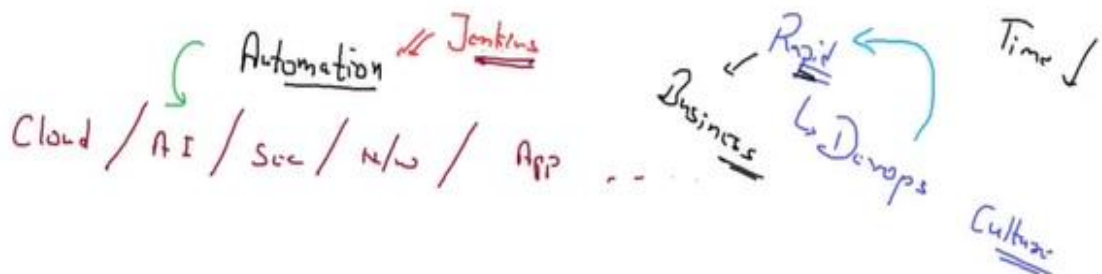




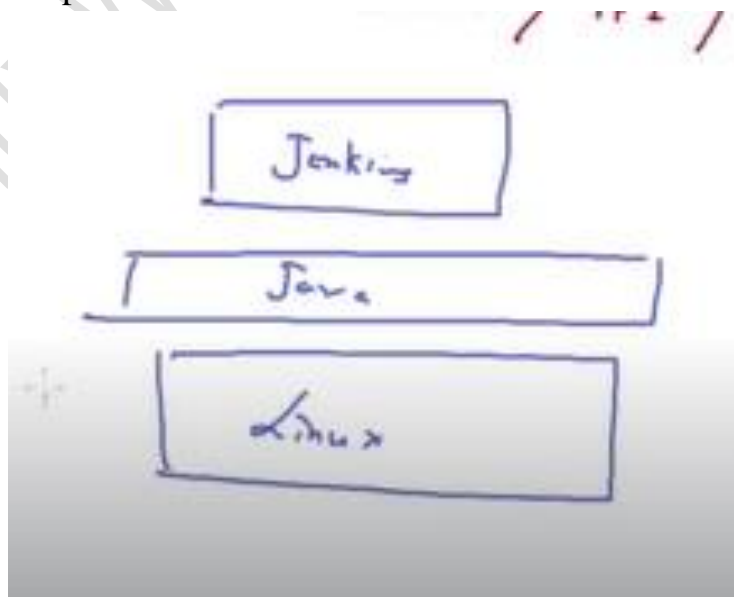
Jenkins Session No. 2

Summary 11-02-2024

- In the last class we install the Jenkins tool on the AWS cloud
- As we know Jenkins is an automation tool
- With the help of Jenkins we can automate any kind of technology And way we need to automation we need speed in the real world



- Jenkins is not a single tool for automation we have many more tools for automation but the use case is the same
- Jenkins is software built on top of Java and every operating system supports Java
- In the real world all the companies install Jenkins in the Linux system
- Also we do the same setup in the AWS cloud we use the Linux operating system on top of this we install Java then we launch Jenkins



[Jenkins]

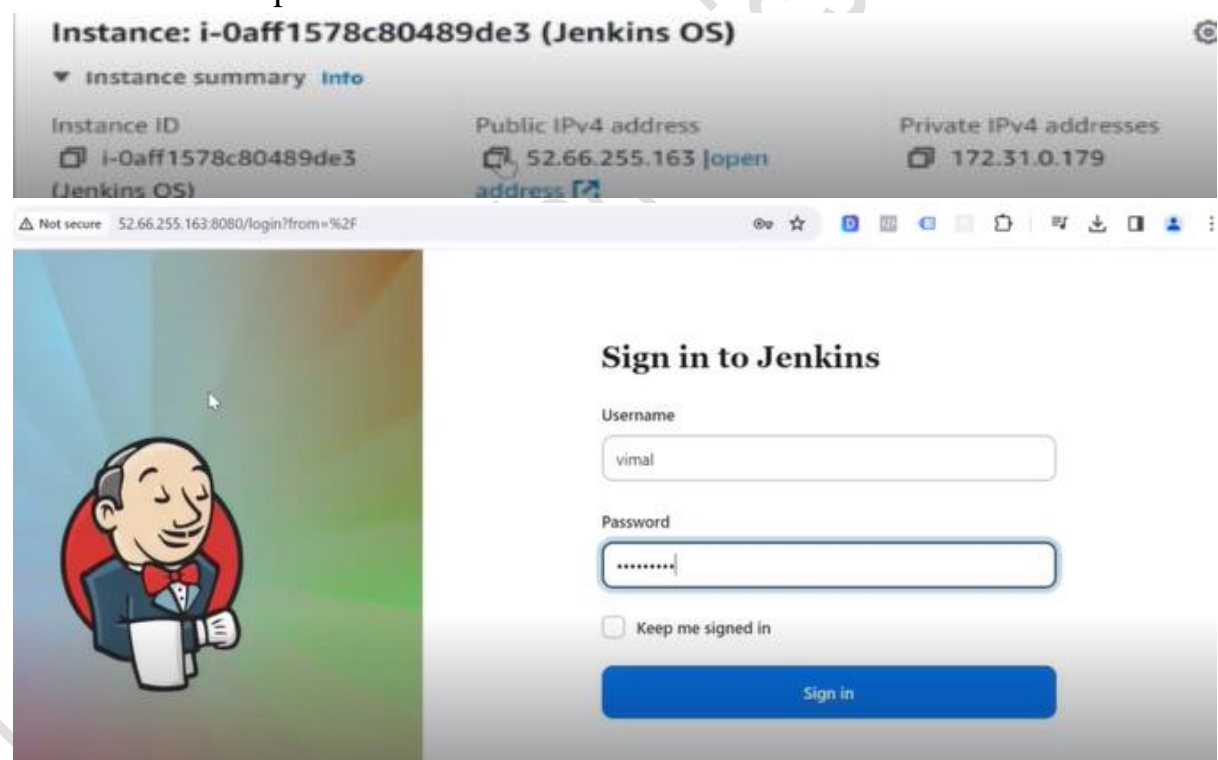
- Let's start the practical
- For that first we need to connect with the instance and see the status of the Jenkins is running or not

#sudo su – root

#systemctl status Jenkins

```
/m/'
Last login: Sat Feb  3 11:59:28 2024 from 13.233.177.4
[ec2-user@ip-172-31-0-179 ~]$ sudo su - root
Last login: Sat Feb  3 11:59:36 UTC 2024 on pts/0
[root@ip-172-31-0-179 ~]# systemctl status jenkins
• jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: disabled)
   Active: active (running) since Sun 2024-02-11 10:37:28 UTC; 1min 28s ago
     Main PID: 2072 (java)
       Tasks: 44 (limit: 4660)
        Memory: 1.2G
          CPU: 36.562s
        CGroup: /system.slice/jenkins.service
                └─2072 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/
httpPort=8080
```

- After that take the public IP address of the instance and paste it into the browser with the port number 8080



- Jenkins helps to automate if you do it manually and it takes a lot of time
- For example if you run some command in the Linux operating system like cal or date command and you need to see it every two minutes later
- For that we go and type the command after that we see the output of the command
- With the help of Jenkins we automate all the technology of the program

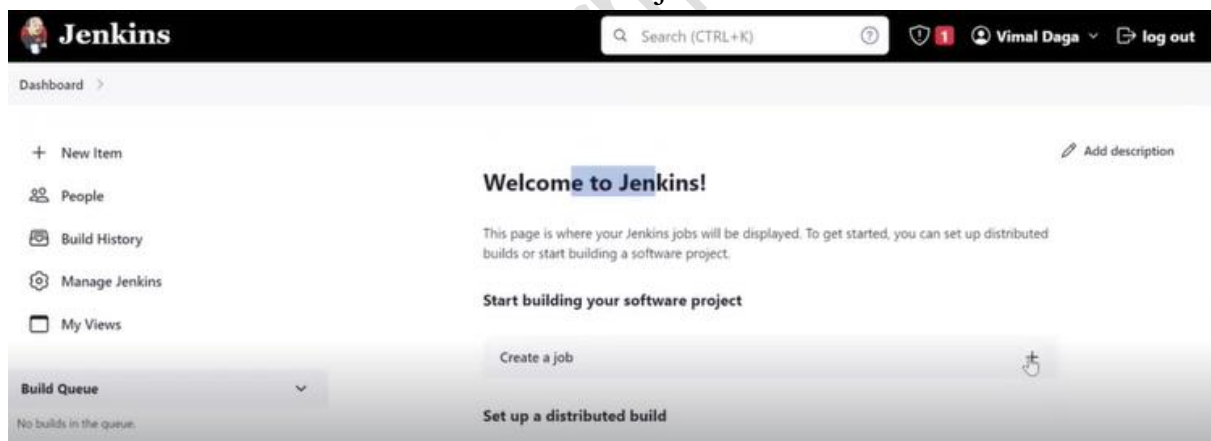
[Jenkins]

```
[root@ip-172-31-0-179 ~]# date
Sun Feb 11 10:41:51 UTC 2024
[root@ip-172-31-0-179 ~]# cal
    February 2024
Su Mo Tu We Th Fr Sa
                1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29
```

- For example we have some programs in that program we need some command and this command we need to use again and again
- For that Jenkins creates a job and puts all the steps on it then we don't need to type again and again
- Jobs means in the Jenkins items or projects

Job == Item

- To create a job go to the dashboard of the Jenkins
- After that click on the new item or create a job both are the same

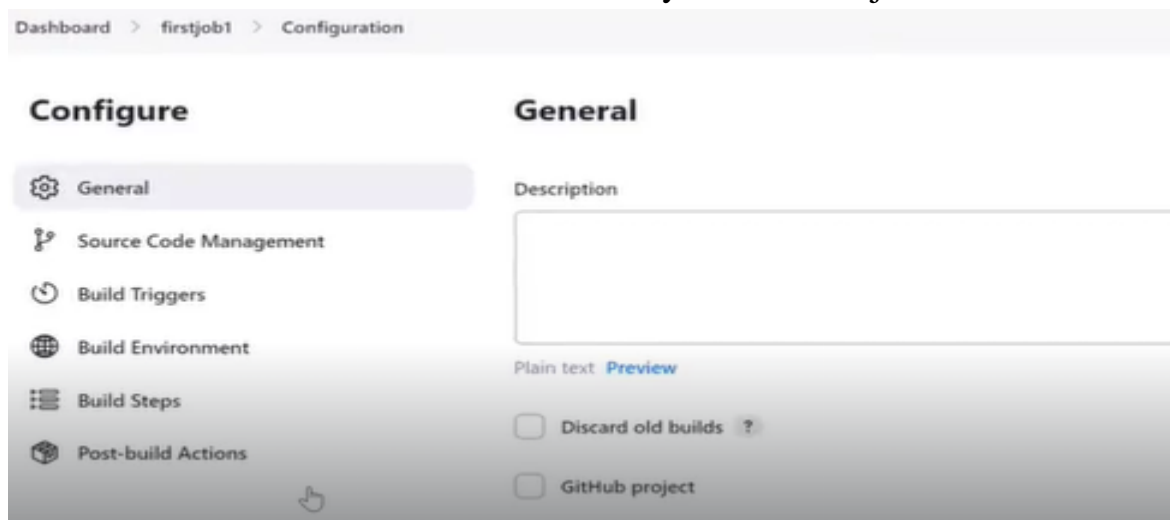


- Then give the job name after that click on the freestyle project

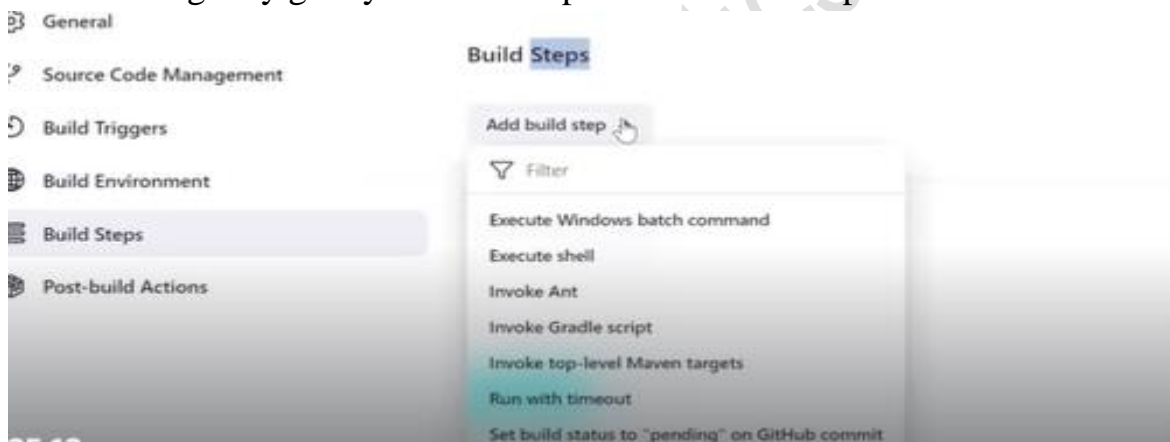


[Jenkins]

- Then click on the ok after that Jenkins asks you to set the job



- Set up the job based on the requirement I require to run the date or cal command for that we go to the build steps and click on the added build step after clicking they give you a lot of options for build steps



- We use execute shell because if you want to run any command we need shall

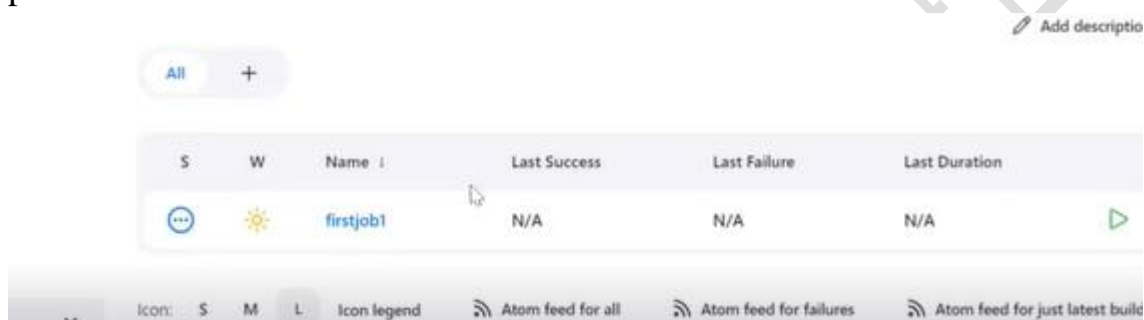


- In that shall we type the required commands in sequence and Jenkins run this command in the same order
- After that click on the save
- Now your first job is created

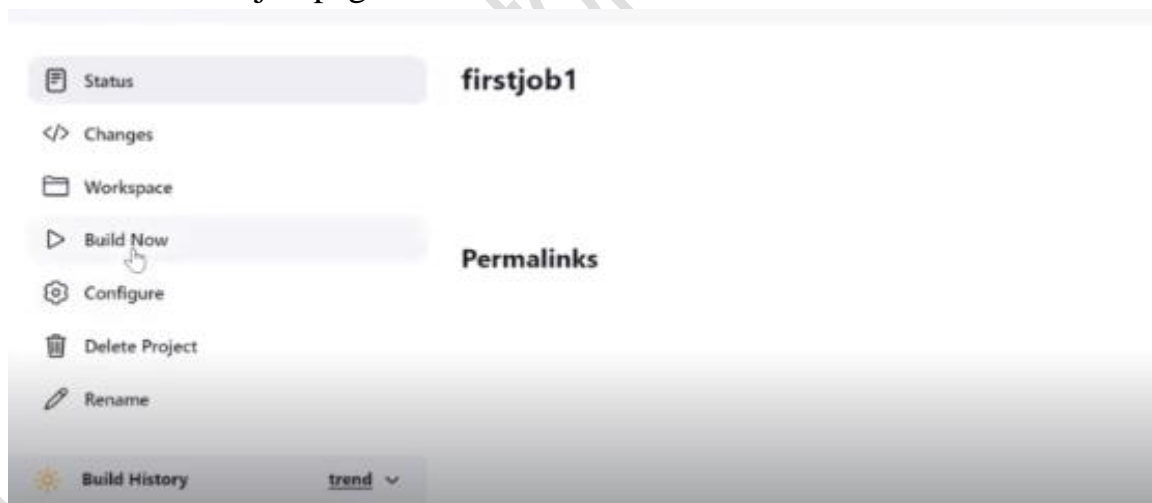
[Jenkins]



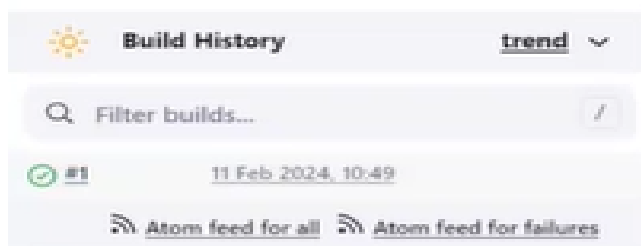
- As we can see Jenkins creates a job but this job is only created not run yet
- After creating a job it's your choice when you want to run the job this process is known as **build**



- If you want to run the job for that go to the job page and click on the job
- You land on the job page after that click on the build now

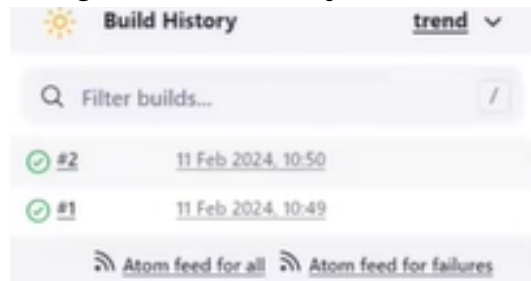


- After that we see in the build history your job is running



[Jenkins]

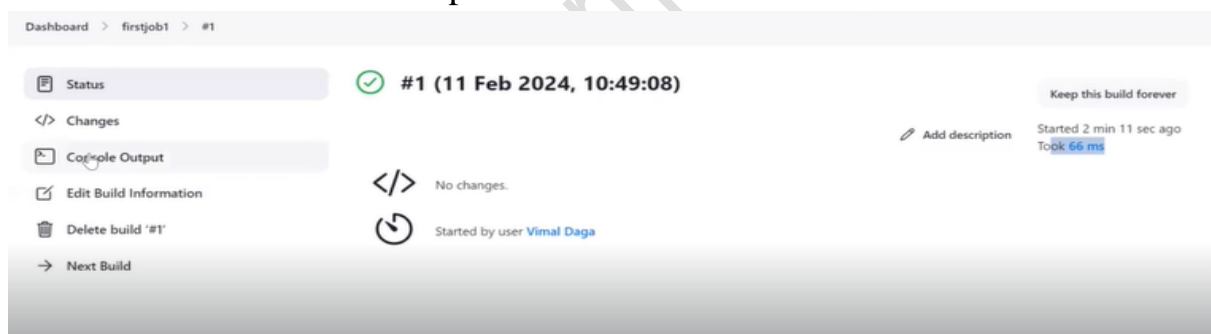
- Green mark means your job ran successfully also they give the time of the job
- If you want to edit the job for that go to the configure then make some changes and save the job after that Jenkins gives one more build number



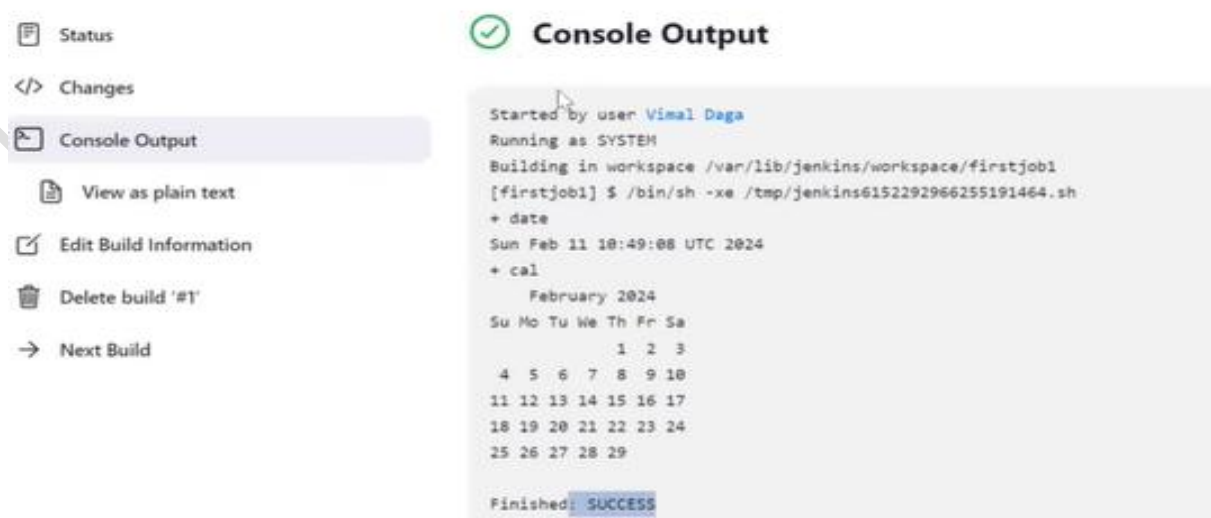
- As we can they add one more number after making some change
- Note:- whenever you change the job they add a new number in the build history
- If you want to see the output of the job then click on the job history after that they give the all details of the job



- Then click on the console output



- After we see the output

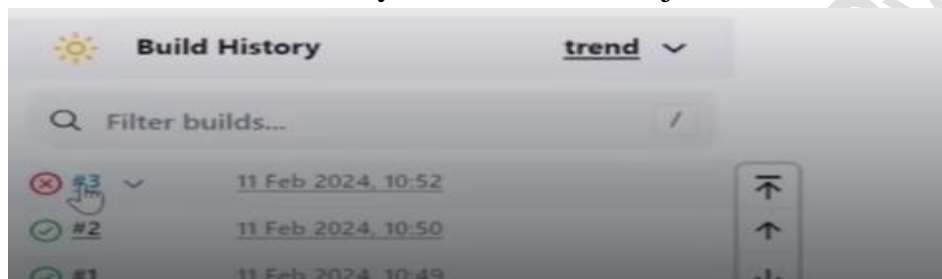


[Jenkins]

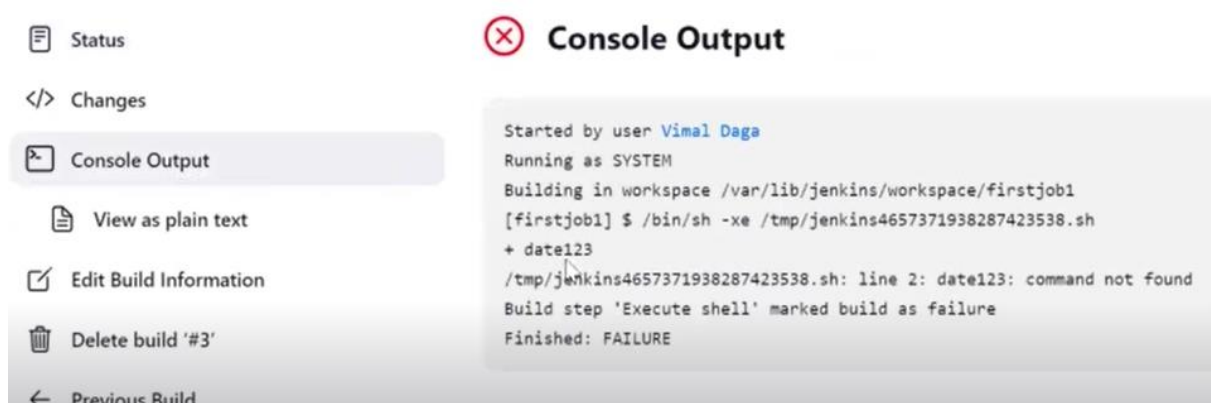
- If you again make some changes and type the wrong command after saving the job



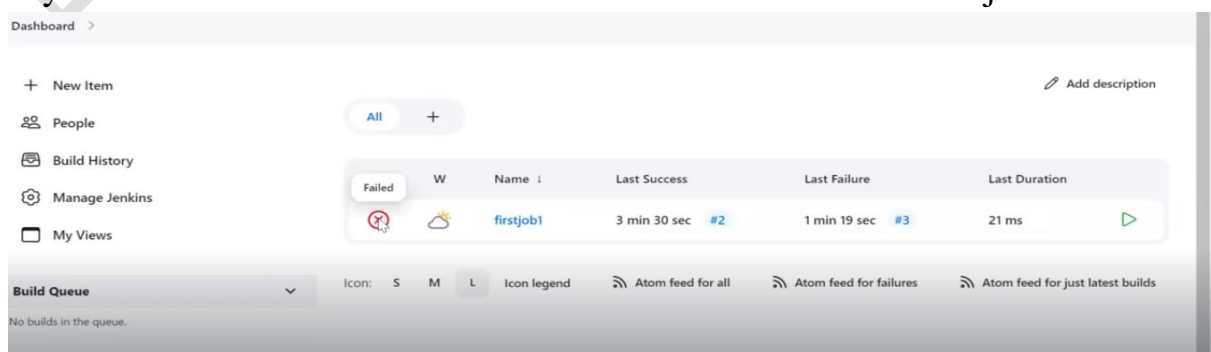
- After that click on Build now
- Then see the build history we see one more job number is added



- As we can see the job is a failure if you want to why your job is failing then click on job 3 and then click on the console output

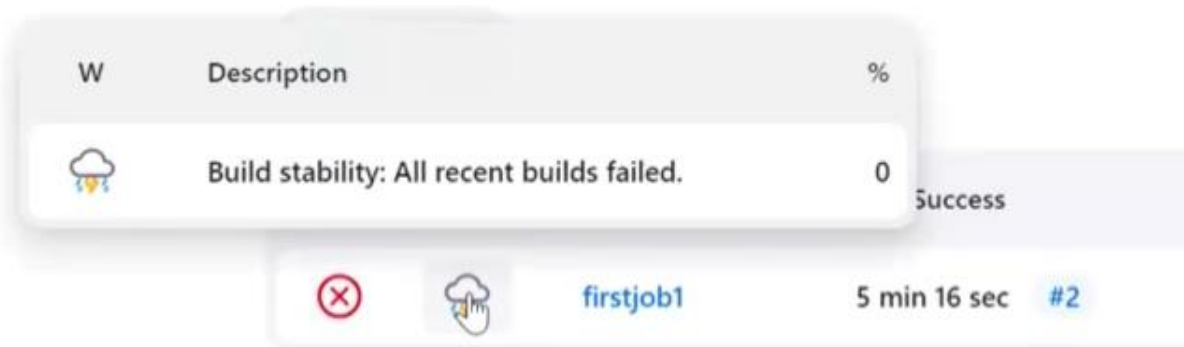


- They give all the details
- If you click on the dashboard of Jenkins after that we see the last job is fail

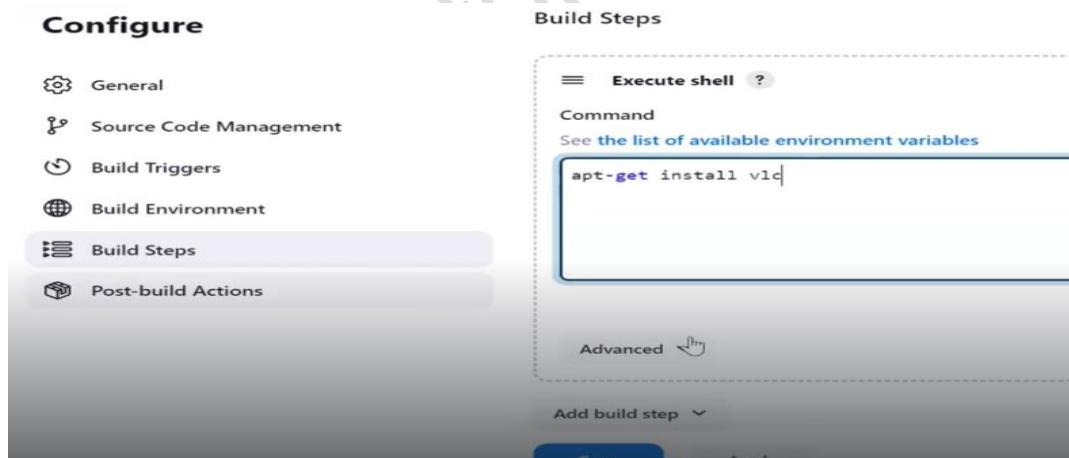


[Jenkins]

- The good thing about Jenkins is it gives small animations for the current status



- If a lot of jobs get fail then they change the cloud
- One important thing is whenever you run any command in Jenkins all the commands are running in the same operating system where your Jenkins is installed
- For example if you run any command and it is not available in your operating system then your Jenkins job fails
- We go to the job and make changes and we need to install apt-get install vlc then save the job

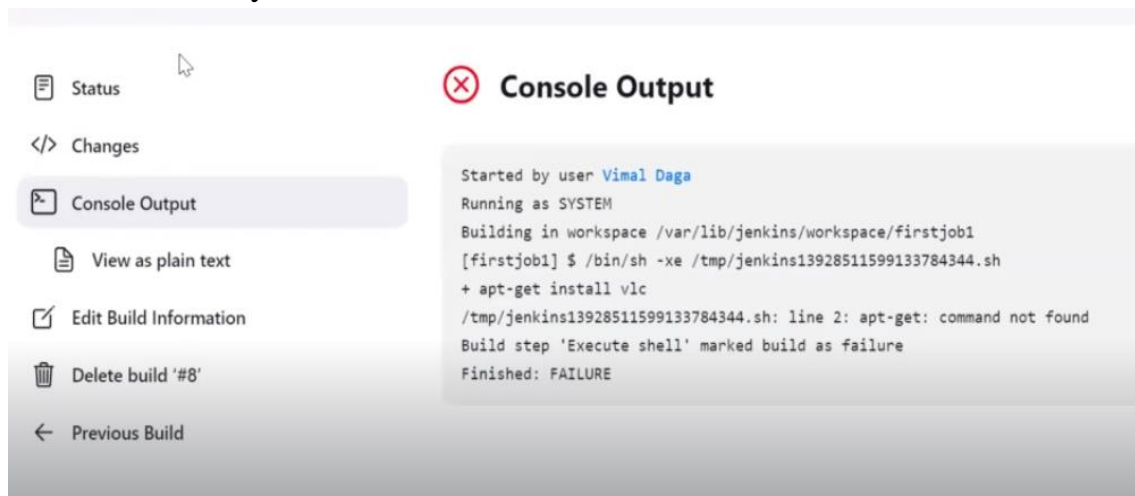


- After that run the job
- As we can see the job is failing

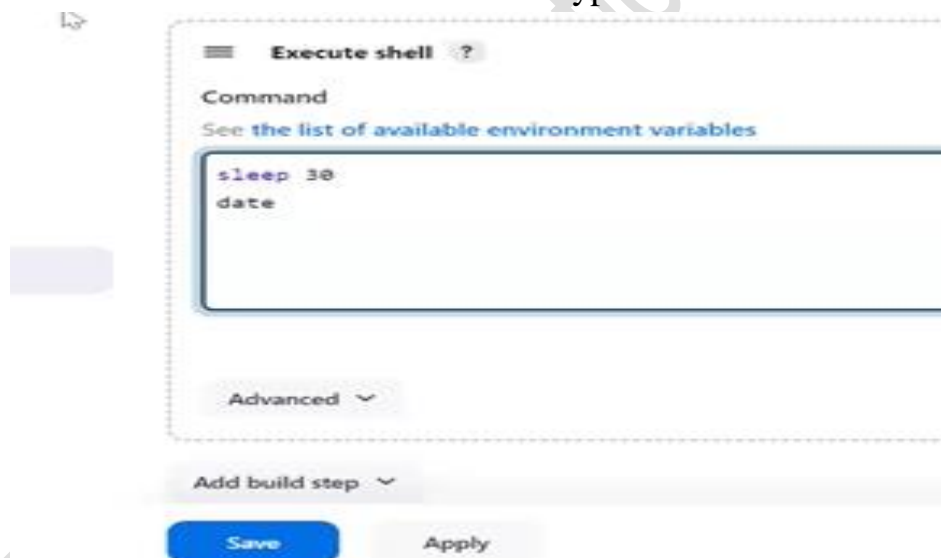


[Jenkins]

- After that go to the console output and we see the job is fail
- And the error is they tell us we don't have this command



- To visualization above we go and make some changes in the job in that job we use the sleep command for 30 seconds with the help of this command we stop the command for some time after that we type the data command and



- Save this job if we run this job then behind the scenes, Jenkins goes and connects to the base Operating system and asks please run this job for me
- If we run the job then we see the job in the build history job is running



- To see this process we go to the command prompt

[Jenkins]

#ps -aux | grep jen

```
[root@ip-172-31-0-179 ~]# ps -aux | grep jen
jenkins 2072  4.0 31.2 3950948 1248376 ?        Ssl  10:37   1:03 /usr/bin/java -Djava.awt.headless=true -jar
/usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080
jenkins 3524  0.0  0.0 232036 3300 ?          S    11:02   0:00 /bin/sh -xe /tmp/jenkins9807507553262829586.
sh
jenkins 3526  0.0  0.0 221360 1020 ?          S    11:02   0:00 sleep 30
root 3535  0.0  0.0 222312 2000 pts/0      S+   11:02   0:00 grep --color=auto jen
```

- With the help of the above command we see all the processes as we can after running the job one new program is coming
- After 30 sec we see the process is stop means the job ran successfully

```
[root@ip-172-31-0-179 ~]# ps -aux | grep jen
jenkins 2072  4.0 31.2 3950948 1248276 ?        Ssl  10:37   1:03 /usr/bin/java -Djava.awt.headless=true -jar
/usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080
root 3587  0.0  0.0 222312 2004 pts/0      S+   11:03   0:00 grep --color=auto jen
```



- Means behind the scenes Jenkins runs shell script on your Operating system
- If you go inside the job file you can see they run all the commands for you

```
/usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080
jenkins 3598  0.0  0.0 232036 3308 ?          S    11:04   0:00 /bin/sh -xe /tmp/jenkins1453888559302202307.
sh
jenkins 3602  0.0  0.0 221360 1016 ?          S    11:04   0:00 sleep 30
root 3604  0.0  0.0 222312 2024 pts/0      S+   11:04   0:00 grep --color=auto jenkins
[root@ip-172-31-0-179 ~]# cat /tmp/jenkins1453888559302202307.sh
sleep 30
date
```

- This above technique is good for the troubleshooting
- When you install Jenkins in the Operating system behind the scene they create one Jenkins ID means a Linux user

#rpm -q Jenkins

```
[root@ip-172-31-0-179 ~]# rpm -q jenkins
jenkins-2.426.3-1.1.noarch
[root@ip-172-31-0-179 ~]#
```

- As we can see we have Jenkins software after that we see the account of the Jenkins

```
[root@ip-172-31-0-179 ~]# id jenkins
uid=992(jenkins) gid=992(jenkins) groups=992(jenkins)
[root@ip-172-31-0-179 ~]#
```

- You know every user has some special powers and only the root user has all the power to do anything

[Jenkins]

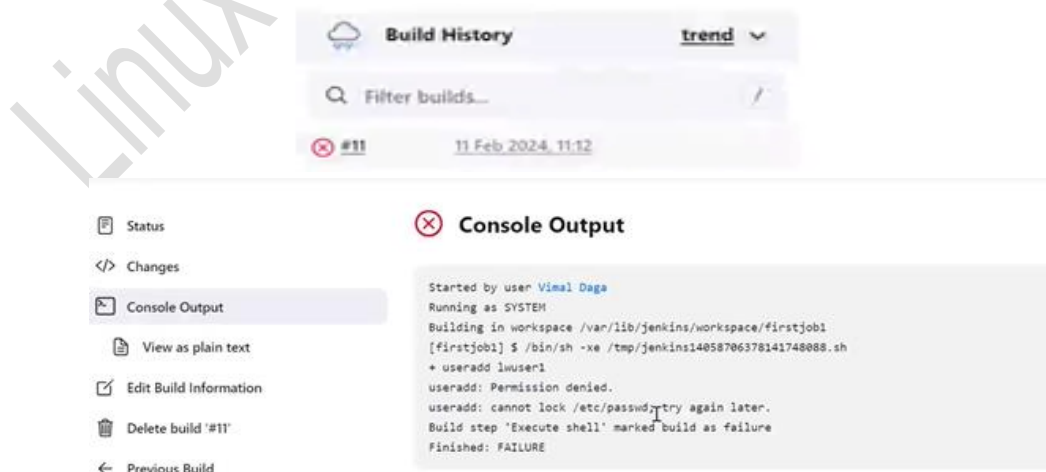
- For example we create one user and this user has the power to run the date command but this user does not have the power to create a new user or install something

```
[root@ip-172-31-0-179 ~]# useradd vimal
[root@ip-172-31-0-179 ~]# su - vimal
[vimal@ip-172-31-0-179 ~]# whoami
vimal
[vimal@ip-172-31-0-179 ~]#
[vimal@ip-172-31-0-179 ~]# date
Sun Feb 11 11:07:55 UTC 2024
[vimal@ip-172-31-0-179 ~]# useradd dfbg
useradd: Permission denied.
useradd: cannot lock /etc/passwd; try again later.
[vimal@ip-172-31-0-179 ~]# yum install httpd
Error: This command has to be run with superuser privileges (under the root user on most systems).
[vimal@ip-172-31-0-179 ~]# exit
logout
[root@ip-172-31-0-179 ~]#
```

- Means whenever we run the job then OS runs the process with the Jenkins Account
- In Linux if you want to run any program then we need a user account without any account we can not run any program
- Jenkins account has some limited powers for example if you want to create a user from Jenkins



- After running the job then the job fails because Jenkins has some limited powers



[Jenkins]

- That is a reason we need to give extra power to the Jenkins account
- And with the help of sudo we give the extra power to the Jenkins
- To give the first we go to the **sudoers file**

```
[root@ip-172-31-0-179 ~]# vi /etc/sudoers
```

- Then make some changes in the file

```
##  
## The COMMANDS section may have other options a  
##  
## Allow root to run any commands anywhere  
root    ALL=(ALL)    ALL  
  
vimal    ALL=(ALL)    ALL  
  
## Allows members of the 'sys' group to run network  
to
```

- As we can we give the Vimal account root power
- Now this account do anything
- To check if it working or not we need to create the first password for the Vimal account

```
[root@ip-172-31-0-179 ~]# passwd vimal  
Changing password for user vimal.  
New password:
```

- Then we log in to the Vimal account

```
[root@ip-172-31-0-179 ~]# su - vimal  
Last login: Sun Feb 11 11:16:50 UTC 2024 on pts/0
```

- After we need to tell the Vimal account you have sudo power for that we use sudo

```
[root@ip-172-31-0-179 ~]# su - vimal  
Last login: Sun Feb 11 11:16:50 UTC 2024 on pts/0  
[vimal@ip-172-31-0-179 ~]$ sudo useradd pop  
  
We trust you have received the usual lecture from the local System  
Administrator. It usually boils down to these three things:  
  
#1) Respect the privacy of others.  
#2) Think before you type.  
#3) With great power comes great responsibility.  
  
For security reasons, the password you type will not be visible.
```

- Now we successfully add a new user

```
[vimal@ip-172-31-0-179 ~]$ id pop  
uid=1002(pop) gid=1002(pop) groups=1002(pop)  
[vimal@ip-172-31-0-179 ~]$
```

[Jenkins]

- If you do not type sudo first then you can not run the command

```
[vimal@ip-172-31-0-179 ~]$ yum install dialog
Error: This command has to be run with superuser privileges (under the root user on most systems).
```

- That is why we need to type sudo first

```
[vimal@ip-172-31-0-179 ~]$ sudo yum install httpd
Last metadata expiration check: 0:41:08 ago on Sun Feb 11 10:37:13 2024.
```

- This concept is known as **Privilege escalation**
- Same thing we do with Jenkins go to the sudoer file and edit the file
- Give root power

```
[root@ip-172-31-0-179 ~]# vi /etc/sudoers
```

```
##
## Allow root to run any commands anywhere
root    ALL=(ALL)    ALL

vimal    ALL=(ALL)    ALL
```

```
##
## Allow root to run any commands anywhere
root    ALL=(ALL)    ALL

vimal    ALL=(ALL)    ALL

jenkins  ALL=(ALL)    ALL

## Allow ALL=(ALL)    ALL the 'sys' group to run networking, soft
```

- After giving root power then again we run the job with sudo power if you do not use the sudo then Jenkins does not run the command
- In Jenkins we always want our command to be non-interactive because if your command is interactive then they give the error
- non-interactive means any command asking for yes or no for example yum install httpd this command always asks for yes or no
- with the help -y we create any command non-interactive

```
[root@ip-172-31-0-179 ~]# yum install httpd -y
```

- if you want to sudo command goes non-interactive then we go to the sudoer file and make some change

```
[root@ip-172-31-0-179 ~]# vi /etc/sudoers
```

```
##
## Allow root to run any commands anywhere
root    ALL=(ALL)    ALL

jenkins  ALL=(ALL)    ALL

vimal    ALL=(ALL)    ALL

## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
```

```
##
## Allow root to run any commands anywhere
root    ALL=(ALL)    ALL

jenkins  ALL=(ALL)    NOPASSWD: ALL

vimal    ALL=(ALL)    ALL

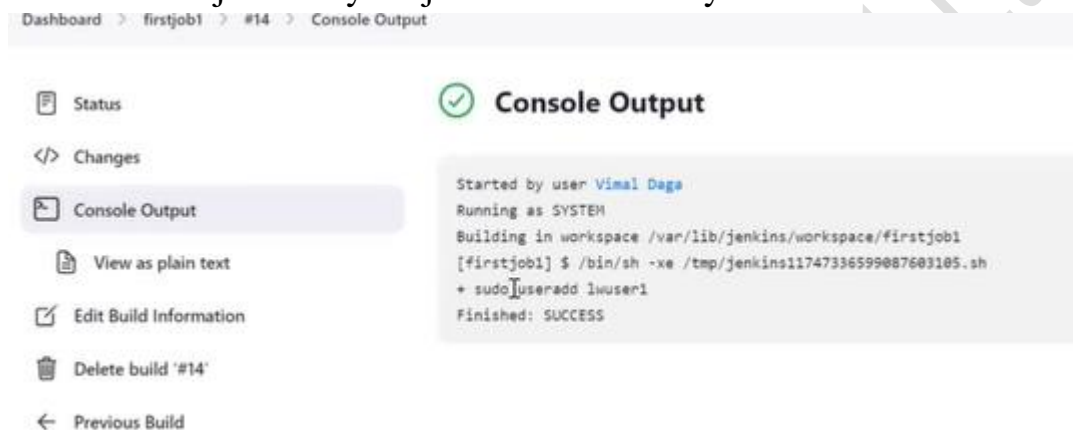
## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROC
```

- Only add one keyword in front of Jenkins is **NOPASSWD**
- After that Jenkins does all the things for example we again make changes in the job

[Jenkins]



- After we run the job then your job run successfully



- As we can see the new user is coming up

```
[root@ip-172-31-0-179 ~]# id lwuser1
uid=1003(lwuser1) gid=1003(lwuser1) groups=1003(lwuser1)
[root@ip-172-31-0-179 ~]#
```

- Now we can do anything with Jenkins
- If you want to set up a web server for that we need to follow some steps
- First we need to install the httpd web server
- Second we go to the folder **cd/var/www/HTML**
- Third create one file and Four write all the content
- Fifth we restart the system

```
use case to setup webserver - auto
-----

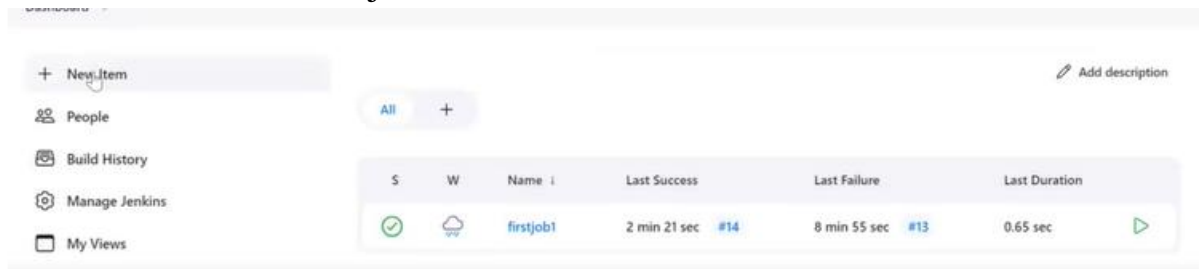
1. yum install httpd

2. cd /var/www/html
   vi index.html
   content

3. systemctl start httpd
```

[Jenkins]

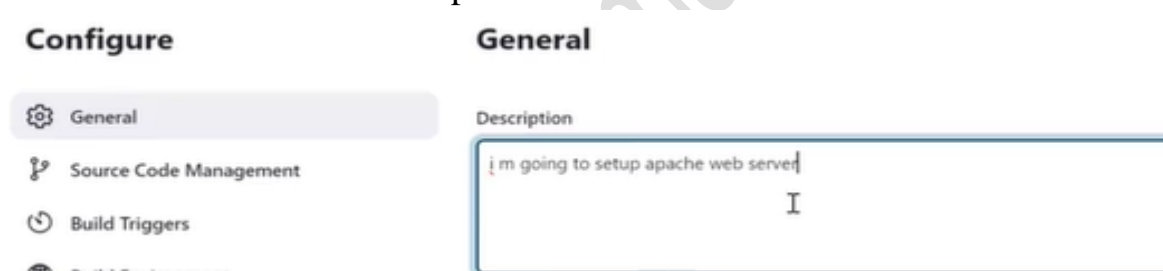
- Now we need to set up this web server automatically
- For that we create a new job



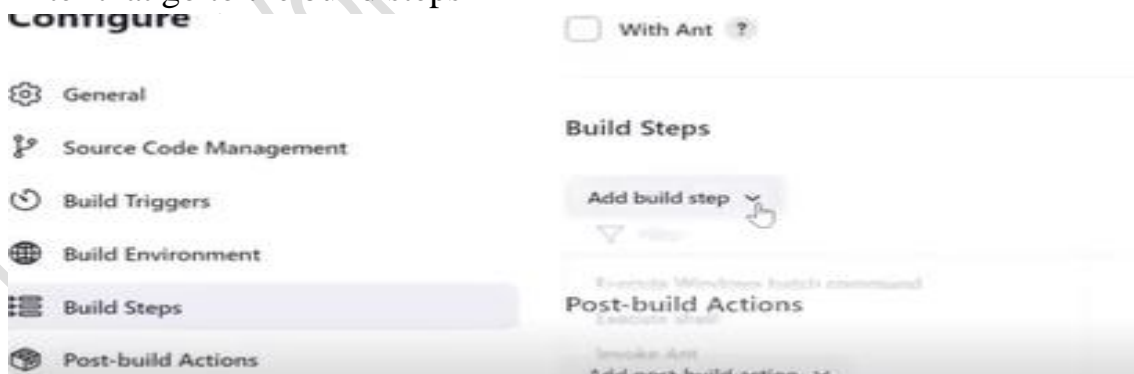
- Give the name and select the freestyle project



- Then write some in the description



- After that go to the build steps

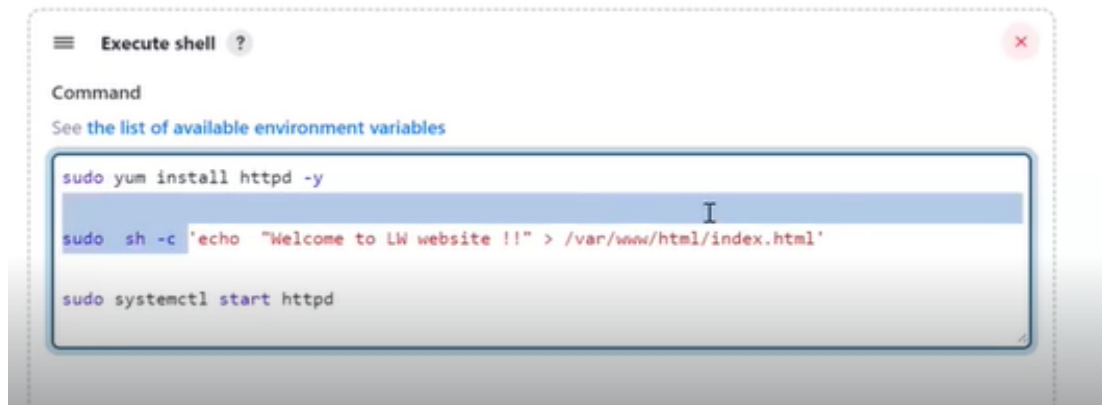


- Then select execute shell

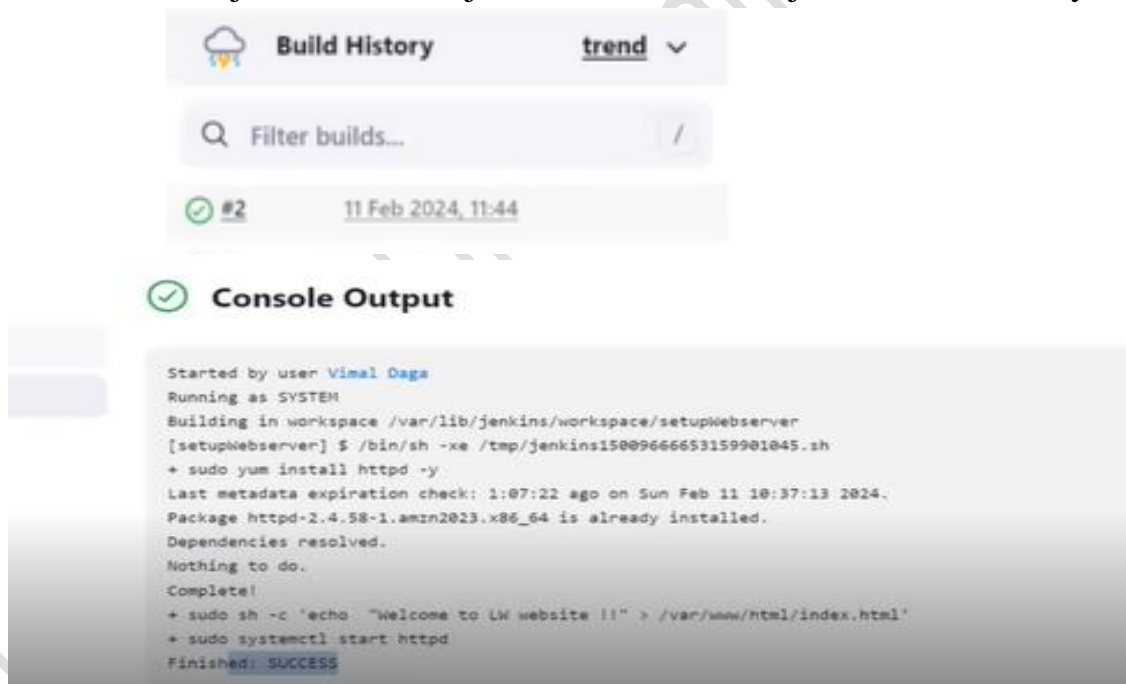


[Jenkins]

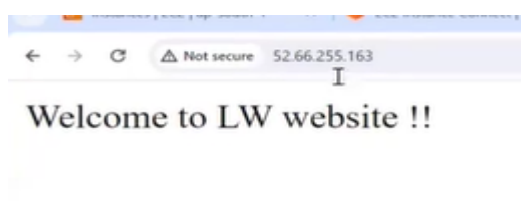
- Then run all the commands required for the web server with sudo power and non-interactive command
- After that we use the echo command with the help of this command we print anything
- In the second we use **sh -c** means this command gives shell and -c means make the command in the string



- Then save this job and run the job as we can see the job run successfully

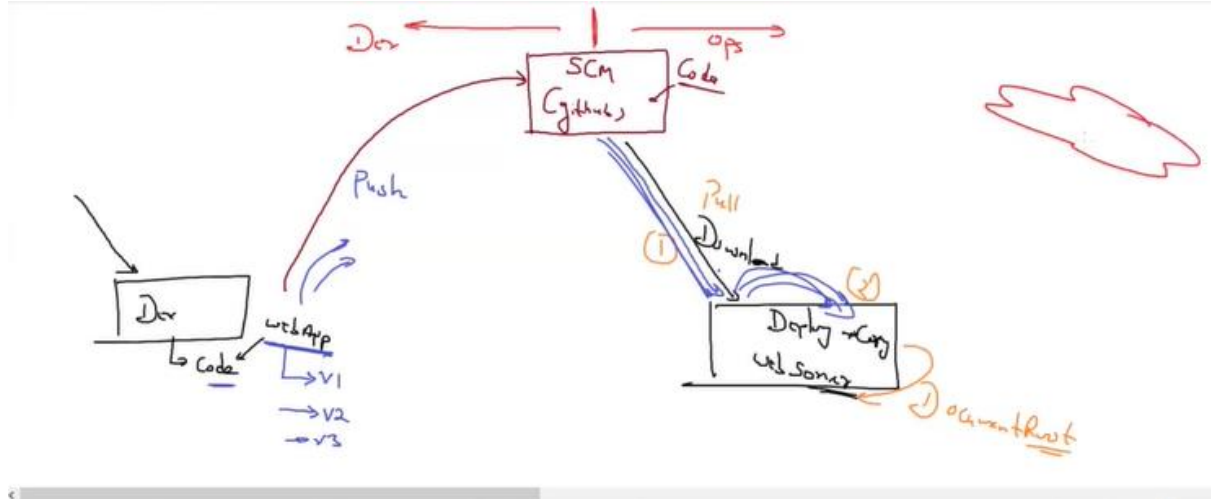


- Now we automatically configure the web server
- After that go to the instance copy the public IP address and paste it into the browser

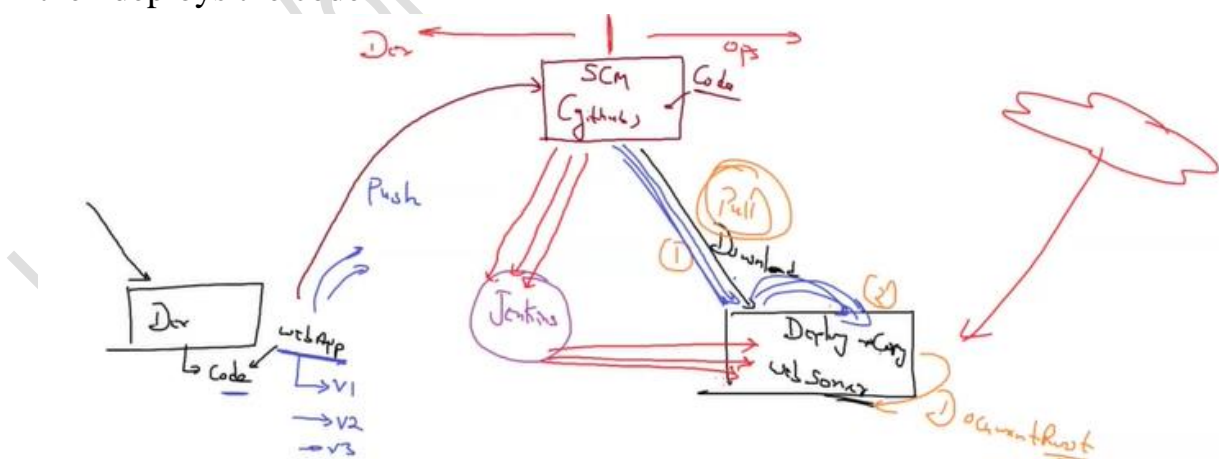


[Jenkins]

- Now we see the web server is ready
- Now a time set up created
- But in the real world if you create a website they follow some steps
- This Means the developer writes code or creates a web app after that developer pushes the sends this code to the SCM tools eg GitHub after that operation team downloads the code and deploys the code



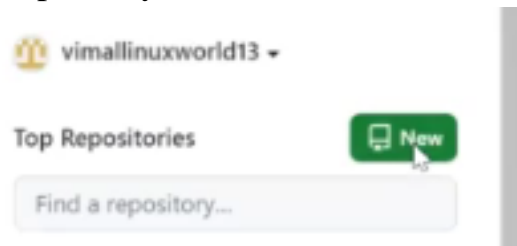
- With all the steps we create a website and if a new version comes then we need to do this thing again and again
- That's why we want all the steps to do automatically with Jenkins
- For that we create a Jenkins program and this all the things with the Jenkins program
- This means the Jenkins program uploads the code and downloads the code then deploys the code



- To make this automated first we need to know about the SCM tools
- For example GitHub
- First we need to learn something about the GitHub
- GitHub is a repository tool in that tool we store our code

[Jenkins]

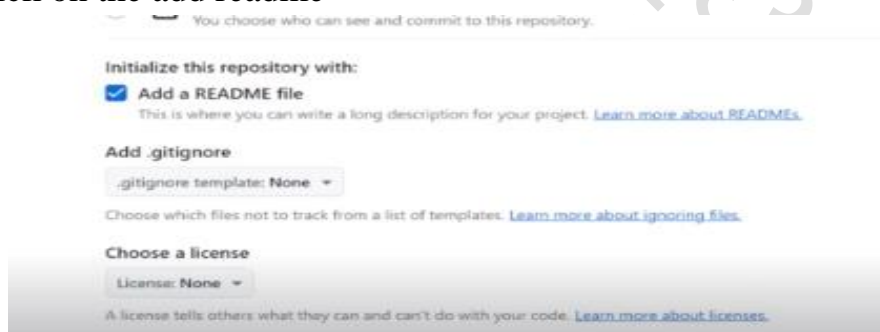
- To create a repository first click on the new



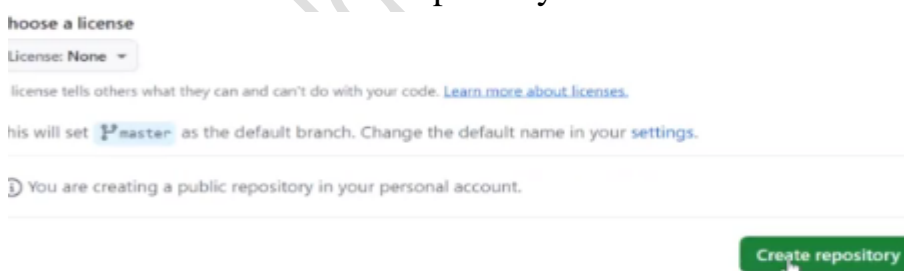
- After that give the repository name



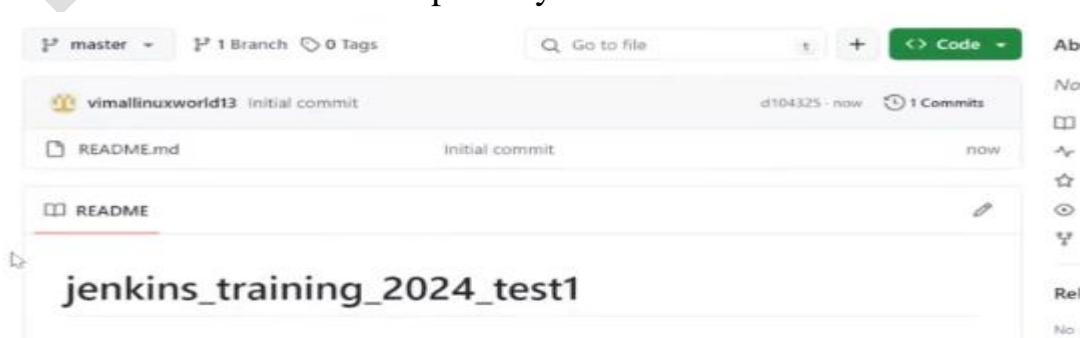
- Click on the add readme



- After that click on the Create repository

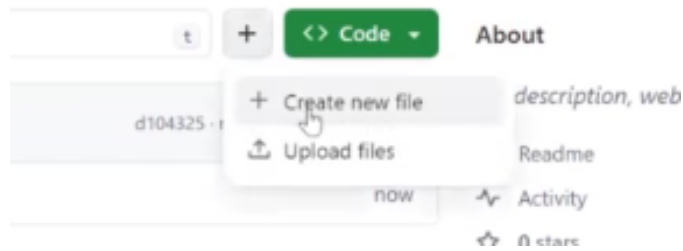


- As we can see we create a repository



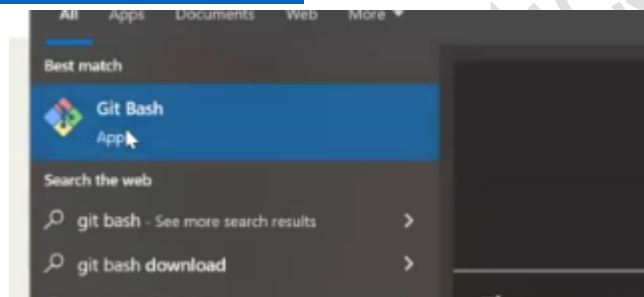
[Jenkins]

- Now your developer comes and writes the code in that repository
- To write a code we click on the Create file

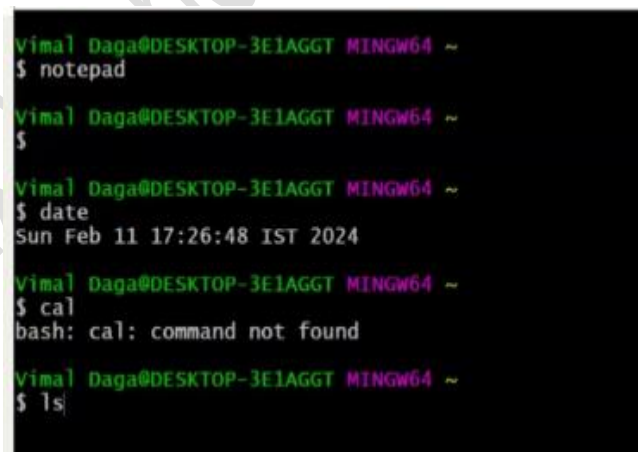


- But this is not a good practice
- Good practice developer writes all code on their laptops and then push it into GitHub for that we use the Git bash program
- Download this program from the internet and this program provides one command prompt
- To download click on the link:-

<https://git-scm.com/downloads>



- A good thing about this tool is it gives you Windows and also give you Linux command

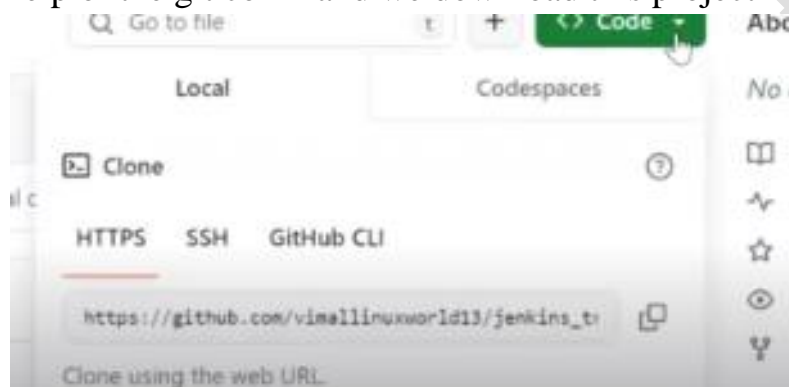


- This tool gives you the git command
- This git command helps to push the code inside the GitHub
- Let's say we create a folder

[Jenkins]

```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~  
$ cd Documents/  
  
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents  
$ mkdir jenkins_training_2024  
  
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents  
$ cd jenkins_training_2024/  
  
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/jenkins_training_2024  
$ ls  
  
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/jenkins_training_2024  
$ |
```

- To push the code we need to tell first where we want to push the code for that we copy the project URL
- With the help of the git command we download this project



```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/jenkins_training_2024  
$ git clone https://github.com/vimallinuxworld13/jenkins_training_2024_test1.git  
  
Cloning into 'jenkins_training_2024_test1'...  
remote: Enumerating objects: 3, done.  
remote: Counting objects: 100% (3/3), done.  
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0  
Receiving objects: 100% (3/3), done.
```

- And the good thing about this URL is they download all the things
- If you want to see the download readme file

```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/jenkins_training_2024  
$ ls  
jenkins_training_2024_test1/  
  
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/jenkins_training_2024  
$ cd jenkins_training_2024_test1/  
  
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/jenkins_training_2024/jenkins_training_2024_test1 (master)  
$ ls  
README.md  
  
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/jenkins_training_2024/jenkins_training_2024_test1 (master)  
$ |
```

- Inside the folder all information they store
- For example we create a file

```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/jenkins_training_2024_test1 (master)  
$ notepad index.html
```

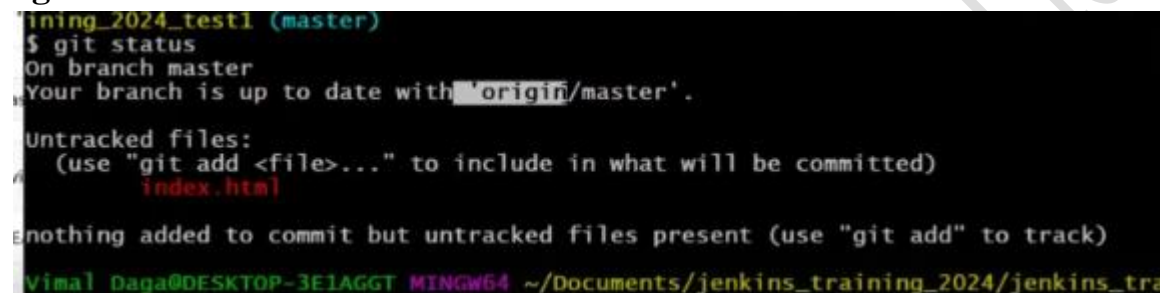

[Jenkins]

- Inside the file we write an HTML code

```
<body bgcolor='aqua'>
Welcome to Vimal Site !!
</body>
```

- Good thing of git bash is they track all the files with the help of this command we see the status

#git status

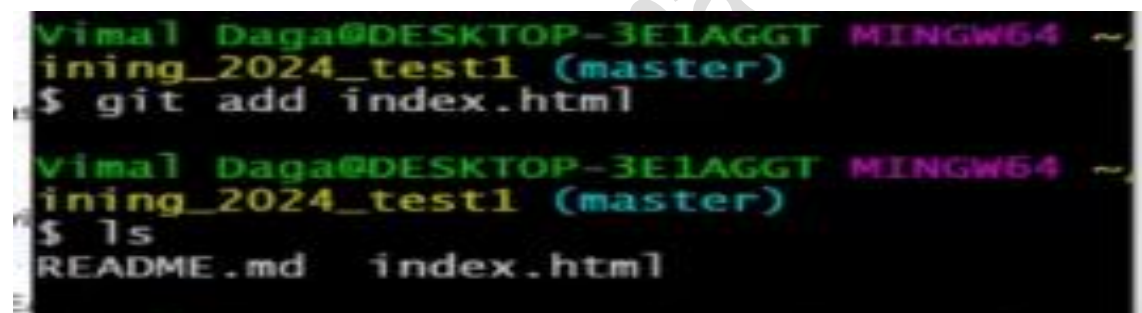


```
ining_2024_test1 (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  index.html

nothing added to commit but untracked files present (use "git add" to track)
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~/Documents/jenkins_training_2024/jenkins_tra
```

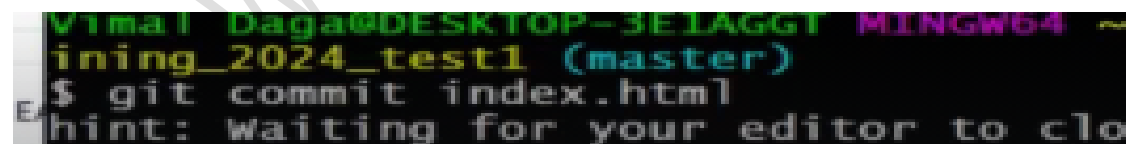
- After that we add the file



```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~
ining_2024_test1 (master)
$ git add index.html

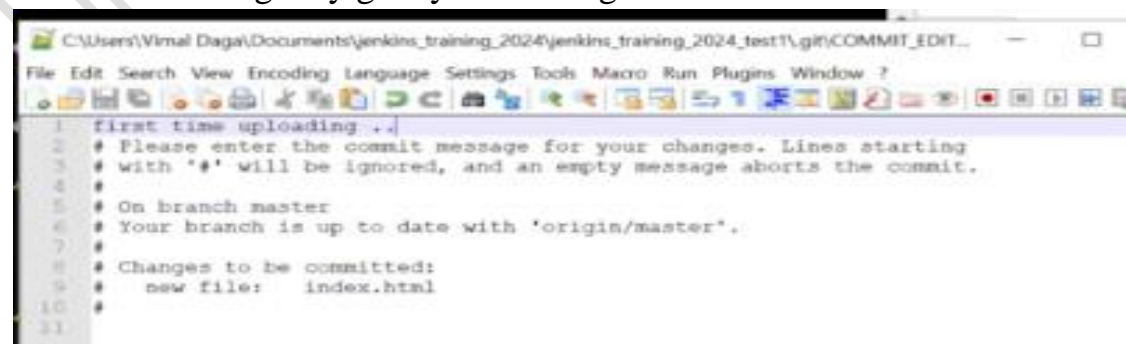
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~
ining_2024_test1 (master)
$ ls
README.md  index.html
```

- After that we need to commit means we are ready to push the file into GitHub



```
Vimal Daga@DESKTOP-3E1AGGT MINGW64 ~
ining_2024_test1 (master)
$ git commit index.html
hint: Waiting for your editor to close the file
```

- After committing they give you warning



```
C:\Users\Vimal Daga\Documents\jenkins_training_2024\jenkins_training_2024_test1\git\COMMIT_EDIT_...
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
1 first time uploading ..
2 # Please enter the commit message for your changes. Lines starting
3 # with '#' will be ignored, and an empty message aborts the commit.
4 #
5 # On branch master
6 # Your branch is up to date with 'origin/master'.
7 #
8 # Changes to be committed:
9 #   new file:   index.html
10 #
11
```

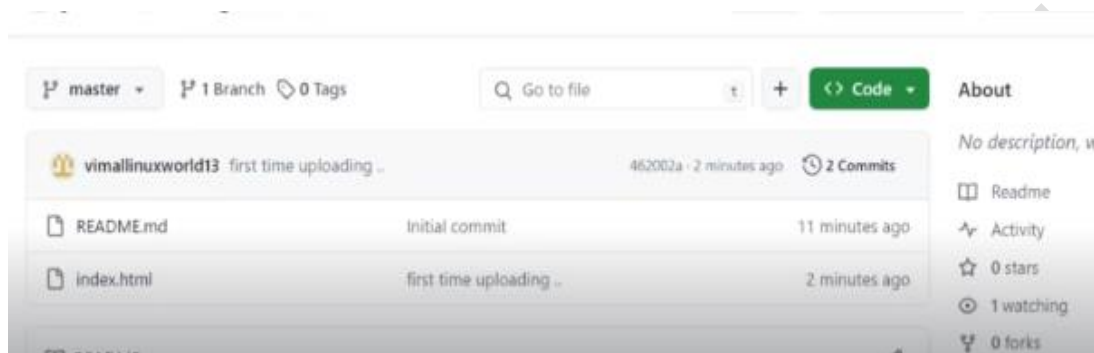
- Then save it

[Jenkins]

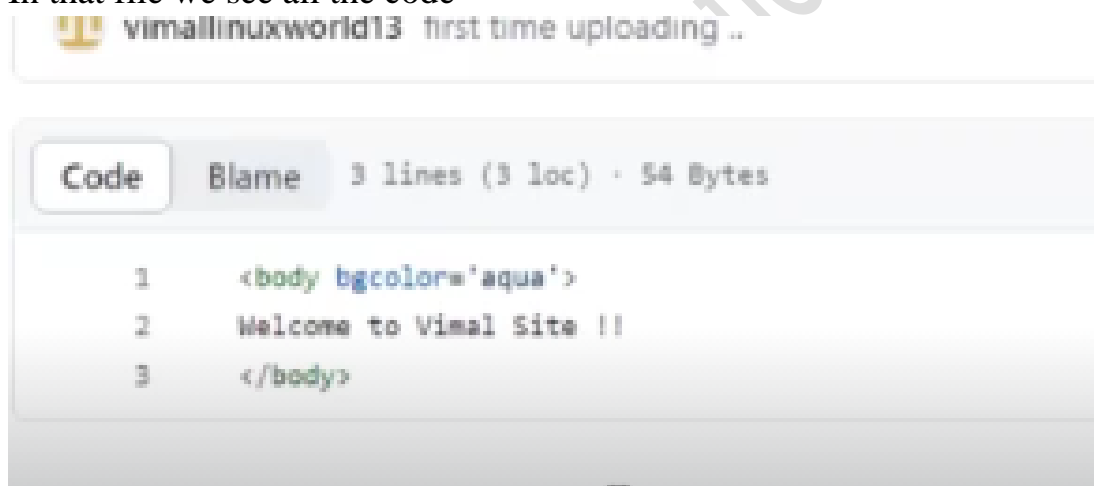
- After that push the file

```
jenkins_training_2024_test1 (master)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 338 bytes | 338.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/vimallinuxworld13/jenkins_training_2024_test1.git
d104325..462002a master -> master
```

- Now we can see one more file has been added to the GitHub



- In that file we see all the code

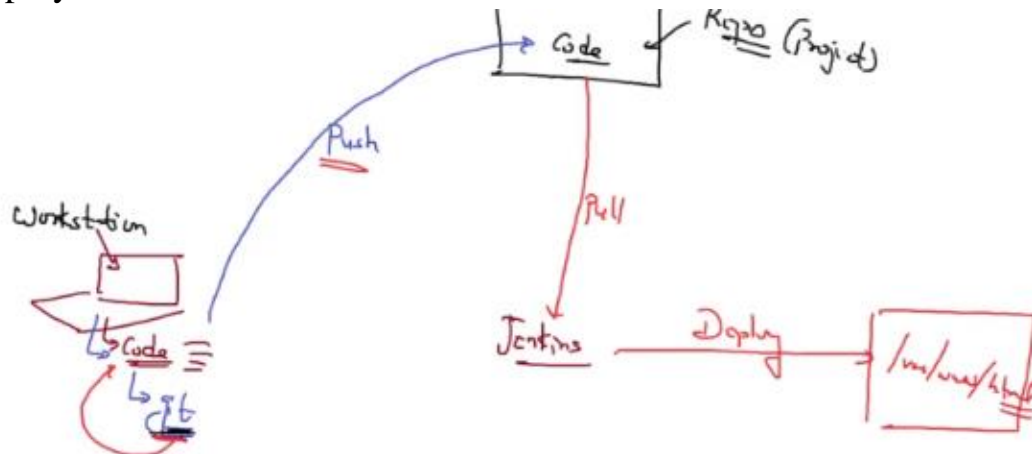


- Below command we follow for the push of the code

```
508 mkdir jenkins_training_2024
509 cd jenkins_training_2024/
510 ls
511 git clone https://github.com/vimallinuxworld13/jenkins_training_2024_test1.git
512 ls
513 cd jenkins_training_2024_test1/
514 ls
515 ls
516 ls -a
517 notepad index.html
518 ls
519 git status
520 ls
521 git add index.html
522 ls
523 git commit index.html
524 git status
525 git push
526 history
```


[Jenkins]

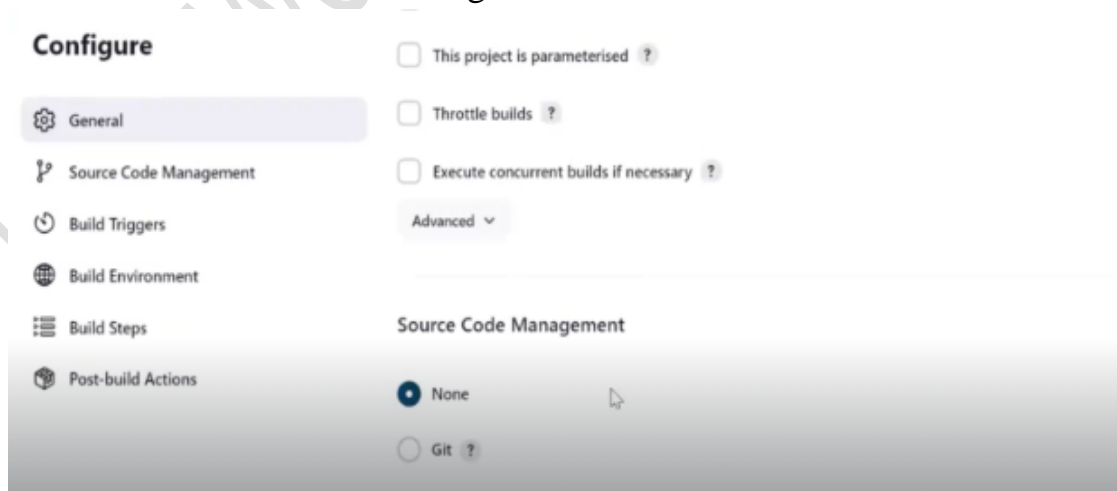
- After pushing the code then we need Jenkins to download the code and deploy the code



- For that we create a new job on Jenkins with a freestyle project



- Jenkins has some special steps for that
- Click on the source code management

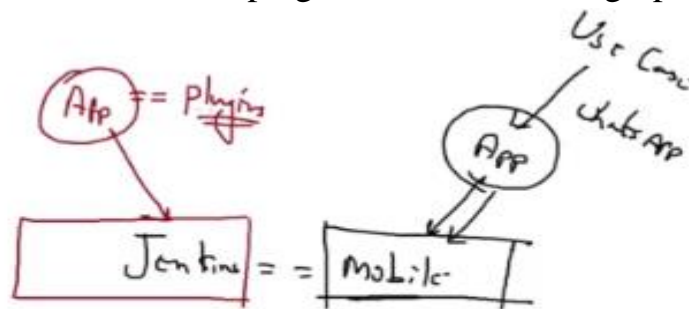


- Then click on git after that give the URL of your project when you save they give you an error and this error happens when we don't have **plugins** and you don't have the **git command**

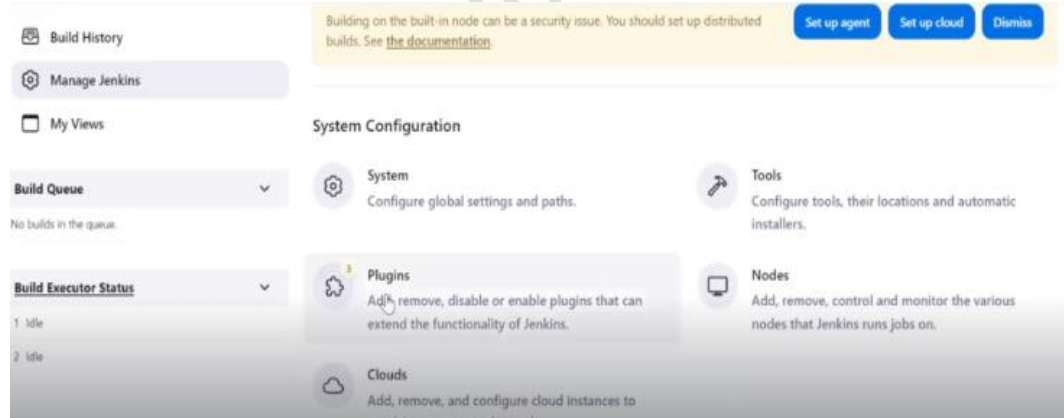
[Jenkins]



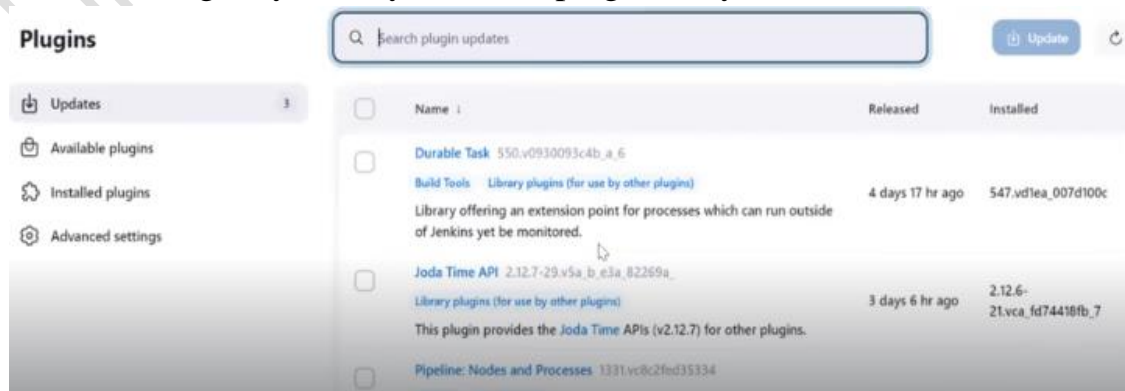
- Because Jenkins is like mobile and particular work we do with the particular app for example Whatsapp but in Jenkins we need plugins
- In Jenkins we have a lot of plugins means we have git plugins



- To find the plugin we go to the manage Jenkins then click on the plugins

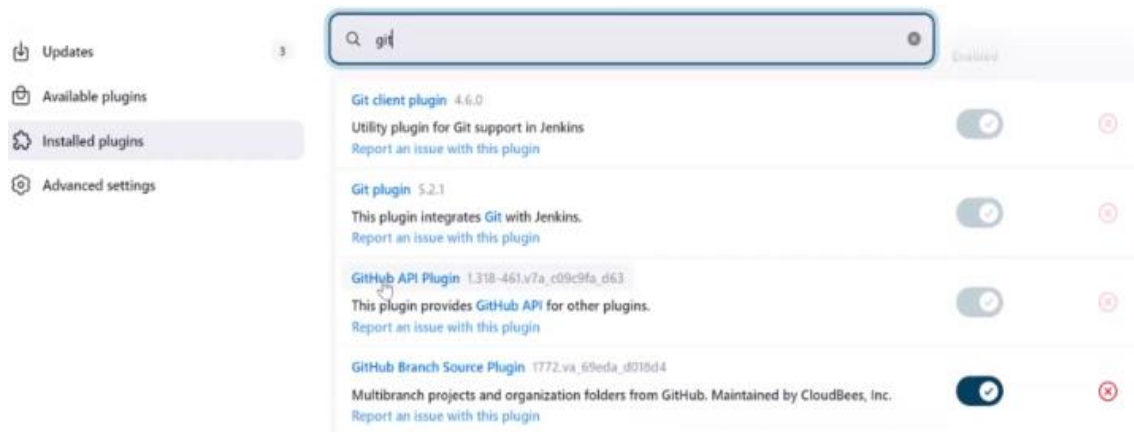


- After clicking they show you all the plugins they have



[Jenkins]

- After that click on the installed plugins then search Git

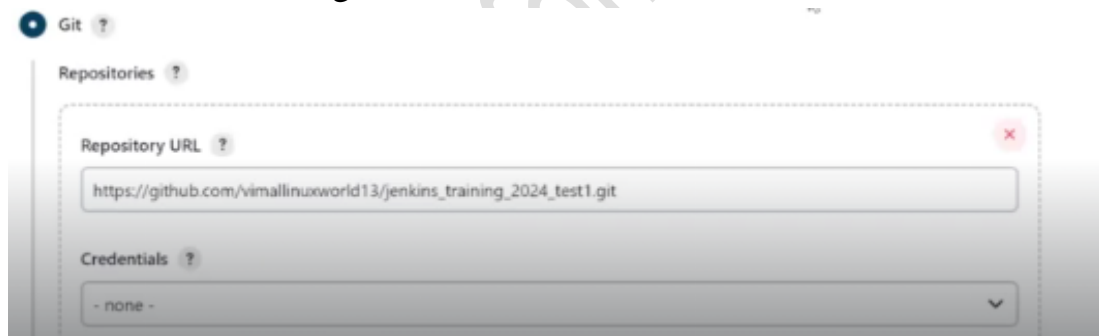


- And with the help of this plugin we go to GitHub and download the code
- If your plugin is ok then we need to install the git command
- To install the git command we go to the instance terminal and run the below command

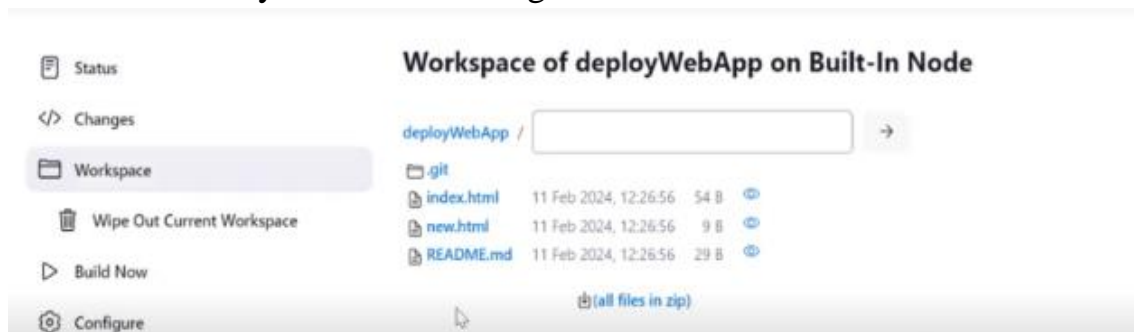
#yum install git

```
[root@ip-172-31-0-179 ~]# yum install git
```

- Now we have the git command where your Jenkins is running
- After that we need to give the URL one more time

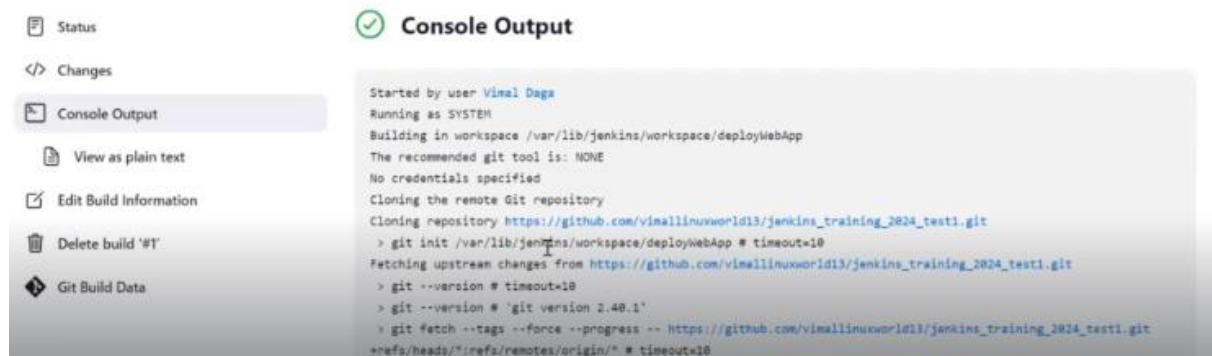


- As we can see the git hub URL is successfully added
- After that Jenkins knows how to pull the code
- Then we run the job and we see in Jenkins they create a **workspace** means they create a folder behind the scene
- In that folder they store all the things downloaded from GitHub

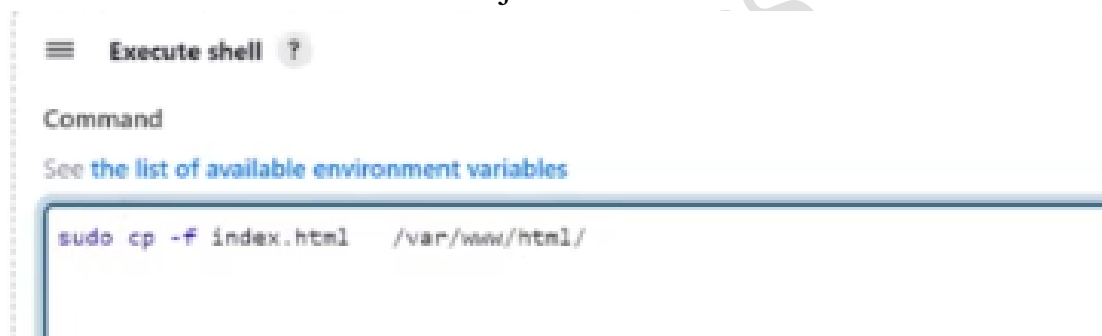


[Jenkins]

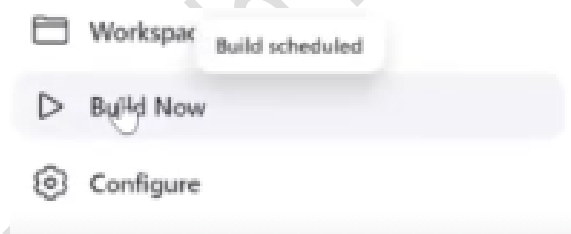
- As we can see they download the files and store them in the workspace
- If you see the console output the job run successfully means we are successfully integrating **Jenkins with the GitHub**



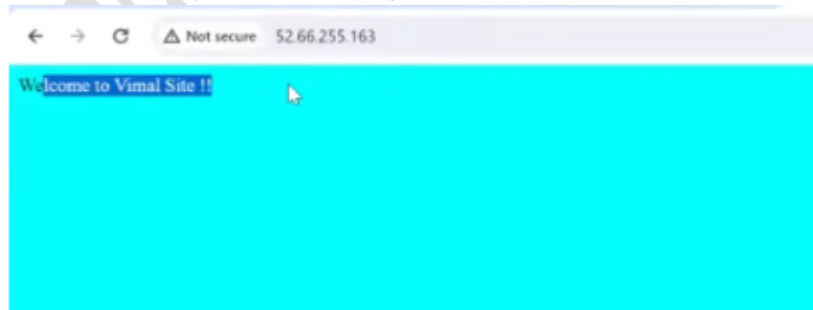
- After that we copy our file
- To copy all the files in the job then go to the executing shell and write the below command after that save the job



- After that click on the build



- Then we see Jenkins change the website color



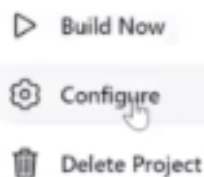
- This above setup is one time set up
- Whenever the developer writes the code again after in the Jenkins only we want to run the job again

[Jenkins]

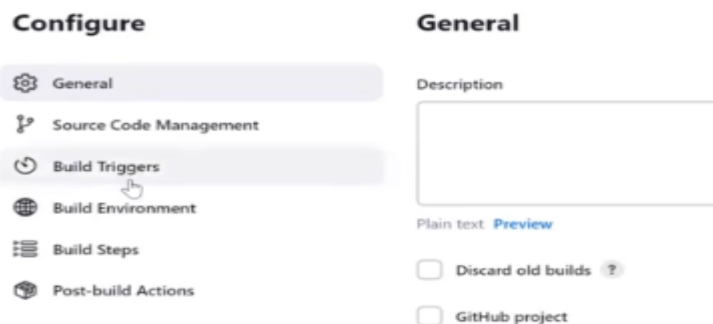
- Then Jenkins automatically pulls the code and deploys the code
- But in this process we want to click the **build now** option again and again this is not a good practice



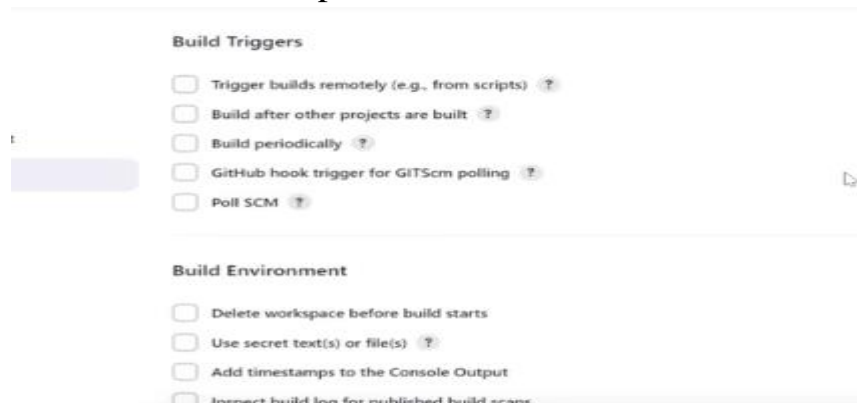
- If you want this part to run automatically we don't want to come and click the button manually this concept is known as a **trigger**
- Anything you want to automate then we need to trigger this part in Jenkins we have a lot of triggers
- To see the triggers go and click on the configure after clicking we see the build trigger



- Then click on the build triggers

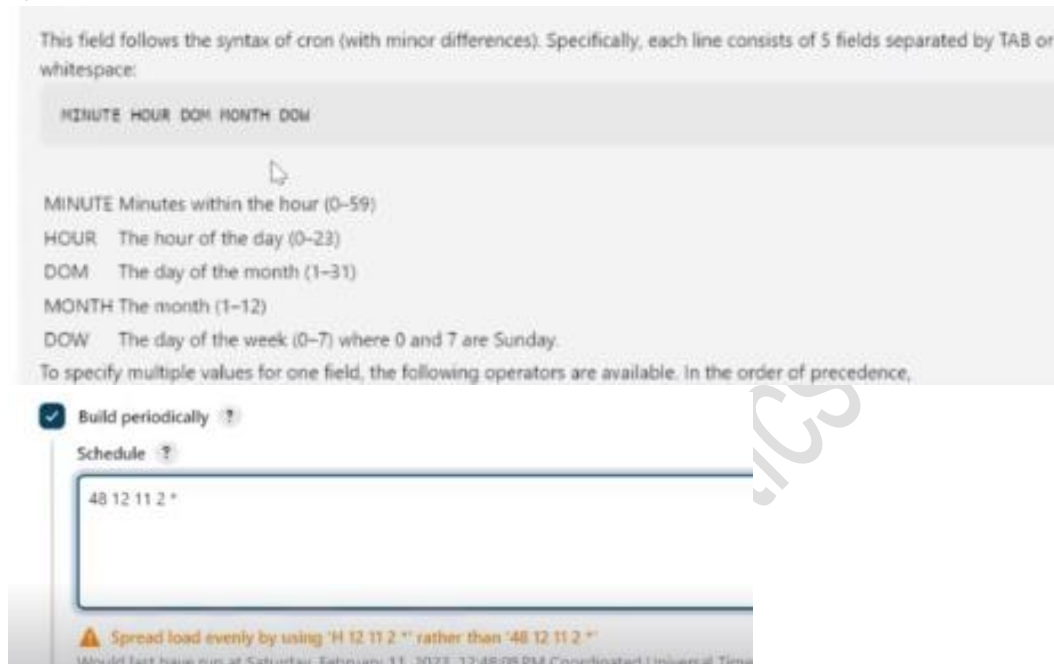


- For triggers we have a lot of options

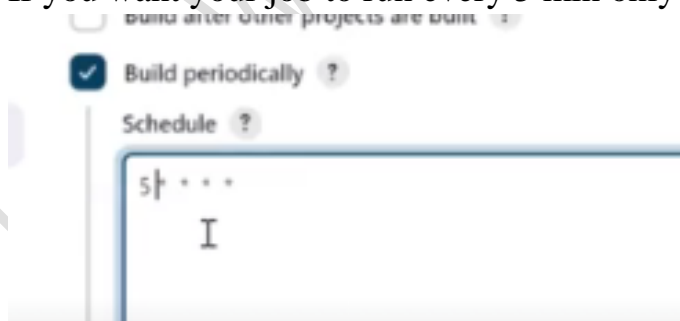


[Jenkins]

- And we use **build periodically trigger**
- **Building periodically** is like an alarm means if you run your job every morning at 7 am then we set the alarm for 7 am
- **At 7 am every day** they run the job automatically means we create a backup
- Below syntax is followed by **build periodically trigger** always write in that syntax



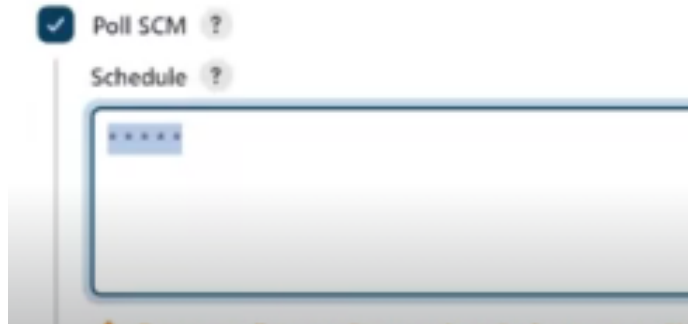
- Then save now it is automated
- If the developer changes anything in the code after pushing the code In GitHub Jenkins does all the automatically
- Without clicking the job they run the job automatic
- If you want your job to run every 5 min only change the alarm



- But this trigger is not good because if your developer does not change the code then we don't need this trigger
- If you use this trigger then all the time they go and download the code and deploy the code again and again
- In the above example we use a **poll SCM trigger**

[Jenkins]

- Because this trigger is a little bit smart if the developer changes the code then this trigger runs the job if the developer does not change the code this trigger does not do anything
- And Both trigger has the same syntax
- Means triggering the job every minute but when the code changes then run the job



- In the **build periodically triggering the** job every minute means they go and download the code every minute
- But In the poll SCM they trigger when code change
- Now we create a full automation project if we log out from your Jenkins but the developer changes the code after that Jenkins triggers the job for you