## Groovy

### 声明式管道基础知识

<https://www.jenkins.io/doc/book/pipeline/>

|  |  |
| --- | --- |
|  | [pipeline](https://www.jenkins.io/doc/book/pipeline/syntax#declarative-pipeline)是特定于声明式管道的语法，它定义了一个“块”，其中包含用于执行整个管道的所有内容和指令。 |
|  | [agent](https://www.jenkins.io/doc/book/pipeline/syntax#agent)是特定于声明式 Pipeline 的语法，指示 Jenkins 为整个 Pipeline 分配执行器（在节点上）和工作空间。 |
|  | stage是一个语法块，描述 [此 Pipeline 的一个阶段](https://www.jenkins.io/doc/book/pipeline/#stage)。[在管道语法](https://www.jenkins.io/doc/book/pipeline/syntax#stage)页面上阅读有关stage声明性管道语法中的块的更多信息。如上所述，块在脚本化管道语法中是可选的[。](https://www.jenkins.io/doc/book/pipeline/#scripted-pipeline-fundamentals)stage |
|  | [steps](https://www.jenkins.io/doc/book/pipeline/syntax#steps)是特定于声明式管道的语法，描述了要在此 中运行的步骤stage。 |
|  | sh是执行给定 shell 命令的管道[步骤（由](https://www.jenkins.io/doc/book/pipeline/syntax#steps)[管道：节点和进程插件](https://plugins.jenkins.io/workflow-durable-task-step)提供 ）。 |
|  | junit是另一个用于聚合测试报告的管道[步骤（由](https://www.jenkins.io/doc/book/pipeline/syntax#steps)[JUnit 插件](https://plugins.jenkins.io/junit)提供 ）。 |
|  | sh是执行给定 shell 命令的管道[步骤（由](https://www.jenkins.io/doc/book/pipeline/syntax#steps)[管道：节点和进程插件](https://plugins.jenkins.io/workflow-durable-task-step)提供 ）。 |

### shell 命令

<https://www.runoob.com/linux/linux-shell.html>

用户使用 Linux 的桥梁。Shell 既是一种命令语言，又是一种程序设计语言。

Shell 是指一种应用程序，这个应用程序提供了一个界面，用户通过这个界面访问操作系统内核的服务。

像 **#!/bin/sh**，它同样也可以改为 **#!/bin/bash**。

**#!** 告诉系统其后路径所指定的程序即是解释此脚本文件的 Shell 程序。

**#!** 是一个约定的标记，它告诉系统这个脚本需要什么解释器来执行，即使用哪一种 Shell。

扩展名为 sh（sh代表shell），扩展名并不影响脚本执行

### 运行 Shell 脚本有两种方法：

**1、作为可执行程序**

将上面的代码保存为 test.sh，并 cd 到相应目录：

chmod +x ./test.sh #使脚本具有执行权限

./test.sh #执行脚本

注意，一定要写成 **./test.sh**，而不是 **test.sh**，运行其它二进制的程序也一样，直接写 test.sh，linux 系统会去 PATH 里寻找有没有叫 test.sh 的，而只有 /bin, /sbin, /usr/bin，/usr/sbin 等在 PATH 里，你的当前目录通常不在 PATH 里，所以写成 test.sh 是会找不到命令的，要用 ./test.sh 告诉系统说，就在当前目录找。

**2、作为解释器参数**

这种运行方式是，直接运行解释器，其参数就是 shell 脚本的文件名，如：

/bin/sh test.sh

/bin/php test.php

这种方式运行的脚本，不需要在第一行指定解释器信息，写了也没用。

### Shell 变量

your\_name="runoob.com"

变量名和等号之间不能有空格

set命令用来修改 Shell 环境的运行参数，也就是可以定制环境。

如果命令行下不带任何参数，直接运行set，会显示所有的环境变量和 Shell 函数。

默认情况下，脚本执行后，屏幕只显示运行结果，没有其他内容。如果多个命令连续执行，它们的运行结果就会连续输出。有时会分不清，某一段内容是什么命令产生的。

set -x用来在运行结果之前，先输出执行的那一行命令。

可以看到，执行echo bar之前，该命令会先打印出来，行首以+表示。这对于调试复杂的脚本是很有用的。

shopt，即shell option，用于显示和设置shell中的行为选项，通过这些选项以增强shell易用性。

**2、用法**

shopt [-sup] [optionName]

* -s：打开某个选项
* -u：关闭某个选项
* -p：列出所有可设置的选项

shopt的选项很多，常用的有extglob，用于开启扩展模式匹配。开启之后Shell可以另外识别出5个模式匹配操作符，能使文件匹配更加方便。

- 查看extglob选项的状态

shopt extglob

- 开启extglob

shopt -s extglob

- 关闭extglob

shopt -u extglob

开启extglob，Shell可以另外识别出5个模式匹配操作符：

匹配操作符 含义

?(pattern-list) 所给模式匹配0次或1次

\*(pattern-list) 所给模式匹配0次以上，包括0次

+(pattern-list) 所给模式匹配1次以上，包括1次

@(pattern-list) 所给模式仅仅匹配一次

！(pattern-list) 不匹配括号内的所给模式

Linux rm（英文全拼：remove）命令用于删除一个文件或者目录。

**语法**

rm [options] name...

**参数**：

* -i 删除前逐一询问确认。
* -f 即使原档案属性设为唯读，亦直接删除，无需逐一确认。
* -r 将目录及以下之档案亦逐一删除。

### if else

### fi

if 语句语法格式：

if condition

then

command1

command2

...

commandN

fi

写成一行（适用于终端命令提示符）：

if [ $(ps -ef | grep -c "ssh") -gt 1 ]; then echo "true"; fi

末尾的 **fi** 就是 **if** 倒过来拼写，后面还会遇到类似的。

if else 的 **[...]** 判断语句中大于使用 **-gt**，小于使用 **-lt**。

if [ "$a" -gt "$b" ]; then

...

fi

如果使用 **((...))** 作为判断语句，大于和小于可以直接使用 **>** 和 **<**。

if (( a > b )); then

...

fi

以下实例判断两个变量是否相等：

### 实例

a=10  
b=20  
**if** **[** $a == $b **]**  
**then**  
   **echo** "a 等于 b"  
**elif** **[** $a -gt $b **]**  
**then**  
   **echo** "a 大于 b"  
**elif** **[** $a -lt $b **]**  
**then**  
   **echo** "a 小于 b"  
**else**  
   **echo** "没有符合的条件"  
**fi**

[Shell if 条件判断 - 小白一生 - 博客园 (cnblogs.com)](https://www.cnblogs.com/liudianer/p/12071476.html)

[ -d DIR ] 如果 FILE 存在且是一个目录则为真。

Git 提供了一个名为 git clean 的命令，可以帮助您从工作目录中删除这些未被跟踪的文件和目录。

使用 -d 选项除了未被跟踪的文件外，还可以删除未被跟踪的目录。当您想从工作目录中清除所有生成的构建工件以及其他未被跟踪的目录时，此选项非常有用。

-f 选项告诉 Git 强制删除未被跟踪的文件和目录。默认情况下，Git 不会删除被 .gitignore 忽略的

[Git clean 命令全面指南|极客笔记 (deepinout.com)](https://deepinout.com/git/git-tutorial/16_a_comprehensive_guide_to_the_git_clean_command.html#ftoc-heading-4)

**-x**

不要使用标准的忽略规则（见[gitignore[5]](https://git-scm.com/docs/gitignore/zh_HANS-CN)），但仍然使用命令行中用 -e 选项给出的忽略规则。 这允许删除所有未跟踪的文件，包括构建产品。 这可以用来（可能与 **git restore** 或 **git reset** 一起使用）创建一个原始的工作目录来测试一个干净的构建。

Git reset 是一种用于撤销或恢复 Git 仓库中所做更改的命令。它用于将当前分支移动到特定提交，使其成为分支的新 HEAD。这可以用于将仓库重置为先前的状态或丢弃自上次提交以来所做的更改。

* **硬重置（Hard）：** 该模式是最严格的一种，它将分支的 HEAD 重置为以前的提交，在该提交之后丢弃所有已做出的更改。当你想要丢弃对文件所做出的某些更改或从头开始时，该模式很有用。
* **硬重置**
* 要执行硬重置，请使用以下命令：
* git reset --hard HEAD^
* Bash
* Copy
* 这将将 HEAD 重置为上一个提交并丢弃自该提交以来的所有更改。在使用此模式时要小心，因为它将永久删除对文件所做的所有更改。

git pull命令是最重要的Git命令之一，它从远程存储库中获取并合并更改到本地存储库。

### 理解Git Pull

在Git中，仓库有两个副本-一个本地副本和一个远程副本。远程副本通常位于远程服务器上，多个开发人员可以访问它以合作完成项目。本地存储库位于您的计算机上并映射远程存储库。

在提交任何更改之前，确保使用最新版本的代码库很重要。 git pull命令可帮助您从远程存储库检索更改并将其整合到存储库的本地分支中。

$? 是上一个程序执行是否成功的标志，如果执行成功则$? 为0，否则 不为0

INT1 -ne INT2           INT1和INT2两数不等为真 ,<>

1. 回到上一层目录  
     
    cd..

git clone 命令是 Git 中一个重要的部分，它可以让开发者从远程服务端下载整个仓库到本地机器。使用这个命令，开发者可以获取到包含所有代码、历史记录和配置信息的仓库的完整副本

[Jenkins: 如何更新当前构建名称和描述\_DevOps云学堂的博客-CSDN博客](https://blog.csdn.net/weixin_40046357/article/details/128108157)

通过不打开即可查看构建细节本身，我们可以获得所有重要细节。这可以通过更新currentBuild.displayName和currentBuild.description变量来实现。

链接的 HTML 代码很简单，它类似这样：

<a href="url">链接文本</a>

* href：指定链接目标的URL，这是链接的最重要属性。可以是另一个网页的URL、文件的URL或其他资源的URL。

（就是会有一个显示为设置的链接文本的按钮，点击按钮跳转到href指定的url界面）

### 处理目录的常用命令

接下来我们就来看几个常见的处理目录的命令吧：

* ls（英文全拼：list files）: 列出目录及文件名
* -l ：长数据串列出，包含文件的属性与权限等等数据；(常用)

Linux mkdir（英文全拼：make directory）命令用于创建目录。

**语法**

mkdir [-p] dirName

**参数说明**：

* -p 确保目录名称存在，不存在的就建一个。

.表示执行的意思，就是执行这个文件。

./呢就表示执行当前目录下的某个文件，就比如当前目录有一个脚本a[.sh](https://zhidao.baidu.com/search?word=.sh&fr=iknow_pc_qb_highlight)，那么./a.sh就表示执行它。

${ } ：  
${ }用于变量替换。一般情况下，$var 与${var} 并没有啥不一样。但是用 ${ } 会比较精确的界定变量名称的范围。

## Python：

[【python】re模块的用法（1）——基本语法\_import re\_AI小白龙的博客-CSDN博客](https://blog.csdn.net/qq_34106574/article/details/83859835)

#### re

模块（Regular Expression [正则表达式](https://so.csdn.net/so/search?q=%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F&spm=1001.2101.3001.7020)）提供各种正则表达式的匹配操作

#### os

就是“operating system”的缩写，顾名思义，os模块提供的就是各种 Python 程序与操作系统进行交互的接口。通过使用os模块，一方面可以方便地与操作系统进行交互，另一方面页可以极大增强代码的可移植性。如果该模块中相关功能出错，会抛出OSError异常或其子类异常。

[Python os 模块详解 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/150835193)

#### (1)sys.argv

很多人会想，我如何给我的程序在外部传递参数呢？这个，就可以实现。如：

Tesy.py

Import sys

Print sys.argv[number]

一般情况下，number为0是这个脚本的名字，1，2…则为命令行下传递的参数.如：

Test.py脚本内容：

import sys

print sys.argv[0]

print sys.argv[1]

print sys.argv[2]

print sys.argv[3]

那么

[root@databak scripts]# python test.py arg1 arg2 arg3

test.py

arg1

arg2

arg3

看到，对应的关系了吗？

sys模块中的argv变量通过使用点号指明——sys.argv——这种方法的一个优势是这个名称不会与任何在你的程序中使用的argv变量冲突。另外，它也清晰地表明了这个名称是sys模块的一部分。

sys.argv变量是一个字符串的 列表 （列表会在后面的章节详细解释）。特别地，sys.argv包含了 命令行参数 的列表，即使用命令行传递给你的程序的参数。

这里，当我们执行python using\_sys.py we are arguments的时候，我们使用python命令运行using\_sys.py模块，后面跟着的内容被作为参数传递给程序。Python为我们把它存储在sys.argv变量中。

记住，脚本的名称总是sys.argv列表的第一个参数。所以，在这里，'using\_sys.py'是sys.argv[0]、'we'是sys.argv[1]、'are'是sys.argv[2]以及'arguments'是sys.argv[3]。注意，Python从0开始计数，而非从1开始。

#### Json

模块如下语句引入：

import json

#### openpyxl

模块中与字体，颜色，背景，边框，字体，行宽，行高，单元格格式等相关的内容都放在了styles模块中

单元格颜色填充PatternFill(patternType=‘ ’)

sys.path指定模块搜索路径的列表。默认情况下，python导入文件或者模块，会在sys.path里找模块的路径。如果在当前搜索路径列表sys.path中找不到该模块的话，就会报错。

用法

最常用的用法就是向sys.path中添加搜索路径

import sys

sys.path.append(path) # path 代表的是一个路径

如果是路径相关的操作，建议使用os的子模块os.path

#### 3.4 os.path.dirname()

与上一个函数正好相反，返回的是最后一个分隔符前的整个字符串：

>>> os.path.dirname("/ityouknow/justdopython/IAmBasename")

'/ityouknow/justdopython'

>>>

>>> os.path.dirname("/ityouknow/justdopython/IAmBasename/")

'/ityouknow/justdopython/IAmBasename'

[Python os 模块详解 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/150835193)

#### 3.2 os.path.abspath()

将传入路径规范化，返回一个相应的绝对路径格式的字符串。

也就是说当传入路径符合“绝对路径”的格式时，该函数仅仅将路径分隔符替换为适应当前系统的字符，不做其他任何操作，并将结果返回。所谓“绝对路径的格式”，其实指的就是一个字母加冒号，之后跟分隔符和字符串序列的格式：

**>>>** os**.**path**.**abspath("a:/just/do/python")

'a:\\just\\do\\python'

**>>>** *# 我的系统中并没有 a 盘*

当指定的路径不符合上述格式时，该函数会自动获取当前工作路径，并使用os.path.join()函数将其与传入的参数组合成为一个新的路径字符串。示例如下：

**>>>** os**.**path**.**abspath("ityouknow")

'D:\\justdopython\\ityouknow'

通过调用 \_\_file\_\_ 属性输出的绝对路径，我们可以很轻易地找到该模块（或包）的源文件。

#### ssh

**1、查看所有控制台命令：**

     $ ssh -p 29418 admin@127.0.0.0 gerrit --help     (使用自己的gerrit地址IP)

[Gerrit常用命令\_gerrit命令\_a420344的博客-CSDN博客](https://blog.csdn.net/a420344/article/details/120302183)

[gerrit query 命令的简单应用 - 个人文章 - SegmentFault 思否](https://segmentfault.com/a/1190000020443777)

ssh -p <port> <host> gerrit query

[--format {TEXT | JSON}]

[--current-patch-set]

[--patch-sets | --all-approvals]

[--files]

[--comments]

[--commit-message]

[--dependencies]

[--submit-records]

[--all-reviewers]

[--start <n> | -S <n>]

[--no-limit]

[--]

<query>

[limit:<n>]

<port> ： 是指gerrit服务器的端口， 一般29418  
<host>： 是指gerrit服务器的地址， 一般username@gerrit.XXX.com， 或者gerrit.XXX.com

#### Json.loads

[Python中json模块的load/loads方法实战及参数详解\_json.loads()方法\_荒野雄兵的博客-CSDN博客](https://blog.csdn.net/daerzei/article/details/100598901)

loads操作的是字符串

#### workbook()

1. **from** openpyxl **import** Workbook
2. *# 创建一个 workbook*
3. wb **=** Workbook()
4. *# 获取被激活的 worksheet*
5. ws **=** wb**.**active
6. *# 设置单元格内容*
7. ws['A1'] **=** 42
8. *# 设置一行内容*
9. ws**.**append([1, 2, 3])
10. *# python 数据类型可以被自动转换*
11. **import** datetime
12. ws['A2'] **=** datetime**.**datetime**.**now()
13. *# 保存 Excel 文件*
14. wb**.**save("sample.xlsx")

#### **re.sub()**

的详细用法，该函数主要**用于替换字符串中的匹配项**。

*# 获取被激活的 worksheet*

ws **=** wb**.**active

font **=** Font(bold**=**True)

*# 遍历范围内的单元格*

**for** row **in** ws['A1:C3']:

**for** cell **in** row:

cell**.**font **=** font

*# 设置整行*

row **=** ws**.**row\_dimensions[1]

row**.**font **=** font

#### *# 设置整列*

column **=** ws**.**column\_dimensions["A"]

column**.**font **=** font

run.font.bold = True #加粗

#### time

[Python学习：python time模块之time.strftime()\_Carol.Carol的博客-CSDN博客](https://blog.csdn.net/sinat_37960022/article/details/110946414)

[Python函数： time.strftime()、time.localtime()、time.time()\_time.strftime()函数的作用\_小小白2333的博客-CSDN博客](https://blog.csdn.net/Ajdidfj/article/details/123043872)

#### [list.append(obj)](https://www.runoob.com/python3/python3-att-list-append.html) 在列表末尾添加新的对象

[正则表达式re.compile()的使用\_re.compile 写正则\_艾莉宝贝的博客-CSDN博客](https://blog.csdn.net/weixin_42793426/article/details/88545939)

#### **compile**

**函数用于编译正则表达式，生成一个 Pattern 对象**，它的一般使用形式如下：

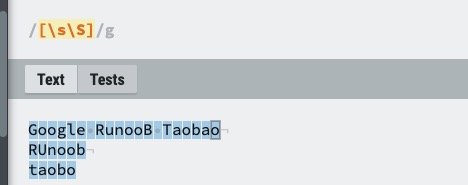
re.compile(pattern[, flag])

其中，pattern 是一个字符串形式的[正则表达式](https://so.csdn.net/so/search?q=%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F&spm=1001.2101.3001.7020)，flag 是一个可选参数，表示匹配模式，比如忽略大小写，多行模式等。

#### 正则表达式

* 字符类：用方括号 [ ] 包围的字符集合，用于匹配方括号内的任意一个字符。
* 元字符：例如 \d、\w、\s 等，用于匹配特定类型的字符，如数字、字母、空白字符
* 特殊字符：例如点号 .、星号 \*、加号 +、问号 ? 等，它们具有特殊的含义和功能。

|  |  |
| --- | --- |
| **[ABC]** | 匹配 **[...]** 中的所有字符，例如 **[aeiou]** 匹配字符串 "google runoob taobao" 中所有的eoua 字母。  https://www.runoob.com/wp-content/uploads/2014/03/E691DDE1-E5CB-4EA8-8D16-759BD0D2B09D.jpg |
| **[\s\S]** | 匹配所有。\s 是匹配所有空白符，包括换行，\S 非空白符，不包括换行。 |



|  |  |
| --- | --- |
| \* | 匹配前面的子表达式零次或多次。例如，**zo\*** 能匹配 **"z"** 以及 **"zoo"**。**\*** 等价于 **{0,}**。 |