
DOIS | 2018 · 深圳站
DevOps 落地，从这里开始
DevOps 国际峰会
暨 DevOps 金融峰会
指导单位：云计算开源产业联盟
主办单位：DevOps时代
高效运维社区
GreatOPS Community
时间：2018年11月2日-3日
地址：深圳市南山区圣淘沙大酒店（翡翠店）

DOIS

Jenkins工作坊
手把手带你玩转pipeline

李华强　Jenkins老兵

沟通群二维码图片

DOIS

自我介绍

DOIS

- 软件配置管理（SCM）领域的一个老兵，先后就职于北电网络，爱立信，飞维美地，乐视，乐融等多家企业从事SCM，DevOps相关的工作。
- Jenkins的忠实粉丝以及最佳实践的推广者，Jenkins官方Certified Jenkins Engineer (CJE)和Certified CloudBees Jenkins Platform Engineer (CCJPE)认证者。

DOIS

目录

Jenkins Pipeline总体介绍

DOIS

- Pipeline 是Jenkins2.0最核心的特性，帮助Jenkins实现CI到CD转变 https://jenkins.io/2.0/
- Pipeline，简单来说，就是一套运行于Jenkins上的工作流框架，将原本独立运行于单个或者多个节点的任务连接起来，实现单个任务难以完成的复杂发布流程。
- 工程实践：Pipeline is code，CD is code，...， everything is code

什么是Jenkins Pipeline

DOIS

- Jenkins Pipeline是一组插件，让Jenkins可以实现持续交付流水线的落地和实施。
- 持续交付流水线(CD Pipeline)是将软件从版本控制阶段到交付给用户或客户的完整过程的自动化表现。软件的每一次更改（在源代码管理中提交）都要经过一个复杂的过程才能被发布。
- Pipeline提供了一组可扩展的工具，通过Pipeline Domain Specific Language (DSL) syntax可以达到CD Pipeline即代码的目的 (CD Pipelines as code)。
- 通常，此“Pipeline即代码”将被写入Jenkinsfile 存储在项目的代码库

为什么要用Pipeline

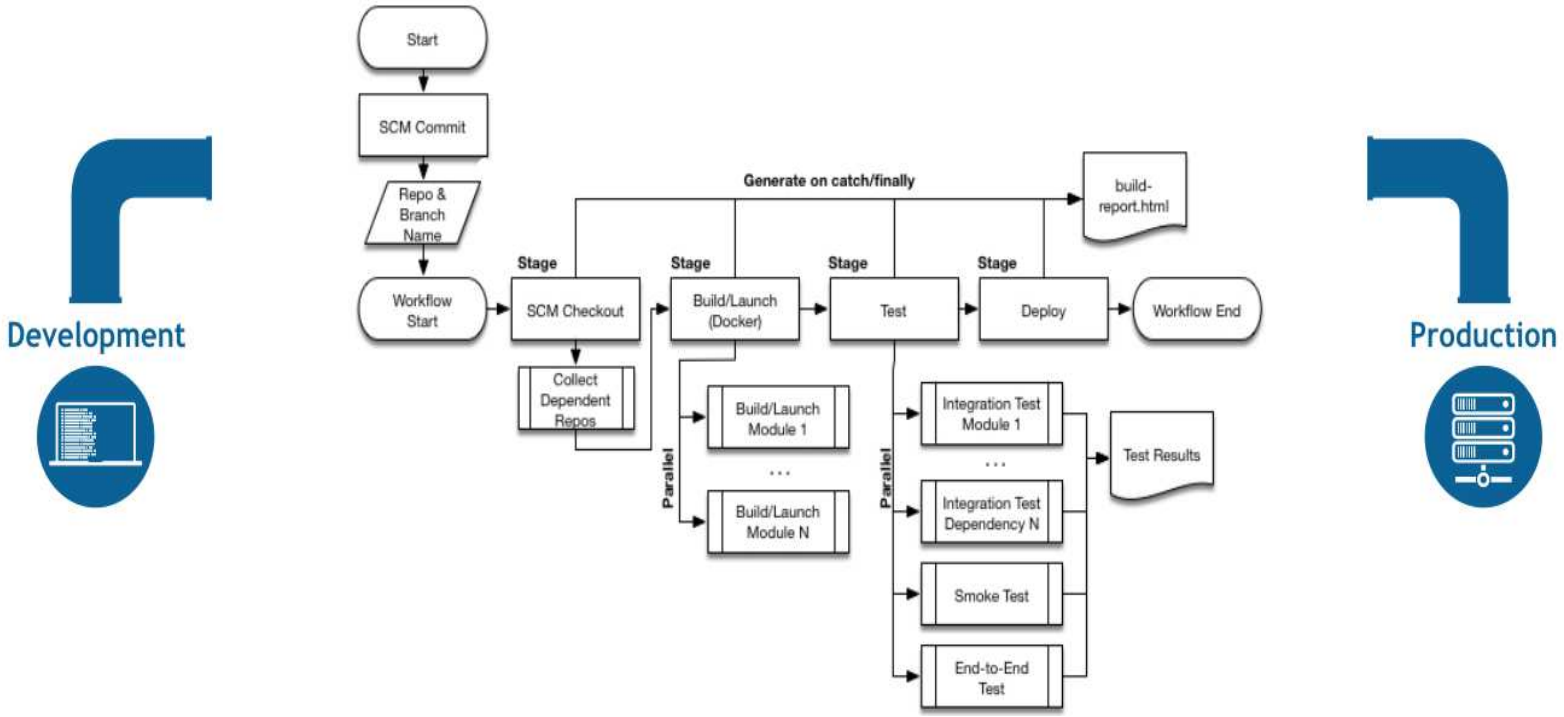
DOIS

用户可以利用Pipeline 的核心功能，构建一系列的复杂功能：

- **代码**：Pipeline以代码的形式实现，通常被检入源代码控制，使团队能够编辑，审查和迭代其CD流程。
- **可持续性**：Jenkins重启或者中断后都不会影响Pipeline Job。
- **停顿**：Pipeline可以选择停止并等待人工输入或批准，然后再继续Pipeline运行。
- **多功能**：Pipeline支持现实世界的复杂CD要求，包括fork/join子进程，循环和并行执行工作的能力。
- **可扩展**：Pipeline插件支持其DSL的自定义扩展以及与其他插件集成的多个选项。

Pipeline案例：持续交付流水线

DOIS



Jenkins Pipeline入门

DOIS

- Pipeline脚本是由Groovy语言实现
  - 无需专门学习Groovy
- Pipeline支持两种语法：
  - Declarative 声明式（在Pipeline 2.5中引入）
  - Scripted Pipeline 脚本式
- 如何创建基本的Pipeline：
  - 直接在Jenkins Web UI 网页界面中输入脚本。
  - 通过创建一个Jenkinsfile可以检入项目的代码库。
- 最佳实践
  - 通常推荐在 Jenkins中直接从源代码控制(SCM)代码库中载入Jenkinsfile

Jenkins Pipeline核心概念

DOIS

- Stage
  - 阶段，一个Pipeline可以划分为若干个Stage，每个Stage代表一组操作，例如："Build", "Test", "Deploy" 。
  - 注意，Stage是一个逻辑分组的概念，可以跨多个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，例如： sh 'make'

Declarative Pipeline -1

DOIS

声明式Pipeline的基本语法和表达式遵循与Groovy语法相同的规则 ，但有以下例外：

- 声明式Pipeline必须包含在固定格式**pipeline{}**块内；
- 每个声明语句必须独立一行，行尾无需使用分号；
- 块(Blocks **{}**)只能包含章节(Sections)， 指令(Directives)，步骤(Steps)或赋值语句

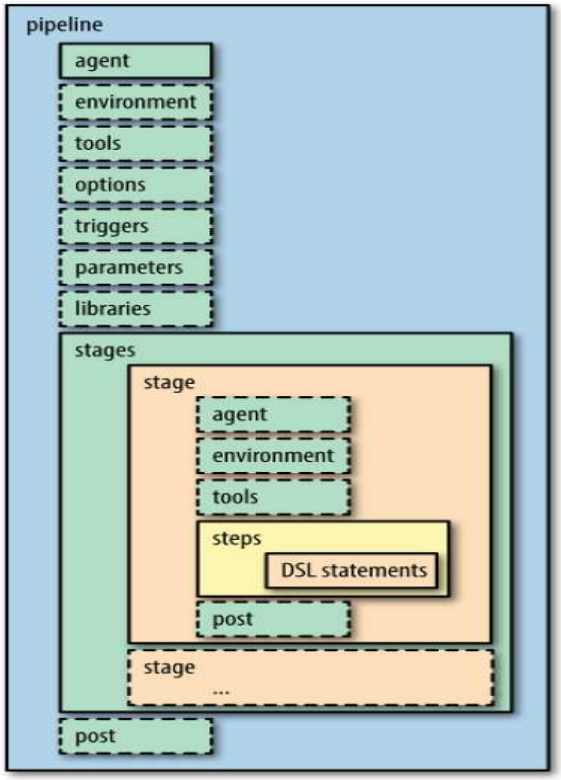
Declarative Pipeline -2

DOIS

- 块(Blocks **{}**)
  - 由大括号括起来的语句，如
  - Pipeline {}, Section {}, parameters {}, script {}
- 章节(Sections)
  - 通常包含一个或多个指令或步骤
  - agent, post, stages, steps
- 指令(Directives)
  - environment, options, parameters, triggers, stage, tools, when
- 步骤(Steps)
  - Pipeline Steps reference
  - 执行脚本式pipeline：使用 script{}

Declarative Pipeline -3

DOIS



DOIS

目录

工作坊环境部署

DOIS

- 云主机信息（统一分配IP）

容器部署（预置插件，配置，demo jobs等）

- /bin/systemctl restart docker.service
- docker pull huaqiangli/jenkins:dois2018-sz-3
- docker run -d -p 8080:8080 -p 50000:50000 huaqiangli/jenkins:dois2018-sz-3

访问Jenkins

- http://<your ip>:8080/
- 登录: admin/jenkins2018
- 修改全局配置的 Jenkins URL 为 http://<your ip>:8080/
- 讲师 Jenkins： http://106.75.175.85:8080/

实战-1

DOIS

- 如何安装相关插件支持pipeline类型project演示
- 如何创建一个基本的pipeline project演示
- 两种代码集成方式（pipeline script，pipeline script from scm）介绍
- 两种类型的pipeline 语法（脚本式，声明式）介绍
- 脚本式 pipeline示例
- 声明式 pipleline示例
- 查看stage view 演示
- 如何使用Snippet Generator等帮助工具获得pipeline 步骤指令等语法演示
- 如何下载代码（checkout，git，svn）
- 如何编译，归档，发送邮件（sh，archive,archiveArtifacts, mail,emailext）
- 如何使用replay功能演示

如何安装相关插件支持pipeline类型project演示

- 查看系统中安装的插件
- 注意主要的插件组
  - Pipeline
  - Blue Ocean

脚本式 pipeline示例

DOIS

- 演示如何创建一个基本的pipeline project
- 创建一个pipeline 项目，pipeline-demo-1，使用“Github+Maven” sample
- 查看 pipeline项目中的各个部分
- 注意两种代码集成方式（pipeline script，pipeline script from scm），支持两种类型的pipeline 语法（脚本式，声明式）
- 运行pipeline，查看log，stageview等

声明式 pipleline示例

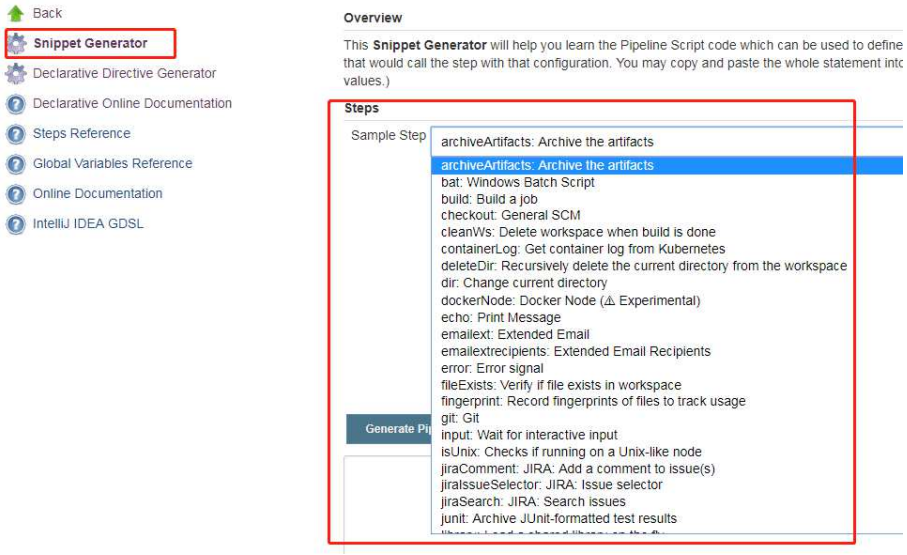
DOIS

- 创建一个pipeline 项目， pipeline-demo-2
- 可以拷贝1.2-pipeline-dec-sample
- 实现pipeline-demo-1对应的功能
- 注意点：
  - tools 指令会作为一个"Declarative: Tool Install" stage出现在stage view中
  - 日志中“The archive step is deprecated, please use archiveArtifacts instead.“
- 更新pipeline，使用archiveArtifacts ，可以参考1.2-pipeline-dec-sample-2

如何使用Snippet Generator

DOIS

- 演示如何使用Snippet Generator等帮助工具获得pipeline 步骤指令等语法
- 演示如何下载代码（checkout，git，svn）



如何下载代码示例

DOIS

- 演示如何下载代码（checkout，git）
- 在pipeline-demo-2的基础上使用checkout以及git语法
- 可以参考1.3-pipeline-dec-checkout 以及1.3-pipeline-dec-git

如何使用replay功能

DOIS

- 在pipeline-demo-2的基础上，replay修改一些代码后运行
- 注意点
  - 失败的build不能进行replay
  - Replay会重新生成一个build number进行运行
  - Replay不会修改pipeline项目配置

DOIS

目录

实战-2

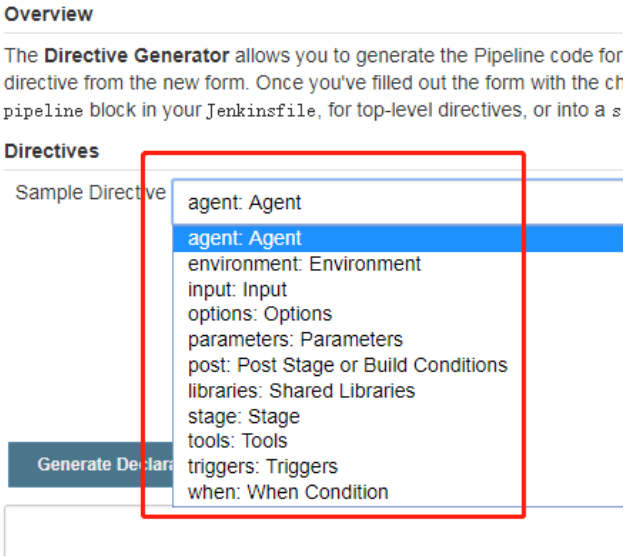
DOIS

- 声明式pipeline 主要指令介绍（agent environment options parameters triggers stage tools input when post）
- 如何使用agent示例
- 如何使用environment示例
- 如何使用parameters示例
- 如何使用triggers示例
- 如何使用options示例
- 如何使用并行stages示例
- 如何使用when跳过某个stage示例
- 如何使用input进行人工确认示例
- 如何使用post模拟post build actions示例
- 如何使用pipeline in SCM 示例
- 如何使用多分支流水线（multibranch pipeline）示例

声明式pipeline 主要指令介绍

DOIS

- agent
- environment
- options
- parameters
- triggers
- stage
- tools
- input
- when
- post



如何使用agent示例

DOIS

- 在pipeline-demo-2的基础上，修改agent部分使用label选择一个节点运行

```
agent {
  label 'slave-1'
}
```

- 注意点： slave-1 是添加在master节点上的label

如何使用environment示例

DOIS

- 在pipeline-demo-2的基础上，添加一个pipeline级别的environment以及一个stage级别的environment，可以参考2.2-pipeline-environment，注意不同级别environment定义变量的作用域不同
- environment中定义环境变量时可以使用 credentials函数，可以参考2.2-pipeline-environment-2，注意环境变量中插入一组环境变量

如何使用parameters示例

DOIS

- 创建一个新的pipeline项目pipeline-demo-3，可以拷贝2.3-pipeline-dec-param-1
- 注意点：
  - 第一次构建的时候UI参数不显示，建议使用默认值
  - Jenkins 2.112 之前版本，Snippet Generator生成不正确的Choice parameter代码。choices: ['choice1', 'choice2', 'choice3'] 需要改成 choices: 'choice1\nchoice2\nchoice3 ‘
  - 第一次运行pipeline后，UI中参数定义被实例化
- 结合UI定义的参数，参考2.3-pipeline-dec-param-2
- 脚本式pipeline如何定义参数？参考
  - 2.3-pipeline-scripted-param-1
  - 2.3-pipeline-scripted-param-2

如何使用triggers示例

DOIS

- 在pipeline-demo-3的基础上添加trigger定义

  triggers { cron 'H/15 * * * *' }

- 可以参考2.4-pipeline-dec-trigger
- 注意点：运行后UI中Build Triggers被实例化
- 脚本式pipeline中如何定义trigger呢？可以参考2.4-pipeline-scripted-trigger

如何使用options示例

DOIS

- 在pipeline-demo-3的基础上添加options定义

```
options {
    buildDiscarder logRotator(artifactDaysToKeepStr: '', artifactNumToKeepStr: '', daysToKeepStr: '', numToKeepStr: '7')
    disableConcurrentBuilds()
    quietPeriod 5
    timeout(30)
    timestamps()
 }
```

注意点：

- Declarative Directive Generator生成的timestamps不正确，应该是timestamps()
- 注意pipeline运行后对应配置的实例化
- 脚本式pipeline中如何实现类似功能呢？

如何使用并行stages示例

DOIS

- 参考 2.6-pipeline-dec-parallel以及2.6-pipeline-dec-parallel-2
- 拷贝2.6-pipeline-dec-parallel-2创建pipeline-demo-4
- 注意点
  - Pipeline脚本讲解，注意stash、unstash以及parallel的用法
  - Open Blue Ocean 查看

如何使用when跳过某个stage示例

DOIS

- 在pipeline-demo-4的基础上，定义一个布尔参数用来控制一个stage的执行

```
parameters {
    booleanParam defaultValue: true, description: 'a boolean parameter', name: 'Run_unit_test'
}
when { expression { return params.Run_unit_test }}
```

- 运行并查看日志，stage view，blue ocean等
- 可以参考2.7-pipeline-dec-when

如何使用input进行人工确认示例

DOIS

- 在pipeline-demo-4的基础上，添加

```
stage('Deploy') {
        steps {
            echo 'Deploying....  '
            script{
                env.deployEnv = input message: 'select the env to deploy', ok: 'deploy', parameters: [choice(name: 'deployEnv',
choices: 'DEV\nQA\nSTAGE\nPRODUCTION', description: 'Are you sure to deploy?')]
            }
            echo "deploy to ${env.deployEnv}"
        }
}
```

- 运行pipeline，从日志中和stage view中分别进行人工确认
- 从blue ocean界面能否进行人工确认？

如何使用post模拟post build actions示例

DOIS

- 在pipeline-demo-4的基础上，添加

```
post {
      always {
        echo "things always to run"
      }
      aborted {
        echo "things to run only for when the pipeline is aborted"
      }
      success {
        echo "things to run only for when the pipeline is Successful"
      }
      failure {
        echo "things to run only for when the pipeline is Failed"
      }
  }
```

- 可以参考2.9-pipeline-dec-post
- 完整运行流水线以及abort流水线查看对应日志输出

如何使用pipeline in SCM

DOIS

- 参看
  - 2.10-pipeline-jenkinsfile
  - 2.10-pipeline-jenkinsfile-2
  - 2.10-pipeline-jenkinsfile-3
- 注意点
- Jenkinsfile in SCM repo
- Skip Default Checkout
- Checkout scm

如何使用多分支流水线（multibranch pipeline）示例

- 可以参考2.11-multi-pipeline
- 注意点：
  - 不同于基本的pipeline类型
  - 以Jenkinsfile为标识，扫描各个分支
  - 每个分支对应folder中的一个子项目

目录

如何使用blue Ocean演示

DOIS

- 界面介绍
- 巧用编辑器生成代码
  - http://<jenkins url>/blue/organizations/jenkins/pipeline-editor/
  - 完成之后ctrl+s生成pipeline代码

如何使用Shared Libraries扩展pipeline

DOIS

- 官方文档
  - https://jenkins.io/doc/book/pipeline/shared-libraries/
- 示例jobs
  - 3.1-pipeline-shared-libs
  - https://github.com/huaqiangli/jenkins-libs

参考资料

DOIS

- 官方文档
- https://jenkins.io/doc/book/pipeline/syntax/
- https://jenkins.io/doc/book/pipeline/
- https://github.com/jenkinsci/pipeline-examples
- https://jenkins.io/doc/book/blueocean/creating-pipelines/
- https://jenkins.io/doc/book/blueocean/pipeline-editor/
- https://jenkins.io/doc/book/pipeline/shared-libraries/
- 工作坊pipeline脚本

  https://github.com/huaqiangli/dois2018SZ

  https://github.com/huaqiangli/jenkins-libs

DOIS

Thanks

DevOps 时代社区　荣誉出品

DOIS

想第一时间看到高效运维社区的新动态吗？

高维在线