Script — code in Groovy script

Scripted pipeline

→ Groovy based syntax

→ Imperative (you tell Jenkins exactly what to do, step by step)

→ flexible → you can write complex logic with groovy codes, loops, conditions etc

Declarative pipeline

Build (compile)

|

Test

|

QA

|

Deploy

e.g Scripted pipeline

```
node {
    stage ("Build") {
        echo "Building"
    }
    stage ("Test") {
        echo "Testing"
    }
    stage ("Deploy") {
        echo "Deploy"
    }
}
```

Pros

1. More control & flexibility
2. Can use full Groovy features (variables, loops, conditions)
3. Good for complex pipeline

Cons

1. Harder to read & maintain
2. Higher learning curve for beginners.

>> Declarative pipeline ÷ uses Domain specific language (DSL) on top of Groovy.

- Declarative – you define what you want rest Jenkins will take care.
- Simpler then ~~Dec~~ Scripted pipeline

Open Jenkins on EC2 Instance

+ New item → pipeline

Lab

Jenkins / project-pipeline / Configuration

Configure

- General
- Triggers
- Pipeline
- Advanced

Definition

Pipeline script

Script ?

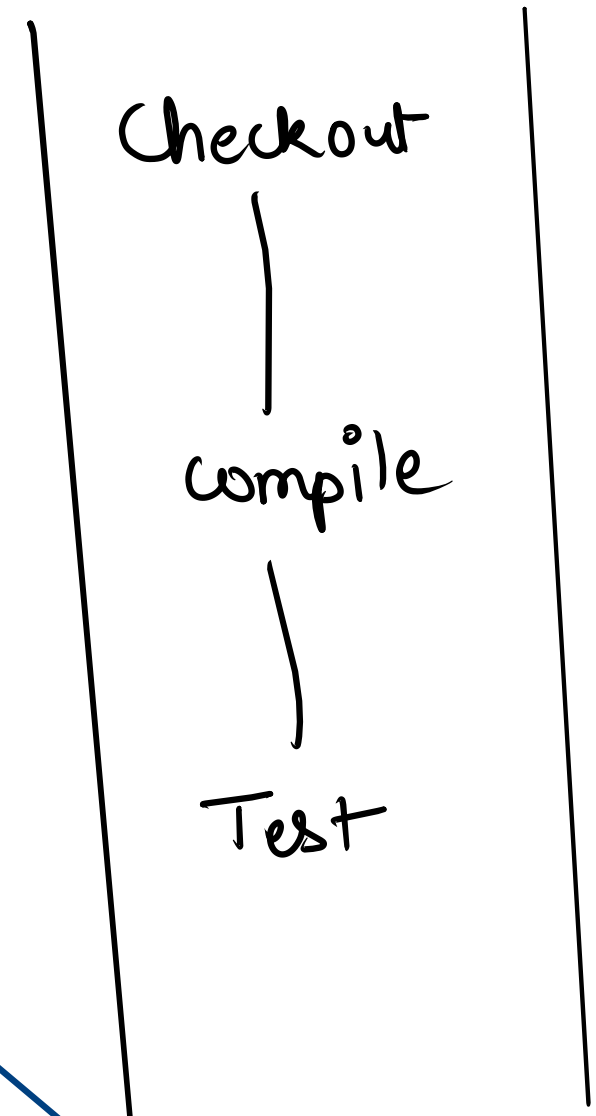
```
pipeline{
    agent any
    stages{
        stage("first stage: here we checkout the code from github"){
            steps{
                git 'https://github.com/akshu20791/addressbook-cicd-project
                echo "cloning the code in jenkins workspace"
            }
        }
        stage("compile the code by akshat"){
            steps{
                sh 'mvn compile'
                echo "compiling the project"
            }
        }
```

Use Groovy Sandbox ?

```
pipeline{
    agent any
    stages{
        stage("first stage: here we checkout the code from github"){
            steps{
                git 'https://github.com/akshu20791/addressbook-cicd-project'
                echo "cloning the code in jenkins workspace"
            }
        }
        stage("compile the code by akshat"){
            steps{
                sh 'mvn compile'
                echo "compiling the project"
            }
        }
        stage("runnng the test case"){
            steps{
                sh 'mvn test'
            }
        }

    }
}
```

Save



use your forked
repo link not mine

Jenkins / project-pipeline

- Status
- Changes
- Build Now
- Configure
- Delete Pipeline
- Stages
- Rename
- Pipeline Syntax

project-pipeline

Permalinks

- Last build (#1), 1 min 59 sec ago

Execute the pipeline

Builds

Filter /

Today

#1 6:20 AM

After this is build click on #1

Click on pipeline Overview →

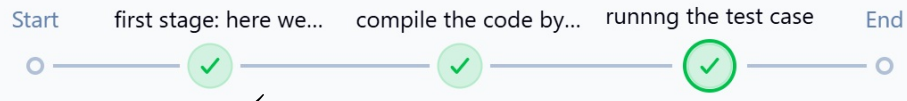
Jenkins / project-pipeline / #1 / Pipeline Overview

#1

Rerun

Manually run by admin | Started 2 min 51 sec ago | Queued 41 ms | Took 2 min 6 sec

Graph



Search

first stage: here we checkout the coc

compile the code by akshat 1m 45s

runnng the test case 13s

runnng the test case 13s Started 3m 7s ago Jen

mvn test

```
0 + mvn test
1 [INFO] Scanning for projects...
2 [WARNING]
```

you can see the logs if error is there that also you can see.

we will now deploy the project on Tomcat.

Go to machine

```
wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.108/bin/apache-tomcat-9.0.108.zip

ls

apt install unzip -y

unzip apache-tomcat-9.0.108.zip

cd apache-tomcat-9.0.108
```

now my jenkins and tomcat both works on port 8080 it is not possible to deploy different app on same port . so we will change the default port of tomcat to 9090

cd conf

vi server.xml

press i and come down

Replace it with 9090

```
<!-- A "Connector" represents an endpoint by which requests are received
     and responses are returned. Documentation at :
     Java HTTP Connector: /docs/config/http.html
     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="8080" 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"
           connectionTimeout="20000"
           redirectPort="8443"
           maxParameterCount="1000"
           />
-->
<!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443
     This connector uses the NIO implementation. The default
     SSLImplementation will depend on the presence of the APR/native
-- INSERT --                                                    73,1
```

```
<!-- A "Connector" represents an endpoint by which requests are received
     and responses are returned. Documentation at :
     Java HTTP Connector: /docs/config/http.html
     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"
           connectionTimeout="20000"
           redirectPort="8443"
           maxParameterCount="1000"
           />
```

cd ..

cd bin

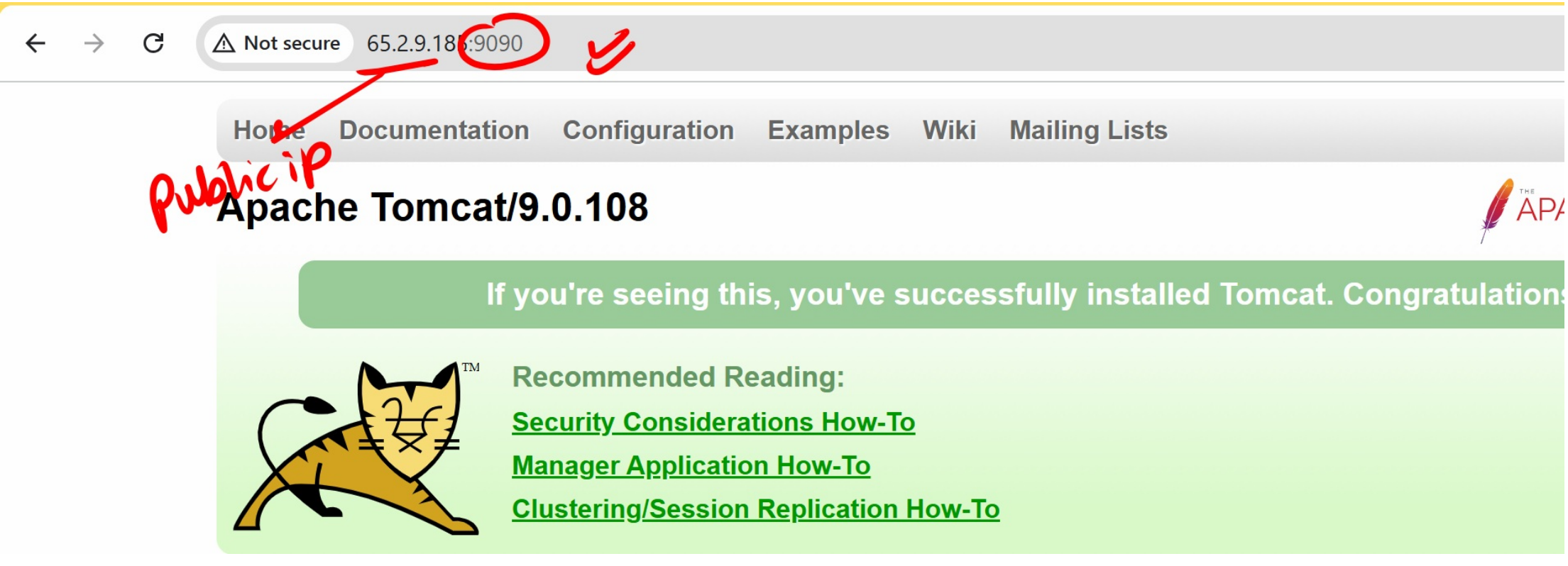
chmod +x *.sh

ls

./startup.sh

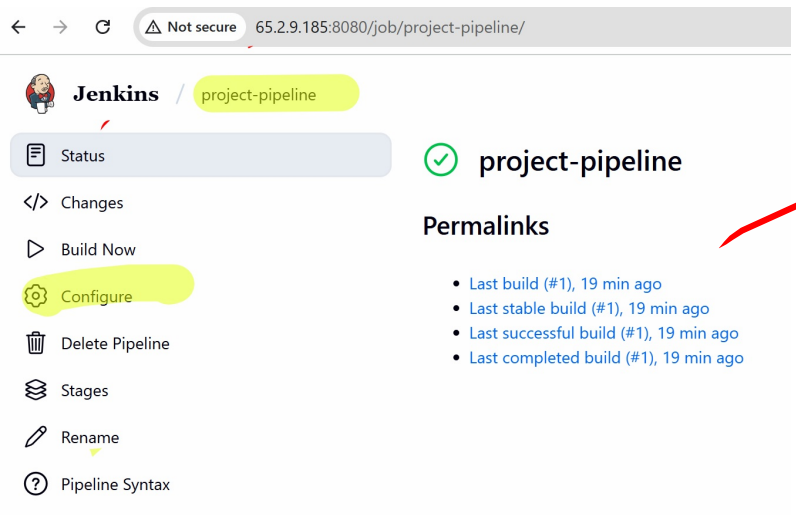
(you will see that tomcat is started)

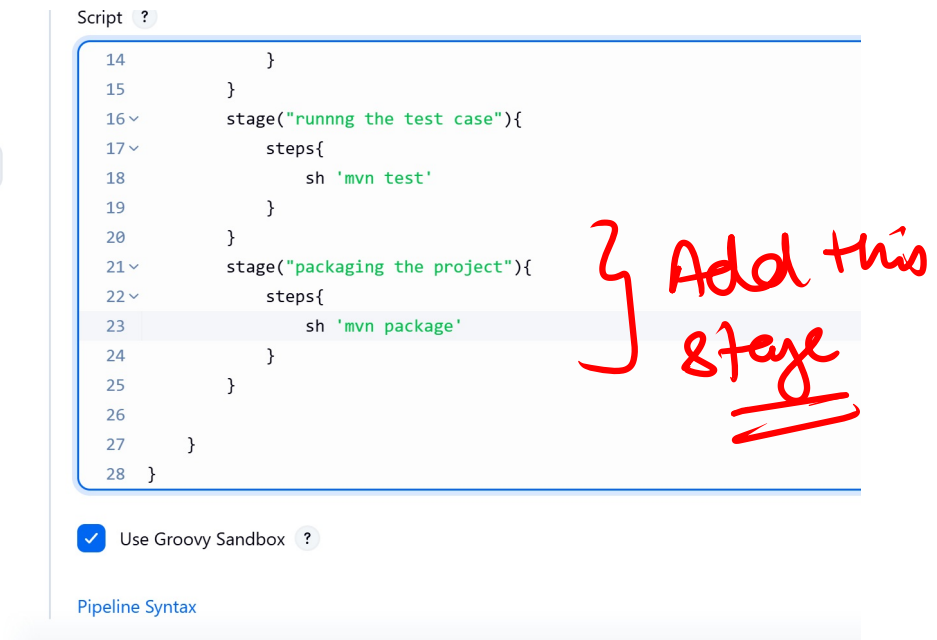
```
root@ip-172-31-43-137:/home/ubuntu/apache-tomcat-9.0.108/bin# ./startup.sh
Using CATALINA_BASE:   /home/ubuntu/apache-tomcat-9.0.108
Using CATALINA_HOME:   /home/ubuntu/apache-tomcat-9.0.108
Using CATALINA_TMPDIR: /home/ubuntu/apache-tomcat-9.0.108/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /home/ubuntu/apache-tomcat-9.0.108/bin/bootstrap.jar:/home/ubuntu/apache-tomca
Using CATALINA_OPTS:
Tomcat started.
root@ip-172-31-43-137:/home/ubuntu/apache-tomcat-9.0.108/bin#
```



we need to deploy the application in tomcat.

now we will package the project:





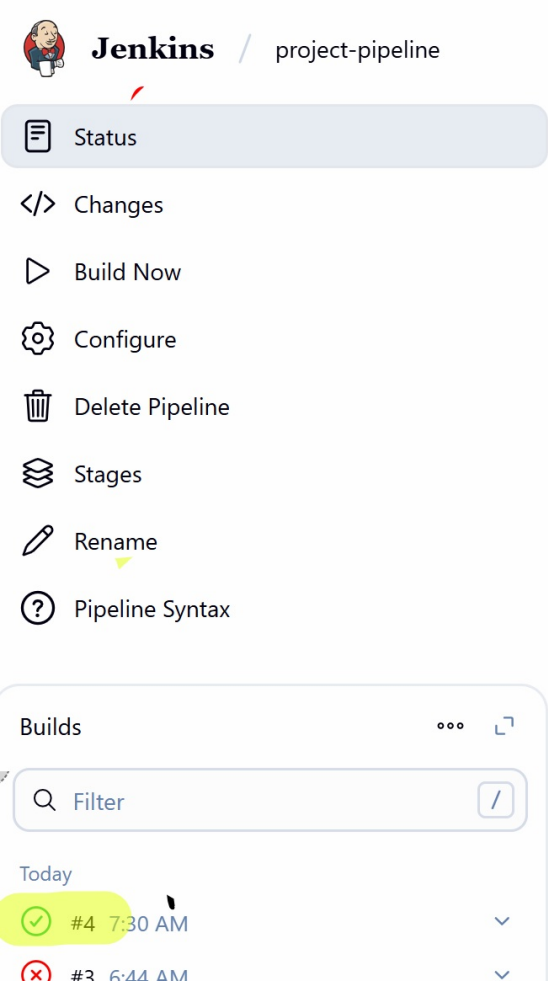
use your own forked repo

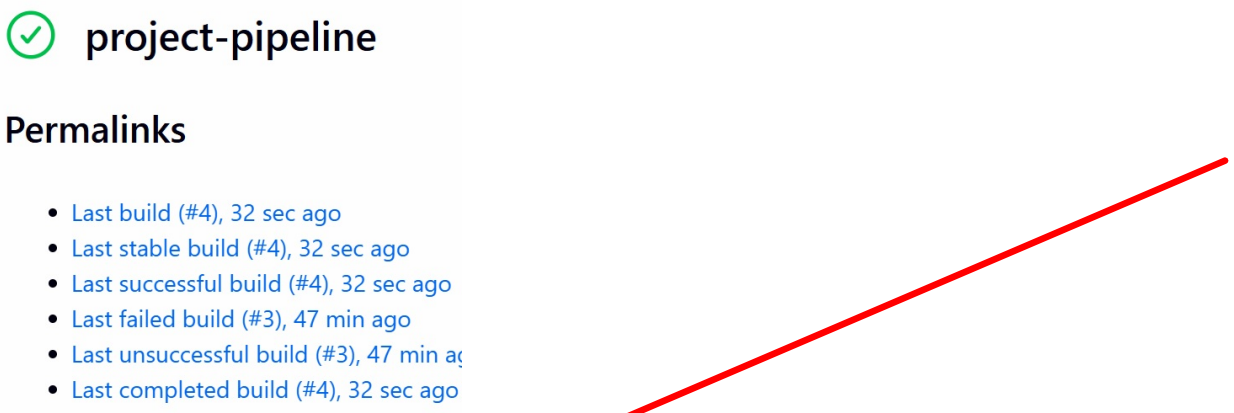
```
pipeline{
    agent any
    stages{
        stage("first stage: here we checkout the code from github"){
            steps{
                git 'https://github.com/akshu20791/addressbook-cicd-project'
                echo "cloning the code in jenkins workspace"
            }
        }
        stage("compile the code by akshat"){
            steps{
                sh 'mvn compile'
                echo "compiling the project"
            }
        }
        stage("runnng the test case"){
            steps{
                sh 'mvn test'
            }
        }
        stage("packaging the project"){
            steps{
                sh 'mvn package'
            }
        }

    }
}
```

Save & build the Job.

After build the Job → click on the built

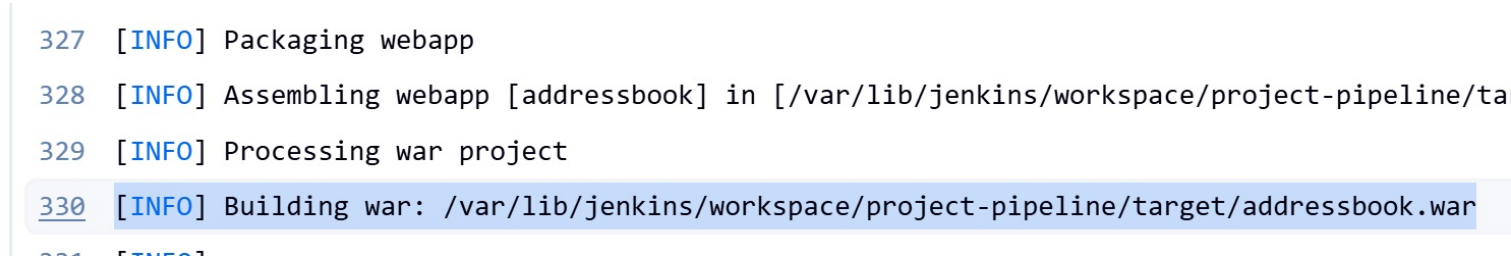




click on pipeline overview

click on package stage

scroll at bottom and you will see where the war file is created



We need to copy the war file in the tomcat - webapps directory

```
root@ip-172-31-43-137:/home/ubuntu# ls
apache-tomcat-9.0.108  apache-tomcat-9.0.108.zip  jenkins.sh
root@ip-172-31-43-137:/home/ubuntu# cd apache-tomcat-9.0.108
root@ip-172-31-43-137:/home/ubuntu/apache-tomcat-9.0.108# ls
BUILDING.txt  CONTRIBUTING.md  LICENSE  NOTICE  README.md  RELEASE-NOTES  RUNNING.txt  bin  conf  lib  logs  temp  webapps  work
root@ip-172-31-43-137:/home/ubuntu/apache-tomcat-9.0.108# cd webapps
root@ip-172-31-43-137:/home/ubuntu/apache-tomcat-9.0.108/webapps# pwd
/home/ubuntu/apache-tomcat-9.0.108/webapps
root@ip-172-31-43-137:/home/ubuntu/apache-tomcat-9.0.108/webapps#
```

war file is present at : /var/lib/jenkins/workspace/project-pipeline/target/addressbook.war

and it need to be pasted to : /home/ubuntu/apache-tomcat-9.0.108/webapps

so we need to add a stage to copy addressbook.war to webapp directory

go back to the pipeine job - configure\

```
pipeline{
    agent any
    stages{
        stage("first stage: here we checkout the code from github"){
            steps{
                git 'https://github.com/akshu20791/addressbook-cicd-project'
                echo "cloning the code in jenkins workspace"
            }
        }
        stage("compile the code by akshat"){
            steps{
                sh 'mvn compile'
                echo "compiling the project"
            }
        }
        stage("runnng the test case"){
            steps{
                sh 'mvn test'
            }
        }
        stage("packaging the project"){
            steps{
                sh 'mvn package'
            }
        }
        stage("depoy the project"){
            steps{
                sh 'sudo cp /var/lib/jenkins/workspace/project-pipeline/target/addressbook.war /home/ubuntu/apache-tomcat-9.0.108/webapps'
            }
        }
    }
}
```

use your forked repo
not mine :)

jenkins user will copy the addressbook.war to webapps directory

We need to give jenkins user the sudo permissins

visudo

```
root@ip-172-31-43-137:/home/ubuntu/apache-tomcat-9.0.108/webapps# visudo
```

i-04393463e4f167953 (jenkins-machine-akshat)

PublicIPs: 3.109.5.254 PrivateIPs: 172.31.43.137

```
# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
jenkins ALL=(ALL:ALL) NOPASSWD: ALL
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

Add this content below root

Press esc ctrl x -> y -> enter to come out of the nano editor

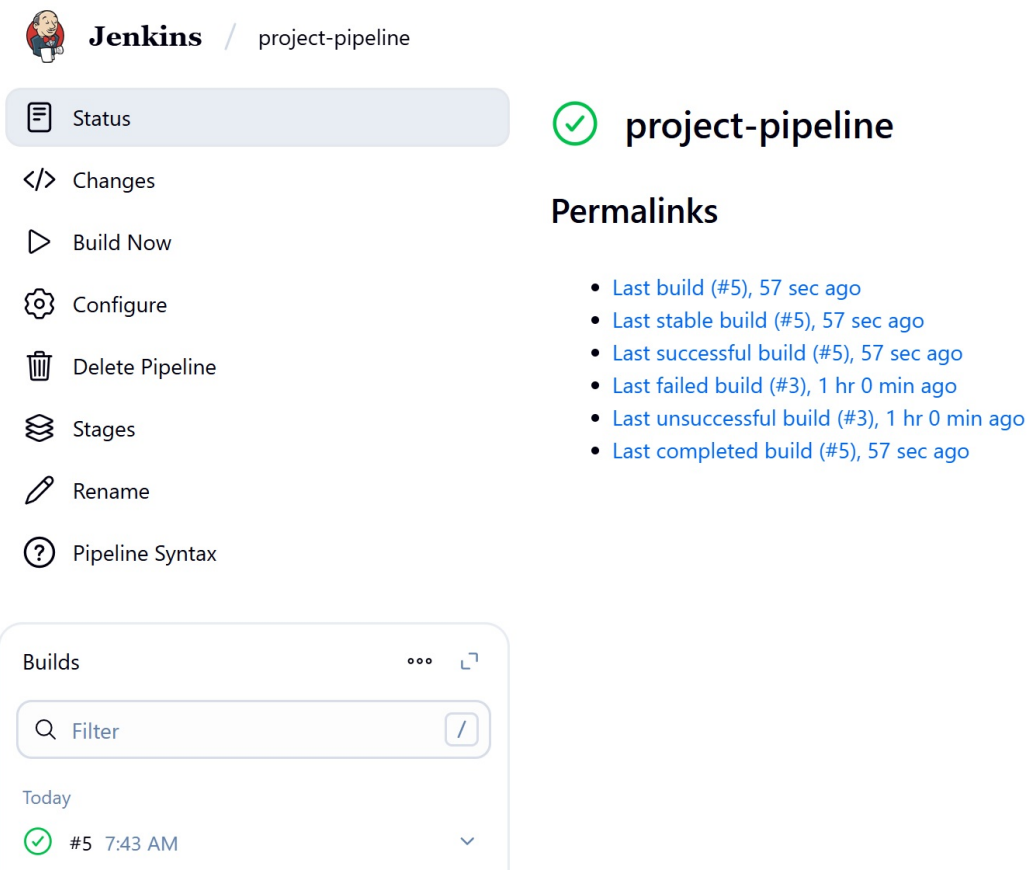
Restart jenkins

service jenkins restart

```
root@ip-172-31-43-137:/home/ubuntu/apache-tomcat-9.0.108/webapps# service jenkins restart
root@ip-172-31-43-137:/home/ubuntu/apache-tomcat-9.0.108/webapps#
```

i-04393463e4f167953 (jenkins-machine-akshat)

go to jenkins -> go to your pipeine job
and build now



Not secure 3.109.5.254:9090/addressbook/

Filter contacts...

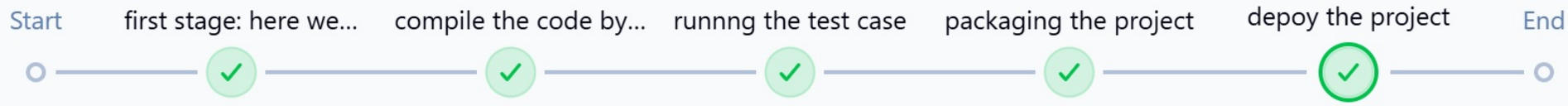
First Name	Last Name	Email
George	White	george@wh
Daniel	Thompson	daniel@thor
Timothy	Jones	timothy@jor
Peter	Wilson	peter@wilsc
Dan	Robinson	dan@robins
Dan	Davis	dan@davis.c
Olivia	Davis	olivia@davis
Dan	Smith	dan@smith.

If this page is not visible :
1) Check the security group if you have allowed traffic from 9090 port
2) check if tomcat is running
3) check pipeline logs if the war file is copied
4) check the webapps directory of tomcat if the addressbook.war is present

#5

Manually run by admin Started 3 min 54 sec ago Queued 28 ms Took 20 sec

Graph



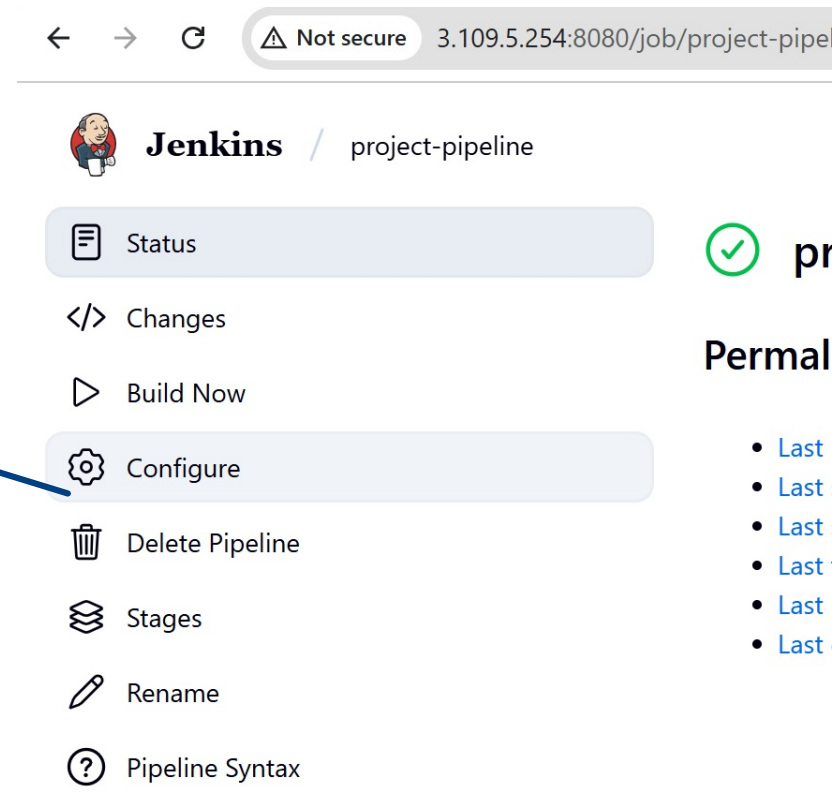
As of now you are building the job manually.....
but we need to put the condition that whenever developer push the code to github automatically job need to be trigger...

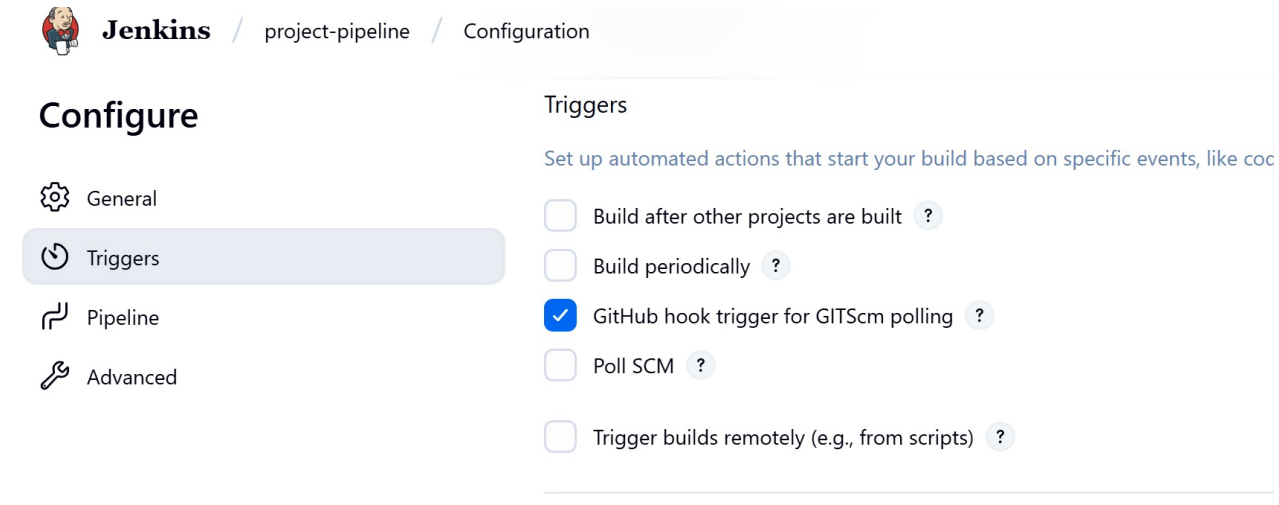
GITHUB WEBHOOKS

with github webhook as soon as developer push the code to github it will trigger the pipeline

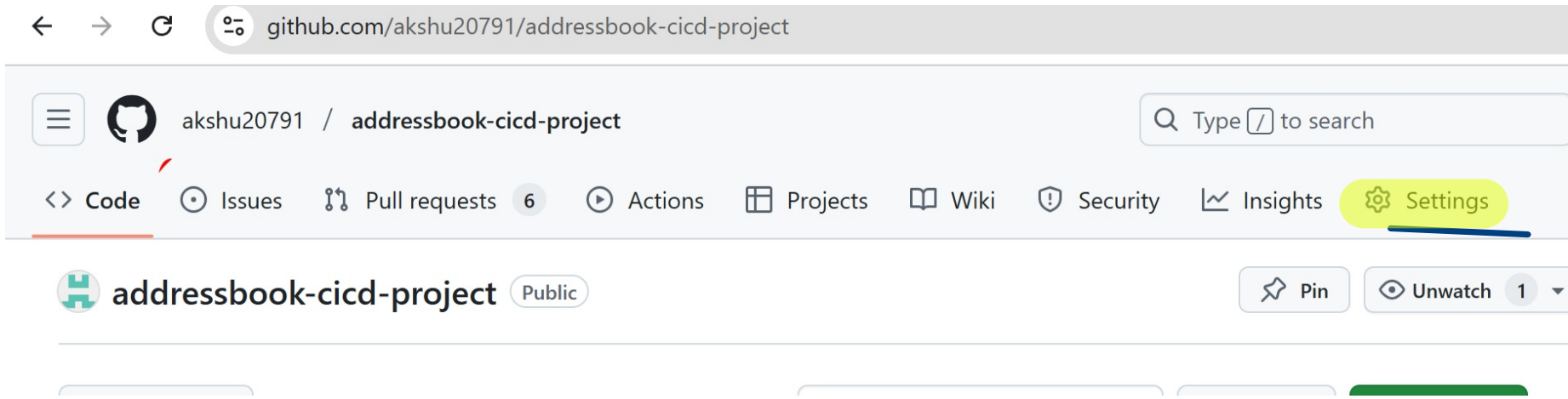
Go to job:
click on configure

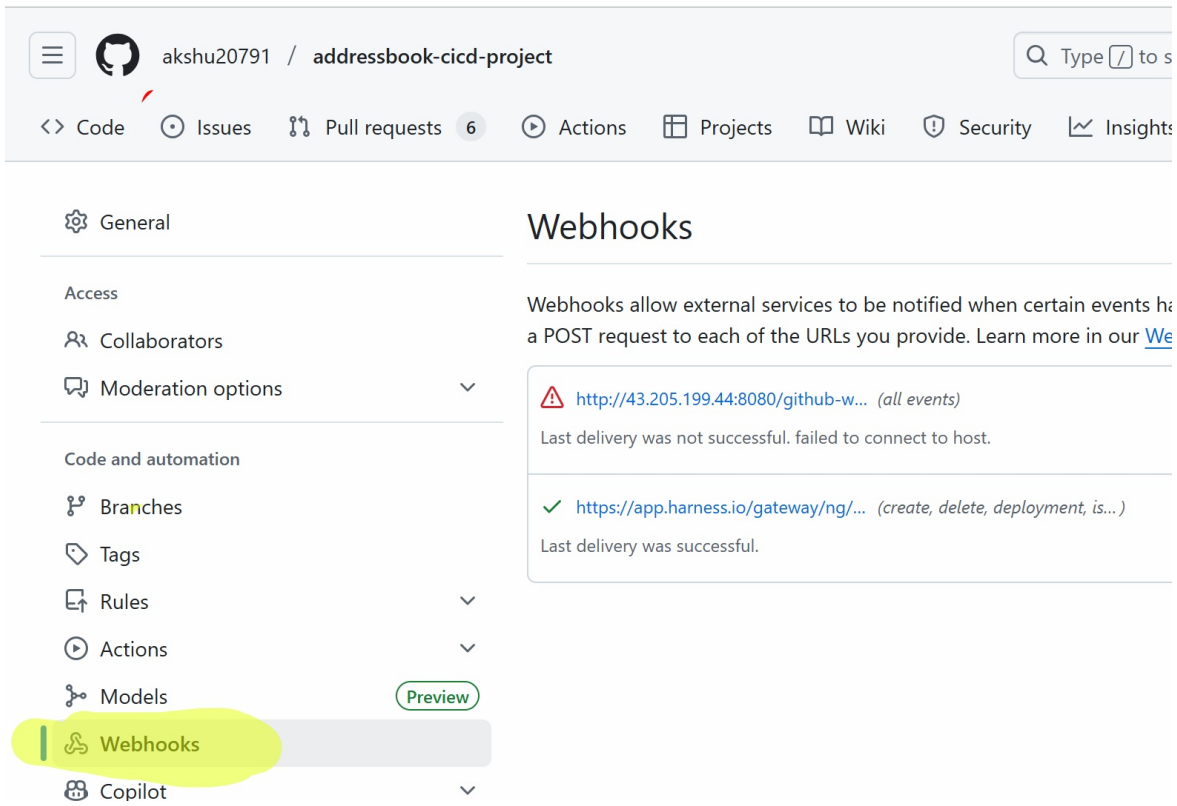
in trigger
click on github webhook



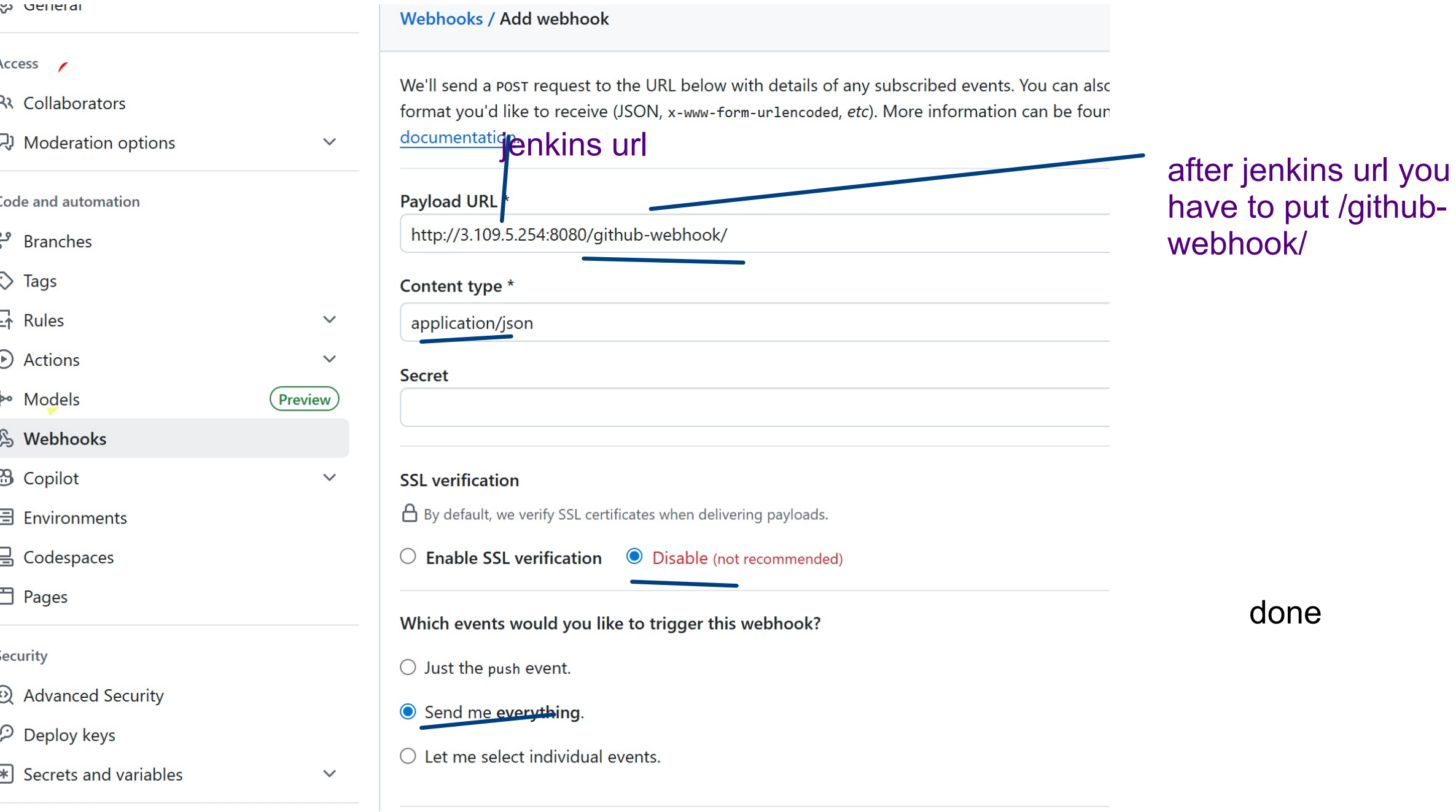


go to the github repo (your gihub repo) which you have forked





+ Add webhook

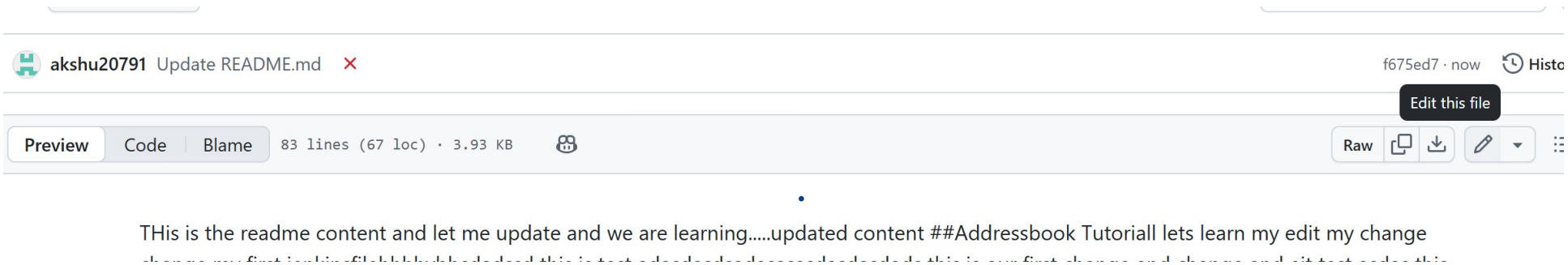


after jenkins url you have to put /github-webhook/

done

now go and update in readme file in github repo

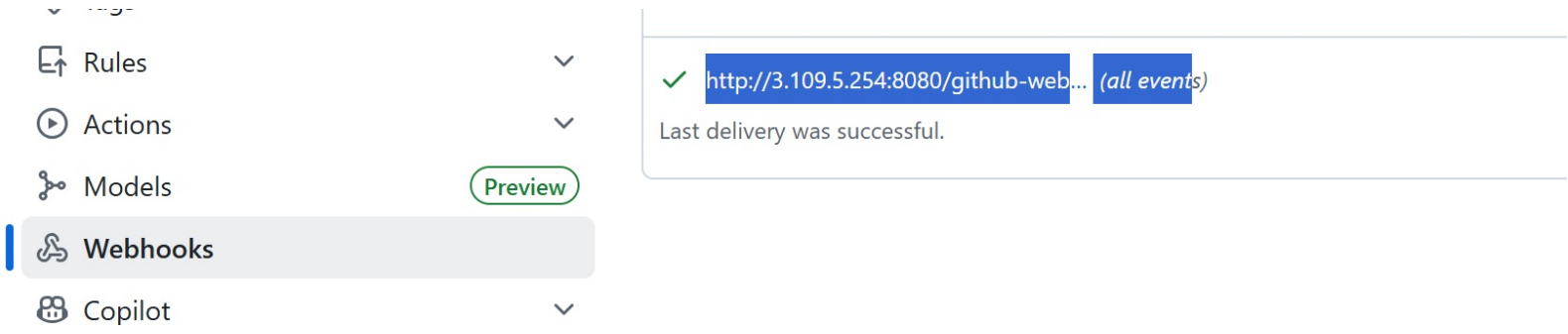
cick on readme.md file



put any content

commit

now check setting



Jenkins / project-pipeline

- Status
- Changes
- Build Now
- Configure
- Delete Pipeline
- Stages
- Rename
- Pipeline Syntax
- GitHub Hook Log

Builds

Filter /

Pending

#6

In the quiet period. Expires in 2.9 sec

project-pipeline

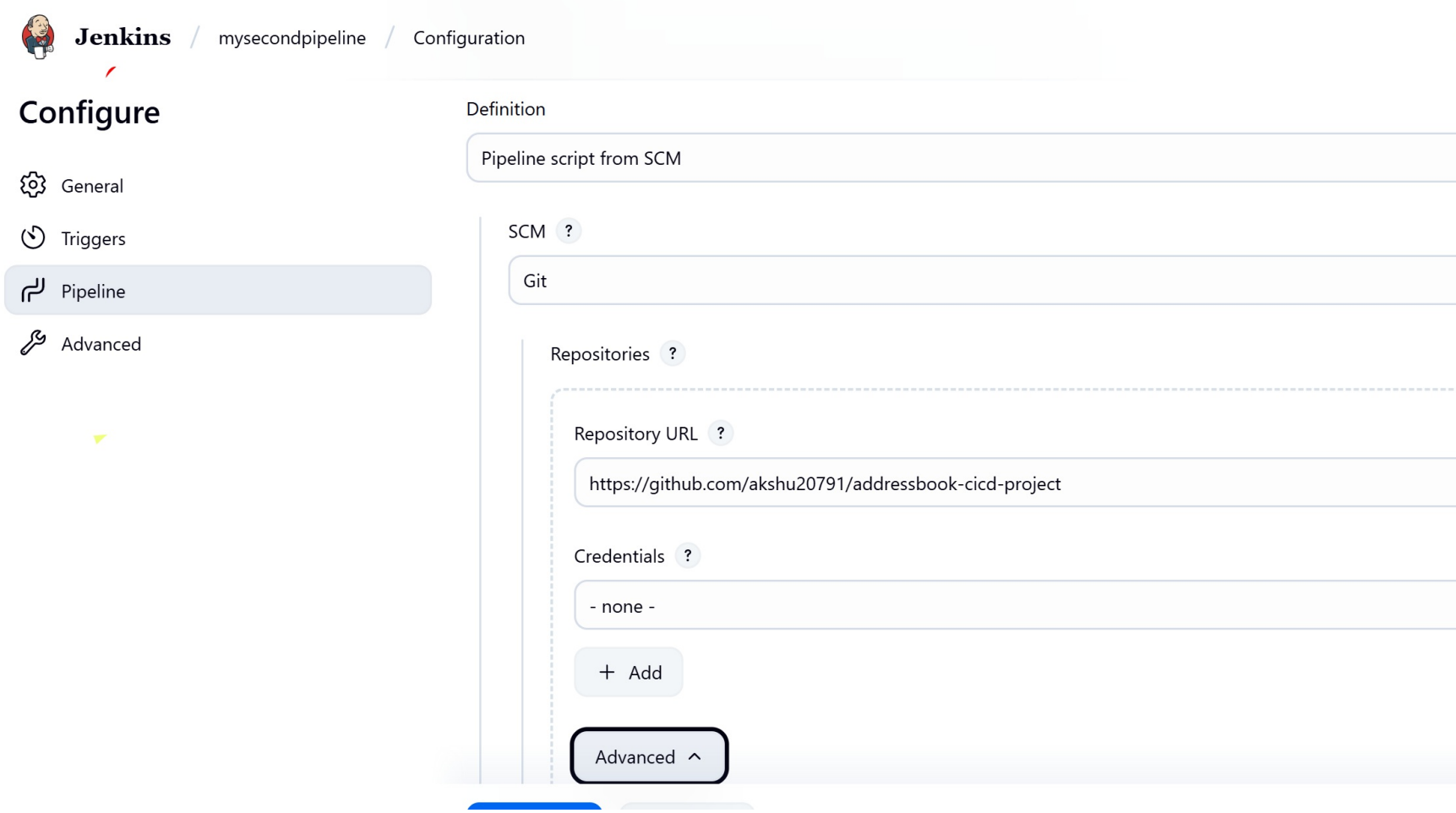
Permalinks

- Last build (#5), 18 min ago
- Last stable build (#5), 18 min ago
- Last successful build (#5), 18 min
- Last failed build (#3), 1 hr 17 min
- Last unsuccessful build (#3), 1 hr
- Last completed build (#5), 18 mir

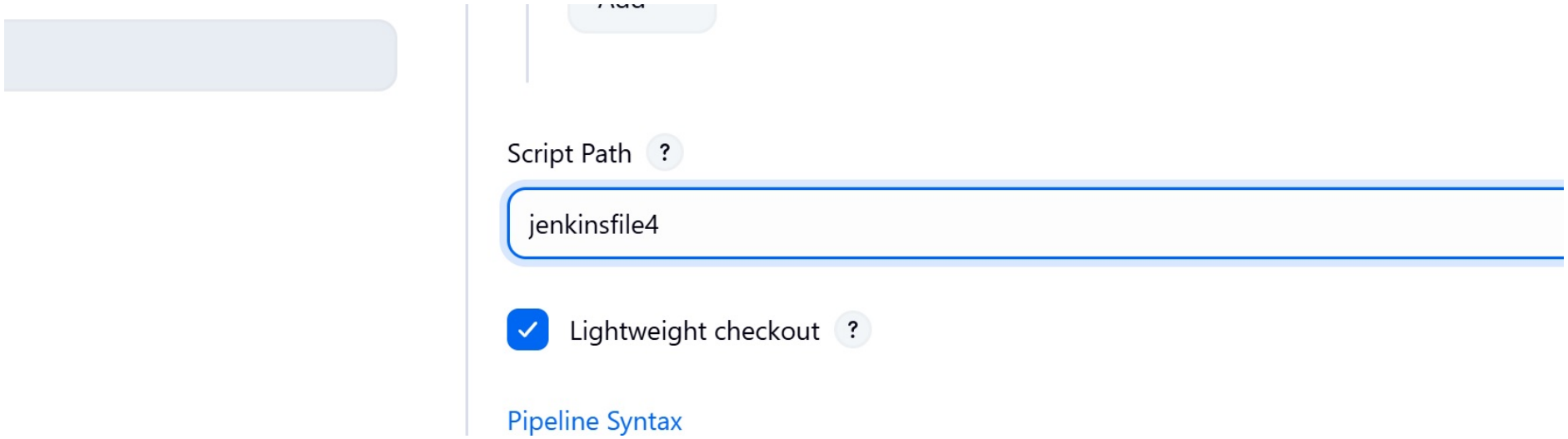
Also you can put your jenkins file in github and call it from github only

copy the existing jenkins file which you created and create a new file in github with name jenkinsfile and paste it in github

go to jenkins -> create a new job



click on script path and give your jenkinsfile name which you created in github



build the job