🚀 Deploying Weather App with Jenkins and Docker: An End-to-End project🌦️

Hey there, tech enthusiasts! In this blog, I'll walk you through the exciting journey of deploying a Python-based Flask Weather App using Jenkins and Docker. Whether you're a beginner or an experienced DevOps engineer, this guide will help you understand the process step-by-step. Let's dive in!

Project Overview🛠️

Our goal is to:

➢ Set up a Jenkins pipeline to automate the deployment process.
➢ Use Docker to containerize our Weather App.
➢ Deploy the app and make it available for users to fetch weather information.

By the end of this blog, you'll have a fully functional Weather App running in a Docker container, automated through Jenkins! 🌍

Step 1: Setting Up Jenkins🖥️

First, let's set up Jenkins on your server. If you haven't installed Jenkins yet, follow these simple steps:

1. Install Jenkins:
   - On Ubuntu, use:

     sudo apt update
     sudo apt install jenkins

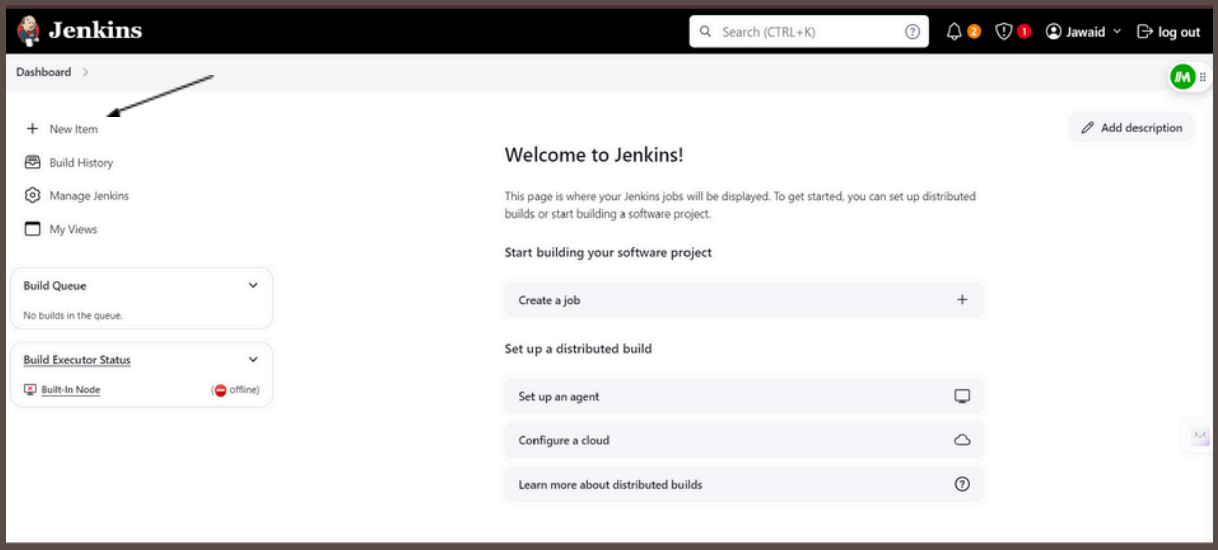
2. Start Jenkins:

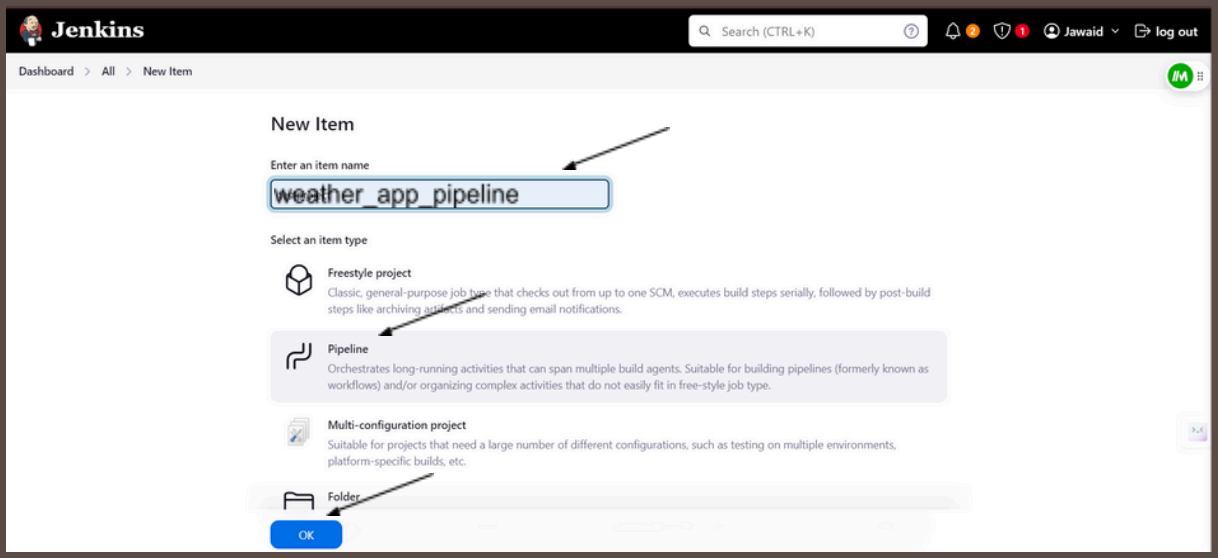
   sudo systemctl start jenkins

3. Access Jenkins:
   - Open your browser and navigate to `http://your-server-ip:8080`.
   - Unlock Jenkins using the password from `/var/lib/jenkins/secrets/initialAdminPassword`.

4. Install Plugins:
   - Install recommended plugins during the initial setup.

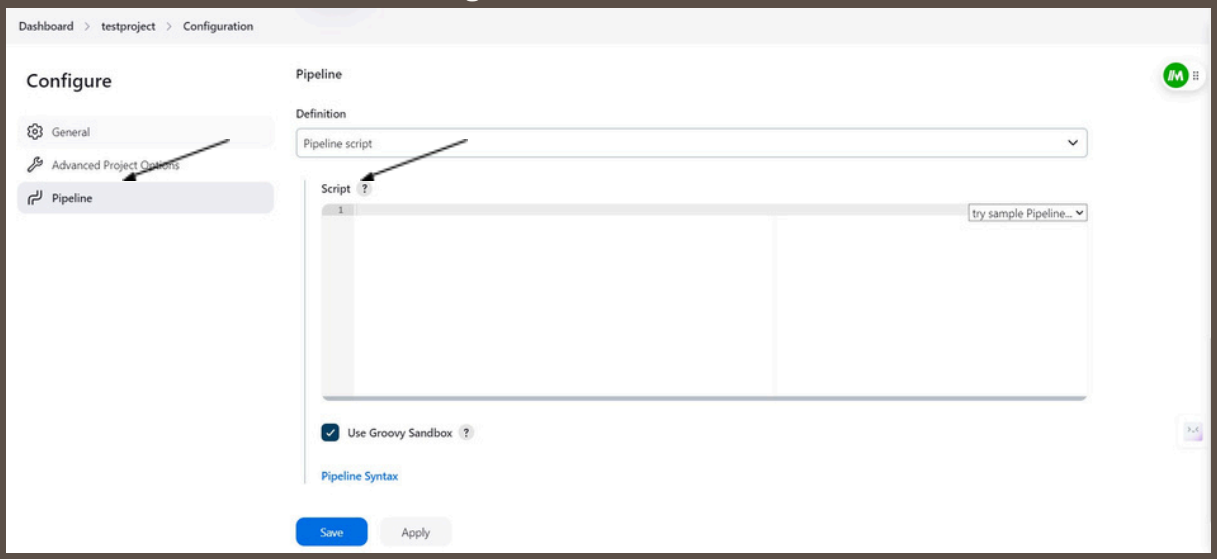
5. Create a New Pipeline:
   - Go to Jenkins Dashboard ➡ New Item ➡ Pipeline.





Step 2: Writing the Jenkinsfile 📝

Now, let's create a `Jenkinsfile` to define the pipeline. This file will automate the steps to clone the code, build the Docker image, and run the container.



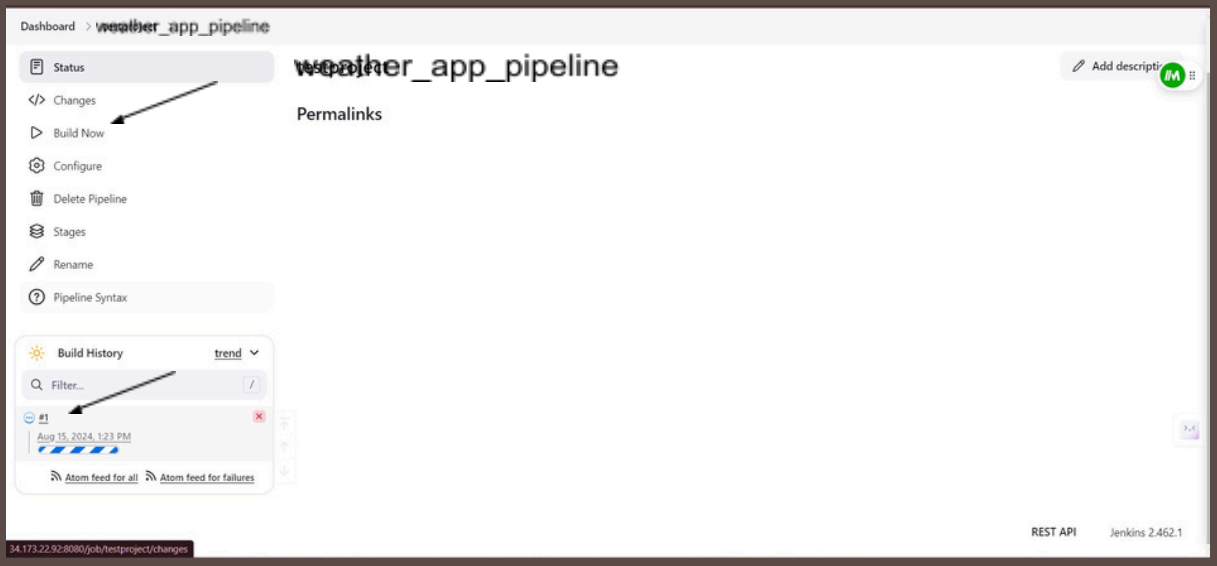

```
pipeline {
    Agent any

    stages {
        stage("clone_code") {
            steps {
                script {
                    sh 'git clone -b main https://github.com/JawaidAkhtar/weather_app.git'
                    echo "Code cloned successfully from GitHub"
                }
            }
        }
        stage("build_docker_image") {
            steps {
                script {
                    // Print the current user and directory for debugging
                    sh 'whoami'
                    sh 'pwd'

                    // Change directory using dir block
                    dir('weather_app') {
                        sh 'docker build -t weather-app-v2 .'
                    }
                    echo "Successfully built the Docker image named weather-app-v2"
                }
            }
        }
        stage("run_container") {
            steps {
```

```
                script {
                    sh 'docker run -d -p 5000:5000 --name myweather-app weather-app-v2:latest'
                    echo "myweather-app is running on port 5000"
                }
            }
        }
    }
}
```

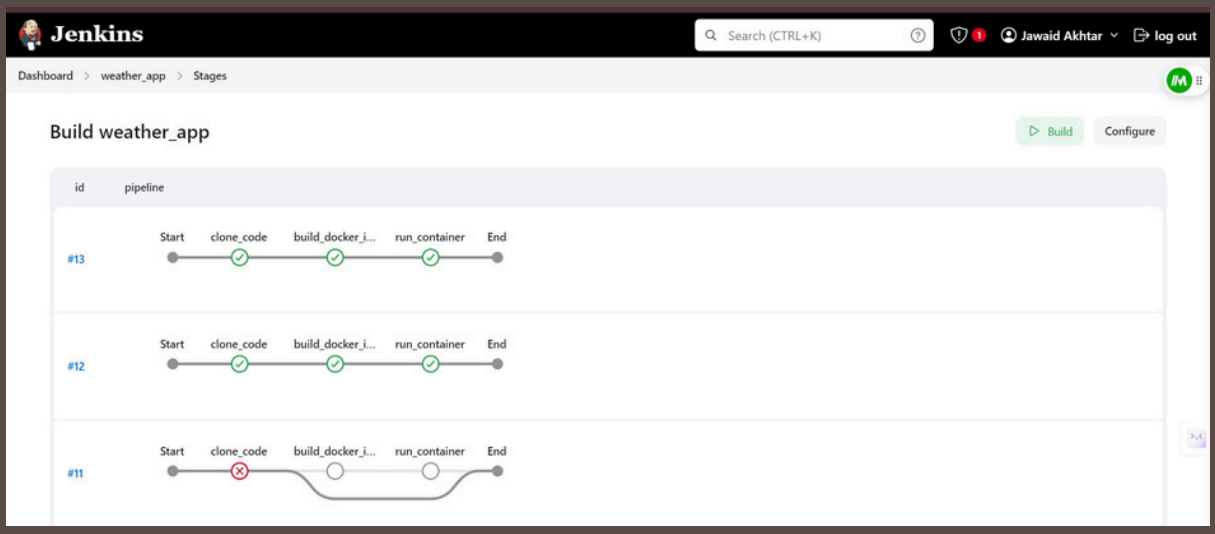
- clone_code: Clones the Weather App repository from GitHub.
- build_docker_image: Builds a Docker image named `weather-app-v2:latest`.
- run_container: Runs the Docker container and maps port 5000.

Once you've set up the Jenkinsfile, trigger the pipeline:

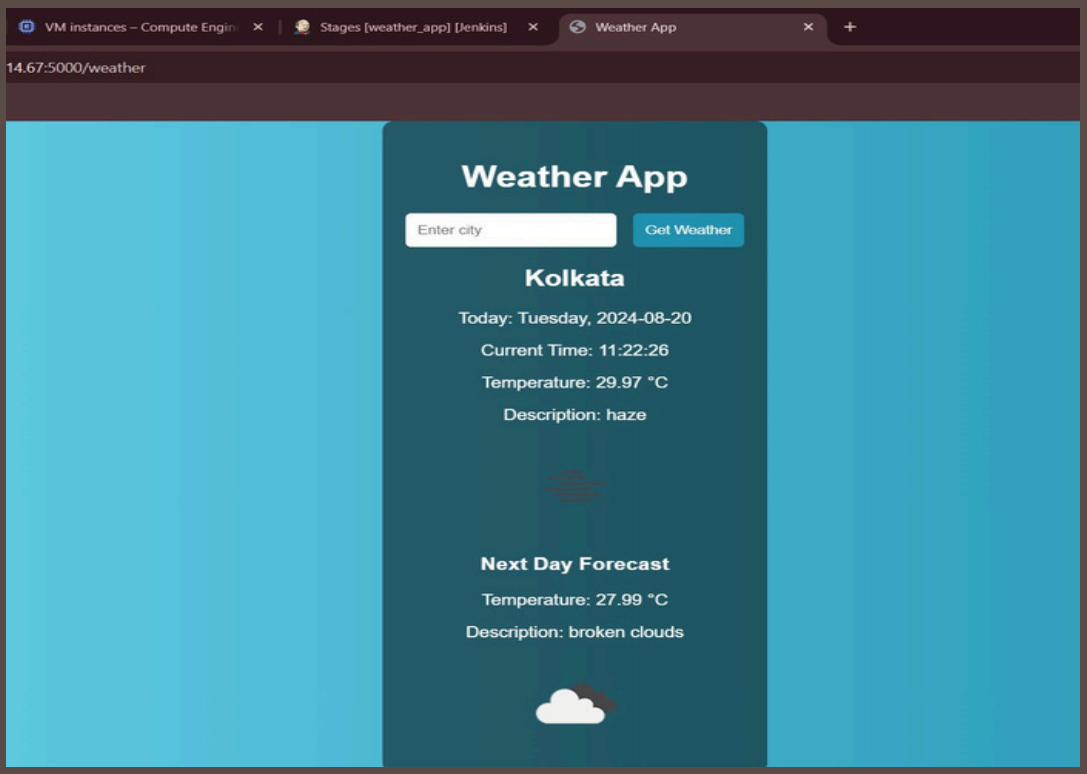
1. Go to Jenkins Dashboard ➡ weather_app_pipeline.
2. Click **Build Now**. 🛠️



As you can see, stages `clone_code`, `build_docker_image`, and `run_container` have all passed successfully. 🎉



Step 4: Accessing the Weather App



- Displays the current weather, including temperature, description, and time.
- Provides a next-day weather forecast.

During the pipeline run, Docker commands are executed to build and manage the container. You can verify the running container using:

docker ps

This command lists all running containers. You'll see something like this:

CONTAINERIDIMAGE COMMAND CREATED STATUS
PORTS NAMES
b7ff68cb9472 weather-app-v2:latest "pythonapp.py"9secondsago Up8seconds
0.0.0.0:5000->5000/tcp myweather-app
```

Docker Tips:
- Use `docker images` to list all Docker images.
- Use `docker stop <container_id>` to stop a running container.

Congrats! You've successfully deployed a Weather App using Jenkins and Docker. This end-to-end automation not only simplifies the deployment process but also ensures that your app is always up-to-date with the latest code changes.

Feel free to explore more by adding additional stages, like testing or deploying to a production environment. The sky's the limit! ☁️
What's Next? 🚀
- Share your progress on LinkedIn with #DevOps and #CI/CD hashtags.
- Expand the project by integrating more complex workflows in Jenkins.
- Stay tuned for more blogs on DevOps and CI/CD automation.

Thanks for following along! 😊 Feel free to drop your comments or questions below. Until next time, happy coding! 👨‍💻
```