

# JENKINS

BASICS

**GAURAV SHARMA**

<https://www.linkedin.com/in/gauravsharma2988/>

# AGENDA

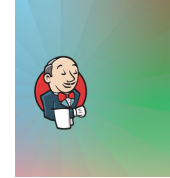


- What is Jenkins ?
- CI tool and examples
- Difference between Bamboo and Jenkins
- Features of Jenkins
- How to setup jenkins server/client?
- What are Jenkins plugins ?
- What is a jenkins job?
- What is upstream/downstream job ?
- 🌟 How to take a backup of a job for recovery ?
- Different types of projects in jenkins
- What is a view ?
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post Build Actions
- Groovy scripting
- Jenkinsfile
- Jenkinsfile in-detail
- Jenkins pipeline
- Blueocean
- Jenkinsfile scenario



# What is Jenkins ?

- A self-contained open source automation server
- Jenkins architecture is designed for distributed build environments.
- Jenkins controller is the original node in the Jenkins installation. It administers the Jenkins agents and orchestrates their work, including scheduling jobs on agents and monitoring agents.
- Latest version of Jenkins is Version 2.426.2 (as of Jan 2024)
- Java-based and can be installed on a variety of os
- Web-based UI
- Can be used to automate all sorts of tasks related to building, testing and delivering or deploying software
- Official jenkins documentation [link](#)



Sign in to Jenkins

Username

Password




☐ Keep me signed in

Sign in



# CI tool & Examples

There are different CICD tools available in market. Few of them are :

- GitLab CICD :  **GitLab**
  - Built-in continuous integration and continuous deployment platform provided by GitLab
  - Tightly integrated with GitLab providing an end-to-end solution for managing the software development lifecycle
- Azure Devops 
  - Set of development tools and services provided by Microsoft as part of the Azure cloud platform
  - Offers a comprehensive suite of tools to support the entire development lifecycle, from planning and coding to building, testing, and deploying applications
  - Key components : Azure Repo, Azure Boards, Azure Pipeline etc
- Bamboo 
  - Continuous integration and continuous deployment (CI/CD) server developed by Atlassian
  - Designed to automate the build, test, and deployment processes for software applications, helping development teams to streamline their workflows
  - Particularly well-suited for organizations already using other Atlassian products, as it seamlessly integrates into the broader Atlassian ecosystem

# Difference between Bamboo and Jenkins



**Bamboo** : Bamboo is a commercial product, and its usage typically involves a licensing fee. There are different licensing options based on the number of build agents and users.

**Jenkins** : Jenkins is open-source and free to use. There are no licensing fees, making it a cost-effective choice for many organizations.

## 1 Vendor and Ecosystem

## 2 Licensing Model

## 3 Community & support

## 4 Plugin Ecosystem

**Bamboo** : Developed by Atlassian, Bamboo is part of the Atlassian suite of tools, which includes Jira, Confluence, Bitbucket, and others. It integrates seamlessly with other Atlassian products, providing a cohesive ecosystem for development and project management.

**Jenkins** : Jenkins is an open-source project with a large and active community. It is not tied to any specific vendor, and its extensive plugin ecosystem allows integration with a wide variety of tools and services.

**Bamboo** : While Bamboo has a marketplace for add-ons and plugins, it may not have the same extensive library of plugins as Jenkins. The Atlassian Marketplace provides various integrations, but the options might be more limited compared to Jenkins.

**Jenkins** : Jenkins has a vast and active plugin ecosystem with thousands of plugins available. This makes it highly extensible and allows integration with a wide range of tools, technologies, and services.

**Bamboo** : While Atlassian provides support for Bamboo, the community may be smaller compared to Jenkins. Users rely on Atlassian's support channels for assistance.

**Jenkins** : Jenkins has a large and active community that contributes to ongoing development and provides support through forums, mailing lists, and various online resources.



# Features of Jenkins

**Continuous Integration (CI)** : Jenkins automates the integration of code changes from multiple contributors, ensuring that the codebase is continuously built and tested.

**Extensibility with Plugins** : Jenkins has a vast ecosystem of plugins that extend its functionality, allowing seamless integration with various tools and technologies.

**Distributed Builds** : Jenkins supports the distribution of build and test tasks across multiple machines, enabling parallel execution and faster build times.

**Flexible Workflow Creation** : Jenkins provides a customizable workflow system, allowing users to define and automate complex build and deployment pipelines based on project requirements.

**Wide Range of Integrations** : Jenkins can integrate with version control systems, build tools, testing frameworks, and deployment platforms, making it adaptable to different development stacks.

**Real-time Monitoring and Reporting** : Jenkins offers real-time monitoring of build activities, along with detailed logs and reports, helping developers and teams identify issues and track build progress.



# How to setup jenkins server?

- Jenkins setup on mac :

Latest LTS version installation -

brew install jenkins-lts -- [Installs jenkins-lts](#)

brew services start jenkins-lts - [Starts the jenkins-lts service](#)

Jenkins successful installation verification -

brew services list -- [Lists the installed services](#)

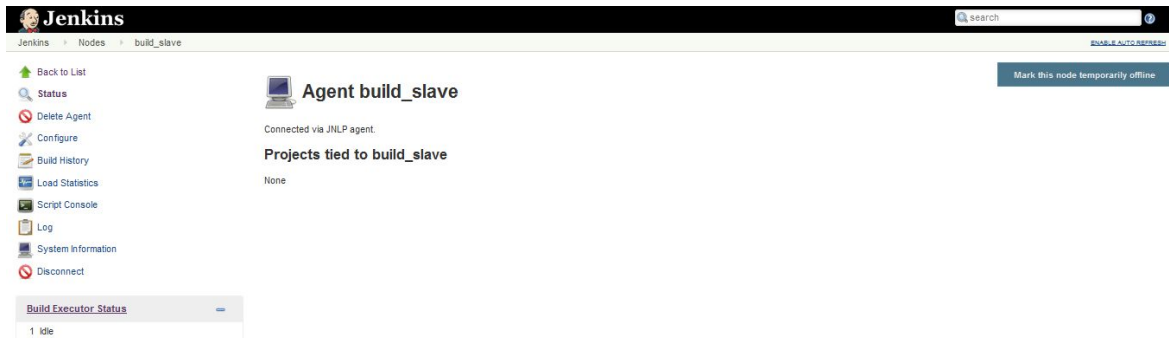
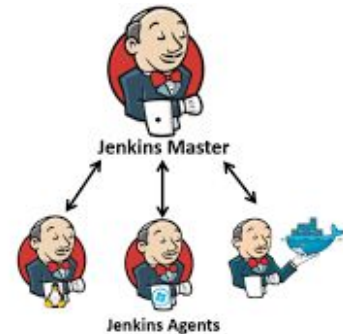
ps -A | grep -i jenkins -- [Lists the jenkins pid](#)

- Jenkins setup on windows : [Link](#)
- Jenkins setup on linux : [Link](#)

# How to setup jenkins agent?



- Jenkins agents can be launched in physical machines, virtual machines, Kubernetes clusters, and with Docker images
- Jenkins agent setup [Link](#)

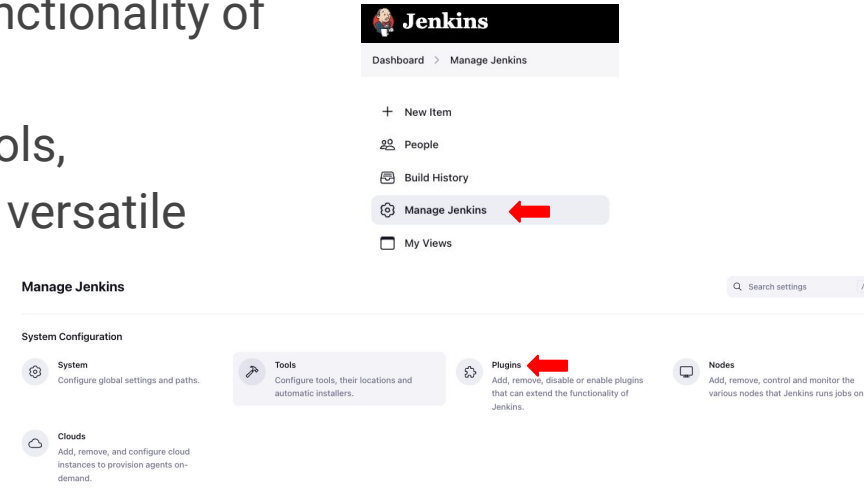




# What are Jenkins plugins ?



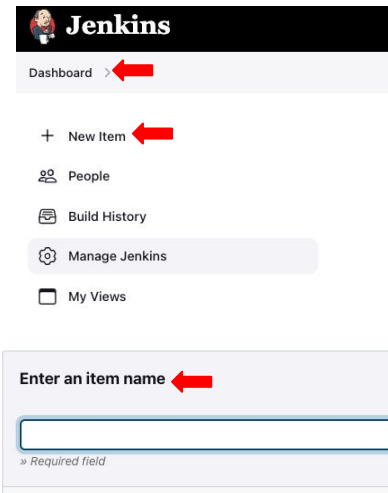
- Extensions or add-ons that enhance the functionality of the Jenkins automation server
- Enable Jenkins to integrate with various tools, technologies, and services and making it a versatile platform for CI/CD deployment
- Common categories of plugins :
  - Source Code Management (SCM) - Git,SVN etc
  - Build Tool Integration - Maven,Gradle etc
  - Notification and Reporting - Slack
  - Containerization and Orchestration - Docker,K8s etc
  - Integration with Issue Tracking and Collaboration Tools - JIRA etc
  - Monitoring and Metrics Plugins - Prometheus





# What is a Jenkins job ?

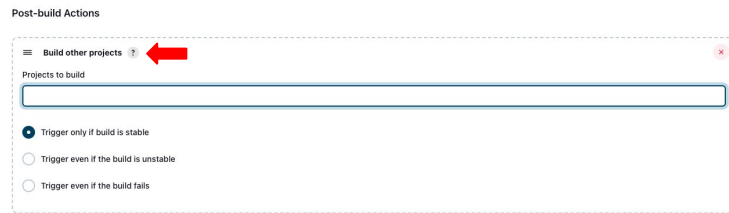
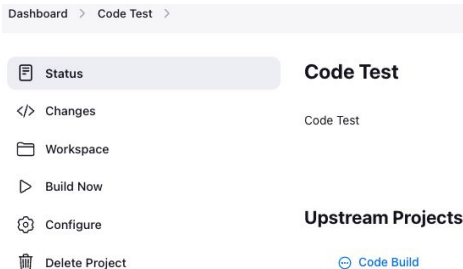
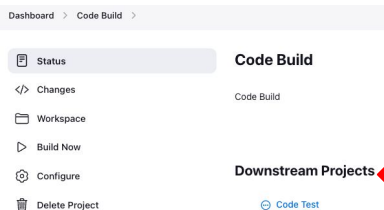
- Fundamental unit of work that represents a task or a set of tasks to be executed as part of a build or deployment process.
- Each job defines a specific set of actions that Jenkins should perform such as compiling code, running tests, or deploying an application.
- Key characteristics :
  - **Configuration** : Configured through a web-based interface or by defining a configuration file (often written in XML or YAML)
  - **Build Steps** : Consist of one or more build steps which represent the individual tasks to be executed.
  - **Build Environment** : Each job runs in a specific environment that can be configured based on the requirements of the build or deployment process.
  - **Triggers** : Jobs can be triggered manually by users or automatically based on certain events.
  - **Build Artifacts** : Jobs may produce build artifacts, which are the outputs generated during the build process.
  - **Post-Build Actions** : After the main build steps are completed, a job can perform post-build actions, such as archiving artifacts, triggering downstream jobs, or sending notifications about the build status.
  - **Build History** : Jenkins maintains a build history for each job, allowing users to review past builds, view build trends, and access information about previous executions.





# What is upstream/downstream job ?

- This configuration allows to create a workflow where changes in one part of the process automatically trigger subsequent jobs, ensuring a streamlined and automated pipeline
- Upstream\Downstream relationships between jobs can be setup through the Jenkins UI
  - **Upstream Configuration** : In the configuration of an upstream job, you can specify which downstream jobs should be triggered based on the trigger condition
  - **Downstream Configuration**: In the configuration of a downstream job, you can specify which upstream job or jobs should trigger



# How to take a backup of a jenkins job for recovery ?




- When a jenkins job is created , a corresponding folder(workspace) is created for every job in the path \$JENKINS\_HOME/jobs/
- All the details of a particular job are stored in the corresponding config file located in the job specific workspace
- Job specific folder located at \$JENKINS\_HOME/jobs/{Job-Name} can be backed up to recover from any unexpected incident
- While recovering, the backed up jenkins job folder can be copied to the location \$JENKINS\_HOME/jobs/ in jenkins server and jenkins service needs to be restarted to get this job listed in the jenkins UI.




# Different types of projects in jenkins

When creating a Jenkins job, you can choose from various project types or build project templates based on the type of automation or task you want Jenkins to perform. Few of them are :


- Freestyle Project
- Pipeline
- Multiconfiguration Project
- Multibranch Pipeline




**Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.




**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.




**Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



**Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



**Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.

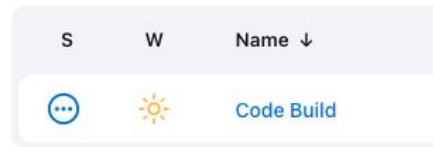
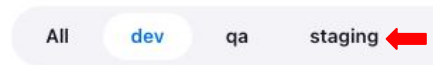


**Organization Folder**  
Creates a set of multibranch project subfolders by scanning for repositories.



# What is a view in Jenkins?

- Tabbed categories that organize content and jobs
- Provide an organizational layer that allows users to group and categorize jobs based on different criteria such as projects, teams, or functionalities
- Displayed on the main dashboard (Screenshot for ref.)
- A view can be created by clicking on the + as referenced in the below screenshot



View in Jenkins





# Source Code Management

- Enables Jenkins to fetch the latest source code from the repository
- Key points
  - SCM Integration : Select the specific SCM tool
  - Repository URL : Provide the URL of the version control repo
  - Credentials : specify the repo credentials
  - Branches/Tags : Specify the branch or tag from the repository that Jenkins should build
  - Polling or Webhooks : Allowing Jenkins to be notified instantly when changes occur

The screenshot shows the 'Add Repository' configuration form in Jenkins. It includes a 'Repository URL' field with a red error message 'Please enter Git repository.' Below it is a 'Credentials' dropdown menu currently set to 'none', with an 'Add' button. An 'Advanced' dropdown is also visible. At the bottom, the 'Branches to build' section shows a 'Branch Specifier (blank for \'any\')' field with '\*master' entered.



# Build Triggers

- Determine when a job should be executed based on certain events or conditions
- Different triggering mechanisms :
  - Manual Build Trigger
  - SCM Polling
  - Webhooks
  - Build after Other Projects
  - Scheduled Builds
  - Triggering on Pull Requests

## Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts) ?
- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☐ GitHub hook trigger for GITScm polling ?
- ☐ Poll SCM ?





# Build Environment

- Allows the configuration of a custom workspace for each job which provides isolation, avoids conflicts with other jobs and allows for precise control over where build artifacts and temporary files are stored
- Contribute to the adaptability and customization, accommodating diverse project requirements and ensuring consistent and reliable builds
- Provides different valuable options which can be used as required in the project

## Build Environment

- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s) ?
- ☐ Add timestamps to the Console Output
- ☐ Inspect build log for published build scans
- ☐ Terminate a build if it's stuck
- ☐ With Ant ?



# Build Steps

Build steps provides a powerful and versatile feature which enables the automation of various tasks and the creation of complex build processes tailored to specific project requirements

Execute Windows batch command

Execute shell

Invoke Ant

Invoke Gradle script

Invoke top-level Maven targets

Run with timeout

Set build status to "pending" on GitHub commit



# Post Build Actions

- Post-build actions allow you to define additional steps that should be performed once the build process has finished
- These actions are configured as part of the job configuration and can include a variety of tasks depending on the specific requirements of your CI/CD pipeline
- Some common post-build actions in Jenkins include:
  - Archiving Artifacts
  - Triggering Downstream Jobs
  - Sending Notifications
  - Publishing JUnit Test Results
  - Publishing HTML Reports
  - Deploying Artifacts
  - Cleaning Up Workspace
  - Setting Build Status

## Post-build Actions

Add post-build action ▼



# Groovy scripting

- Groovy is a scripting language used to define Jenkins pipelines
- Complex workflows, define stages, steps, and conditions, and orchestrate the entire build, test, and deployment process can be created with groovy
- Groovy's extensibility in Jenkins allows users to create custom shared libraries and plugins

```
pipeline {
  agent any

  stages {
    stage('Build') {
      steps {
        echo 'Building...'
        // Your build commands here
      }
    }
    stage('Test') {
      steps {
        echo 'Testing...'
        // Your test commands here
      }
    }
    stage('Deploy') {
      steps {
        echo 'Deploying...'
        // Your deployment commands here
      }
    }
  }
}
```



# Jenkinsfile

- A Jenkinsfile can be written using two types of syntax
  - Declarative
  - Scripted
- Pipeline concepts :
  - **Pipeline** -
    - User-defined model of a CD pipeline.
    - Defines entire build process, which typically includes stages for building, testing and delivering.
    - Key part of Declarative Pipeline syntax.
  - **Node** -
    - Machine which is part of the Jenkins environment and is capable of executing a Pipeline.
    - Key part of Scripted Pipeline syntax.
  - **Stage** -
    - Defines a conceptually distinct subset of tasks performed through the entire Pipeline
  - **Step** -
    - Defines what to do at a particular point in time or "step" in the process



# Jenkinsfile contd.

## Declarative

```
pipeline {
  agent any

  stages {
    stage('Build') {
      steps {
        echo 'Building..'
      }
    }
    stage('Test') {
      steps {
        echo 'Testing..'
      }
    }
    stage('Deploy') {
      steps {
        echo 'Deploying....'
      }
    }
  }
}
```

## Scripted

```
stage('Build') {
  /* .. snip .. */
}

stage('Test') {
  parallel linux: {
    node('linux') {
      checkout scm
      try {
        unstash 'app'
        sh 'make check'
      }
      finally {
        junit '**/target/*.xml'
      }
    }
  },
  windows: {
    node('windows') {
      /* .. snip .. */
    }
  }
}
```

**Declarative** Jenkinsfiles use a simplified, structured syntax with predefined pipeline stages while **Scripted** Jenkinsfiles allow for greater flexibility and expressiveness through a Groovy-based scripting syntax for defining custom pipeline logic.



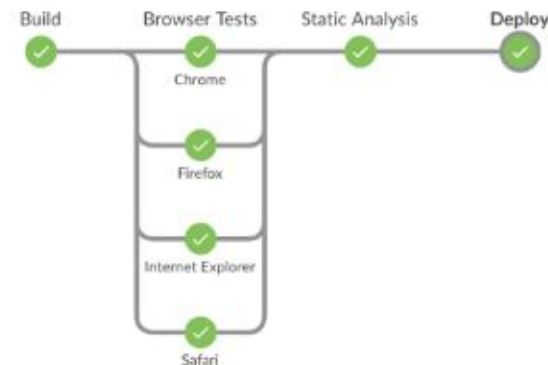
# Jenkins pipeline

- Suite of plugins which supports implementing/integrating continuous delivery pipelines into Jenkins.
- Definition of a Jenkins Pipeline is written into a text file called a Jenkinsfile.
- Foundation of Pipeline-as-code
- Provides a powerful and extensible framework for orchestrating complex CI/CD workflows, fostering collaboration, automation, and traceability throughout the software delivery process
- Benefits from the Blue Ocean plugin, offering a modern and user-friendly visual interface for creating, visualizing, and interacting with pipelines



# BlueOcean

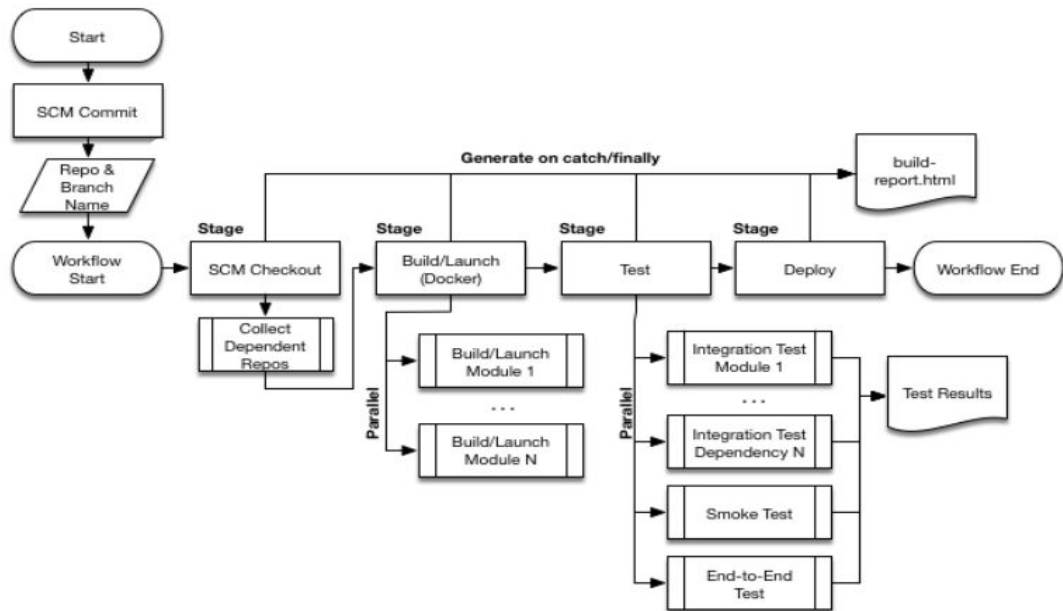
- User interface and visualization plugin for Jenkins
- Provides more modern and intuitive experience for creating, visualizing, and interacting with CI/CD pipelines
- Easy to work with Jenkins pipelines
- Key features
  - Visual Pipeline Editor
  - Pipeline Visualization
  - Integrated Code Editor
  - Pipeline Creation Wizard
  - Rich Pipeline Analytics







# Jenkinsfile : CD scenario modeled in Jenkins Pipeline



# THE END

Devops/SRE-DeepDive

