Jenkins Basics - GitHub Integration & CI Pipeline Project

Project Overview

Deliverables:

- GitHub + Jenkins integration project
- Jenkins pipeline with test reports

Tools Used: Git, GitHub, Jenkins, pytest, Docker, Flask

Repository: `https://github.com/prashant-haptiq/Hodo-App.git`

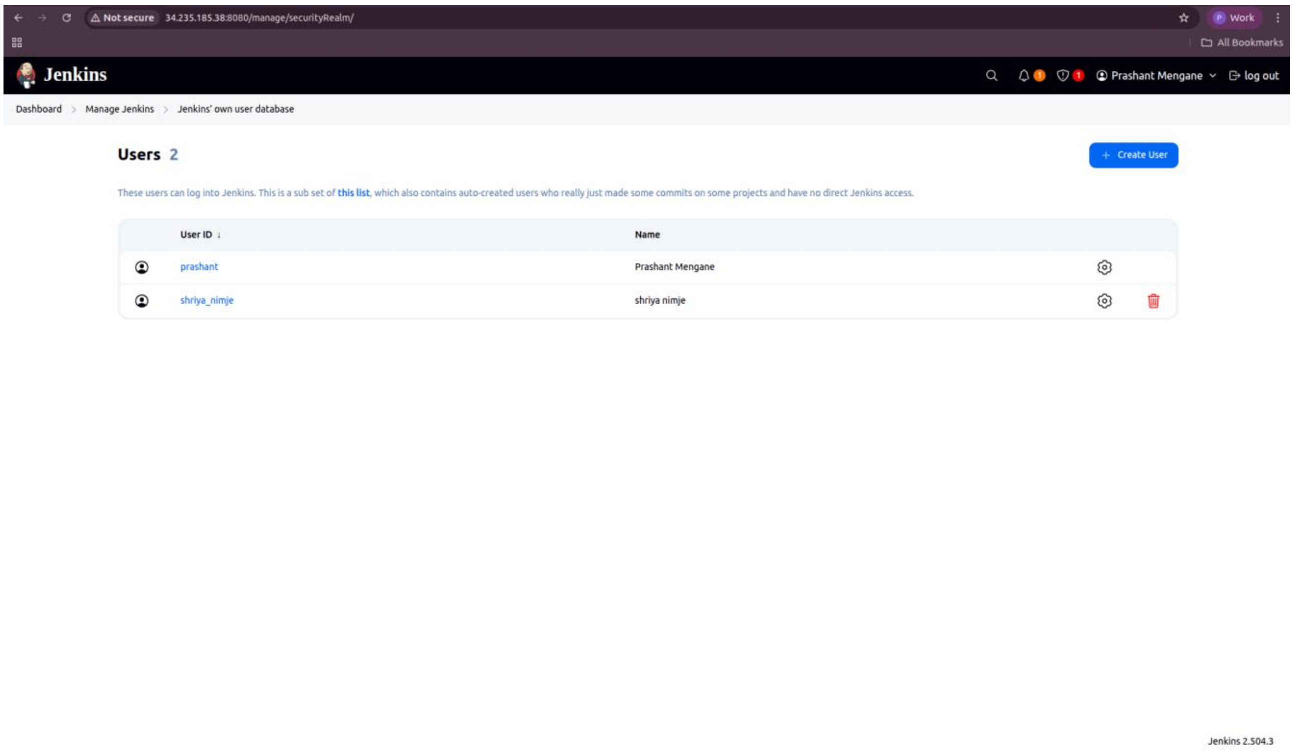
Implementation Steps

Step 1: Jenkins Setup & Configuration

1.1 Additional User Creation

Created New User:

- Navigate to: **Manage Jenkins → Manage Users → Create User**



34.235.185.38:8080/manage/securityRealm/

Dashboard > Manage Jenkins > Jenkins' own user database

Users 2

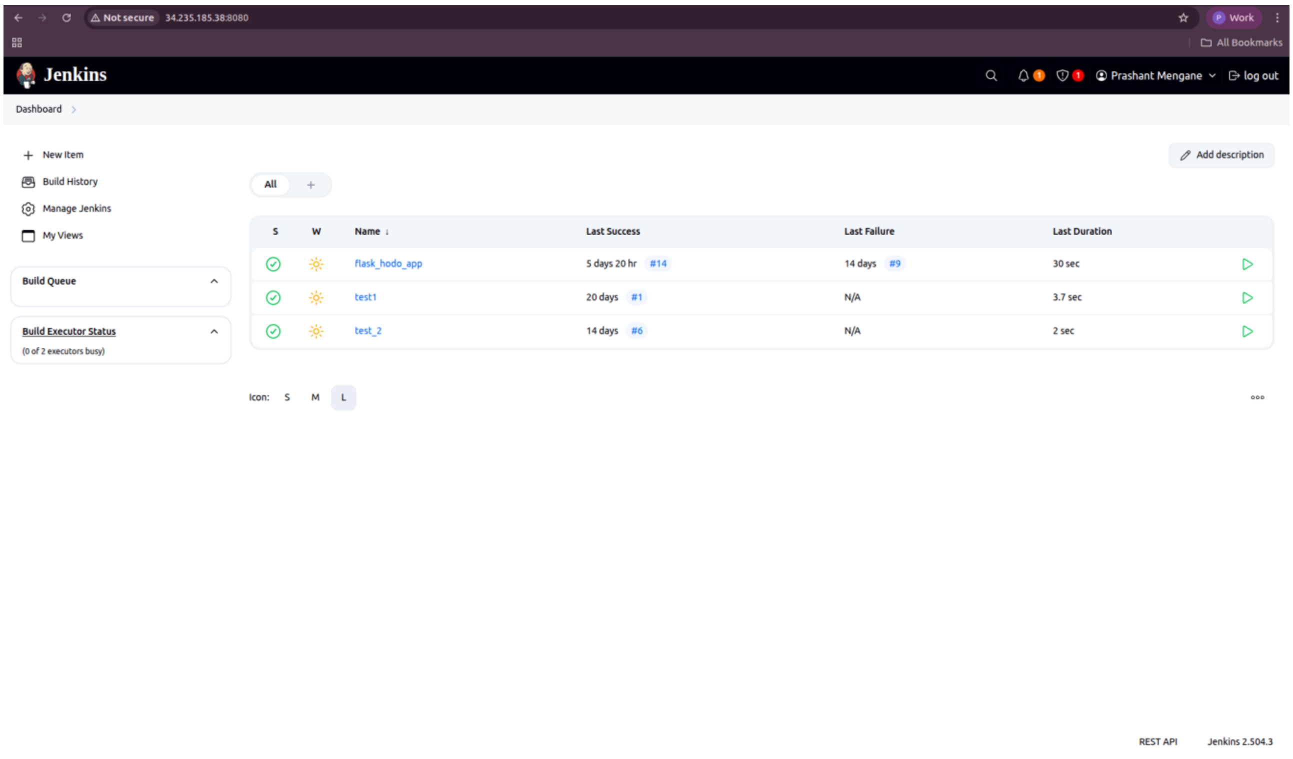
These users can log into Jenkins. This is a sub set of this list, which also contains auto-created users who really just made some commits on some projects and have no direct Jenkins access.

User ID ↓	Name
prashant	Prashant Mengane
shriya_nimje	shriya nimje

Jenkins 2.504.3

1.2 Jenkins Dashboard

- Successfully accessed Jenkins dashboard
- Installed additional plugins: Git, GitHub, Pipeline, Docker


Dashboard

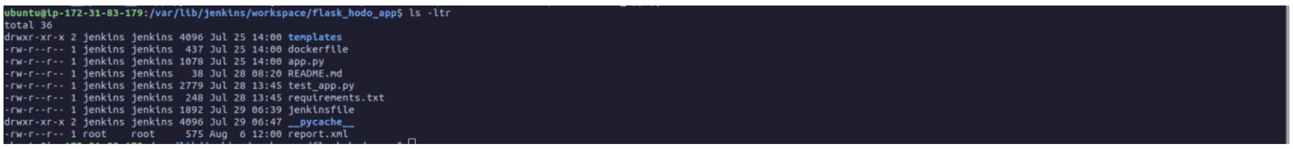
S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀	flask_hodo_app	5 days 20 hr #14	14 days #9	30 sec
✓	☀	test1	20 days #1	N/A	3.7 sec
✓	☀	test_2	14 days #6	N/A	2 sec

Build Executor Status (0 of 2 executors busy)

1.3 Jenkins Build Workspace Location

Default Build Storage Location on EC2:

- Jenkins workspace path: `/var/lib/jenkins_home/workspace/`
- Project builds saved at: `/var/lib/jenkins_home/workspace/hodo-app-ci-pipeline/`
- Build artifacts and reports stored in project workspace directory
- Each build creates temporary workspace for pipeline execution



```
ubuntu@ip-172-31-83-179:/var/lib/jenkins/workspace/flask_hodo_app$ ls -ltr
total 36
drwxr-xr-x 2 jenkins jenkins 4096 Jul 25 14:00 templates
-rw-r--r-- 1 jenkins jenkins  437 Jul 25 14:00 dockerfile
-rw-r--r-- 1 jenkins jenkins 1078 Jul 25 14:00 app.py
-rw-r--r-- 1 jenkins jenkins   38 Jul 28 08:20 README.md
-rw-r--r-- 1 jenkins jenkins 2779 Jul 28 13:45 test_app.py
-rw-r--r-- 1 jenkins jenkins  248 Jul 28 13:45 requirements.txt
-rw-r--r-- 1 jenkins jenkins 1892 Jul 29 06:39 jenkinsfile
drwxr-xr-x 2 jenkins jenkins 4096 Jul 29 06:47 __pycache__
-rw-r--r-- 1 root    root     575 Aug  6 12:00 report.xml
```

Step 2: GitHub Integration Setup

2.1 GitHub Personal Access Token

GitHub Steps:

1. GitHub Settings → Developer settings → Personal access tokens
2. Generate new token with permissions: `repo`, `admin:repo_hook`
3. Token name: `jenkins-integration`

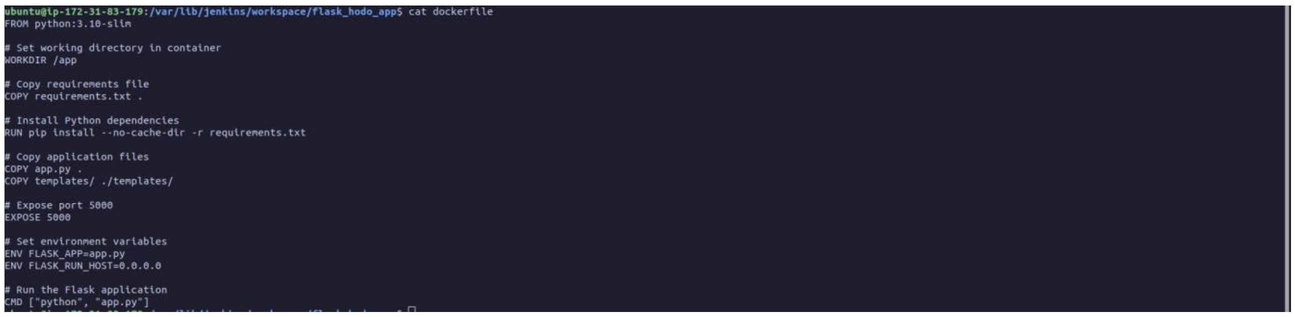
Step 3: GitHub Repository Setup

3.1 Repository: Hodo-App

Repository Structure:

```
Hodo-App/
├── app.py              # Flask application
├── requirements.txt    # Python dependencies
├── Dockerfile         # Docker configuration
├── Jenkinsfile        # Pipeline configuration
├── tests/
│   └── test_app.py    # pytest test files
└── README.md
```

Step 4: Docker Configuration



```
ubuntu@ip-172-31-83-179:/var/lib/jenkins/workspace/flask_hodo_app$ cat dockerfile
FROM python:3.10-slim

# Set working directory in container
WORKDIR /app

# Copy requirements file
COPY requirements.txt .

# Install Python dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Copy application files
COPY app.py .
COPY templates/ ./templates/

# Expose port 5000
EXPOSE 5000

# Set environment variables
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0.0

# Run the Flask application
CMD ["python", "app.py"]
```

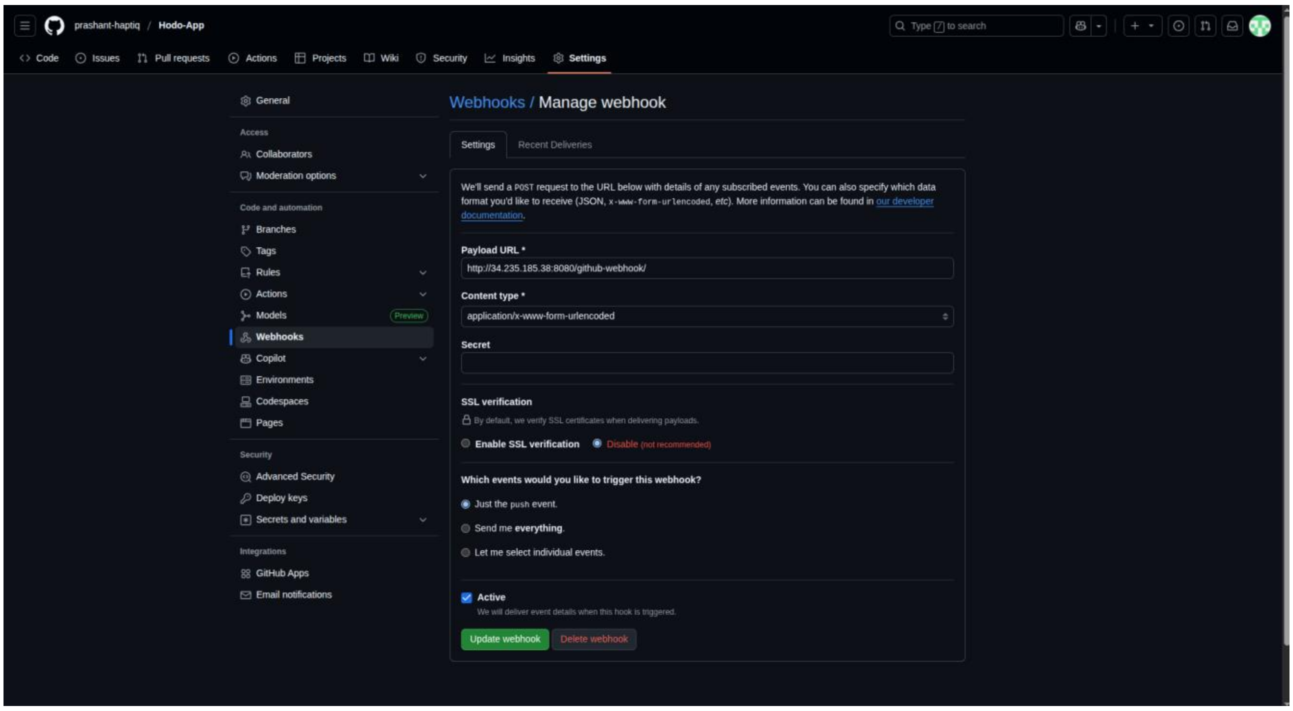
Step 5: Webhook Configuration

5.1 GitHub Webhook Setup

Repository Settings → Webhooks → Add webhook

Webhook Configuration:

- Payload URL: `http://34.235.185.38:8080/github-webhook/`
- Content type: `application/json`
- Trigger events: `Just the push event`
- Active: □



Webhooks / Manage webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in our developer documentation.

Payload URL *: http://34.235.185.38:8080/github-webhook/

Content type *: application/x-www-form-urlencoded

Secret:

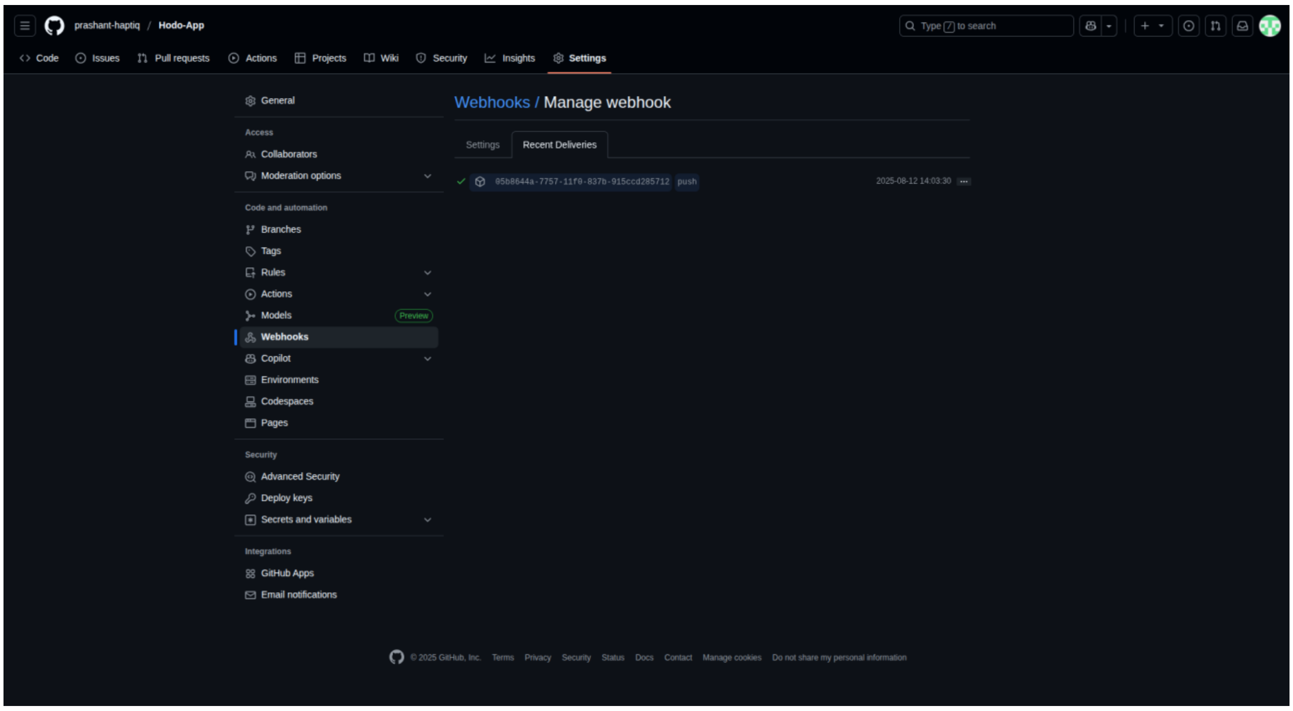
SSL verification: By default, we verify SSL certificates when delivering payloads.
Enable SSL verification [✓ Disable (not recommended)]

Which events would you like to trigger this webhook?
[✓ Just the push event.]
Send me everything.
Let me select individual events.

[✓ Active] We will deliver event details when this hook is triggered.

5.2 Webhook Delivery Test

- Made a test commit and push to master branch
- Verified webhook delivery in GitHub
- Status: □ Successful delivery



Webhooks / Manage webhook

Recent Deliveries

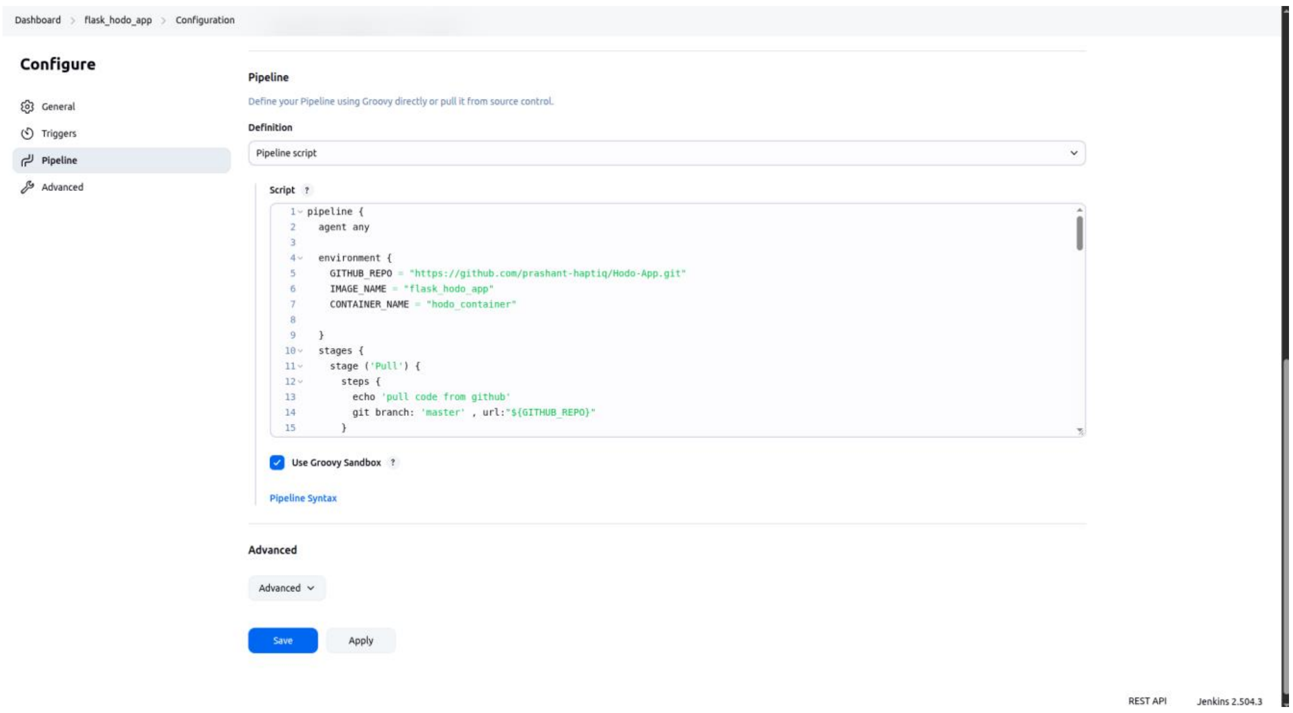
✓ 05b8644a-7757-11f0-837b-915ccd285712 push 2025-08-12 14:03:30

Step 6: Jenkins Pipeline Creation

6.1 Pipeline Job Configuration

New Item → Pipeline

- Job name: `hodo-app-ci-pipeline`
- Pipeline definition: `Pipeline script from SCM`
- SCM: Git
- Repository URL: `https://github.com/prashant-haptiq/Hodo-App.git`
- Credentials: GitHub Personal Access Token
- Branch: `*/master`
- Script Path: `Jenkinsfile`



Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition: Pipeline script

Script

```
pipeline {
  agent any

  environment {
    GITHUB_REPO = "https://github.com/prashant-haptiq/Hodo-App.git"
    IMAGE_NAME = "flask_hodo_app"
    CONTAINER_NAME = "hodo_container"

  }
  stages {
    stage ('Pull') {
      steps {
        echo 'pull code from github'
        git branch: 'master' , url:"${GITHUB_REPO}"
      }
```

Use Groovy Sandbox: ✓

6.2 Your Jenkinsfile Implementation

```
pipeline {
  agent any
  environment {
    GITHUB_REPO = "https://github.com/prashant-haptiq/Hodo-App.git"
    IMAGE_NAME = "flask_hodo_app"
    CONTAINER_NAME = "hodo_container"
  }
  stages {
    stage ('Pull') {
      steps {
        echo 'pull code from github'
        git branch: 'master' , url:"${GITHUB_REPO}"
      }
    }
    stage ('Build') {
      steps {
        echo 'Building docker image'

        sh "docker build -t ${IMAGE_NAME} ."
      }
    }
    stage ('Test') {
      steps {
        echo 'Running test inside the container using pytest'
        sh '''
          #running test inside the new container which we build from image
          #--rm flag will automatically remove the container after the test
          #if the test fails , so the pipeline stage
          #runnig pytest with verbose reporting (-rA)
          #-v (volume mount) ensures that the file which is generated
'report.xml' is saved in jenkins workspace ie. ec2 instance
          docker run --rm -v ${PWD}:/app ${IMAGE_NAME} pytest -rA --junit-
xml=report.xml
          '''
      }
    }
    stage ('Deploy') {
      steps {
        echo 'running container'
        sh '''
          #stopping old container if exists and starting a new one , to avoid
port conflicts
          docker stop ${CONTAINER_NAME} || true
          docker rm ${CONTAINER_NAME} || true
          #run the new container in detached mode
          docker run -d --name ${CONTAINER_NAME} -p 5000:5000 ${IMAGE_NAME}
          echo 'application is running at http://34.235.185.38:5000'

          '''
      }
    }
  }
  post {
    always {
      #this will print the test report in jenkins UI
      junit 'report.xml'
    }
    success {
      echo 'Application is running on port 5000'
    }
    failure {
      echo 'ERROR! check logs'
    }
  }
}
```

Step 7: Pipeline Execution & Testing

7.1 Manual Build Test

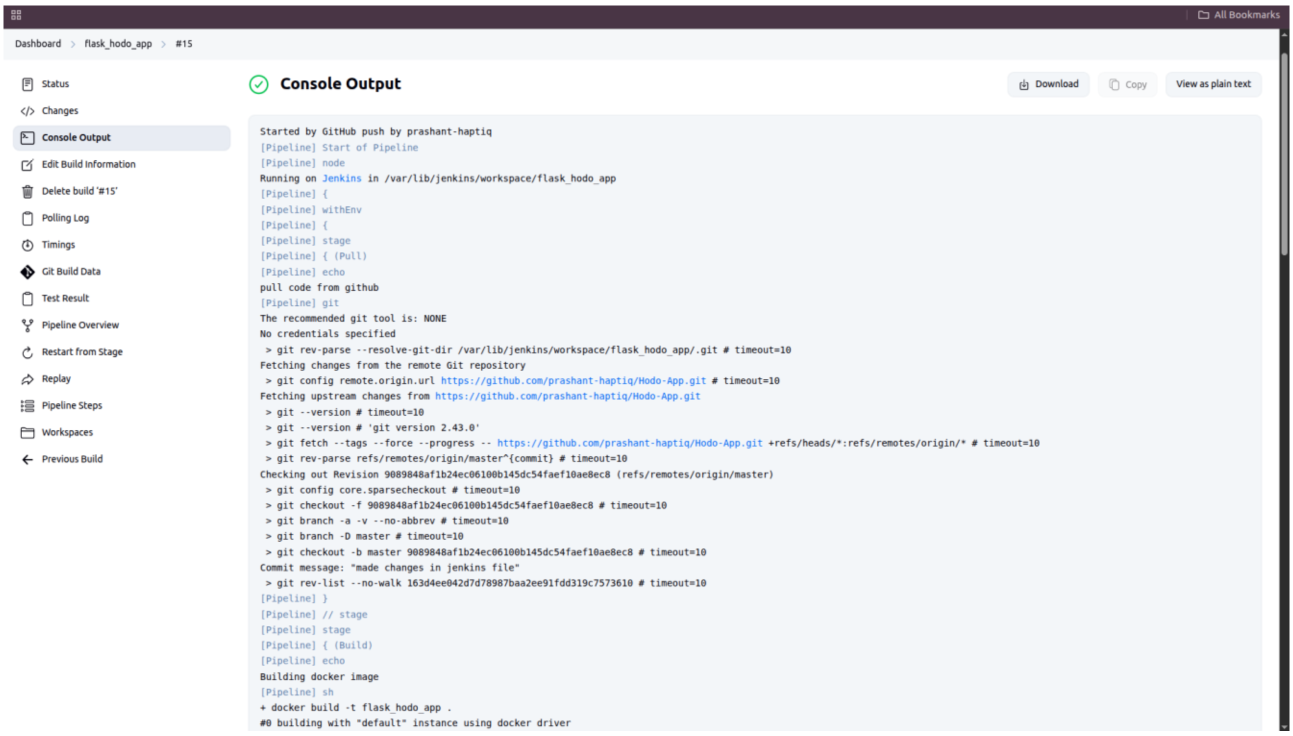
- Triggered first build manually: "Build Now"
- Monitored pipeline execution through Jenkins console
- All 4 stages completed successfully: Pull → Build → Test → Deploy

Pipeline Stages:

1. **Pull**: Code pulled from GitHub master branch
2. **Build**: Docker image `flask_hodo_app` built successfully
3. **Test**: pytest executed inside container with JUnit XML report
4. **Deploy**: Container deployed on port 5000

7.2 Jenkins Console Output

Key Console Messages:



```
Started by GitHub push by prashant-haptiq
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/flask_hodo_app
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Pull)
[Pipeline] echo
pull code from github
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/flask_hodo_app/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/prashant-haptiq/Hodo-App.git # timeout=10
Fetching upstream changes from https://github.com/prashant-haptiq/Hodo-App.git
 > git --version # timeout=10
 > git --version # 'git version 2.43.0'
 > git fetch --tags --force --progress -- https://github.com/prashant-haptiq/Hodo-App.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 9089848af1b24ec06100b145dc54faef10ae8ec8 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 9089848af1b24ec06100b145dc54faef10ae8ec8 # timeout=10
 > git branch -a -v --no-abbrev # timeout=10
 > git branch -D master # timeout=10
 > git checkout -b master 9089848af1b24ec06100b145dc54faef10ae8ec8 # timeout=10
Commit message: "made changes in jenkins file"
 > git rev-list --no-walk 163d4ee042d7d78987baa2ee91fdd319c7573610 # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Building docker image
[Pipeline] sh
+ docker build -t flask_hodo_app .
#0 building with "default" instance using docker driver
```

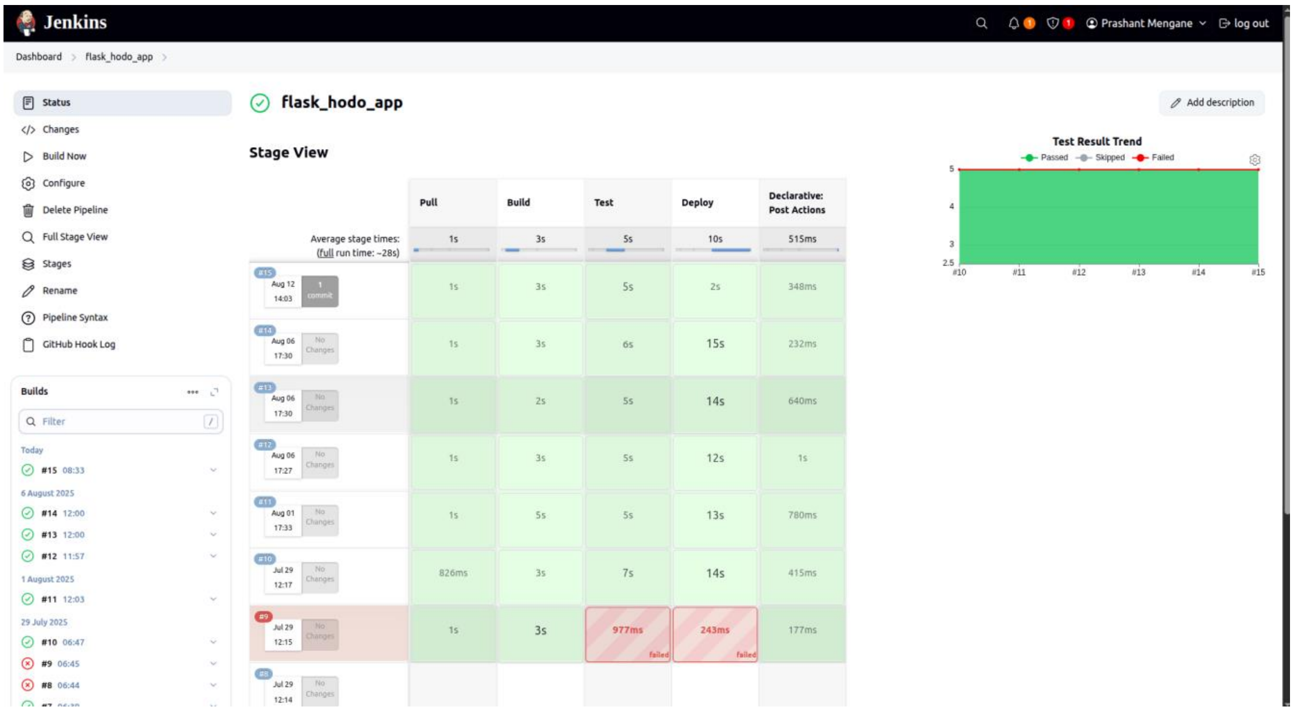
7.3 Webhook Automation Test

Test Process:

1. Made code change in Hodo-App repository
2. Committed and pushed to master branch:

```
git add jenkinsfile; git commit -m “added changes in jenkinsfile”; git push
origin master
```

3. Webhook automatically triggered Jenkins build
4. Pipeline executed successfully with Build #15



	Pull	Build	Test	Deploy	Declarative: Post Actions
Average stage times: (full run time: ~28s)	1s	3s	5s	10s	515ms
#15 Aug 12 14:03 1 commit	1s	3s	5s	2s	348ms
#14 Aug 06 17:30 No Changes	1s	3s	6s	15s	232ms
#13 Aug 06 17:30 No Changes	1s	2s	5s	14s	640ms
#12 Aug 06 17:27 No Changes	1s	3s	5s	12s	1s
#11 Aug 01 17:33 No Changes	1s	5s	5s	13s	780ms
#10 Jul 29 12:17 No Changes	826ms	3s	7s	14s	415ms
#9 Jul 29 12:15 No Changes	1s	3s	977ms failed	243ms failed	177ms
#8 Jul 29 12:14 No Changes					

Step 8: Test Reports Integration

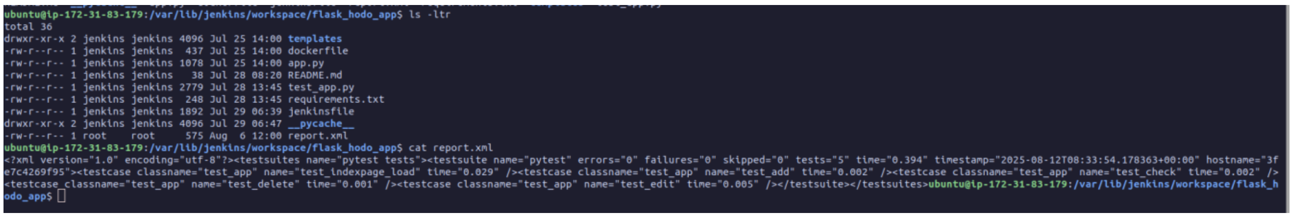
8.1 pytest Test Execution

Test Command in Container:

```
docker run --rm -v ${PWD}:/app flask_hodo_app pytest -rA --junit-xml=report.xml
```

pytest Output:

- Tests discovered and executed
- JUnit XML report generated: `report.xml`
- Volume mount ensures report saved in Jenkins workspace



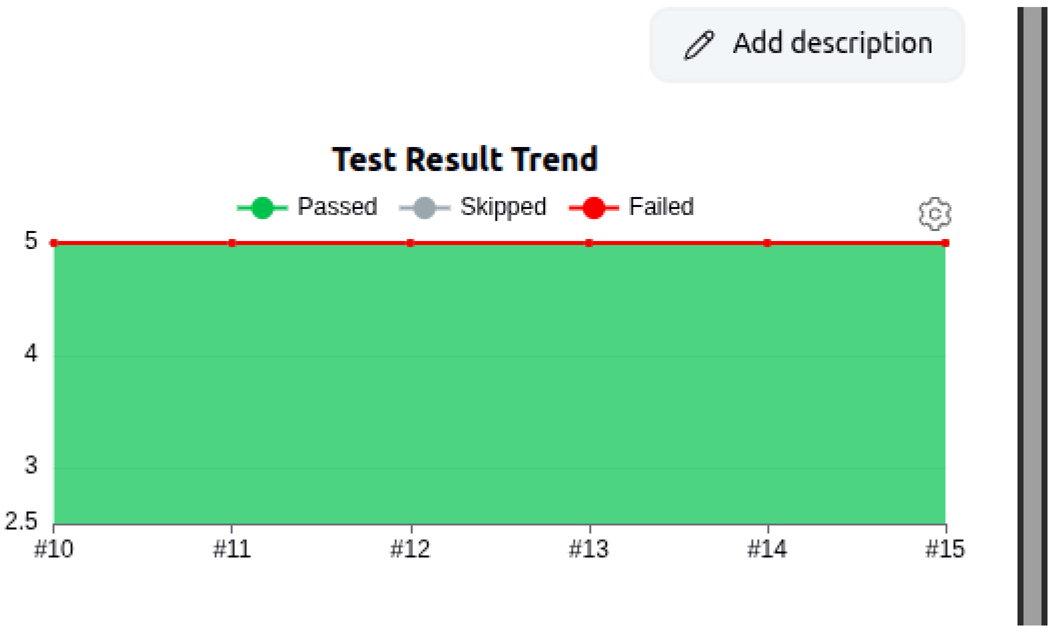
```
ubuntu@ip-172-31-83-179:/var/lib/jenkins/workspace/flask_hodo_app$ ls -ltr
total 36
drwxr-xr-x 2 jenkins jenkins 4096 Jul 25 14:00 templates
-rw-r--r-- 1 jenkins jenkins  437 Jul 25 14:00 dockerfile
-rw-r--r-- 1 jenkins jenkins 1078 Jul 25 14:00 app.py
-rw-r--r-- 1 jenkins jenkins   38 Jul 28 08:20 README.md
-rw-r--r-- 1 jenkins jenkins 2779 Jul 28 13:45 test_app.py
-rw-r--r-- 1 jenkins jenkins  248 Jul 28 13:45 requirements.txt
-rw-r--r-- 1 jenkins jenkins 1892 Jul 29 06:39 jenkinsfile
drwxr-xr-x 2 jenkins jenkins 4096 Jul 29 06:47 __pycache__
-rw-r--r-- 1 root    root     575 Aug  6 12:00 report.xml
ubuntu@ip-172-31-83-179:/var/lib/jenkins/workspace/flask_hodo_app$ cat report.xml
<?xml version="1.0" encoding="utf-8"?><testsuites name="pytest tests"><testsuite name="pytest" errors="0" failures="0" skipped="0" tests="5" time="0.394" timestamp="2025-08-12T08:33:54.178363+00:00" hostname="3f
e7c4269f95"><testcase classname="test_app" name="test_indexpage_load" time="0.029" /><testcase classname="test_app" name="test_add" time="0.002" /><testcase classname="test_app" name="test_check" time="0.002" />
<testcase classname="test_app" name="test_delete" time="0.001" /><testcase classname="test_app" name="test_edit" time="0.005" /></testsuite></testsuites>ubuntu@ip-172-31-83-179:/var/lib/jenkins/workspace/flask_h
odo_app$ 
```

8.2 Jenkins Test Results

Test Report Integration:

- Jenkins `post` section: `junit 'report.xml'`
- Test results automatically published in Jenkins UI
- Test trends visible across builds

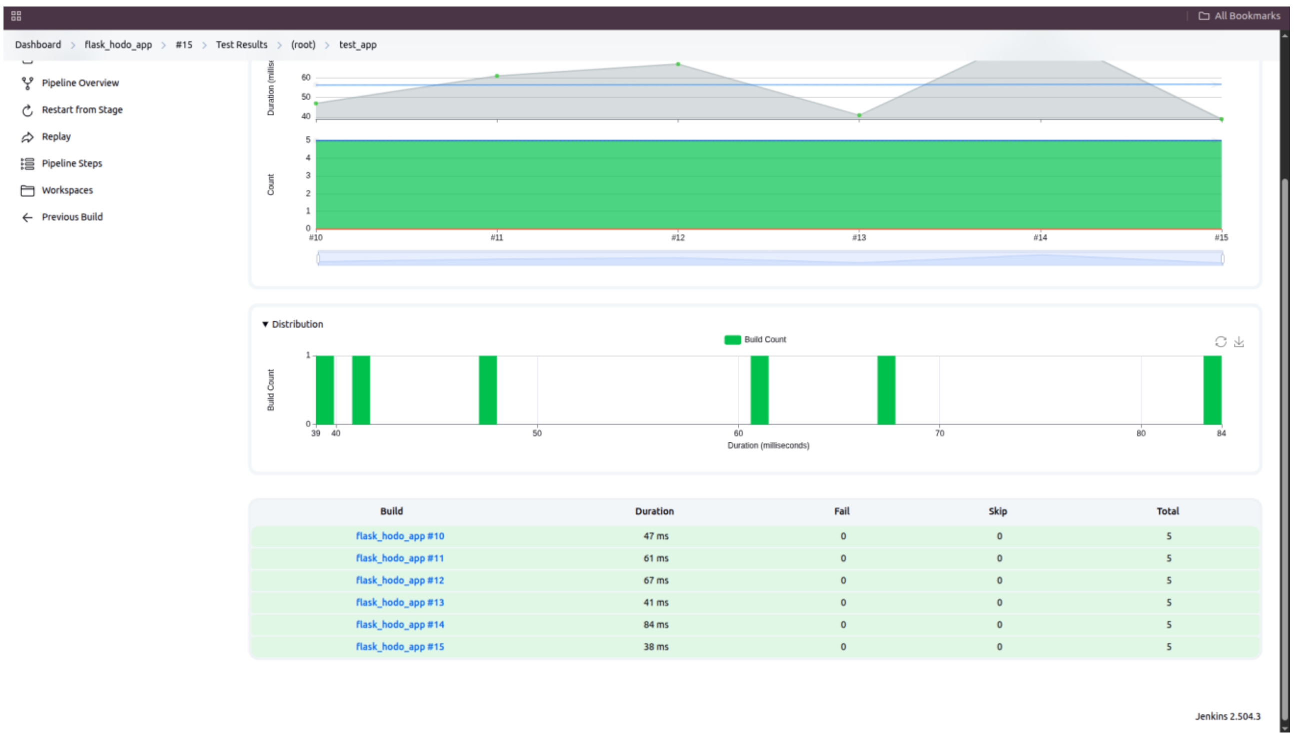
Test Statistics:



8.3 Test Results Dashboard

Jenkins Test Results Features:

- Test result trends across builds
- Individual test case details
- Failure analysis and history
- Test duration tracking



Build	Duration	Fail	Skip	Total
flask_hodo_app #10	47 ms	0	0	5
flask_hodo_app #11	61 ms	0	0	5
flask_hodo_app #12	67 ms	0	0	5
flask_hodo_app #13	41 ms	0	0	5
flask_hodo_app #14	84 ms	0	0	5
flask_hodo_app #15	38 ms	0	0	5

Jenkins 2.504.3

Results & Deliverables

□ GitHub + Jenkins Integration Project

Successfully Implemented:

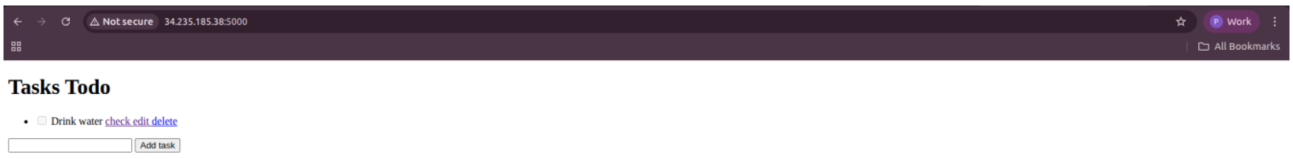
- GitHub repository: `prashant-haptiq/Hodo-App`
- Jenkins pipeline job configured and working
- Webhook integration functional
- Automatic build triggering on master branch push
- Flask application successfully containerized

□ Jenkins Pipeline with Test Reports

Pipeline Features:

- 4 stages: Pull → Build → Test → Deploy
- pytest test execution inside Docker container
- JUnit XML test report generation and publishing
- Automatic container deployment on port 5000
- Container management (stop/remove old, start new)

- Application URL: `http://34.235.185.38:5000`



Tasks Todo

- Drink water check edit delete

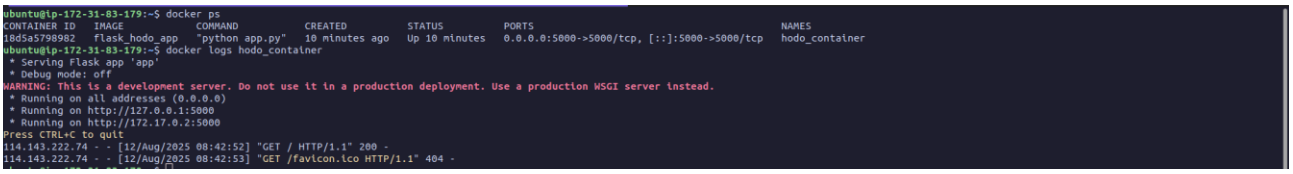
Add task

Application Deployment Verification

Live Application

Deployment Details:

- Container Name: `hodo_container`
- Port Mapping: `5000:5000`
- Application URL: `http://34.235.185.38:5000`
- Status: Running successfully



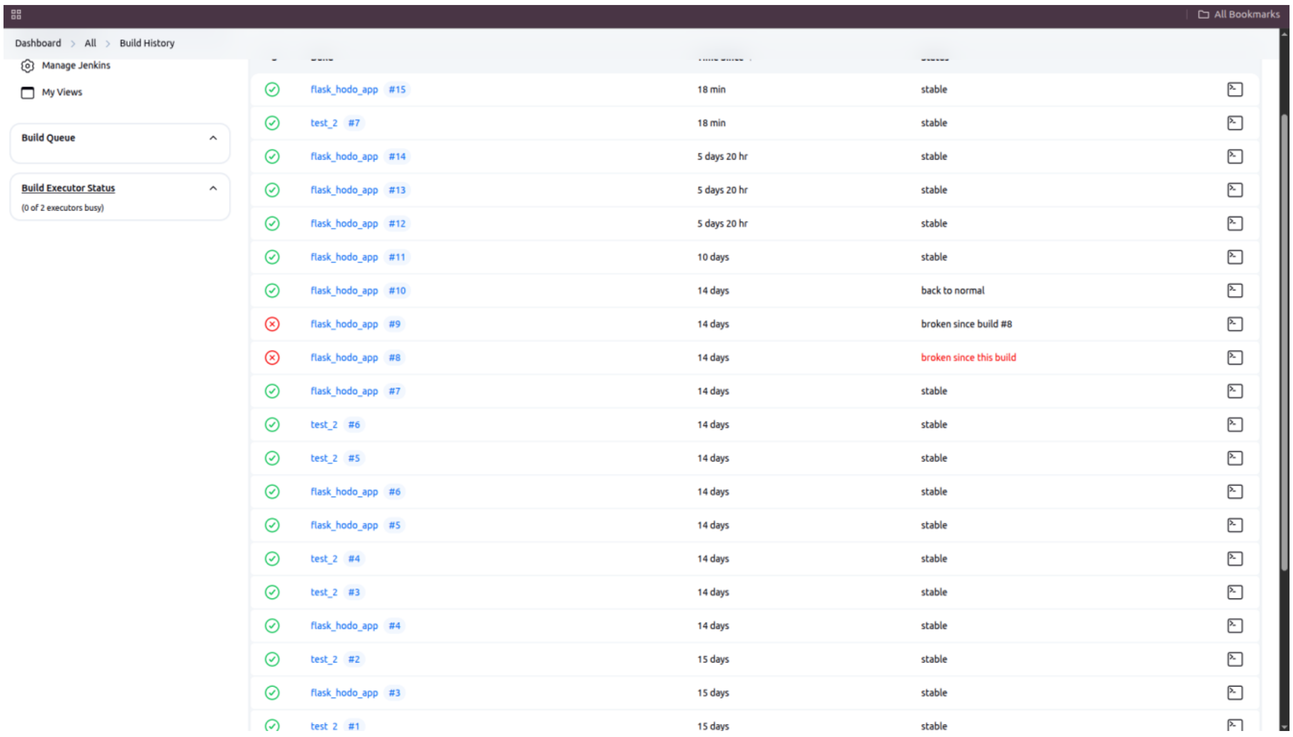
```
ubuntu@ip-172-31-83-179:~$ docker ps
CONTAINER ID   IMAGE           COMMAND           CREATED          STATUS          PORTS                                         NAMES
18d5a5798982   flask_hodo_app  "python app.py"   10 minutes ago   Up 10 minutes   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp   hodo_container
ubuntu@ip-172-31-83-179:~$ docker logs hodo_container
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
114.143.222.74 - - [12/Aug/2025 08:42:52] "GET / HTTP/1.1" 200 -
114.143.222.74 - - [12/Aug/2025 08:42:53] "GET /favicon.ico HTTP/1.1" 404 -
```

Jenkins Dashboard Overview

Main Dashboard

Dashboard Features:

- Build history with success/failure status
- Recent builds across all projects
- System information and node status
- Plugin management and system configuration



Dashboard > All > Build History

Manage Jenkins

My Views

Build Queue

Build Executor Status

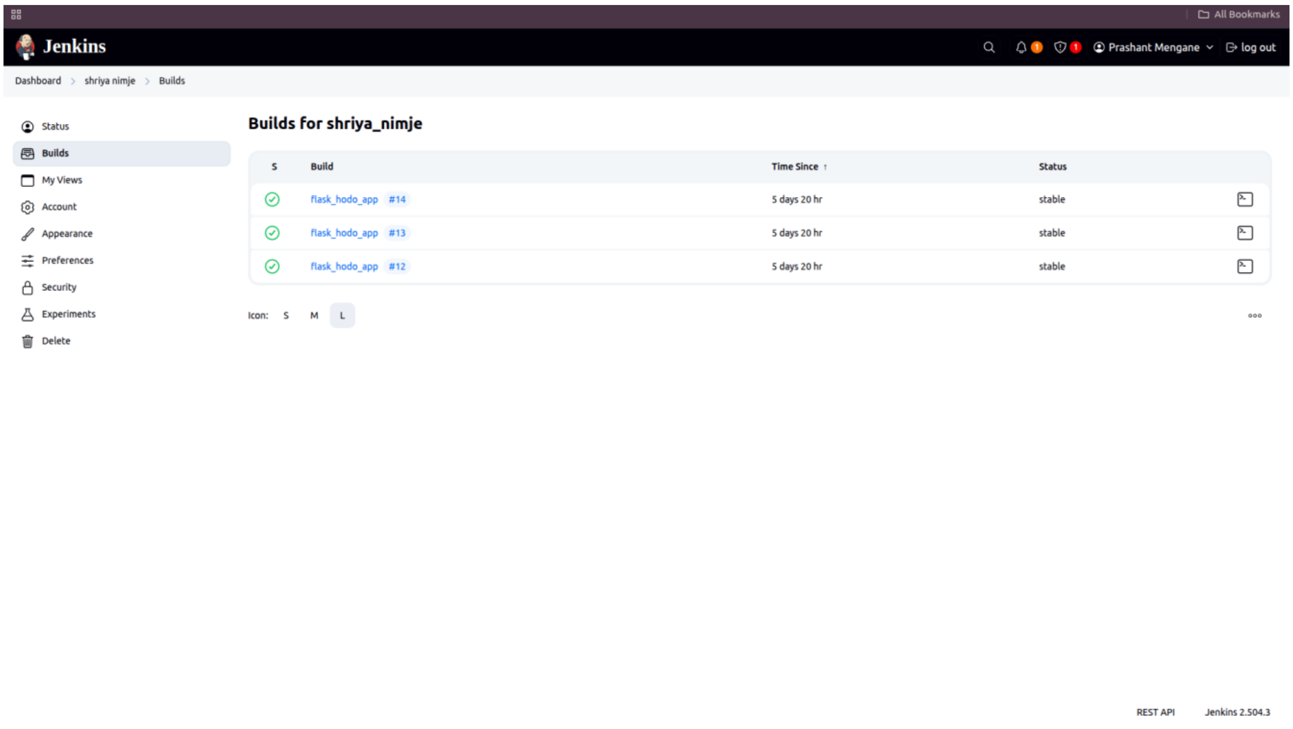
(0 of 2 executors busy)

✓	flask_hodo_app #15	18 min	stable
✓	test_2 #7	18 min	stable
✓	flask_hodo_app #14	5 days 20 hr	stable
✓	flask_hodo_app #13	5 days 20 hr	stable
✓	flask_hodo_app #12	5 days 20 hr	stable
✓	flask_hodo_app #11	10 days	stable
✓	flask_hodo_app #10	14 days	back to normal
✗	flask_hodo_app #9	14 days	broken since build #8
✗	flask_hodo_app #8	14 days	broken since this build
✓	flask_hodo_app #7	14 days	stable
✓	test_2 #6	14 days	stable
✓	test_2 #5	14 days	stable
✓	flask_hodo_app #6	14 days	stable
✓	flask_hodo_app #5	14 days	stable
✓	test_2 #4	14 days	stable
✓	test_2 #3	14 days	stable
✓	flask_hodo_app #4	14 days	stable
✓	test_2 #2	15 days	stable
✓	flask_hodo_app #3	15 days	stable
✓	test_2 #1	15 days	stable

User Management

Users in Jenkins:

- Additional user created with appropriate permissions
- builds by user


All Bookmarks

Jenkins

Prashant Mengane log out

Dashboard > shriya nimje > Builds

- Status
- Builds
- My Views
- Account
- Appearance
- Preferences
- Security
- Experiments
- Delete

Builds for shriya_nimje

S	Build	Time Since ↑	Status
✓	flask_hodo_app #14	5 days 20 hr	stable
✓	flask_hodo_app #13	5 days 20 hr	stable
✓	flask_hodo_app #12	5 days 20 hr	stable

Icon: S M L

REST API Jenkins 2.504.3