|  |
| --- |
| jenkins2.0pipeline总结 |
| **作者：王金成**  **2018/5/11** |
| **快捷键：**  Ctrl + 1 标题1  Ctrl + 2 标题2  Ctrl + 3 标题3  Ctrl + 4 实例  Ctrl + 5 程序代码  Ctrl + 6 正文 |
| **格式说明：**  蓝色字体：注释  黄色背景：重要  绿色背景：注意 |

目 录

[第1章 pipeline模式介绍 1](#_Toc514319648)

[1.1 先决条件 1](#_Toc514319649)

[1.2 pipeline定义 1](#_Toc514319650)

[1.3 pipeline基本概念 1](#_Toc514319651)

[1.3.1 stage阶段 1](#_Toc514319652)

[1.3.2 node节点 1](#_Toc514319653)

[1.3.3 step步骤 1](#_Toc514319654)

[1.4 常用参数说明 1](#_Toc514319655)

[1.5 语法注意事项 2](#_Toc514319656)

[1.6 获取代码注意事项 2](#_Toc514319657)

[1.6.1 私钥Id查询地 2](#_Toc514319658)

[1.6.2 在配置时如下 2](#_Toc514319659)

[1.7 做动态赋值 3](#_Toc514319660)

[1.7.1 背景 3](#_Toc514319661)

[1.7.2 赋值方法 3](#_Toc514319662)

[第2章 pipeline脚本 3](#_Toc514319663)

[2.1.1 上述pipeline脚本中调用的shell脚本如下 5](#_Toc514319664)

[2.2 对应界面执行显示为 10](#_Toc514319665)

# pipeline模式介绍

Jenkins Pipeline是一套插件，支持将连续输送Pipeline实施和整合到Jenkins。Pipeline 提供了一组可扩展的工具，用于通过Pipeline DSL为代码创建简单到复杂的传送Pipeline 。

## 先决条件

要使用Jenkins Pipeline，您将需要：

* Jenkins 2.x或更高版本（旧版本回到1.642.3可能会工作，但不推荐）
* Pipeline插件

## pipeline定义

脚本Pipeline是用Groovy写的 。Groovy语法的相关位将在本文档中根据需要进行介绍，因此，当了解Groovy时，不需要使用Pipeline。

可以通过以下任一方式创建基本Pipeline：

* 直接在Jenkins网页界面中输入脚本。
* 通过创建一个Jenkinsfile可以检入项目的源代码管理库。

用任一方法定义Pipeline的语法是一样的，但是Jenkins支持直接进入Web UI的Pipeline，通常认为最佳实践是在Jenkinsfile Jenkins中直接从源代码控制中加载Pipeline。

## pipeline基本概念

### stage阶段

一个pipeline可以划分为若干个stage，每个stage代表一组操作。

注意：stage是一个逻辑分组的概念，可以跨越多个node

### node节点

一个node就是一个jenkins节点，或者是master或者是agent，是执行step的具体运行期环境

### step步骤

step是最基本的操作单元，小到创建一个目录，大到构建一个Docker镜像，由各类jenkins plugin提供

## 常用参数说明

|  |  |  |
| --- | --- | --- |
| 参数 | 含义 | 子参数 |
| dir | 目录 | delete/mkdir [删除/创建] |
| git | 拉取git仓库里的代码 | credentialsID/url/branches [私钥ID/地址/分支] |
| sh | 执行/bin/bash命令 |  |
| sleep | 暂停几秒 |  |
| input | 交互形式 |  |
| environment | 设置变量 |  |
| 基本语法参数 | 说明 | 注意 |
| agent | 指在jenkins中执行的位置 | 必须放置在顶层 |
| any | 在任何可用的agent上执行 | 列如：agent any |
| stage | 一个组操作的开始位置 | 此处定义的名称为此组执行显示名称 |
| steps | 你要执行什么动作 |  |

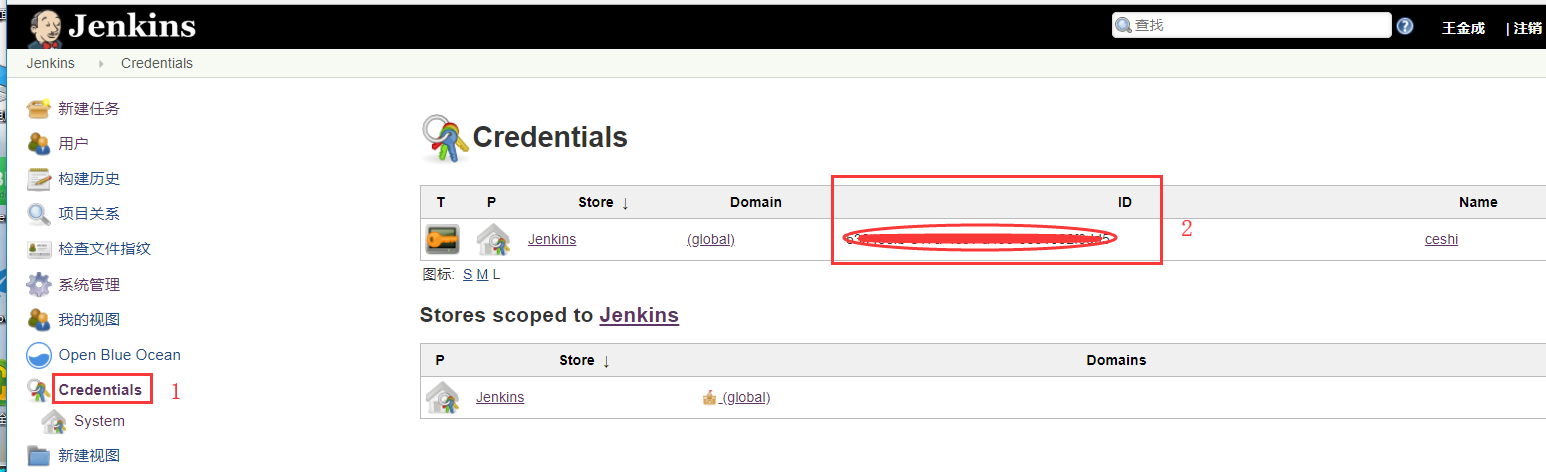
## 语法注意事项

1. 缩进默认为4个
2. 同级在同一个级别位
3. 要区分单引号与双引号的区别
4. 执行什么参数后面必须加上单引或双引
5. 在调用脚本里变量时，应用双引号

## 获取代码注意事项

pipeline模式配置里是没有配置gitlab连接方式的地方，此处在拉取时应选取jenkins里配置的连接管理账号中的ID号，如：

### 私钥Id查询地



### 在配置时如下

git credentialsId: 'ID号', url: 'Git仓库地址'

## 做动态赋值

### 背景

现在有这么一个需求，我需要在pipeline脚本里定义一个变量，但是这个变量的值是通过命令的方式取出来的。

### 赋值方法

environment {

def VARSION = sh returnStdout: true,script: 'cd ~/workspace/ceshi-5 && git tag|tail -2|head -1'

}

含义不清，得高人指教而来

# pipeline脚本

pipeline {

agent any

environment {

def VARSION = sh returnStdout: true,script: 'cd ~/workspace/ceshi-5 && git tag|tail -2|head -1'

}

stages{

stage('初始化') {

steps {

dir('/var/lib/jenkins/workspace/ceshi') {

deleteDir()

}

}

}

stage('拉取代码') {

steps {

sleep 1

git credentialsId: 'ID号', url: 'GIT仓库地址'

}

}

stage('测试代码') {

steps {

sleep 1

sh 'cd ~/workspace && ./devops-ceshi.sh ceshi-5 Test\_Code'

}

}

stage('Ant打包') {

steps {

sleep 1

sh 'cd ~/workspace && ./devops-ceshi.sh ceshi-5 Ant\_Bale'

}

}

stage('推送代码') {

steps {

sleep 1

sh 'cd ~/workspace && ./devops-ceshi.sh ceshi-5 Push\_Code'

}

}

stage('停服务') {

steps {

sleep 1

input('是否停服务')

sh 'cd ~/workspace && ./devops-ceshi.sh ceshi-5 Kill\_Serve'

}

}

stage('解压or升级') {

steps {

sleep 1

input ("备份的版本号: ${VARSION} 确定请继续")

sh "cd ~/workspace && ./devops-ceshi.sh ceshi-5 Upgrade ${VARSION}"

}

}

stage('起服务') {

steps {

sleep 1

input('是否起服务')

sh 'cd ~/workspace && ./devops-ceshi.sh ceshi-5 Start\_Serve'

}

}

}

}

### 上述pipeline脚本中调用的shell脚本如下

#!/bin/bash

# 变量内容

# 代码存放路径

DEST\_DIR=/var/lib/jenkins/workspace

# 任务名称

TASK=$1

# 执行哪个函数

EXECUTE=$2

# sonar-scanner家目录

SONAR=/usr/local/sonar-scanner/bin

# git版本提取

VARSION=$3

# 函数

# 校测代码

Test\_Code(){

${SONAR}/sonar-scanner -X -Dsonar.host.url=sonarqube\_URL -Dsonar.login=账户 -Dsonar.password=密码 -Dsonar.language=java -Dsonar.projectName=${TASK} -Dsonar.projectVersion=1.0 -Dsonar.sourceEncoding=GBK -Dsonar.projectKey=${TASK} -Dsonar.my.property=value -Dsonar.java.binaries=${DEST\_DIR} -Dsonar.sources=./ -Dsonar.projectBaseDir=${DEST\_DIR}/${TASK}

}

# 编译代码

Ant\_Bale(){

source /etc/profile

ant -buildfile ${DEST\_DIR}/${TASK}-build.xml

}

# 传送代码

Push\_Code(){

[ -d /tmp/${TASK} ] || mkdir -p /tmp/${TASK} && rm -r /tmp/${TASK}/\*

cp -r ${DEST\_DIR}/${TASK}.war /tmp/${TASK} && cd /tmp/${TASK} && unzip ${TASK}.war

rsync -r /tmp/${TASK}/ ansible主机:~/${TASK}-rpt/

}

# kill服务

Kill\_Serve(){

ssh ansible主机 "ansible -i ~/hosts ceshi -m shell -a 'cd && ./stop.sh'"

}

# Ansible升级

Upgrade(){

ssh ansible主机 "ansible-playbook -i ~/hosts -e host=ceshi -e lj=~/${TASK}-rpt/ -e v=${VARSION} ~/剧本路径"

}

# 起服务

Start\_Serve(){

ssh ansible主机 'ansible -i ~/hosts IP -m shell -a "cd ~ && ./start.sh"'

ssh ansible主机 'ansible -i ~/hosts IP -m shell -a "cd ~ && ./start.sh"'

}

# 锁文件

Lock\_File(){

touch /tmp/zte.lock

}

# 解锁文件

Unbind\_Lock\_File(){

rm -f /tmp/zte.lock

}

# 版本回退

Varsion\_Back(){

ssh ansible主机 "ansible -i ~/hosts ceshi -m shell -a 'cd 代码路径 && mv resweb.war-${VARSION} resweb.war'"

}

# 执行与判断

if [ -f /tmp/zte.lock ];then

echo "脚本正在运行，程序退出"

fjdsfnds

exit

fi

# 运行哪些命令

case ${EXECUTE} in

Test\_Code)

Lock\_File

Test\_Code

if [ $? -ne 0 ];then

echo "检测代码失败"

Unbind\_Lock\_File

hjaid

exit

fi

Unbind\_Lock\_File

;;

Ant\_Bale)

Lock\_File

Ant\_Bale

if [ $? -ne 0 ];then

echo "ant打包失败"

Unbind\_Lock\_File

jhdaod

exit

fi

Unbind\_Lock\_File

;;

Push\_Code)

Lock\_File

Push\_Code

if [ $? -ne 0 ];then

echo "推送代码失败"

Unbind\_Lock\_File

dhajsd

exit

fi

Unbind\_Lock\_File

;;

Upgrade)

Lock\_File

Upgrade

if [ $? -ne 0 ];then

echo "Ansible升级失败"

Unbind\_Lock\_File

dhjahda

exit

fi

Unbind\_Lock\_File

;;

Varsion\_Back)

Lock\_File

Varsion\_Back

if [ $? -ne 0 ];then

echo "版本回退失败"

Unbind\_Lock\_File

ndjadadkja

exit

fi

Unbind\_Lock\_File

;;

Kill\_Serve)

Lock\_File

Kill\_Serve

if [ $? -ne 0 ];then

echo "kill 失败"

Unbind\_Lock\_File

dajfa

exit

fi

Unbind\_Lock\_File

;;

Start\_Serve)

Lock\_File

Start\_Serve

if [ $? -ne 0 ];then

echo "启动失败"

Unbind\_Lock\_File

dnakjda

exit

fi

Unbind\_Lock\_File

;;

\*)

echo "脚本运行错误，输入正确的选项"

jfhsjd

exit

;;

esac

**注：由于pipeline语法太多，需消耗学习成本，故此用shell脚本取代一些功能**

## 对应界面执行显示为

