Docker容器化部署微服务

官方文档：https://docs.docker.com/get-started/overview/

中文文档：https://dockerdocs.cn/get-started/overview/index.html

Docker架构：

图片版权归属

Docker主要概念：

Docker引擎(Docker Engine)：是一个服务端-客户端结构的应用，主要有这些部分：Docker守护进程、Docker Engine API、Docker客户端。

Docker注册中心(Docker registry)：用于存储Docker的镜像。Docker Hub
是一个公共的注册中心，任何人都可以使用，默认配置下，Docker将会在这里寻找镜像。另外，用户可以自行构建私有注册中心。Docker Datacenter (DDC)的用户，可以直接使用 Docker Trusted Registry (DTR)。

容器(Containers)：镜像的可运行的实例。容器可通过API或CLI（命令行）进行操控。

镜像(Images)：一个只读模板，用于指示创建容器。镜像分层(layers)构建的，而定义这些层次的文件叫Dockerfile。

服务(Services)：允许用户跨越不同的Docker守护进程（Docker
daemons）的情况下增加容器，并将这些容器分为管理者（managers）和工作者（workers），让他们为swarm共同工作。

Dockerfile：包含着用户想要如何构建镜像的所有命令的文本，用于构建镜像。

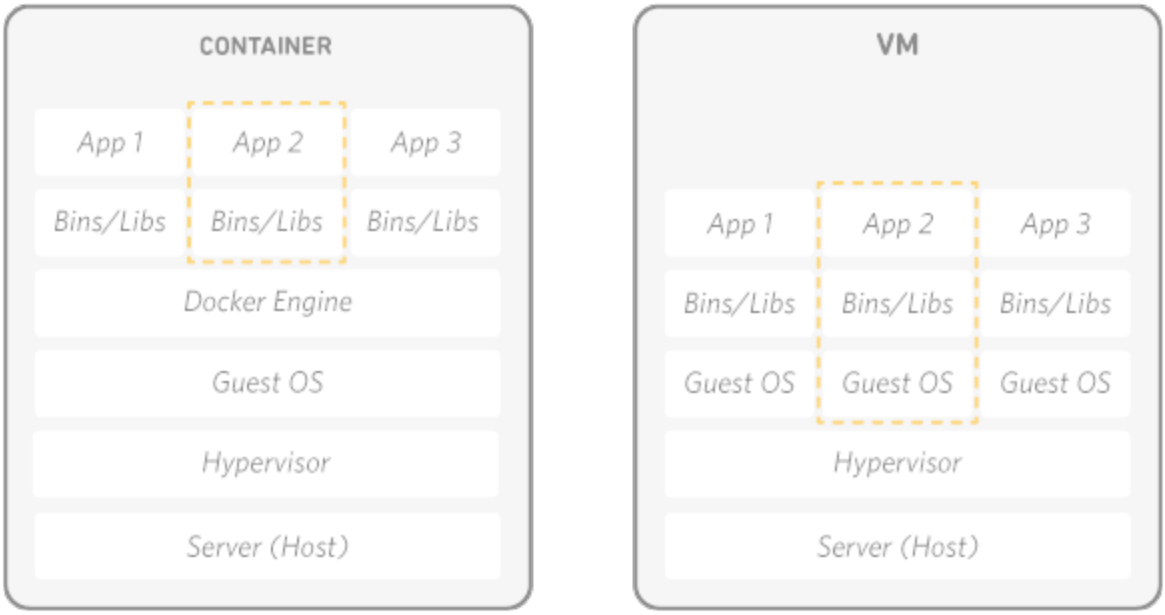
Docker Compose：定义了服务（service）、网络、卷（volume）；是一款旨在帮助定义和共享多容器应用程序的工具。使用Compose，我们可以创建一个YAML文件来定义服务，编排微服务，并且可以使用一个命令启动所有服务。

Docker与虚拟机的区别：

Docker容器与虚拟机类似，但二者在原理上不同。容器是将操作系统层虚拟化，虚拟机则是虚拟化硬件，因此容器更具有便携性、高效地利用服务器。

虚拟机：是一个完整的操作系统，需要占用大量的磁盘、内存和 CPU 资源，一台机器只能开启几十个的虚拟机。

Docker：只是宿主机的一个进程，只需要将应用以及相关的组件打包，在运行时占用很少的资源，一台机器可以开启成千上万个 Docker。



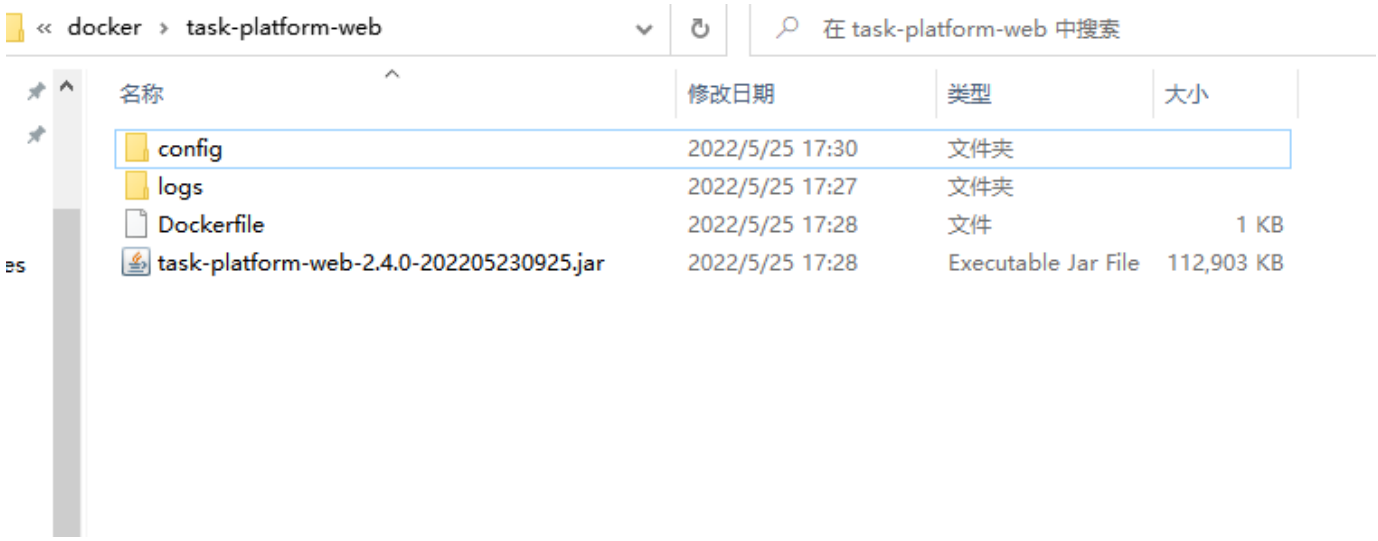
图片版权归属

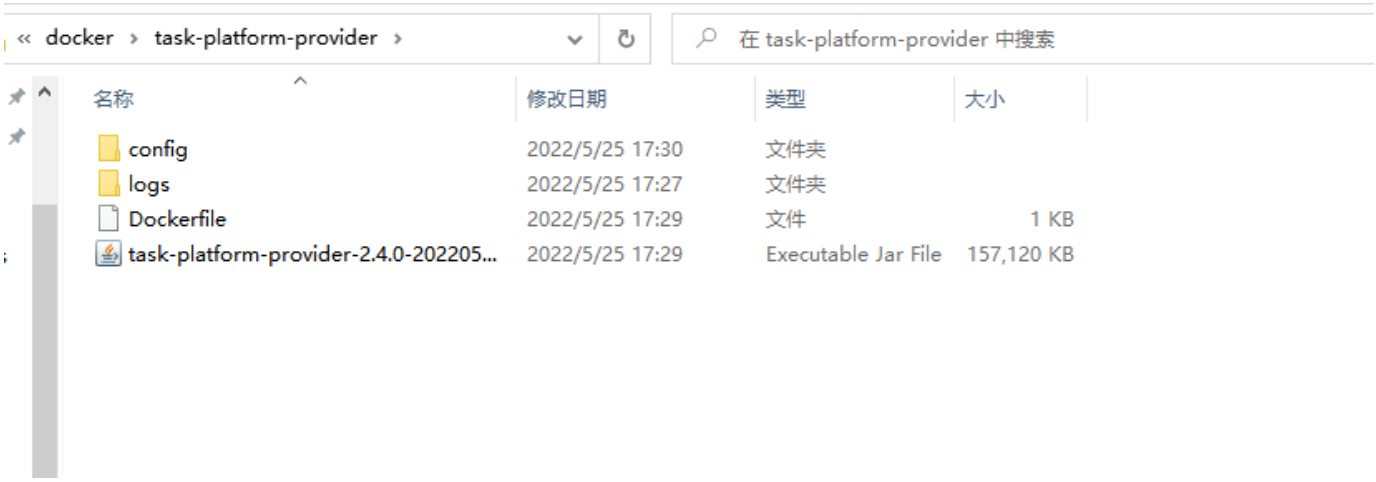
以下以任务平台TP-V2.4.0为基础示例，探索如何docker容器化部署微服务。

本地测试

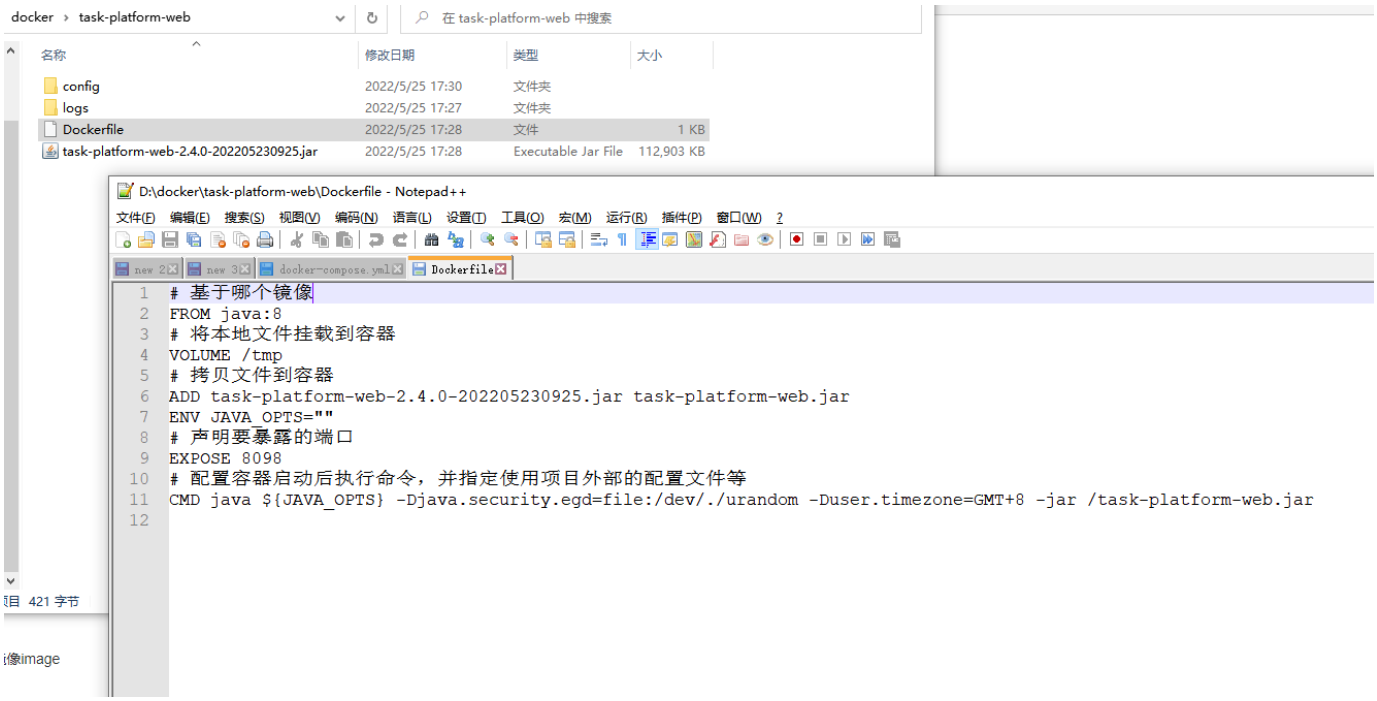
一、下载docker、docker-compose（跳过）

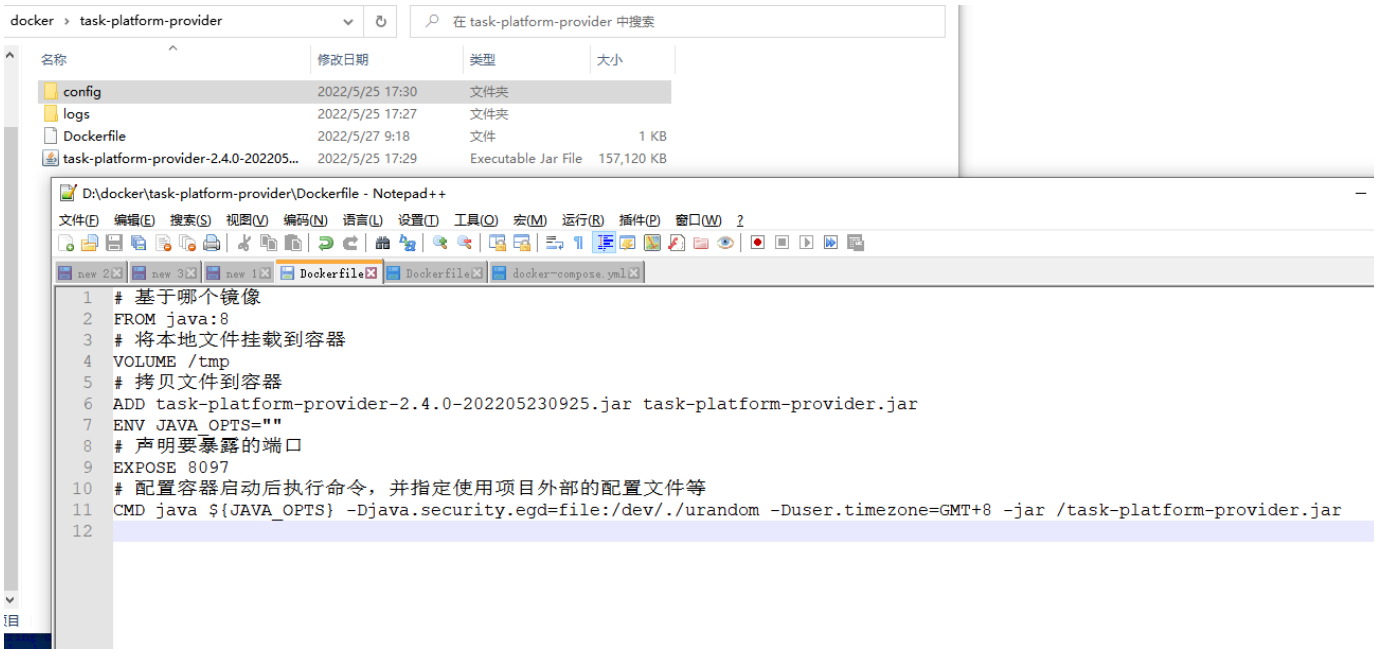
二、新建docker目录，目录下新建task-platform-web、task-platform-provider目录，存放各自的jar包、配置文件、Dockerfile文件



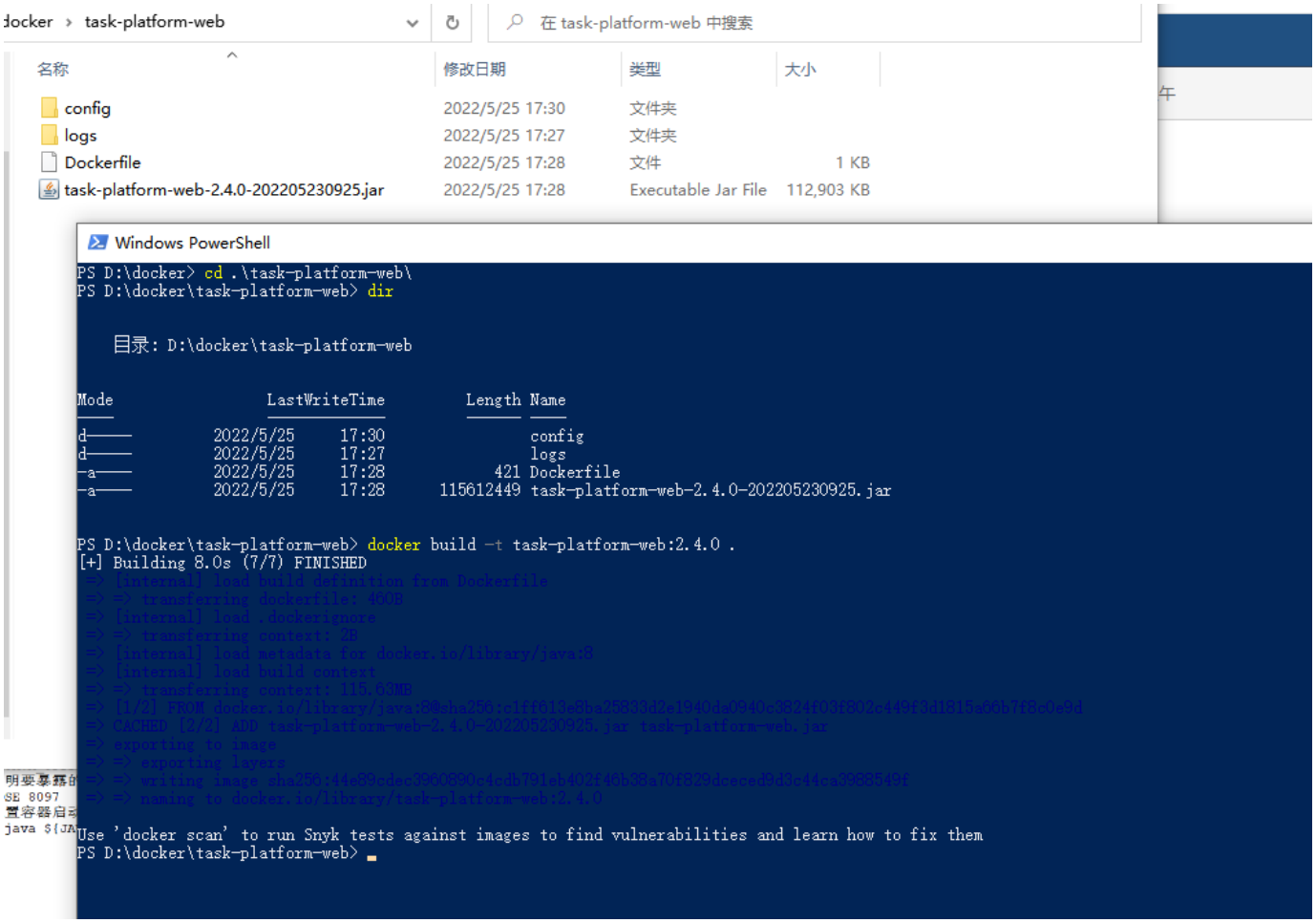


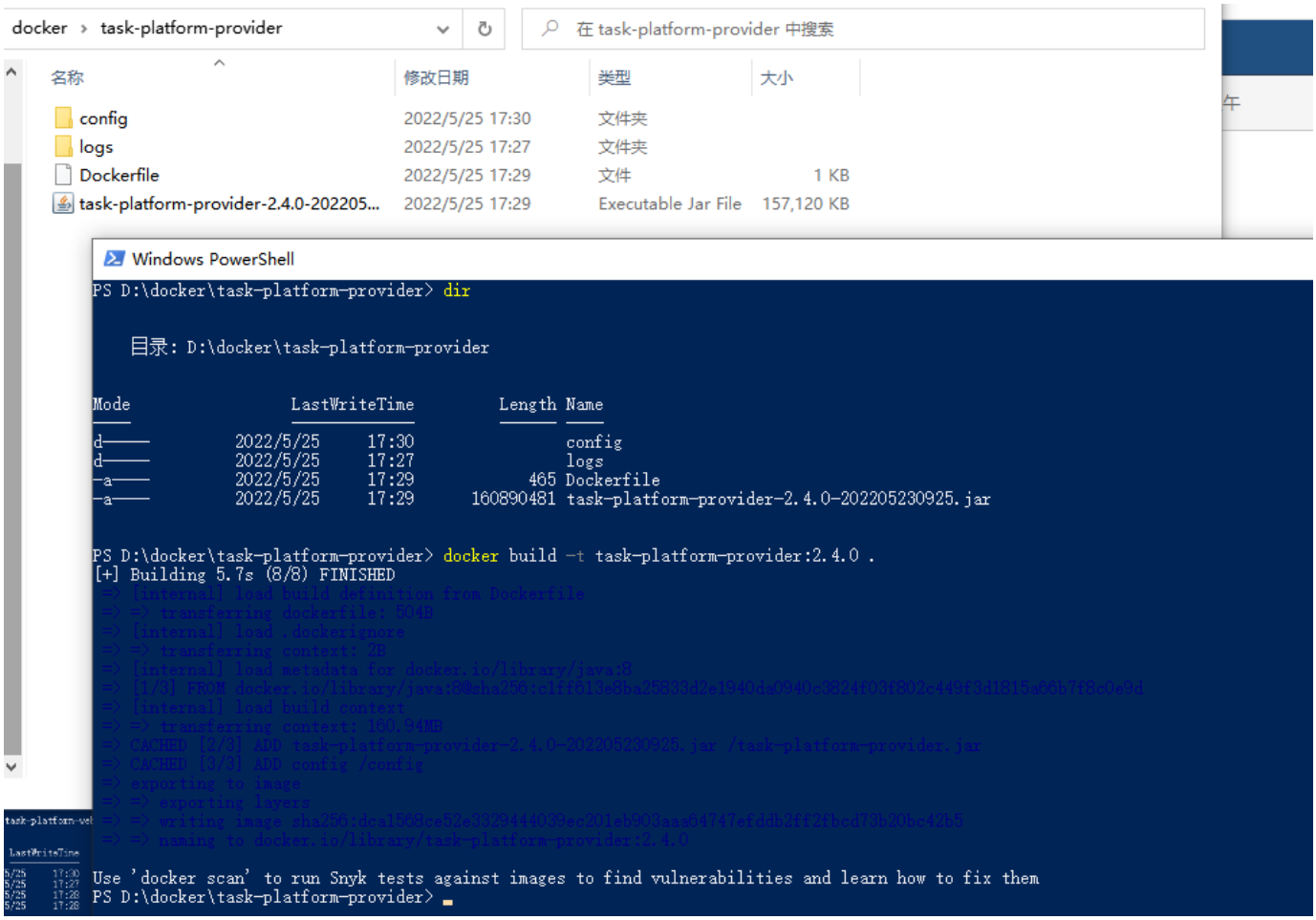
三、编写Dockerfile文件，用于构建镜像image

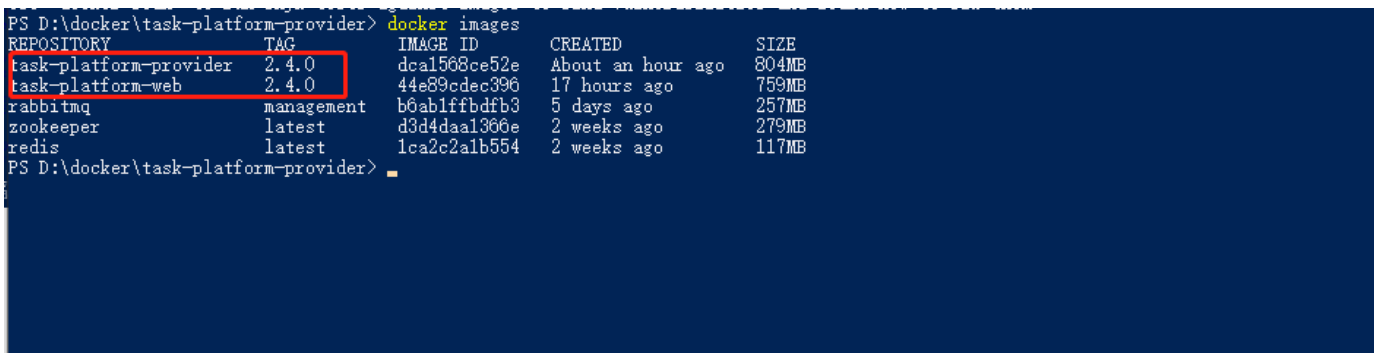


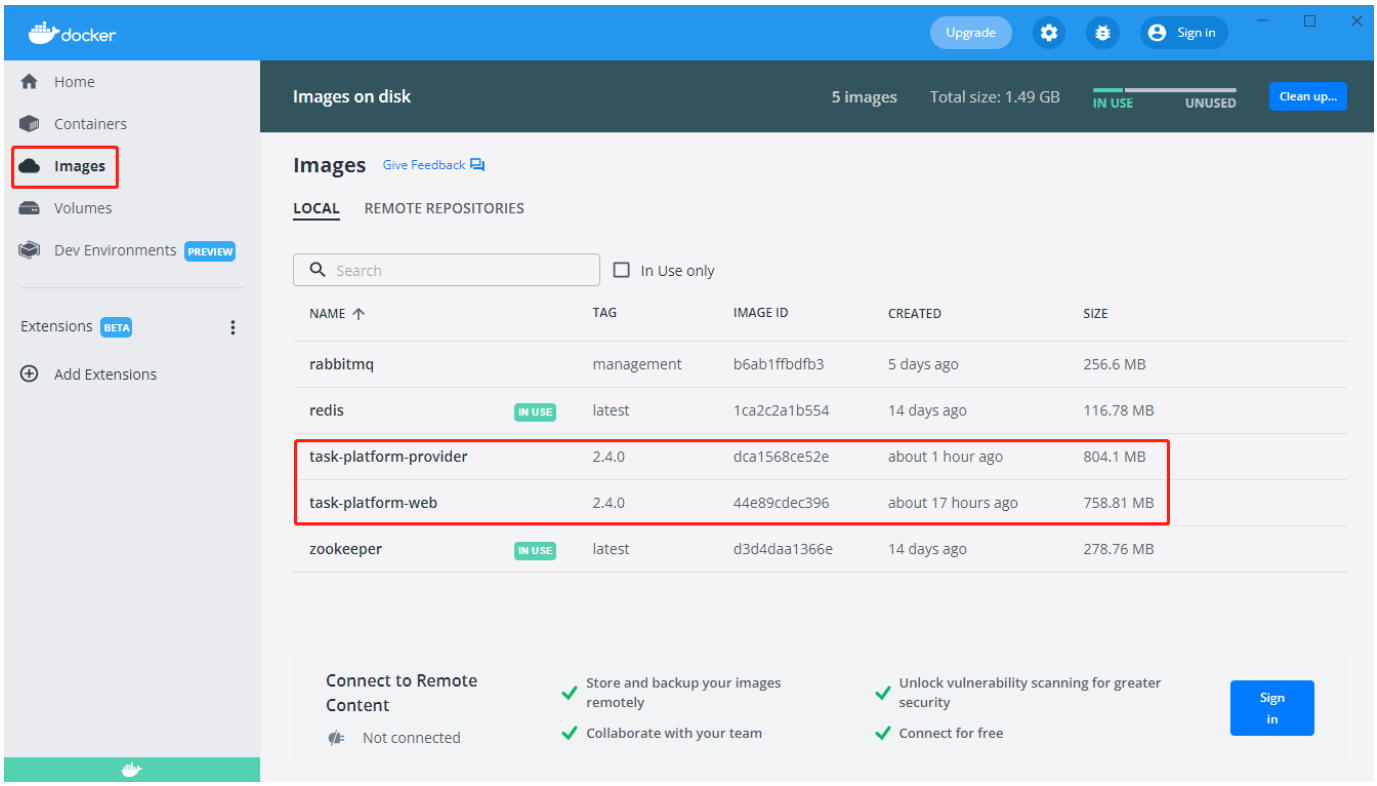


四、Dockfile文件编写完成后、进入对应目录构建镜像（docker build -t 镜像名:版本tag .）

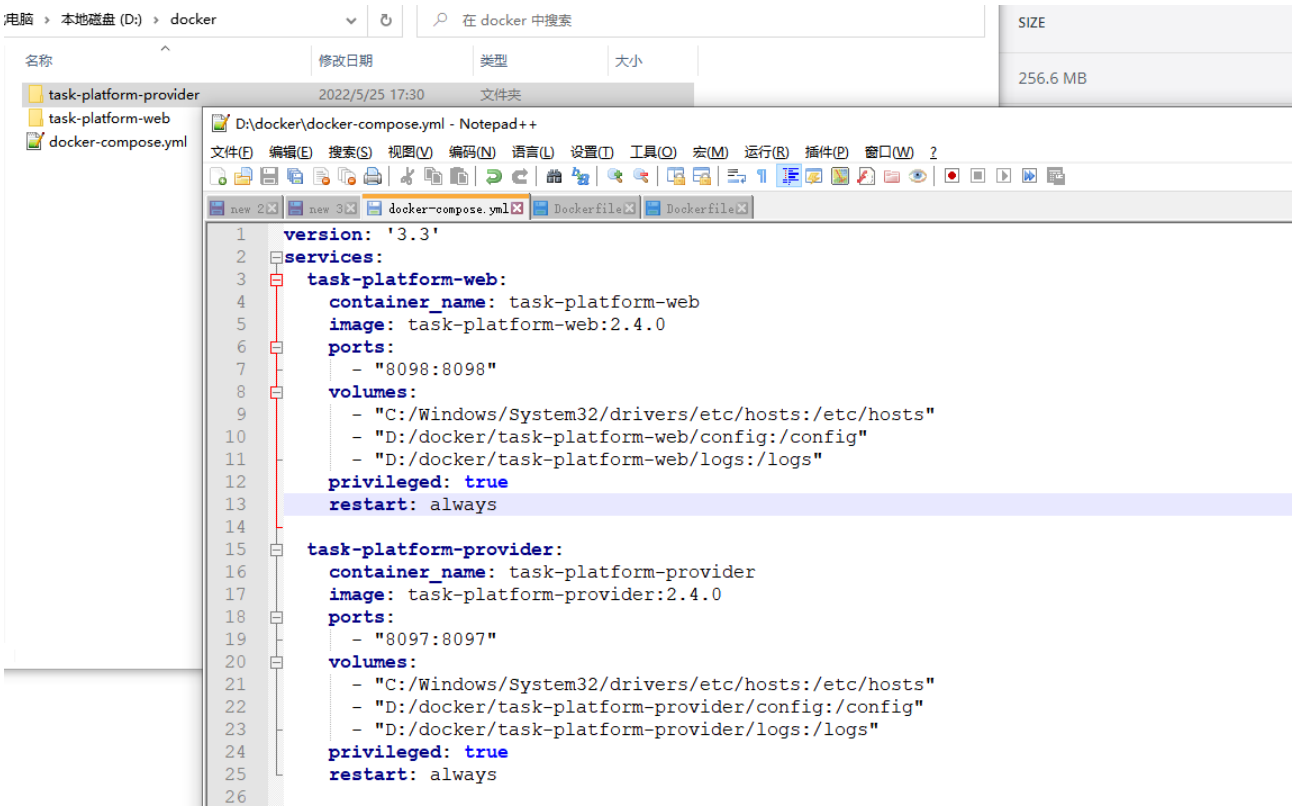


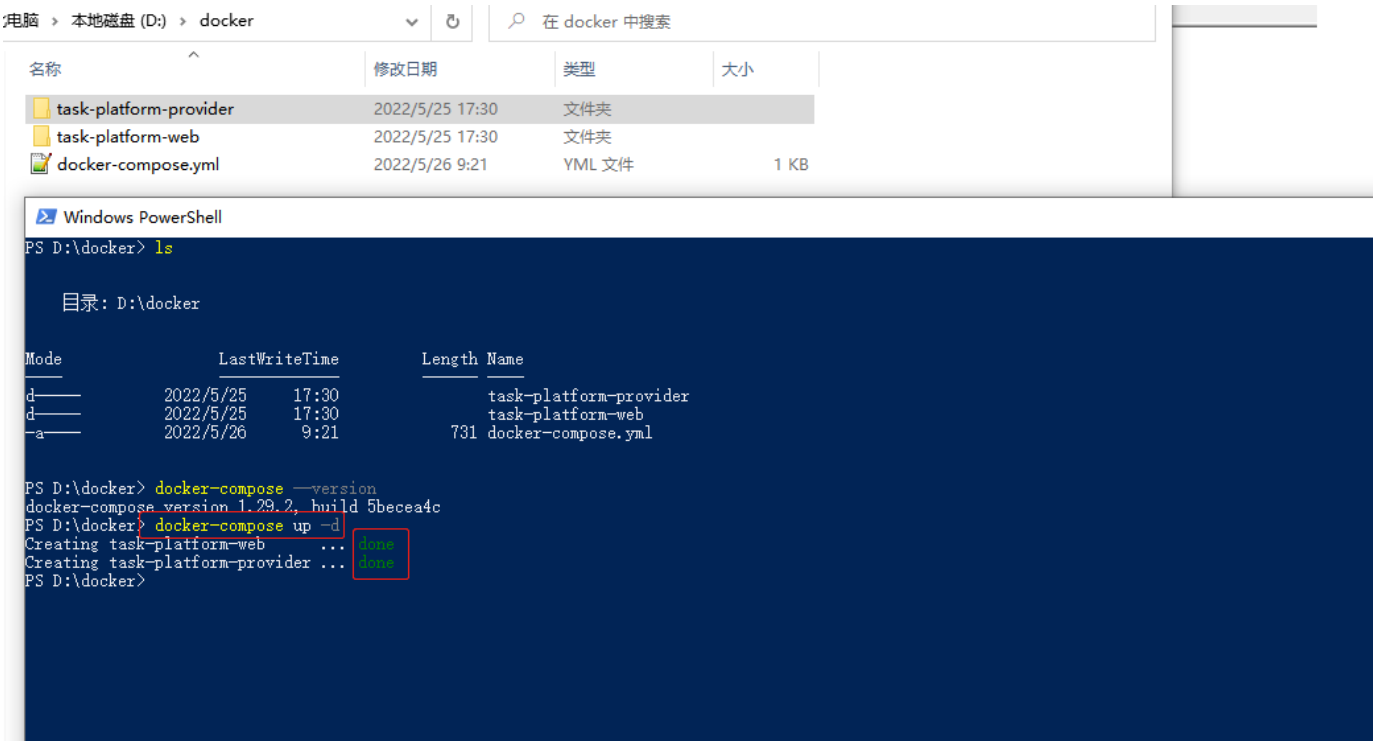

docker › task-platform-provider
在 task-platform-provider 中搜索
名称
修改日期
类型
大小
config 2022/5/25 17:30 文件夹
logs 2022/5/25 17:27 文件夹
Dockerfile 2022/5/25 17:29 文件 1 KB
task-platform-provider-2.4.0-202205... 2022/5/25 17:29 Executable Jar File 157,120 KB
Windows PowerShell
PS D:\docker\task-platform-provider> dir
目录: D:\docker\task-platform-provider
Mode LastWriteTime Length Name
d----- 2022/5/25 17:30 config
d----- 2022/5/25 17:27 logs
-a---- 2022/5/25 17:29 465 Dockerfile
-a---- 2022/5/25 17:29 160890481 task-platform-provider-2.4.0-202205230925.jar
PS D:\docker\task-platform-provider> docker build -t task-platform-provider:2.4.0 .
[+] Building 5.7s (8/8) FINISHED
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
PS D:\docker\task-platform-provider>

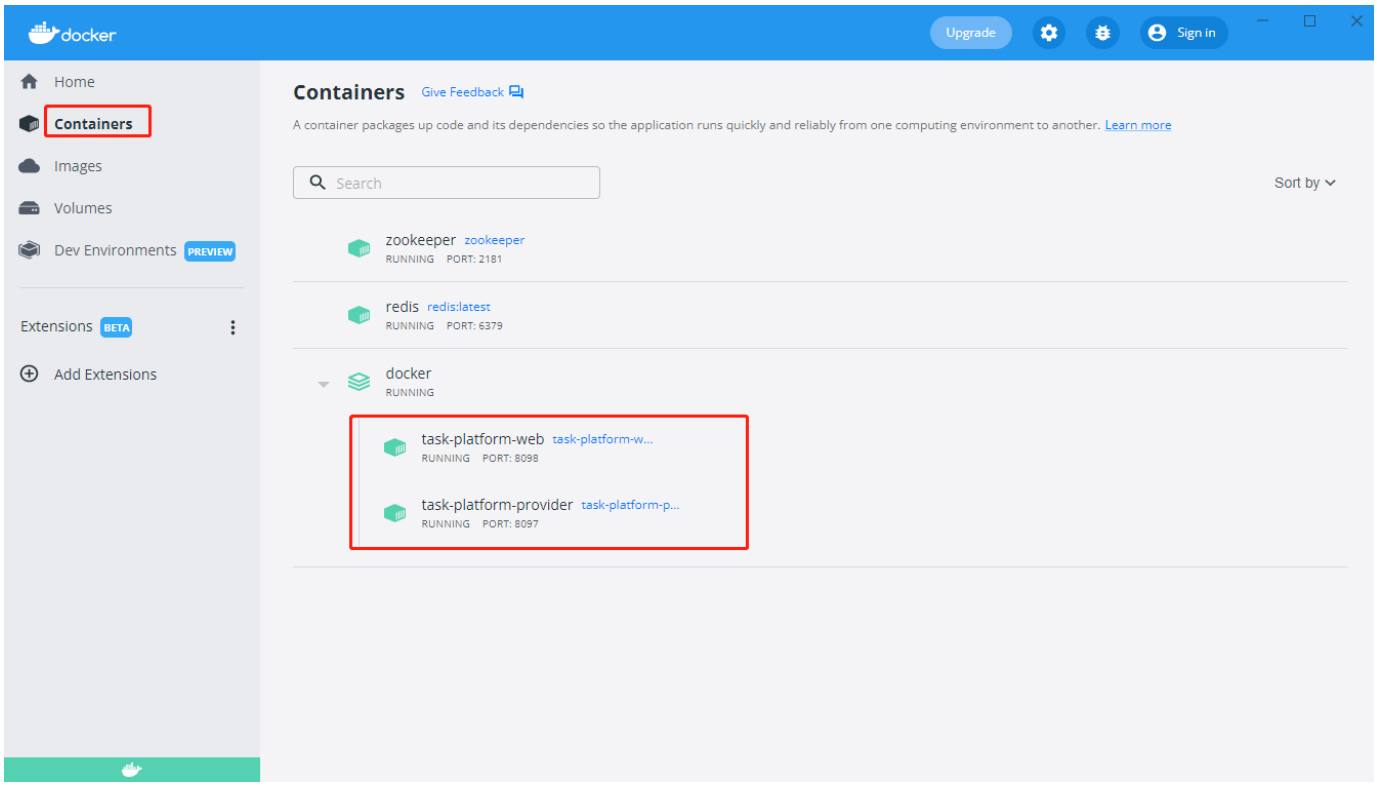

PS D:\docker\task-platform-provider> docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
task-platform-provider 2.4.0 dca1568ce52e About an hour ago 804MB
task-platform-web 2.4.0 44e89cdec396 17 hours ago 759MB
rabbitmq management b0ab1ffbdfb3 5 days ago 257MB
zookeeper latest d3d4daa1366e 2 weeks ago 279MB
redis latest 1ca2c2a1b554 2 weeks ago 117MB
PS D:\docker\task-platform-provider>



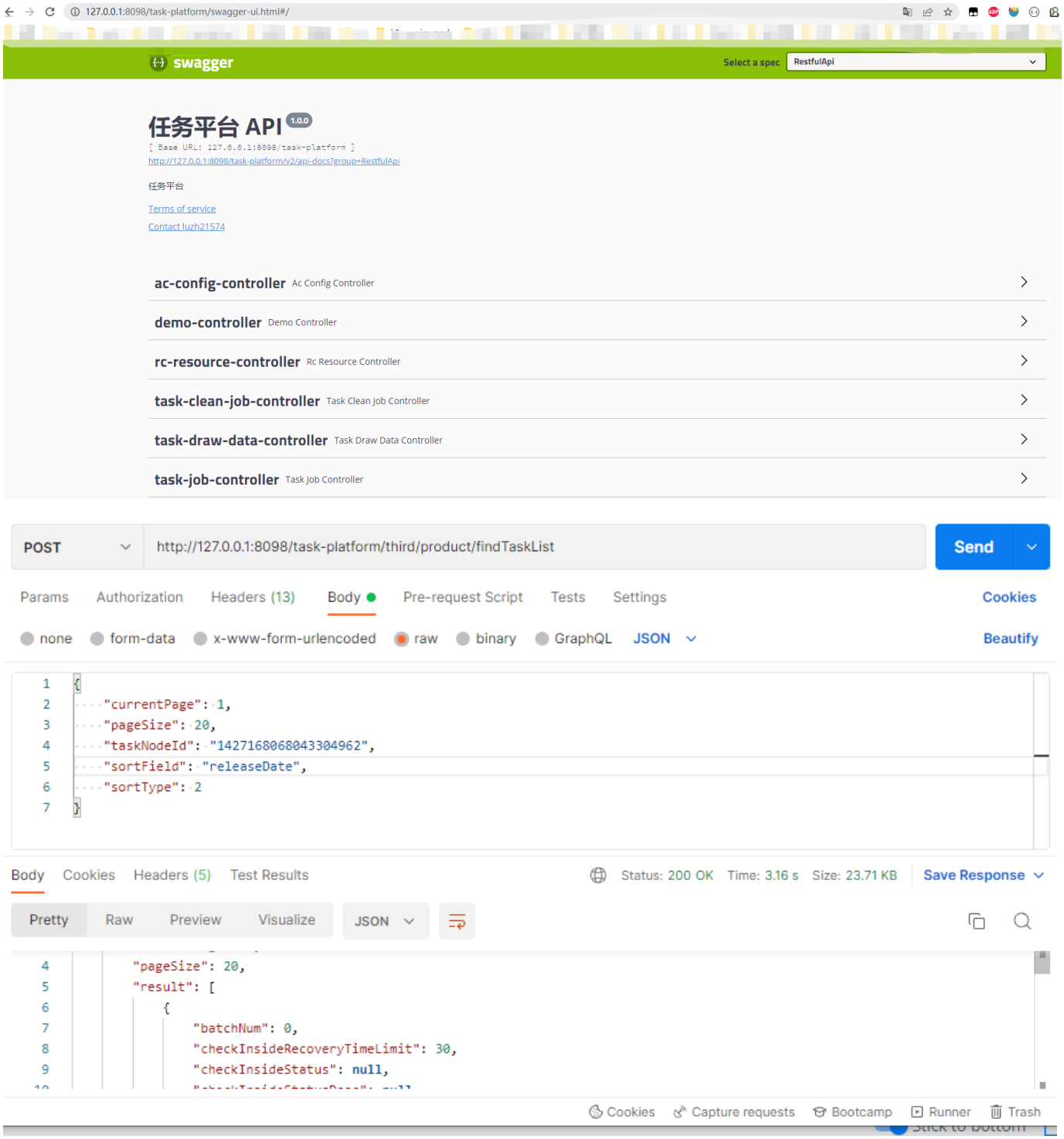
五、编写docker-compose.yml文件，编排微服务，并用docker-compse创建容器启动所有服务（docker-compose up -d）




电脑 › 本地磁盘 (D:) › docker
在 docker 中搜索
名称
修改日期
类型
大小
task-platform-provider 2022/5/25 17:30 文件夹
task-platform-web 2022/5/25 17:30 文件夹
docker-compose.yml 2022/5/26 9:21 YML 文件 1 KB
Windows PowerShell
PS D:\docker> ls
目录: D:\docker
Mode LastWriteTime Length Name
d----- 2022/5/25 17:30 task-platform-provider
d----- 2022/5/25 17:30 task-platform-web
-a---- 2022/5/26 9:21 731 docker-compose.yml
PS D:\docker> docker-compose --version
docker-compose version 1.29.2, build 5becea4c
PS D:\docker> docker-compose up -d
Creating task-platform-web ... done
Creating task-platform-provider ... done
PS D:\docker>

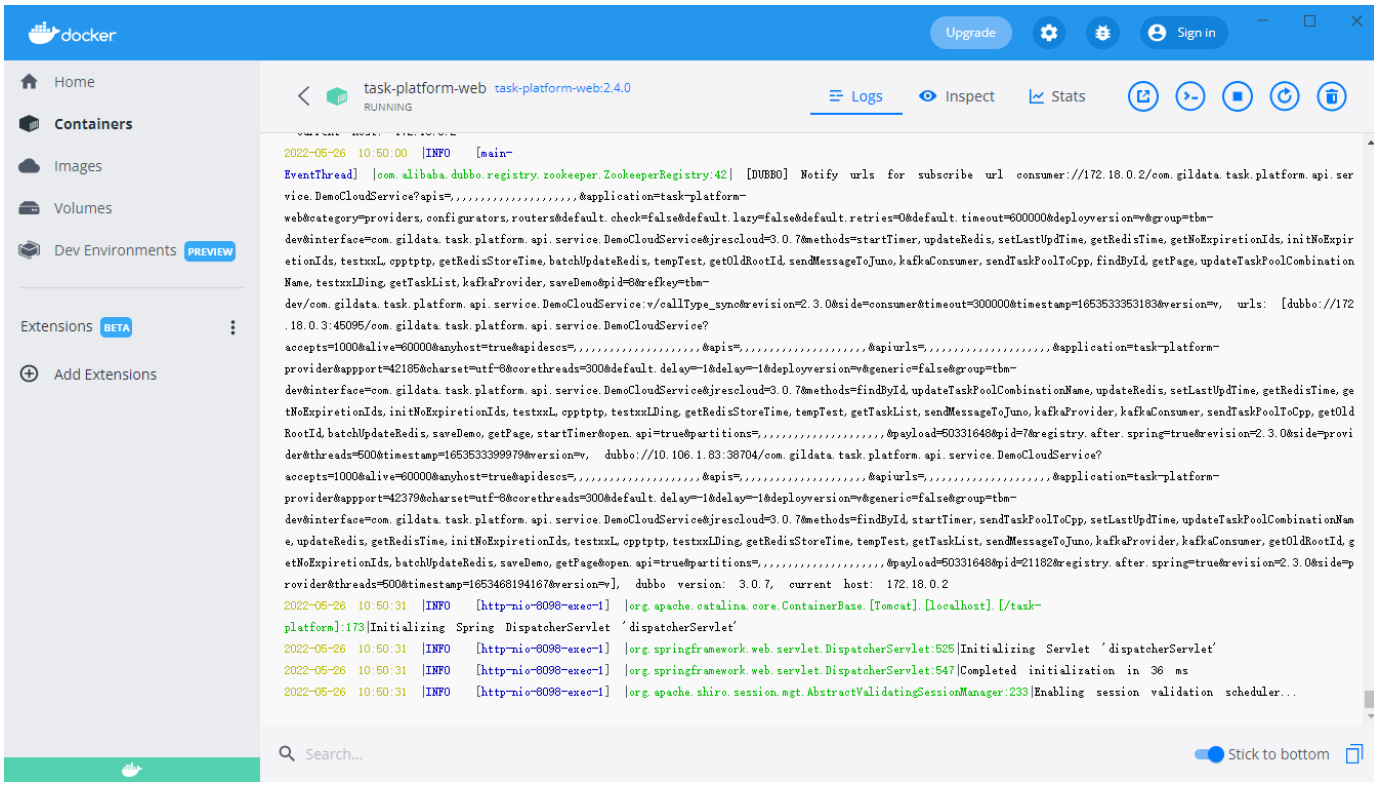

docker
Upgrade
Sign in
Home
Containers
Images
Volumes
Dev Environments PREVIEW
Extensions BETA
Add Extensions
Containers
Give Feedback
A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more
Search
Sort by
zookeeper zookeeper
RUNNING PORT: 2181
redis redis:latest
RUNNING PORT: 6379
docker
RUNNING
task-platform-web task-platform-w...
RUNNING PORT: 8098
task-platform-provider task-platform-p...
RUNNING PORT: 8097

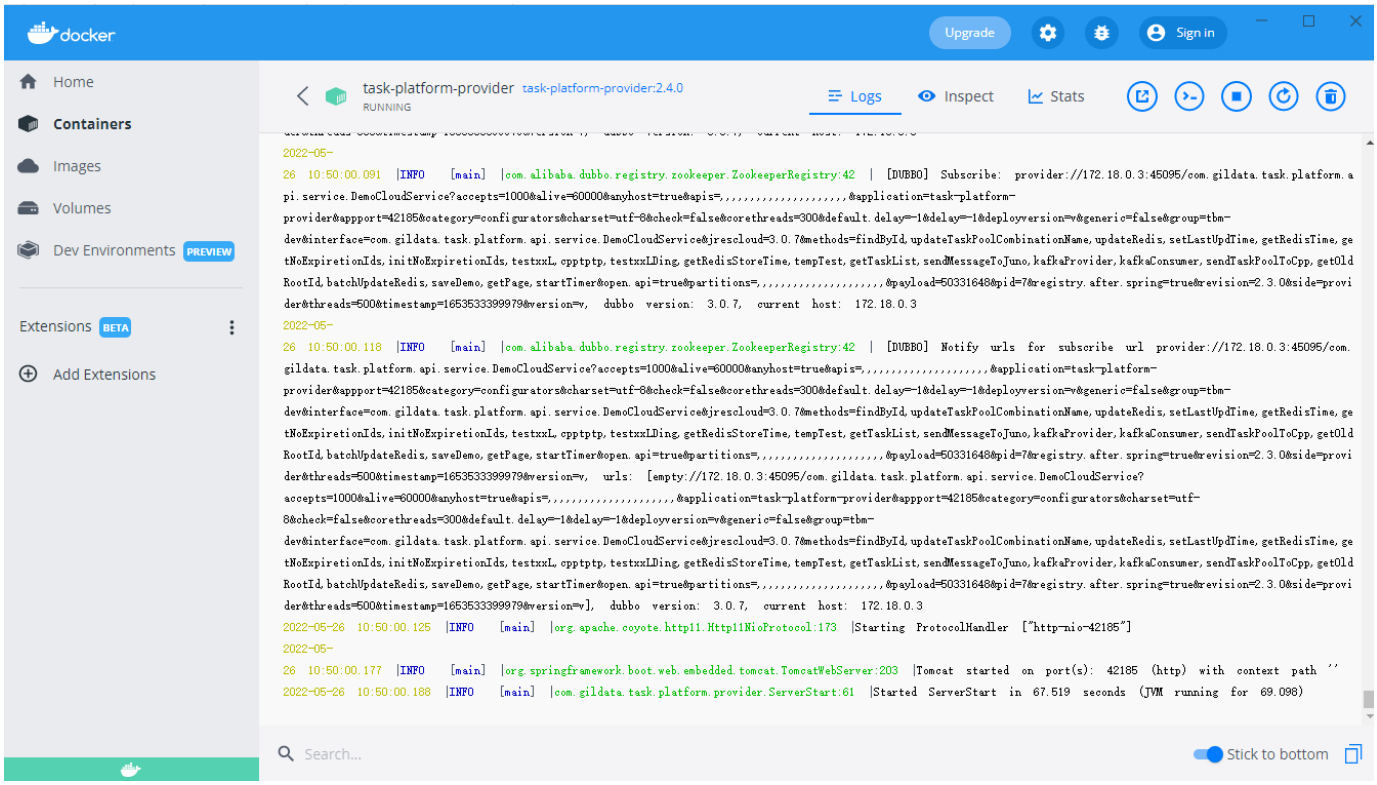
六、容器启动后，验证服务


127.0.0.1:8098/task-platform/swagger-ui.html#/
swagger
Select a spec
RestfulApi
任务平台 API
1.0.0
[Base URL: 127.0.0.1:8098/task-platform]
http://127.0.0.1:8098/task-platform/v2/api-docs?group=RestfulApi
任务平台
Terms of service
Contact luzh21574
ac-config-controller Ac Config Controller
demo-controller Demo Controller
rc-resource-controller Rc Resource Controller
task-clean-job-controller Task Clean Job Controller
task-draw-data-controller Task Draw Data Controller
task-job-controller Task Job Controller

POST
http://127.0.0.1:8098/task-platform/third/product/findTaskList
Send
Params
Authorization
Headers (13)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"currentPage": 1,
"pageSize": 20,
"taskNodeId": "1427168068043304962",
"sortField": "releaseDate",
"sortType": 2
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 3.16 s
Size: 23.71 KB
Save Response
Pretty
Raw
Preview
Visualize
JSON
"pageSize": 20,
"result": [
{
"batchNum": 0,
"checkInsideRecoveryTimeLimit": 30,
"checkInsideStatus": null,
Cookies
Capture requests
Bootcamp
Runner
Trash

可进入容器查看日志：





当然，实际上微服务容器化部署比这复杂很多，涉及网络连接等，这里只以最简单的流程做说明。

测试环境

结合IDEA Docker插件演示

一、测试环境安装Docker，安装docker-compose（跳过）

二、docker开放2375端口用于远程连接

1. vim /usr/lib/systemd/system/docker.service

2. ExecStart属性后添加参数-H tcp://0.0.0.0:2375

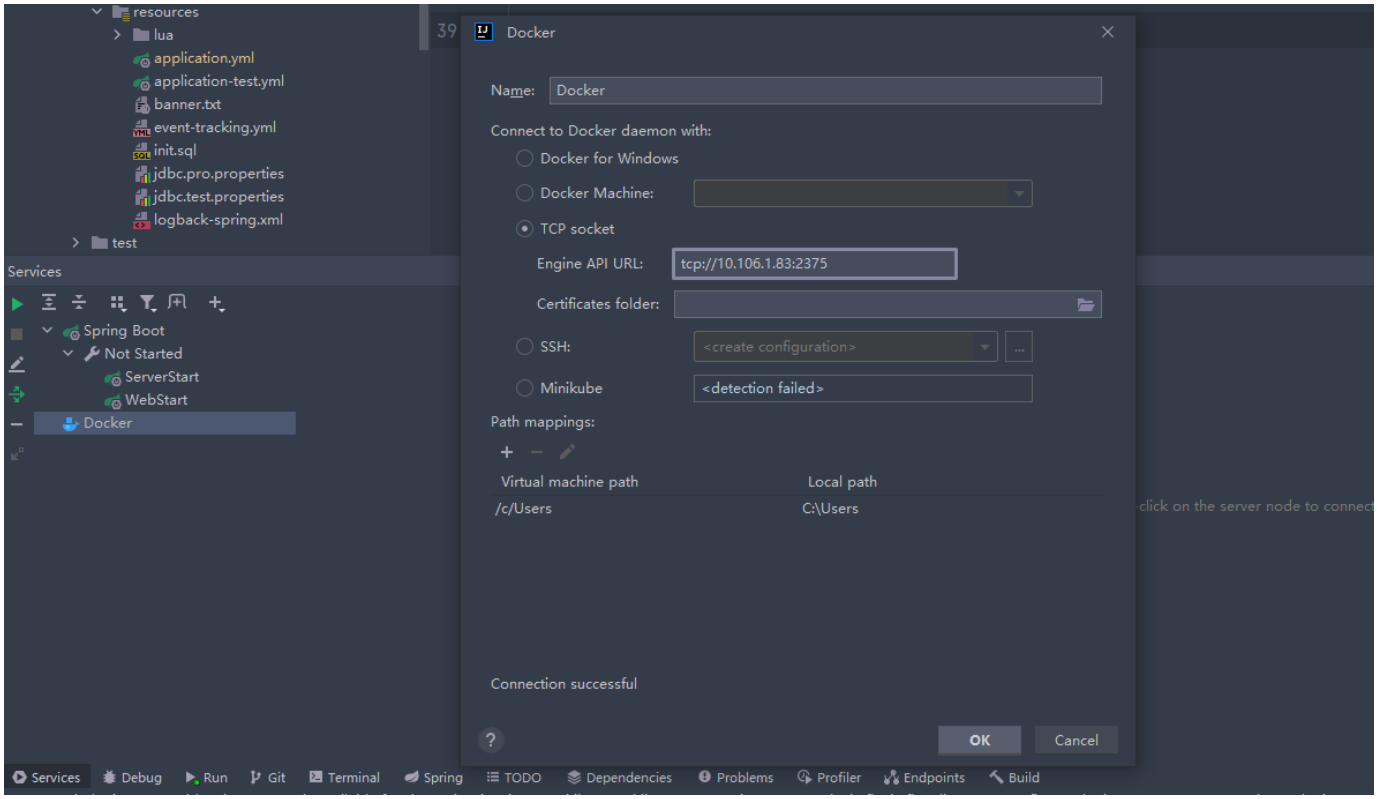
```
root@ubuntuserverx64:~# vi /usr/lib/systemd/system/docker.service

[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
BindsTo=containerd.service
After=network-online.target firewalld.service containerd.service
Wants=network-online.target
Requires=docker.socket

[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock -H tcp://0.0.0.0:2375
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutSec=0
RestartSec=2
Restart=always

# Note that StartLimit* options were moved from "Service" to "Unit" in systemd 229.
# Both the old, and new location are accepted by systemd 229 and up, so using the old location
# to make them work for either version of systemd.
StartLimitBurst=3

# Note that StartLimitInterval was renamed to StartLimitIntervalSec in systemd 230.
# Both the old, and new name are accepted by systemd 230 and up, so using the old name to make
# this option work for either version of systemd.
StartLimitInterval=60s

# Having non-zero Limit*s causes performance problems due to accounting overhead
# in the kernel. We recommend using cgroups to do container-local accounting.
LimitNOFILE=infinity
LimitNPROC=infinity
LimitCORE=infinity

# Comment TasksMax if your systemd version does not support it.
# Only systemd 226 and above support this option.
TasksMax=infinity

# set delegate yes so that systemd does not reset the cgroups of docker containers
Delegate=yes

# kill only the docker process, not all processes in the cgroup
KillMode=process

[Install]
WantedBy=multi-user.target
~
~
```

3. 重载docker守护线程systemctl daemon-reload

4. 重启dockersystemctl restart docker

5. 测试是否开启成功curl http://localhost:2375/version

三、IDEA安装Docker插件，添加Docker服务，连接测试环境Docker



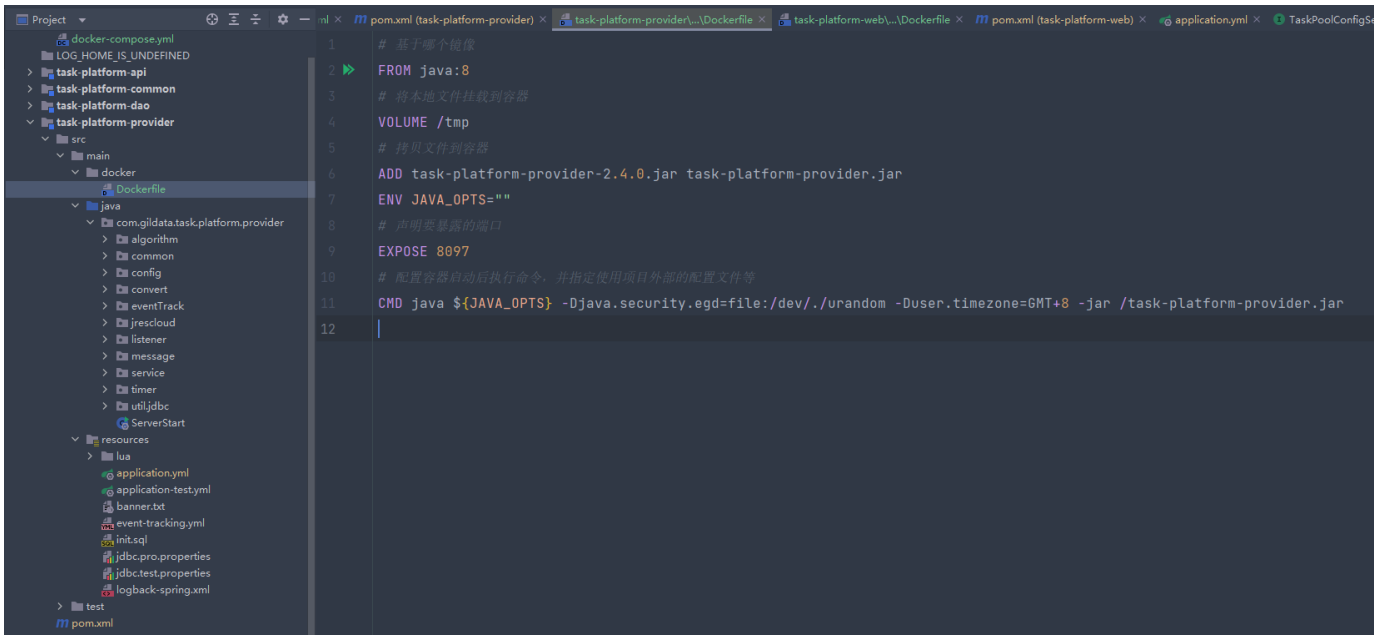
四、pom文件配置Docker build插件

```
<build>
<plugins>
<!--SpringBoot-->
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
<executions>
<execution>
<goals>
<goal>repackage</goal>
</goals>
</execution>
</executions>
</plugin>
<!--dockermaven-->
<plugin>
<groupId>com.spotify</groupId>
<artifactId>docker-maven-plugin</artifactId>
<version>1.2.0</version>
<executions>
<execution>
<id>build-image</id>
<phase>package</phase>
<goals>
<goal>build</goal>
</goals>
</execution>
</executions>
<configuration>
<!---->
<imageName>${project.artifactId}</imageName>
<imageTags>
<imageTag>latest</imageTag>
<imageTag>${project.version}</imageTag>
</imageTags>
<!--docker-->
<dockerHost>http://10.106.1.83:2375</dockerHost>
<dockerDirectory>src/main/docker</dockerDirectory>
<resources>
<!--jar-->
<resource>
<targetPath>/</targetPath>
<directory>${project.build.directory}</directory>
<include>${project.build.finalName}.jar</include>
</resource>
</resources>
</configuration>
</plugin>
</plugins>
</build>
```

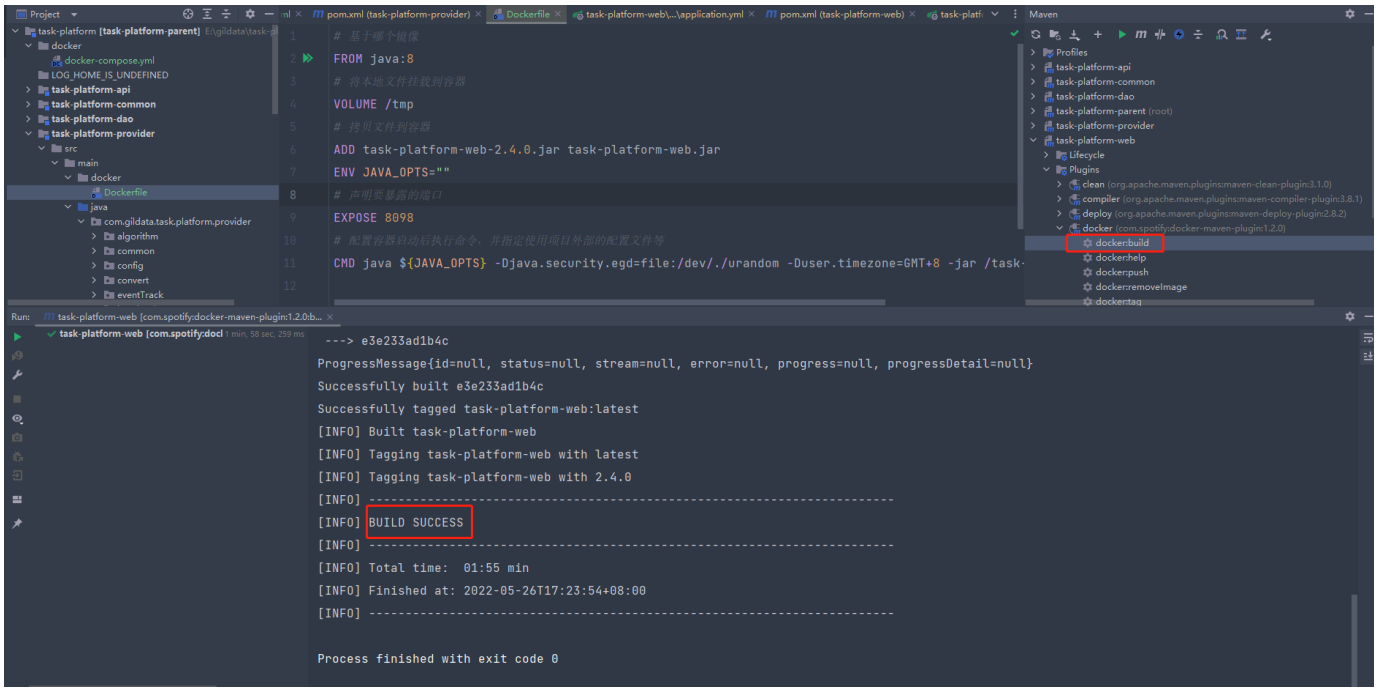
```xml
<build>
    <plugins>
        <!--SpringBoot的打包插件-->
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <executions>
                <execution>
                    <goals>
                        <goal>repackage</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
        <!--制作docker镜像的maven插件-->
        <plugin>
            <groupId>com.spotify</groupId>
            <artifactId>docker-maven-plugin</artifactId>
            <version>1.2.0</version>
            <executions>
                <execution>
                    <id>build-image</id>
                    <phase>package</phase>
                    <goals>
                        <goal>build</goal>
                    </goals>
                </execution>
            </executions>
```

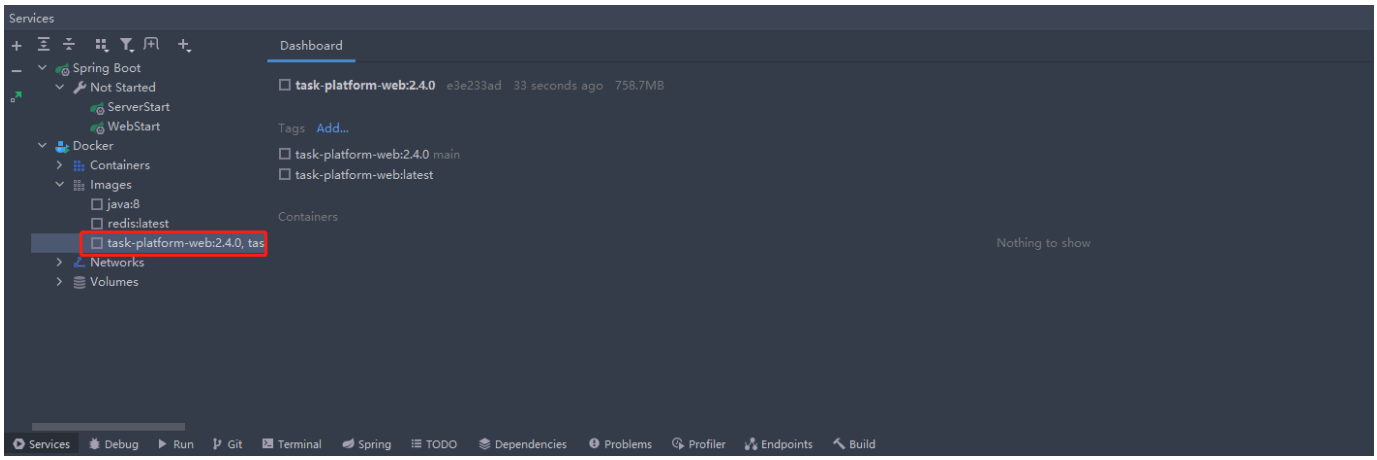
```xml
            <configuration>
                <!--镜像名-->
                <imageName>${project.artifactId}</imageName>
                <imageTags>
                    <imageTag>latest</imageTag>
                    <imageTag>${project.version}</imageTag>
                </imageTags>
                <!--docker所在宿主机的地址-->
                <dockerHost>http://10.106.1.83:2375</dockerHost>
                <dockerDirectory>src/main/docker</dockerDirectory>
                <resources>
                    <!--打包后jar所在的位置-->
                    <resource>
                        <targetPath>/</targetPath>
                        <directory>${project.build.directory}</directory>
                        <include>${project.build.finalName}.jar</include>
                    </resource>
                </resources>
            </configuration>
        </plugin>
    </plugins>
</build>
```

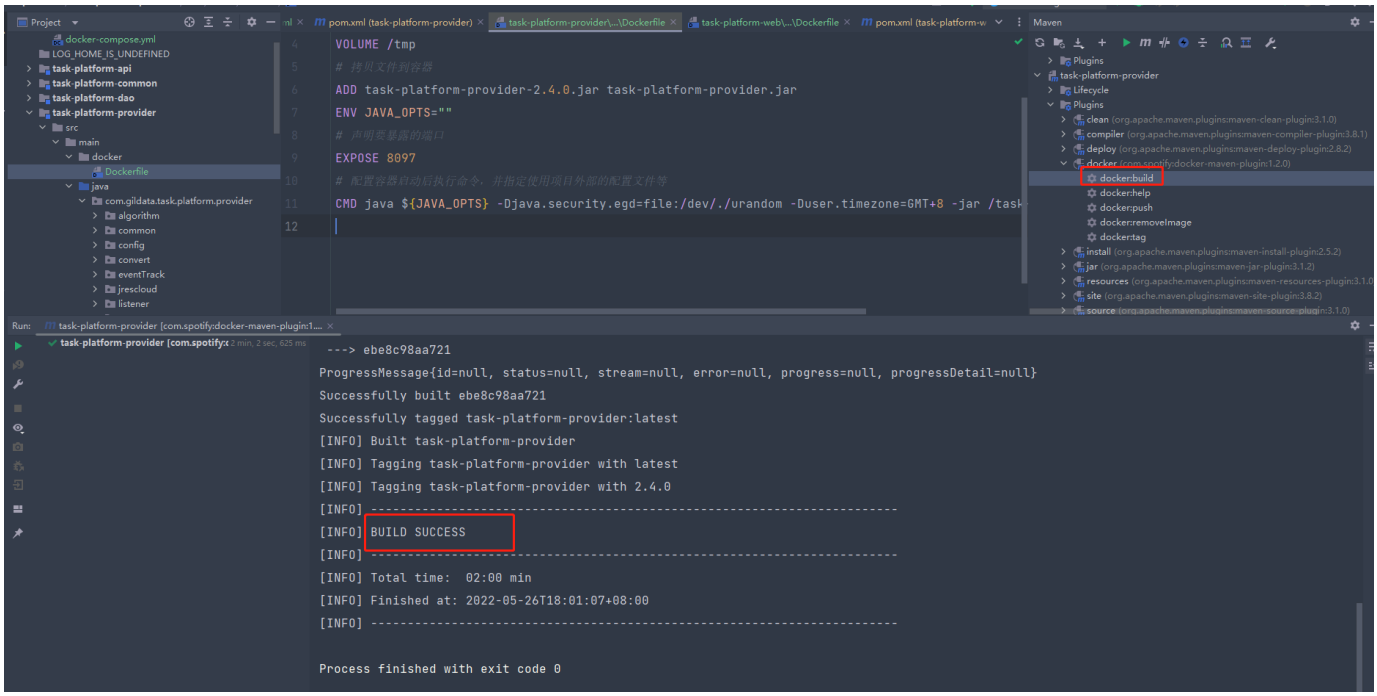
五、编辑Dockerfile文件

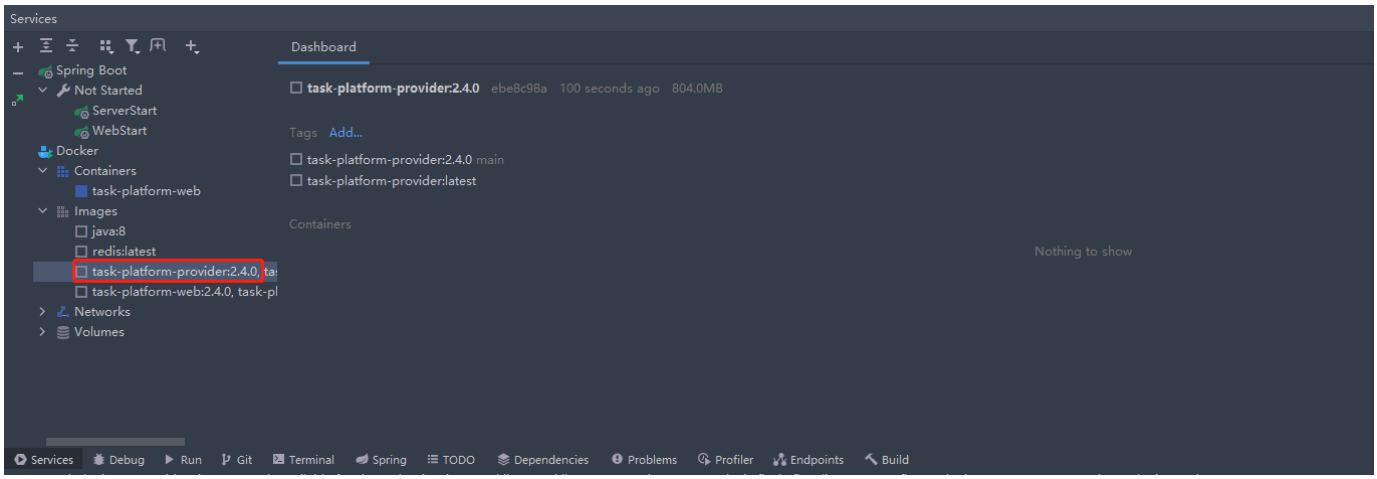


六、maven打包项目，运行docker build，读取pom文件的配置获取Dockerfile文件和jar包，构建镜像到测试环境

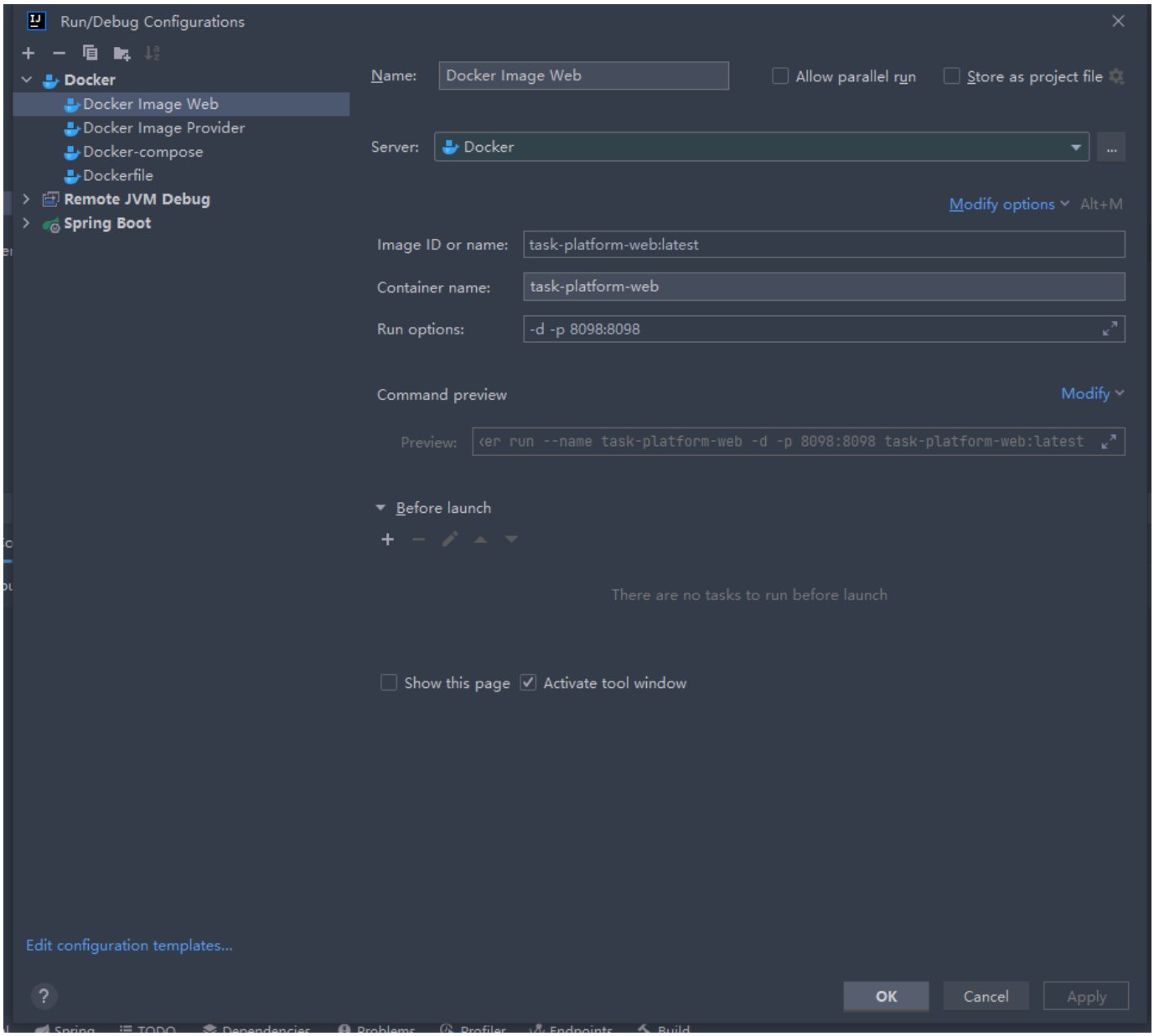

Project
task-platform [task-platform-parent]
docker
docker-compose.yml
LOG_HOME_IS_UNDEFINED
task-platform-api
task-platform-common
task-platform-dao
task-platform-provider
src
main
docker
Dockerfile
java
com.gildata.task.platform.provider
algorithm
common
config
convert
eventTrack
pom.xml (task-platform-provider)
Dockerfile
task-platform-web\...\application.yml
pom.xml (task-platform-web)
基于哪个镜像
FROM java:8
将本地文件挂载到容器
VOLUME /tmp
拷贝文件到容器
ADD task-platform-web-2.4.0.jar task-platform-web.jar
ENV JAVA_OPTS=""
声明要暴露的端口
EXPOSE 8098
配置容器启动后执行命令，并指定使用项目外部的配置文件等
CMD java ${JAVA_OPTS} -Djava.security.egd=file:/dev/./urandom -Duser.timezone=GMT+8 -jar /task-
Maven
Profiles
task-platform-api
task-platform-common
task-platform-dao
task-platform-parent (root)
task-platform-provider
task-platform-web
Lifecycle
Plugins
clean (org.apache.maven.plugins:maven-clean-plugin:3.1.0)
compiler (org.apache.maven.plugins:maven-compiler-plugin:3.8.1)
deploy (org.apache.maven.plugins:maven-deploy-plugin:2.8.2)
docker (com.spotify:docker-maven-plugin:1.2.0)
docker:build
docker:help
docker:push
docker:removeImage
docker:tag
Run:
task-platform-web [com.spotify:docker-maven-plugin:1.2.0:b...
task-platform-web [com.spotify:docd
---> e3e233ad1b4c
ProgressMessage{id=null, status=null, stream=null, error=null, progress=null, progressDetail=null}
Successfully built e3e233ad1b4c
Successfully tagged task-platform-web:latest
[INFO] Built task-platform-web
[INFO] Tagging task-platform-web with latest
[INFO] Tagging task-platform-web with 2.4.0
[INFO] --
[INFO] BUILD SUCCESS
[INFO] --
[INFO] Total time: 01:55 min
[INFO] Finished at: 2022-05-26T17:23:54+08:00
[INFO] --
Process finished with exit code 0

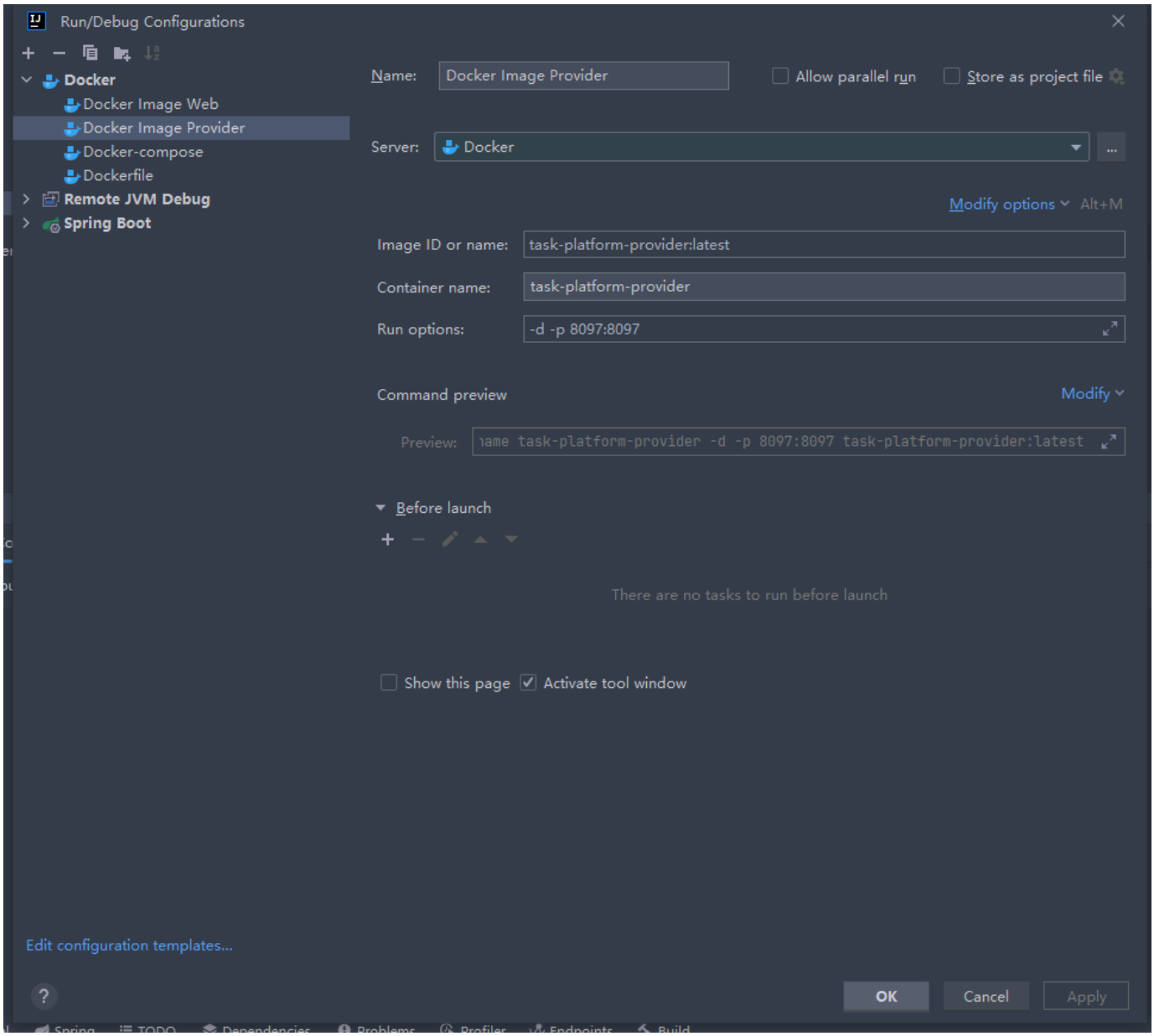

Services
Spring Boot
Not Started
ServerStart
WebStart
Docker
Containers
Images
java:8
redis:latest
task-platform-web:2.4.0, tas
Networks
Volumes
Dashboard
task-platform-web:2.4.0 e3e233ad 33 seconds ago 758.7MB
Tags Add...
task-platform-web:2.4.0 main
task-platform-web:latest
Containers
Nothing to show
Services
Debug
Run
Git
Terminal
Spring
TODO
Dependencies
Problems
Profiler
Endpoints
Build





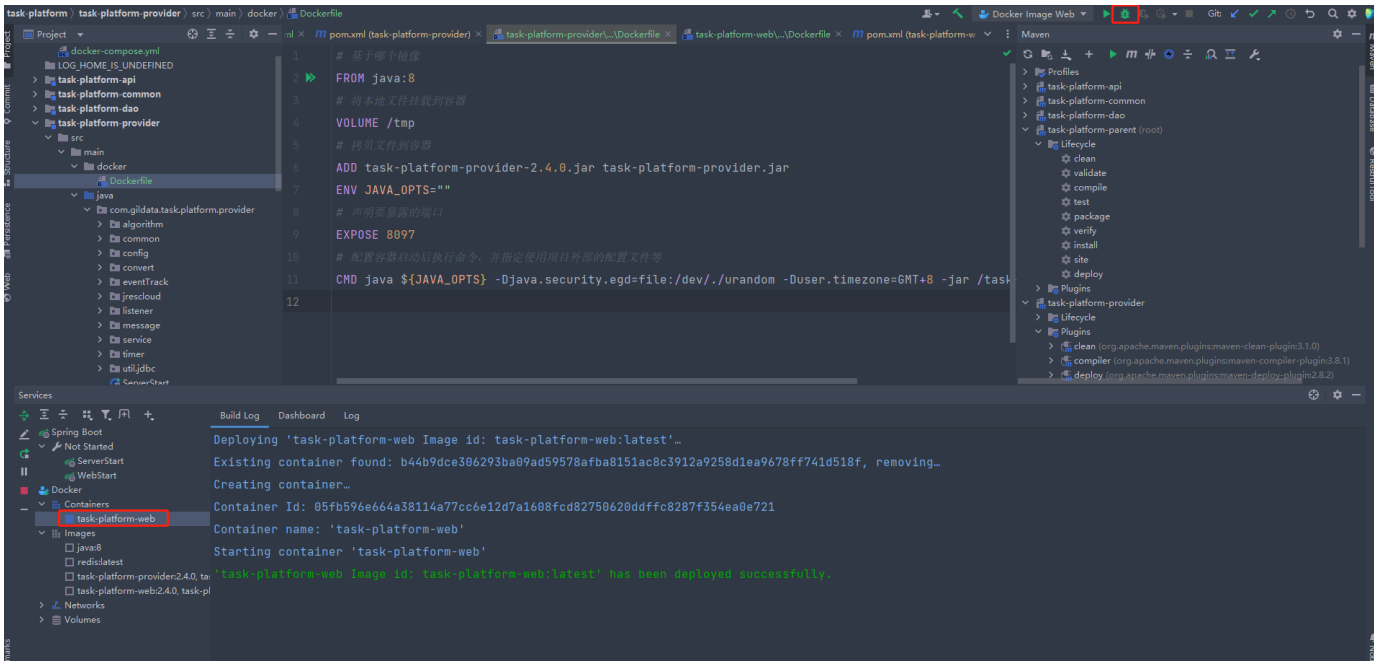
七、添加Docker image启动器，编辑构建容器参数

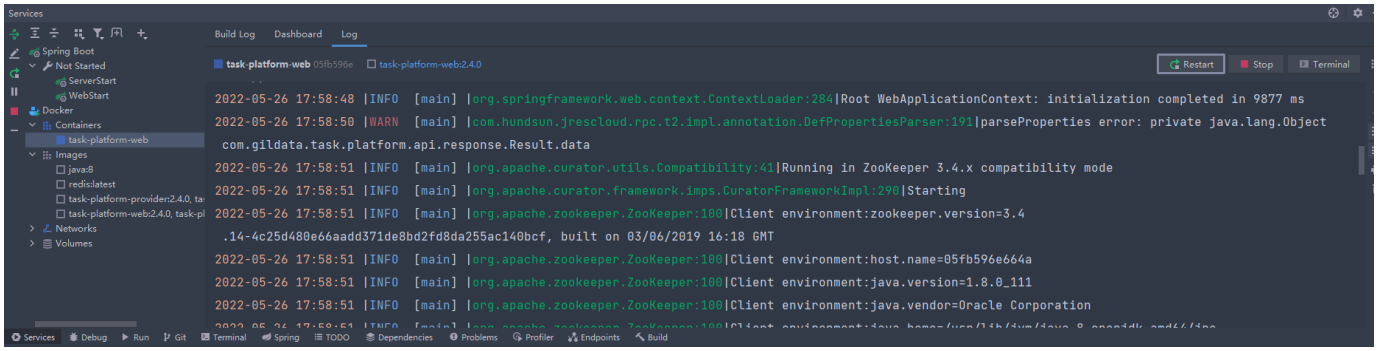

Run/Debug Configurations
Docker
Docker Image Web
Docker Image Provider
Docker-compose
Dockerfile
Remote JVM Debug
Spring Boot
Name: Docker Image Web
Allow parallel run
Store as project file
Server: Docker
Modify options Alt+M
Image ID or name: task-platform-web:latest
Container name: task-platform-web
Run options: -d -p 8098:8098
Command preview
Modify
Preview: ‹er run --name task-platform-web -d -p 8098:8098 task-platform-web:latest
Before launch
There are no tasks to run before launch
Show this page
Activate tool window
Edit configuration templates...
OK
Cancel
Apply

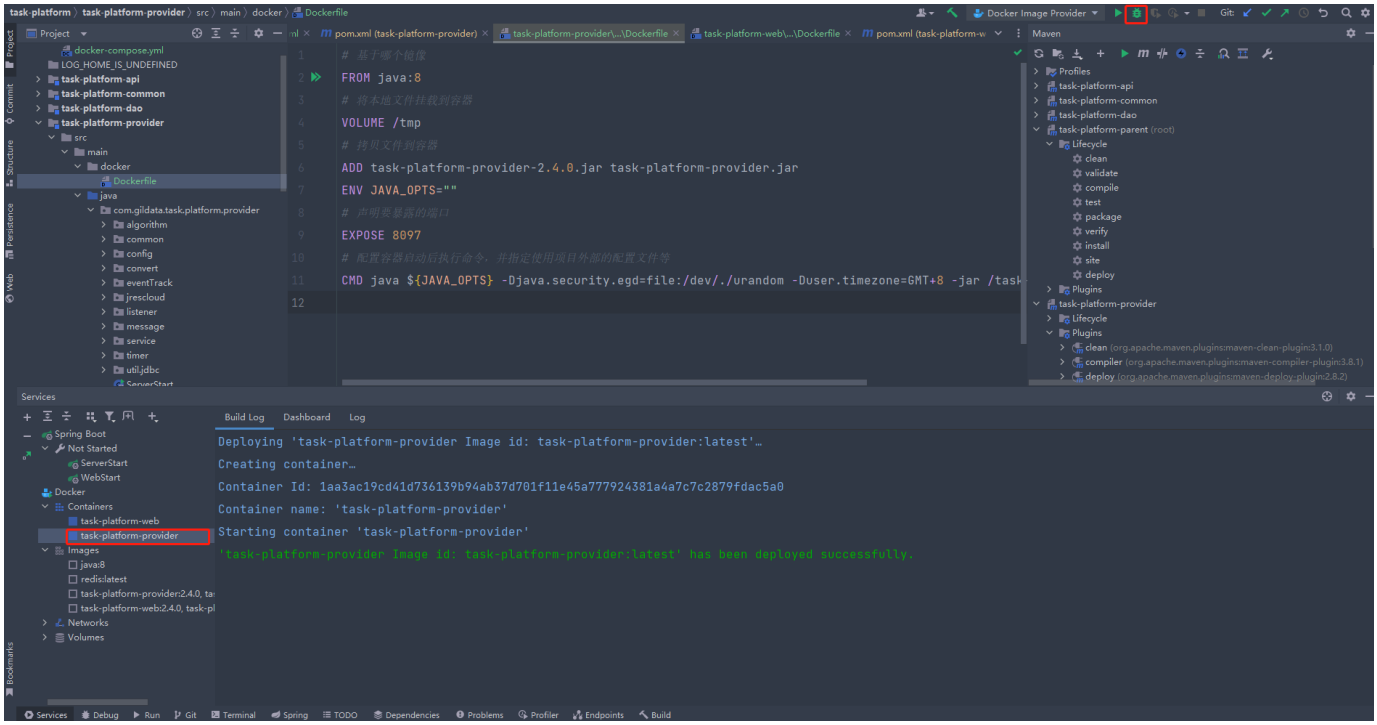


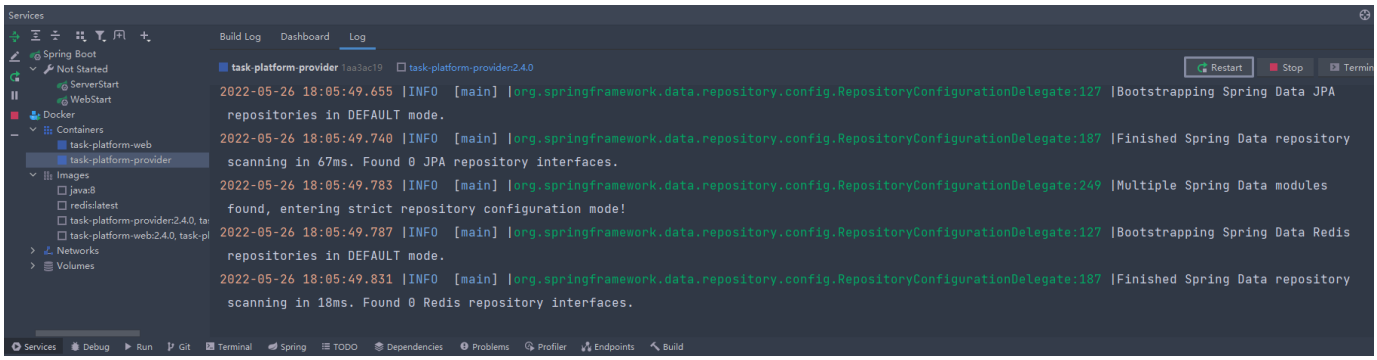
八、构建容器

1. 可以根据第七步的配置，通过docker image构建容器，依次启动服务

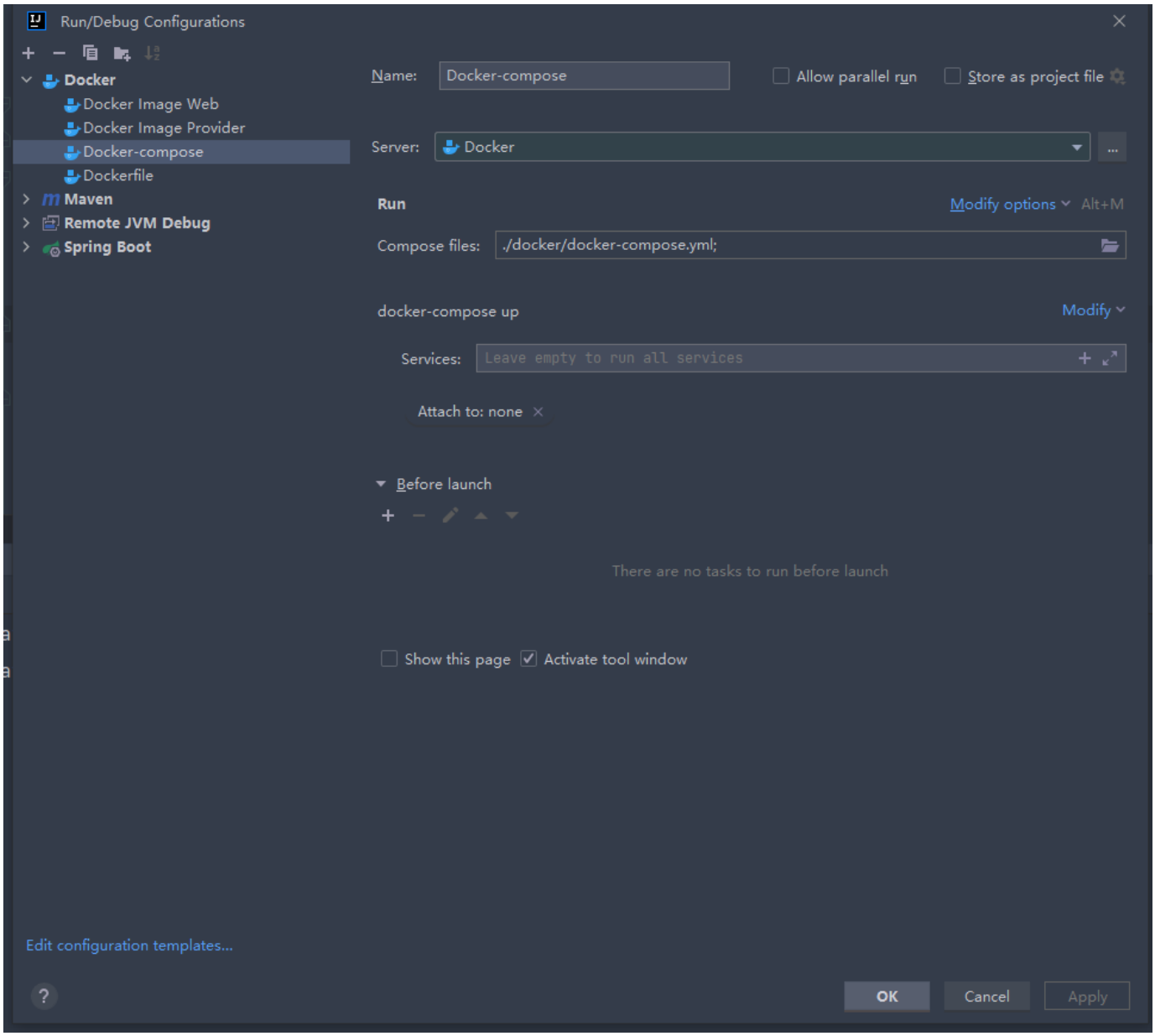

task-platform > task-platform-provider > src > main > docker > Dockerfile
Docker Image Web
Project
pom.xml (task-platform-provider)
task-platform-provider\...\Dockerfile
task-platform-web\...\Dockerfile
pom.xml (task-platform-w
Maven
docker-compose.yml
LOG_HOME_IS_UNDEFINED
task-platform-api
task-platform-common
task-platform-dao
task-platform-provider
src
main
docker
Dockerfile
java
com.gildata.task.platform.provider
algorithm
common
config
convert
eventTrack
jrescloud
listener
message
service
timer
util.jdbc
基于哪个镜像
FROM java:8
将本地文件挂载到容器
VOLUME /tmp
拷贝文件到容器
ADD task-platform-provider-2.4.0.jar task-platform-provider.jar
ENV JAVA_OPTS=""
声明要暴露的端口
EXPOSE 8097
配置容器启动后执行命令，并指定使用项目外部的配置文件等
CMD java ${JAVA_OPTS} -Djava.security.egd=file:/dev/./urandom -Duser.timezone=GMT+8 -jar /task
Profiles
task-platform-api
task-platform-common
task-platform-dao
task-platform-parent (root)
Lifecycle
clean
validate
compile
test
package
verify
install
site
deploy
Plugins
task-platform-provider
clean (org.apache.maven.plugins:maven-clean-plugin:3.1.0)
compiler (org.apache.maven.plugins:maven-compiler-plugin:3.8.1)
deploy (org.apache.maven.plugins:maven-deploy-plugin:2.8.2)
Services
Build Log
Dashboard
Log
Spring Boot
Not Started
ServerStart
WebStart
Docker
Containers
task-platform-web
Images
java:8
redis:latest
task-platform-provider:2.4.0, ta
task-platform-web:2.4.0, task-pl
Networks
Volumes
Deploying 'task-platform-web Image id: task-platform-web:latest'…
Existing container found: b44b9dce306293ba09ad59578afba8151ac8c3912a9258d1ea9678ff741d518f, removing…
Creating container…
Container Id: 05fb596e664a38114a77cc6e12d7a1608fcd82750620ddffc8287f354ea0e721
Container name: 'task-platform-web'
Starting container 'task-platform-web'
'task-platform-web Image id: task-platform-web:latest' has been deployed successfully.

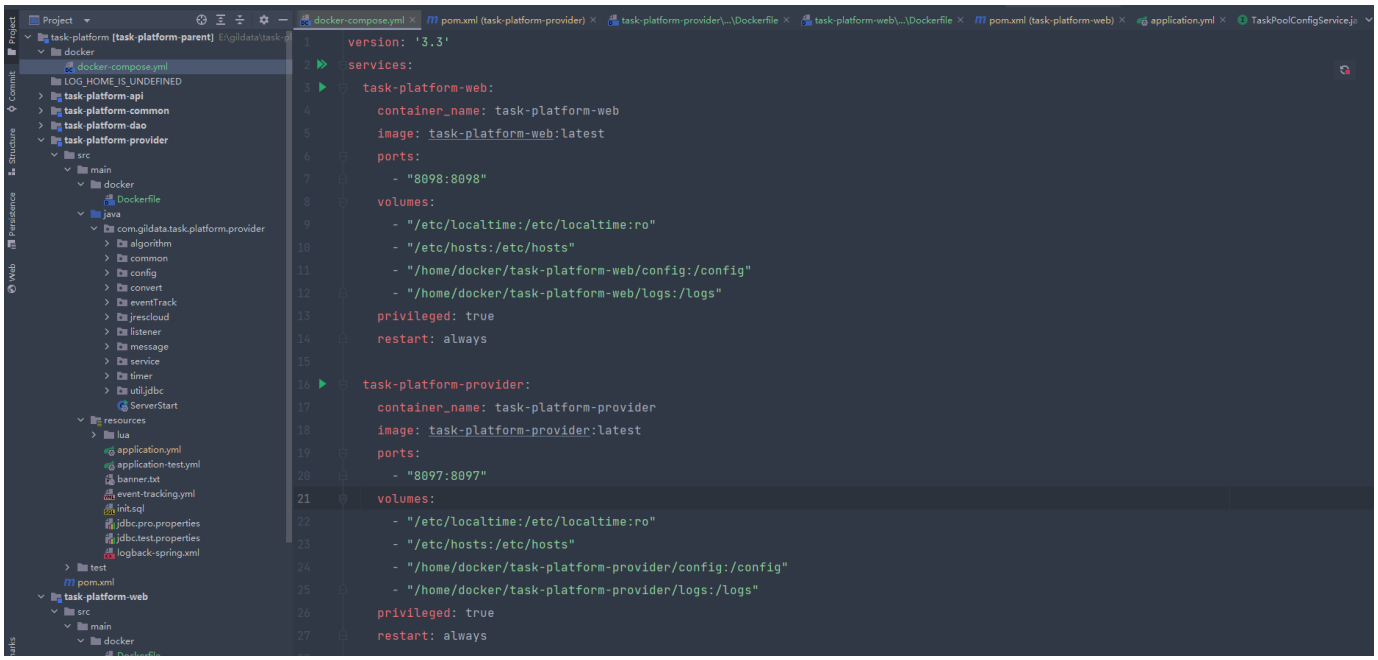

Services
Build Log
Dashboard
Log
task-platform-web 05fb596e
task-platform-web:2.4.0
Restart
Stop
Terminal
2022-05-26 17:58:48 |INFO [main] |org.springframework.web.context.ContextLoader:284|Root WebApplicationContext: initialization completed in 9877 ms
2022-05-26 17:58:50 |WARN [main] |com.hundsun.jrescloud.rpc.t2.impl.annotation.DefPropertiesParser:191|parseProperties error: private java.lang.Object
com.gildata.task.platform.api.response.Result.data
2022-05-26 17:58:51 |INFO [main] |org.apache.curator.utils.Compatibility:41|Running in ZooKeeper 3.4.x compatibility mode
2022-05-26 17:58:51 |INFO [main] |org.apache.curator.framework.imps.CuratorFrameworkImpl:290|Starting
2022-05-26 17:58:51 |INFO [main] |org.apache.zookeeper.ZooKeeper:100|Client environment:zookeeper.version=3.4
.14-4c25d480e66aadd371de8bd2fd8da255ac140bcf, built on 03/06/2019 16:18 GMT
2022-05-26 17:58:51 |INFO [main] |org.apache.zookeeper.ZooKeeper:100|Client environment:host.name=05fb596e664a
2022-05-26 17:58:51 |INFO [main] |org.apache.zookeeper.ZooKeeper:100|Client environment:java.version=1.8.0_111
2022-05-26 17:58:51 |INFO [main] |org.apache.zookeeper.ZooKeeper:100|Client environment:java.vendor=Oracle Corporation
Services
Debug
Run
Git
Terminal
Spring
TODO
Dependencies
Problems
Profiler
Endpoints
Build

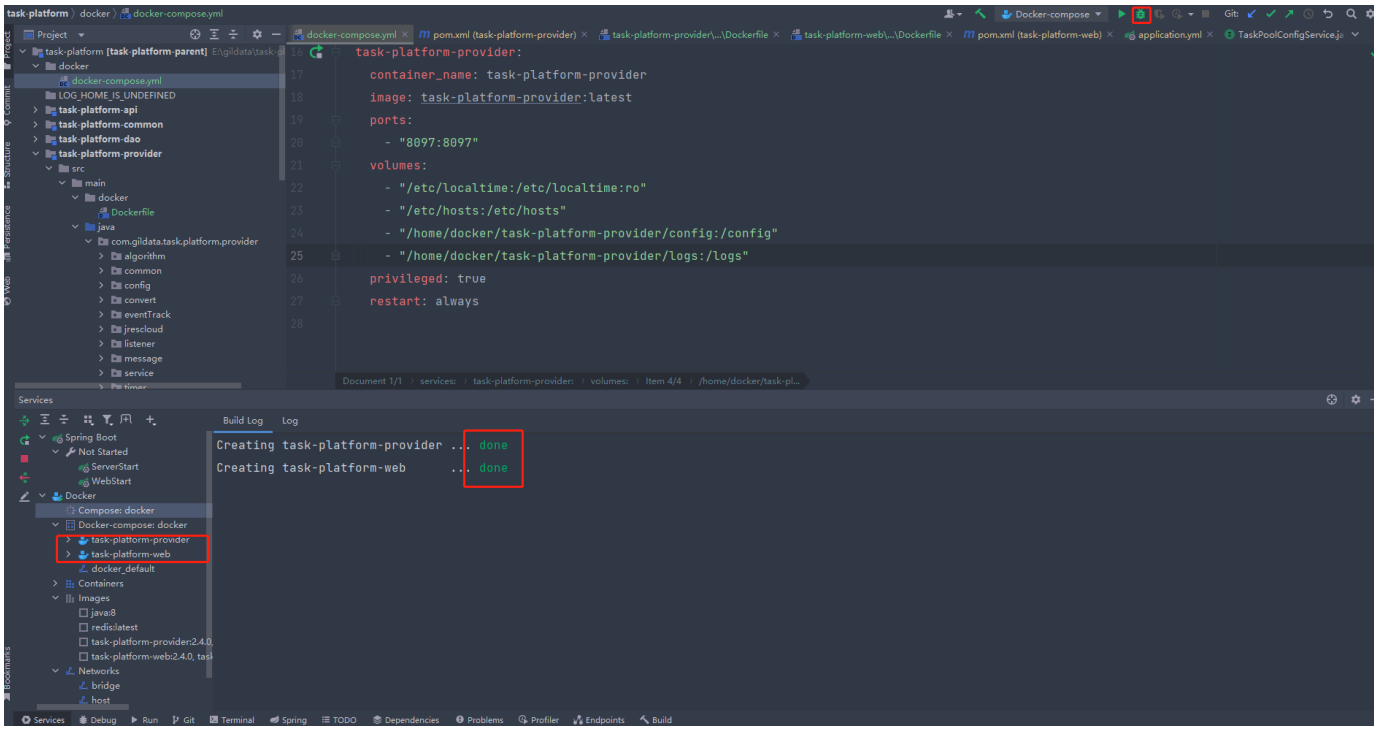

task-platform > task-platform-provider > src > main > docker > Dockerfile
Docker Image Provider
Project
pom.xml (task-platform-provider)
task-platform-provider\...\Dockerfile
task-platform-web\...\Dockerfile
pom.xml (task-platform-w
Maven
docker-compose.yml
LOG_HOME_IS_UNDEFINED
task-platform-api
task-platform-common
task-platform-dao
task-platform-provider
src
main
docker
Dockerfile
java
com.gildata.task.platform.provider
algorithm
common
config
convert
eventTrack
jrescloud
listener
message
service
timer
util.jdbc
基于哪个镜像
FROM java:8
将本地文件挂载到容器
VOLUME /tmp
拷贝文件到容器
ADD task-platform-provider-2.4.0.jar task-platform-provider.jar
ENV JAVA_OPTS=""
声明要暴露的端口
EXPOSE 8097
配置容器启动后执行命令，并指定使用项目外部的配置文件等
CMD java ${JAVA_OPTS} -Djava.security.egd=file:/dev/./urandom -Duser.timezone=GMT+8 -jar /task
Profiles
task-platform-api
task-platform-common
task-platform-dao
task-platform-parent (root)
Lifecycle
clean
validate
compile
test
package
verify
install
site
deploy
Plugins
task-platform-provider
clean (org.apache.maven.plugins:maven-clean-plugin:3.1.0)
compiler (org.apache.maven.plugins:maven-compiler-plugin:3.8.1)
deploy (org.apache.maven.plugins:maven-deploy-plugin:2.8.2)
Services
Build Log
Dashboard
Log
Spring Boot
Not Started
ServerStart
WebStart
Docker
Containers
task-platform-web
task-platform-provider
Images
java:8
redis:latest
task-platform-provider:2.4.0, ta
task-platform-web:2.4.0, task-pl
Networks
Volumes
Deploying 'task-platform-provider Image id: task-platform-provider:latest'…
Creating container…
Container Id: 1aa3ac19cd41d736139b94ab37d701f11e45a777924381a4a7c7c2879fdac5a0
Container name: 'task-platform-provider'
Starting container 'task-platform-provider'
'task-platform-provider Image id: task-platform-provider:latest' has been deployed successfully.
Services
Debug
Run
Git
Terminal
Spring
TODO
Dependencies
Problems
Profiler
Endpoints
Build



2. 通过docker-compose.yml文件编排服务启动







重点问题：

当我在测试环境上按照在本地部署一样进行时，容器部署后，服务启动失败，具体表现为容器无法访问外网，连宿主机都无法ping通，调试了很久也无法解决，怀疑和网络配置有关或者服务器的原因。

尝试了下将容器的网络模式由bridge（默认；宿主机与容器通过虚拟网桥通过端口映射）改为host（和宿主机共享网络）模式，服务成功运行，docker-compose.yml文件修改：

```yaml
version: '3.3'
services:
  task-platform-web:
    container_name: task-platform-web
    image: task-platform-web:latest
    volumes:
      - "/etc/localtime:/etc/localtime:ro"
      - "/etc/hosts:/etc/hosts"
      - "/home/docker/task-platform-web/config:/config"
      - "/home/docker/task-platform-web/logs:/logs"
    network_mode: "host"
    privileged: true
    restart: always

  task-platform-provider:
    container_name: task-platform-provider
    image: task-platform-provider:latest
    volumes:
      - "/etc/localtime:/etc/localtime:ro"
      - "/etc/hosts:/etc/hosts"
      - "/home/docker/task-platform-provider/config:/config"
      - "/home/docker/task-platform-provider/logs:/logs"
    network_mode: "host"
    privileged: true
    restart: always
```