

e.g. Scripted pipeline

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

Pros

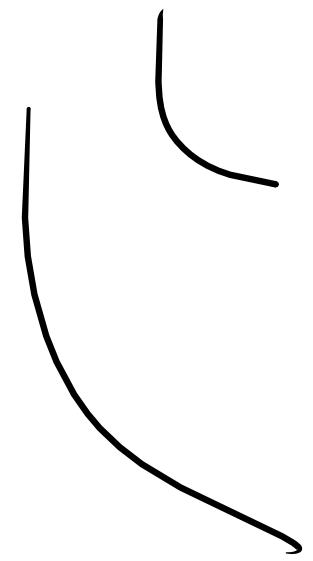
- ① More control & flexibility
- ② Can use full Groovy features
(variables, loops, conditions)
- ③ Good for complex pipeline

Cons

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

>> Declarative pipeline ÷ uses Domain specific language (DSL)

on top of Groovy.



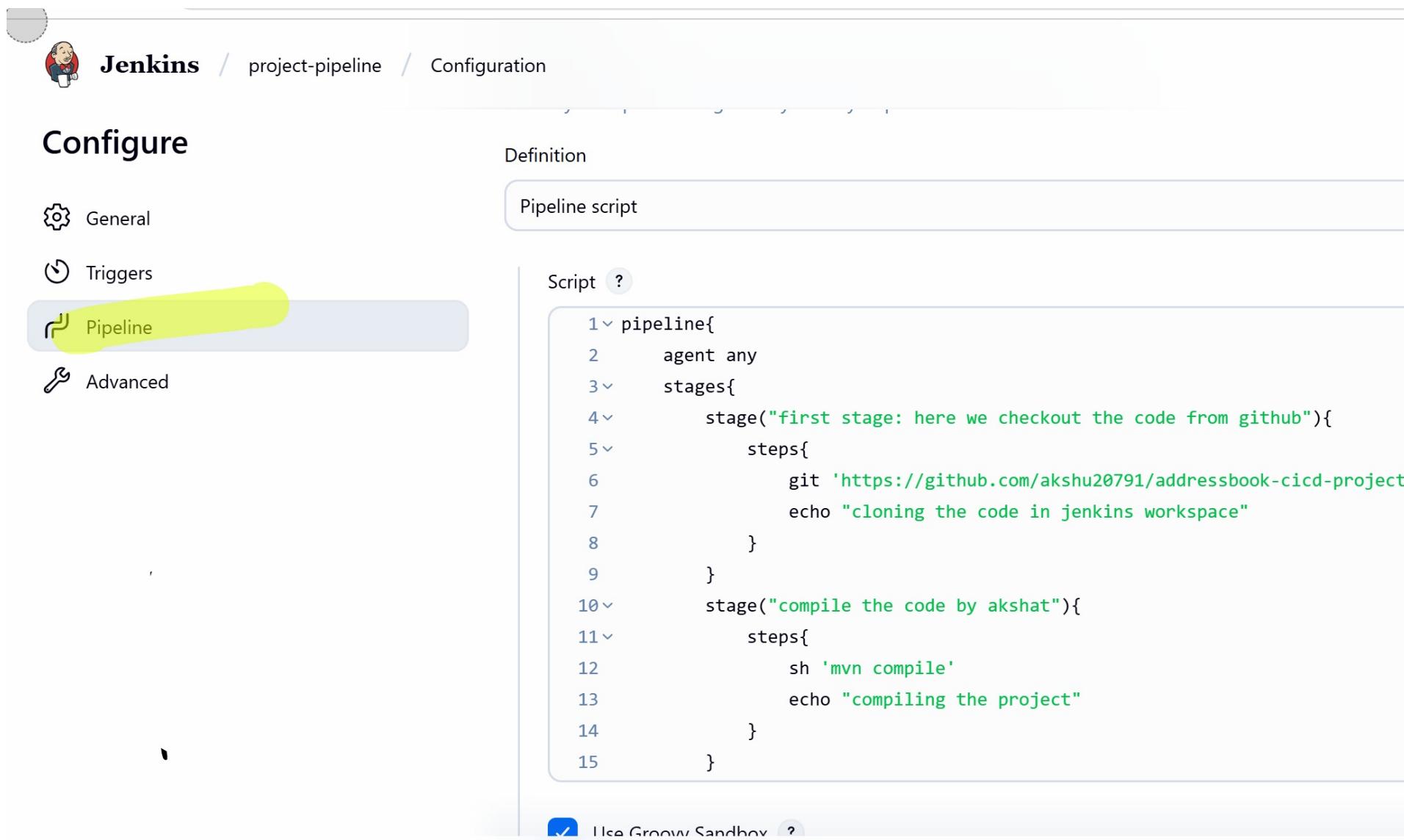
Declarative - you define what you want
rest Jenkins will take care.

Simpler than ~~Scripted~~ Scripted pipeline

Open Jenkins on EC2 Instance

Lab

+ New item → pipeline



The screenshot shows the Jenkins Pipeline configuration interface. The left sidebar has tabs: General, Triggers, Pipeline (which is selected and highlighted with a yellow bar), and Advanced. The main area is titled 'Definition' and 'Pipeline script'. It contains a code editor with the following Groovy script:

```
1 < pipeline{
2     agent any
3     stages{
4         stage("first stage: here we checkout the code from github"){
5             steps{
6                 git 'https://github.com/akshu20791/addressbook-cicd-project'
7                 echo "cloning the code in jenkins workspace"
8             }
9         }
10        stage("compile the code by akshat"){
11            steps{
12                sh 'mvn compile'
13                echo "compiling the project"
14            }
15        }
}
```

At the bottom of the code editor, there is a checkbox labeled '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("running the test case"){  
            steps{  
                sh 'mvn test'  
            }  
        }  
    }  
}
```

Save

Checkout

compile

Test

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

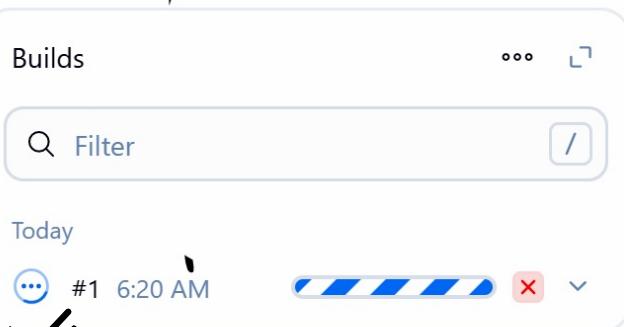
Execute the pipeline

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

```
graph LR; Start((Start)) --> S1["first stage: here we..."]; S1 --> S2["compile the code by..."]; S2 --> S3["running the test case"]; S3 --> End((End))
```

Search

- ✓ first stage: here we checkout the co
- ✓ compile the code by akshat 1m 45s
- ✓ running the test case 13s

✓ running the test case 13s Started 3m 7s ago Jen

✓ mvn test

- 0 + mvn test
- 1 [INFO] Scanning for projects...
- 2 [WARNING]
- 3 [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 gogo

```
<!-- 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#
```

Not secure 65.2.9.18:9090

Home Documentation Configuration Examples Wiki Mailing Lists

Apache Tomcat/9.0.108

The Apache

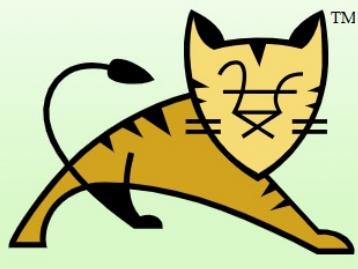
If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:

[Security Considerations How-To](#)

[Manager Application How-To](#)

[Clustering/Session Replication How-To](#)



TM

we need to deploy the application in tomcat .

now we will package the project:

The screenshot shows the Jenkins interface for a pipeline named "project-pipeline". The pipeline has four stages: "Last build (#1)", "Last stable build (#1)", "Last successful build (#1)", and "Last completed build (#1), 19 min ago". A red arrow points from the pipeline name to the pipeline script area.

```
14      }
15    }
16    stage("running the test case"){
17      steps{
18        sh 'mvn test'
19      }
20    }
21    stage("packaging the project"){
22      steps{
23        sh 'mvn package'
24      }
25    }
26  }
27}
28}
```

A red bracket on the right side of the script area is annotated with the text "Add this stage" in red ink.

```
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("running the test case"){
      steps{
        sh 'mvn test'
      }
    }
    stage("packaging the project"){
      steps{
        sh 'mvn package'
      }
    }
  }
}
```

use your own forked
repo

Save & build the Job.
After build the Job → Click on the built

The screenshot shows the Jenkins interface for the 'project-pipeline' job. On the left, there's a sidebar with options like Status, Changes, Build Now, Configure, Delete Pipeline, Stages, Rename, and Pipeline Syntax. Below that is a 'Builds' section with a 'Filter' input and a table showing two builds: '#4 7:30 AM' (green circle) and '#3 6:44 AM' (red circle). The main content area has a green checkmark icon and the text 'project-pipeline'. It includes a 'Permalinks' section with a list of recent builds and a 'Logs' section at the bottom showing four log entries:

- 327 [INFO] Packaging webapp
- 328 [INFO] Assembling webapp [addressbook] in [/var/lib/jenkins/workspace/project-pipeline/ta]
- 329 [INFO] Processing war project
- 330 [INFO] Building war: /var/lib/jenkins/workspace/project-pipeline/target/addressbook.war

A red arrow points from the text 'click on pipeline overview' to the 'project-pipeline' heading. Another red arrow points from the text 'click on package stage' to the 'Logs' section. A third red arrow points from the text 'scroll at bottom and you will see where the war file is created' to the last log entry.

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

```
ubuntu@ip-172-31-43-137:~$ sudo su
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 pipeline 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("running the test case"){  
            steps{  
                sh 'mvn test'  
            }  
        }  
        stage("packaging the project"){  
            steps{  
                sh 'mvn package'  
            }  
        }  
        stage("deploy 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 permisins

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)

D: 11/14/2023 10:53:51 P: 11/14/2023 17:23:17

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

The screenshot shows the Jenkins interface for the 'project-pipeline' job. On the left, there's a sidebar with various options: Status (which is selected), Changes, Build Now, Configure, Delete Pipeline, Stages, Rename, and Pipeline Syntax. The main content area displays the pipeline name 'project-pipeline' with a green checkmark icon. Below it, there's a section titled 'Permalinks' with a list of build links. Underneath, a 'Builds' section shows a table with one row, indicating a successful build #5 from 57 seconds ago. A 'Filter' input field is also present.

Build	Status	Duration
#5	Success	57 sec ago

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

**Jenkins**

/ project-pipeline

/ #5

/ Pipeline Overview



< #5

Manually run by admin

Started 3 min 54 sec ago

Queued 28 ms

Took 20 sec

Graph



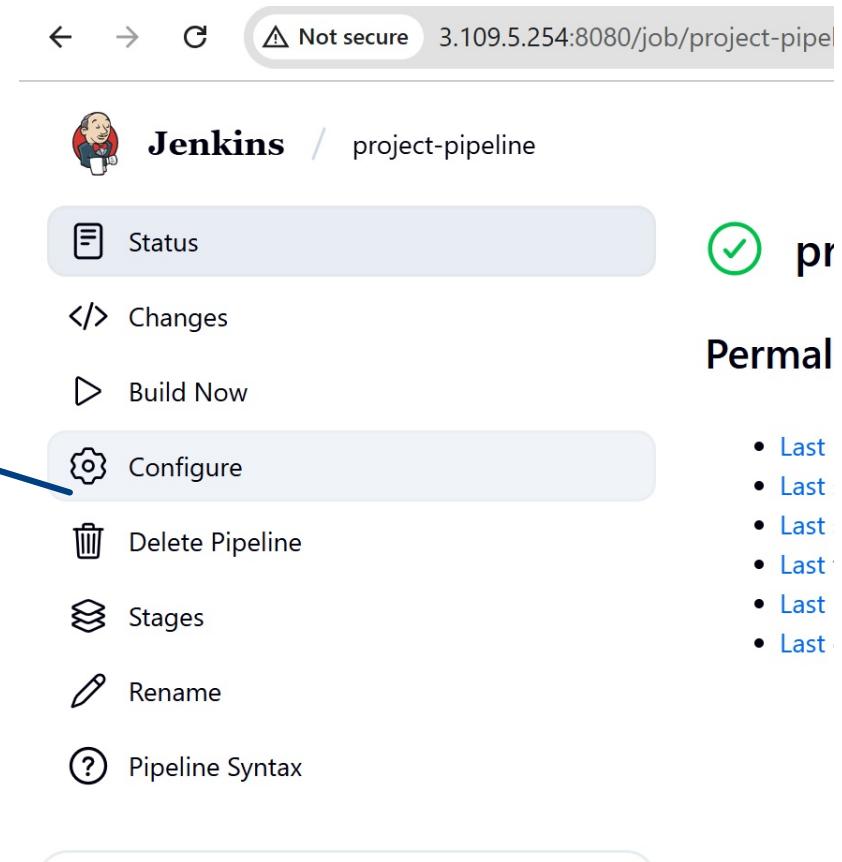
As of now you are building the job manually.....
but we need to put the condition that whenever developer push the code to github automatically job need to be trigger...

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



Configure

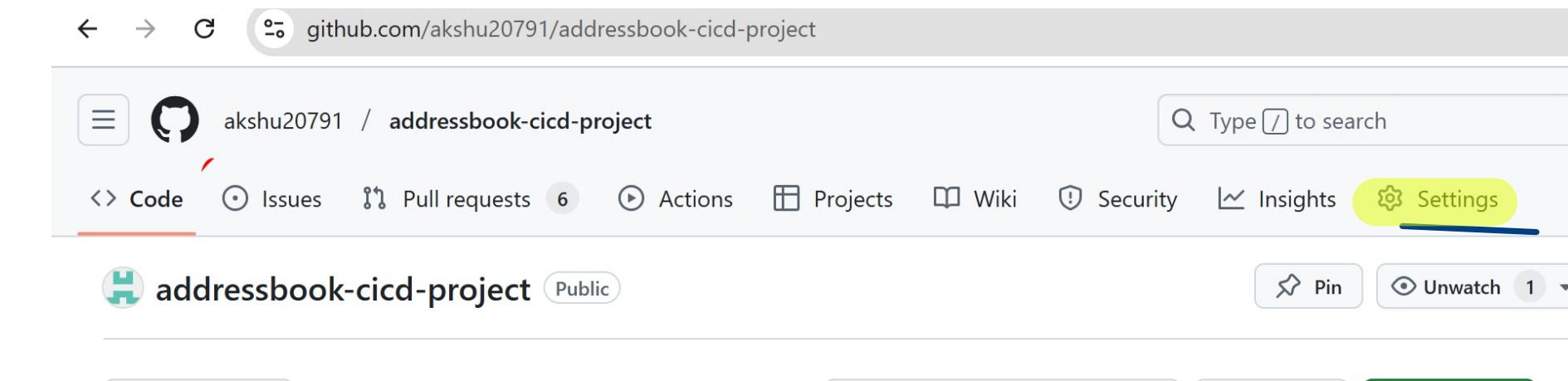
General

Triggers

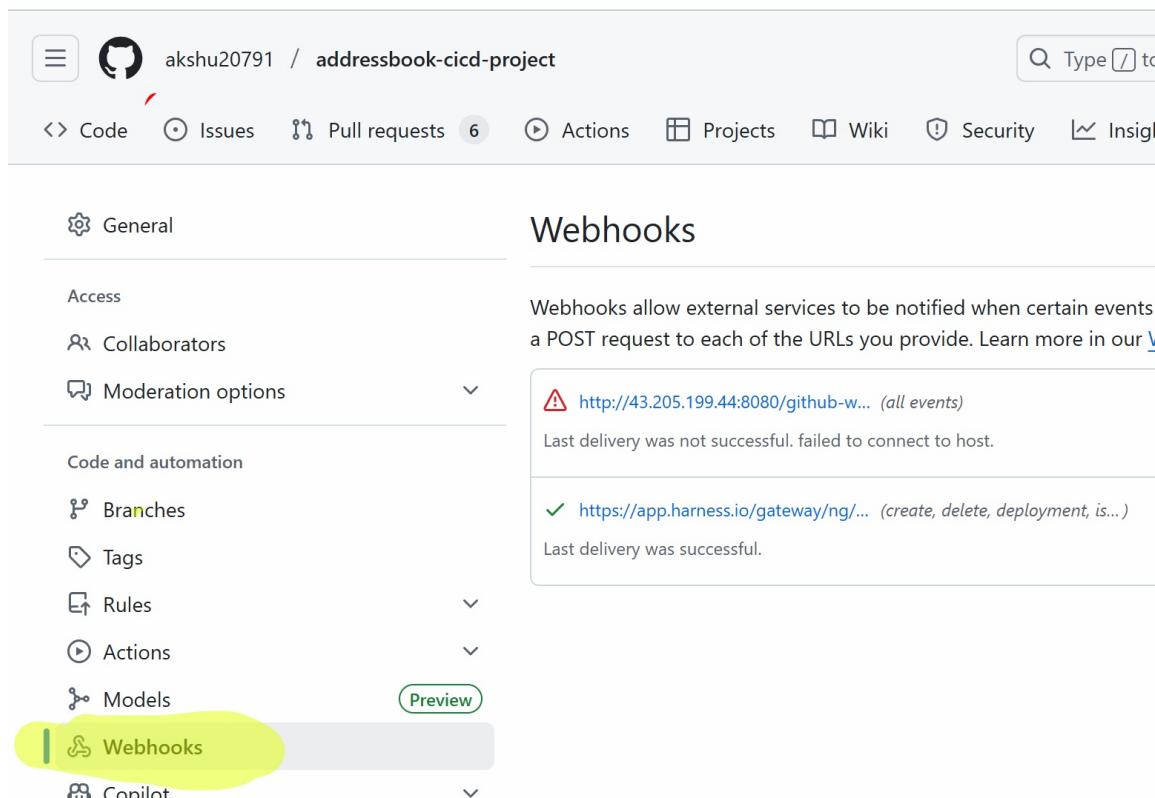
Pipeline

Advanced

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



The screenshot shows a GitHub repository page for 'akshu20791 / addressbook-cicd-project'. The 'Settings' tab is highlighted in yellow. The page includes a search bar, navigation links for Code, Issues, Pull requests (6), Actions, Projects, Wiki, Security, Insights, and Settings, and a 'Public' badge. Below the header, there's a title 'addressbook-cicd-project' and a 'Pin' button.



The screenshot shows the 'Settings' page for the repository. The left sidebar has sections like General, Access, Collaborators, and Moderation options. The main area is titled 'Webhooks' and contains a description: 'Webhooks allow external services to be notified when certain events happen. Send a POST request to each of the URLs you provide.' It lists two webhook entries: one with a red warning icon for 'http://43.205.199.44:8080/github-w...' (all events) which failed to connect, and one with a green checkmark for 'https://app.harness.io/gateway/ng/...' (create, delete, deployment, is...) which was successful. A green button labeled '+ Add webhook' is visible on the right.

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Models

Preview

Webhooks

Copilot

Environments

Codespaces

Pages

Security

Advanced Security

Deploy keys

Secrets and variables

Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found [documentation](#).

jenkins url

Payload URL *

http://3.109.5.254:8080/github-webhook/

Content type *

application/json

Secret

(empty)

SSL verification

By default, we verify SSL certificates when delivering payloads.

Enable SSL verification Disable (not recommended)

Which events would you like to trigger this webhook?

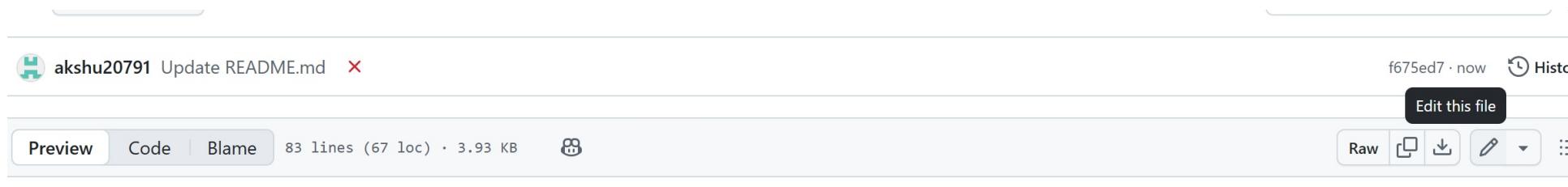
- Just the push event.
- Send me everything.
- Let me select individual events.

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

done

now go and update in readme file in github repo

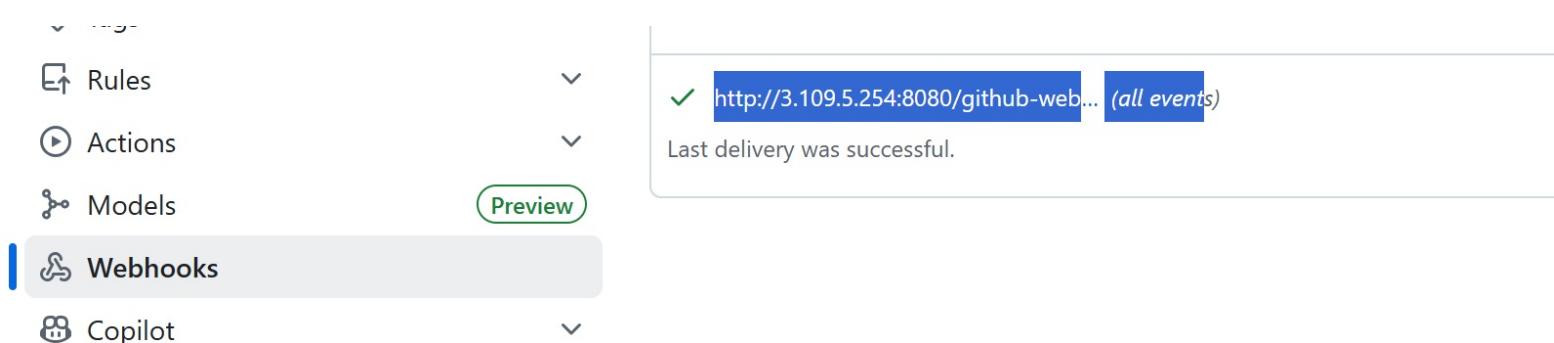
click on readme.md file



put any content

commit

now check setting





Jenkins / project-pipeline

Status

Changes

Build Now

Configure

Delete Pipeline

Stages

Rename

Pipeline Syntax

GitHub Hook Log



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 min](#)

Builds

...

Filter /

Pending

#6

In the quiet period. Expires in 2.9 sec

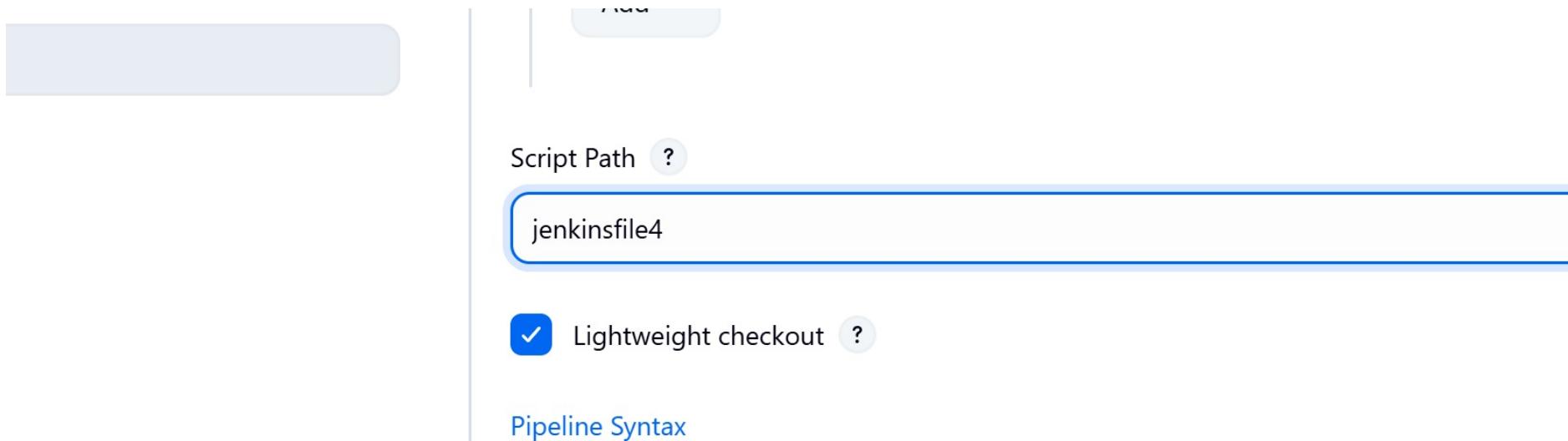
Also you can put your jenkins file in github and call it from github only

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

The screenshot shows the Jenkins interface for configuring a new pipeline job named "mysecondpipeline". The left sidebar has tabs for General, Triggers, Pipeline (which is selected), and Advanced. The main area is titled "Definition" and shows "Pipeline script from SCM". Under "SCM", "Git" is selected. In the "Repositories" section, the "Repository URL" is set to <https://github.com/akshu20791/addressbook-cicd-project>. The "Credentials" section shows "- none -" and a "+ Add" button. At the bottom of the configuration panel is an "Advanced" button.

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



build the job