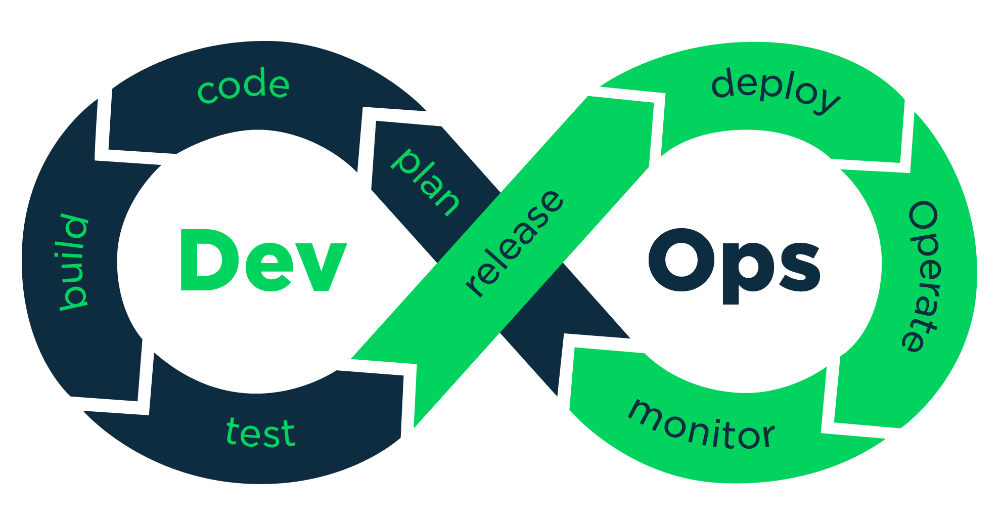
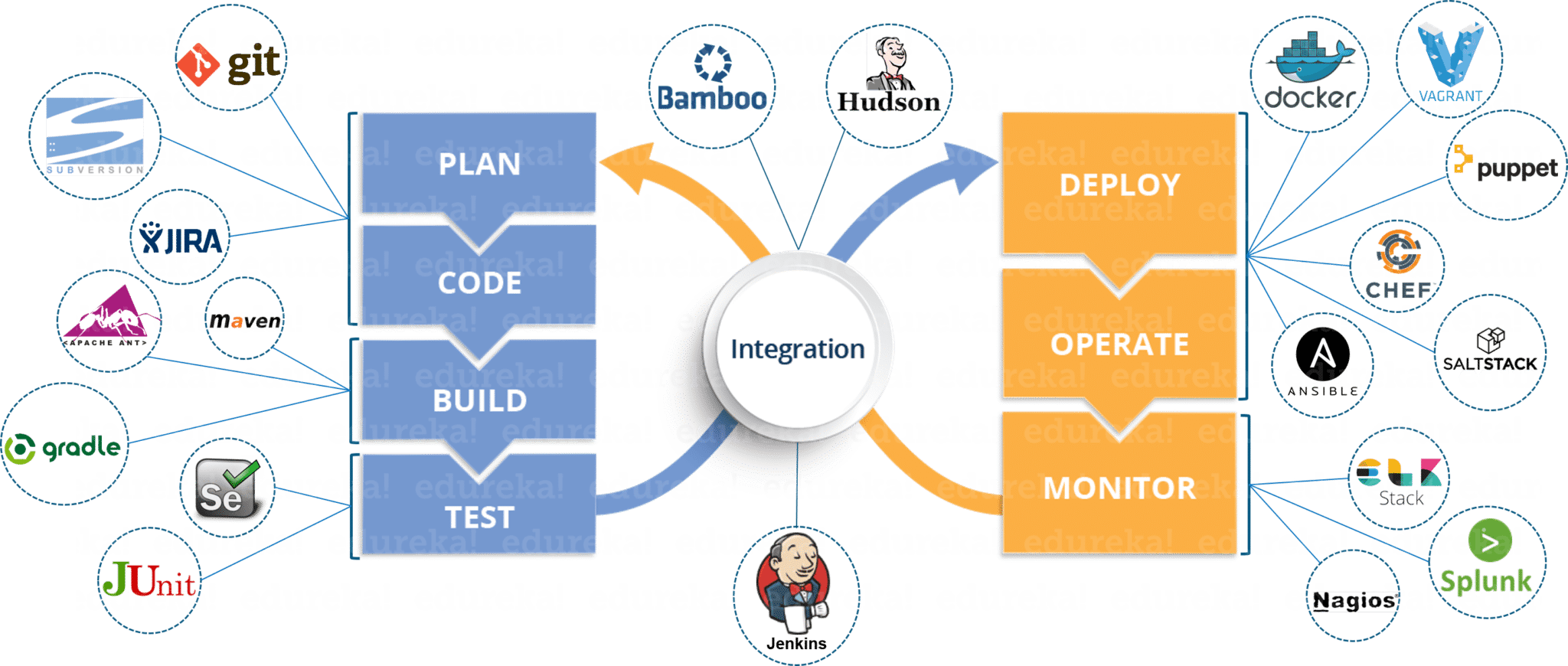
**Declarative and Scripted pipeline job in Jenkins -PART 1**

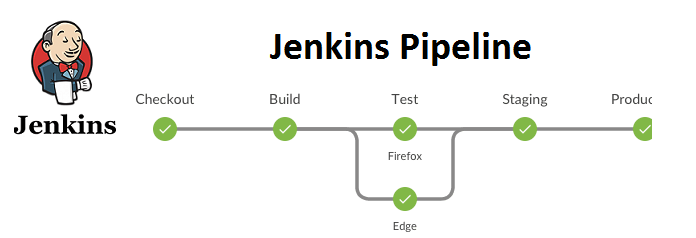
**What is Pipeline in Devops?**

* A pipeline is a collection of jobs that brings the software from version control into the hands of the end users by using automation tools.
* It is a feature used to incorporate continuous delivery in our software development workflow.
* There are various pipeline tools in the market and Jenkins is one among them.
* Example: Bamboo, Hudson etc.



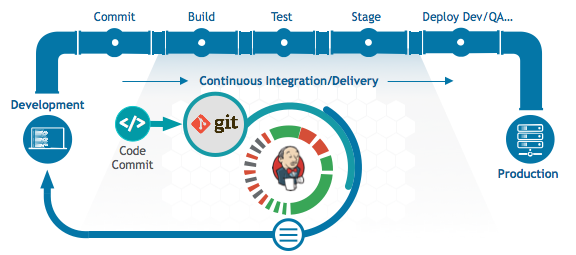


**What is a Jenkins pipeline?**

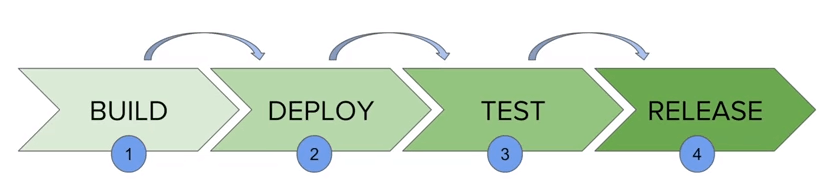


* They represent multiple Jenkins jobs as one whole workflow in the form of a pipeline.
* What do these pipelines do? These pipelines are a collection of Jenkins jobs which trigger each other in a specified sequence.

**Jenkins Pipeline – Core element of Continuous Integration/Continuous Delivery (CICD)**



**How it works?**



* A Jenkins pipeline, every job has some sort of dependency – with at least with one or more events
* These pipelines are a core element of a CI/CD (Continuous Integration/Continuous Delivery) system
* These pipelines enable development teams to automate the building, testing, and deployment of their applications.

**Types of Pipeline jobs:**

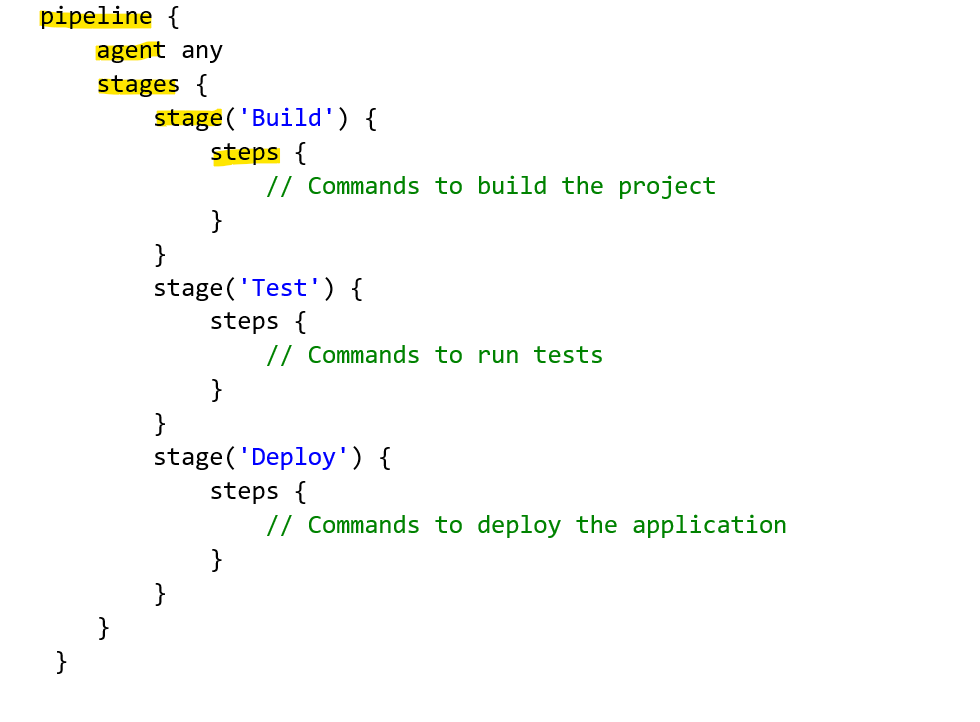
There are two types of syntax used for defining your Jenkins pipeline jobs

1. **Declarative**
2. **Scripted**

**Declarative pipeline - Syntax**

**Declarative:**

The declarative pipeline provides a more structured and simpler syntax for defining pipelines. It is designed to be easy to read and understand, especially for those new to Jenkins and CI/CD concepts.



|  |  |
| --- | --- |
|  |  |

**Scripted pipeline - Syntax**

**Scripted:**

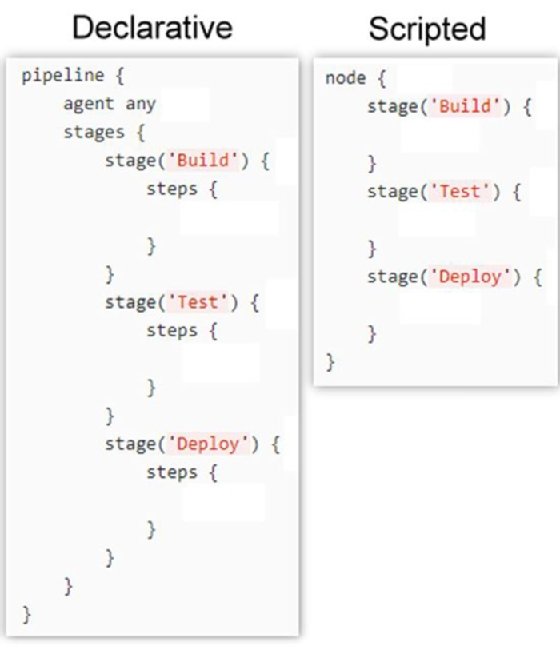
The scripted pipeline uses a more flexible and complex scripting syntax, allowing for advanced customization and conditional logic. It provides more control and is suitable for complex build processes.

|  |  |
| --- | --- |
|  |  |

**Pipeline Syntax differences –Declarative vs Scripted**

**Pipeline syntax differences**

* Declarative pipelines always begin with the word pipeline.
* Scripted pipelines, on the other hand, always begin with the word node.
* Declarative pipelines break down stages into individual stages that can contain multiple steps.
* Scripted pipelines use Groovy code and references to the Jenkins pipeline DSL within the stage elements without the need for steps.
* These are the key differences that allow a developer to quickly differentiate between a scripted pipeline and a declarative pipeline.



**Jenkins Pipeline Job( code) Concepts**

##### **Pipeline concepts**

**Pipeline**

* This is a user defined block which contains all the processes such as build, test, deploy, etc.
* It is a collection of all the stages in a Jenkinsfile.
* All the stages and steps are defined within this block.
* It is the key block for a declarative pipeline syntax.

**Node**

* A node is a machine that executes an entire workflow.
* It is a key part of the scripted pipeline syntax.
* There are various mandatory sections which are common to both the declarative and scripted pipelines, such as stages, agent and steps that must be defined within the pipeline.

**Agent**

* An agent is a directive that can run multiple builds with only one instance of Jenkins.
* This feature helps to distribute the workload to different agents and execute several projects within a single Jenkins instance.
* It instructs Jenkins to allocate an executor for the builds.
* A single agent can be specified for an entire pipeline or specific agents can be allotted to execute each stage within a pipeline. Few of the parameters used with agents are:

**Any**

Runs the pipeline/ stage on any available agent.

**None**

This parameter is applied at the root of the pipeline and it indicates that there is no global agent for the entire pipeline

And each stage must specify its own agent.

**Label**

Executes the pipeline/stage on the labelled agent.

**Docker**

This parameter uses docker container as an execution environment for the pipeline or a specific stage.

**Stages**

* This block contains all the work that needs to be carried out.
* The work is specified in the form of stages.
* There can be more than one stage within this directive.
* Each stage performs a specific task.

**Steps**

* A series of steps can be defined within a stage block.
* These steps are carried out in sequence to execute a stage.
* There must be at least one step within a steps directive.

**Stage Section in Jenkins Pipeline Build**

The “stage” section is required to segregate the work category as listed inline:





A specified pipeline will consist of several steps that can be grouped in several stages. For example:

|  |  |  |
| --- | --- | --- |
| Stage 1 | Pull code from repository | Pull Code CheckIn |
| Stage 2 | Build your project and artifacts | Build Project/Artifacts of Project |
| Stage 3 | Deploy your application | Deploy from centralized repo to specified environment |
| Stage 4 | Perform functional tests | QA performance will happen |
| Stage 5 | Perform performance tests | ·       (UI and performance can test. |

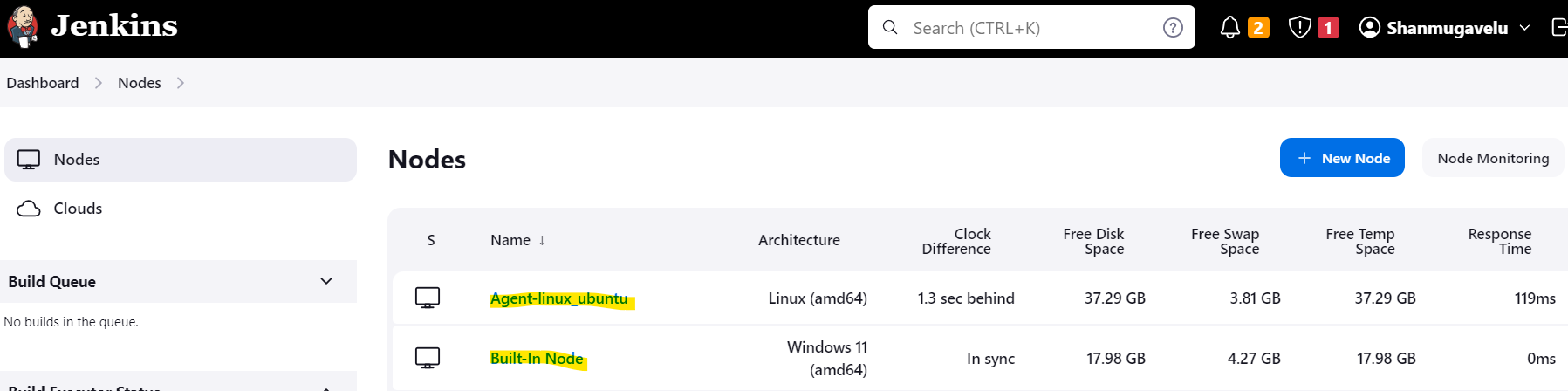
**Where to write or place/run ?the pipeline code ?**

We can define or write pipeline code (Scripted and Declarative) either in

1. In the web UI (Jenkins UI)
2. In Jenkins file (in Repo)

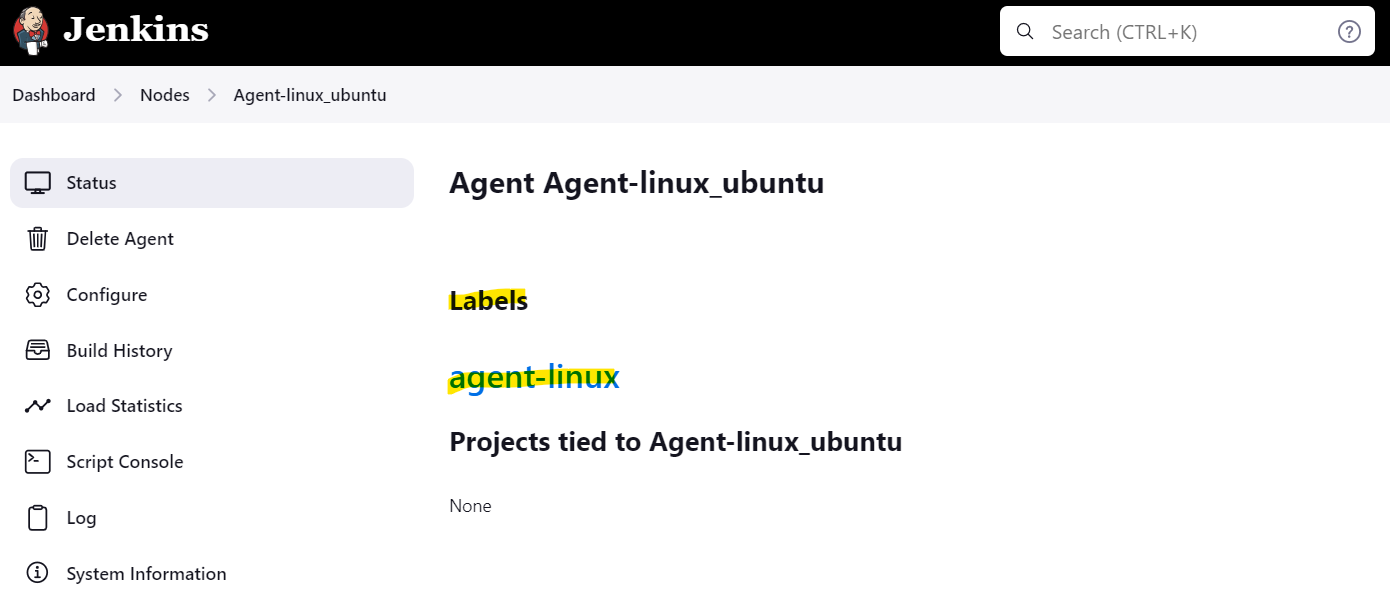
**Creating a Declarative Pipeline job (code in Jenkins web UI):**

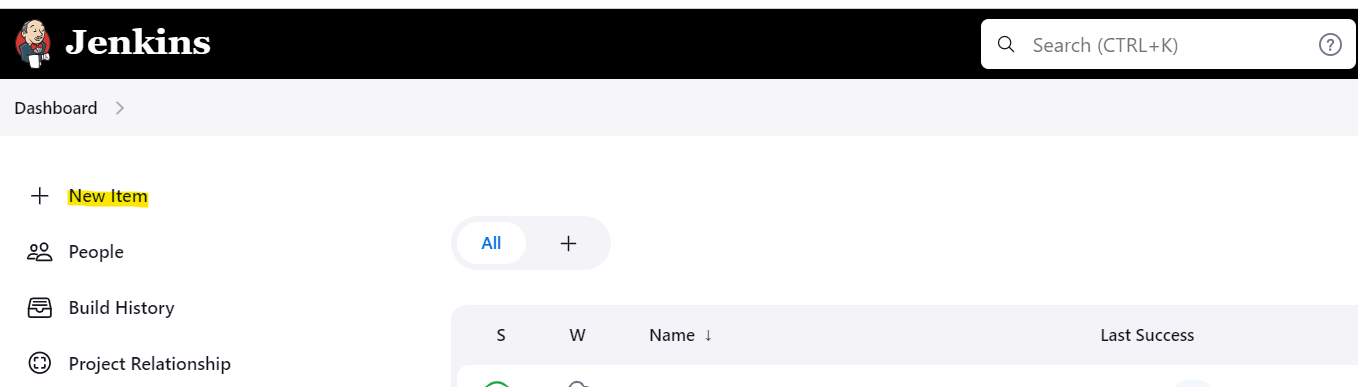
**Checking the Master and Slave node availability**

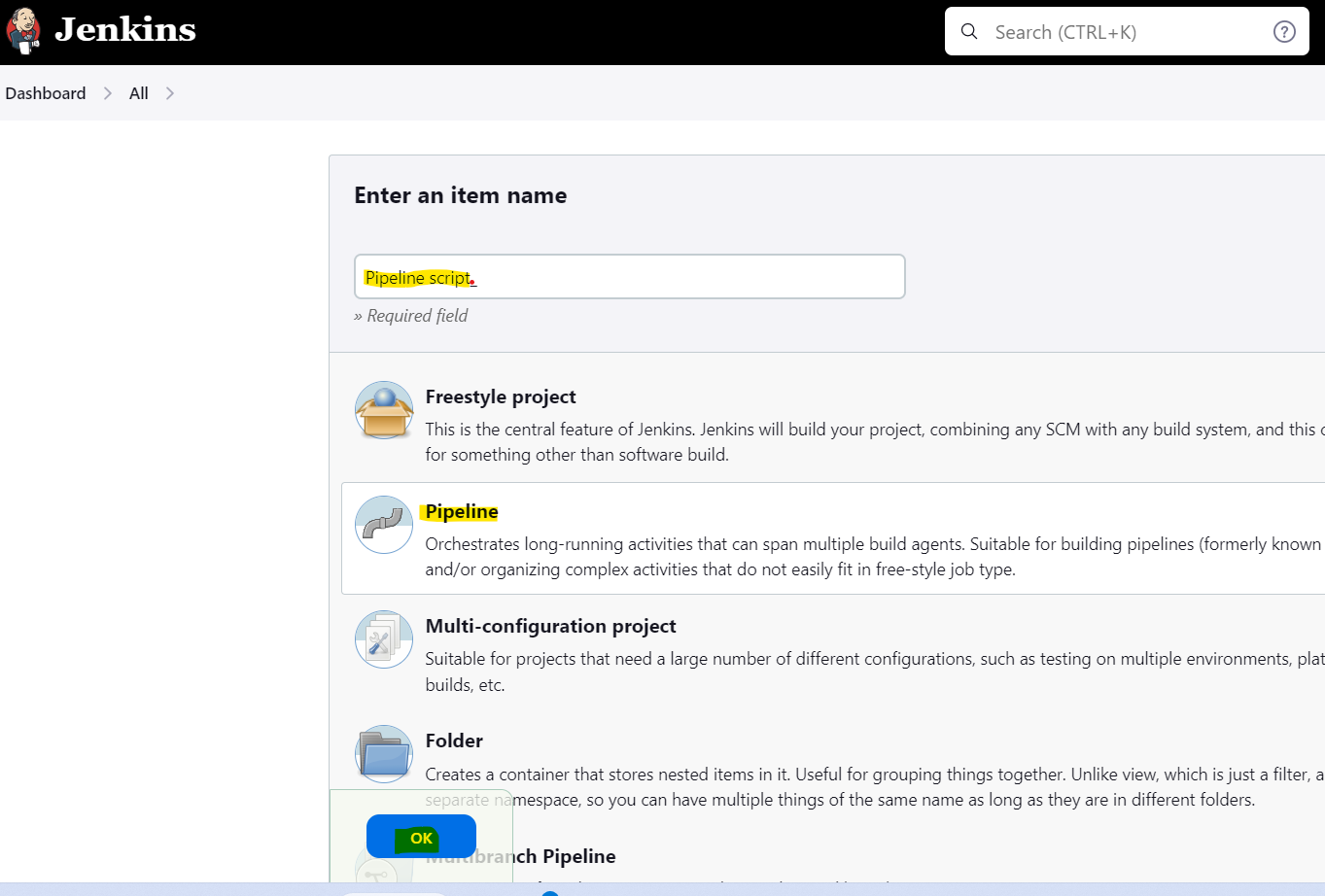


1. **Create the pipeline job and run from the Slave node (Agent node)**

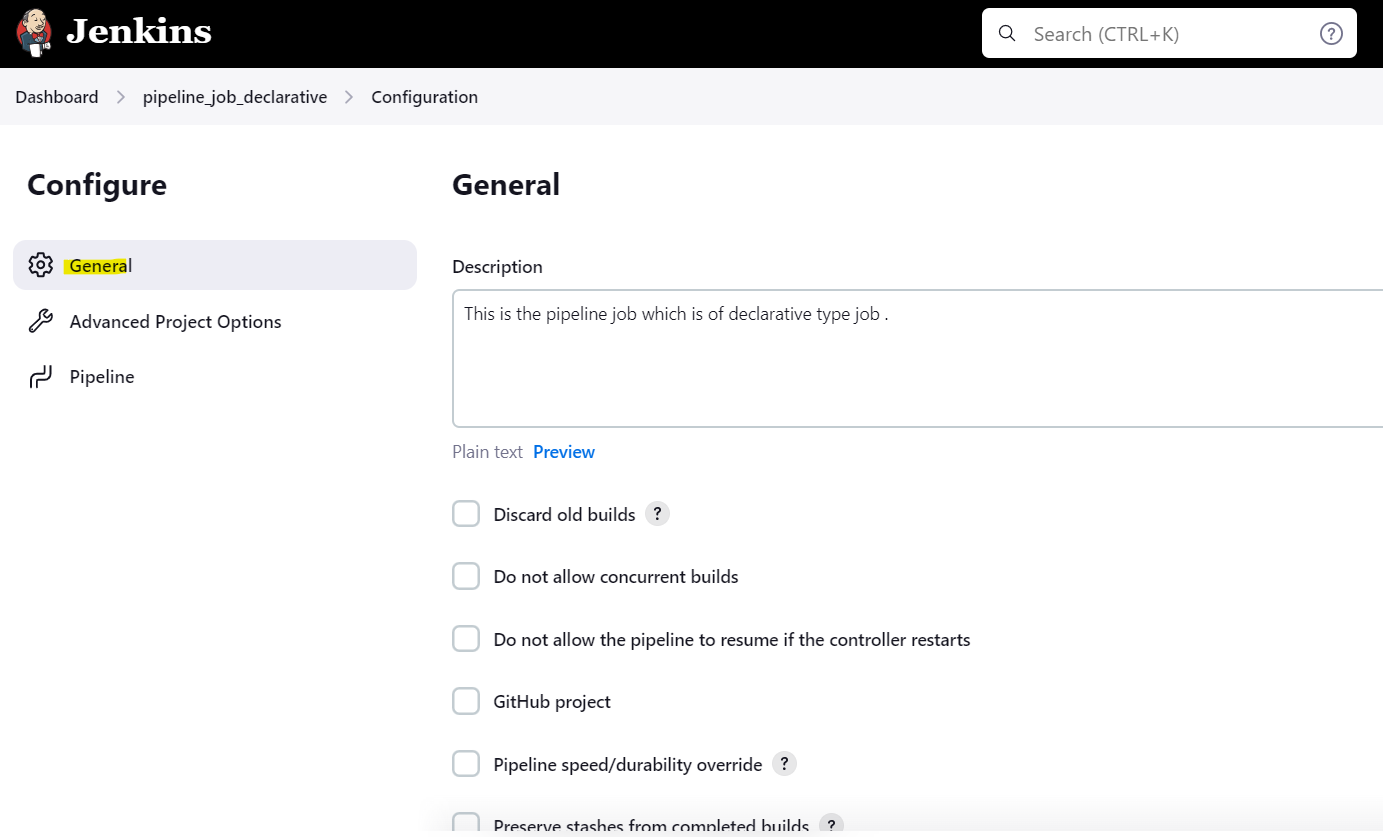
Check the Label for the Agent node

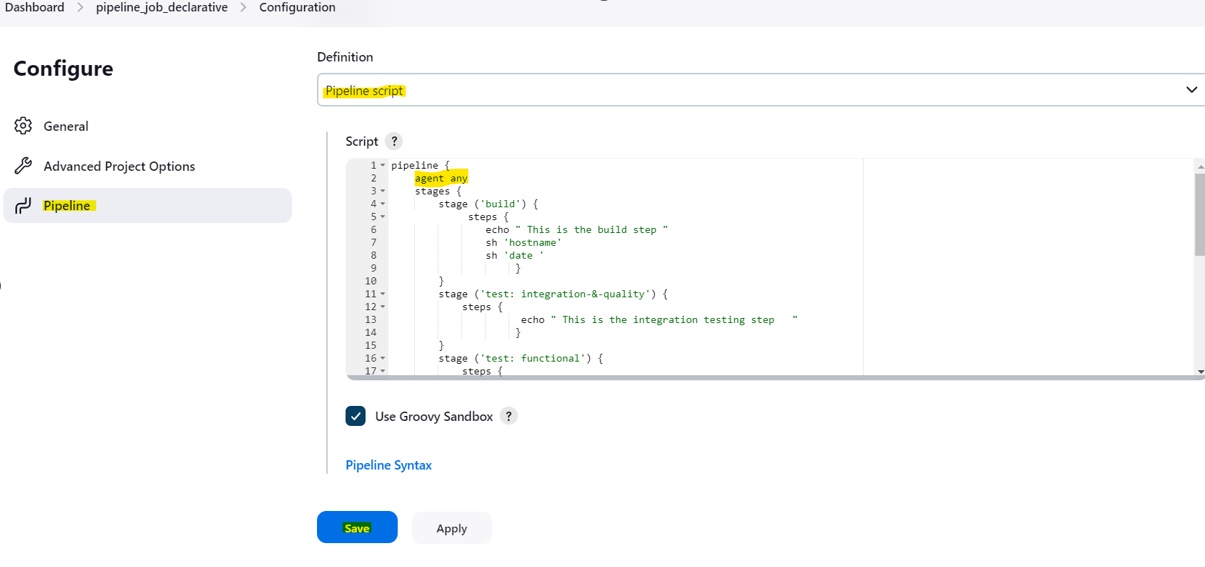




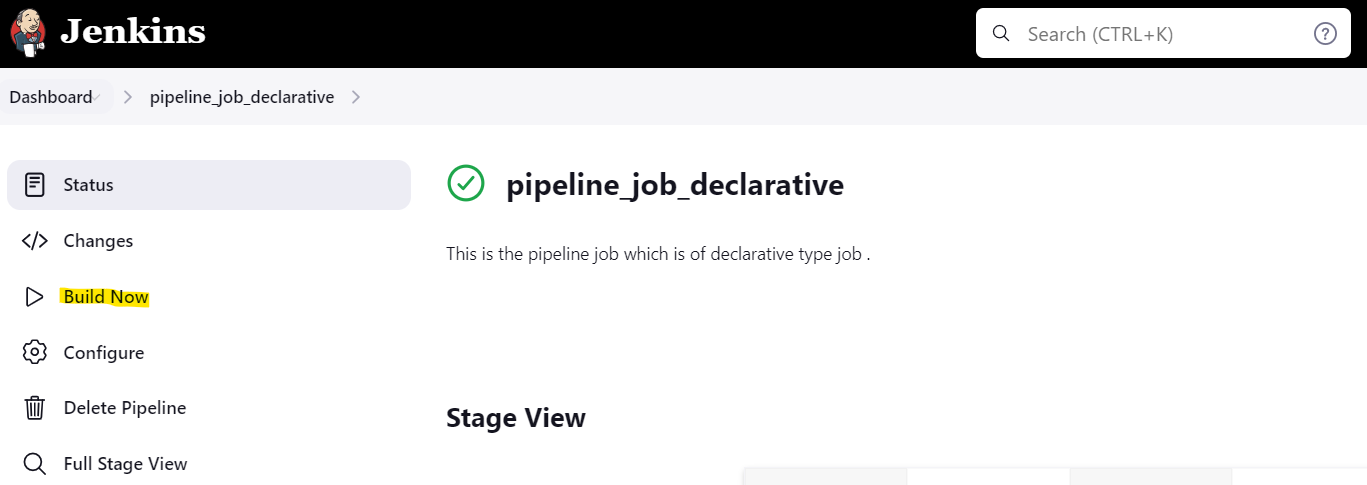


Configure the Jenkins job for declarative pipeline ->

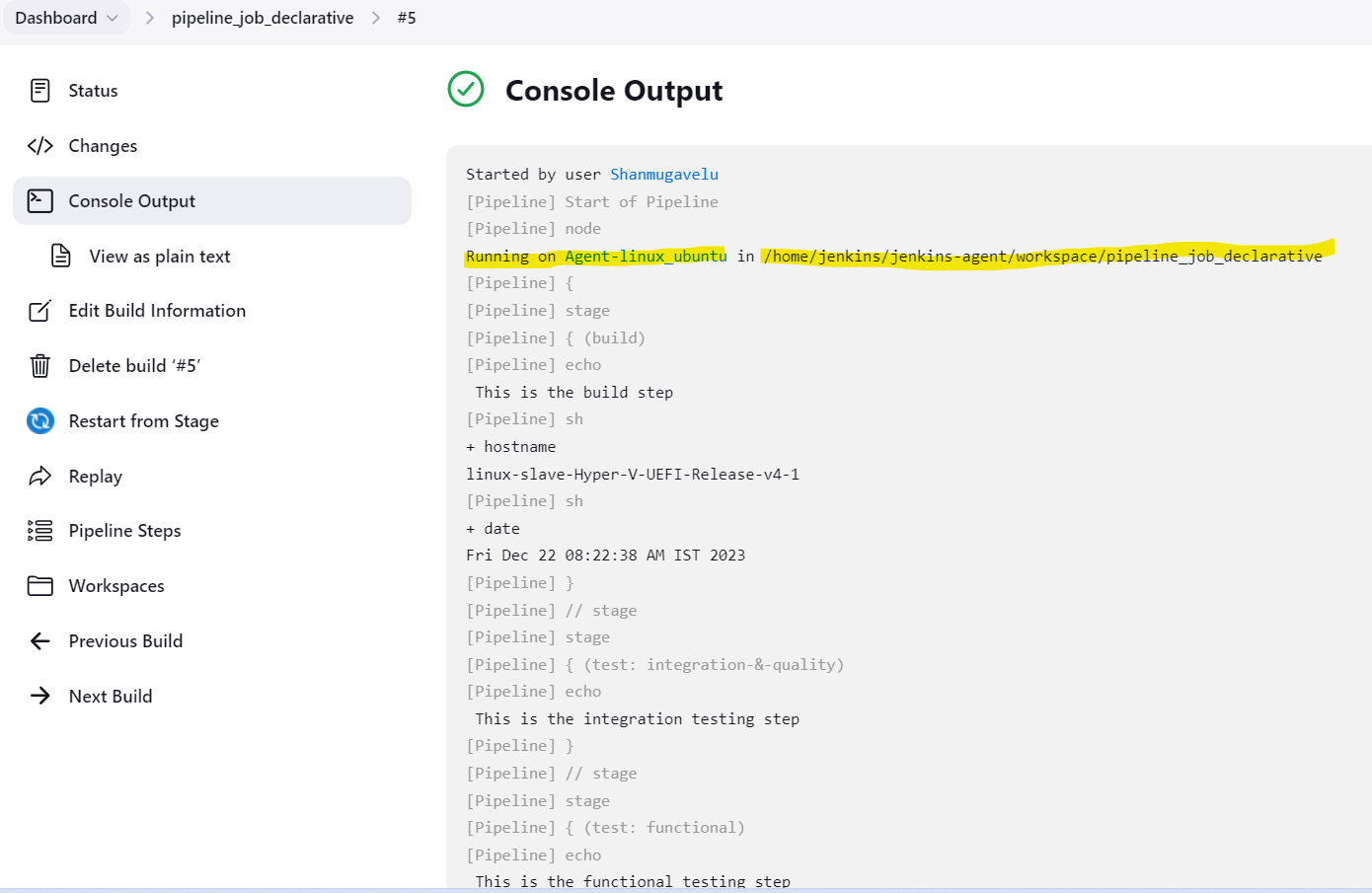




Build the Job **->**

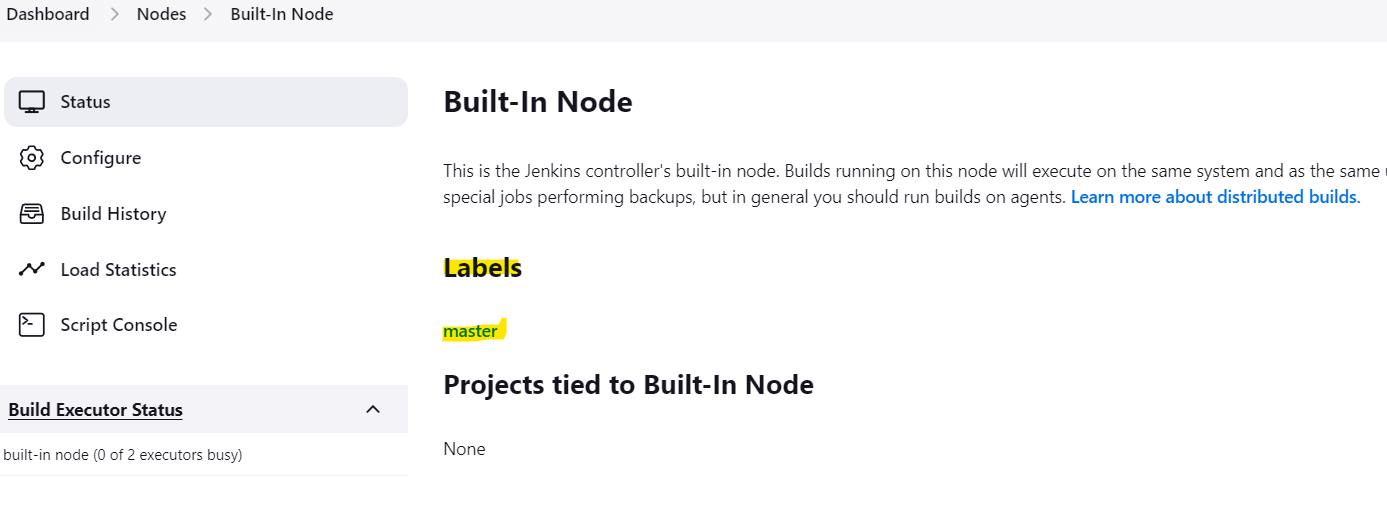


Check the Job output**->**



1. **Create the declarative pipeline and run from the Master (Built-in node)**

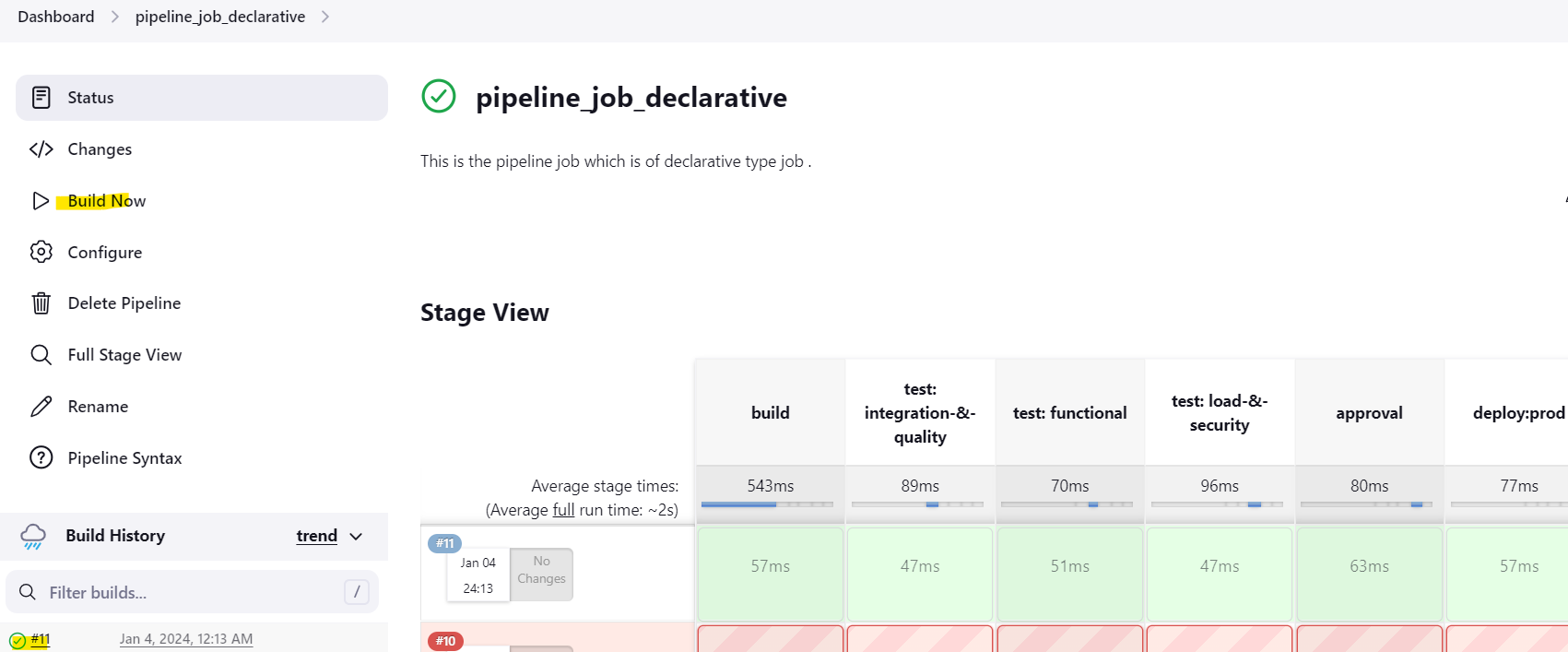
Check the Label for the master node

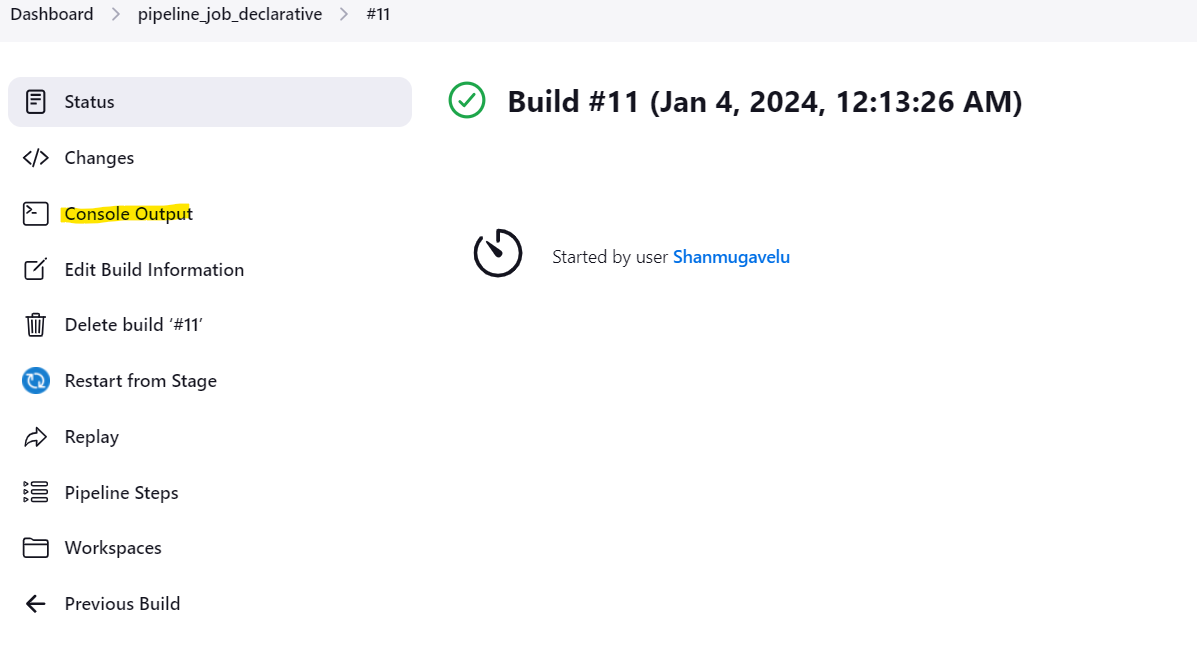


Configure the Jenkins job for declarative pipeline ->

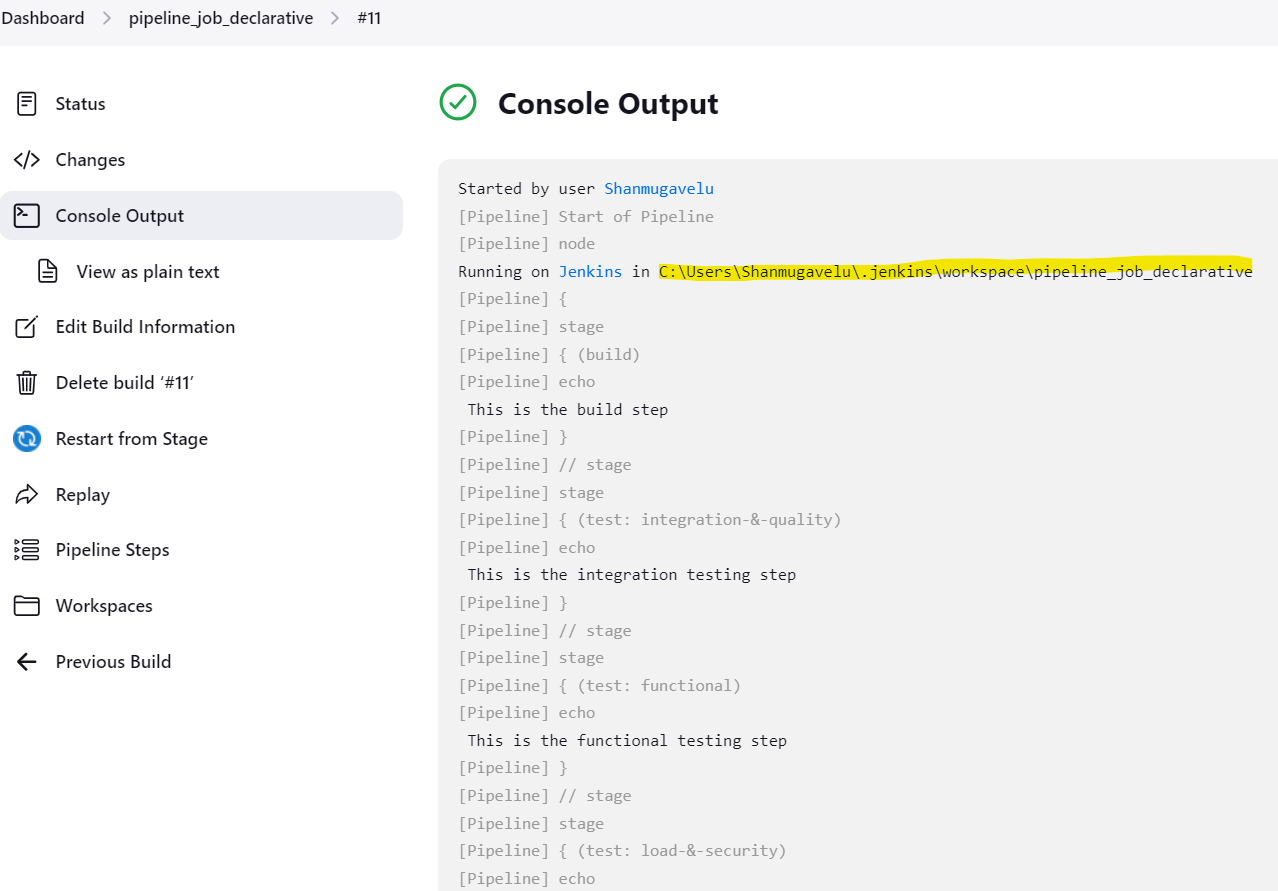


Build the Job **->**





Check the Job output**->**



## Written by Vinod Sairam

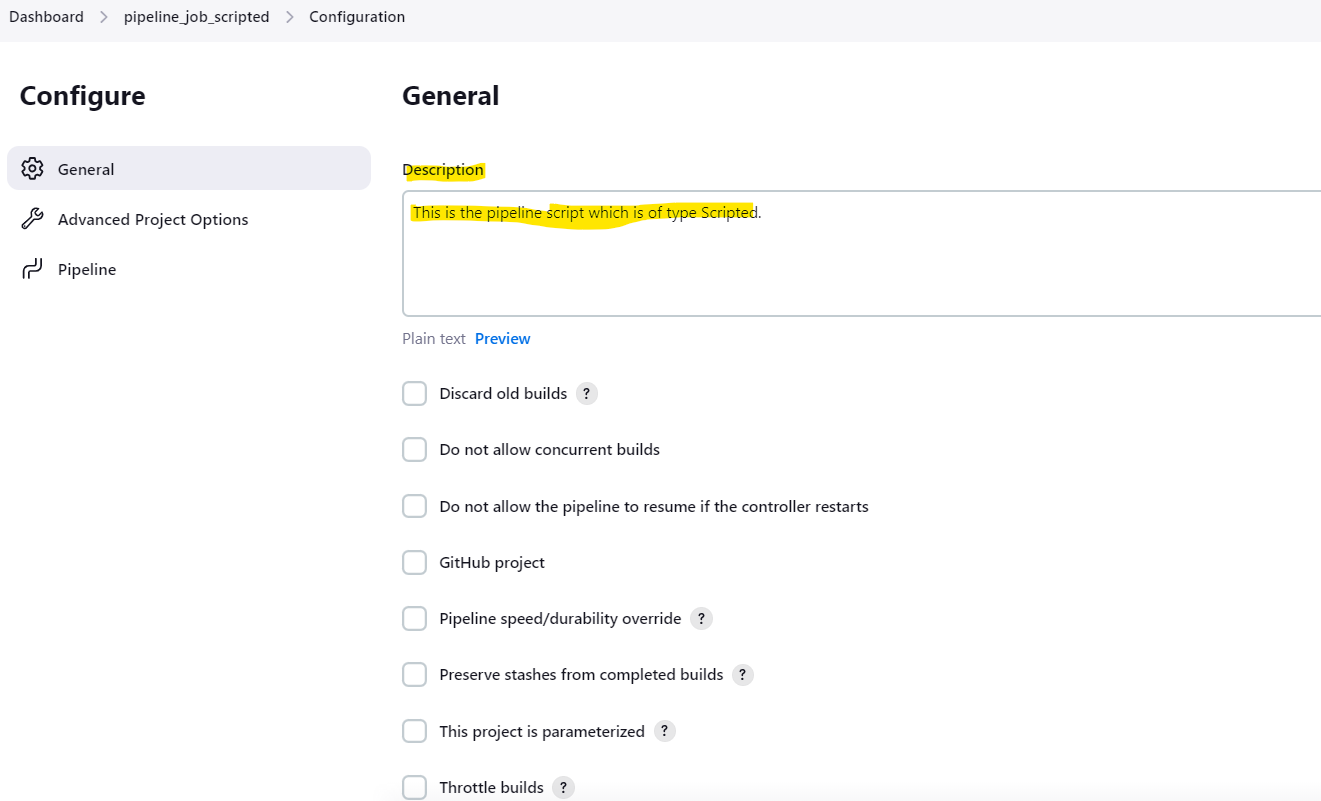
The Database Administrator who helps to manage Enterprise Databases who develops Devops Engineering ideas to support Database tasks, Infrastructure tasks and to reach the right audience.

**Assisted by Shanmugavelu. M**

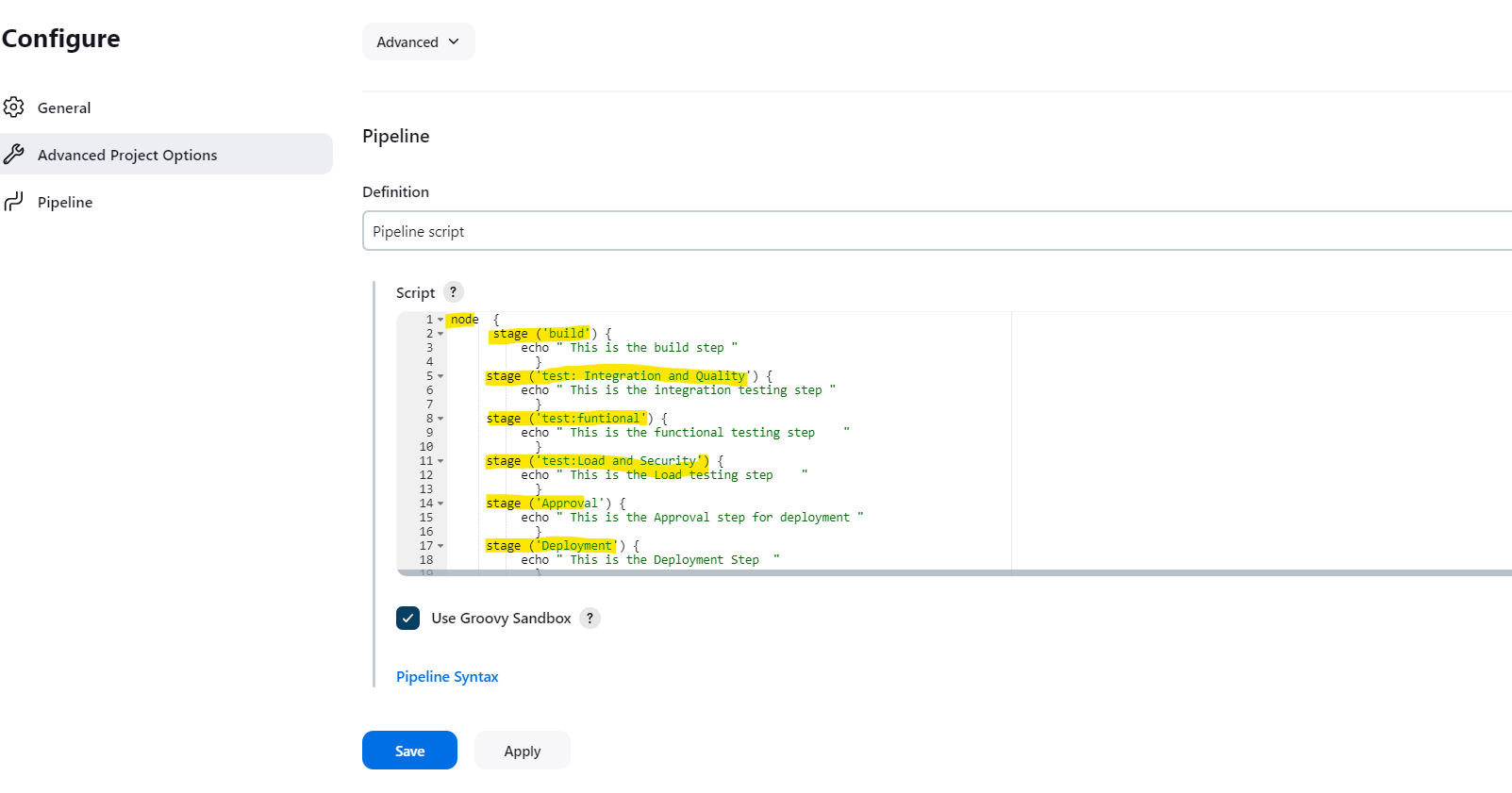
**Declarative and Scripted pipeline job in Jenkins -PART 2**

**Creating a Scripted Pipeline job (code in Jenkins web UI):**

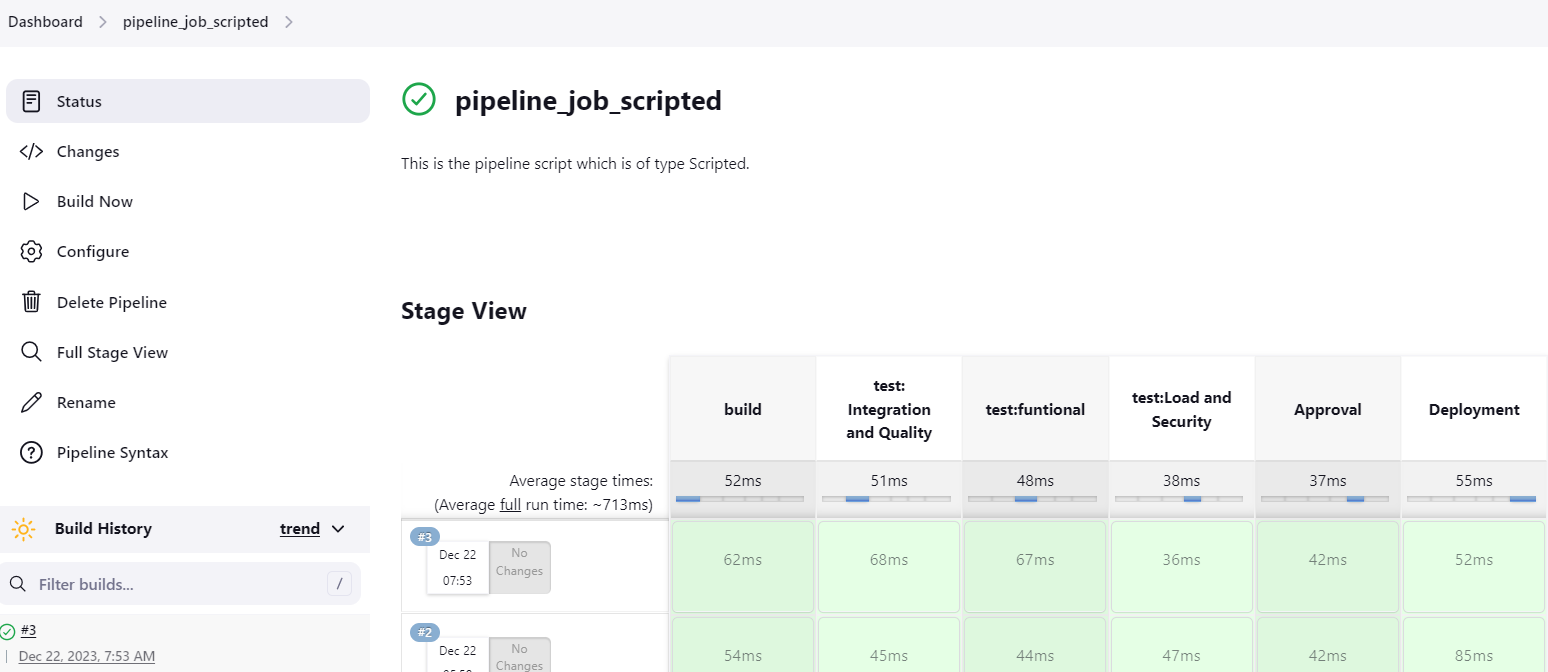
Configure the Jenkins job for scripted pipeline ->

****

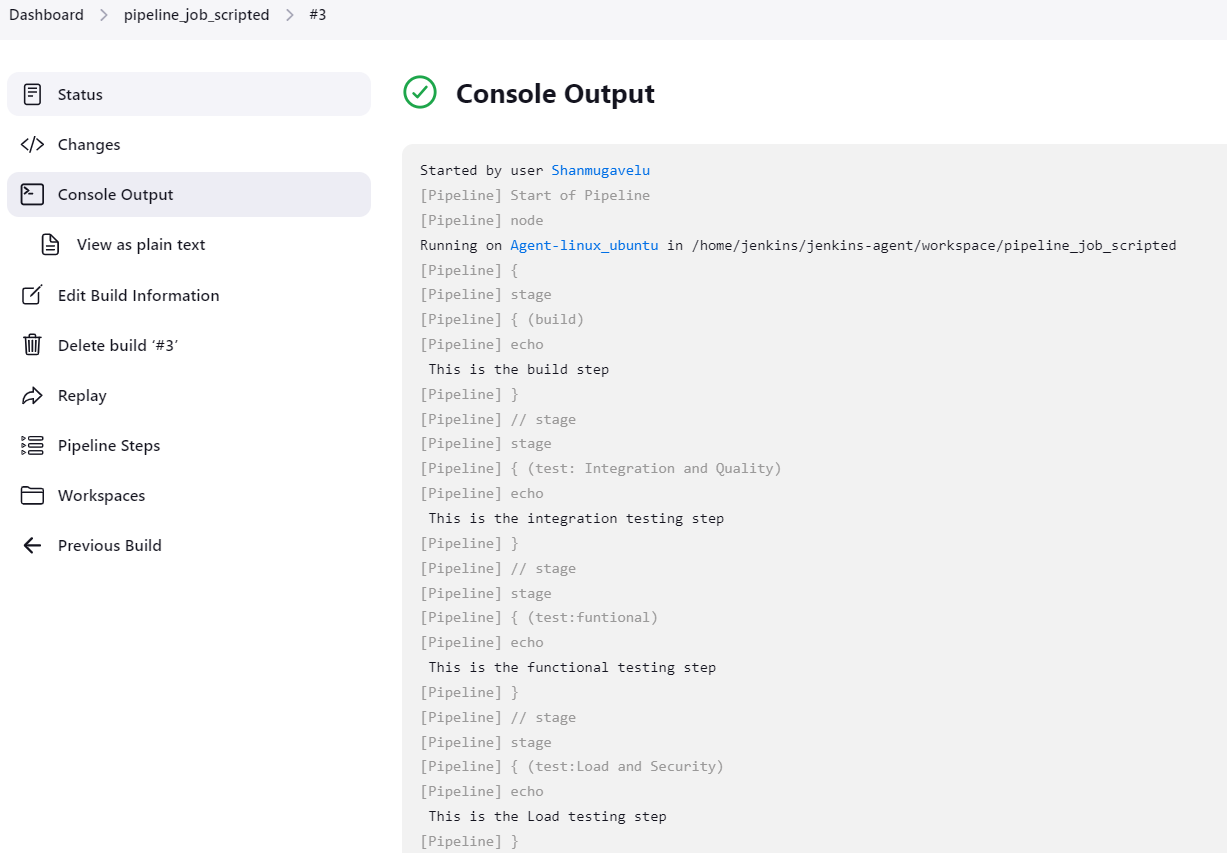
Write the code in the Jenkins Web UI ->

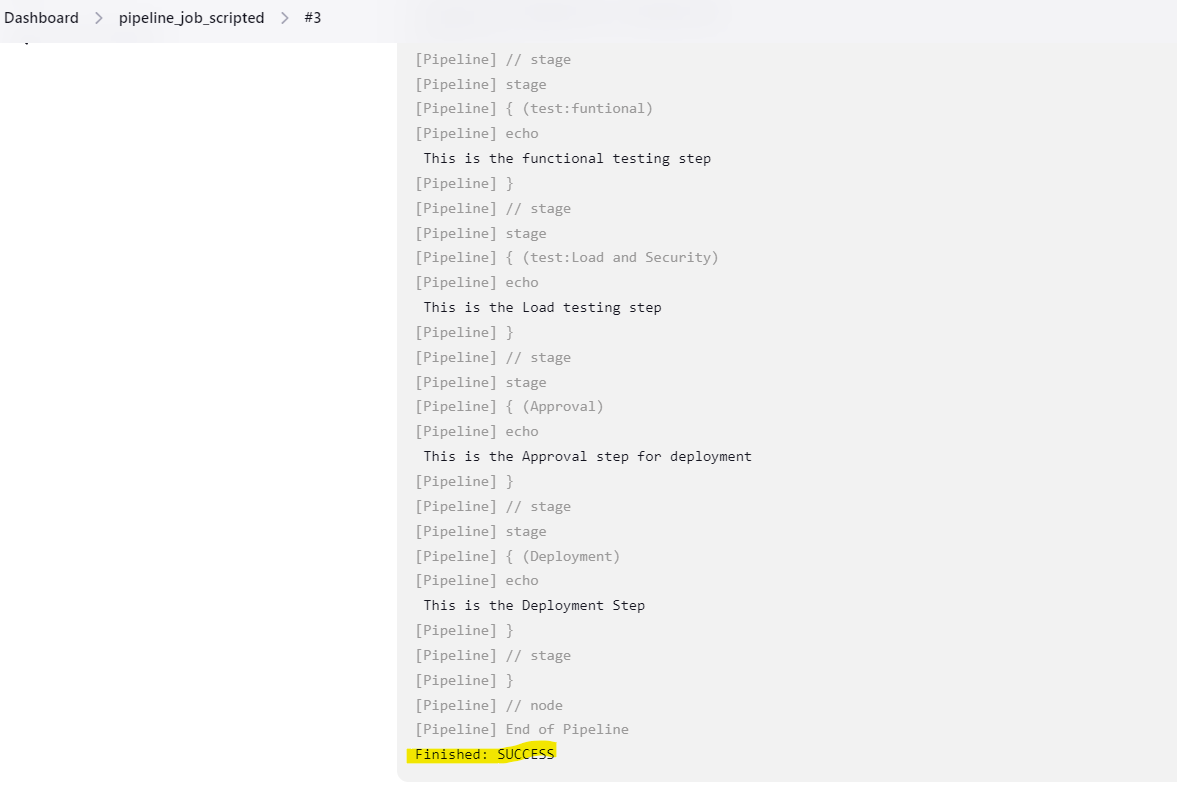


Build the Job **->**



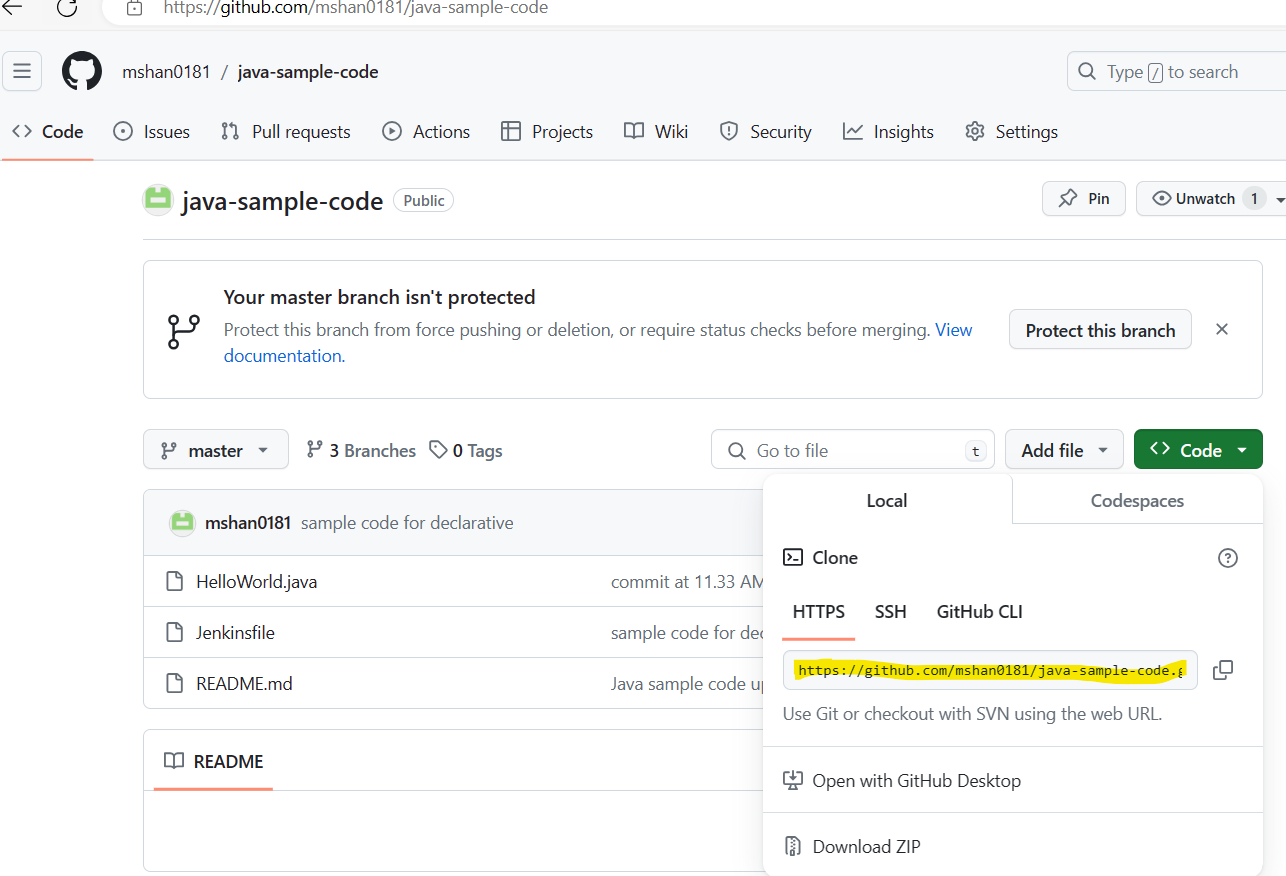
Check the Job output**->**



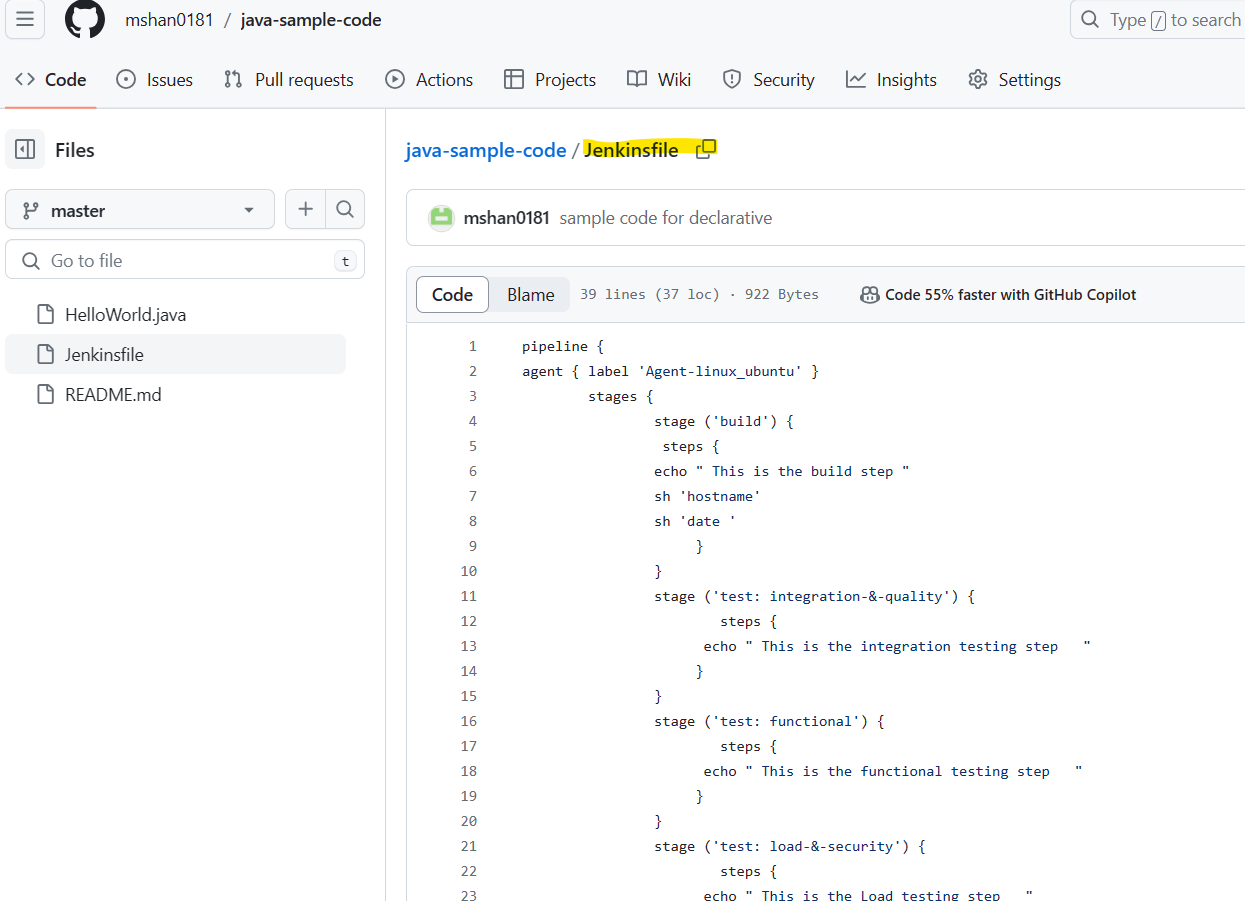


**Creating a Pipeline job using Jenkins file –(code using Jenkins file in Repo):**

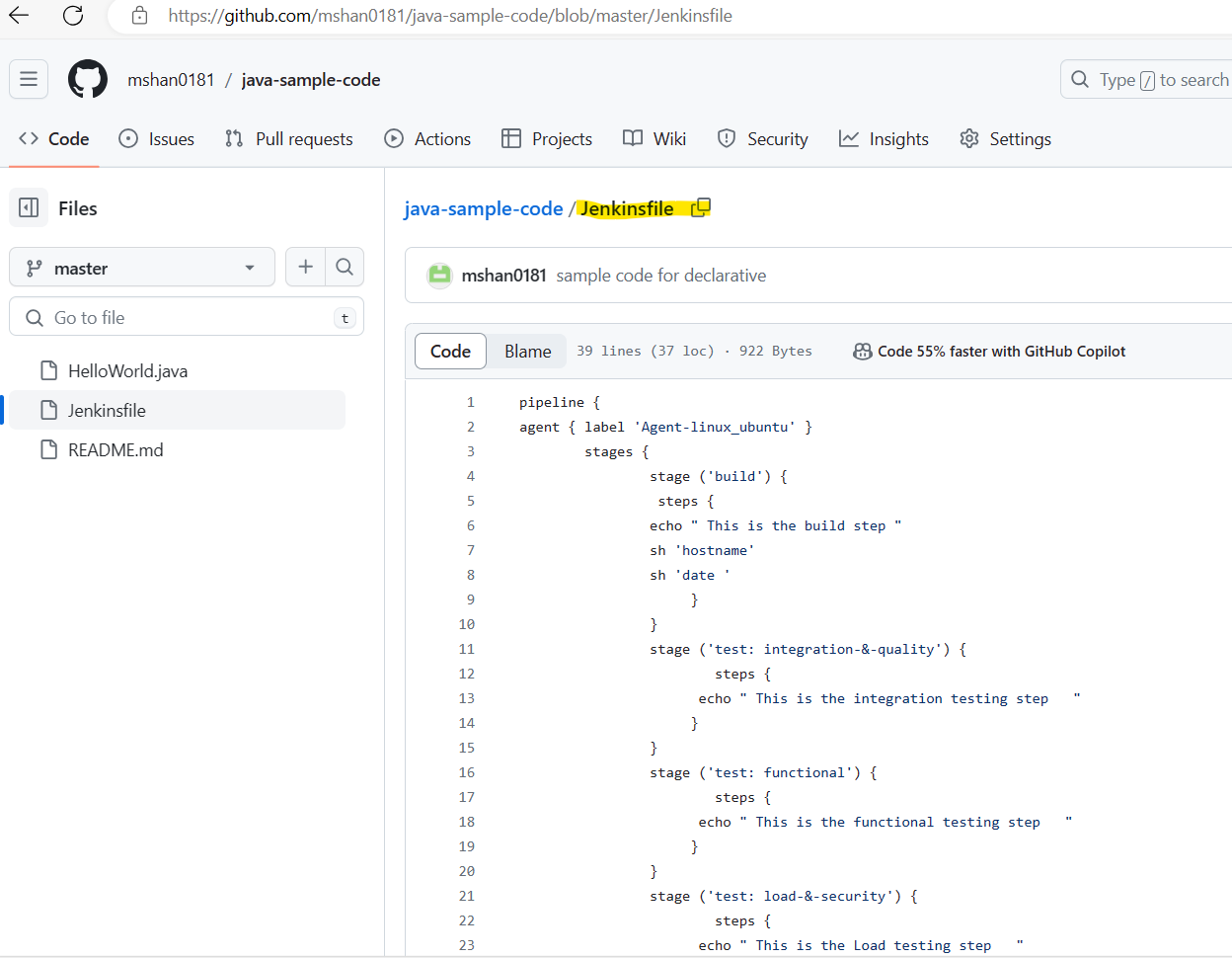
Copy the repository URL GITHUB ->



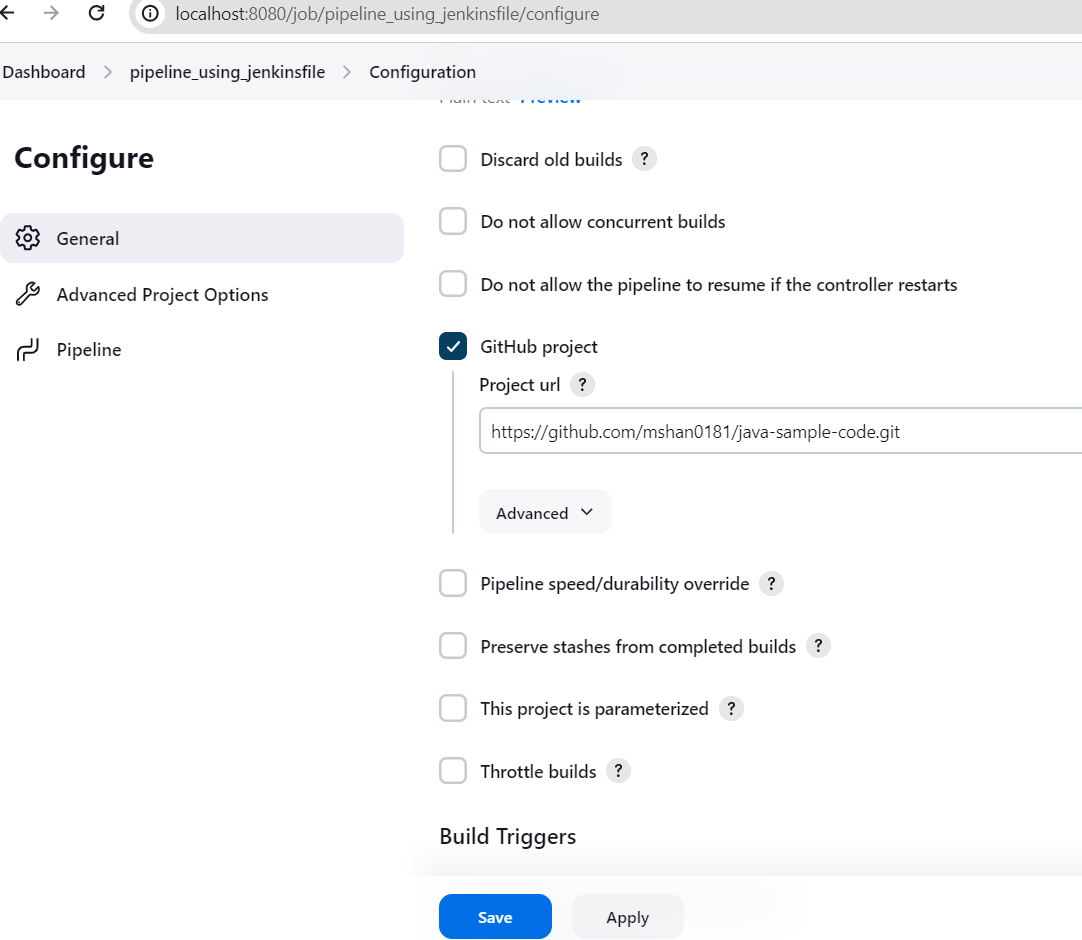
Create the Jenkins file ( with code) in GITHUB Repo ->

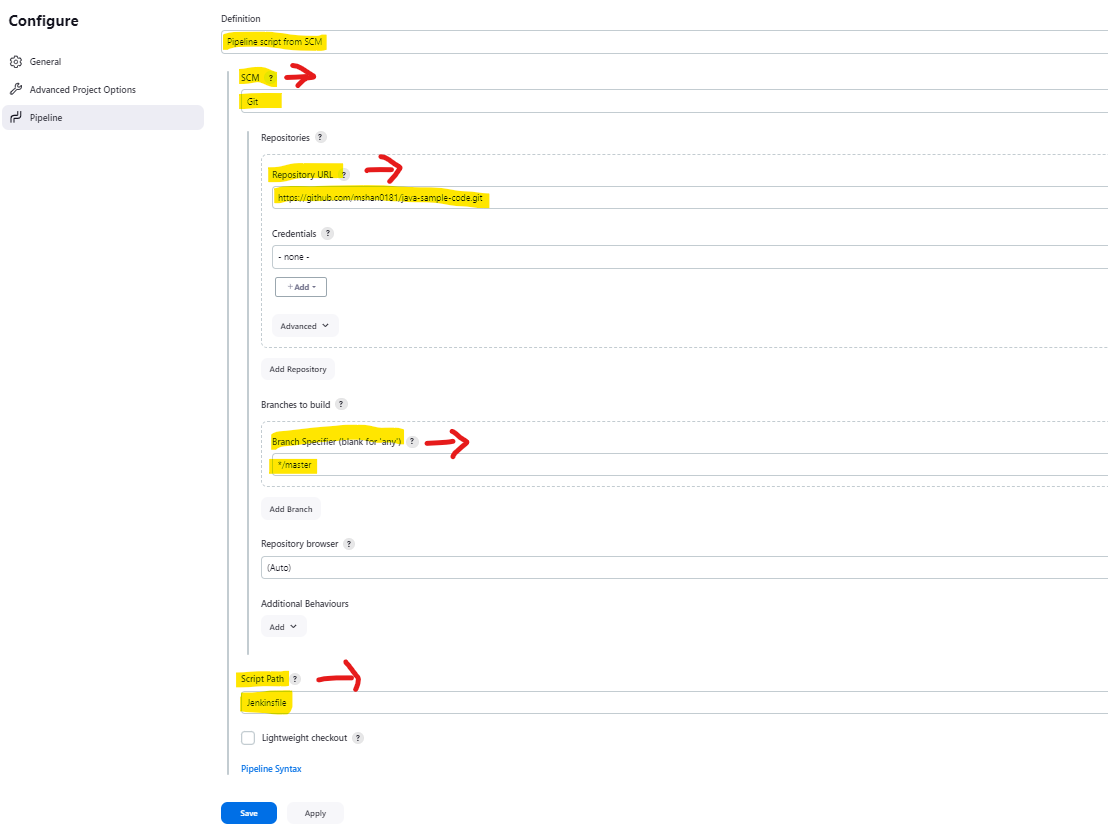


Copy the Jenkins file URL from GITHUB->

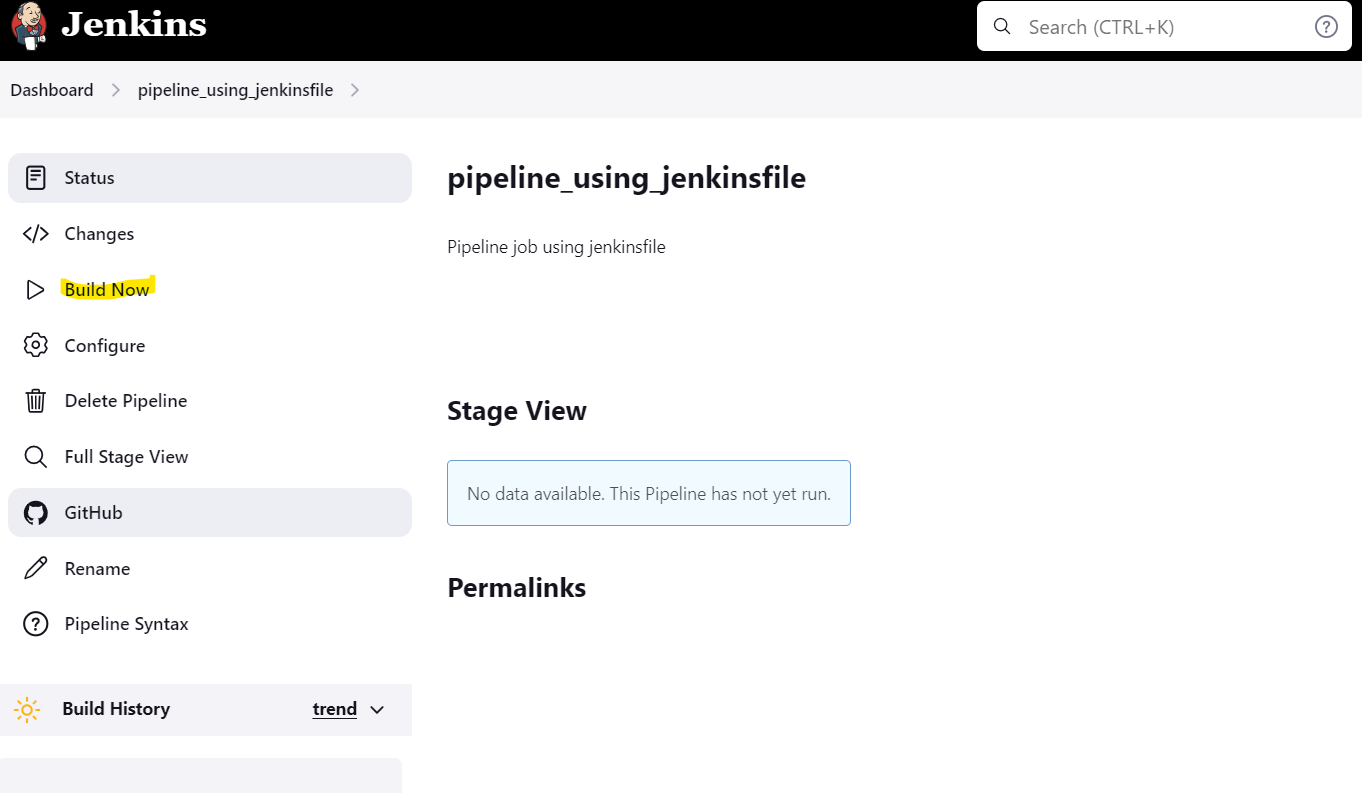


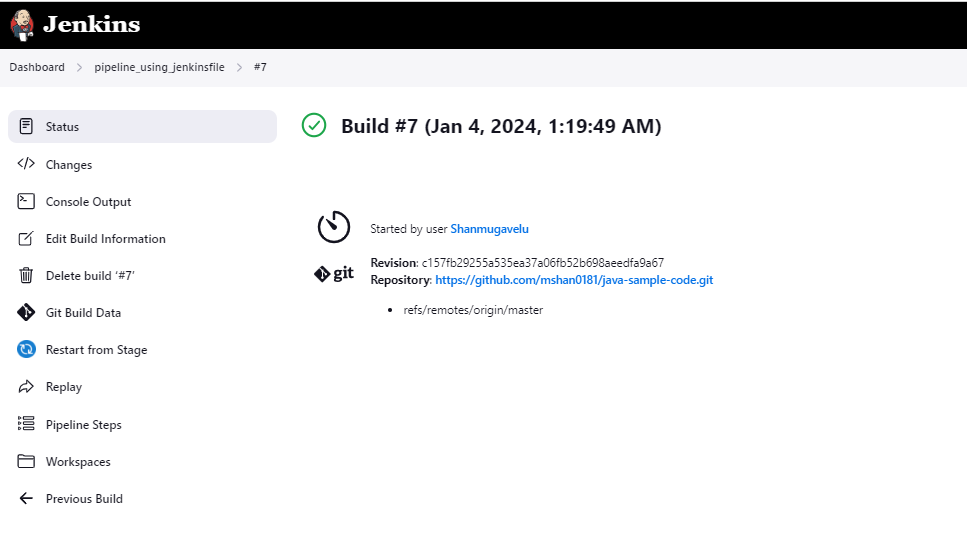
Configure the Jenkins job with Jenkinsfile option ->

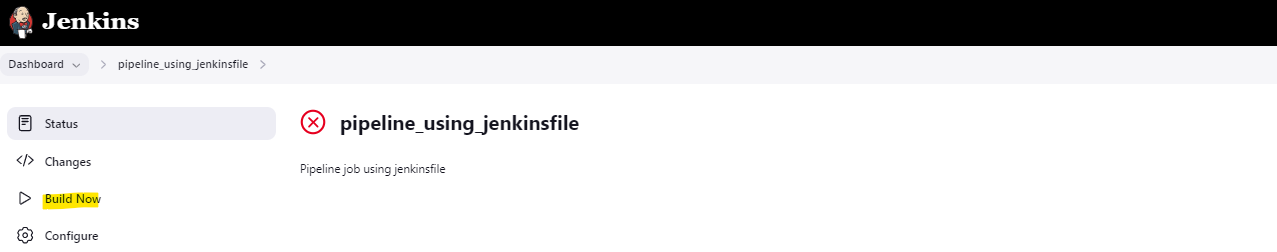


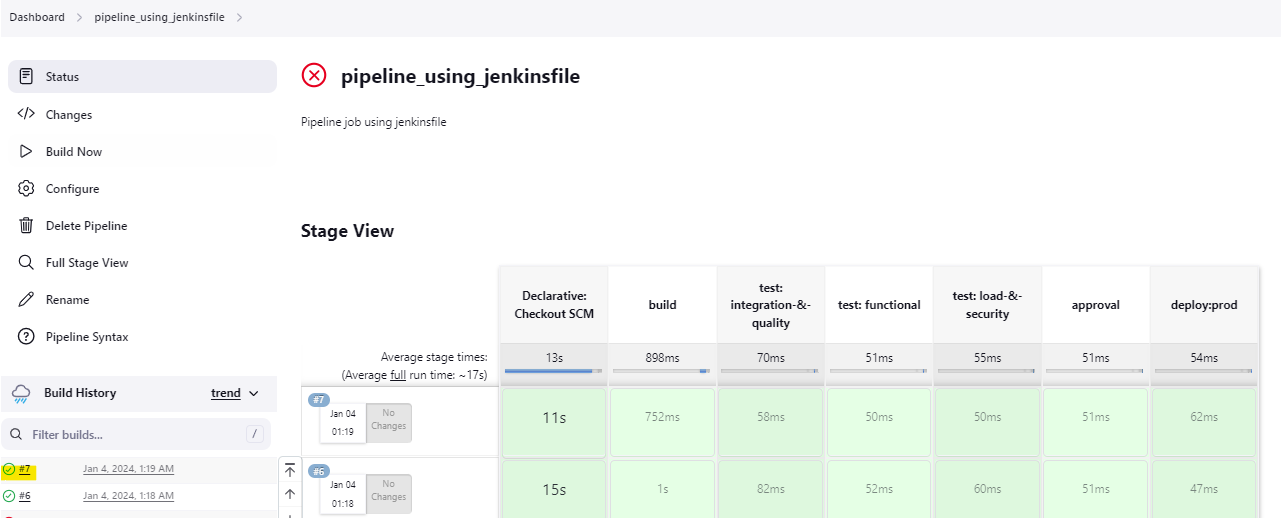


Build the Jenkins job ->

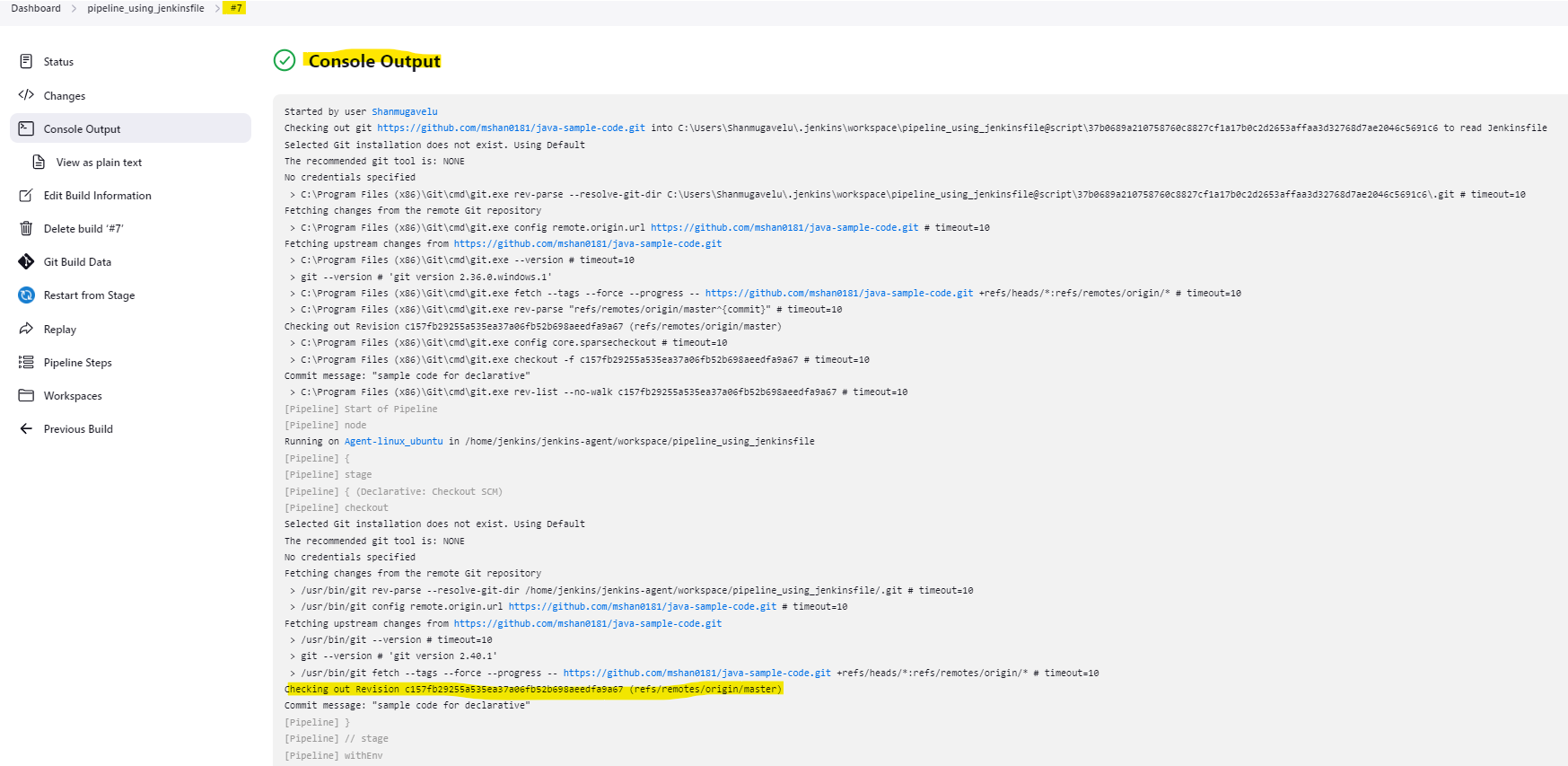


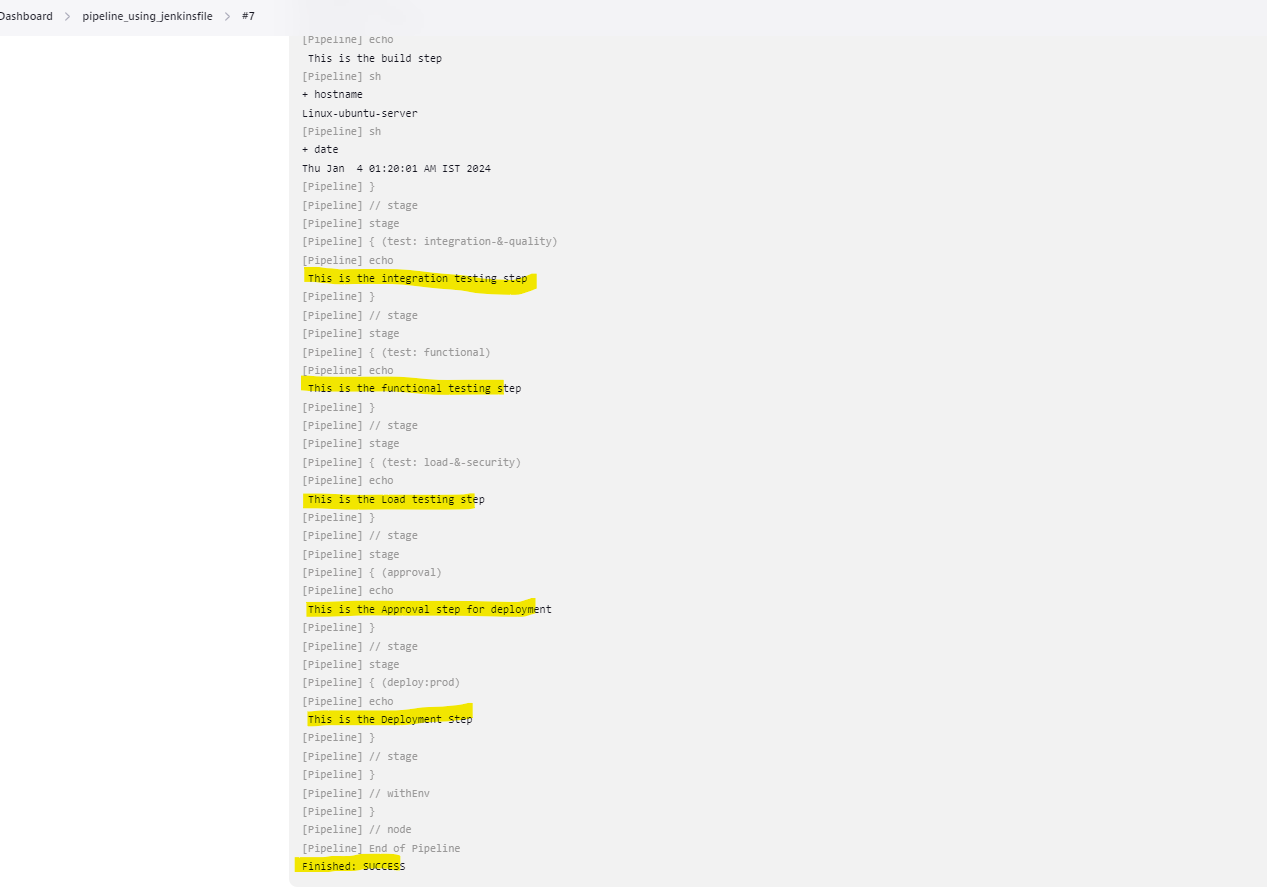




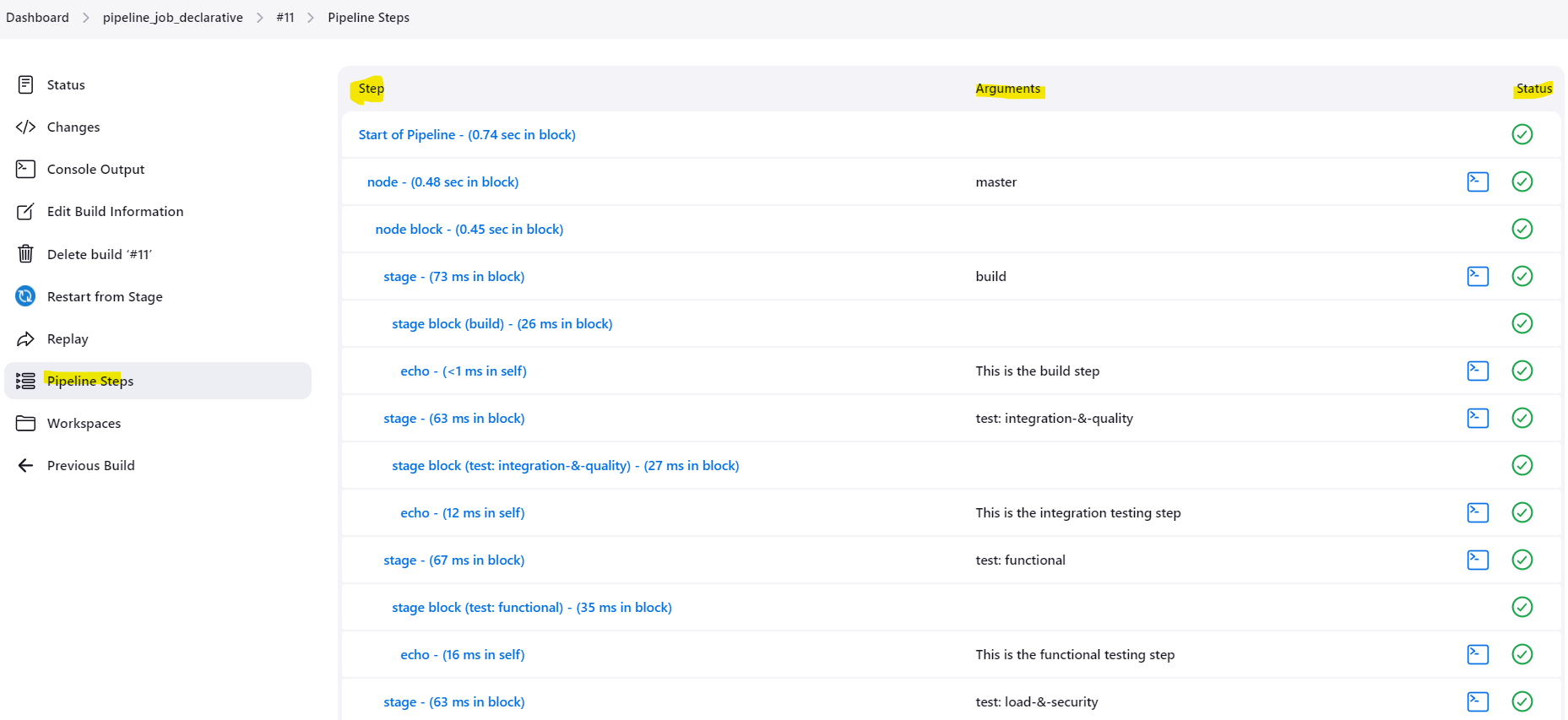


Check the job output ->

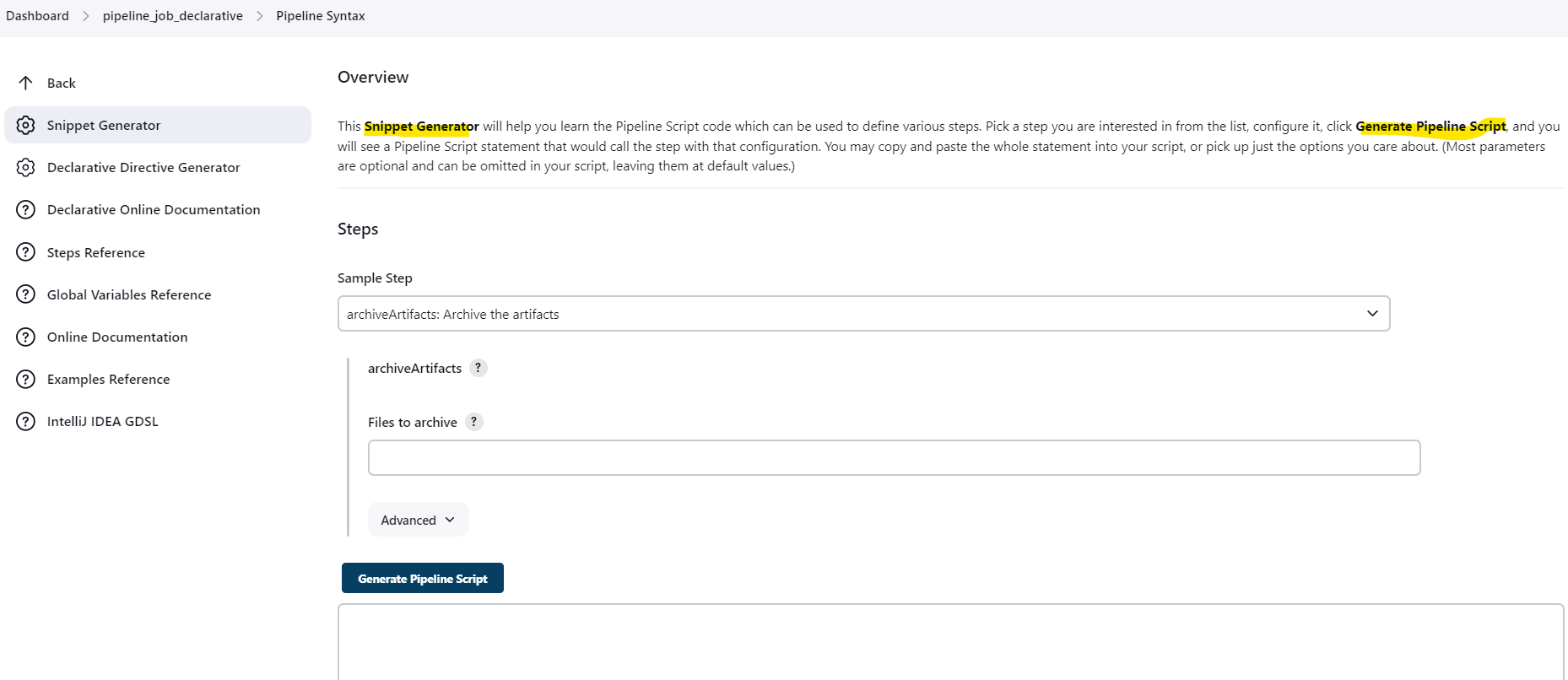


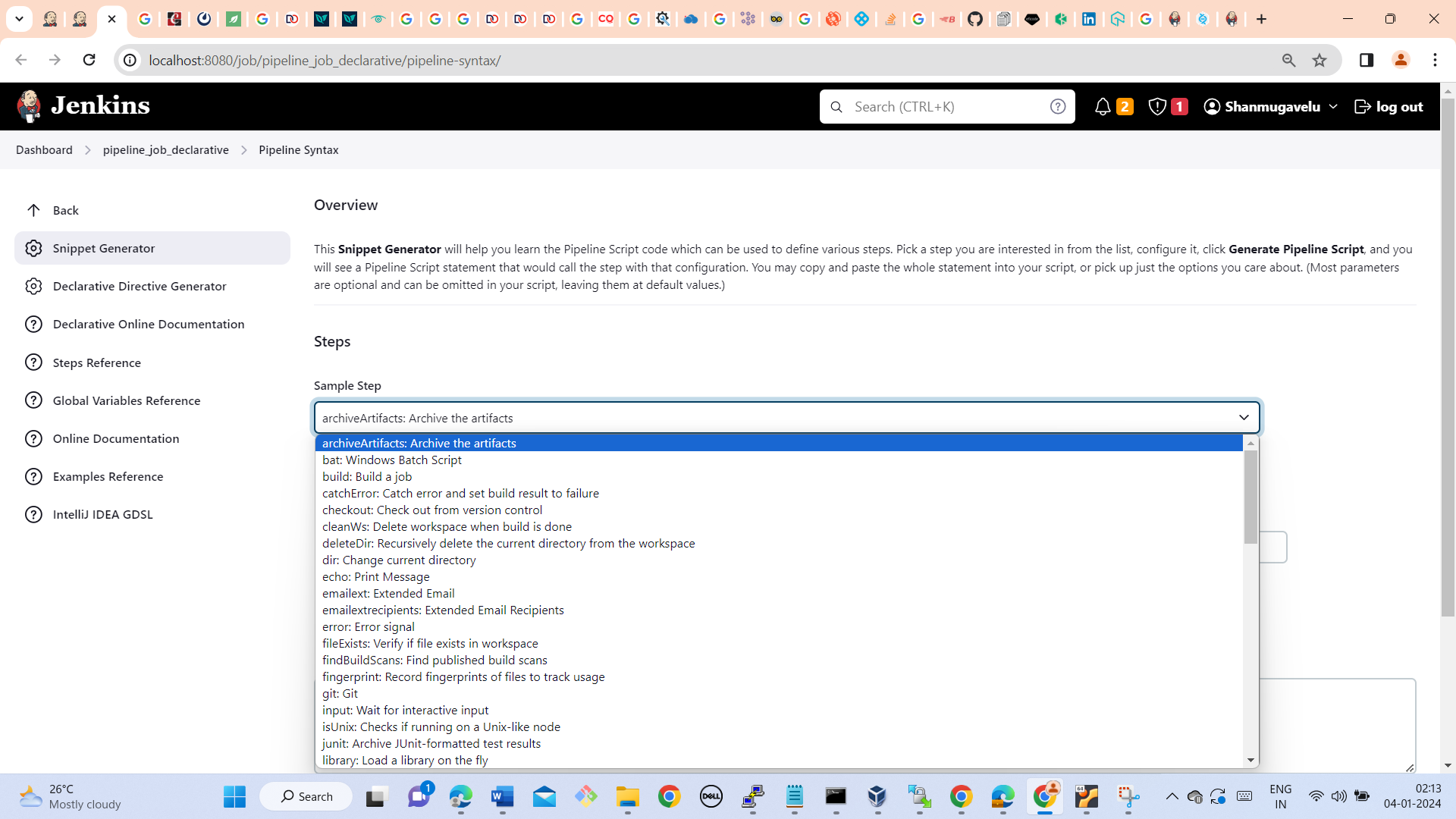


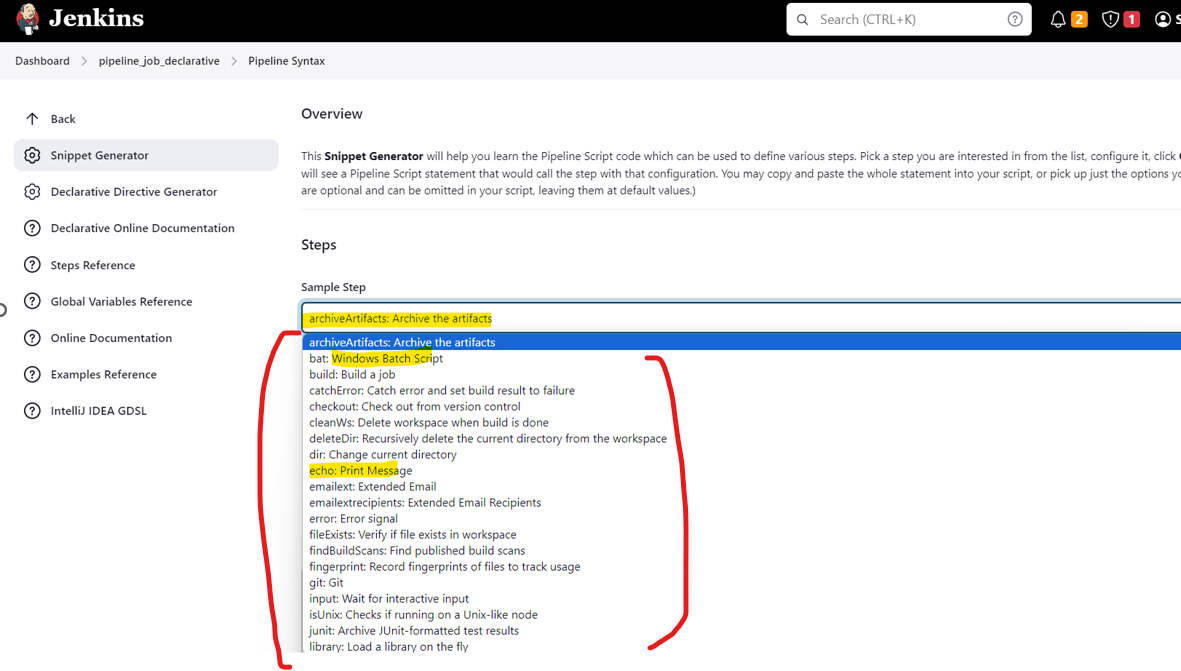
**Option ->Checking Pipeline Steps -Individual steps**

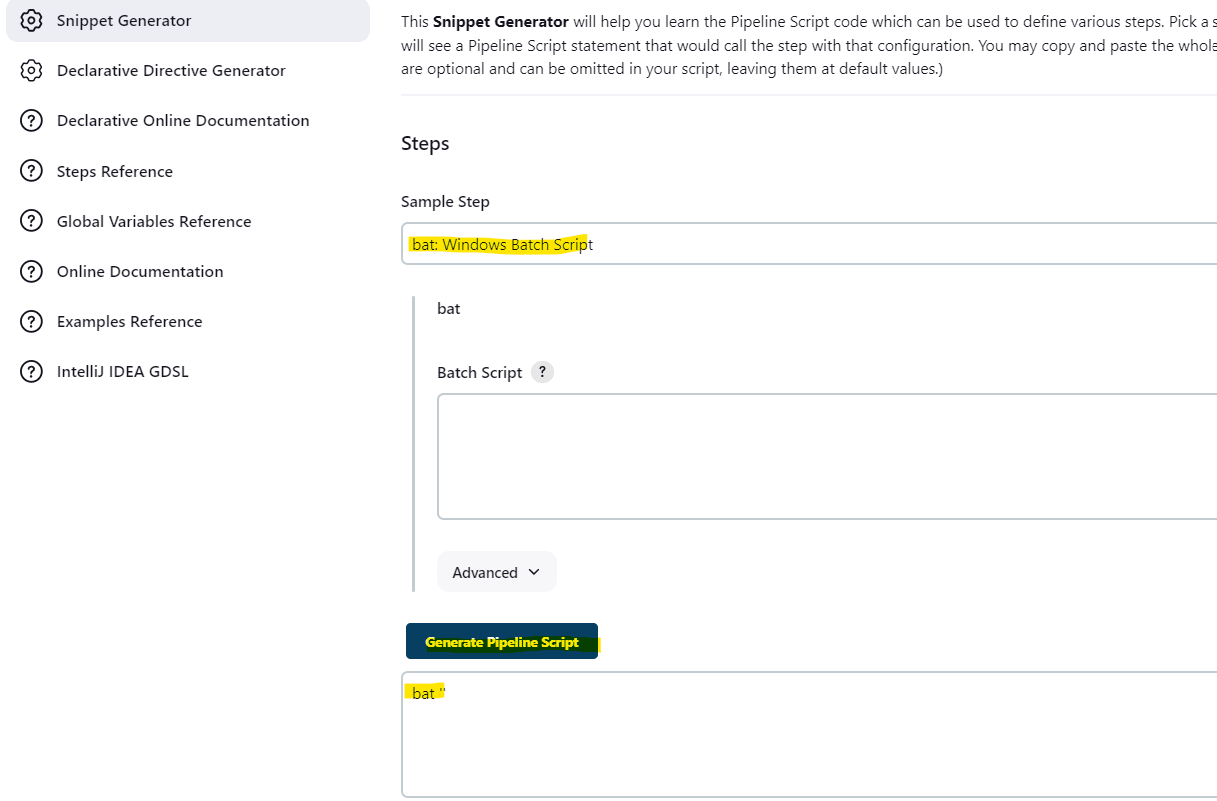


**Snipper Generator – Help/guide To generate Pipeline script**





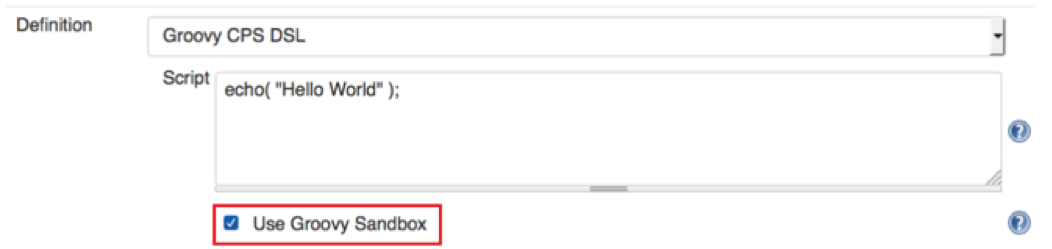


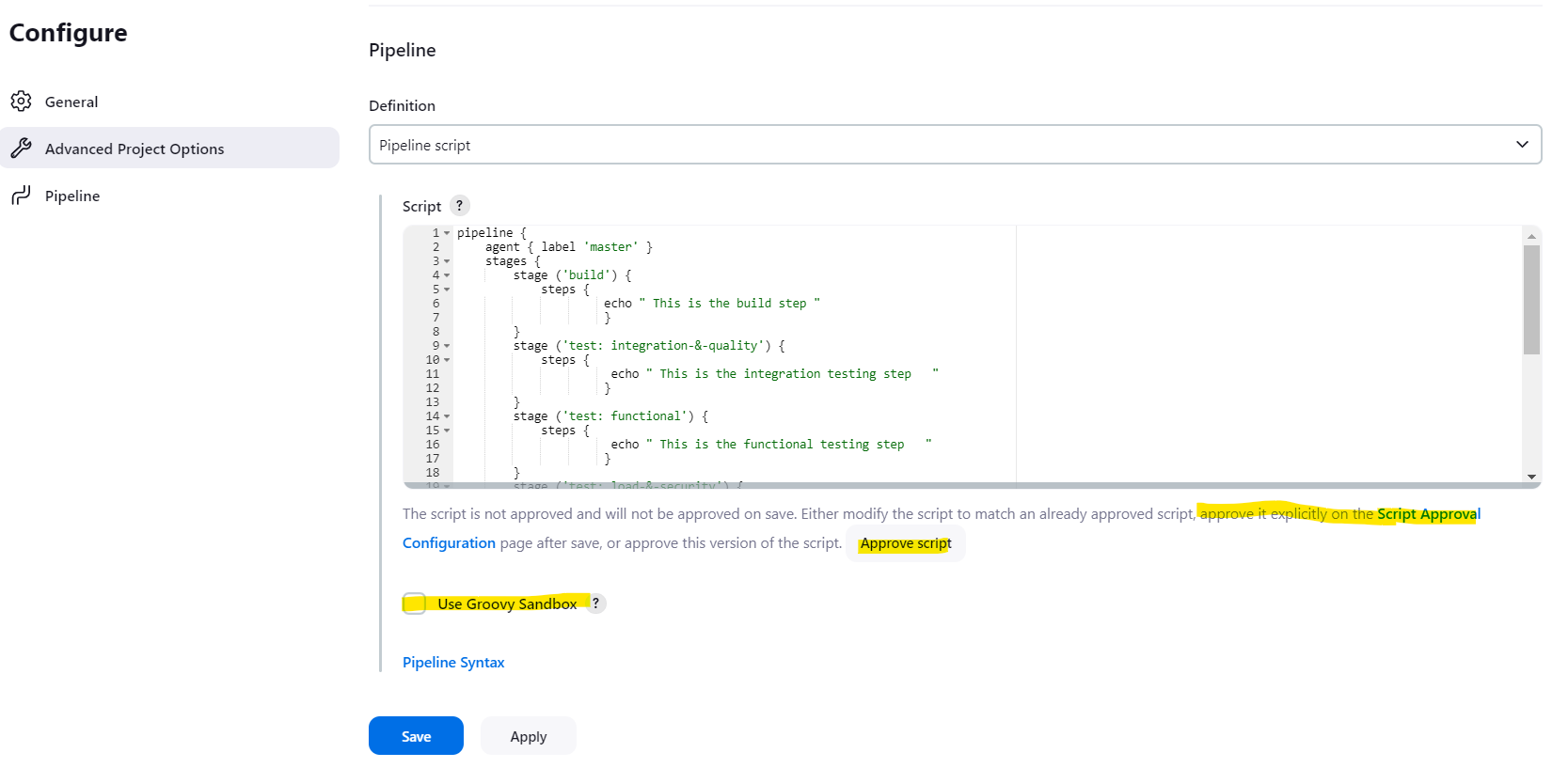


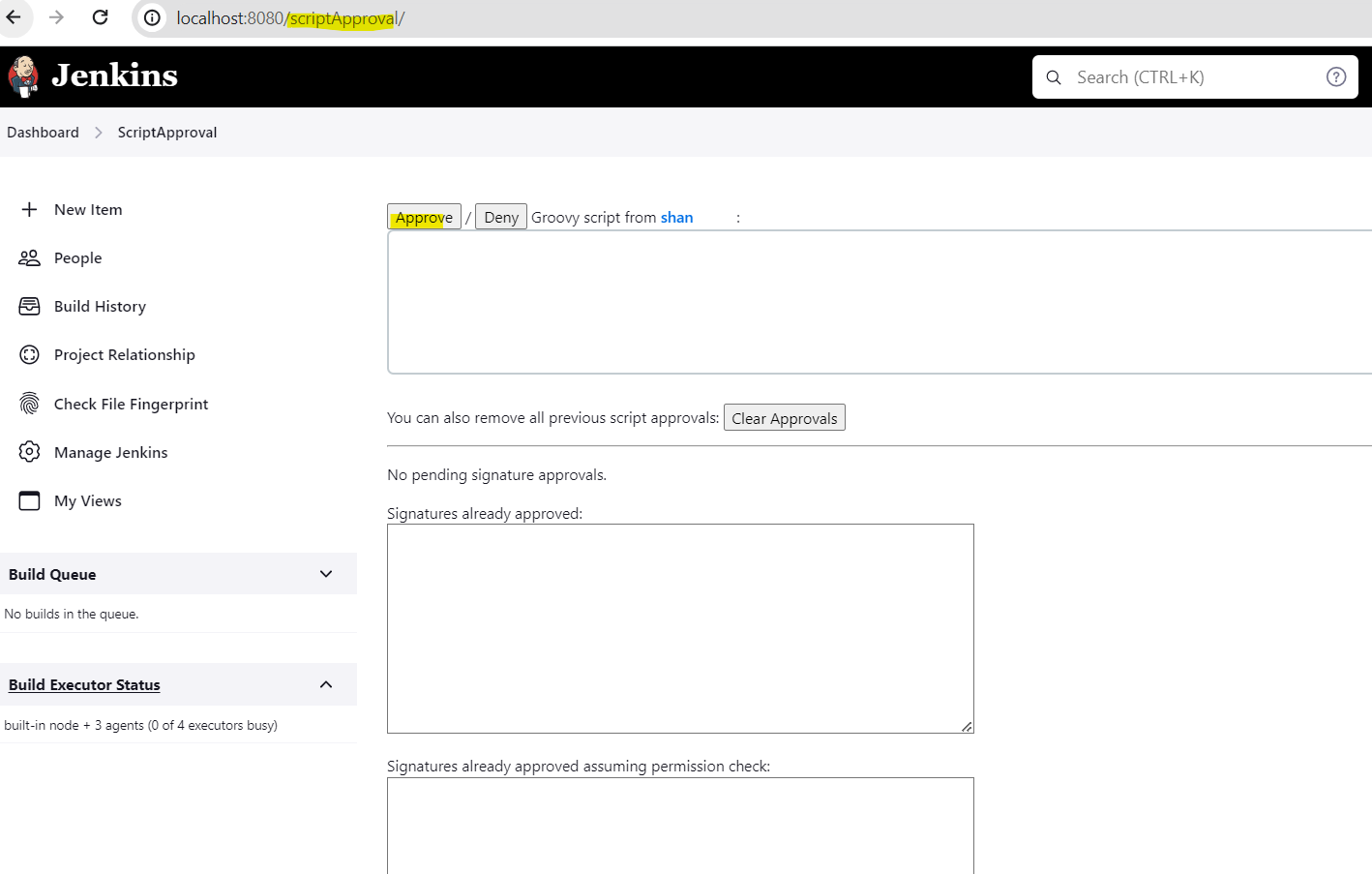
**Sandbox Security option in Jenkins**

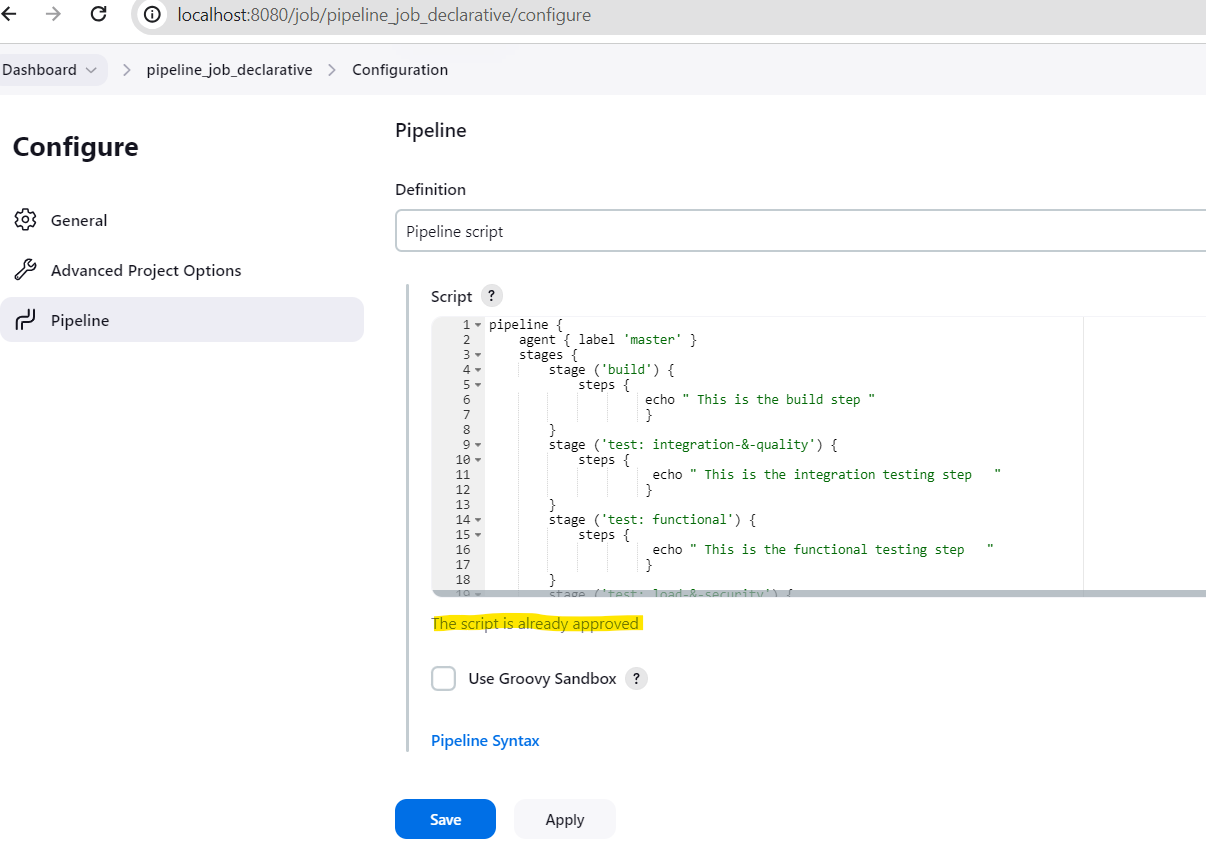
**This is option is for Restricted Access**

* Jenkins limits the execution of any Groovy script by providing a sandbox.
* The option “**Use Groovy Sandbox**,” shown below, is available in the Pipeline tab, and it allows the scripts to be run by any user without requiring administrator privileges.
* In this case, the script is run only by using the internal accessible APIs (which allow you to develop your script by using Groovy).
* When unchecked, if the script has operations that require approval, an administrator will have to provide them. This method is known as “**Script approval.**” By default, all Jenkins pipelines run in a Groovy sandbox.
* If the option is checked and unauthorized operations are used, the script will fail when run.









**What to choice between two – Declarative (or) Scripted ?**

* Scripted vs. declarative pipelines are different only in their programmatic approach.
* One uses a declarative programming model, while the other uses an imperative programming model.
* What about the choice between declarative vs. scripted pipelines for a new project?
* The answer to that question is most definitely the declarative pipeline.
* The development industry has largely moved toward a declarative programming model for CI/CD pipelines.
* Both [GitHub Actions](https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/How-to-publish-GitHub-Actions-artifacts-example) and GitLab CI support only [YAML pipelines](https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Jenkins-YAML-Pipeline-Example), which are very similar to declarative Jenkins pipelines.
* Furthermore, declarative pipelines are easier to maintain and they tend to have a lower learning curve.
* Though both these pipelines are based on the groovy DSL, the scripted pipeline uses stricter groovy based syntaxes because it was the first pipeline to be built on the groovy foundation.
* Since this Groovy script was not typically desirable to all the users, the declarative pipeline was introduced to offer a simpler and more optioned Groovy syntax.
* Declarative pipeline is a relatively new feature that supports the pipeline as code concept. It makes the pipeline code easier to read and write. This code is written in a Jenkinsfile which can be checked into a source control management system such as Git.
* Whereas, the scripted pipeline is a traditional way of writing the code. In this pipeline, the Jenkinsfile is written on the Jenkins UI instance.
* The declarative pipeline is defined within a block labelled ‘pipeline’ whereas the scripted pipeline is defined within a ‘node’
* The declarative syntax is the best approach to use when new [CI/CD workflows](https://www.jenkins.io/doc/book/pipeline/syntax/) are built.

## Written by Vinod Sairam

The Database Administrator who helps to manage Enterprise Databases who develops Devops Engineering ideas to support Database tasks, Infrastructure tasks and to reach the right audience.

**Assisted by Shanmugavelu. M**