Jenkins Pipeline语法详解

蒋刚毅（Cay）

预习资料

- Pipeline使用之语法详解 https://jianggy.gitbooks.io/jenkins2/content/chapter03.html
- Pipeline使用之项目样例 https://jianggy.gitbooks.io/jenkins2/content/chapter06.html

Docker方式部署Jenkins

1.下载Jenkins的docker镜像：
docker pull hub.c.163.com/library/jenkins:latest
2.启动Jenkins镜像：
docker run -itd -u root --name jenkins -p 8080:8080 -p 50000:50000 –v /var/jenkins_home:/var/jenkins_home hub.c.163.com/library/jenkins
3.进入Jenkins容器
docker exec -it jenkins bash
4.访问Jenkins服务：
http://IP:8080/jenkins

Pipeline DSL Syntax

- **Declarative Pipeline**

对用户来说，语法更严格，有固定的组织结构，更容易生成代码段，同时可支持BlueOcen图形化脚本操作，使其成为用户更理想的选择。

- **Scripted Pipeline**

更加灵活，因为Groovy本身只能对结构和语法进行限制，对于更复杂的Pipeline来说，用户可以根据自己的业务进行灵活的实现和扩展。

Jenkinsfile (Declarative Pipeline)

```
pipeline {
    agent any 1

    stages {
        stage('Build') { 2
            steps { 3
                sh 'make' 4
            }
        }
        stage('Test'){
            steps {
                sh 'make check'
                junit 'reports/**/*.xml' 5
            }
        }
        stage('Deploy') {
            steps {
                sh 'make publish'
            }
        }
    }
}
```

Jenkinsfile (Scripted Pipeline)

```
node {
    stage('Example') {
        if (env.BRANCH_NAME == 'master') {
            echo 'I only execute on the master branch'
        } else {
            echo 'I execute elsewhere'
        }
    }
}
```

Declarative Pipeline

- Declarative Pipeline遵循与Groovy相同的语法规则，但有以下几点例外：
  - Pipeline的顶层必须是块，具体来说就是：pipeline { }。
  - 不用分号作为语句分隔符，每个声明必须独立一行。
  - 块里只能包含Sections（章节）、Directives（指令）、Steps（步骤）或赋值语句。
  - 属性引用以无参方法的方式调用。例如，输入被视为input（）

```
pipeline {
/* insert Declarative Pipeline here */
    }
```

Sections（章节）

Sections通常包含一个或多个Directives或Steps。

- **agent**
  - agent指定整个Pipeline或特定stage在Jenkins环境中执行的位置。在pipeline代码块的顶层agent必须进行定义，但在stage级使用是可选的。
- **post**
  - 定义Pipeline或stage运行结束后的操作。post支持以下类型的代码块：always，changed，failure，success，unstable和aborted。
- **stages**
  - 包含一个或多个stage的序列，Pipeline的大部分工作在此执行。建议stages至少包含至少一个stage指令，用于连接各个交付过程，如构建，测试和部署等。
- **steps**
  - steps包含一个或多个在stage块中执行的step序列。

Directives （指令）

- **environment**
  - environment指令指定一系列键值对，这些键值对将被定义为所有step或stage中step的环境变量，具体取决于environment指令在Pipeline中的位置。
- **options**
  - options指令允许在Pipeline内配置Pipeline专用选项。
- **parameters**
  - parameters指令提供用户在触发Pipeline时的参数列表。这些参数值通过params对象可用于Pipeline步骤
- **triggers**
  - triggers指令定义了Pipeline自动化触发的方式。
- **stage**
  - stage指令包含在stages中，包含step、agent（可选）或其他特定包含于stage中的指令。实际上，Pipeline完成的所有实际工作都包含在一个或多个stage指令中。
- **tools**
  - 通过tools可自动安装工具，并放置环境变量到PATH。如果agent none，这将被忽略。
- **when**
  - when指令允许Pipeline根据给定的条件确定是否执行该阶段。

Parallel(并行)

- **Parallel**
  - Declarative Pipeline的stages中可能包含多个嵌套的stage, 对相互不存在依赖的stage可以通过并行的方式执行，以提升pipeline的运行效率。
  - 另外，通过在某个stage中设置“failFast true”，可实现当这个stage运行失败的时候，强迫所有parallel stages中止运行。

Steps（步骤）

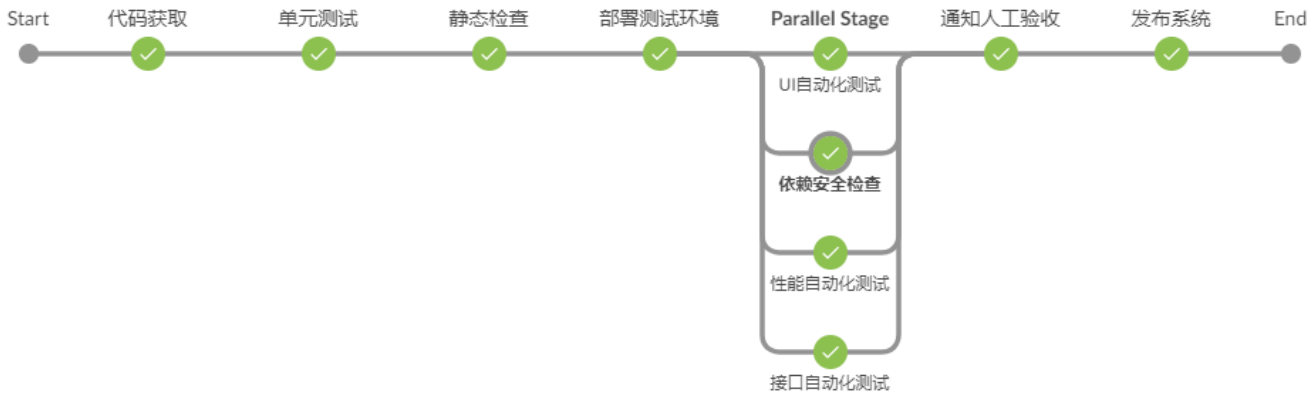
- **基本概念**
  - Pipeline最基本的部分是“step”，step告诉Jenkins要做什么，并且作为 Pipeline语法的基本构建块。
- **Pipeline Stepsreference**
  - Declarative Pipeline可使用Pipeline Steps手册中的所有可用步骤：https://jenkins.io/doc/pipeline/steps/
- **script**
  - script步骤中可以引用script Pipeline语句，并在Declarative Pipeline中执行。对于大多数用例，script在Declarative Pipeline中的步骤不是必须的，但它可以提供一个有用的加强。

Scripted Pipeline

- Groovy脚本不一定适合所有使用者，因此Jenkins创建了Declarative Pipeline，为编写Jenkins Pipeline提供了一种更简单、更有意义的语法。但是不可否认，由于脚本化的pipeline是基于groovy的一种DSL语言，所以与Declarative pipeline相比为Jenkins用户提供了更巨大的灵活性和可扩展性。

```
//Jenkinsfile (Scripted Pipeline)
node {
    stage('Example') {
        if (env.BRANCH_NAME == 'master') {
            echo 'I only execute on the master branch'
        } else {
            echo 'I execute elsewhere'
        }
    }
}
```

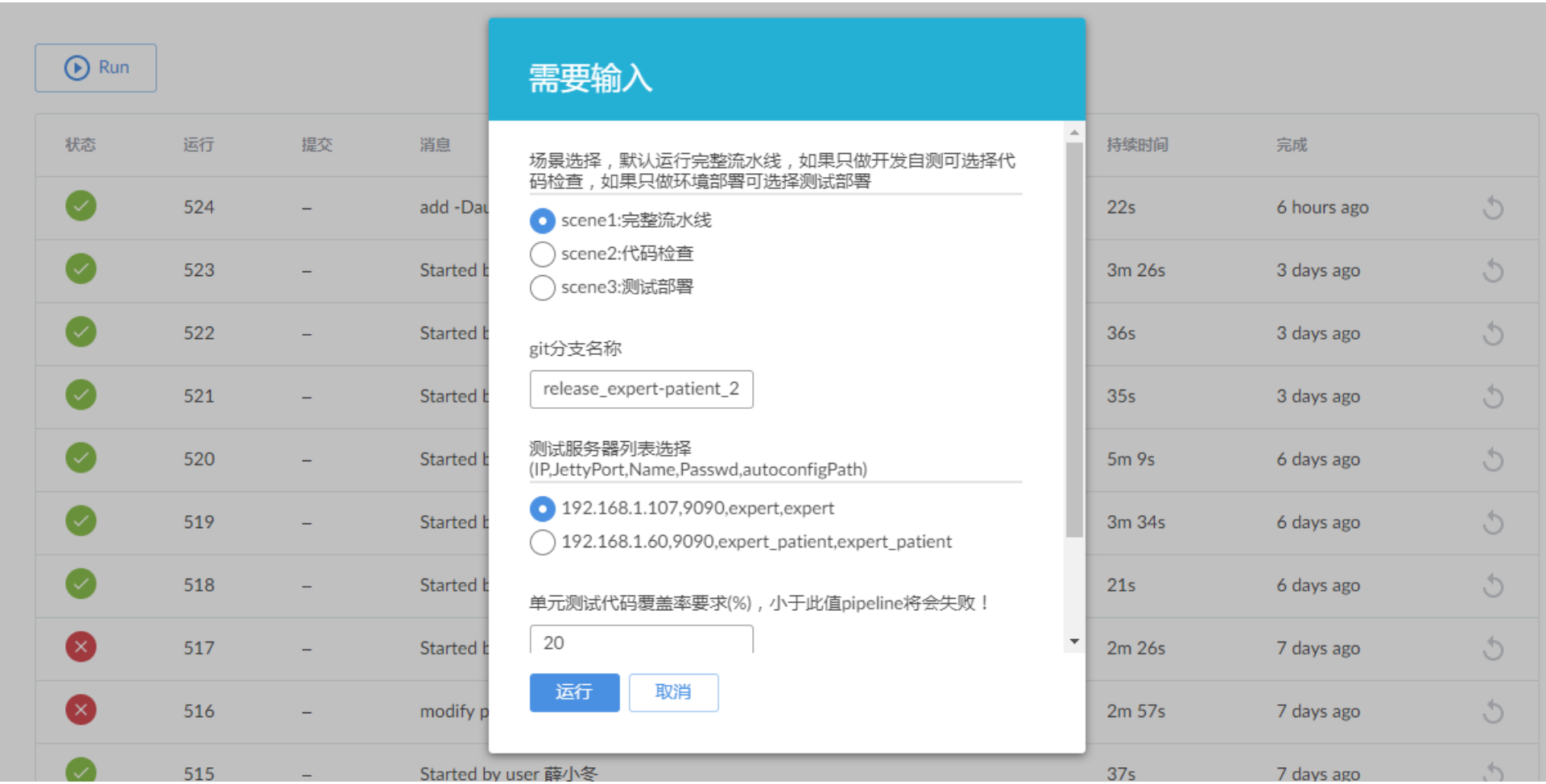
持续交付流水线样例



Steps 依赖安全检查

✓	maven3 — Use a tool from a predefined Tool Installation	<1s
✓	Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step.	<1s
✓	jdk8 — Use a tool from a predefined Tool Installation	<1s
✓	Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step.	<1s
✓	Invoke OWASP Dependency-Check analysis	39s

持续交付流水线样例


Run
状态
运行
提交
消息
持续时间
完成
524 – add -Dau 22s 6 hours ago
523 – Started b 3m 26s 3 days ago
522 – Started b 36s 3 days ago
521 – Started b 35s 3 days ago
520 – Started b 5m 9s 6 days ago
519 – Started b 3m 34s 6 days ago
518 – Started b 21s 6 days ago
517 – Started b 2m 26s 7 days ago
516 – modify p 2m 57s 7 days ago
515 – Started by user 薛小冬 37s 7 days ago
需要输入
场景选择，默认运行完整流水线，如果只做开发自测可选择代码检查，如果只做环境部署可选择测试部署
scene1:完整流水线
scene2:代码检查
scene3:测试部署
git分支名称
release_expert-patient_2
测试服务器列表选择
(IP,JettyPort,Name,Passwd,autoconfigPath)
192.168.1.107,9090,expert,expert
192.168.1.60,9090,expert_patient,expert_patient
单元测试代码覆盖率要求(%)，小于此值pipeline将会失败！
20
运行
取消

课后习题

- 待补充