

视频笔记：

● Jenkins 基础介绍

- 持续集成：快速发现错误、防止分支大幅偏离主干
- 持续集成(部署?)流程-CI，提交到代码仓库。持续集成服务器：检测代码变动、自动构建编译测试打包。
- 持续集成 CI、持续交付 CD、持续部署 CD，包括 Develop、Build、Test、Release、Deploy to production。CI 1~3，CD 1~4，CD 1~5。
- Jenkins 持续集成管理工具。跨平台。
https://en.wikipedia.org/wiki/Comparison_of_continuous_integration_software
- Jenkins 部署和安装方法。 待做。
- 首页：Configure System、Configure Global Security、Manage Plugins、Manage Nodes 等。
- **例子 1：新建 Jenkins Job。** New Item - Freestyle project - 输入 Project name (example_job1) 后可进入 configure 界面。General 设置、Build Triggers 设置(可 remotely 通过 token 在浏览器远程触发, stable、unstable、fails, Build periodically 和 Poll SCM(有变动后才运行))、Build Environment 设置等。进入 Project 后，手工触发编译：Build with Parameters 并点 Build 按钮。回到 Jenkins 首页可以看到 Project 列表。（可参考第 1 讲，具体的参数配置）
- **例子 2：** example_job2。Freestyle project、勾选 Build After other projects are built、Projects to watch 填 example_job1 并选 Trigger only if build is stable，Build 中 Execute shell 的 Command 填 echo "this is job2"。并 Apply。现象 1：可在 example_job1 中看到 Downsteam Projects 中有 example_job2；example_job2 中看到 Upsteam Projects 中有 example_job1。
- example_job2 的 Build 中 Execute shell 的 Command 中添加 echo "parameter is", \$tester_home，并在 example_job1 中选择 Build with Parameters, 参数仍填 hello 并点 Build 按钮。在 example_job2 中的 Console Ouput 内看不到 example_job1 传递的参数（Console Ouput 在 Build History 中点击#序号后打开）。
- 取消勾选 example_job2 的勾选 Build After other projects are built。在 example_job1 的 Post-build Actions 中的 Projects to build 填写 example_job2 并选 Trigger only if build is stable。并 Appley。同样会有现象 1。

● Jenkins 邮件及节点配置

- 邮件服务器的配置：Jenkins 首页选择 confiure，在 Extended E-mail Notification 中填写 SMTP server 并勾选 Use SMTP Authentication，填写用户名和密码后勾选 Use SSL，填写 SMTP port 和默认收件人（们）Default Recipients (\$DEFAULT_RECIPIENTS) 等信息。Default Triggers 中为发邮件的情况，如 Failure - Any。
- 在 E-mail Notification 中再填写一遍，Default Recipients 此处为 Reply-To Address，勾选 Test configuration by sending test e-mail 并填写 Test e-mail recipient。上方 Jenkins Locatioin 的 System Admin e-mail addres 内同时填写用户名。点击 Tset configuration 按钮检测，“默认收件人邮箱”即可以收到一封来自“用户名邮箱”的测试邮件。
- 在 example_job1 中的 Add post-build action 选择 Editable E-mail Notification，Project Recipient List 默认即为\$DEFAULT_RECIPIENTS，其他类似。Advance Settings 可以看到选择的 Failur - Any，也可以添加在当前作用域范围内的 Always 选项（在 example_job2 中的 Add post-build action 选择 Editable E-mail Notification 后看不到新增的选项）等。并 Apply。选择 Build with Parameters 填写参数并点 Build 按钮，可在 Conslose Ouput 可以看到不同步骤是否有 Checking if email needs to be generated。“默认收件人邮箱”即可以收到一封邮件。
- 可以观察到邮件标题和内容是按照 Jenkins 首页的 configure 内 Default Subject 和 Default Content 的格式发送的。
- Extended E-mail Notification 可能需要在 Plugin Manager 里面添加。还可以添加 Locale 插件选择其他语言。

● 分布式架构

- Master 和 Slave 架构，Console Ouput 中可以看到 Building on master in waorkspace...等语句。
- Jenkins 首页选择 Manage Nodes 后点击 New Node，输入名字如 mac，#of executors 代表最多同时运行的 build 的数目，可选 1。Remote root directory 任务下放目录，可先创建好后并填写，如/A/B。Labels 选定是在哪里运行，如 mac。Usage 选择 Only build...，从而确保环境匹配。Launch method 可选 viia Java Web Start，Node Properties 可添加环境变量等。点击名字后下载 slave.jar 并复制该行命令后在 shell 运行，即可以看到 Agent discovery successful。
- 在首页 configure Security 内 Agent protocols 一般会选上 Java Web Start Agent Protocol/1/2/4。
- 可在 master 机器 shell 中查看 slave.log 文件查看错误信息，如果查询不到说明网络连接或者端口有问题。
- 创建成功后可点进 example_job2 的 configure 可以看到 General 下多了一个选项 Restrict where this project can be run, 勾选后在 Label Expression 可以填写 Labels 的内容如 mac, 即会在其上运行（可删掉 Command 中的 echo "parameter is", \$tester_home）。此时在 example_job1 中点击 Build with Parameters 并 Build，可以观察到 Build History 运行，结束后点进 Downsteam Projects 的 example_job2 其内的 Build History 也开始运行(结束前会显示 Pending), 在 example_job2 的 Console Output 中可以看到 Building remotely on mac(mac) in workspace /A/B/workspace/example_job2 等语句。观察到 example_job1 的 console 显示其 Building on master in work space 等语句，目录不包含/A/B 路径。

● 对源代码仓库的集成

- **例子 3：把 svn 的代码能够在 master 运行并收到结果通知。**新建 newJob 名为 example_svn，Restrict where this project can be run 的 Label Expression 填写 mac，Souce Code Management 选择 Subversion，Repository URL 填写 SVN(?)地址,Credentials 点击 Add 并添加内容(SVN的用户名和密码)(第三讲 05:50),Repository depth 一般选 infinity,Build 中 Execute shell 的 Command 输入 pwd 回车 python ./C/D/test.py(什么目录?)，添加 post-build action 为 E-mail Notification。（已在本地的目录下有了代码）在 Console Output 中等待，并看到 cheking out...等内容和文件列表，同时运行了其中的.py 文件，并发送了邮件。
- 修改目录下的代码为错的(不是编译错误,是 self.assertEqual('a'.upper(),'B'))，并在 shell 中输入 svn commit -m "incorrect test.py" test.py 提交代码。example_svn 开始 build，查看 Console Output，会看到提示 FAILED (failures=1)，同时有邮件发送（Jenkins 邮件及节点配置勾选了 Always）。
- example_svn 的 configure 中 Build Triggers 勾选 Build periodically 并在 Schedule 内填写 H/2 * * * *（每隔两分钟），可以看到 Build History 中每两分钟会出现一次 Build。
- **例子 4：github 集成。**新建 newJob 名为 example_github，勾选 GitHub project，url 填写项目地址。勾选 Restrict where this project can be run 并在 Label Expression 填写 mac,Source Code Management 下勾选 Git，Repositories URL 填写以.git 结尾的仓库地址，Branches to build 为*/master，Build 选择 Execute shell 并在 Command 中填写 pwd 回车 echo "git remote url is" + \$GIT_URL 回车 echo "build num"+ &BUILD_NUMBER 回车 python test/test.py，可以通过点击 the list of...查看详细说明。并 Apply。点击 Build History 并进入 Console Output，可以看到相应的结果。

- 附：使用 github，复制.git 网址，git clone https://.../xx.git 下载到本地。可查看 git log，可通过 vim test.py 修改下载到的代码。输入 git add test.py，git commit test.py，输入描述信息。最终输入 git push origin master 即可上床提交代码。
- example_github 的 configure 中 Build Triggers 勾选 Poll SCM 并在 Shedule 内填写 TZ=Asia/Beijing 回车 H/Z * * * *从而修改时区并每两分钟 Build 一次，然而并没有每两分钟 Build 一次。

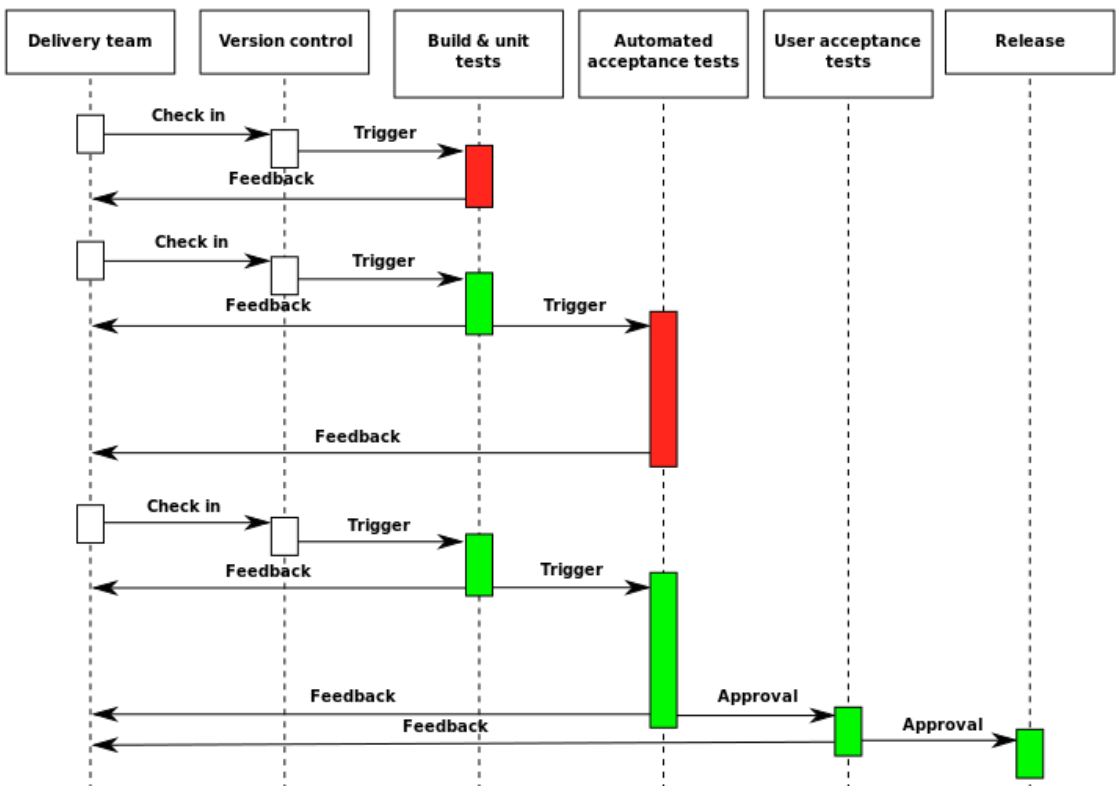
● 集成 Maven TestNG

- 例子：Maven 集成。略。
- pluginMagager 中安装 HTML Publisher, 修改 example_maven 的 Add post-build action 为 Publish HTML reports，在 HTML directory to archive 中填写路径，Index page[s]中填写路径下的 emailable-report.html。可以直接在 HTML Report 中查看相应的结果。
- 例子 5: appium 集成。新建 newJob 名为 example_appium，勾选 Restrict where this project can be run 并在 Label Expression 填写 mac，Build 选择 Execute shell 并在 Command 中填写 cd /Users/.../appium_sample/sampe-code/examples/python;python android_contacts.py。并 Apply。点击 Build Now。
- 集成成功和脚本成功。
- Command 中填写 cd /Users/.../appium_sample/sampe-code/examples/python;python android_contacts.py >result.txt 2>&1; ./result_analysis.sh。真正检测成功或失败。

```
python android_contacts.py > result.txt 2>&1
grep -i failure result.txt
if [ $? == 0 ]; then
    echo "failure is found"
    exit 1
else
    exit 0
fi
```

● DevOps 和持续交付、pipeline 机制

- 一系列 Jenkins 插件将整个持续交付流程用解释性代码 Jenkinsfile 来描述



- 例子 6: Pipeline。新建 newJob 名为 example_pepipeline, 下方勾选 Pipeline。Pipeline 的 Definition 选择 Pipeline script, Script 选择 try sample Pipeline，需提前在 configureTools 中添加 Name 为 M3 的 Maven，版本可为 3.5.2。并 Apply。点击 Build Now。可以看到 Stage View，其中可以查看每个阶段（Preparation、Build、Results）的 Logs 信息，在 Console Output 中可以看到三个阶段合起来的信息。
- Declarative Pipeline Syntax（推荐）、Scripted Pipeline Syntax

```
pipeline{
  agent any
  environment {
    paral = "abc"
  }
  stages{
    stage('Preparation'){
      //get code from GitHub
      steps{
        sh "echo git 'https://xx.git'"
      }
    }
    stage('Build'){
      steps{
        sh "echo build"
        sh "echo '${paral}'"
      }
    }
    stage('test'){
      steps{
        sh "echo test"
      }
    }
  }
}
```

```
pipeline{
  agent any
  environment {
    mvnHome = tool 'M3'
  }
  stages{
    stage('Preparation'){
      steps{
        git 'https://xx.git'
      }
    }
    stage('Build'){
      steps{
        script{
          if (isUnix()) {
            sh "'${mvnHome}/bin/mvn' -Dmaven.test.failure.ignore clean package"
          } else {
            bat("'%{mvnHome}\bin\mbn' -Dmaven.test.failure.ignore clean package/")
          }
        }
      }
    }
    stage('Results'){
      steps{
        junit '**/target/surefire-reports/TEST-*.xml'
        archive 'target/*.jar'
      }
    }
  }
}
```

- 将上图右代码拷贝到 example_pepipeline 的 Pipeline 下的 Sciprt 栏中，其中 Definition 为 Pipeline script。并 Apply。点击 Build Now。
- 例子 7: Selenium sample use declarative pipeline。新建 newJob 名为 example_selenium_pipeline，下方勾选 Pipeline。将上图右修改后的代码拷贝到 example_pepipeline 的 Pipeline 下的 Sciprt 栏中，其中 Definition 为 Pipeline script。其中代码修改 agent { label 'mac' }和 mvnHome = tool 'maven'两处，并修改 git 后地址。build 中的 script 内修改为 sh "cd ./SeleniumTest; '\${mvnHome}/bin/mvn' -Dmaven.test.failure.ignore clean package"。并 Apply。点击 Build Now。可以在 Stage View 中看到 Preparation 的 Logs 中看到 git 的进度；Results 的 Logs 中看到 Archive Junit-formatted test results --**/target/...../TEST-*.xml --(self time 1s)和 Archive artifacts --target/*.jar --(self time 88ms)等内容。点进 Build History 中的序号，可以看到 Test Result，依次点击 All Tests 下的 Package、Class、Test name 可看到 Passed 等信息。
- 例子 8: Pipeline 自定义模块之自定义测试报告。点击 example_selenium_pipeline 的 Configure，在上图右修改后的代码中 stage('Results')内 steps 段落下方添加 post { always { } }，点击 Pipeline Syntax，在 Sample Step 选择 publishHTML: Publish HTML reports，在 HTML directory to archive 中填写 Selenium Test/target/surefire-reports，Index page[s]中填写 emailable-report.html，Report title 可填写 HTML_Selenium Report，点击 Generate Pipeline Script，将生成的代码（第 7 讲 16:47）添加到 always { }中。并 Apply。点击 Build Now。左侧会出现 HTML-Selenium Report，点击可以看到报告。点进 Build History 中的序号，可以看到本次相对于例子 7 内多了 Test Result。
- 例子 9: Pipeline 自定义模块之发送邮件。点击 example_selenium_pipeline 的 Configure, 点击 Pipeline Syntax，在 Sample Step 选择 emailxxt: Extended Email，填写 To、Subject 和 Body 等信息，如 Body 填写\$DEFAULT

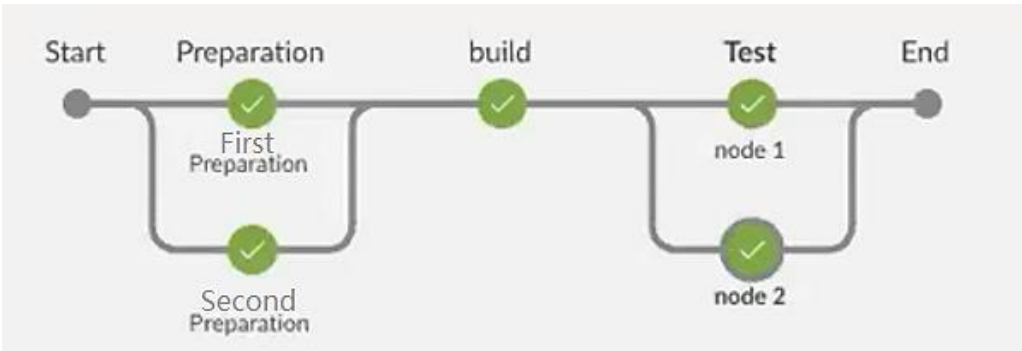
_CONTENT，点击 Generate Pipeline Script，将生成的代码（第 7 讲 22:32）添加到例子 8 中生成代码的下一行，（仍在 always { }内）。并 Apply。点击 Build Now。可以收到邮件。

例子 10: paraller samples。新建 newJob 名为 example_parallel_pipeline，下方勾选 Pipeline。将下图内代码拷贝到 Pipeline 下的 Sciprt 栏中，其中 Definition 为 Pipeline script。并 Apply。点击 Build Now。可以在 Stage View 看到各个 Stage 运行所需的时间。

```
pipeline {
  agent none
  stages {
    stage('Preparation') {
      parallel {
        stage('First Preparation') {
          agent {
            node {
              label 'mac'
            }
          }
          steps {
            git 'https://github.com/xxx/Code.git'
          }
        }
        stage('Second Preparation') {
          agent any
          steps {
            sh 'pwd'
            sh 'echo preparation2'
          }
        }
      }
    }
  }
}
```


```
stage('build') {
  agent any
  steps {
    sh 'pwd'
    echo 'abc'
  }
}
stage('Test') {
  parallel {
    stage('node 1') {
      agent any
      steps {
        sh 'pwd'
        sh 'sleep 20s'
        sh 'echo hstream1'
      }
    }
    stage('node 2') {
      agent {
        label 'mac'
      }
      steps {
        sh 'pwd'
        sh 'sleep 20s'
        sh 'echo hello2'
        sh 'python ./test/test.py'
      }
    }
  }
}
```

- 代码：整个的 agent 是 none，stage Preparation 内 parallel 中的 First Preparation 和 Second Preparation 是并行。stage Preparation 和 stage build 是串行，stage Test 内 parallel 中的 node 1 和 node 2 是并行。
- 点击左侧 Open Blue Ocean，并点进 MESSAGE 为 Stared by user XXX，看到具体的并行串行的图示。点击绿色的勾可以看到具体的指令和 agent 等内容。



例子 11: SCM Jenkinsfile example。选择 example_parallel_pipeline，其中 Pipeline 的 Definition 为 Pipeline script form SCM，代表着 Jenkinsfile 可以放在源代码管理仓库，跟开发者代码放在一起。SCM 可选择 Git，Repositor URL 填写.git 结尾的路径地址，Branches to build 一般为*/master。并 Apply。点击 Build Now。可以看到在 Open Blue Ocean 界面中每个 Step 下多了一条 General SCM。

Blue Ocean

- Plugin Manager 中安装 Blue Ocean 和 Blue Ocean Pipeline Editor。可以控制已有 Pipeline，也可以创建新的 Pipeline。
- 在 Jenkins 中点击左侧 Open Blue Ocean, 进入 Pipelines 界面点击右侧 New Pepeline, Where do you store your code 可选的 GitHub，并选择账户后选择 repositories 并点击 Creat Pipeline，输入 Naming conflict 后创建 Pipeline。
- 在某一 Pipeline 界面中，点击右上侧铅笔图标，即可以在图形化界面中添加或删除节点，在添加时点击已有节点下方的+, 右侧输入节点名字。并点击 Add step 按钮，可选择 Shell Script 选项，可输入 pwd 并点击 save。输入 Description 后并选择 Commit to，点击 Save & run。但是添加时可能没有错误检测，如没添加 agent 会导致节点运行错误。

答疑课

- Jenkins 运行失败的原因：Jenkins 的 master 环境不能满足所有测试环境，集成到 Jenkins 等 SSH 的环境要一致。
- 排查：确保脱离编译器时在 Shell 能运行、Master 连接 Node 后的系统环境和本地 Node 运行环境是否一样，通过 export 查看。

总结

- 持续集成测试流程和 Jenkins 介绍
- Jenkins 在 Ubuntu/Centos/Docker 系统上的搭建部署
- 使用 Jenkins 来创建任务和配置
- 创建和部署 slaves 和 nodes
- SVN 的集成和任务管理配置
- Git/Github 的集成和任务管理配置
- Jenkins 实例：Maven & Jankins with Selenium Jenkins 2.0
- DevOps 和持续交付
- Jenkins 2.0 Pipeline 机制
- Pipeline 定义和使用
- Blue Ocean 的使用