

Jenkins+Gitlab+apache

作者：Liu hai

环境准备：

jenkins CentOS 7 4G内存2CPU IP: 10.10.10.11

Gitlab CentOS 7 4G内存2CPU IP: 10.10.10.12

webServer CentOS 7 2G内存2CPU IP: 10.10.10.13

一、Gitlab 安装部署

安装版本：社区版

官网：<https://about.gitlab.com/installation>

国内镜像源：<https://mirrors.tuna.tsinghua.edu.cn>

安装

下载GitLab安装包，如果没有网络需要提前准备好rpm包

```
wget https://omnibus.gitlab.cn/el/7/gitlab-jh-14.6.0-jh.0.el7.x86_64.rpm
```

安装下载的rpm包

```
rpm -Uvh gitlab-jh-14.6.0-jh.0.el7.x86_64.rpm  
# 或  
# 推荐会自动解决依赖问题  
yum -y localinstall gitlab-jh-14.6.0-jh.0.el7.x86_64.rpm
```

配置Gitlab

```
vim /etc/gitlab/gitlab.rb  
  
external_url 'http://10.0.0.12' #修改此行为自己的IP  
#增加下面行，可选邮件通知设置  
gitlab_rails['smtp_enable'] = true # 开启邮箱  
gitlab_rails['smtp_address'] = "liuhai.qq.com" # SMTP邮件服务器  
gitlab_rails['smtp_port'] = 25 # SMTP邮件服务器端口  
gitlab_rails['smtp_user_name'] = "liuhai@qq.com" # 自己的邮箱  
gitlab_rails['smtp_password'] = "授权码" # 自己邮箱的授权码  
gitlab_rails['smtp_domain'] = "qq.com" # SMTP服务器的域  
gitlab_rails['smtp_authentication'] = "login"  
gitlab_rails['smtp_enable_starttls_auto'] = true
```

执行初始化并启动服务

```
gitlab-ctl reconfigure
```

常用命令

```
gitlab-rails #用于启动控制台进行特殊操作，如修改管理员密码、打开数据库控制台( gitlab-rails dbconsole)
等
gitlab-psql #数据库命令行
gitlab-rake #数据备份恢复等数据操作

gitlab-ctl #客户端命令行操作行
gitlab-ctl stop #停止gitlab
gitlab-ctl start #启动gitlab
gitlab-ctl restart #重启gitlab
gitlab-ctl status #查看组件运行状态
gitlab-ctl tail nginx #查看某个组件的日志
```

访问Gitlab

<http://10.10.10.12>

第一次登录使用root用户，密码在/etc/gitlab/initial_root_password中，完成登录后及时修改密码！！

二、jenkins 安装部署

安装jdk环境

jdk版本可自由选择，官方推荐的是jdk11

```
# 解压
tar -xf jdk-8u181-linux-x64.tar.gz -C /usr/local/
# 创建一个软链接 方便版本的切换
ln -s /usr/local/jdk1.8.0_181 /usr/local/java
# 编辑环境变量
vim /etc/profile.d/java.sh
export JAVA_HOME=/usr/local/java
export PATH=$JAVA_HOME/bin:$PATH
# 使环境生效
. /etc/profile.d/java.sh
# 验证
java --version
```

安装

方案一：

官网的方案，会因为网络问题而失败

```
# 建立源
wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat/jenkins.repo
# 下载密钥
rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
# 直接安装
yum -y install jenkins
```

方案二：

先下载好rpm包，更推荐这种方式

```
# 安装epel源
yum -y install epel-release
# 本地安装
yum -y localinstall jenkins-2.319.1-1.1.noarch.rpm
```

配置

指定java路径

```
vim /etc/sysconfig/jenkins
JENKINS_JAVA_CMD="/usr/local/java/bin/java"
```

启动服务

```
systemctl restart jenkins
```

访问

<http://10.10.10.11:8080>

将插件一遍一遍的重复重试完成插件的安装，配置管理员密码重启就可以了!!

三、web服务器

简单的apach就行了,安装启动

```
yum -y install httpd
systemctl start httpd
```

以上都是环境准备

四、配置jenkins拉取Gitlab中的项目并推送到web服务器

Gitlab中的配置

首先新建一个Jenkins（可自定义）用户，为管理员身份

然后以刚才建立的用户的身份登录Gitlab，创建一个项目

最后创建一个测试页面index.html用来做测试，让Jenkins拉取，并推送到web服务器中

jenkins中配置拉取Gitlab中的项目

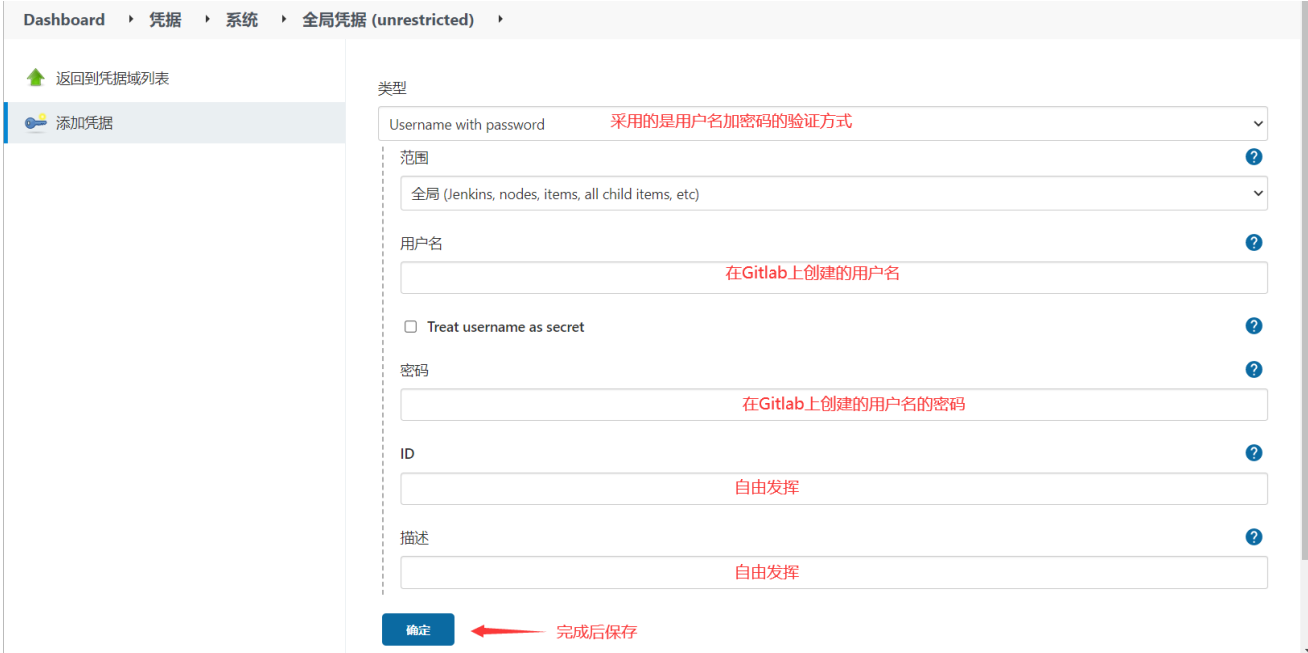
1、添加凭据，也就是说Jenkins使用什么身份拉取Gitlab中项目



2、添加全局凭据



3、添加用户，采用的方式可以是多种这里采用用户名和密码的方式




4、新建任务用于构建 --- 到这一步还只能是拉取下来，拉取下来的项目文件存放在/var/lib/jenkins/workspace/下

输入一个任务名称

web 自定义一个名字


* 任务名 'web' 已存在

 **构建一个自由风格的软件项目**
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. 甚至可以构建软件以外的系统.

 **构建一个maven项目**
构建一个maven项目.Jenkins利用你的POM文件.这样可以大大减轻构建配置.

 **流水线**
精心地组织一个可以长期运行在多个节点上的任务.适用于构建流水线 (更加正式地应称为工作流), 增加或者组织难以采用自由风格的任务类型.

 **构建一个多配置项目**
适用于多配置项目.例如多环境测试.平台指定构建.等等.

 **(0) 文件夹**
Creates a set of multibranch project subfolders by scanning for repositories.

 **多分支流水线**
根据一个SCM仓库中检测到的分支创建一系列流水线.

 **文件夹**
创建一个可以嵌套存储的容器.利用它可以进行分组.视图仅仅是一个过滤器.而文件夹则是一个独立的命名空间. 因此你可以有多个相同名称的内容.只要它们在不同的文件 夹里即可.

如果你想根据一个已经存在的任务创建, 可以使用这个选项

确定 完成

源码管理

☐ 无

☒ Git

Repositories

Repository URL

http://10.10.10.12/tom/web.git

克隆地址 密码验证就使用http的 公私钥就用git

Credentials

Jenkins/***** (123)

添加

选择拉取的用户

高级...

Add Repository

Branches to build

指定分支 (为空时代表any)

*/main

拉取那个分支 就是存放项目的分支

Add Branch

源码库浏览器

(自动)

Additional Behaviours

新增

构建触发器

保存

应用

5、测试构建---点击立即构建---然后在工作空间中如果可以看到Girlab中的内容就说明拉取成功了

节点 master 上的工作空间 web



也可以在/var/lib/jenkins/workspace/下看

```
[root@jenkins web]# pwd
/var/lib/jenkins/workspace/web
[root@jenkins web]# ls
index.html  README.md
[root@jenkins web]#
```

jenkins中配置推送项目到web服务器

方案一：

配置Publish Over SSH，用于将拉取下来的代码推送到web服务器中

先要安装插件 Publish Over SSH

插件安装：系统管理 --> 插件管理 --> 可选插件 --> 搜索Publish Over SSH --> 安装即可（其他的插件也一样）

然后在Jenkins中生产公私钥文件，将公钥传到web服务器中

Publish over SSH 先安装这个插件才可以使用推送的功能

Jenkins SSH Key

Passphrase

Path to key

Key

jenkins上生成的私钥文件

SSH Servers

SSH Server

Name

webservers 自定义

Hostname

10.10.10.13 web服务器ip

Username

root wed服务器的用户

Remote Directory

/var/www/html 发布目录，也就是拉取后推送到的地方

Success 看到这个就代表成功了

点击测试

高级...

Test Configuration

删除

方案二：

采用ansible的方式进行推送，也就是说可以不使用 Publish Over SSH，而使用ansible的copy模块
ansible安装略....

配置：

将下面的打开

```
vim /etc/ansible/ansible.cfg
# uncomment this to disable SSH key host checking
# 不要询问指纹信息
host_key_checking = False
# 设置以root用户执行ansible
remote_user = root
```

然后在/var/lib/jenkins/.ssh中生成公私钥文件，然后将公钥文件拷贝到web服务器中

再在构建设置中选择执行shell的方式

在命令框中就可以ansible命令来进行拷贝了

执行 shell

命令

查看 可用的环境变量列表

高级...

增加构建步骤

- Invoke Ant
- Invoke Gradle script
- Run with timeout
- Send files or execute commands over SSH
- Set build status to "pending" on GitHub commit
- 执行 Windows 批处理命令
- 执行 shell 选择shell
- 调用顶层 Maven 目标

报如下错误：

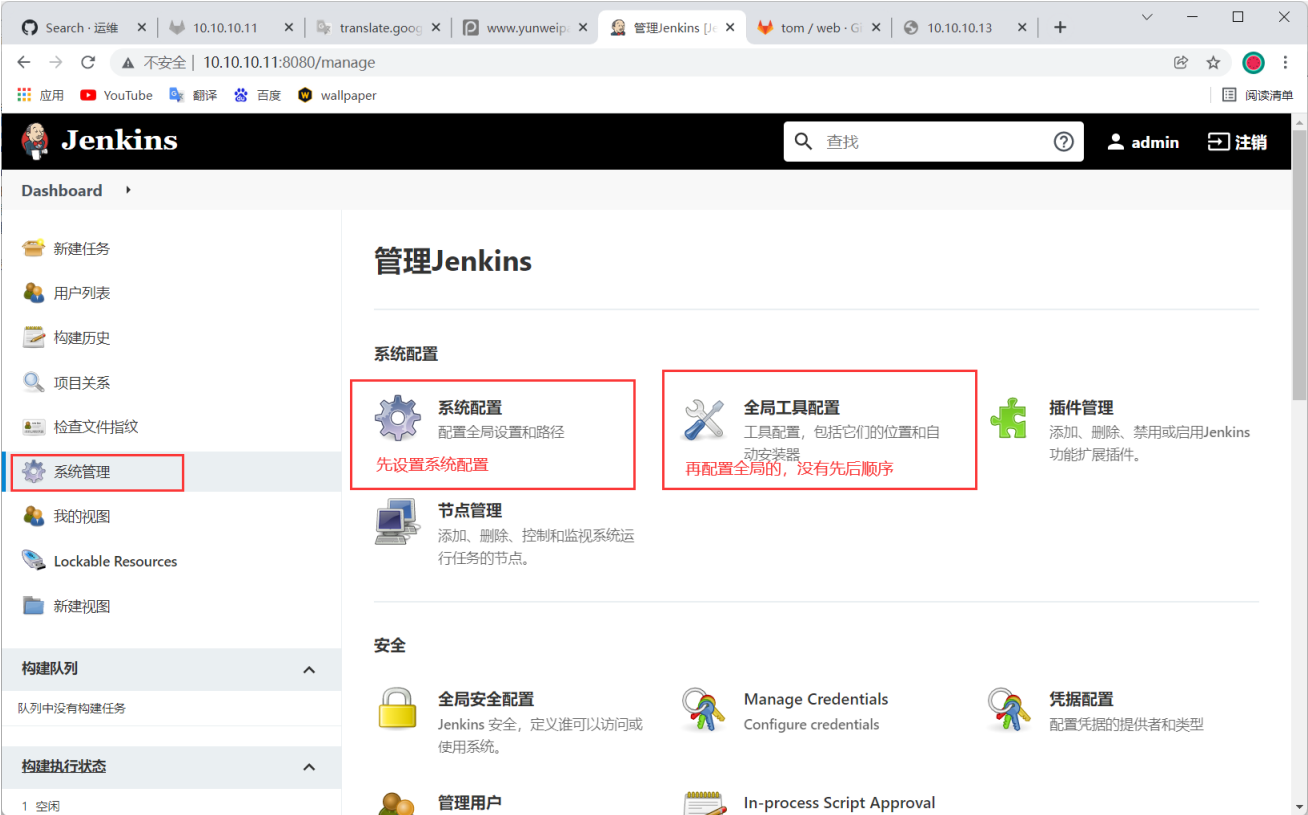
```
10.10.10.13 | UNREACHABLE! => {
  "changed": false,
  "msg": "Failed to connect to the host via ssh: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).",
  "unreachable": true
}
```

解决方案：

<https://www.cnblogs.com/rwxwsblog/p/5658703.html>

系统配置

配置系统环境



设置编码（本实验的可选项，不做不影响结果）

全局属性

☐ Disable deferred wipeout on this node

☐ 工具位置

☒ 环境变量

键值对列表

键

LANG

值

zh_CH.UTF-8

新增

删除

添加一个环境变量

指定jdk的路径

JDK

JDK 安装

新增 JDK

JDK

别名

jdk8 自定义

JAVA_HOME

/usr/local/java jenkins中jdk的绝对路径

☐ 自动安装

删除 JDK

新增 JDK

系统下JDK 安装列表

指定Maven路径（这个在该项目中用不到，后面需要构建的时候就会用到，在这里也可以先不配置，自由选择）

```
# 解压
tar -xf apache-maven-3.5.4-bin.tar.gz -C /usr/local/
# 创建一个软链接 方便版本的切换
ln -s /usr/local/maven-3.5.4 /usr/local/maven3
# 编辑环境变量
vim /etc/profile.d/maven3.sh
export MAVEN_HOME=/usr/local/maven3
export PATH=$MAVEN_HOME/bin:$PATH
# 使环境生效
. /etc/profile.d/maven3.sh
# 验证
mav -v
# 配置加速
vim /usr/local/maven3/conf/settings.xml
# 找到mirrors标签，加入以下内容：
<mirror>
  <id>alimaven</id>
  <name>aliyun maven</name>
  <url>https://maven.aliyun.com/nexus/content/groups/public/</url>
  <mirrorOf>central</mirrorOf>
</mirror>
```

完成后就可以配置了

Maven 需要在服务器中安装

Maven 安装

新增 Maven

Maven

Name

maven3 自定义

MAVEN_HOME

/usr/local/maven3 maven3 绝对路径

☐ 自动安装

删除 Maven

新增 Maven

系统下Maven 安装列表

验证

点击立即构建

就可以验证了，如果在web服务器上的/var/www/html可以看到Gitlab中的文件就代表成功了

也可以直接查看构建结果，或访问WEB网页

