Jenkins CI/CD

By Satya

What is CI/CD

- CI/CD is a method to frequently deliver apps to customers by introducing automation into the stages of app development. The main concepts attributed to CI/CD are continuous integration, continuous delivery, and continuous deployment
- CI/CD is the most important part of DevOps that is used to integrate various DevOps stages. Jenkins is the most famous Continuous Integration tool

What is Jenkins

- Jenkins is an open-source automation tool written in Java
- Jenkins achieves Continuous Integration with the help of plugins. Plugins allow the integration of Various DevOps stages. If you want to integrate a particular tool, you need to install the plugins for that tool.
- **Advantages of Jenkins:**

  It is an open-source tool with great community support.

  It is easy to install.

  It has 1000+ plugins to ease your work.

  It is free of cost.

  It is built with Java and hence, it is portable to all the major platforms.

What is a Jenkins pipeline

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.

The key feature of this pipeline is to define the entire deployment flow through code.

It basically follows the '**pipeline as code**' discipline. Instead of building several jobs for each phase, you can now code the entire workflow and put it in a **Jenkinsfile**. Below is a list of reasons why you should use the Jenkins Pipeline.

Jenkins Pipeline Advantages

It models simple to complex pipelines as code by using **Groovy DSL** (Domain Specific Language)

The code is stored in a text file called the Jenkinsfile which can be **checked into a SCM** (Source Code Management)

Improves user interface by incorporating **user input** within the pipeline

It is durable in terms of unplanned restart of the Jenkins master

It can restart from saved **checkpoints**

It supports complex pipelines by incorporating conditional loops, fork or join operations and allowing tasks to be performed in parallel

It can integrate with several other plugins

- What is a Jenkinsfile

A **Jenkinsfile** is a text file that stores the entire workflow as code

The Jenkinsfile is written using the Groovy DSL. It is written based on two syntaxes

- Declarative pipeline syntax

```
pipeline {

}
```

- Scripted pipeline syntax

```
node {

}
```

This is a user defined block. All the stages and steps are defined within this block. It is the key block for a declarative pipeline

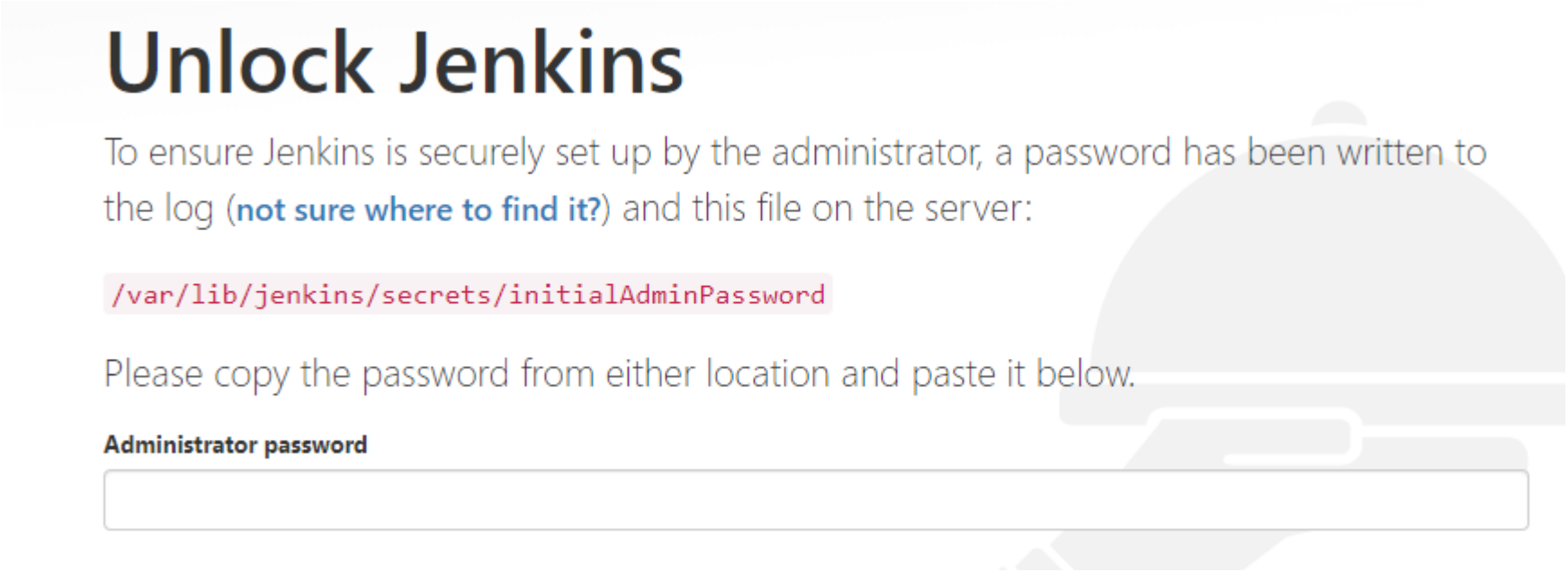
A node is a machine that executes an entire workflow. It is a key part of the scripted pipeline syntax.

```
#Declarative syntax
pipeline {
    agent any

    stages {
        stage("Build") {
            steps {
                echo "Some code compilation here..."
            }
        }

        stage("Test") {
            steps {
                echo "Some tests execution here..."
                echo 1
            }
        }
    }
}
```

```
#Scripted syntax
node {
    stage("Build") {
        echo "Some code compilation here..."
    }

    stage("Test") {
        echo "Some tests execution here..."
        echo 1
    }
}
```

- Once Jenkins installed successfully, Lunch Jenkins

  http://<Instance ip address>:8080/




```
# cat /var/lib/jenkins/secrets/initialAdminPassword
bd64c9d595c54b3eab2e32027a15cf15
```

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Create First Admin User

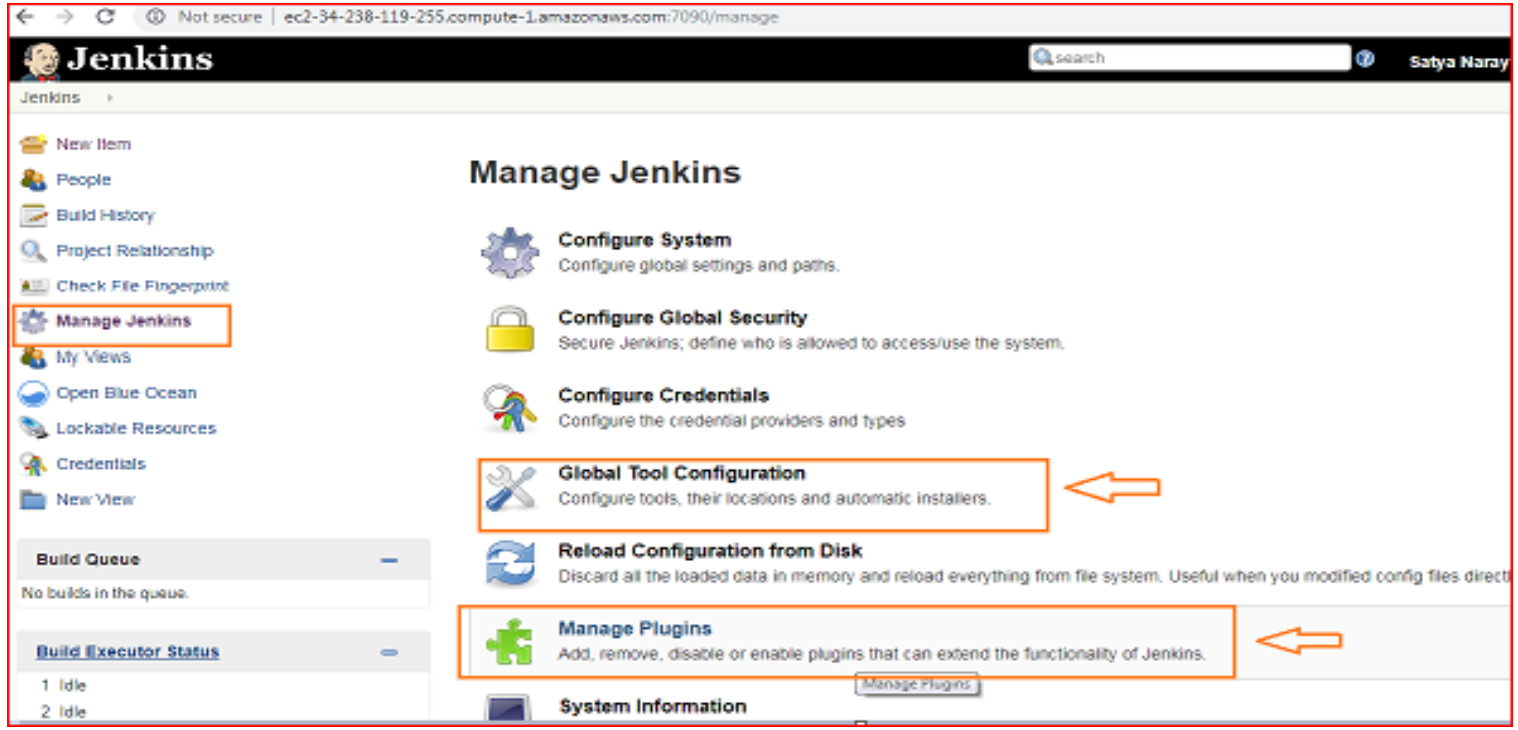
Username:	admin
Password:	••••••••••
Confirm password:	••••••••••
Full name:	Satya
E-mail address:	satyasahuaws@gmail.co

Configure Jenkins

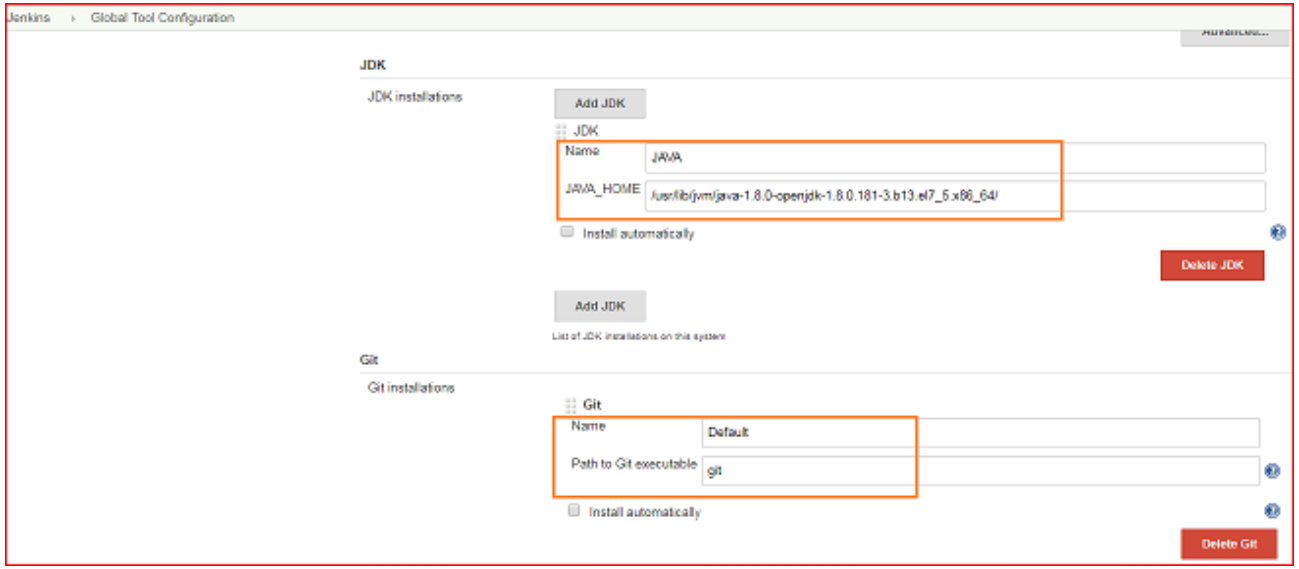
- After Jenkins is installed, JDK, Maven and Githas to be configured

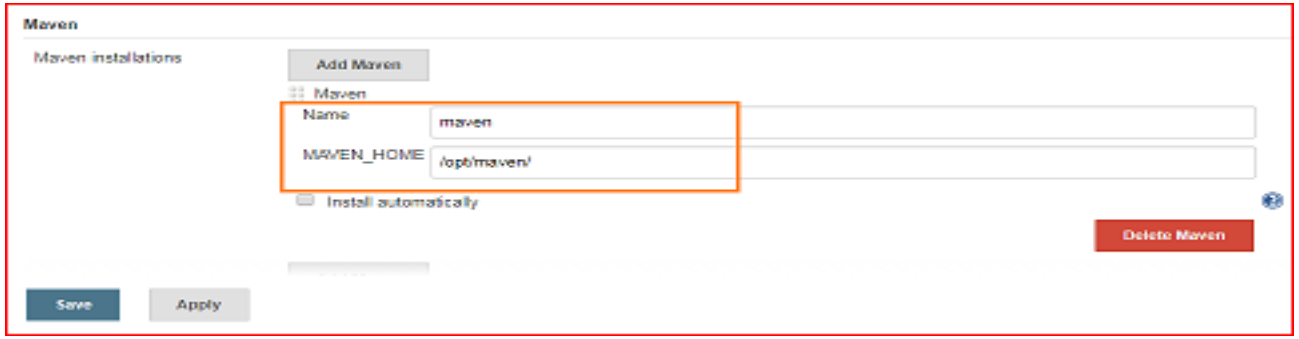
Follow the below steps to start with configuration.

Go to Jenkins >> Manage Jenkins >> Global Tool configuration >>



Add JDK, Git, Maven path as below >> Apply and Save





Jenkins Jfrog integration

Manage Jenkins -> Manage Plugins -> Available -> Artifactory

Artifactory

Artifactory servers

Use the Credentials Plugin

Artifactory

Server ID: Artifactory Version 4.15.0

URL: http://localhost:8081/artifactory/

Advanced...

Default Deployer Credentials

Username: jenkins

Password: ••••••••••

Found Artifactory 4.15.0

Test Connection

Use Different Resolver Credentials

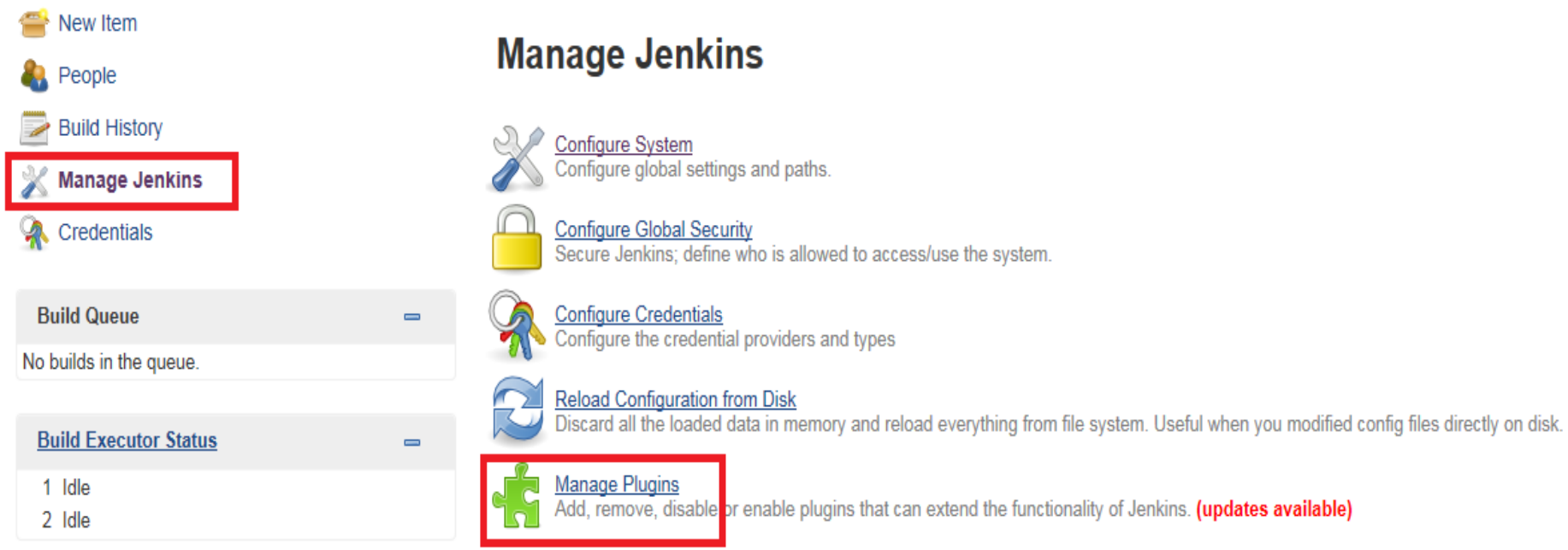
Delete

Save

Apply

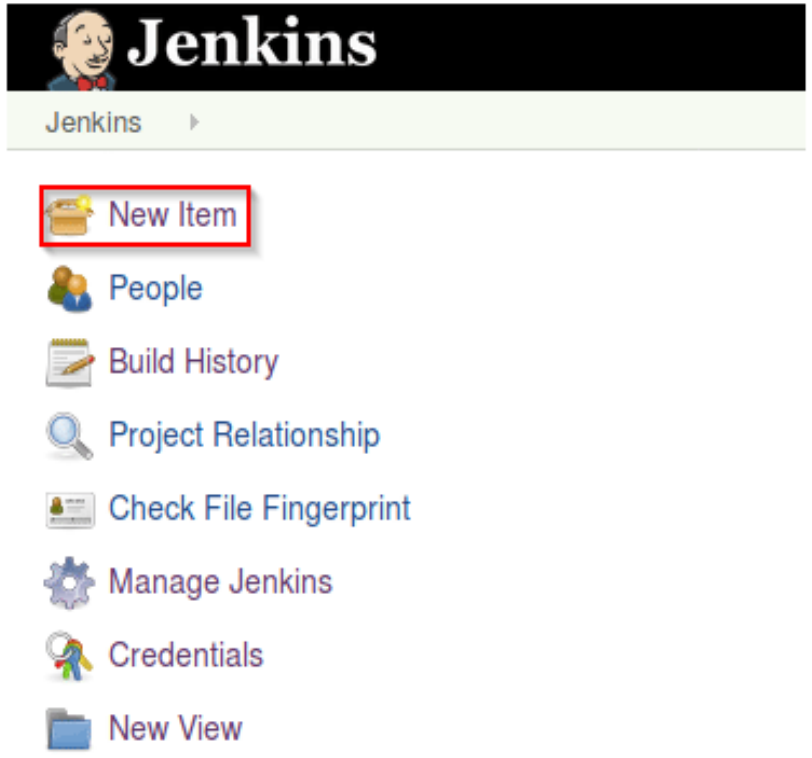
Installing Plugins

Go to Jenkins >> Manage Jenkins >> Manage plugins >>

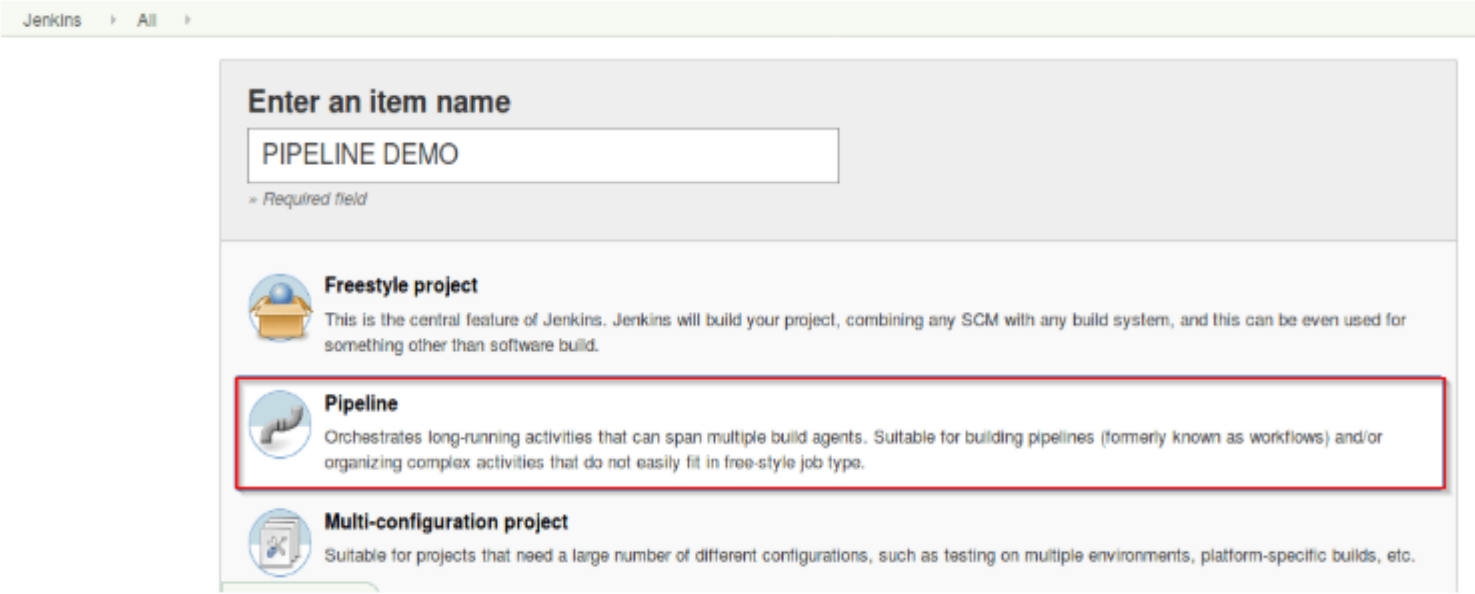


Creating your first Jenkins pipeline.

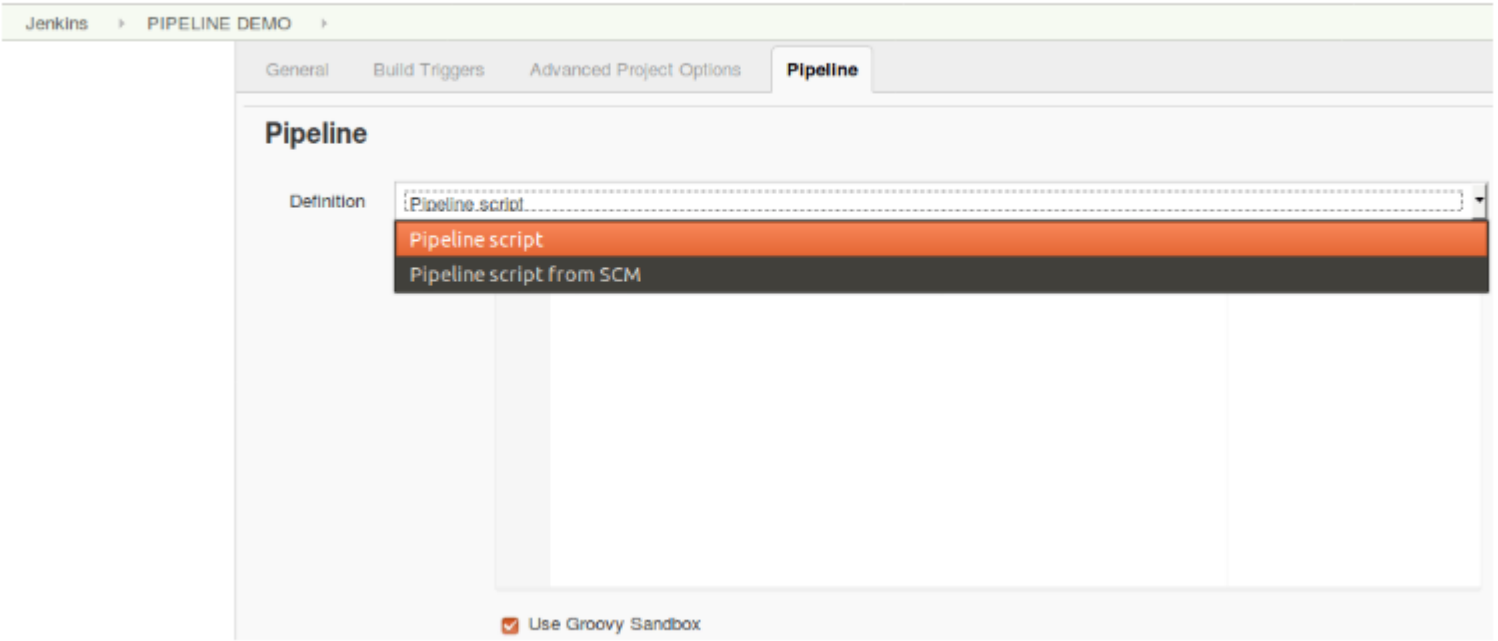
Step 1: Log into Jenkins and select 'New item' from the dashboard.



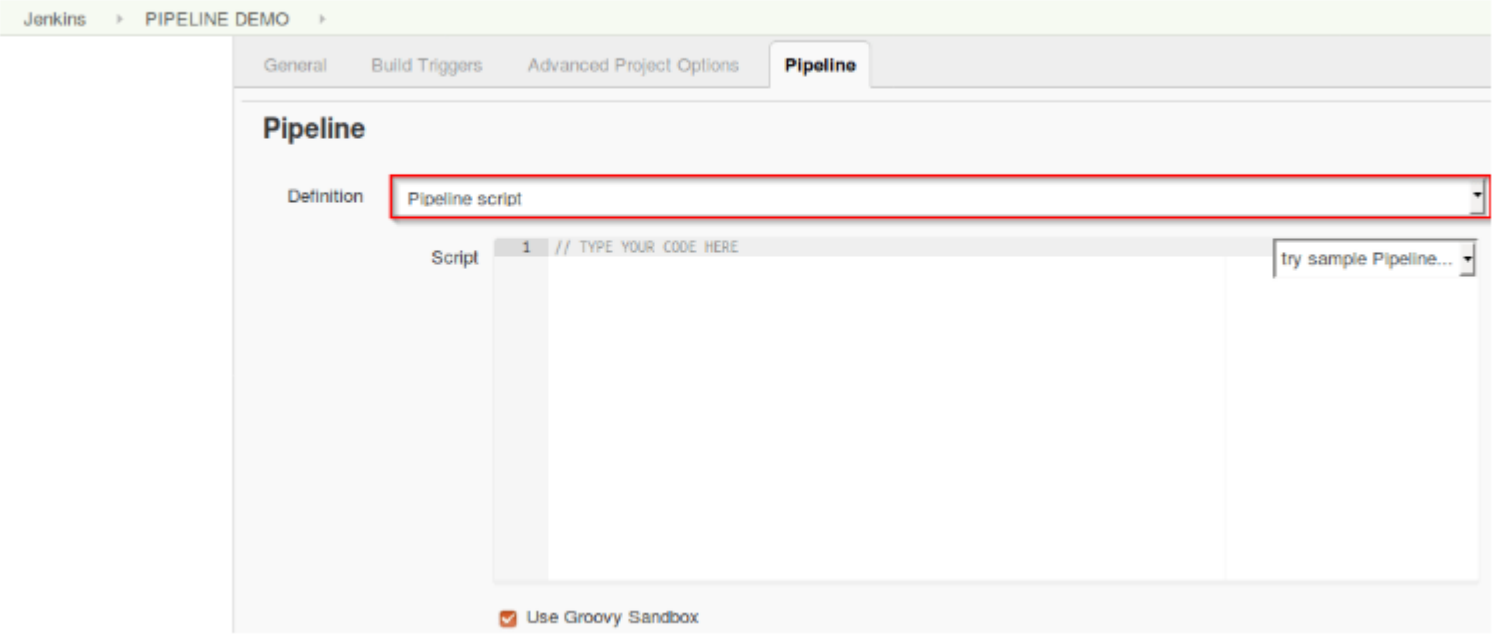
Step 2: Next, enter a name for your pipeline and select 'pipeline' project. Click on 'ok' to proceed.



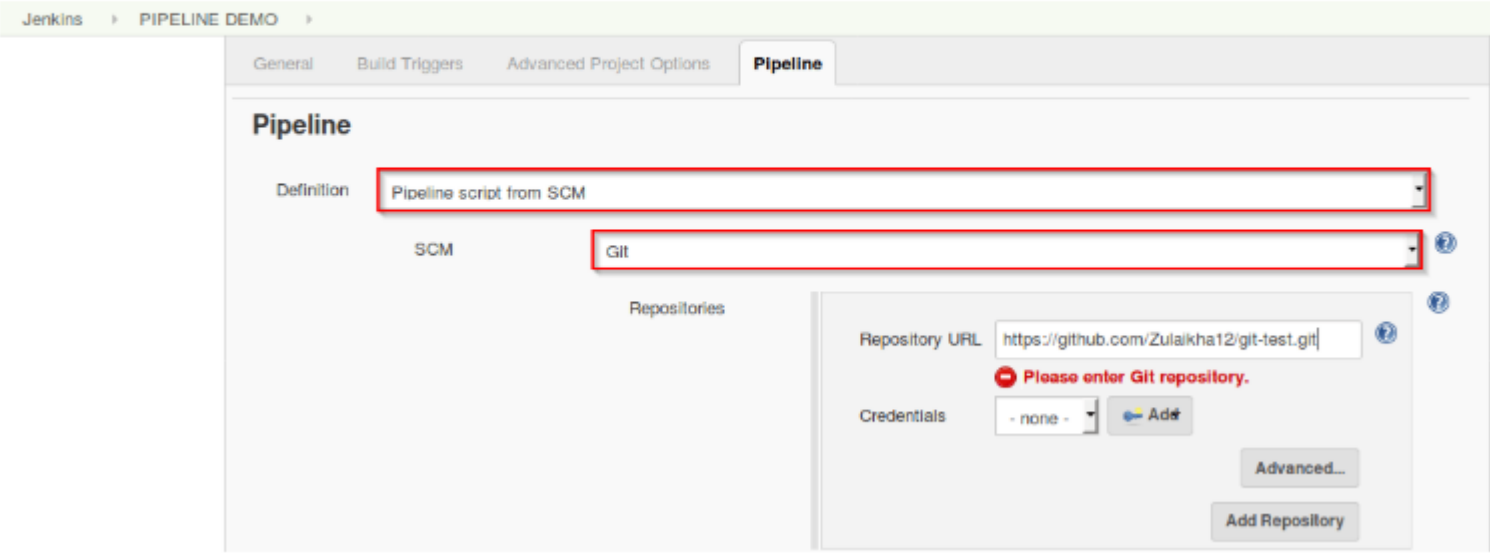
Step 3: Scroll down to the pipeline and choose if you want a declarative pipeline or a scripted one.



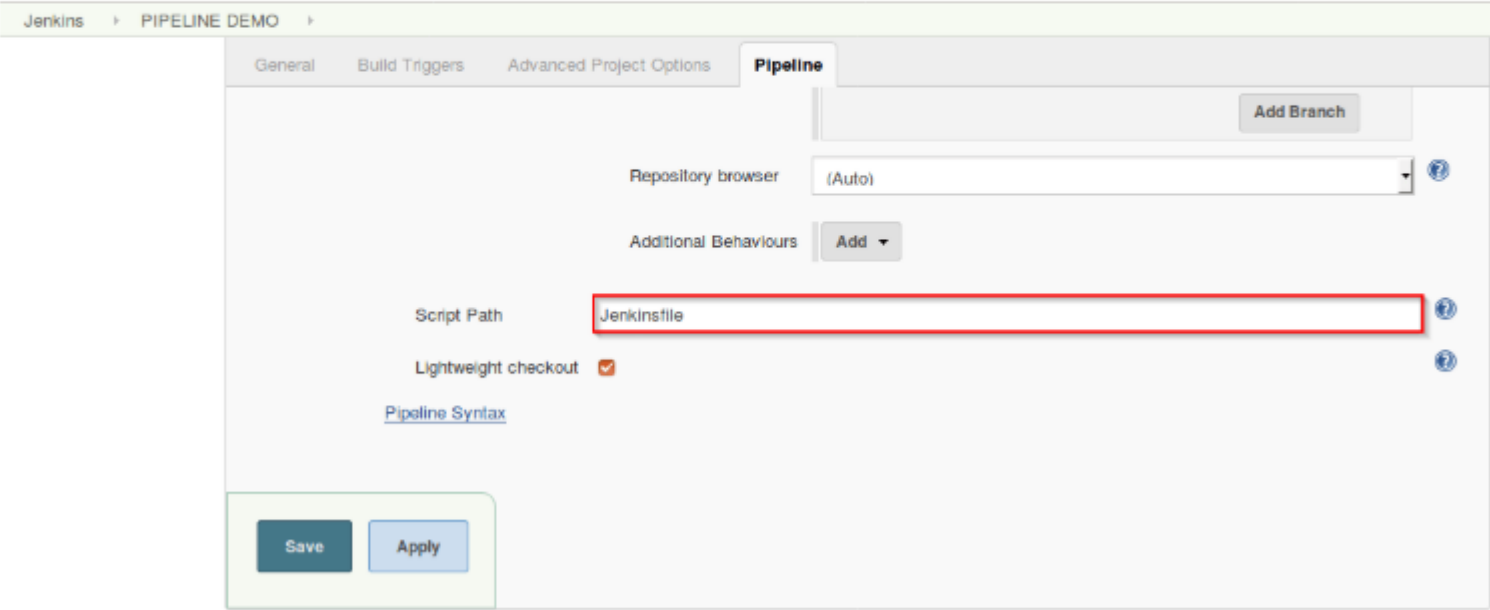
Step 4a: If you want a scripted pipeline then choose 'pipeline script' and start typing your code.



Step 4b: If you want a declarative pipeline then select 'pipeline script from SCM' and choose your SCM. In my case I'm going to use Git throughout this demo. Enter your repository URL.



Step 5: Within the script path is the name of the Jenkinsfile that is going to be accessed from your SCM to run. Finally click on 'apply' and 'save'. You have successfully created your first Jenkins pipeline.



- Pipeline code validation at startup

```
pipeline {
    agent any

    stages {
        stage("Build") {
            steps {
                echo "Some code compilation here..."
            }
        }

        stage("Test") {
            steps {
                echo "Some tests execution here..."
                echo 1
            }
        }
    }
}
```

pipeline-sandbox < 71

Pipeline | Changes | Tests | Artifacts | Logout

Branch: –
Commit: –

<1s
a minute ago

No changes
Started by user Szymon Stepniak

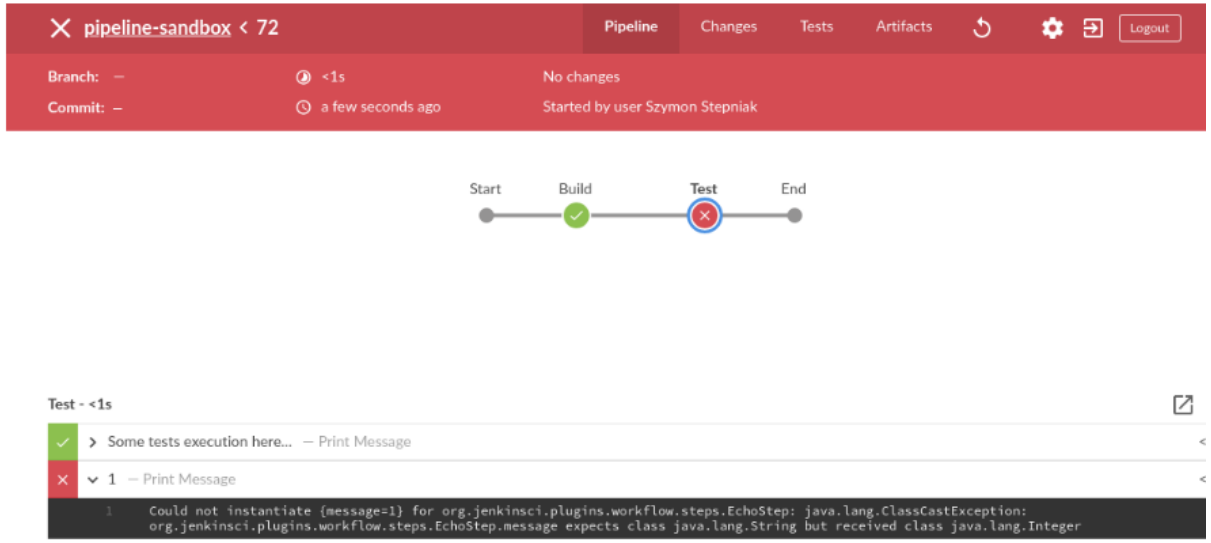
Start — Build — End

Build

No steps This stage has no steps

```
Started by user Szymon Stepniak
Running in Durability level: MAX_SURVIVABILITY
org.codehaus.groovy.control.MultipleCompilationErrorsException: startup failed:
WorkflowScript: 14: Expecting "class java.lang.String" but got "1" of type class java.lang.Integer instead @ line 14, column 22.
                    echo 1
                         ^

1 error

        at org.codehaus.groovy.control.ErrorCollector.failIfErrors(ErrorCollector.java:310)
        at org.codehaus.groovy.control.CompilationUnit.applyToPrimaryClassNodes(CompilationUnit.java:1085)
        at org.codehaus.groovy.control.CompilationUnit.doPhaseOperation(CompilationUnit.java:603)
        at org.codehaus.groovy.control.CompilationUnit.processPhaseOperations(CompilationUnit.java:581)
```

```
node {
    stage("Build") {
        echo "Some code compilation here..."
    }

    stage("Test") {
        echo "Some tests execution here..."
        echo 1
    }
}
```



The declarative pipeline in this case handles such use case much better.

Restart from stage

Another cool feature that only declarative pipeline has is "Restart from stage"

```
pipeline {
    agent any

    stages {
        stage("Build") {
            steps {
                echo "Some code compilation here..."
            }
        }

        stage("Test") {
            steps {
                echo "Some tests execution here..."
            }
        }
    }
}
```

localhost:9090/job/scripted-vs-declarative-pipeline/5/restart/

Jenkins

Jenkins › scripted-vs-declarative-pipeline › #5 › Restart from Stage

- Back to Project
- Status
- Changes
- Console Output
- Edit Build Information
- Delete build '#5'
- Open Blue Ocean
- Build timeline
- Restart from Stage
- Replay

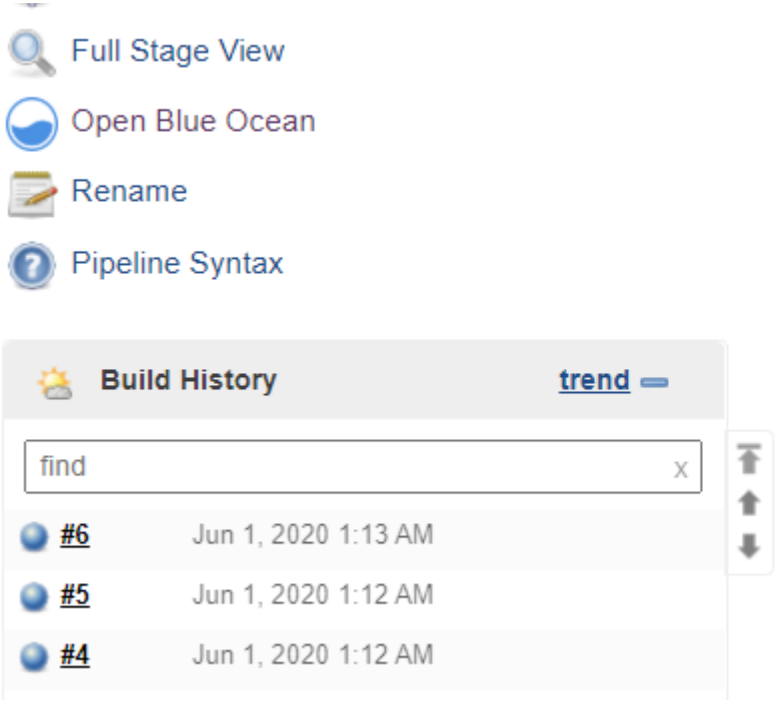
Restart #5 from Stage

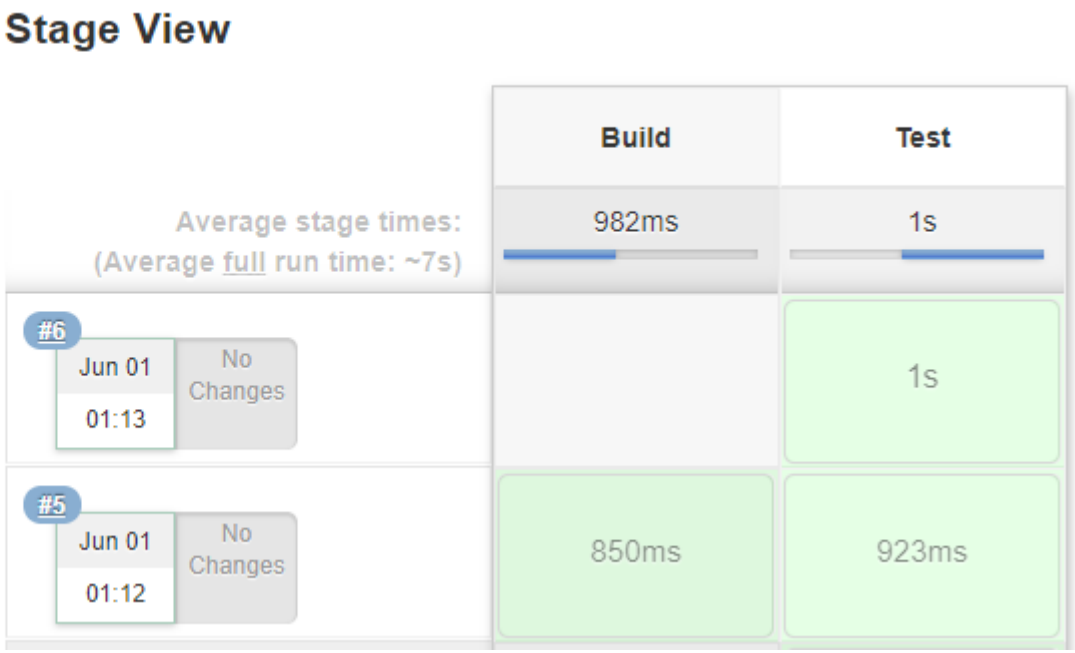
Stage Name Build

Build

Test

Run





No restart option you can see in scripted pipeline.

Q&A