
基于 *Jenkins* *CICD* 的代码发布与回滚

目录

目录.....	2
文档信息	3
文档约定	3
案例概述	3
案例知识点	4
案例环境	5
基于 JENKINS+GIT+DOCKER 发布 JAVA 项目	5
1、配置 GIT 仓库.....	5
2、配置 DOCKER 服务器	6
3、部署镜像仓库.....	7
5、配置 JENKINS 环境	10
6、JENKINS 创建项目并发布测试	14
7、版本回滚.....	20
基于 JENKINS+GIT+ANSIBLE 发布 PHP 项目	24
1、部署 PHP 运行环境.....	24
2、安装 ANSIBLE 插件	25
3、上传 PHP 项目代码到 GIT 仓库	26
4、JENKINS 创建项目并发布测试	27

文档信息

文档作者	房佳亮
文档版本	Version1.0
文档版权	内部资料禁止传播
文档归类	Linux 运维架构师系列
系统环境	CentOS-7.X-x86_64
作者邮箱	crushlinux@163.com
修订信息	2021-04-19
技术交流	

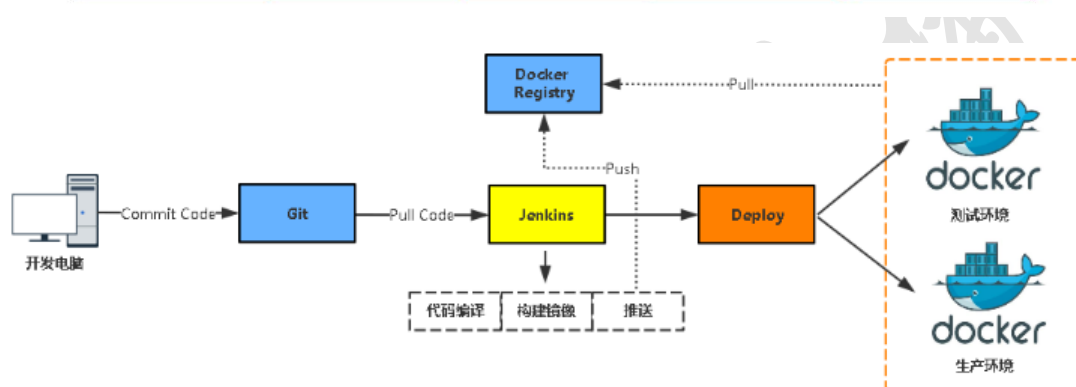
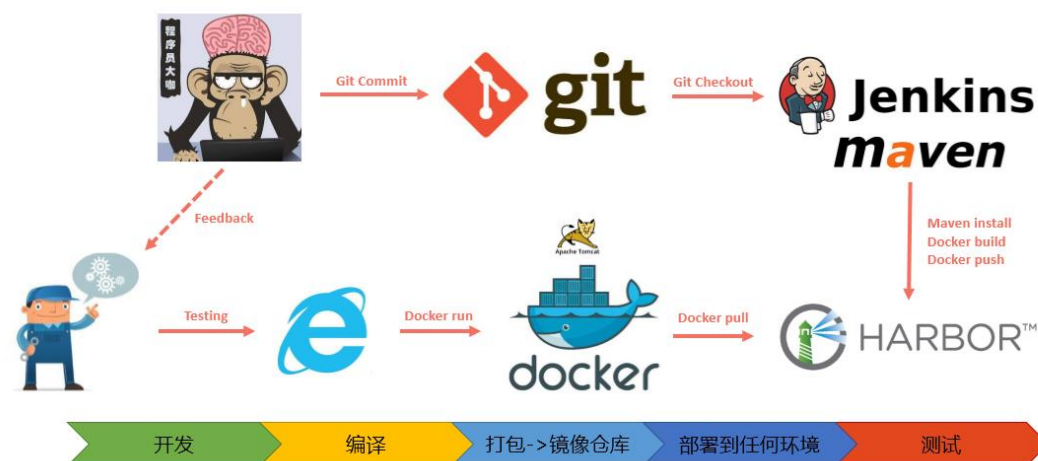
文档约定

[绿色背景]	知识重点
[红色背景]	错误警告
[黄色背景]	注意事项

执行命令

案例概述

在之前的 jenkins 持续集成章节中主要实现的是 jenkins 的项目构建及部署。那在本章将结合新项目来实现自动化构建及发布，并将项目打包成镜像上传到私有仓库，来实现一键发布和回滚等操作。



案例知识点

1、Jenkins 介绍

Jenkins，原名 Hudson，2011 年改为现在的名字，它是一个开源的实现持续集成的软件工具。官方网站：<https://jenkins.io/>。

Jenkins 能实施监控持续集成过程中所存在的问题，提供详细的日志文件和提醒功能，还能用图表的形式直观地展示项目构建的趋势和稳定性。

2、Maven 介绍

Maven 项目对象模型(Project Object Model，POM)，可以通过一小段描述信息来管理项目的构建，报告和文档的软件项目管理工具。

3、Docker 介绍

Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用及依赖包到任意一个可移植的镜像中，然后发布到任何流行的 Linux 机器上。

4、Ansible 介绍

Ansible 是一种自动化运维工具，基于 Python 开发，实现了批量系统配置、批量程序部署、批量运行命令等功能。

案例环境

操作系统	IP 地址	主机名	角色
CentOS7.5	192.168.200.111	git	Git/Docker registry
CentOS7.5	192.168.200.112	jenkins	Jenkins
CentOS7.5	192.168.200.113	docker	Docker

基于 Jenkins+Git+Docker 发布 Java 项目

所有主机上操作

```
[root@localhost ~]# setenforce 0
[root@localhost ~]# iptables -F
[root@localhost ~]# systemctl stop firewalld
[root@localhost ~]# systemctl stop NetworkManager
```

案例需求

开发者将开发完成的代码提交到 Git 代码版本仓库后，点击 Jenkins 任务按钮自动拉取代码编译构建，并自动部署到 Web 服务器，用户可访问最新项目版本。

1、配置 Git 仓库

Git 是一个开源的分布式版本控制系统，可以有效、高速的处理从很小到非常大的项目版本管理，是目前企业中使用最为广泛的代码版本管理系统。

1) 使用 yum 安装 Git 并配置

```
[root@git ~]# yum -y install git
[root@git ~]# useradd git
[root@git ~]# echo "123456" | passwd --stdin git
```

2) 配置 Jenkins 主机免交互拉取 Git 仓库

在 Jenkins 主机上创建密钥对，将 id_rsa.pub 内容追加到 Git 服务器上的 /home/git/.ssh/authorized_keys 文件中。

```
[root@jenkins ~]# ssh-keygen # 一路回车即可
[root@jenkins ~]# ssh-copy-id git@192.168.200.111
```

测试免交互登录

```
[root@jenkins ~]# ssh git@192.168.200.111
[git@git ~]$ exit
登出
Connection to 192.168.200.111 closed.
```

3) 在 Git 服务器创建 probe 版本仓库，一般对 Git 的规范的方式要以.git 为后缀。如下：

```
[root@git ~]# su - git
```

```
[git@localhost ~]$ mkdir probe.git
[git@localhost ~]$ cd probe.git
[git@localhost probe.git]$ git --bare init
初始化空的 Git 版本库于 /home/git/probe.git/
[git@localhost probe.git]$ exit
登出
```

4) 从 Github 拉取开源 Java 博客系统 psi-probe。

```
[root@jenkins ~]# git clone https://github.com/psi-probe/psi-probe.git
[root@jenkins ~]# cd psi-probe/
```

5) 移除旧的推送地址，添加新的 Git 提交地址。如下：

```
[root@jenkins psi-probe]# git remote remove origin
[root@jenkins psi-probe]# git remote add origin git@192.168.200.111:/home/git/probe.git
```

6) 提交代码到 Git 仓库并创建 Tag。如下：

```
[root@jenkins psi-probe]# touch psi-probe-web/src/main/webapp/a.html
[root@jenkins psi-probe]# git add .
[root@jenkins psi-probe]# git config --global user.email "crushlinux@163.com"
[root@jenkins psi-probe]# git config --global user.name "crushlinux"
[root@jenkins psi-probe]# git commit -m "a"
[master 5354df1] a
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 web/src/main/webapp/a.html
```

7) 创建标签：

```
[root@jenkins ~]# git tag 1.0.0
```

8) 推送到 Git 服务器。如下：

```
[root@jenkins ~]# git push origin 1.0.0
Counting objects: 28806, done.
Compressing objects: 100% (8010/8010), done.
Writing objects: 100% (28806/28806), 9.07 MiB | 18.02 MiB/s
Writing objects: 100% (28806/28806), 20.74 MiB | 18.02 MiB/s, done.
Total 28806 (delta 17127), reused 28777 (delta 17105)
To git@192.168.200.111:/home/git/probe.git
* [new tag]          1.0.0 -> 1.0.0
```

2、配置 Docker 服务器

1) 安装 Docker，在所有主机上操作

```
[root@localhost ~]# wget -O /etc/yum.repos.d/CentOS-Base.repo
http://mirrors.aliyun.com/repo/Centos-7.repo

[root@localhost ~]# yum -y install yum-utils device-mapper-persistent-data lvm2
[root@localhost ~]# yum-config-manager --add-repo
http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
```

```
[root@localhost ~]# ls /etc/yum.repos.d/
backup  Centos-aliyun.repo  CentOS-Media.repo  docker-ce.repo

[root@localhost ~]# yum -y install docker-ce
[root@localhost ~]# systemctl start docker
[root@localhost ~]# systemctl enable docker

[root@docker ~]# docker version
Client:
  Version:           18.09.6
  API version:        1.39
  Go version:         go1.10.8
  Git commit:         481bc77156
  Built:              Sat May  4 02:34:58 2019
  OS/Arch:            linux/amd64
  Experimental:       false

Server: Docker Engine - Community
Engine:
  Version:           18.09.6
  API version:        1.39 (minimum version 1.12)
  Go version:         go1.10.8
  Git commit:         481bc77
  Built:              Sat May  4 02:02:43 2019
  OS/Arch:            linux/amd64
  Experimental:       false/
```

2) git 主机配置阿里云镜像加速器

```
[root@docker ~]# cat << END > /etc/docker/daemon.json
{
    "registry-mirrors": [ "https://nyakyfun.mirror.aliyuncs.com" ]
}
END
[root@docker ~]# systemctl daemon-reload
[root@docker ~]# systemctl restart docker
```

3、部署镜像仓库

Docker Hub 作为 Docker 默认官方公共仓库；用户如果想自己搭建私有镜像仓库，官方提供了 registry 镜像，使其搭建私有仓库变的非常简单。

1) 在 git 部署 docker 私有仓库

```
[root@git ~]# docker run -d -v /opt/registry:/var/lib/registry -p 5000:5000 --restart=always
--name registry registry
[root@git ~]# docker ps -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
539aaa4a938b	registry	"/entrypoint.sh /etc..."	15 seconds ago
Up 13 seconds	0.0.0.0:5000->5000/tcp	registry	
[root@git ~]# docker images			
REPOSITORY	TAG	IMAGE ID	CREATED
SIZE			
registry	latest	f32a97de94e1	3 months ago
25.8MB			

2) 测试 registry 可用性

由于 Docker CLI 默认以 HTTPS 访问，而部署的 registry 并未提供 HTTPS，因此，需要在 pull 镜像的 Docker 主机添加 HTTP 可信任。

Docker 和 jenkins 主机上操作

```
[root@docker ~]# vim /etc/docker/daemon.json
{"insecure-registries":["192.168.200.111:5000"]}
[root@docker ~]# systemctl daemon-reload
[root@docker ~]# systemctl restart docker
```

3) 打标签并推送镜像到 registry

```
[root@docker ~]# cat centos-7-x86_64.tar.gz | docker import - centos:7
[root@docker ~]# docker tag centos:7 192.168.200.111:5000/centos:7
[root@docker ~]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED
SIZE			
192.168.200.111:5000/centos	7	8ca6d50d77c1	7 seconds ago
589MB			
centos	7	8ca6d50d77c1	7 seconds ago
589MB			

```

[root@docker ~]# docker push 192.168.200.111:5000/centos:7
The push refers to repository [192.168.200.111:5000/centos]
c664eb1dcbf0: Pushed
7: digest: sha256:e6a3608bb01ef56b61d985020c5b55aef135dd50501232ab6e6ca589e01f8700
size: 529

```

4) 查看 registry 上传的镜像:

```
[root@docker ~]# curl http://192.168.200.111:5000/v2/_catalog
{"repositories":["centos"]}
[root@docker ~]# curl http://192.168.200.111:5000/v2/centos/tags/list
{"name":"centos","tags":["7"]}
```

5) 从 registry 下载镜像:

```
[root@docker ~]# docker rmi 192.168.200.111:5000/centos:7
```



```
[root@docker ~]# docker pull 192.168.200.111:5000/centos:7
7: Pulling from centos
Digest: sha256:d855e3c2525a97f919850f211bb62096bb92d59b6891e4f4ab04cad3e5864bd1
Status: Downloaded newer image for 192.168.200.111:5000/centos:7
[root@localhost ~]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
192.168.200.111:5000/centos	7	5f618d4dd78d	About a minute ago	589MB

6) 构建 Tomcat 基础镜像

在 jenkins 和 docker 主机上安装 JDK:

```
[root@docker ~]# tar xf jdk-8u191-linux-x64.tar.gz
[root@docker ~]# mv jdk1.8.0_191/ /usr/local/java
```

```
[root@docker ~]# ls -l apache-tomcat-8.5.16.tar.gz
-rw-r--r-- 1 root root 9417469 7 月 27 2017 apache-tomcat-8.5.16.tar.gz

[root@docker ~]# cat Dockerfile
FROM centos:7
MAINTAINER crushlinux

ENV VERSION=8.5.16
ENV JAVA_HOME /usr/local/java

ADD ./apache-tomcat-${VERSION}.tar.gz /tmp
RUN cd /tmp && \
    mv apache-tomcat-${VERSION} /usr/local/tomcat && \
    rm -rf apache-tomcat-${VERSION}.tar.gz /usr/local/tomcat/webapps/* && \
    mkdir /usr/local/tomcat/webapps/ROOT

EXPOSE 8080
CMD ["/usr/local/tomcat/bin/catalina.sh", "run"]
```

```
[root@docker ~]# docker build -t 192.168.200.111:5000/tomcat-85 .
[root@docker ~]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
192.168.200.111:5000/tomcat-85	latest	8fdce2f6f1eb	25 seconds ago	224MB
192.168.200.111:5000/centos	7	5f618d4dd78d	10 minutes ago	589MB
centos	7	9f38484d220f	3 months ago	202MB

7) 构建镜像并上传到 registry:

```
[root@docker docker-tomcat]# docker push 192.168.200.111:5000/tomcat-85
The push refers to repository [192.168.200.111:5000/tomcat-85]
ac6dcc6c07d0: Pushed
dad381e36213: Pushed
c664eb1dcbf0: Pushed
latest: digest:
sha256:761e8fa35fadedf1a1717bcd1628cb45b48f58cd24561cab9c66b03096c5dc92c size: 952
```

5、配置 Jenkins 环境

Jenkins 是一个开源软件项目，是基于 Java 开发的一种持续集成工具，用于代码编译、部署、测试等工作。Jenkins 也是一个跨平台的集成工具，大多数主流的平台都支持，而且安装很简单，这里将以部署 war 包方式安装。

官网下载地址：<https://jenkins.io/download/>。

1) 修改 jenkins 运行用户

```
[root@jenkins ~]# vim /etc/sysconfig/jenkins
JENKINS_USER="root"
[root@jenkins ~]# /etc/init.d/jenkins restart
Restarting jenkins (via systemctl): [ 确定 ]
```

2) Jenkins 配置全局工具配置

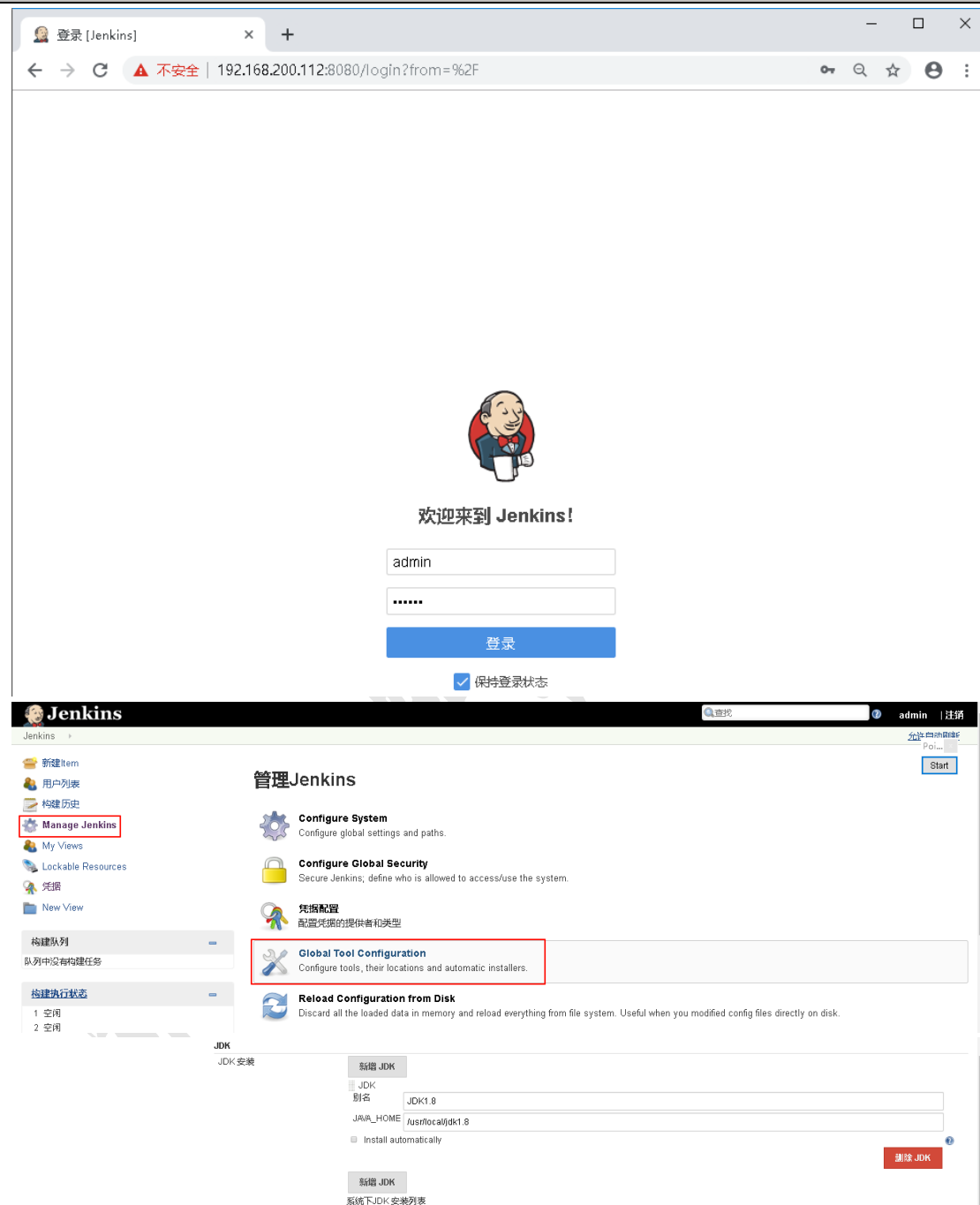
在 Jenkins 首页中点击“Manage Jenkins”->“Global Tool Configuration”->“JDK”->新增“JDK”，设置 JDK 别名为“JDK1.8”。去掉“Install automatically”选项，设置“JAVA_HOME”为本案例中 JDK 实际安装路径。

```
[root@jenkins ~]# tar xf jdk-8u191-linux-x64.tar.gz
[root@jenkins ~]# mv jdk1.8.0_191/ /usr/local/java
[root@jenkins ~]# vim /etc/profile
export JAVA_HOME=/usr/local/java
export CLASSPATH=$JAVA_HOME/lib/tools.jar:$JAVA_HOME/lib/dt.jar
export PATH=$JAVA_HOME/bin:$PATH
[root@jenkins ~]# rm -rf /usr/bin/java
[root@jenkins ~]# source /etc/profile
[root@jenkins ~]# java -version
java version "1.8.0_191"
Java(TM) SE Runtime Environment (build 1.8.0_191-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.191-b12, mixed mode)

[root@jenkins ~]# cd /var/lib/jenkins/updates
[root@jenkins updates]# sed -i
's/http://updates.jenkins-ci.org/download/https://mirrors.tuna.tsinghua.edu.cn/jenkins/g' default.json
```

```
[root@jenkins updates]# sed -i 's/http:\\\\www.google.com/https:\\\\www.baidu.com/g'
default.json
[root@jenkins jenkins]# /etc/init.d/jenkins restart
Restarting jenkins (via systemctl):
```

[确定]



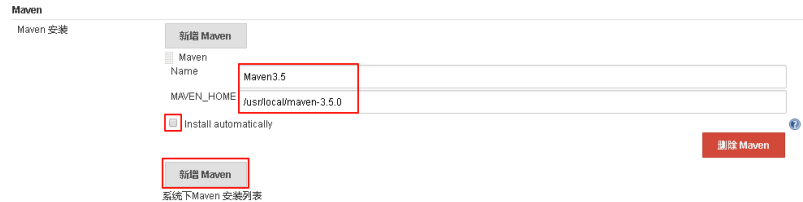
在“Global Tool Configuration”配置界面中找到 Maven 配置选项，然后点击“新增 Maven”并设置别名为“Maven3.5”。

```
[root@jenkins ~]# tar xf apache-maven-3.5.0-bin.tar.gz
[root@jenkins ~]# mv apache-maven-3.5.0 /usr/local/maven-3.5.0
```

为 maven 更换阿里云镜像站

```
[root@jenkins ~]# vim /usr/local/maven-3.5.0/conf/settings.xml
<mirror>
```

```
<id>nexus-aliyun</id>
<mirrorOf>central</mirrorOf>
<name>Nexus aliyun</name>
<url>http://maven.aliyun.com/nexus/content/groups/public</url>
</mirror>
```



Git 配置

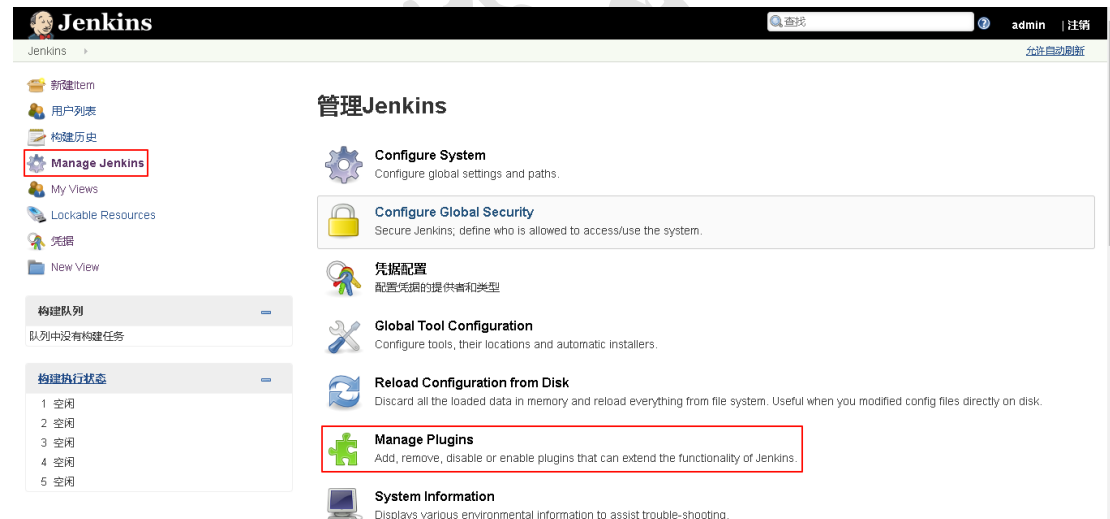
```
[root@jenkins ~]# which git
/usr/bin/git
```



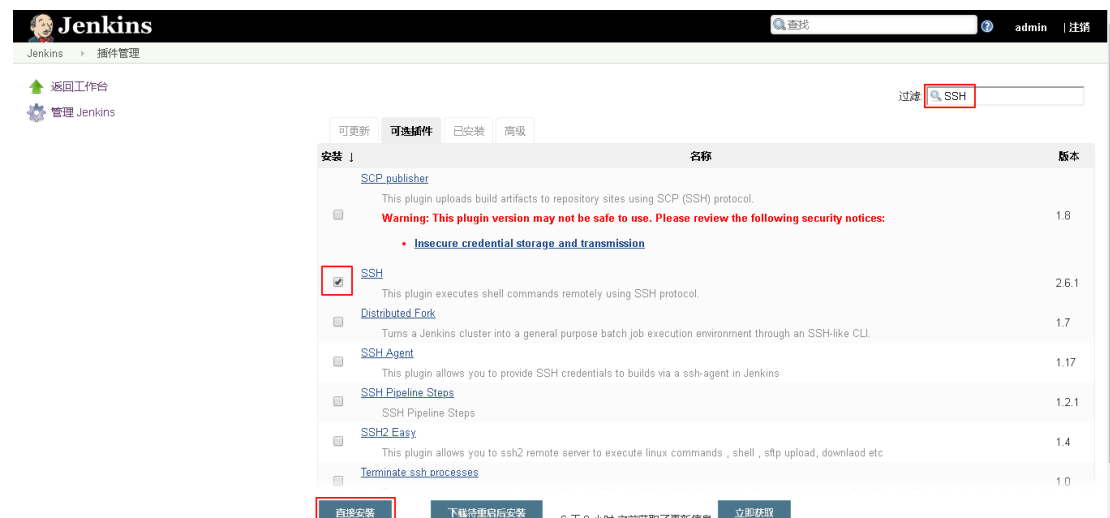
以上全局基本配置完毕后，点击保存即可完成。

3) Jenkins 安装必要插件

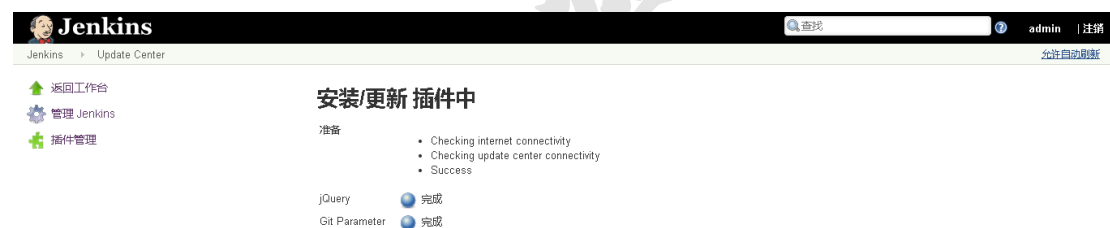
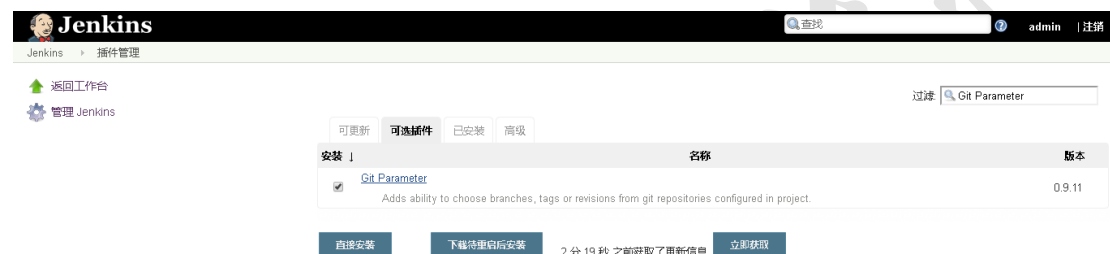
Manage Jenkins-> Manage Plugins



安装 SSH 插件



安装 Git Parameter 插件



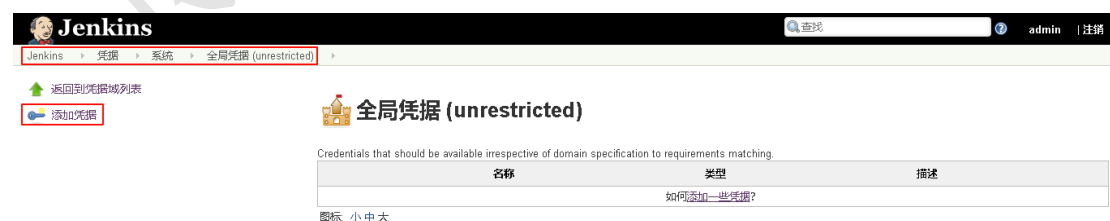
插件说明：

SSH：用于 SSH 远程 Docker 主机执行 Shell 命令。

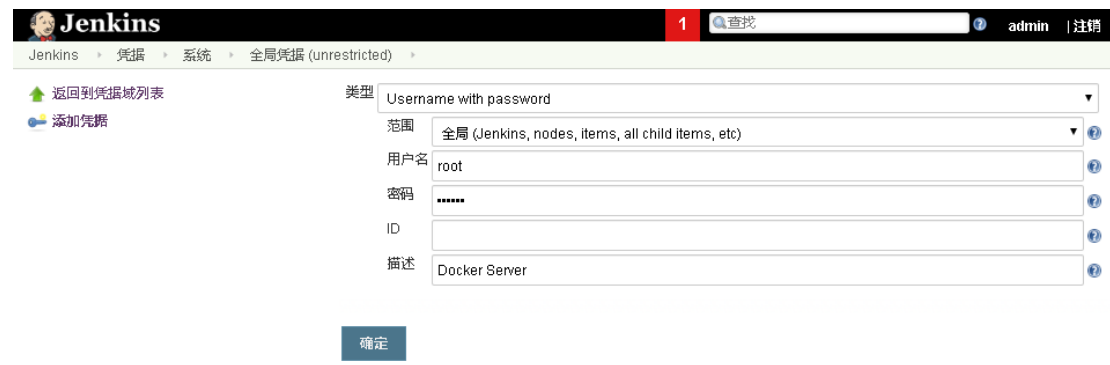
Git Parameter：动态获取 Git 仓库 Branch、Tag。

配置 SSH 插件

创建一个用于连接 Docker 主机的凭据。主页面-> 凭据-> 系统-> 全局凭据-> 添加凭据。

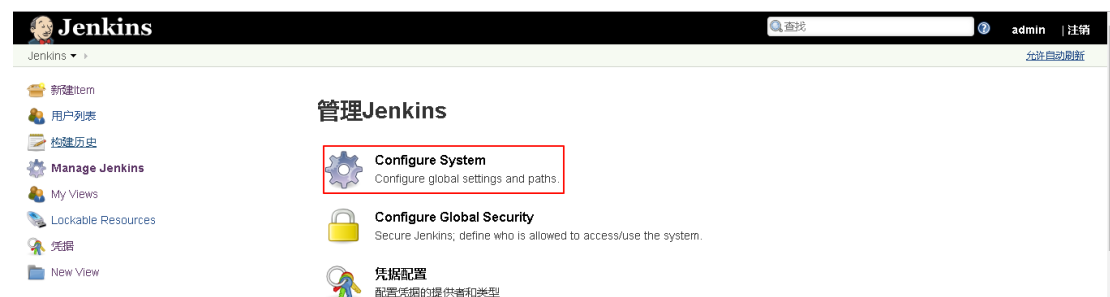


输入连接 Docker 主机的用户名和密码




The screenshot shows the Jenkins 'Add Credentials' page. The breadcrumb trail is 'Jenkins > 凭据 > 系统 > 全局凭据 (unrestricted)'. On the left, there are links for '返回到凭据域列表' and '添加凭据'. The main form is titled 'Username with password'. It includes a '范围' (Scope) dropdown set to '全局 (Jenkins, nodes, items, all child items, etc)', a '用户名' (Username) field with 'root', a '密码' (Password) field with masked characters, an empty 'ID' field, and a '描述' (Description) field with 'Docker Server'. A '确定' (OK) button is at the bottom.

第二步：添加 SSH 远程主机。Manage Jenkins-> Configure System。



配置 SSH remote hosts



The screenshot shows the 'SSH remote hosts' configuration page. The breadcrumb trail is 'SSH sites'. The form has fields for 'Hostname' (192.168.200.113), 'Port' (22), 'Credentials' (root (Docker Server) with a '+ 添加' button), 'Pty' (checkbox), 'serverAliveInterval', and 'timeout'. A 'Check connection' button is at the bottom right. A 'Successful connection' message is displayed above the button.

6、Jenkins 创建项目并发布测试

1) 主页面-> 新建 Item-> 输入任务名称，构建一个 Maven 项目



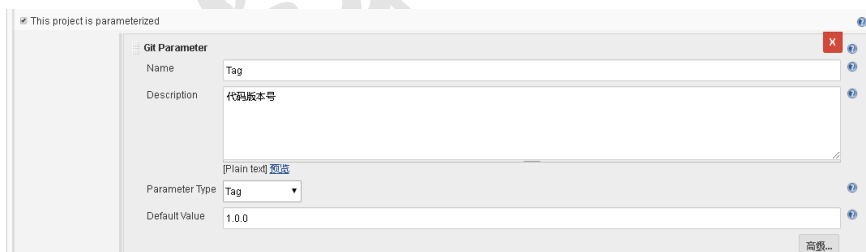


注意：如果没有显示“构建一个Maven 项目”选项，需要在管理插件里安装“Maven Integration”插件。

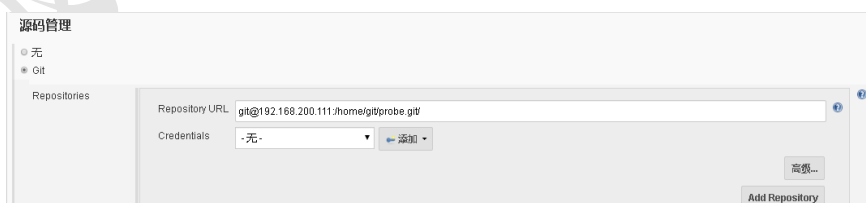
2) 配置 Git 参数化构建



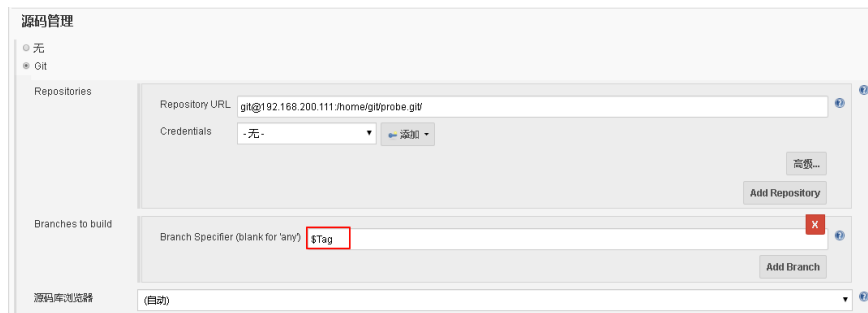
3) 动态获取 Git 仓库 tag，与用户交互选择 Tag 发布



4) 指定项目 Git 仓库地址



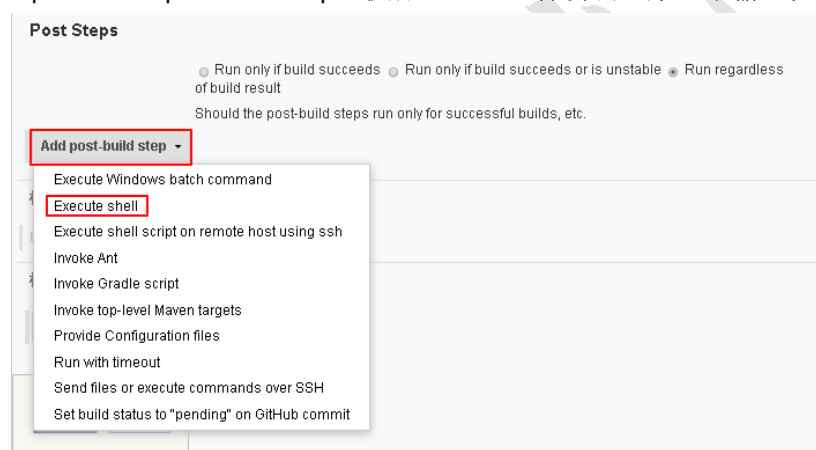
5) 修改*/master 为\$Tag, Tag 是上面动态获取的变量名，表示根据用户选择打代码版本



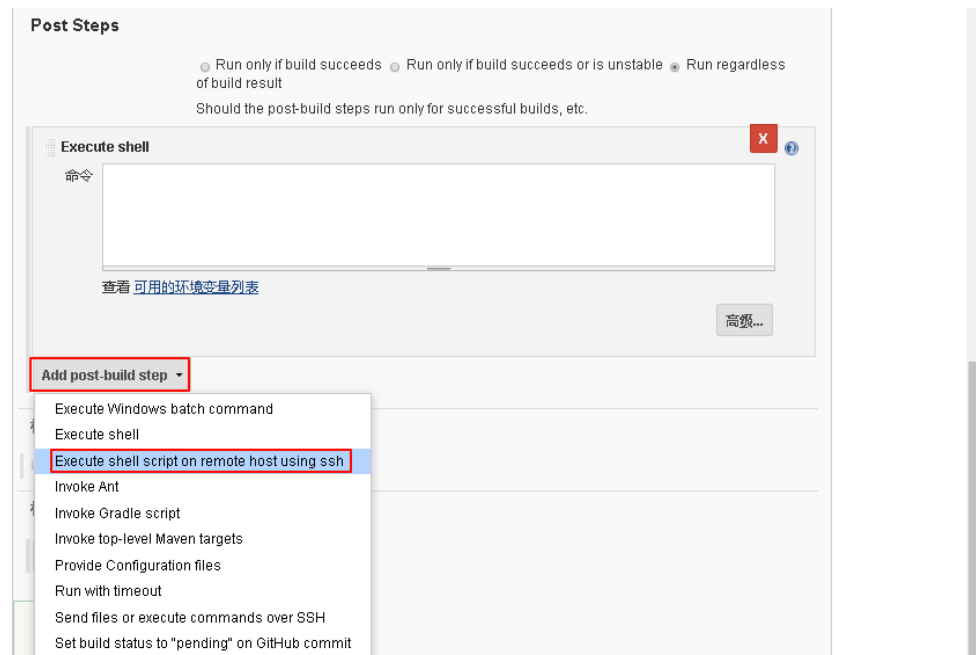
6) 设置 maven 构建命令选项 “clean package -Dmaven.test.skip=true”



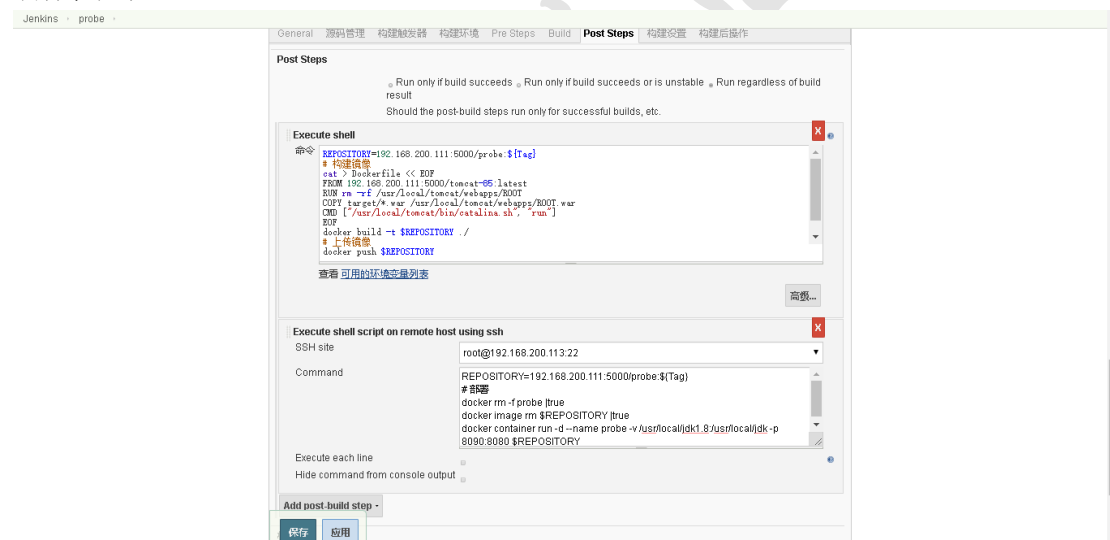
7) 利用 pom.xml 文件构建项目。在 Jenkins 本机镜像构建与推送到镜像仓库，并 SSH 远程连接到 Docker 主机使用推送的镜像创建容器
选择 Post Steps ->Add post-build step ->执行 shell，这样会调出第一个输入框



选择 Post Steps ->Execute shell script on remote host using ssh，这样会调出第二个输入框。



pom.xml: 它是声明项目描述符遵循 **POM** 模型版本。模型本身的版本很少改变，虽然如此，但它仍然是必不可少的，这是为了当 **Maven** 引入新的特性或者其他模型变更的时候，确保稳定性。



第一个命令框内容

```

REPOSITORY=192.168.200.111:5000/probe:${Tag}
# 构建镜像
cat > Dockerfile << EOF
FROM 192.168.200.111:5000/tomcat-85:latest
RUN rm -rf /usr/local/tomcat/webapps/ROOT
COPY psi-probe-web/target/*.war /usr/local/tomcat/webapps/ROOT.war
CMD ["/usr/local/tomcat/bin/catalina.sh", "run"]
EOF
docker build -t $REPOSITORY ./
# 上传镜像
docker push $REPOSITORY
    
```

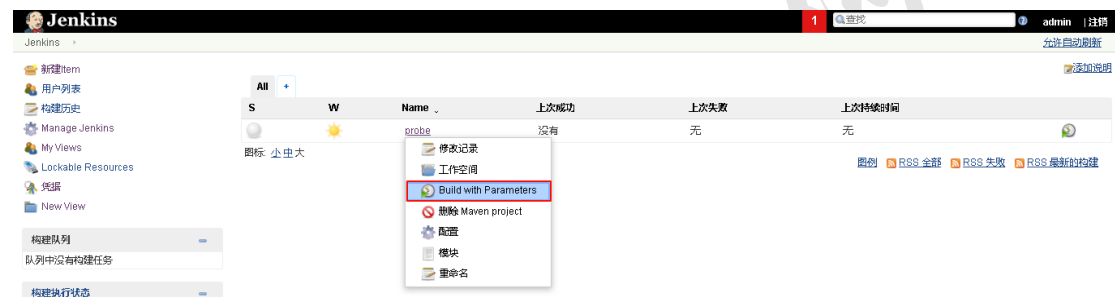
第二个命令框内容

```
REPOSITORY=192.168.200.111:5000/probe:${Tag}
# 部署
docker rm -f probe | true
docker container run -d --name probe -v /usr/local/java:/usr/local/java -p
8090:8080 $REPOSITORY
```

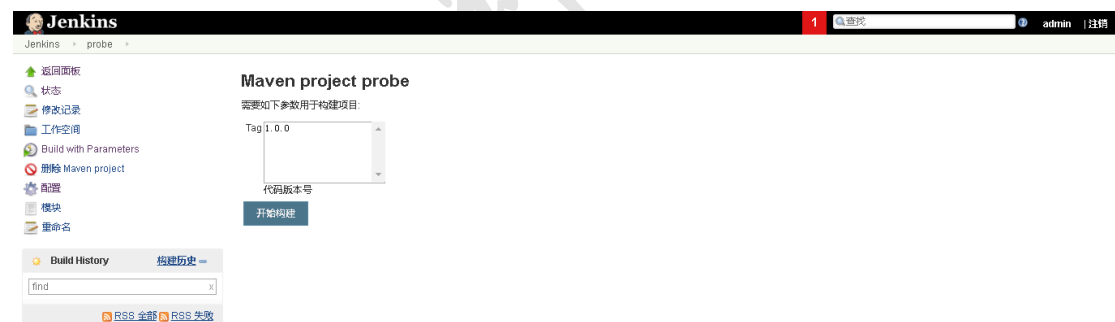
```
[root@docker ~]# vim /etc/docker/daemon.json
{"insecure-registries":["192.168.200.111:5000"]}
[root@docker ~]# systemctl daemon-reload
[root@docker ~]# systemctl restart docker
```

注意：从部署命令当中可以看到最后一行容器名称是 probe，暴露宿主端口 8090，即使用宿主机 IP:8090 就可以访问 probe 项目。

8) probe 项目已配置完成，开始构建



9) 选择 tag，开始构建。



10) 在任务控制台输出构建日志的开始信息

定上一个正常版本镜像，然后重新执行创建容器命令即可回到之前正常的版本。

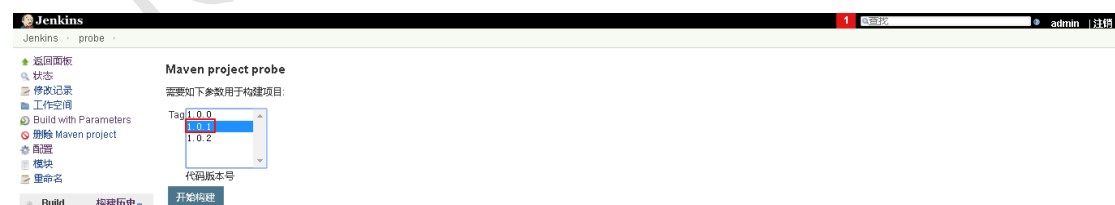
提交 1.0.1 版本代码

```
[root@jenkins ~]# cd psi-probe/
[root@jenkins psi-probe]# echo "test1" > psi-probe-web/src/main/webapp/test1.html
[root@jenkins psi-probe]# git add .
[root@jenkins psi-probe]# git commit -m "test1"
[master 7b36b14] test1
1 file changed, 1 insertion(+)
create mode 100644 web/src/main/webapp/test1.html
[root@jenkins psi-probe]# git tag 1.0.1
[root@jenkins psi-probe]# git push origin 1.0.1
Counting objects: 12, done.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 574 bytes | 0 bytes/s, done.
Total 7 (delta 2), reused 0 (delta 0)
To git@192.168.200.111:/home/git/probe.git
* [new tag]          1.0.1 -> 1.0.1
```

提交 1.0.2 版本代码

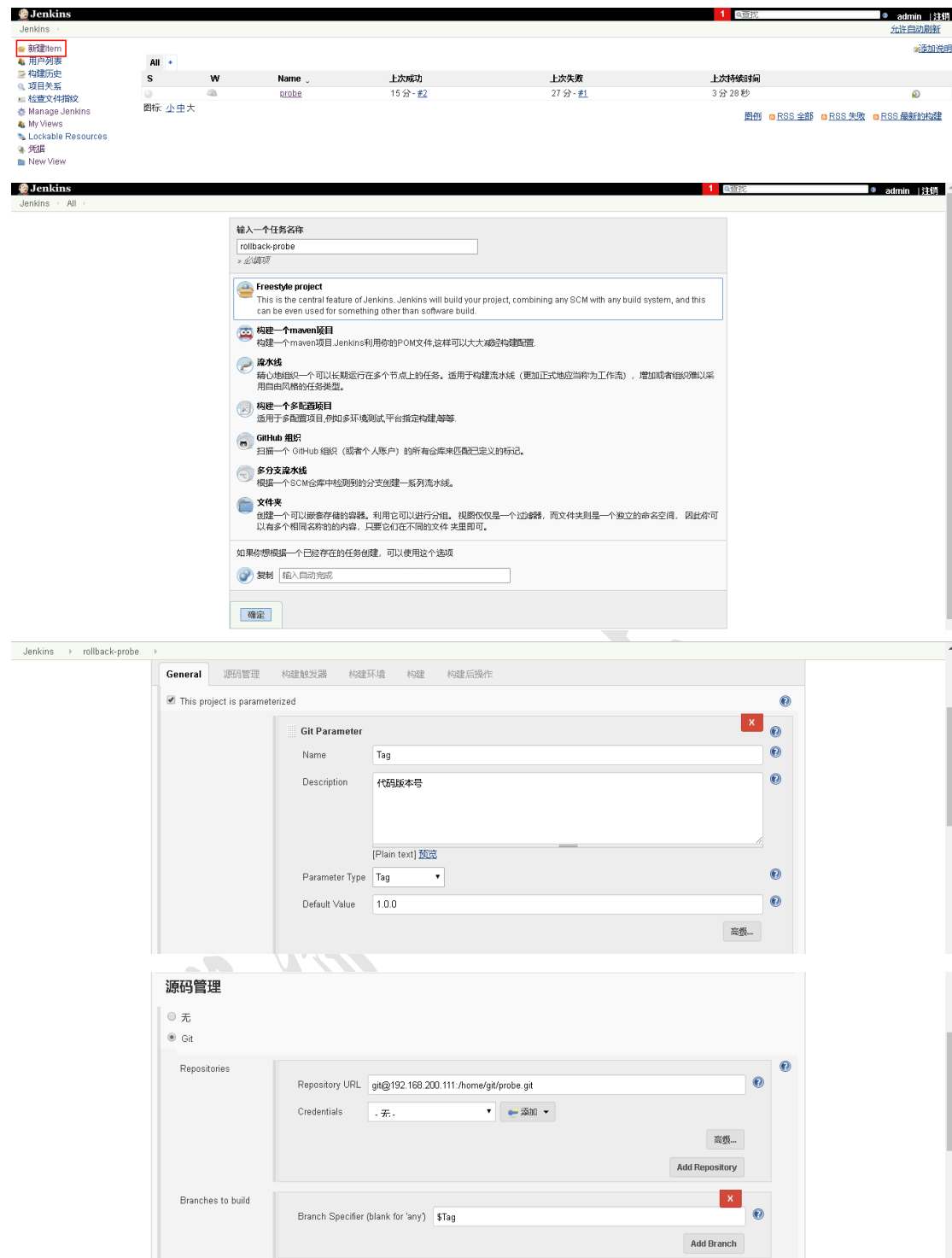
```
[root@jenkins psi-probe]# echo "test2" > psi-probe-web/src/main/webapp/test2.html
[root@jenkins psi-probe]# git add .
[root@jenkins psi-probe]# git commit -m "test2"
[master eb617d6] test2
1 file changed, 1 insertion(+)
create mode 100644 web/src/main/webapp/test2.html
[root@jenkins psi-probe]# git tag 1.0.2
[root@jenkins psi-probe]# git push origin 1.0.2
Counting objects: 12, done.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 577 bytes | 0 bytes/s, done.
Total 7 (delta 2), reused 0 (delta 0)
To git@192.168.200.111:/home/git/probe.git
* [new tag]          1.0.2 -> 1.0.2
```

发布 1.0.1 版本



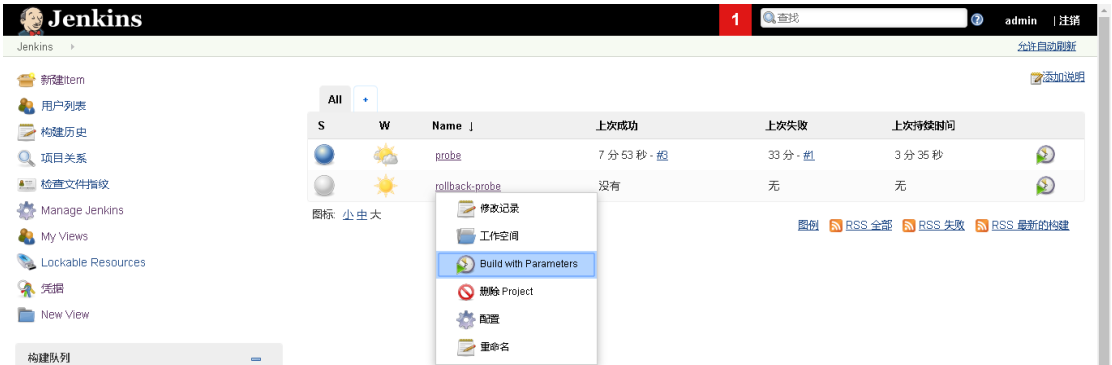
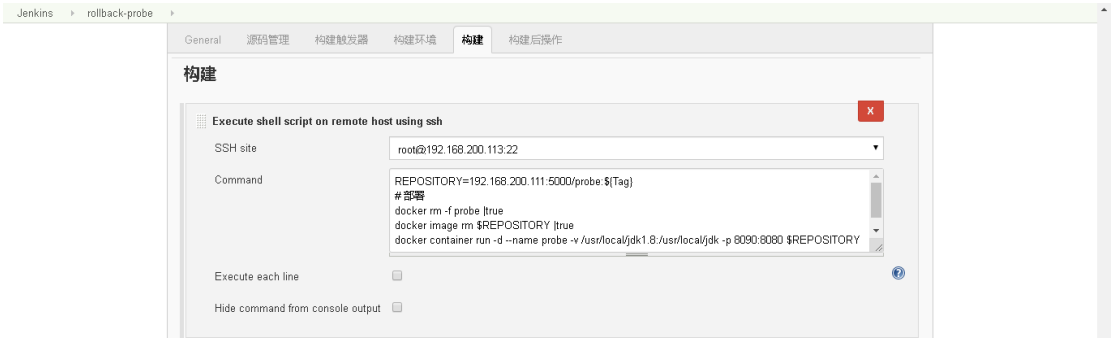
```
[root@docker ~]# docker exec -it probe /bin/bash
[root@1b57d0ebb506 /]# ls /usr/local/tomcat/webapps/ROOT
META-INF WEB-INF a.html css flags index.jsp js test1.html
```

1) 创建一个自由软件项目风格任务

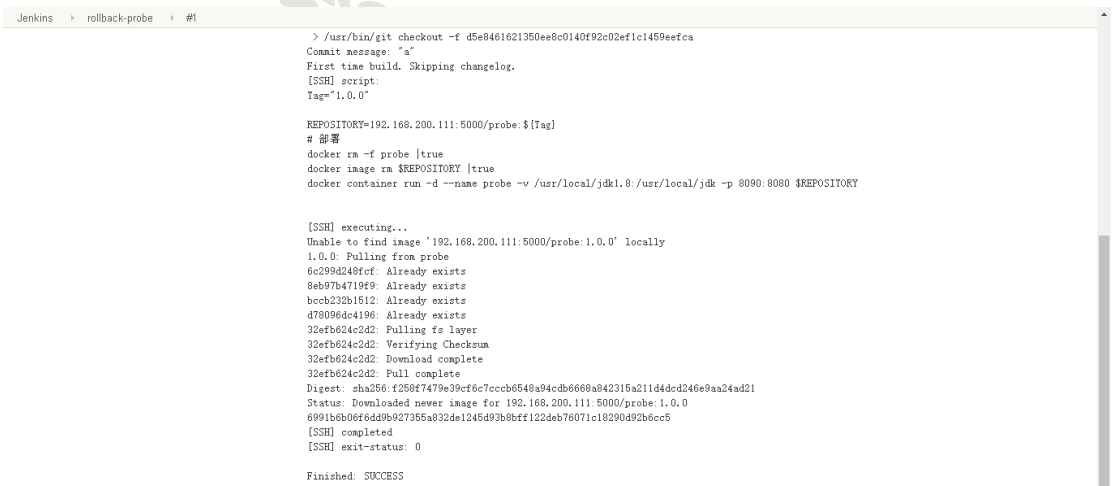
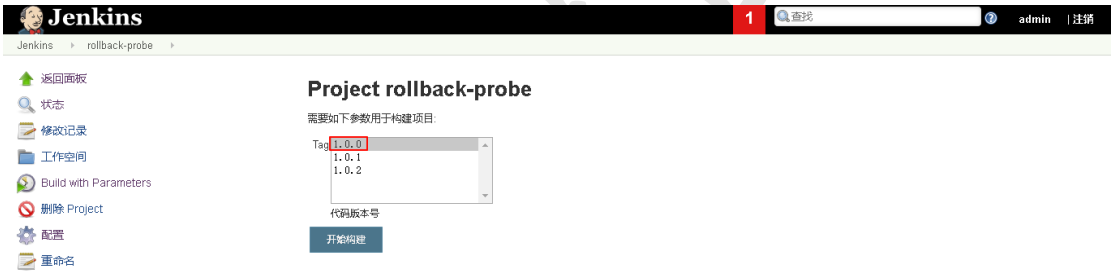


```

REPOSITORY=192.168.200.111:5000/probe:${Tag}
# 部署
docker rm -f probe | true
docker container run -d --name probe -v /usr/local/java:/usr/local/java -p
8090:8080 $REPOSITORY
    
```



回滚到 1.0.0 版本



[root@docker ~]# docker ps -l			
CONTAINER ID	IMAGE	COMMAND	
CREATED	STATUS	POR	
TS	NAMES	6991b6b06f6d	192.168.200.111:5000/probe:1.0.0
"/usr/local/tomcat/b..."	20 seconds ago	Up 18 seconds	0.
0.0.0:8090->8080/tcp	probe		

```
[root@docker ~]# docker exec -it probe /bin/bash
[root@6991b6b06f6d /]# ls /usr/local/tomcat/webapps/ROOT
META-INF WEB-INF a.html css flags index.jsp js
```

基于 Jenkins+Git+Ansible 发布 PHP 项目

1、部署 PHP 运行环境

PHP 是一个动态程序，负责解析 PHP-FPM 服务，而这个服务不支持静态页面处理，一般结合 Nginx 解决这个问题。Nginx 本身是一个静态 Web 服务器，并不支持解析 PHP 程序，但它支持了 FastCGI 接口来调用动态服务来解析 PHP 程序。

当客户端请求 PHP 页面时，Nginx 通过 fastcgi 接口转发给本地 9000 端口的 PHP-FPM 子进程处理，处理完成后返回 Nginx。

1) 安装 Nginx

配置 Nginx 网络源

```
[root@docker ~]# vim /etc/yum.repos.d/nginx.repo
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/7/$basearch/
gpgcheck=0
enabled=1
```

安装并启动

```
[root@docker ~]# yum -y install nginx
[root@docker ~]# systemctl start nginx
```

2) 安装 PHP

安装 php 依赖的第三方库，命令如下：

```
[root@docker ~]# yum -y install gd-devel libxml2-devel libcurl-devel libjpeg-devel libpng-devel
gcc openssl-*
```

编译安装 php

```
[root@docker ~]# tar xf php-5.6.39.tar.gz -C /usr/src/
[root@docker ~]# cd /usr/src/php-5.6.39/
[root@docker php-5.6.39]# ./configure --prefix=/usr/local/php \
--with-config-file-path=/usr/local/php/etc \
--with-mysql --with-mysqli --with-openssl --with-zlib \
--with-curl --with-gd --with-jpeg-dir --with-png-dir \
--with-iconv --enable-fpm --enable-zip --enable-mbstring && make -j 2 && make install
```

配置 php-fpm,命令如下：

```
[root@docker php-5.6.39]# cp php.ini-production /usr/local/php/etc/php.ini
```



```
[root@docker php-5.6.39]# cp /usr/local/php/etc/php-fpm.conf.default
/usr/local/php/etc/php-fpm.conf
[root@docker php-5.6.39]# vim /usr/local/php/etc/php-fpm.conf
149 user = nginx
150 group = nginx
[root@docker php-5.6.39]# cp sapi/fpm/init.d.php-fpm /etc/init.d/php-fpm
[root@docker php-5.6.39]# chmod +x /etc/rc.d/init.d/php-fpm
[root@docker php-5.6.39]# service php-fpm start
Starting php-fpm  done
```

3) Nginx 代理 PHP

添加虚拟主机配置如下:

```
[root@docker ~]# vim /etc/nginx/conf.d/default.conf
location / {
    root    /usr/share/nginx/html;
    index  index.html index.htm index.php;
}

location ~ \.php$ {
    root /usr/share/nginx/html;
    fastcgi_pass 127.0.0.1:9000;
    fastcgi_index index.php;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    include fastcgi_params;
}

[root@docker ~]# systemctl restart nginx
```

2、安装 Ansible 插件

1) 主页面-> 系统管理->管理插件



The screenshot shows the Jenkins web interface. In the left sidebar, 'Manage Jenkins' is highlighted with a red box. The main content area, titled '管理Jenkins', lists several configuration options. 'Manage Plugins' is also highlighted with a red box and includes a red triangle icon with the text '可用更新' (Available Update).

2) 安装 Ansible 插件



3) 点击”直接安装”按钮。只安装 Ansible 插件是不行的,还要在 Jenkins 主机上安装 ansible:

```
[root@jenkins ~]# rpm -i epel-release-latest-7.noarch.rpm
[root@jenkins ~]# yum -y install ansible
[root@jenkins ~]# vim /etc/ansible/hosts
[webserver]
192.168.200.113
```

其中: [webserver]表示 Web 主机分组名称,该分组下可以写多个主机,也可以写多个分组区分不同角色服务器。

3、上传 PHP 项目代码到 Git 仓库

1) 在 Git 服务器创建 wordpress 版本仓库

```
[root@git ~]# su - git
[git@git ~]$ mkdir wordpress.git
[git@git ~]$ cd wordpress.git
[git@git wordpress.git]$ git --bare init
初始化空的 Git 版本库于 /home/git/wordpress.git/
```

2) 下载开源 PHP 博客系统 wordpress

```
[root@jenkins ~]# wget https://wordpress.org/latest.tar.gz
[root@jenkins ~]# tar xf latest.tar.gz
[root@jenkins ~]# cd wordpress/
```

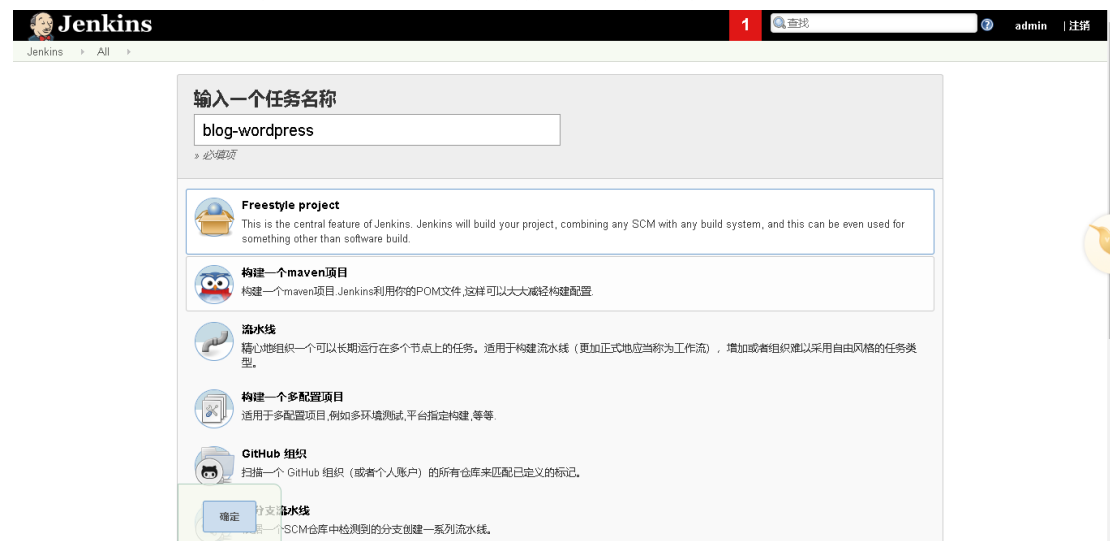
3) 提交到 Git 仓库

```
[root@jenkins ~]# git init
[root@jenkins ~]# git remote add origin git@192.168.200.111:/home/git/wordpress.git
[root@jenkins ~]# git add .
[root@jenkins ~]# git commit -m "wp"

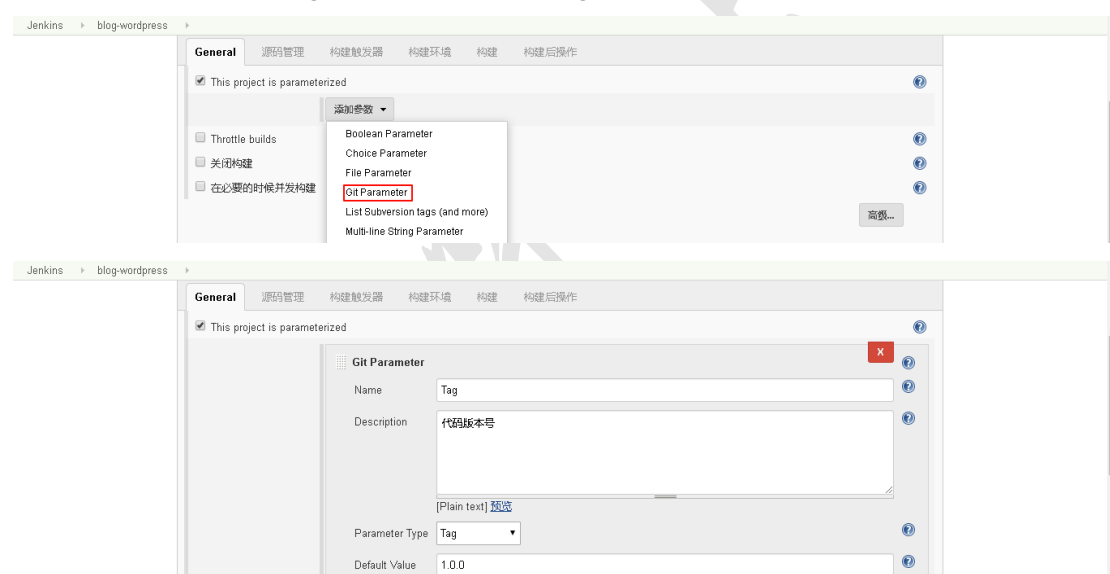
[root@jenkins wordpress]# git tag 1.0.0
[root@jenkins wordpress]# git push origin 1.0.0
```

4、Jenkins 创建项目并发布测试

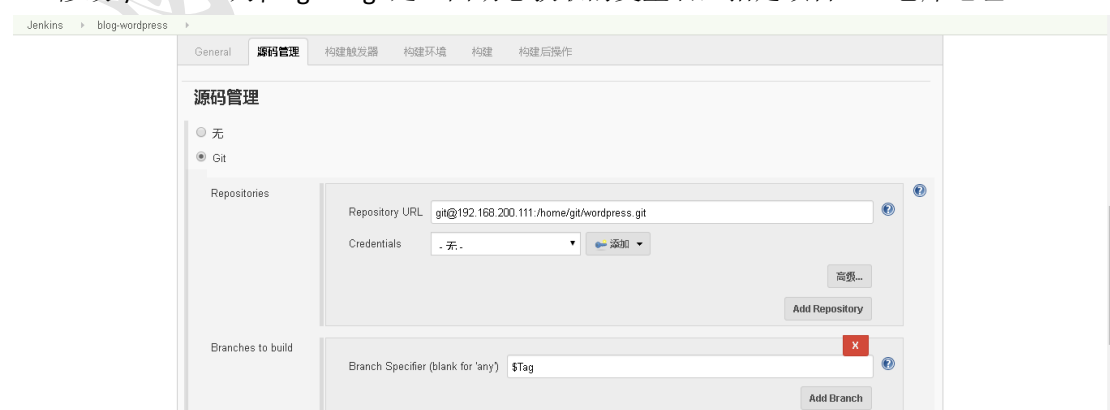
1) 主页面-> 新建任务-> 创建一个自由软件项目风格任务



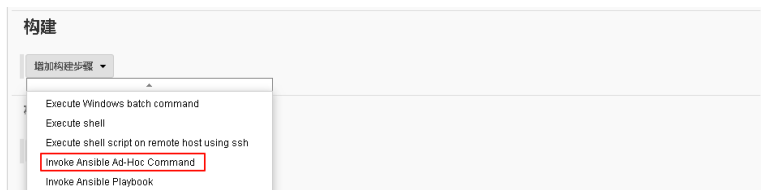
2) 动态获取 Git 仓库 tag，与用户交互选择 Tag 发布，



3) 修改*/master 为\$Tag。Tag 是上面动态获取的变量名，指定项目 Git 仓库地址。

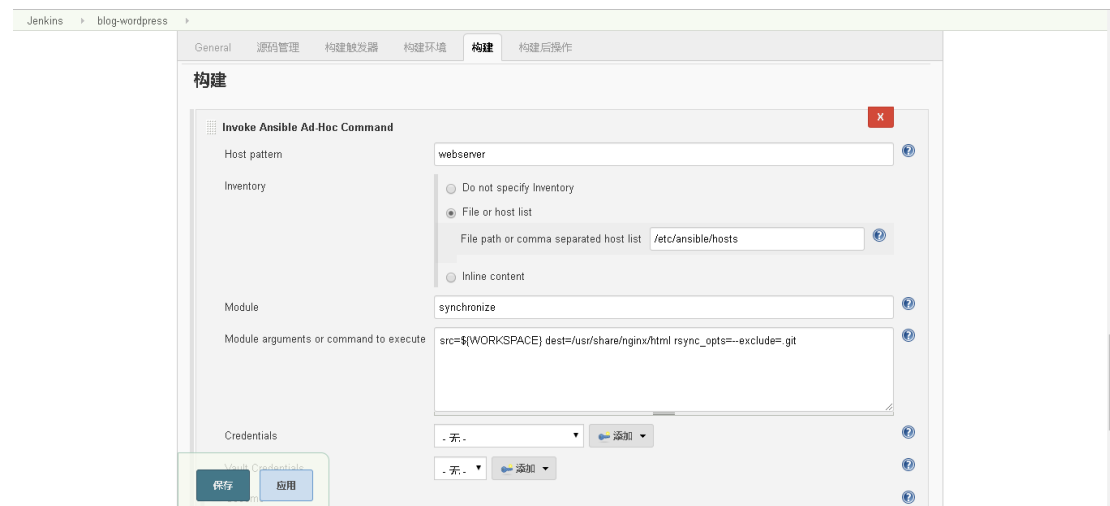


4) 使用 ansible 插件，选择 Invoke Ansible Ad-Hoc Command

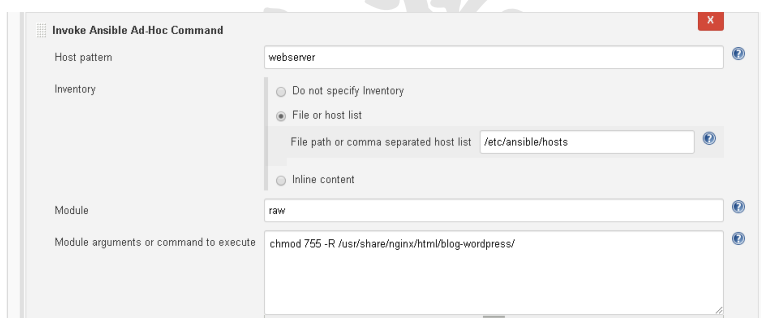


5) 使用 synchronize 模块同步数据

```
src=${WORKSPACE} dest=/usr/share/nginx/html rsync_opts=--exclude=.git
```



6) 使用 raw 模块校正项目权限



参数说明：

- Host pattern: 指定刚在/etc/ansible/hosts 中定义的主机分组名称
- Inventory: 主机清单
- Module: 模块名
- Module arguments or command to execute: 模块参数或执行命令

```
[root@jenkins ~]# ssh-keygen
[root@jenkins ~]# ssh-copy-id root@192.168.200.113
```

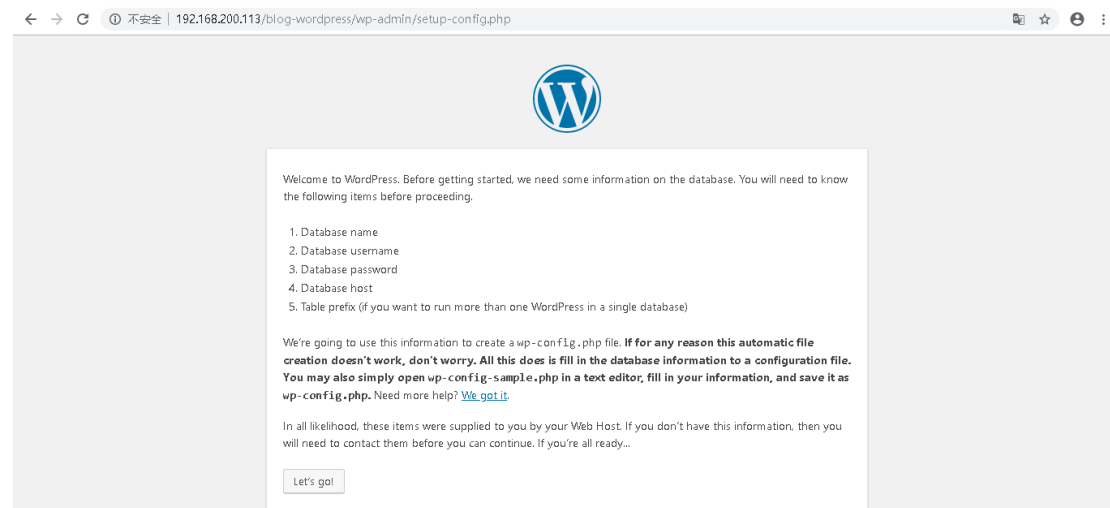
7) 主页面-> 右击 blog-wordpress -> Build with Parameters



8) 构建日志信息会控制台输出



9) 浏览器访问测试 <http://192.168.200.113/blog-wordpress/wp-admin/>



10) 模拟实际生产环境提交代码，作用是可以清楚看到两次发版代码的不同。

```
[root@jenkins ~]# cd wordpress/  
[root@jenkins wordpress]# echo "Hello Wordpress" > test.html
```

11) 将修改的 test.html 提交到 Git 仓库

```
[root@jenkins wordpress]# git add .  
[root@jenkins wordpress]# git commit -m "hw"  
[master 79fb4a8] hw  
1 file changed, 1 insertion(+)  
create mode 100644 test.html  
[root@jenkins wordpress]# git tag 1.0.1  
[root@jenkins wordpress]# git push origin 1.0.1  
Counting objects: 4, done.  
Compressing objects: 100% (2/2), done.  
Writing objects: 100% (3/3), 273 bytes | 0 bytes/s, done.  
Total 3 (delta 1), reused 0 (delta 0)  
To git@192.168.200.111:/home/git/wordpress.git  
* [new tag] 1.0.1 -> 1.0.1
```

12) 在 Jenkins 执行构建任务



发布成功后，访问：<http://192.168.200.113/blog-wordpress/test.html>，页面显示“Hello Wordpress!”。说明刚修改的代码已发布成功！



ⓘ 不安全 | 192.168.200.113/blog-wordpress/test.html

Hello Wordpress

内部资料 禁止传播