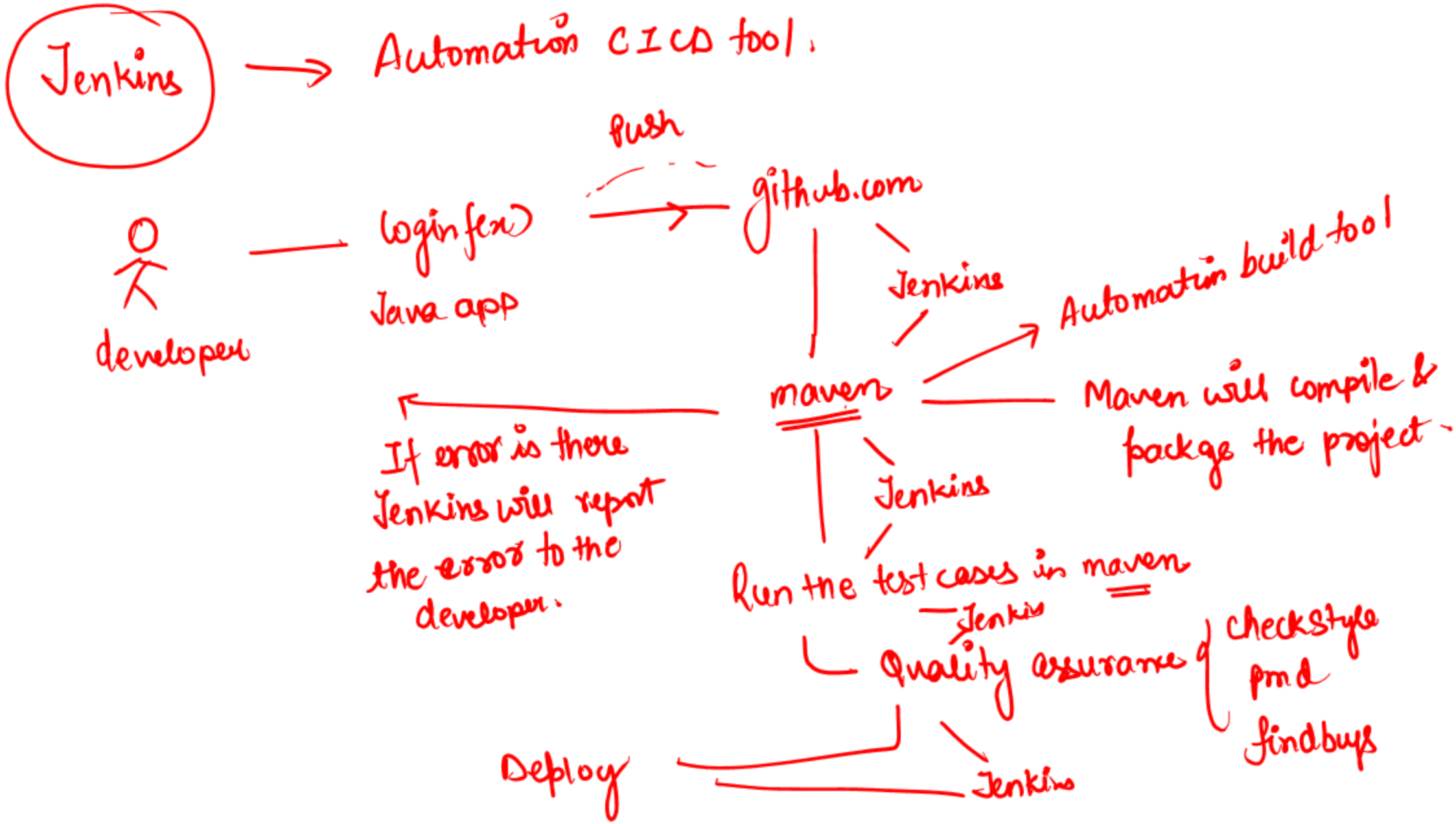
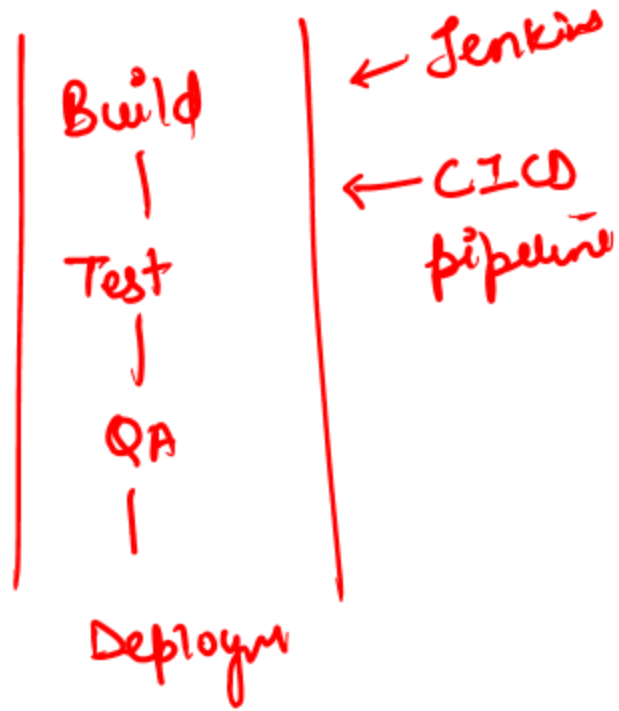
# Jenkins


Jenkins
Automation CICD tool.
Push
github.com
developer
login(ex)
Java app
Jenkins
Automation build tool
maven
Maven will compile & package the project.
If error is there Jenkins will report the error to the developer.
Jenkins
Run the test cases in maven
Jenkins
Quality assurance
checkstyle
pmd
findbugs
Deploy
Jenkins

Jenkins → automation tool used for building, testing and deploying app.

- Open source tool
- Jenkins software developed in Java.
- Jenkins follow plugin based architecture.
  - e.g. → mailer plugin.
    maven plugin.
- To automate the execution of pipeline we write the script in Groovy langugage.
- Jenkins can be Installed in windows, mac, ubuntu, Linux



Maven → Build automation and project management tool

→ It helps building (compiling, packaging), testing Java app automatically
→ (Jar/war)

→ configuration file → pom.xml
→ Dependencies
→ plugins
→ Project Information
→ Build Instructions

} Developer

- https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html

lifecycle handles the creation of your project's web site.

## A Build Lifecycle is Made Up of Phases

Each of these build lifecycles is defined by a different list of build phases, wherein a build phase represents a stage in the lifecycle.

For example, the default lifecycle comprises of the following phases (for a complete list of the lifecycle phases, refer to the Lifecycle Reference):

- `validate` - validate the project is correct and all necessary information is available
- `compile` - compile the source code of the project
- `test` - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
- `package` - take the compiled code and package it in its distributable format, such as a JAR.
- `verify` - run any checks on results of integration tests to ensure quality criteria are met
- `install` - install the package into the local repository, for use as a dependency in other projects locally
- `deploy` - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

These lifecycle phases (plus the other lifecycle phases not shown here) are executed sequentially to complete the `default` lifecycle. Given the lifecycle phases above, th

Gradle — competitor of maven

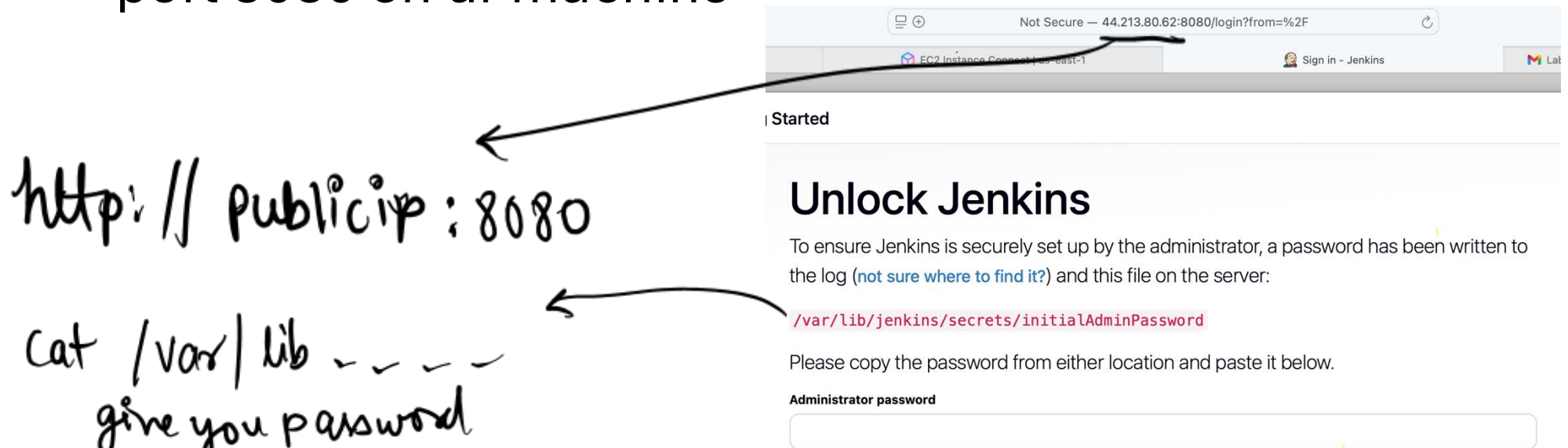
- build automation tool used to build, test and deploy the applications
- you can use gradle for Java, Kotlin / Android apps, Groovy --
- Gradle uses scripts
  - build.gradle (Groovy)
  - build.gradle.kts (kotlin)

# Lab 1 : Install Jenkins in the ubuntu machine

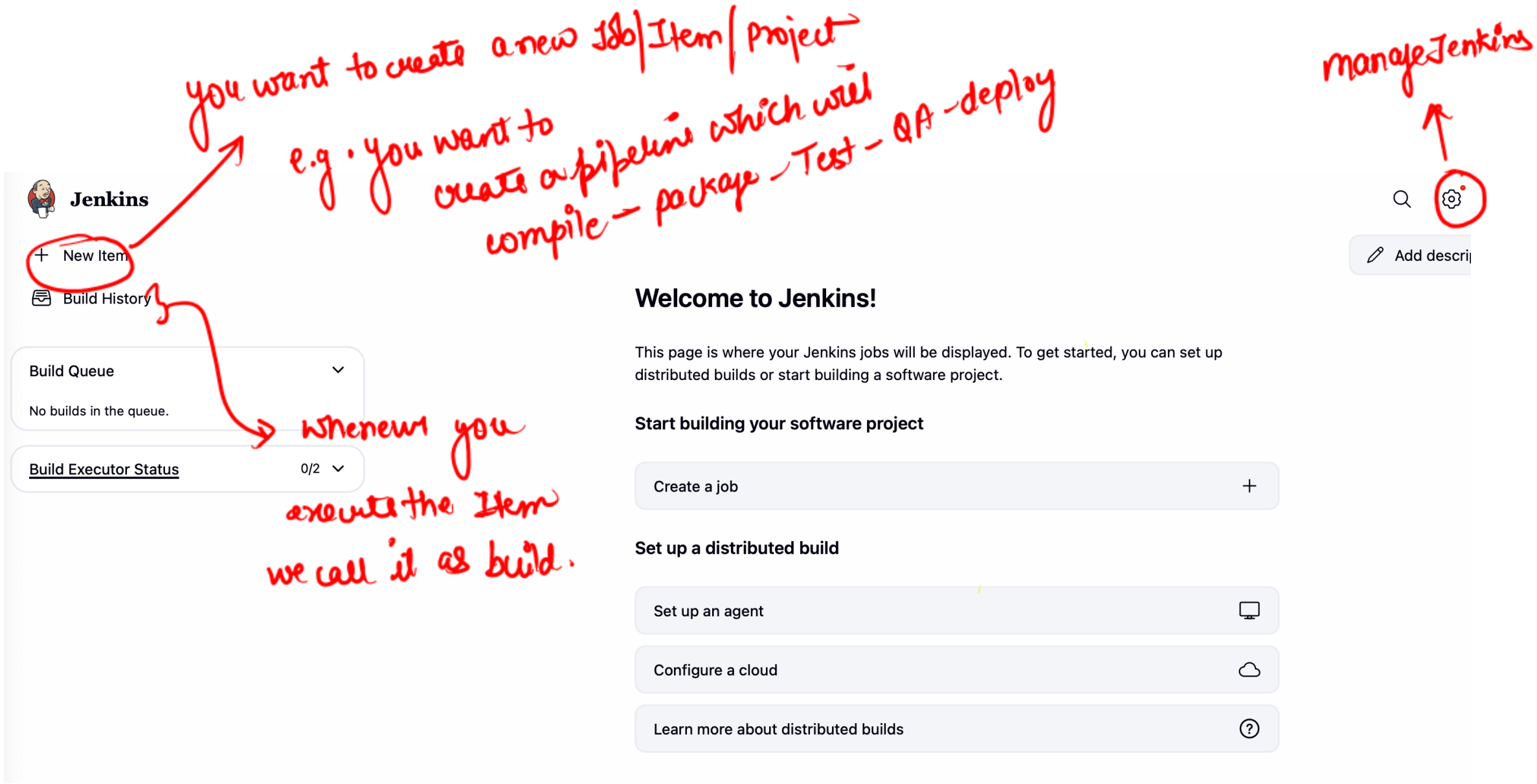
Use the script to download Jenkins

https://raw.githubusercontent.com/akshu20791/Deployment-script/refs/heads/main/jenkins.sh

Jenkins works on port 8080 so you need to allow traffic coming from port 8080 on ur machine



- Install suggested plugins
- Put user , pass and other details and login to jenkins


you want to create a new Job/Item/project
e.g. you want to create a pipeline which will compile - package - Test - QA - deploy
manageJenkins
Jenkins
New Item
Build History
Build Queue
No builds in the queue.
Build Executor Status
0/2
whenever you execute the Item we call it as build.
Add descri
Welcome to Jenkins!
This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.
Start building your software project
Create a job
Set up a distributed build
Set up an agent
Configure a cloud
Learn more about distributed builds

# Lab 2: User management in jenkins

Lets suppose you are working in a team and you have developer , testers ,interns , admin etc in the team to which u don’t want to give your main Jenkins account. So we can create the users in Jenkins and give them the restricted permissions

There are multiple ways of doing it ...
1) Role based access control

2) Matrix based method

Doc link:
https://drive.google.com/file/d/15eDDN5auA10kqJkv-P2wsFDBwqJcK5BS/view?usp=sharing

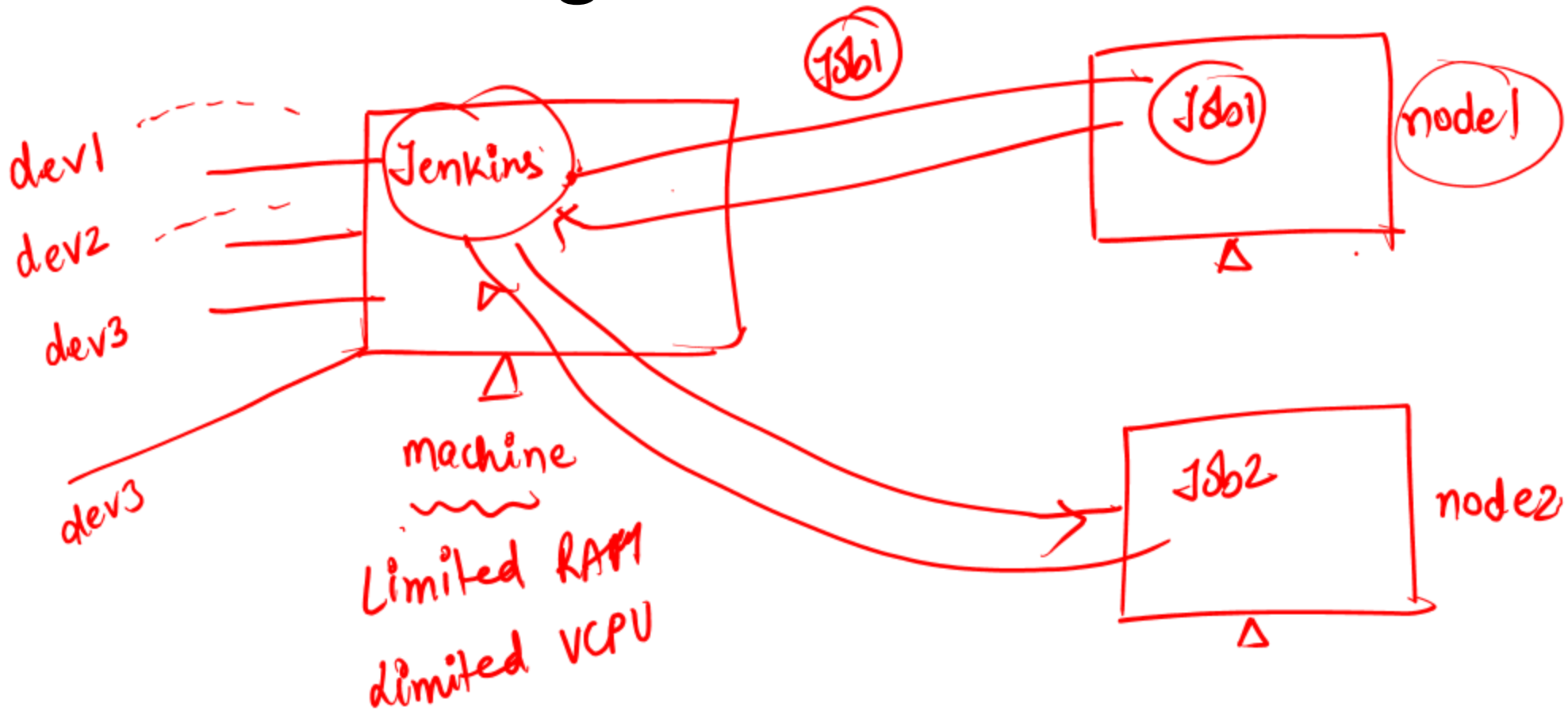
# Lab 3: Create a simple hello world job

- We need to create a job in Jenkins which will print hello world message in the machine and we will see the logs from Jenkins

Solution:

https://drive.google.com/file/d/1q-_ak9eN0X_hCii3UHLEiCSzubGG2oMw/view?usp=sharing

# Lab 4: Working in master node architecture




Jenkins delegates the tasks to the nodes (machines) so as to reduce the load on Jenkins machine (master)

- Go to aws account -> ec2 ->Instances -> Launch a new instance with all traffic enabled and name that machine as node machine
- Connect to the node machine

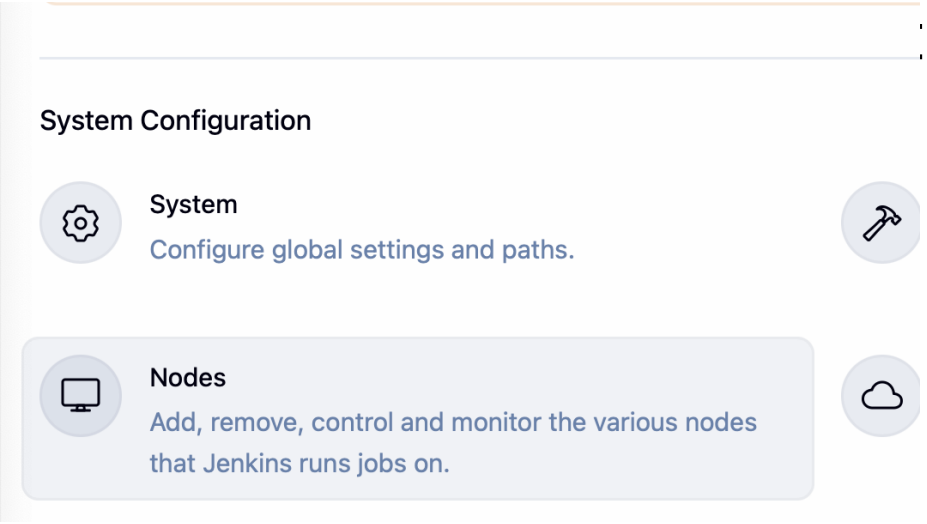
  In the terminal write :

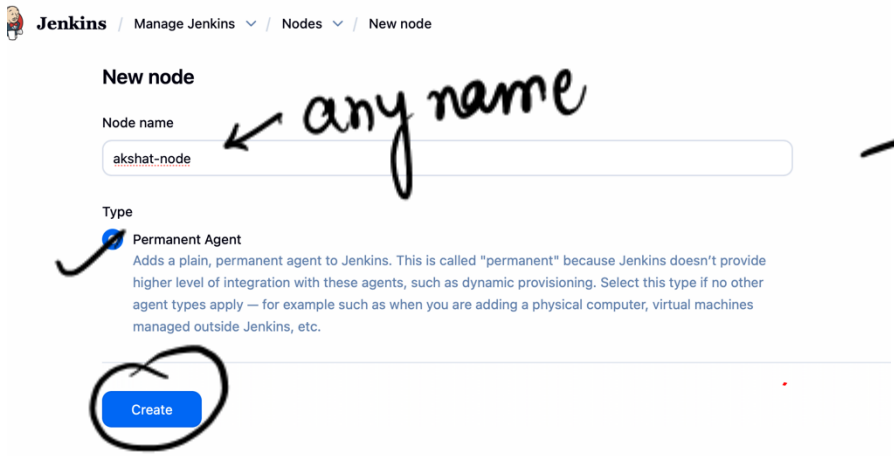
  sudo su

  apt update

  apt install openjdk-17-jdk -y

- Go back to your Jenkins master machine -> manage Jenkins -> In sy nodes -> + NEW NODES






Save

Jenkins / Manage Jenkins / Nodes

Name ?

akshat-node

Description ?

How many Jobs this node can execute concurrently

Plain text Preview

Number of executors ?

1

Remote root directory ?

/opt/akshat

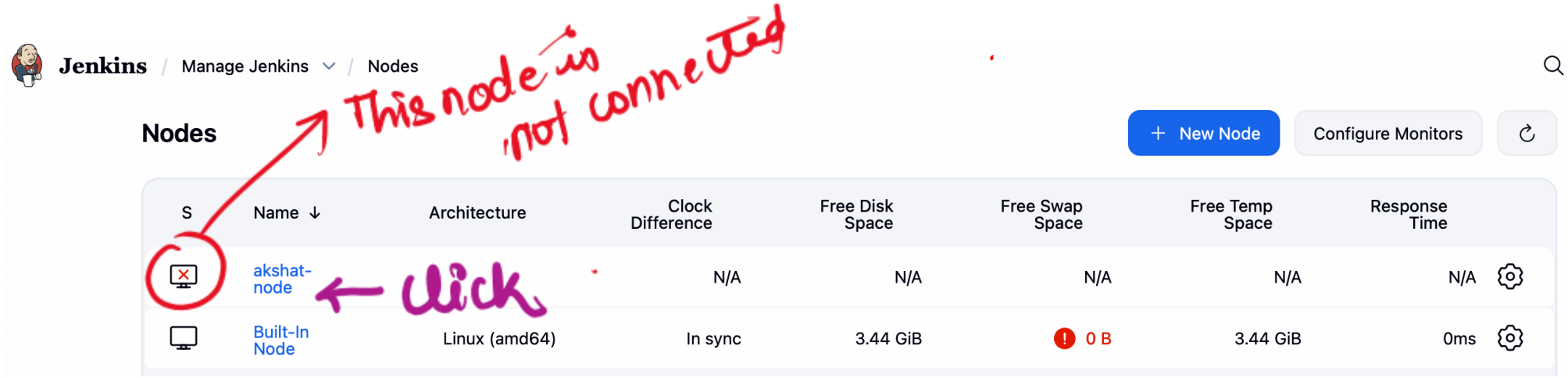
any location (Here your connection files are present)

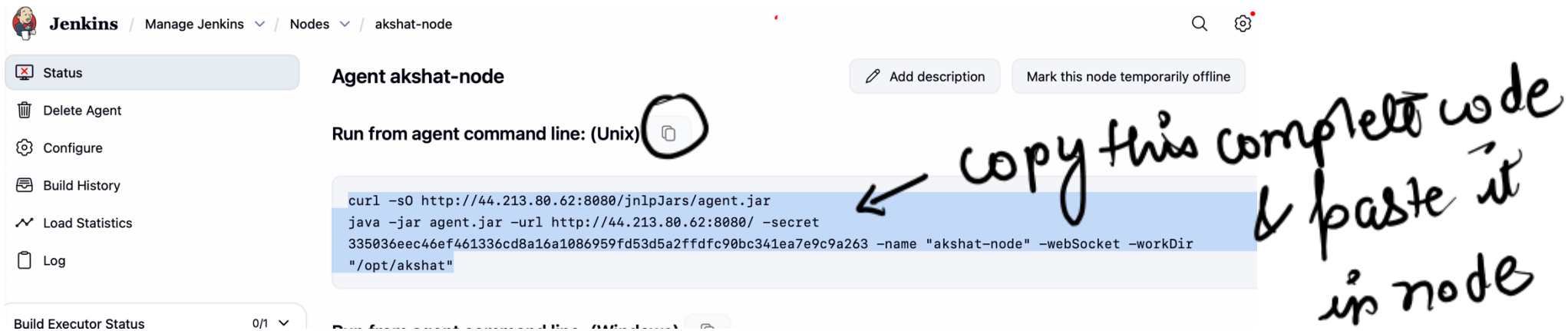
Labels ?

node1

Identifier

Usage ?


Jenkins / Manage Jenkins / Nodes
This node is not connected
Nodes
+ New Node
Configure Monitors
S
Name ↓
Architecture
Clock Difference
Free Disk Space
Free Swap Space
Free Temp Space
Response Time
akshat-node
click
N/A
N/A
N/A
N/A
N/A
Built-In Node
Linux (amd64)
In sync
3.44 GiB
0 B
3.44 GiB
0ms


Jenkins / Manage Jenkins / Nodes / akshat-node
Status
Delete Agent
Configure
Build History
Load Statistics
Log
Build Executor Status 0/1
Agent akshat-node
Add description
Mark this node temporarily offline
Run from agent command line: (Unix)
curl -sO http://44.213.80.62:8080/jnlpJars/agent.jar
java -jar agent.jar -url http://44.213.80.62:8080/ -secret
335036eec46ef461336cd8a16a1086959fd53d5a2ffdfc90bc341ea7e9c9a263 -name "akshat-node" -webSocket -workDir
"/opt/akshat"
copy this complete code & paste it in node

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-66-234:/home/ubuntu# curl -sO http://44.213.80.62:8080/jnlpJars/agent.jar
java -jar agent.jar -url http://44.213.80.62:8080/ -secret 335036eec46ef461336cd8a16a1086959fd53d5a2ffdfc90bc341ea7e9c9a263 -name "akshat-node" -webS
rkDir "/opt/akshat"
Nov 18, 2025 11:12:58 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /opt/akshat/remoting as a remoting work directory
Nov 18, 2025 11:12:58 AM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /opt/akshat/remoting
Nov 18, 2025 11:12:58 AM hudson.remoting.Launcher createEngine
INFO: Setting up agent: akshat-node
Nov 18, 2025 11:12:58 AM hudson.remoting.Engine startEngine
INFO: Using Remoting version: 3327.v868139a_d00e0
Nov 18, 2025 11:12:58 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /opt/akshat/remoting as a remoting work directory
Nov 18, 2025 11:12:58 AM hudson.remoting.Launcher$CuiListener status
INFO: WebSocket connection open
Nov 18, 2025 11:12:58 AM hudson.remoting.Launcher$CuiListener status
INFO: Connected
```

**i-069660b9f5bf64f88 (node-akshat-jenkins)**

PublicIPs: 18.206.16.190 PrivateIPs: 172.31.66.234

(node machine)

Jenkins / Manage Jenkins / Nodes

## Nodes

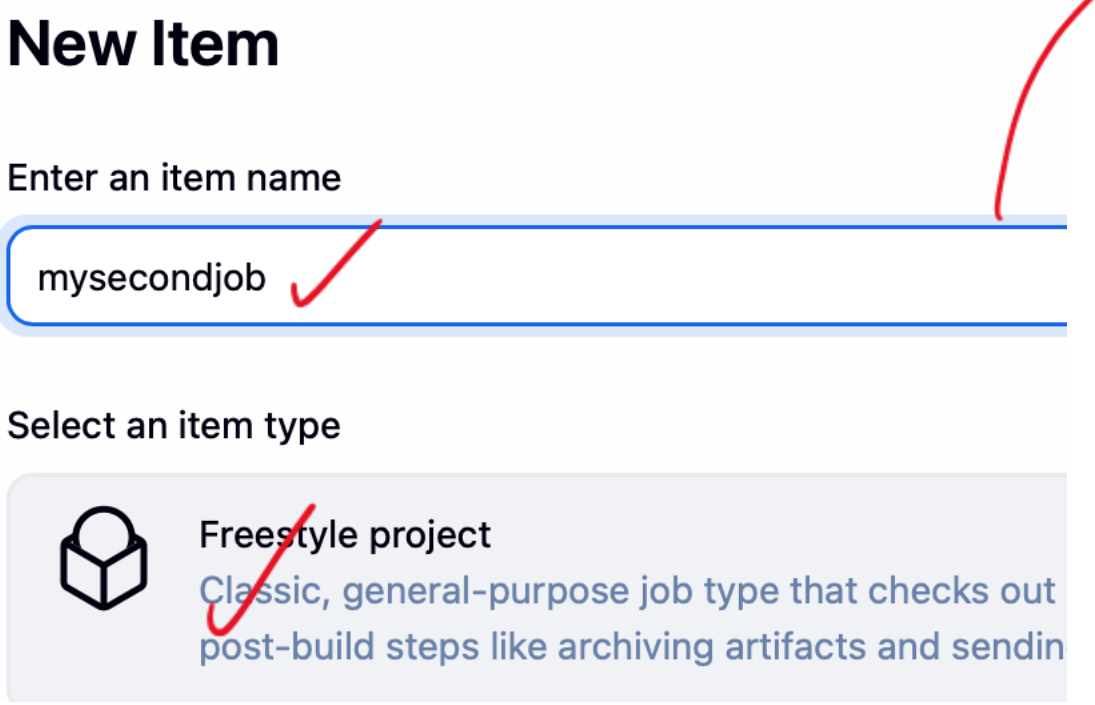
X mark is removed

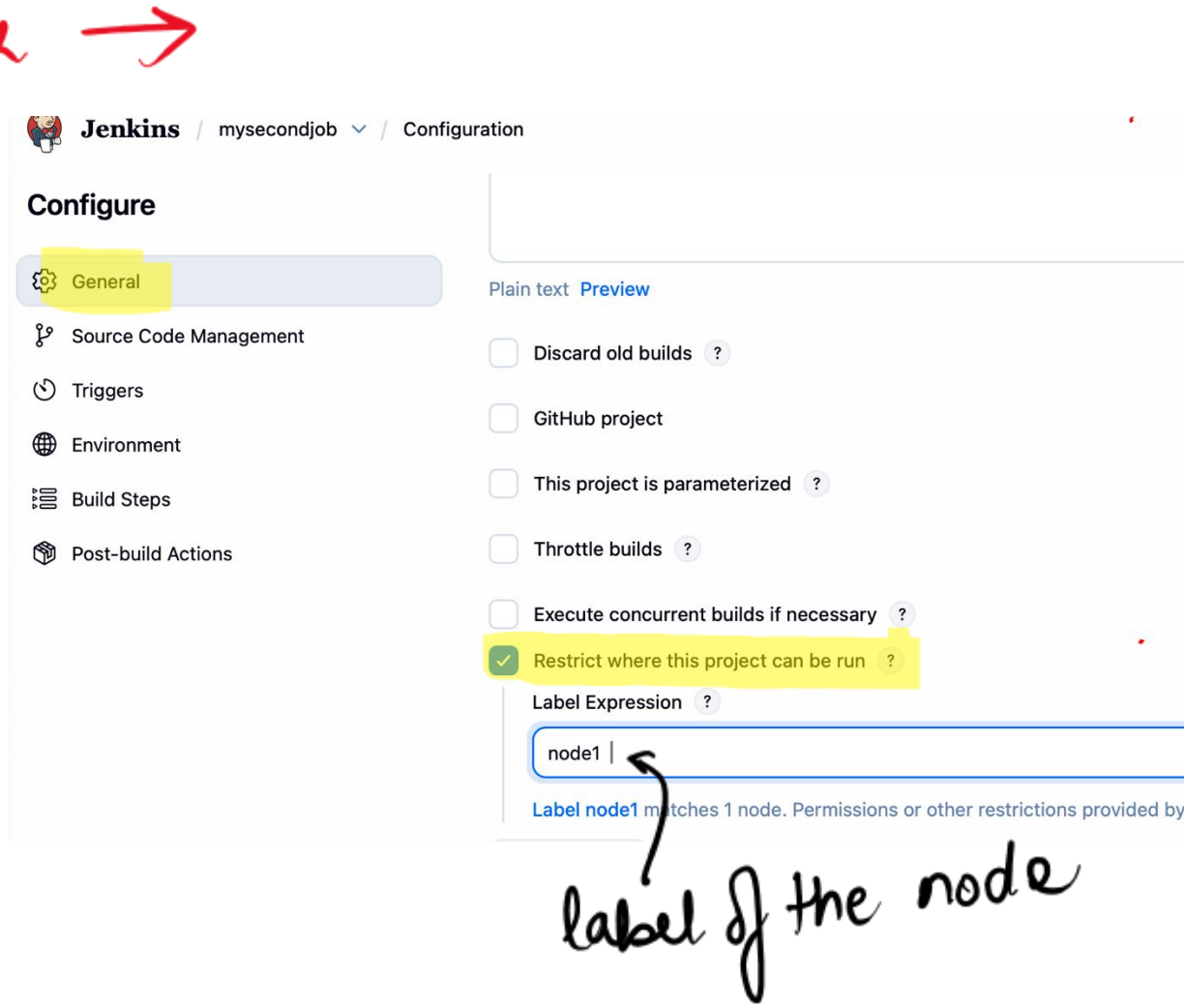
| S | Name ↓ | Architecture | Clock Difference | Free Disk Space | Free Swa Spac |
|---|---|---|---|---|---|
| | akshat-node | Linux (amd64) | In sync | 4.09 GiB | |
| | Built-In Node | Linux (amd64) | In sync | 3.44 GiB | |
| | **Data obtained** | **35 sec** | **35 sec** | **35 sec** | **3** |

- Lets run the job in the node

Go to ur Jenkins machine

+new item



Jenkins / mysecondjob / Configuration

## Configure

- General
- Source Code Management
- Triggers
- Environment
- Build Steps
- Post-build Actions

Use secret text(s) or file(s)

Add timestamps to the Console Output

Inspect build log for published build scans

Terminate a build if it's stuck

With Ant

### Build Steps

Automate your build process with ordered tasks like code compilation, testing,

**Execute shell**

Command

See the list of available environment variables

```
echo hello world
```

Save and Build

Jenkins / mysecondjob

- Status
- Changes
- Workspace
- Build Now
- Configure
- Delete Project
- Rename
- Credentials

**mysecondjob**

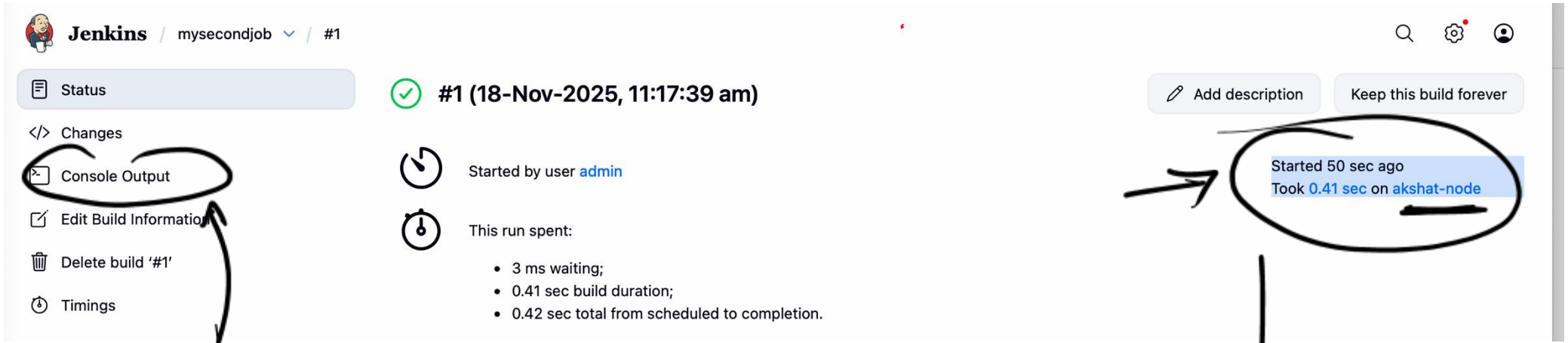
**Permalinks**

Builds

Today

#1 11:[illegible] am

click (after build)




go see the logs

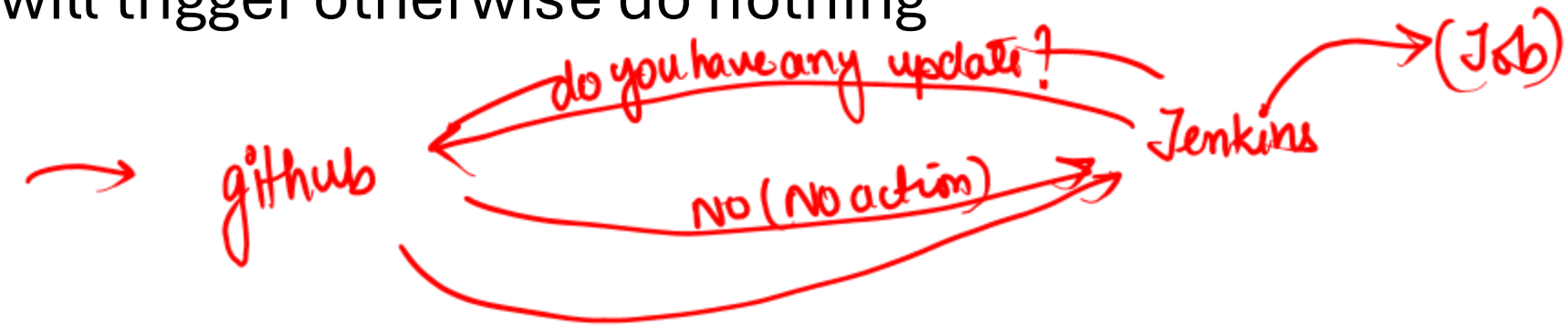
Hence proved that our job is executed in akshatnode.

# Lab 5: Automate the Jenkins job whenever the developer push the code to github

- As of now if the developer push the code to github it does not automatically build the project …we need to manually build the project by clicking on build now button

- This could be done via multiple ways like :

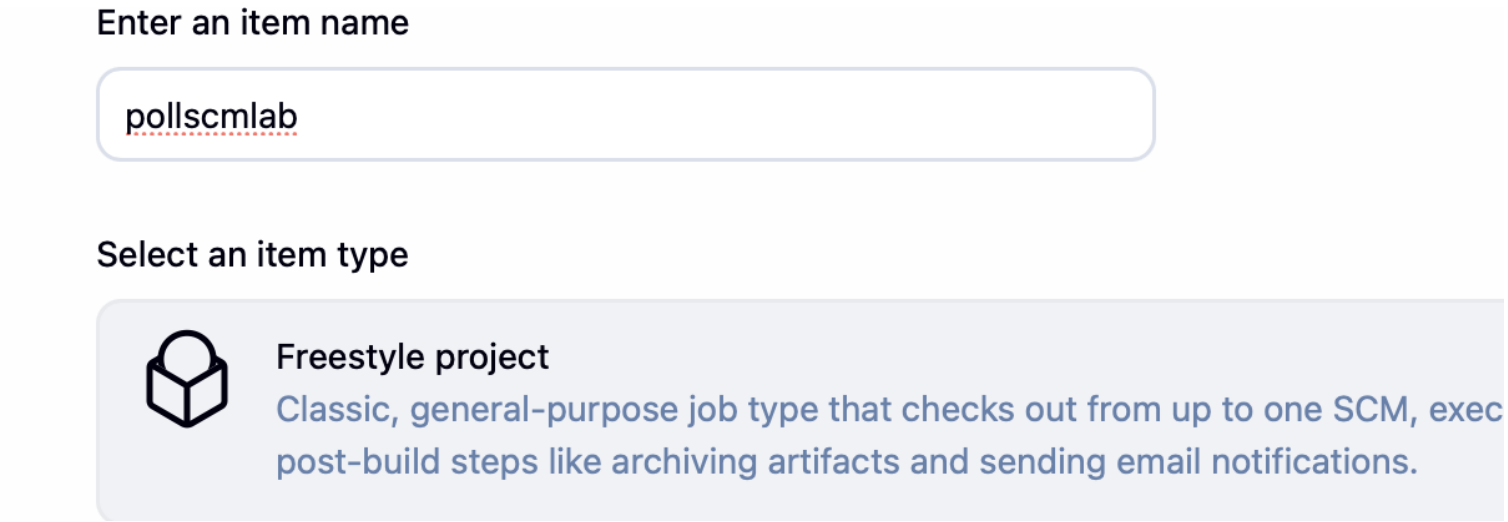
> POLL SCM : Jenkins will keep checking the github for updates after a particular duration (CRON JOB ) and if there is any update in github it will trigger otherwise do nothing




> GITHUB WEBHOOK : Whenever there is any update in github ...its github who will go to Jenkins and trigger the job....

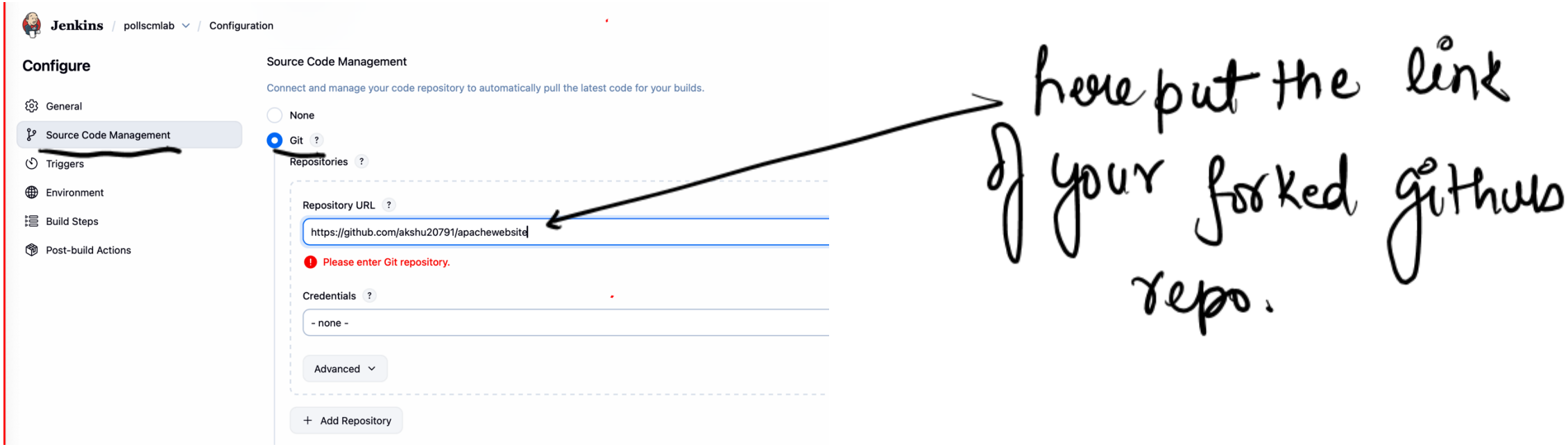
# LAB 5.1 : POLL SCM LAB

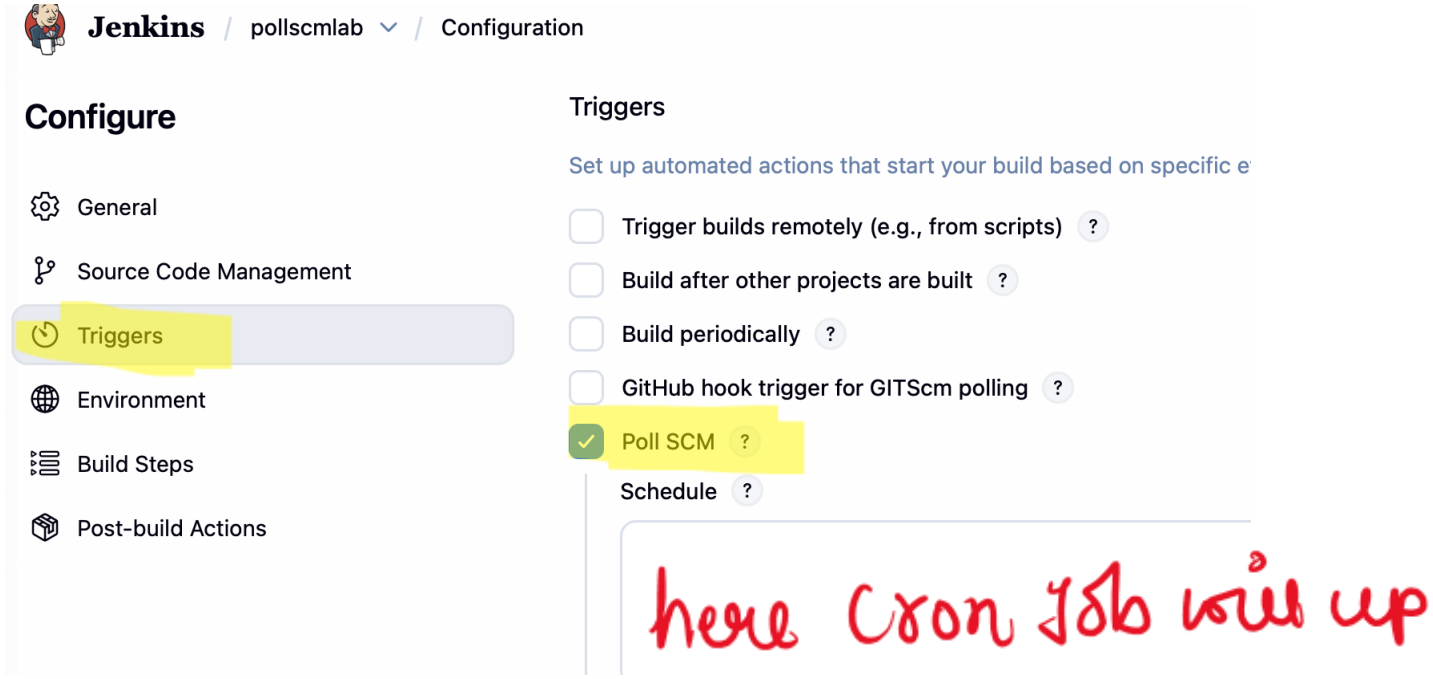
- Go to Jenkins -> +NEW ITEM -> Give any name and freestyle job



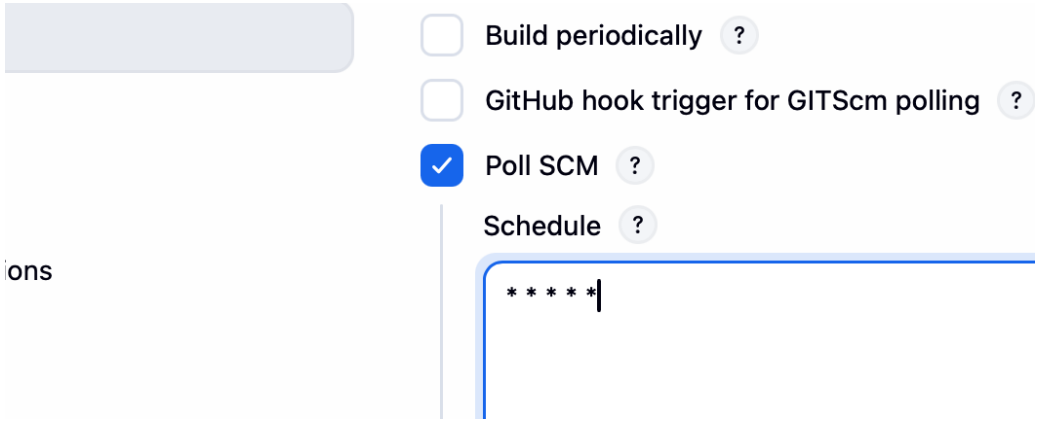

FORK THE GITHUB REPO :
https://github.com/akshu20791/apachewebsite

(Google if you don’t know how to fork)


Jenkins / pollscmlab / Configuration
Configure
General
Source Code Management
Triggers
Environment
Build Steps
Post-build Actions
Source Code Management
Connect and manage your code repository to automatically pull the latest code for your builds.
None
Git
Repositories
Repository URL
https://github.com/akshu20791/apachewebsite
Please enter Git repository.
Credentials
- none -
Advanced
Add Repository
here put the link of your forked github repo.


Jenkins / pollscmlab / Configuration
Configure
General
Source Code Management
Triggers
Environment
Build Steps
Post-build Actions
Triggers
Set up automated actions that start your build based on specific e
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
GitHub hook trigger for GITScm polling
Poll SCM
Schedule
here Cron Job will up

- https://crontab.guru
- * * * *


In build steps -> add build steps

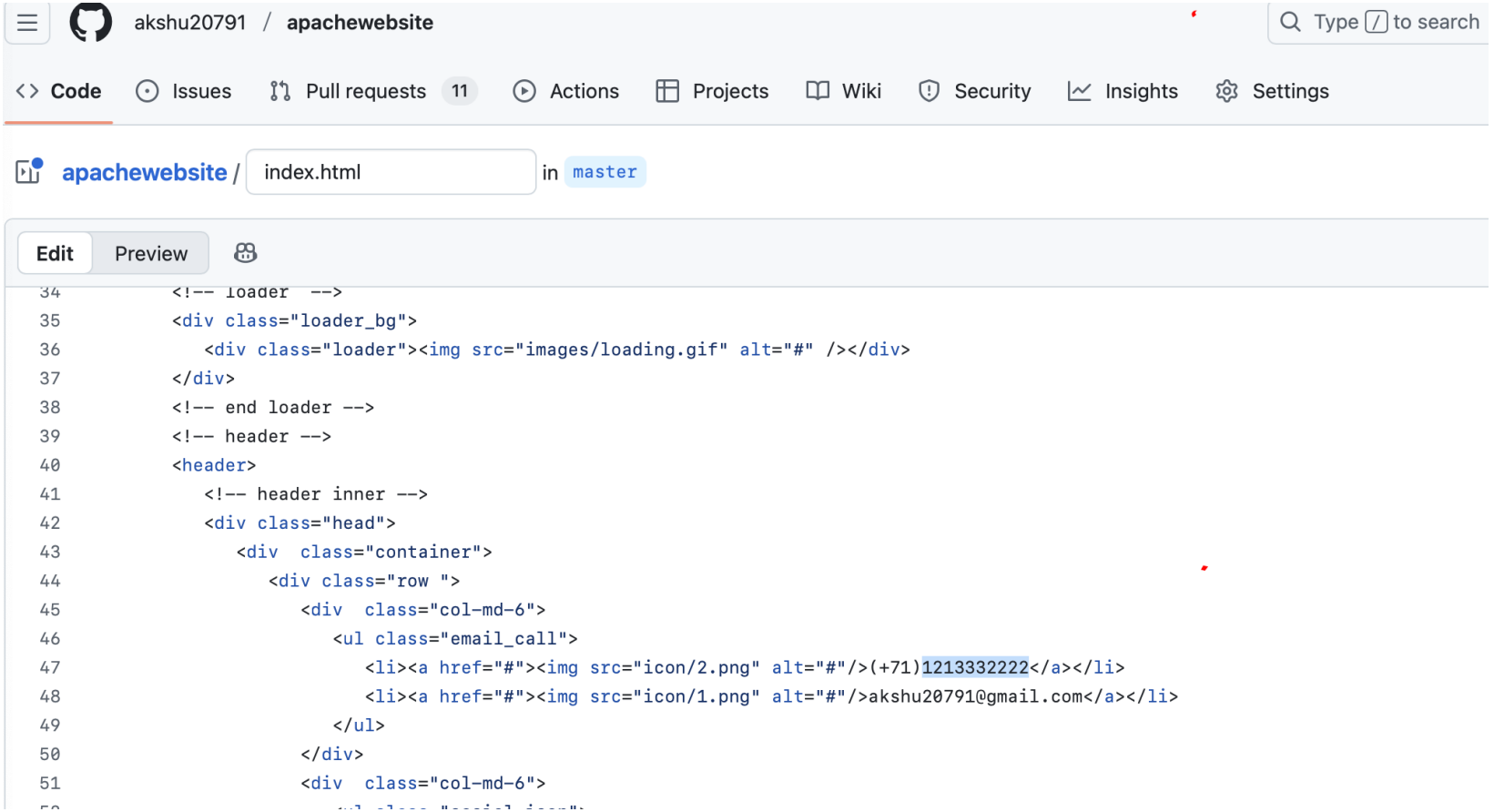
execute shell

echo hello world

Save

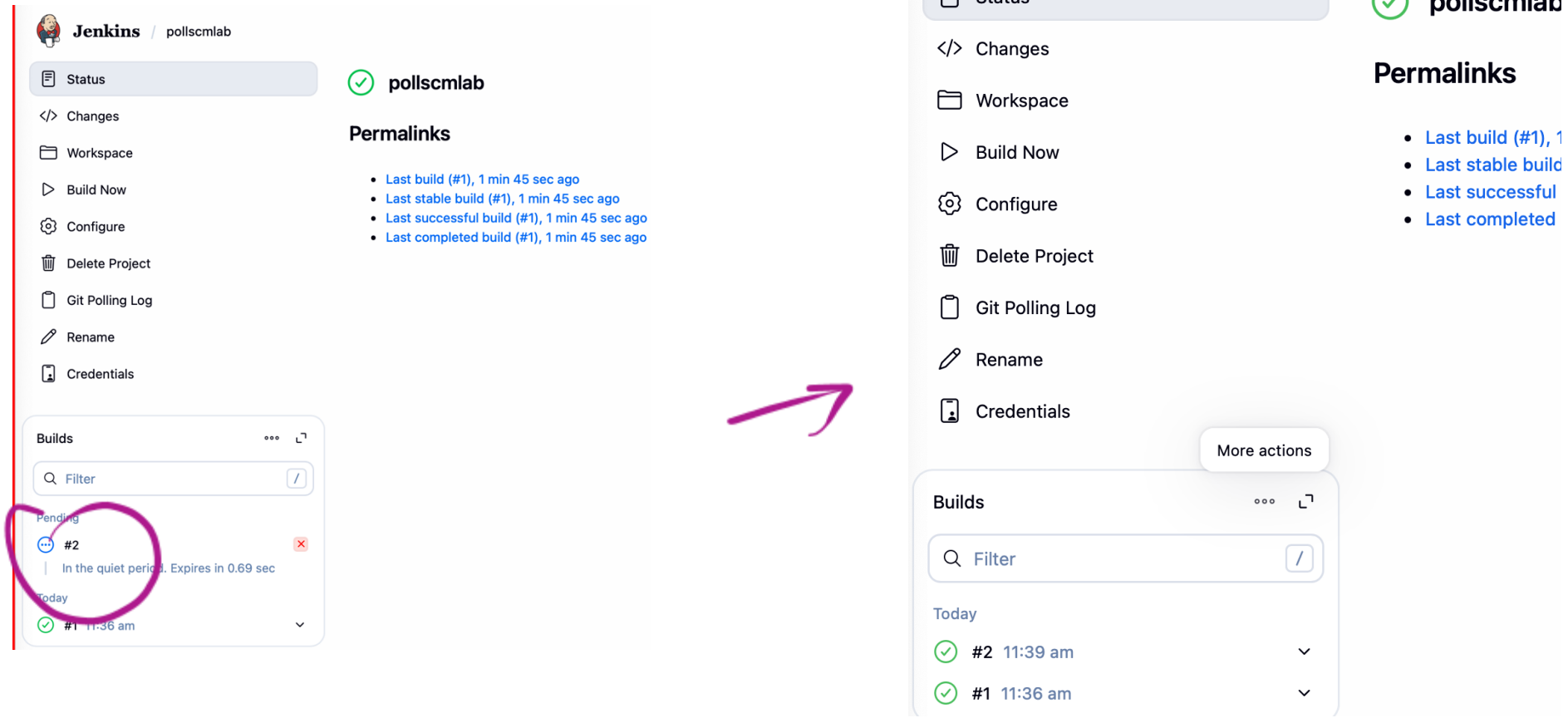
Build now

- Go to the github account -> go to the repo which u have forked -> edit the index.html file and add any content




Commit

- Keep checking the Jenkins ...u will see automatically the Jenkins job will be trigger after 1 min because u have made some changes in github which is automatically identified by Jenkins.

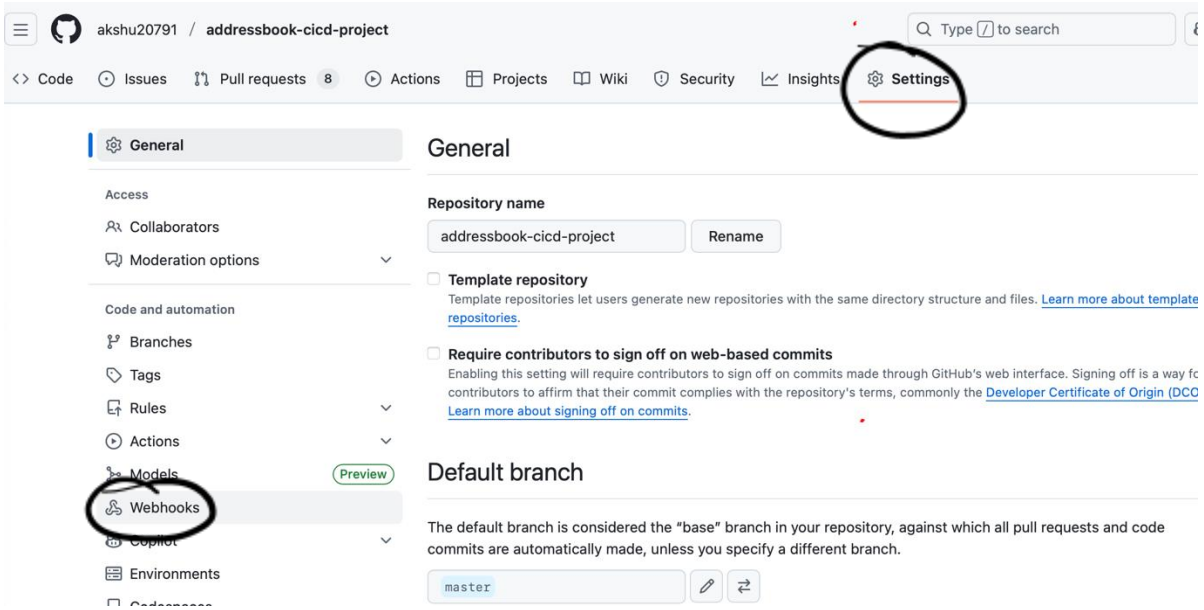


# Lab 5.2: Github webhooks

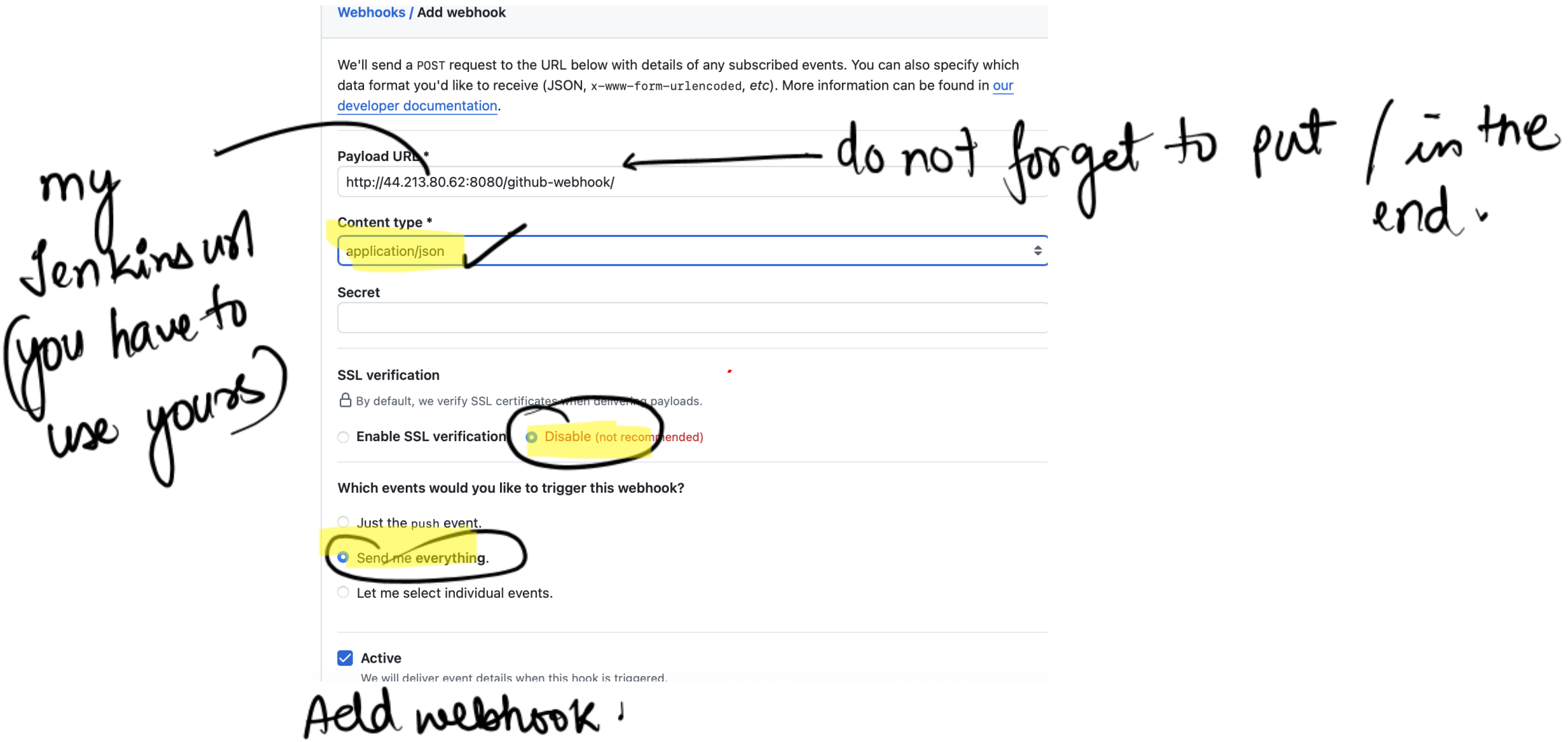
- Fork https://github.com/akshu20791/addressbook-cicd-project

(you will be using ur forked github repo everywhere)

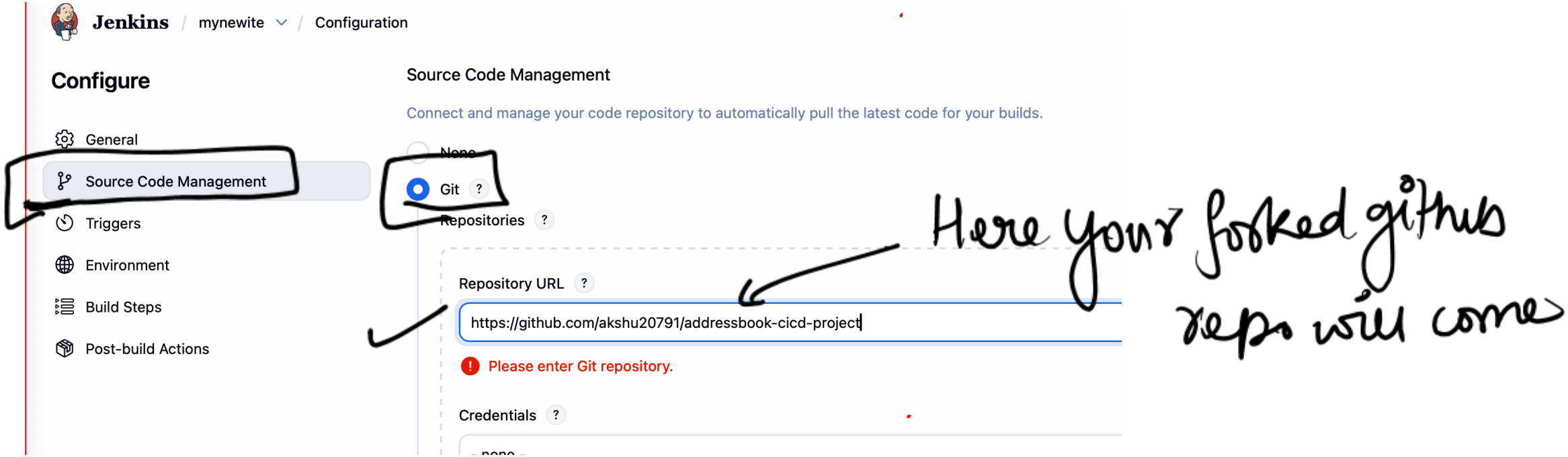
Go to github -> go to ur githubrepo -> setings → webhooks



# Click on Add webhook -> in payload url : put your Jenkins url http://publicip:8080/github-webhook/



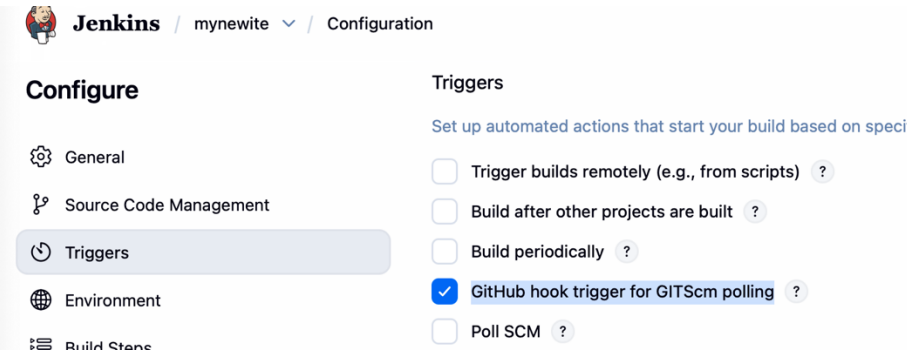
- Go to Jenkins -> + new item -> free style item



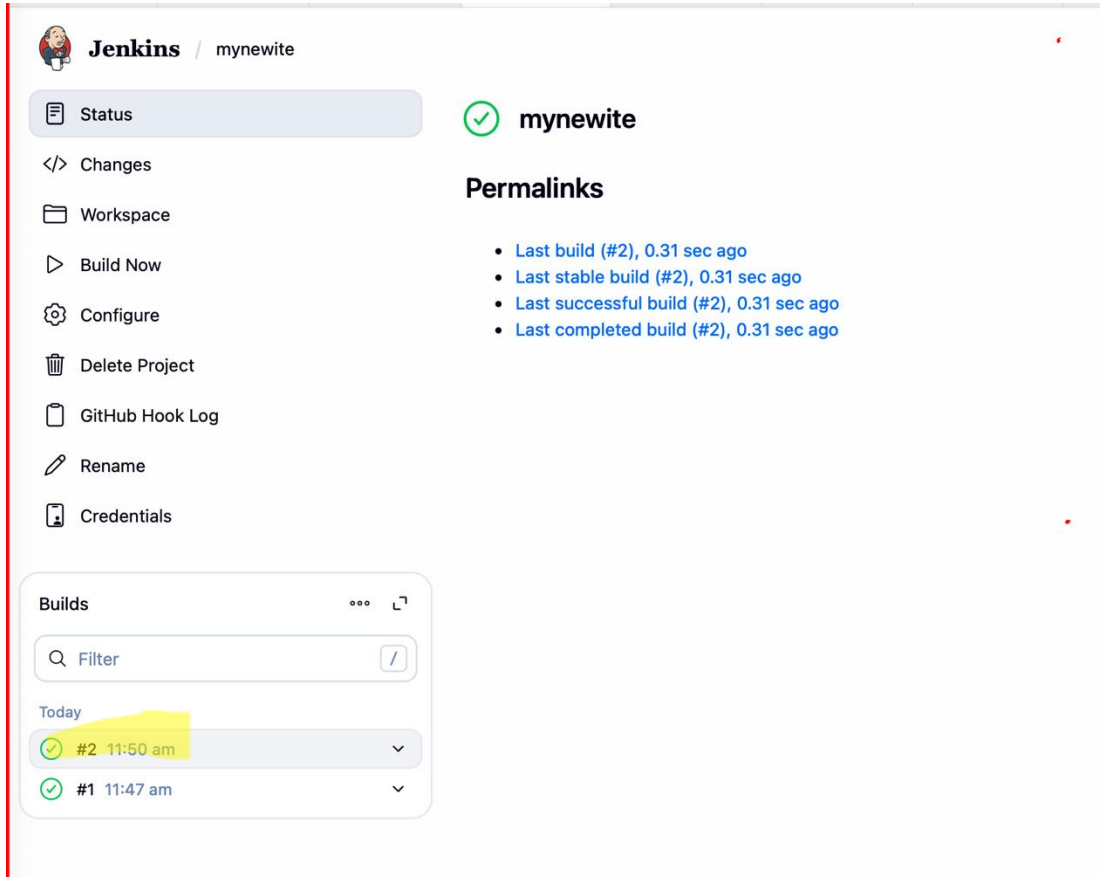

In trigger -> select GitHub hook trigger for GITScm polling

In build step -> add build step -> Execute shell -> echo hello world

save



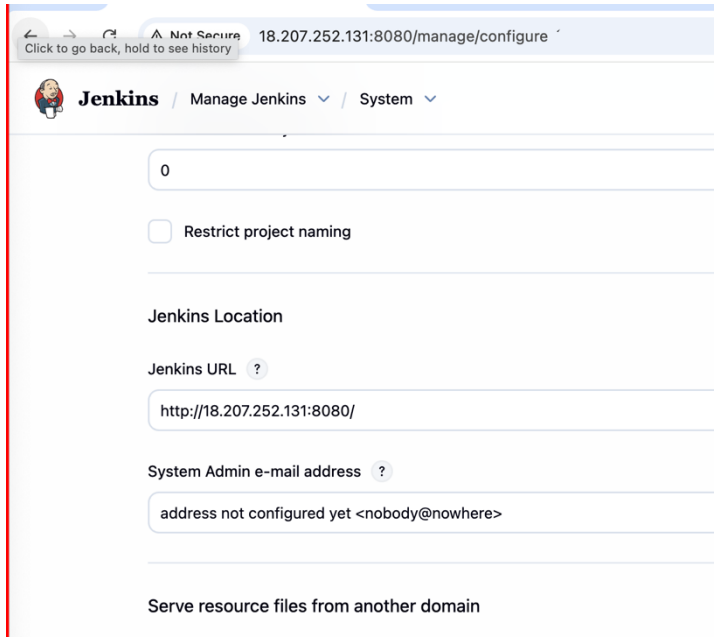
- Now build the job once
- Now go to github and edit readme file and add any content in it



# Lab 6: Slow Jenkins due to public ip change

- If you restart the machine you will reliase the public ip of the machine is changes your jenkins will open on the new public ip but it will be slow. To bring to the same speed follow:

  Go to manage Jenkins -> System -> Scroll down in Jenkins url update the latest url



# Lab 7: Forget admin pass in jenkins

- Go to the machine where Jenkins is running

> sudo su

> service jenkins stop

> cd /var/lib/jenkins

> vi config.xml

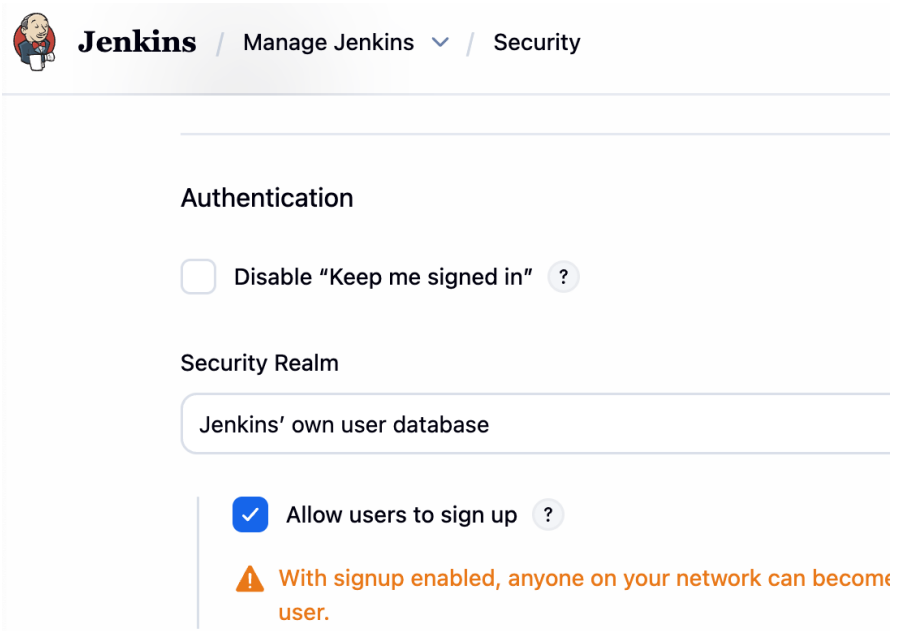
# find the line <useSecurity>true</useSecurity>

Change it to :
<useSecurity>false</useSecurity>

> service jenkins restart

**Jenkins will run without any login .....now lets create a new password**

- Go to manage Jenkins -> Security –>
  Security realm : Jenkins own user database
  Tick Allow users to signup




- Go back to manage Jenkins you will see

Users

Click on users -> click on your root user

-> Click on security ->there you can change

The pass

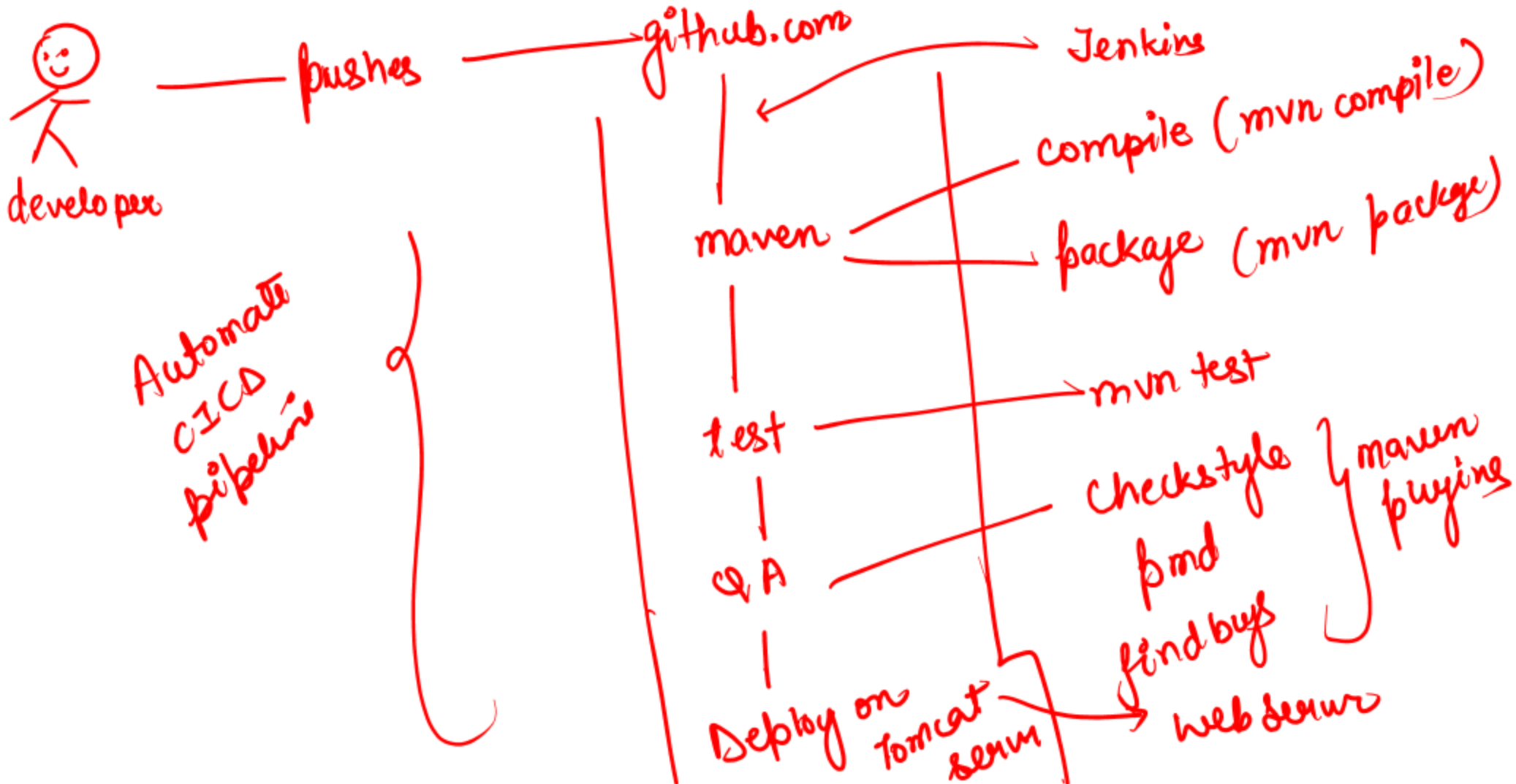
- After this is done ...go to your Jenkins machine
  - sudo su
  - service jenkins stop

    cd /var/lib/jenkins

    vi config.xml

Change <useSecurity>false</useSecurity> to <useSecurity>true</useSecurity>

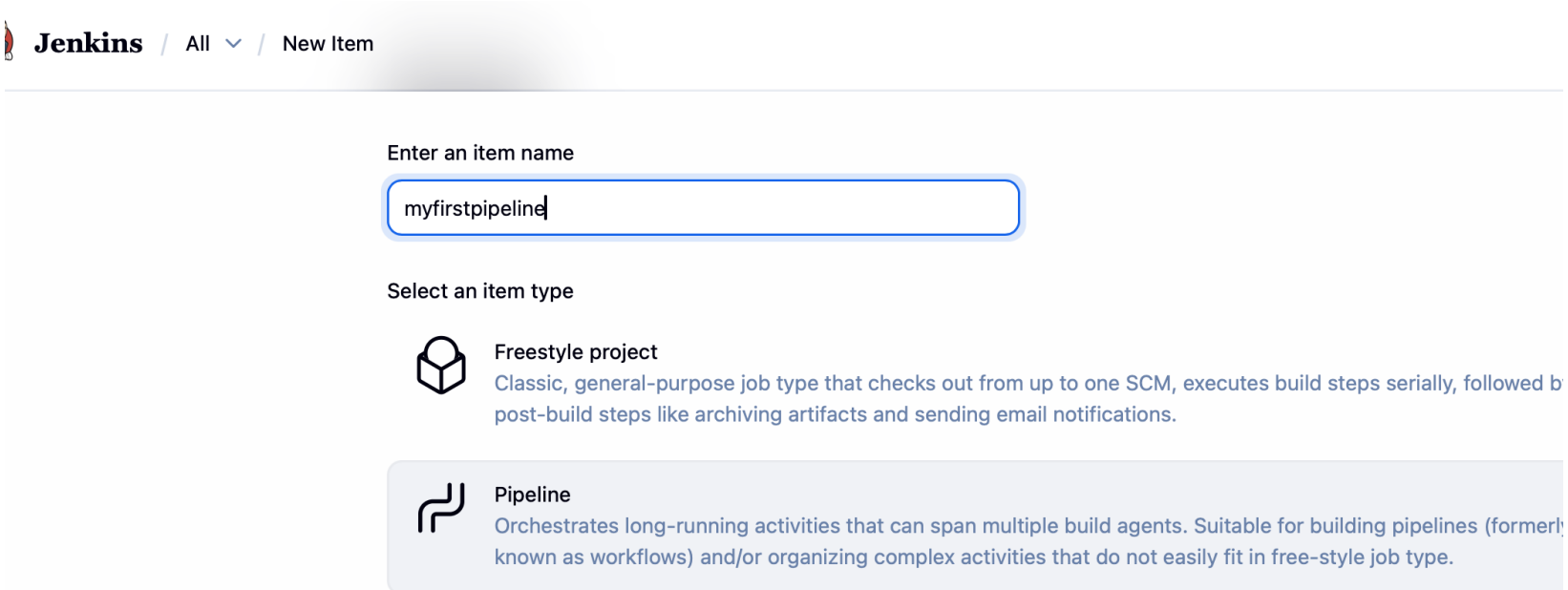
# Lab 8: Create end to end CICD pipeline



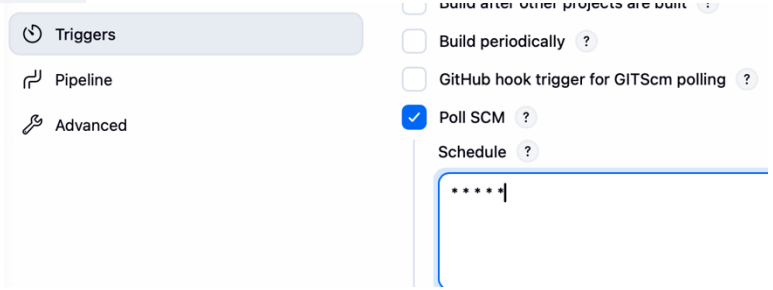
- Fork the repo : https://github.com/akshu20791/addressbook-cicd-project

(if not done yet)

- Go to jenkins -> + New item -> give some name and select pipeline




- Click on trigger on left side -> poll scm -> * * * * *



- Lets suppose your github repo is private:

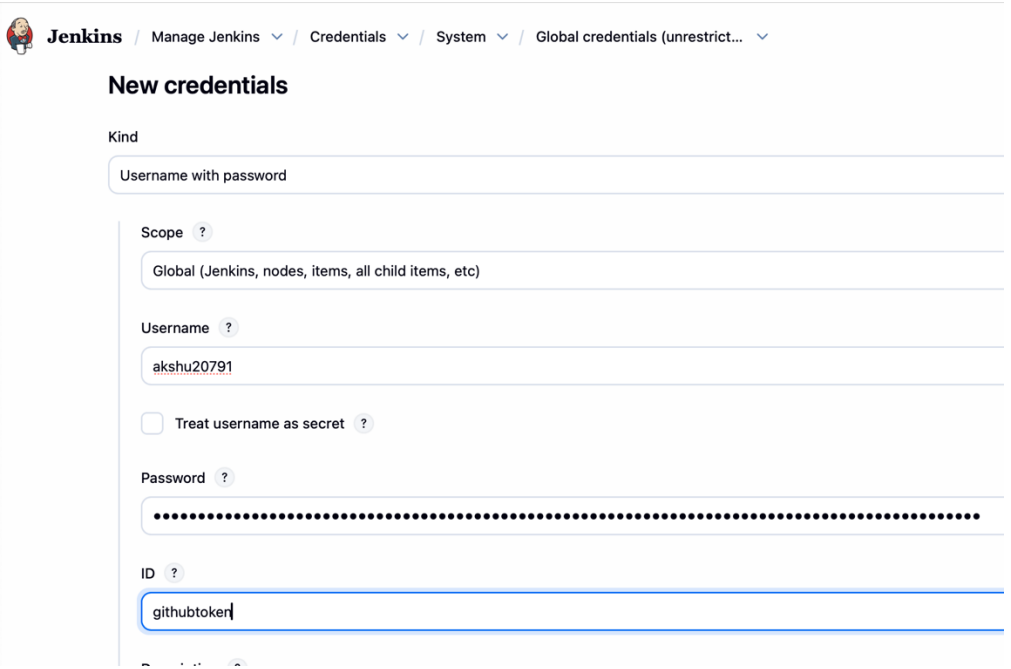
Go to manage jenkins -> credentials -> In front of system click on global -> + add credentials
Kind: Username and pass

Username: your github username

Pass: personal access token

Id: githubtoken

create



Script ?

```
pipeline{
    agent any
    stages{
        stage("checkout the code"){
         steps{
            git credentialsId: 'githubtoken', url: 'https://github.com/akshu20791/addressbook-cicd-project/'
            echo "git checkout done"
         }
        }
        stage("compile the code"){
            steps{
                sh "mvn compile"
            }
        }
        stage("package the code"){
            steps{
                echo "we are packaging the project"
                sh "mvn package"
            }
        }
        stage("run the test cases"){
            steps{
                echo "run the test cases"
                sh "mvn test"
            }
        }
    }
}
```

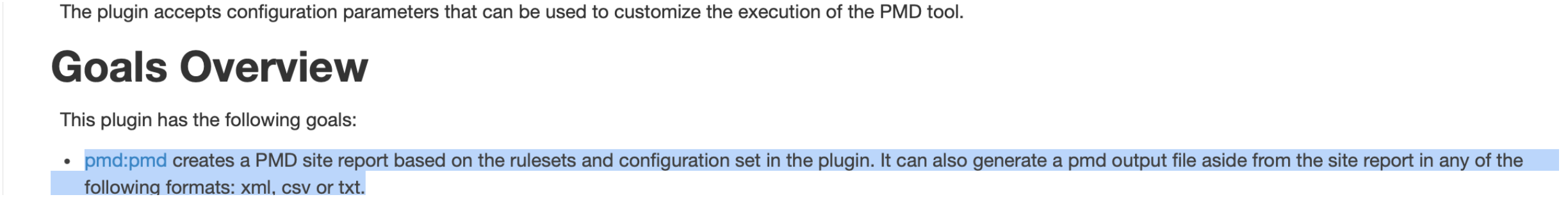
Save
And build to see if the code is correct till this point

- Now we need to do Quality assurance

- To perform QA -> go to manage jenkins -> Plugins -> Available plugins -> Search for warnings plugin (https://plugins.jenkins.io/warnings-ng) in the search bar of available plugins -> Select and install the plugin

- Go back to the job and lets add QA part in the script as well

https://maven.apache.org/plugins/maven-pmd-plugin/



The plugin accepts configuration parameters that can be used to customize the execution of the PMD tool.

## Goals Overview

This plugin has the following goals:

- pmd:pmd creates a PMD site report based on the rulesets and configuration set in the plugin. It can also generate a pmd output file aside from the site report in any of the following formats: xml, csv or txt.

So the goal will be pmd:pmd and it will generate the report as well for you !!!

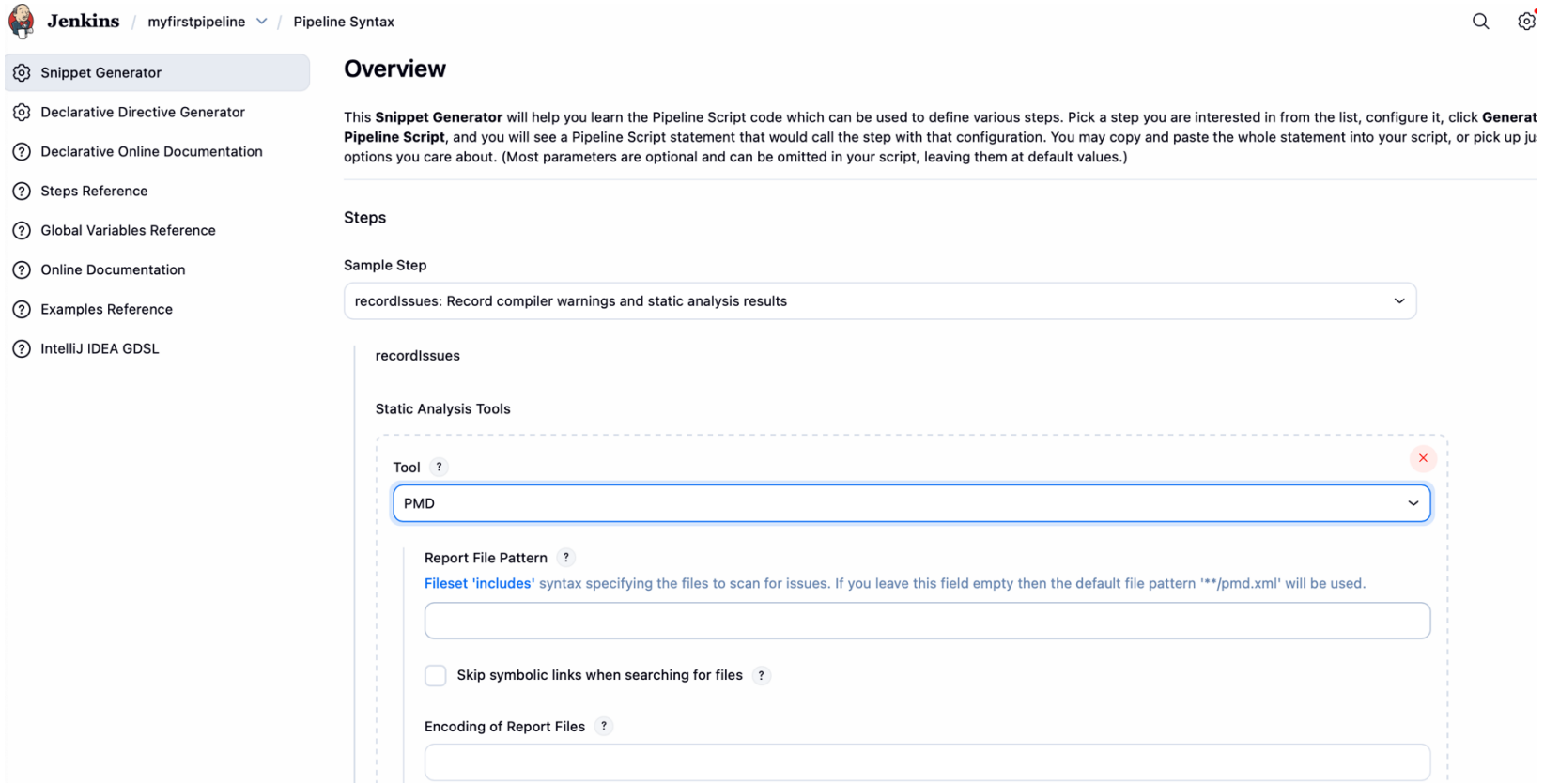
```
13              }
14          }
15          stage("package the code"){
16              steps{
17                  echo "we are packaging the project"
18                  sh "mvn package"
19              }
20          }
21          stage("run the test cases"){
22              steps{
23                  echo "run the test cases"
24                  sh "mvn test"
25              }
26          }
27          stage("qa via pmd plugin"){
28              steps{
29                  echo "performing qa"
30                  sh "mvn pmd:pmd"
31              }
32          }
33      }
```

Use Groovy Sandbox ?

Pipeline Syntax

- We want to generate the reports as well....so how will I get the code??

click on pipeline syntax




Generate the pipeline script

Copy the script which comes in front of you

And we will put it in the code

```
        }
        stage("run the test cases"){
            steps{
                echo "run the test cases"
                sh "mvn test"
            }
        }
        stage("qa via pmd plugin"){
            steps{
                echo "performing qa"
                sh "mvn pmd:pmd"
                recordIssues sourceCodeRetention: 'LAST_BUILD', tools: [pmdParser()]
            }
        }
    }
}
```

se Groovy Sandbox ?

- Now lets build the project

Save and click on build now

After build is successful lets now check the logs

1 match | Contains | addressbook.war

Jenkins / myfirstpipeline / #7 / Console Output

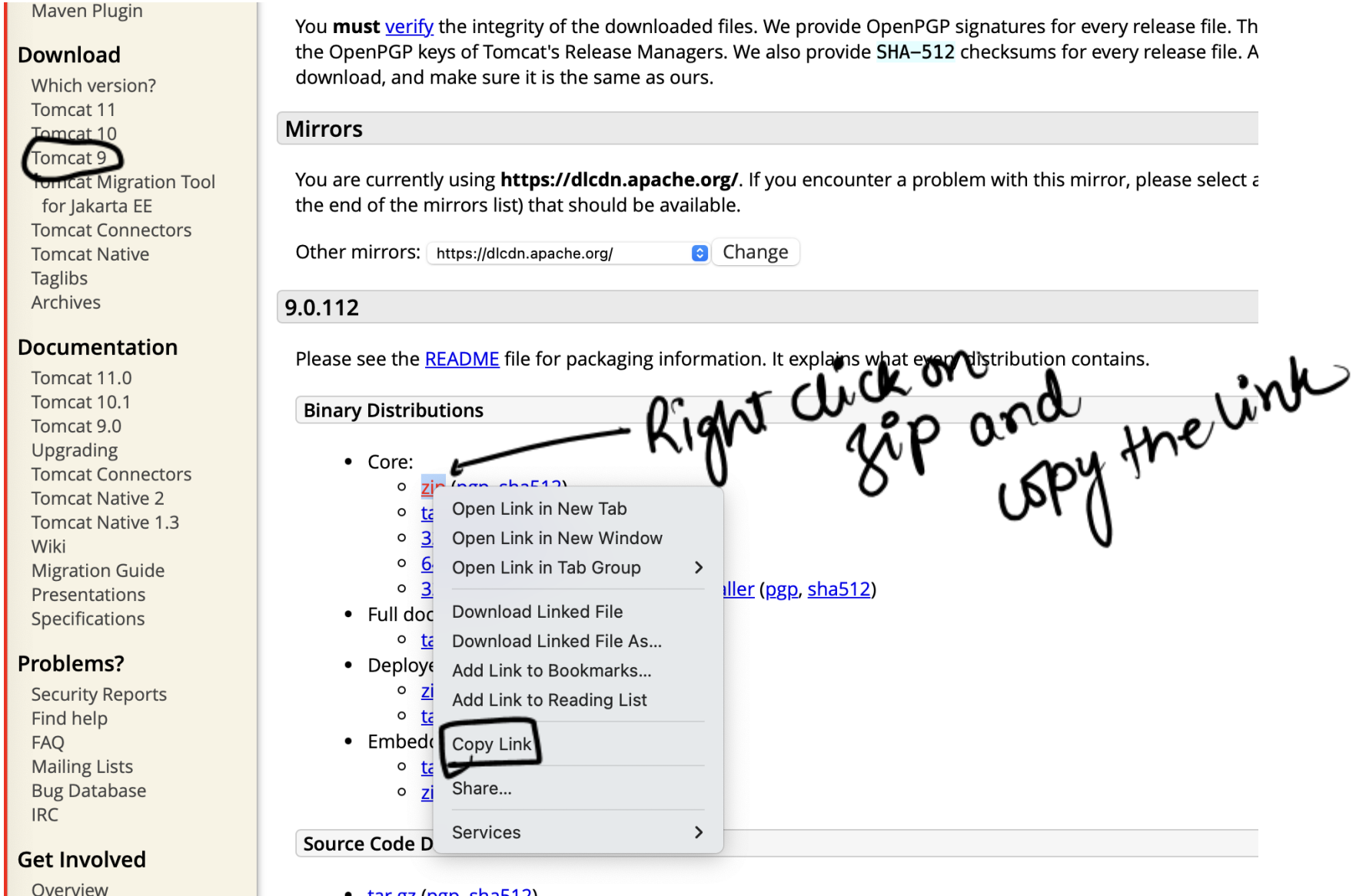
```
Downloaded from central: https://repo.maven.apache.org/maven2/org/sonatype/sisu/sisu-guice/3.1.0/sisu-guice-3.1.0-no_aop.jar (357 kB at 8
kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/asm/asm/3.3.1/asm-3.3.1.jar (44 kB at 105 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/google/guava/guava/10.0.1/guava-10.0.1.jar (1.5 MB at 3.6 MB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/eclipse/sisu/org.eclipse.sisu.inject/0.0.0.M2a/org.eclipse.sisu.inject-
0.0.0.M2a.jar (202 kB at 464 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/javax/annotation/jsr250-api/1.0/jsr250-api-1.0.jar
Downloading from central: https://repo.maven.apache.org/maven2/javax/inject/javax.inject/1/javax.inject-1.jar
Progress (1): 2.5 kB
Progress (2): 2.5 kB | 4.1/5.8 kB
Progress (2): 2.5 kB | 5.8 kB

Downloaded from central: https://repo.maven.apache.org/maven2/javax/inject/javax.inject/1/javax.inject-1.jar (2.5 kB at 50 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/javax/annotation/jsr250-api/1.0/jsr250-api-1.0.jar (5.8 kB at 96 kB/s)
[INFO] Packaging webapp
[INFO] Assembling webapp [addressbook] in [/var/lib/jenkins/workspace/myfirstpipeline/target/addressbook]
[INFO] Processing war project
[INFO] Building war: /var/lib/jenkins/workspace/myfirstpipeline/target/addressbook.war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  13.095 s
[INFO] Finished at: 2025-11-19T09:56:02Z
[INFO] ------------------------------------------------------------------------
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
```

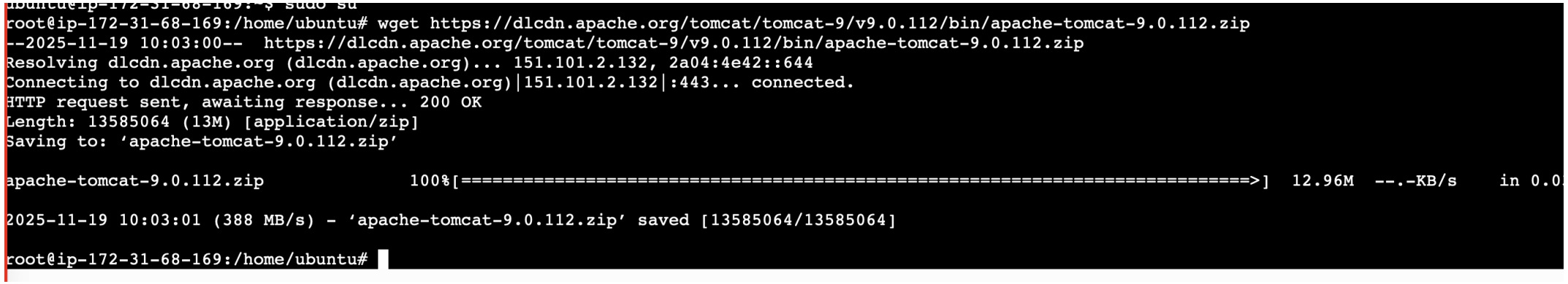
This is the location of the war file in the machine. We need to now deploy this war file in tomcat

- Lets now install the tomcat

Search install tomcat on google -> click on tomcat9



- sudo su (if you are connecting to machine from start)




- apt install unzip –y
- ls
- unzip apache-tomcat-9.0.112.zip

**Now we will start the tomcat ….but the problem tomcat works on port 8080 ….and jenkins also works on port 8080…how can two s/w work on same ports?**

We need to change the tomcat port to other port (like e.g 1234, or 9090, or any other port number which is unused less then 65000)

Get inside the tomcat directory ....there you will see conf directory

Get inside the conf directory and open server.xml file

```
root@ip-172-31-68-169:/home/ubuntu# cd apache-tomcat-9.0.112
root@ip-172-31-68-169:/home/ubuntu/apache-tomcat-9.0.112# ls
BUILDING.txt  CONTRIBUTING.md  LICENSE  NOTICE  README.md  RELEASE-NOTES  RUNNING.txt  bin  conf  lib  logs  temp  weba
root@ip-172-31-68-169:/home/ubuntu/apache-tomcat-9.0.112# cd conf
root@ip-172-31-68-169:/home/ubuntu/apache-tomcat-9.0.112/conf# ls
catalina.policy      context.xml           jaspic-providers.xsd  server.xml        tomcat-users.xsd
catalina.properties  jaspic-providers.xml  logging.properties    tomcat-users.xml  web.xml
root@ip-172-31-68-169:/home/ubuntu/apache-tomcat-9.0.112/conf# vi server.xml
```

On line 71 you will see 8080 default port change it to 9090

```
        Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
    -->
    <Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443"
               maxParameterCount="1000"
               />
    <!-- A "Connector" using the shared thread pool-->
    <!--
    <Connector executor="tomcatThreadPool"
```

Change

```
        Java AJP  Connector: /docs/config/ajp.html
        APR (HTTP/AJP) Connector: /docs/apr.html
        Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
    -->
    <Connector port="9090" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443"
               maxParameterCount="1000"
               />
    <!-- A "Connector" using the shared thread pool-->
    <!--
    <Connector executor="tomcatThreadPool"
               port="8080" protocol="HTTP/1.1"
```

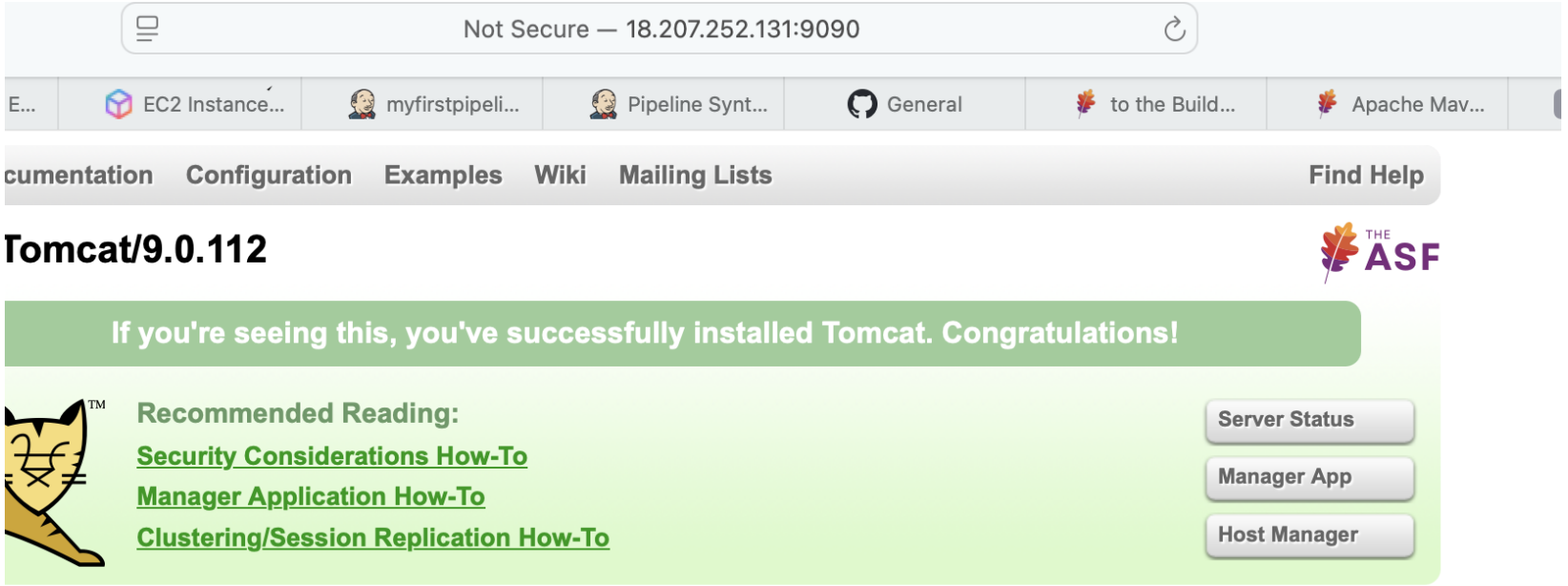
- Now we will start tomcat

For that we need to go bin directory and give rwx permission to all the binary files ending with sh

```
root@ip-172-31-68-169:/home/ubuntu/apache-tomcat-9.0.112/conf# cd ..
root@ip-172-31-68-169:/home/ubuntu/apache-tomcat-9.0.112# ls
BUILDING.txt  CONTRIBUTING.md  LICENSE  NOTICE  README.md  RELEASE-NOTES  RUNNING.txt  bin  conf  lib  logs  temp  webapps  work
root@ip-172-31-68-169:/home/ubuntu/apache-tomcat-9.0.112# cd bin
root@ip-172-31-68-169:/home/ubuntu/apache-tomcat-9.0.112/bin# chmod 700 *.sh
root@ip-172-31-68-169:/home/ubuntu/apache-tomcat-9.0.112/bin# ls
bootstrap.jar       ciphers.bat                configtest.bat  digest.sh         setclasspath.sh  startup.sh           tool-wrapper.sh
catalina-tasks.xml  ciphers.sh                 configtest.sh   makebase.bat      shutdown.bat     tomcat-juli.jar      version.bat
catalina.bat        commons-daemon-native.tar.gz  daemon.sh       makebase.sh       shutdown.sh      tomcat-native.tar.gz  version.sh
catalina.sh         commons-daemon.jar         digest.bat      setclasspath.bat  startup.bat      tool-wrapper.bat
root@ip-172-31-68-169:/home/ubuntu/apache-tomcat-9.0.112/bin# ./startup.sh
Using CATALINA_BASE:   /home/ubuntu/apache-tomcat-9.0.112
Using CATALINA_HOME:   /home/ubuntu/apache-tomcat-9.0.112
Using CATALINA_TMPDIR: /home/ubuntu/apache-tomcat-9.0.112/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /home/ubuntu/apache-tomcat-9.0.112/bin/bootstrap.jar:/home/ubuntu/apache-tomcat-9.0.112/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
root@ip-172-31-68-169:/home/ubuntu/apache-tomcat-9.0.112/bin#
```

Now check if the inbound rule has 9090 port allowed ...

And open tomcat on publicip:9090




- Now we need to deploy the app on tomcat ....we need to copy the war file which we saw previously in the tomcat webapps directory

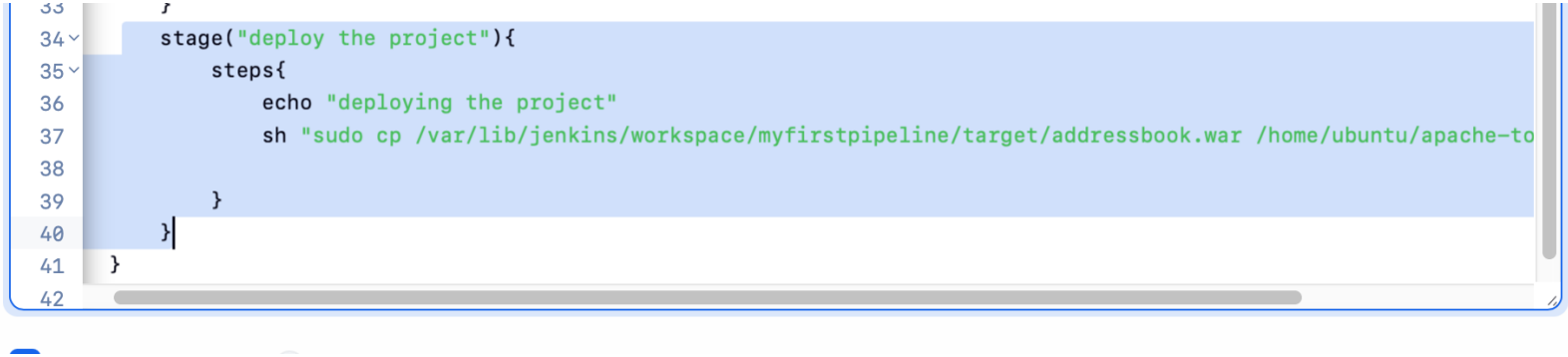
```
root@ip-172-31-68-169:/home/ubuntu/apache-tomcat-9.0.112/bin# cd ..
root@ip-172-31-68-169:/home/ubuntu/apache-tomcat-9.0.112# ls
BUILDING.txt  CONTRIBUTING.md  LICENSE  NOTICE  README.md  RELEASE-NOTES  RUNNING.txt  bin  conf  lib  logs  temp  webapps  work
root@ip-172-31-68-169:/home/ubuntu/apache-tomcat-9.0.112# cd webapps
root@ip-172-31-68-169:/home/ubuntu/apache-tomcat-9.0.112/webapps# pwd
/home/ubuntu/apache-tomcat-9.0.112/webapps
root@ip-172-31-68-169:/home/ubuntu/apache-tomcat-9.0.112/webapps#
```

location where we need to copy the addressbook .war file

- Now we know that our addressbook.war is created /var/lib/jenkins/workspace/myfirstpipeline/target/addressbook.war (you will have a different location ..you need to check logs)

And we need to copy It in webapps directory of tomcat

Go to your pipeline -> configure -> add this script

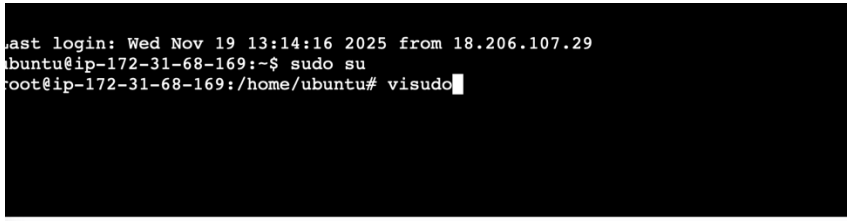


- Reference: https://github.com/akshu20791/addressbook-cicd-project/blob/master/jenkinsfile12

But when you run the script you will get an error that jenkins user does not have sudo permission

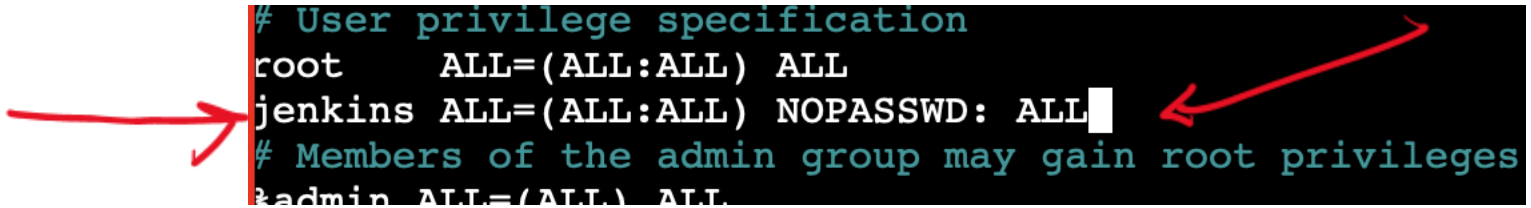
Go to the machine,

> visudo




Scroll down

Add the content below root




Press ctrl x -> y -> enter to come out of the file

service jenkins restart

- Build the job now

You should be able to build the job now

After the build is success

Example: http://18.207.252.131:9090/addressbook/

Here http://18.207.252.131 is public ip of the machine in which jenkins is running

9090 is the tomcat port