

Flask Hello World API

This project is a simple Flask-based API that returns a "Hello, World!" message and allows for dynamic greetings using user input. It's an ideal starting point for developers learning Flask.

Key Features

- **Hello World Endpoint:** Returns a static "Hello, World!" message.
- **Dynamic Greeting:** Responds with personalized greetings based on user input (e.g., `/hello/John` returns "Hello, John!").
- **Automated Testing:** Includes tests using pytest to ensure endpoint functionality.

Requirements

- Python 3.6+
- Flask 2.0+
- pytest 6.0+

Installation

Clone the repository:

```
git clone https://github.com/manjyyot/flask-hello-world-api.git
cd flask-hello-world-api
```

1.

Set up a virtual environment:

```
python -m venv venv
source venv/bin/activate # macOS/Linux
.\venv\Scripts\activate # Windows
```

2.

Install dependencies:

```
pip install -r requirements.txt
```

3.

Running the Application

Start the Flask server:

```
flask run
```

- 1.
2. Visit <http://127.0.0.1:5000/> to see the API in action.

API Endpoints

- **GET /:** Returns "Hello, World!"
- **GET /hello:** Returns "Hello!"
- **GET /hello/{name}:** Returns a personalized greeting.

Running Tests

To run tests, use:

```
pytest
```

AWS EC2 - Installing Jenkins

1. **Launch an EC2 instance:**
 - Create a new EC2 instance using your preferred operating system (e.g., Ubuntu 20.04).
 - Ensure you have appropriate security groups set up to allow SSH (port 22) and HTTP (port 8080) access.

Connect to your EC2 instance:

```
ssh -i "your-key.pem" ubuntu@your-ec2-public-ip
```

2.

Update your system:

```
sudo apt update
sudo apt upgrade -y
```

3.

Install Java (Jenkins requires Java):

```
sudo apt install openjdk-11-jdk -y
```

4.

Add Jenkins repository and install Jenkins:

```
sudo wget -q -O - https://pkg.jenkins.io/jenkins.io.key | sudo apt-key
add -
sudo sh -c 'echo deb http://pkg.jenkins.io/debian/ / >
/etc/apt/sources.list.d/jenkins.list'
sudo apt update
sudo apt install jenkins -y
```

5.

Start Jenkins:

```
sudo systemctl start jenkins
sudo systemctl enable jenkins
```

6.

7. Open Jenkins on port 8080:

- Visit <http://<your-ec2-public-ip>:8080> in your browser.

You'll be prompted for the Jenkins unlock key, which can be found by running:

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

-
-
-
-
-
-
-
8. **Install suggested plugins** and set up your Jenkins instance by following the on-screen instructions.

```
ubuntu@ip-10-0-0-59:~$ jenkins --version
2.496
```

Jenkins Pipeline Setup

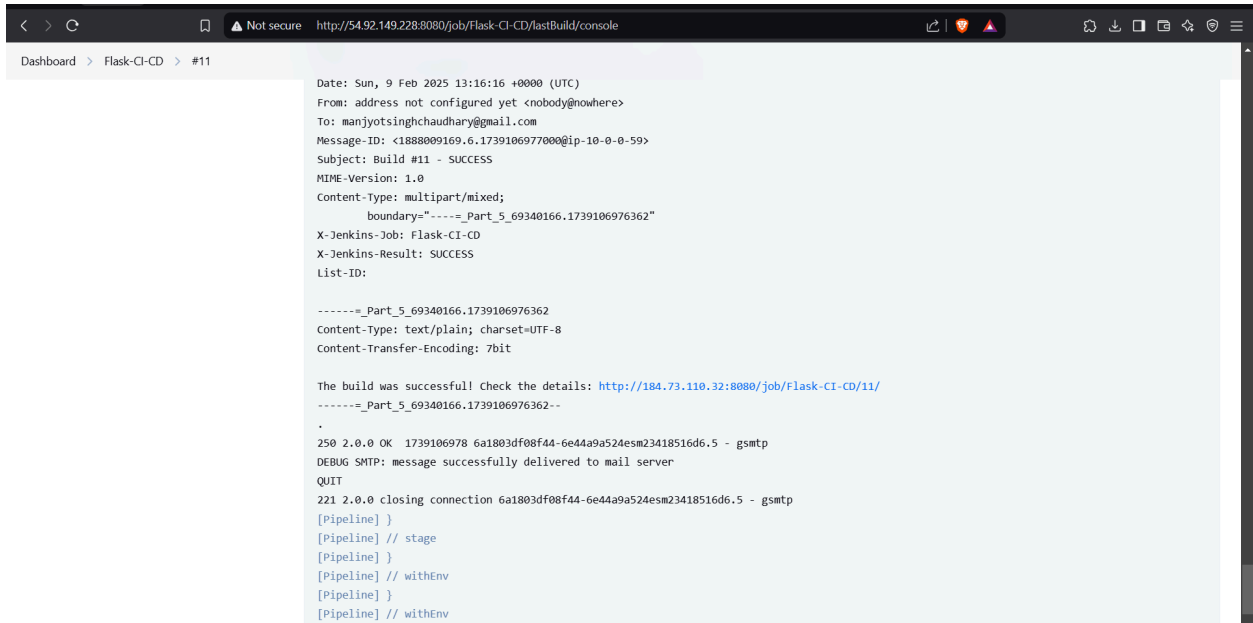
After installing Jenkins, you can set up your Jenkins pipeline for Continuous Integration and Continuous Deployment (CI/CD) by adding the Jenkinsfile to your repository, as described in the pipeline section above.

Jenkins CI/CD

Every push to the main branch of this repository triggers an automated build pipeline on the Jenkins server. The steps of this pipeline are as follows:

1. **Checkout Code:** Pull the latest changes from the GitHub repository.
2. **Set Up Python Environment:** Create a virtual environment and install required dependencies.
3. **Docker Login:** Log in to Docker Hub using credentials stored in Jenkins.
4. **Build Docker Image:** Build the Docker image for the application.
5. **Push Docker Image:** Push the newly built Docker image to Docker Hub under the `iammanjyyot/my-flask-hello-world` repository.
6. **Deploy to Production:** Deployment steps can be added here based on your production environment.

The screenshot shows the Jenkins web interface. At the top, the browser address bar displays the URL `http://184.73.110.32:8080/job/Flask-CI-CD/lastBuild/pipeline-graph/`. The Jenkins logo and name are visible in the top left. Below the navigation bar, the breadcrumb trail reads: `Dashboard > Flask-CI-CD > #7 > Pipeline Overview`. The main heading is `< Build #7`, accompanied by a green checkmark icon. To the right of the heading are three buttons: `Rebuild` (green), `Console` (grey), and `Configure` (grey). The central part of the page displays the pipeline graph, which consists of a horizontal line with several steps marked by green checkmarks: `Start`, `Checkout SCM`, `Checkout Code`, `Set Up Python E...`, `Docker Login`, `Build Docker Im...`, `Push Docker Im...`, `Deploy to Produ...`, `Post Actions`, and `End`. On the right side, a 'Details' box provides build statistics: `Started 8 min 48 sec ago`, `Queued 8.4 sec`, and `Took 39 sec`.



Email Notifications

Jenkins will send an email notification with the build result (success or failure) to manjyotsinghchaudhary@gmail.com after each build, along with the Docker Hub update.

Jenkinsfile

groovy

```
pipeline {
    agent any
    environment {
        DOCKER_CREDENTIALS = 'docker-hub-cred'
    }
    stages {
        stage('Checkout Code') {
            steps {
                checkout scm
            }
        }
        stage('Set Up Python Environment') {
            steps {
                script {
                    sh '/bin/ -c "python3 -m venv venv && source venv/bin/activate && pip install -r requirements.txt"'
                }
            }
        }
    }
}
```

```

        }
    }
}
stage('Docker Login') {
    steps {
        script {
            withCredentials([usernamePassword(credentialsId:
DOCKER_CREDENTIALS, passwordVariable: 'DOCKER_PASSWORD',
usernameVariable: 'DOCKER_USERNAME')]) {
                sh 'echo $DOCKER_PASSWORD | docker login -u
$DOCKER_USERNAME --password-stdin'
            }
        }
    }
}
stage('Build Docker Image') {
    steps {
        script {
            sh 'docker build -t
iammanjyyot/my-flask-hello-world:latest .'
        }
    }
}
stage('Push Docker Image') {
    steps {
        script {
            sh 'docker push
iammanjyyot/my-flask-hello-world:latest'
        }
    }
}
stage('Deploy to Production') {
    steps {
        script {
            echo 'Deploying to production...'
        }
    }
}
}

```

```

    }
    post {
        always {
            cleanWs()
        }
        success {
            emailext(
                subject: "Build #${BUILD_NUMBER} - SUCCESS",
                body: "The build was successful! Check the details:
${BUILD_URL}",
                to: "manjyotsinghchaudhary@gmail.com"
            )
        }
        failure {
            emailext(
                subject: "Build #${BUILD_NUMBER} - FAILURE",
                body: "The build failed. Check the console output for
details: ${BUILD_URL}",
                to: "manjyotsinghchaudhary@gmail.com"
            )
        }
    }
}
}
}
}

```

The screenshot displays the Jenkins web interface for a specific build. The browser address bar shows the URL `http://184.73.110.32:8080/job/Flask-CI-CD/lastBuild/`. The Jenkins logo and navigation menu are visible at the top. The main content area shows the build status as **#7 (Feb 8, 2025, 3:35:34 PM)** with a green checkmark icon. A sidebar on the left lists various build-related actions like 'Status', 'Changes', 'Console Output', etc. The main panel provides details about the build's execution, including a timeline of events (started by GitHub push, run spent time breakdown) and the source code revision and repository information.

Jenkins

Dashboard > Flask-CI-CD > #7

#7 (Feb 8, 2025, 3:35:34 PM) [Add description](#) [Keep this build forever](#)

Started by GitHub push by Manjyot Started 6 min 34 sec ago
Took 39 sec

This run spent:

- 8.4 sec waiting;
- 39 sec build duration;
- 48 sec total from scheduled to completion.

git **Revision:** 019cc4a4e187efeed2a2e56c19c16fc8dd2f0c05
Repository: <https://github.com/Manjyot/flask-hello-world.git>

- refs/remotes/origin/main

Changes

- Updated jenkins file ([commit: 019cc4a](#)) ([details](#) / [githubweb](#))



Successfully email is sent by the JenkinsServer to the user after every push on the main branch.