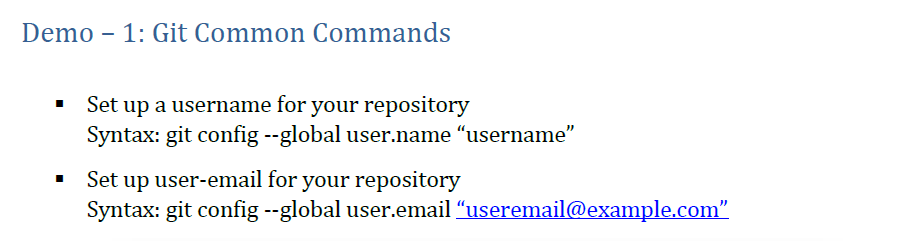
**Centos Image on AWS t2.large 40 GB**

**/ProComputers CentOS-7.9-x86\_64-Minimal-8GiB-HVM-20220707\_053634-20255663-8070-4ad8-85c6-4d4f796c9265**

Command to install git:

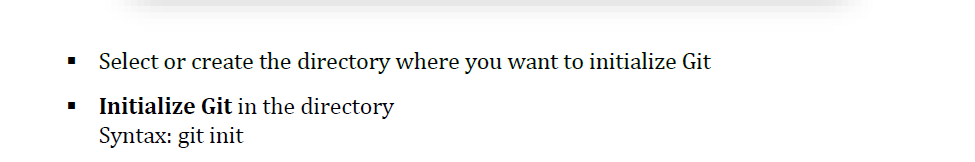
Syntax: yum install git

git --version



Example: git config – global user.name “Omer Hussain”

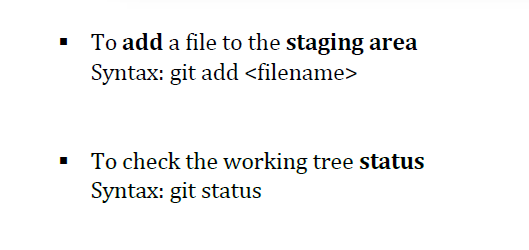
git config –global user.email [abbadi998@gmail.com](mailto:abbadi998@gmail.com)



Example: mkdir demo

cd demo

git init



Example: touch demo.txt

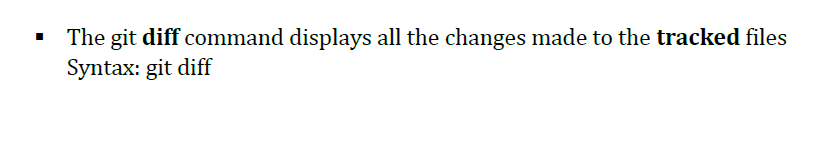
git add demo.txt

git status

**To Commit the stage files to your local repository:**

**Syntax: git commit**

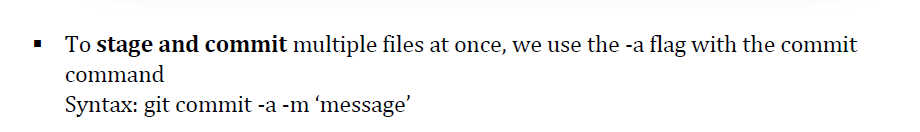
**Example: git commit -m “ this is my new commit” demo.txt**



**Example: vi demo.txt**

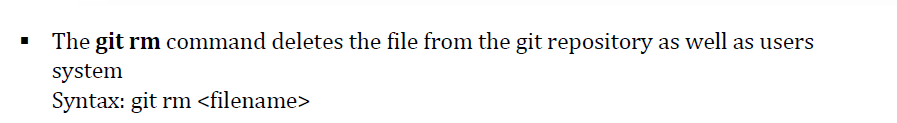
**Add some line (press ctrl c , shift :wq!)**

**git diff**

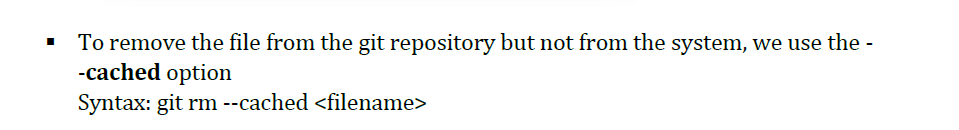


**Example: touch demo2.txt**

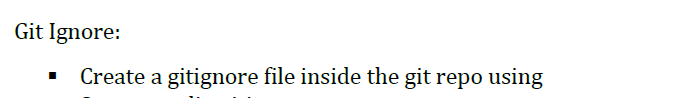
**git commit -a -m “new File added”**



**Example git rm demo.txt**

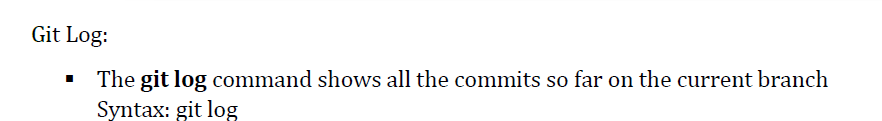


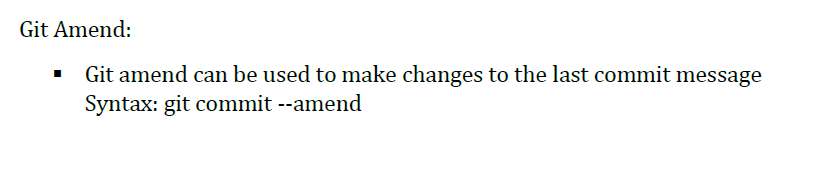
**Example git rm –cached demo1.txt**

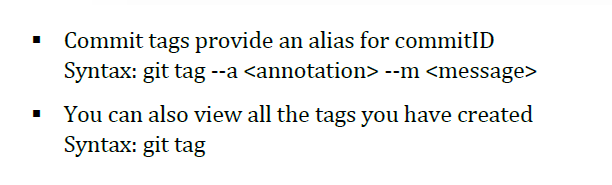


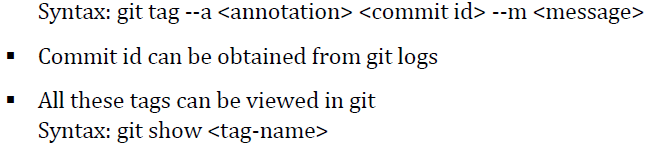
**Example: vi .gitignore**

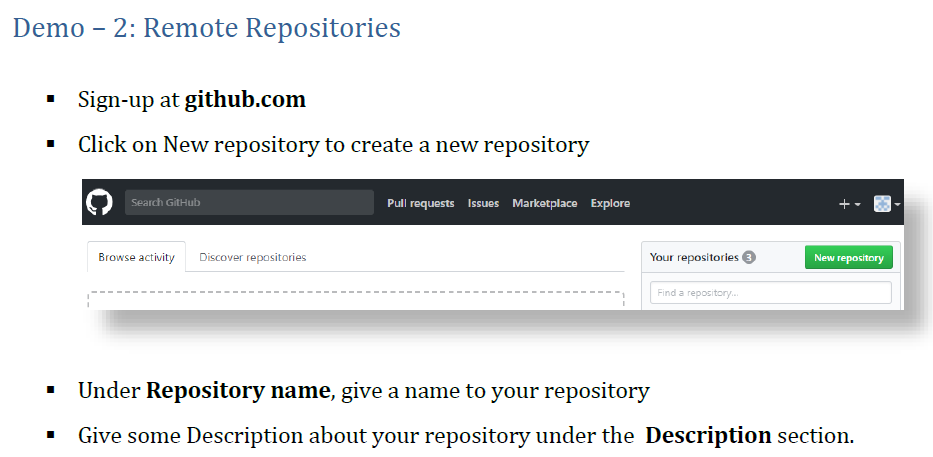
**\*.txt (ctrl+c , shift :wq!)**

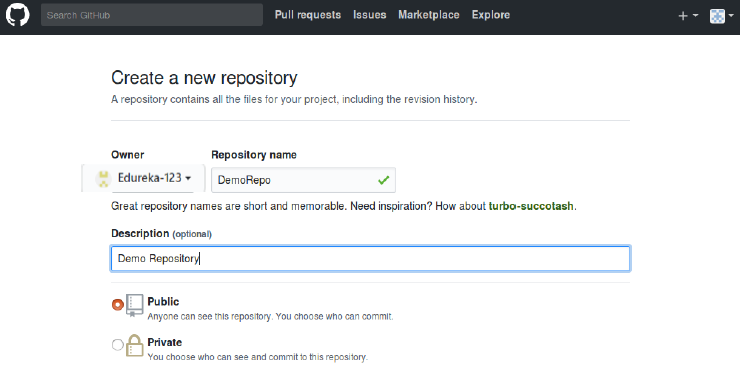


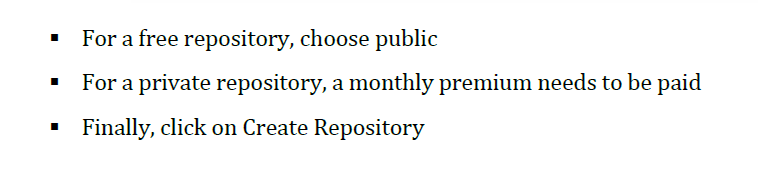


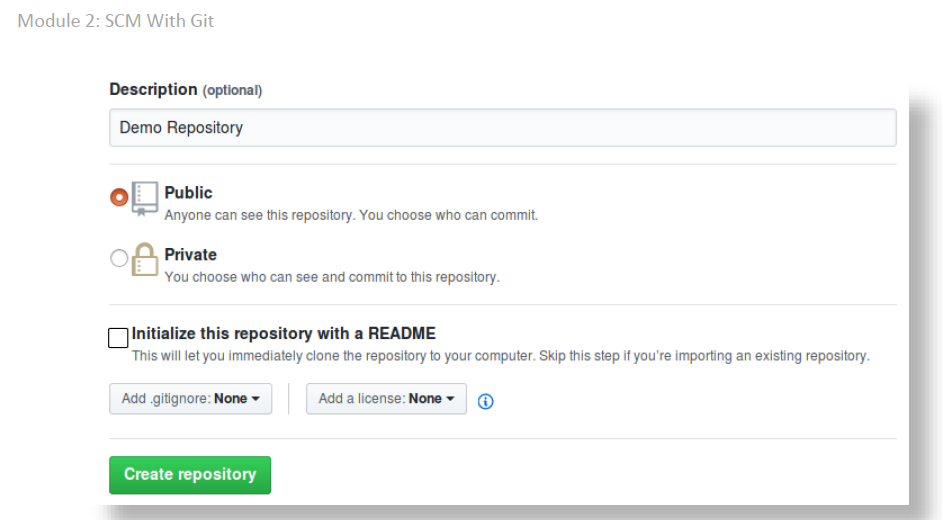


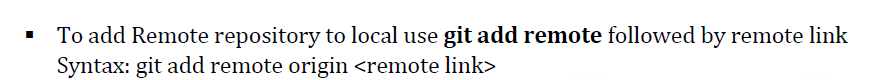


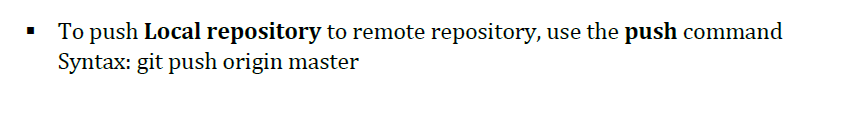


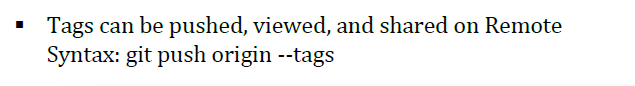


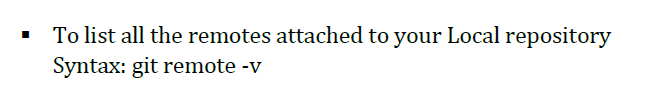


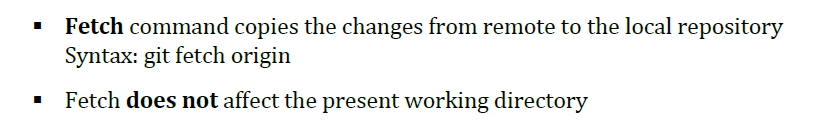








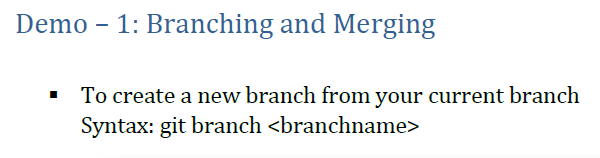


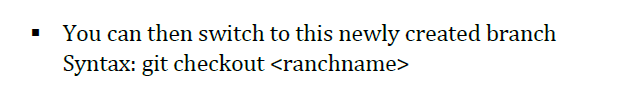


**Pull copies changes from remote to local repository**

**It then merges the changes with the present working directory**

**Syntax: git pull origin**

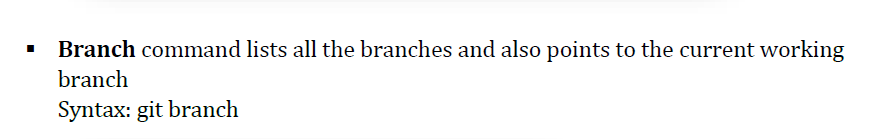


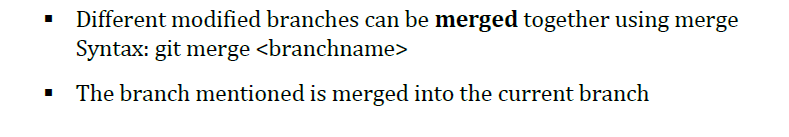


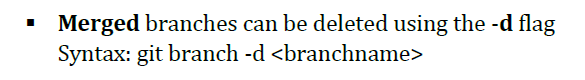
**Typo: git checkout branchname**

**Creating and switching to a new branch can be done by using -b flag**

**Syntax: git checkout -b branchname**



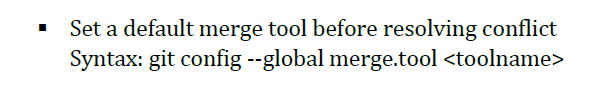




**Umerged: branches can be deleted usind -D flag**

**Syntax: git branch -D branchname**



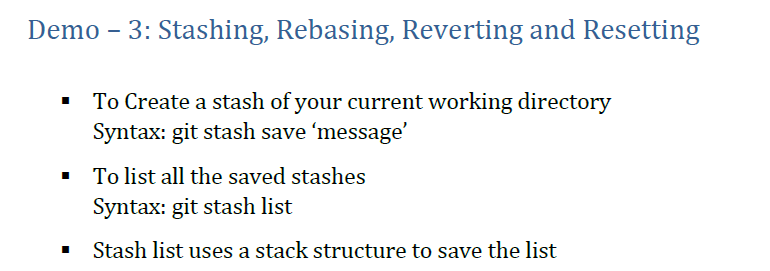


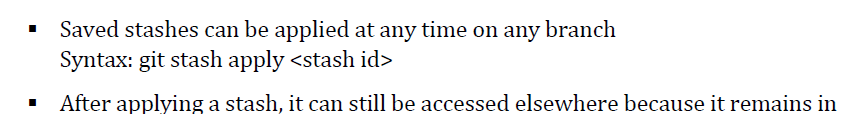
**Merge-tool automatically detects the conflicts and displays them**

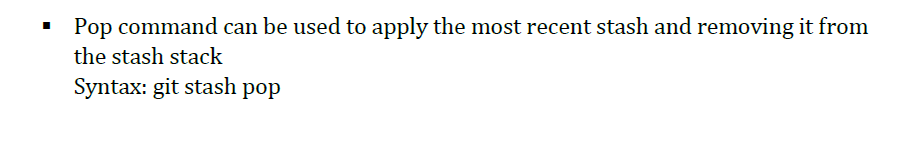
**Syntax: git mergetool**

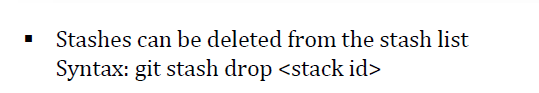
**commit the resolve changes**

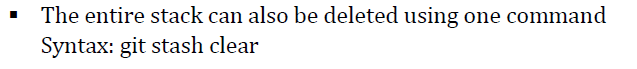
**merge the branch again**

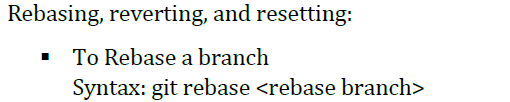


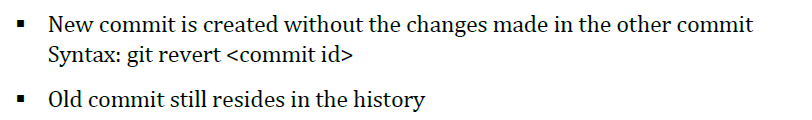
 **Stash**

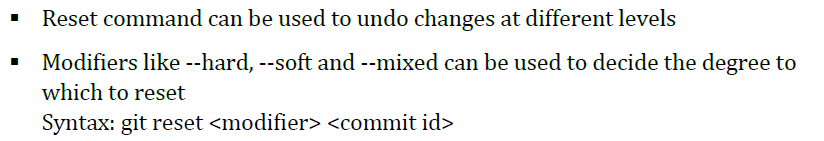












**Jenkins Installation:**

**sudo yum -y install java-11-openjdk**

**curl --silent --location http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo | sudo tee /etc/yum.repos.d/jenkins.repo**

**sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key**

**sudo yum install jenkins**

**sudo systemctl start jenkins**

**sudo systemctl status jenkins**

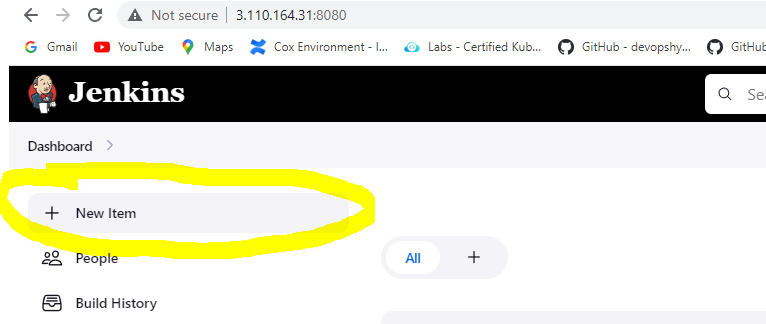
**sudo cat /var/lib/jenkins/secrets/initialAdminPassword**

**Access Jenkins via browser :** [**http://publicIP:8080**](http://publicIP:8080)

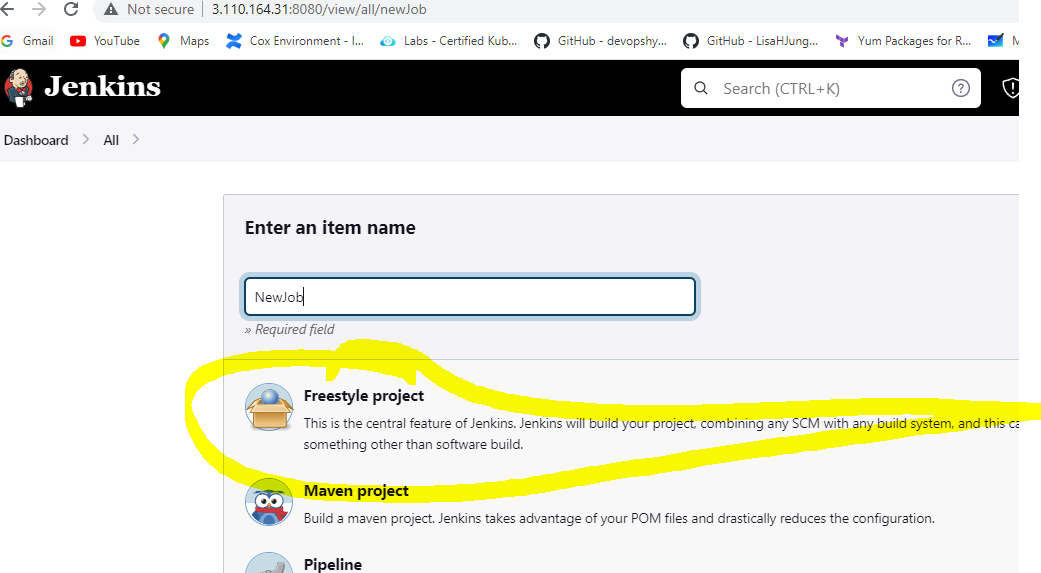
**How to create a sample job in Jenkins**

**To create new project/Item/Job**

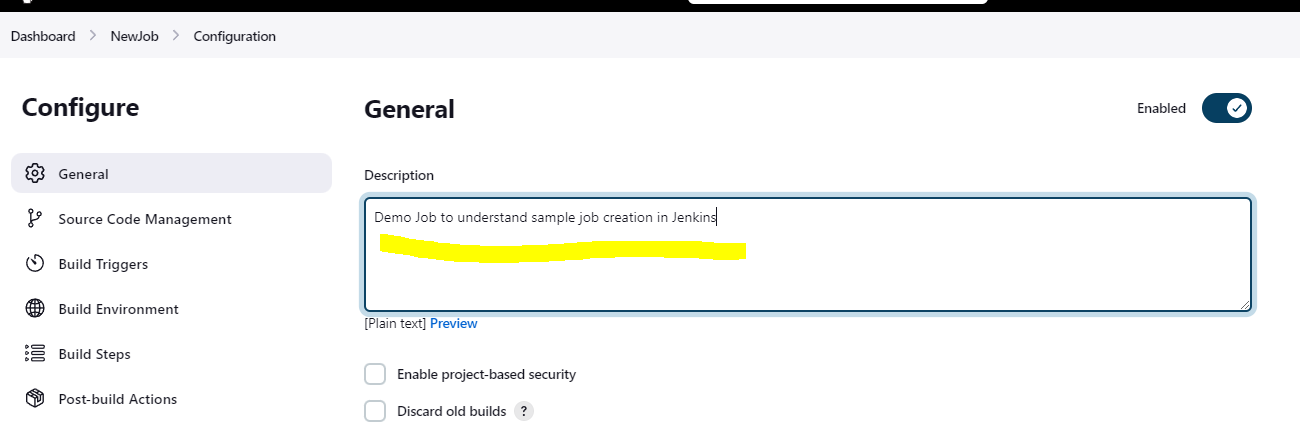
**Click on new Item as showing below**



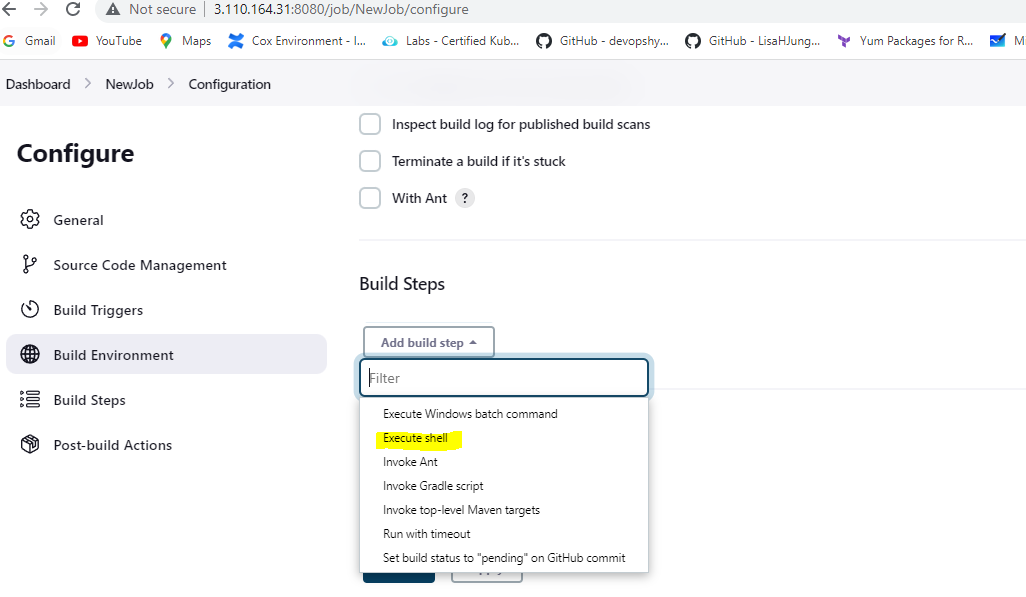
**Enter the job name as Newjob and select Freestyle project as shown in the below figure and click ok**



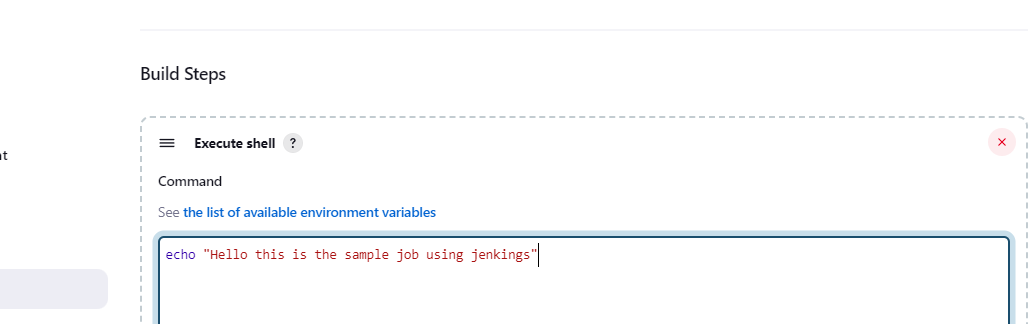
**Enter the description**



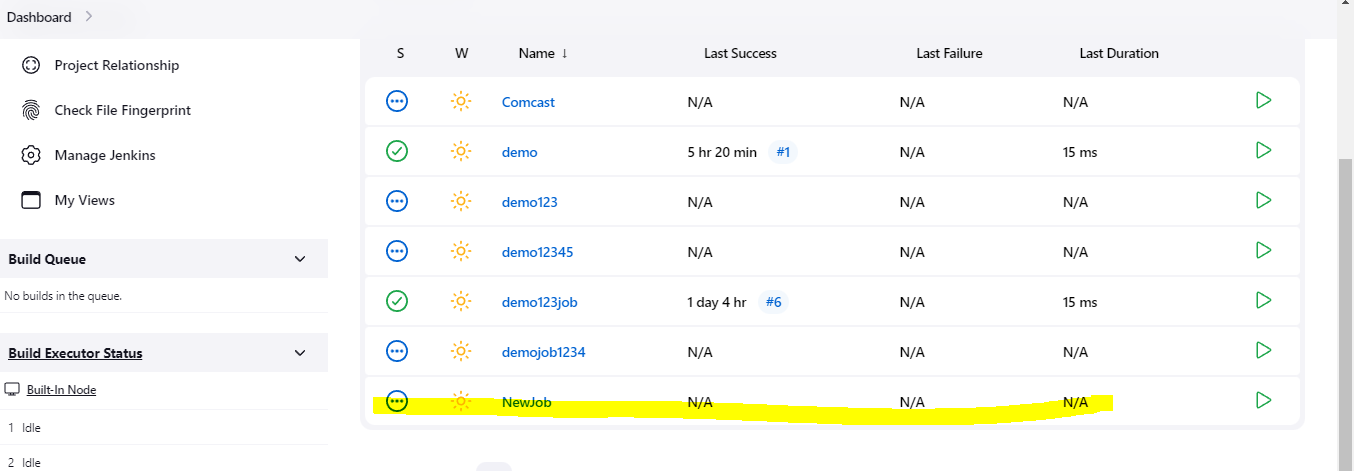
**Click on BuildStep and click on add built step and select Execute shell as shown in the figure below**



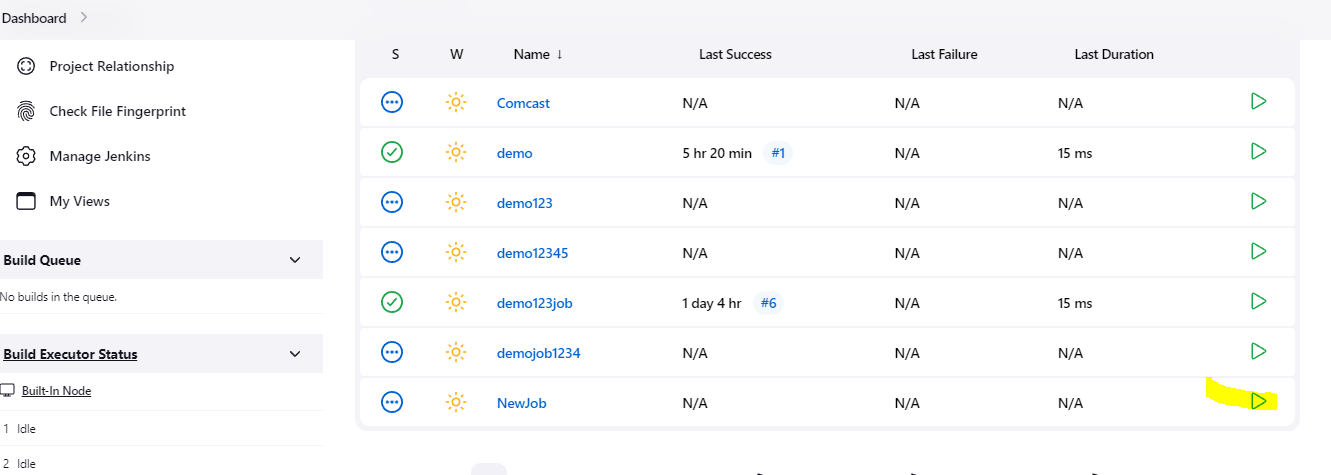
**Enter any linux command as below we are echoing the statement and click apply**



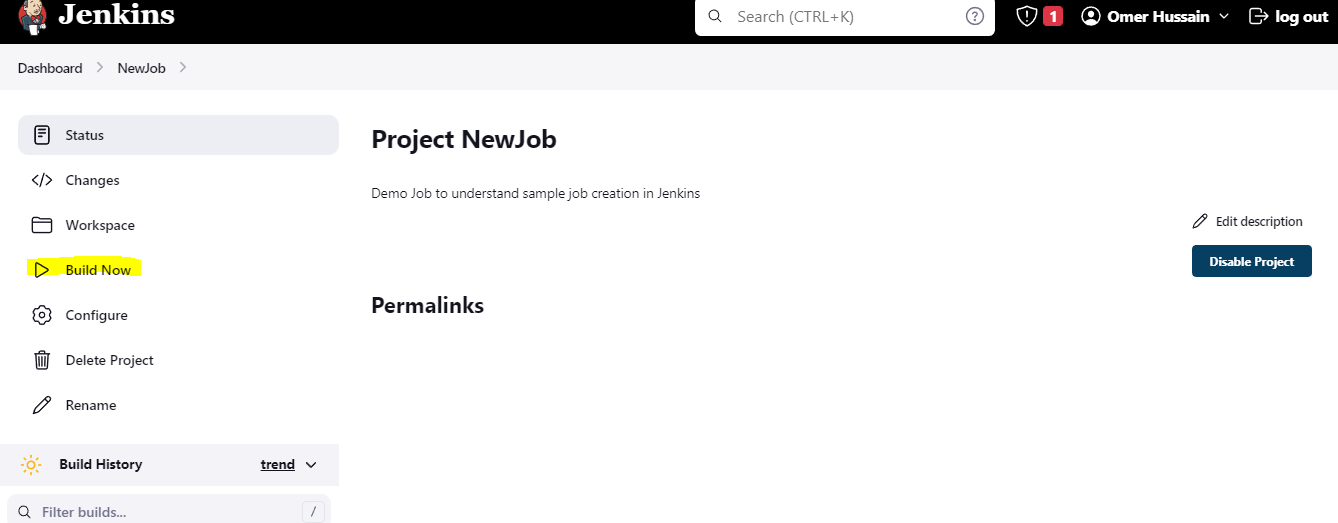
**Go to dashboard to see all the jobs created and you will be able to see the job which we created just now as shown in the figure. The Column S will show the status if the job is executed or not. The blue color signifies that the job is not executed, and green signal signifies the job is executed. The column W will let you know if the job failed any time. You would see cloud symbol if the job failed and sun symbol if the job is successful. Other Columns are self-explanatory as NAME, Last success, Last failure and Last duration of the job**



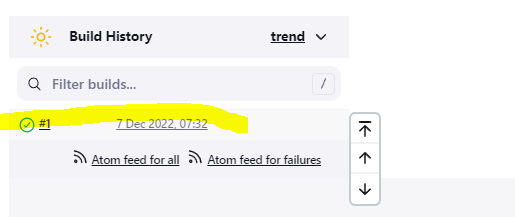
**Either Click on the green button as highlighted and shown below to run the job**

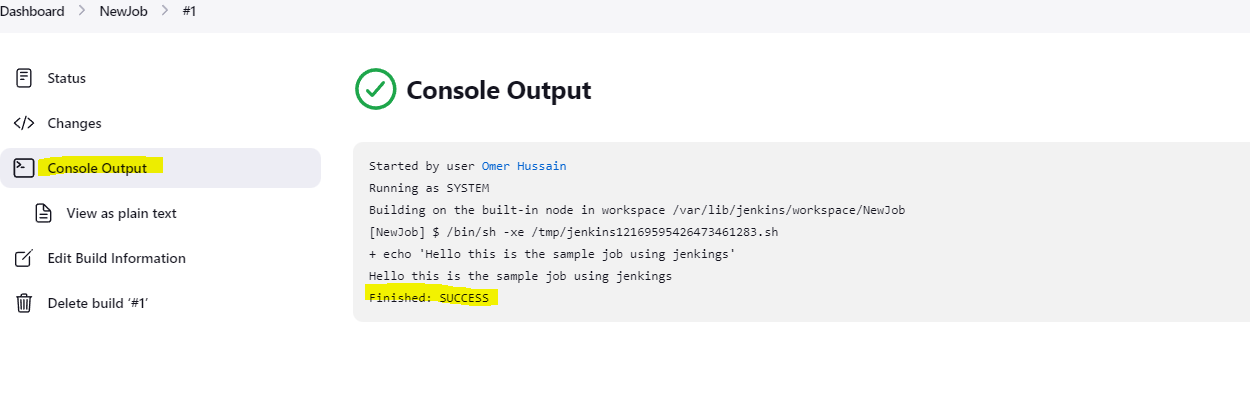


**Or click on the Name of the job and it will land u on this page and click on build now as highlighted in the image below:**

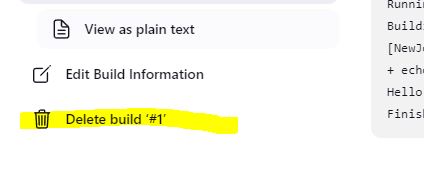


**Once the job is build you will see the job under Build history as highlighted in the image below. Click on the job number and it will take to the next page and then click on the console to see the output as highlighted in the second Image.**

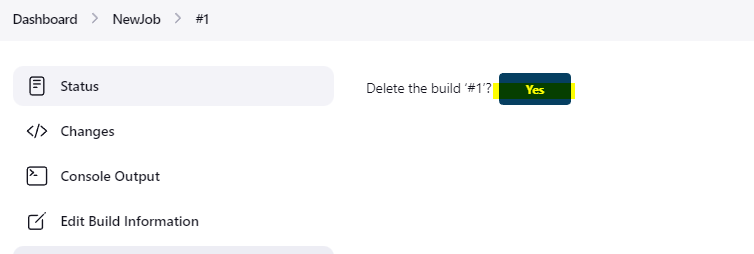




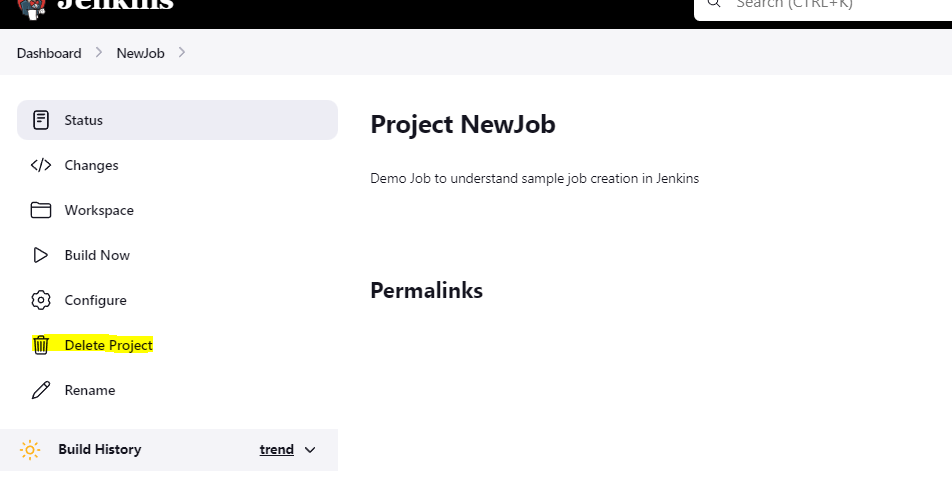
**To delete the build click on the Delete build option as highlighted in the image below:**



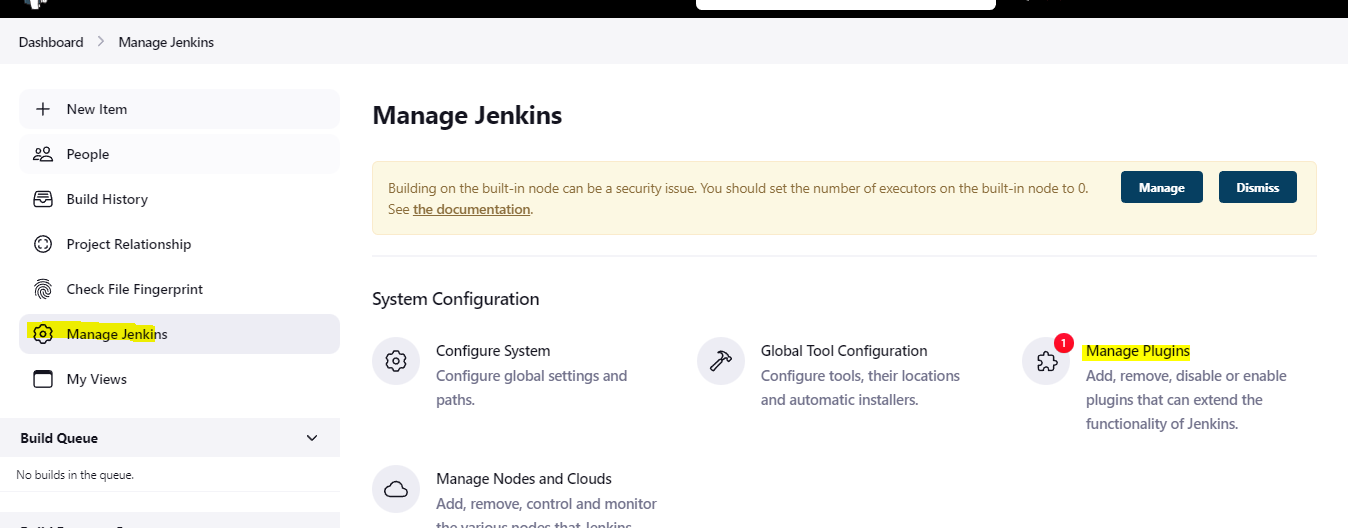
**Click yes it will delete the build for you as shown in the image**



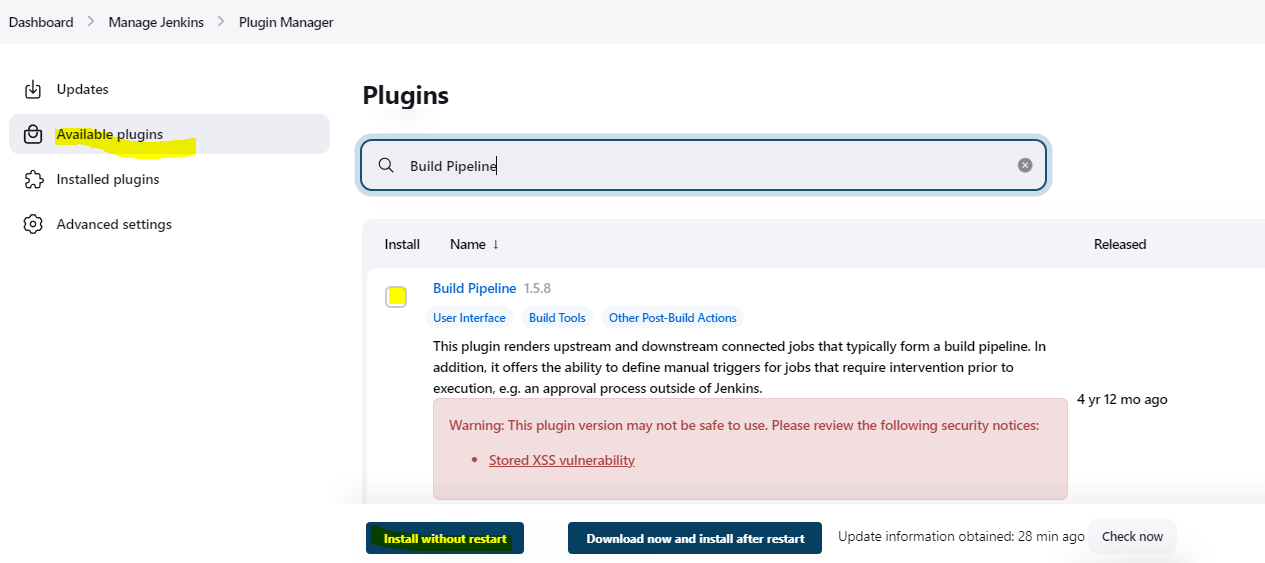
**To delete the Job/Project/Item go to the name of the project after clicking on the project name it will land to the Image as shown in the picture below and click on delete project to delete the project by selecting delete project option**



**To install any plugin click on Manage Jenkins options and then click on manage plugin as shown in the image below:**

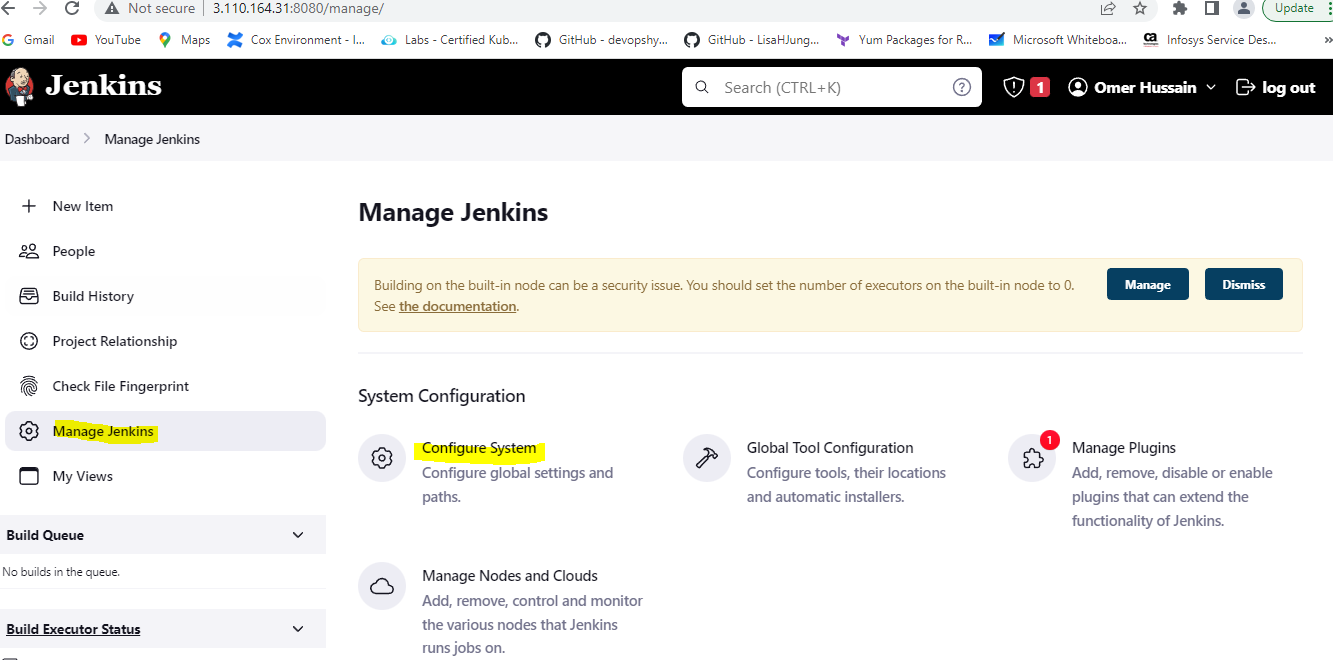


**Select Available plugin search for plugin you want to install select the checkbox beside the plugin name and install without restart and the plugin will get installed on jenkins**

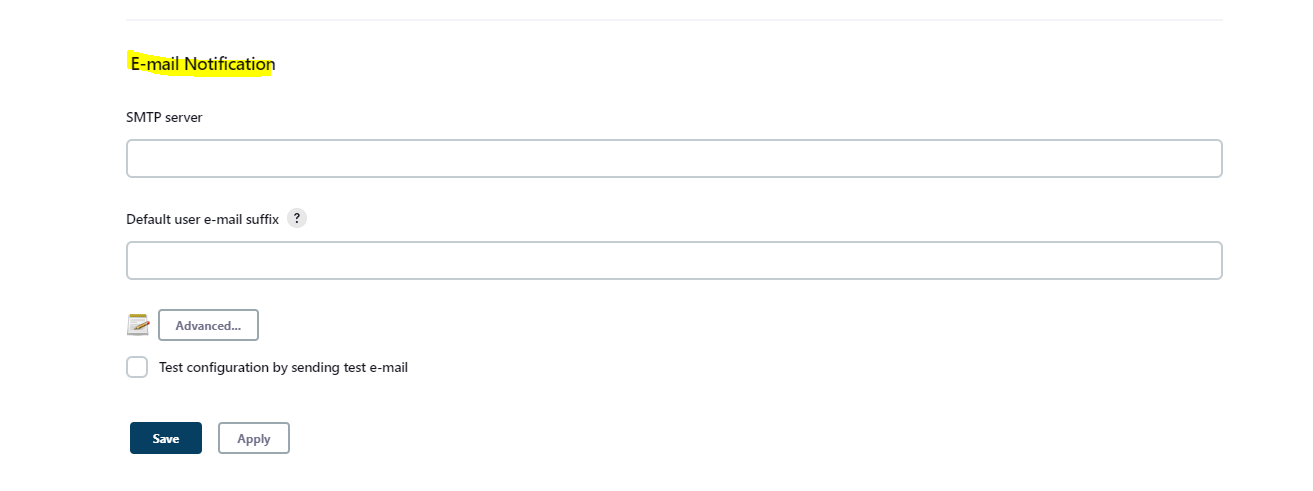


**To setup Email notification on Jenkins**

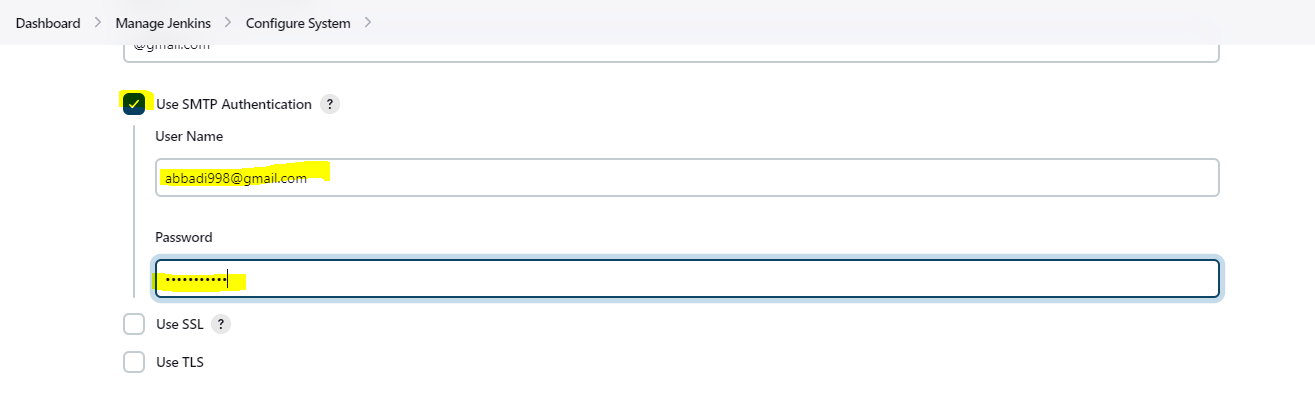
**Click on Manage Jenkins and then click on configure system as shown in the image**



**Scroll down until you see Email notification:**



**To set up gmail as ur email service update the details as below click on advance to see all the options and select the smtp authentication to enter the gmail and the Application password (Generate it as mentioned below)**



**Check the checkbox beside test configuration by sending test e-mail option to check if we are able to receive email and enter your personal gmail**



**Note: By default gmail will block any app accessing the email if you want it work have to make changes as below**

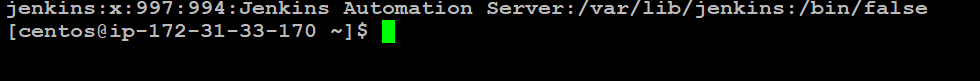
1. **Obtain application specific password**
   * sing-in to gmail account >> navigate to settings >> privacy and security settings
   * setup two step verification settings (because without two step verification we cannot generate application specific password)
   * after setting up two step verification setting in gmail account navigate back to security and privacy settings
   * click on application specific password >> give the name of the application in the drop down as Jenkins (google by default does not have any specific application password setting for Jenkins) >> this will generate password note down the password generated

**Configuring Master and Slave node on Jenkins**

**Groundwork:**

**On Master Node (Where Jenkins is installed)**

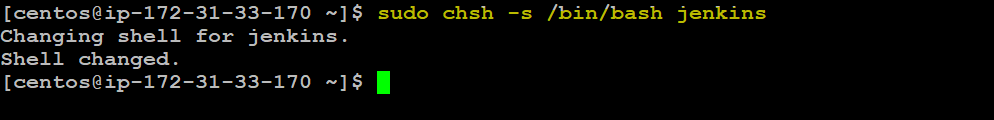
**By default, the Jenkins user will be created after we installed Jenkins on the server**



**As you can see in the above image the Jenkins server created after we installed Jenkins on master machine doesn’t have the login shell as highlighted in the image shown below**



**To establish the password less authentication between master and slave node. We should be having Jenkins user on both master and slave machine. As on the master machine we have Jenkins User which is service user we must enable the login for this user to generate the key and copy it to the slave machine. Below is the command to enable the login shell in the Jenkins user in the master node.**



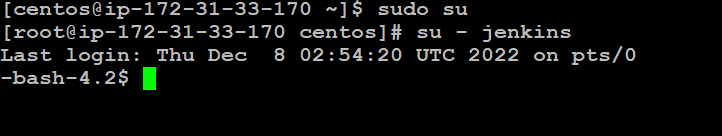
**As shown in the image above we have enabled the login shell in Jenkin user running on master node**

**Command: sudo chsh -s /bin/bash jenkins**

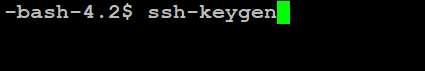
**After we have enabled the login shell in order to login to the Jenkins user running on master machine we use the below command**

**Command: sudo su (to change as the root user in ec2 instance doesn’t need to provide support of root)**

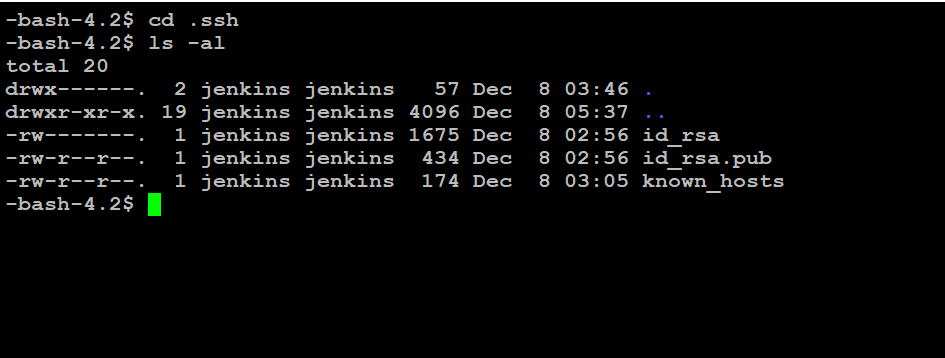
**su – Jenkins**



**Use the below command to generate keypair on the master node**



**Once the above command is executed we will check in .ssh folder to see if the private and public key got generated. As shown in the image below the key id\_rsa(private key) and id\_rsa.pub (Public key). Now we have to copy the public key to the slave node but before that we have to create the Jenkins user on the slave node**

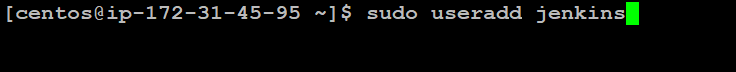


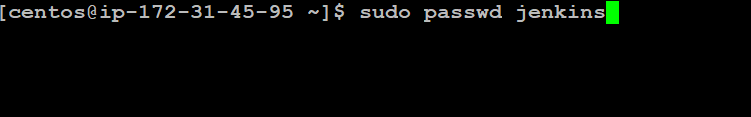
**Slave Node:**

**Create the Jenkins user on the slave node by running the below command as shown in the image**

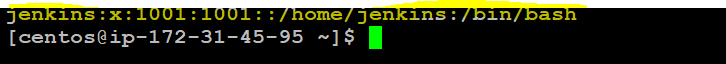
**Command: sudo user add jenkins**

**sudo passwd Jenkins**

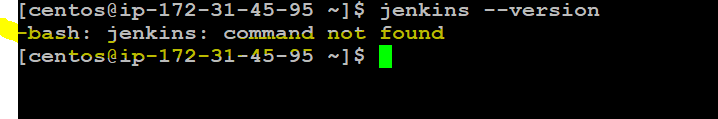


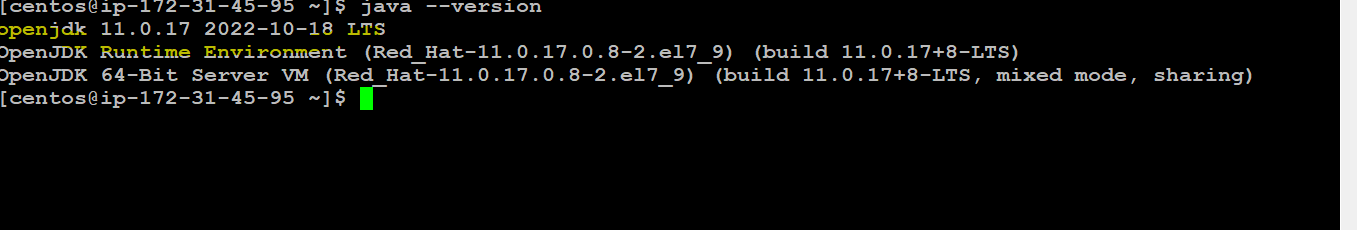


**After these above command executed we will have Jenkins user on the slave machine as shown in the image below**



**We don’t require any Jenkins software running on slave node only dependencies we need is java 11.**





**Master node:**

**Now as we have created Jenkins user on the slave machine. we must copy the public key from master to slave machine. In order to achieve the above task, we use the below command.**

**Master node Command: sudo su**

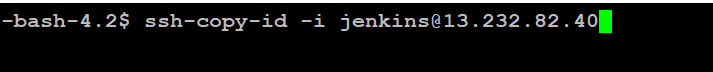
**su – jenkins**

**ssh -copy-id -I jenkins@PublicIPOfTheSlaveNode**

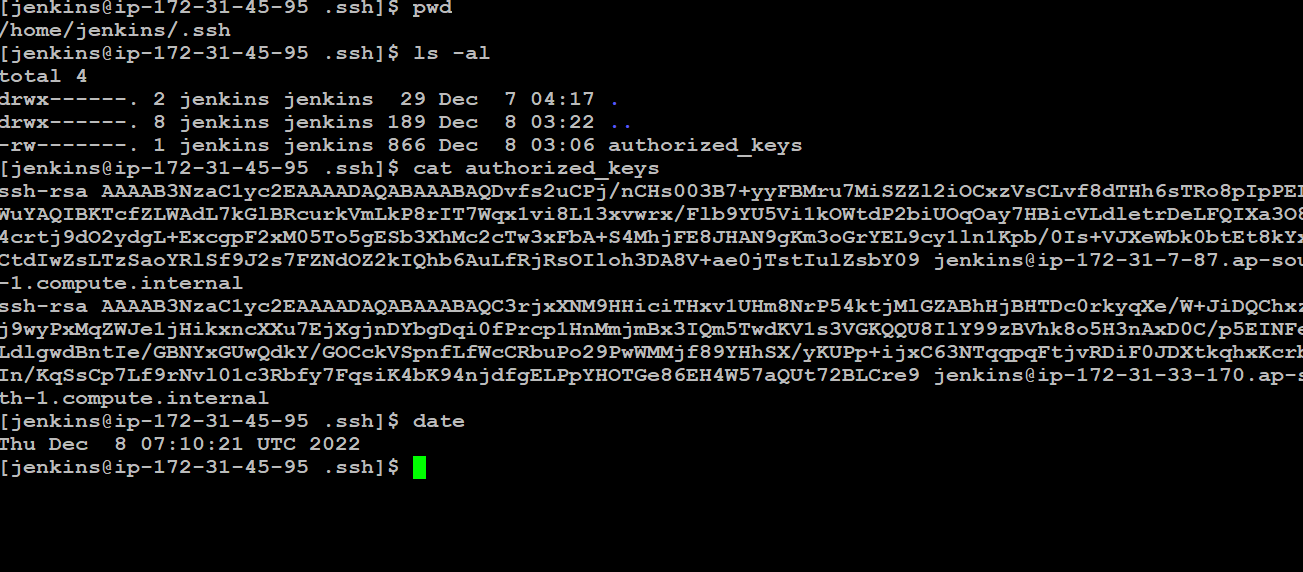
**Once the above command is get executed it will prompt you to enter the password enter the password, we chose for slave node and it will copy the public key of master node onto the slave node.**

**As you shown in the images below.**

**Master node**

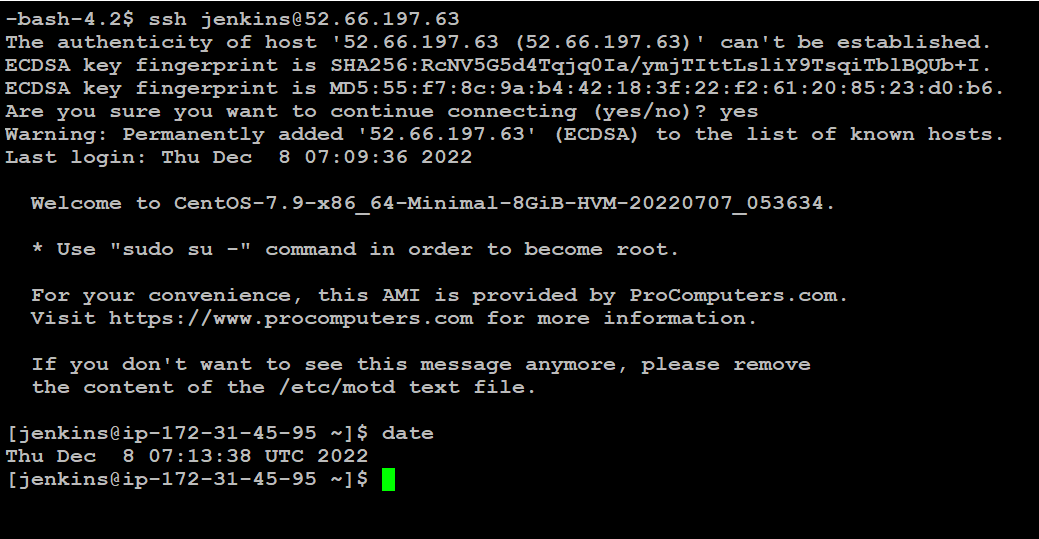


**Slave node**



**Verify if we have password less login from master node to slave node by running the below command from master node**

**Command: ssh jenkins@PublicIpOfSlaveNode**



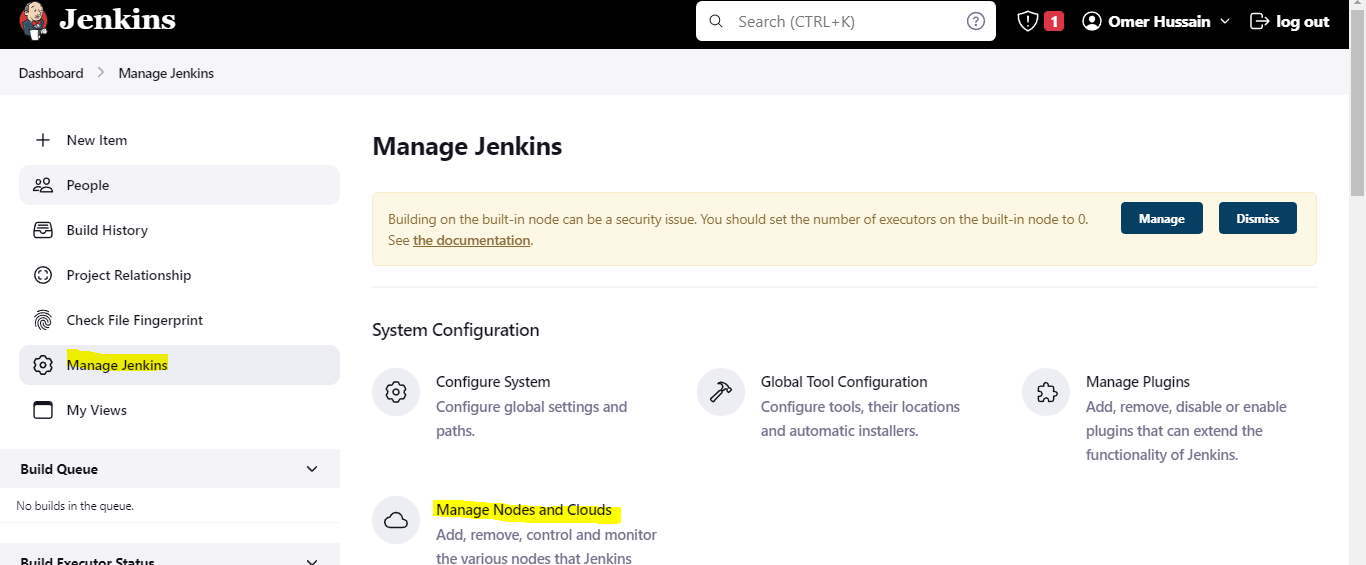
**Adding Slave node in Jenkins Steps**

**To add the slave node on Jenkins steps as follows:**

**Login to Jenkins URL**

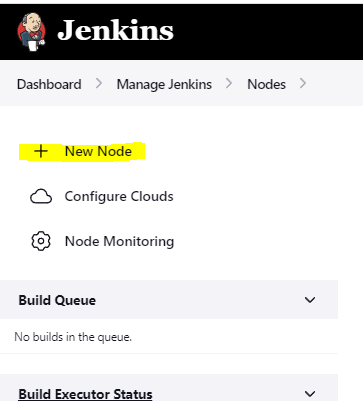
**Click on Manage Jenkins**

**Click on Manage Node and Clouds in Jenkins as highlighted in the image below**

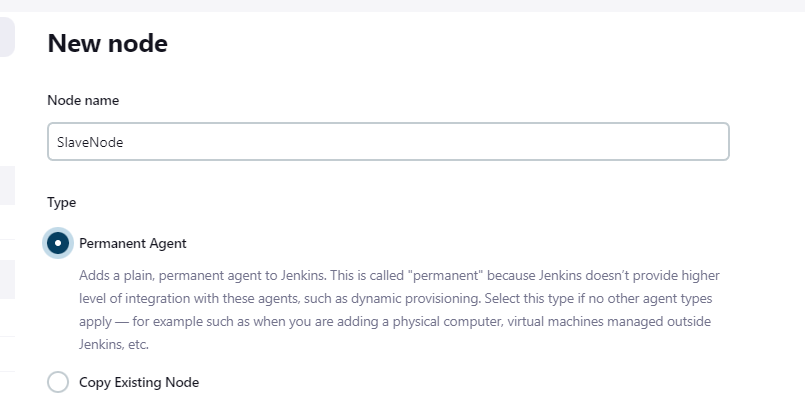


**Click on New Node to add the SlaveNode as highlighted in the image below**

**Using Maven as a continuos**



**Give the name to the node as below**

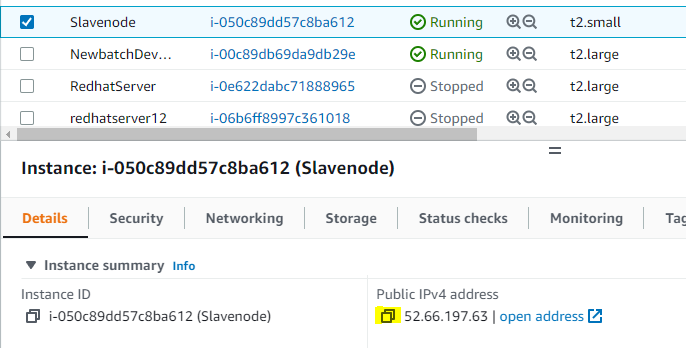


**Enter the details as shown in the image below**

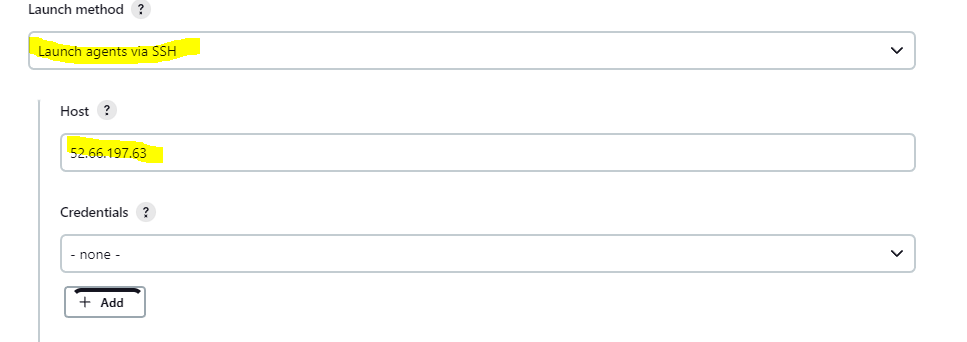


**Remote root directory is the directory of the slave node . Whenever we create any user by default it will in the /home directory with the username so as the Jenkins directory i.e. /home/jenkins**

**Copy the public IP of the Slave node from aws console and enter the details as shown in the image**

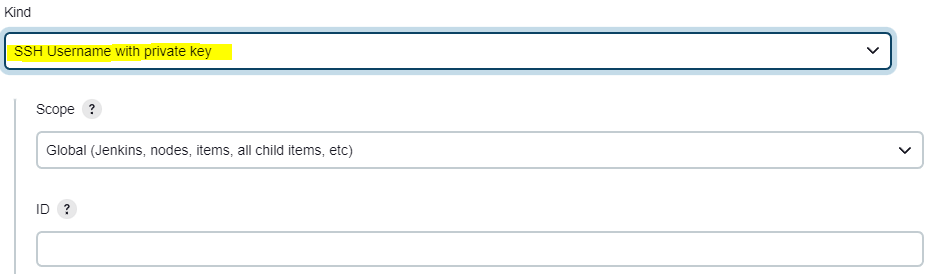


**Select the launch method as ssh and enter the PublicIP of slave node as below**

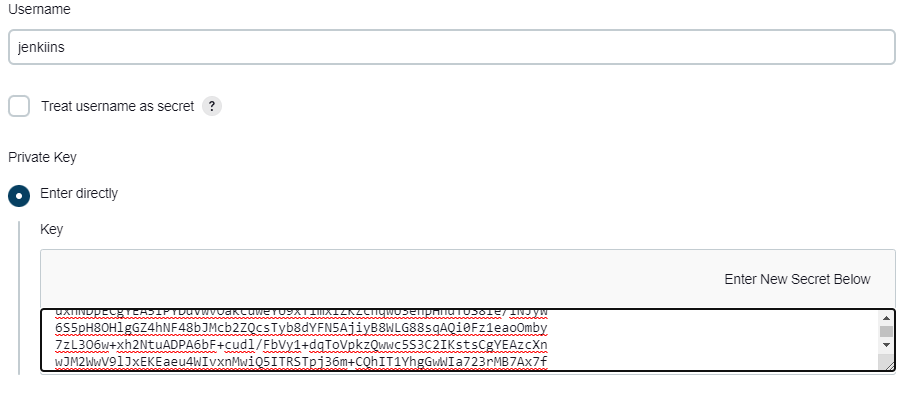


**Click on Add to add the credentials**

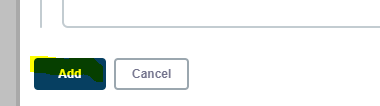
**Select the Kind as SSH username with private key as below**



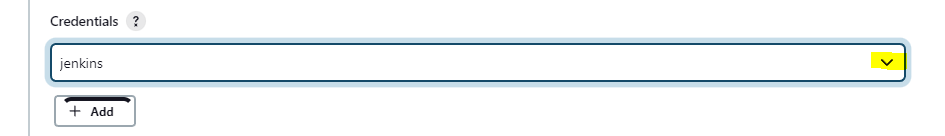
**Enter the username as Jenkins and copy the private key from master node and paste in the Jenkins as below**



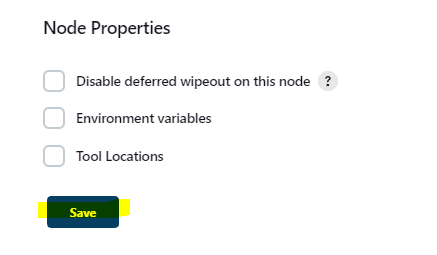
**Click on Add to add the credentials**



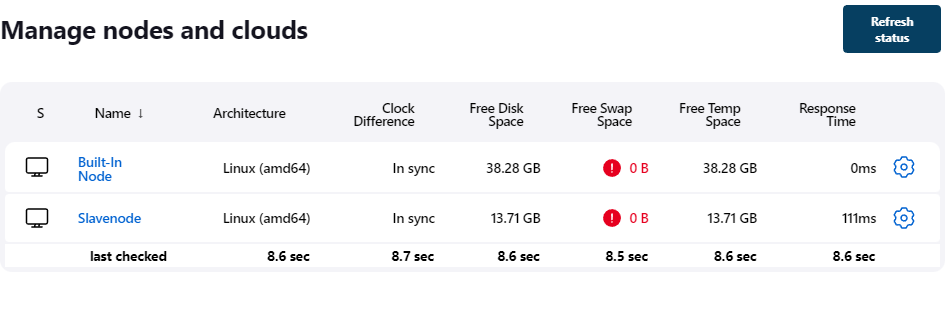
**Click on Jenkins credentials by clicking on the dropdown and option from credentials as shown below**



**Once done click on save to save the changes**



**Once everything is done I will be able to see the slave node by clicking the option manage Jenkins and then manage node as below**

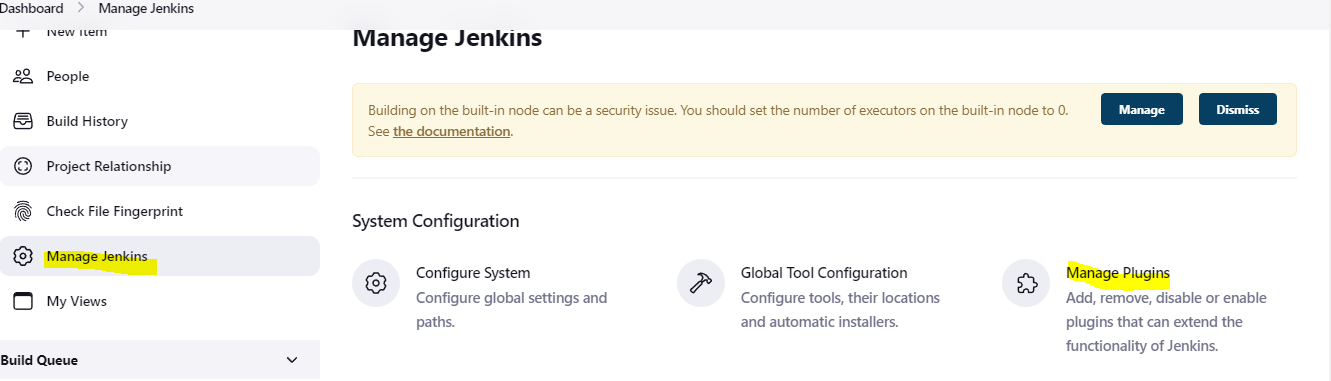


**Now I will be able to execute any job on the master and slave node by restricting its executing as below**

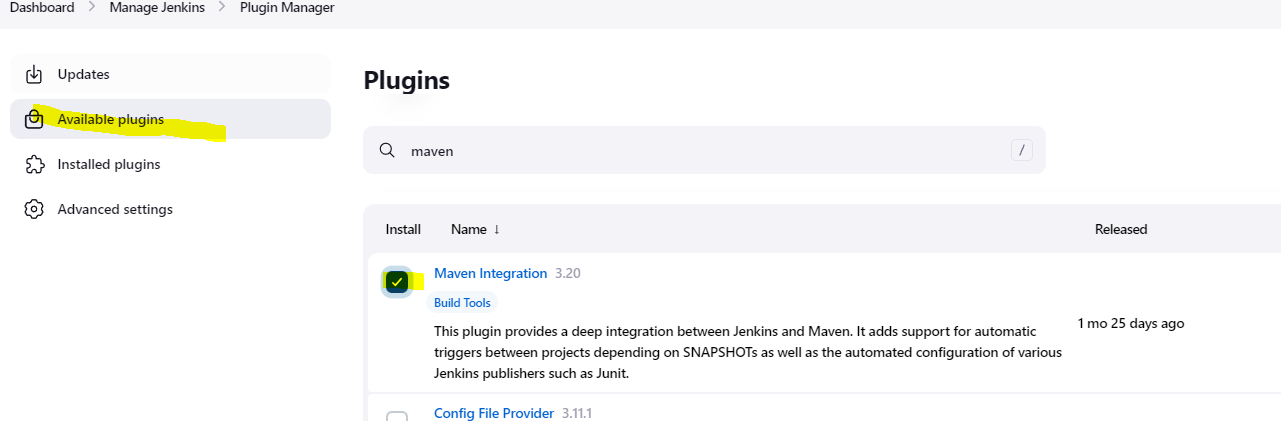


**Maven as a continuous Build tool with Jenkins**

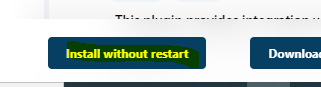
**Installing Maven as a build tool in Jenkins Click on the option Manage Jenkins and select option manage plugin as shown below**



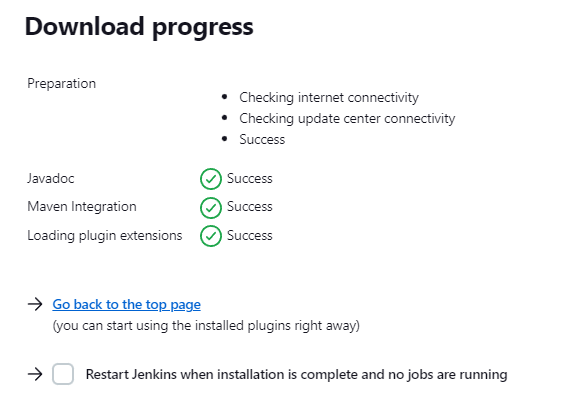
**Click on Available plugin and search for maven and add maven as shown below**



**Once done click on install without restart as below**

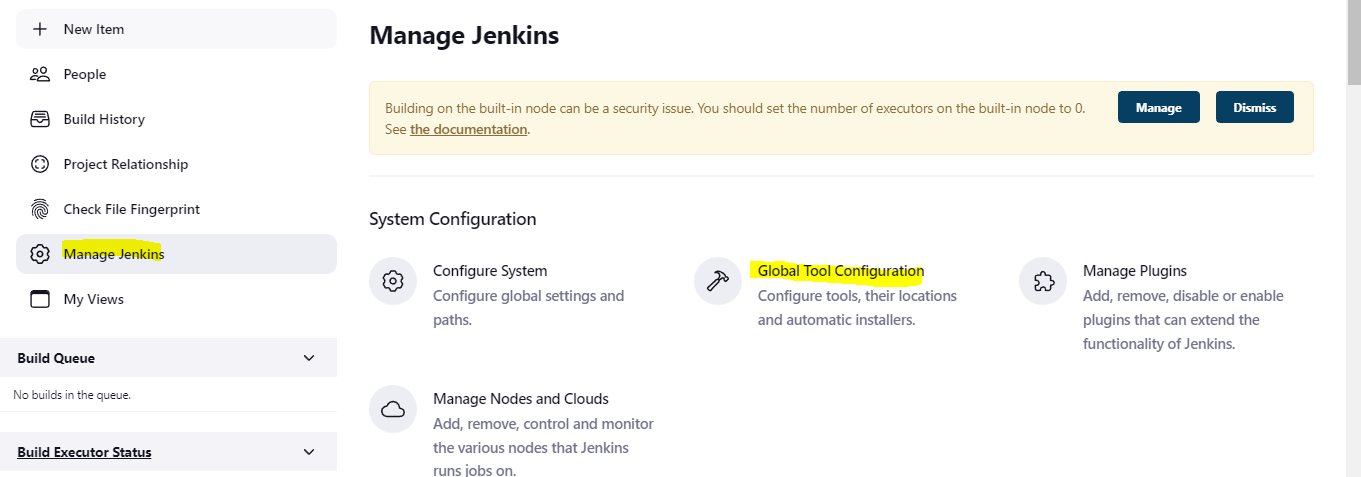


**You will get the output as below**

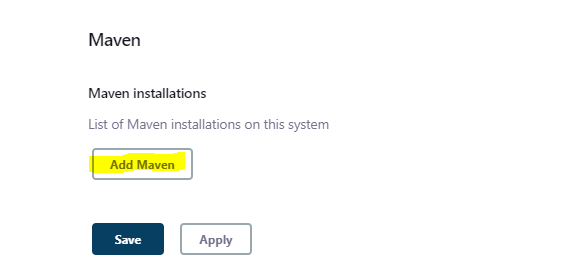


**Configuring Maven Plugin:**

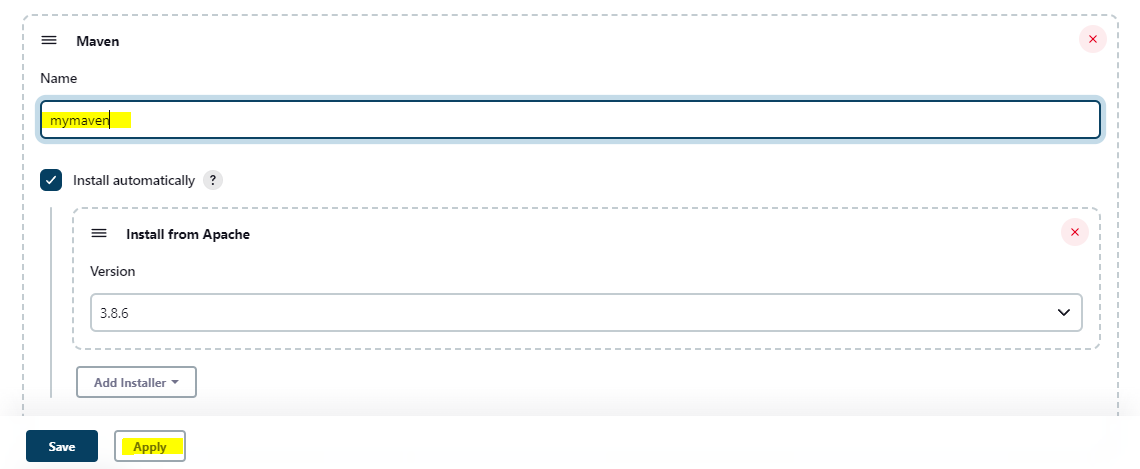
**Click on Manage Jenkins-Global Tool Configuration**



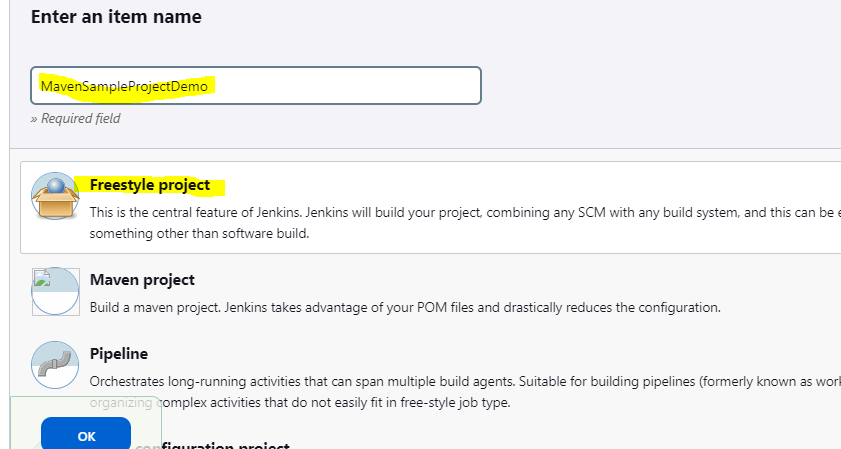
**Scroll Down go to Maven -Add Maven as shown below**



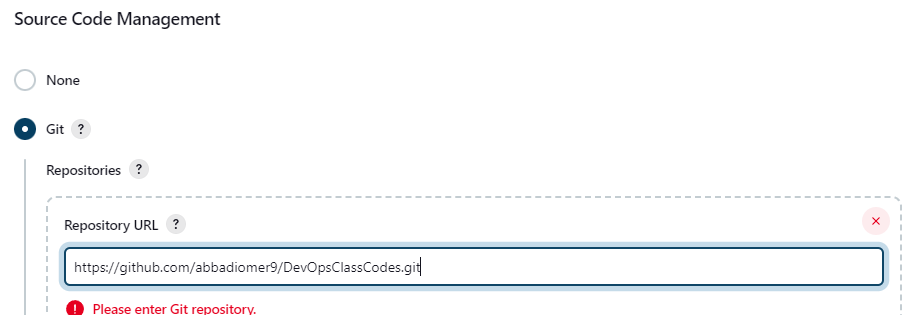
**Enter the name as mymaven(Name can be anything of your choice) and click apply as shown below:**



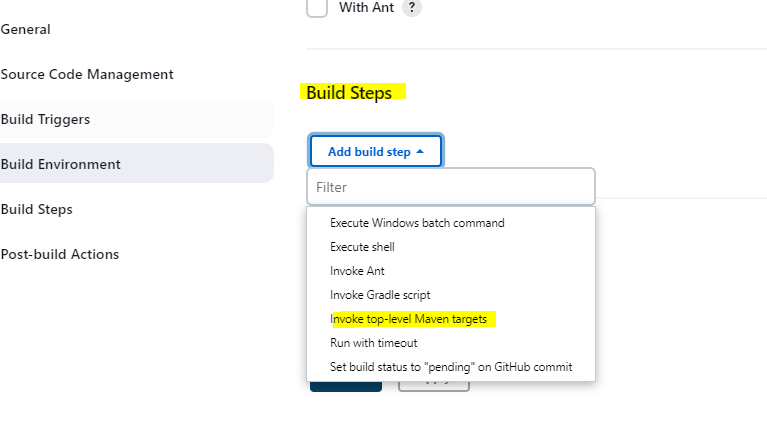
**Click on NewItem to create new build as we do for any project and enter the details as below**



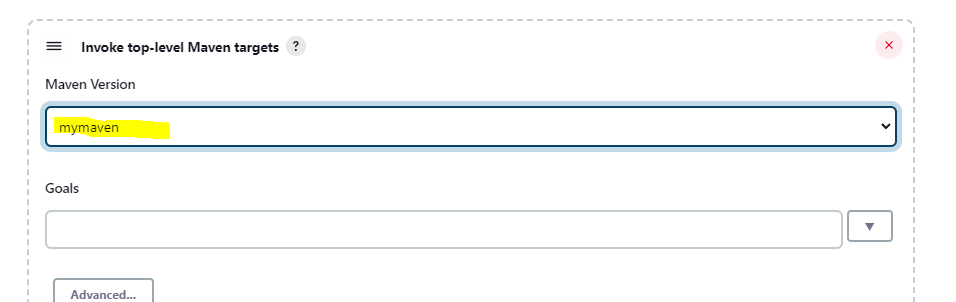
**Enter the description and select the github repository which has the java code as shown below**

**GITHUB URL: https://github.com/abbadiomer9/DevOpsClassCodes.git**

**Scroll down and under Build Step : Add build step select the option invoke-top-level Maven Targets**



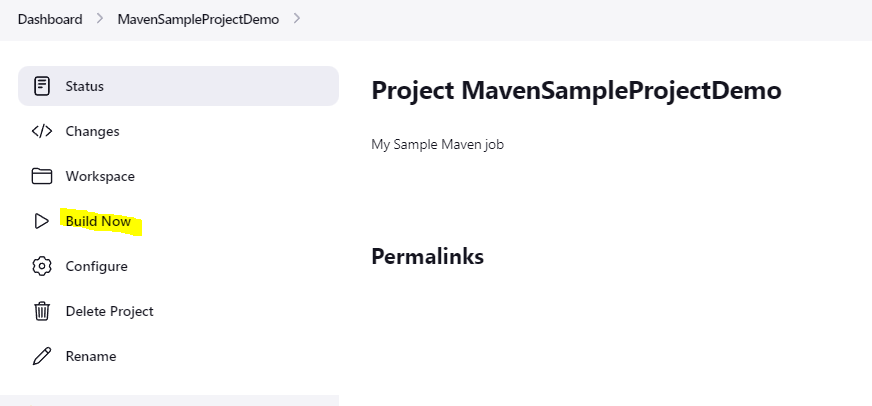
**Select the name of the maven we configured as shown in the image below**



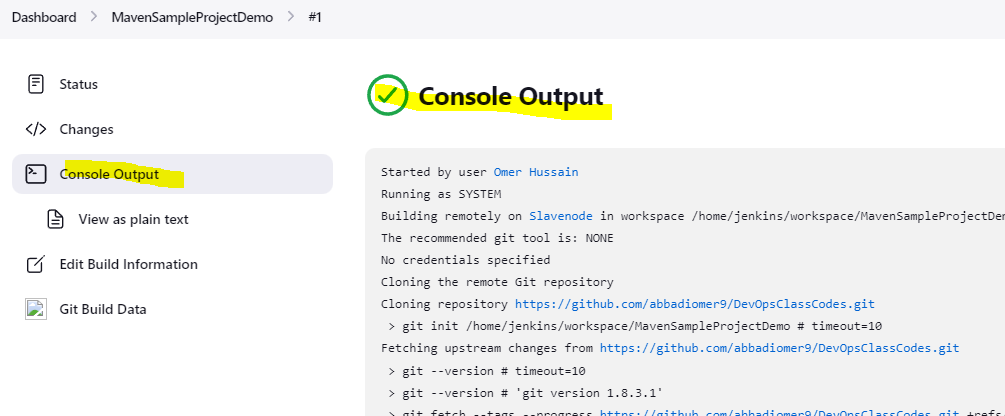
**Under Goals enter compile and click on apply to save as below**



**Go to the maven project created and click on build now as below**



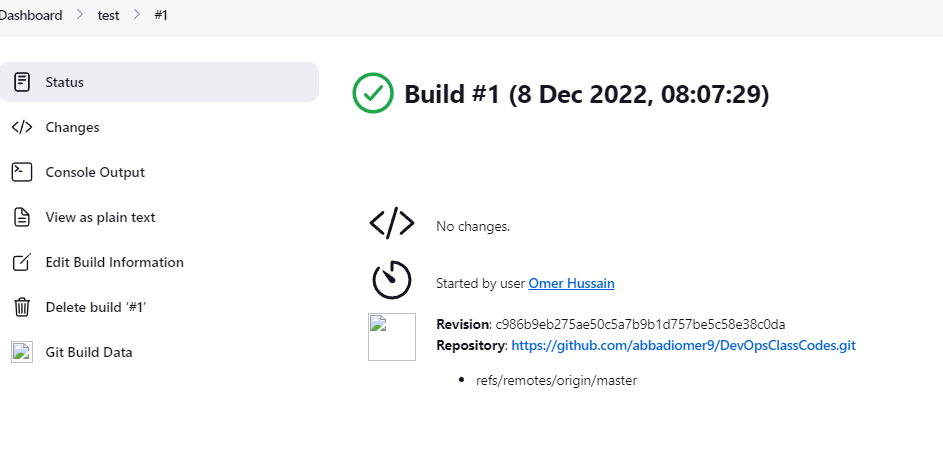
**Once the build is done go to console output to check the status of the build as below**



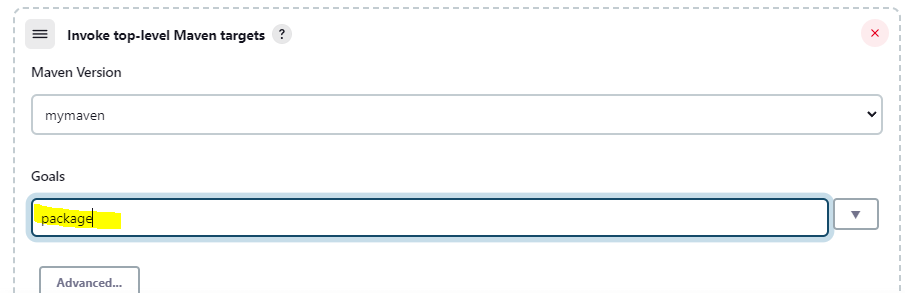
**IN the same way create two more item by the Name test, packaging and follow all the same steps but the goals should be different as shown in the image below.**

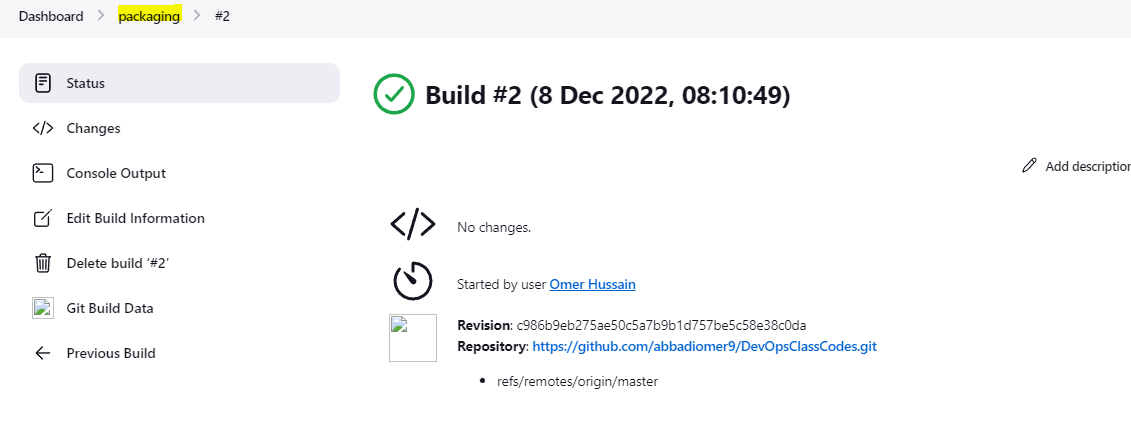
**TEST JOB**





**Packaging Job:**

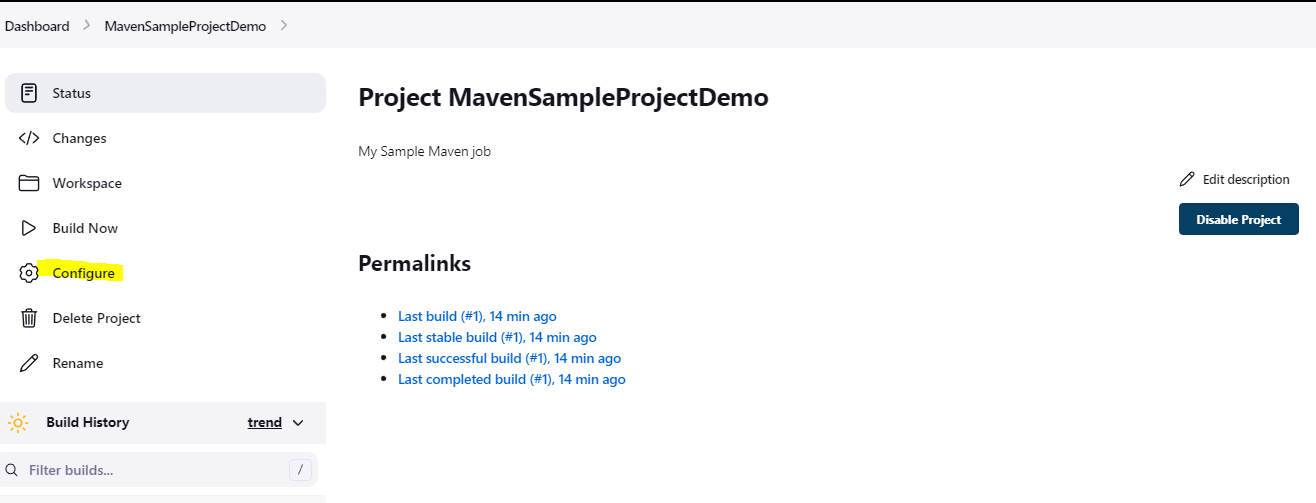




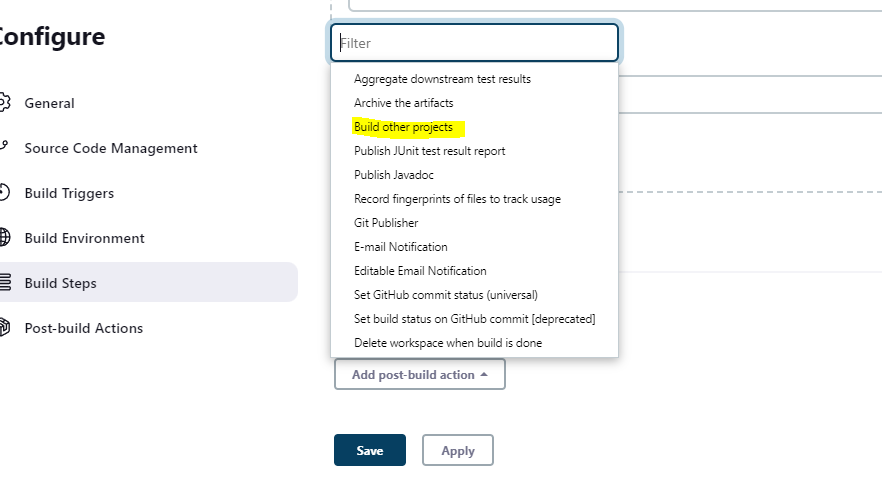
**Creating a Pipeline using the post build option:**

**We use the above three job created to create a pipeline using post build action as below**

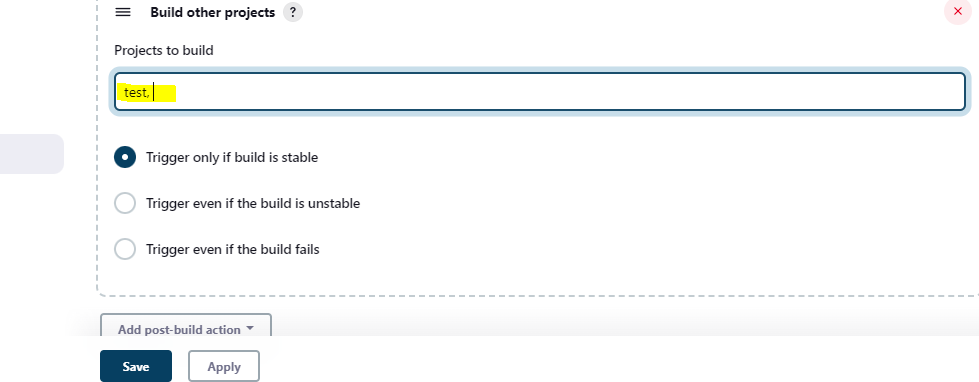
**Access the first job we created i.e. MavenSampleProjectDemo and click on configure as shown in the image**



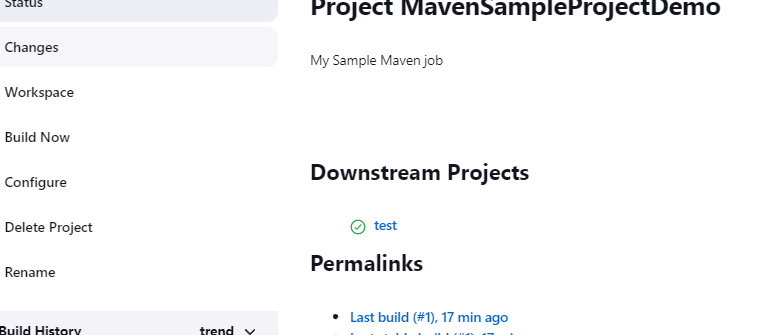
**Under Post build action add post build action and select build other projects as shown below**



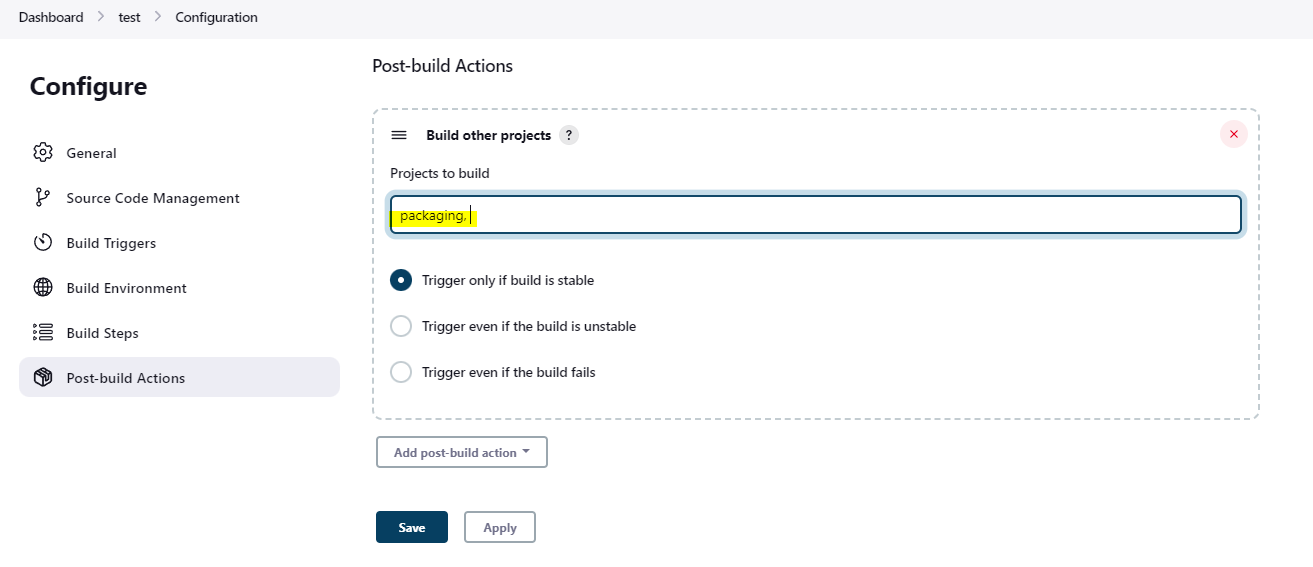
**Select test which is our second job and click apply to save the configuration**



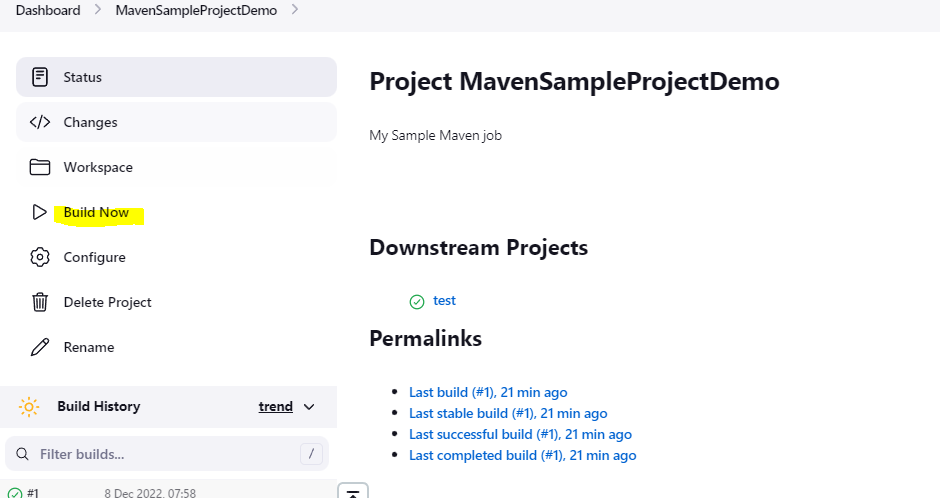
**You will see output something like below**

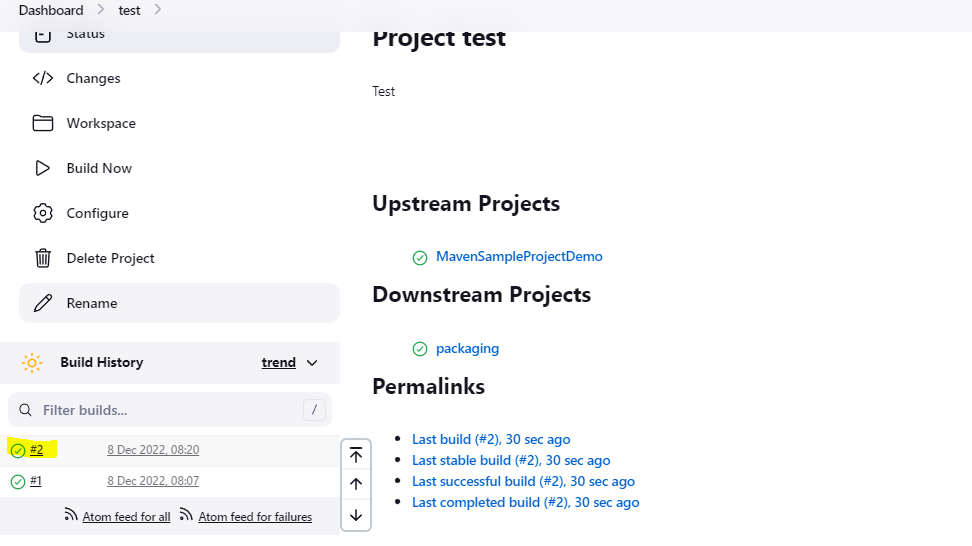


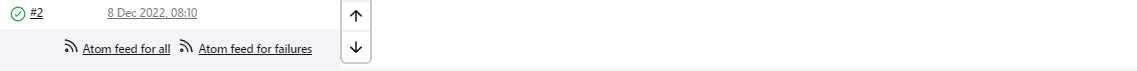
**Access our test project and follow the same step but this time select as packaging as post build action and click apply i.e. our third job which we have created as shown below.**



**So now if I manually build our first job i.e. MavenSampleProjectDemo other two jobs test and packaging will get build automatically which is our pipeline.**

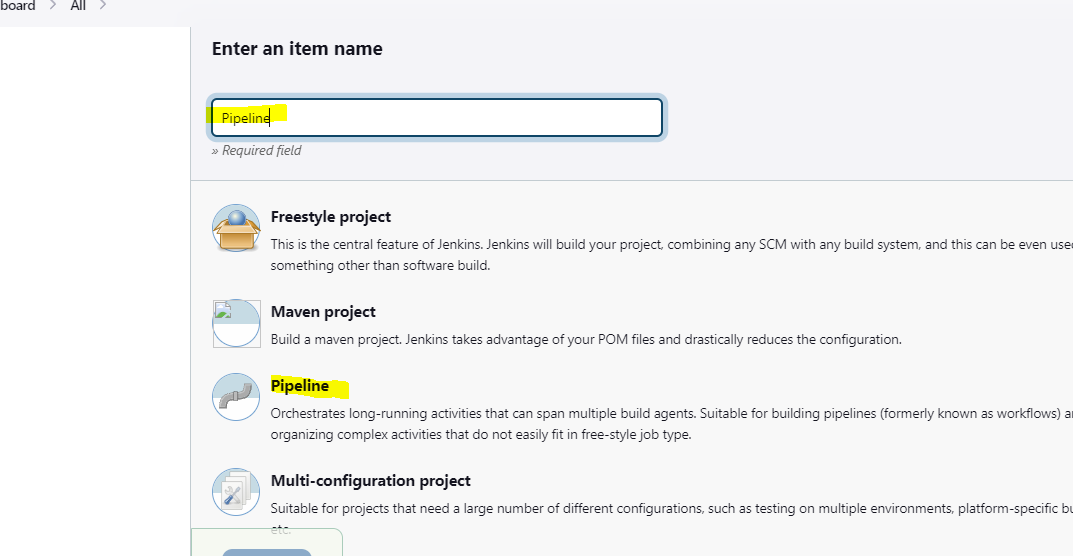




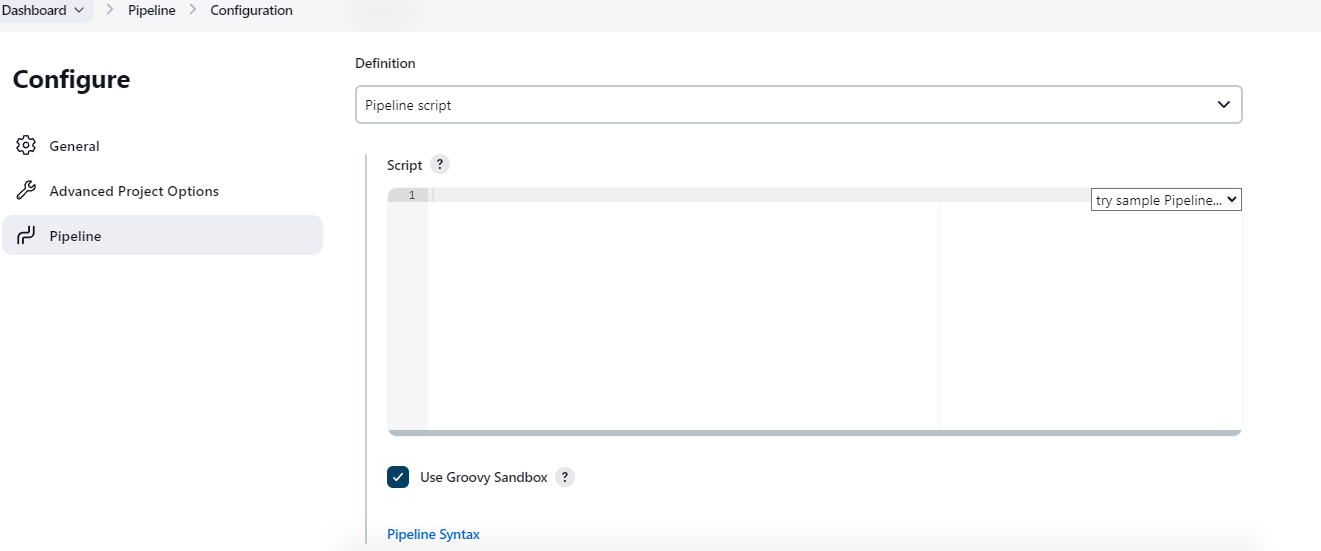


**The above result can also be achieved by code i.e., pipeline as a code as shown below**

**Click on NewItem and enter the project name as pipeline but this time we select pipeline option as shown below**



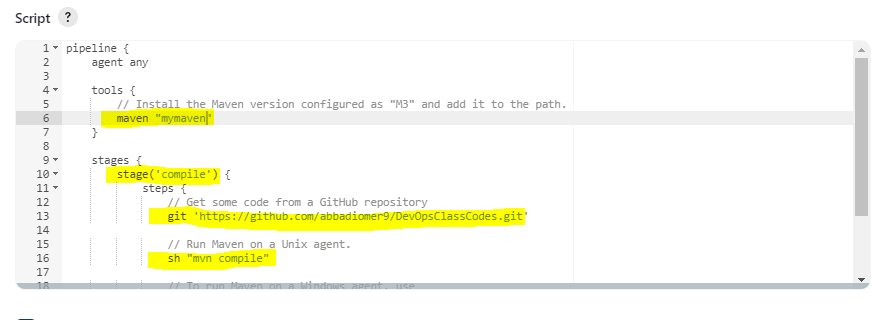
**Enter the description and scroll down to definition as shown below**



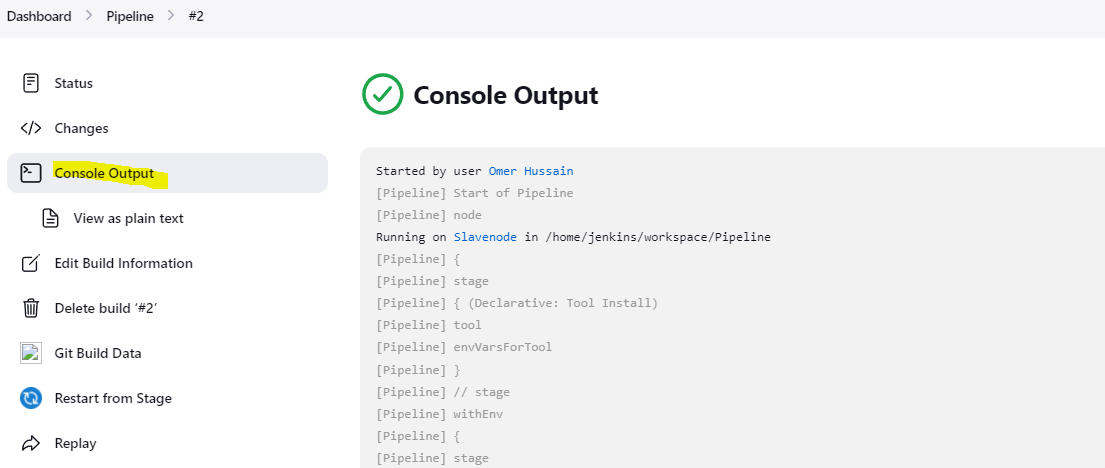
**Select the option from the drop-down menu which says github+maven as shown below. The script will get automatically populated**



**In the below example we made few changes we want to build our first job i.e., compile so we name the stage as compile, maven as mymaven replaced git hub url with our github code url and change the maven command to maven compile and click apply to save as shown below**



**Click on build now and check the output under console. You will see the job is build by using the code same way we can add test and packaging as well**



**Ansible**

**Installation**

**sudo yum install wget**

**wget https://dl.fedoraproject.org/pub/epel/7/x86\_64/Packages/e/epel-release-7-14.noarch.rpm**

**sudo rpm -ivh epel-release-7-11.noarch.rpm\sudo yum repolist**

**sudo yum-config-manager --enable epel**

**sudo yum install ansible**

**sudo useradd ansible (Same task must be perform on slave machine)**

**sudo passwd ansible(Same task must be perform on slave machine)**

**sudo vi /etc/sudoers (Same task must be perform on slave machine)**

**ansible ALL=(ALL) NOPASSWD: ALL**

**su – ansible (MasterNode)**

**ssh-keygen**

**ssh-copy-id -i ansible@PublicIPOfSlaveNode**

**Ansible Inventory: This is the file where we define all the slave/Remote/Target machine for ansible to perform tasks**

**Path: sudo vi /etc/ansible/hosts**

**To add group of slave nodes the syntax is as below**

**[demo] Note: Inside the bracket the name can be anything of your choice and its case sensitive below the groupname just enter the Public/PrivateIp if its in the same subnet/Fully Qualified Domain Name (FQDN) of our slave node**

**Example:**

**[demo]**

**52.66.197.63**

**Ansible Modules**

**Powerhouse to perform any task on ansible. In other words, it can also be called as function. All the ansible modules will get downloaded at the time of installation and it is written in python language. So, for any ansible module to get executed on the slave node only dependency we require is python must be > 2.7 running on the slave machine. As we know that we don’t require ansible running on the slave machine.**

**To check the modules on the Masternode we can use the below command:**

**Command: ansible-doc -l**

**Example of modules: yum, command, ping, git, copy, setup etc.**

**Ansible Playbook:**

**In a layman’s term we can call playbook as a glue that binds ansible modules, inventory. Playbooks are series of tasks that can be executed on the slave node.**

**Ansible Adhoc Command: Single task or One Liner**

**-m = Module**

**-a = Flag for the module**

**--become= To run the command as sudo or root user**

**-vvv(verbose)= To see the output in the terminal usually use to troubleshoot if the task is getting failed**

**Demo= Name of our GroupOfServer**

**ansible -m ping demo**

**ansible -m command -a uptime demo**

**ansible -m command -a “df -h” demo**

**ansible -m yum -a “name=httpd state=present” demo --become**

**ansible -m yum -a “name=httpd state=present” demo --become -vvv**

**ansible -m service -a “name=http state=started” demo –become**

**ansible -m copy -a “src=/tmp/a.txt dest=/tmp” demo**

**Sample Playbook**

**---**

**- hosts: demo**

**become: true**

**beocme\_user: root**

**tasks:**

**- name: install the package**

**yum: name=httpd state=present**

**- name: Start the service**

**service: name=httpd start=started**

**Save the file with an extension .yml**

**Example a.yml**

**To execute the playbook run the command as below:**

**ansible-playbook a.yml --syntax-check(to check the syntax)**

**ansible-playbook a.yml --diff –check(To dry run it will only show the changes but those changes will not be implemented on the slave node)**

**ansible-playbook a.yml –diff (To execute the playbook)**

**All the tasks will get executed as it is defined in the sequence.**