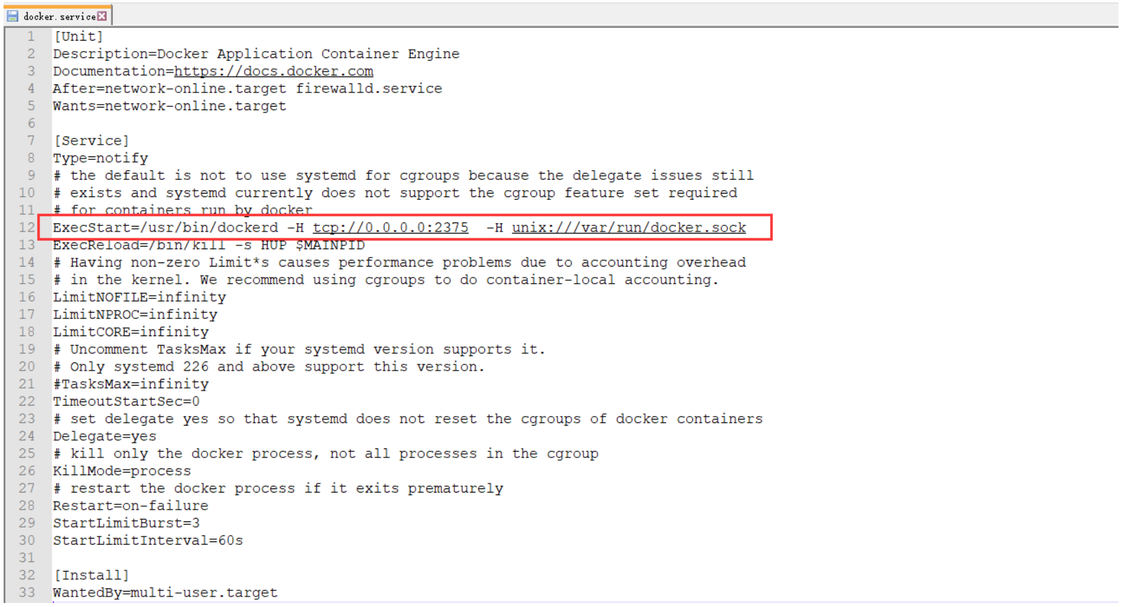
04-基于Docker容器，使用Intellij IDEA一键部署权限管理系统到云环境

本节内容，我们使用IntelliJ IDEA的Docker插件帮助我们将当前权限管理应用制作成Docker镜像、运行在指定的远程机器(阿里云)上。

实现步骤：1.在CentSO系统上开启Docker的远程连接，如果你对docker安装和基本的操作还不熟悉，请参考我的docker课程之后在继续下面的内容。

- 编辑此文件：/lib/systemd/system/docker.service，把ExecStart=/usr/bin/dockerd-current \改为ExecStart=/usr/bin/dockerd-current -H tcp://0.0.0.0:2375 -H unix://var/run/docker.sock \，如下：



docker.service

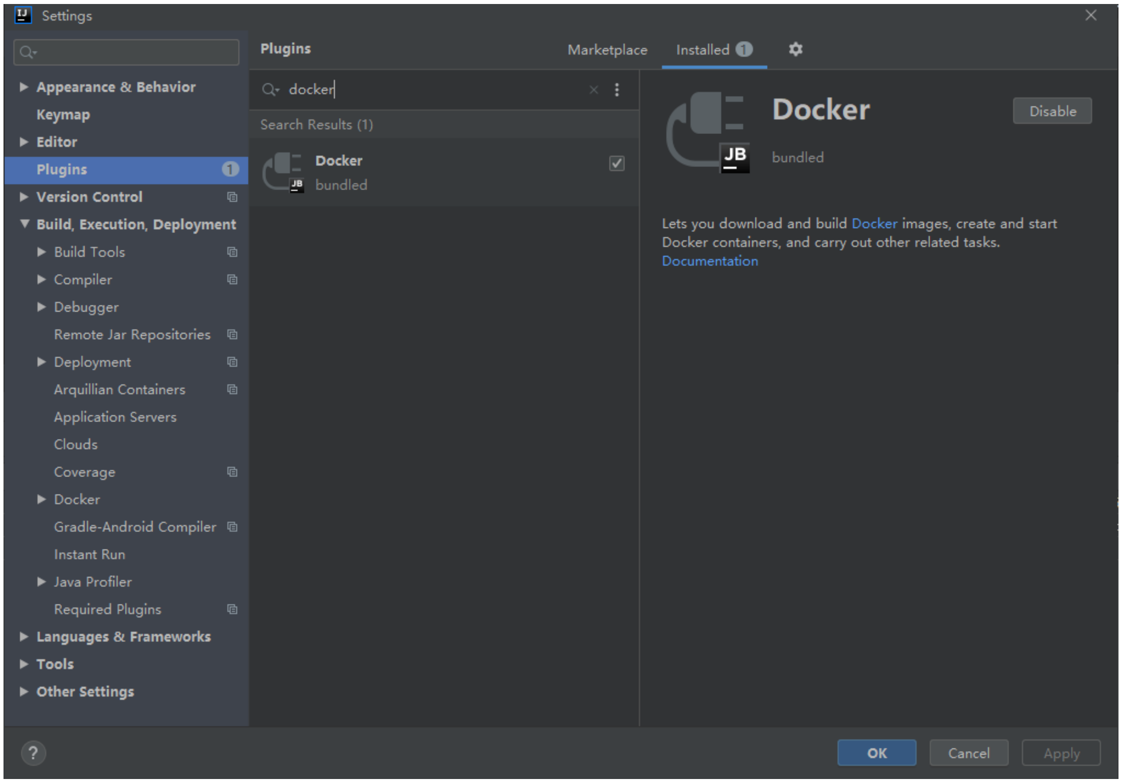
```
[Unit]
Description=Docker Application Container Engine
Documentation=https://docs.docker.com
After=network-online.target firewalld.service
Wants=network-online.target

[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 tcp://0.0.0.0:2375  -H unix:///var/run/docker.sock
ExecReload=/bin/kill -s HUP $MAINPID
# 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
# Uncomment TasksMax if your systemd version supports it.
# Only systemd 226 and above support this version.
#TasksMax=infinity
TimeoutStartSec=0
# 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
# restart the docker process if it exits prematurely
Restart=on-failure
StartLimitBurst=3
StartLimitInterval=60s

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

- 重新加载配并重启docker：

```
systemctl daemon-reload && systemctl restart docker
```

- 确保端口2375已开启，如果使用阿里云等云服务，记得在安全策略上配置端口2375

2.Intellij IDEA安装Docker插件，打开Idea，从File->Settings->Plugins->Install JetBrains plugin进入插件安装界面，在搜索框中输入docker，可以看到Docker integration，点击右边的Install按钮进行安装。安装后重启Idea。

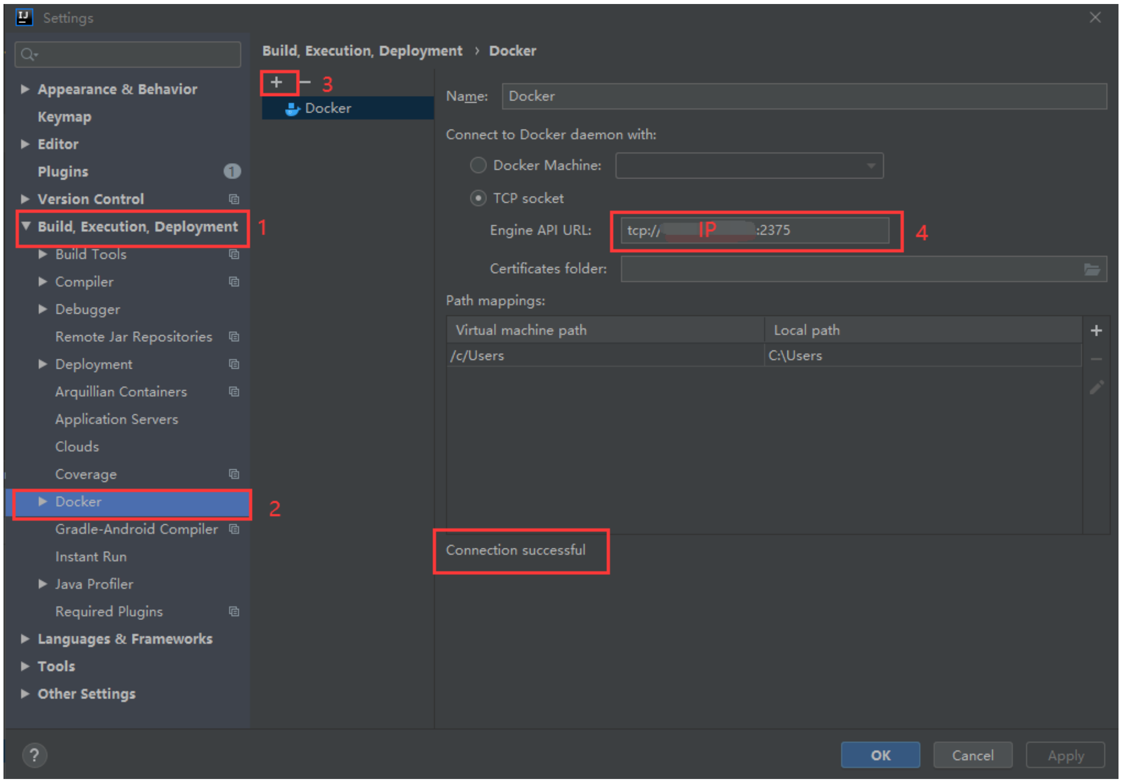


docker

Docker ☑

Lets you download and build Docker images, create and start Docker containers, and carry out other related tasks.

3.重启后配置docker，连接到远程docker服务。从File->Settings->Build,Execution,Deployment->Docker打开配置界面。在设置页面，按照下图的数字顺序创建一个Docker server并进行设置，输入Docker服务所在机器的IP地址，如果连接成功页面上会立即提示"Connection successful"



Name: Docker

Connect to Docker daemon with:

○ Docker Machine:

● TCP socket

Engine API URL: tcp://[redacted]:2375

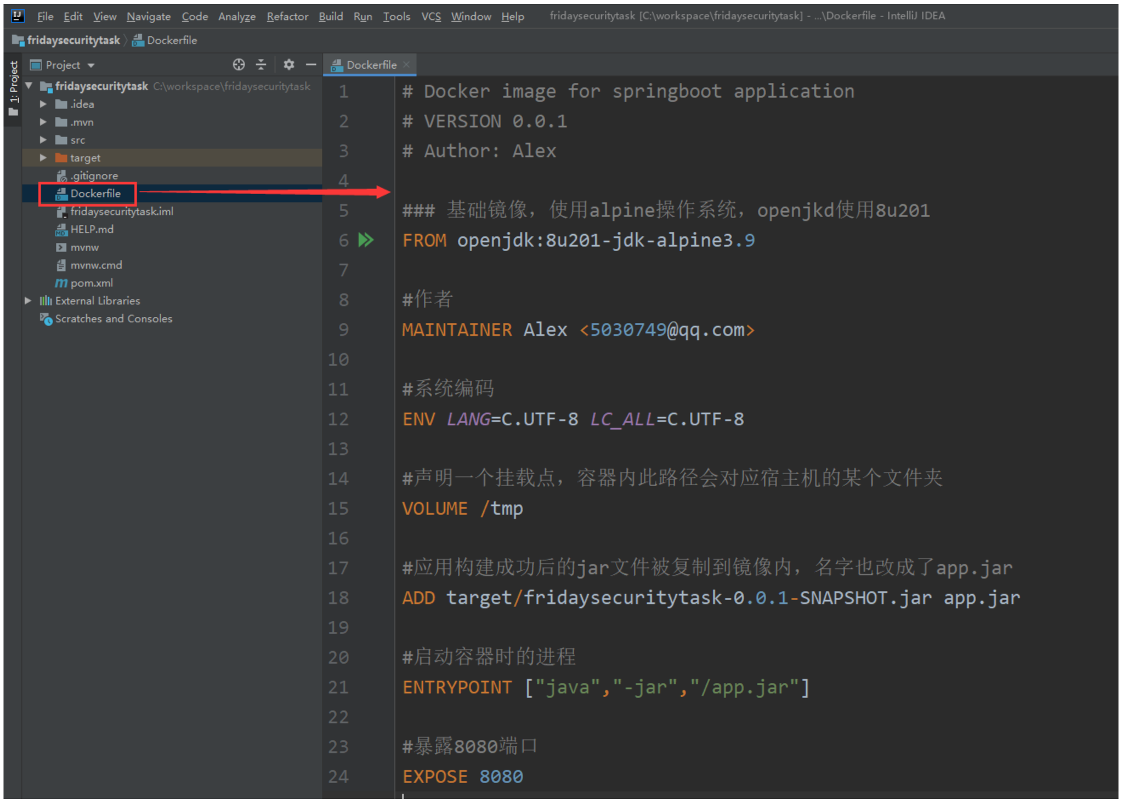
Certificates folder:

Path mappings:

Virtual machine path	Local path
/c/Users	C:\Users

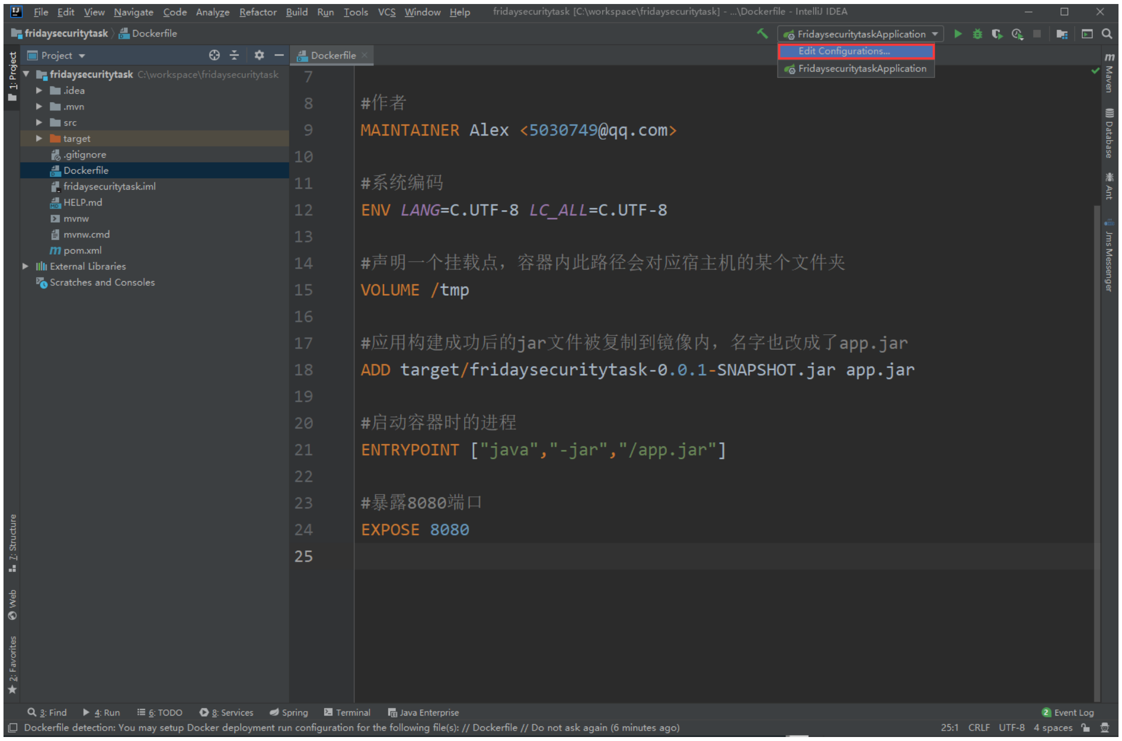
Connection successful

4.在fridaysecuritytask项目目录下创建Dockerfile，内容如下：


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
fridaysecuritytask [C:\workspace\fridaysecuritytask] - ...\Dockerfile - IntelliJ IDEA
fridaysecuritytask
Dockerfile
1: Project
Project
fridaysecuritytask C:\workspace\fridaysecuritytask
.idea
.mvn
src
target
.gitignore
Dockerfile
fridaysecuritytask.iml
HELP.md
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles
Dockerfile

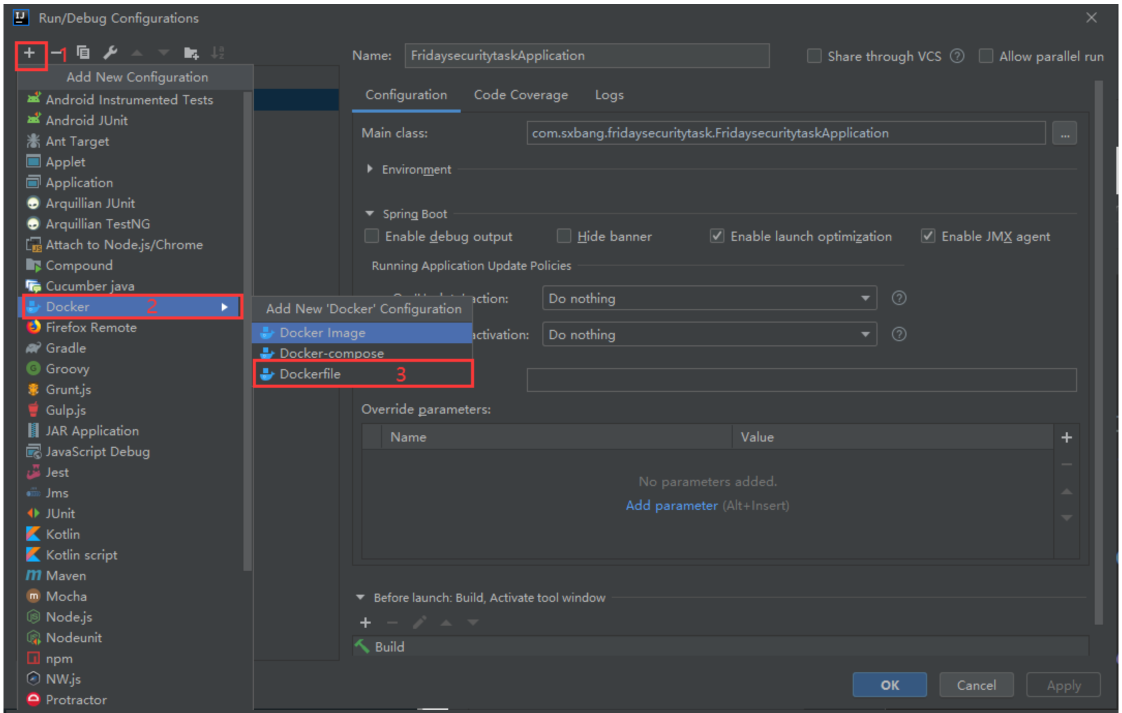
```
# Docker image for springboot application
# VERSION 0.0.1
# Author: Alex

### 基础镜像，使用alpine操作系统，openjkd使用8u201
FROM openjdk:8u201-jdk-alpine3.9

#作者
MAINTAINER Alex <5030749@qq.com>

#系统编码
ENV LANG=C.UTF-8 LC_ALL=C.UTF-8

#声明一个挂载点，容器内此路径会对应宿主机的某个文件夹
VOLUME /tmp

#应用构建成功后的jar文件被复制到镜像内，名字也改成了app.jar
ADD target/fridaysecuritytask-0.0.1-SNAPSHOT.jar app.jar

#启动容器时的进程
ENTRYPOINT ["java","-jar","/app.jar"]

#暴露8080端口
EXPOSE 8080
```

5.按照下图操作，创建一个Dockerfile的配置



```

#作者
MAINTAINER Alex <5030749@qq.com>

#系统编码
ENV LANG=C.UTF-8 LC_ALL=C.UTF-8

#声明一个挂载点，容器内此路径会对应宿主机的某个文件夹
VOLUME /tmp

#应用构建成功后的jar文件被复制到镜像内，名字也改成了app.jar
ADD target/fridaysecuritytask-0.0.1-SNAPSHOT.jar app.jar

#启动容器时的进程
ENTRYPOINT ["java","-jar","/app.jar"]

#暴露8080端口
EXPOSE 8080

```

Dockerfile detection: You may setup Docker deployment run configuration for the following file(s): // Dockerfile // Do not ask again (6 minutes ago)



Name: FridaysecuritytaskApplication

□ Share through VCS □ Allow parallel run

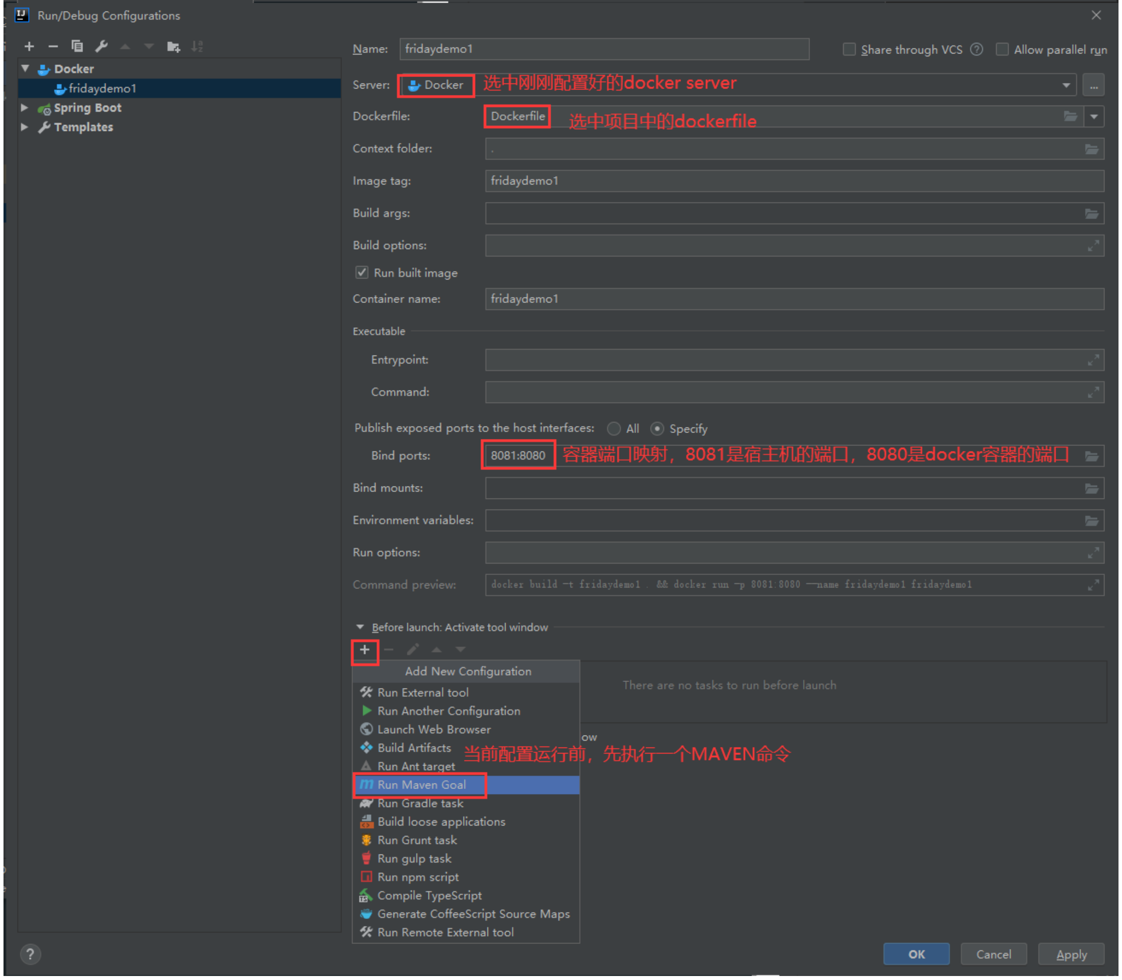
Main class: com.sxbang.fridaysecuritytask.FridaysecuritytaskApplication

□ Enable debug output □ Hide banner ☑ Enable launch optimization ☑ Enable JMX agent

...action: Do nothing

...activation: Do nothing

Before launch: Build, Activate tool window



Name: fridaydemo1

□ Share through VCS □ Allow parallel run

Server: Docker 选中刚刚配置好的docker server

Dockerfile: Dockerfile 选中项目中的dockerfile

Context folder: .

Image tag: fridaydemo1

Build args:

Build options:

☑ Run built image

Container name: fridaydemo1

Entrypoint:

Command:

Publish exposed ports to the host interfaces: ○ All ● Specify

Bind ports: 8081:8080 容器端口映射，8081是宿主机的端口，8080是docker容器的端口

Bind mounts:

Environment variables:

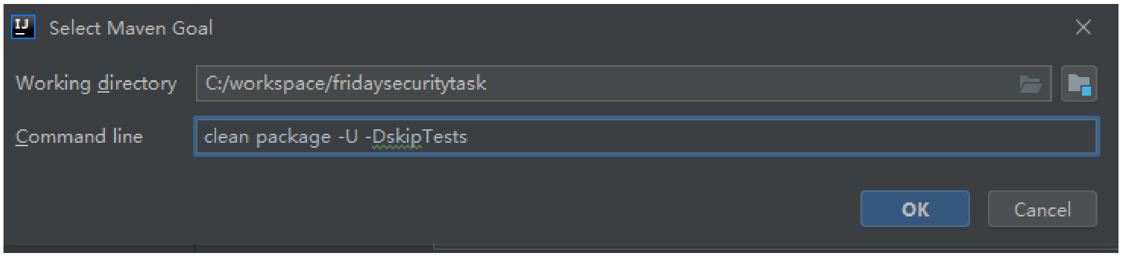
Run options:

Command preview: docker build -t fridaydemo1 . && docker run -p 8081:8080 --name fridaydemo1 fridaydemo1

Before launch: Activate tool window

当前配置运行前，先执行一个MAVEN命令

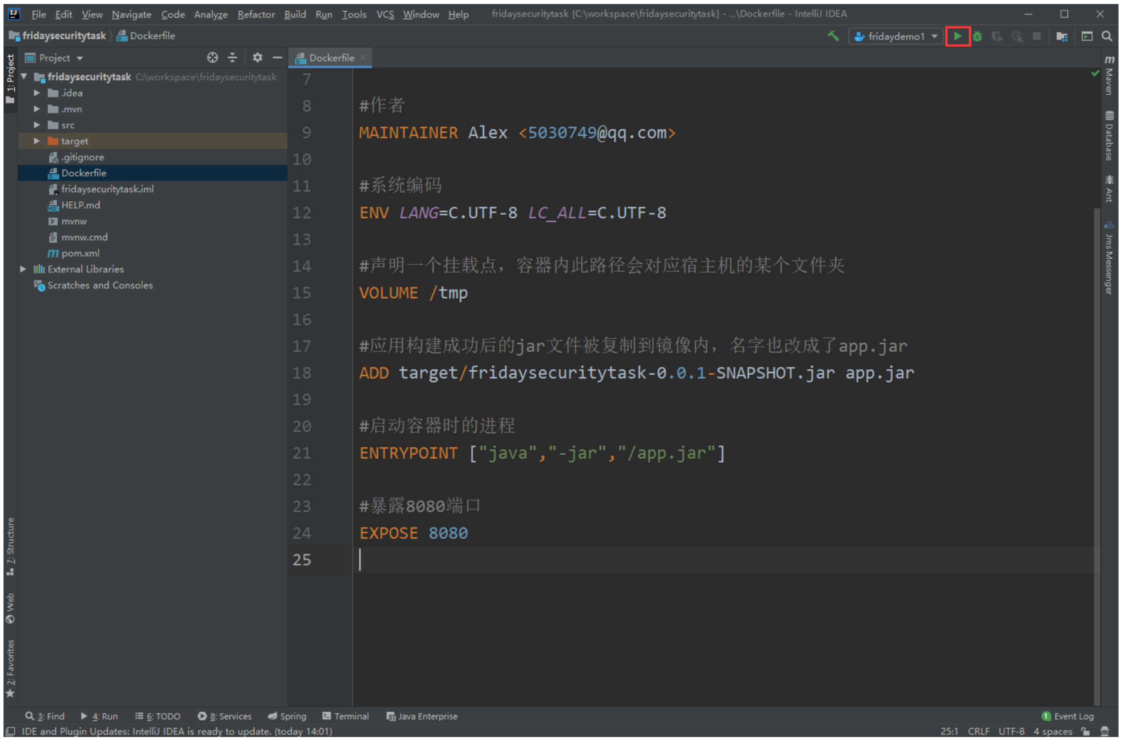
在个"Run Maven Goal"点击后，输入要执行的maven命令clean package -U -DskipTests，表示每次在构建镜像之前，都会将当前工程清理掉并且重新编译构建：



Working directory: C:/workspace/fridaysecuritytask

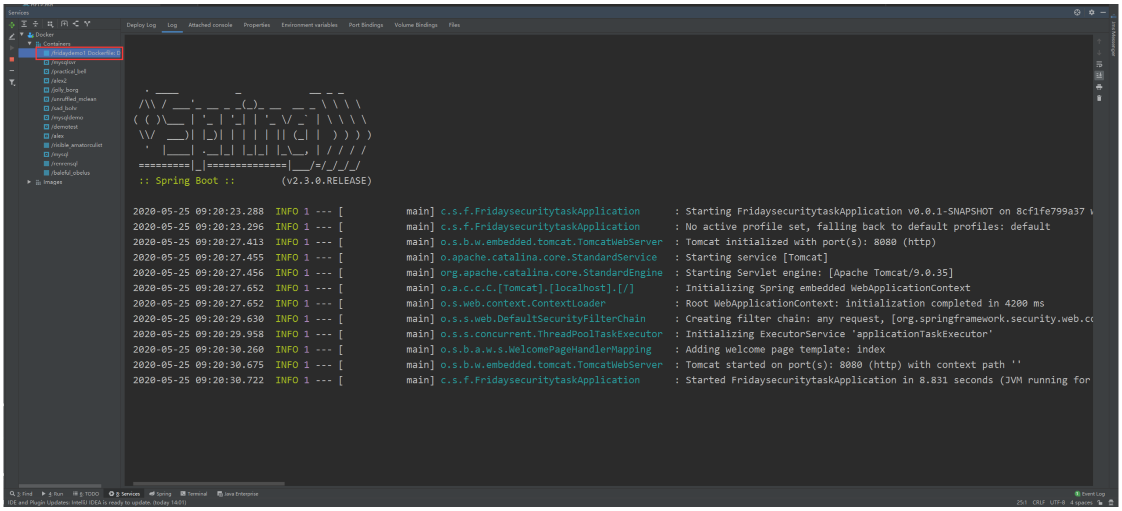
Command line: clean package -U -DskipTests

6.点击三角按钮运行验证

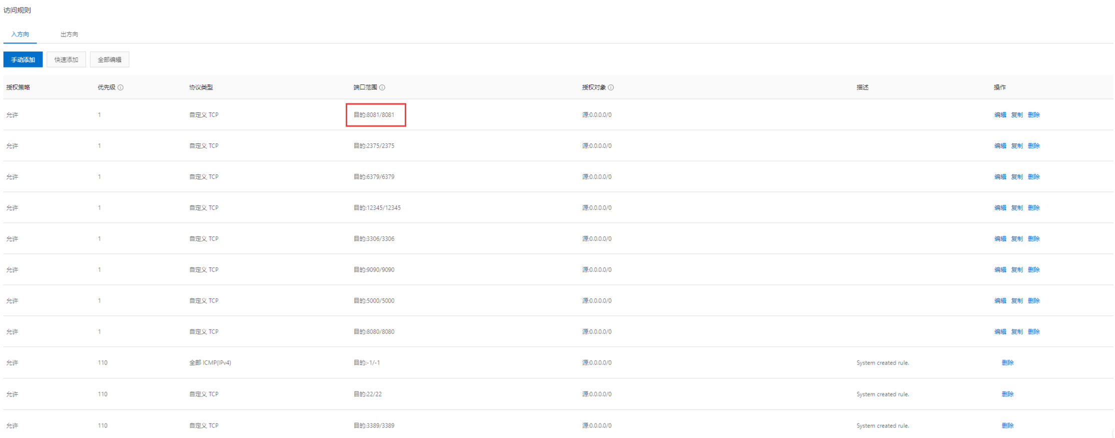

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
fridaysecuritytask [C:\workspace\fridaysecuritytask] - ...\Dockerfile - IntelliJ IDEA
fridaysecuritytask
Dockerfile
fridaydemo1
Project
fridaysecuritytask C:\workspace\fridaysecuritytask
.idea
.mvn
src
target
.gitignore
Dockerfile
fridaysecuritytask.iml
HELP.md
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles
3: Find
4: Run
6: TODO
8: Services
Spring
Terminal
Java Enterprise
Event Log
IDE and Plugin Updates: IntelliJ IDEA is ready to update. (today 14:01)
25:1 CRLF UTF-8 4 spaces

```

#作者
MAINTAINER Alex <5030749@qq.com>

#系统编码
ENV LANG=C.UTF-8 LC_ALL=C.UTF-8

#声明一个挂载点，容器内此路径会对应宿主机的某个文件夹
VOLUME /tmp

#应用构建成功后的jar文件被复制到镜像内，名字也改成了app.jar
ADD target/fridaysecuritytask-0.0.1-SNAPSHOT.jar app.jar

#启动容器时的进程
ENTRYPOINT ["java","-jar","/app.jar"]

#暴露8080端口
EXPOSE 8080

```

启动运行成功，使用浏览器访问：http://宿主机IP:8081，如果是阿里云等云服务，记得在安全组规则中增加8081端口.

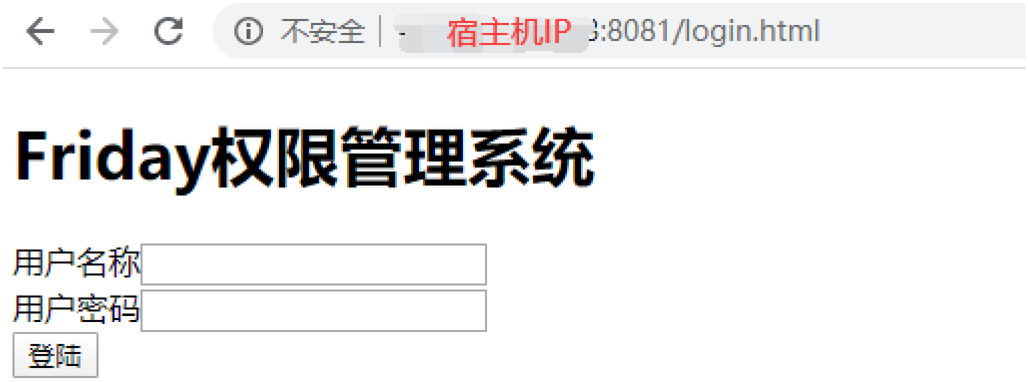


```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.3.0.RELEASE)

2020-05-25 09:20:23.288  INFO 1 --- [           main] c.s.f.FridaysecuritytaskApplication      : Starting FridaysecuritytaskApplication v0.0.1-SNAPSHOT on 8cf1fe799a37 w
2020-05-25 09:20:23.296  INFO 1 --- [           main] c.s.f.FridaysecuritytaskApplication      : No active profile set, falling back to default profiles: default
2020-05-25 09:20:27.413  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2020-05-25 09:20:27.455  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2020-05-25 09:20:27.456  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.35]
2020-05-25 09:20:27.652  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2020-05-25 09:20:27.652  INFO 1 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 4200 ms
2020-05-25 09:20:29.630  INFO 1 --- [           main] o.s.s.web.DefaultSecurityFilterChain     : Creating filter chain: any request, [org.springframework.security.web.co
2020-05-25 09:20:29.958  INFO 1 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2020-05-25 09:20:30.260  INFO 1 --- [           main] o.s.b.a.w.s.WelcomePageHandlerMapping    : Adding welcome page template: index
2020-05-25 09:20:30.675  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2020-05-25 09:20:30.722  INFO 1 --- [           main] c.s.f.FridaysecuritytaskApplication      : Started FridaysecuritytaskApplication in 8.831 seconds (JVM running for
```



访问规则

授权策略	优先级 ⓘ	协议类型	端口范围 ⓘ	授权对象 ⓘ	描述	操作
允许	1	自定义 TCP	目的:8081/8081	源:0.0.0.0/0		编辑 复制 删除
允许	1	自定义 TCP	目的:2375/2375	源:0.0.0.0/0		编辑 复制 删除
允许	1	自定义 TCP	目的:6379/6379	源:0.0.0.0/0		编辑 复制 删除
允许	1	自定义 TCP	目的:12345/12345	源:0.0.0.0/0		编辑 复制 删除
允许	1	自定义 TCP	目的:3306/3306	源:0.0.0.0/0		编辑 复制 删除
允许	1	自定义 TCP	目的:9090/9090	源:0.0.0.0/0		编辑 复制 删除
允许	1	自定义 TCP	目的:5000/5000	源:0.0.0.0/0		编辑 复制 删除
允许	1	自定义 TCP	目的:8080/8080	源:0.0.0.0/0		编辑 复制 删除
允许	110	全部 ICMP(IPv4)	目的:-1/-1	源:0.0.0.0/0	System created rule.	删除
允许	110	自定义 TCP	目的:22/22	源:0.0.0.0/0	System created rule.	删除
允许	110	自定义 TCP	目的:3389/3389	源:0.0.0.0/0	System created rule.	删除



不安全 | - [redacted] 宿主机IP [redacted]:8081/login.html

Friday权限管理系统

用户名称

用户密码