Setup : AWS EC2 Instance, Jenkins, Docker Application

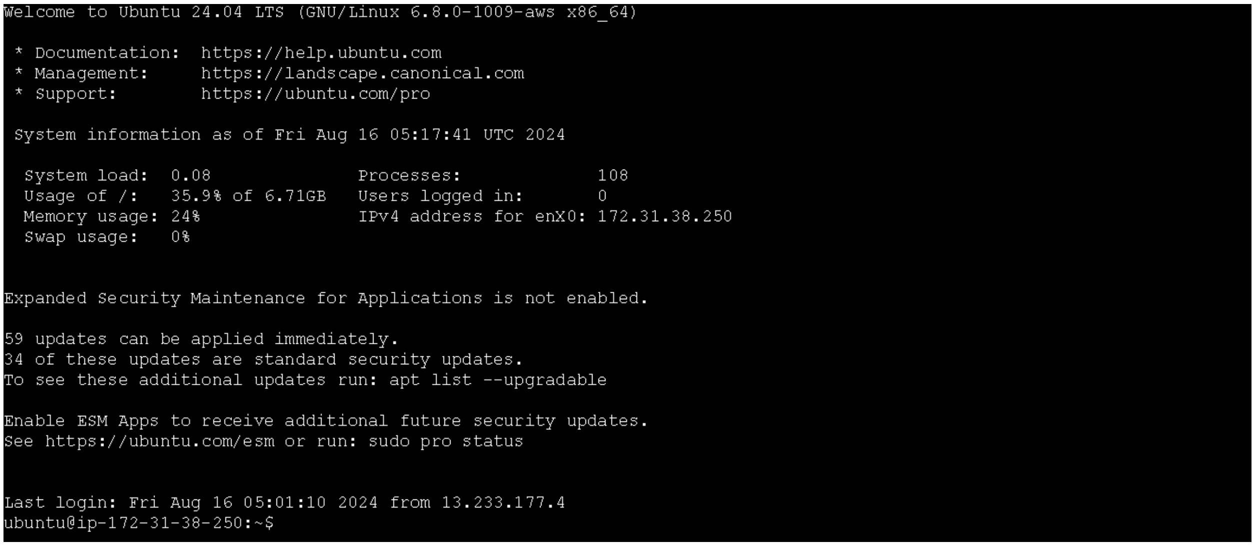
Step 1: Create AWS EC2 Instance

Launch an EC2 Instance:

- Choose the desired Amazon Machine Image (AMI) (e.g., Ubuntu 20.04 LTS).
- Select instance type (e.g., t2.micro).
- Configure instance details (e.g., number of instances, network settings).
- Add storage (default 8 GB).
- Add tags (optional).
- Configure security group to allow SSH (port 22) and HTTP (port 80) access.
- Review and launch the instance.

Connect to EC2 Instance:

- Use SSH to connect to the instance.
- ssh -i your-key.pem ubuntu@your-ec2-public-ip



```
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1009-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Fri Aug 16 05:17:41 UTC 2024

  System load:  0.08               Processes:             108
  Usage of /:   35.9% of 6.71GB    Users logged in:       0
  Memory usage: 24%                IPv4 address for enX0: 172.31.38.250
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

59 updates can be applied immediately.
34 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Fri Aug 16 05:01:10 2024 from 13.233.177.4
ubuntu@ip-172-31-38-250:~$
```

Step 2: Install Java

Update your system:

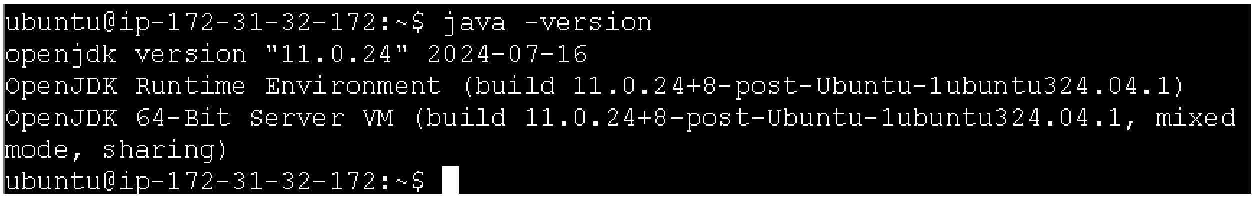
- sudo apt update

Install OpenJDK 11:

- sudo apt install openjdk-11-jre

Verify Java installation:

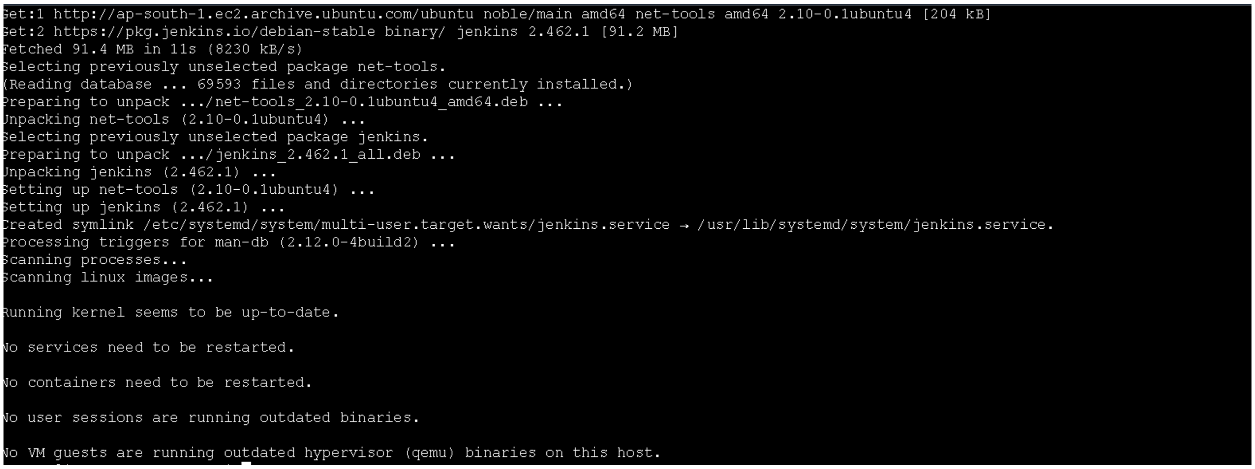
- java -version



```
ubuntu@ip-172-31-32-172:~$ java -version
openjdk version "11.0.24" 2024-07-16
OpenJDK Runtime Environment (build 11.0.24+8-post-Ubuntu-1ubuntu324.04.1)
OpenJDK 64-Bit Server VM (build 11.0.24+8-post-Ubuntu-1ubuntu324.04.1, mixed
mode, sharing)
ubuntu@ip-172-31-32-172:~$
```

Step 3: install Jenkins

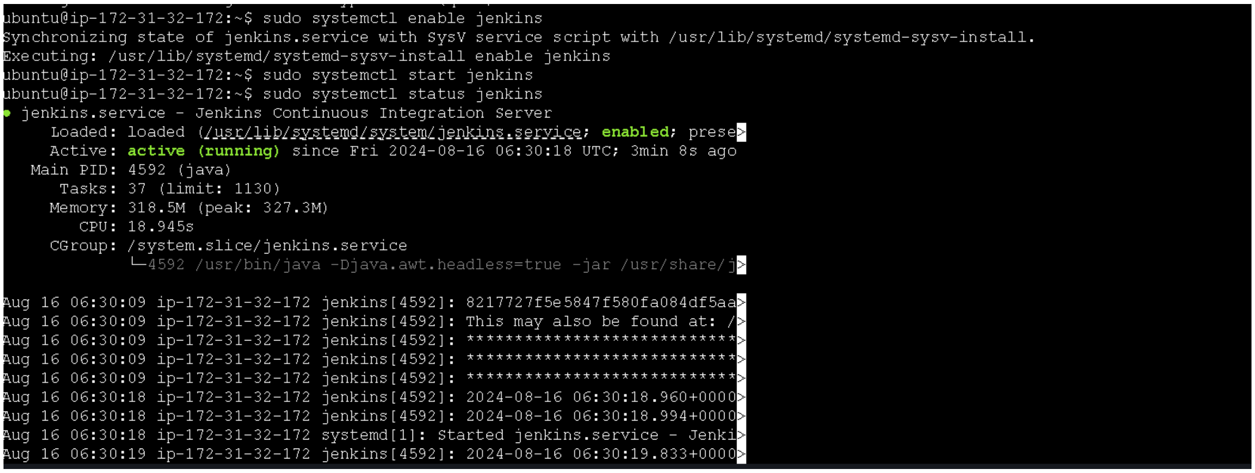
- sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
  echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list **>** /dev/null
- sudo apt-get update
- sudo apt-get install jenkins



```
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 net-tools amd64 2.10-0.1ubuntu4 [204 kB]
Get:2 https://pkg.jenkins.io/debian-stable binary/ jenkins 2.462.1 [91.2 MB]
Fetched 91.4 MB in 11s (8230 kB/s)
Selecting previously unselected package net-tools.
(Reading database ... 69593 files and directories currently installed.)
Preparing to unpack .../net-tools_2.10-0.1ubuntu4_amd64.deb ...
Unpacking net-tools (2.10-0.1ubuntu4) ...
Selecting previously unselected package jenkins.
Preparing to unpack .../jenkins_2.462.1_all.deb ...
Unpacking jenkins (2.462.1) ...
Setting up net-tools (2.10-0.1ubuntu4) ...
Setting up jenkins (2.462.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /usr/lib/systemd/system/jenkins.service.
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

Step 4: start Jenkins

- `sudo systemctl enable jenkins`
- `sudo systemctl start jenkins`
- `sudo systemctl status jenkins`



```
ubuntu@ip-172-31-32-172:~$ sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
ubuntu@ip-172-31-32-172:~$ sudo systemctl start jenkins
ubuntu@ip-172-31-32-172:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; prese>
     Active: active (running) since Fri 2024-08-16 06:30:18 UTC; 3min 8s ago
   Main PID: 4592 (java)
      Tasks: 37 (limit: 1130)
     Memory: 318.5M (peak: 327.3M)
        CPU: 18.945s
     CGroup: /system.slice/jenkins.service
             └─4592 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/j>

Aug 16 06:30:09 ip-172-31-32-172 jenkins[4592]: 8217727f5e5847f580fa084df5aa>
Aug 16 06:30:09 ip-172-31-32-172 jenkins[4592]: This may also be found at: />
Aug 16 06:30:09 ip-172-31-32-172 jenkins[4592]: ****************************>
Aug 16 06:30:09 ip-172-31-32-172 jenkins[4592]: ****************************>
Aug 16 06:30:09 ip-172-31-32-172 jenkins[4592]: ****************************>
Aug 16 06:30:18 ip-172-31-32-172 jenkins[4592]: 2024-08-16 06:30:18.960+0000>
Aug 16 06:30:18 ip-172-31-32-172 jenkins[4592]: 2024-08-16 06:30:18.994+0000>
Aug 16 06:30:18 ip-172-31-32-172 systemd[1]: Started jenkins.service - Jenki>
Aug 16 06:30:19 ip-172-31-32-172 jenkins[4592]: 2024-08-16 06:30:19.833+0000>
```

https://www.jenkins.io/doc/book/installing/linux/ - this is the guide i'm following.

Step 5: Install Docker

Install Docker:

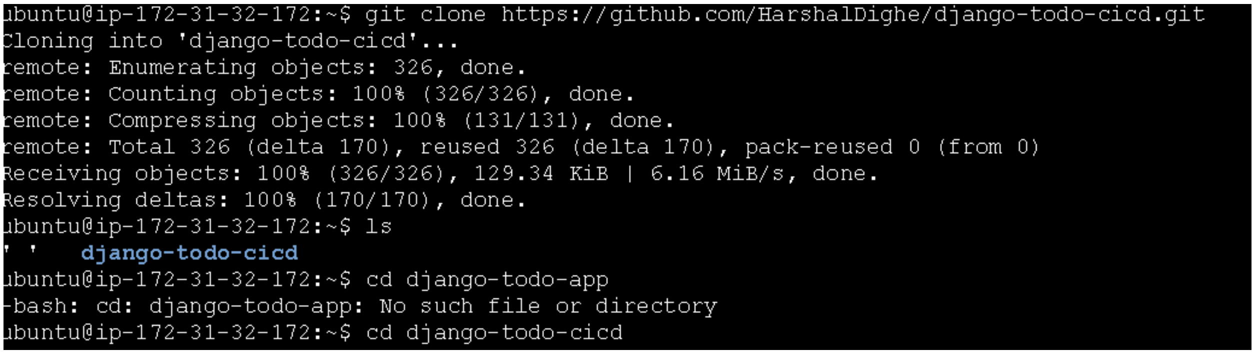
- sudo apt install docker.io

Add the current user to the Docker group:

- sudo chown $USER /var/run/docker.sock

Step 6: clone your repo

- git clone https://github.com/HarshalDighe/django-todo-cicd.git
- cd <project>



```
ubuntu@ip-172-31-32-172:~$ git clone https://github.com/HarshalDighe/django-todo-cicd.git
Cloning into 'django-todo-cicd'...
remote: Enumerating objects: 326, done.
remote: Counting objects: 100% (326/326), done.
remote: Compressing objects: 100% (131/131), done.
remote: Total 326 (delta 170), reused 326 (delta 170), pack-reused 0 (from 0)
Receiving objects: 100% (326/326), 129.34 KiB | 6.16 MiB/s, done.
Resolving deltas: 100% (170/170), done.
ubuntu@ip-172-31-32-172:~$ ls
' '   django-todo-cicd
ubuntu@ip-172-31-32-172:~$ cd django-todo-app
-bash: cd: django-todo-app: No such file or directory
ubuntu@ip-172-31-32-172:~$ cd django-todo-cicd
```

Step 7: Setup Node.js Application with Docker

Create a Dockerfile:

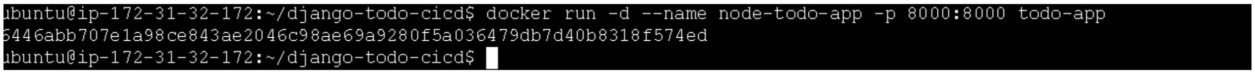
- FROM python:3
- RUN pip install Django==3.2
- COPY . .
- EXPOSE 8000
- CMD ["python","manage.py","runserver","0.0.0.0:8000"]

Build Docker image:

- docker build . -t node-app

Run the Docker container:

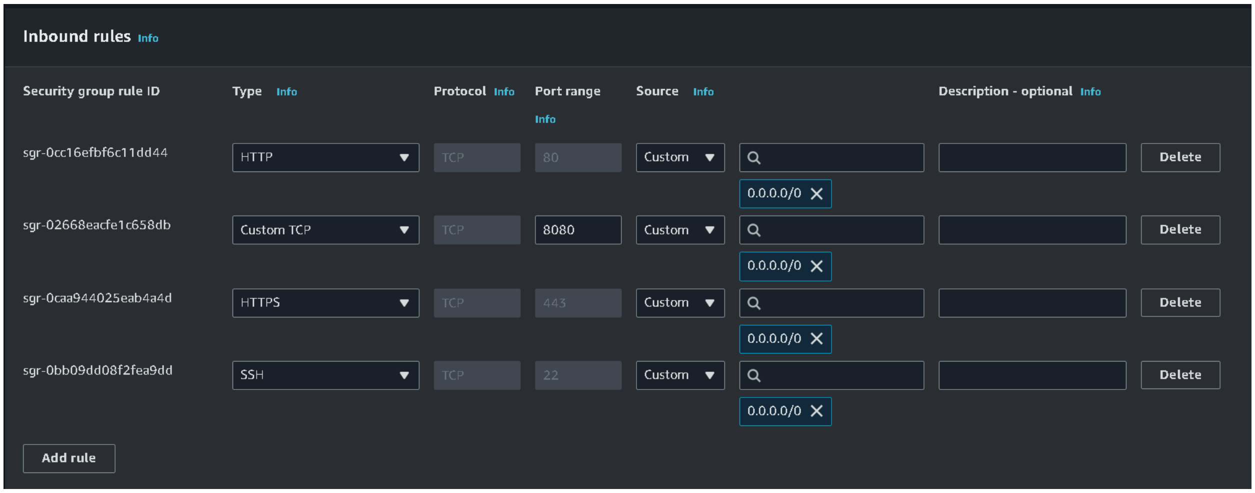
- docker run -d --name node-todo-app -p 8000:8000 node-app



```
ubuntu@ip-172-31-32-172:~/django-todo-cicd$ docker run -d --name node-todo-app -p 8000:8000 todo-app
5446abb707e1a98ce843ae2046c98ae69a9280f5a036479db7d40b8318f574ed
ubuntu@ip-172-31-32-172:~/django-todo-cicd$ 
```

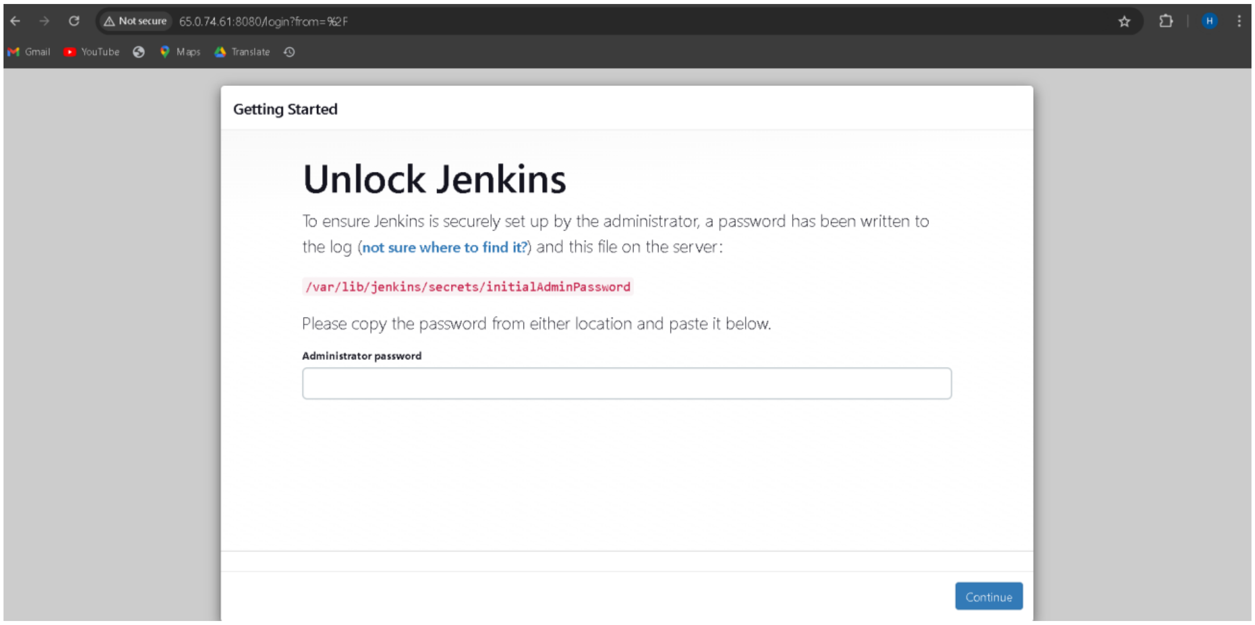
Step 8:unblock your 8080 and 8000 port

- Go to instance security and click on security groups.
- Edit inbound rule.
- Add 8080,8000 rule,choose anywhere ip4 and save it
- Copy your instance public ip and hit in browser.



Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-0cc16efbf6c11dd44	HTTP	TCP	80	Custom	0.0.0.0/0 ✕		Delete
sgr-02668eacfe1c658db	Custom TCP	TCP	8080	Custom	0.0.0.0/0 ✕		Delete
sgr-0caa944025eab4a4d	HTTPS	TCP	443	Custom	0.0.0.0/0 ✕		Delete
sgr-0bb09dd08f2fea9dd	SSH	TCP	22	Custom	0.0.0.0/0 ✕		Delete



Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

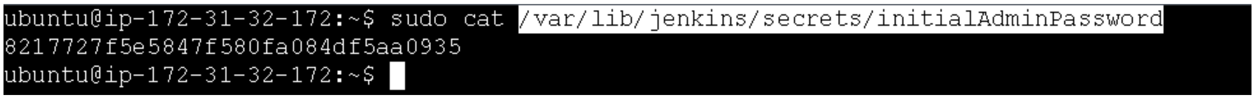
`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Step 9: to unlock Jenkins

- Your password is here `/var/lib/jenkins/secrets/initialAdminPassword`
- sudo cat `/var/lib/jenkins/secrets/initialAdminPassword`

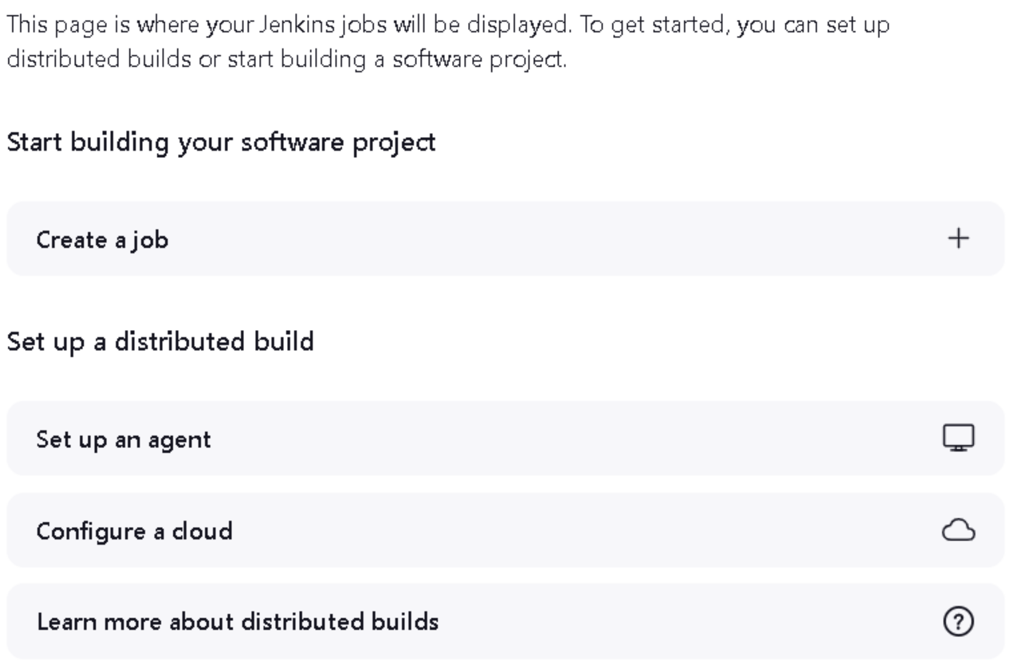


```
ubuntu@ip-172-31-32-172:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
8217727f5e5847f580fa084df5aa0935
ubuntu@ip-172-31-32-172:~$
```

- paste it.

Step 10: you have to access Jenkins

- set up agent



This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job +

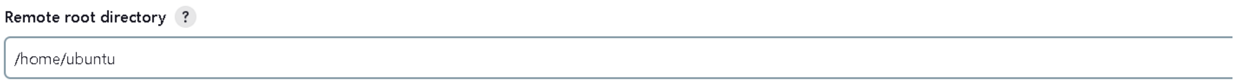
Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds

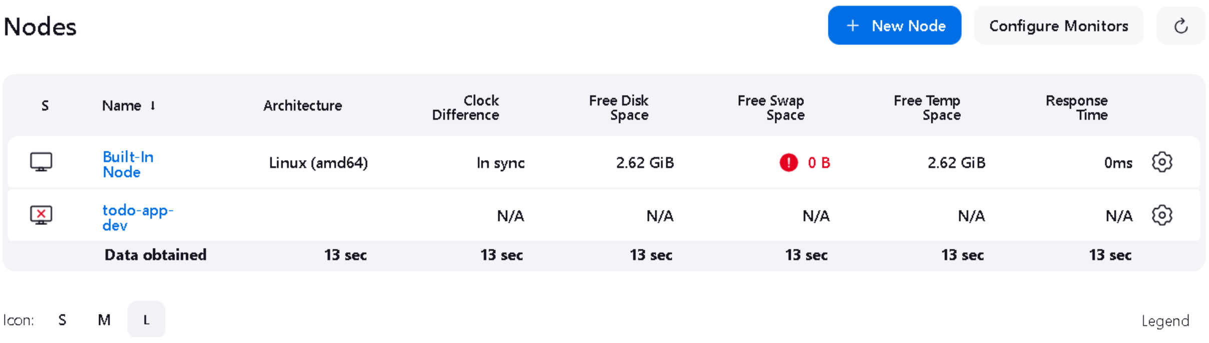
- remote root dir of your server



Remote root directory ?

/home/ubuntu

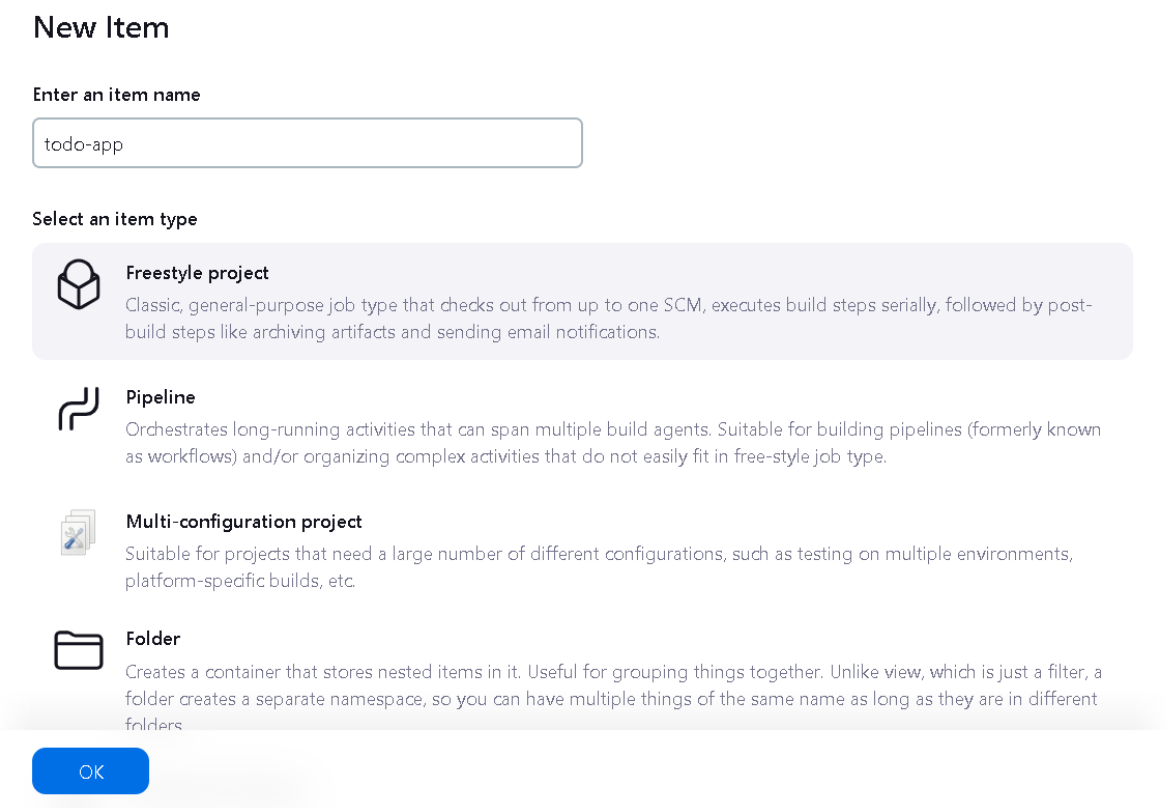
- your node is ready



Nodes

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time	
🖥	Built-In Node	Linux (amd64)	In sync	2.62 GiB	❗ 0 B	2.62 GiB	0ms	⚙
☒	todo-app-dev		N/A	N/A	N/A	N/A	N/A	⚙
	Data obtained	13 sec	13 sec	13 sec	13 sec	13 sec	13 sec	

- create a freestyle job.



New Item

Enter an item name

todo-app

Select an item type

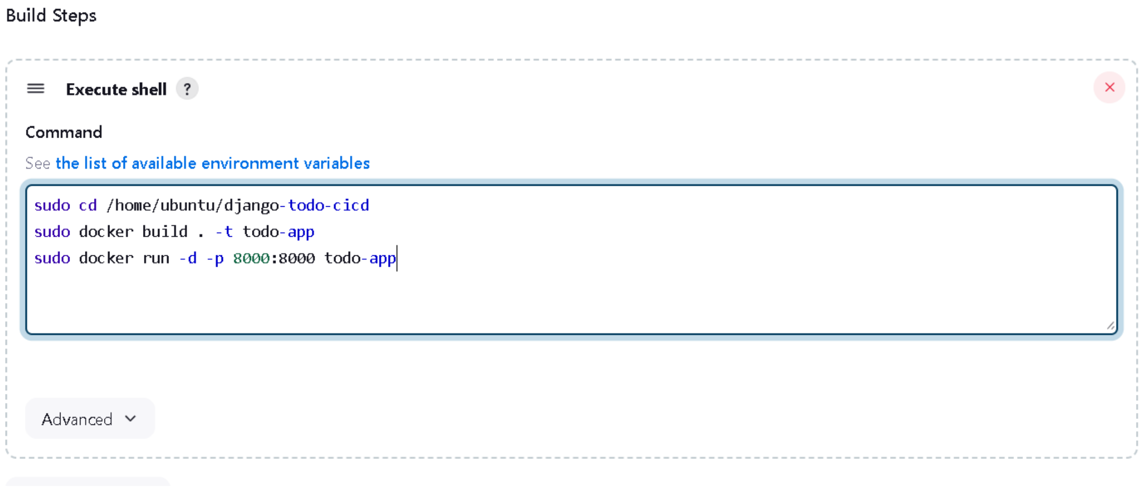
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

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
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders

- click on ok
- add build step (execute shell)



Build Steps

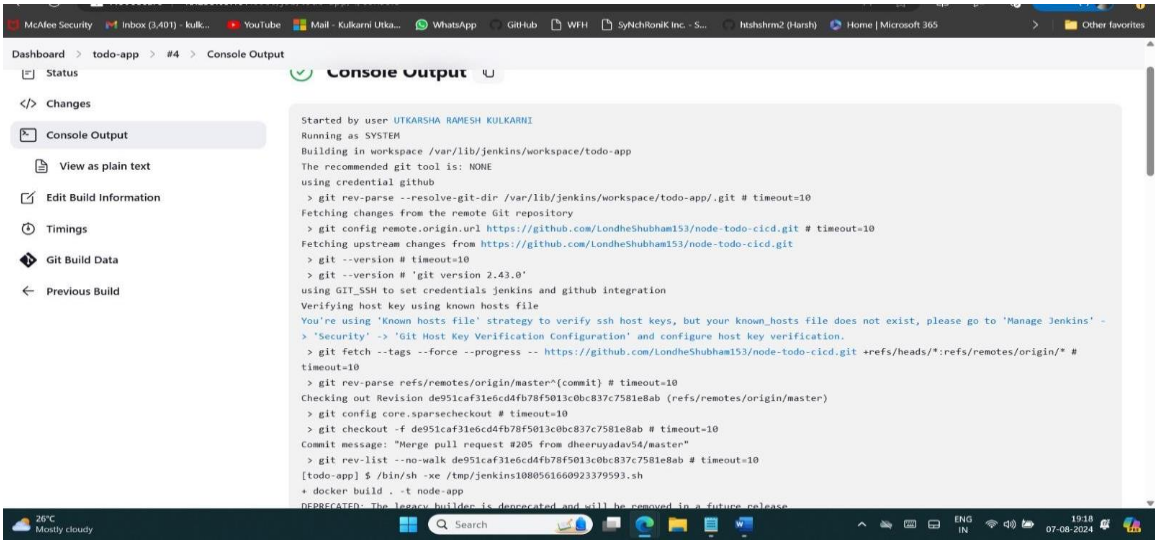
Execute shell

Command

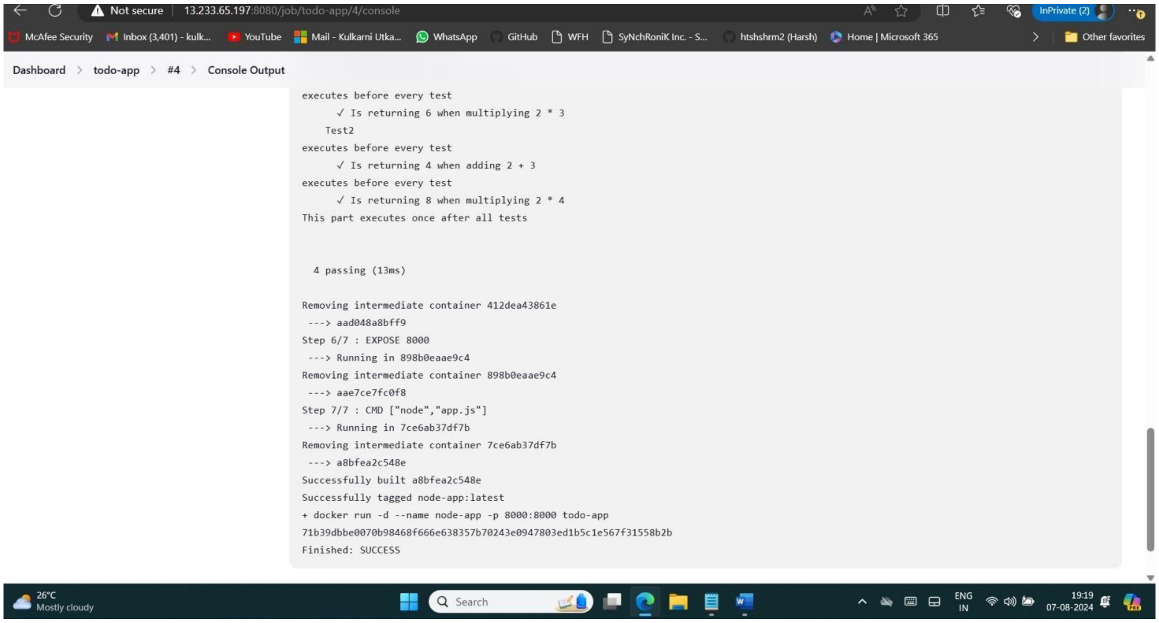
See the list of available environment variables

```
sudo cd /home/ubuntu/django-todo-cicd
sudo docker build . -t todo-app
sudo docker run -d -p 8000:8000 todo-app
```

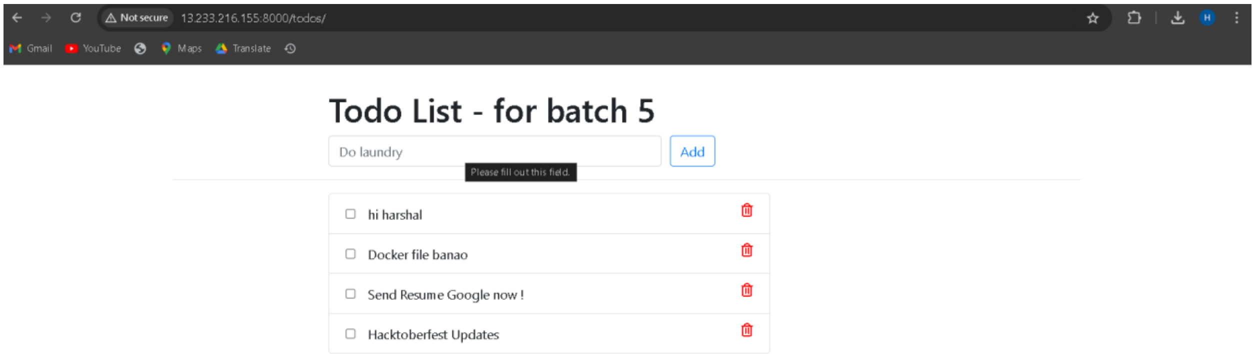
- go your instance and get permission to accesses Jenkins (chmod 777 django-todo-cicd)



```
Started by user UTKARSHA RAMESH KULKARNI
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/todo-app
The recommended git tool is: NONE
using credential github
 > 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/LondheShubham153/node-todo-cicd.git # timeout=10
Fetching upstream changes from https://github.com/LondheShubham153/node-todo-cicd.git
 > git --version # timeout=10
 > git --version # 'git version 2.43.0'
using GIT_SSH to set credentials jenkins and github 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/LondheShubham153/node-todo-cicd.git +refs/heads/*:refs/remotes/origin/* #
timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision de951caf31e6cd4fb78f5013c0bc837c7581e8ab (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f de951caf31e6cd4fb78f5013c0bc837c7581e8ab # timeout=10
Commit message: "Merge pull request #205 from dheeruyadav54/master"
 > git rev-list --no-walk de951caf31e6cd4fb78f5013c0bc837c7581e8ab # timeout=10
[todo-app] $ /bin/sh -xe /tmp/jenkins1080561660923379593.sh
+ docker build . -t node-app
DEPRECATED: The legacy builder is deprecated and will be removed in a future release
```



```
executes before every test
      ✓ Is returning 6 when multiplying 2 * 3
    Test2
executes before every test
      ✓ Is returning 4 when adding 2 + 3
executes before every test
      ✓ Is returning 8 when multiplying 2 * 4
This part executes once after all tests


  4 passing (13ms)

Removing intermediate container 412dea43861e
 ---> aad048a8bff9
Step 6/7 : EXPOSE 8000
 ---> Running in 898b0eaae9c4
Removing intermediate container 898b0eaae9c4
 ---> aae7ce7fc0f8
Step 7/7 : CMD ["node","app.js"]
 ---> Running in 7ce6ab37df7b
Removing intermediate container 7ce6ab37df7b
 ---> a8bfea2c548e
Successfully built a8bfea2c548e
Successfully tagged node-app:latest
+ docker run -d --name node-app -p 8000:8000 todo-app
71b39dbbe0070b98468f666e638357b70243e0947803ed1b5c1e567f31558b2b
Finished: SUCCESS
```


Not secure 13.233.216.155:8000/todos/
Gmail
YouTube
Maps
Translate
Add
Please fill out this field.

Todo List - for batch 5

Do laundry

- □ hi harshal
- □ Docker file banao
- □ Send Resume Google now !
- □ Hacktoberfest Updates