# Working with Jenkins

**Step 1: Set Up Your Python Application**

1. Prepare Your Python Application:
   Ensure your application has a Dockerfile. Here's an example for a simple Flask app:

Directory Structure: as a separate folder

app/

├── **app.py**

├── **requirements.txt**

└── **Dockerfile**

app.py:

```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello():
   return "Hello, World!"
if __name__ == "__main__":
   app.run(host="0.0.0.0", port=5000)
```

requirements.txt:

```
flask
```

Dockerfile:

```
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["python", "app.py"]
```

**Step 2:** Push the application code (including the Dockerfile) to a GitHub repository.

**Step 3: Configure Jenkins Pipeline**

1. Create a New Pipeline Job:
    - **In Jenkins, click New Item > Pipeline > Provide a name > Click OK.**
2. Configure the Pipeline:
    - Under the "Pipeline" section, choose Pipeline script.
    - Use the following example for a Python app:

```
pipeline {
    agent any
    stages {
        stage('Clone Repository') {
            steps {
                git 'https://github.com/your-username/your-python-repo.git'
            }
        }
        stage('Build Docker Image') {
            steps {
                script {
                    docker.build('your-dockerhub-username/python-app')
                }
            }
        }
        stage('Push Docker Image') {
            steps {
                withDockerRegistry([credentialsId: 'docker-hub-credentials', url: ''']) {
                    script {
                        docker.image('your-dockerhub-username/python-app').push('latest')
                    }
                }
```

```
        }
    }
    stage('Run Tests') {
        steps {
            script {
                docker.image('your-dockerhub-username/python-app:latest').inside {
                    sh 'pytest tests/'
                }
            }
        }
    }
}
```

3. **Test the Pipeline:**

- Save and trigger a build.
- Ensure the Docker image is pushed to the registry.

**Step 4:** Deploy the Application

1. **Run the Container Locally**:

   docker run -d -p 5000:5000 your-dockerhub-username/python-app:latest

2. **Access the Application**:
    1. Open http://localhost:5000 in a browser to see your Python app running.