Using jenkins

26 February 2025 09:46

```
git config --global user.name <>
```

```
git config --global user.email <>
```

```
git config --global --list
```

```
# Step 1: Create project directory and navigate into it
mkdir my_python_project && cd my_python_project

# Step 2: Initialize a Git repository
git init

# Step 3: Create a pyproject.toml for the project
cat << EOF > pyproject.toml
[build-system]
requires = ["hatchling"]
build-backend = "hatchling.build"

[project]
name = "my_python_project"
version = "0.1.0"
dependencies = []

[project.scripts]
myapp = "my_python_project.main:main"
EOF

# Step 4: Create source code directory and main Python file
mkdir -p src/my_python_project
cat << EOF > src/my_python_project/main.py
def main():
    print("Hello from Jenkins CI/CD Pipeline!")

if _name_ == "_main_":
    main()
EOF

# Step 5: Create a basic test file
mkdir tests
cat << EOF > tests/test_main.py
import unittest
from my_python_project.main import main
import io
import sys

class TestMain(unittest.TestCase):
    def test_main(self):
        captured_output = io.StringIO()
        sys.stdout = captured_output
        main()
        sys.stdout = sys._stdout_
        self.assertEqual(captured_output.getvalue().strip(), "Hello from Jenkins
CI/CD Pipeline!")

if _name_ == "_main_":
    unittest.main()
EOF

# Step 6: Create a Dockerfile
cat << EOF > Dockerfile
FROM python:3.9-slim

WORKDIR /app

COPY dist/*.whl .
RUN pip install *.whl

CMD ["myapp"]
EOF

# Step 7: Create a .gitignore file
cat << EOF > .gitignore
_pycache_/
*.py[cod]
*$py.class
dist/
build/
*.egg-info/
venv/
EOF

# Step 8: Build the initial wheel file (requires 'build' package)
pip install build
```

```
python -m build --wheel

# Step 9: Commit everything to Git
git add .
git commit -m "Initial project setup with Python code and Dockerfile"

# Step 10: Create a Jenkinsfile for the pipeline
cat << EOF > Jenkinsfile
pipeline {
    agent any

    environment {
        DOCKER_IMAGE = 'my-python-project:latest'
    }

    stages {
        stage('Checkout') {
            steps {
                // For local Git repo (adjust path if needed)
                dir('/home/user/my_python_project') {
                    git branch: 'main', url: 'file:///home/user/my_python_project'
                }
            }
        }

        stage('Build Wheel') {
            steps {
                sh 'pip install build'
                sh 'python -m build --wheel'
            }
        }

        stage('Test') {
            steps {
                sh 'pip install pytest'
                sh 'pytest tests/'
            }
        }

        stage('Build Docker Image') {
            steps {
                sh 'docker build -t ${DOCKER_IMAGE} .'
            }
        }

        stage('Deploy') {
            steps {
                sh 'docker stop my-python-container || true'
                sh 'docker rm my-python-container || true'
                sh 'docker run -d --name my-python-container ${DOCKER_IMAGE}'
            }
        }
    }

    post {
        success {
            echo 'Pipeline completed successfully!'
        }
        failure {
            echo 'Pipeline failed!'
        }
    }
}
EOF

# Step 11: Add Jenkinsfile to Git and commit
git add Jenkinsfile
git commit -m "Add Jenkinsfile for CI/CD pipeline"

# Step 12: Display project structure
echo "Project structure created:"
ls -R
```

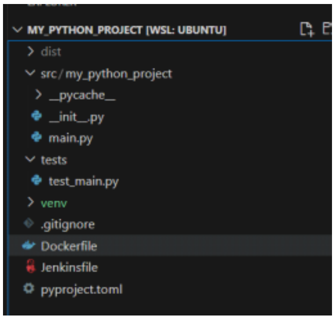
```
[build-system]
        requires = ["hatchling"]
        build-backend = "hatchling.build"

        [project]
        name = "my_python_project"
        version = "0.1.0"
        dependencies = []

        [project.scripts]
        myapp = "my_python_project.main:main"

        [tool.hatch.build.targets.wheel]
        packages = ["src/my_python_project"]
```

```
    git@github.com:jineshranawatcode/my_python_project.git

sudo usermod -aG docker jenkins

eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_ed25519


python3 -m venv venv
                  . venv/bin/activate
                   pip install --upgrade pip
                   pip install build
                   python3 -m build --wheel
```

Few changes in test_main.py

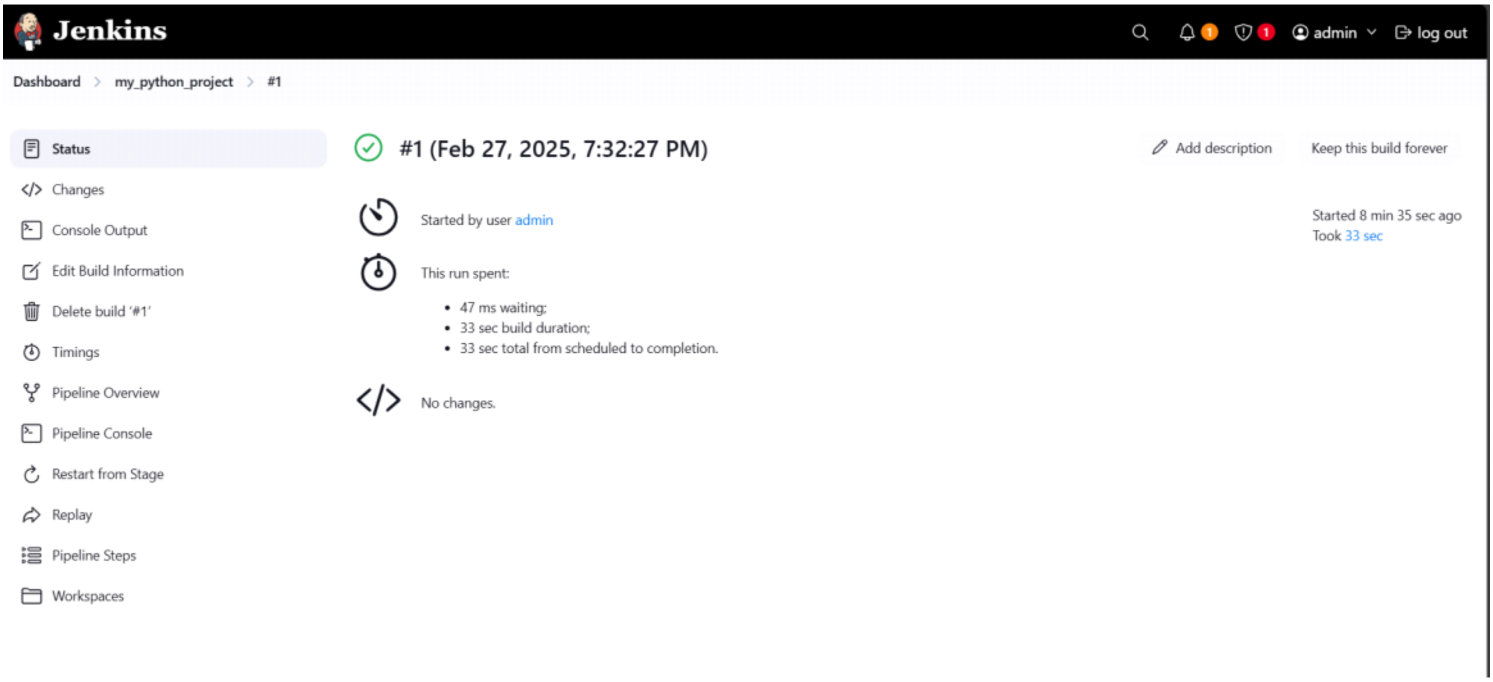
```
import unittest
import os
import sys
import io
sys.path.insert(0,
os.path.abspath(os.path.join(os.path.dirname(__file__),
"../src")))
from my_python_project.main import main

class TestMain(unittest.TestCase):
    def test_main(self):
        captured_output = io.StringIO()
        sys.stdout = captured_output
        main()
        sys.stdout = sys.__stdout__  # Fix: Corrected to
restore sys.stdout
        self.assertEqual(captured_output.getvalue().strip(
), "Hello from Jenkins CI/CD Pipeline!")
if __name__ == "__main__":
    unittest.main()
```

File structure be like



```
MY_PYTHON_PROJECT [WSL: UBUNTU]
> dist
v src / my_python_project
  > __pycache__
    __init__.py
    main.py
v tests
    test_main.py
> venv
  .gitignore
  Dockerfile
  Jenkinsfile
  pyproject.toml
```

In jenkins:



#1 (Feb 27, 2025, 7:32:27 PM)

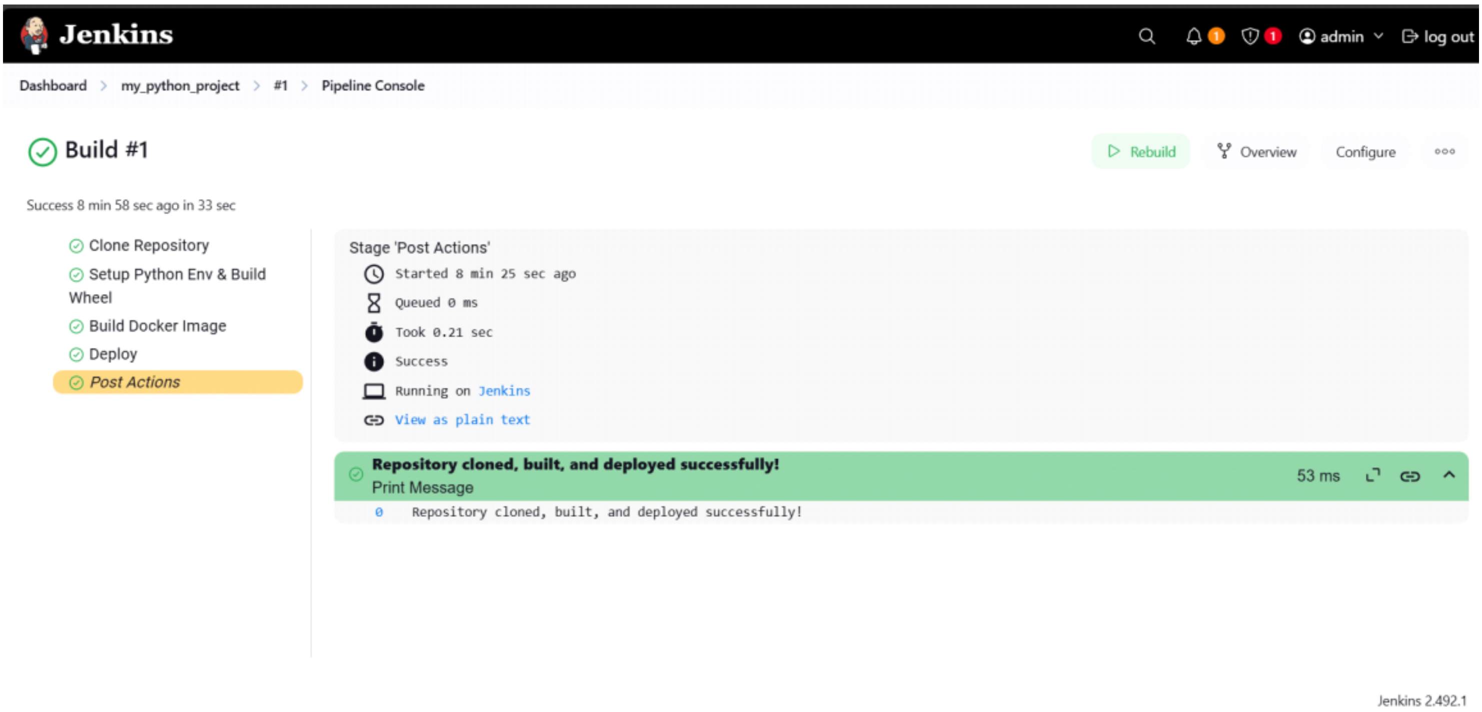
Started by user admin

Started 8 min 35 sec ago
Took 33 sec

This run spent:

- 47 ms waiting;
- 33 sec build duration;
- 33 sec total from scheduled to completion.

No changes.



Build #1

Success 8 min 58 sec ago in 33 sec

- Clone Repository
- Setup Python Env & Build Wheel
- Build Docker Image
- Deploy
- *Post Actions*

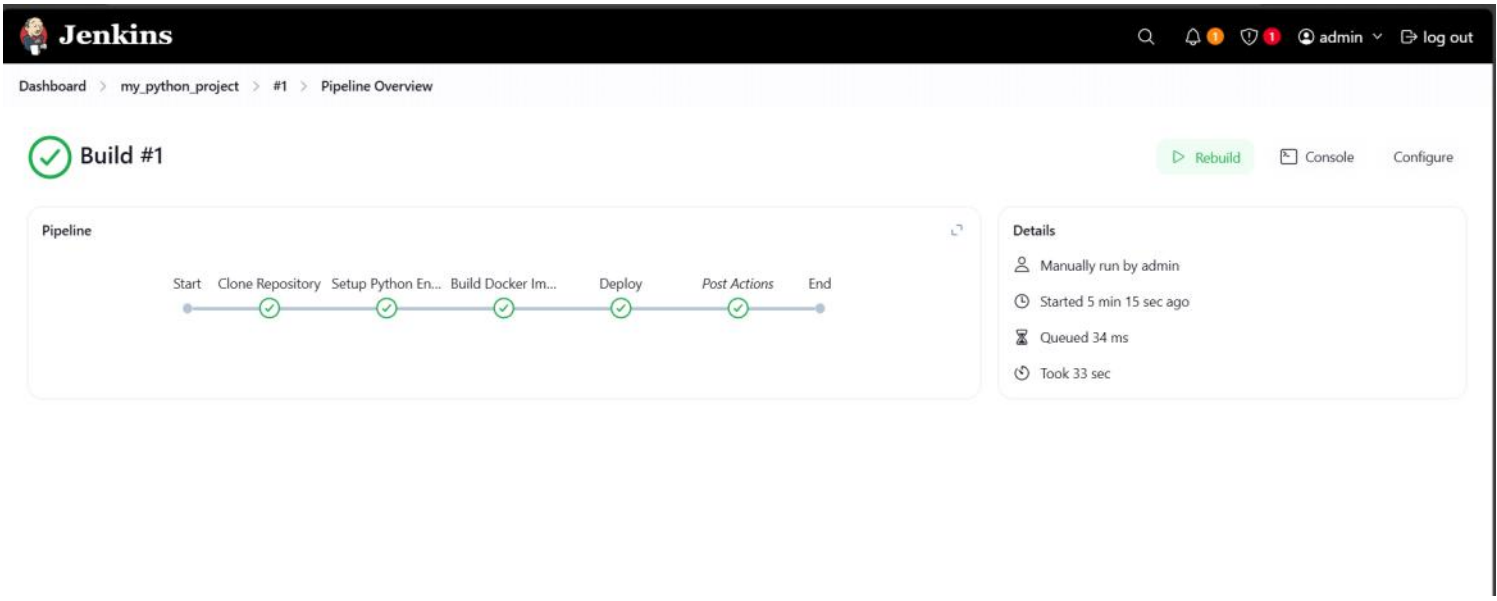
Stage 'Post Actions'

```
Started 8 min 25 sec ago
Queued 0 ms
Took 0.21 sec
Success
Running on Jenkins
View as plain text
```

Repository cloned, built, and deployed successfully!
Print Message 53 ms

```
0    Repository cloned, built, and deployed successfully!
```

Jenkins 2.492.1



Build #1

Details

- Manually run by admin
- Started 5 min 15 sec ago
- Queued 34 ms
- Took 33 sec