabled), Webhooks (Create a container webhook), Network ports configuration (Publish all exposed network ports to random host ports), Manual network port publishing (with a '+ publish a new network port' button), Access control (Enable access control), and Actions (Auto remove, Deploy the container). Two buttons at the bottom are 'Administrators' (selected) and 'Restricted'. A green arrow points from the 'Name' field to the 'Image' field, and another green arrow points to the 'Deploy the container' button.

在Deploy之前先选择网络模式(Host与Bridge)**二选一**，如果用Bridge就在Manual network port publishing里手动填写映射关系，一般绑定至主机的80端口

```
host:80 -> container:8080
代表容器的8080端口绑定至服务器80端口，在外部使用服务器ip:80就能访问容器8080(http://服务器ip)
```

This screenshot shows the 'Manual network port publishing' dialog. It has fields for host (e.g. 80, 80-88, ip:80 or ip:80-88 (optional)) and container (e.g. 80 or 80-88). There are buttons for TCP, UDP, and a delete icon. A green arrow points to the 'host' field.

如果不知道容器会用到哪个端口，就在下面的设置里选择**Host**模式



设置完网络模式后点Deploy，如果成功则在Containers内可以看到刚刚启动的容器和网络信息。

## 7. 访问&测试

如果刚刚启动容器时是Host模式，在自己电脑的浏览器用‘服务器ip:app使用的端口’访问；

如果是Bridge模式，且app使用的端口为b，该端口映射到了主机的a，则用服务器ip:映射的端口a访问；

浏览器应该能正常显示你们的页面，按上学期软件工程和CPT202老师PPT上要求的流程来测试

## 8. 容器异常退出

在正常情况下，我们刚刚构建的镜像应该一直保持运行状态(因为APP不会自己退出)，如果发现Container状态为Exited，则通过Portainer-Containers-找到该容器并分析日志，例如：

```

:: Spring Boot :: (v3.2.3)
2024-04-08T06:46:15.810Z INFO 1 --- [main] w.cpt202.backend.BackendApplication : Starting BackendApplication v0.0.1-SNAPSHOT using Java 21.0.2 with PID 1 (/opt/myApp.jar started by root in /opt)
2024-04-08T06:46:15.821Z INFO 1 --- [main] w.cpt202.backend.BackendApplication : No active profile set, falling back to 1 default profile: 'default'
2024-04-08T06:46:19.993Z WARN 1 --- [main] trationDelegate$BeanPostProcessorChecker : Bean 'org.springframework.ws.config.annotation.DelegatingWsConfiguration' of type [org.springframework.ws.config.annotation.DelegatingWsConfiguration$$SpringCGLIB$0] is not eligible for getBean processing: not eligible for auto-proxying. The currently created BeanPostProcessor [annotationActionEndpointMapping] is declared through a non-static factory method on that class; consider declaring it as static instead.
2024-04-08T06:46:20.097Z INFO 1 --- [main] .w.s.a.AnnotationActionEndpointMapping : Supporting [WS-Addressing August 2004, WS-Addressing 1.0]
2024-04-08T06:46:20.097Z INFO 1 --- [main] o.s.b.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8000 (http)
2024-04-08T06:46:21.072Z INFO 1 --- [main] m.a.s.ContainerStandardService : Starting service...
2024-04-08T06:46:21.072Z INFO 1 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engines: [Apache Tomcat/10.1.19]
2024-04-08T06:46:21.190Z INFO 1 --- [main] o.a.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded webapplicationcontext
2024-04-08T06:46:21.205Z INFO 1 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 5034 ms
2024-04-08T06:46:23.516Z WARN 1 --- [main] ConfigServletWebServerApplicationContext : Exception encountered during context initialization - cancelling refresh attempt: org.springframework.context.ApplicationContextException: Failed to start bean 'webServerStartStop'
2024-04-08T06:46:24.072Z INFO 1 --- [main] .s.b.a.l.ConditionEvaluationReporter : Error starting ApplicationContext. To display the condition evaluation report re-run your application with 'debug' enabled.
2024-04-08T06:46:24.164Z ERROR 1 --- [main] o.s.b.d.LoggingFailureAnalysisReporter :

APPLICATION FAILED TO START

Description:
Web server failed to start. Port 8000 was already in use.
Action:
Identify and stop the process that's listening on port 8000 or configure this application to listen on another port.

```

Log显示SpringBoot启动时8000端口被占用导致启动失败，那么去系统中停止其他使用8000端口的进程(或改为Bridge模式用另一个端口)即可。以下是其他常见的问题：

|              | 可能原因         | 排查方法                                       |
|--------------|--------------|--------------------------------------------|
| 启动失败         | DockerFile有误 | 检查DockerFile中java版本是否正确，其余指令是否正确           |
| 启动后一段时间内Exit | Jar文件问题      | 检查自己的电脑运行java -jar app.jar是否能成功            |
| 启动后一段时间内Exit | 端口冲突         | 停止其他使用该端口的进程或改为Bridge模式                    |
| 启动后一段时间内Exit | 缺少组件         | 默认情况下只会用到Java 如果你们还用到了其他组件需要在DockerFile内声明 |
| 启动后一段时间内Exit | OOM          | OOM一般都是Jar文件(App自身)的问题                     |
| 启动成功但是无法访问   | 网络问题         | 检查访问时端口号，协议(http/https)是否正确                |

## F. Docker生命周期

### 1. 下线/重启

使用Portainer内置的功能即可

| Name       | State   | Quick Actions                                     | Stack | Image                         | Created             | IP Address | Published Ports      | Ownership      |
|------------|---------|---------------------------------------------------|-------|-------------------------------|---------------------|------------|----------------------|----------------|
| portainer1 | running | Start, Stop, Kill, Restart, Pause, Resume, Remove | -     | portainer/portainer-ce-litest | 2024-03-28 23:36:08 | 172.17.0.2 | 8000:8001, 9000:9000 | administrators |
| Test240408 | running | Start, Stop, Kill, Restart, Pause, Resume, Remove | -     | 0404:latest                   | 2024-04-08 14:38:36 | 172.17.0.3 | 5555:8000, 5556:8080 | administrators |

### 2. 升级

如果之前已经上线了一个App，现在测试后改动了代码，准备在服务器上也升级到更新的版本。那么先停止以前的容器，然后准备好相关文件(jar)，回到上一个章节的**使用Portainer构建Docker镜像**部分使用DockerFile重新构建即可，这次只需要上传新的jar并更新版本号。然后再启动新的容器

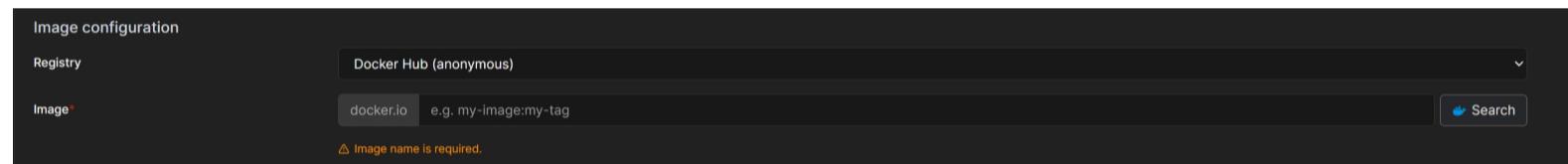
例如原来的镜像名字是  
GroupXX-1.0  
那么改为  
GroupXX-2.0  
(-后面的数字更大即可)

### 3. 数据持久化(可选)

在之前的方法中，容器的所有内容都与宿主机是隔离的，容器退出后，其中的数据也会一并删除，如果想保留数据，则在构建镜像时选择使用**容器数据卷**。容器退出后数据会保留在宿主机内

### 4. 快速部署(可选)

如果在构建镜像后把镜像上传至仓库，以后在所有装了Docker的电脑内使用仓库名+镜像名即可快速部署（大陆地区可能无法访问，自行解决网络问题）



## G. 不使用Docker部署

如果决定不使用Docker部署，那么参考以下流程：

- [1. 登录服务器](#)
- [2. 放行端口](#)
- [3. 安装Mysql与其他组件\(例如Java\)](#)
- [4. 调试代码](#)
- [4. 打包为Jar](#)
- [4. 本地测试](#)
- [5. 上传文件](#)
- [6 使用java -jar XXX.jar 启动App](#)

---

如有错误敬请指正

PDF Available on: [CPT202 Docker Tutorial, XJTLU AY2024 \(github.com\)](#)

Html Available on: [CPT202 Docker Tutorial-Online Version](#)

Automated deployment scripts Available on: [Automated deployment-Linux](#)

License: GPLv3

Tianxin Zhao