

使用系统服务管理 spring boot 程序

背景:

spring boot 架构的 java 项目，一般的启动方式类似如下：

```
nohup java -jar -Xmx1024M /rebuild/louxe-open-api.jar --server.port=8083 >
/rebuild/out-open-api.log &
```

停止程序，一般使用 kill pid

但其实也支持使用系统服务的方式进行管理，这样就可以使用以下方式进行更加优雅的管理程序：

```
serveice louxe-open-api start|stop|restart|status
systemctl start|stop|restart|status louxe-open-api.service
```

官方文档：

<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#cloud-deployment-cloud-foundry-services>

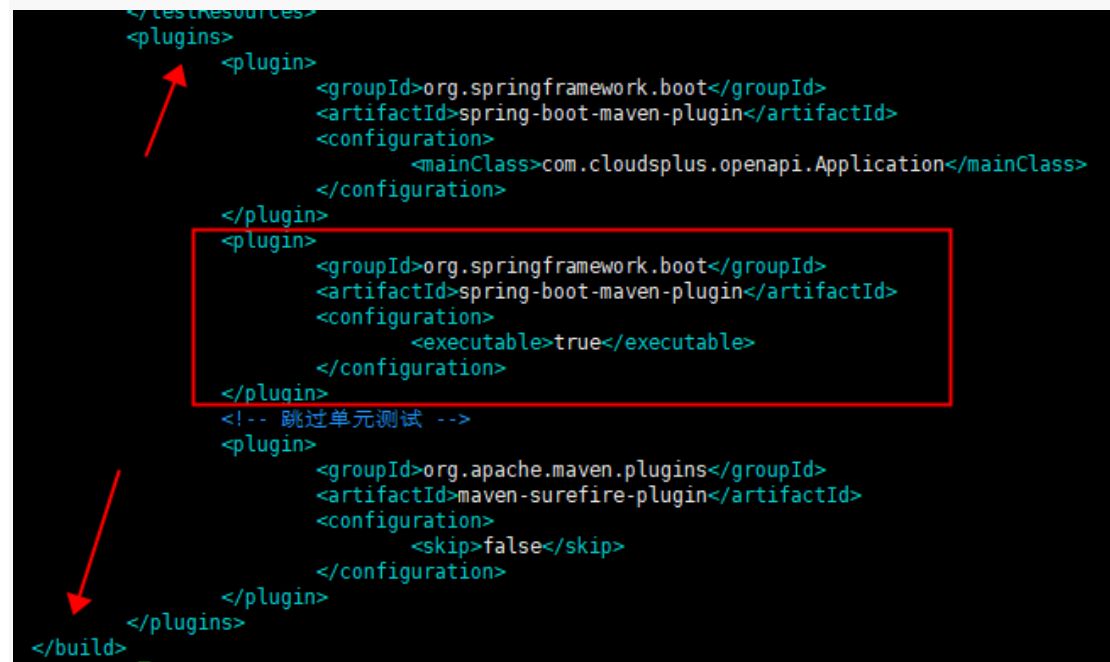
在第 61 小节

一、打包前配置

项目编译前的 pom.xml 文件中，在<plugins>段落添加以下内容，同其他插件的顺序无要求

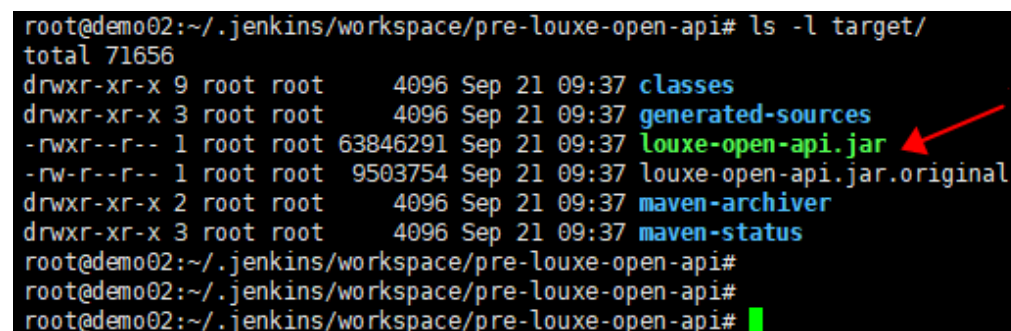
```
<plugin>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-maven-plugin</artifactId>
  <configuration>
    <executable>true</executable>
  </configuration>
</plugin>
```

比如：



```
<!-- testResources -->
<plugins>
  <plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <configuration>
      <mainClass>com.cloudsplus.openapi.Application</mainClass>
    </configuration>
  </plugin>
  <plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <configuration>
      <executable>true</executable>
    </configuration>
  </plugin>
  <!-- 跳过单元测试 -->
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <configuration>
      <skip>false</skip>
    </configuration>
  </plugin>
</plugins>
</build>
```

添加这个后，我们发现打包出的 jar 包自动就有了 x 权限



```
root@demo02:~/jenkins/workspace/pre-louxe-open-api# ls -l target/
total 71656
drwxr-xr-x 9 root root    4096 Sep 21 09:37 classes
drwxr-xr-x 3 root root    4096 Sep 21 09:37 generated-sources
-rwxr--r-- 1 root root 63846291 Sep 21 09:37 louxe-open-api.jar
-rw-r--r-- 1 root root 9503754 Sep 21 09:37 louxe-open-api.jar.original
drwxr-xr-x 2 root root    4096 Sep 21 09:37 maven-archiver
drwxr-xr-x 3 root root    4096 Sep 21 09:37 maven-status
root@demo02:~/jenkins/workspace/pre-louxe-open-api#
root@demo02:~/jenkins/workspace/pre-louxe-open-api#
root@demo02:~/jenkins/workspace/pre-louxe-open-api#
```

其原理大致就是会将一些内容嵌入到 jar 里面，使 jar 能够使用系统服务进行管理，嵌入的内容有如下：

名称	描述
mode	脚本模式。默认为 auto。
initInfoProvides	在 Provides “INIT INFO” 的部分。默认 spring-boot-application 为 Gradle 和 \${project.artifactId}Maven。
initInfoRequiredStart	在 Required-Start “INIT INFO” 的部分。默认为 \$remote_fs \$syslog \$network。
initInfoRequiredStop	在 Required-Stop “INIT INFO” 的部分。默认为 \$remote_fs \$syslog \$network。
initInfoDefaultStart	在 Default-Start “INIT INFO” 的部分。默认为 2 3 4 5。
initInfoDefaultStop	在 Default-Stop “INIT INFO” 的部分。默认为 0 1 6。
initInfoShortDescription	在 Short-Description “INIT INFO” 的部分。默认 Spring Boot Application 为 Gradle 和 \${project.name}Maven。
initInfoDescription	在 Description “INIT INFO” 的部分。对于 Maven，默认 Spring Boot Application 为 Gradle 和 \${project.description}（回退 \${project.name}）。
initInfoChkconfig	在 chkconfig “INIT INFO” 的部分。默认为 2345 99 01。
confFolder	默认值 CONF_FOLDER。默认为包含 jar 的文件夹。
inlinedConfScript	引用应在默认启动脚本中内联的文件脚本。这可用于设置环境变量，例如 JAVA_OPTS 在加载任何外部配置文件之前。
logFolder	的默认值 LOG_FOLDER。仅对 init.d 服务有效。
logFilename	的默认值 LOG_FILENAME。仅对 init.d 服务有效。
pidFolder	的默认值 PID_FOLDER。仅对 init.d 服务有效。
pidFilename	中的 PID 文件名的默认值 PID_FOLDER。仅对 init.d 服务有效。
useStartStopDaemon	start-stop-daemon 命令是否可用，应该用于控制过程。默认为 true。
stopWaitTime	的默认值 STOP_WAIT_TIME。仅对 init.d 服务有效。默认为 60 秒。

从表中可以看出，有些参数是可以通过通过配置文件传入的，参考第二、第三节内容。当然这些内容，我们也可以通过 embeddedLaunchScript 自定义，但大多数情况下，不需要。

二、部署前配置文件

将打包后的 jar 包放到目标机器，比如路径是 /rebuild/louxe-open-api/louxe-open-api.jar

添加配置文件，该配置文件默认和 jar 位于同一路径下，且名字于 jar 的名字相同，后缀是 conf, 如/rebuild/louxe-open-api/louxe-open-api.conf

内容例如：

```
JAVA_HOME=/usr/local/java
LOG_FOLDER=/rebuild/louxe-open-api
PID_FOLDER=/rebuild/
JAVA_OPTS="-Xms256M -Xmx1024M "
RUN_ARGS="--server.port=8083"
```

其中 PID_FOLDER、LOG_FOLDER 这两个个参数只适合 init.d 管理的服务，对于 systemd 管理的服务，参考 3.2 节进行配置。

需要注意的是 PID_FOLDER 参数，比如配置为 PID_FOLDER=/rebuild/ 则 pid 文件的路径是：/rebuild/louxe-open-api/louxe-open-api.pid，且属主是 root

以上是最常用的几个参数，当然也有很多其他参数，如有需要可以配置，如下（摘抄于官方文档）

变量	描述
MODE	操作的“模式”。默认值取决于 jar 的构建方式，但通常是 auto（意味着它通过检查它是否是所调用目录中的符号链接来尝试猜测它是否是 init 脚本 init.d）。您可以显式设置它以 service 使 stop start status restart 命令有效，或者 run 如果要在前台运行脚本。
USE_START_STOP_DAEMON	start-stop-daemon 命令是否可用，应该用于控制过程。默认为 true。
PID_FOLDER	pid 文件夹的根名称（/var/run 默认情况下）。
LOG_FOLDER	用于放置日志文件的文件夹的名称（/var/log 默认情况下）。
CONF_FOLDER	从中读取.conf 文件的文件夹的名称（默认情况下与 jar 文件相同）。
LOG_FILENAME	LOG_FOLDER（<appname>.log 默认情况下）日志文件的名称。
APP_NAME	应用程序的名称。如果 jar 从符号链接运行，则脚本会猜测应用程序名称。如果它不是符号链接或您想要显式设置应用程序名称，这可能很有用。

RUN_ARGS	传递给程序的参数（Spring Boot 应用程序）。
JAVA_HOME	默认情况下 java 使用 PATH 默认值发现可执行文件的位置，但如果存在可执行文件，则可以显式设置它\$JAVA_HOME/bin/java。
JAVA_OPTS	启动时传递给 JVM 的选项。
JARFILE	jar 文件的显式位置，以防脚本用于启动实际上未嵌入的 jar。
DEBUG	如果不为空，则-x 在 shell 进程上设置标志，以便于在脚本中查看逻辑。
STOP_WAIT_TIME	在强制关闭之前停止应用程序时等待的时间（60 默认情况下）。

三、配置成系统服务

3.1 使用 init 进行管理

Ubuntu14 及之前、CentOS/RedHat6 及之前，都是用 init 进行服务管理

```
ln -sv /rebuild/louxe-open-api/louxe-open-api.jar /etc/init.d/louxe-open-api
```

#创建系统服务，通过软连接的形式，服务的名字随意写

service louxe-open-api start|stop|restart|status 等均可以使用

当然也可以将其加入开机启动等。

注意：

- 1) 项目启动用户以 jar 文件的属主为主
- 2) 建议使用非 root 用户运行，这也是官方强烈建议的，同时也是运维人员的常识
- 3) 非 root 运行，保证日志文件属主正确，比如

```
chown lx2.lx2 out-open-api.log
```

同时如果使用 ansible 进行部署的话，为了减少报错，需要使用 file 模块提前确认此文件的权限。

3.2 使用 systemd 进行管理

CentOS/Redhat7 及之后、Ubuntu16 及之后使用 systemd 管理服务

添加如下文件：

```
[root@iZuf61fwacpba8u3ty82udZ .ssh]# cat /etc/systemd/system/louxe-open-api.service
```

```
[Unit]
```

```
Description=louxe-open-api service
```

```
After=syslog.target
```

```
[Service]
```

```
User=lx2
```

```
ExecStart=/rebuild/louxe-open-api/louxe-open-api.jar
```

```
SuccessExitStatus=143
```

```
[Install]
```

```
WantedBy=multi-user.target
```

不同的项目需要修改以下三项：

- 1)Description: 描述信息，随意写
- 2)ExecStart: 可执行文件的路径
- 3)User: 启动服务的用户，建议非 root

然后就可以使用

```
systemctl start|stop|restart|status louxe-open-api[.service]
```

进行管理

注意：

- 1) 项目启动用户以 User 参数为准，同时注意 jar 文件的属主也要是 User
- 2) 建议使用非 root 用户运行，这也是官方强烈建议的，同时也是运维人员的常识
- 3) 非 root 运行，保证日志文件属主正确，比如
同时如果使用 ansible 进行部署的话，为了减少报错，需要使用 file 模块提前确认此文件的权限。
- 4) 使用 systemd 管理服务，控制台日志将会被 journal 统一管理，二进制形式，默认保存到/var/log/journal/目录。另外也没有 pid 文件。

四、附录-ansible 脚本

4.1 使用 init 管理服务

此时服务的管理,应该使用 service 模块,程序用户 lx2,未写备份的步骤,需要时可以添加:

```
---
- hosts: pre-1
  remote_user: root
  vars:
    - tdocbasedir: /rebuild
    - projectname: louxe-open-api.jar
  tasks:
    - name: 1.stop jar
      service:
        name=louxe-open-api
        state=stopped
        ignore_errors: yes

    - name: 2.sync file to dest {{tdocbasedir}}
      copy:
        src=/root/.jenkins/workspace/pre-louxe-open-api/target/{{projectname}}
        dest={{tdocbasedir}}
        mode=0744
        owner=lx2
        group=lx2

    - name: 3.restart jar
      service:
        name=louxe-open-api
        state=restarted
...
```

4.2 使用 systemd 管理服务

此时应该使用 ansible 的 systemd 模块,程序用户 lx2,未写备份的步骤,需要时可以添加。

```
---
- hosts: pre-1
  remote_user: root
  vars:
```



```
- tdocbasedir: /rebuild
- projectname: louxe-open-api.jar
tasks:
- name: 1.stop jar
  systemd:
    name=louxe-open-api
    state=stopped
    ignore_errors: yes

- name: 2.sync file to dest {{tdocbasedir}}
  copy:
    src=/root/.jenkins/workspace/pre-louxe-open-api/target/{{projectname}}
    dest={{tdocbasedir}}
    mode=0744
    owner=lx2
    group=lx2

- name: 3.restart jar
  systemd:
    name=louxe-open-api
    state=restarted
...
```