**Overview of Jenkins & CI/CD Concepts**

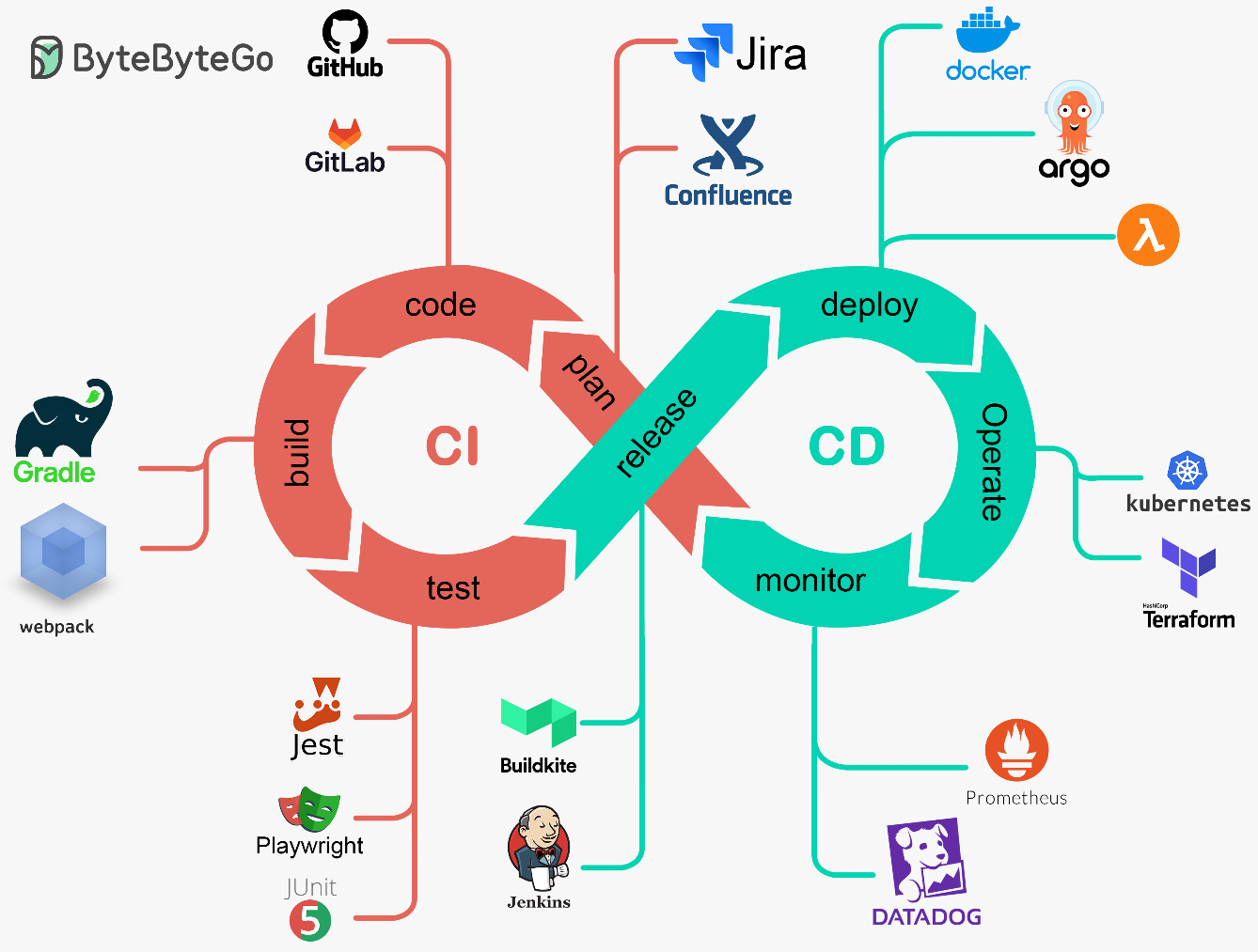
**Learning Objectives:**

* Understand CI/CD and its role in modern software development.
* Explain Jenkins architecture and core concepts.
* Describe how a simple Jenkins pipeline works.
* Read and write basic Jenkinsfile syntax.
* Understand build triggers and automation concepts in Jenkins.

**1. Introduction to CI/CD**

**What is CI/CD?**

* **CI (Continuous Integration):**
  + Developers regularly push code to a shared repository.
  + Automated builds and tests run on every commit.
  + Goal: detect integration issues early.
* **CD (Continuous Delivery/Deployment):**
  + **Continuous Delivery**: Automatically prepares builds for release.
  + **Continuous Deployment**: Automatically deploys every successful build to production.



**Benefits:**

* Faster development cycles
* Better code quality
* Early bug detection
* Reduced integration issues
* Consistent delivery process

**2. Jenkins Overview**

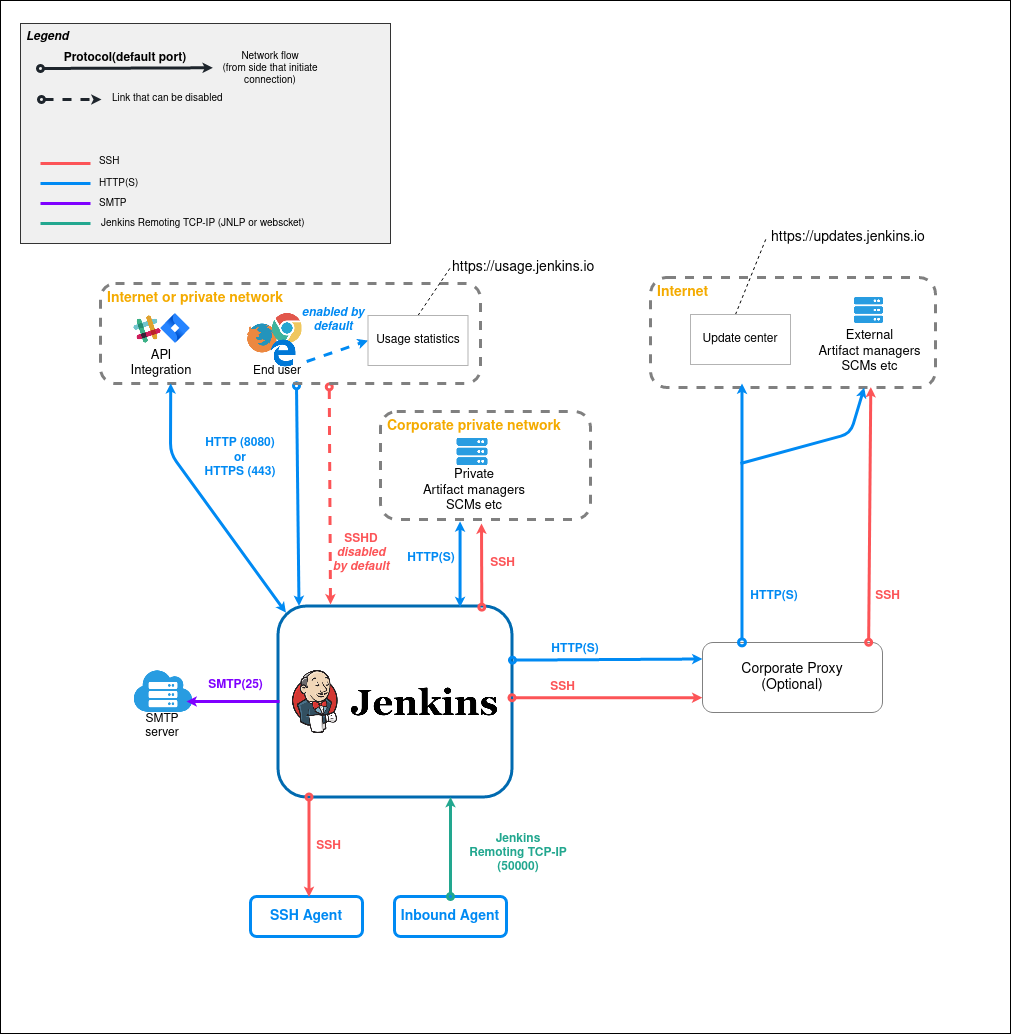
**What is Jenkins?**

* An open-source automation server used for CI/CD.
* Written in Java.
* Extensible via plugins (over 1,800 available).
* Integrates with Git, Docker, Kubernetes, Maven, Gradle, Slack, etc.

**Jenkins Architecture:**

Developer → Git Push → Jenkins (Master) → Build Agents (Workers) → Report/Test/Deploy

* **Master Node:** UI, job scheduling, orchestration.
* **Agents (Slaves):** Execute builds on different environments.
* **Plugins:** Extend functionality (e.g., GitHub, Slack, Docker).



**3. Jenkins Installation Overview**

(Note: Theoretical only — not hands-on.)

**Common Installation Options:**

* WAR file (java -jar jenkins.war)
* System packages (e.g., .deb, .rpm, brew)
* Docker image (jenkins/jenkins)
* Cloud-based solutions (Jenkins X, Jenkins on AWS)

**Post-Install Setup:**

* Admin password & user creation
* Plugin installation wizard
* Setting up the first job

**4. Jenkins Jobs & Freestyle Projects**

**Types of Jobs:**

* **Freestyle Project:** Basic UI-based configuration.
* **Pipeline Project:** Scripted build and deploy flow using a Jenkinsfile.

**Freestyle Project Features:**

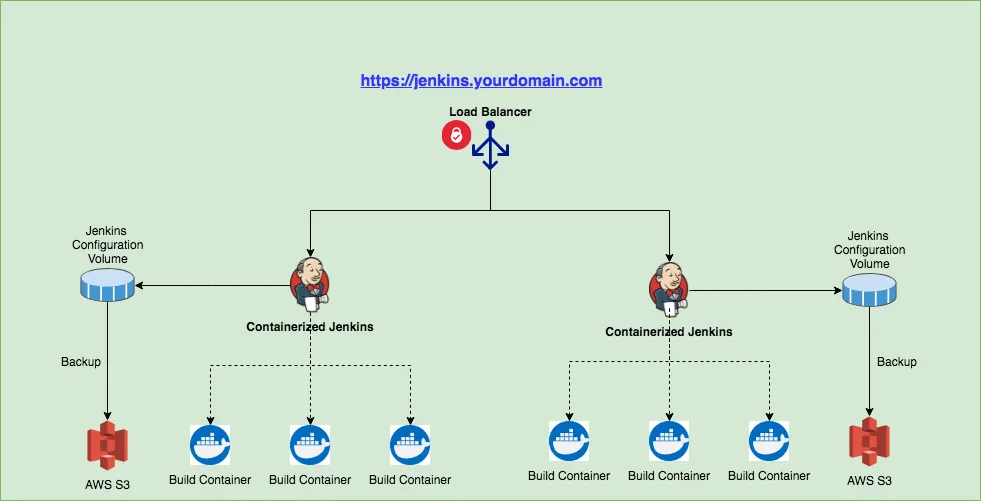
* Source Code Management (SCM): Git, SVN
* Build triggers (poll SCM, webhook, timer)
* Build steps (e.g., shell script, Maven build)
* Post-build actions (e.g., send email, archive artifacts)

**5. Introduction to Pipelines**

**What is a Jenkins Pipeline?**

* A **Pipeline** defines the entire CI/CD process in code (Jenkinsfile).
* Stored in version control with the application code.
* Enables reproducibility, collaboration, and automation.

**Two Types of Pipelines:**

1. **Declarative Pipeline** (Recommended):
   * Simpler, structured syntax.
2. **Scripted Pipeline**:
   * Groovy-based, flexible but complex.

**6. Jenkinsfile Basics**

**What is a Jenkinsfile?**

* A plain-text file that defines a Jenkins Pipeline.
* Typically stored in the root of your project (/Jenkinsfile).

**Basic Declarative Pipeline Syntax:**

pipeline {

agent any

stages {

stage('Build') {

steps {

echo 'Building...'

}

}

stage('Test') {

steps {

echo 'Testing...'

}

}

stage('Deploy') {

steps {

echo 'Deploying...'

}

}

}

}

**Key Elements:**

* pipeline: Starts the pipeline block
* agent: Defines where the pipeline runs (any, docker, specific label)
* stages: High-level steps like Build/Test/Deploy
* steps: Actual shell commands or actions to perform

**7. Triggers & Automation**

**Common Build Triggers:**

* **Manual:** Click "Build Now"
* **SCM Polling:** Jenkins polls Git at intervals (e.g., every 5 mins)
* **Webhooks:** GitHub/GitLab notifies Jenkins when code is pushed
* **CRON jobs:** Time-based scheduled builds

**Example CRON syntax:** H/5 \* \* \* \* → Every 5 minutes

@daily → Once per day

**8. Real-world Use Case Sample**

