教程（一）快速入门

什么是pipeline

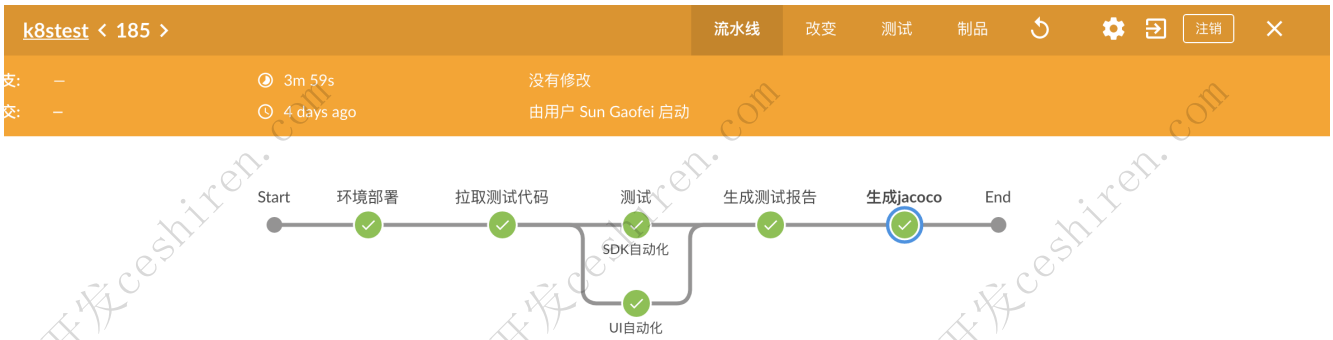
先介绍下什么是Jenkins 2.0，Jenkins 2.0的精髓是Pipeline as Code，是帮助Jenkins实现CI到CD转变的重要角色。什么是Pipeline，简单来说，就是一套运行于Jenkins上的工作流框架，将原本独立运行于单个或者多个节点的任务连接起来，实现单个任务难以完成的复杂发布流程。Pipeline的实现方式是一套Groovy DSL，任何发布流程都可以表述为一段Groovy脚本，并且Jenkins支持从代码库直接读取脚本，从而实现了Pipeline as Code的理念。
Pipeline的几个基本概念：

1. Stage:
   阶段，一个Pipeline可以划分为若干个Stage，每个Stage代表一组操作。注意，Stage是一个逻辑分组的概念，可以跨多个Node。
2. 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。
3. 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。

本节介绍Jenkins Pipeline的一些核心概念，并介绍在运行的Jenkins实例中定义和使用Pipelines的基础知识。

为什么使用pipeline

先看效果图：



这是SDP的pipeline，我们看到在这一个pipeline中我们整合了以前需要好多个job才能做的事情。
而pipeline的优势之一，就是它通过pipeline as code的理念，把原来需要N个job才能完成的流程全都放到了一个job里。
并且提供了良好的workflow视图供我们查看。 上面的图中我们点击任何一个阶段都会有详细的日志和运行参数可以查看，方便我们观察每一个阶段的工作。

而pipeline的第二个优势在于通过pipeline的代码你能一眼纵观整条pipeline的所有工作内容。
在以前我们通过UI配置job的时候我们想要查看这个job的全貌是比较难的，一个是因为UI太多，另一个也是因为一条pipeline分散到各个job中，来回切换很麻烦。
所以你难以得知整个工作流的全貌而变得难以维护。而pipeline则可以把配置全以代码形式的方式放在一个脚本框中，每一个阶段的工作内容全都一目了然。

第三个优势在于pipeline通过代码编写利于维护， 尤其是pipeline from scm的支持，我们可以在gitlab中维护我们所有的pipeline，jenkins会直接读取git中的脚本进行构建。

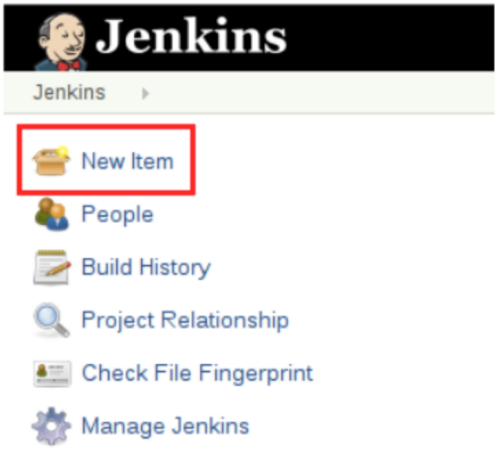
第四个优势在与pipeline可以扩展groovy语言， 我们通过自己开发jenkins的shared library，封装很多通用功能。
这样能强化job的能力，做到以前我们做不到的事情。

Pipeline定义

Pipeline脚本是用Groovy写的 。
可以通过以下任一方式创建基本Pipeline：
pipeline script：直接在Web UI的script输入框里面输入pipeline script语句即可，参考说明可以点击输入框下边的Pipeline Syntax，里面有很多示例操作说明，非常好用。
pipeline script from SCM：需要配置SCM代码存储Git地址或SVN地址，指定script文件Jenkinsfile所在路径，每次构建job会自动去指定的目录执行script文件
以上两种方法定义Pipeline的语法都是一样的。

在Web UI中定义Pipeline

要在Jenkins Web UI中创建基本Pipeline Job，请按照下列步骤操作：
单击Jenkins主页上的New Item。

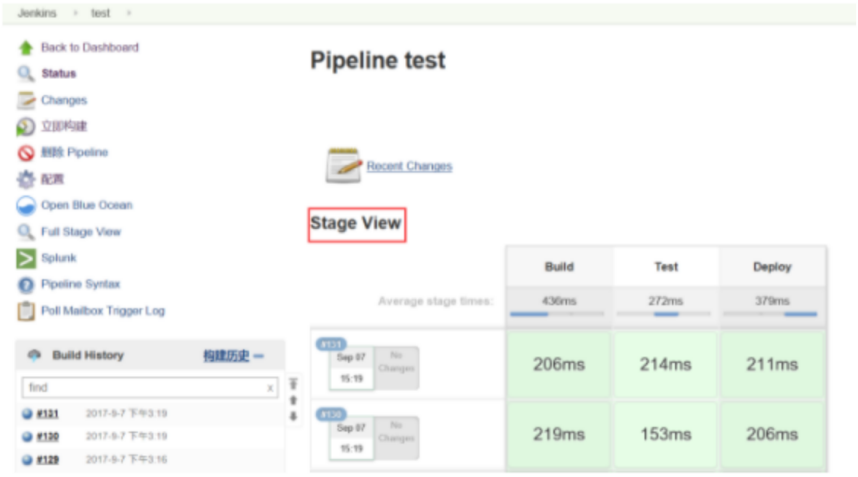


输入Pipeline的名称，选择Pipeline，然后单击确定。

在脚本文本区域中，输入Pipeline，然后单击保存。

```
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"
            }
        }
    }
}
```

跑一下然后单击控制台输出以查看Pipeline的完整输出



在SCM中定义pipeline

复杂的Pipeline难以在Pipeline配置页面的文本区域内进行写入和维护。为了解决这一问题，jenkins Pipeline支持在文本编辑器中编写脚本文件jenkinsFile，Jenkins可以通过从SCM选项的控件中加载Pipeline脚本。我们可以配置git的地址，而我们统一的地址是：https://gitlab.4pd.io/qa/jenkinsfile

选择SCM选项中的Pipeline脚本后，不要在Jenkins UI中输入任何Groovy代码; 只需指定要检索的Pipeline脚本的路径就可以了。jenkins会自动的从git中下载jenkinsfile

内置文档

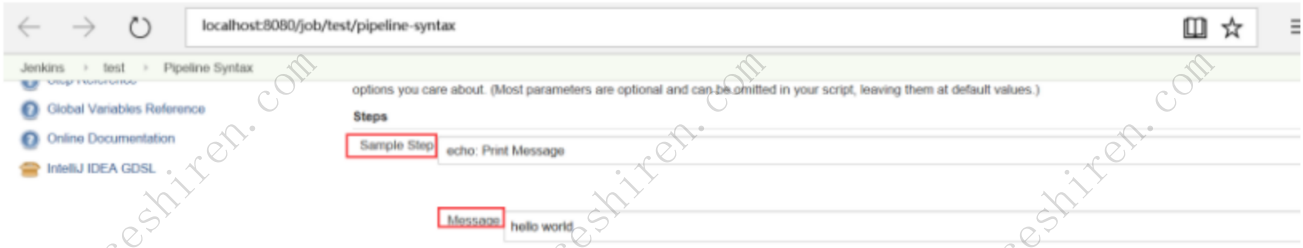
Pipeline配有内置的文档功能，可以更轻松地创建不同复杂性的Pipeline。根据Jenkins实例中安装的插件自动生成和更新内置文档。内置文档可以在全局范围内找到：localhost:8080/pipeline-syntax/，假设您有一个Jenkins实例在本地端口8080上运行。同样的文档也作为pipeline语法链接到任何配置的Pipeline的侧栏中项目。

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。Snippet Generator动态填充Jenkins实例可用的步骤列表。可用的步骤数量取决于安装的插件，它明确地暴露了在Pipeline中使用的步骤。要使用代码段生成器生成步骤代码片段：

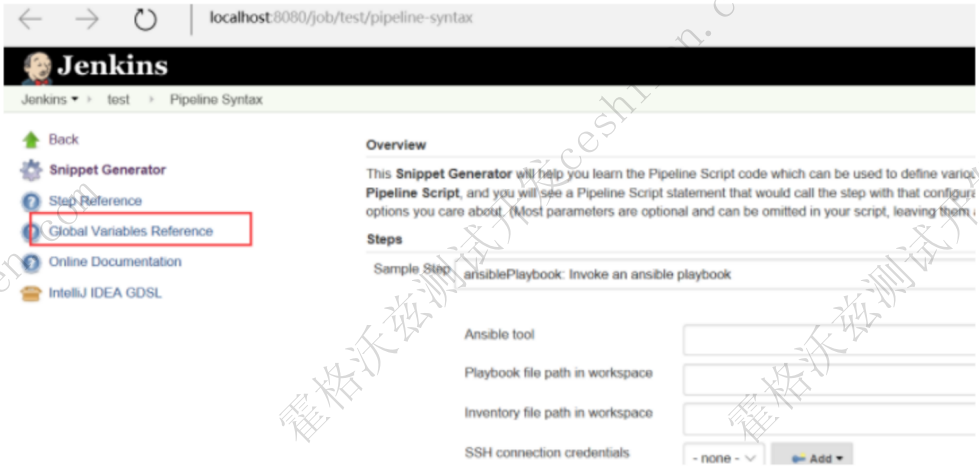
1、从配置的流水线或本地主机：8080 / pipeline-syntax导航到Pipeline语法链接Pipeline Syntax

2、在“ Sample Step”下拉菜单中选择所需的步骤,使用“Sample Step”下拉列表下方的动态填充区域配置所选步骤,如message为“hello world”，单击生成Pipeline脚本以创建一个可以复制并粘贴到Pipeline中的Pipeline代码段



全局变量引用

除了代码片段生成器之外，Pipeline还提供了一个内置的“ 全局变量引用”。像Snippet Generator一样，它也是由插件动态填充的。与代码段生成器不同的是，全局变量引用仅包含Pipeline提供的变量，这些变量可用于Pipeline。



在Pipeline中默认提供的变量是：
ENV
Pipeline脚本可访问的环境变量，例如：
env.PATH或env.BUILD_ID。请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等等

引用官方文档：https://jenkins.io/doc/book/pipeline/getting-started/
Groovy语法及入门：
Groovy入门之语法和变量定义
Groovy进阶之函数、闭包和类
精通 Groovy