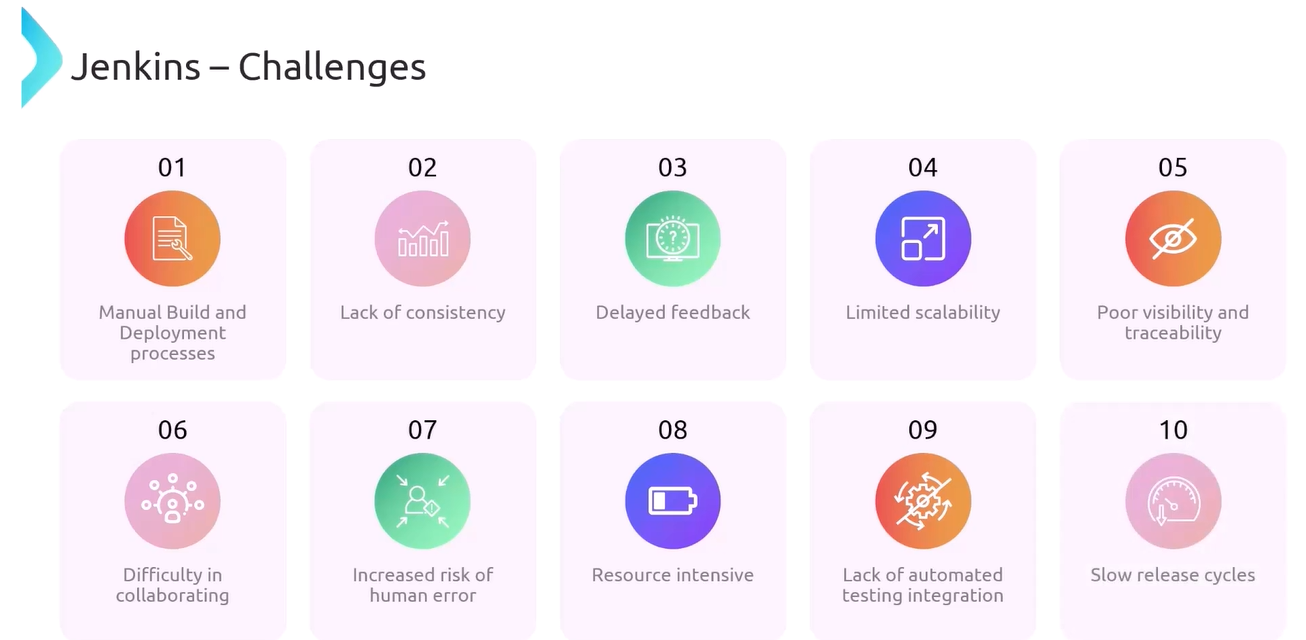
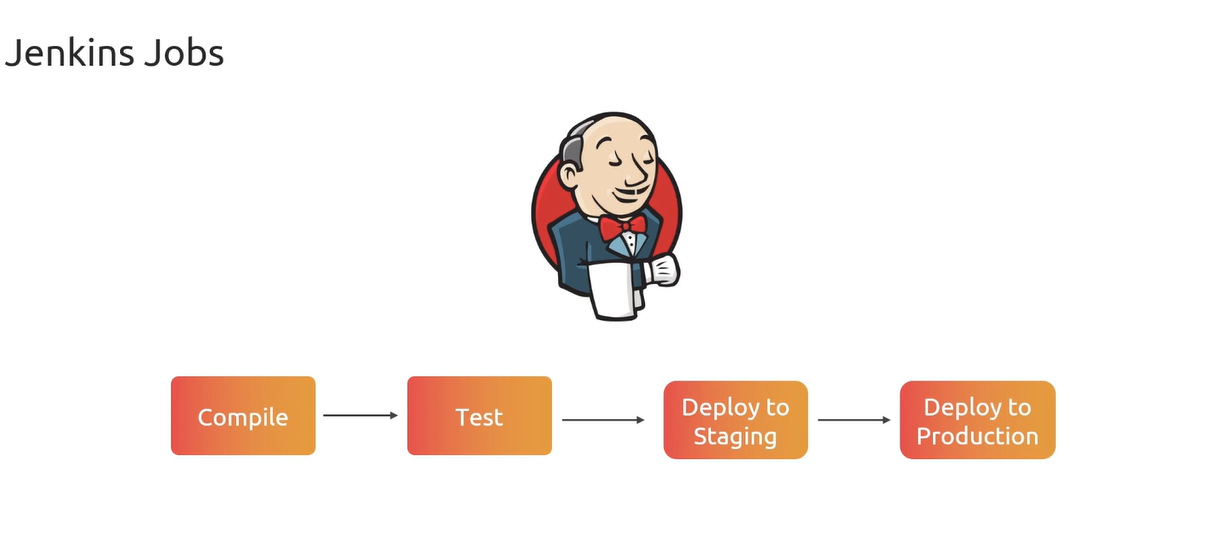
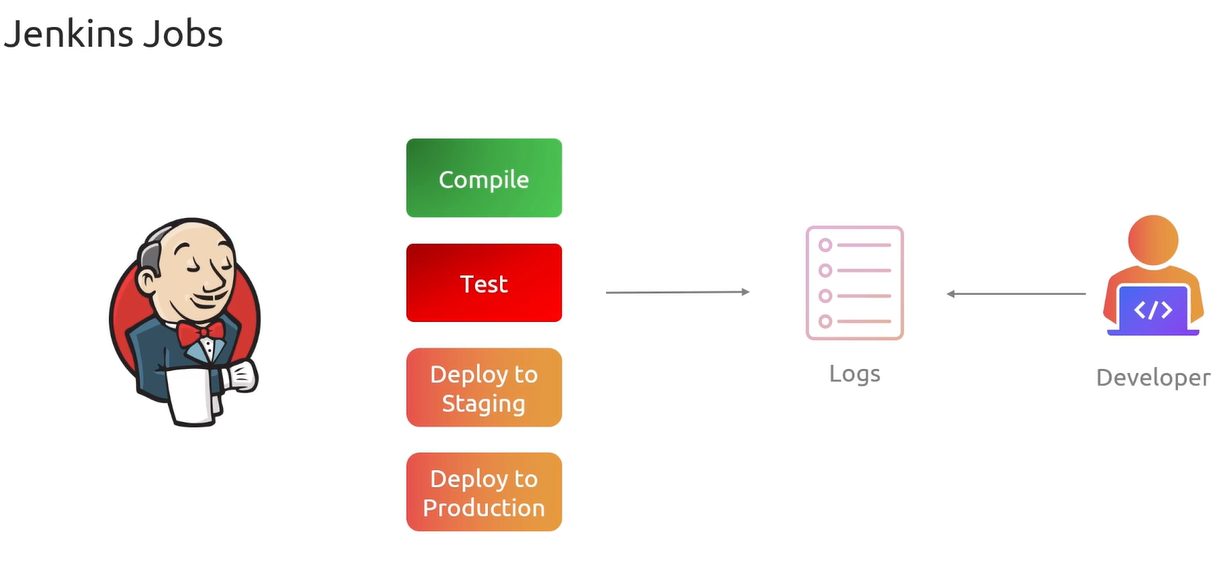
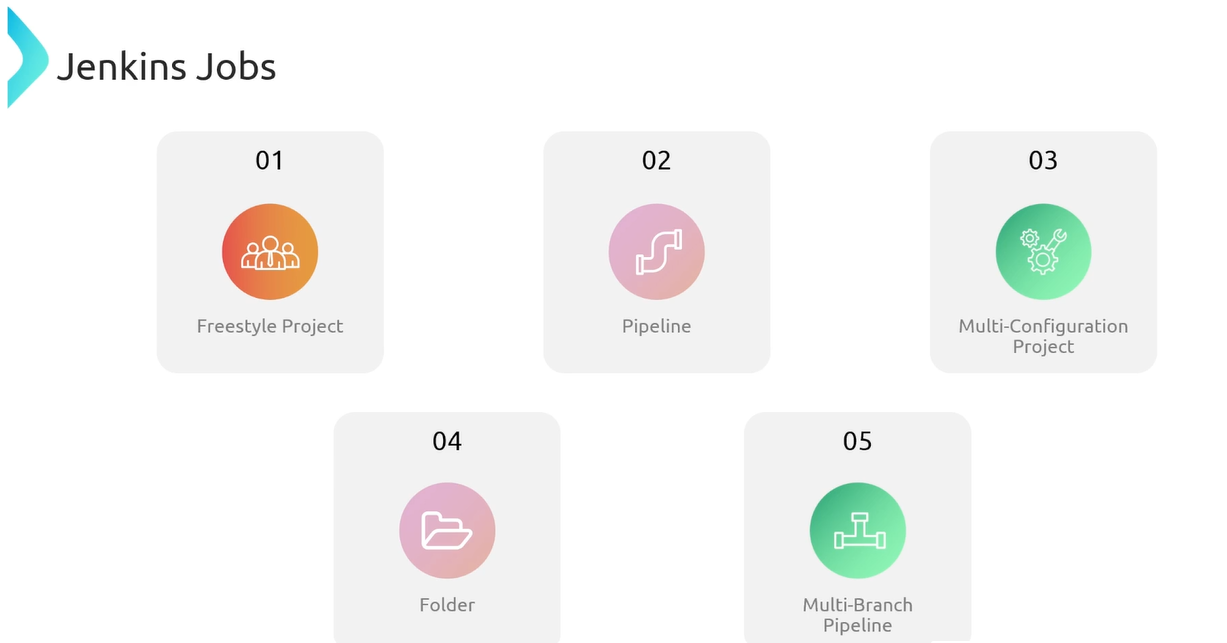


****

****

****

****

**What is Jenkins?**

Jenkins is an open-source automation server widely used for **Continuous Integration (CI) and Continuous Deployment (CD)** in software development. It helps developers automate the build, test, and deployment processes, ensuring efficient and reliable software delivery.

**Key Features of Jenkins**

1. **Continuous Integration & Continuous Deployment (CI/CD)** – Automates the integration and deployment of code.
2. **Extensible with Plugins** – Supports over 1,800 plugins to integrate with tools like Git, Docker, Kubernetes, AWS, and more.
3. **Distributed Builds** – Can execute builds on multiple machines for better performance.
4. **Pipeline as Code** – Uses **Jenkinsfile** (written in Groovy) to define build pipelines.
5. **Easy Integration with SCMs** – Works with Git, GitHub, Bitbucket, SVN, etc.
6. **Job Scheduling & Monitoring** – Automates task execution with cron-like scheduling

**How Jenkins Works?**

1. **Developers commit code** → Jenkins detects changes in version control (Git, SVN).
2. **Jenkins pulls the code** and triggers the **build process**.
3. **Jenkins runs tests** to check the code's correctness.
4. **Artifacts are generated** (JAR, WAR, Docker image, etc.).
5. **Deployment** – The build is deployed to a testing/staging/production server

**Jenkins Architecture**

1. **Master Node** – Controls jobs, scheduling, and distributes builds.
2. **Worker Nodes (Agents)** – Execute builds and report back to the master.
3. **Plugins** – Extend functionality (e.g., Git, Docker, Slack notifications).
4. **Jenkins Pipeline**
5. Jenkins allows defining **CI/CD workflows** using **Declarative** or **Scripted** pipelines.

Example of a Simple Jenkinsfile (Declarative Pipeline)

pipeline {

agent any

stages {

stage('Build') {

steps {

echo 'Building the application...'

}

}

stage('Test') {

steps {

echo 'Running tests...'

}

}

stage('Deploy') {

steps {

echo 'Deploying the application...'

}

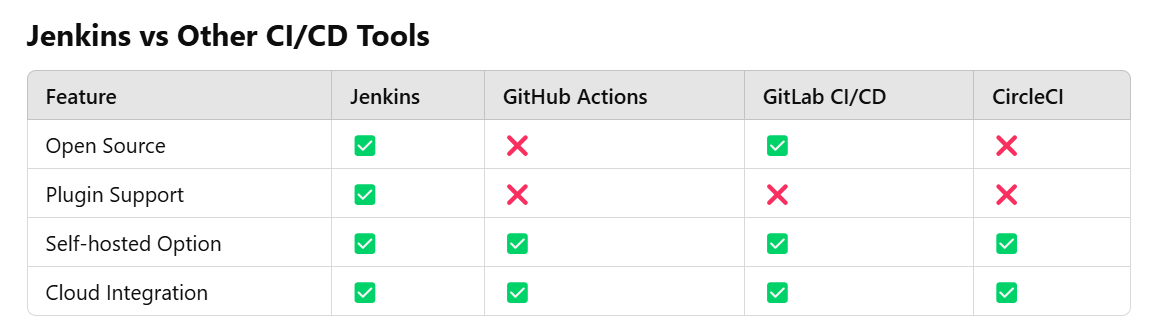
}

}

}

==================  
  
  
**Advantages of Jenkins**

✔ **Free & Open Source**  
✔ **Highly Extensible** with plugins  
✔ **Easy Integration** with DevOps tools  
✔ **Automation** of builds & deployments  
✔ **Improves Code Quality** through automated testing



**Use Cases of Jenkins**

* **CI/CD Automation**
* **Microservices Deployment**
* **Infrastructure as Code (IaC) with Terraform/Ansible**
* **Automated Testing**
* **Containerized Builds (Docker & Kubernetes)**

### ****Conclusion****

Jenkins is a powerful **CI/CD automation tool** that accelerates software development by automating testing, building, and deployment processes. It is highly extensible, widely used in DevOps, and integrates seamlessly with modern cloud and containerized environments.

**Build Triggers in Jenkins**

Build triggers in Jenkins define the conditions under which a job should start automatically. These triggers help automate the build process without manual intervention. Jenkins provides multiple types of build triggers, allowing flexibility in continuous integration and delivery (CI/CD) workflows.

**Types of Build Triggers in Jenkins**

1. **Manual Trigger**
   * A user manually clicks the **"Build Now"** button to start a job.
2. **SCM Polling (Poll SCM)**
   * Jenkins checks the version control system (e.g., Git, SVN) at regular intervals.
   * If changes are detected, the build starts.
   * Example cron syntax for polling every 5 minutes

H/5 \* \* \* \*

 **Webhook (Push Trigger)**

* A Git repository (e.g., GitHub, GitLab, Bitbucket) notifies Jenkins about new commits.
* Jenkins starts the build immediately when a push event occurs.
* Requires configuration in both Jenkins and the version control system.

 **Build After Other Projects (Upstream Trigger)**

* A job starts after another job (upstream job) completes successfully.
* Useful for pipeline workflows where multiple jobs are dependent.

 **Build Periodically**

* Schedules the job to run at specific time intervals using cron syntax.
* Example to run the build every day at midnight:

0 0 \* \* \*

 **GitHub Hook Trigger for GITScm Polling**

* Specifically used for GitHub webhooks.
* Triggers a build when there is a push event in the repository.

 **Trigger Build Remotely (via API)**

* Jenkins provides a URL that external applications or scripts can use to trigger builds.
* Requires setting an authentication token.
* Example API call:

curl -X POST <http://jenkins-url/job/job-name/build?token=TOKEN_NAME>

 **Parameterized Builds**

* Allows triggering a build with custom parameters.
* Useful when builds need specific values, such as branch names or environment variables.

 **Scheduled Triggers in Pipeline (Jenkinsfile)**

* In declarative pipelines, builds can be scheduled inside a pipeline block.

pipeline {

triggers {

cron('H 12 \* \* 1-5') // Runs at 12 PM from Monday to Friday

}

}

**Custom Triggers (Plugins)**

* Jenkins supports plugins that provide additional triggers.
* Example: **Jenkins Job DSL Plugin** allows custom automation rules.

**Choosing the Right Build Trigger**

* **For code changes** → Use **webhooks** or **SCM polling**.
* **For periodic builds** → Use **Build Periodically** or **cron in a pipeline**.
* **For dependent jobs** → Use **Build after other projects are built**.
* **For API-based builds** → Use **Trigger builds remotely**.