View the lesson (Creating Your First Pipeline)

What is a Jenkins Pipeline?

Jenkins Pipeline (or simply "Pipeline") is a suite of plugins that supports implementing and integrating continuous delivery pipelines into Jenkins.

- A continuous delivery pipeline is an automated expression of your process for getting software from version control right through to your users and customers.
- Jenkins Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines "as code". The definition of a Jenkins Pipeline is typically written into a text file (called a Jenkinsfile) which in turn is checked into a project's source control repository.

Tips:
Jenkins pipelines can be defined using a text file called JenkinsFile. You can implement pipeline as code using JenkinsFile, and this can be defined by using a domain specific language (DSL). With JenkinsFile, you can write the steps needed for running a Jenkins pipeline.

> There are two types of the syntax used for defining your JenkinsFile.

1. Declarative: Declarative pipeline syntax offers an easy way to create pipelines. It contains a predefined hierarchy to create Jenkins pipelines.

```
pipeline {
    /* insert Declarative Pipeline here */
}
```

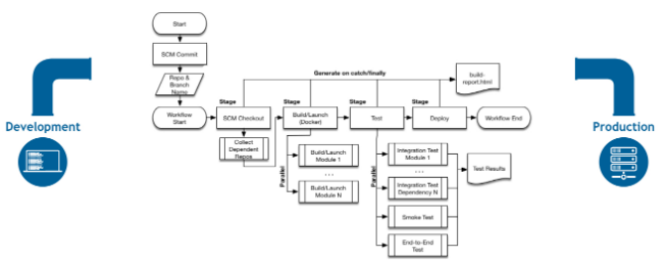
2. Scripted: Scripted Jenkins pipeline runs on the Jenkins master with the help of a lightweight executor. It uses very few resources to translate the pipeline into atomic commands.

```
node {
    stage ('Build') {
        //...
    }
    stage ('Test') {
        //...
    }
}
```

Pipeline Concepts

- Jenkins is, fundamentally, an automation engine that supports a number of automation patterns. Pipeline adds a powerful set of automation tools onto Jenkins, supporting use cases that span from simple continuous integration to comprehensive CD pipelines. By modeling a series of related tasks, users can take advantage of the many features of Pipeline:
- **Code:** Pipelines are implemented in code and typically checked into source control, giving teams the ability to edit, review, and iterate upon their delivery pipeline.
- **Durable:** Pipelines can survive both planned and unplanned restarts of the Jenkins master.
- **Pausable:** Pipelines can optionally stop and wait for human input or approval before continuing the Pipeline run.
- **Versatile:** Pipelines support complex real-world CD requirements, including the ability to fork/join, loop, and perform work in parallel.
- **Extensible:** The Pipeline plugin supports custom extensions to its DSL and multiple options for integration with other plugins.

> The flowchart below is an example of one CD scenario easily modeled in Jenkins Pipeline:



Pipeline Concepts

Pipeline: A Pipeline is a user-defined model of a CD pipeline. A Pipeline's code defines your entire build process, which typically includes stages for building an application, testing it and then delivering it.

Node: A node is a machine that is part of the Jenkins environment and is capable of executing a Pipeline.

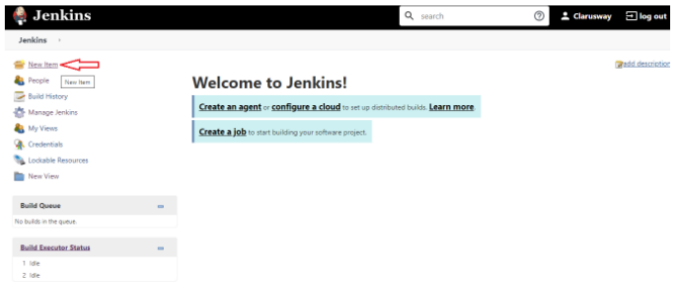
Stage: A stage block defines a conceptually distinct subset of tasks performed through the entire Pipeline (e.g. "Build", "Test" and "Deploy" stages), which is used by many plugins to visualize or present Jenkins Pipeline status/progress.

Step: A single task. Fundamentally, a step tells Jenkins what to do at a particular point in time (or "step" in the process). For example, to execute the shell command make use of the sh step: sh 'make'.

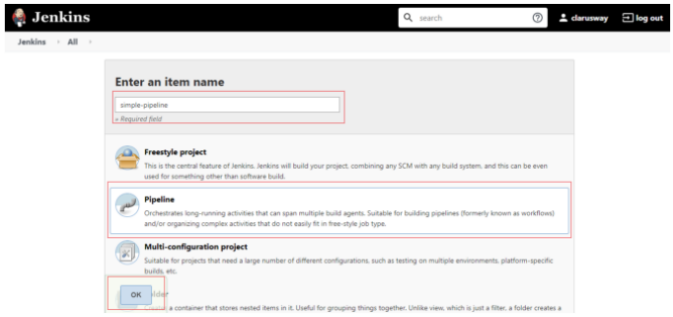
Single Step Pipeline

Let's create our first Jenkins Pipeline `Clarusway_Way to Reinvent Yourself`

1 After logging in to Jenkins click on the "New Item" menu option



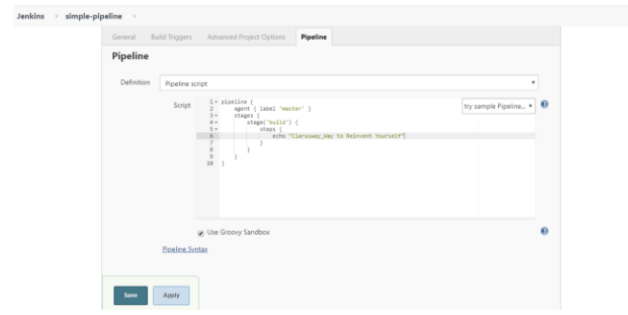
2 Type in the name of the Jenkins Pipeline, Click on the "Pipeline". Then press the "OK" button.



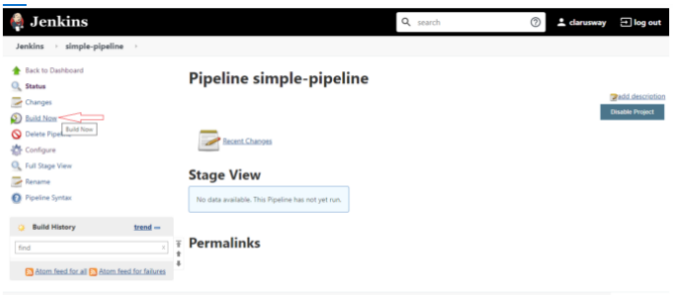
3 It will take you directly to the "Configuration" page of the project, select the section called "Pipeline", paste the following code

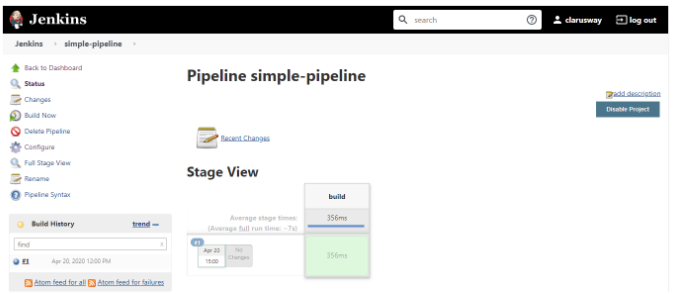
Jenkinsfile (Declarative Pipeline)

```
pipeline {
    agent { label 'master' }
    stages {
        stage('build') {
            steps {
                echo "Clarusway_Way to Reinvent Yourself"
            }
        }
    }
}
```



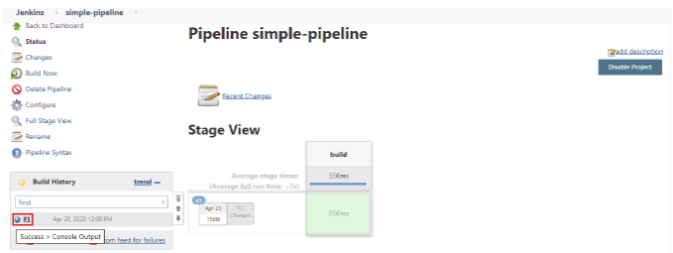
4 Click on the "Build Now" button.

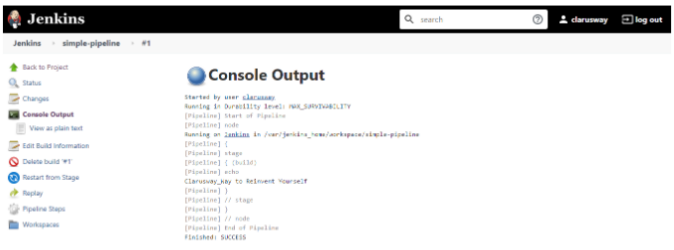




Tips: It will start running the pipeline and within a few seconds, you'll see an indicator of your first job being successful (blue dot on the left side).

5 If you click on that blue dot on the left side it will take you to the "Console Output"





> Inside the `pipeline` there can be `stages` (We have one). Inside the `stages`, there can be several `stage elements`.
>
> Inside each `stage`, there must be `steps`. The steps themselves are Jenkins commands.
>
> `echo` will just print something on the console. It can be useful for displaying values as the pipeline makes progress.

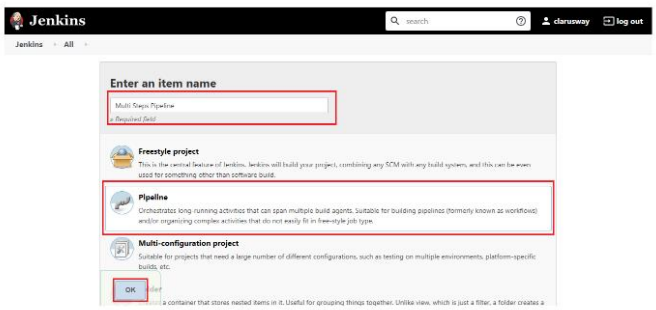
Running Multiple Steps

Pipelines are made up of multiple steps that allow you to build, test and deploy applications. Jenkins Pipeline allows you to compose multiple steps in an easy way that can help you model any sort of automation process.

- Think of a step as a single command that does a single action. When a step succeeds it moves onto the next step. When a step fails to execute correctly the Pipeline will fail.
- When all the steps in the Pipeline have successfully completed, the Pipeline is considered successfully executed.

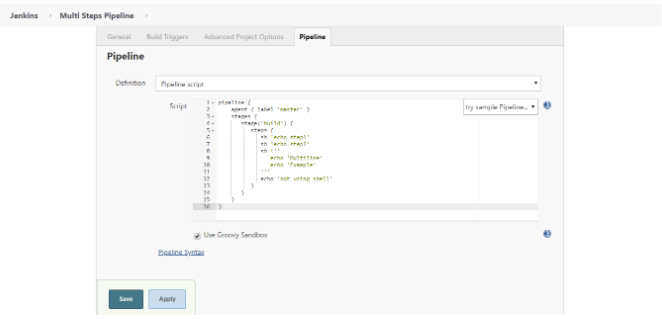
> Example

1

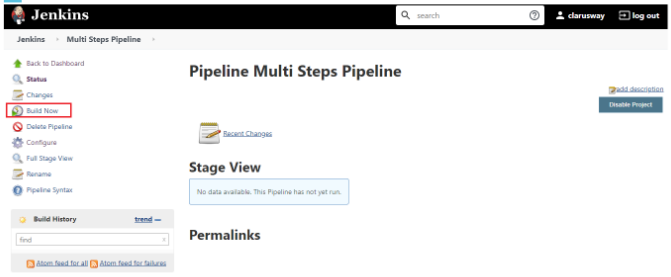



```
pipeline {
    agent { label 'master' }
    stages {
        stage('build') {
            steps {
                sh 'echo step1'
                sh 'echo step2'
                sh '''
                    echo 'Multiline'
                    echo 'Example'
                '''
                echo 'not using shell'
            }
        }
    }
}
```

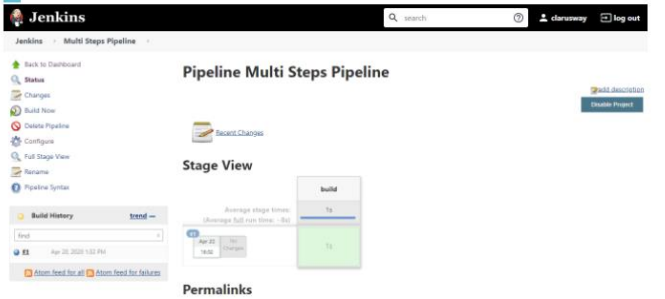
2



3



4



5

Complementary Lesson about Creating Your First Pipeline