

# Jenkins Fundamentals

## Jenkins Fundamentals:

### Objectives:

- Understand what Jenkins is and how it works
- Set up Jenkins on an EC2 instance
- Run basic Freestyle and Pipeline jobs

### 1. What is Jenkins?

Jenkins is an **open-source automation server** used to:

- Build, test, and deploy applications
- Automate any part of your software delivery process (CI/CD)
- Integrate with hundreds of tools (GitHub, Docker, AWS, Terraform, etc.)

### Key Features:

Feature	Purpose
Freestyle Jobs	GUI-based jobs with basic logic
Pipeline Jobs	Code-defined pipelines using Jenkinsfile
Plugins	Extend Jenkins (e.g., Git, Docker, AWS CLI, Slack)
Distributed Builds	Run builds on multiple agents (Jenkins master/agent)

## Feature

## Purpose

Webhook  
Integration

Trigger jobs automatically from  
GitHub/GitLab etc.

## 2. Provision Jenkins on AWS EC2

We'll use an **Ubuntu-based EC2 instance** to host Jenkins.

### Step-by-Step Setup

#### 1. Launch EC2 Instance:

- AMI: Ubuntu
- Instance type: t2.micro(free-tier)
- Key pair: Use an existing one or create a new one
- Security Group:
  - Allow: **SSH (22), HTTP (80), Jenkins UI (8080)**
  - Optional later: **HTTPS (443)**

**Name and tags** Info

Name:  [Add additional tags](#)

**▼ Application and OS Images (Amazon Machine Image)** Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Recents **Quick Start**

Amazon Linux macOS **Ubuntu** Windows Red Hat SUSE Linux Debian

[Browse more AMIs](#)  
Including AMIs from AWS, Marketplace and the Community

**▼ Summary**

**Number of instances** Info:

**Software Image (AMI)**: Canonical, Ubuntu, 24.04, amd64 noble image  
ami-0e35ddab05955cf57

**Virtual server type (instance type)**: t2.micro

**Firewall (security group)**: New security group

**Storage (volumes)**: 1 volume(s) - 8 GiB

[Cancel](#) [Launch instance](#) [Preview code](#)

**i-0336a7dc06b171b97 (jenkins)**

[sg-05762e8f14f696a95 \(launch-wizard-1\)](#)

**▼ Inbound rules**

Name	Security group rule ID	Port range	Protocol	Source	Security group
-	sgr-0815240d39be4273c	8080	TCP	0.0.0.0/0	<a href="#">launch-wizard-</a>
-	sgr-03a586c56c97d3e05	80	TCP	0.0.0.0/0	<a href="#">launch-wizard-</a>
-	sgr-003f9939f37b9b93e	22	TCP	0.0.0.0/0	<a href="#">launch-wizard-</a>

## 2. Connect via SSH:

```
ssh -i /path/to/your-key.pem ubuntu@<your-ec2-public-ip>
```

## 3. Install Java (Jenkins dependency):

```
sudo apt update
```

```
sudo apt install -y openjdk-17-jdk
```

## 4. Install Jenkins:

```
-> curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.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 -y jenkins
```

## 5. Start and enable Jenkins:

```
sudo systemctl enable jenkins
```

```
sudo systemctl start jenkins
```

## 6. Access Jenkins UI:

- Go to: <http://<your-ec2-public-ip>:8080>

- Unlock Jenkins with:

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

## Getting Started

# Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

.....

[Continue](#)

- Install **Suggested Plugins**
- Create **admin user**

## Getting Started

# Create First Admin User

Username

ajinkya

Password

.....

Confirm password

.....

Full name

ajinkya

E-mail address

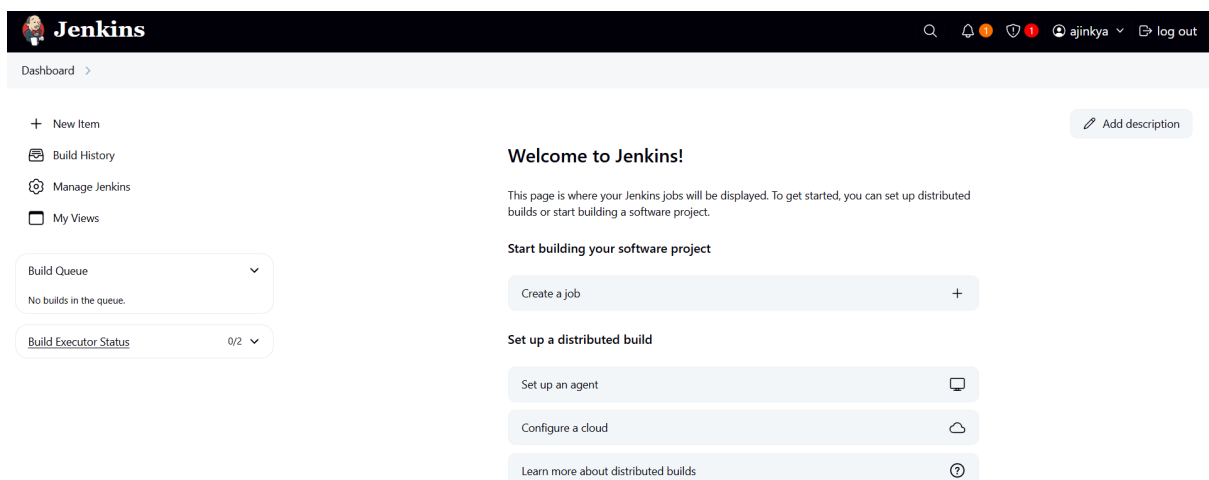
ajinkya@gmail.com

Jenkins 2.504.1

[Skip and continue as admin](#)[Save and Continue](#)

### 3. Jenkins UI Concepts

Component	Description
Dashboard	View and manage all jobs
New Item	Create jobs (freestyle, pipeline, multi-branch, etc.)
Build Now	Trigger a job manually
Build History	See past builds (logs, console output, status)
Configure	Set up job details (SCM, build steps, triggers)
Manage Jenkins	Global settings, plugins, nodes, security, etc.



# Practical 01: First Jenkins job using freestyle and pipeline

## a) Freestyle Job

- Create job → “Freestyle Project”

### New Item

Enter an item name

pr-01-freestyle

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

- Add a **Build Step** → Execute Shell

- Example script:

```
echo "Hello from Jenkins!"
```

### Configure

- General
- Source Code Management
- Triggers
- Environment
- Build Steps**
- Post-build Actions

- ☐ Terminate a build if it's stuck
- ☐ With Ant ?

#### Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

##### Execute shell ?

Command

See [the list of available environment variables](#)

```
echo "Hello from Jenkins!!!"
```

Advanced ▾

Add build step ▾

Save

Apply

- Save and build the job using build now.
- Go to build #1 to see console output and our message.

Dashboard > pr-01-freestyle > #1 > Console Output

Status

</> Changes

Console Output

Edit Build Information

Delete build '#1'

Timings

### Console Output

Download

Copy

View as plain text

```
Started by user ajinkya
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/pr-01-freestyle
[pr-01-freestyle] $ /bin/sh -xe /tmp/jenkins1423071144167938310.sh
+ echo Hello from Jenkins!!!
Hello from Jenkins!!!
Finished: SUCCESS
```

## b) Pipeline Job

- Create job → “Pipeline”

### New Item

Enter an item name

pr-01-pipeline

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

- Paste in this example Jenkinsfile:

```
pipeline {  
  agent any  
  stages {  
    stage('Greet') {  
      steps {  
        echo 'Hello from a pipeline!'  
      }  
    }  
  }  
}
```

- Save and build the job using build now.
- Go to build #1 to see console output and our message.

Dashboard > pr-01-pipeline > #1

- Status
- Changes
- Console Output
- Edit Build Information
- Delete build '#1'
- Timings
- Pipeline Overview
- Restart from Stage
- Replay
- Pipeline Steps
- Workspaces

✓ Console Output

Download

Copy

View as plain text

Started by user [ajinkya](#)

[Pipeline] Start of Pipeline

[Pipeline] node

Running on [jenkins](#) in /var/lib/jenkins/workspace/pr-01-pipeline

[Pipeline] {

[Pipeline] stage

[Pipeline] { (Greet)

[Pipeline] echo

Hello from a pipeline!

[Pipeline] }

[Pipeline] // stage

[Pipeline] }

[Pipeline] // node

[Pipeline] End of Pipeline

Finished: SUCCESS

[www.linkedin.com/in/ajinkya-pame-4a752b346](https://www.linkedin.com/in/ajinkya-pame-4a752b346)

We will see below things further,

## **Core Jenkins Skills for DevOps Jobs**

### **Goals:**

- Master pipeline scripting (Declarative + Scripted)
- Use Git, GitHub, Docker, and AWS CLI in Jenkins
- Automate builds, tests, and deployments

### **Projects:**

#### **1. CI/CD for a Node.js or Python app**

- GitHub integration (webhook)
- Build → Test → Dockerize → Deploy (to EC2 or ECS)

#### **2. Use Jenkins Shared Libraries**

- Write reusable pipeline functions

