CI/CD: Note-App Deployment using Declarative Pipeline

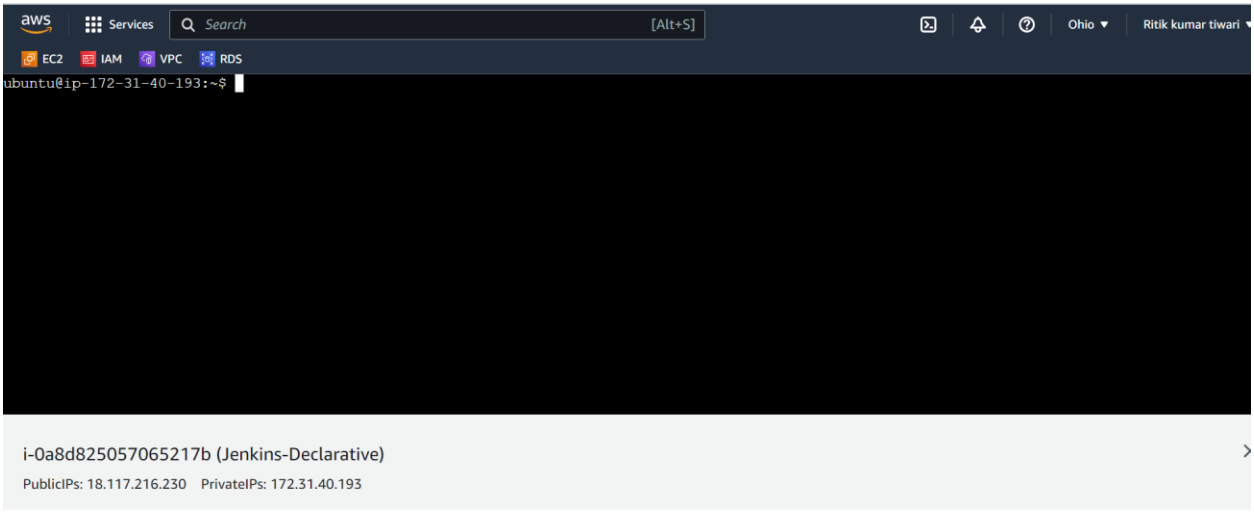
LINKEDIN: ***https://www.linkedin.com/in/ritikktiwari/***

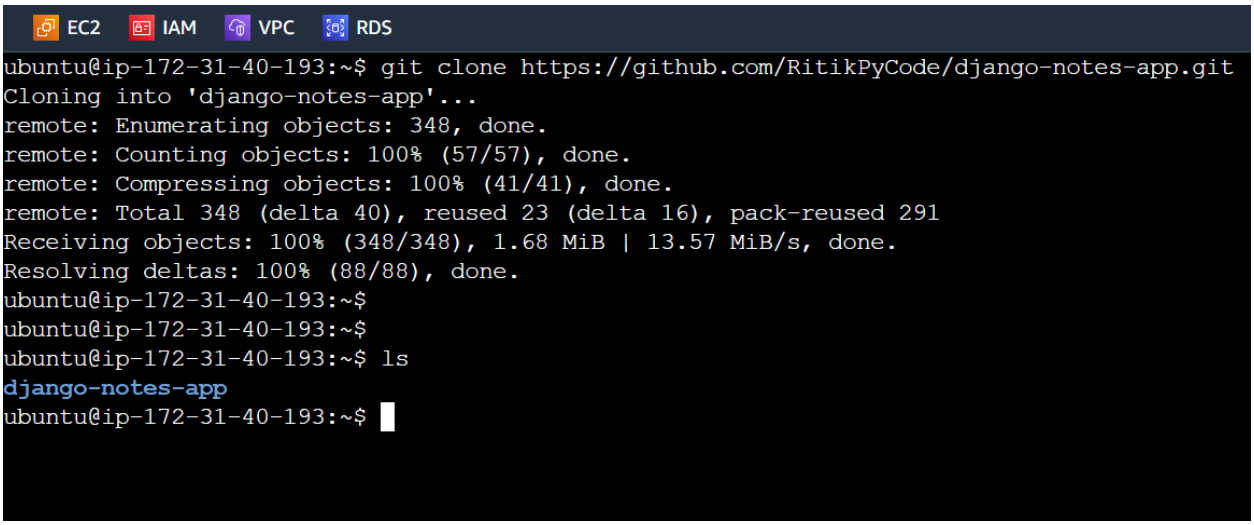
by: Ritik Kumar Tiwari

AWS EC2 Instance

- Go to AWS Console
- Select ubuntu Operating system
- Instances(running)
- Launch instances

GitHub URL: https://github.com/RitikPyCode/django-notes-app





```
ubuntu@ip-172-31-40-193:~$ cd django-notes-app/
ubuntu@ip-172-31-40-193:~/django-notes-app$ ls
 Dockerfile      README.md   db.sqlite3           manage.py   notesapp   requirements.txt
'Project code'   api         docker-compose.yml   mynotes     procfile   staticfiles
ubuntu@ip-172-31-40-193:~/django-notes-app$
```

```
ubuntu@ip-172-31-40-193:~/django-notes-app$ sudo apt-get update
Hit:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
ubuntu@ip-172-31-40-193:~/django-notes-app$
```

```
ubuntu@ip-172-31-40-193:~/django-notes-app$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
```

```
ubuntu@ip-172-31-40-193:~/django-notes-app$ docker --version
Docker version 20.10.21, build 20.10.21-0ubuntu1~22.04.3
ubuntu@ip-172-31-40-193:~/django-notes-app$
```

Ubuntu has not docker permission:

```
ubuntu@ip-172-31-40-193:~/django-notes-app$ docker ps
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2F
1.24/containers/json": dial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-40-193:~/django-notes-app$ whoami
ubuntu
ubuntu@ip-172-31-40-193:~/django-notes-app$
```

We need to modify the user permission and append the ubuntu user in docker group:

Then reboot your system once and wait for few minutes.

```
ubuntu@ip-172-31-40-193:~/django-notes-app$ sudo usermod -aG docker $USER
```

```
ubuntu@ip-172-31-40-193:~/django-notes-app$ sudo reboot
ubuntu@ip-172-31-40-193:~/django-notes-app$
```

```
ubuntu@ip-172-31-40-193:~/django-notes-app$ ls
 Dockerfile      README.md   db.sqlite3          manage.py   notesapp   requirements.txt
'Project code'   api         docker-compose.yml  mynotes     procfile   staticfiles
ubuntu@ip-172-31-40-193:~/django-notes-app$ cat Dockerfile
FROM python:3.9

WORKDIR /app/backend

COPY requirements.txt /app/backend
RUN pip install -r requirements.txt

COPY . /app/backend

EXPOSE 8000

CMD python /app/backend/manage.py runserver 0.0.0.0:8000
ubuntu@ip-172-31-40-193:~/django-notes-app$
```

Docker file:

FROM python:3.9

WORKDIR /app/backend

COPY requirements.txt /app/backend

RUN pip install -r requirements.txt

COPY . /app/backend

EXPOSE 8000

CMD python /app/backend/manage.py runserver 0.0.0.0:8000

We have successfully built.

```
ubuntu@ip-172-31-40-193:~/django-notes-app$ docker build -t notes-app .
Sending build context to Docker daemon  8.389MB
Step 1/7 : FROM python:3.9
3.9: Pulling from library/python
d52e4f012db1: Extracting [==================================================>]  49.55MB/49.55MB
7dd206bea61f: Download complete
2320f9be4a9c: Download complete
6e5565e0ba8d: Download complete
d3797e13cc41: Download complete
9421d62bef94: Download complete
19d7d75bfdca: Download complete
67f28346abde: Download complete
```

Jenkins Installation Pre-requisites.

sudo apt update
sudo apt install openjdk-17-jre
java -version
sudo apt-get update

```
ubuntu@ip-172-31-40-193:~/django-notes-app$ java -version
openjdk version "17.0.7" 2023-04-18
OpenJDK Runtime Environment (build 17.0.7+7-Ubuntu-0ubuntu122.04.2)
OpenJDK 64-Bit Server VM (build 17.0.7+7-Ubuntu-0ubuntu122.04.2, mixed mode, sharing)
ubuntu@ip-172-31-40-193:~/django-notes-app$
```

Installation of Jenkins

curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \

/usr/share/keyrings/jenkins-keyring.asc > /dev/null

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \

https://pkg.jenkins.io/debian-stable binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update

sudo apt-get install Jenkins

service Jenkins status

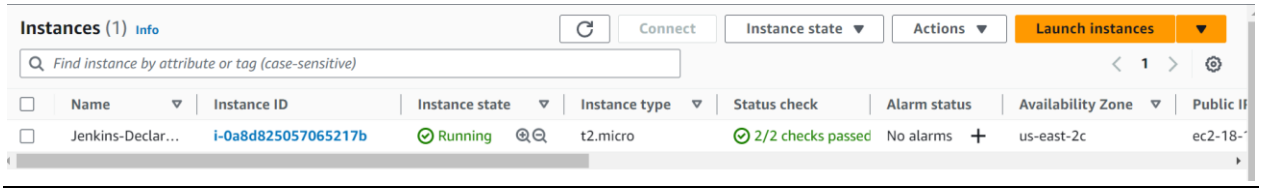
```
ubuntu@ip-172-31-40-193:~/django-notes-app$ service jenkins status
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2023-07-10 19:21:21 UTC; 58s ago
   Main PID: 5902 (java)
      Tasks: 44 (limit: 1141)
     Memory: 289.6M
        CPU: 38.728s
     CGroup: /system.slice/jenkins.service
             └─5902 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080
```

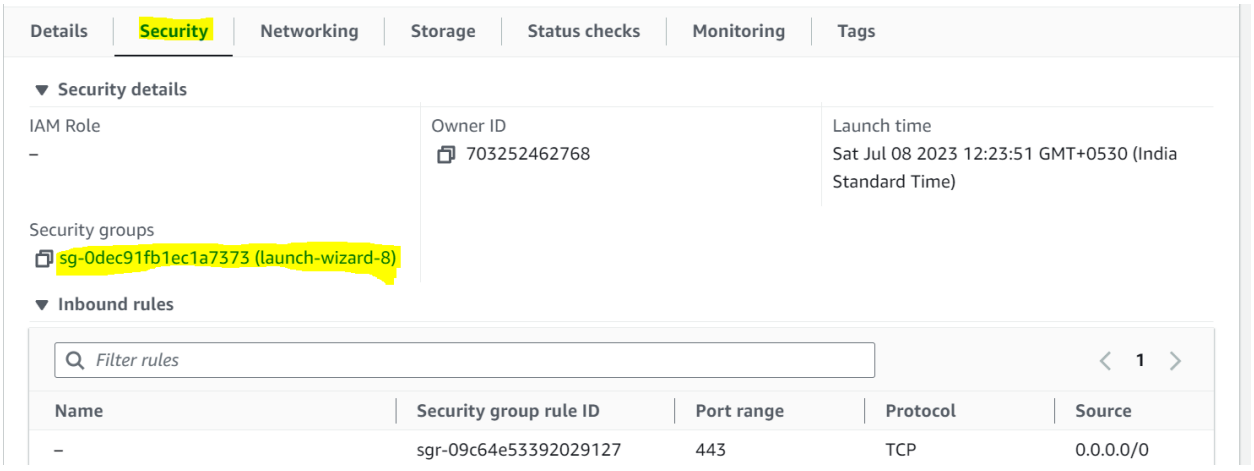
Now you need to go to your Brower and paste the public IP address of your server eg: http://PublicI:8080/

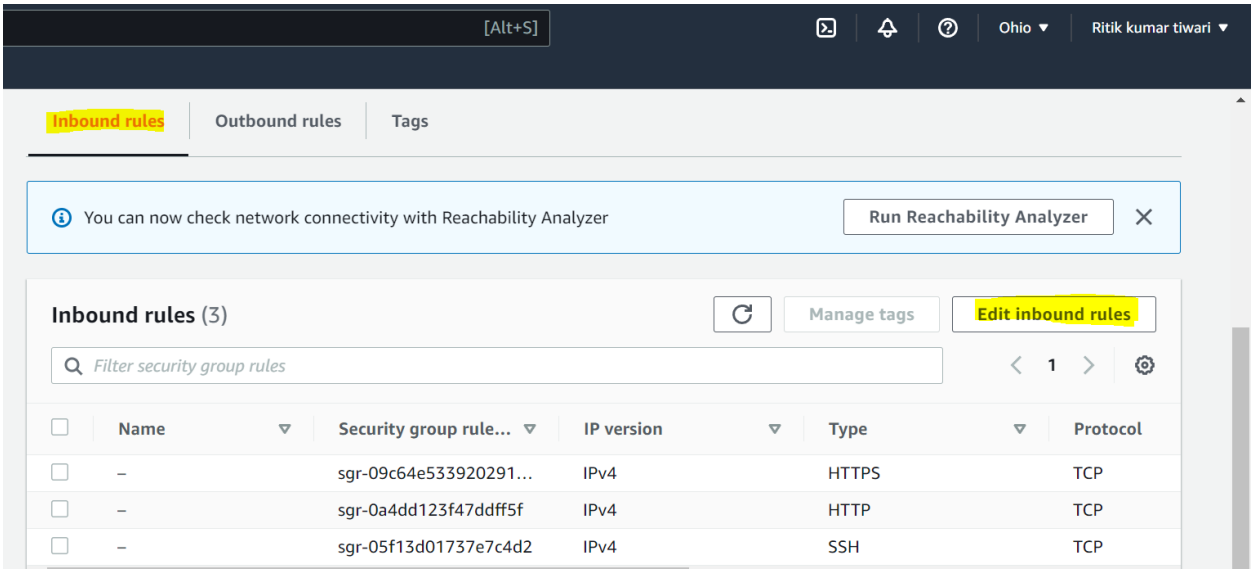
Jenkins runs on 8080 port.

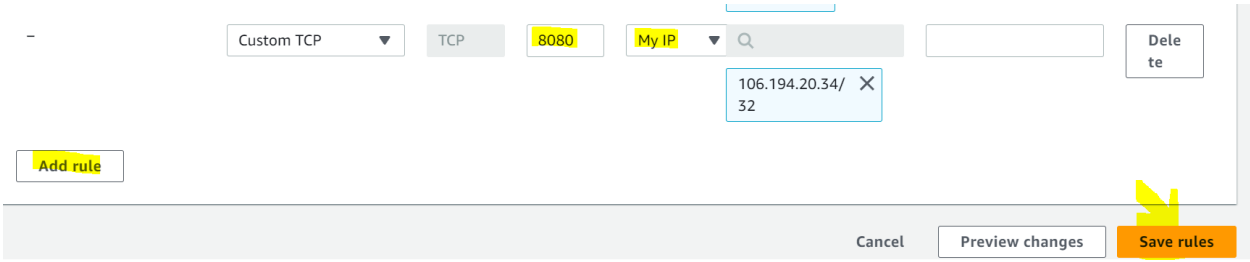
If port is not enabled, then you need to enable that port: 3000 on your aws security group:

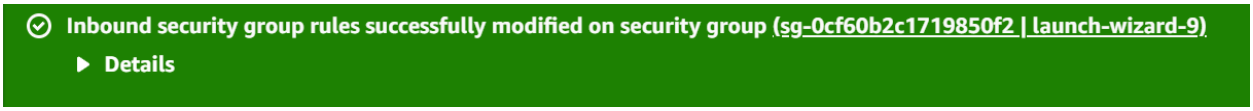
Steps:


Instances (1) Info
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Jenkins-Declar...
i-0a8d825057065217b
Running
t2.micro
2/2 checks passed
No alarms
us-east-2c

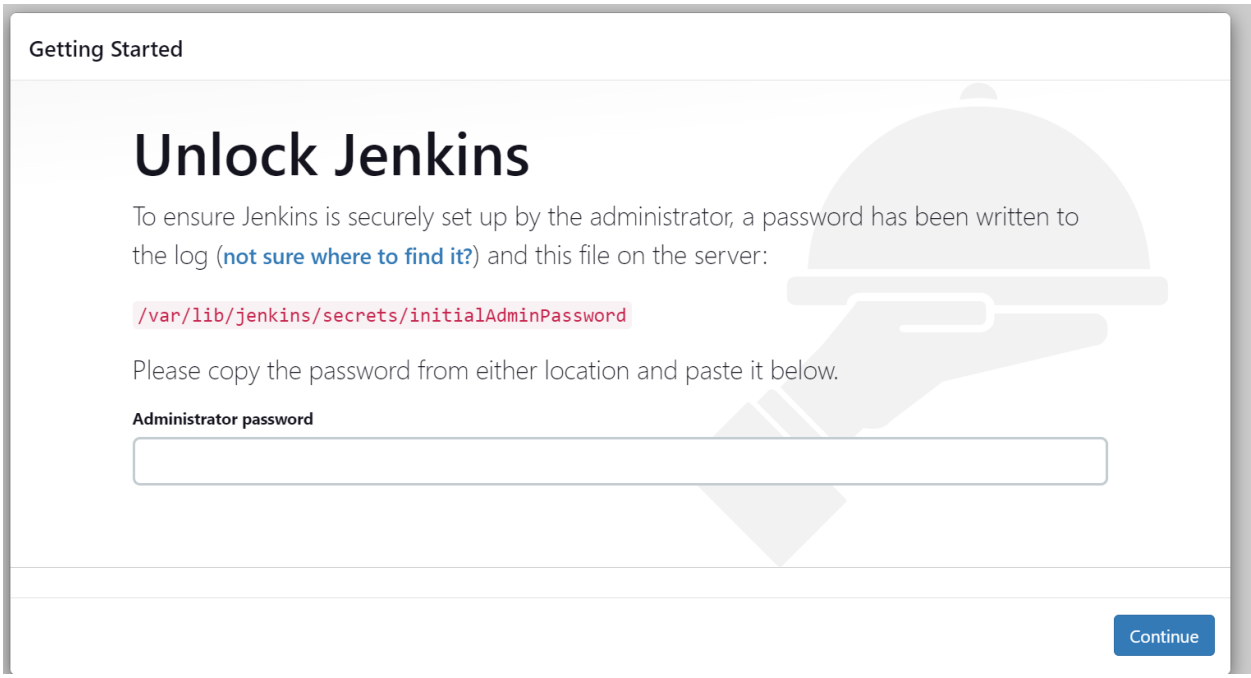

Details
Security
Networking
Storage
Status checks
Monitoring
Tags
Security details
IAM Role
–
Owner ID
703252462768
Launch time
Sat Jul 08 2023 12:23:51 GMT+0530 (India Standard Time)
Security groups
sg-0dec91fb1ec1a7373 (launch-wizard-8)
Inbound rules
Filter rules
Name
Security group rule ID
Port range
Protocol
Source
–
sgr-09c64e53392029127
443
TCP
0.0.0.0/0


[Alt+S]
Ohio
Ritik kumar tiwari
Inbound rules
Outbound rules
Tags
You can now check network connectivity with Reachability Analyzer
Run Reachability Analyzer
Inbound rules (3)
Manage tags
Edit inbound rules
Filter security group rules
Name
Security group rule...
IP version
Type
Protocol
–
sgr-09c64e533920291...
IPv4
HTTPS
TCP
–
sgr-0a4dd123f47ddff5f
IPv4
HTTP
TCP
–
sgr-05f13d01737e7c4d2
IPv4
SSH
TCP



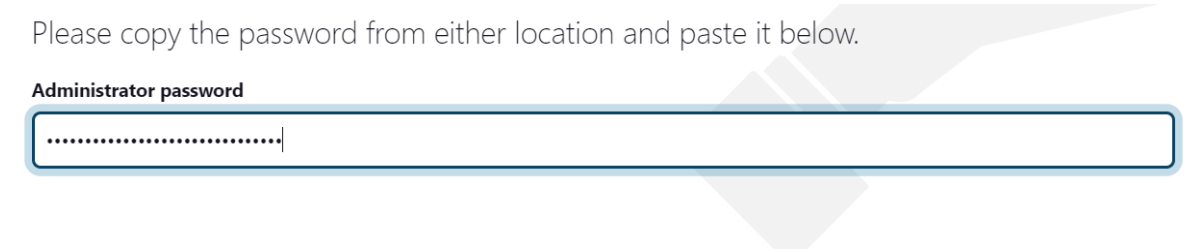


Paste it on your browser: http://PublicI:8080/

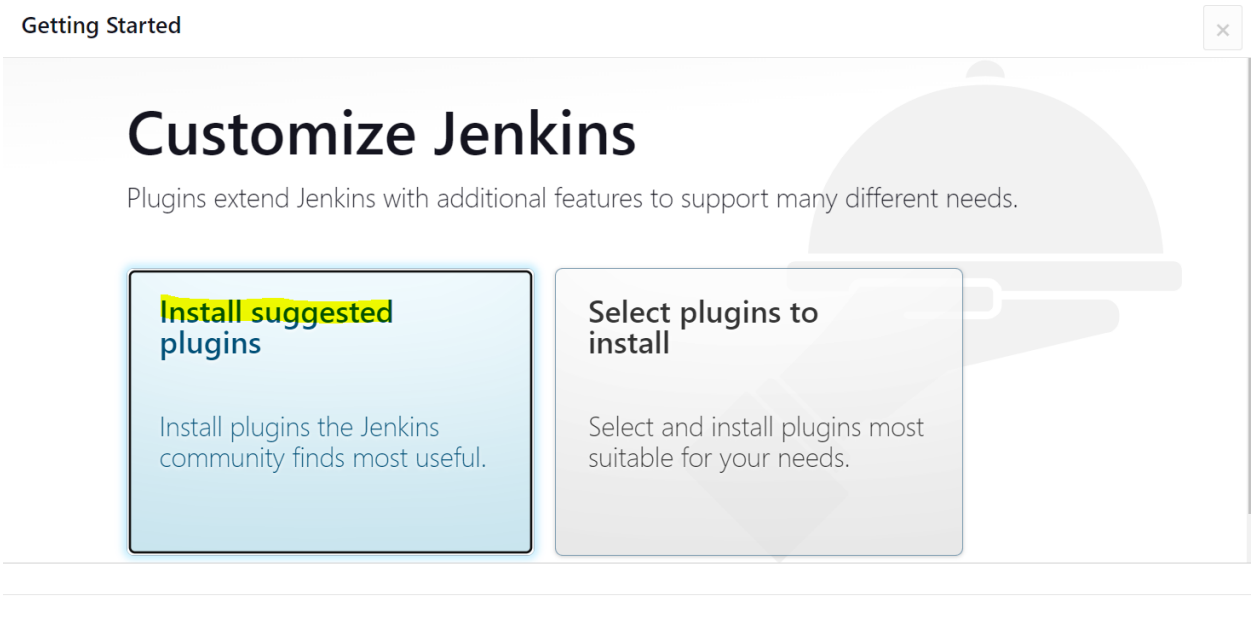


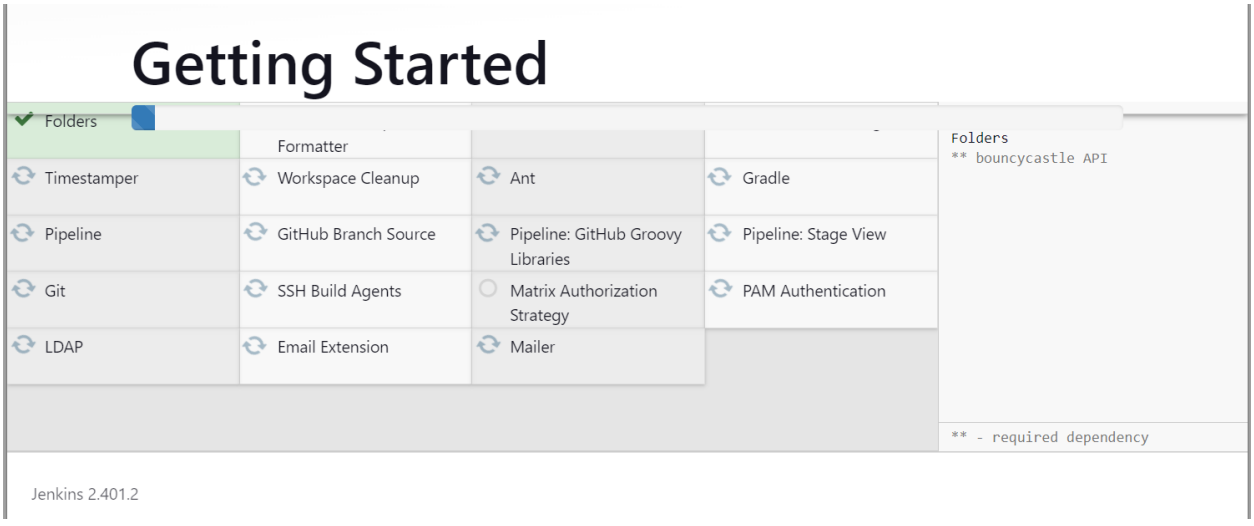
Copy and Cat this command with sudo permission:
/var/lib/jenkins/secrets/initialAdminPassword, you will get initial password.

```
ubuntu@ip-172-31-40-193:~/django-notes-app$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
5c3d4da49bdb41d0b005456e08c0d49b
ubuntu@ip-172-31-40-193:~/django-notes-app$
```



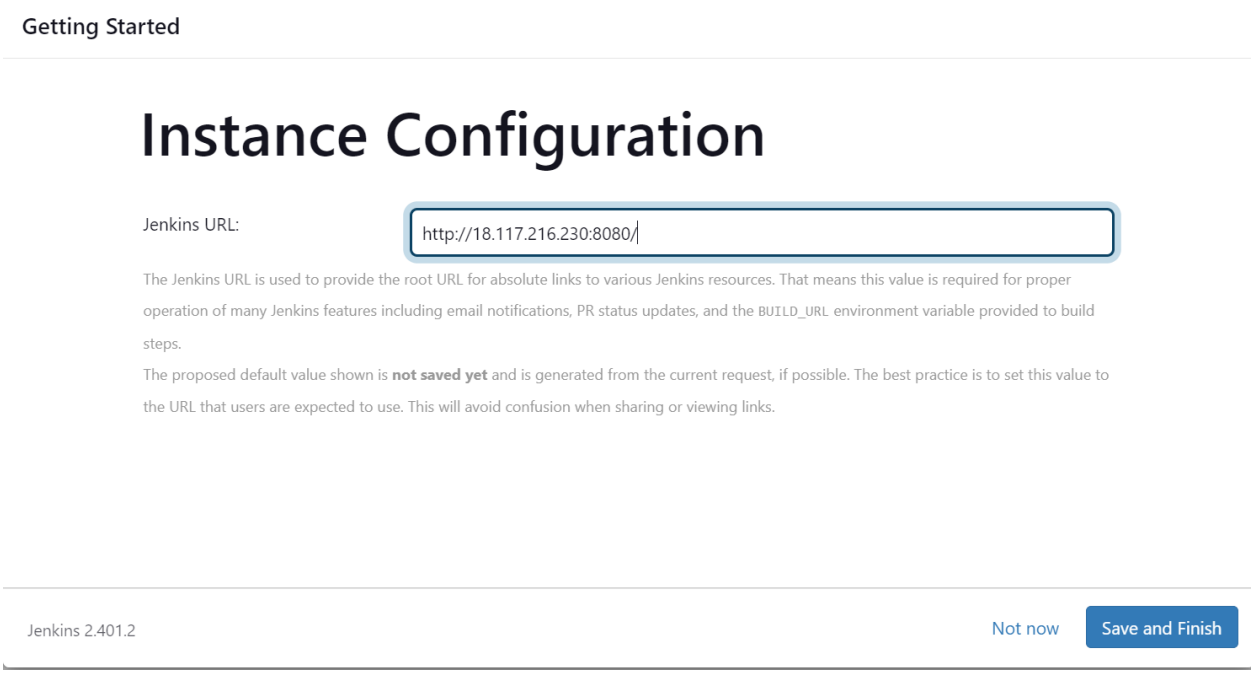
Select suggested.



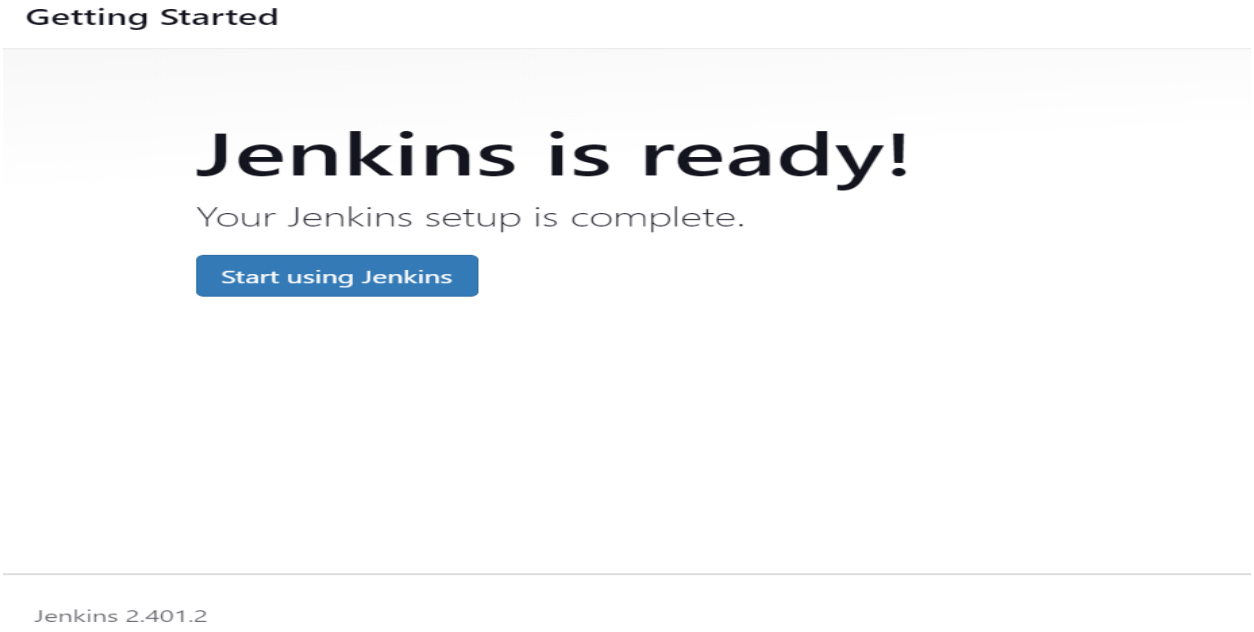


Create your username and Password:

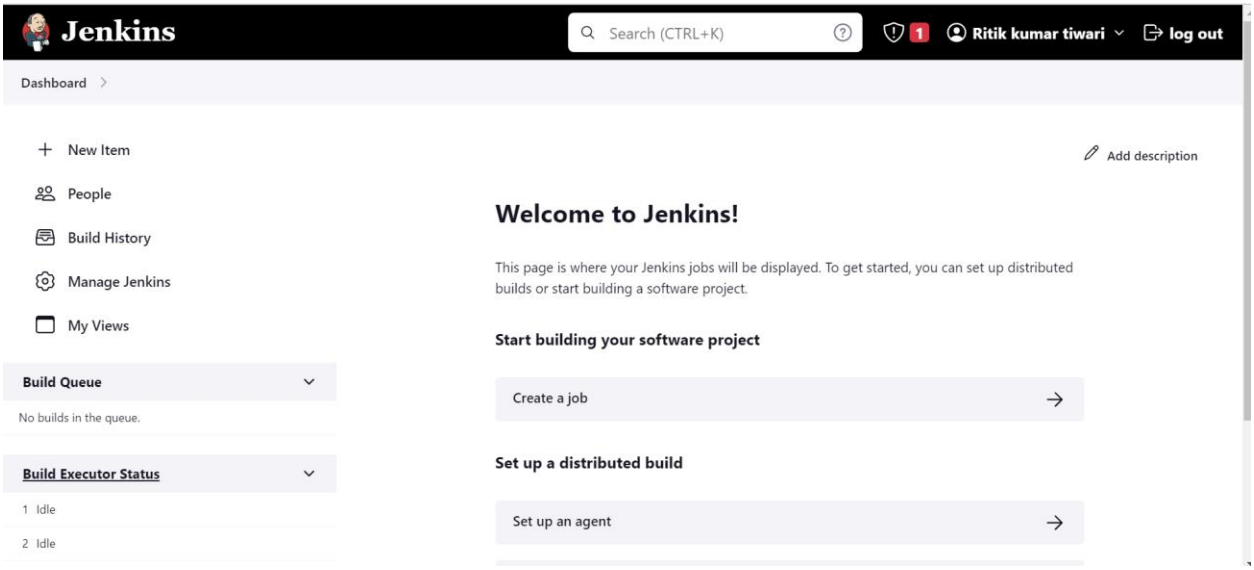
Save and Finish

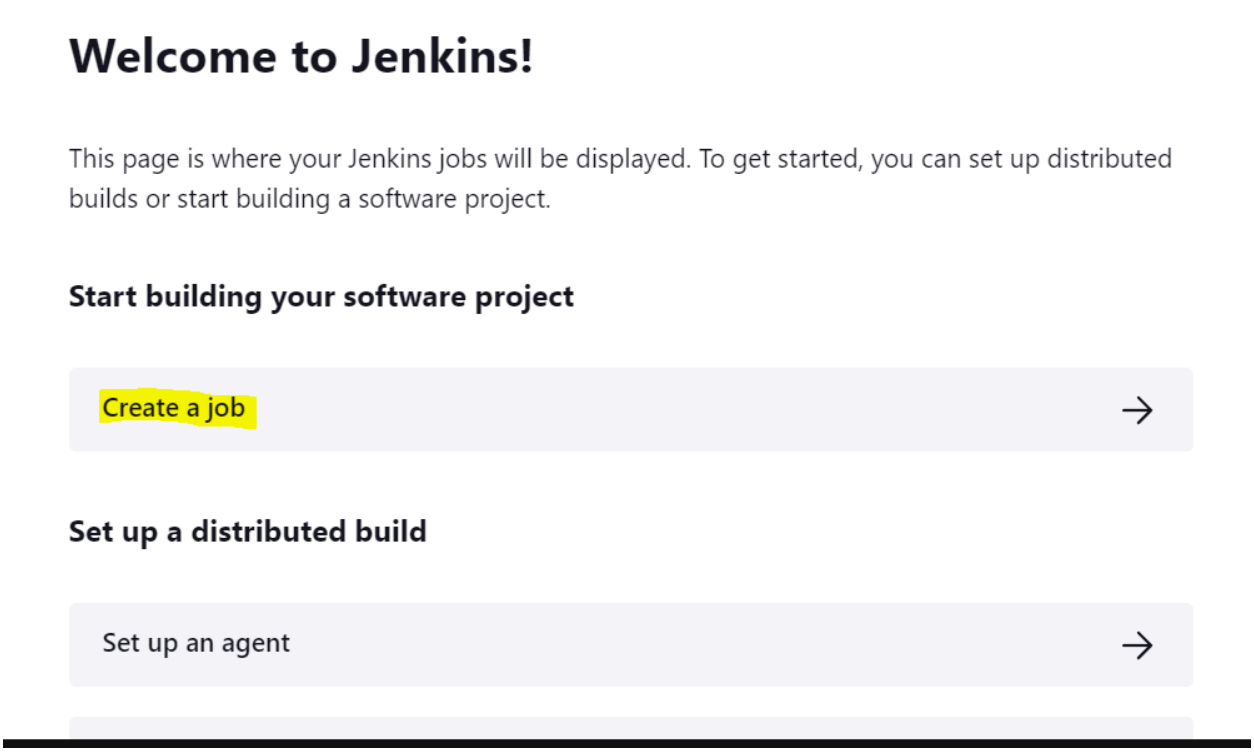


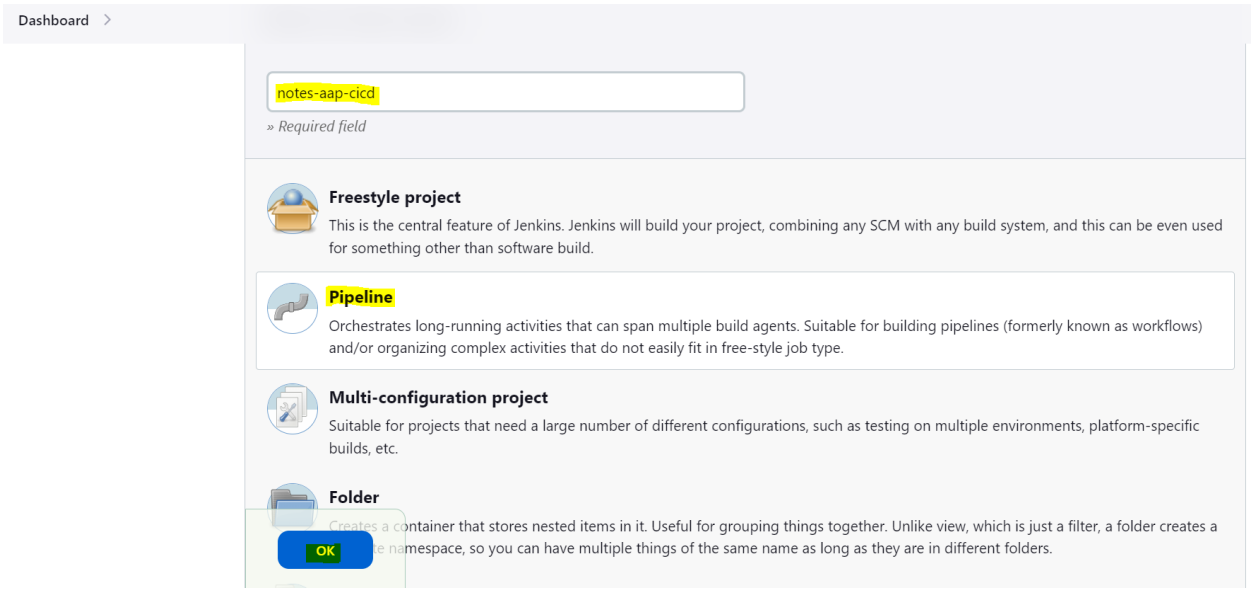
Start using Jenkins

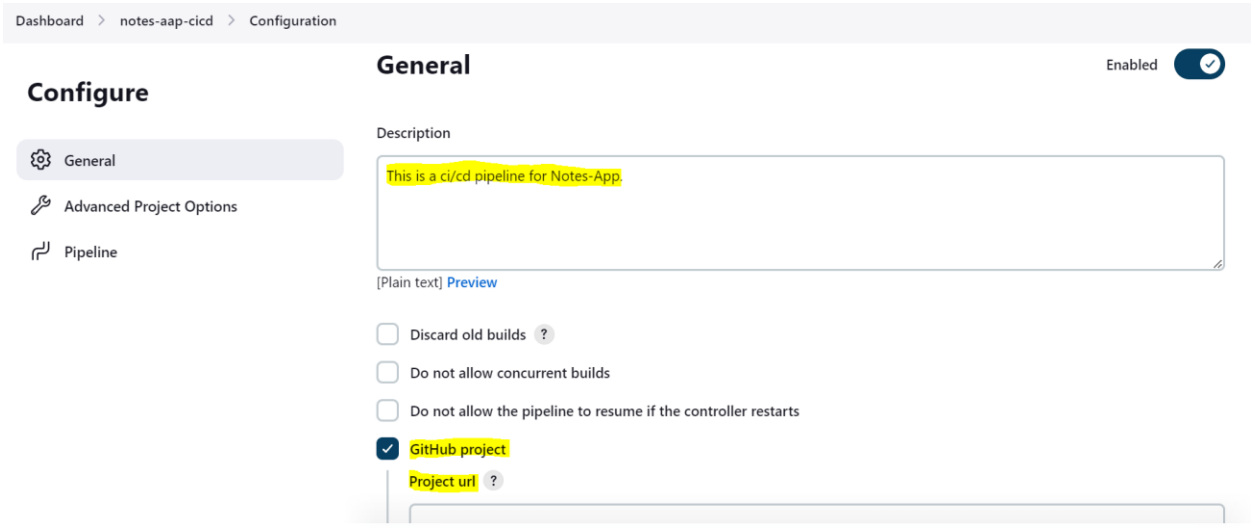


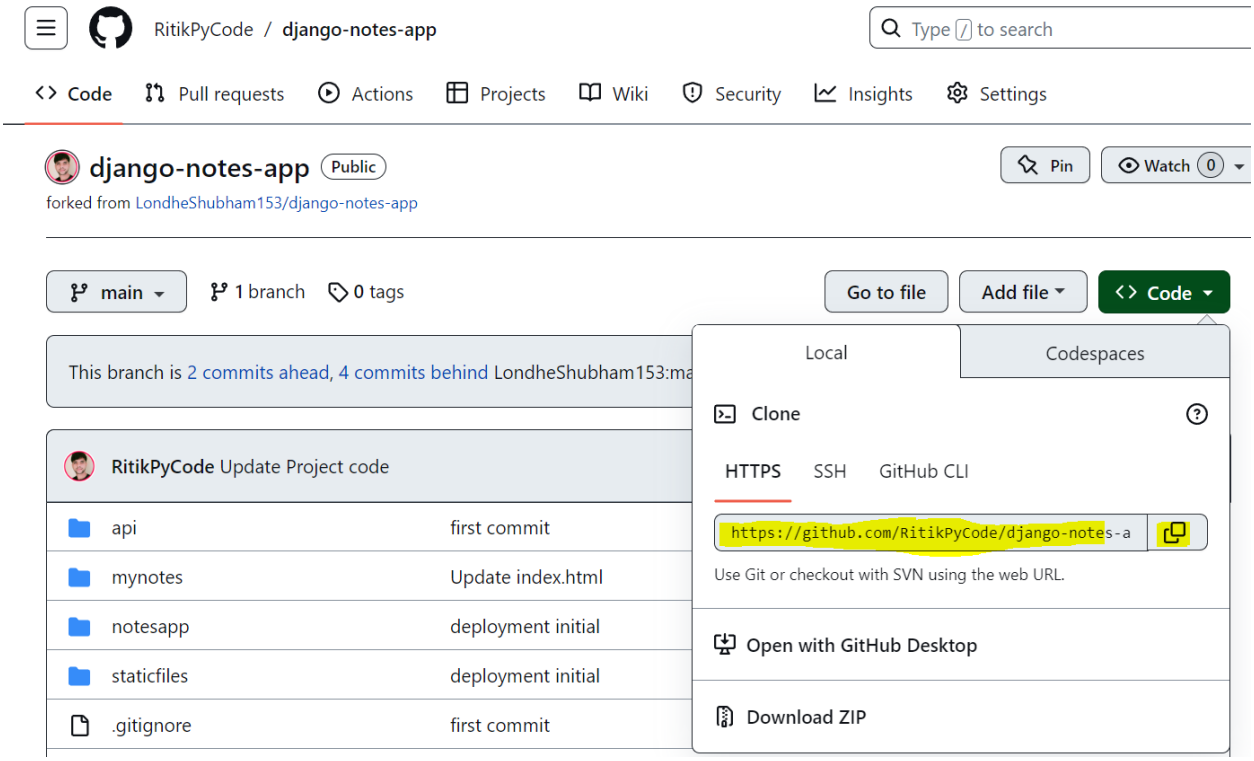
Dashboard layout

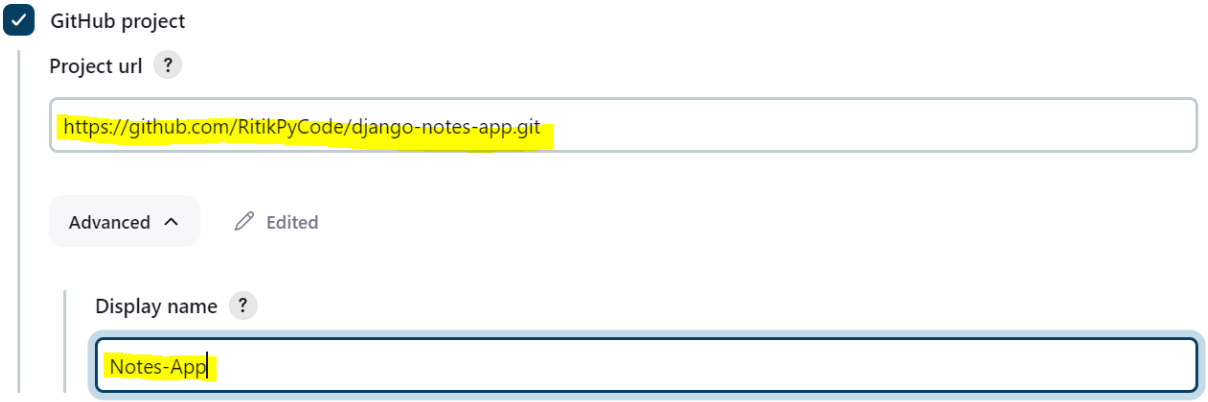





Dashboard
notes-aap-cicd
» Required field
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
Folder
OK


Dashboard
notes-aap-cicd
Configuration
Configure
General
Advanced Project Options
Pipeline
General
Enabled
Description
This is a ci/cd pipeline for Notes-App.
[Plain text] Preview
Discard old builds
Do not allow concurrent builds
Do not allow the pipeline to resume if the controller restarts
GitHub project
Project url


RitikPyCode / django-notes-app
Type / to search
Code
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
django-notes-app
Public
forked from LondheShubham153/django-notes-app
Pin
Watch 0
main
1 branch
0 tags
Go to file
Add file
Code
This branch is 2 commits ahead, 4 commits behind LondheShubham153:ma
RitikPyCode Update Project code
api first commit
mynotes Update index.html
notesapp deployment initial
staticfiles deployment initial
.gitignore first commit
Local
Codespaces
Clone
HTTPS
SSH
GitHub CLI
https://github.com/RitikPyCode/django-notes-a
Use Git or checkout with SVN using the web URL.
Open with GitHub Desktop
Download ZIP


GitHub project
Project url
https://github.com/RitikPyCode/django-notes-app.git
Advanced
Edited
Display name
Notes-App

Build Triggers

- [ ] Build after other projects are built ?
- [ ] Build periodically ?
- [x] GitHub hook trigger for GITScm polling ?
- [ ] Poll SCM ?
- [ ] Quiet period ?
- [ ] Trigger builds remotely (e.g., from scripts) ?

This is pipeline structure, where all steps are there. Code, Build, push to Docker Hub, Deploy

Script ?

try sample Pipeline...

```
pipeline {
    agent any

    stages{

        stage("Code"){

        }
        stage("Build"){

        }
        stage("Push to Docker Hub"){

        }
        stage("Deploy"){

        }
    }

```

This is Groovy syntax, and it's the structure of Declarative Pipeline , Let me show you the output of every stages.

Script ?

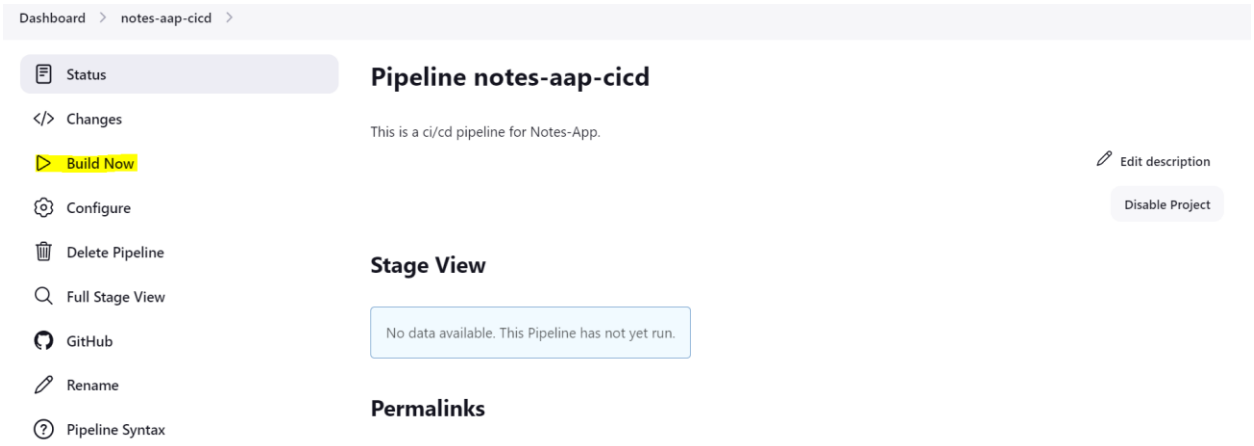
```
pipeline {
    agent any

    stages{

        stage("Code"){
         steps{
            echo "Cloning the code from github"
            }
        }
        stage("Build"){
            steps{
                echo "Building the Images"
            }
        }
        stage("Push to Docker Hub"){
         steps{
            echo "Pushing to the code on docker HUB"
            }
        }
        stage("Deploy"){
            steps{
            echo "Deploying the container"
            }
        }
    }
```

Save Apply

```
pipeline {
   agent any

   stages{

      stage("Code"){
       steps{
         echo "Cloning the code from github"
        }
      }
      stage("Build"){
         steps{
            echo "Building the Images"
        }
      }
```

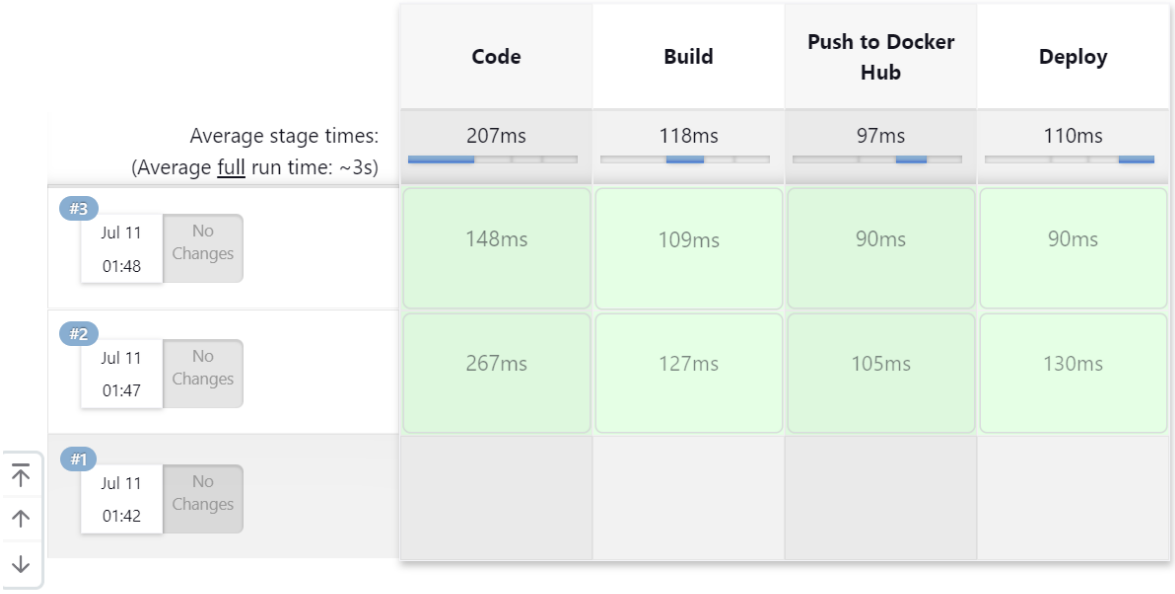
```
    stage("Push to Docker Hub"){
     steps{
        echo "Pushing to the code on docker HUB"
        }
    }
    stage("Deploy"){
        steps{
        echo "Deploying the container"
        }
    }
  }

}
```

Click on Build Now,



Code, Build, Push to Docker HUB, Deploy

Stage View



Permalinks

Finally, Demo has been completed.

Check now if your pipeline running fine. Because we have to check every stage. If we get any error, we will do troubleshooting no problem.

SAVE and BULD NOW

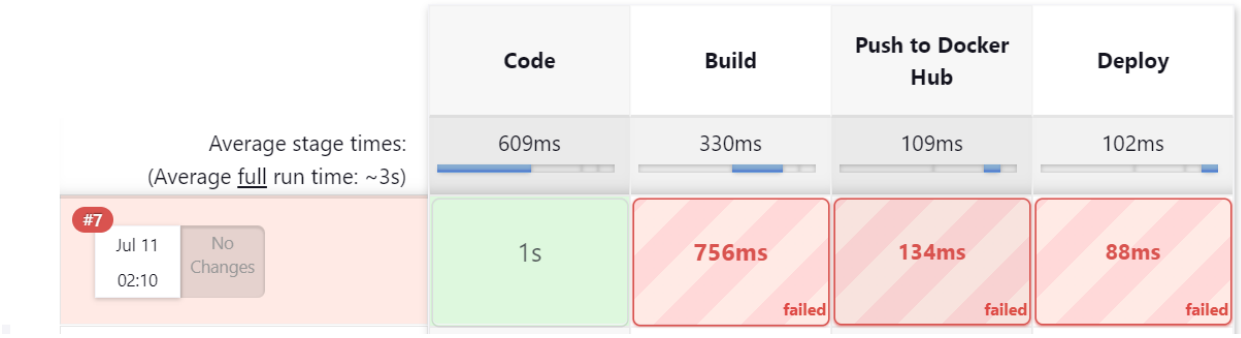
```
pipeline {
    agent any

    stages{

        stage("Code"){
         steps{
            echo "Cloning the code from github"
            git url:"https://github.com/RitikPyCode/django-notes-app.git", branch:"main"

            }
        }
        stage("Build"){
            steps{
                echo "Building the Images"
                sh "docker build -t my-note-app ."
            }
        }
        stage("Push to Docker Hub"){
         steps{
            echo "Pushing to the code on docker HUB"
            }
        }
        stage("Deploy"){
            steps{
            echo "Deploying the container"
            }
        }
    }

}
```

Cloning the Code

```
pipeline {
   agent any

   stages{

      stage("Code"){
       steps{
          echo "Cloning the code from github"
          git url:"https://github.com/RitikPyCode/django-notes-app.git", branch:"main"

          }
       }
      stage("Build"){
```

```
        steps{
            echo "Building the Images"
            sh "docker build -t my-note-app ."
        }
    }
    stage("Push to Docker Hub"){
     steps{
        echo "Pushing to the code on docker HUB"
}
        }
    }
    stage("Deploy"){
        steps{
        echo "Deploying the container"
        }
    }
  }

}
```

Stage View



First Stage has been completed successfully.

/var/lib/jenkins/workspace/notes-aap-cicd

```
Started by user Ritik kumar tiwari
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/notes-aap-cicd
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Code)
```

```
Building the Images
[Pipeline] sh
+ docker build -t my-note-app .
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post
"http://%2Fvar%2Frun%2Fdocker.sock/v1.24/build?
buildargs=%7B%7D&cachefrom=%5B%5D&cgroupparent=&cpuperiod=0&cpuquota=0&cpusetcpus=&cpusetmems=&cpushares=0&docker
file=Dockerfile&labels=%7B%7D&memory=0&memswap=0&networkmode=default&rm=1&shmsize=0&t=my-note-
app&target=&ulimits=null&version=1": dial unix /var/run/docker.sock: connect: permission denied
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
```

Go to this path /var/lib/jenkins/workspace/notes-aap-cicd . all files are here.

```
ubuntu@ip-172-31-40-193:~$ cd /var/lib/jenkins/workspace/notes-aap-cicd
ubuntu@ip-172-31-40-193:/var/lib/jenkins/workspace/notes-aap-cicd$ ls
 Dockerfile      README.md   db.sqlite3          manage.py   notesapp   requirements.txt
'Project code'   api         docker-compose.yml  mynotes     procfile   staticfiles
ubuntu@ip-172-31-40-193:/var/lib/jenkins/workspace/notes-aap-cicd$
```

```
ubuntu@ip-172-31-40-193:/var/lib/jenkins/workspace/notes-aap-cicd$ sudo usermod -aG docker jenkins
ubuntu@ip-172-31-40-193:/var/lib/jenkins/workspace/notes-aap-cicd$ sudo reboot
ubuntu@ip-172-31-40-193:/var/lib/jenkins/workspace/notes-aap-cicd$
```

sudo usermod -aG docker jenkins

sudo reboot

Please refresh the Jenkins software then build now again, you will pass all the stages successfully.

Console Output

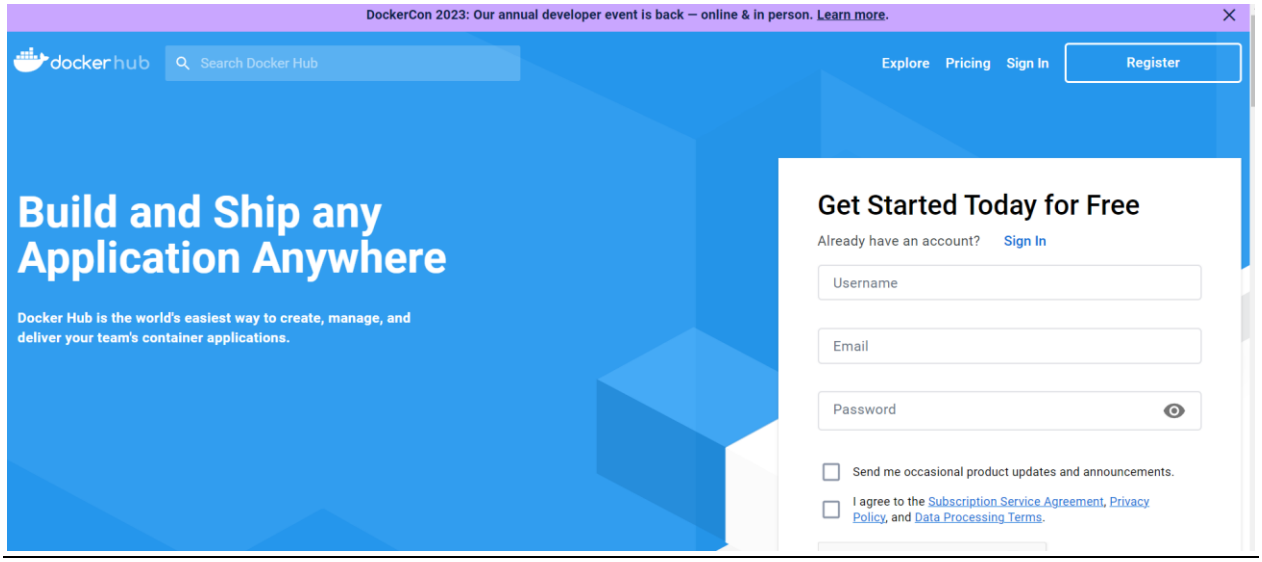
```
Started by user Ritik kumar tiwari
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/notes-aap-cicd
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Code)
```

Stage View

	Code	Build	Push to Docker Hub	Deploy
Average stage times: (Average full run time: ~10s)	759ms	3s	98ms	92ms
#9 Jul 11 02:25 No Changes	1s	18s	103ms	101ms

Now Pushing the code from DOCKER HUB.

First you have to sign In or Signup yourself on docker hub: https://hub.docker.com/

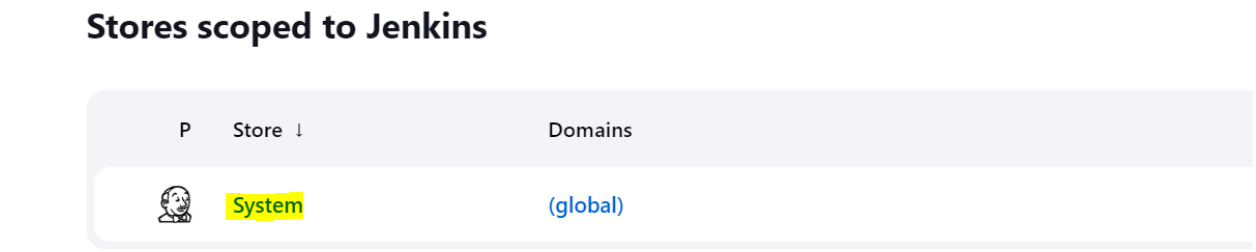


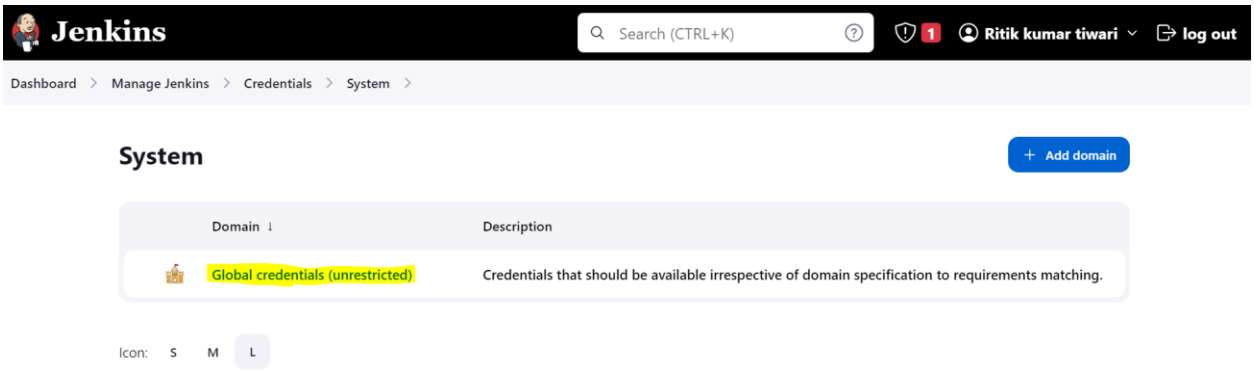

```
ubuntu@ip-172-31-40-193:/$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create on
e.
Username: ritikt2023
Password:
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ubuntu@ip-172-31-40-193:/$
```

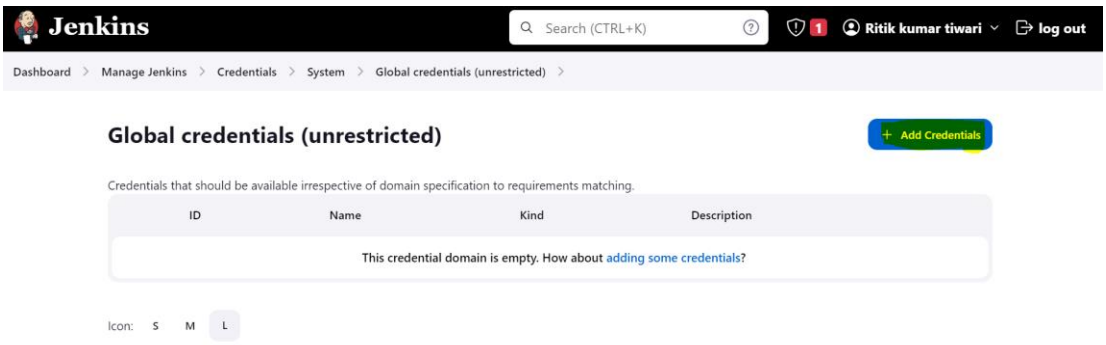
Draw-back related credentials, issue has been resolved below:

sh "docker login", then creds but if we type all the creds then it will be publicly, then we have devops concept: environment variable, it will not show you your creds.

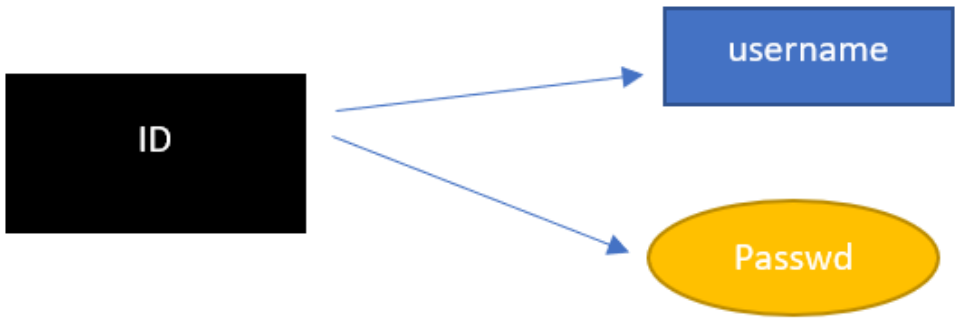
First go to Jenkins Dashboard > Manage Jenkins > Security: Credentials > click on system

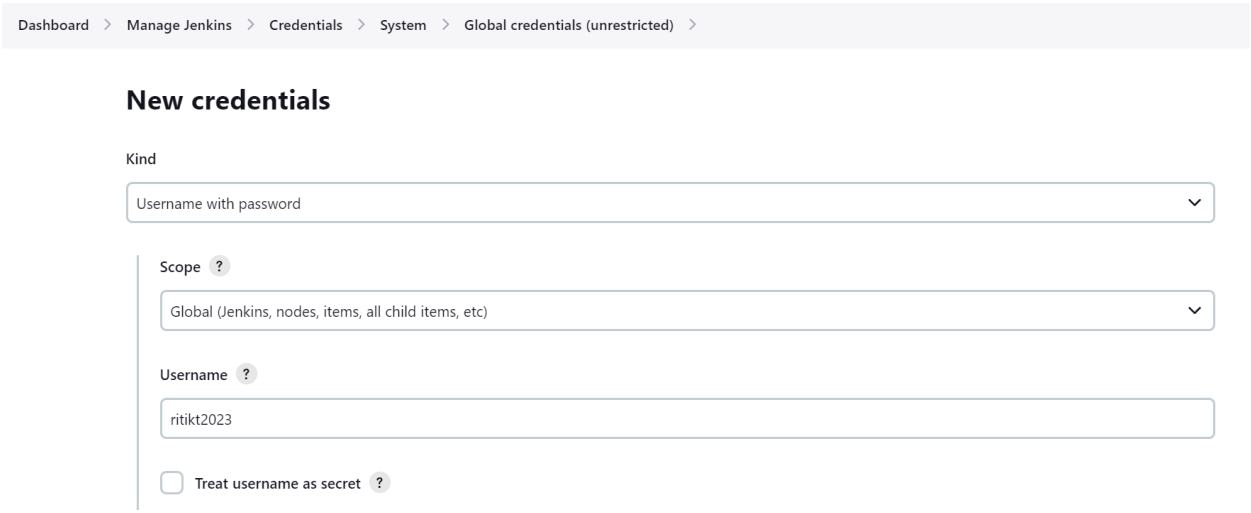


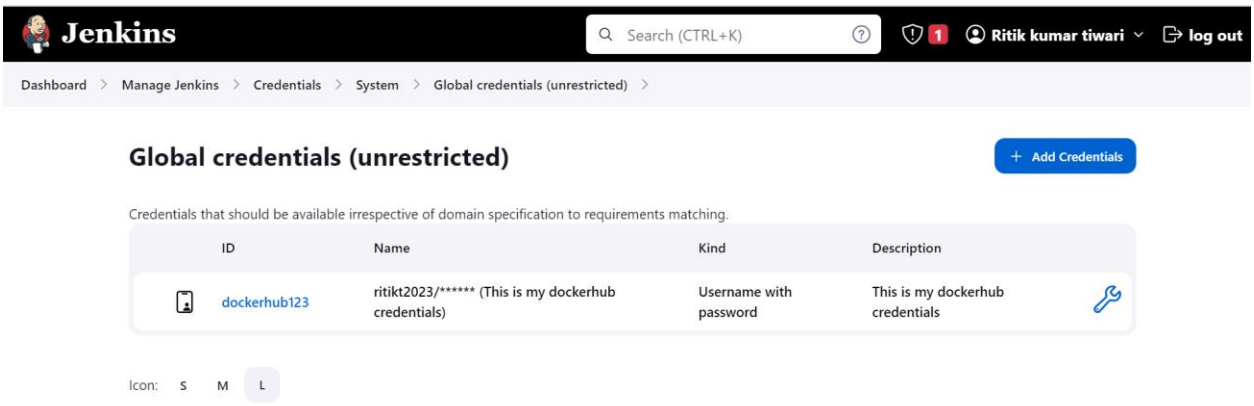




We need to assign ID for username and password, because if anyone ask then we can give our ID no. which contains username and password.





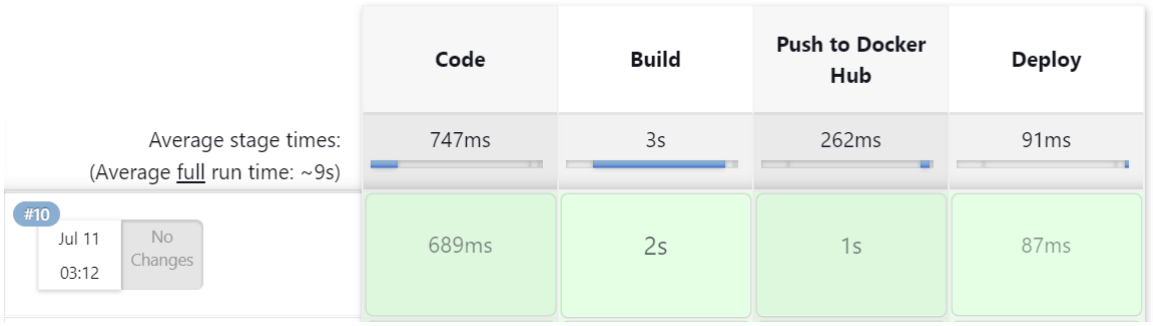


Now back to dashboard and come in configuration

```
pipeline {
   agent any

   stages{

      stage("Code"){
       steps{
         echo "Cloning the code from github"
         git url:"https://github.com/RitikPyCode/django-notes-app.git", branch:"main"
```

```
        }
    }
    stage("Build"){
        steps{
            echo "Building the Images"
            sh "docker build -t my-note-app ."
        }
    }
    stage("Push to Docker Hub"){
      steps{
        echo "Pushing to the code on docker HUB"
        withCredentials([usernamePassword(credentialsId:"dockerhub123", passwordVariable:
"dockerhubPass", usernameVariable:"dockerhubUser")]){
        sh "docker login -u ${env.dockerhubUser} -p ${env.dockerhubPass}"
        }
        }
    }
    stage("Deploy"){
        steps{
        echo "Deploying the container"
        }
    }
  }

}
```

Stage View

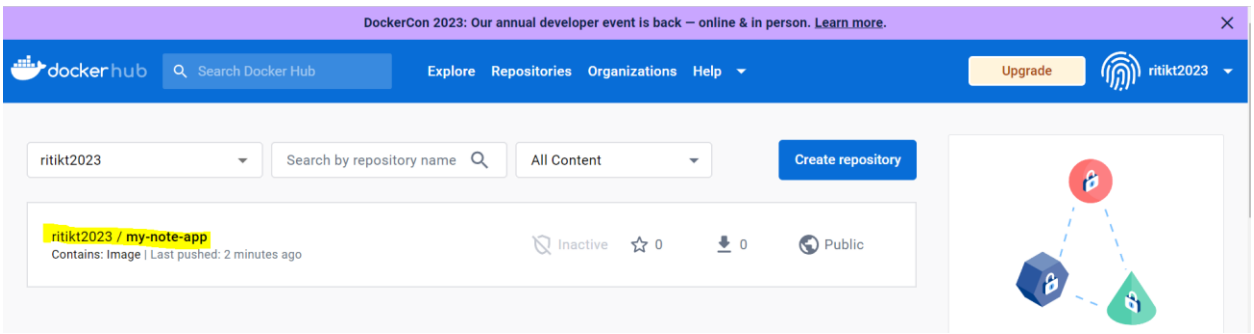


Now we are going to push our image on docker hub using declarative pipeline.

```
pipeline {
   agent any

   stages{

      stage("Code"){
       steps{
         echo "Cloning the code from github"
         git url:"https://github.com/RitikPyCode/django-notes-app.git", branch:"main"

         }
      }
      stage("Build"){
         steps{
            echo "Building the Images"
            sh "docker build -t my-note-app ."
         }
      }
      stage("Push to Docker Hub"){
```

```
        steps{
            echo "Pushing to the code on docker HUB"
            withCredentials([usernamePassword(credentialsId:"dockerhub123", passwordVariable:
"dockerhubPass", usernameVariable:"dockerhubUser")]){
            sh "docker tag my-note-app ${env.dockerhubUser}/my-note-app:latest" (we have to tag our old
image then add usename)
            sh "docker login -u ${env.dockerhubUser} -p ${env.dockerhubPass}"
            sh "docker push ${env.dockerhubUser}/my-note-app:latest"

            }
            }
        }
        stage("Deploy"){
            steps{
            echo "Deploying the container"
            }
        }
    }

}
```

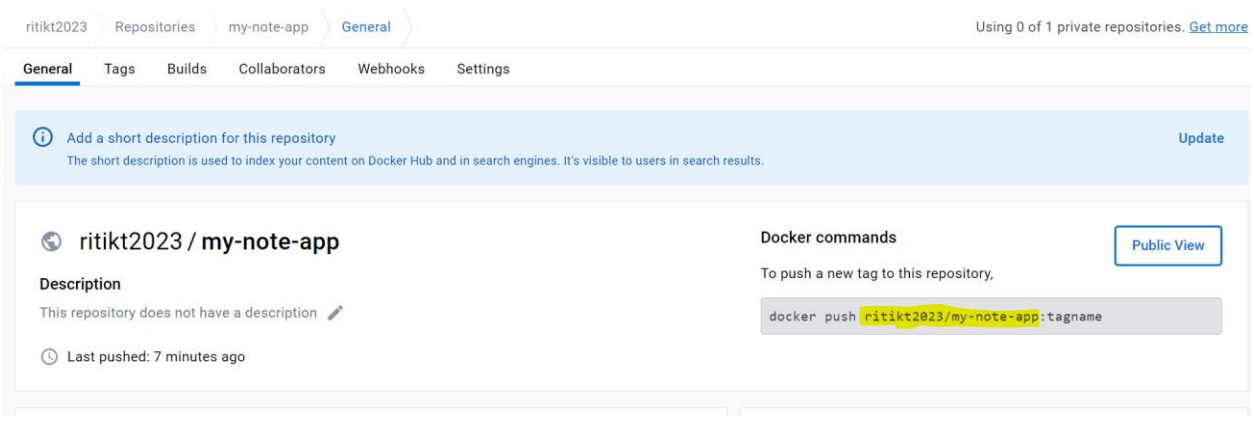
Now refresh your docker hub website:



Stage Deploy:

sh "docker run -d -p 8000:8000 ritikt2023/my-note-app:latest"

```
        stage("Deploy"){
            steps{
            echo "Deploying the container"
            sh "docker run -d -p 8000:8000 ritikt2023/my-note-app:latest"
            }
        }
    }

}
```

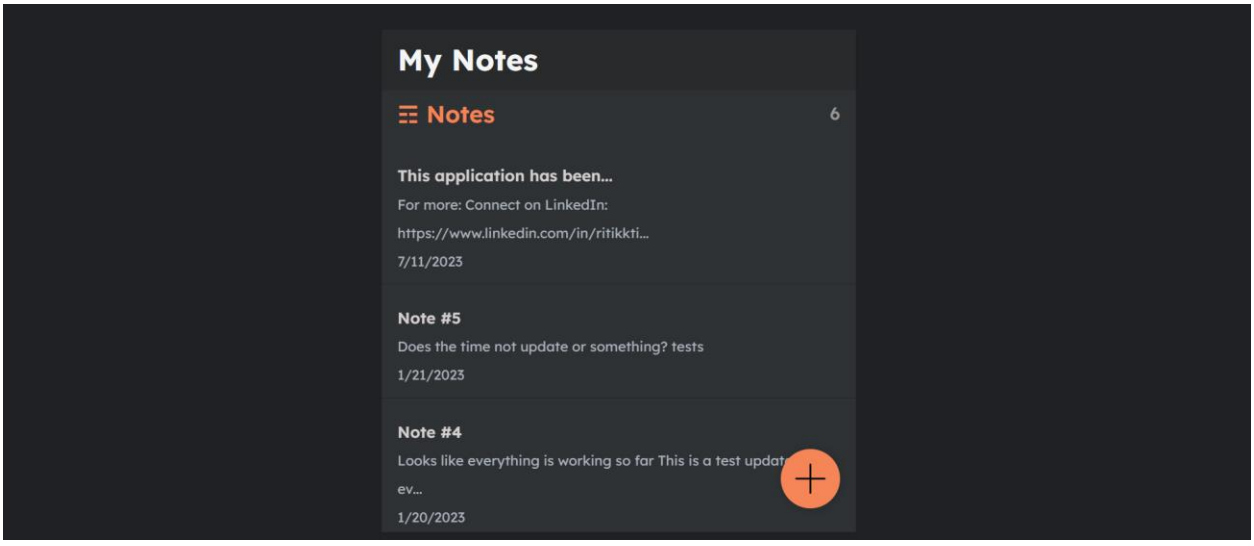


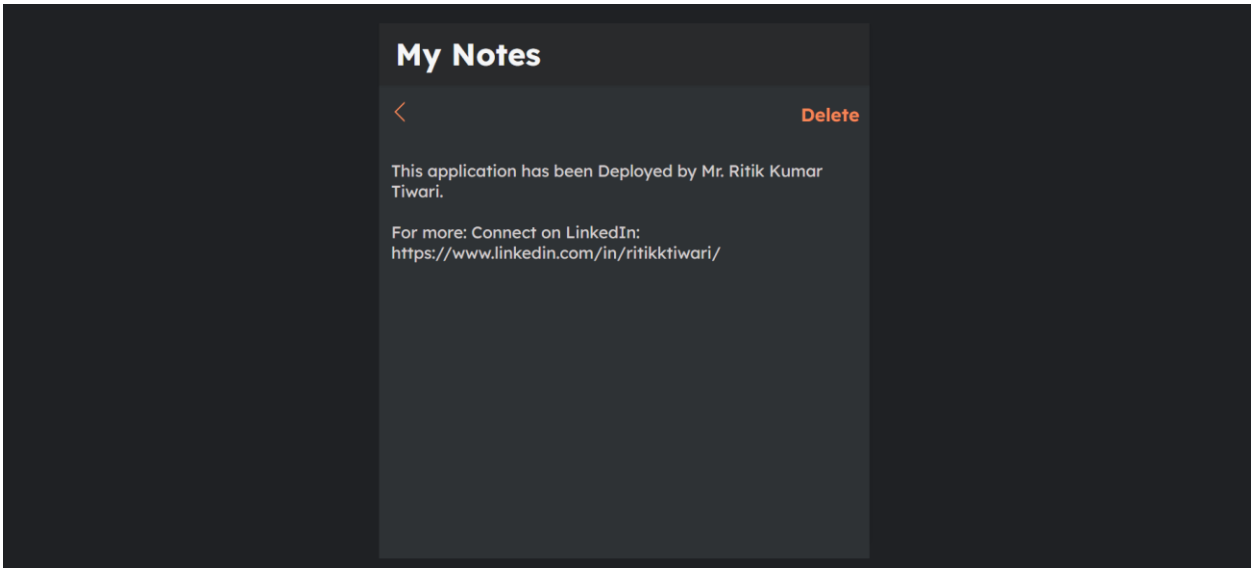
Add 8000 port where your application will run.

Stage View

	Code	Build	Push to Docker Hub	Deploy
Average stage times: (Average full run time: ~11s)	768ms	3s	2s	295ms
#13 Jul 11 03:37 No Changes	666ms	1s	5s	1s

Application has been deployed. You can check with your public Ip address with 8000 ports. Eg: http://18.117.216.230:8000/





If you are going to build now again then pipeline will get failed. The reason is Port is already allocated

Stage View

	Code	Build	Push to Docker Hub	Deploy
Average stage times: (Average full run time: ~11s)	765ms	3s	2s	339ms
#14 Jul 11 03:44 No Changes	742ms	2s	5s	691ms failed

Console Output

```
+ docker run -d -p 8000:8000 ritikt2023/my-note-app:latest
d152e277452511e1b64d912d0ee2e256086e0548054d01192bfff60220a02c3b
docker: Error response from daemon: driver failed programming external connectivity on endpoint clever_wescoff
(29b80c912703e74ebb29884312baf342a1504b77b443bdd2cc17099abf5656ee): Bind for 0.0.0.0:8000 failed: port is already
allocated.
```

***Docker Compose** is a tool that was developed to help define and share multi-container applications. With Compose, we can create a YAML file to define the services and with a single command, can spin everything up or tear it all down.*

Install the DOCKER-COMPOSE

sudo apt-get install docker-compose

```
ubuntu@ip-172-31-40-193:/$ sudo apt-get install docker-compose
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-docker python3-dockerpty python3-docopt python3-dotenv python3-texttable python3-websocket
The following NEW packages will be installed:
  docker-compose python3-docker python3-dockerpty python3-docopt python3-dotenv python3-texttable python3-websocket
0 upgraded, 7 newly installed, 0 to remove and 75 not upgraded.
Need to get 290 kB of archives.
```

```
ubuntu@ip-172-31-40-193:/$ docker-compose -version
docker-compose version 1.29.2, build unknown
ubuntu@ip-172-31-40-193:/$
```

It's type of yml file where are the things are explained.

Code Blame 6 lines (6 loc) · 86 Bytes

```
1   version : "3.3"
2   services :
3     web :
4       build : .
5       ports :
6           - "8000:8000"
```

Container or image ko down kiya aur phir up kiya inhi code se.

If we are going to run that application with the help of docker-compose what will happen, it will throw error,

```
ubuntu@ip-172-31-40-193:~/django-notes-app$ docker-compose up -d
Creating network "django-notes-app_default" with the default driver
Building web
Sending build context to Docker daemon  8.389MB
Step 1/7 : FROM python:3.9
 ---> 1d7821476b67
Step 2/7 : WORKDIR /app/backend
 ---> Using cache
```

for this error we need to kill running container.

```
ERROR: for django-notes-app_web_1  Cannot start service web: driver failed programming external connectivity on endpoint django-notes-app_web_1
920397336b9aab01f6291ef6accd4034eee1a0e51c6292ad885d17124365203): Bind for 0.0.0.0:8000 failed: port is already allocated

ERROR: for web  Cannot start service web: driver failed programming external connectivity on endpoint django-notes-app_web_1 (9920397336b9aab01f
91ef6accd4034eee1a0e51c6292ad885d17124365203): Bind for 0.0.0.0:8000 failed: port is already allocated
ERROR: Encountered errors while bringing up the project.
ubuntu@ip-172-31-40-193:~/django-notes-app$
```

```
ubuntu@ip-172-31-40-193:~/django-notes-app$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                                       NAMES
7d65280937fd   e4ca11059a74   "/bin/sh -c 'python …"   46 minutes ago   Up 46 minutes   0.0.0.0:8000->8000/tcp, :::8000->8000/tcp   suspicious_cla
rke
ubuntu@ip-172-31-40-193:~/django-notes-app$ docker kill 7d65280937fd
7d65280937fd
ubuntu@ip-172-31-40-193:~/django-notes-app$
```

Now run docker-compose up -d, this will decide automatically, image or container name.

```
ubuntu@ip-172-31-40-193:~/django-notes-app$ docker-compose up -d
Starting django-notes-app_web_1 ... done
ubuntu@ip-172-31-40-193:~/django-notes-app$ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED         STATUS         PORTS                                       NAMES
df250b605130   django-notes-app_web   "/bin/sh -c 'python …"   5 minutes ago   Up 17 seconds   0.0.0.0:8000->8000/tcp, :::8000->8000/tcp   django-
notes-app_web_1
ubuntu@ip-172-31-40-193:~/django-notes-app$
```

Now there is no container after down

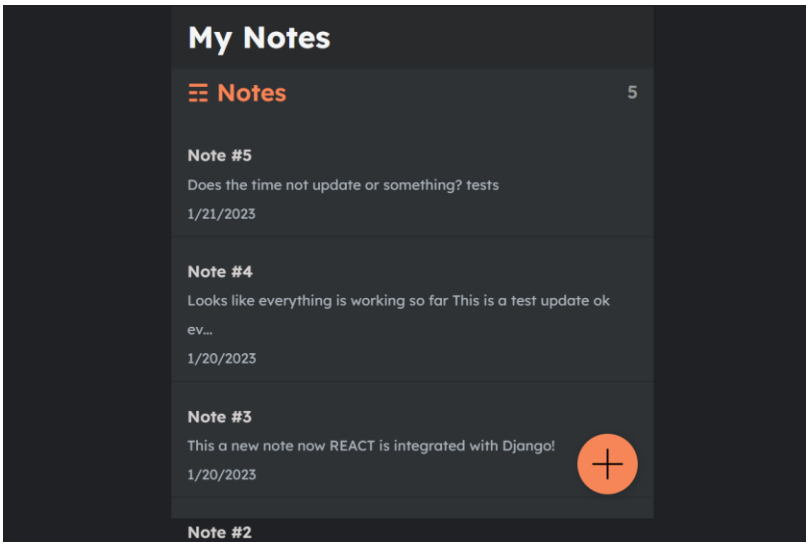
```
ubuntu@ip-172-31-40-193:~/django-notes-app$ docker-compose down
Stopping django-notes-app_web_1 ... done
Removing django-notes-app_web_1 ... done
Removing network django-notes-app_default
ubuntu@ip-172-31-40-193:~/django-notes-app$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
ubuntu@ip-172-31-40-193:~/django-notes-app$
```

This is the solution of above error. Let me add this code in groovy syntax.

```
        }
        stage("Deploy"){
            steps{
            echo "Deploying the container"
            sh "docker-compose down && docker-compose up -d"
            }
        }
    }

}
```

Stage View

	Code	Build	Push to Docker Hub	Deploy
Average stage times: (Average full run time: ~13s)	825ms	3s	3s	680ms
#15 Jul 11 04:30 No Changes	691ms	1s	4s	3s



Lets change the little bit docker-compose file: build replace with image, now we are taking images from docker hub.

django-notes-app / docker-compose.yml in main

Edit Preview

```
1    version : "3.3"
2    services :
3      web :
4         image : ritikt2023/my-note-app:latest
5         ports :
6             - "8000:8000"
7
```

We have committed few things, now you can run as much as possible, never getting error.

Stage View

	Code	Build	Push to Docker Hub	Deploy
Average stage times: (Average full run time: ~15s)	1s	3s	3s	1s
#16 Jul 11 04:42 1 commit	3s	2s	5s	13s

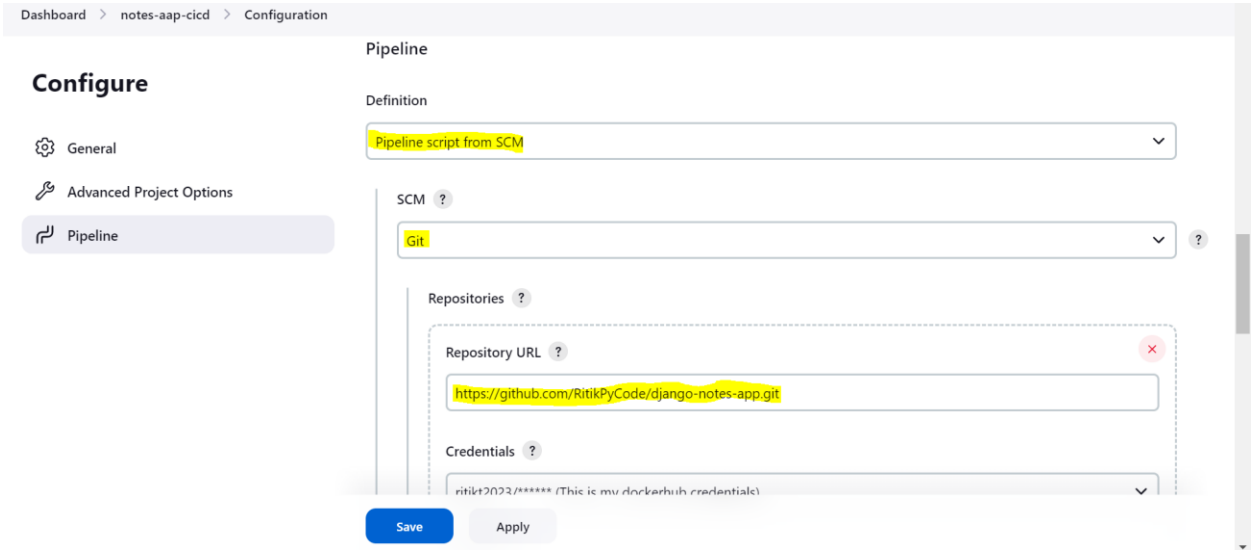
This is called robust pipeline.

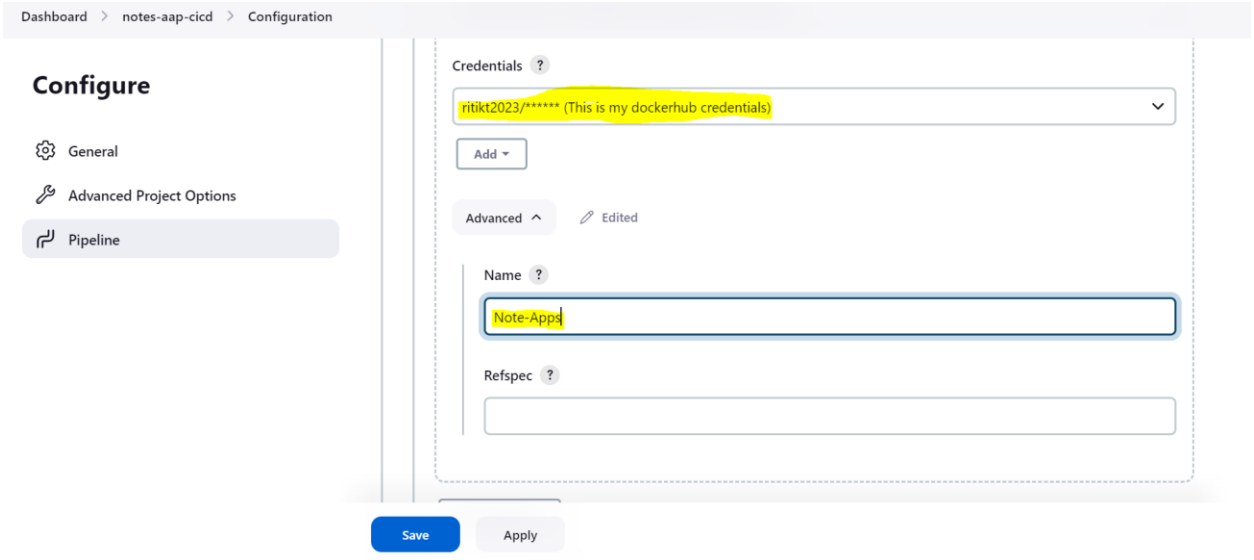
A Robust Pipeline = Predictability + Power + Profit. The foundation of building a pipeline is about blocking out the time and being committed to focused prospecting.

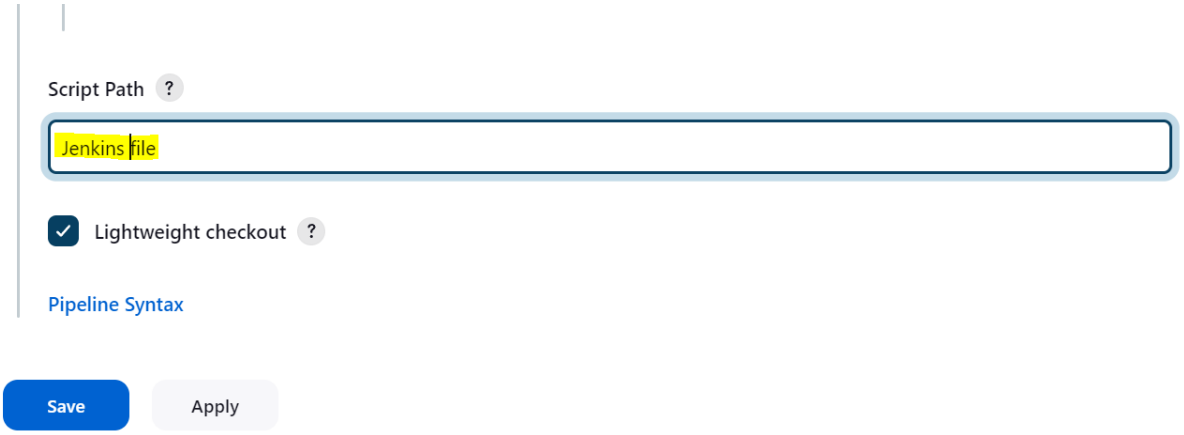
Now let's try our whole code will come from GitHub. Through adding file Jenkins files

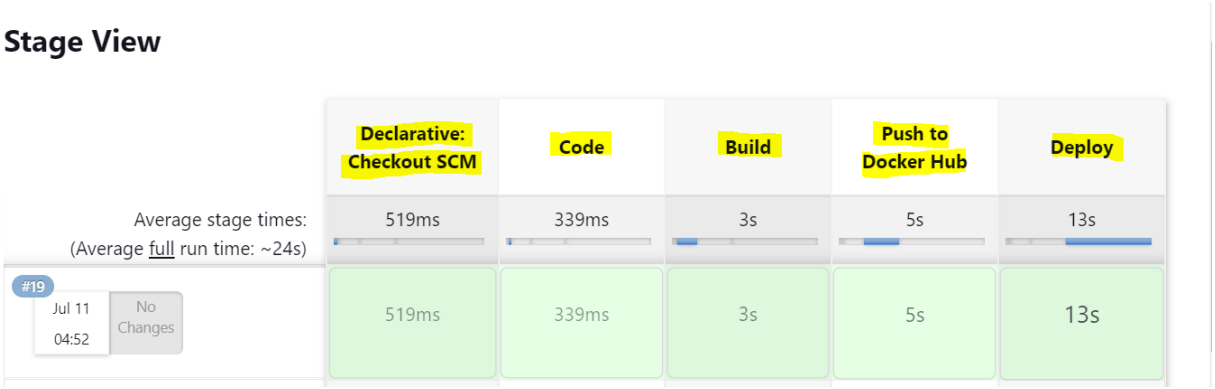
Copy whole pipeline code, go to GitHub> add new file> paste same code as Jenkins file> commit it.

Then:







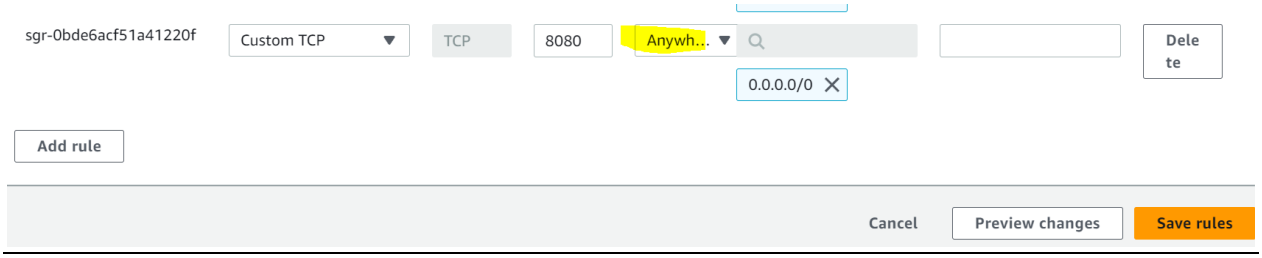


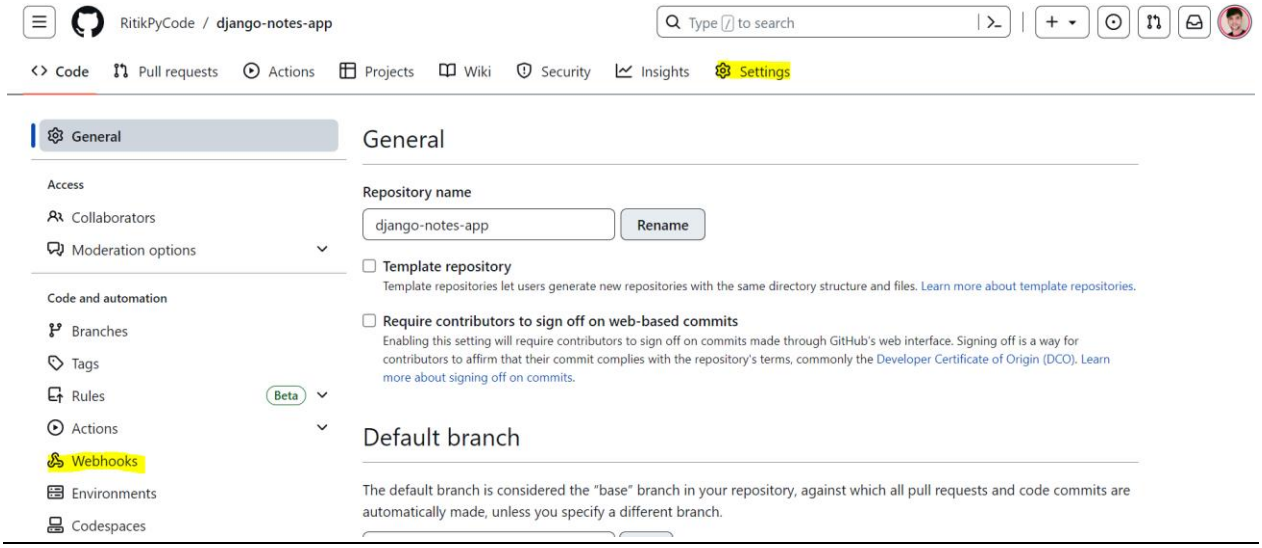
Application has been deployed successfully.

Webhook concept:

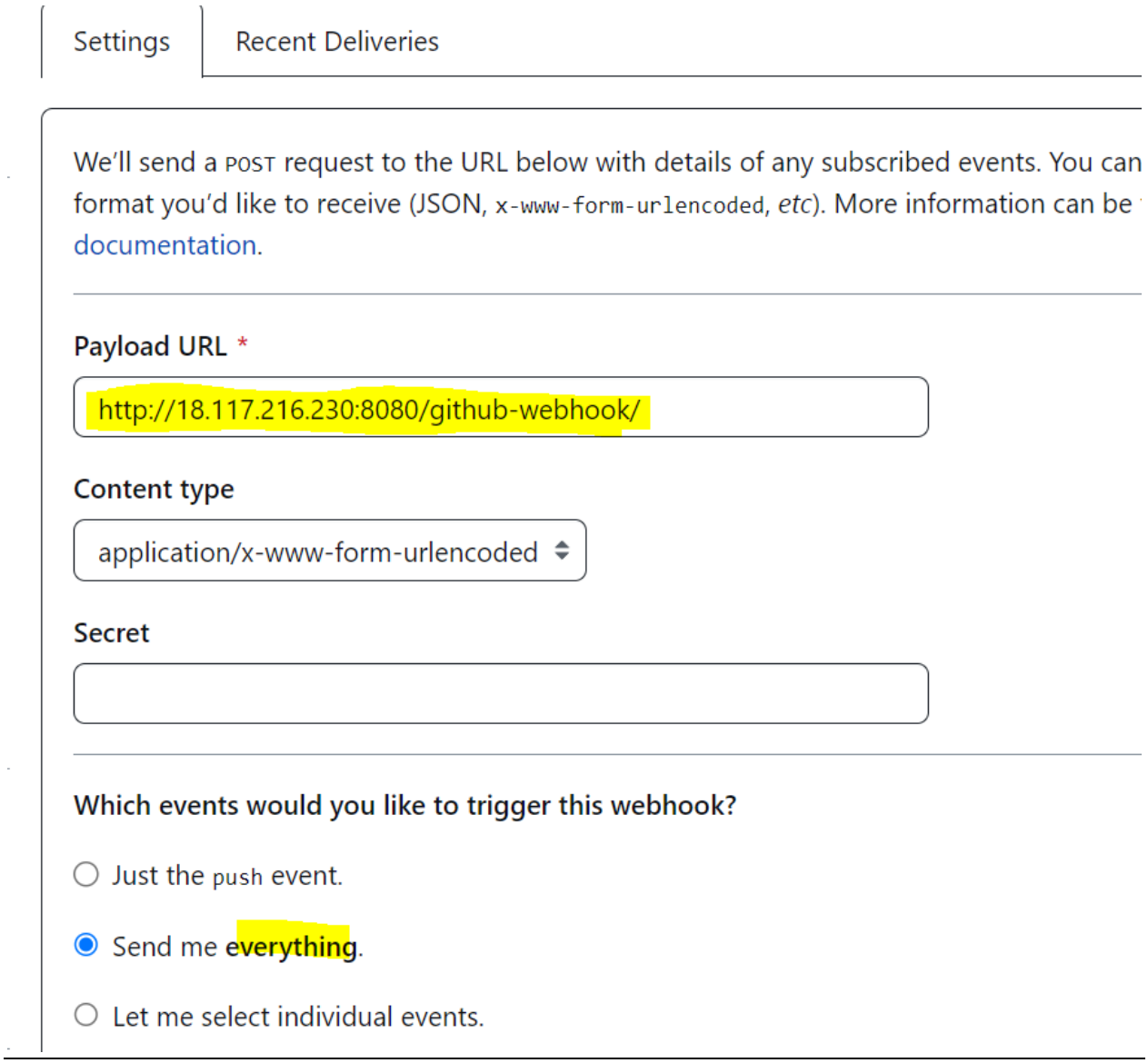
A webhook is a mechanism by which GitHub notifies Jenkins of changes to a repository. Jenkins can then automatically build and test the code in response to the webhook notification.

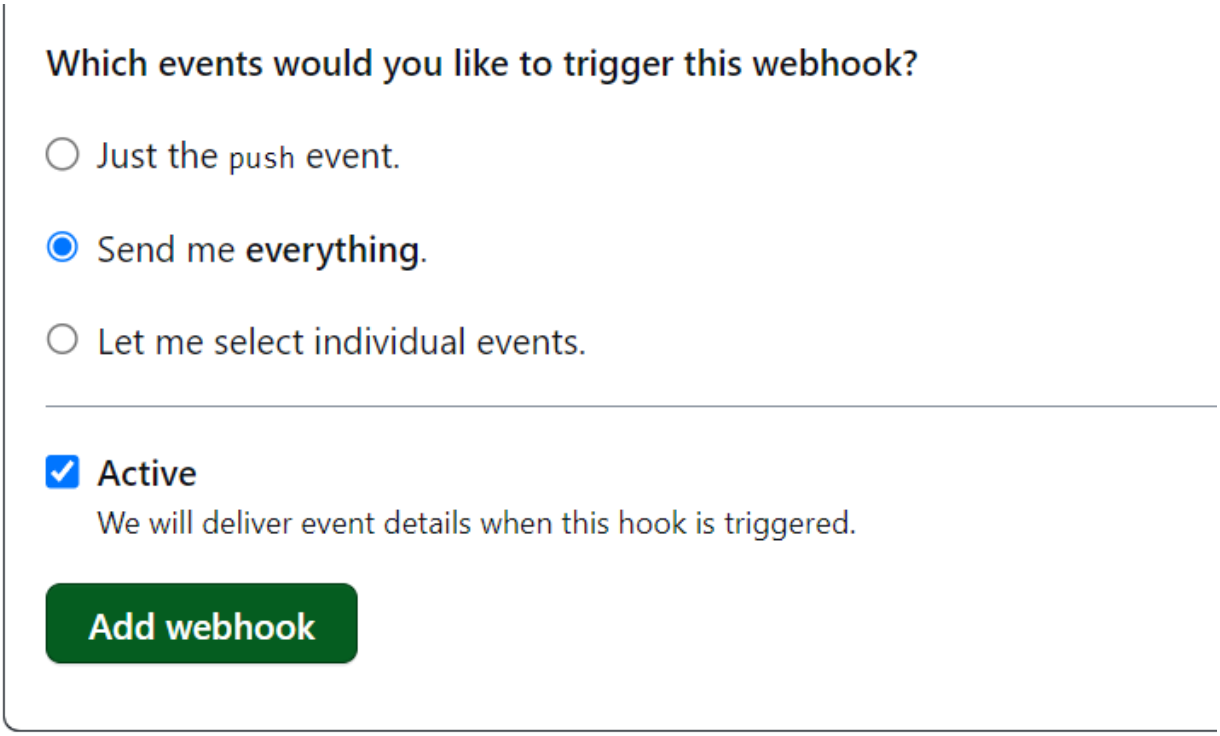
First edit your inbound rule port no. 8080 , My Ip to Anywhere.

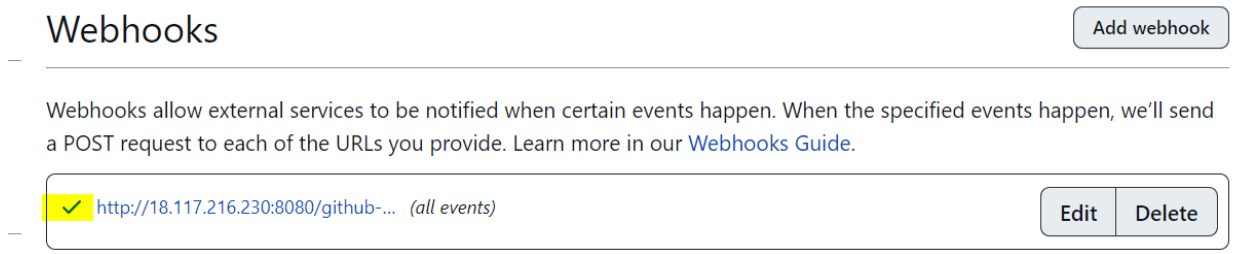




Jenkins URL in Payload URL section:

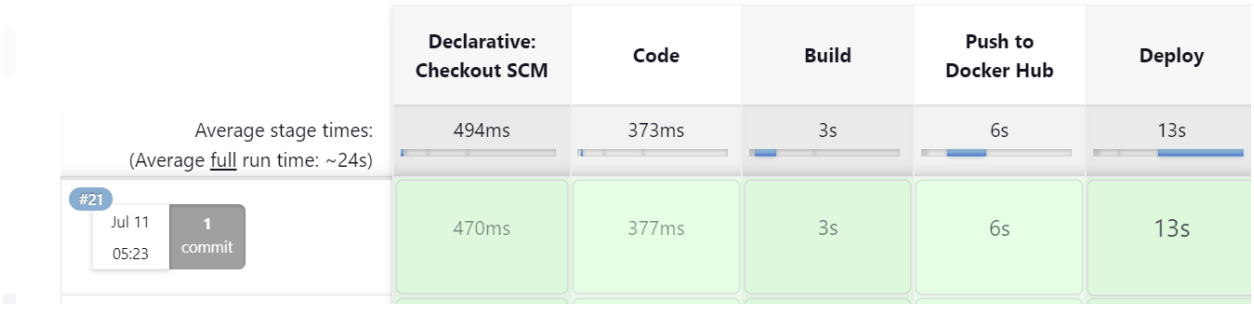






Now webhook is configured. If you change any think in code and commit it… that will be triggered automatically, it's called continuous deployment. With out clicking on build now.

Stage View



Author: Ritik Kumar Tiwari

LinkedIn: https://www.linkedin.com/in/ritikktiwari/