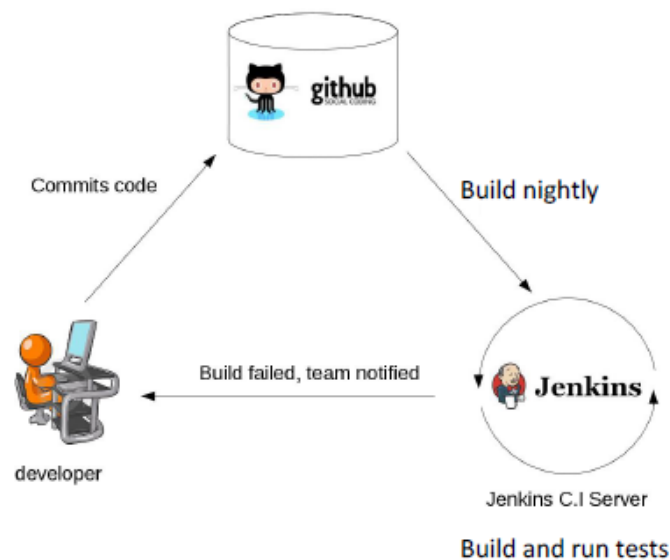


# Jenkins

Jenkins is an open source Continuous Integration tool written in Java. It is used to manually, periodically, or automatically build software development projects.

## Why Jenkins?

Jenkins is a software that allows continuous integration. Jenkins will be installed on a server where the central build will take place. The following flowchart demonstrates a very simple workflow of how Jenkins works.



A build is triggered whenever new code is committed to the central repository.

Broken builds are usually treated as a high priority issue and are fixed quickly.

## Jenkins installation steps:

```
sudo yum update -y
```

```
sudo wget -O /etc/yum.repos.d/jenkins.repo \
```

```
https://pkg.jenkins.io/redhat-stable/jenkins.repo
```

```
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
```

```
sudo yum upgrade
```

```
sudo amazon-linux-extras install java-openjdk11 -y
```

```
sudo yum install jenkins -y
```

```
sudo systemctl enable jenkins
```

```
sudo systemctl start jenkins
```

```
sudo systemctl status jenkins
```

### How to configure jenkins for the first time ?

Search on browser: IP Address:port ---> default 8080

1. it will ask for initialAdminPassword

copy and paste `/var/lib/jenkins/secrets/initialAdminPassword`

2. Install suggested plugins --->[auto installed plugins – git, github branch source, SSH build agents, pipeline, email extension, mailer, github groovy libraries]

3. Configure username and password

### How to provide sudo access to jenkins for all the folders in the jenkins server ?

```
cd /etc/
```

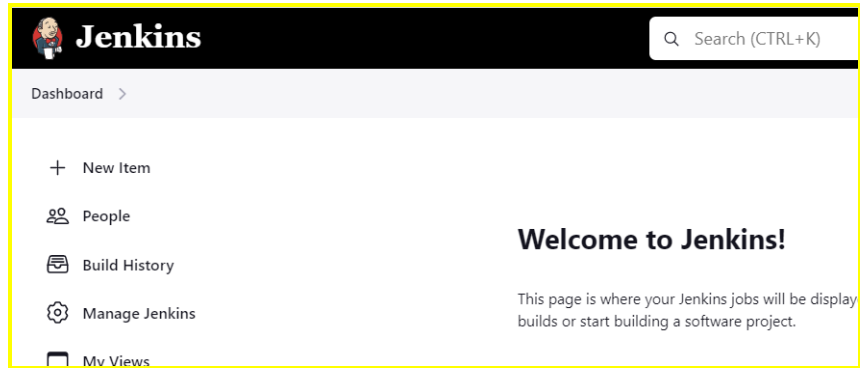
```
sudo vi sudoers (add below line)
```

```
jenkins ALL=(ALL) NOPASSWD: ALL
```

## Jenkins job creation steps-

Job is used to automate the tasks


- 1) click on the new item




Provide a name for the job and then select the job type among the options

job\_name


» 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.




**Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



**Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



**Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.



**Organization Folder**  
Creates a set of multibranch project subfolders by scanning for repositories.





## 1. General

### General

Enabled 

Description

[Plain text] [Preview](#)

- ☐ Discard old builds 
- ☐ GitHub project
- ☐ This project is parameterized 
- ☐ Throttle builds 
- ☐ Execute concurrent builds if necessary 

**Discard old builds:** This determines when, if ever, build records for this project should be discarded. Build records include the console output, archived artifacts, and any other metadata related to a particular build.

Keeping fewer builds means less disk space will be used in the Build Record Root Directory, which is specified on the Configure System screen.

Jenkins offers two options for determining when builds should be discarded:

**Days to keep builds:** discard builds when they reach a certain age; for example, seven days old.

**Max # of builds to keep:** discard the oldest build when a certain number of builds already exist.

These two options can be combined and used at the same time, so you can keep builds for 14 days, but only up to a limit of 50 builds, for example. If either limit is exceeded, then any builds beyond that limit will be discarded.

#### **This project is parameterized**

Parameters allow you to prompt users for one or more inputs that will be passed into a build

#### **Throttle builds**

Enforces a minimum time between builds based on the desired maximum rate.

#### **Execute concurrent builds if necessary**

When this option is checked, multiple builds of this project may be executed in parallel.

## 2. Source code management zone - Git repo configuration

The git plugin provides fundamental git operations for Jenkins projects. It can poll, fetch, checkout, and merge contents of git repositories.

### 3. Build trigger - to trigger Jenkins job automatically

Build Triggers

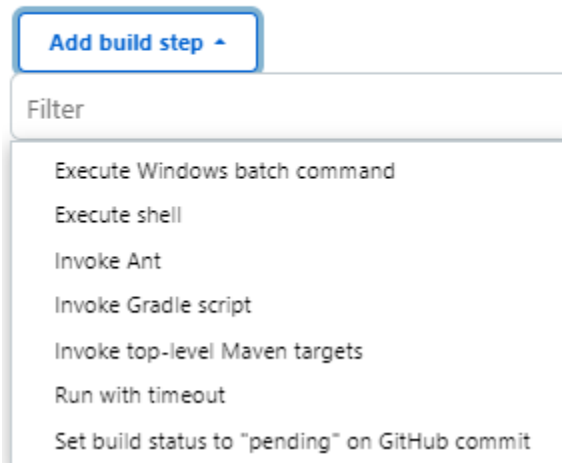
- ☐ Trigger builds remotely (e.g., from scripts) ?
- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ GitHub hook trigger for GITScm polling ?
- ☐ Poll SCM ?

#### Build triggers:

1. **build after other projects are built**
2. **build periodically** --> we need to provide cron expression
3. **Github hook trigger** --> trigger the job automatically after every job
  - a. select the option "github hook trigger" in jenkins job configuration section
  - b. In git hub --> in repository settings --> webhooks --> create a webhook ( in webhook , give payload url --> jenkins-url/github-webhook/  
  
ex: http://13.55.22.99:8080/github-webhook/  
or  
https://cdp-jenkins-citi.us/github-webhook/
4. **Poll scm** --> we have to give a cron schedule. Jenkins will poll/check for any new commits. If there is a commit, it will trigger the job.

#### 4. Build step - to compile source code

##### Build Steps



#### 5. Post build actions - to deploy source code, build report, email notification and so on

- Aggregate downstream test results
- Archive the artifacts
- Build other projects
- Publish JUnit test result report
- Record fingerprints of files to track usage
- Git Publisher
- E-mail Notification
- Editable Email Notification
- Set GitHub commit status (universal)
- Set build status on GitHub commit [deprecated]
- Delete workspace when build is done

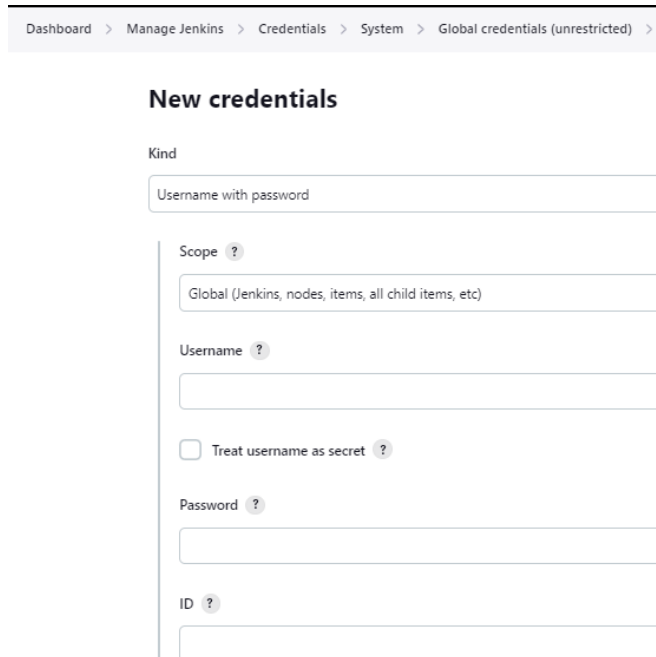
#### ASSIGNMENT

- 1.create Jenkins jobs for all the tasks that you automated using shell script.
2. clone a private using jenkins job
3. create a file which has the data of CPU , memory and disk space for every hour in a day ?
4. end-end automation of the website --> that means if there is a update in the code it should automatically reflect in my website

## Steps to Clone a private repo

we have to create credentials

Manage jenkins --> credentials --> system-- Global credentials- Add credentials



The screenshot shows the Jenkins 'New credentials' form. At the top is a breadcrumb trail: 'Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >'. Below this is the title 'New credentials'. The form has several fields: 'Kind' with a dropdown menu showing 'Username with password'; 'Scope' with a dropdown menu showing 'Global (Jenkins, nodes, items, all child items, etc)'; 'Username' with a text input field; a checkbox labeled 'Treat username as secret'; 'Password' with a text input field; and 'ID' with a text input field. Each field has a small question mark icon next to its label.

1. kind: username and password
2. username: github-username
3. password: PAT-token
4. ID: name for the credentials
5. description: anything

### Q) How to run builds parallely ?

While configuring the job we should tick the option **"Execute concurrent builds if necessary"**

### Q) What is parameterized plugin ? or This project is parameterized - what is this option ?

If you want to pass parameters to Jenkins job and you can use it later – then you can use this option.

## Types of parameters:

\*\*\***string parameter** --> provide the parameter by filling in the blanks while building the job

\*\*\***choice parameter** --> you can choose from the drop-down. the options will given in configure section

**file parameter** --> used if you want to upload a file in the trigger time

credential parameter

boolean parameter

## how to change the number of executors ?

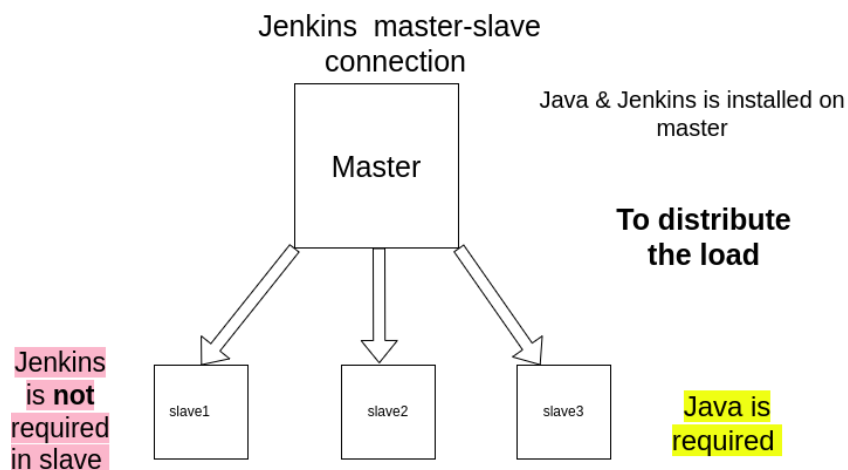
Manage jenkins --> Manages nodes and clouds --> select the node and then change the number of executors

## What are executors in jenkins?

Using executors we can specify the number of runners for the job. We can run multiple jobs parallely in if we have executors.

## Why Master-slave connection required in Jenkins ?

jenkins To distribute the load across multiple servers / to run jobs on slave servers.





## How to set up Jenkins master-slave connection?

**Manage Jenkins --> Manage nodes and clouds --> + new node**

Dashboard > Manage Jenkins > Nodes > New node

### New node

Node name

Type

☒ Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example

Create

### Steps:

1. Manage jenkins --> manage nodes and clouds
2. click on new node
3. give name and description
4. give the root directory path for the slave: /home/ec2-user
5. Launch agent via SSH
6. give IP address , add credentials ( give slave machine username and password )
7. Verifying strategy : non-verification strategy

Save and Click on launch agent.

Dashboard > Manage Jenkins > Nodes >

Name ?

node\_name

Description ?

[Plain text] [Preview](#)

Number of executors ?

1

Remote root directory ?

home/ec2-user

⚠ Are you sure you want to use a relative path for the FS?

Labels ?

Usage ?

Use this node as much as possible

Launch method ?

Launch agent by connecting it to the controller

Save

## How to Create Users in Jenkins ?

1. manage jenkins --> manage users --> create user ( give username and password )

## Q) How to restrict access to the users? How to give access to users ?

Manage Jenkins --> configure global security --> **Matrix based security** ( select the access to be given to new user from the matrix )

Dashboard > Manage Jenkins > Configure Global Security

Authorization

Matrix-based security

User/group	Overall	Credentials	Agent	Job	Run	View	SCM
	Administer	ManageDomains Create Delete Update	View Configure Connect Create Delete Disconnect Build Cancel Configure Create Delete	Discover Move Read Workspace Delete Replay Update Configure Delete Read Tag			
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Authenticated Users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Add user... Add group... ?

**Jenkins pipeline:** It is a collection of different stages. We have to write a script to perform any tasks.

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

### Pipeline syntax:

```
pipeline {  
  agent any  
  stages {  
    stage ( "name" ) {  
      steps {  
        //write your commands here  
      }  
    }  
  }  
}
```

## Jenkins pipeline examples:

1. Write a Jenkins pipeline to clone and build a java project in a slave ?

```
pipeline {  
    agent { label 'slave' }  
    stages {  
        stage (clone) {  
            steps {  
                sh "pwd"  
                sh "cd hello-world ; git pull https://github.com/ValaxyTech/hello-world.git"  
            }  
        }  
        stage ("mvn build") {  
            steps {  
                sh "pwd"  
                sh "cd hello-world ;mvn clean install"  
            }  
        }  
    }  
}
```

## Write a Jenkins pipeline to deploy a Java project?

```
pipeline{
    agent{label 'slave1'}

    stages{
        stage('clone'){
            steps{
                git url:"https://....git", credentialsId:"github_creds", branch:"master"
            }
        }

        stage('build'){
            steps{
                sh "mvn clean install"
            }
        }

        stage('deploy'){
            steps{
                sh "pwd"
                sh "cp target/app.war  apache-tomcat-8.0.15/webapps"
            }
        }
    }
}
```

## What is jenkinsfile?

It is a file where we write / keep all our jenkins pipeline job configuration which is written in Declarative syntax or Groovy script .

- We keep this jenkinsfile in a SCM (git) and this will be sourced in the pipeline job

**steps:**

1. **select pipeline script from SCM**
2. give repository url and credentials where the jenkins is there
3. path to jenkinsfile

## Which syntax / language have you used ?

Basically we have 2 types of syntax

1. declarative pipeline
2. scripted pipeline

=> I have used **Declarative pipeline** to create my CICD pipelines ( declarative syntax is also called as **DSL** - domain specific language )

=> **scripted Pipeline** : I have little bit of knowledge on this. we can use groovy scripting

**Agent** : Agent is written in jenkinsfile to specify where the job should run. We can define agent at the pipeline level or at each stage level.

## How do you specify agents ? or types of agents ?

1. any ==> it runs the pipeline/stage on any available executor
2. label ==> executes on particular labeled slave or choose from group of slaves
3. none ==> we use this at pipeline level which we are not defining agent at pipeline and we specifically need to define agent for each stage . you can specify separate agents for separate stages

ex: in the below example , I have specified separate agents for each stage.

```
pipeline{
```

```
agent none

stages{

  stage('clone'){

    agent any

    steps{

      git url:"https://github.com/Jamunamn/my-sample-website.git", credentialsId:"private_clone1",
branch:"master"

    }

  }

  stage('build'){

    agent{label 'slave1'}

    steps{

      sh "mvn clean install"

    }

  }

  stage('deploy'){

    agent{label 'slave2'}

    steps{

      sh "pwd"

      sh "cp webapp/target/webapp.war /home/ec2-user/apache-tomcat-8.0.15/webapps"

    }

  }

}

}
```

4. docker: creates docker container to run the jenkins job and the container will be deleted once the job is completed

5. Pod template: uses kubernetes pod to run the job and the pod will be deleted once the job is completed

**Jenkins shared Library** : It is used to create reusable snippets that can be used in pipeline. Shared library is useful if you want to collaborate and share the stages / steps of a pipeline across projects.

#### normal scenario / single file

##### Jenkinsfile

```
pipeline{
  agent{label 'slave1'}
  stages{
    stage('clone'){
      steps{

        git
        uri:"https://github.com/Jamunamn/my-
        sample-website.git",
        credentialsid:"private_clone1",
        branch:"master"
      }
    }

    stage('build'){
      steps{
        sh "mvn clean install"
      }
    }

    stage('deploy'){
      steps{
        sh "cp
        webapp/target/webapp.war /home/ec2-
        user/apache-tomcat-8.0.15/webapps"
      }
    }
  }
}
```

#### Jenkins shared library

##### jenkinsfile

```
pipeline{
  agent{label 'slave1'}
  stages{
    stage('clone'){
      steps{
        clone()
      }
    }

    stage('build'){
      steps{
        build()
      }
    }

    stage('deploy'){
      steps{
        deploy()
      }
    }
  }
}
```

##### library

```
def clone {
  git uri:"$uri", branch:""
}
```

```
def build {
  sh "mvn clean install"
}
```

```
def deploy {
  sh "cp $src $dest"
}
```



## How to restart jenkins ?

CLI: **sudo service jenkins restart**

UI: **jenkins-homepage-url/restart** ==> restart now

**jenkins-homepage-url/safeRestart** ==> restart after current running jobs are finished or plugin installation is completed.

### Additional commands:

sudo service jenkins stop ==> stop the jenkins

sudo service jenkins start ==> start the jenkins

## How do you check whether jenkins is running or not ?

UI: hit the jenkins url in the browser

CLI: **sudo service jenkins status**

( or )

**ps -eaf | grep jenkins**

**Plugins:** A plugin is a software add-on that is installed on a program, enhancing its capabilities. each plugin brings a new feature to jenkins.

## How to install plugins ?

manage jenkins --> manage plugins

## How to install customized plugins ?

1. Jenkins UI method:

manage jenkins --> manage plugins --> Advanced --> upload the plugin .hpi / .jpi

jpi ==> jenkins plugin interface

hpi ==> hudson plugin interface

## 2. CLI method in jenkins server

go to `/var/lib/jenkins/plugins` and upload the customized plugin

List of Plugins :

1. git
2. parameterized plugin
3. docker
4. kubernetes
5. Ansible
6. Jira
7. maven

Important plugins: \*\*\*\*\*

-----

**Sonarqube** : To analyze code quality.

**E-mail notification / extended E-mail notification** : to send emails

download: manage jenkins --> manage plugins --> download emailext plugin

configuration: manage jenkins --> configure system --> search email ext --> add SMTP server , add SMTP port

```
stage('email'){  
    steps {  
        emailext to: 'abc@gmail.com', subject: 'job execution', body: 'your job has completed the  
execution. Please check the console logs for the status of the job.'  
    }  
}
```

**Blue ocean plugin:** used for better visualization and better user interface and experience

**Gearman plugin:** This is also called High availability plugin. This supports multiple jenkins masters/controllers. Normally , If jenkins master is down you cannot trigger a single job. In this case , you have to resolve the issue and wait till the jenkins is restarted. But If we use the Gearman plugin, we will have a backup server to trigger jenkins jobs.

Download: Manage jenkins --> Manage plugins --> Download Gearman

Configure: Manage jenkins --> configure system --> Search Gearman --> add Gearman server --> Gearman port

**Selenium Plugin:** this is used to automate test cases.

### Q) How to take backup of jenkins ?

you have to create a copy of /var/lib/jenkins --> jenkins\_backup. Generally I'll do it once a week. We can use this backup whenever the jobs are deleted by mistake or the master server crashes

### Q) What to do when you forgot the jenkins admin credentials ?

go to /var/lib/jenkins in config.xml make this as false <useSecurity>true</useSecurity>

restart jenkins

It will ask for initial admin password --> sudo cat /var/lib/jenkins/secrets/initialAdminPassword

Create username and password

### Q) How to change the port of the jenkins ?

go to /var/lib/jenkins in config.xml change the port to 8082

restart jenkins

use IP:new-port ex: IP:8082

**Assignment :**

1. Multi-branch pipeline ?

2. upstream and downstream jobs?