Day 26 : Jenkins Declarative Pipeline

This is #90DaysofDevops challenge under the guidance of Shubham Londhe.

Day 26 TASK

Check this for task:

https://github.com/LondheShubham153/90DaysOfDevOps/blob/master/2023/day26/tasks.md

What is Pipeline :A pipeline is a collection of steps or jobs interlinked in a sequence.

Declarative: Declarative is a more recent and advanced implementation of a pipeline as a code.

Scripted: Scripted was the first and most traditional implementation of the pipeline as a code in Jenkins. It was designed as a general-purpose DSL (Domain Specific Language) built with Groovy.

Why you should have a Pipeline?

The definition of a Jenkins Pipeline is written into a text file (called a `Jenkinsfile`) which in turn can be committed to a project's source control repository.

A Jenkinsfile is a text file that contains the definition of a Jenkins Pipeline. It is typically stored in source control along with the code that it is building or deploying.

A Jenkins Pipeline is a way to define a set of steps that will be executed to build, test, and deploy software. It allows you to define your build process as code, which can be versioned and managed like any other code. The Jenkinsfile specifies the steps that should be taken at each stage of the pipeline, and can also specify when and how the pipeline should be triggered.

This is the foundation of "Pipeline-as-code"; treating the CD pipeline as a part of the application to be versioned and reviewed like any other code.

Creating a `Jenkinsfile` and committing it to source control provides a number of immediate benefits:

- Automatically creates a Pipeline build process for all branches and pull requests.
- Code review/iteration on the Pipeline (along with the remaining source code).

Pipeline syntax

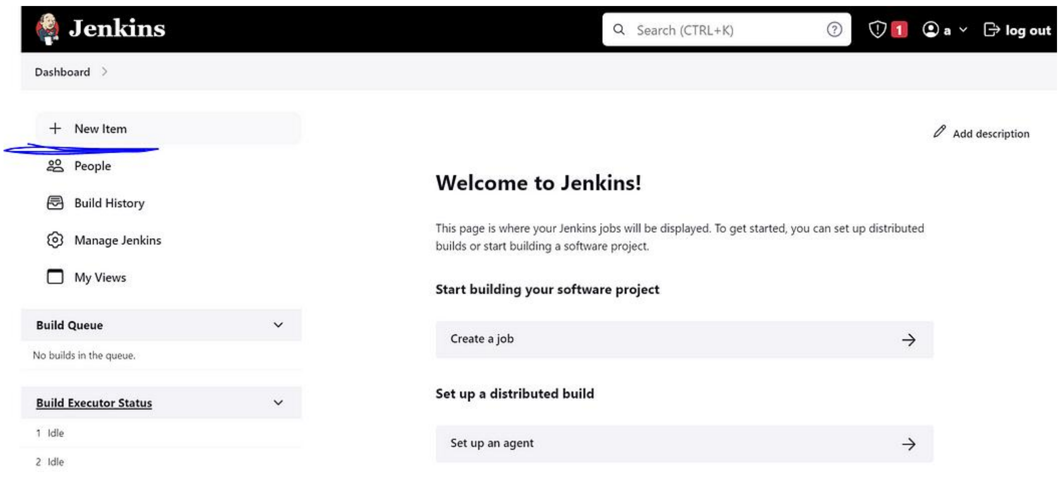
```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                //
            }
        }
        stage('Test') {
            steps {
                //
            }
        }
        stage('Deploy') {
            steps {
                //
            }
        }
    }
}
```

Task-01

1. Create a New Job, this time select Pipeline instead of Freestyle Project.
2. Follow the Official Jenkins Hello world example
3. Complete the example using the Declarative pipeline

First create an EC2 instance , install Jenkins on it. Access Jenkins using the public IP of the EC2 instance and the 8080 port.

Once you are done with the Jenkins installation and are able to access the same on the browser then open the Jenkins Dashboard and select “New Item”.



Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

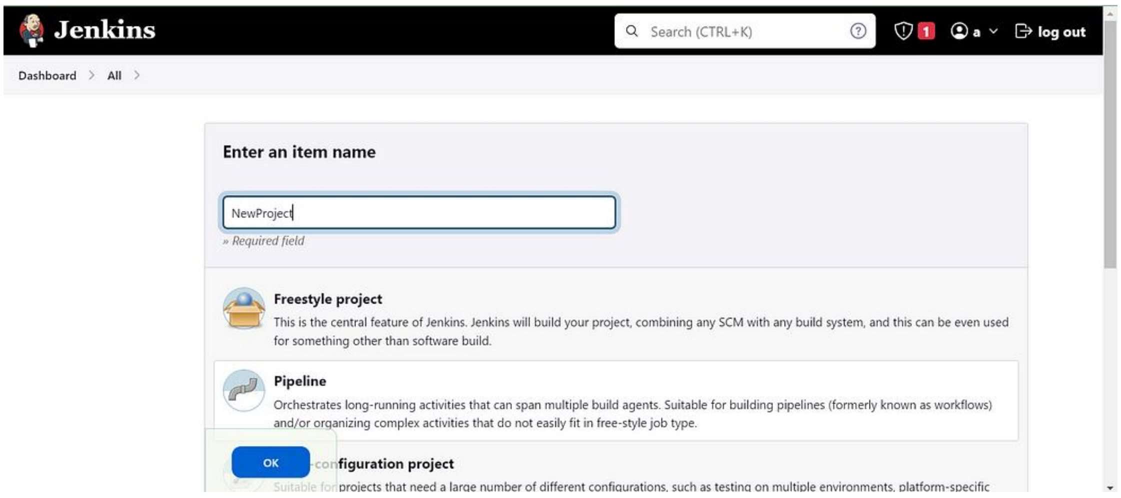
Set up a distributed build

No builds in the queue.

1 Idle

2 Idle

Here you need to add the Project Name and then select the project type as “pipeline” in this case because we will be creating the JenkinnFile further.



Enter an item name

NewProject

» Required field

Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline

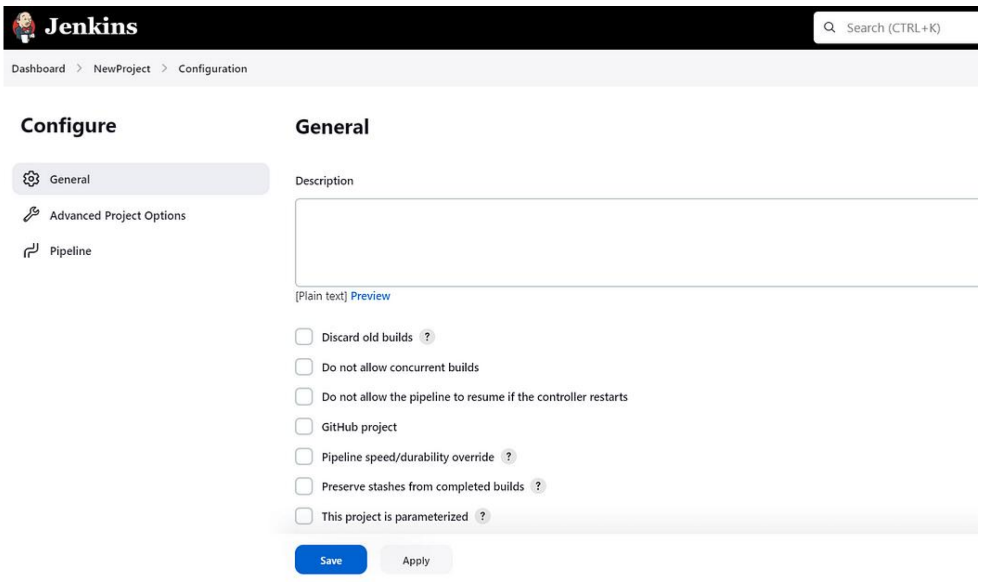
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific

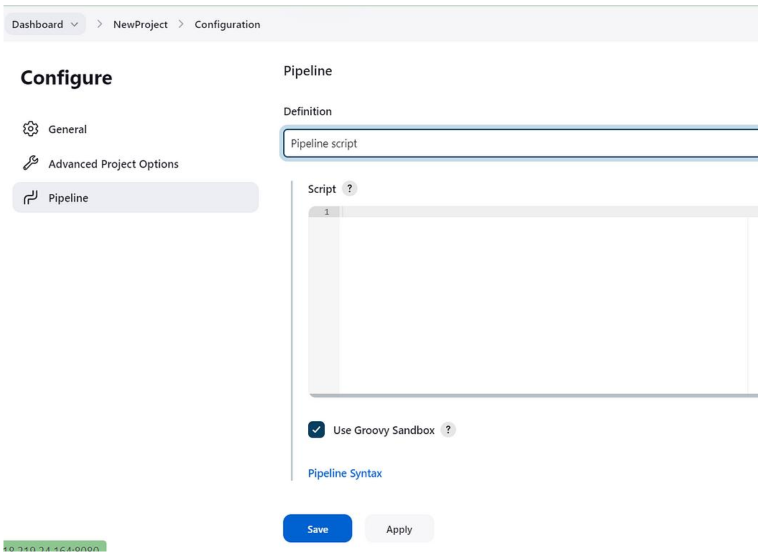
Click on “OK”.

Now we will land into the “Project Configuration” section.


Jenkins
Search (CTRL+K)
Dashboard > NewProject > Configuration
Configure
General
Advanced Project Options
Pipeline
General
Description
[Plain text] Preview
- [ ] Discard old builds ?
- [ ] Do not allow concurrent builds
- [ ] Do not allow the pipeline to resume if the controller restarts
- [ ] GitHub project
- [ ] Pipeline speed/durability override ?
- [ ] Preserve stashes from completed builds ?
- [ ] This project is parameterized ?

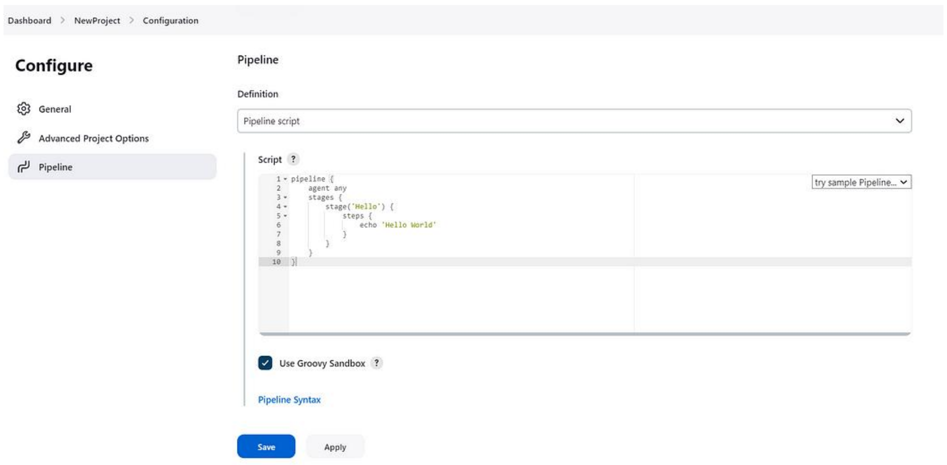
Save Apply

Here you need to go to the "pipeline" section and there you need to select the "Pipeline script" in definition.


Dashboard > NewProject > Configuration
Configure
General
Advanced Project Options
Pipeline
Pipeline
Definition
Pipeline script
Script ?
- [x] Use Groovy Sandbox ?

Pipeline Syntax
Save Apply

Now we will write a basic Pipeline Script of "hello world".



```
pipeline {
    agent any
    stages {
        stage('Hello') {
            steps {
                echo 'Hello World'
            }
        }
    }
}
```

Click on the **save** button to save the Pipeline.

```
pipeline {
    agent any
    stages {
        stage('Hello') {
            steps {
                echo 'Hello World'
            }
        }
    }
}
```

Code explanation

1. **Pipeline**

- The Declarative pipeline should start with the **pipeline** block and this is the **mandatory** block.

2. Agent

- Agent signifies where the Jenkins build job should run. In this case, we have selected agents as any.
- Jenkins supports a wide variety of agents. The entire list of supported agents in Jenkins can be found here

3. Stages/stage

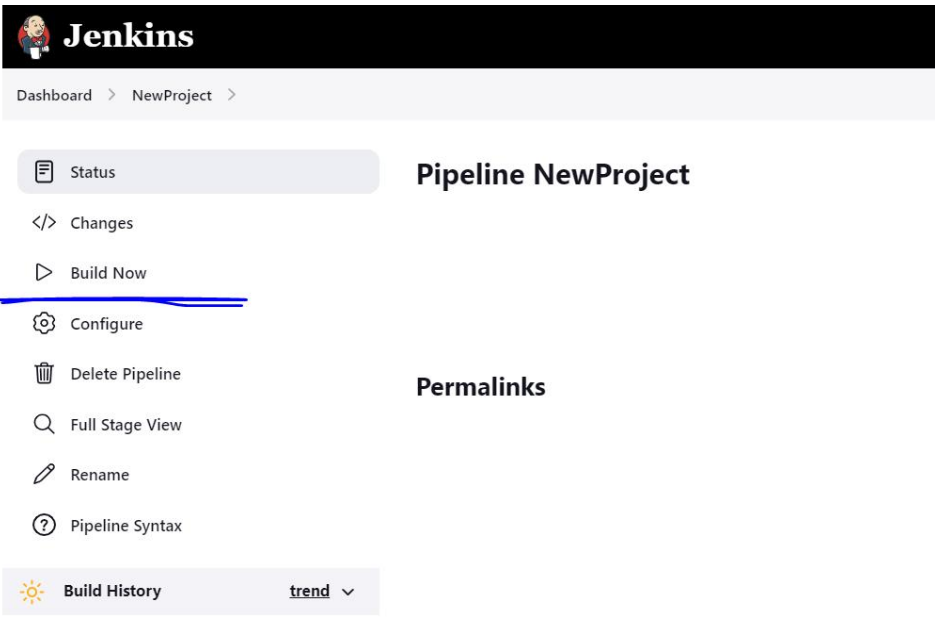
- stages block consists of different executable stage blocks.

- At least one stage block is mandatory inside the stages block.
- Here we have names stage as “Hello”

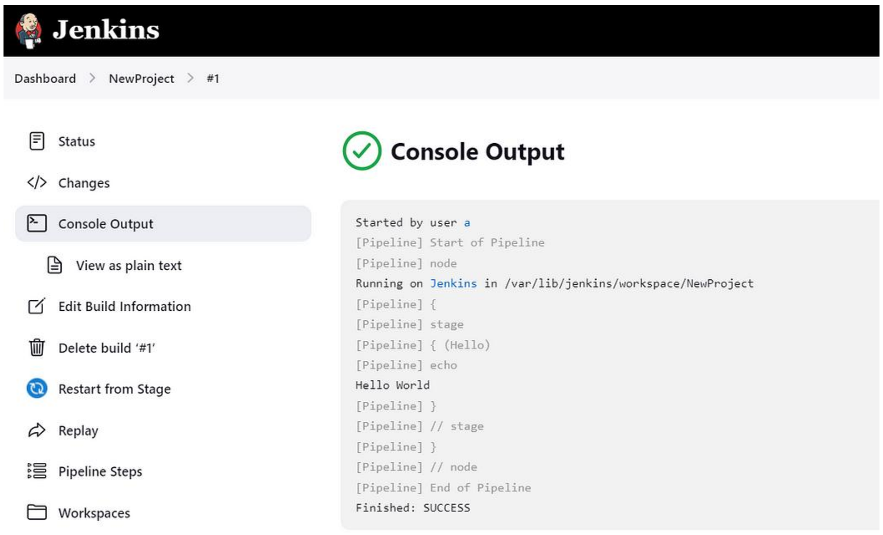
4. Steps

- Steps blocks consist of the actual operation which needs to be performed inside Jenkins.
- In the above example, we are printing “**Hello World**“

Now we can start the build process by manually clicking on the build now tab present.



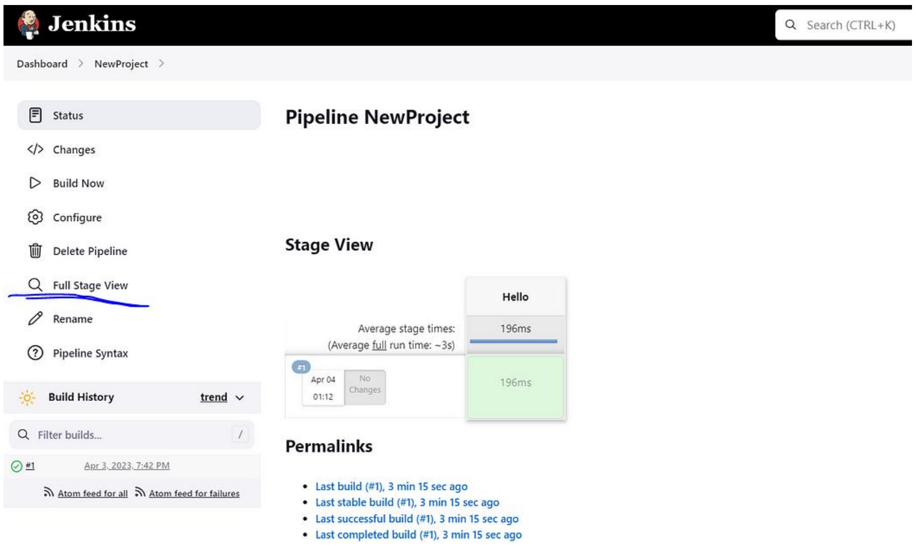
Once the build is complete you can check the output of the build by clicking on the “Console Output” tab present there.



```
Started by user a
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/NewProject
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Hello)
[Pipeline] echo
Hello World
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

You can see here the pipeline is running successfully and it did print "Hello World".

Now you check the multi stage view by clicking on the "Full Stage View" in the project main page.



Please, feel free to drop any questions in the comments below. I would be happy to answer them.

If this post was helpful, please do follow and click the clap

_Thank you for reading

_Rajani