**1.What is Jenkins?**

Jenkins is an open-source automation server used for Continuous Integration (CI) and Continuous Delivery (CD). It helps automate building, testing, and deploying software projects and its MASTER-SALVE architecture.

Slave:

A screenshot of a computer

AI-generated content may be incorrect.

**2.On Which Platforms Can You Install Jenkins: Jenkins is cross-platform and can be installed on:**

**1. Operating Systems**

| **Platform** | **Installation Method** |
| --- | --- |
| Linux | .deb (Ubuntu/Debian), .rpm (RHEL/CentOS), Docker |
| Windows | MSI installer, Docker, Chocolatey |
| macOS | Homebrew (brew install jenkins), Docker |

**2. Cloud Platforms**

| **Platform** | **How to Run Jenkins** |
| --- | --- |
| AWS | EC2, ECS, EKS |
| Azure | Azure VM, Azure Kubernetes Service (AKS) |

**3. Containerized (Docker & Kubernetes)**

| **Method** | **Command / Setup** |
| --- | --- |
| Docker | docker run -p 8080:8080 jenkins/jenkins:lts |
| Kubernetes | Helm chart: helm install jenkins jenkins/jenkins |

**4. Pre-packaged Solutions**

* Jenkins X (Cloud-Native Jenkins for Kubernetes)
* OpenShift Jenkins (Red Hat’s PaaS solution)

**3.How to Install Jenkins?**

**1. On Ubuntu/Debian (Linux)**

*# Install Java (required)*

sudo apt update

sudo apt install openjdk-11-jdk -y

*# Add Jenkins repo & install*

curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null

echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt update

sudo apt install jenkins -y

*# Start Jenkins*

sudo systemctl start jenkins

sudo systemctl enable jenkins

Access Jenkins at: http://<SERVER\_IP>:8080

**2. On Windows**

1. Download the Windows installer from [jenkins.io](https://www.jenkins.io/download/).
2. Run the .msi file and follow the setup wizard.
3. Jenkins runs as a Windows service (accessible at http://localhost:8080).

**3. Using Docker**

docker run -d -p 8080:8080 -p 50000:50000 --name jenkins -v jenkins\_home:/var/jenkins\_home jenkins/jenkins:lts

**Access at: http://localhost:8080**

Conclusion

✅ Jenkins runs on Linux, Windows, macOS, Docker, Kubernetes, and major clouds (AWS/Azure/GCP).

🚀 Next Steps:

* After installing, unlock Jenkins (cat /var/lib/jenkins/secrets/initialAdminPassword on Linux).
* Install recommended plugins

**4.AWS:Comparison Table where We can install Jenkins**

| **Method** | **Best For** | **Cost** | **Scalability** | **Maintenance** |
| --- | --- | --- | --- | --- |
| **EC2** | Full control, custom setups | $$$ | Manual | High |
| **ECS/EKS** | Docker/Kubernetes users | $$$$ | High | Medium |

**ECS (Docker-native)**

**EKS (Managed Kubernetes)**

**5.Azure:Comparison Table**

| **Method** | **Best For** | **Cost** | **Scalability** | **Persistence** |
| --- | --- | --- | --- | --- |
| **Azure VM** | **Full control, production** | **$$$** | **Manual** | **Yes (Disks)** |
| **AKS** | **Kubernetes-native CI/CD** | **$$$$** | **Auto** | **Yes (PVC)** |
| **ACI** | **Temporary/dev testing** | **$** | **No** | **No** |
|  |  |  |  |  |

**1.Below any file need to run CI pipeline Without JENKINSFILE**

**A. Freestyle Project (Jenkins without needing a Jenkinsfile (using Freestyle jobs)**

If your pipeline only runs tests (via pytest), it will:

* Install dependencies from requirements.txt
* Run your test suite
  + Add a **shell build step**:

pip install -r requirements.txt

pytest

* **Output**: Test results (console logs/JUnit reports)
* **No build artifacts** (like .tar.gz or .whl) are created by default.

**B: Using setup.py : Ensure your project has a setup.py file (minimal example) or pyproject.toml and DON’T want to use jenkinsfile**

from setuptools import setup

setup(

name="your-package",

version="0.1",

packages=["your\_module"],

)

*# Install build tools*

**pip install setuptools wheel**

*# Generate .tar.gz and .whl in dist/ folder*

**python setup.py sdist bdist\_wheel**

* + This creates:
    - dist/your-package-0.1.tar.gz (source distribution)
    - dist/your\_package-0.1-py3-none-any.whl (universal wheel)

**C. GitHub Actions (.github/workflows/python-ci.yml)**

**File Needed**: .github/workflows/python-ci.yml  
**Best for**: Projects hosted on GitHub.

**Example Workflow** (Runs tests & builds .tar.gz/.whl):

name: Python CI

on: [push, pull\_request]

jobs:

test:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v4

- name: Set up Python

uses: actions/setup-python@v5

with:

python-version: "3.10"

- name: Install dependencies

run: |

python -m pip install --upgrade pip

pip install -r requirements.txt

pip install pytest

- name: Run tests

run: pytest

build:

needs: test

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v4

- name: Build package

run: |

pip install setuptools wheel build

python -m build

- name: Upload artifacts

uses: actions/upload-artifact@v3

with:

path: dist/

**Features**:

✅ Runs on every git push/PR  
✅ Tests Python code  
✅ Builds tar.gz and .whl files  
✅ Stores build artifacts

**Phase 1: Continuous Integration (CI)**

Goal: Automate code integration and testing.

When to Implement: First (foundational step).

**Phase 2: Continuous Delivery (CD)**

Goal: Automate deployment preparation but require manual approval.

When to Implement: After CI is stable.

**Phase 3: Continuous Deployment (CD)**

Goal: Fully automate deployments to production.

When to Implement: Only after CI and Continuous Delivery are flawless.

**2.Below file need to run CI pipeline with JENKINSFILE**

**1. Core Files Required**

| **File** | **Purpose** |
| --- | --- |
| **Jenkinsfile** | **Defines the CI/CD pipeline (stages for test, build, deploy, etc.).** |
| **setup.py** | **Required for building Python packages (source .tar.gz and wheel .whl).** |
| **requirements.txt** | **Lists Python dependencies (used for pip install).** |
| **tox.ini (optional)** | **Standardized local/test environment configuration (if using tox).** |

**Supporting Files**

**A.Jenkinsfile**

**groovy**

pipeline {

agent {

label 'spark' // Ensure Spark/Python are installed

}

stages {

**stage('Checkout') {**

steps {

git 'https://github.com/your-repo/pyspark-ci-project.git'

}

}

**stage('Install Dependencies') {**

steps {

sh 'python -m pip install -r requirements.txt'

}

}

**stage('Test') {**

steps {

sh 'python -m pytest tests/' // Run unit tests

}

}

**stage('Build') {**

steps {

sh 'python setup.py sdist bdist\_wheel' // Create .tar.gz and .whl

}

}

**stage('Publish Artifact') {**

steps {

// Publish to Artifactory/Nexus/PyPI (example: Twine)

withCredentials([usernamePassword(

credentialsId: 'nexus-creds',

usernameVariable: 'USER',

passwordVariable: 'PASSWORD'

)]) {

sh 'twine upload --repository-url https://your-repo.com dist/\* --username $USER --password $PASSWORD'

}

}

}

}

}

**B. setup.py (Minimal Example)**

from setuptools import setup, find\_packages

setup(

name="your\_project",

version="0.1",

packages=find\_packages(),

install\_requires=open("requirements.txt").read().splitlines(),

)

**C. requirements.txt**

pytest>=7.0.0

setuptools>=60.0.0

wheel>=0.37.0

**D. tox.ini (Optional for Local Testing)**

[tox]

envlist = py310

[testenv]

deps =

-rrequirements.txt

commands =

pytest

**3.After your Jenkins CI pipeline runs successfully and generates build artifacts (e.g., .tar.gz or .whl files), you need to store them in an artifact repository for versioning, sharing, and deployment. The most common tools for this are:**

1. **JFrog Artifactory (Recommended for Enterprises)**
2. **Nexus (Alternative to Artifactory)**
3. **Amazon S3 / Google Cloud Storage (For Raw Storage)**
4. **AWS CodeArtifact**
5. **Google Artifact Registry**
6. **Azure Artifacts**

**3. Docker-Based CI Pipeline (Run in Containers)**