

Jenkins 部署和踩坑总结

作者: CodingOX

- 原文链接: https://ld246.com/article/1612765490260
- 来源网站:链滴
- 许可协议: 署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)



基础安装

下载 Tomcat, 目前不要选择 10+的版本,选择 9 比较好,下载地址:链接

下载 Jenkins, 推荐下载Generic Java package (.war), 下载地址: 链接

然后将下载后的 jar 包部署到 Tomcat 的 webapps 目录下,然后重启。

Tomcat 的启动和关闭可能会涉及到权限问题,记得授权。

启动后,通过http://ip:8080/jenkins/进行访问,第一次速度有一些慢,然后就是获取密码,创建新 户。

插件推荐

包括如下内容:

- Maven 相关插件: Maven Integration plugin
- Git插件: Git plugin
- ●环境变量插件: Environment Injector
- 远程部署: Publish Over SSH
- 通用的 Webhook插件: Generic Webhook Trigger Plugin
- BlueOcean 服务编排: Blue Ocean 和Blue Ocean Pipeline Editor

配置相关



- Jenkins Location这个地方建议修改为: http请求和补充自己的email
- Git glugin中的user.name和user.email修改
- •邮件通知建议使用QQ邮箱, 163很容易识别为垃圾邮箱。

● Publish over ssh如果有用到,请在这里设置,通过证书或者密码登录都是可以的,设置完成后, 击Test Configuration测试下。

311 361 461		
Name	vm-162	
Hostname	192.168.0.162	
Username	root	•
Remote Directory	1	•
Use password auth	entication, or use a different key	
Passphrase / Passwor	Concealed	Change Password
Path to key		•
Key		•
	_	
Jump host		

通用配置

系统管理-> 全局通用配置 中需要配置: Maven、JDK、Docker 等。下面通过 JDK 举例,其他按照 己的实际情况配置,但都请不要勾选"自动安装"。

JDK 安装	新增 JDK		
	JDK		
	别名	adopt11	
	JAVA_HOME	/usr/java/jdk-11.0.9.1+1	
	□ 自动安装	取消	(
			删除 JDK

其他的类似。





选择新建一个 maven 项目:

hello	-maven
» 必填	项
	构建一个自由风格的软件项目
61	
	这是Jenkins的主要功能.Jenkins将会结合任何SCM和任何构建系统来构建你的项目, 甚
	这是Jenkins的主要功能.Jenkins将会结合任何SCM和任何构建系统来构建你的项目, 表 构建一个maven项目

然后填写git的相关信息

Repositories	Paracilary URI http://000.000.0.000.0000/seat/dealers.eller	
	Kepository OKL http://iaz.io6.0.ioz.3000/roof/docker-gitea	
	Credentials root/****** (gitea 85) *	0
	Add	席级 Repository
Branches to build	指定分支(为空时代表any) */master	× 0
		增加分支
源码库浏览器	(自动)	~ 6

那么这个流程何时开始编译,此时就需要通过 git项目的 webhook 了,下面这个选项是插件:Generi Webhook Trigger Plugin带来的。

构建触发器



只需要填写 token, 然后webhook 的时候发送相关指令即可, 发送格式写的很清楚

• 通过带参数的形式: http://192.168.0.161:8080/jenkins/generic-webhook-trigger/invoke?to en=hello

● 通过 header的形式等

然后设置 maven 的构建指令以及运行

B clean package -Dmaven.test.skip=true -Dautoconfig.skip Run only if build succeeds Run only if build succeeds or is unstable Run regardless of build result
B clean package -Dmaven.test.skip=true -Dautoconfig.skip Image: Second state in the second state is t
Run only if build succeeds O Run only if build succeeds or is unstable O Run regardless of build result
hould the post-build steps run only for successful builds, etc.
/docker-hi-0.0.2.jar /home/admin/docker-hi/docker-hi-0.0.2.jar -temp.sh /home/admin/docker-hi/deploy-temp.sh dmin/docker-hi/deploy-temp.sh docker-hi-0.0.2 restart

强调几点:

- maven的指令,就是上面的Goals and options的开头不能提交mvn
- Post Steps表示的构建完成后的步骤,通常是进行项目启动

最后可以通过 email 进行消息通知。

远程部署

基本设置

通常项目不仅要部署到本机,还可能需要部署其他主机,此时有 2 种办法,一种是通过主机信任后通 lscpl传输,一种是通过插件lPublish over sshl,我们先聊聊后者。

在^{II}Post Steps^{II}中新增^{II}Send files or execute commands over SSH^{II},如下所示:



然后可以看到下图:

SH Publishers	SSH Server			
	Name	vm-162		
				离级
	Transfers	Transfer Set		
		Source files] 6
		Remove prefix	Either Source files, Exec command or both must be supplied	
		Remote directory		
		Exec command		•
			Either Source files, Exec command or both must be supplied All of the transfer fields (excent for Exec timeout) support substitution of Jenkins environment variables	
		Add Transfer Set		

假如你没有 SSH Server 的设置,那是因为你没有在全局设置中进行配置,可以参考我上面¹插件推荐 中的最后配置

然后输入如下内容:

ame	vm-162		
			高级
ransfers	Transfer Set		
	Source files	target/*.jar,deploy-jenkins.sh,deploy-temp.sh	•
	Remove prefix		•
	Remote directory	/home/admin/docker-hi/	
	Exec command	sh /home/admin/docker-hi/deploy-temp.sh docker-hi-0.0.2 restart	
		All of the transfer fields (except for Exec timeout) support substitution of Jenkins environment variables	
	Exclude files		
	Pattern separator	[,]+	
	No default excludes		
	Make empty dirs		
	Flatten files		
	Clean remote	0	
	Remote directory		

● □ Sourec File□表示的是你要传输的文件,这个路径其实就是相对你项目的路径。□Jenkins□会从□Git 上拉取文件,然后在自己的 □workspace□中进行编译。□

- I Remove prefix 表示的去除前缀,这里因为我勾选了Flattern files 所以不需要去除前缀
- Remote Directory 表示的远程主机上的目录
- I Exec command□表示的是远程主机上执行的命令,通常是通过脚本进行启动、关闭、重启等□
- I Flattern files□表示的去除前缀,就是将只传输文件到□Remote Director□,不带文件前缀

脚本文件

这个脚本文件,是最大的坑,查资料,卡了半天,总算再查另外一个问题的时候,查到了解决方案。

你可能会遇到问题:无法通过Jenkins 的这个插件使用脚本启动远程项目。同时你的日志中可能报错: RROR: Exception when publishing, exception message [Exec exit status not zero. Status [1]]

其实这个问题可能是因为:Jenkins 启动任务完成后,会关闭工作进程启动的子进程,所以远程任务 关闭了【可能】

所以在写脚本的时候,需要注意3个细节:

- 文件开始加入: □ source /etc/profile□
- 然后设置参数: □ export BUILD_ID=dontkillme□

● 最后在获取进程的时候, 排除到 □ Jenkins□, 比如我们 Java 应用这么写: □\$(ps -ef | grep -w "\$S RVICE_NAME" | grep "java" | awk '{print \$2}')□

```
下面给出一个案例脚本:
#!/bin/bash
# 搭配使用才行!
source /etc/profile
# jenkins 使用
export BUILD ID=dontkillme
## exec shell name
EXEC SHELL NAME=$1\.sh
## service name
SERVICE NAME=$1
SERVICE DIR=/home/admin
JAR NAME=$SERVICE NAME\.jar
PID=$SERVICE NAME\.pid
WORK DIR=$SERVICE DIR/docker-hi
mkdir -p $WORK DIR
#function start
start() {
 cd $WORK DIR
 if [ ! -d "log" ]; then
  mkdir log
 fi
# nohup java -Xms256m -Xmx512m -jar $JAR NAME >log/$SERVICE NAME.out 2>&1 &
 nohup java -Xms256m -Xmx512m -jar $JAR NAME >log/$SERVICE NAME.out 2>&1 &
 echo $! >$WORK DIR/$PID
 echo "#### start $SERVICE NAME"
}
# function stop x
stop() {
 cd $WORK DIR
 if [ -f "$WORK DIR/$PID" ]; then
  kill $(cat $WORK DIR/$PID)
  rm -rf $WORK DIR/$PID
 fi
 echo "#### stop $SERVICE NAME"
 sleep 6
 TEMP PID=$(ps -ef | grep -w "$SERVICE NAME" | grep "java" | awk '{print $2}')
 if [ "$TEMP PID" == "" ]; then
  echo "#### $SERVICE NAME process not exists or stop success"
 else
  echo "#### $SERVICE NAME process pid is:$TEMP PID ."
  kill -9 $TEMP PID
 fi
}
# function clean
clean() {
 cd $WORK DIR
 if [ ! -d "lastDeploy" ]; then
```

```
mkdir lastDeploy
 else
  rm lastDeploy/$SERVICE_NAME*
 fi
 if [ -f "$JAR NAME" ]; then
  mv $JAR NAME lastDeploy
 fi
}
case "$2" in
start)
 start
 ;;
stop)
 stop
;;
restart)
 stop
 sleep 2
 start
 echo "#### restart $SERVICE NAME"
 ;;
clean)
 stop
 sleep 2
 clean
 echo "#### clean $SERVICE NAME"
 ;;
*)
 ## restart
 stop
 sleep 2
 start
 ;;
esac
exit 0
```

Blue Ocean

安装了插件Blue Ocean 和 Blue Ocean Pipeline Editor在项目面板的左边,会出现选项:打开 Blue cean

Dashboard -
🖀 新建任务
🍓 用户列表
🦻 构建历史
🔍 项目关系
🏭 检查文件指纹
💮 系统管理
🍓 我的视图
● 打开 Blue Ocean
Lockable Resources
新建视图

可以选择新建流水线

Jenkins		流水线	化加加加	Ð	
流水线 Q Search pipelines					创建流水线
收藏					
🗸 docker-gitea	۲ master		③ 40 minutes a	go	*
名称		健康状态	分支	PR	
docker-gitea		-	1个通过	343	*

其中第一步就是选择 git 仓库:

	选择代码仓库	
	Bitbucket Cloud	Bitbucket Server
	GitHub	GitHub Enterprise
	🚯 Git	
0	连接到 Git 仓库	
T	注意:任何包含 Jenkinsfile 的仓库将	会自动构建 更多 Jenkinsfile 的介绍。
	仓库 URL	
	http://192.168.0.162:3000/root/do	ocker-gitea
	Jenkins needs a user credential to aut Use existing credential: root/**** http://192.168.0.162:3000/root/ Create new credential 用户名 密码 创建证书 	thorize itself with git. ** (Git username/password for docker-gitea)
	创建流水线	

输入地址后,会要求是使用已经创建好的凭证还是创建一个新的凭证,这个随你。 然后会要求你输入这个流水线的名称,比如blue,创建完成后,跳转到该界面

🔵 blue	☆ ♦						活动	分支	Pull Requests
状态	這行	提交	分支	Ŧ	消息	神中國國	完」	æ	
\odot	1	-	master		Branch indexing	8	•÷)		۲

简单说下上述面板的几个意思:

- 活动:表示的每一次构建任务的情况。
- 分支:这个概念其实就是 git的分支概念,本质上你可视化流程的操作,最终会在项目根路径下形 一个文档jenkinsfile,并通过git进行管理。

此时选择分支,然后将鼠标移动到 master分支,选择后面的笔表示。

ocinana					and the second se	
🤶 blue י	☆ ‡				活动	分支 Pull Requests
健康状态	状态	分支	推交	最近一次消息	完成	
٠	0	master	鼠标移动到这里来	Branch indexing	3 minutes ag	go ④④/☆

接下来的流程就是创建,点击+进行流程创建,基本都是shell脚本,创建好后如下所示:

Start

点击确定后提交到 git仓库。

最终生成的文件如下所示:

```
pipeline {
    agent any
    stages {
        stage('git') {
            steps {
                git(url: 'http://192.168.0.162:3000/root/docker-gitea.git', branch: 'master', credentialsId: '
            itea')
            }
        }
        stage('maven') {
            steps {
            stage('maven') {
                steps {
                steps {
                sh 'mvn -B clean package -Dmaven.test.skip=true -Dautoconfig.skip'
            }
        }
        }
    }
}
```

```
stage('deploy') {
   parallel {
    stage('remote deploy-1') {
      steps {
       sh "scp target/*.jar root@192.168.0.162:/home/admin/docker-hi/docker-hi-0.0.2.jar
ssh root@192.168.0.162 \'bash -s\' < /home/admin/docker-hi/deploy-temp.sh docker-hi-0.0
2 restart'''
     }
    }
    stage('local deploy') {
      steps {
       sh "\\cp target/docker-hi-0.0.2.jar /home/admin/docker-hi/docker-hi-0.0.2.jar
\\cp deploy-temp.sh /home/admin/docker-hi/deploy-temp.sh
sh /home/admin/docker-hi/deploy-temp.sh docker-hi-0.0.2 restart'''
      }
    }
   }
  }
 }
}
```

Linux互信

严格来讲这篇文章不算是 IJenkinsI 相关内容, 但是我认为这个蛮关键的, 所以这里记录下。

为了远程部署方便,除了通过^{II}Over SSHII插件,还有一个办法就是II通过将IIJenkinsII所在主机的公钥写入目标机器的IIauthorized_keysIII

比如 [Jenkins]所在的机器是[192.168.0.161], 目标机器是[192.168.0.162。

- 在□ Jenkins□所在的机器执行: □ssh-keygen -t rsa□, 默认情况下该指令会在□□~/.ssh□下生成私钥 公钥□id_rsa.pub□
- 在 Jenkins所在的机器执行: □cat ~/.ssh/id_rsa.pub□ 然后复制内容□``
- 『在目标机器』 192.168.0.162 『查看下 『~/.ssh 『是否有文件 『authorized_keys

● □□如果没有那么创建一个,如果有的话直接将第二步复制的内容追加其中,记得要在新的一行粘贴容。

然后你可以尝试在1610的主机尝试登陆1620

此时用户是默认的本机 hostname ssh 192.168.0.162

上述指令其实是最简答的写法,严格的写法是加上对方主机的 [hostname]。

ssh hostname@192.168.0.162

上述流程配置好后,就可以通过^Iscp^I进行文件传输,比如这样

scp target/*.jar root@192.168.0.162:/home/admin/docker-hi/docker-hi-0.0.2.jar

也可以在远程主机执行本地的脚本

ssh root@192.168.0.162 'bash -s' < /home/admin/docker-hi/deploy-temp.sh docker-hi-0.0.2 restart

当然也可以在远程主机执行远程的脚本

ssh root@192.168.0.162 /home/admin/docker-hi/deploy-temp.sh docker-hi-0.0.2 restart

区别就是不带 bash -s'