

# Jenkins持续集成课件

## 1.什么是CI/CD

- 持续集成

频繁地（一天多次）将代码集成到主干。将软件个人研发的部分向软件整体部分 交付，频繁进行集成以便更快地发现其中的错误。快速发现错误。每完成一点更新，就集成到主干，可以快速发现错误，定位错误也比较容易，防止分支大幅偏离主干。如果不是经常集成，主干又在不断更新，会导致以后集成的难度变大，甚至难以集成。持续集成的目的，就是让产品可以快速迭代，同时还能保持高质量。它的核心措施是，代码集成到主干之前，必须通过自动化测试。只要有一个测试用例失败，就不能集成

- 持续交付 (Continuous delivery)

频繁地将软件的新版本，交付给质量团队或者用户，以供 评审。如果评审通过，代码就进入生产阶段。持续交付在持续集成的基础上，将集成后的代码部署到更贴近真实运行环境的「类生产环境」(production-like environments)中。持续交付优先于整个产品生命周期的软件部署，建立在高水平自动化持续集成之上。

- 持续部署 (continuous deployment)

是持续交付的下一步，指的是代码通过评审以后，自动部署到 生产环境。持续部署的目标是，代码在任何时刻都是可部署的，可以进入生产阶段。持续部署的前提 是能自动化完成测试、构建、部署等步骤。

## 2.什么是jenkins

Jenkins是一个开源的、可扩展的持续集成、交付、部署（软件/代码的编译、打包、部署）的 基于web界面的平台。允许持续集成和持续交付项目，无论用的是什么平台，可以处理任何类型的构建或持续集成。一般部署在服务器端，使用分布式的构建策略处理不同的任务。

Jenkins官网: <https://www.jenkins.io/>

Jenkins war包下载: <http://mirrors.jenkins.io/war-stable> 本课件是基于2.289版本

## 3.jenkins搭建

- jdk安装

参考群文件java环境搭建

- jenkins安装

群文件下载jenkins.war或者官网下载war包

### 1. 启动jenkins并安装

- 命令执行如下命令，启动后窗口不要关

```
java -DJENKINS_HOME=D:\jenkins\jenkins_data -jar
D:\jenkins\jenkins.war
```

```

ver
2021-11-02 03:21:48.705+0000 [id=29] INFO jenkins.install.SetupWizard#init:
*****
*****
*****
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:
a05c9f9779b2463b9491c4606617ca77
This may also be found at: D:\jenkins\jenkins_data\secrets\initialAdminPassword
*****
*****
*****
2021-11-02 03:21:56.128+0000 [id=57] INFO h.m.DownloadService$Downloadable#load: Obtained the updated data file fo
r hudson.tasks.Maven.MavenInstaller
2021-11-02 03:21:56.129+0000 [id=57] INFO hudson.util.Retrier#start: Performed the action check updates server suc
cessfully at the attempt #1
2021-11-02 03:21:56.134+0000 [id=57] INFO hudson.model.AsyncPeriodicWork#lambda$doRun$0: Finished Download metadat
a. 7,656 ms
2021-11-02 03:21:58.969+0000 [id=44] INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization
2021-11-02 03:21:58.981+0000 [id=23] INFO hudson.WebAppMain$3#run: Jenkins is fully up and running

```

## 2. 浏览器完成配置步骤

- 浏览器打开<http://localhost:8080>

入门

# 解锁 Jenkins

为了确保管理员安全地安装 Jenkins，密码已写入到日志中（[不知道在哪里?](#)）该文件在服务器上：

`D:\jenkins\jenkins_data\secrets\initialAdminPassword` 找到这个文件复制其中的内容输入到下边的输入框

请从本地复制密码并粘贴到下面。

管理员密码

继续

- 安装推荐的插件

点击后需要耐心等待



## ■ 创建用户

新手入门

# 创建第一个管理员用户

用户名:

admin

密码:

.....

确认密码:

.....

全名:

admin

电子邮件地址:

2879897713@qq.com

Jenkins 2.289.1

[使用admin账户继续](#) [保存并完成](#)

## ■ 实例配置

默认即可

# 实例配置

Jenkins URL:

http://localhost:8080/

Jenkins URL 用于给各种Jenkins资源提供绝对路径链接的根地址。这意味着对于很多Jenkins特色是需要正确设置的，例如：邮件通知、PR状态更新以及提供给构建步骤的BUILD\_URL环境变量。

推荐的默认值显示在尚未保存，如果可能的话这是根据当前请求生成的。最佳实践是要设置这个值，用户可能会需要用到。这将会避免在分享或者查看链接时的困惑。

Jenkins 2.289.1

现在不要

保存并完成


## 开始使用

# Jenkins已就绪！

Jenkins安装已完成。

开始使用Jenkins

Jenkins 2.289.1

 **Jenkins**

Dashboard

新建Item

用户列表

构建历史

Manage Jenkins

My Views

Lockable Resources

New View

构建队列

队列中设有构建任务

构建执行状态

1 空闲

2 空闲

欢迎来到 Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds

添加说明

## 4.jenkins插件安装

jenkins插件本身很多，我们这里安装的只是课程上会用到的

进入[Manage Jenkins]->[Manage Plugins]，选择可选插件标签，搜索以下插件并安装，所有都安装完了以后按照提示重启

- HTML Publisher 收集普通的html报告，将其转换为jenkins的一个页面
- Groovy 用来设置一条命令，兼容第三方的css和js的命令
- allure 收集allure测试报告的

## 5.全局工具配置

进入Manage Jenkins-->Global Tool Configuration，中文的话就是系统管理-->全局工具配置，依次配置各个工具

- jdk

点击新增，取消自动安装，填写jenkins所在服务器的jdk路径



JDK

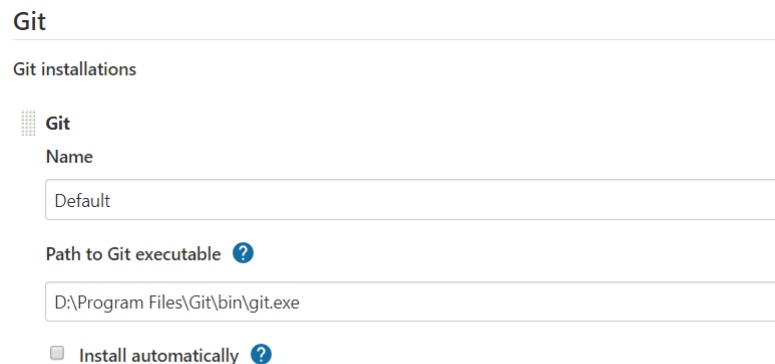
JDK 安装

新增 JDK

JDK	别名	JAVA_HOME	Install automatically
JDK	default	C:\Program Files\Java\jdk1.8.0_121	<input type="checkbox"/>

- git

先在jenkins所在服务器上手动安装git，取消自动安装



Git

Git installations

Git	Name	Path to Git executable	Install automatically
Git	Default	D:\Program Files\Git\bin\git.exe	<input type="checkbox"/>

- allure

先在jenkins所在服务器上手动安装allure，取消自动安装

## Allure Commandline

Allure Commandline 安装

新增 Allure Commandline

Allure Commandline

别名

Default


安装目录

D:\allure-2.11.0

☐ Install automatically

## 6.git凭据配置


进入Manage Jenkins-->Manage Credentials，按照下图操作

 凭据

类型	提供者	存储 ↓	域	唯一标识
----	-----	------	---	------

图标 小 中 大

### Stores scoped to Jenkins

提供者	存储 ↓	域
	Jenkins	全局

添加凭据

类型

Username with password

范围

全局 (Jenkins, nodes, items, all child items, etc)

用户名

sandroad 输入git用户名

☐ Treat username as secret

密码

..... 输入git密码

ID

描述

沙陌git用户

确定

## 7. 邮件配置

以下以 QQ 邮箱为例，企业中需要和邮箱管理员沟通

### 1. 先开启qq邮箱的smtp服务

- 开启 QQ 邮箱 SMTP 服务，打开 QQ 邮箱进入设置



- 进入帐户



- 开启 SMTP 服务



- 发送验证码到指定号码



- 点击确定，SMTP 服务已开启



### 2. jenkins中配置邮箱

进入系统管理-->系统配置，做如下配置

- 增加系统管理员邮件地址

Jenkins Location	
Jenkins URL	http://192.168.2.184:8080/jenkins/
系统管理员邮件地址	2879897713@qq.com

- 邮件配置测试，注意这里只是测试  
找邮件通知的区域，进行配置及测试

邮件通知

SMTP服务器: smtp.qq.com (一般都是以smtp开头，加上某个邮箱的xx.com，公司里需要询问邮箱管理员)

用户默认邮件后缀: @qq.com

☒ 使用SMTP认证

用户名: 2879897713@qq.com (用户名)

密码: ..... (注意这里不是qq邮箱的登录密码，是在上一步我们拿到的qq邮箱授权码)

☐ 使用SSL协议

☐ Use TLS

SMTP端口: ( ? )

Reply-To Address: ( ? )

字符集: UTF-8

☒ 通过发送测试邮件测试配置 (勾选测试)

Test e-mail recipient: 2879897713@qq.com (输入收件人邮箱)

Test configuration (点击测试)

点击测试后提示成功，并且邮箱会收到一封邮件，说明邮件发送没有什么问题

- 配置扩展邮箱

该配置将会作为我们jenkins任务执行完成后的邮件模板

找到Extended E-mail Notification区域，做如下配置，注意点高级才能出现用户名和密码

Extended E-mail Notification

SMTP server: smtp.qq.com

SMTP Port: 25

SMTP Username: 2879897713@qq.com

SMTP Password: ..... (授权码)

如果想看任务发送邮件时的详细日志可以勾选下述选项，主要是邮件发送有问题时进行调试使用的

新增

☒ Enable Debug Mode

☐ Require Administrator for Template Testing

☐ Enable watching for jobs

☐ Allow sending to unregistered users




## 8.自动化任务配置

- 点击首页的新建Item


### 输入一个任务名称

apiautotest任务名称


» 必填项

 **Freestyle project**


This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build tool used for something other than software build.

 **Pipeline**


Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines and/or organizing complex activities that do not easily fit in free-style job type.

 **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.

 **Organization Folder**

Creates a set of multibranch project subfolders by scanning for repositories.

 **多分支流水线**

根据 SCM 仓库中检测到的分支创建一系列流水线。

确定

- 定义构建参数

☒ This project is parameterized

Choice Parameter

名称 ?

env\_name

选项 ?

test  
prepro  
pro

这里写各个环境的名称，和代码中定义的一致

- 配置自动化脚本源码地址

### 源码管理

☐ 无

☒ Git

Repositories

Repository URL

https://gitee.com/sandroad/apiautotest.git输入自动化脚本代码git仓库地址

Credentials

sandroad/\*\*\*\*\* (沙陌git用户)添加

高级...

Add Repository

Branches to build

指定分支 (为空时代表any)

\*/master这里默认是主干

- 构建触发器配置

触发器的意思就是在什么时候去执行任务，下面会讲解三种触发方式，工作中不一定是三种都配

1. 由上游其他任务驱动
2. 定时执行
3. 定时轮询版本库(一般测试任务不用)

**构建触发器**

☒ Build whenever a SNAPSHOT dependency is built

☐ Schedule build when some upstream has no successful builds

☐ 触发远程构建 (例如, 使用脚本)

☒ Build after other projects are built

关注的项目: 开发的代码打包部署任务

☒ 只有构建稳定时触发

☐ 即使构建不稳定时也会触发

☐ 即使构建失败时也会触发

☒ Build periodically

日程表: 0 0 \* \* \*

⚠️ 分散负载应该用 'H 0 \* \* \*' 而不是 '0 0 \* \* \*'

Would last have run at 2020年7月23日 星期四 上午12时00分18秒 CST; would next run at 2020年7月24日 星期五 上午12时00分18秒 CST.

☒ Poll SCM

日程表: H H/2 \* \* \*

Would last have run at 2020年7月23日 星期四 上午09时57分49秒 CST; would next run at 2020年7月23日 星期四 上午11时57分49秒 CST.

☐ 忽略钩子 post-commit

- 配置构建

1. 配置兼容第三方测试报告的命令

System.setProperty("hudson.model.DirectoryBrowserSupport.CSP", "")

**Execute system Groovy script**

Groovy command

Groovy Script

System.setProperty("hudson.model.DirectoryBrowserSupport.CSP", "")

☐ Use Groovy Sandbox

Additional classpath

Add entry

高级...

2. 配置执行命令

咱们的脚本执行最终是以python run.py test去执行的，因此选择命令执行插件

windows下选择：Execute Windows batch command

mac或者linux下选择：Execute shell

Execute Windows batch command

命令

python run.py %env\_name%

这表示上边定义的环境参数, %env\_name%表示引用

mac下命令写法python3 run.py \${env\_name}

参阅 [可用环境变量列表](#)

高级...

- 配置构建后操作

- 测试报告收集

选择allure report插件

构建后操作

Allure Report

☐ Disabled

Results:

Path

report/shop

这里填的是自动化项目测试报告目录下的结果数据目录

新增

Paths to Allure results directories relative from workspace.  
E.g. **target/allure-results**.

Properties

新增

高级...

- 配置邮件发送

选择Editable Email Notification 插件

Editable Email Notification

☐ Disable Extended Email Publisher

Allows the user to disable the publisher, while maintaining the settings

Project From

Project Recipient List

2879897713@qq.com,2879897713@qq.com

这里填写收件人邮箱, 多个的话用英文逗号分割

Comma-separated list of email address that should receive notifications for this project.

Project Reply-To List

\$DEFAULT\_REPLYTO

Content Type ?

HTML (text/html) 选择邮件内容格式为HTML

Default Subject ?

\$DEFAULT\_SUBJECT

Default Content ?

<!DOCTYPE html>  
<html>  
<head>  
<meta charset="UTF-8">  
<title>\$PROJECT\_NAME-第\$BUILD\_NUMBER次构建日志</title>  
</head>  
  
<body leftmargin="8" marginwidth="0" topmargin="8" marginheight="4" offset="0">  
<div>  
<table width="95%" cellpadding="0" cellspacing="0" style="width: 100%; border-collapse: collapse;">  
<tr>  
<td style="width: 50%; vertical-align: top; padding: 5px;">  
<div>  
<table border="1" style="width: 100%; border-collapse: collapse;">  
<tr>  
<th style="text-align: left; padding: 5px;">构建时间  
</th>  
<tr>  
<td style="padding: 5px;">\${BUILD\_TIME}</td>  
</tr>  
</table>  
</div>  
</td>  
<td style="width: 50%; vertical-align: top; padding: 5px;">  
<div>  
<table border="1" style="width: 100%; border-collapse: collapse;">  
<tr>  
<th style="text-align: left; padding: 5px;">构建日志  
</th>  
<tr>  
<td style="padding: 5px;">\${BUILD\_LOG}</td>  
</tr>  
</table>  
</div>  
</td>  
</tr>  
</table>  
</div>  
</body>  
</html>

填写邮件内容模板，在群文件下载

Triggers ?

Always 这是邮件触发方式，点击高级设置才会出来，选择Always

Send To

Developers

Recipient List

新增

高级...

### 3. 任务执行

可以通过选择环境名称来决定执行哪一个环境下的测试

Dashboard > apiautotest >

返回面板

状态

修改记录

工作空间

Build with Parameters

配置

删除 Project

Email Template Testing

Allure Report

重命名

Project apiautotest

需要如下参数用于构建项目:

env\_name

test

开始构建

### 4. 查看邮件报告

## apiautotest - Build # 7 - Unstable! ☆

发件人: 2879897713 <2879897713@qq.com> 图

时间: 2021年11月3日 (星期三) 下午3:40

收件人: 沙陌 <2879897713@qq.com>

邮件可翻译为中文 立即翻译

### 构建信息

## BUILD UNSTABLE

Build URL <http://localhost:8080/job/apiautotest/7/>

Project: apiautotest

Date of build: Wed, 03 Nov 2021 15:40:28 +0800

Build duration: 21 秒

- 项目名称: apiautotest
- 详细测试日志: <http://localhost:8080/job/apiautotest/7/console>
- 详细测试报告: <http://localhost:8080/job/apiautotest/allure>
- 触发原因: Started by user admin
- 项目 Url: <http://localhost:8080/job/apiautotest/7/>

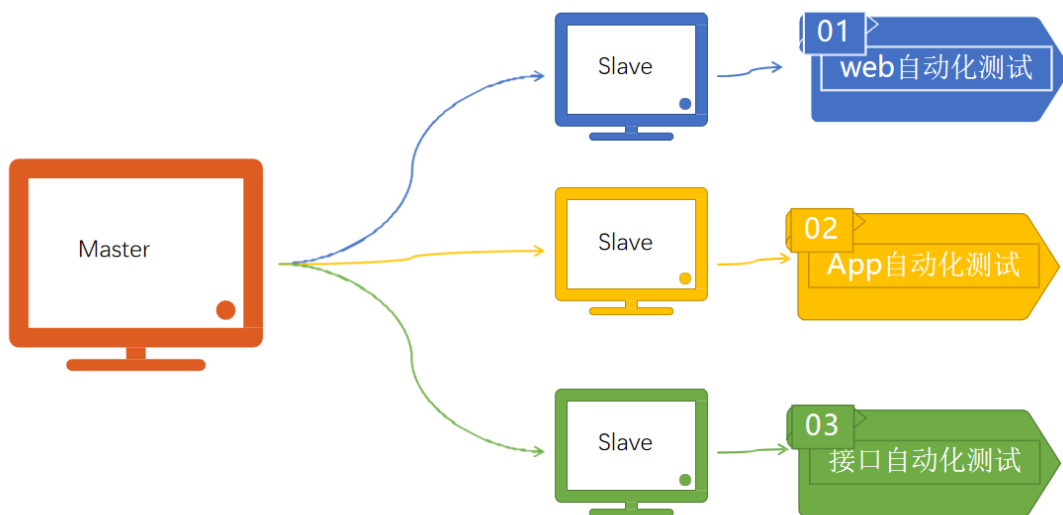
### CHANGES

Revision [efdff356ebf28a4adcba4588a04b66605](#)

edit

run.py

## 9.jenkins分布式执行



上图中master指的是jenkins所在服务器, 用来统筹管理各个任务及配置

slave指的是各个自动化任务执行的机器, 也叫作节点

master通过管理节点, 及任务中的节点配置将不同的任务分配到不同的设备上执行

### • 节点配置及启动

节点指的就是执行任务的机器

#### 1. 开启java web代理服务

打开jenkins的[Manager Jenkins]->[Configure Global Security] 页面, 进行如下 设置并保存

### 代理

TCP port for inbound agents

☐ 指定端口:  ☒ 随机选取 ☐ 禁用

代理协议

☒ Java Web Start Agent Protocol/4 (TLS 加密)

A TLS secured connection between the master and the agent performed by TLS upgrade of the socket.

#### 2. 节点配置

打开jenkins的[Manager Jenkins]->[Manage Nodes and Clouds] 页面

Dashboard > Nodes >

返回工作台

管理 Jenkins

新建节点

Configure Clouds

节点监控

构建队列

节点名称

自动化执行节点

Permanent Agent

添加一个普通、固定的节点到Jenkins。之所以叫做“固定”，是因为不受Jenkins管理的物理机、虚拟机等等。

确定

名称

自动化执行节点

描述

Number of executors

1

远程工作目录

D:\jenkinsslave 填写节点机器的某个目录，该目录是用来执行jenkins任务的工作目录

标签

auto\_node 节点标签名称

用法

Use this node as much as possible

### 3. 节点启动

第二步保存后如下，两种连接方式选择其一即可

Agent 自动化执行节点

节点连接Jenkins的方式如下:

Launch 在浏览器中启动节点

如果这种方式无法启动就用第二种方式

在命令行中启动节点

java -jar agent.jar -jnlpUrl http://localhost:8080/computer/%E8%87%AA%E5%8A%A8%E5%8C%96%E6%89%A7%E8%A1%E8%8A%E8%82%E7%82%06c7d97170456b4ae02d119434574870a5b5cc751bcf7279865bb574de18374f -workDir "D:\jenkinsslave"

Run from agent command line, with the secret stored in a file:

echo 06c7d97170456b4ae02d119434574870a5b5cc751bcf7279865bb574de18374f > secret-file  
java -jar agent.jar -jnlpUrl http://localhost:8080/computer/%E8%87%AA%E5%8A%A8%E5%8C%96%E6%89%A7%E8%A1%E8%8A%E8%82%E7%82%06c7d97170456b4ae02d119434574870a5b5cc751bcf7279865bb574de18374f -secret-file secret-file -workDir "D:\jenkinsslave"

#### =标签

auto\_node

#### ■ 使用launch启动

点击launch后会下载一个文件，将该文件放在节点机器上，点击该文件一步步操作即可，最终出现如下画面则启动成功，注意该弹框不要关闭



#### ■ 使用agent.jar启动



如果launch的不行的话可以采用这种方式

点击上图中框中的agent.jar进行下载，然后放在节点机器上，使用那一串命令进行启动

```
C:\Users\lixio>cd Downloads
C:\Users\lixio\Downloads>java -jar agent.jar -jnlpUrl http://localhost:8080/computer/%E8%97%A4%E5%8A%A8%E5%8C%96%E6%89%A7%E8%A1%8C%E8%8A%82%E7%82%B9/jenkins-agent.jnlp -secret 06c7d97170456b4ae02d119434574870a5b5cc751bcf7279865bb574de18374f
-workDir "D:\jenkinsslave"
十一月 03, 2021 4:34:31 下午 org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
信息: Using D:\jenkinsslave\remoting as a remoting work directory
十一月 03, 2021 4:34:31 下午 org.jenkinsci.remoting.engine.WorkDirManager setupLogging
```

最终结果出现connected，说明连接成功，该窗口也不能关闭，关闭后连接就中断了

查看节点列表，没有红x说明连接成功了

S	名称 ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Windows 10 (amd64)	已同步	36.37 GB	3.44 GB	18.60 GB	0ms
	自动化执行节点	Windows 10 (amd64)	已同步	36.37 GB	3.41 GB	18.60 GB	55ms
获取到的数据		3 分 27 秒	3 分 27 秒	3 分 27 秒	3 分 26 秒	3 分 27 秒	3 分 27 秒

- 节点任务配置

如果要在节点机器上执行自动化测试，那么自动化测试所需要的环境也是必须搭建的，比如python及各种第三方库、java、allure、git等等

找到之前配置的apiautotest任务，打开后点配置，增加节点信息

General

源码管理

构建触发器

构建环境

构建

构建后操作

[Plain text] 预览

添加参数

☐ Throttle builds

☐ 关闭构建

☐ 在必要的时候并发构建

☒ 限制项目的运行节点

标签表达式

auto\_node

这里写节点名称或者节点标签名称

No agent/cloud matches this label expression. Did you mean 'master' instead of 'au'?

高级

保存后执行

那么节点执行时到底是在哪里执行的呢？

master会直接将任务分配给节点，节点配置里有一个远程工作目录，那么会讲自动化的代码拉去到该目录，然后进行执行

Data (D:) > jenkinslave > workspace > apiautotest >			
名称	修改日期	类型	
.git	2021/11/3 17:07	文件	
.pytest_cache	2021/11/3 17:07	文件	
__pycache__	2021/11/3 17:07	文件	
allure-report	2021/11/3 17:07	文件	
api	2021/11/3 17:07	文件	
common	2021/11/3 17:07	文件	
config	2021/11/3 17:07	文件	
data	2021/11/3 17:07	文件	
logs	2021/11/3 17:07	文件	
report	2021/11/3 17:07	文件	
testcases	2021/11/3 17:07	文件	
__init__.py	2021/11/3 17:07	Pyth	
conftest.py	2021/11/3 17:07	Pyth	
pytest.ini	2021/11/3 17:07	配置i	
run.py	2021/11/3 17:07	Pyth	
setting.py	2021/11/3 17:07	Pyth	

## 10.钉钉及微信通知

- 钉钉通知

## 1. 安装钉钉通知插件

## 2. 登录钉钉，配置钉钉机器人

参考链接:

<https://developers.dingtalk.com/document/robots/custom-robot-access>

### 3. jenkins里配置钉钉

进入系统管理-->系统配置，找到钉钉区域，做如下配置

## 钉钉

通知时机

☒ 构建启动时

☒ 构建中断时

☒ 构建失败时

☒ 构建成功时

☒ 构建不稳定时

☒ 未构建时

机器人

id

dd001自定义的

名称

apitest和钉钉里设置时一样

webhook

钉钉机器人的hook地址  
https://oapi.dingtalk.com/robot/send?timestamp=1555757355&nonce=111c2

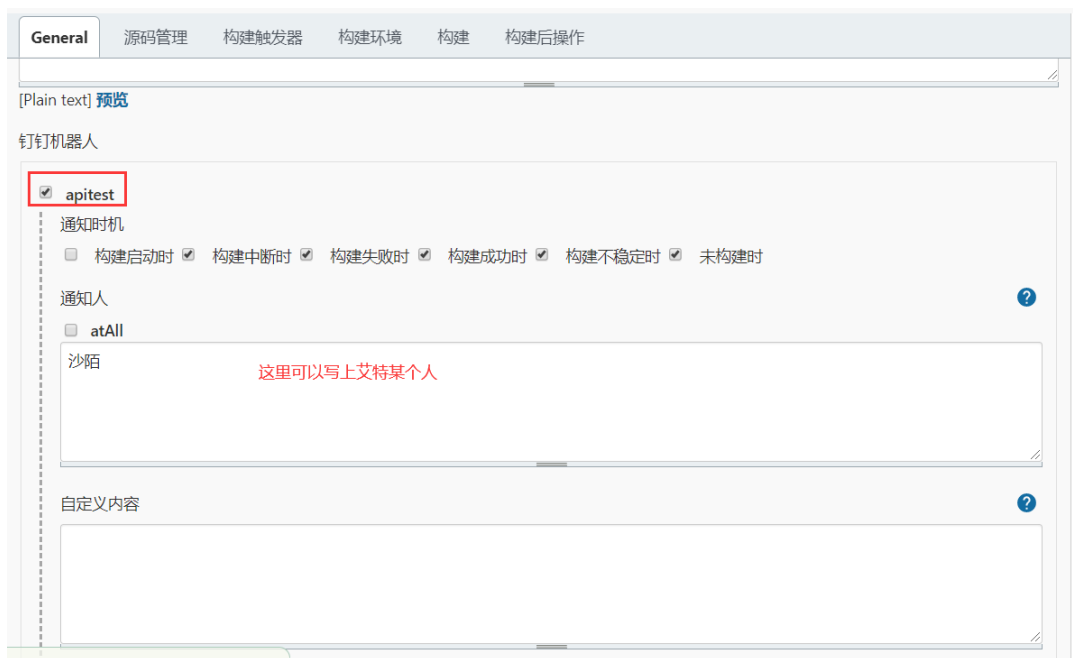
关键字

关键字和钉钉设置一样，任务发送信息里包含这个关键字才会被触发  
api接口

#### 4. jenkins任务里配置钉钉通知

找到之前配置的apiautest任务，进入配置页面，增加如下配置



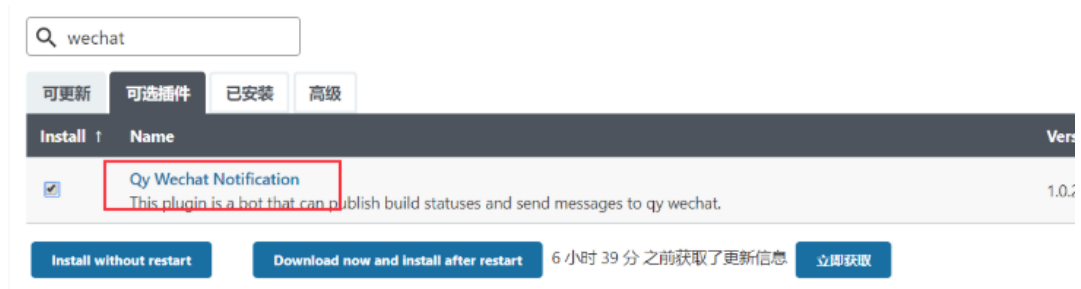


执行任务完成后，钉钉群收到消息



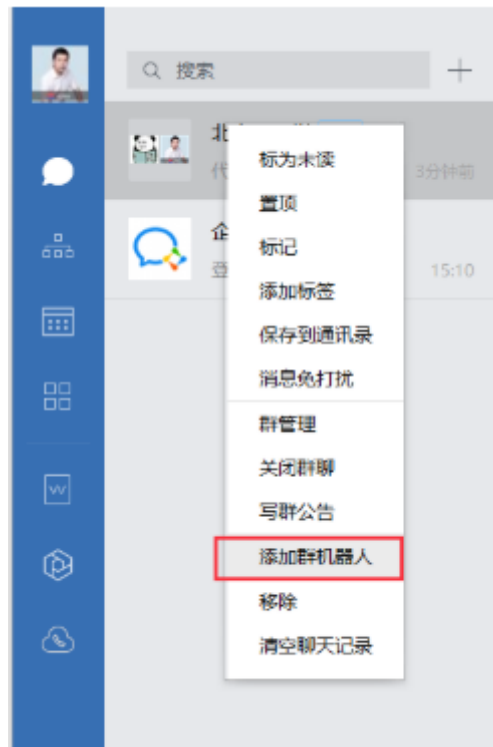
- 企业微信通知

- 安装微信通知插件



- 登录企业微信，添加微信机器人

- 群消息右键，点击添加机器人



- 点击添加一个机器人后，点击新创建一个机器人

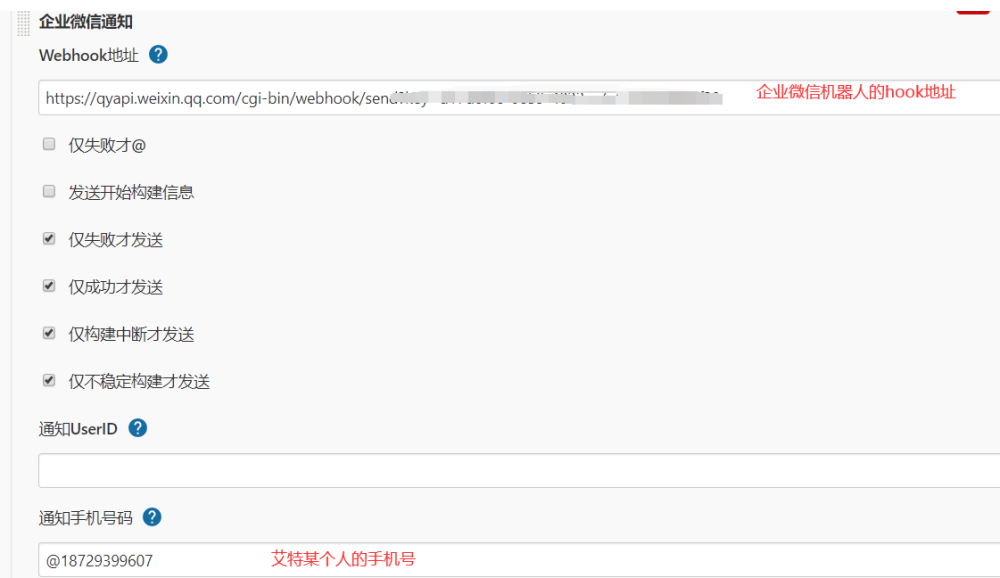


- 输入机器人名称，点击添加



### 3. jenkins任务里配置微信通知

找到之前的apiautotest任务，进入配置页面，找到最下面的构建后操作，添加企业微信通知插件



执行任务后，企业微信中收到消息



# 11.pipeline流水线

中文在线文档: <https://www.jenkins.io/zh/doc/book/pipeline/>

还有一个: <https://www.w3cschool.cn/jenkins/jenkins-qc8a28op.html>

## 1. pipeline基础

- 首先创建在jenkins上创建一个pipeline的流水线任务

新建Item--选择pipeline



- 基本pipeline脚本结构

```
pipeline {
    //agent 表示要执行的节点, any表示任意节点
    agent any
    //stages表示任务执行时的所有步骤集合
    stages {
        //stage就表示一个步骤, 括号里是步骤名称
        stage('拉取项目源码'){
            //每一个stage都可以定义自己执行的节点, 如果没定义, 则用最上方的
            agent {
                // label 后跟的是节点的标签名称
                label 'auto_node'
            }
            steps {
                echo '这是拉取代码这一步'
                echo "Running ${env.BUILD_ID} on ${env.JENKINS_URL}"
                echo "${env.JOB_NAME}"
                echo "${currentBuild.result} sdsdd"
                //sh 'pwd'
                //如果当前节点是windows, 我想执行windows下的命令
                bat 'dir'
            }
        }

        stage('静态代码扫描'){
            steps {
                echo '这是静态代码扫描'
            }
        }
    }
}
```

```

stage('单元测试'){
    steps {
        echo '执行jacoco单元测试'
    }
}

stage('打包依赖服务'){
    steps {
        echo '打包依赖服务'
    }
}

stage('打包当前服务'){
    steps {
        echo '打包当前服务'
    }
}

stage('部署环境'){
    steps {
        echo '部署环境'
    }
}

stage('接口自动化测试'){
    agent {
        label 'auto_node'
    }
    steps {
        echo '接口自动化测试'
    }
}

stage('ui自动化'){
    steps {
        echo 'ui自动化'
    }
}
}

```

## 2. 针对自动化编写pipeline脚本

对于pipeline脚本来说我们不用去记那么多的东西，可以打开自己的pipeline任务，点击流水线语法后，使用下面两个菜单帮我们生成部分脚本



- 对于一个项目来说可能具备多服务多环境的现象  
所以我们要创建环境节点参数，按照下图生成

返回

片段生成器

Declarative Directive Generator

Declarative Online Documentation

步骤参考

全局变量参考

在线文档

Examples Reference

IntelliJ IDEA GDSL

Overview

The **Directive Generator** allows you to generate the Pipeline code for a Declarative Pipeline directive, such as the **choice** directive. You can generate the Pipeline code for a new directive from the dropdown, and then choose the contents of the directive from the new form. Once you've filled out the form, click the **Generate Declarative Directive** button and the Pipeline code will appear in the box below. You can copy that code into your Pipeline script, or into a stage block for stage directives. See [the online syntax documentation](#) for more information.

Directives

Sample Directive

parameters: Parameters

See [the online documentation](#) for more information on the parameters directive.

Choice Parameter

名称

env\_name

参数名称

选项

test

prepro

pro

各个环境的名称

描述

[Plain text] 预览

新增

Generate Declarative Directive

parameters {  
 choice choices: ['test', 'prepro', 'pro'], name: 'env\_name'  
}

这是生成后的脚本，复制这里

在pipeline脚本的最上方增加如下：

```
parameters {  
  choice choices: ['test', 'prepro', 'pro'], name: 'env_name'  
}
```

配好后先执行一次任务

- 选择项目部署的节点

```
agent {
    label "auto_node"
}
```

- 拉取自动化测试项目代码

## 生成脚本

[返回](#)

片段生成器

Declarative Directive Generator

Declarative Online Documentation

步骤参考

全局变量参考

在线文档

Examples Reference

IntelliJ IDEA GDSL

### 概览

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement if you want, or you can omit optional and can be omitted in your script, leaving them at default values.)

### 步骤

示例步骤

git: Git

git

仓库 URL

https://gitee.com/sandroad/apiautotest.git 自动化脚本git仓库地址

分支

master 分支名称, 默认就是master

凭据

sandroad/\*\*\*\*\* (沙陌git用户)

添加

选择用户

☒ Include in polling?
 ☒ Include in changelog?

生成流水线脚本

git credentialsId: 'b4ef4f5a-295b-40a6-848f-aa3b23b21951', url: 'https://gitee.com/sandroad/apiautotest.git'

点击后生成下面的脚本

复制生成的脚本，放在拉代码的阶段的steps下

```
stage('拉取自动化测试脚本源码'){
    steps {
        echo '这是拉取代码这一步'
        git credentialsId: 'b4ef4f5a-295b-40a6-848f-aa3b23b21951', url: 'https://gitee.com/sandroad/apiautotest.git'
    }
}
```

- 执行测试脚本

```
stage('执行测试脚本'){
    steps {
        echo '执行测试脚本'
        bat 'python run.py %env_name%'
        //mac下
        //sh 'python run.py ${env_name}'
    }
}
```

- allure报告

## 生成脚本

片段生成器

Declarative Directive Generator

Declarative Online Documentation

步骤参考

全局变量参考

在线文档

Examples Reference

IntelliJ IDEA GDLS

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement optional and can be omitted in your script, leaving them at default values.)

步骤

示例步骤

allure: Allure Report

allure

☐ Disabled

Results:

Path

report/shop

这里是自动化测试项目中测试结果数据目录

新增

Paths to Allure results directories relative from workspace.  
E.g. **target/allure-results**.

Properties

新增

生成流水线脚本

点击后生成脚本

allure includeProperties: false, jdk: "", results: [[path: 'report/shop']]

复制生成的脚本，放在生成报告的阶段的steps下

```
stage('生成allure测试报告'){
    steps {
        echo '生成allure测试报告'
        allure includeProperties: false, jdk: '', results:
        [[path: 'report/shop']]
    }
}
```

## 邮件发送

### 生成脚本

邮件标题: \$PROJECT\_NAME - Build # \$BUILD\_NUMBER - \$BUILD\_STATUS!

返回

片段生成器

Declarative Directive Generator

Declarative Online Documentation

步骤参考

全局变量参考

在线文档

Examples Reference

IntelliJ IDEA GDLS

概览

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options and values.)

步骤

示例步骤

emailx: Extended Email

emailx

To

2879897713@qq.com

收件人, 多个用,分割

Recipient Providers

新增

Subject

\$PROJECT\_NAME - Build # \$BUILD\_NUMBER - \$BUILD\_STATUS!

邮件标题

Body

</td>

<td height="350px" style="overflow:hidden;display:block">\${JELLY\_SCRIPT,template="html"}<br/>

</tr>

</table>

</div>

</body>

邮件内容, 在群文件复制过来

生成流水线脚本

点击生成脚本

emailx body: ""<!DOCTYPE html>

<html>



复制生成的脚本，放在邮件发送的阶段的steps下

```
stage('邮件发送'){
    steps {
        echo '邮件发送'
        email body: '''<!DOCTYPE html>
            <html>
            <head>
            <meta charset="UTF-8">
            <title>${PROJECT_NAME}-第${BUILD_NUMBER}次构建日志</title>
            </head>

            <body leftmargin="8" marginwidth="0" topmargin="8"
marginheight="4"
            offset="0">
            <div>
            <table width="95%" cellpadding="0" cellspacing="0"
                style="font-size: 11pt; font-family: Tahoma,
Arial, Helvetica, sans-serif">

                <tr>
                    <th align="center" colspan="2"><br />
                        <h2>构建信息</h2>
                    </th>
                </tr>
                <tr>
                    <td>
                        <ul>
                            <li>项目名称 :   ${PROJECT_NAME}</li><br />
                            <li>详细测试日志 :   <a
href=${BUILD_URL}console target='_blank\''>${BUILD_URL}console</a></li>
                            <br />
                            <li>详细测试报告 :   <a
href=${JOB_URL}allure target='_blank\''>${JOB_URL}allure</a></li><br />
                            <li>触发原因:   ${CAUSE}</li><br />
                            <li>项目   url :   <a href='${BUILD_URL}'
target='_blank\''>${BUILD_URL}</a></li><br />
                        </ul>
                    </td>
                    <td height="350px"
style="overflow:hidden;display:block">${JELLY_SCRIPT,template="html"}
                    <br/>
                </tr>
            </table>
            </div>
            </body>
            </html>''' ,
        subject: '$PROJECT_NAME - Build # ${BUILD_NUMBER} -
${BUILD_STATUS!}',
        to: '2879897713@qq.com'
    }
}
```

- 钉钉通知

钉钉通知生成脚本上有些问题，所以手动编写一下，如下：

```
stage('钉钉通知'){
    steps {
        echo '钉钉通知'
        dingtalk(
            robot:'dd001',//robot指的是你在系统配置中配的钉钉机器人的id
            type:'MARKDOWN',
            atAll: false,
            title: "${JOB_NAME} 测试完成",
            text: ["#### '${JOB_NAME}'项目扫描部署  \n - 任务:
第'${BUILD_NUMBER}'次\n - 状态: '${currentBuild.result}' \n \n \n[查看控制台]('${BUILD_URL}')"]
        )
    }
}
```

- 企业微信通知

由于企业微信通知在jenkins上生成脚本有问题，所以需要使用python来调用企业微信机器人接口，代码如下：

在项目根目录下创建一个wx\_notice.py文件，写入如下代码并提交git

```
# !/usr/bin python3
# encoding: utf-8 -*-
# @author: 沙陌 微信: Matongxue_2
# @Time: 2021/11/4 13:25
# @Copyright: 北京码同学
import sys

from common.client import RequestsClient

class WxNotice(RequestsClient):

    def __init__(self,url,job_name,build_number,result,user,build_url):
        super().__init__()
        self.url = url
        self.method = 'post'
        self.json = {
            "msgtype": "markdown",
            "markdown": {
                "content": f"#### {job_name}测试完成  \n - 任务: 第
{build_number}次\n - 状态: {result} \n - 执行人: {user} \n \n[查看报告]
({build_url}/allure) "
            }
        }

class JenkinsStatus(RequestsClient):
    # http://localhost:8080/job/liushuixian/11/api/json
    def __init__(self,build_url,username,password):
        super().__init__()
        self.url = f'{build_url}/api/json'
        self.method = 'get'
        self.session.auth = (username,password)

if __name__ == '__main__':
```

```

# WxNotice().send()
args = sys.argv
build_url = args[1]
username = args[2]
password = args[3]
wx_url = args[4]
job_name = args[5]
build_number = args[6]
# url = 'http://localhost:8080/job/liushuixian/11/api/json'
# 调用任务执行数据
jenkins_result = JenkinsStatus(build_url, username, password)
jenkins_result.send()
# 获取任务执行人
user = jenkins_result.extract_resp('$.userName')
# 获取任务执行结果
result = jenkins_result.extract_resp('$.result')
print(user, result)

# wx_url = 'https://qyapi.weixin.qq.com/cgi-bin/webhook/send?
key=d17d0f68-0859-4022-a7a1-b5fa480f5f29'
WxNotice(wx_url, job_name, build_number, result, user, build_url).send()

```

在任务中需要增加更多的变量, username、password、wx\_url

String Parameter

名称 ?

username

默认值 ?

admin

描述 ?

在企业微信通知的阶段的steps下

```

stage('微信通知'){
    steps {
        echo '微信通知'
        bat 'python wx_notice.py %BUILD_URL% %username%
%password% %wx_url% %JOB_NAME% %BUILD_NUMBER%'
        //mac下
        //sh 'python wx_notice.py ${BUILD_URL} ${username}
${password} ${wx_url} ${JOB_NAME} ${BUILD_NUMBER}'
    }
}

```

- 执行任务

返回工作台

状态

变更历史

Build with Parameters

配置

删除 Pipeline

完整阶段视图

重命名

Allure Report

流水线语法

## Pipeline liushuixian

需要如下参数用于构建项目:

env\_name

test

username

admin

password

123456

wx\_url

https://qyapi.weixin.qq.com/cgi-bin/webhook/send?key=d17d0f68-0859-4022-a7a1-b5fa480f5f29

开始构建

阶段视图

Average stage times:

#14

Nov 04 15:07

No Changes

拉取自动化测试项目源码	执行测试脚本	生成allure测试报告	邮件发送	钉钉通知	微信通知
3s	22s	4s	1s	275ms	1s
2s	23s	4s	913ms	249ms	1s