**Project Name : Created CICD pipeline of WEBAPP using Docker, Jenkins, and AWS.**

Project Description: CloudCodeOps is an end-to-end DevOps project that automates the deployment of code from the GITHUB repository to AWS using Docker Containers and Jenkins for CI/CD.

Execute the Application From the Following ways:

1. Execute code on a local server( VSCode).
2. Execute code on the remote server (EC2 Ubuntu)
3. Execute code on EC2 with Docker by creating Dockerfile.
4. Execute code on EC2 with Docker and Jenkins.
5. Execute code on EC2 via GITHUB and Jenkins integration.
6. Execute code on VSCODE via Docker-compose
7. Execute code on Jenkins via Docker-compose

**Prerequisites:**

1. **GitHub account.**
2. **Visual studio setup**
3. **AWS EC2 Server**

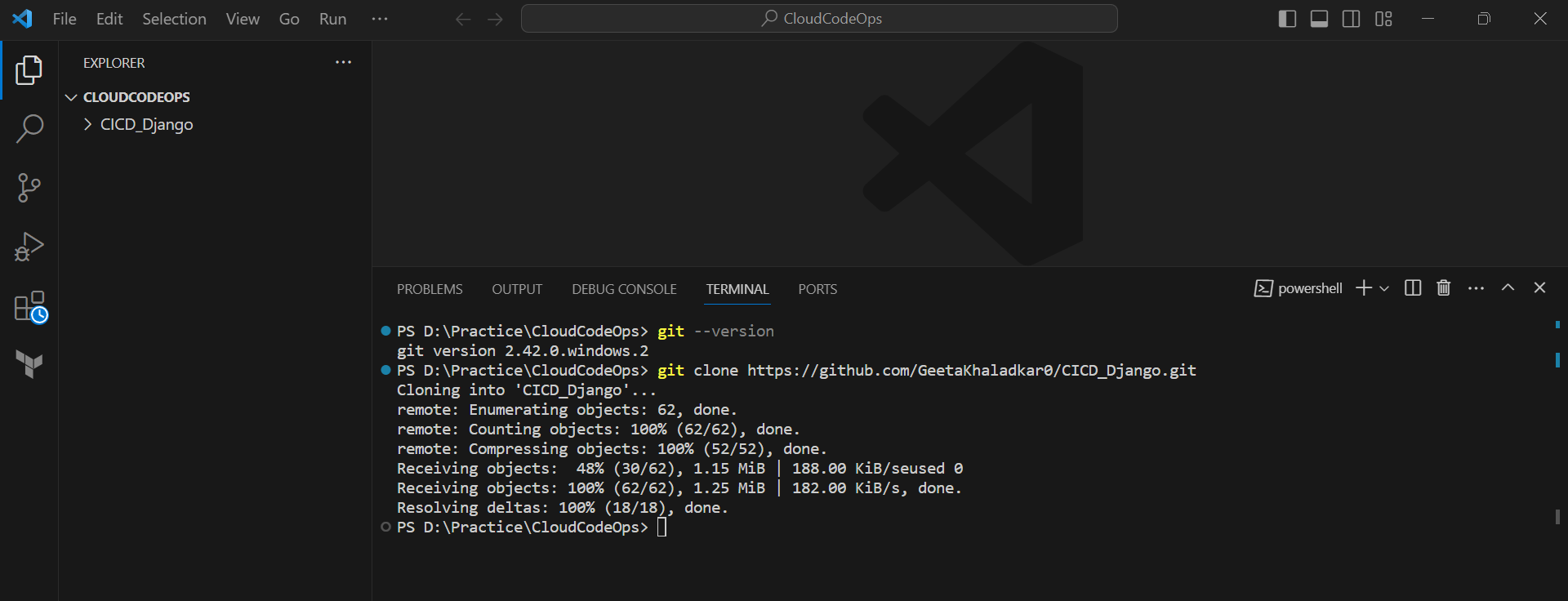
**Ubuntu : Port [ 8000 : Project Execution, 8080: Jenkins]**

1. **Installation : GIT, Python, Django, Jenkins ,Docker,Docker-**

**Step 1: Take a code from Github and run it on Local Machine**

1. **Create one project folder and Open VSCode in it.**
2. **Open Terminal and check git version for confirmation of installation.**
3. **Clone github repository in that folder.**

**git clone <URL>**

****

1. **Now create virtual environment as in one folder there are many applications and each applications have specific requirements of installation so, virtual environment required.**
   * **Install virtual environment**

**-** PS D:\Practice\CloudCodeOps> **pip install virtualenv**

* + **Check version**

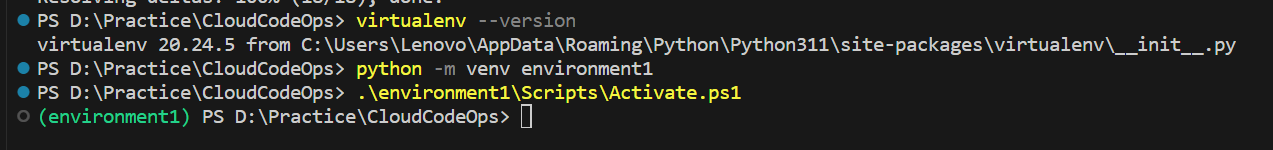
**-** PS D:\Practice\CloudCodeOps> **virtualenv –version**

* + **Create Virtual environment**

**-** PS D:\Practice\CloudCodeOps> **python -m venv environment1**

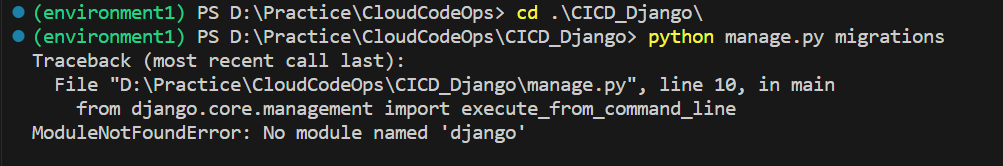
* + **Activate Environment**

**-** PS D:\Practice\CloudCodeOps> **.\environment1\Scripts\Activate.ps1**



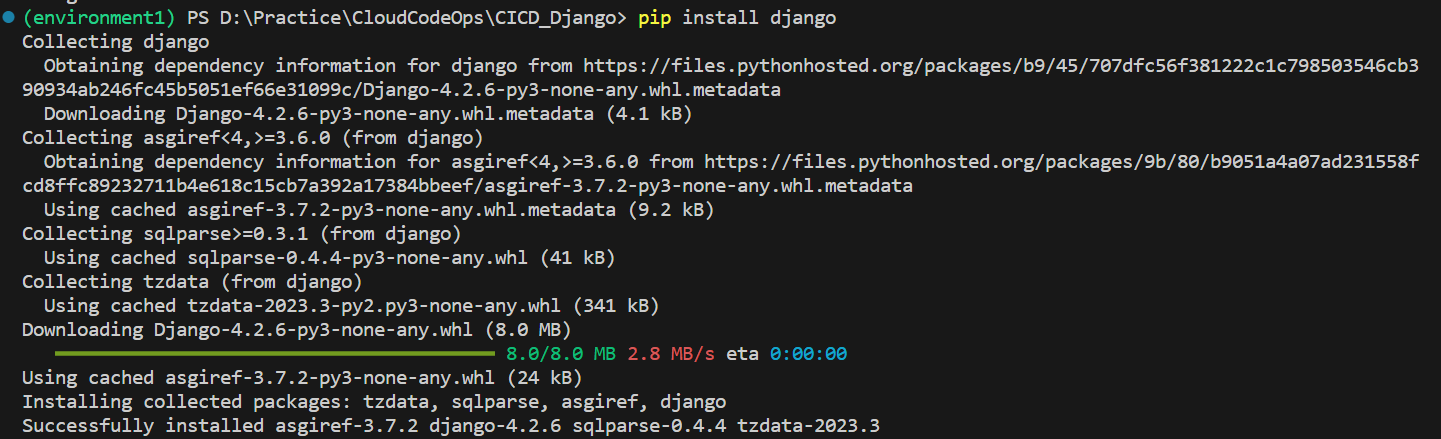
1. **Now we are ready to run our project in this Virtual Environment.There is readme.md file in project folder.So open folder and follow the steps.**

* **Open project folder to run it**

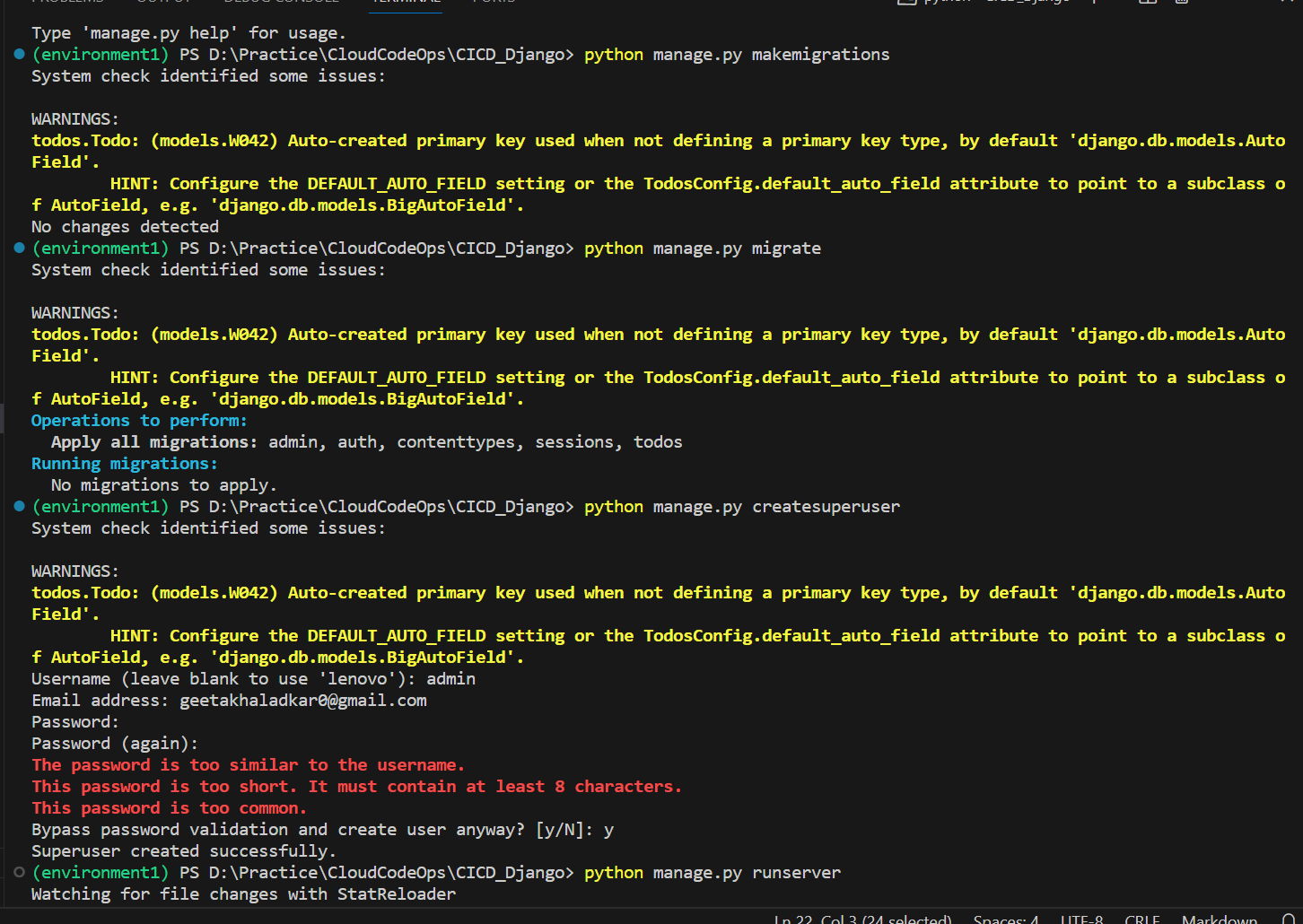
****

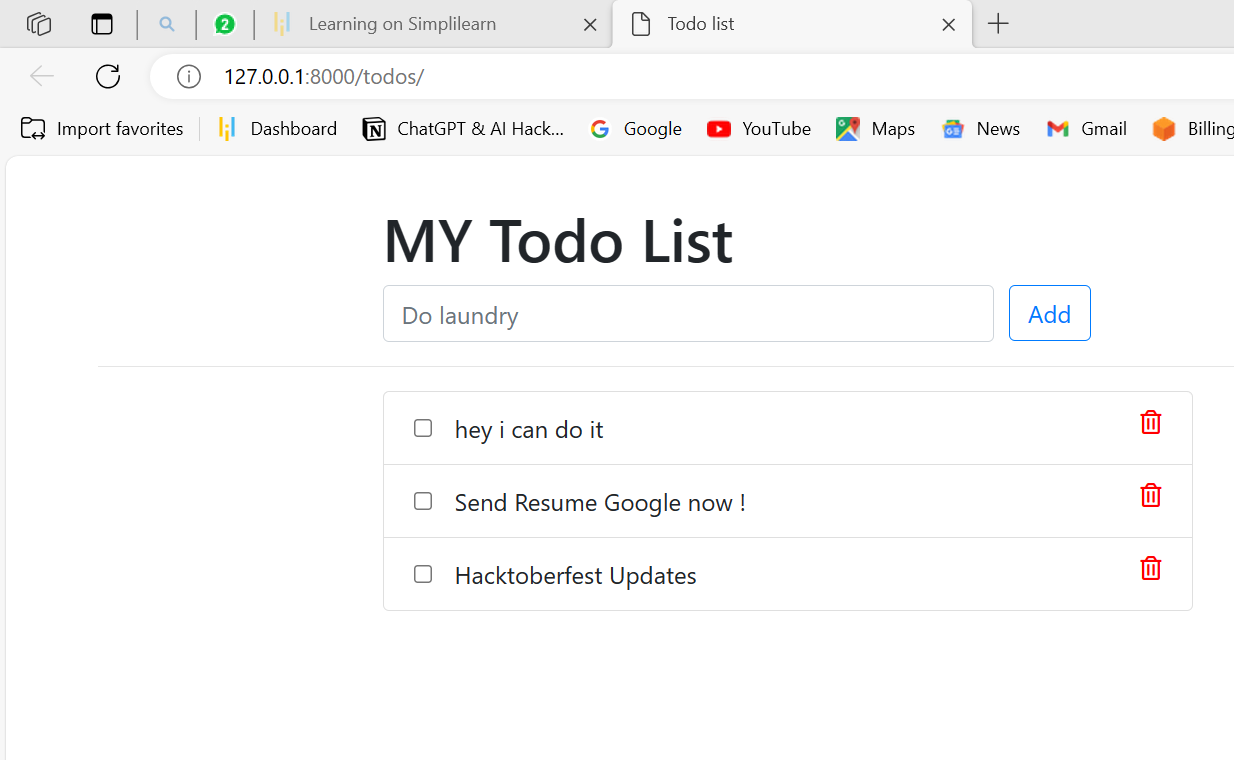
* **Execute command but Django not available so install Django.**

**-** (environment1) PS D:\Practice\CloudCodeOps\CICD\_Django> **pip install Django**

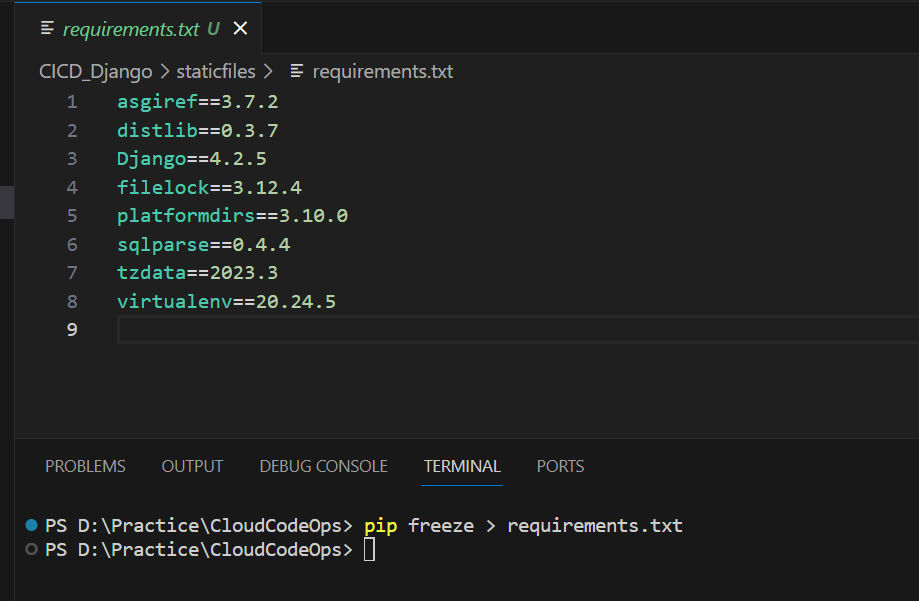
****

* (environment1) PS D:\Practice\CloudCodeOps\CICD\_Django> **python manage.py migrations**
* (environment1) PS D:\Practice\CloudCodeOps\CICD\_Django>**python manage.py makemigrations**
* (environment1) PS D:\Practice\CloudCodeOps\CICD\_Django> **python manage.py migrate**
* (environment1) PS D:\Practice\CloudCodeOps\CICD\_Django> **python manage.py createsuperuser**
* (environment1) PS D:\Practice\CloudCodeOps\CICD\_Django> **python manage.py runserver**
* **Click on link which displayed on output .**

****

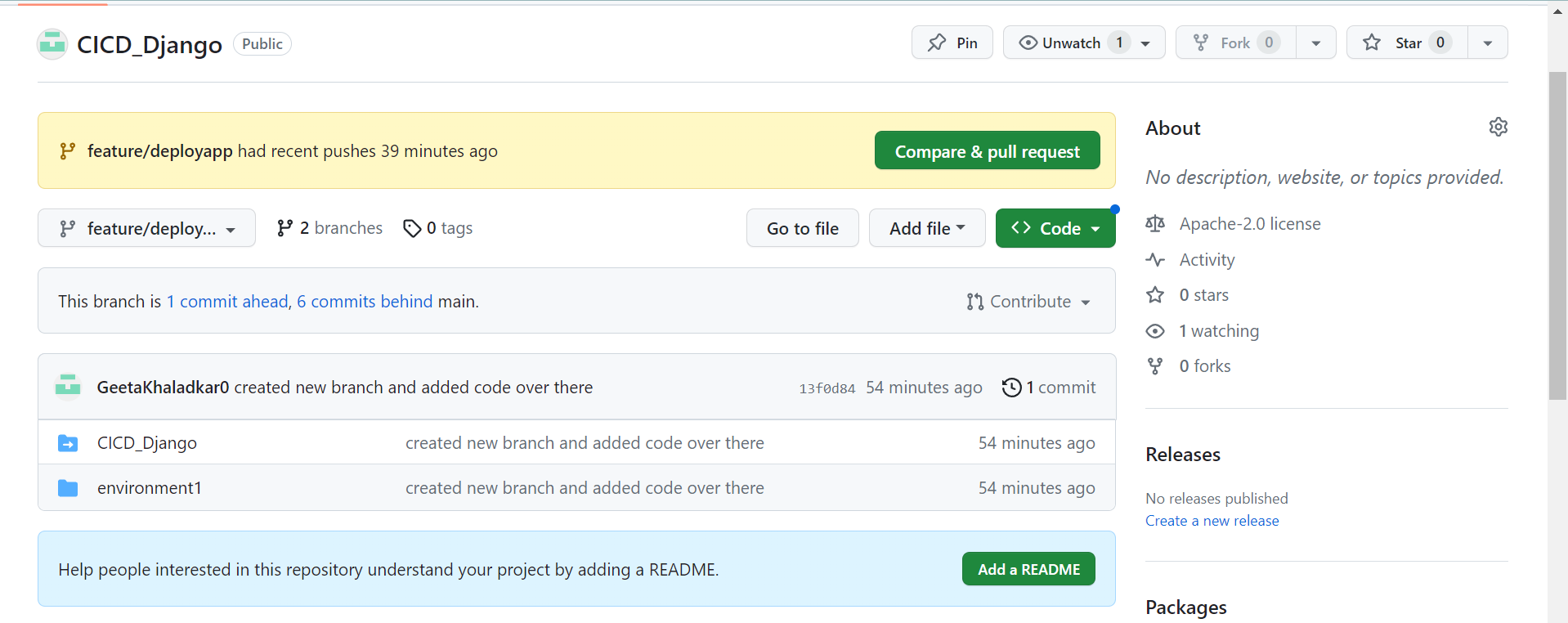
****

1. **Here as compared to readme file we added more things which are not mentioned so, create 1 requirements.txt file to avoid confusion and error free execution.**

****

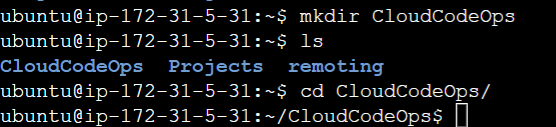
1. **If there is any situation and we need to do some changes in code and don’t want to interfere with main code .so first initialize GIT and create new branch then add and commit code there.**

* **git init**
* **git –version**
* **git config - -global user.name “Geeta”**
* **git config - -global user.email “**[**geetakhaladkar0@gmail.com**](mailto:geetakhaladkar0@gmail.com)**”**
* **git remote add origin** [**https://github.com/GeetaKhaladkar0/CICD\_Django.git**](https://github.com/GeetaKhaladkar0/CICD_Django.git)
* **git remote -v**
* **git checkout -b feature/deploy-app**
* **git add .**
* **git commit -m “done changes”**
* **git push --set-upstream origin feature/deployapp**

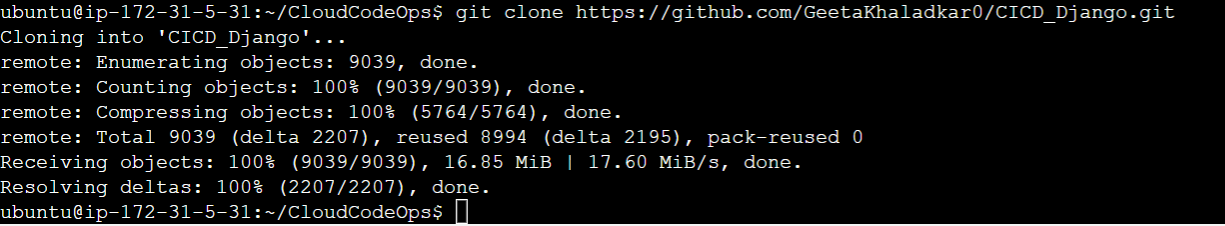
****

**Step 2: Now run code manually on AWS EC2 instance .**

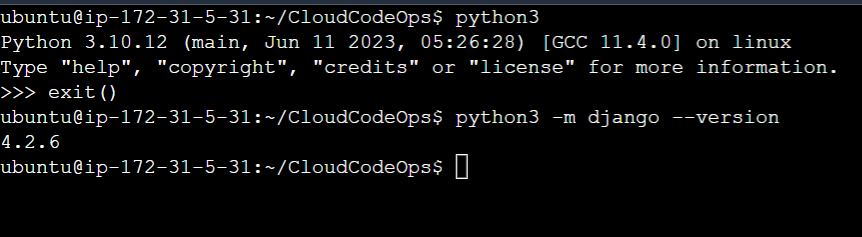
* **Because we want to make pipeline as we take a code from GITHUB and deploy it through Jenkins.**
* **Port: [8080 : Jenkins, 8000 : Execution]**
* **Create directory.**
* **mkdir CloudCodeOps**
* **cd CloudCodeOps**

****

* **Clone the code from github**
* **git clone** [**https://github.com/GeetaKhaladkar0/CICD\_Django.git**](https://github.com/GeetaKhaladkar0/CICD_Django.git)

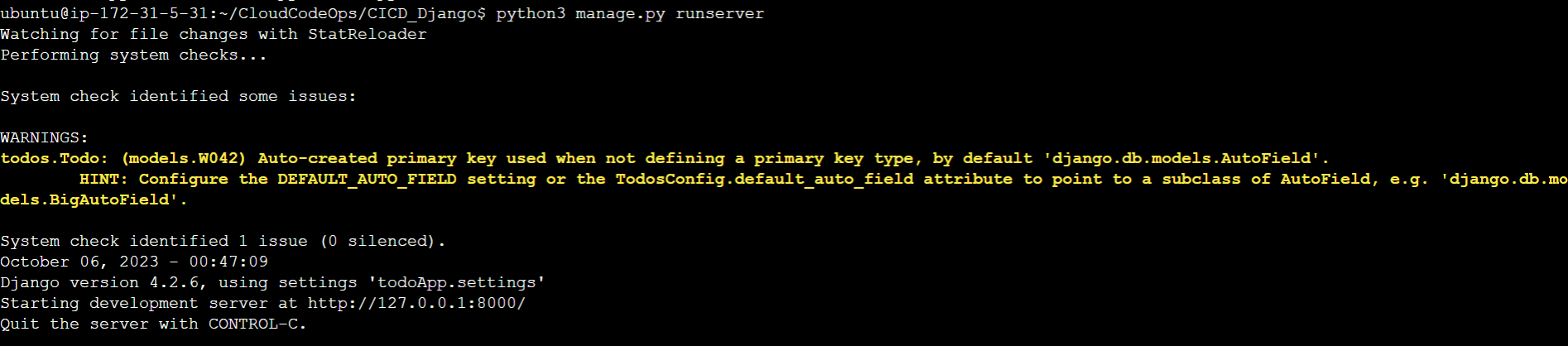
****

* **Here we are not creating any virtual environment .Directly getting ready to execute code because it is total system to use.**
* **Install requirements in this ubuntu system:**
  + **sudo apt-get update**
  + **sudo apt install Django**
  + **sudo apt install python3-pip**
  + **python3 -m django –version**

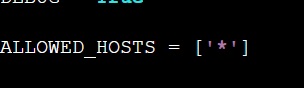
****

1. **Now execute code.**

* **python3 manage.py runserver**

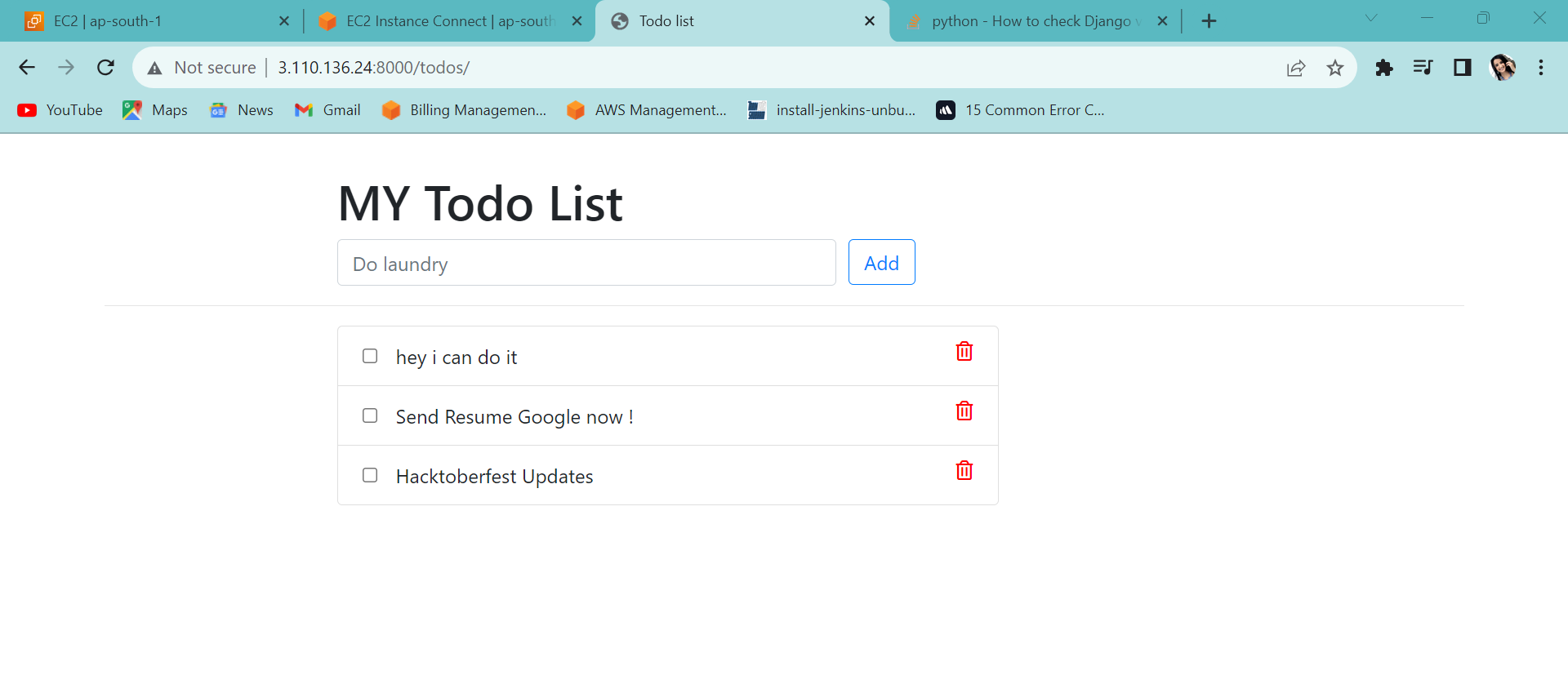
****

* **Here after execution we can see that Code is running on “127.0.0.1:8000”. This 127 IP is localhost IP.**
* **we running on remote means exposing project remotely so there must be 0.0.0.0:8000. Edit Inbound Rule in EC2. [0.0.0.0] means accessing from anywhere.**
* **Now, do one change in code .**
  + **In settings.py allocation Host must be [‘\*’] .It means accessible from anywhere all.**

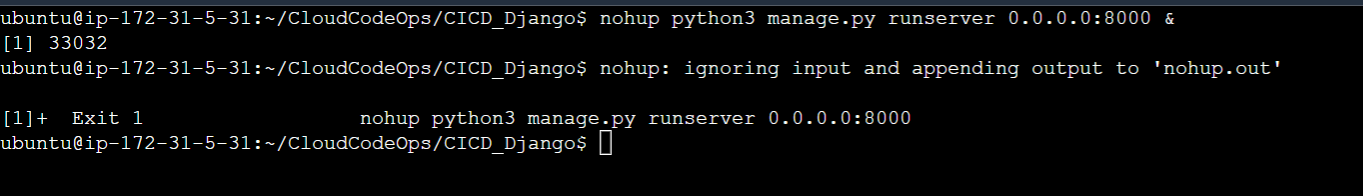
****

****

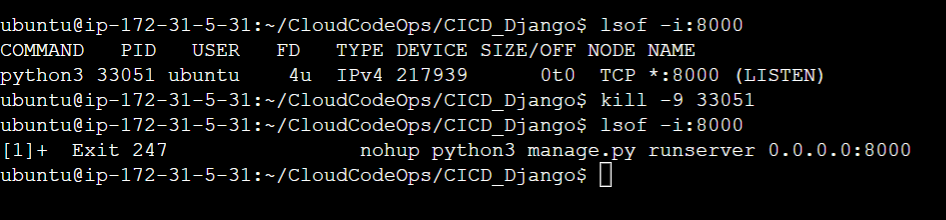
* **Copy Public IP of EC2 and run with port 8000.**

****

* **While running code terminal continuously showing log files and we cant do any work on terminal.**
  + **Add nohup in starting of command and & in end.**

****

* **List the services which are running on port 8000.**
  + **lsof -i: 8000**
  + **kill -9 <process ID>**

****

**Step 3: Now run code on AWS EC2 instance with help of Docker -automation .**

* **We have Ec2 Ubuntu server .On that server various applications, services are running at a time so server having too much load.**
* **So, for better performance we need to create virtual environment.**
* **Docker gives a container which acts like a virtual environment.**

**Inside that container we run our application by installing our prerequisites of our project.**

* **Docker having two modules: Image and Container**

**Docker Image: It is a copy of project which can create by Dockerfile.**

**Docker Container: It acts as virtual environment as Docker image run on container.**

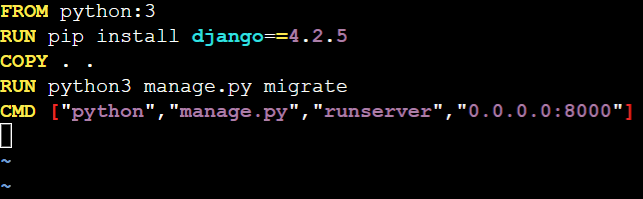
**Prerequisite: Docker**

1. **Install Docker on Ec2 server.**
   * **Sudo apt-install docker.io**
   * **Now we are ready to create virtual environment as container.**
2. **Create Container for this Django project through Dockerfile**
   * **vi Dockerfile**

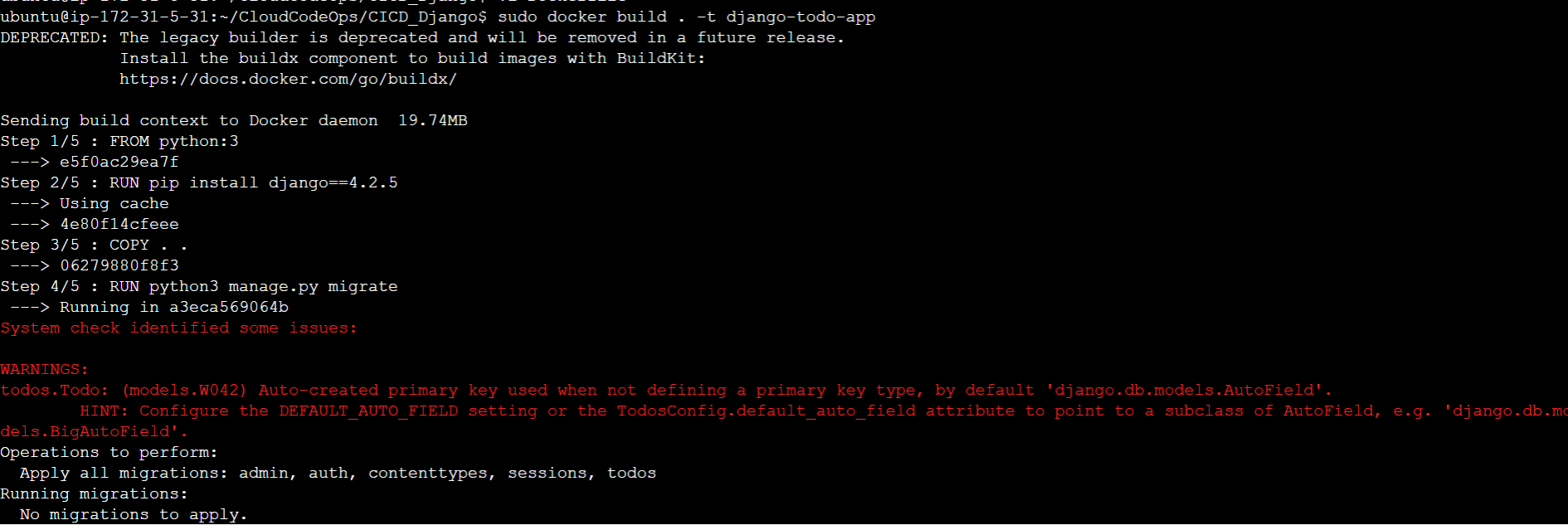
* **Whatever we done manually till yet now onwards we can automate it.**

1. **Firstly, Update OS (FROM).Docker call Ubuntu image where Python3 already installed – FROM python:3**
2. **Execute python Dependencies as it’s a Django project. So, install it. – RUN pip install Django == 4.2.5.**
3. **Copy the code from this server as we clone it already and paste it into Docker Container – COPY . .**
4. **Now ,we migrate that manage.py file – RUN python manage.py migrate**
5. **Finally, we put a command to execute code. Command is in array format.**

**CMD [“python”, “manage.py”, ”runserver”, “0.0.0.0:8000”]. ESC :wq (Enter) to end file.**

****

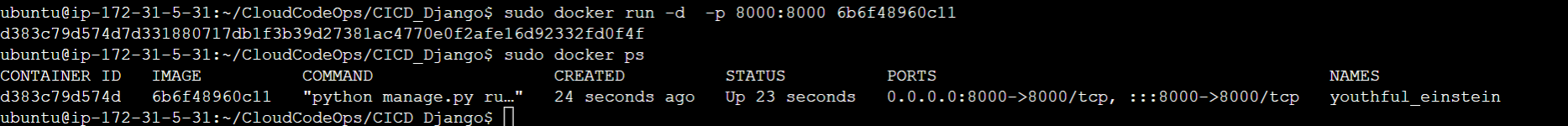
1. **Now, build docker image.Sudo is uder for permission from server which like superuser. sudo docker build . -t Django-todo-app.**

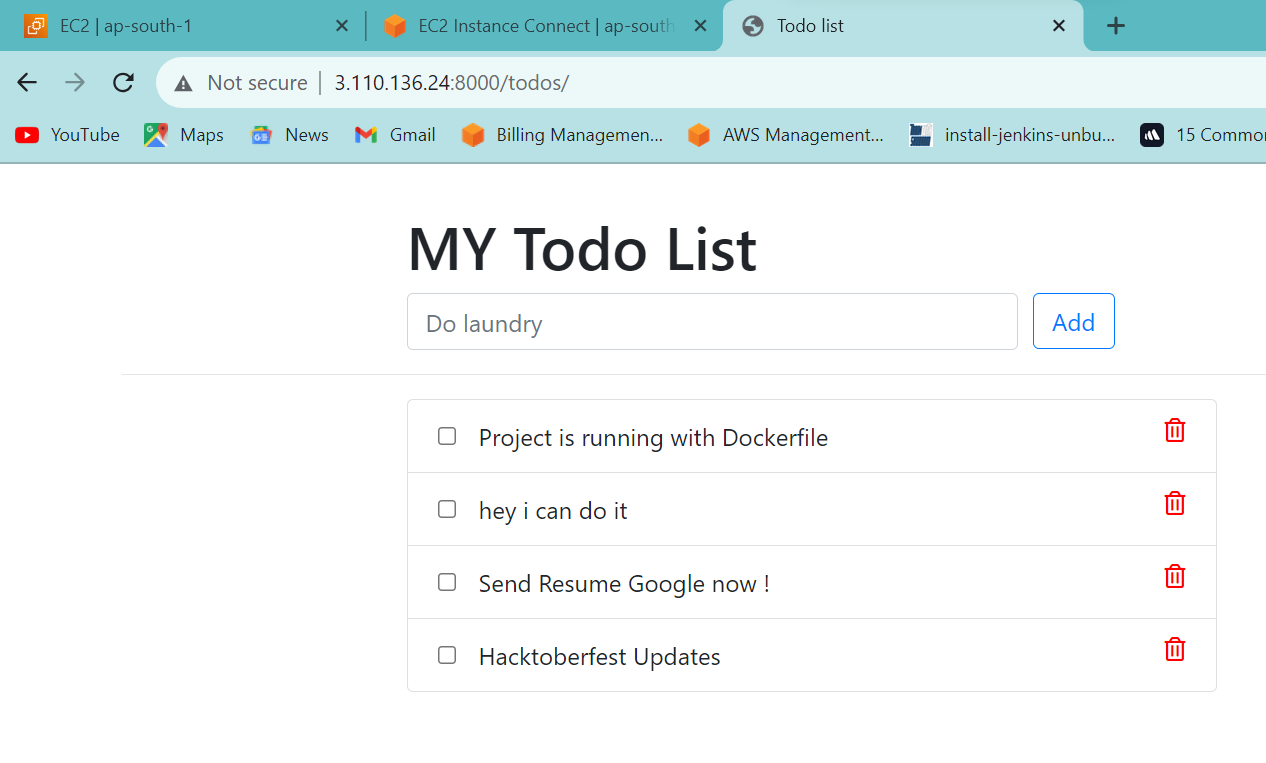
****

1. **After building image we get one ID and now we run that ID to execute code in container. While running image we need to do port mapping .**

**Sudo docker run -d -p 8000:8000 <containerID>**

**-d is use for run image in background.**

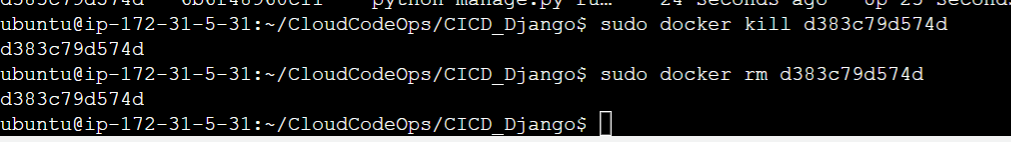
****

****

* **Free the container by killing container ID after executing code.**

**sudo docker kill <containerID>**

**sudo docker rm <containerID>**

****

**Step 4: Now run code on AWS EC2 instance with help of Docker and Jenkins (Continious Integration).**

**Manually Jenkins Installation Steps on UBUNTU server:**

1. **Jenkin need java on which Jenkins is running. So first install it. But before update the system.**

* **sudo apt-get update**
* **sudo apt install openjdk-11-jre**
* **java --version**

1. **Now install Jenkins.First curl means it connect with jenkin URL and take all libraries from there.**

**curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null**

1. **Push Jenkin 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**

1. **Update system**

* **Sudo apt-get update**

1. **Install Jenkins**

* **sudo apt-get install jenkins.**

1. **After rebooting or restarting EC2 ,Jenkins will started automatically each time.**

* **sudo systemctl enable Jenkins**

1. **Start Jenkins**

* **Sudo systemctl start jenkins**

1. **Status of Jenkins**

* **Sudo systemctl status Jenkins**

1. **Enable port 8080 in EC2 inbound rules as Jenkins run on port 8080.**

* **Copy <public IP:8080> and paste on URL Jenkin is running**

1. **Get the password .copy path from Jenkins Dashboard and paste on terminal**

* **sudo cat /var/lib/jenkins/secrets/initialAdminPassword**
* **Paste password on dashboard and install it with suggested plugins**

**Jenkins is a tool which is used for automation.By this we can automated scripts, dockerfiles and can do CICD.**

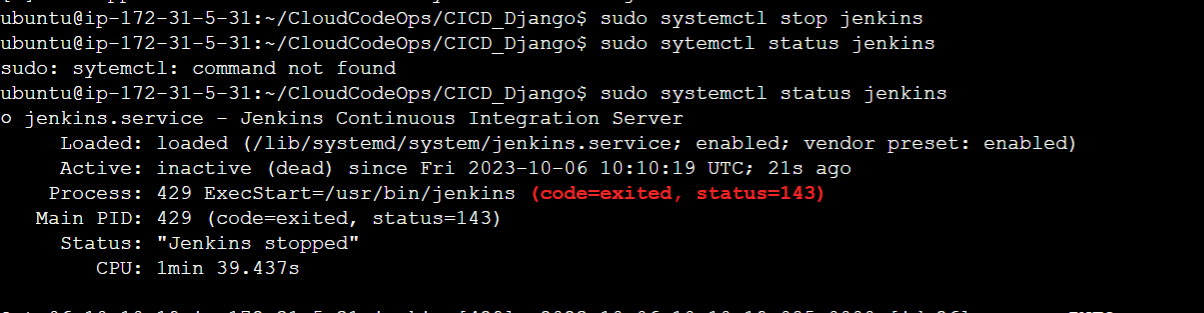
**Now, we install Jenkins manually so we can automate that also by Dockerfile.**

**Jenkins can run on one system only one time on port 8080 so first disable currently running Jenkins means stop Jenkins**

**Automate Jenkins installation through Dockerfile:**

1. **Stop Jenkins**

* **sudo systemctl stop Jenkins**

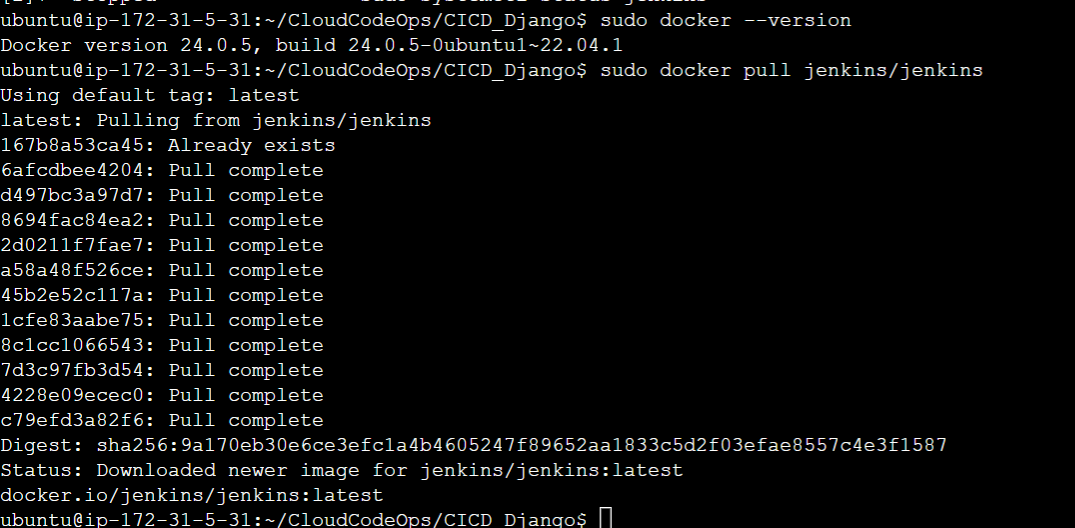
****

1. **Now, automate Jenkins through Docker. So, Docker must available in server.**

* **Docker installation : sudo apt install docker.io**
* **Check Docker version : sudo docker –version**
* **Now pull Jenkins image from Docker . image:jenkins/jenkins**

**sudo docker pull Jenkins/Jenkins**

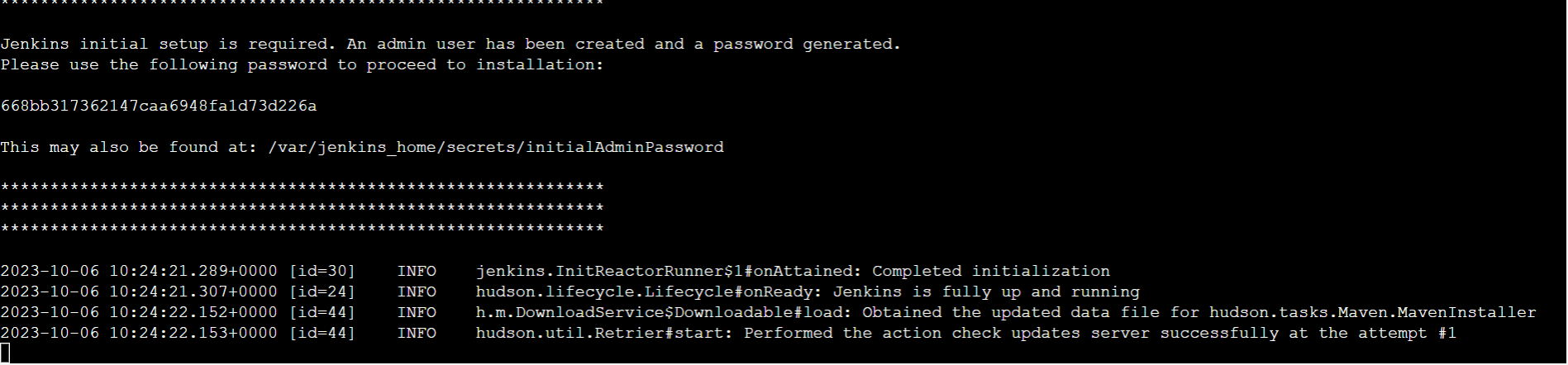
**In this image all installations steps are alredy combined.**

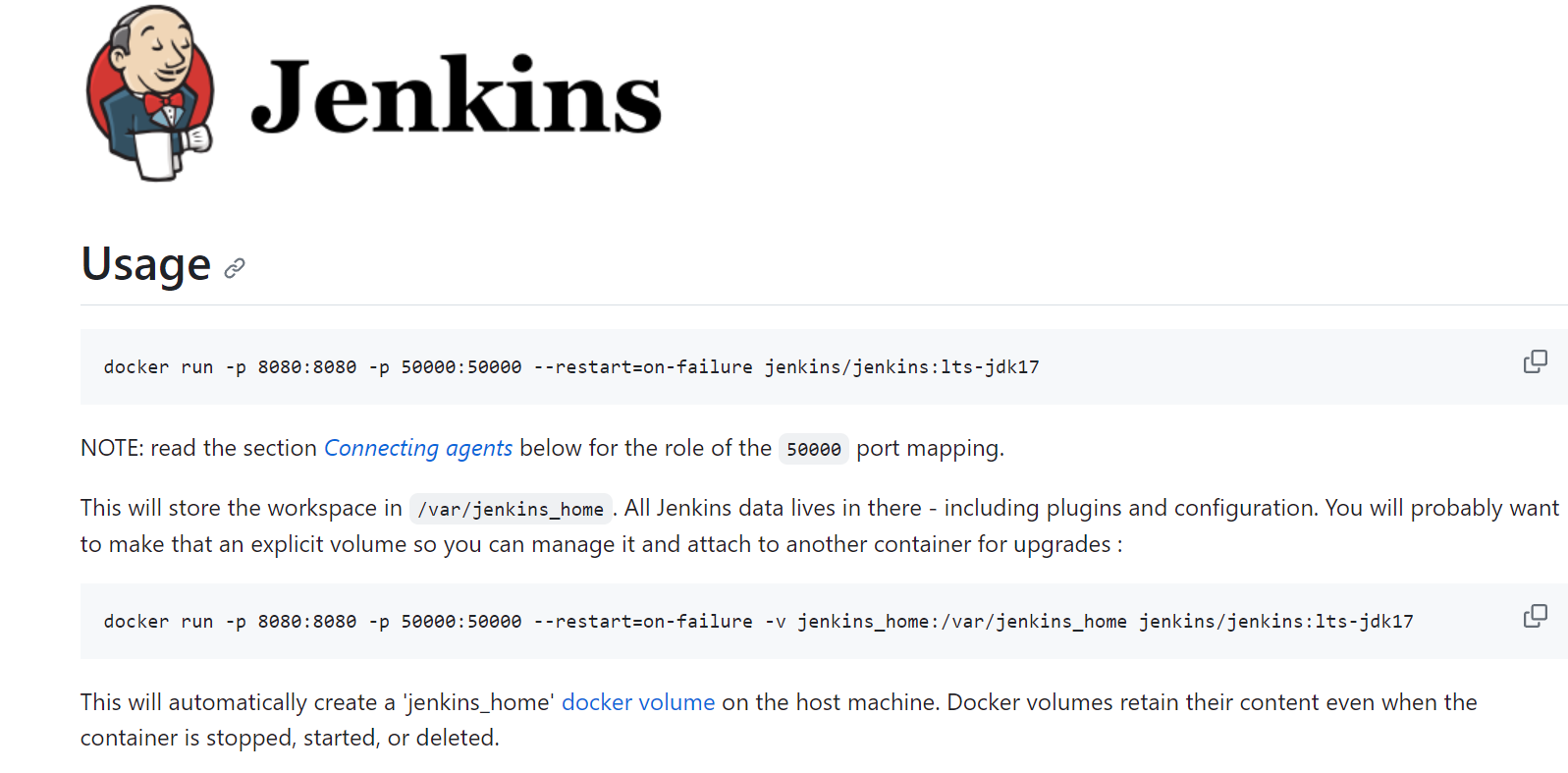
****

* **Check that Jenkins installed or not. Run that image and map the port 8080 and assign volume.**

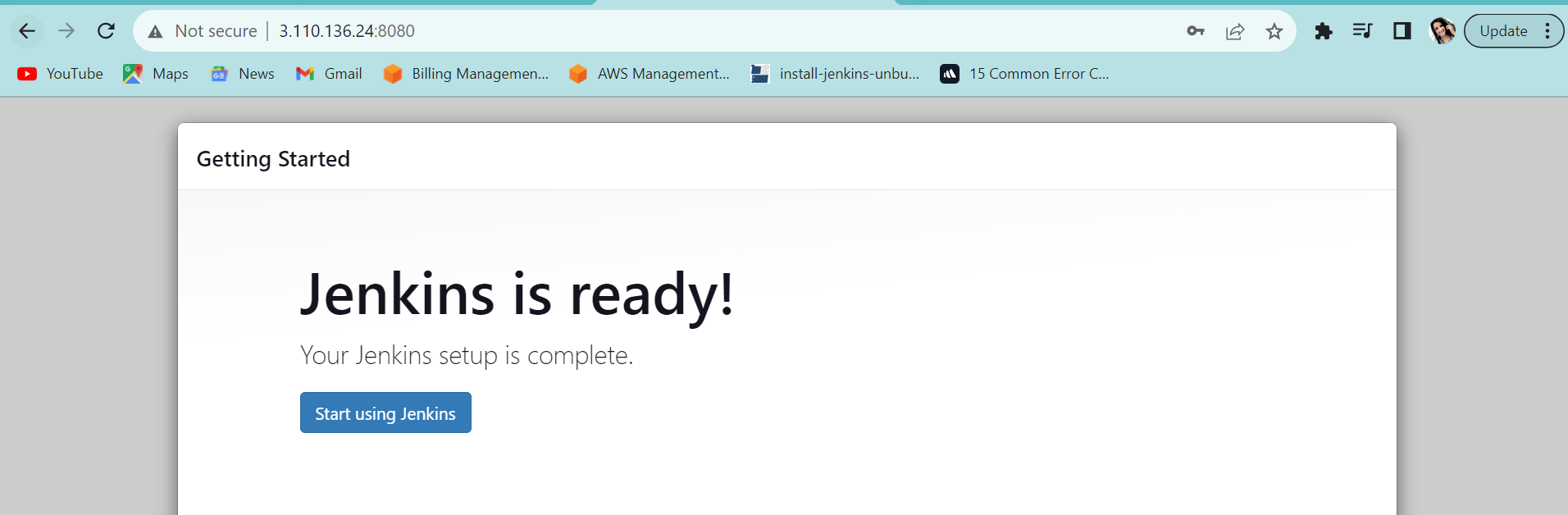
**nohup sudo docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v jenkins\_home:/var/jenkins\_home jenkins/jenkins:lts-jdk17 &**

* **After executing this command we get the admin password which need to paste on Jenkins Dashboard.**
* **With the help of [nohup &] terminal is accessible .**

****

****

* **Jenkins is ready**

****

**Now, By using Jenkins we can run our code through pipeline.**

**CI – continuous integration CD – contiious deployment**

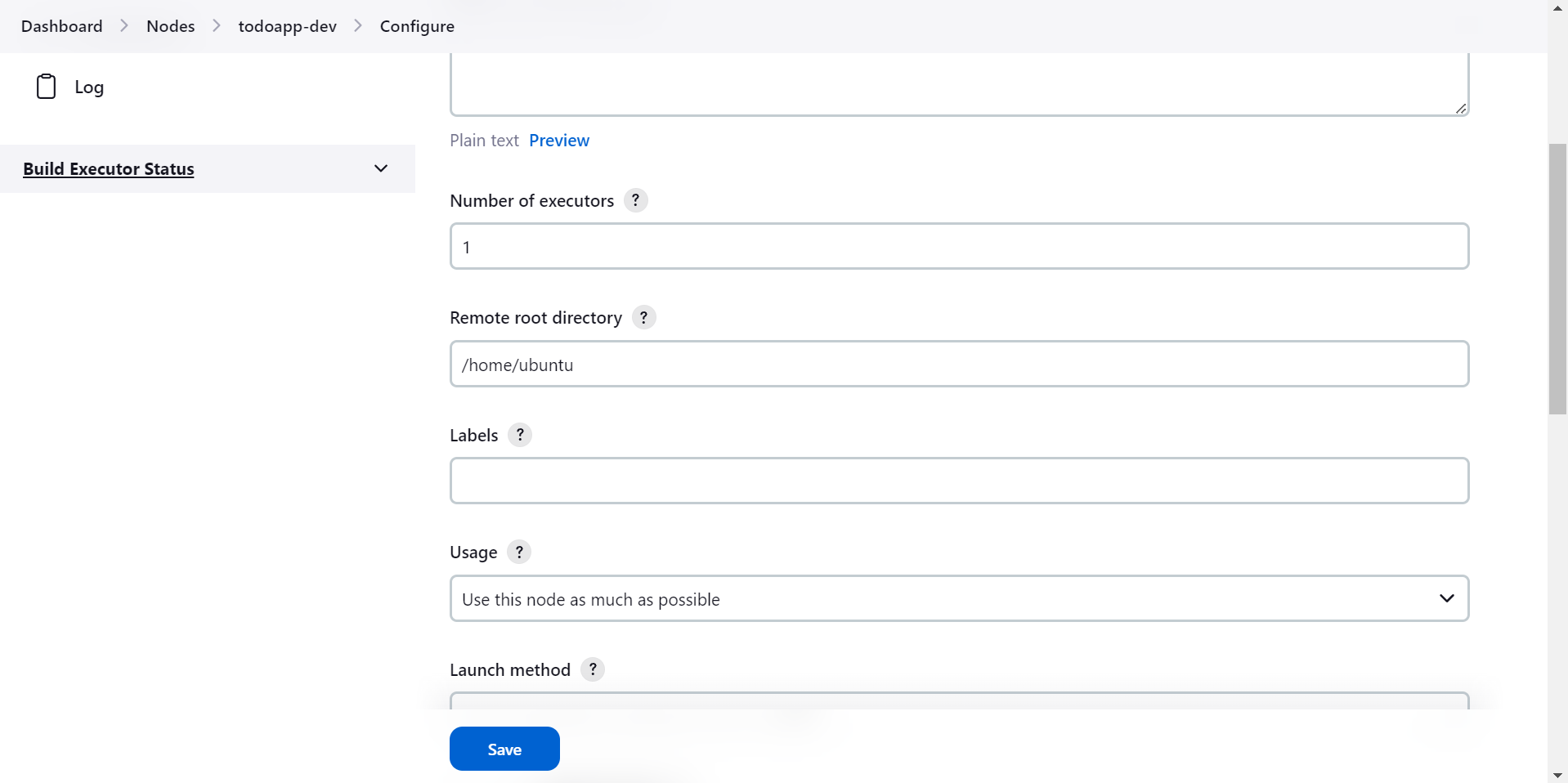
**First we want to make CICD pipeline to follow steps and this work need setup which called NODE**

**Agent is responsible to perform all operations which are running under pipeline.**

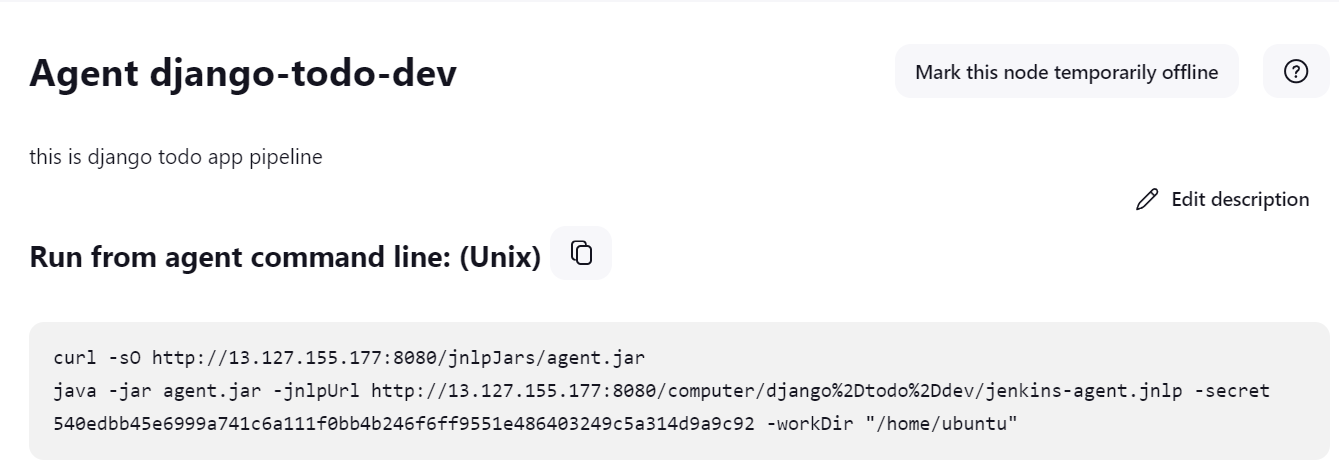
**NODE is environment which allows you to create pipeline**

**We are using Jenkins which is installed by manually**

1. **Create an Agent**

****

**Jenkin URL and Agent URL must run on same public IP so, add this command on terminal which we get after creating agent.**

****

1. **Create a Job**

**Here we are doing only continuous Deployment. Means only we took a code from Ec2 server and run with the help of Jenkins.**

1. **Create a job -> give name -> SCM: None -> Build steps :Execute shell.**

* **Whenever we build pipeline we need to give execution steps to build code on Jenkins.**
* **First give permission to Jenkins to access code (read,write ,execute).**

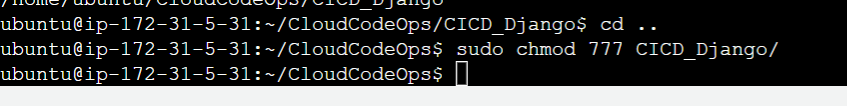
**cd ..**

**sudo chmod 777 <project folder>**

**sudo vi etc/sudoers**

**jenkins ALL=(ALL) NOPASSWD: ALL**

* **add this in bottom of sudoers file and exit with :x**

****

**:wq – saves the changes, and exits Vim**

**:x – save the changes made, and exits Vim**

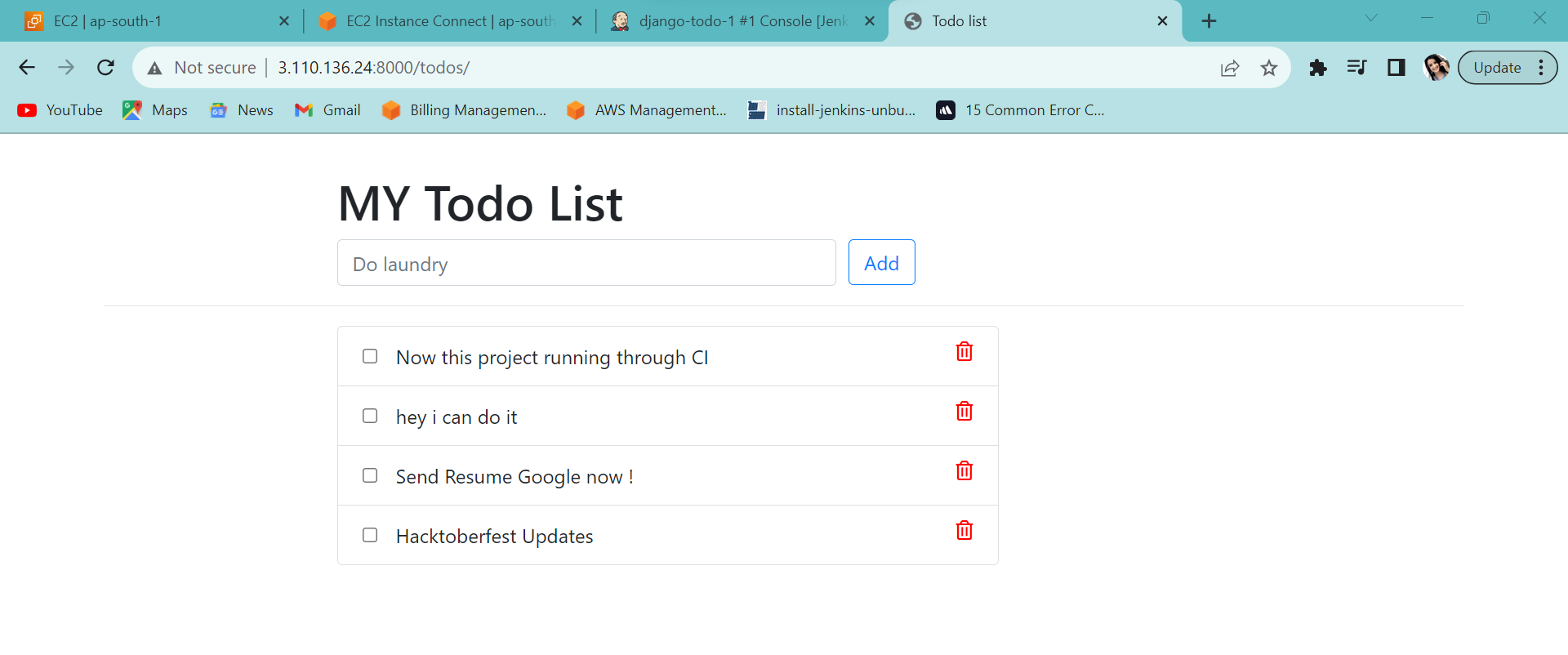
* **Write code in execution shell for building pipeline.**

**#!/bin/bash**

**sudo docker build -t todo-dev -f /home/ubuntu/CloudCodeOps/CICD\_Django/Dockerfile /home/ubuntu/CloudCodeOps/CICD\_Django**

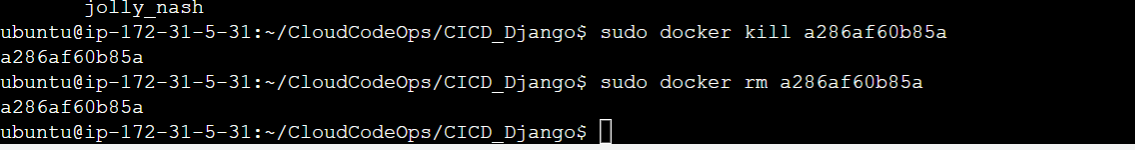
**sudo docker run -d -it -p 8000:8000 todo-dev**

****

****

**Step 5: Now run code on Jenkins with GITHUB integration (Continious Integration Continious Deployment).**

* **Use Personal Access Token for integration.**
* **On Ec2 server Jenkins must need to be installed. So, Jenkins will start.**
* **Jenkins will take a code from GITHUB and Build it on Jenkins .**
* **Jenkins GITHUB integration with secret key.**
* **First, free the 8000 port and then start the process.**

****

**Jenkins and GITHUB are two separate platforms and to integrate / communicate this we need a security layer which is ” Personal Access Token”.**

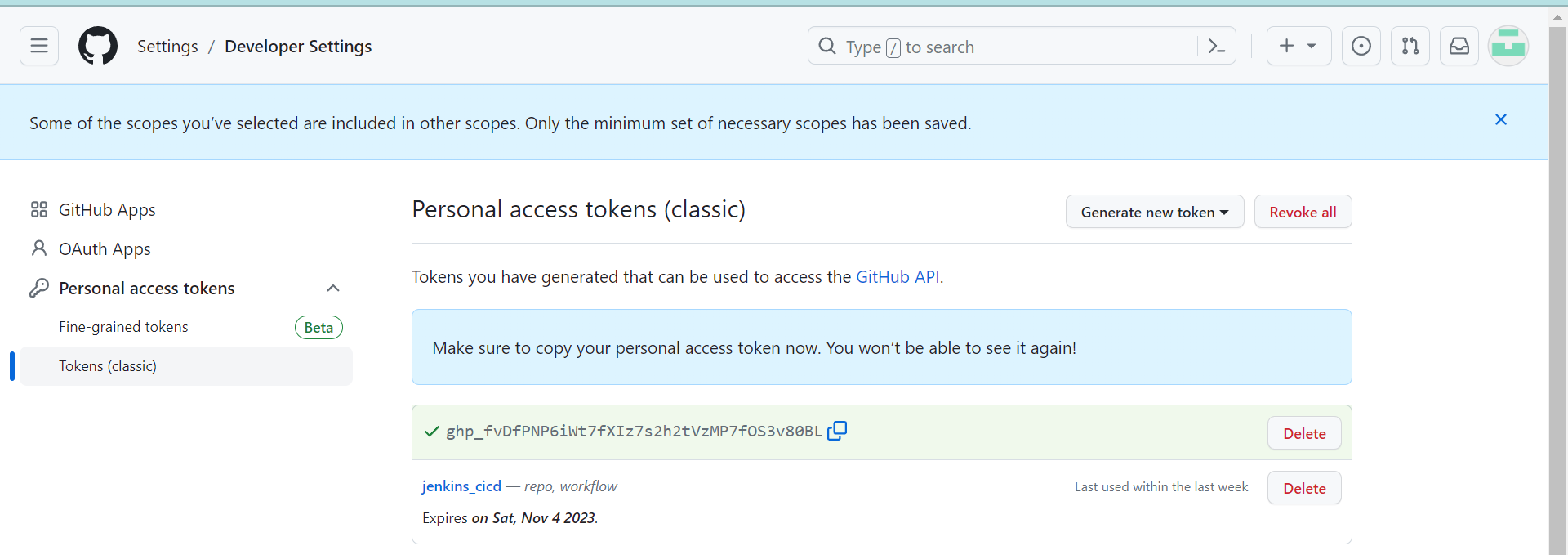
**Steps:**

**Github-> setting -> developer settings -> personal access tokens ->**

**Tokens(classic) -> generate token -> Name -> Generate**

**Copy Token and paste somewhere .Its confidential.**

**ghp\_fvDfPNP6iWt7fXIz7s2h2tVzMP7fOS3v80BL**

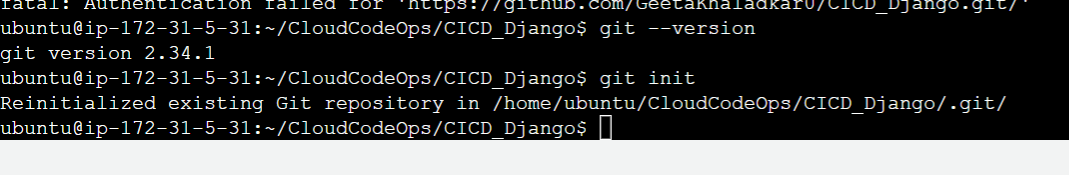
****

1. **Code is in Ec2 server so, first push code on github**

* **Check git version and initialize it.**

**git –version**

**git init**

****

* **If On EC2 there is another GITHUB URL and we want to set our URL**

**git remote set-url origin https://github.com/GeetaKhaladkar0/CICD\_Django.git**

* **Set the configuration. First Create repository on Github then put commands on terminal.**

**git remote add origin** <https://github.com/GeetaKhaladkar0/CICD_Django.git>

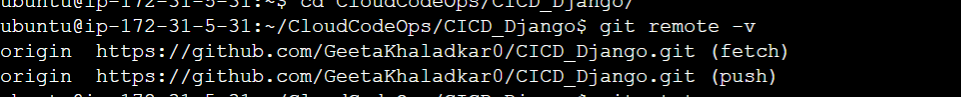
**git branch -m Main**

**git push -u origin main**



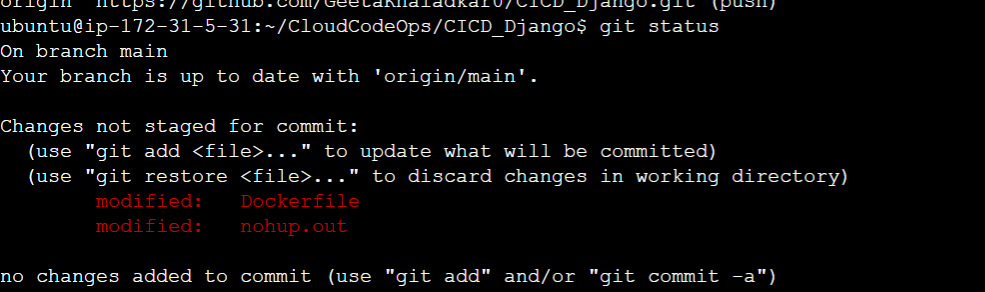
* **See the configuration .If not set the configuration**

**git remote -v**

****

* **Check the status**

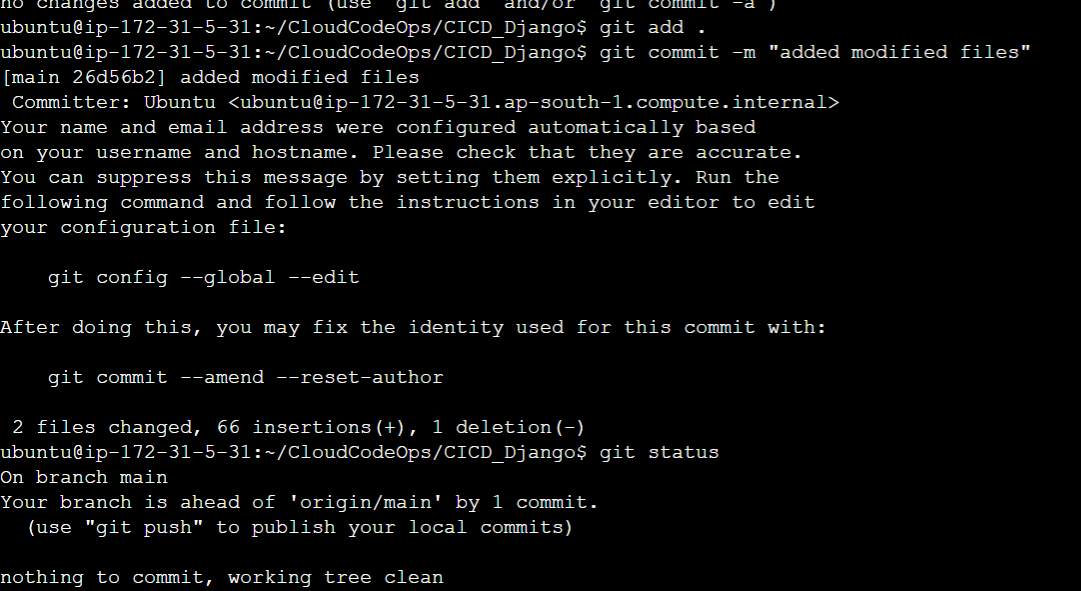
**git status**

****

* **Add a code and commit it**

**git add .**

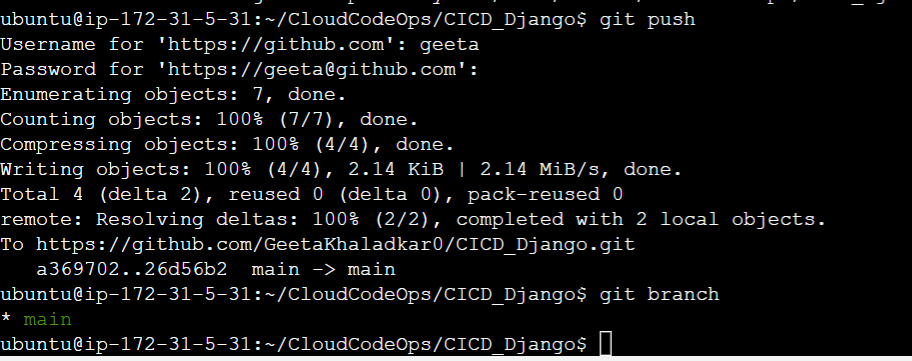
**git commit -m “message”**

****

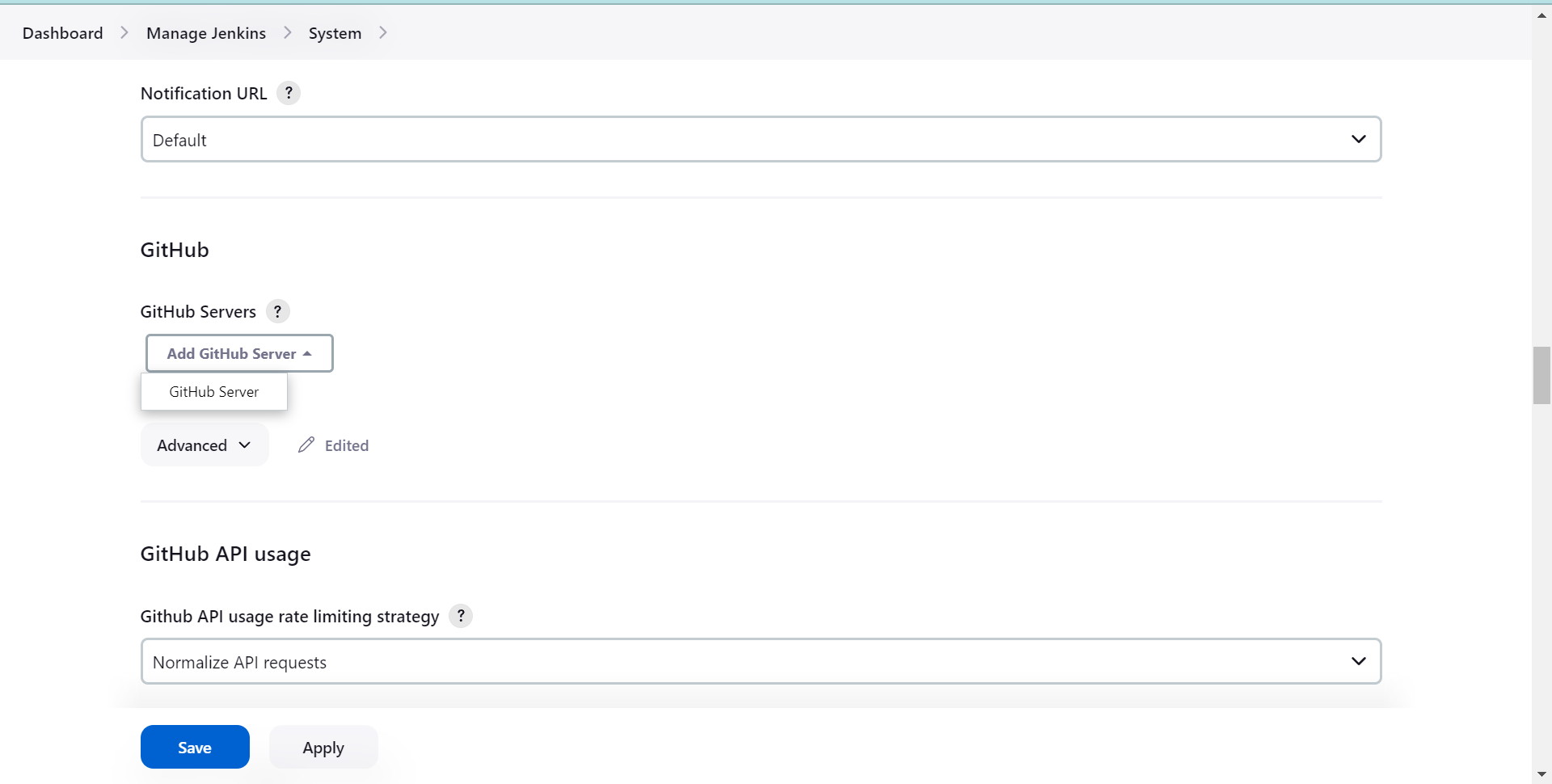
* **Push code on GITHUB**

**git push**

**Username : Geeta Password : Personal Access Token**

****

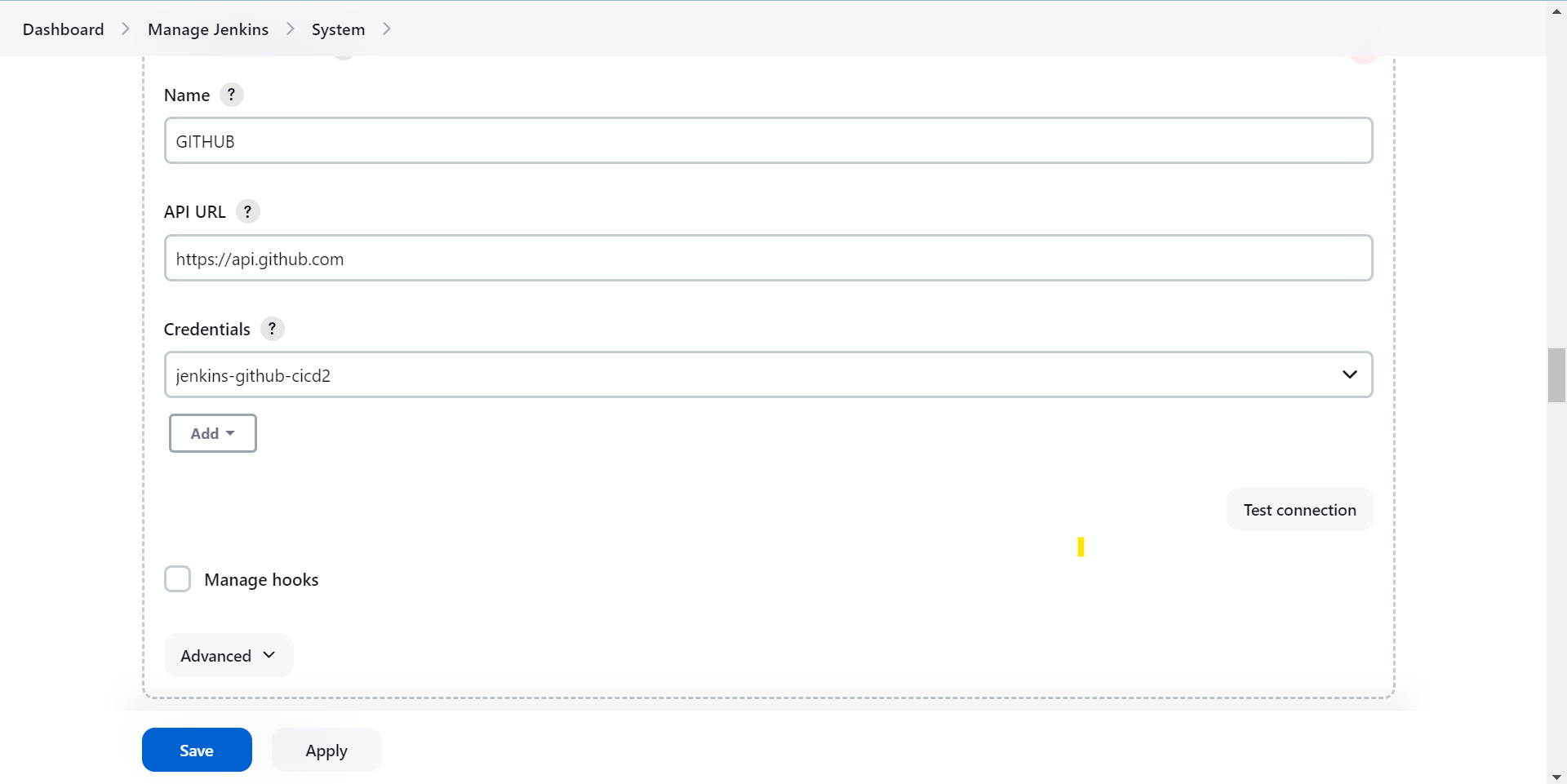
* **Now for integration we need GIT Client plugin on Jenkins and it is preinstalled while installation on Jenkins.**
* **Go to Jenkins System Configuration -> GITHUB -> Add Github Servers -> Name -> Credentials : SECRET TEXT (Personal Access Token) -> Test Connection**

****

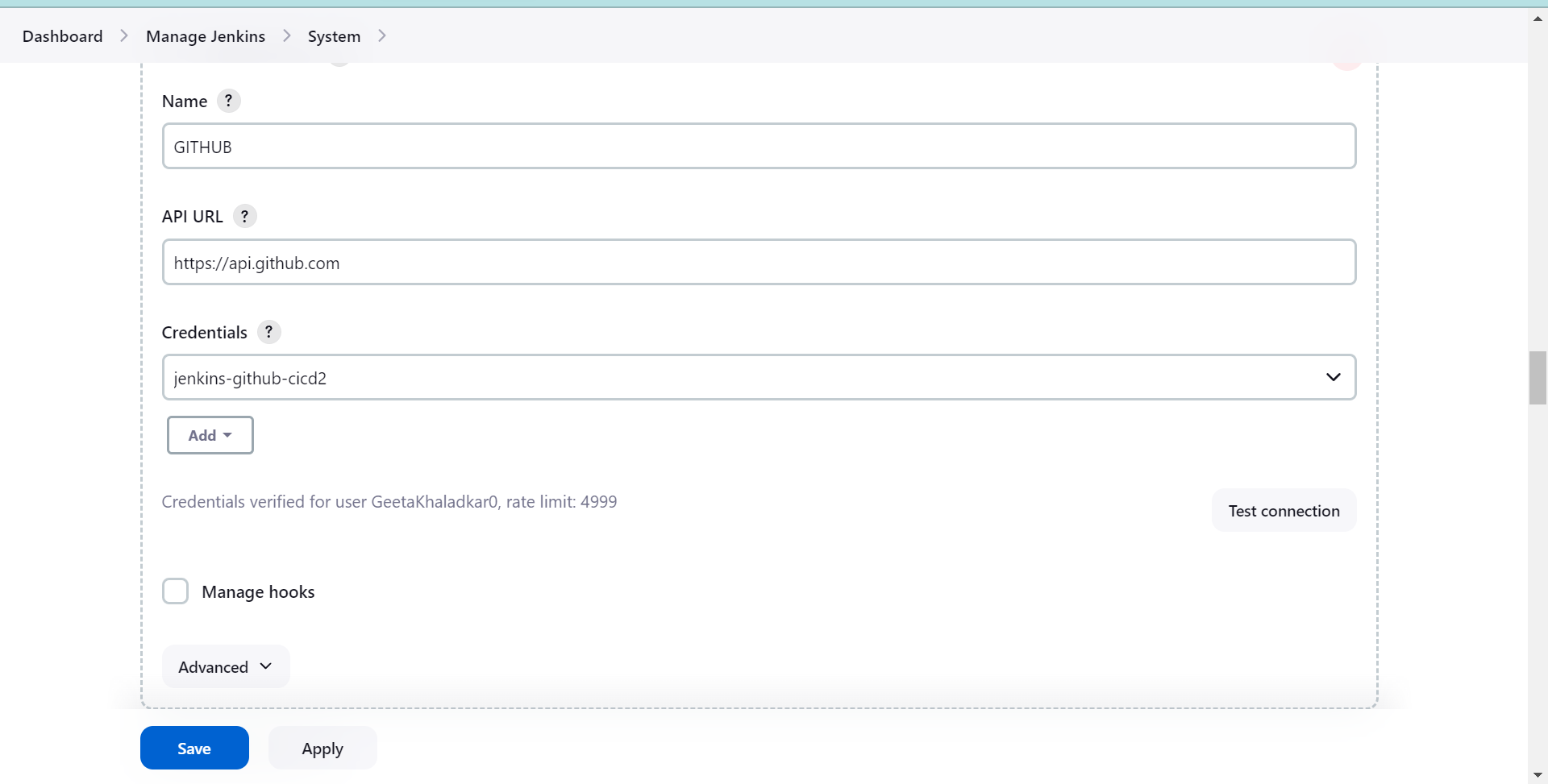
**Select Secret Text in kind and Paste Personal Access Token in secret .**

****

**Select Credentials from dropdown**

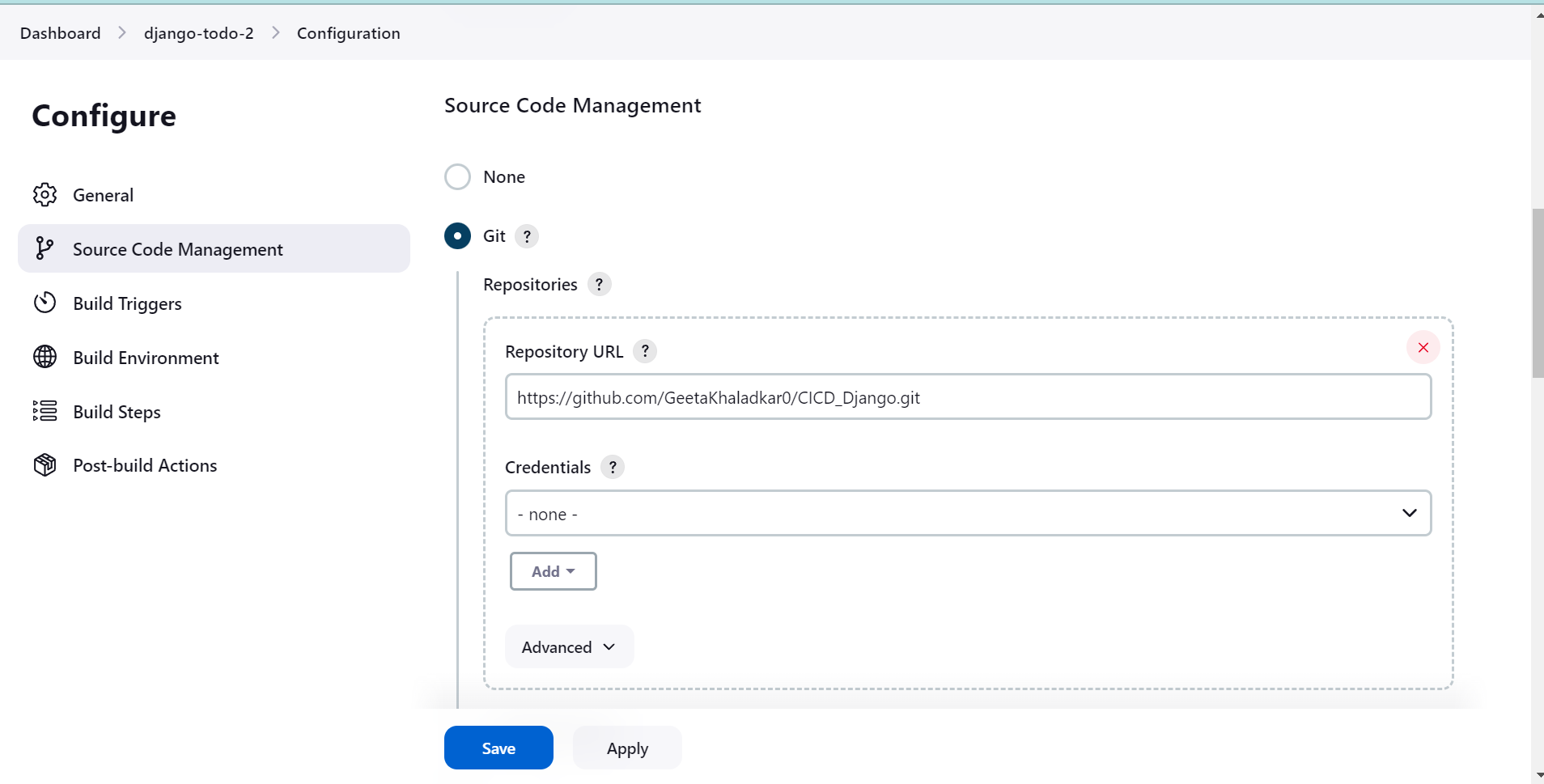
****

**Test the connection and save it.Only from PAT jenkins identifies GITHUB Email ID.**

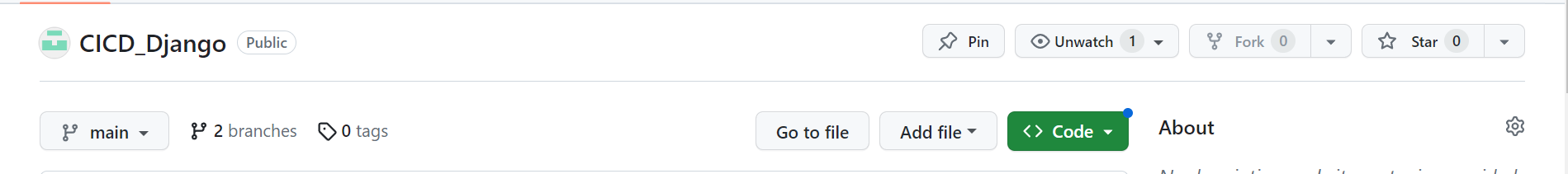
****

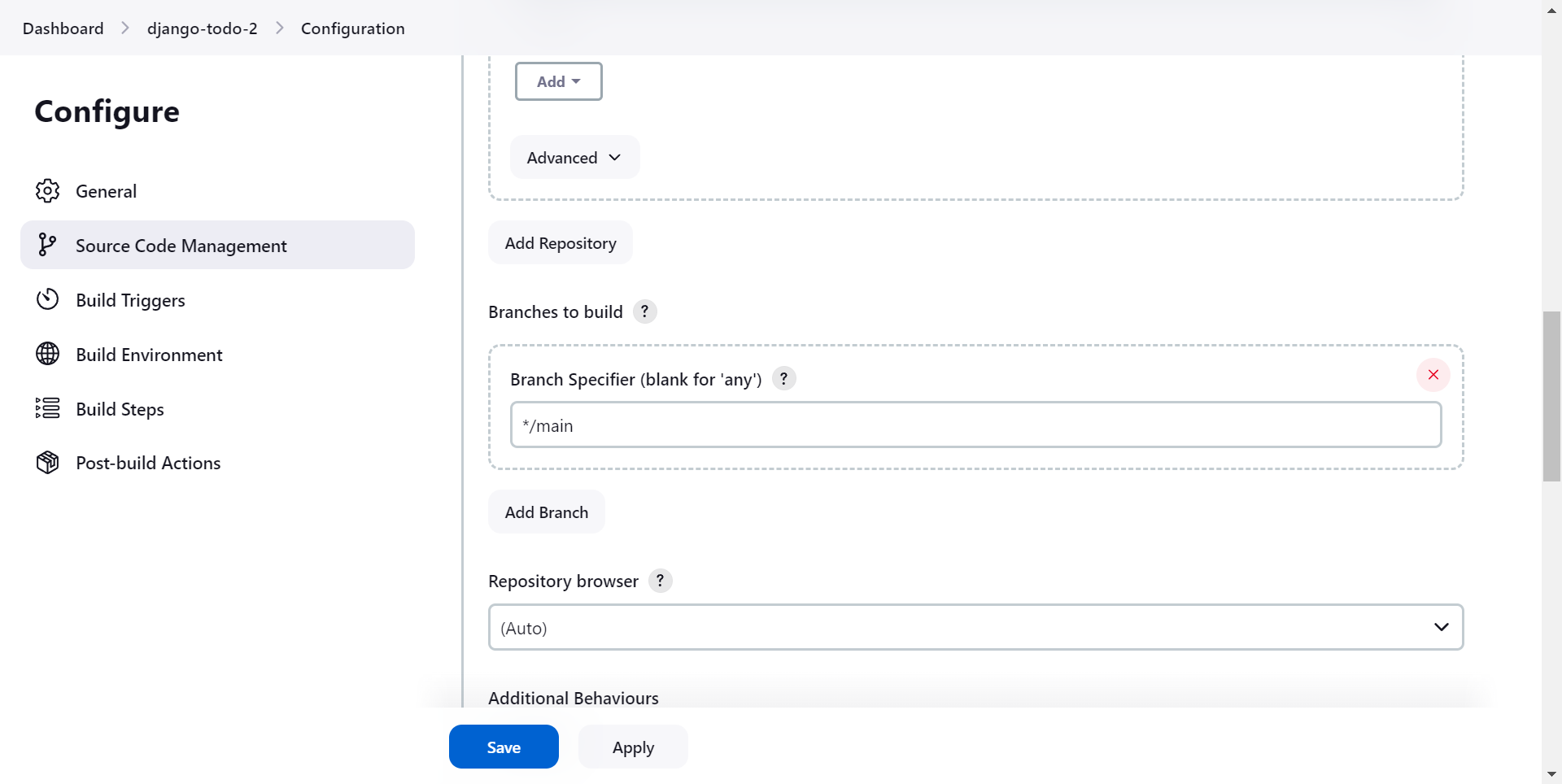
**Now we are making CICD Pipeline.**

* **Create new JOB**
* **In SCM : GIT**

****

**Select Branch where code is available on GITHUB.**

****

****

**Now we want to add Build Steps: Execute Shell**

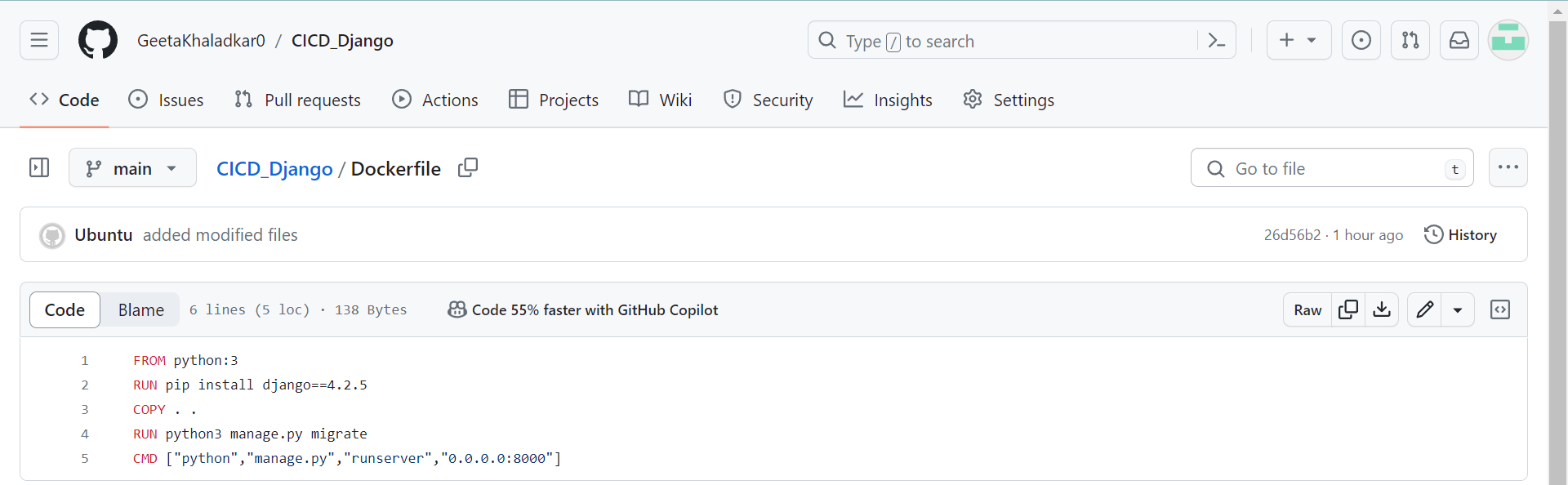
* **We already Given SCM:GITHUB URL. So code will access from there.**
* **Dockerfile is mandatory in project folder as it automates process.**

**In Dockerfile we create a container which having one Ubuntu OS with python and Django installation.**

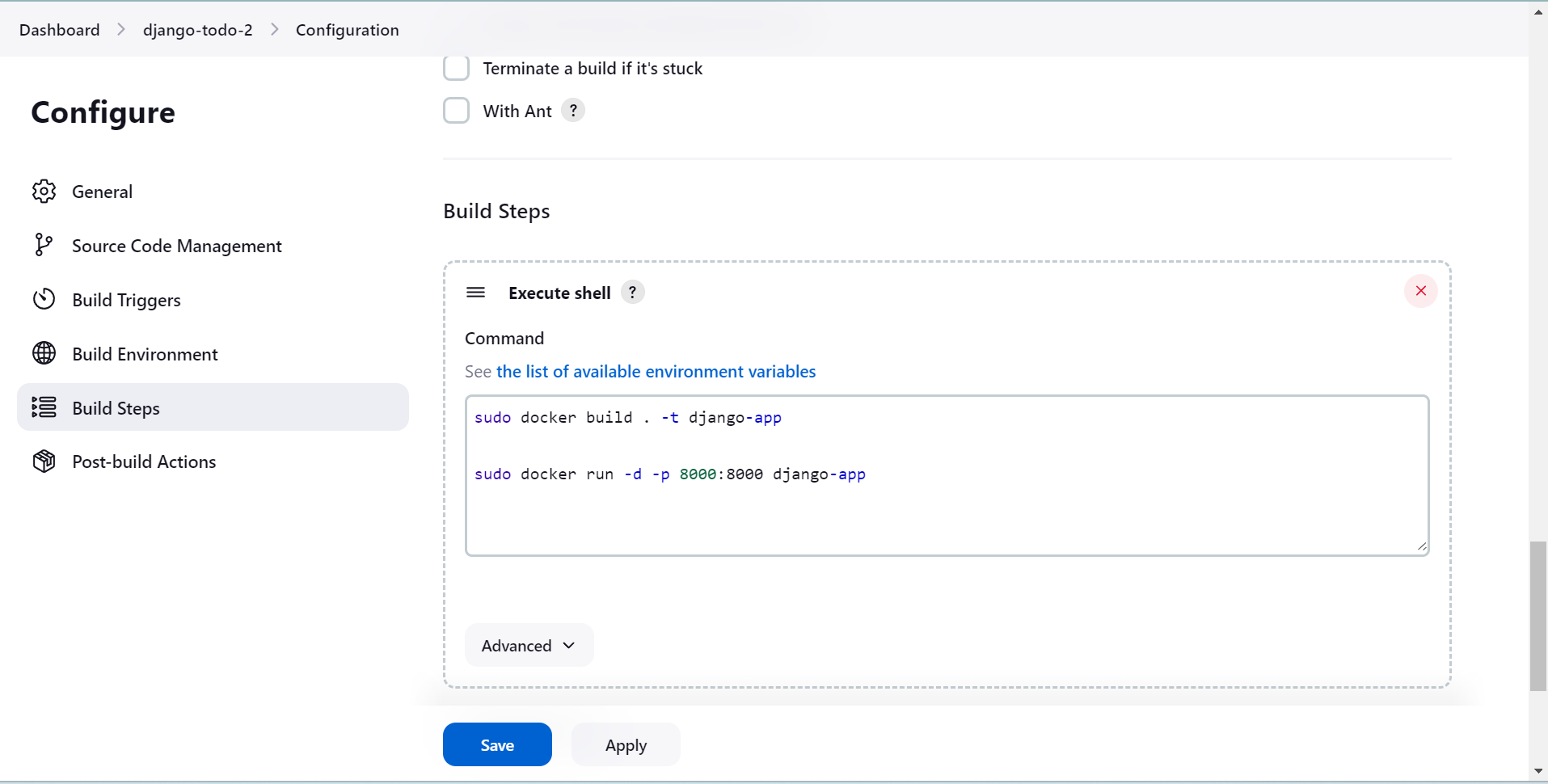
**Taking Project code from Ec2 server and paste in container.**

**Migrate manage.py file and then finally runserver.**

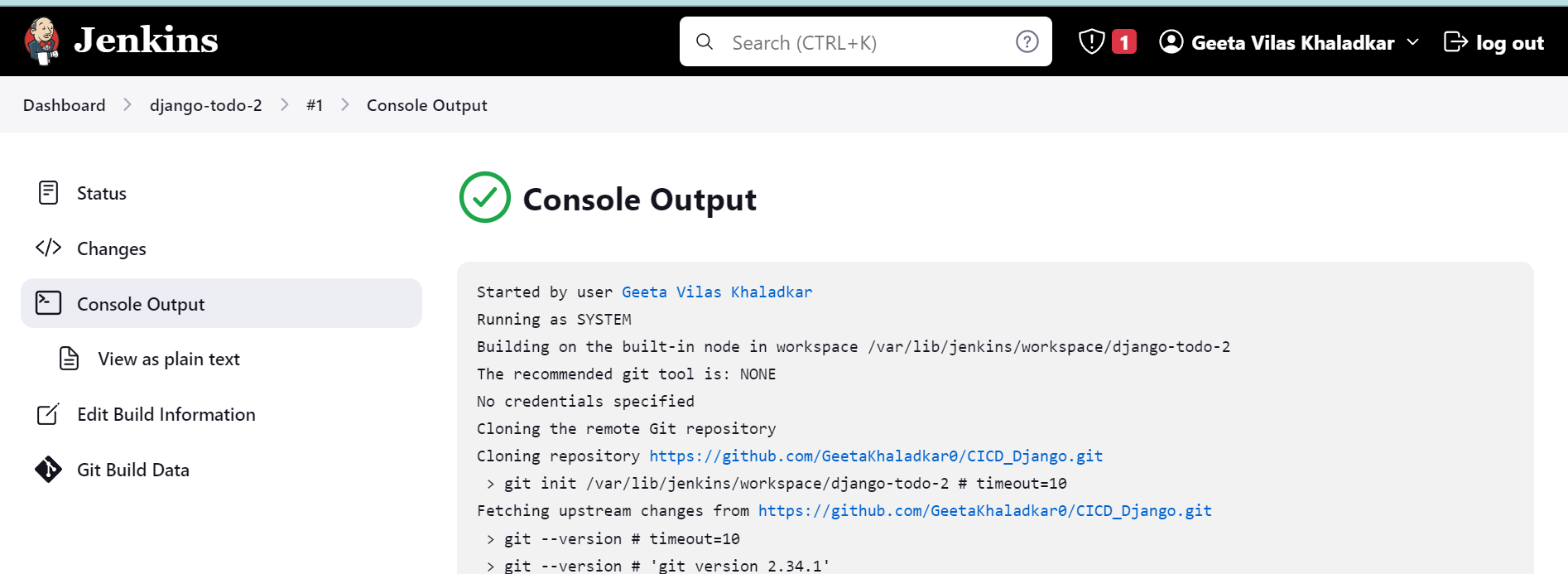
* **So only build Dockerfile and whole code automatically execute.**

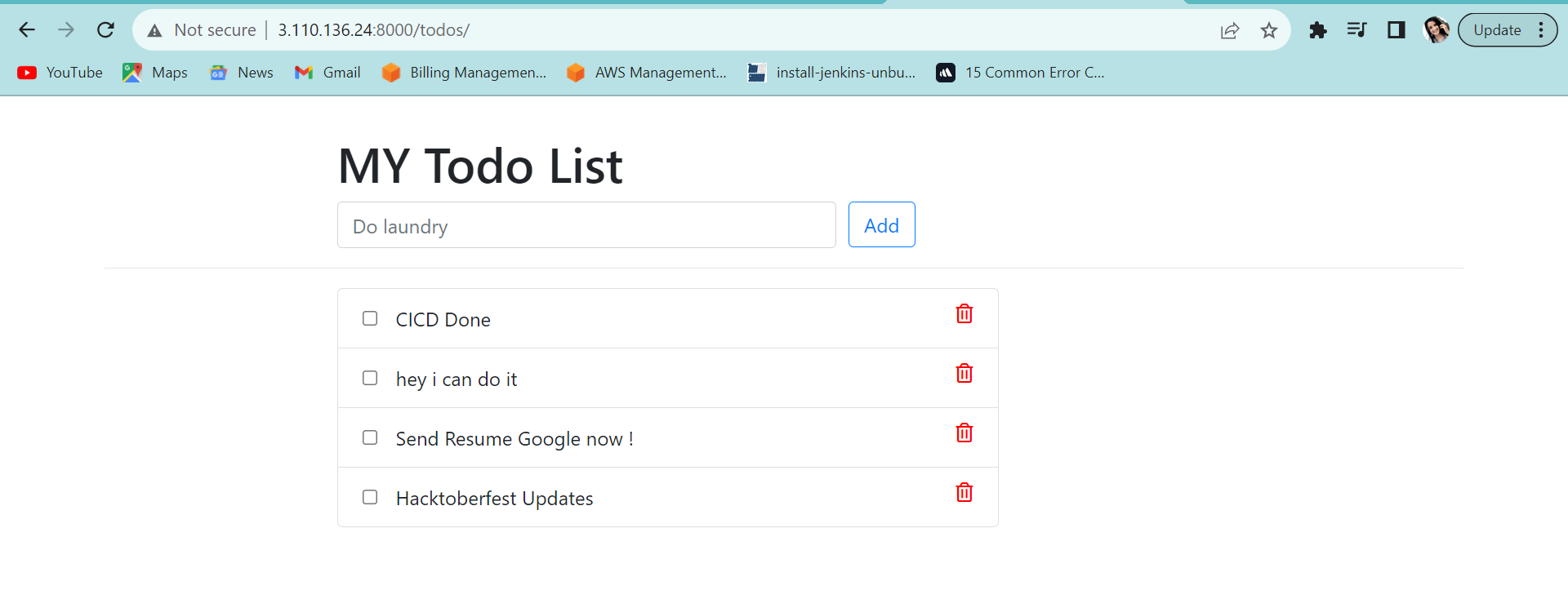
****

* **Write build steps in Jenkins Execute shell to execute code.**

****

* **Now , Build Now**

****

****

**Now , we are thinking that where is CICD?**

* **If Developer will change in code on local server and push it on GITHUB.**
* **Then Jenkins already integrated with GITHUB so, It is ready to build it.**
* **Only need to click on play button or build now.**

**Step 6: Run code on VSCODE via Docker-Compose**

**INSTALL YAML in VSCode**

* **Check python version**

PS D:\Practice\YML-dockercompose> **python --version**

**Python 3.11.5**

PS D:\Practice\YML-dockercompose> **pip3.11 install pyyaml**

**Defaulting to user installation because normal site-packages is not writeable**

**Requirement already satisfied: pyyaml in c:\users\lenovo\appdata\roaming\python\python311\site-packages (6.0.1)**

PS D:\Practice\YML-dockercompose> **pip show pyyaml**

**Name: PyYAML**

**Version: 6.0.1**

**Summary: YAML parser and emitter for Python**

**Home-page: https://pyyaml.org/**

**Author: Kirill Simonov**

**Author-email: xi@resolvent.net**

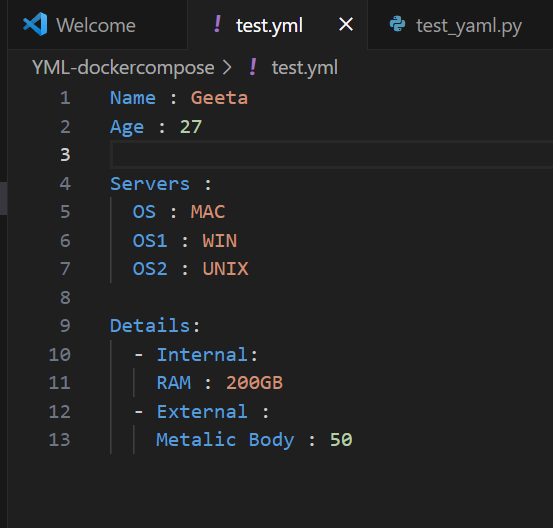
**License: MIT**

**Location: C:\Users\Lenovo\AppData\Roaming\Python\Python311\site-packages**

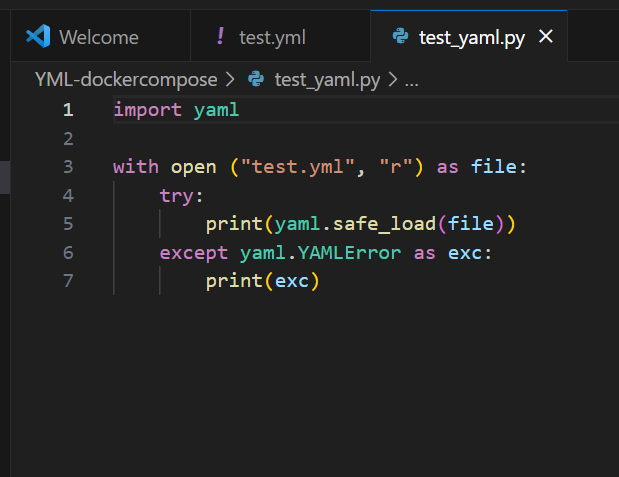
**Requires:**

**Required-by:**

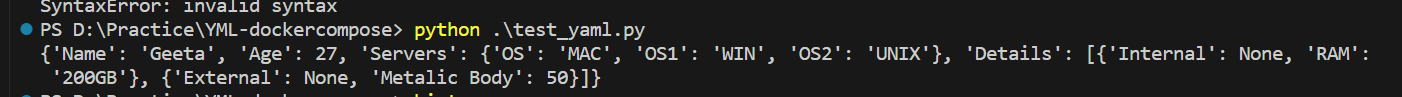
**DEMO: Create one yaml file and read it .**

****

**For reading yaml file create one python file and import yaml in that and read data from there.**

****

**For execution run : python <filename>.py**

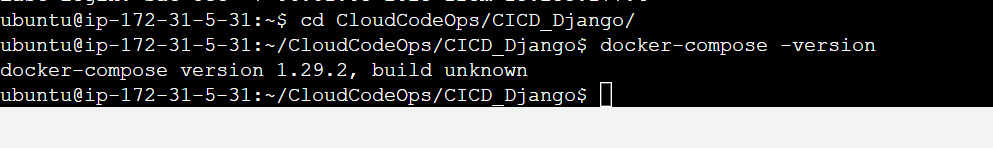
****

**Docker Compose:**

* **It is one yaml file which basically use for joining or connecting with containers.**
* **Check Docker-compose is in your system or not**

**Docker-compose -v**

**Sudo apt install docker-compose**

****

**Now create Docker compose file**

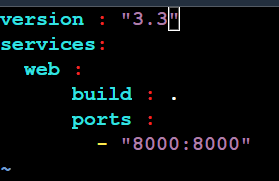
**Vi docker-compose.yml**

**Version is “3.3”**

**Docker-compose having services .Now there is ony one service :web**

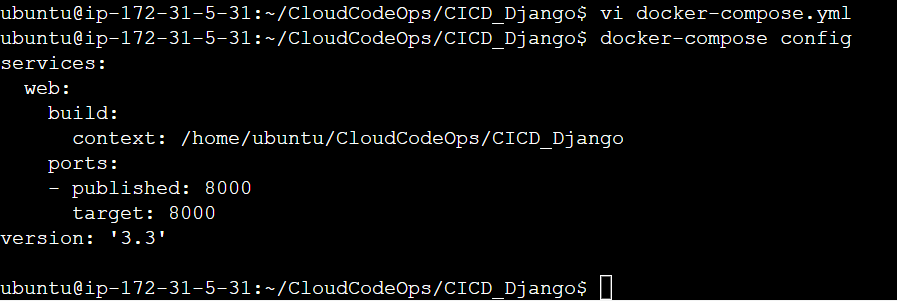
**Web service is build on current folder where project files are present so build (.)**

**Web services will run on port 8000 so, It must bind with server port to container port.**

****

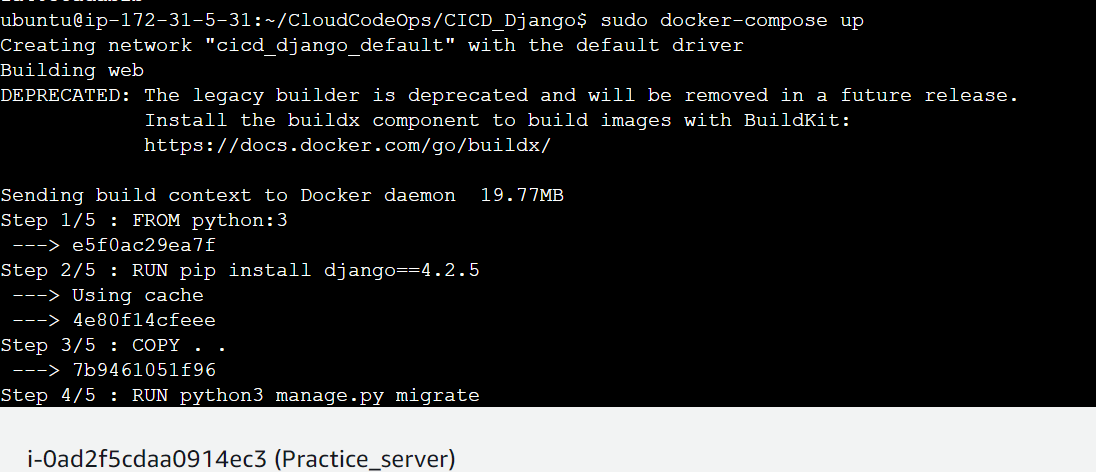
**Check file is correct or not**

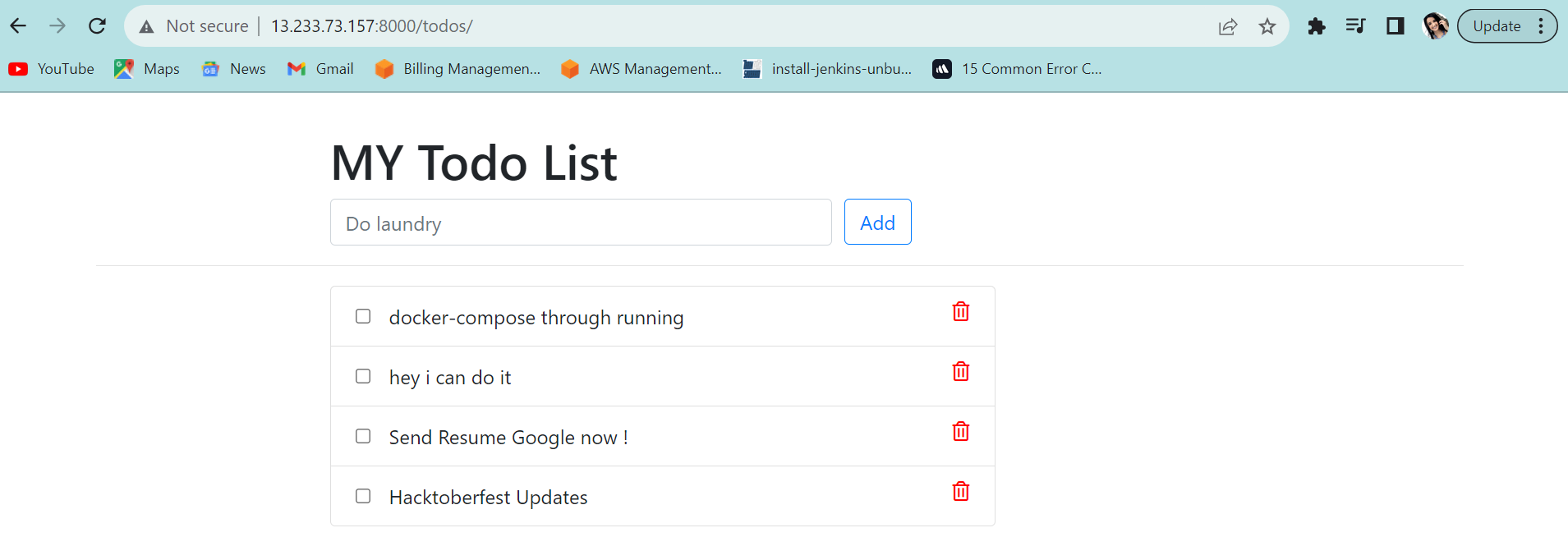
**Docker-compose config**

****

**Run docker compose**

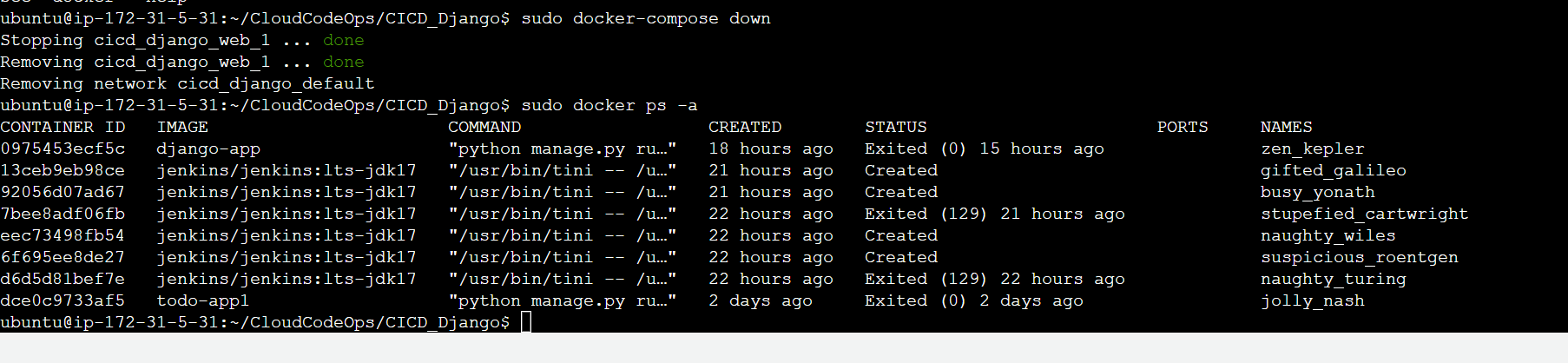
**sudo docker-compose up**

****

****

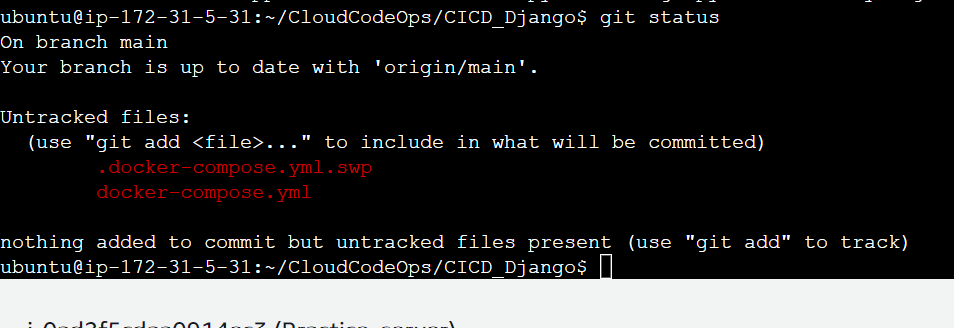
**Free the container which allocating port :8000**

**Sudo docker-compose down**

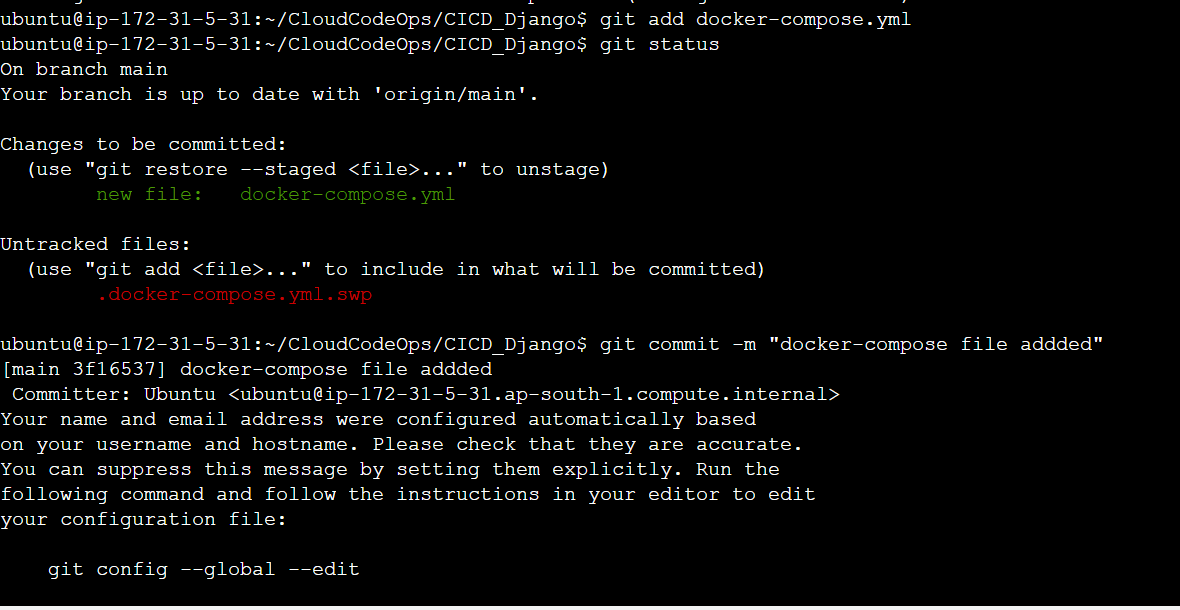
****

**Step 7: Run code Jenkins via Docker-Compose**

* **Check the git status**

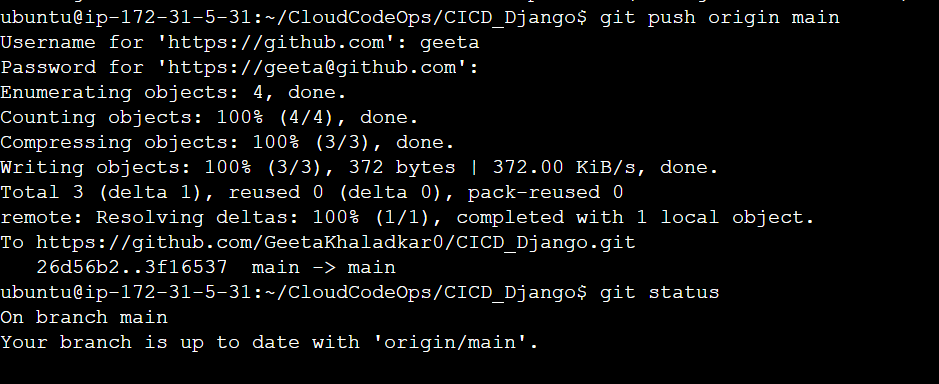
****

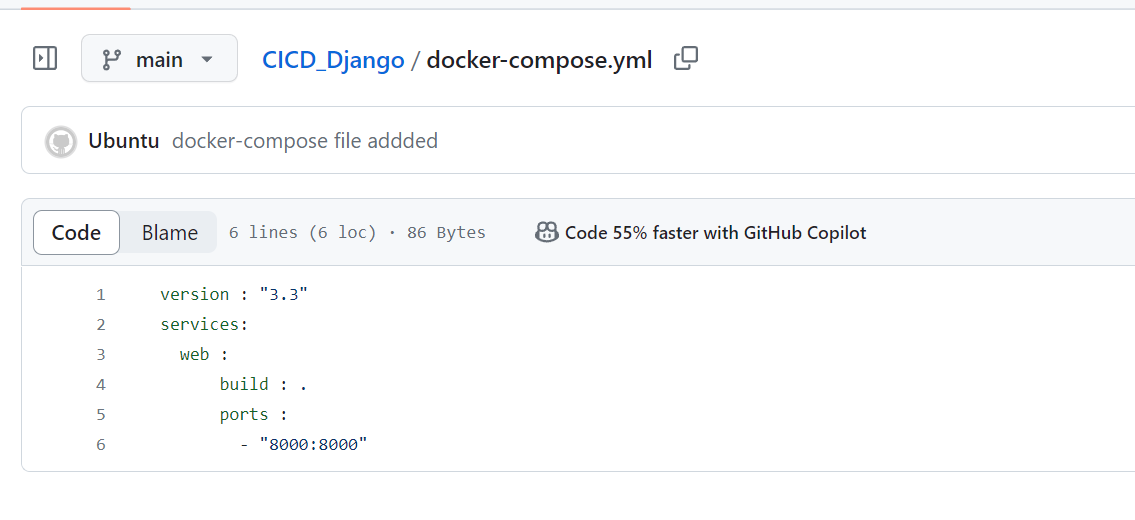
* **Push docker-compose.yml file on github**

****

**Add Username : Geeta Password : Personal Access Token**

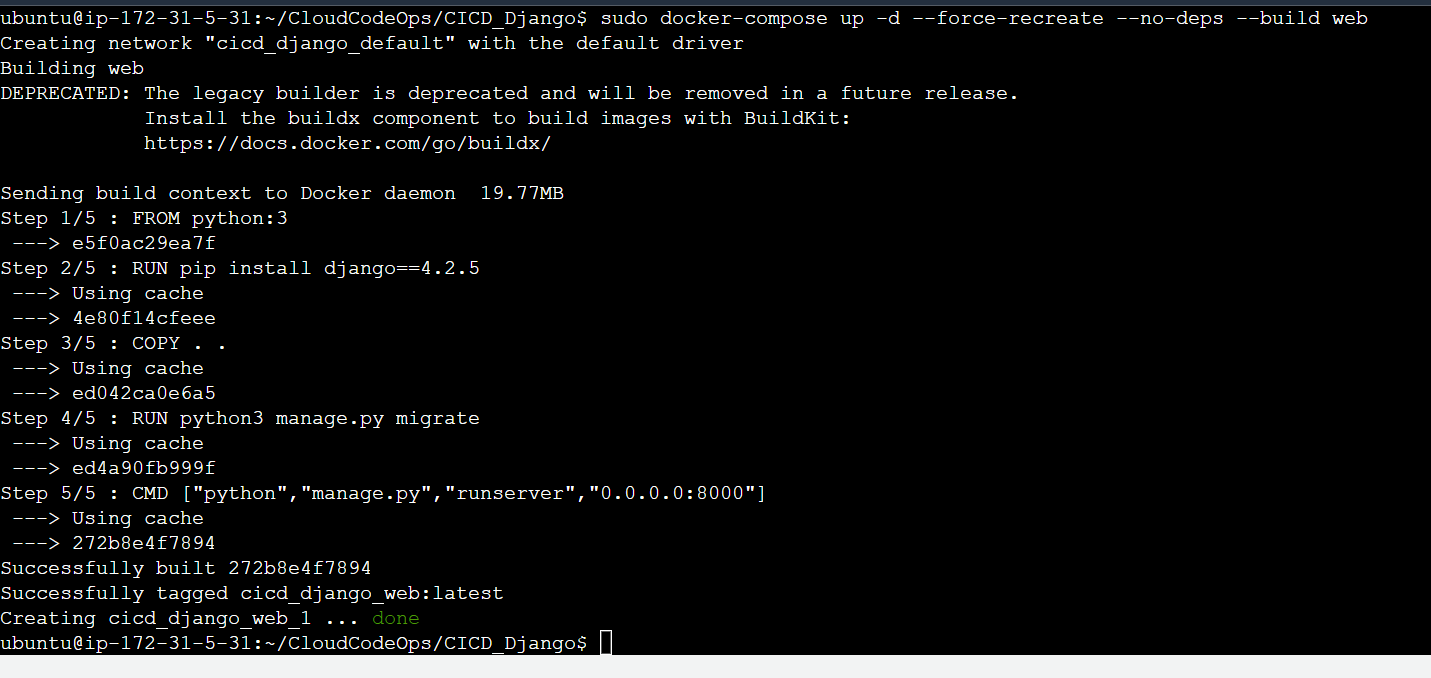
**ghp\_fvDfPNP6iWt7fXIz7s2h2tVzMP7fOS3v80BL**

****

****

**Sudo docker-compose up -d –force-recreate –no-deps –build web**

* **(docker-compose up ) means run the file**
* **(--force-recreate) means forcefully recreate the docker container.**
* **-(--no-deps) means no dependencies**
* **(--build) means build (web) services which name written in docker-compose.yml.**

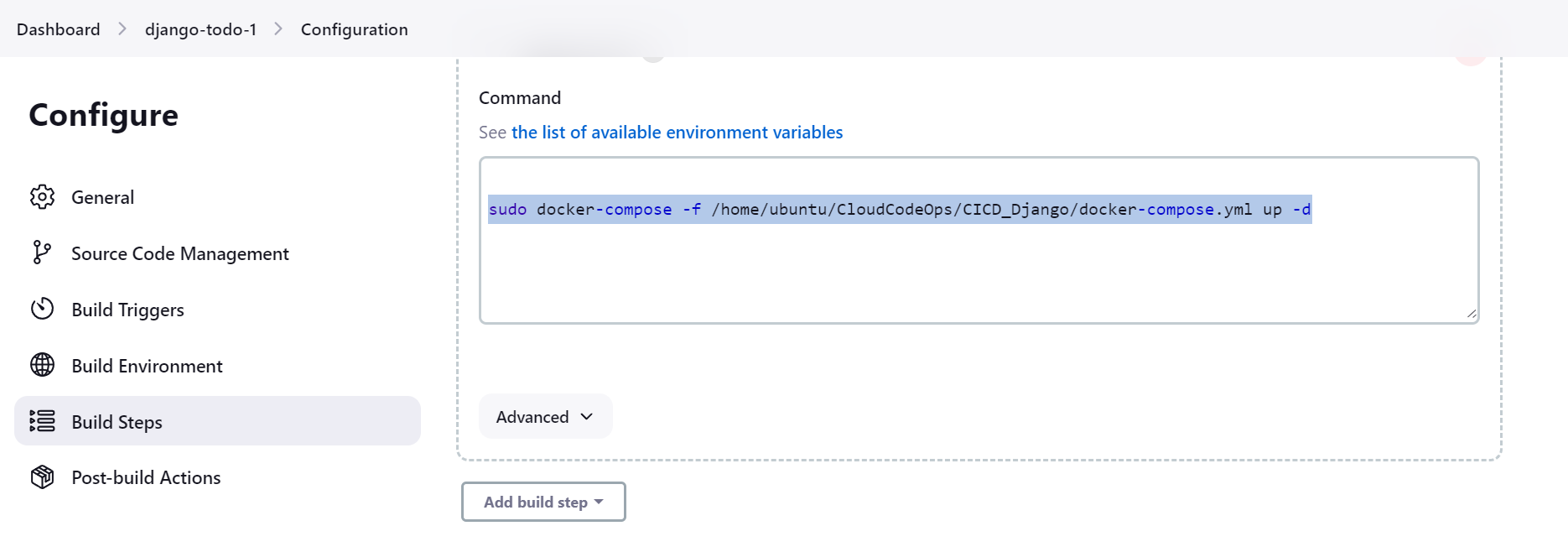
****

* **Go on Jenkins -> execute shell ->**

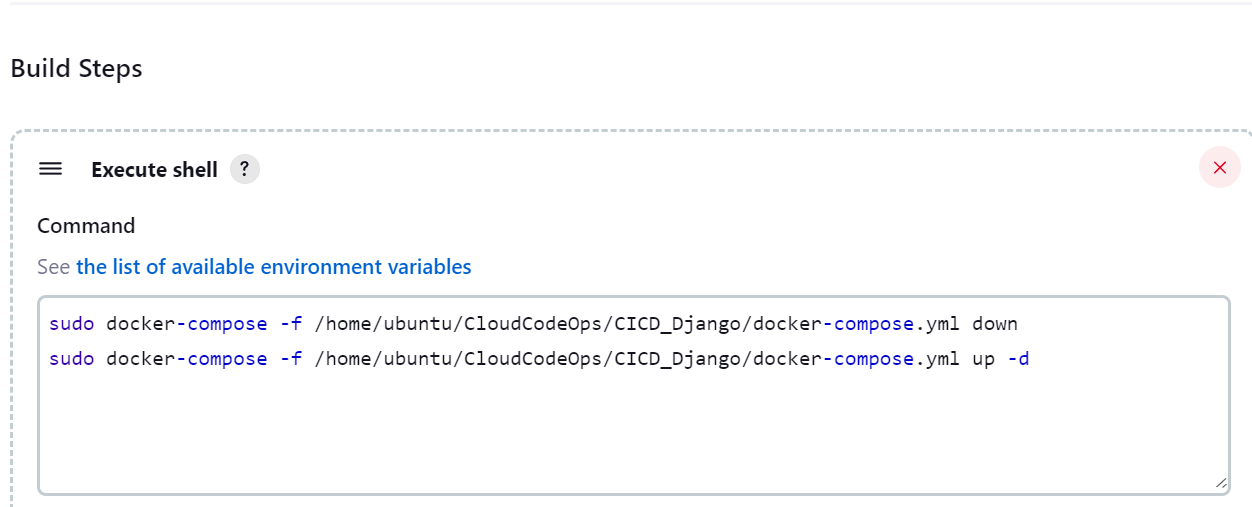
**Sudo docker-compose -f /home/ubuntu/CloudCodeOps/CICD\_Django/docker-compose.yml up -d**

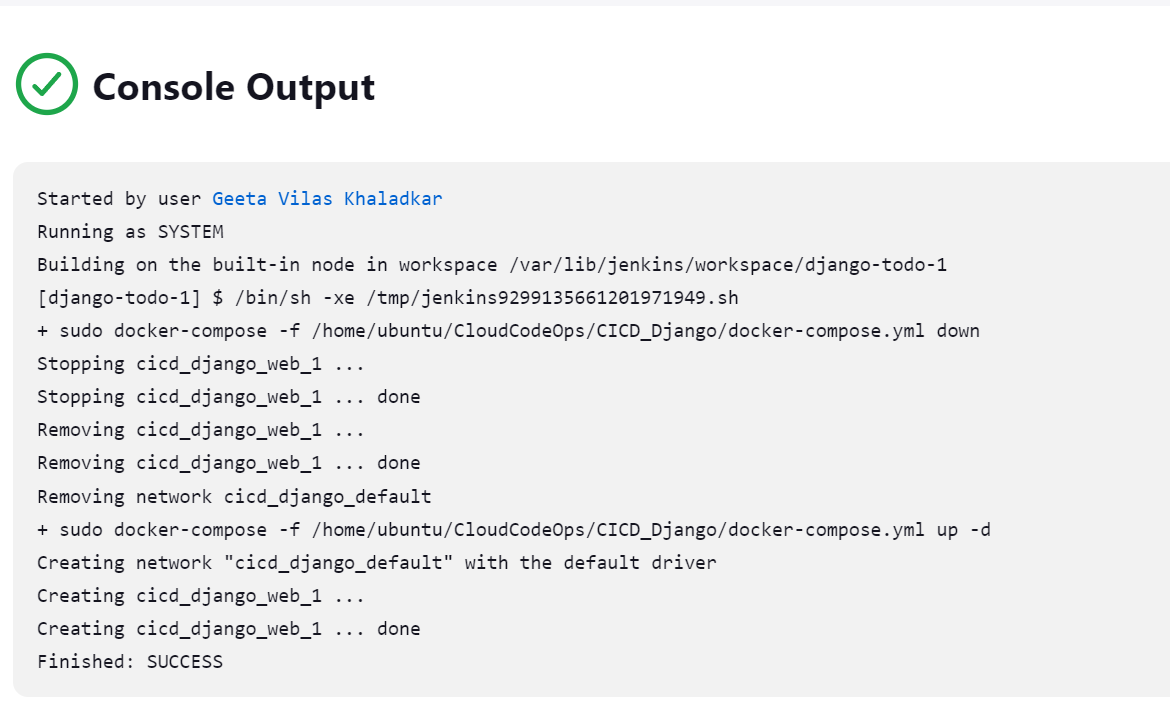
**If we don’t want to continuously free port before building the project then first down container**

**Sudo docker-compose -f /home/ubuntu/CloudCodeOps/CICD\_Django/docker-compose.yml down**

****

**Don’t need to free port manually so, It will down the container first and then run the container.**

****

****

**In CI/CD integration,**

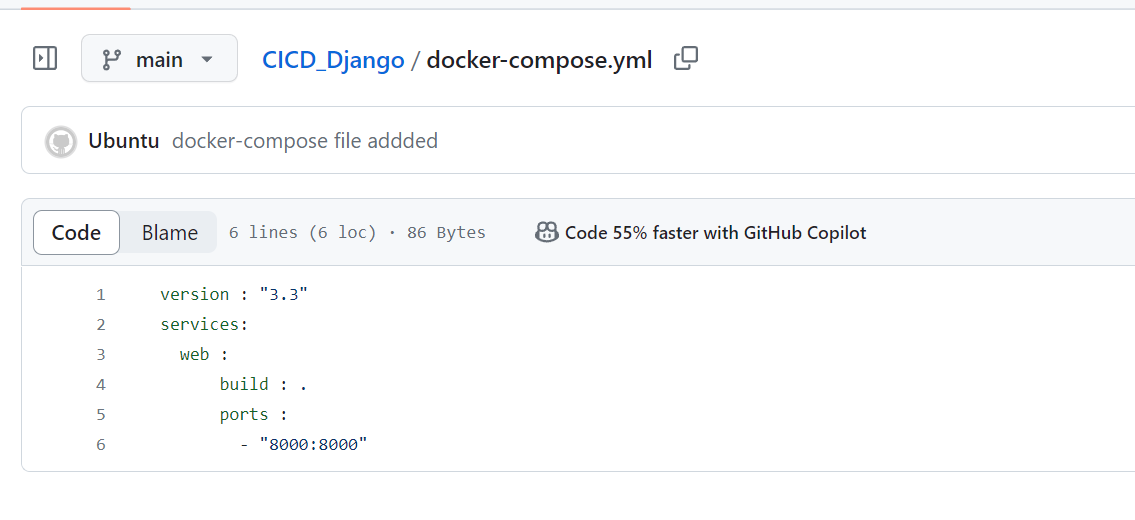
**Jenkins will pull code from GITHUB where GITHUB has a docker-compose file.**

**Docker-compose having services. Now there is only one service: web**

**Web service is built on the current folder where project files are present so build (.)**

**Web services will run on port 8000, so It must bind with the server port to the container port.**

**In execution, the shell only runs the docker-compose file it builds webapp.**

****