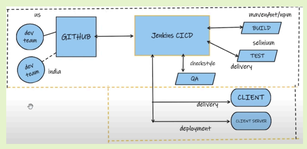
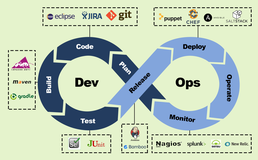
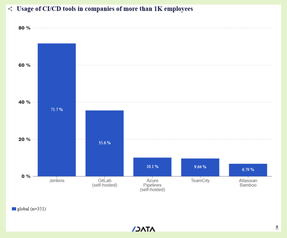
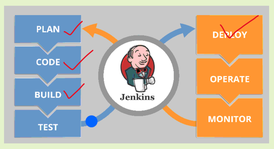
✅ **Jenkins Series Topics**

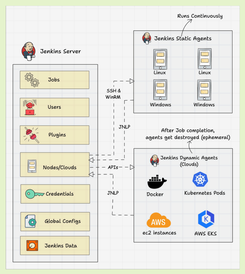
1. **CI/CD Methods Before Jenkins Was Adopted**
2. **What Benefits Does Jenkins Bring That Made It a Popular CI/CD Tool?**
3. **Jenkins Architecture**
4. **Jenkins Server Web UI Overview**
5. **Operating Systems Supported by Jenkins**
6. **Jenkins Installation on a VM (Ubuntu OS)**
7. **Jenkins UI Dashboard and Jobs Overview**
8. **Jenkins Freestyle Project vs Declarative Pipeline**
9. **Jenkins Master-Slave Setup**
10. **Jenkins User Management (RBAC - Role-Based Access Control)**
11. **Post-Build Actions: Email Alerts for Build Success/Failure with Report Attachments**
12. GITHUB Integration - Public and Private Repositories
13. ****Devops Project Jenkins JAVA Maven Docker Tomcat server
14. Jenkins Shared Library jenkins
15. ** Jenkins CI/CD Pipeline: Docker , Kubernetes, SonarQube, Prometheus, and Grafana Integration DevSecOps with OWASP, Trivy, and SonarQube in Jenkins**
16. **Jenkins Backup & Restore - Disaster Recovery.

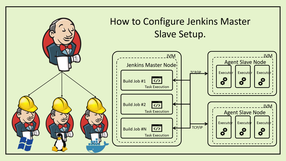






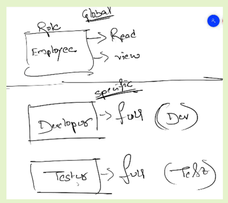






plugin: role base auth stratergy

Why is Jenkins User Management Important?
1. Access Control: User management allows administrators to control access to Jenkins resources.
Different users may have different levels of authorization, ensuring that only authorized personnel can perform critical actions.
2. Security: By managing user accounts,
3. Collaboration: In a collaborative development environment, multiple team members may interact with Jenkins. User management ensures that each team member has the appropriate permissions, fostering collaboration without compromising security.



mvp-01
login
admin
client

Jenkins multistage pipeline

dev — Jenkinsfile
stage — Jenkinsfile
prod — Jenkinsfile

Build Stage
Test Stage
Quality Check Stage
Stagin

build test stage qa

❓ Developer -item role - dev related jobs , after dev it could be anything
Pattern - dev.* -
Provide full access to dev item role

❓ Tester- item role- test related jobs, after test can see only test

Pattern - test.* -
Provide full access to dev item role all

🎯 What is Maven?
Maven is a build automation tool used primarily for Java projects.

🪶 Main Uses:

Compile Java code

Download and manage dependencies (like libraries)

Package the application (JAR/WAR)

Run tests

Generate reports

📄 What is pom.xml in Maven?
pom.xml stands for Project Object Model. It's the core configuration file for any Maven project.

Think of it as the blueprint for your project.

✅ Key Tags Explained:

groupId: Your company or domain name (e.g., com.insight)

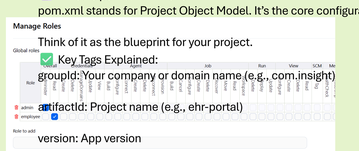
artifactId: Project name (e.g., ehr-portal)

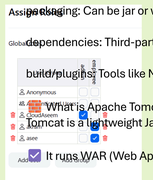
version: App version

packaging: Can be jar or war (WAR is used for web apps)

dependencies: Third-party libraries you need

build/plugins: Tools like Maven Compiler, WAR Plugin





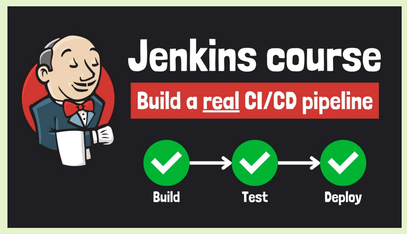
🐱 What is Apache Tomcat?
Tomcat is a lightweight Java web server and servlet container.

✅ It runs WAR (Web Application Archive) files — these are the standard package for Java web apps.

When you:

Build your app with Maven → it creates a .war file

Deploy .war file to Tomcat → Tomcat runs it as a web application



“CI/CD Pipeline with Java | Maven | Jenkins | Tomcat”

🔁 *How Maven, POM, and Tomcat Work Together*
Developer writes Java code and defines dependencies in pom.xml

Maven reads pom.xml, downloads necessary libraries, compiles code, and packages it into a .war file

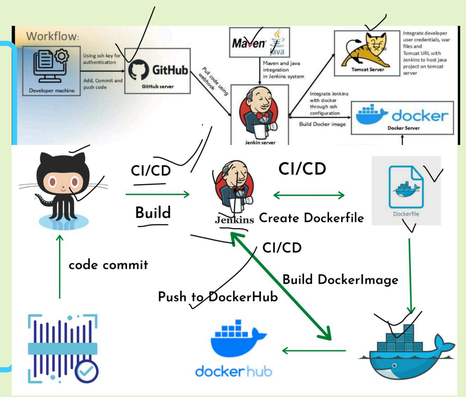
Tomcat takes the .war file and deploys it as a web app accessible via a browser (e.g., http://localhost:8080/myapp)

📌 *Real Example Use Case*
In your Jenkins pipeline:

Maven builds the .war file from the Java code

Jenkins copies this .war into a Tomcat Docker container

Tomcat serves it to the end user



📌 [What is Jenkins Shared Library?
"A Jenkins Shared Library is a way to centralize and reuse code across multiple Jenkins pipelines. Instead of copying and pasting stages or steps in every Jenkinsfile, you write them once in a shared library — and reuse them everywhere!"

Why Use It?

Clean & DRY pipelines (Don't Repeat Yourself)

Easier to maintain

Ideal for large teams and enterprise projects

Folder Structure of a Shared Library

```
(root)
└── vars/
    └── myPipeline.groovy
└── src/
    └── org/
        └── cloudaseem/
            └── utils.groovy
└── resources/
    └── templates/
└── README.md
```

Explanation:

vars/: Stores global pipeline functions (e.g., myPipeline.groovy)

src/: Stores custom classes (like Utils.groovy)

resources/: Stores templates or config files

README.md: Optional, for documentation

How to Create a Shared Library
"Step-by-step guide to creating your own shared library:"

Create a new GitHub repo (e.g., jenkins-shared-library)

Add the folder structure as shown above

Inside vars/myPipeline.groovy:

```
groovy

def call(String name = 'World') {
  echo "Hello, ${name} from Shared Library!"
}
```

Disaster Recovery (DR) strategies for Jenkins

This document outlines essential Disaster Recovery (DR) strategies for Jenkins, ensuring minimal downtime and data loss in the event of failure. The three main methods covered include:

Automated EBS Volume Backups

AMI Backups of Jenkins Server

Jenkins Directory Backup to S3 Bucket and Restore Procedures

1. Automated EBS Volume Backups

Objective: Capture point-in-time snapshots of the Jenkins data volume.

Approach:

Identify the EBS volume attached to the Jenkins EC2 instance.

Use AWS Backup or a Lambda + CloudWatch Events combination to schedule regular snapshots.

Sample Lambda Strategy:

Trigger: CloudWatch Event (Daily/Weekly)

Function: Create snapshot using the create-snapshot API

Tag snapshots for retention policies

Benefits:

Quick volume-level restore

Minimal configuration required

2. AMI Backup of Jenkins Server

Objective: Backup the complete Jenkins EC2 instance (OS + Jenkins config + installed tools).

Steps:

Navigate to EC2 Dashboard > Instances

Select Jenkins instance > Actions > Create Image (AMI)

Provide image name and description

Schedule AMI creation using Lambda + CloudWatch if needed

Best Practices:

Tag AMIs with creation date and retention period

Combine with EBS backup for complete DR coverage

Use Case:

Fast recovery by launching new instance from AMI

```
#!/bin/bash
TIMESTAMP=$(date +%F-%H-%M)
BACKUP_DIR="/var/lib/jenkins"
S3_BUCKET="s3://cloudaseem-jenkins-backup-bucket"
tar -czf /tmp/jenkins-backup-$TIMESTAMP.tar.gz $BACKUP_DIR
aws s3 cp /tmp/jenkins-backup-$TIMESTAMP.tar.gz $S3_BUCKET/
rm -f /tmp/jenkins-backup-$TIMESTAMP.tar.gz
```

Restore Tip:

```
aws s3 cp s3://cloudaseem-jenkins-backup-bucket/jenkins-backup-2025-05-28-19-05.tar.gz .
sudo systemctl stop jenkins
sudo tar -xzf jenkins-backup-2025-05-28-19-05.tar.gz -C /
sudo systemctl start jenkins
```