**Jenkins**

🡪Jenkins is a open-source automation server used for **Continuous Integration and Continuous Deployment**.

🡪It is used to automate the projects without any interference of humans.

🡪Jenkins was originally developed by Kohsuke Kawaguchi, a Japanese software developer, while he was working in the Sun MicroSystems in 2004.

🡪It was initially called Hudson, but named as Jenkins in 2011.

**Why Jenkins?**

* Automates repetitive tasks
* Supports multiple plugins for integration with various tools.
* Enables Continuous Integration/Continuous Deployment (CI/CD).
* Open-source and easy to configure.

**Jenkins used for?**

* Continuous integration for application and infrastructure code.
* Continuous delivery pipelines to deploy the application to different environments using Jenkins pipeline as a code.
* Infrastructure component deployment and management using IaC Tools.
* Run batch operations using Jenkins jobs.
* Run ad-hoc operations like backups, cleanups, remote script execution, event triggers etc.

**CI/CD:**

🡪CI/CD is a DevOps practise that automates software development, testing and deployment.

🡪It ensures faster and more reliable software releases.

**Continuous Integration:**

* It is a process where you integrate a set of tools or set of processes that follow you before delivering the application to the client.

**Key features:**

* Developers push code changes frequently.
* Automated builds and tests ensure code quality.
* Detects and fixes issues early.
* Rapid Feedback.

**Continuous Deployment:**

* It is a process where you deploy/deliver your application to your end user.

**Key features:**

* Code is tested and automatically deployed to production.
* Requires strong test automation to prevent failures.
* Ensures quick and frequent releases.
* Manual deployment
* Deploy-ready code

**Need for CI/CD:**

🡪Jenkins port number is 8080.

🡪Jenkins overcomes the drawback of both Agile and Waterfall methodologies (requirement adoption/accepting).

🡪Jenkins is heart of the DevOps.

🡪We will integrate all the DevOps tools with Jenkins & deploy the applications onto the web server (Tomcat).

🡪We will integrate git, maven, dockers, Kubernetes, ansible, terraform, SonarQube, nexus for deployment and we use another separate instance i.e., Tomcat.

🡪In Jenkins inorder to CI/CD, we need to create jobs/projects first.

🡪Running the jobs/projects is known as ‘Build’.

🡪Build or deployment both are same.

🡪Jenkins provides different types of jobs for automating builds, testing and deployments.

1. **Creating a free style job:**

* Go to Jenkins Dashboard → Click "New Item"
* Enter Job Name → Select "Freestyle Project" → Click "OK"
* Configure Source Code Management:
* Select Git
* Enter the repository URL
* Set the build triggers:
* Check “POLL SCM”.
* Add Build steps:
* Choose **"Execute Shell"** (for Linux/macOS) or **"Execute Windows Batch Command"**
* Save and click “build now”
* **Result**: Jenkins will fetch the code, build, and execute the command.

1. **Creating a Pipeline Job:**

* Go to Dashboard → Click "New Item"
* Enter Job Name → Select "Pipeline" → Click "OK"
* Scroll to Pipeline Section → Select "Pipeline Script"
* Enter Pipeline Script (Example):
* Click "Save" → Click "Build Now"
* **Result**: Jenkins will execute the CI/CD pipeline with **Build, Test, and Deploy** stages.

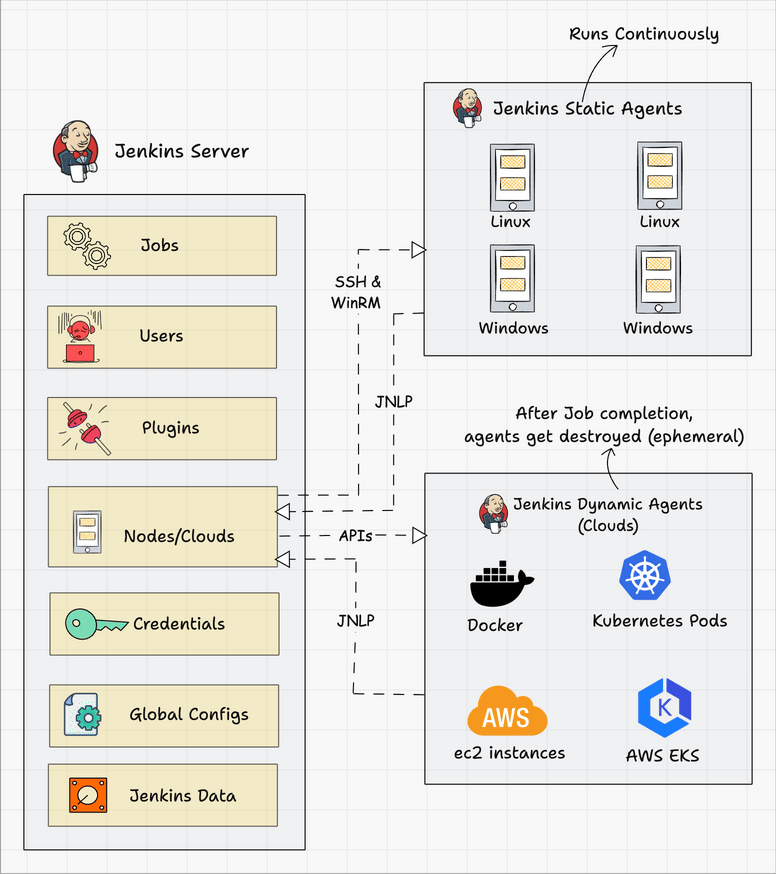
🡪A **Parameterized Job** allows users to provide inputs (parameters) before triggering a build. 🡪This makes Jenkins jobs more flexible by enabling different configurations without modifying the job itself.

**Jenkins important configuration files:**

* Jenkins home directory: /var/lib/Jenkins
* Installed plugins: /var/lib/Jenkins/plugins
* Created job list: /var/lib/Jenkins/workspace
* Nodes info: /var/lib/Jenkins/nodes
* Jenkins log info: /var/lib/Jenkins/log
* List of job: /var/lib/Jenkins/jobs
* Created users list: /var/lib/Jenkins/users

**Jenkins Architecture:**

🡪Jenkins is a master-slave architecture.



* The following are the key components in Jenkins:
* Jenkins Master Node
* Jenkins Agent Nodes/Clouds
* Jenkins Web Interface

1. **Jenkins Master Node:**

* Jenkins server or master node holds all key configurations.
* Jenkins master server is like a control server that orchestrates all the workflow defined in the pipelines.

**Key components in Jenkins Master:**

* **Jenkins Jobs:**
* A job is a collection of steps used to build the source code, test the code, run a shell script, run an Ansible role in remote host.
* It is also called a Jenkins pipeline.
* **Jenkins plugins:**
* Plugins are official and community developed modules that you can install on your Jenkins server.
* You can install the AWS Jenkins plugin and use the abstracted plugin functionalities.
* **Jenkins Global Security:**
* Jenkins’s own user database.
* LDAP configuration.
* SAML Single Sign On(SSO)
* **Jenkins Credentials:**
* When we setup the Jenkins pipelines, there is a scenario it needs to connect to a cloud account, a server, a database, or an API endpoint using secrets.
* Different types of secrets as credentials:

1. Secret Text
2. Username & Password
3. SSH keys

* **Jenkins Data:**
* All the Jenkins data gets stored in the following folder location:

/var/lib/Jenkins/

1. **Jenkins Nodes/Clouds:**

* You can configure multiple agent nodes or clouds for executing Jenkins jobs.
* **Jenkins Global Settings:**
* Under Jenkins global configuration, you have all the configurations of installed plugins and native Jenkins global configurations.
* **Jenkins logs:**
* It provides logging info on all Jenkins server actions including job logs, plugin logs etc.
* **Jenkins agent:**
* These are worker nodes that actually execute all the steps mentioned in a job
* When you create a job, you have to assign an agent to it.
* Every agent has a label as a unique identifier.
* **Jenkins server-agent connectivity:**
* There are 2 ways to connect with the Jenkins master and agent:
* **Using the SSH method:** Uses the SSH protocol to connect to the agent. The port number should be 22 between master and agent. Here the connection gets initiated from the Jenkins master.
* **Using the JNLP method:** Uses the java JNLP protocol (Java Network Launch Protocol). Here the connection gets initiated from the Jenkins Agent. The port assigned is 50000.

**🡪**There are 2 types of Jenkins agents:

1. **Agent Nodes:** These are servers (Windows/Linus) that will be configured as static agents. These agents will be up and running all the time and stay connected to the Jenkins server.
2. **Agent Clouds:** Jenkins cloud agent is a concept of having dynamic agents. Means, whenever you trigger a job, a agent gets deployed as a container on demand and gets deleted once the job is completed.

* **Jenkins Web Interface:**
* [Jenkins 2.0](https://devopscube.com/jenkins-2-tutorials-getting-started-guide/) introduced a very intuitive web interface called "Jenkins Blue Ocean". It has a good visual representation of all the pipelines.

**🡪**In Jenkins we can take the backup.

🡪Jenkins also provides high security.

🡪In Jenkins we can create users.

🡪Jenkins will be automated with the help of created jobs/projects.

**Real-time environments:**

🡪Developers commit the code – Jenkins triggers build – Unit tests run – Integration tests in QA – UAT in staging – Auto or manual deployment in Production.

* **Development Environment:**

🡪Used by developers for frequent code commits.

🡪Jenkins runs unit tests, static code analysis, and build jobs.

* **Testing Environment:**

🡪Runs integration, regression, and performance tests.

🡪Jenkins integrates with tools like Selenium, JUnit, TestNG.

* **Stagging Environment:**

🡪Simulates production-like conditions for final testing.

🡪Jenkins deploy applications for UAT

* **Production Environment:**

🡪It is the live environment where the users can access the application.

🡪Jenkins performs CD for automated rollouts.

**Need of Environments:**

* To ensure an error-free application.
* Productivity needs to be improved.
* Customer satisfaction.
* To deliver the application in time.
* To ensure the security and compliance.

**Jenkins Installation Steps:**

* Create a instance in AWS with the name Jenkins, while creating the instance set it to the t2.medium.
* Connect to the instance using git bash.
* Change to the root-user using:

**sudo su**

* Install java:

**sudo yum install java-1.8.0-amazon-corretto.x86\_64**

* Install git:

**sudo yum install git**

* Install maven:

**sudo yum install maven -y**

* Install Jenkins:

**sudo wget -O /etc/yum.repos.d/jenkins.repo** [**https://pkg.jenkins.io/redhat-stable/jenkins.repo**](https://pkg.jenkins.io/redhat-stable/jenkins.repo)

**sudo rpm --import** [**https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key**](https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key)

**sudo yum install jenkins -y**

* Check the version:

**jenkins –version**

* Start the Jenkins:

**service jenkins start**

**sudo systemctl start jenkins**

**sudo systemctl enable jenkins**

**sudo systemctl status jenkins**

* Java version:

**Java –version** #17.0.14

* Copy the public ip address of the Jenkins instance and paste in the browser with the extension **:8080**
* Now you can see the Jenkins dashboard.

**CI/CD Pipeline:**

🡪It provides feedback to the developers at each and every stage of the building process.

🡪Made procedure too fast than the normal CI/CD automation.

Developers

Feedback

Commit Changes Feedback Feedback Feedback

Customers

Approve & Deploy

CD

Build & Test

CI

Source code control server

fetch

changes

**Before CI:**

Testing

Integration/

Build

GitHub

Repo (VCS)

**After CI:**

Build, Test and

Deploy

GitHub

Repo (VCS)

CI Server

Feedback

**Build Trigger:**

* Build triggers define when and how a Jenkins job should be executed automatically.
* **Types of Build Triggers:**

1. **SCM Polling:**

🡪SCM (Source Code Management) polling is a build trigger in Jenkins that checks a repository for changes at a specified interval.

🡪If changes are found, Jenkins triggers a build automatically.

* Open Jenkins Dashboard – select a job
* Click configure
* Scroll to “Build Triggers” 🡪Check Poll SCM
* Enter a cron expression in the scheduled field.
* Cleck save.

1. **Build Periodically:**

🡪The **Build Periodically** option in Jenkins allows you to schedule jobs at fixed time intervals using a **cron expression**. It is useful for running **nightly builds, weekly reports, or maintenance jobs.**

**🡪**It is a cron based expression.

* Open **Jenkins Dashboard** → Select a **Job**
* Click **Configure**
* Scroll to **"Build Triggers"** → Check **"Build periodically"**
* Enter a **Cron Expression** in the Schedule field
* Click **Save**

1. **Webhooks:**

🡪A **Webhook** is a way for Jenkins to be **instantly notified** when changes occur in a repository

🡪Unlike SCM polling, which **checks for changes at intervals**, a webhook **pushes** the notification to Jenkins, making it faster and more efficient.

* Open **Jenkins Dashboard** → Select a **Job**
* Click **Configure**
* Scroll to **"Build Triggers"**
* Check **"Build when a change is pushed to GitHub"**
* Click **Save**

**Jenkins Plugins:**

🡪Jenkins plugins extend its functionality by adding new features, integrations and automation capabilities.

🡪There are **1,800+ plugins** available for SCM, build tools, cloud services, notifications, and more.

🡪There are 3 ways to create plugins:

1. In jenkins dashboard--->manage jenkins--->manage plugins-->available--->type plugin name.--->install--->
2. We need to download in our local laptops --->upload to jenkins dashboard--->manage jenkins-->manage plugins--->advanced--->upload complete download file----->open --->upload.
3. We need to download the plugins in our local laptop--->copy that plugin to jenkins instance.

this path: -->/var/lib/jenkins/plugins.

--->restart jenkins.