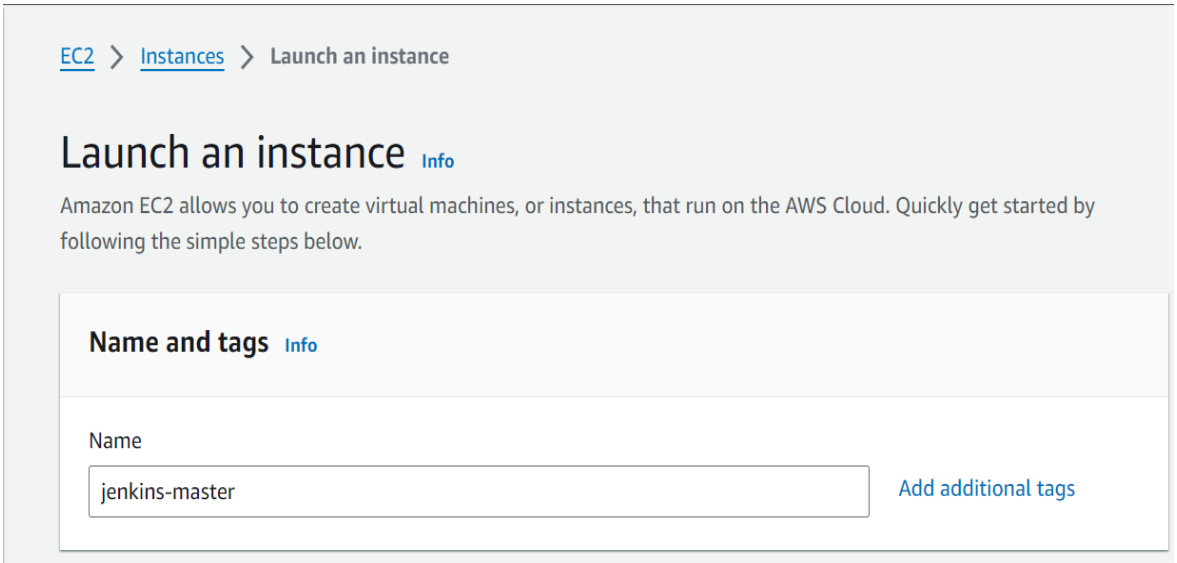
PROJECT -1: CICD USING JENKINS

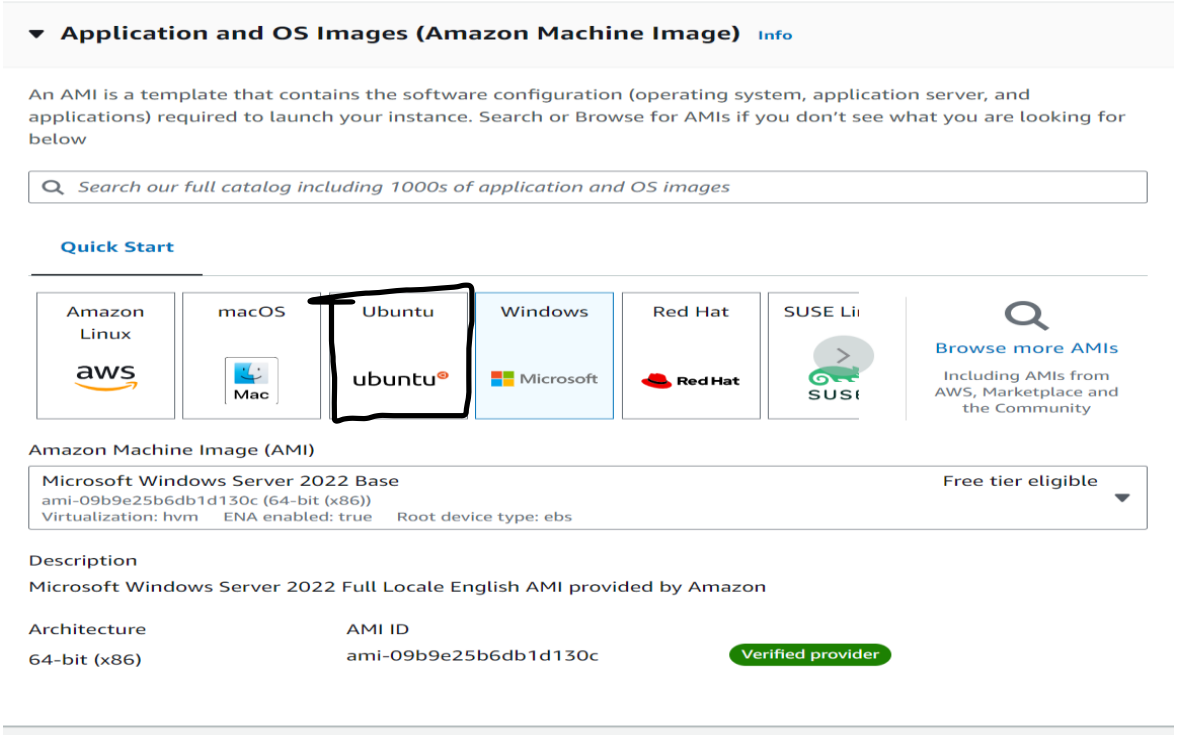
1. Launching AWS EC2 Instance:

- Screenshot of the AWS EC2 console dashboard with the "NAME" and "TAG"



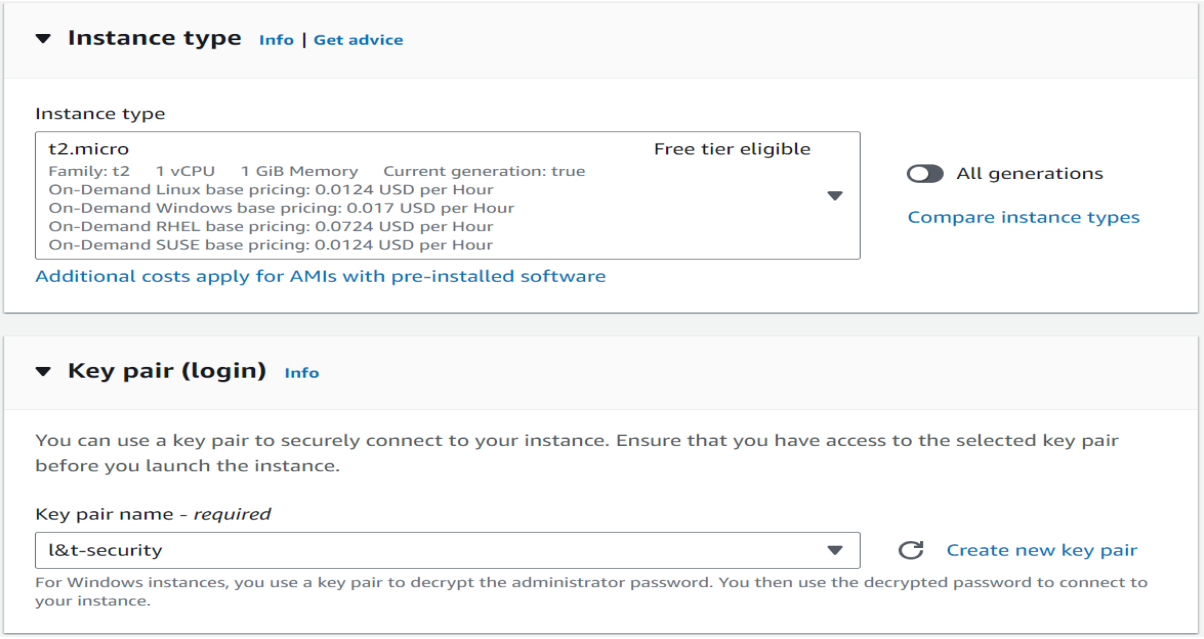
2. Choosing an AMI:

- Screenshot of the "Choose an Amazon Machine Image (AMI)" step. Select the AMI of your choice based on your operating system preferences.

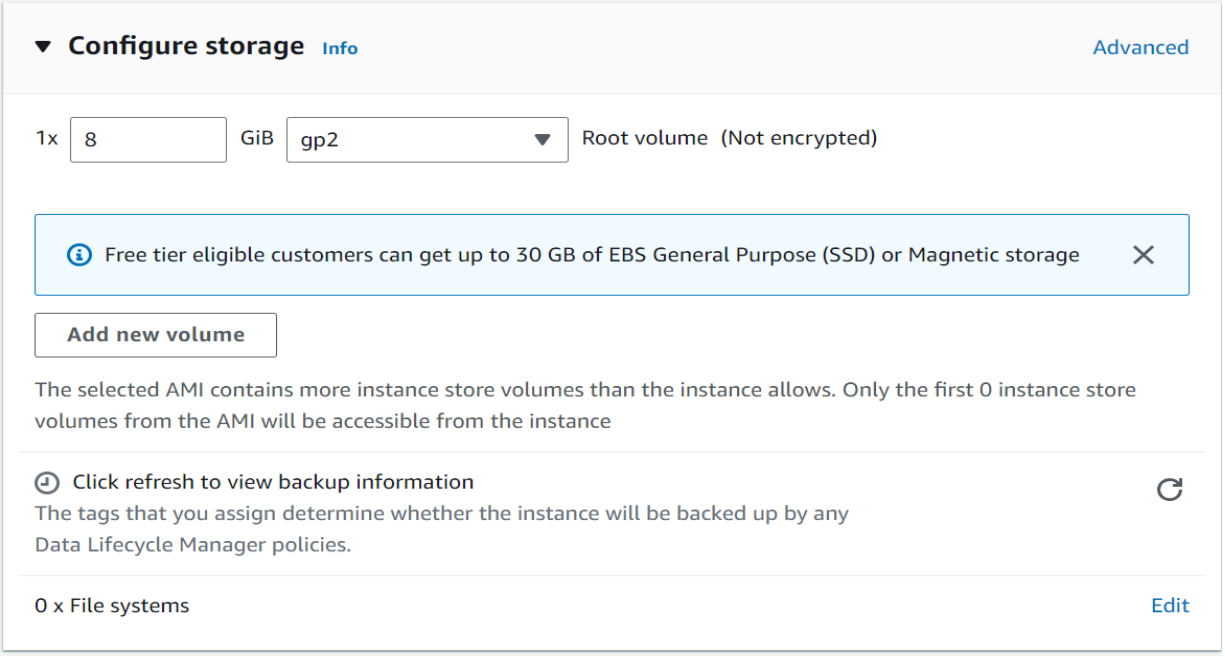


3. Configuring Instance Type:

- Screenshot of the "Choose an Instance Type" step. Select "t2.micro" from the list of available instance types.
- Screenshot of the "Configure Instance Details" step. Click on "Create a new security group" and name it "L&T Security Group".



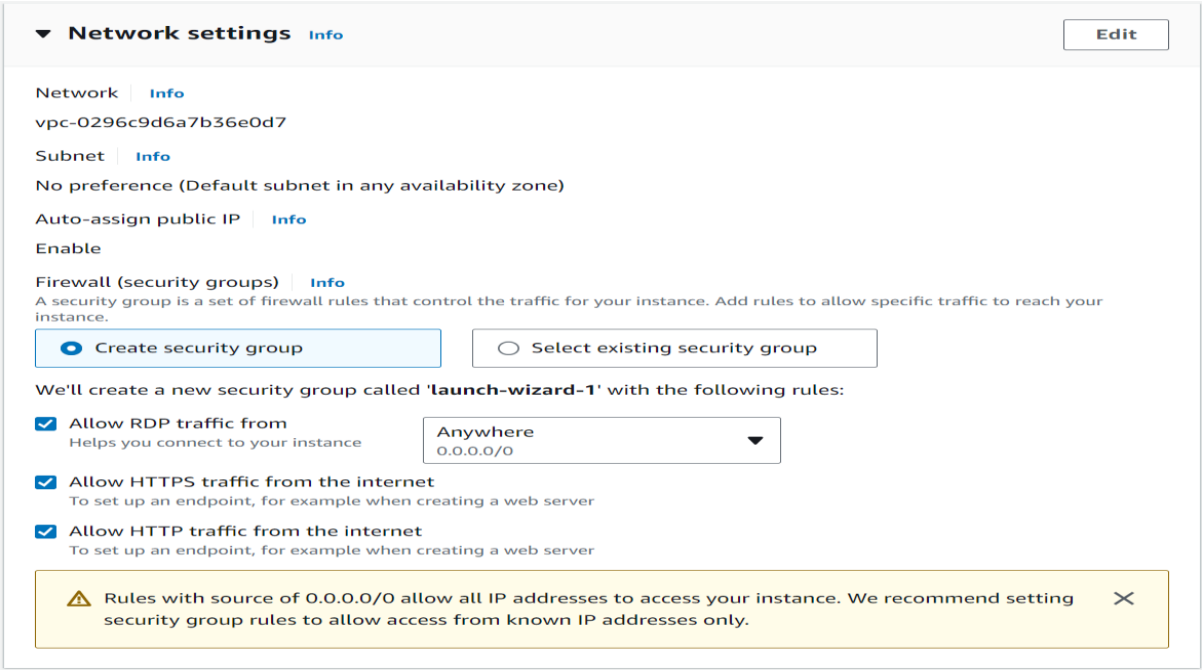
- Screenshot of the EBS volume creation page. Set the size to "8 GB" and choose "gp2" for General Purpose SSD type.



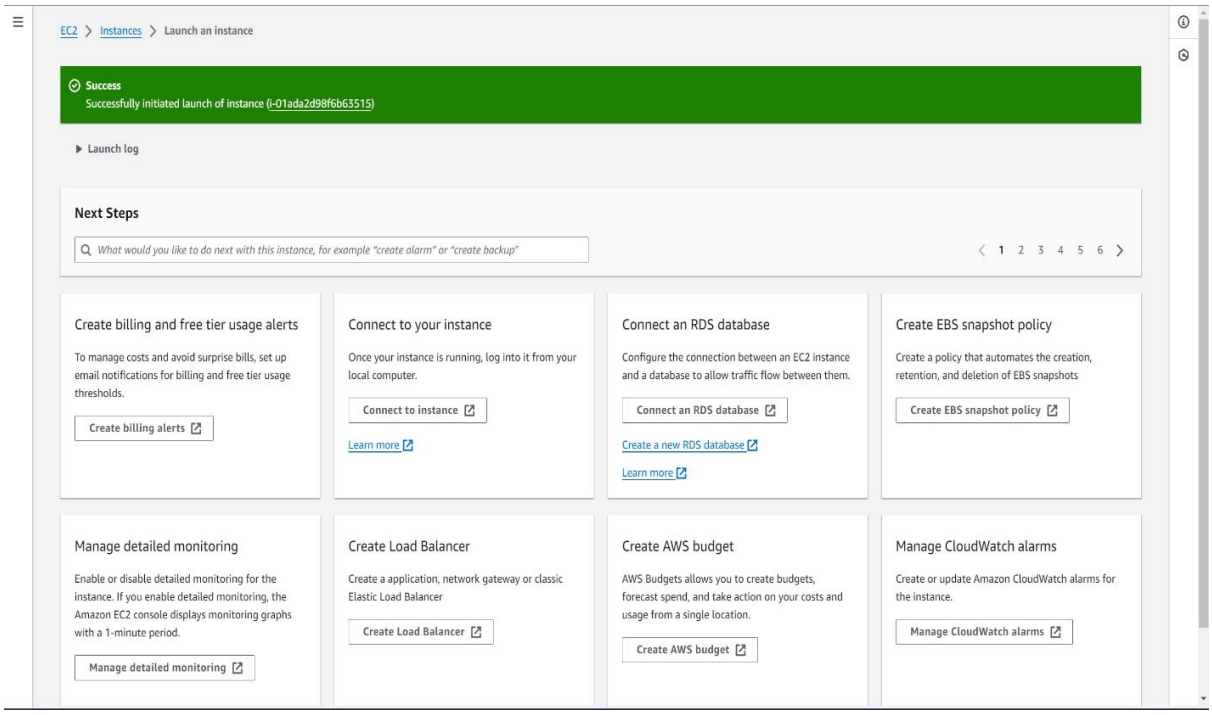
4. Configuring Security Group:

➢ Add two inbound rules:

- One rule for SSH access (TCP port 22) with source set to your IP address or CIDR block for secure access.
- One rule for HTTP access (TCP port 80) with source set to the desired public access (0.0.0.0/0 for everyone or a specific IP range).

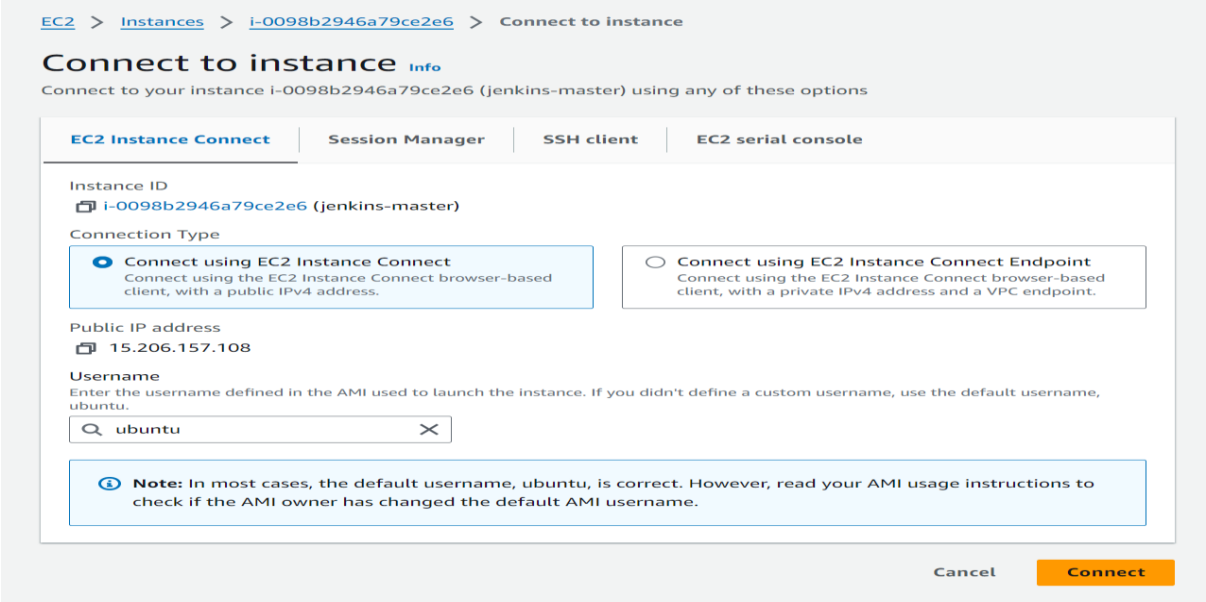


- Screenshot of the "Launch" button and confirmation message. Click on "Launch" to start the instance creation process.



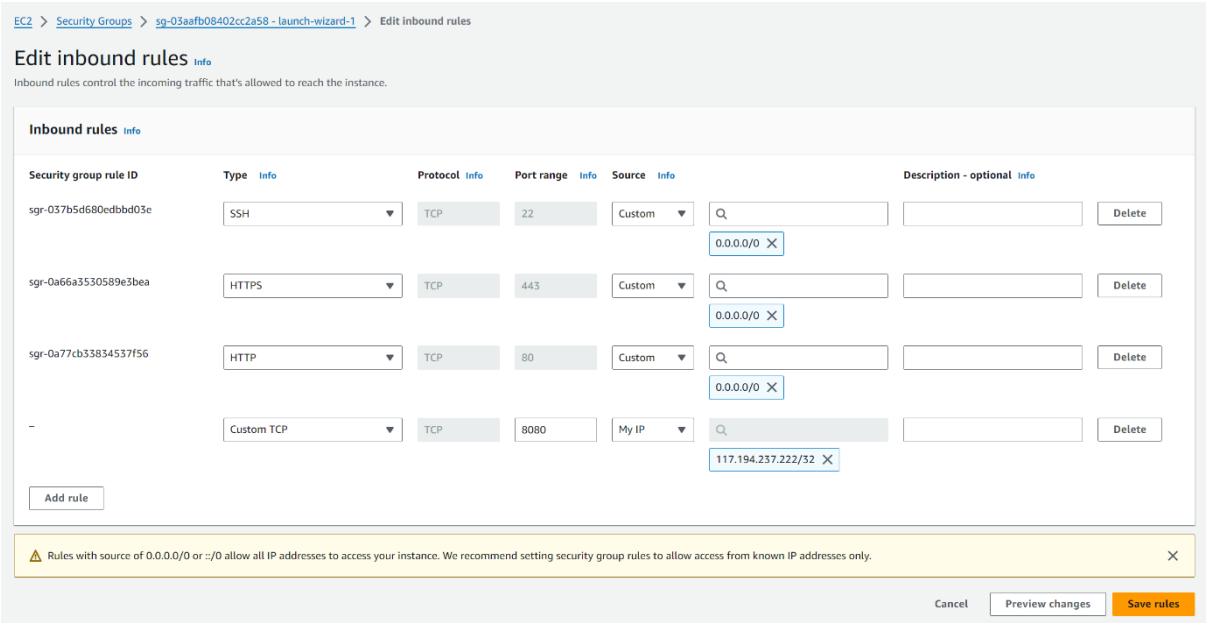
5. Accessing the Instance:

- Screenshot of the EC2 dashboard after the instance is launched. Note the public IP address of the instance
- Screenshot of the EC2 dashboard connecting ubuntu with public IP address



6. Edit Inbound rules:

- Screenshot of the EC2 dashboard Setting Rules on TCP For accessing the EC2 on my IP Address



7. Installation of java and Jenkins:

Step - 1 Install Java

Update your system

```
sudo apt update
```

Install java

```
sudo apt install openjdk-11-jre
```

Validate Installation

```
java -version
```

It should look something like this

```
openjdk version "11.0.12" 2021-07-20 OpenJDK Runtime
Environment (build 11.0.12+7-post-Debian-2) OpenJDK 64-
Bit Server VM (build 11.0.12+7-post-Debian-2, mixed mode,
sharing)
```

Step - 2 Install Jenkins

Just copy these commands and paste them onto your terminal.

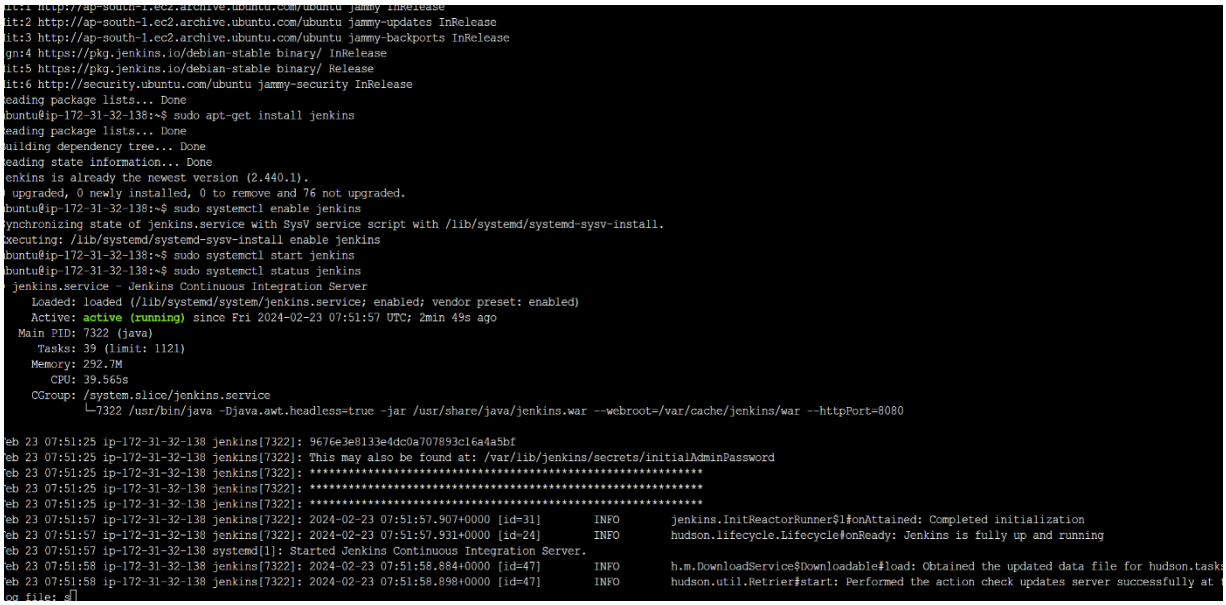
```
curl -fsSL https://pkg.jenkins.io/debian/jenkins.io.key |
sudo tee \   /usr/share/keyrings/jenkins-keyring.asc >
/dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-
keyring.asc] \   https://pkg.jenkins.io/debian binary/ |
sudo tee \   /etc/apt/sources.list.d/jenkins.list >
/dev/null
sudo apt-get update
sudo apt-get install jenkins
```

Step -3 Start jenkins

```
sudo systemctl enable jenkins
sudo systemctl start jenkins
sudo systemctl status Jenkins
```

- Screenshot of Open port 8080 from AWS Console where it asking for Administrator Password for unlocking Jenkins

- Screenshot of Installation and starting Jenkins that shows status of Jenkins is active and running

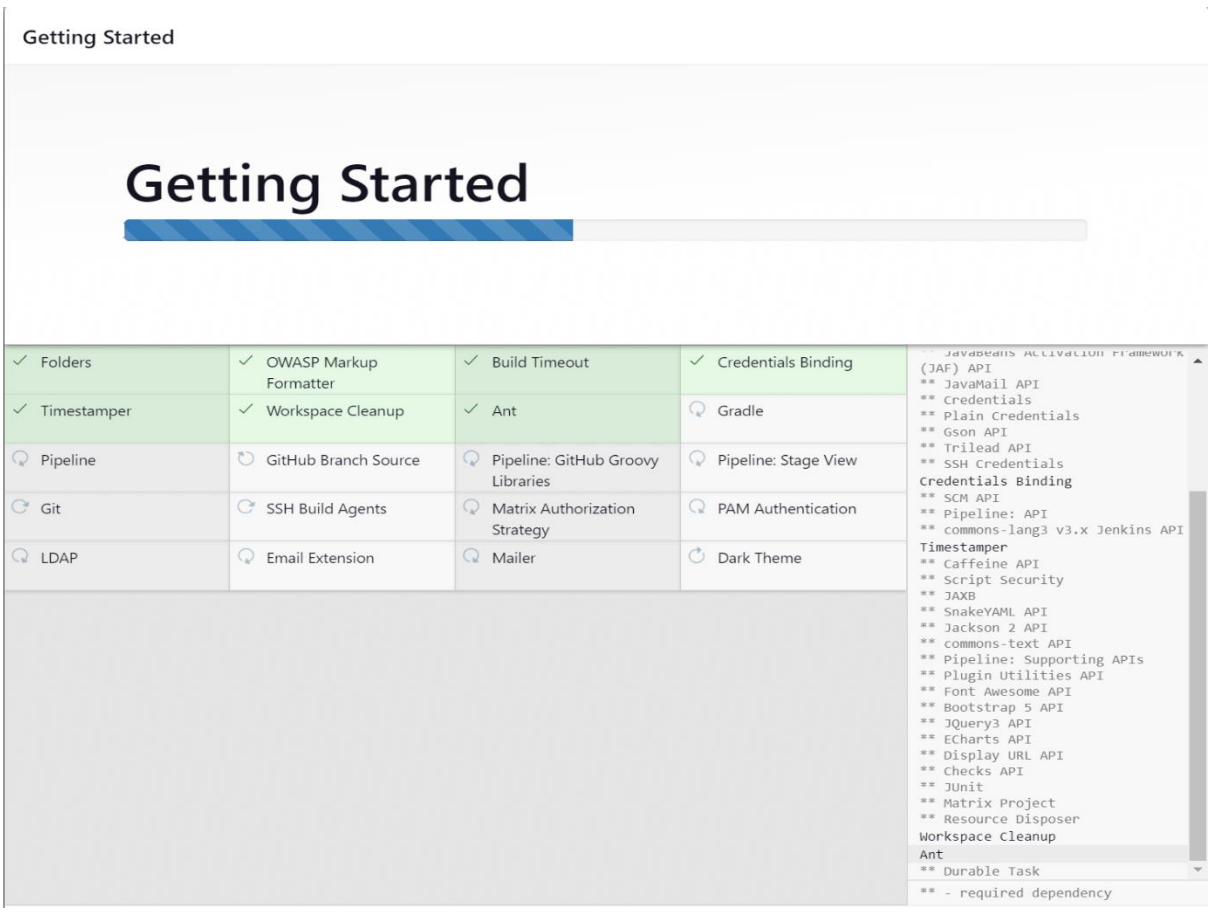


```
it:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
it:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
gn:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
it:5 https://pkg.jenkins.io/debian-stable binary/ Release
it:6 http://security.ubuntu.com/ubuntu jammy-security InRelease
eading package lists... Done
buntu@ip-172-31-32-138:~$ sudo apt-get install jenkins
eading package lists... Done
uilding dependency tree... Done
eading state information... Done
enkins is already the newest version (2.440.1).
 upgraded, 0 newly installed, 0 to remove and 76 not upgraded.
buntu@ip-172-31-32-138:~$ sudo systemctl enable jenkins
ynchronizing state of jenkins.service with SysV service script with /lib/systemd/systemd-sysv-install.
xecuting: /lib/systemd/systemd-sysv-install enable jenkins
buntu@ip-172-31-32-138:~$ sudo systemctl start jenkins
buntu@ip-172-31-32-138:~$ sudo systemctl status jenkins
 jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2024-02-23 07:51:57 UTC; 2min 49s ago
   Main PID: 7322 (java)
      Tasks: 39 (limit: 1121)
     Memory: 292.7M
        CPU: 39.565s
     CGroup: /system.slice/jenkins.service
             └─7322 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

eb 23 07:51:25 ip-172-31-32-138 jenkins[7322]: 9676e3e8133e4dc0a707893c16a4a5bf
eb 23 07:51:25 ip-172-31-32-138 jenkins[7322]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
eb 23 07:51:25 ip-172-31-32-138 jenkins[7322]: *************************************************************
eb 23 07:51:25 ip-172-31-32-138 jenkins[7322]: *************************************************************
eb 23 07:51:25 ip-172-31-32-138 jenkins[7322]: *************************************************************
eb 23 07:51:57 ip-172-31-32-138 jenkins[7322]: 2024-02-23 07:51:57.907+0000 [id=31]	INFO	jenkins.InitReactorRunner$1#onAttained: Completed initialization
eb 23 07:51:57 ip-172-31-32-138 jenkins[7322]: 2024-02-23 07:51:57.931+0000 [id=24]	INFO	hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
eb 23 07:51:57 ip-172-31-32-138 systemd[1]: Started Jenkins Continuous Integration Server.
eb 23 07:51:58 ip-172-31-32-138 jenkins[7322]: 2024-02-23 07:51:58.884+0000 [id=47]	INFO	h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.task
eb 23 07:51:58 ip-172-31-32-138 jenkins[7322]: 2024-02-23 07:51:58.898+0000 [id=47]	INFO	hudson.util.Retrier#start: Performed the action check updates server successfully at
og file: s
```

8. Customize Jenkins Plugins:

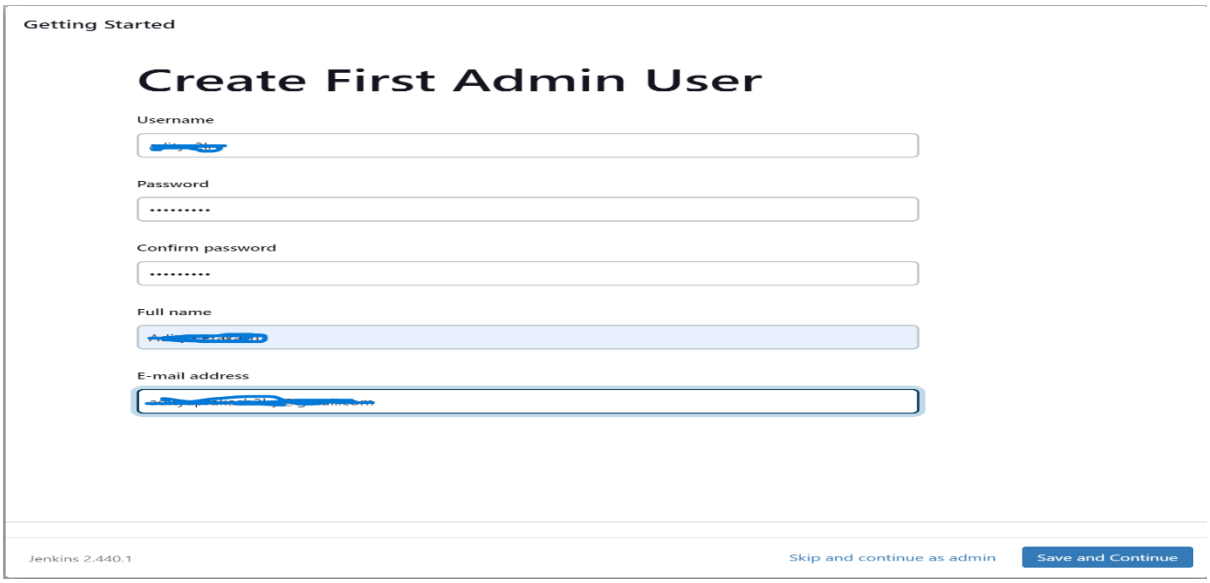
- Screenshot of Installation of suggested plugins extend Jenkins with additional features to support many different needs

- Getting started all the plugins in the customize Jenkins

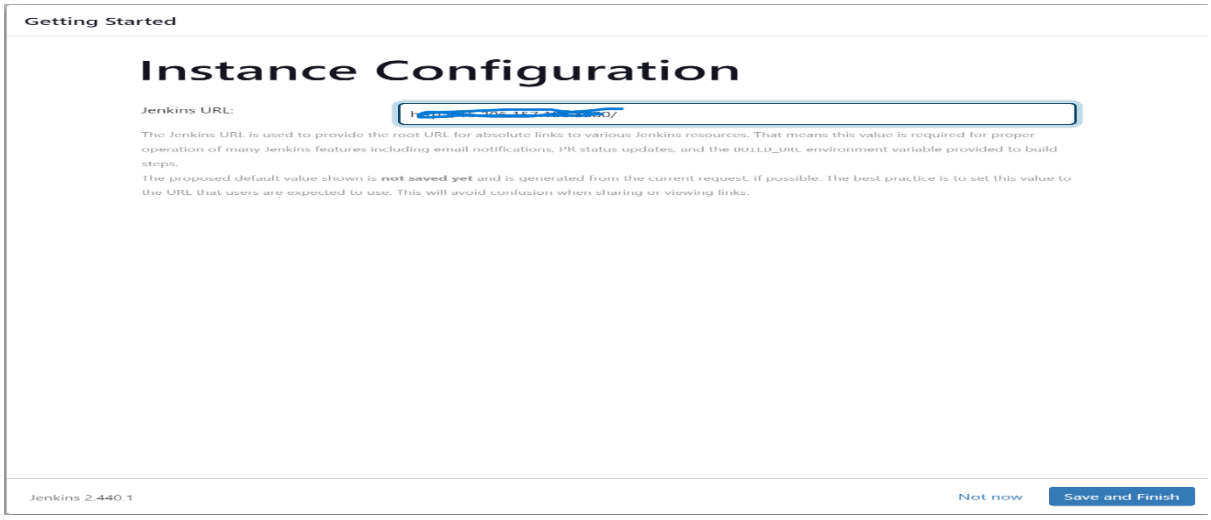


9. Creating First Admin User:

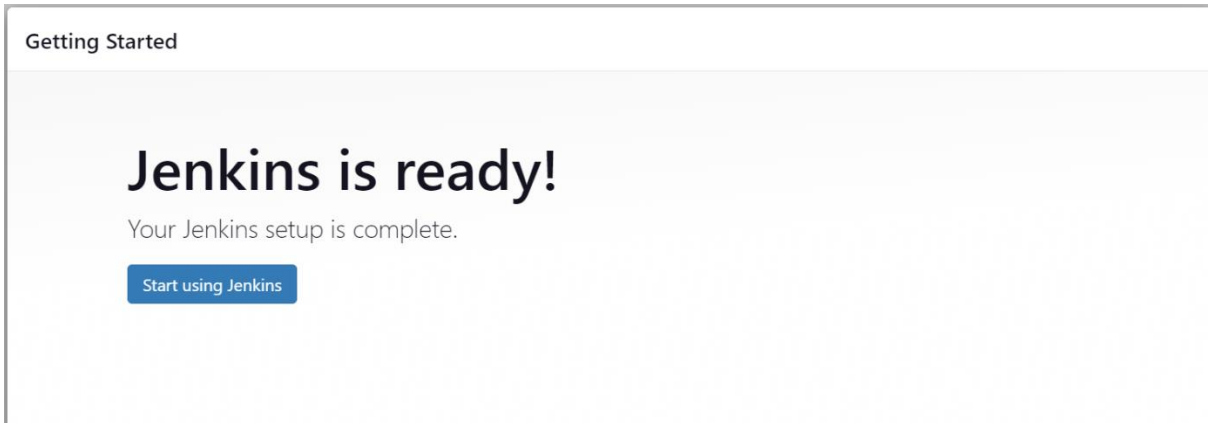
- Screenshot of setting username, password, Full Name, E-mail address



- Screenshot of Instance Configuration setting Jenkins root URL for absolute links to various Jenkins resources

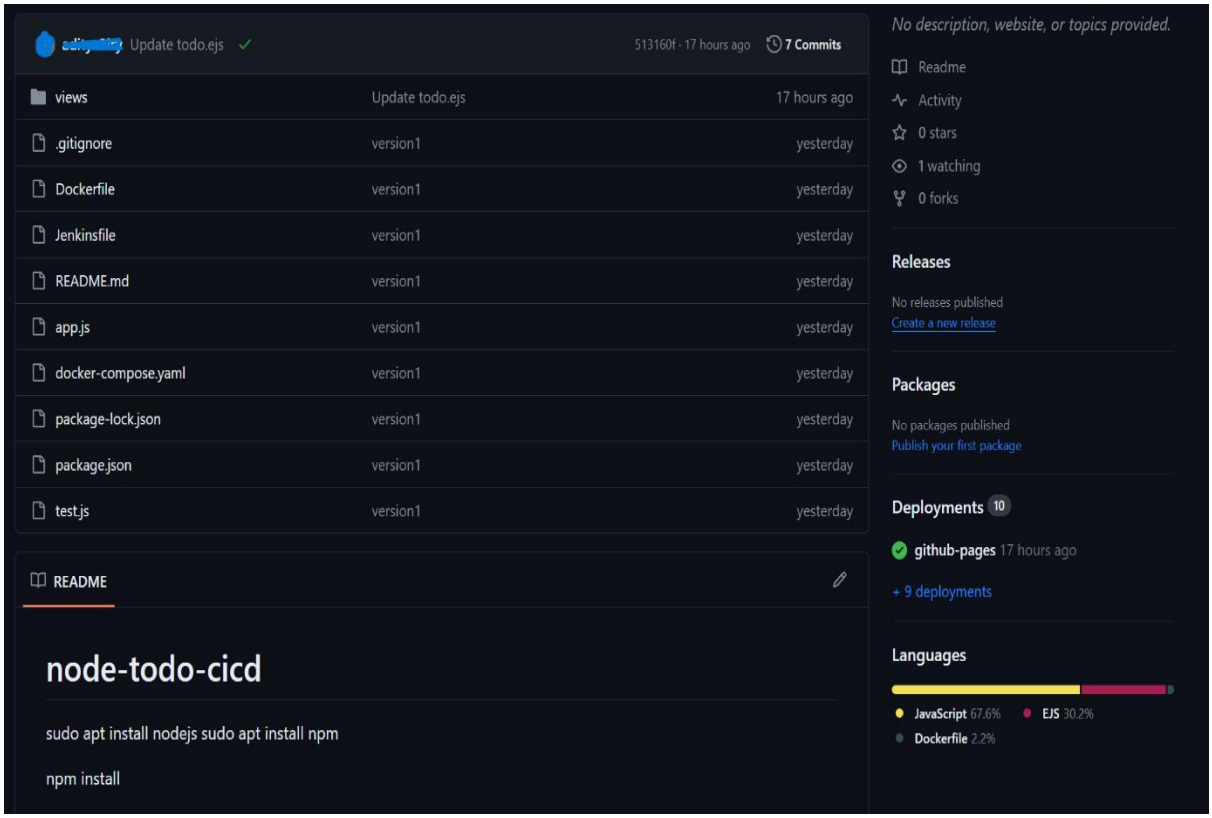


- Screenshot of starting Jenkins

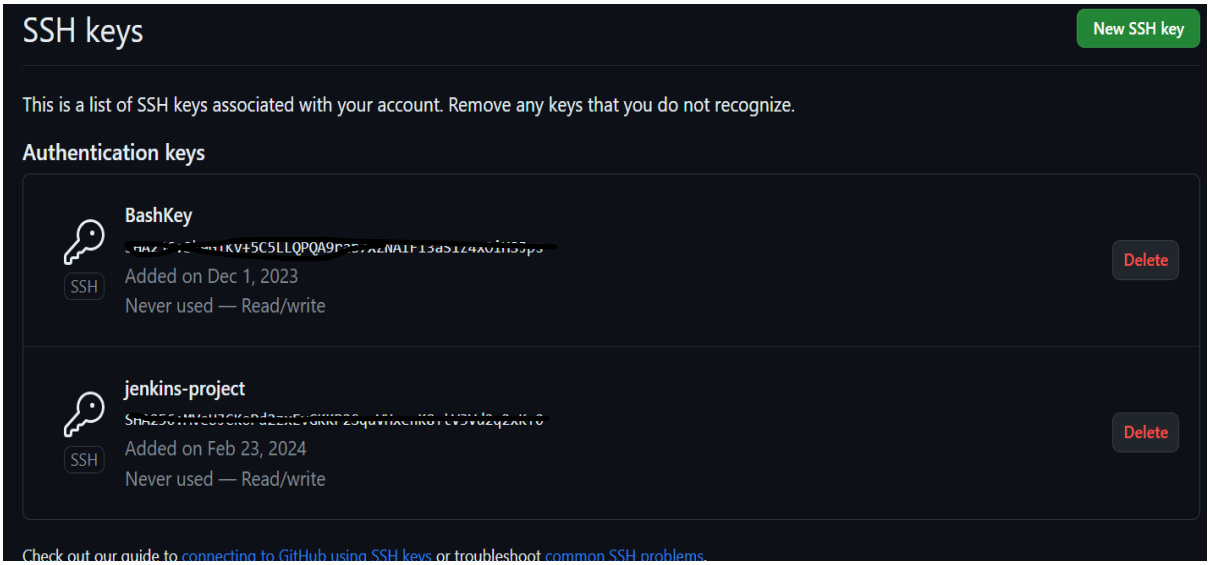


10. Creating GitHub repo:

- Screenshot of Creating a GitHub repository for your project and familiarize yourself with basic Git operations.
- Set up your source code repository. You can use a service like AWS Code Commit, GitHub, or Bitbucket to store your code

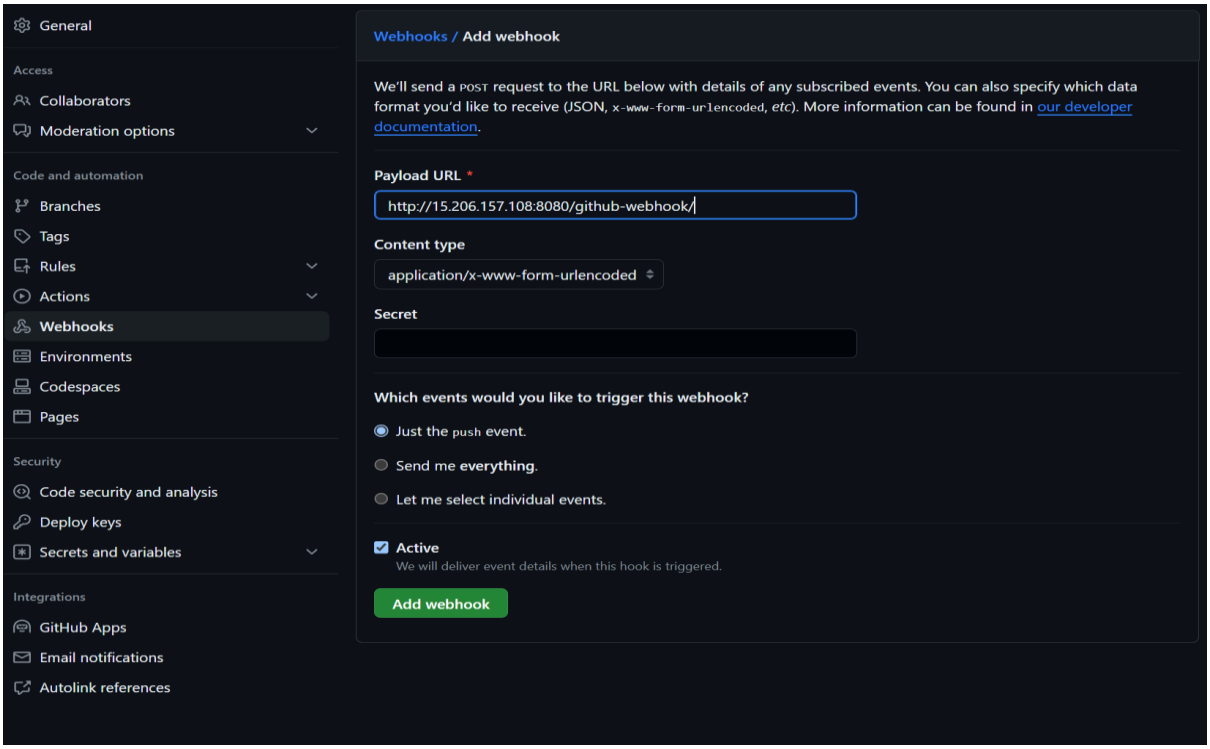


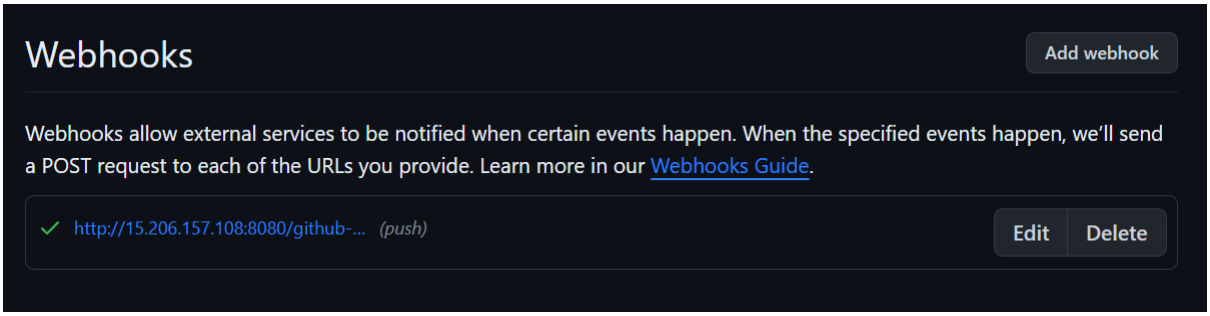
- Connect your source code repository to the pipeline using SSH Key. In the source stage of the pipeline, you can specify the source of your code, such as your Code Commit repository or your GitHub branch.



11.Configure Webhook:

- In your GitHub repository settings, go to "Webhooks" and create a new webhook.
- Set the "Payload URL" to the Jenkins job's URL (e.g., http://your-jenkins-server:8080/job/your-pipeline-name/build).
- Choose the events to trigger the pipeline (e.g., "Push").
- Secret (optional): Create a secret for additional security if needed.
- "Test Delivery" to verify the webhook.



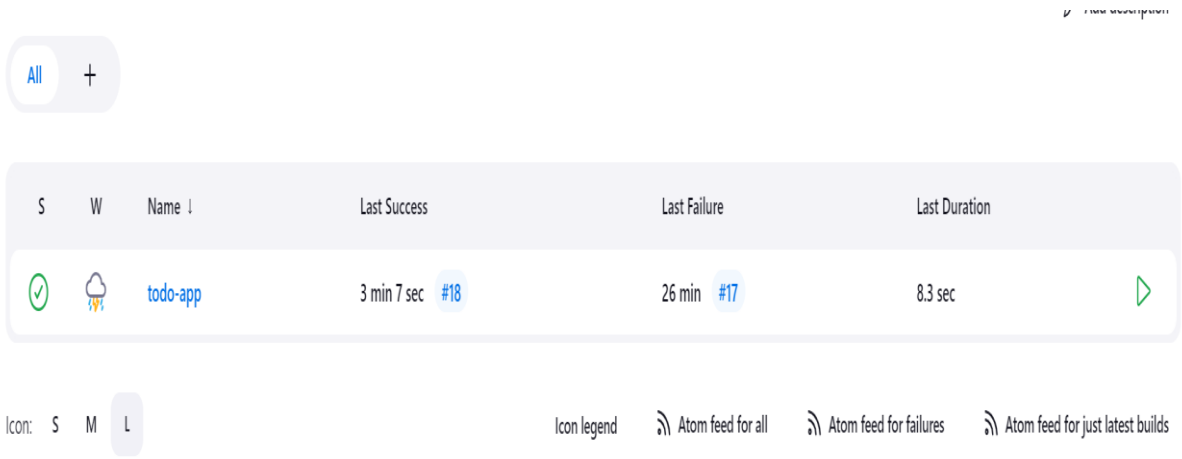


PROJECT -2: CICD USING ELASTIC BEANSTALK

1. **Configure Jenkins:**
   - Create a new user with appropriate permissions in Jenkins.
   - Install the necessary plugins:
     - Git for connecting to GitHub repositories.
     - Pipeline for creating CI/CD pipelines.
     - Docker Pipeline for building and pushing Docker images.

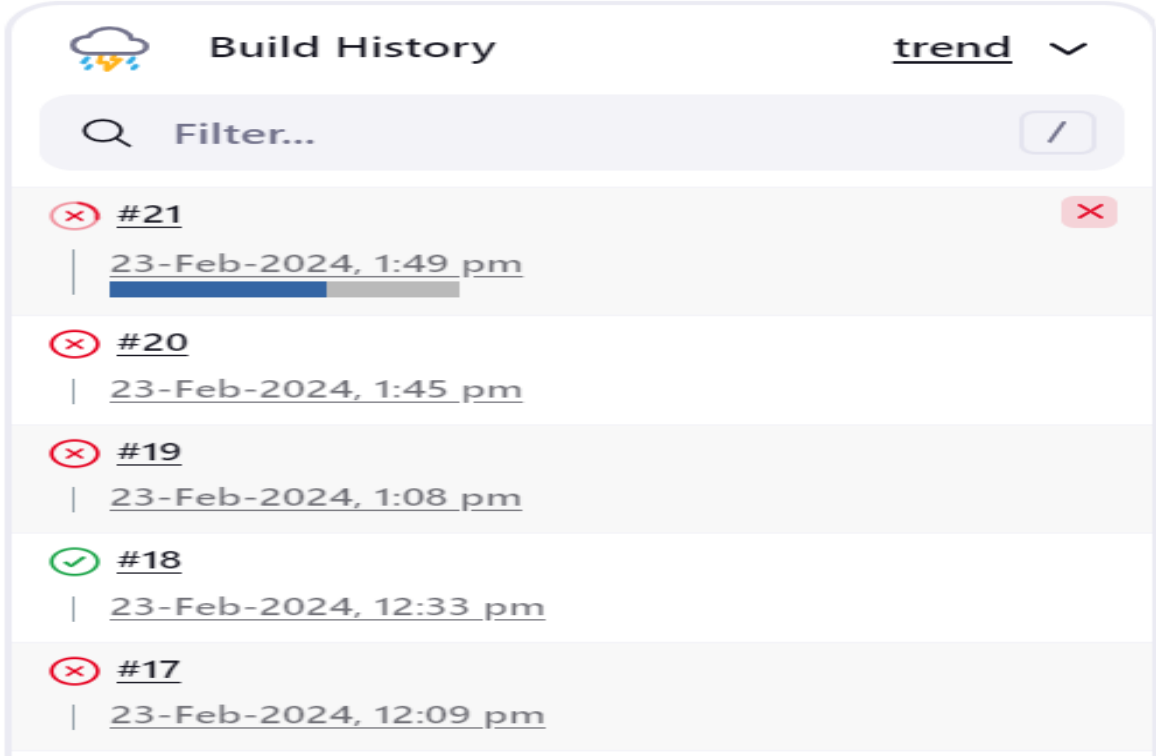
```
ubuntu@ip-172-31-32-138:/var/lib/jenkins/workspace/todo-app$ docker build . -t todo-app
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  25.24MB
Step 1/7 : FROM node:12.2.0-alpine
12.2.0-alpine: Pulling from library/node
e7c96db7181b: Pull complete
a9b145f64bbe: Pull complete
3bcb5e14be53: Pull complete
Digest: sha256:2ab3d9a1bac67c9b4202b774664adaa94d2f1e426d8d28e07bf8979df61c8694
Status: Downloaded newer image for node:12.2.0-alpine
 ---> f391dabf9dce
Step 2/7 : WORKDIR app
 ---> Running in dbc28d5656ac
Removing intermediate container dbc28d5656ac
 ---> 4290ab17b595
Step 3/7 : COPY . .
 ---> 19992a1b364e
Step 4/7 : RUN npm install
 ---> Running in 0f4e1ca339f4
```

2. **Create a Jenkins Pipeline:**
   - Go to "New Item" in Jenkins.
   - Choose "Pipeline" and give it a meaningful name.
   - Select "Pipeline script from SCM" and provide the GitHub repository URL and branch containing your Jenkins file (or paste the script directly).
   - Save the job.



3. **Write the Jenkinsfile:**
   - Use the declarative syntax for a clean and maintainable pipeline.
   - Here's a basic example that pulls code from GitHub, builds a Docker image, and pushes it to Docker Hub:

1. Define a build spec for your application. In the build stage of the pipeline, you can define a build spec that tells CodeBuild how to build your application. This could involve running unit tests, packaging your application into a deployable artifact, and generating configuration files.



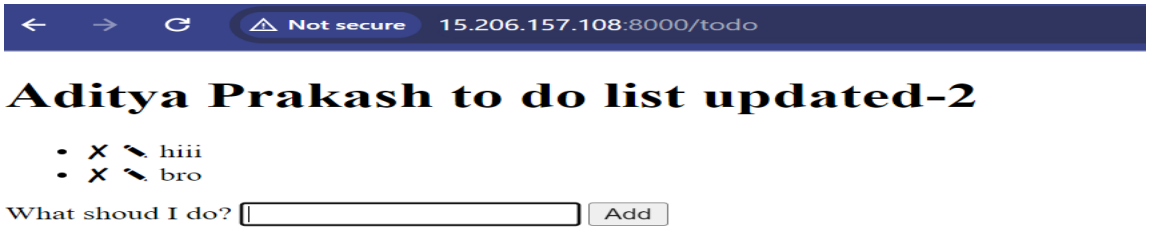
- Configure deployment to Elastic Beanstalk. In the deployment stage of the pipeline, you can specify the Elastic Beanstalk application and environment that you want to deploy your application to. You can also configure Code Pipeline to trigger deployments automatically when changes are made to your code.

Run the Pipeline:

Push changes to your GitHub repository.
The webhook will trigger the Jenkins pipeline automatically.
Monitor the pipeline's progress in the Jenkins UI.

Console Output

```
Started by user Aditya Prakash
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/todo-app
The recommended git tool is: NONE
using credential github-jenkins
 > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/todo-app/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/aditya2lry/To-Do-List.git # timeout=10
Fetching upstream changes from https://github.com/aditya2lry/To-Do-List.git
 > git --version # timeout=10
 > git --version # 'git version 2.34.1'
using GIT_SSH to set credentials this is for github and jenkins integration
Verifying host key using known hosts file
You're using 'Known hosts file' strategy to verify ssh host keys, but your known_hosts file does not exist, please go to 'Manage Jenkins' -> 'Security' -> 'Git Host Key
Verification Configuration' and configure host key verification.
 > git fetch --tags --force --progress -- https://github.com/aditya2lry/To-Do-List.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 4550629f9cbb558b9939c362e955eb3b97d41ecd (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 4550629f9cbb558b9939c362e955eb3b97d41ecd # timeout=10
Commit message: "Update todo.ejs"
 > git rev-list --no-walk 4550629f9cbb558b9939c362e955eb3b97d41ecd # timeout=10
[todo-app] $ /bin/sh -xe /tmp/jenkins119312255845221 3401.sh
+ docker build . -t todo-app
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  25.25MB

Step 1/7 : FROM node:12.2.0-alpine
 ---> f391dabf9dce
```



ALL COMANDS HISTORY:-

1. sudo apt update
2. sudo apt install openjdk-11-jre
3. java -version
4. curl -fsSL https://pkg.jenkins.io/debian/jenkins.io.key | sudo tee
   \ /usr/share/keyrings/jenkins-keyring.asc > /dev/null
5. echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]
   \ https://pkg.jenkins.io/debian binary/ | sudo tee
   \ /etc/apt/sources.list.d/jenkins.list > /dev/null
6. sudo apt-get update
7. sudo apt-get install jenkins
8. sudo systemctl enable jenkins
9. sudo systemctl start jenkins
10. sudo systemctl status jenkins
11. sudo cat /var/lib/jenkins/secrets/initialAdminPassword
12. sudo apt install docker.io
13. FROM node:12.2.0-alpine
14. WORKDIR app
15. COPY . .
16. RUN npm install
17. EXPOSE 8000
18. CMD ["node","app.js"]
19. docker build . -t node-app
20. sudo usermod -a -G docker $USER
21. docker run -d --name node-todo-app -p 8000:8000 todo-node-app
22. Got to jenkins job
23. Execute shell
24. docker build . -t node-app-todo
25. docker run -d --name node-app-container -p 8000:8000 node-app-todo