如果不需要自己编译打包后台服务，直接跳到第5章。

1. 准备工作

1.1 application.properties配置文件中增加下面配置行：

# localhost  
spring.profiles.active=localhost

1.2 生成1个密钥文件

keytool -genkey -alias tomcat -keyalg RSA

默认得到.keystore文件，改名为keystore，拷贝到java工程的resoures目录下。

1.3复制一份application-localhost.properties文件，比如从application-myvm.properties文件复制。

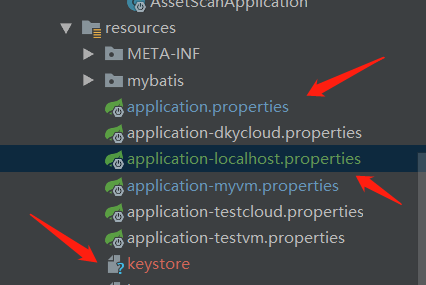
修改要调用其他服务的ip地址为localhost（这个修改是为了避免在实际部署前重新编译打包，可以直接使用已打包的jar文件）

service.ip.default=localhost

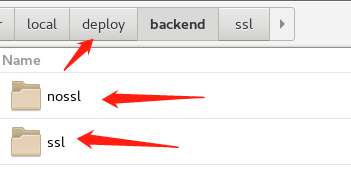
1.4 在application-localhost.properties文件中添加如下配置

# SSL 配置  
#你生成的证书名字  
server.ssl.key-store=classpath:keystore  
#密钥库密码  
server.ssl.key-store-password=12345678  
#server.ssl.keyStoreType=PKCS12  
server.ssl.keyStoreType=JKS  
server.ssl.keyAlias=tomcat  
server.ssl.enabled=**true**

下图是配置文件和密钥文件的示意



1.5 服务器deploy目录下创建两个目录，目录名称自行决定，本文中这两个目录采用ssl和nossl。

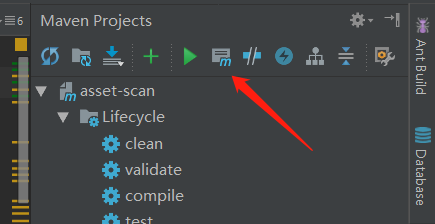


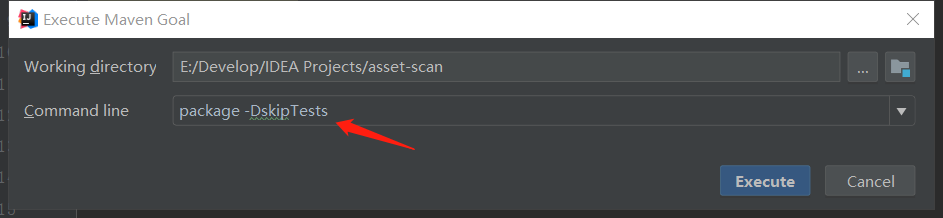
1. Java服务Https协议

在上述的ssl配置生效的前提下，采用localhost配置进行打包。

因为localhost配置中明确了其他服务地址是localhost，常规打包时测试失败，采用自定义maven打包命令：

package -DskipTests





将打包后的文件拷贝到虚拟机的ssl目录下。

所有的java后端服务都采用此方式准备好部署文件。

1. Java服务Http协议

将上述application-localhost.properties配置文件中ssl相关配置注释，仍然使用package –DskipTests打包。并将打包后的文件拷贝到虚拟机的ssl目录下。

1. Django服务

4.1 生成密钥和证书

在virtualenv的虚拟环境pyweb目录下，进入到wwwroot/deploy子目录，用openssl生成密钥和请求证书：

openssl genrsa -out edbserver.key 2048

openssl req -new -key edbserver.key -out edbserver.csr （生成请求文件时，密码设定时直接回车）

openssl x509 -req -days 365 -in edbserver.csr -signkey edbserver.key -out edbserver.crt

4.2 修改common/config.py和common/syslog.py的数据库服务IP地址

g\_mongo\_client = pymongo.MongoClient("mongodb://admin:123456@192.168.182.88:27017/")

mysql\_db = pymysql.connect(host='192.168.182.88', user='root', password='123456', db='asset\_scan', port=13306)

如上述例子，将192.168.182.88换成实际mongo部署环境下的IP。或者由于全部服务部署在一个环境，也可改成127.0.0.1

1. 准备前端包

5.1 前端Https协议

前端代码中，src/global/environment/index.js，设置：

const PROTOCOL = 'https';

并检查对应的后端服务IP是否正确，比如：

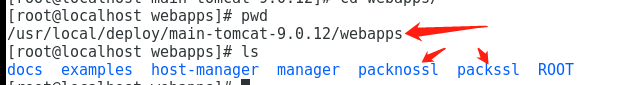
const BASE\_URL = '://192.168.182.88:' + MAIN\_S\_PORT + '/';

export const BASE\_URL2 = '://192.168.182.88:' + EDB\_PORT;

打包

npm run build

存放打包文件到tomcat的webapps/packssl目录下，比如：



5.2 前端http协议

前端代码中，src/global/environment/index.js，设置：

const PROTOCOL = 'https';

并检查对应的后端服务IP是否正确，参考5.1节中图。

打包，存放打包文件到tomcat的webapps/packnossl目录下。

5.3 SSL密钥文件

复制1.2节中的密钥文件keystore到tomcat的conf目录下。

5.4 tomcat两种协议的配置

在tomcat的conf目录下，修改server.xml文件，使其包含以下两个配置：

（部署运行时二选一，另一个注释掉）

http协议：

<Connector port="8080" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" />

https协议：

<Connector

protocol="org.apache.coyote.http11.Http11NioProtocol"

port="8080" maxThreads="200"

scheme="https" secure="true" SSLEnabled="true"

keystoreFile="conf/keystore" keystorePass="12345678"

clientAuth="false" sslProtocol="TLS"/>

1. 部署切换到https协议

6.1 杀掉两个Java后端服务进程

ps -ef | grep java

kill -9 [pid]

6.2 运行两个Java后端服务

进入ssl协议后端包目录下，比如：

cd /usr/local/deploy/backend/ssl

后台模式运行：

nohup java -jar asset-scan-0.0.1-SNAPSHOT.jar &

nohup java -jar scan-task-manage-0.0.1-SNAPSHOT.jar &

6.3 关闭python的uwsgi服务

进入虚拟环境：

cd /usr/local/deploy/

source pyweb/bin/activate

关闭现有服务：

cd /usr/local/deploy/pyweb/wwwroot/

uwsgi --stop deploy/uwsgi.pid

6.4 运行新的uwsgi服务

nohup uwsgi --ini deploy/uwsgissl.ini &

6.5 启动前端

进入到tomcat的webapps目录下，删除ROOT下的所有文件，并把packssl下的所有文件拷贝到ROOT下。

打开tomcat的conf/server.xml，保留https协议配置，注释掉http协议配置。

重启tomcat:

bin/shutdown.sh

bin/startup.sh

6.6 远程访问：

在用浏览器打开网页前，用浏览器访问下面两个链接：

<https://192.168.182.88:8090/api/sysinfo/version?sys_type=1>

<https://192.168.182.88:10091/cnvd/>

1. 部署切换到http协议

7.1 切换到http协议Java后端服务进程

参考6.1和6.2，唯一区别是进入nossl目录。

cd /usr/local/deploy/backend/nossl

7.2 切换到http协议的uwsgi服务

参考6.3和6.4，唯一区别是命令中配置文件为uwsgi.ini

nohup uwsgi --ini deploy/uwsgi.ini &

7.3 启动前端

进入到tomcat的webapps目录下，删除ROOT下的所有文件，并把packnossl下的所有文件拷贝到ROOT下。

打开tomcat的conf/server.xml，保留http协议配置，注释掉https协议配置。

重启tomcat。