Jenkinsfile 讲义(第一期)

什么是 jenkinsifle

在讲 jenkinsfile 之前，得先提一下 pipeline，pipeline 是一个插件，在 jenkins 安装完成后，会有官方推荐的插件下载，其中就一个 pipeline 这个插件

作用：将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流程编排与可视化。

新建一个 pipeline 项目，展示一下普通 pipeline 和 jenkinsfile

官方释义：Jenkinsfile 是 Jenkins 2.x 核心特性 Pipeline 的脚本，由 Groovy 语言实现

个人理解：
Jenkinsfile 就是一个文本文件，内部记录的内容是一套 Jenkins 执行的流程，支持并行执行，Jenkinsfile 文件一般存放于代码仓库的根目录，随着代码走，即使 Jenkins 需要重装或者安装 Jenkins 的机器出了问题，Jenkinsfile 所记录的流程也可以很迅速的重新部署。
和传统 Jenkins 的使用方法不同的是，传统的 Jenkins 使用方案是建立一个 job，然后在 job 内部添加各个行为和事件，复杂一点的会进行 job 间的串联，但是这种运用方式会产生一些问题，当一个复杂任务窜连了多个 job 后，维护起来会变得很麻烦，当 jenkins 的主节点机器发生问题时，灾备的恢复成本也会变得很高，因此，Jenkinsfile 的出现解决了这些问题，它把复杂的流程记录在一个文本文件中，语法简单明了，随着代码一起维护，Jenkinsfile 秉承的一个理念就是 Jenkinsfile as Code。
Jenkinsfile 有 2 种写法，一种是 Declarative，另一种是 Groovy，通常 Declarative 的写法可以满足大部分场景的需要，相对来说学习成本更低，Groovy 的写法更灵活，比如 Jenkins 的 Shared Library 模式就需要用到 Groovy。
在 Jenkins 中有一种工作任务名为 pipeline，pipeline 这种类型的任务又可以有 2 种模式，一种是在 pipeline 的配置界面中，写下 pipeline

语法，另一种模式就是从仓库中检出 Jenkinsfile 文件进行读取，执行，pipeline 的语法和 Jenkinsfile 的语法是一样的，这 2 个模式的区别就是，前者是把执行流程写在了 Jenkins 中，后者则是把执行流程写在了一个名为 Jenkinsfile 的文件中，并把它存放在代码仓库的根目录，随版本一起维护。

pipeline{ ... }
pipeilne 是 jenkinsfile 的起始点，所有的流程都必须包含在 pipeline 语句块里面

agent
agent 必须在顶端定义，也可以在 stage 内部定义
但是一定要写，如果不分配节点，那就写 agent none，不写会报错
顶部写了 agent none 后，就必须在每个 stage 下都给他指定节点

agent 指定运行任务的节点，它有几个预设的关键字
any：在任何可用的节点上执行流水线或阶段，通常默认是 master
none：不分配节点，假如在 pipeline 块顶层声明 agent none，则在每个 stage 块中需要声明指派的节点。
label：指定运行的节点，如 agent { label : 'slave_win'}，表示指定在名为 slave_win 的节点上执行任务。

stages{ ... }
一个完整的 pipeline 中必须包含至少一个 stages，一个 stages 包含一个或多个 stage
stages 的书写方式 是 stages 后面跟个花括号{}，带着用户写

stage(...){...}
stage 必须被包含在 stages 中，不可以脱离 stages 独立存在，流水线所做的实际工作都被封装进一个或多个 stage 中，stage 必须后面跟上括号，stage(")的形式，括号内为此步骤的名称。stage 内部不能包含 stage，但是可以包含 stages。一个 stage 中必须包含至少一个 steps

steps{...}
在给定的 stage 中定义，一个 steps 块中可以定义多个步骤

```
pipeline{
   agent any
```

```
    stages{
        stage('打包'){
            steps{
                sh label:'', script: 'echo build'
            }
        }
    }
}

pipeline{
    agent any
    stages{
        stage('打包'){
            steps{
                bat label:'', script: 'echo build'
            }
        }
    }
}

pipeline{
    agent none
    stages{
        stage('打包'){
            agent {label: 'windows'}
            steps{
                sh label:'', script: 'echo build'
            }
        }
    }
}
```

post{ ... }
非必须，post 块执行的顺序取决于 post 块存在的位置，在每个 stage 下可以包含 post 块，同样，如果把 post 块放在 stages 后，那在整个 stages 执行完成后，才会执行 post 块内部的指令。他是按照顺序执行的
post 语句块里有 11 中常用的

always { ... } 不管构建状态怎么样，始终会运行

aborted { ... } 当构建状态“中止”时运行

success { ... } 如果构建状态为“成功”或尚未设置，则运行

failure { ... } 如果构建状态为“失败”，则运行

```
pipeline{
    agent any
    stages{
        stage('打包'){
            steps{
                sh label:'', script: 'echo build'
            }
        }
        stage('发布'){
            steps{
                sh label:'', script: 'echo deploy'
            }
        }
    }
    post{
        always{
            sh label:'', script: 'echo always'
        }
        success{
            sh label:'', script: 'echo success'
        }
        failure{
            sh label:'', script: 'echo failure'
        }
        aborted{
            sh label:'', script: 'echo aborted'
        }
    }
}
```