**Django Notes App CI/CD Pipeline**

**Project Overview**

This repository hosts the CI/CD pipeline for automating the deployment of a Django Notes App.

The pipeline leverages **Jenkins, Docker, and AWS** services to facilitate continuous integration and continuous deployment processes. Below is an explanation of the various stages in the pipeline.

**Stages:**

1. **Clone Code:**
   * The pipeline begins by cloning the Django Notes App code from the specified GitHub repository.
2. **Build:**
   * A Docker image, named **my-app**, is built using the Django Notes App source code.
3. **Push to Docker Hub:**
   * The Docker image is tagged and securely pushed to Docker Hub using the provided credentials.
4. **Deploy:**
   * The deployment stage utilizes Docker Compose to manage the deployment of the Django Notes App on an AWS EC2 instance.

**Prerequisites**

Before running the pipeline, ensure that the following prerequisites are met:

* Jenkins is set up and properly configured.
* Docker is installed on the Jenkins server.
* AWS credentials are configured within Jenkins.
* Docker Hub credentials are set up in Jenkins.
* Make EC2 server for jenkins and clone the code in this server.
* **Building the code:**

-install docker in ec2

-give permission to ubuntu to use docker.

-build the docker file to make docker image

* **Jenkins installation:**

-install java and install jenkins

-check jenkins by pasting publicip:8080

-enable inbound rules for 8080 port number

* Now jenkins is ready.
* **Create a pipeline job** and add github url so that the jenkins could get the code.

To automate use github webhooks so that whenever there is commit on github the jenkins will automatically starts building it.

* Use declarative pipeline method in jenkins using groovy syntax which includes the stages and inside stages there are steps too.
* **We need 4 stages is this project:**

-stage1: Cloning the code

-stage2: Building the code

-stage3: Pushing the image to docker hub

-stage4: Deploying

* After this when we deploy it is correct. But whenever we try again to deploy it fails saything that this port 8000 is already allocated.  
  to remove this problem we use **docker compose.**

Here docker compose will help to down the running port and builds a new one. So that there is no repition in same port.

* **In order to push the build image to dockerhub we need to login in docker hub. So, to keep the docker credintials safe we use credential manager in jenkins**

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

pipeline {

agent any

stages{

stage("Clone Code"){

steps {

echo "Cloning the code"

git url:"https://github.com/LondheShubham153/django-notes-app.git", branch: "main"

}

}

stage("Build"){

steps {

echo "Building the image"

sh "docker build -t my-note-app ."

}

}

stage("Push to Docker Hub"){

steps {

echo "Pushing the image to docker hub"

withCredentials([usernamePassword(credentialsId:"dockerHub",passwordVariable:"dockerHubPass",usernameVariable:"dockerHubUser")]){

sh "docker tag my-note-app ${env.dockerHubUser}/my-note-app:latest"

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"

sh "docker-compose down && docker-compose up -d"

}

}

}

}