使用 Jenkins 构造 CI/CD 流水线

一、关于 CI/CD 术语的简要介绍

持续集成： CI 是开发人员每天多次将新代码集成到制品库中的过程。这是比传统方法更好的方法，在传统方法中，开发人员将隔离构建新代码，然后在项目生命周期结束时将其集成到主存储库中。主要目标是在初始阶段检测到任何集成错误，以便可以迅速对其进行纠正。每当新代码与现有主存储库合并时，它将触发新的构建。针对这些新版本执行测试运行以检查是否有破损。

持续交付： CD 确保自动执行软件交付过程，并保障将集成代码交付到生产阶段，而不会造成任何错误或延迟。DevOps 实施过程中的 CD 可帮助开发人员一致地将新代码与主分支合并，以便他们可以构建即时软件就绪的产品。它负责检查代码的质量，并执行测试以检查它是否可以将功能构建释放到生产环境中。

持续部署：自动化的最关键部分发生在交付管道的此阶段。只要代码中有重要更改，相应的构建和部署就会同时发生。它是通过连续部署过程实现的，该过程允许对通过 CI 阶段的每个代码修改执行实时部署更改。在此阶段，从初始代码开始直到代码进入生产阶段，都无需进行人工干预。

二、CI/CD 流程工具

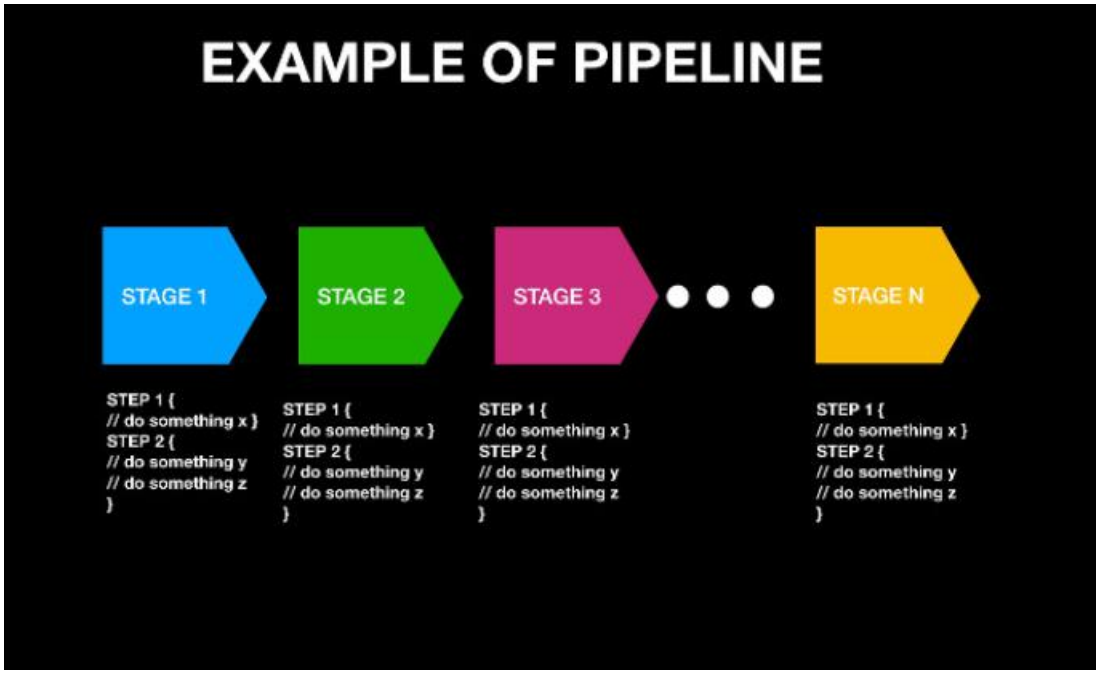
Jenkins 开源工具自动化 CI/CD 流程。

三、什么是流水线？

流水线是一个单独的概念，指的是按顺序连接在一起的事件或作业组：

“ 流水线(pipeline)”是可以执行的一系列事件或作业。

对流水线的可视化理解如下：



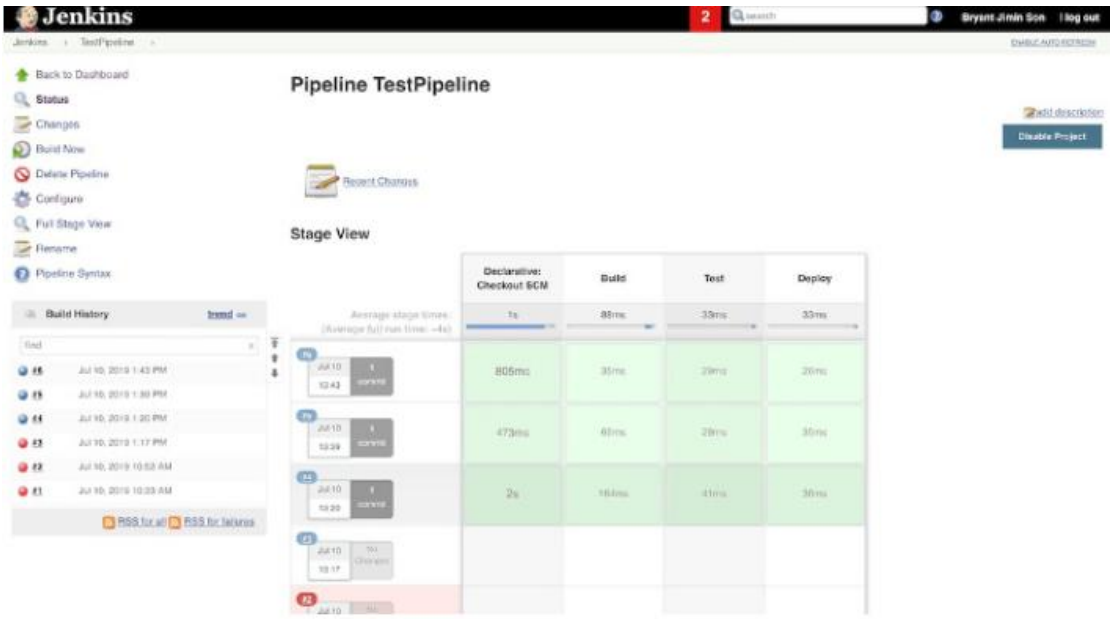
Jenkins 流水线以编码脚本的形式提供，通常称为 “Jenkinsfile”，尽管可以用不同的文件名。下面这是一个简单的 Jenkins 流水线文件的示例：

```
pipeline {
   agent any

   stages {
      stage('Build') {
        steps {
          echo 'Building...'
          echo "Running ${env.BUILD_ID} ${env.BUILD_DISPLAY_NAME} on ${env.NODE_NAME} and JOB ${env.JOB_NAME}"
        }
   }
   stage('Test') {
     steps {
        echo 'Testing...'
     }
   }
   stage('Deploy') {
     steps {
       echo 'Deploying...'
     }
   }
  }
}
```

Jenkins 流水线是一种以定义的方式依次执行 Jenkins 作业的方法，方法是将其编码并在多个块中进行结构化，这些块可以包含多个任务的步骤。

最终成果为：



Final Result

四、构建 Jenkins 流水线

（1）在开始前，环境配置需求：

1.Java 开发工具包（JDK）：如果尚未安装，请安装 JDK 并将其添加到环境路径中，以便可以通过终端执行 Java 命令（如 java jar）。这是利用本教程中使用的 Java Web Archive（WAR）版本的 Jenkins 所必需的（尽管你可以使用任何其他发行版）。

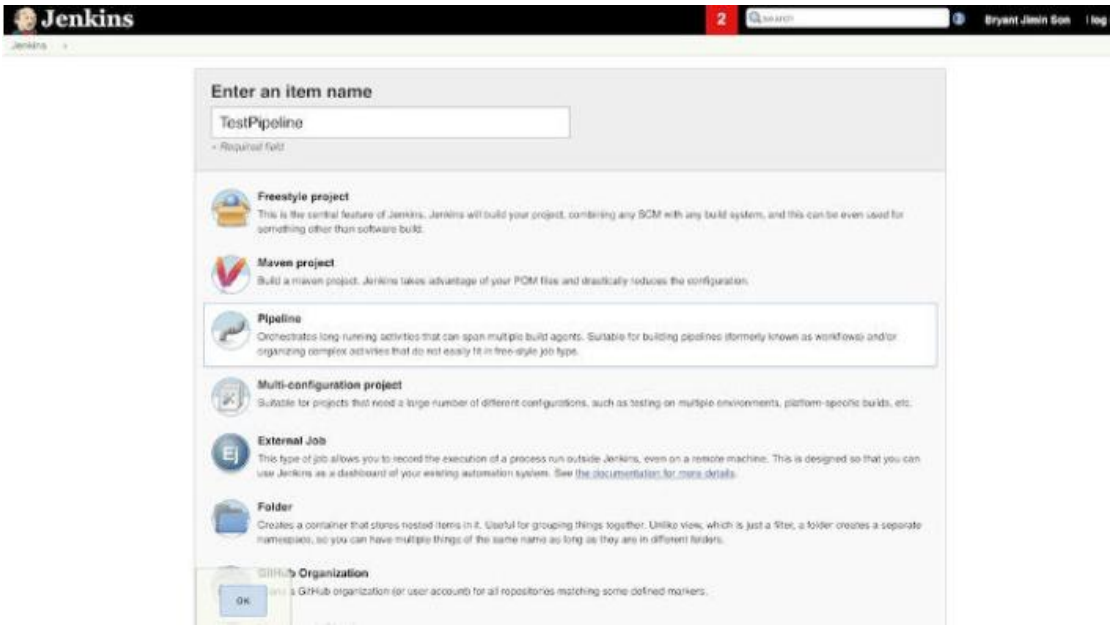
2.然后下载 jenkins（此次用的是 windows 版本）

（2）安装完毕后，以 Java 二进制方式执行 Jenkins，打开终端，进入 jenkins.war 所在的目录（确保 JDK 被添加到环境路径），执行以下命令，便可：

```
java -jar ./jenkins.war
```

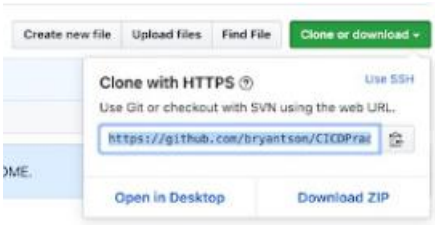
若顺利，则打开 web 浏览器，输入 localhost:8080/便可进入 jenkins 界面。

（3）create a new job（创建一个新作业）

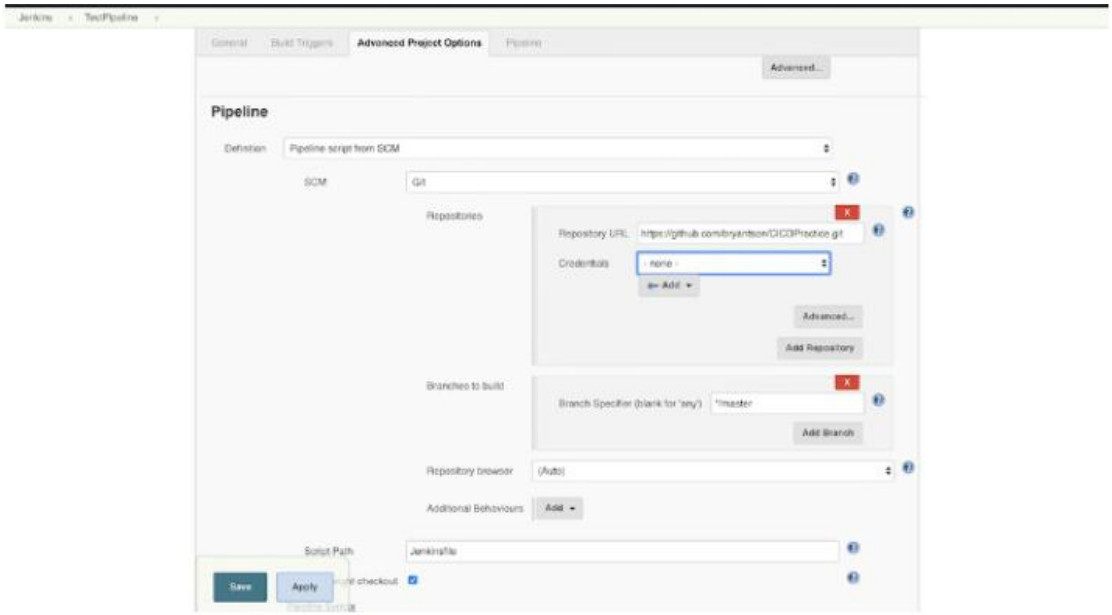


构建流水线的方法有两种，一种是直接编写流水线脚本，另一种是从 SCM 中检索 jenkins file。

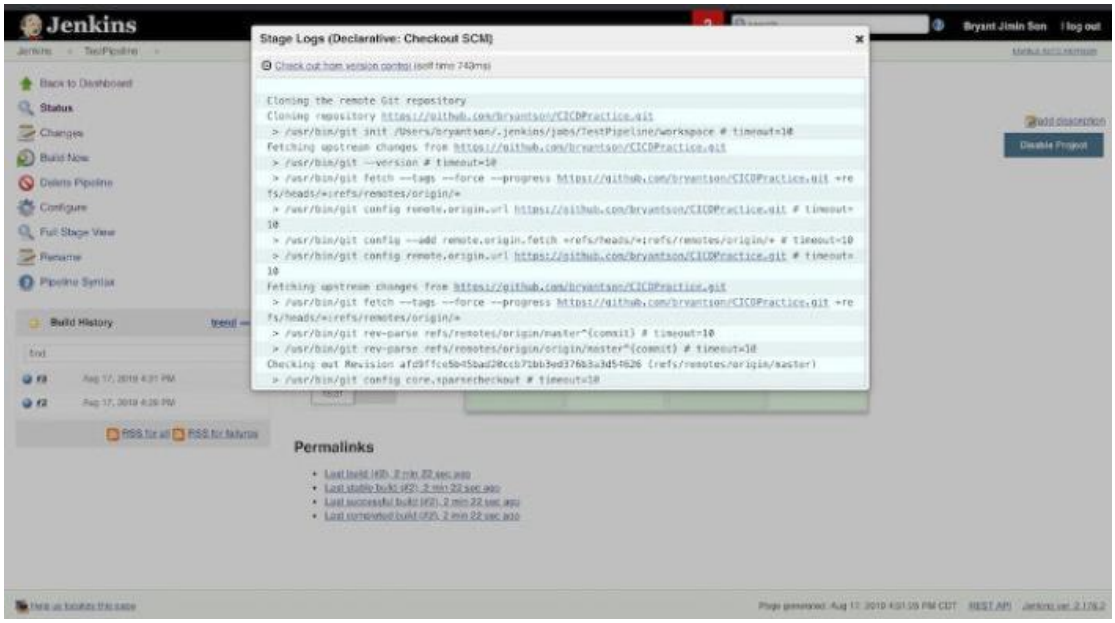
（4）我们采取通过 SCM 配置并执行流水线作业



将其 URL 复制下来，单击“Configure”以修改现有作业。滚动到“Advanced Project Options”设置，但这一次，从 “Destination” 下拉列表中选择 “Pipeline script from SCM” 选项。将 GitHub 存储库的 URL 粘贴到 “Repository URL” 中，然后在 “Script Path” 中键入 “Jenkinsfile”。 单击 “Save” 按钮保存。



要构建流水线，回到 “Task Overview” 页面后，单击 “Build Now” 以再次执行作业。然后，单击某阶段查看“Log”检查进行过程。


Jenkins
Bryant Jimin Son
log out
Stage Logs (Declarative: Checkout SCM)
Check out from version control (self time 749ms)
Cloning the remote Git repository
Cloning repository https://github.com/bryantson/CICDPractice.git
> /usr/bin/git init /Users/bryantson/.jenkins/jobs/TestPipeline/workspace # timeout=10
Fetching upstream changes from https://github.com/bryantson/CICDPractice.git
> /usr/bin/git --version # timeout=10
> /usr/bin/git fetch --tags --force --progress https://github.com/bryantson/CICDPractice.git +refs/heads/*:refs/remotes/origin/*
> /usr/bin/git config remote.origin.url https://github.com/bryantson/CICDPractice.git # timeout=10
> /usr/bin/git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
> /usr/bin/git config remote.origin.url https://github.com/bryantson/CICDPractice.git # timeout=10
Fetching upstream changes from https://github.com/bryantson/CICDPractice.git
> /usr/bin/git fetch --tags --force --progress https://github.com/bryantson/CICDPractice.git +refs/heads/*:refs/remotes/origin/*
> /usr/bin/git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> /usr/bin/git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision afd9ffce5b455bad20ccb71bb3ed376b3a3d54026 (refs/remotes/origin/master)
> /usr/bin/git config core.sparsecheckout # timeout=10
Back to Dashboard
Status
Changes
Build Now
Delete Pipeline
Configure
Full Stage View
Rename
Pipeline Syntax
Build History
Disable Project
Permalinks