Jenkins User Conference China

中国 · 深圳

南山区圣淘沙大酒店（翡翠店） 2019年4月13日

主办单位：DevOps时代 | 高效运维社区 | cloudbees

沟通群二维码

Jenkins工作坊
Jenkins 2 主要特性演练与实操

李华强 Jenkins老兵

自我介绍

- 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 (CCJE)认证者。

工作坊环境搭建

- 云主机信息（统一分配IP）
- 容器部署（预置插件，配置，demo jobs等）
  - service docker start
  - docker pull huaqiangli/jenkins:gops2019-sz-4
  - docker run -d -p 8080:8080 -p 50000:50000 huaqiangli/jenkins:gops2019-sz-4
- 访问Jenkins
  - http://<your ip>:8080/
  - 登录: admin/jenkins2019
  - 修改全局配置的 Jenkins URL 为 http://<your ip>:8080/

名称	外网ip	用户名	密码	备注
Demo	114.67.24.234	root	Jenkins@2019	讲师机
member1	106.75.148.50	root	Jenkins@2019	学员机
member2	106.75.164.134	root	Jenkins@2019	学员机
member3	106.75.168.156	root	Jenkins@2019	学员机
member4	106.75.164.182	root	Jenkins@2019	学员机
member5	106.75.166.175	root	Jenkins@2019	学员机
member6	106.75.164.102	root	Jenkins@2019	学员机
member7	106.75.146.129	root	Jenkins@2019	学员机
member8	106.75.133.82	root	Jenkins@2019	学员机
member9	106.75.145.119	root	Jenkins@2019	学员机
member10	106.75.164.226	root	Jenkins@2019	学员机
member11	106.75.153.86	root	Jenkins@2019	学员机
member12	106.75.168.138	root	Jenkins@2019	学员机
member13	106.75.154.228	root	Jenkins@2019	学员机
member14	106.75.168.239	root	Jenkins@2019	学员机
member15	106.75.152.99	root	Jenkins@2019	学员机

目录

Jenkins 2 介绍

- Jenkins版本
  - 传统的Jenkins： 1.X
  - Jenkins 2 ： 2.X
  - https://jenkins.io/2.0/
- Jenkins 2 主要特性
  - Built-in support for delivery pipelines
  - Improved usability
  - Fully backwards compatible

Jenkins Pipeline总体介绍

- Pipeline 是Jenkins2.X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 as code

什么是Jenkins Pipeline

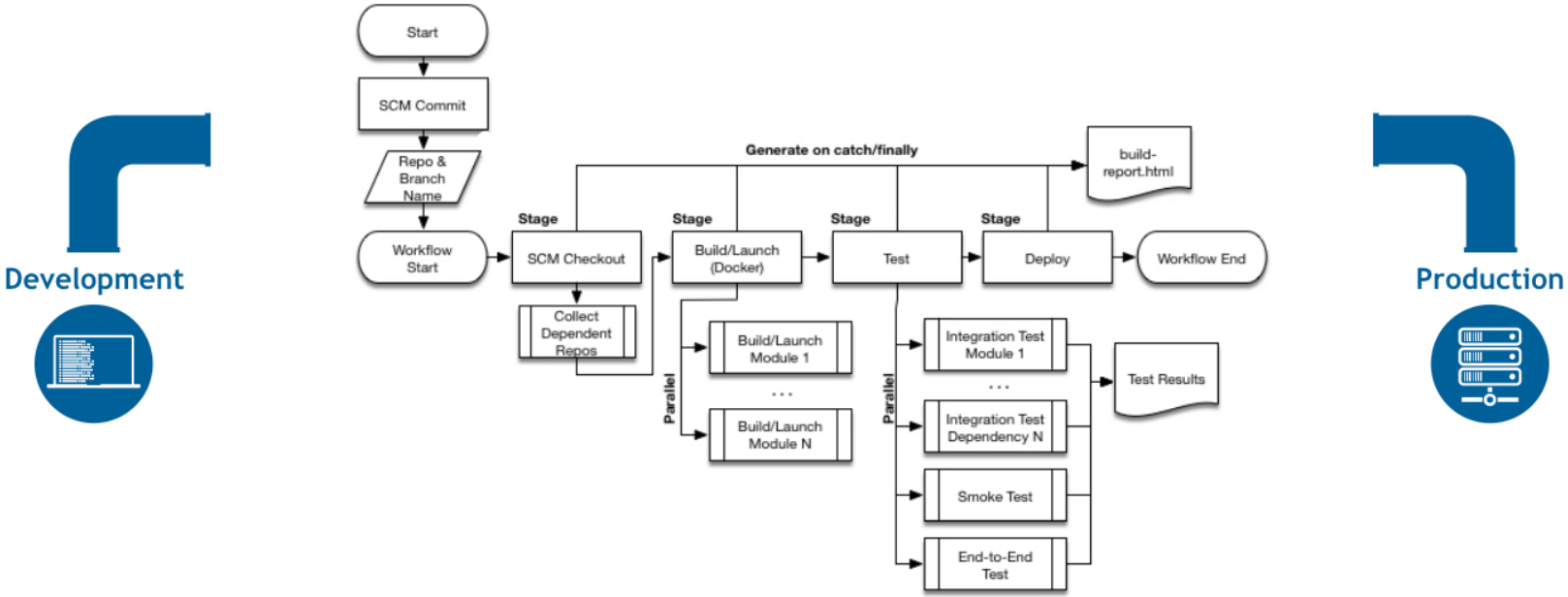
- Jenkins Pipeline是一组插件，让Jenkins可以实现持续交付流水线的落地和实施。
- Jenkins Pipeline是一种新型的project or job类型
- 持续交付流水线(CD Pipeline)是将软件从版本控制阶段到交付给用户或客户的完整过程的自动化表现。
- Pipelines as code，CI/CD Pipelines as code
- 通常，“Pipelines as code”会以 Jenkinsfile 存储在项目代码库中

为什么要用Pipeline

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



Jenkins Pipeline核心概念

- 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

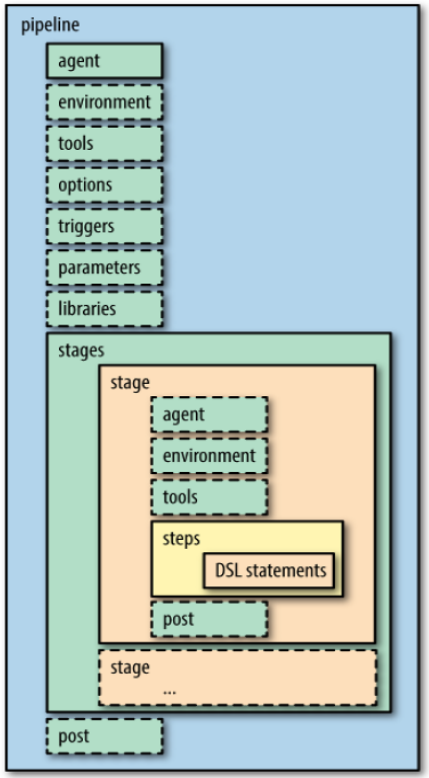
声明式Pipeline的基本语法和表达式遵循与Groovy语法相同的规则 ，但有以下例外：

- 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

- 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



目录

实战-1 pipeline 初探

- 预热传统自由风格项目示例
- 如何安装相关插件支持pipeline类型项目
- 如何创建一个基本的pipeline项目
- 两种代码集成方式介绍
- 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 pipeline示例
- 查看stage view 演示
- 如何使用Snippet Generator帮助工具
- 如何下载代码（checkout，git，svn）示例
- 如何编译，归档，发送邮件示例
- 如何使用replay功能演示

预热-Freestyle 项目示例

- 熟悉传统Jenkins的各项功能，有助于理解和使用pipeline
- 示例
  - 0.1-freestyle-sample
  - 0.2-freestyle-sample

如何安装相关插件支持pipeline类型项目

- 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

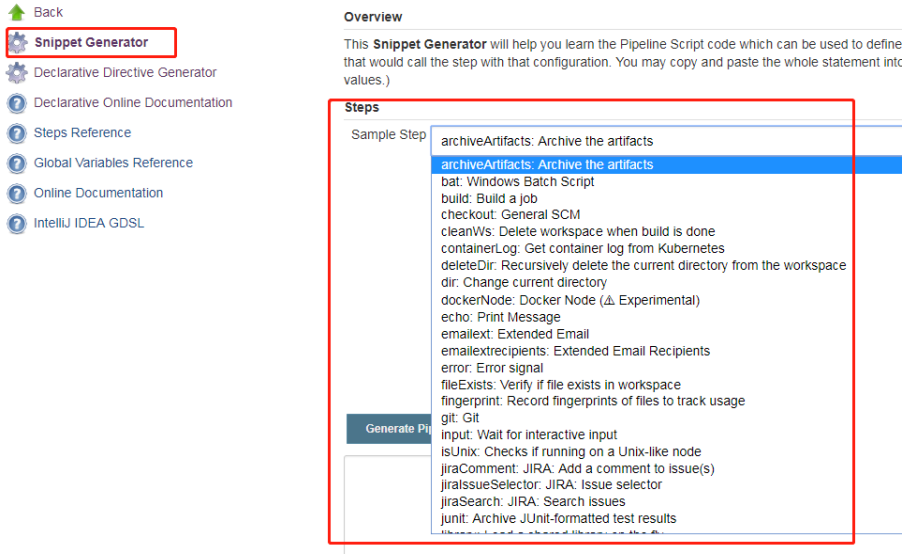
- 演示如何创建一个基本的pipeline project
- 创建一个pipeline 项目，命名 pipeline-demo-1，使用“Github+Maven” sample
- 查看 pipeline项目中的各个部分
- 注意两种代码集成方式（pipeline script，pipeline script from scm），支持两种类型的pipeline 语法（脚本式，声明式）
- 运行pipeline，查看log，stageview等

声明式 pipleline示例

- 创建一个pipeline 项目， 命名 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

- 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

- 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

- 在pipeline-demo-2的基础上，replay修改一些代码后运行
- 参考 1.4-pipeline-replay
- 注意点
  - 失败的build能不能进行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

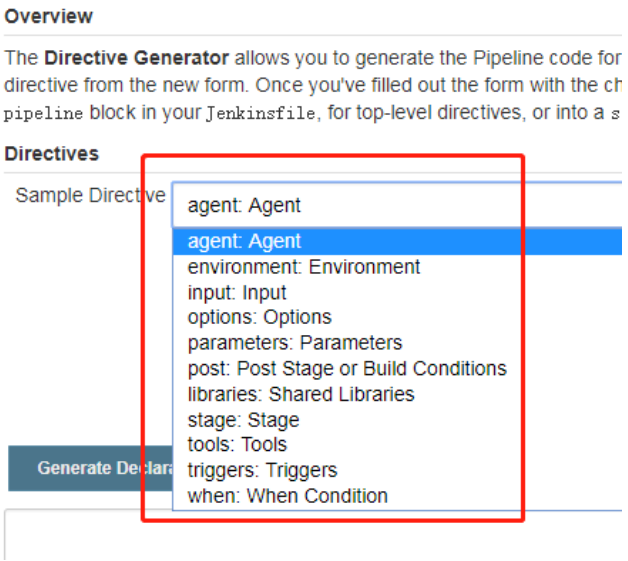
目录

实战-2 pipeline 进阶

- 声明式pipeline 主要指令介绍
- 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
- 如何使用blue Ocean
- 如何使用Shared Libraries扩展pipeline

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

- 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
- 参考 2.1-pipeline-agent

如何使用environment示例

- 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

- 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
- 扩展，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

- 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
- 扩展，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

- 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
- 扩展，脚本式pipeline中如何实现类似功能呢？

如何使用并行stages示例

- 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

- 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

- 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

- 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
- 注意，post 可以在 stage和pipeline 级别设置

如何使用pipeline in SCM

- 参看
  - 2.10-pipeline-jenkinsfile
  - 2.10-pipeline-jenkinsfile-1
  - 2.10-pipeline-jenkinsfile-2
  - 2.10-pipeline-jenkinsfile-3
- 注意点
  - Jenkinsfile in SCM repo
  - Lightweight checkout
  - Skip Default Checkout
  - Checkout scm

如何使用多分支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

如何使用blue Ocean演示

- 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

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

目录

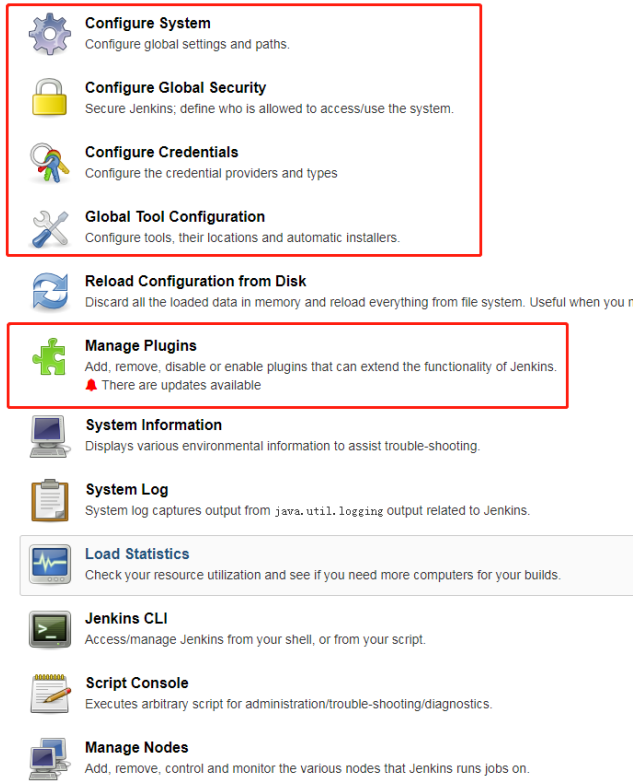
Jenkins高阶讨论

- Jenkins 安装与备份
- 理解Jenkins众多配置
- 理解Master与Agent（Slave）
- 应该选用选择哪种类型的项目
- 如何获得Jenkins帮助
- 使用Jenkins API自动化CICD
- Jenkins功能扩展开发

Jenkins 安装与备份

- 选择合适的安装方式
  - https://jenkins.io/doc/book/installing/
  - Docker镜像，WAR file，Linux Package管理工具 ？
- 物理机 VS 虚拟机 ？容器 VS 主机 ？
- 备份哪些数据？全量备份，增量备份，选择备份？

理解Jenkins众多配置



理解Master与Agent（Slave）

- Master
  - 运行Jenkins instance
  - 基于文件系统的存储（配置、工作区以及 archive，stash等）
  - 提供构建能力
- Slave
  - Jenkins分布式构建架构
  - 提供构建能力
  - SSH VS JNLP
  - 容器化Slave
- 应不应该使用Master运行构建？

应该选用选择哪种类型的项目

是否应该全部迁移或者使用pipeline类型？

- Freestyle project
  - 自由风格项目，Jenkins最主要的项目类型
- Pipeline
  - 流水线项目，适合使用pipeline（workflow）插件功能构建流水线任务
- Maven Project
  - Maven项目专用，类似 Freestyle，更简单
- Multi-configuration project
  - 多配置项目，适合需要大量不同配置（环境，平台等）构建
- Multibranch Pipeline
  - 多分支流水线项目，根据SCM仓库中的分支创建多个Pipeline项目

如何获得Jenkins帮助

- Jenkins 官方文档
  - https://jenkins.io/
  - https://plugins.jenkins.io/
- 善用Jenkins instance 内嵌帮助文档
  - 提示信息，？帮助，api文档，插件主页等
- 其他Jenkins帮助参考
  - https://www.w3cschool.cn/jenkins/
  - 谷歌，百度 搜索参考文档
- 社区交流、互助

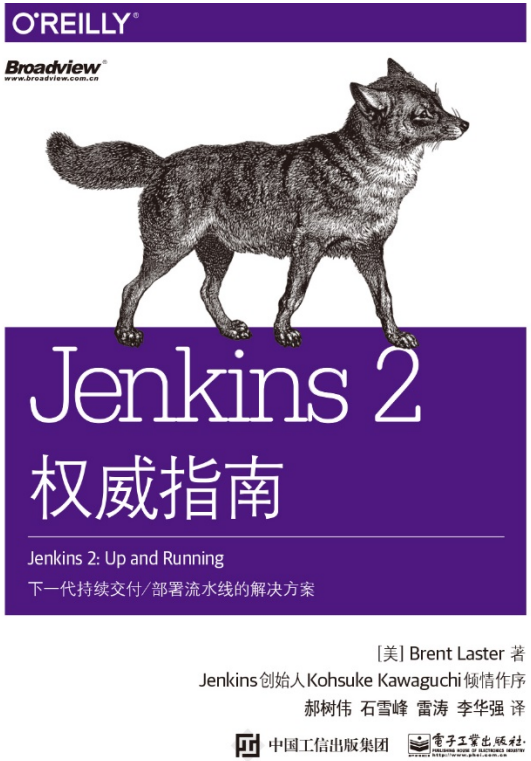
使用Jenkins API自动化CICD流程

- Jenkins Rest API
  - <Jenkins url>/api
  - <Jenkins url>/job/<job name>/api
  - <Jenkins url>/job/<job name>/<build number>/api
  - Pipeline API : https://github.com/jenkinsci/blueocean-plugin/tree/master/blueocean-rest
- API wrappers
  - https://pypi.python.org/pypi/jenkinsapi
  - https://pypi.python.org/pypi/python-jenkins
  - https://rubygems.org/gems/jenkins_api_client
- Jenkins CLI
  - java -jar jenkins-cli.jar [-s JENKINS_URL] command [options...] [arguments...]
- 参考文档
  - https://wiki.jenkins.io/display/JENKINS/Remote+access+API
  - https://wiki.jenkins.io/display/JENKINS/Jenkins+CLI

Jenkins功能扩展开发

- 插件开发
  - https://jenkins.io/doc/developer/plugin-development/
  - https://jenkins.io/doc/developer/
- Extending with Shared Libraries
  - https://jenkins.io/doc/book/pipeline/shared-libraries/
- 何时需要自主开发插件？
- 何时需要通过共享库扩展？

新书推荐


O'REILLY
Broadview
www.broadview.com.cn
Jenkins 2
权威指南
Jenkins 2: Up and Running
下一代持续交付/部署流水线的解决方案
[美] Brent Laster 著
Jenkins创始人Kohsuke Kawaguchi倾情作序
郝树伟 石雪峰 雷涛 李华强 译
中国工信出版集团

Thanks

高效运维社区

开放运维联盟

荣誉出品

想第一时间看到高效运维社区的新动态吗？