



# 持续交付概念和Pipeline入门

蒋刚毅 (Cay)

# 预习资料

## 1.持续交付&Devops综述

<https://jianggy.gitbooks.io/jenkins2/content/chapter1.html>

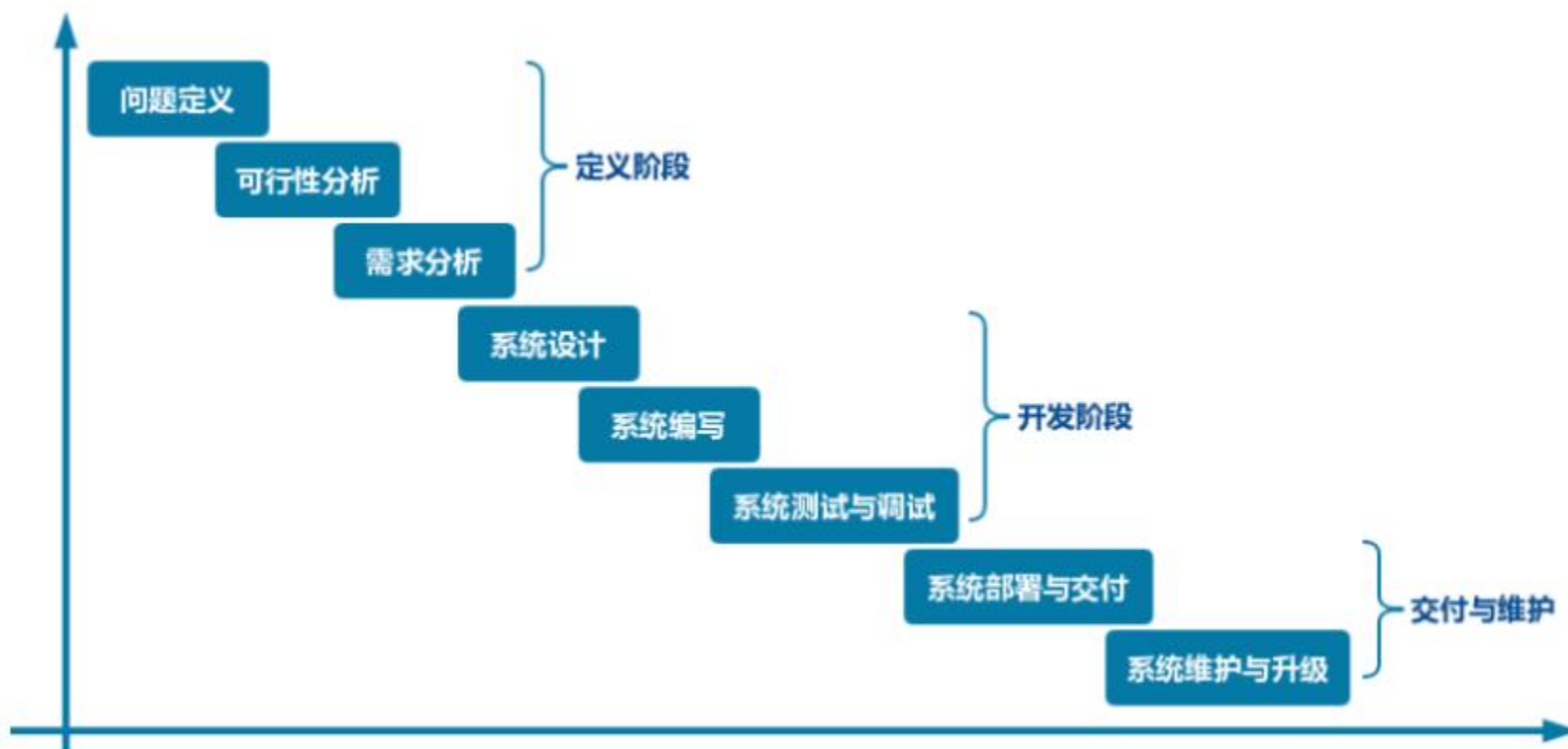
## 2.Jenkins Pipeline之快速入门

<https://jianggy.gitbooks.io/jenkins2/content/di-2-zhang-pipeline-zhi-kuai-su-ru-men.html>

## 3.Groovy语法入门

<https://www.w3cschool.cn/groovy/>

# 传统瀑布流程



# 流水线提升效率



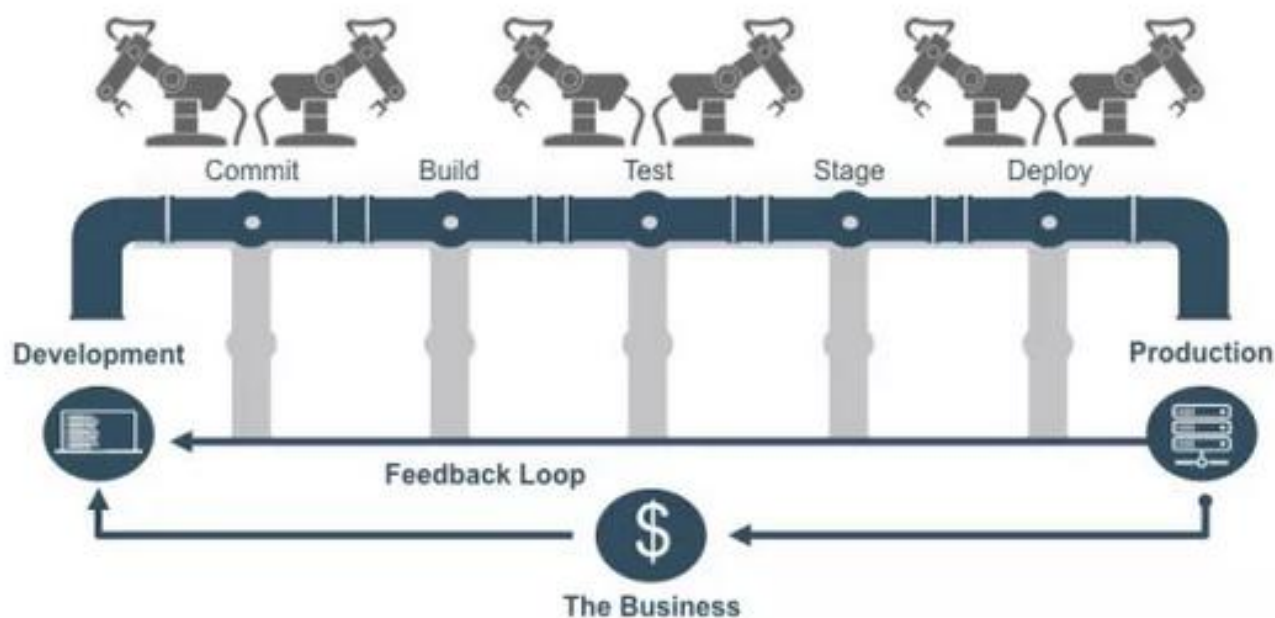
Time to Assemble a Model-T

**12.5 hrs**



**1.5 hrs**

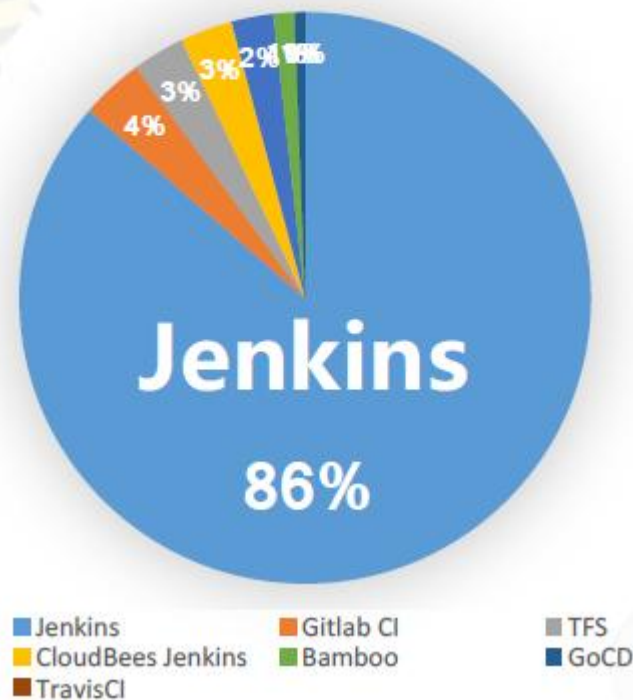
# 持续交付和自动化是答案



# Why Jenkins?

- **65%**以上公司实现了一周一次以上的部署，微服务的时代快速交付成为常态。
- **64%**的公司已经引入持续交付流水线，其中**86%**都在使用Jenkins。
- 包括**1350**个Jenkins插件在内的活跃开源社区，完善的工具生态。
- Pipeline+BlueOcean=Future

## 持续交付流水线工具



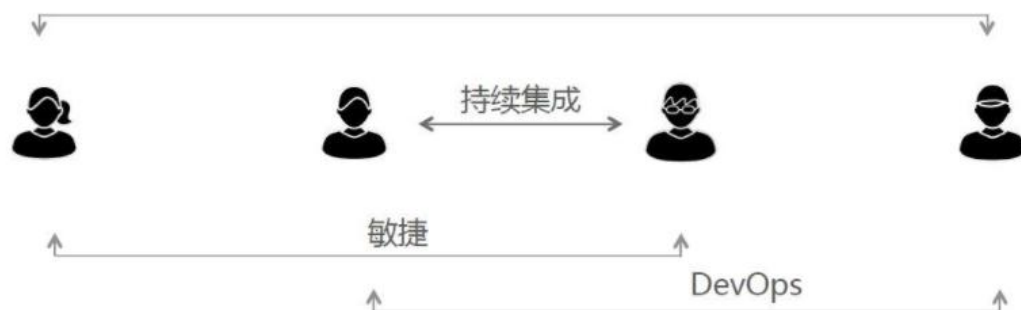
# Why Pipeline?

- **Pipeline as Code:** 任何流程都可以表述为一段Jenkinsfile脚本，并且Jenkins支持从代码库直接读取脚本。
- 以前用N个freestyle的job串联的工作流，现在只需要用1个Pipeline即可实现。
- 可通过groovy脚本无限扩展Pipeline的能力。
- 可通过共享库方式抽象公共逻辑。
- 支持大量开源工具链的集成。

# CI/CD/DevOps

从概念上来说，DevOps更关注Ops(Operations), 持续交付更关注Dev (Development)。他们的目标都是解决相同的问题，即加速软件开发，减少软件开发到交付或上线的时间，并使开发、测试、运维几个角色协作的更紧密。一般来说倾向于两者说的是同一回事，它们只不过是一枚硬币的正反面而已。

## 持续交付



持续交付流派的观点

是一种组织能力，应具有可持续性  
高质量、低成本、无风险地快速交付服务，提供业务价值

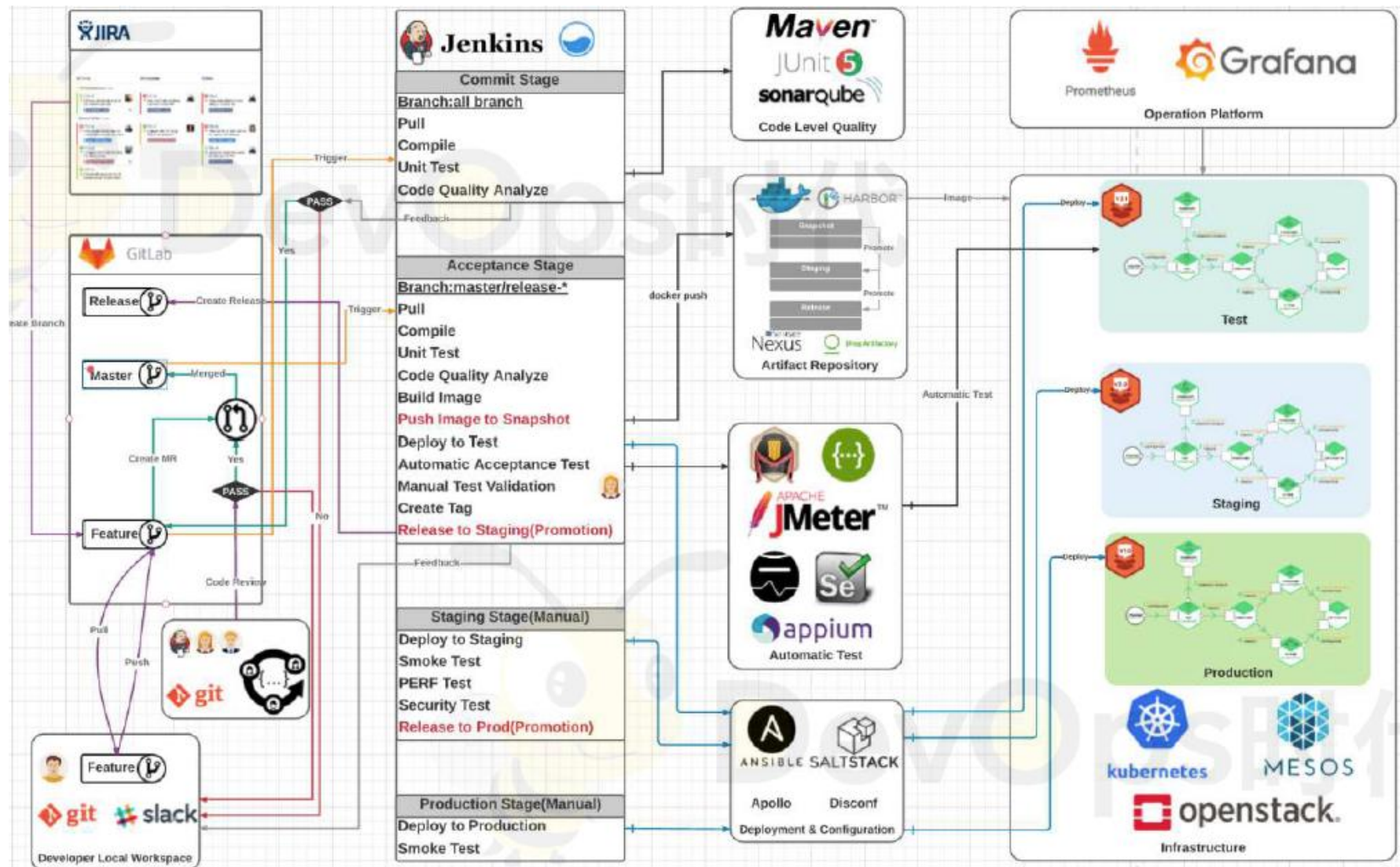
DevOps流派的观点

## 一、研发运营一体化 ( DevOps ) 过程

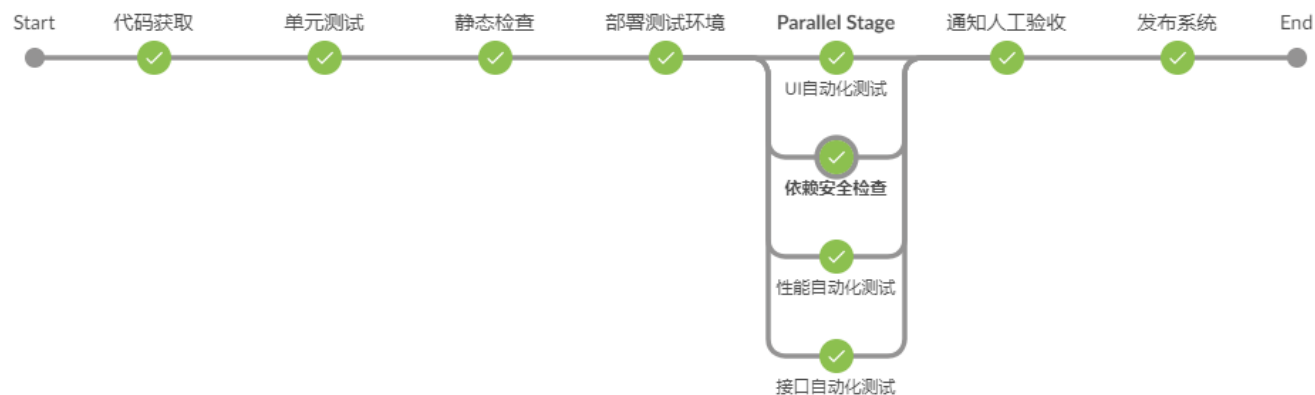
敏捷开发管理			持续交付							技术运营				
需求管理	计划管理	过程管理	配置管理	构建与持续集成	测试管理	部署与发布管理	环境管理	数据管理	度量与反馈	监控服务	数据服务	容量服务	连续性服务	运营反馈
需求收集	需求澄清和拆解	迭代管理	版本控制	构建实践	测试分级策略	部署与发布模式	环境供给方式	测试数据管理	度量指标	应用监控	数据收集能力	容量规划能力	高可用规划	业务知识管理
需求分析	故事与任务排期	迭代活动	版本可追踪性	持续集成	代码质量管理	持续部署流水线	环境一致性	数据变更管理	度量驱动改进	质量管理体系	数据处理能力	容量平台服务	质量管理体系	项目管理
需求与用例	计划变更	过程可视化及流动			测试自动化					事件响应及处置	数据告警能力	运营成本管理		业务连续性管理
需求验收		度量分析								监控平台				运营服务管理



# DevOps工具链



# 持续交付流水线示例



## 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

# 持续交付流水线示例

▶ 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

# Jenkins Pipeline定义

- 使用条件

- ✓ Jenkins 2.x或更高版本
- ✓ Pipeline插件

- **Pipeline定义**

任何Pipeline交付流程都可以表述为一段Groovy脚本，并且Jenkins支持从代码库直接读取脚本，从而实现了Pipeline as Code的理念。

Pipeline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。

# 在Web UI中定义Pipeline

1. 单击Jenkins主页上的New Item。
2. 输入Pipeline的名称，选择Pipeline，然后单击确定。
3. 在脚本文本区域中，输入Pipeline，然后单击保存。
4. 执行构建，单击“构建历史记录”下的# buildId，然后单击控制台输出以查看Pipeline的完整输出。

The screenshot shows the Jenkins web interface for a Pipeline named "test". The left sidebar contains navigation links: Back to Dashboard, Status, Changes, 立即构建 (Build Now), 删除 Pipeline (Delete Pipeline), 配置 (Configure), Open Blue Ocean, Full Stage View, Splunk, Pipeline Syntax, and Poll Mailbox Trigger Log. The main content area displays the "Pipeline test" configuration. A red box highlights the "Stage View" button. Below it, a table shows the average stage times for Build (436ms), Test (272ms), and Deploy (379ms). The "Build History" section on the left shows a list of builds with a search bar and a "构建历史" (Build History) link. The "Recent Changes" section shows a list of changes with a "Recent Changes" link.

	Build	Test	Deploy
Average stage times:	436ms	272ms	379ms
#121 Sep 07 15:19 No Changes	206ms	214ms	211ms
#120 Sep 07 15:19 No Changes	219ms	153ms	206ms

# 初识Pipeline脚本

```
pipeline{
  agent any
  stages{
    stage('Build') {
      steps {
        echo "make"
      }
    }
    stage('Test') {
      steps {
        /* `make check` returns non-zero on test failures,
         * using `true` to allow the Pipeline to continue nonetheless
         */
        echo "make check || true"
      }
    }
    stage('Deploy') {
      steps {
        echo "make publish"
      }
    }
  }
}
```

## 基本概念:

**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。

**stages:**阶段，一个Pipeline可以划分为若干个Stage，每个Stage代表一组操作。

**steps:**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。

# Pipeline内置文档

- **内置文档：** Pipeline内置文档可以更轻松地创建不同复杂性的Pipeline，根据Jenkins中安装的插件自动生成和更新内置文档。

[链接：localhost:8080/pipeline-syntax/](http://localhost:8080/pipeline-syntax/)

- **代码段生成器：** 内置的“Snippet Generator”程序有助于为单个步骤生成代码段。



The screenshot shows the 'Steps' section of the Jenkins Pipeline configuration page. It features a 'Sample Step' dropdown menu with 'stage: Stage' selected. Below this is a 'Stage Name' input field containing the text 'Deploy'. To the right of the input field is a red upward-pointing arrow and a help icon. A blue button labeled 'Generate Pipeline Script' is positioned below the input field. The output area at the bottom displays a code snippet: 

```
stage('Deploy') {  
    // some block  
}
```

- **全局变量引用：**仅包含Pipeline提供的变量，这些变量可用于Pipeline。

## ENV

Pipeline脚本可访问的环境变量，例如：`env.PATH`或`env.BUILD_ID`。可参阅内置的全局变量，以获取管道中可用的完整和最新的环境变量列表。

## PARAMS

将为Pipeline定义的所有参数公开，例如：`params.MY_PARAM_NAME`。

## currentBuild

可获取当前正在执行的Pipeline job的信息，例如属性`currentBuild.result`，`currentBuild.displayName`等。

# groovy语法简介

- Groovy是一种基于JVM（Java虚拟机）的敏捷开发语言，它结合了Python、Ruby和Smalltalk的许多强大的特性，Groovy 代码能够与 Java 代码很好地结合，也能用于扩展现有代码。由于其运行在 JVM 上的特性，Groovy 可以使用其他 Java 语言编写的库。
- Groovy是JVM的一个替代语言（替代是指可以用 Groovy 在Java平台上进行 Java 编程），使用方式基本与使用 Java代码的方式相同，该语言特别适合与Spring的动态语言支持一起使用，设计时充分考虑了Java集成，这使 Groovy 与 Java 代码的互操作很容易。（注意：不是指Groovy替代java，而是指Groovy和java很好的结合编程。

入门教程: <https://www.w3cschool.cn/groovy/>



# 课后习题

- 理解Pipeline As Code的理念
- 理解持续集成、持续交付和DevOps的关系
- 在Jenkins中安装Pipeline插件
- 在Jenkins中建立Pipeline Job，完成一个简单Pipeline脚本的编写和执行。
- 熟悉Pipeline内置文档