Git & Jenkins

PART 1: GIT

Git Repository Basics

- A **Git repository** stores a project's files + revision history and tracks changes over time.
- **Remote repository** → hosted on a server (e.g., GitHub) for collaboration.
- **Local repository** → copy on the developer's machine; developers push work from local → remote.
- Key commands: commit, push, pull.

Git Installation & Setup

- Download from: https://git-scm.com/
- **git config** → first and necessary command on Git CLI.
  - Sets author name and email for commits.
  - git config --global user.name "Name"
  - git config --global user.email "email@example.com"
  - git config --list → view configured details.

Git Init

- **git init <name>** → creates/initializes an **empty local repository**.
- Example: git init demo

Working Tree vs Index (Staging Area)

Concept	Description	Location
Working Tree	Directory with actual project files; reflects current filesystem state including unstaged changes	Local project directory
Index (Staging Area)	Buffer between working tree and repository; holds changes ready to be committed	.git/index

- git add → moves files from **working area → staging area**.
- git commit → moves staged changes to **local repository**.

Key Git Commands

Command	Purpose
git clone <URL>	Copy a remote repository to local machine
git add <filename>	Add specific file to staging area
git add .	Add **all** changed files to staging area
git status	Check status of files (staged/unstaged/untracked)
git commit -m "message"	Save staged changes to local repo with a message
git push	Upload local commits to remote repository
git branch	List all branches
git merge <branch>	Merge specified branch into current branch
git log	View commit history

Git File Status Lifecycle

Untracked → (git add) → Staged
Unmodified → (edit file) → Modified → (git add) → Staged
Staged → (git commit) → Unmodified
Unmodified → (rm file) → Untracked

- **Tracked files** = files Git knows about (from last snapshot or newly staged).
- **Untracked files** = files in working directory NOT in last snapshot and NOT staged.
- After cloning, all files are **tracked and unmodified**.

Git Authentication (Token)

- Git uses a **personal access token** instead of password for git push.
- Path: GitHub → Profile → Settings → Developer Settings → Personal Access Tokens → Tokens Classic → Generate New Token.
- On Windows: verify via **Credential Manager → Windows Credentials → Generic Credentials**.

GIT BRANCHING STRATEGY

What is a Branching Strategy?

- A set of **rules and conventions** for how branches are created, used, merged, and deleted.
- Purpose:
  - Allow **multiple developers** to work simultaneously without conflicts.
  - Keep the **main codebase stable**.
  - Simplify **testing, debugging, and releases**.
  - Improve **collaboration and version control**.

Branch Types

Branch	Description	Lifespan
Master	Main branch; contains final/production-ready version; HEAD always reflects final version	Permanent
Develop	Parallel to master; contains latest development changes for next release; also called **"integration branch"**	Permanent
Feature	Used for developing new features; branches off from develop; deleted after merging back into develop	Short-lived
Release	Created for new version release preparation; deployed to **staging server** for testing; merged back into develop AND master; tagged with version number	Short-lived
Hotfix	Created for **critical bug fixes** in production; branches off from master; merged back into master with a tag	Short-lived

Key Points for MCQs

- **Master and Develop** are the two **main/permanent** branches.
- Feature, Release, and Hotfix are **supporting/short-lived** branches.
- **Release branch** is generally created by **senior developers**.
- Release branch must be merged into **both develop AND master**.
- After merging release → develop, it must be **tagged with a version number**.
- **Hotfix** branches are similar to release branches but arise from **immediate/critical bugs in production**.

PART 2: JENKINS

Jenkins Overview

- **Open-source continuous integration (CI) tool**.
- Automates the software development process, reducing manual effort.
- Achieves CI with the help of **plugins**.
- Automates: source checkout, building, code analysis, testing, and deployment.
- Runs tasks each time a developer changes source code → **detects defects faster**.

Jenkins Benefits

- Large community with regular contributions.
- **Free and open-source** (no subscription cost).
- **Highly configurable** with support for many plugins.
- **Platform-independent** — available for Windows, Linux, macOS.

Jenkins CI Workflow

Developer → Code Repository → Compile/Build → Test → Deploy

Jenkins Installation & Launch

- Download from: https://www.jenkins.io/
- Run: java -jar Jenkins.war
- Access URL: http://localhost:8080/
- First launch → choose **"Install Suggested Plugins"**.

Jenkins Management Options

Accessible via **Manage Jenkins** on the dashboard:

- Configure System
- Configure Global Security
- Global Tool Configuration
- Manage Plugins
- System Information / System Log
- Load Statistics
- Jenkins CLI
- Script Console
- Manage Nodes
- Reload Configuration from Disk

Jenkins Master-Slave Architecture

Component	Role
Master Node	Manages Jenkins environment; handles job scheduling & dispatching; maintains configuration and UI
Slave Nodes (Agents)	Execute jobs assigned by master; can run on different platforms; allow **parallel execution** of builds

- Master **schedules jobs**, assigns them to slaves, monitors slave state (online/offline), and displays build results.
- Slaves execute the actual build tasks.

Jenkins Job Types

Type	Description
Freestyle Project	Simple, manual job configuration through UI; suitable for simple jobs; lacks flexibility/scalability
Pipeline	Script (declarative or scripted) defining stages for build/test/deploy; versioned as code in repository (Jenkinsfile)

Creating Jobs in Jenkins

- **New Job**: Dashboard → New Item → Enter name → Select Freestyle Project → OK → Configure → Build Steps.
- **Copy Existing Job**: New Item → Enter name → Scroll down → "Copy from" → Type existing job name.
- **Dependent Job**: Configure Job → Build Triggers → "Build after other projects are built" → Enter parent job name in "Projects to watch".

Jenkins Job Triggering Methods

#	Method	Description
1	**Manually**	Click "Build Now" in Jenkins UI
2	**Periodically (Scheduled)**	Using **cron-like syntax**
3	**SCM Polling**	Jenkins periodically checks SCM (e.g., Git) for changes; triggers build if new commits detected
4	**Webhooks**	SCM (GitHub/GitLab) sends webhooks to Jenkins on events like pushes or pull requests

#	Method	Description
5	**Upstream/Downstream**	Job runs automatically after successful completion of another job (chained workflows)

Jenkins Cron Expression Format

MINUTES(0-59) HOURS(0-23) DAY_OF_MONTH(1-31) MONTH(1-12) DAY_OF_WEEK(0-7)

- 0 and 7 both represent **Sunday**.

Pipeline as Code (Jenkinsfile)

- Define build/deployment processes **as code** (not via GUI).
- Stored and versioned in source repository.
- File must be named **Jenkinsfile** in the **repository root**.
- Presence of Jenkinsfile makes repo eligible for Jenkins to **automatically manage and execute jobs** based on branches.

Declarative Pipeline Structure

```
pipeline {
  agent any
  stages {
    stage('Hello') {
      steps {
        echo 'Hello World'
      }
    }
  }
  post {
    success { echo 'Pipeline succeeded!' }
    failure { echo 'Pipeline failed!' }
  }
}
```

Section	Purpose
pipeline	Root block; defines entire pipeline
agent	Where the pipeline runs (e.g., any available agent or specific node)
stages	Contains individual stages (build, test, deploy)
steps	Actions within each stage (shell commands, scripts)
post	Actions after all stages complete (notifications, archiving artifacts)

Delivery Pipeline

- **Plugin**: Delivery Pipeline plugin (install via Manage Jenkins → Plugins → Available).
- Used to visualize **chained dependent jobs** as a pipeline view.
- Create: Dashboard → "+" (New View) → Enter name → Select "Delivery Pipeline View" → Add components.
- Features: pipeline instances count, enable start of new pipeline build, enable rebuild, show total build time.

Setting Up Maven & Git in Jenkins

- **Maven**: Manage Jenkins → Tools → Add Maven details.
- **Git Plugin**: Manage Jenkins → Tools → Enter Git path OR install Git plugin.

PART 3: SELENIUM (Brief)

What is Selenium?

- **Free, open-source** automated testing framework for **web applications**.
- Works across different **browsers and operating systems**.
- Platform-independent; supports **distributed testing** via Selenium Grid.

Selenium Suite Components

Component	Description
Selenium IDE	Record interactions with web apps; toolkit for quick web testing
Selenium RC	Early tool; used a server as intermediary between test code and browser
WebDriver	Open-source framework for automating web browsers; interacts **directly** with browser (no server needed like RC)
Selenium Grid	Server that allows tests to use browser instances on **remote machines**; uses hub-and-spoke model

Selenium Advantages

- Open source
- Cross-browser testing
- Cross-OS support (Windows, Linux, macOS)
- Supports mobile devices

Selenium Disadvantages

- Supports only **web-based applications** (NOT desktop apps)
- No built-in IDE (slower script development than QTP)
- Cannot access controls **within** the browser
- **No default test report generation**

Selenium WebDriver

- Permits **cross-browser testing**.
- Open-source collection of APIs for automating web app testing.
- Supports languages: **Python, Java, JavaScript, C#, PHP, Ruby**.
- Interacts **directly** with the browser engine (no additional server like Selenium RC).

HIGH-YIELD MCQ TRAPS

1. **git init** creates a local repo, NOT a remote one.
2. **Index = Staging Area** (same thing, different names).
3. Staging area is at **.git/index.**
4. **Master and Develop** = permanent branches; rest are short-lived.
5. **Hotfix** branches off from **master**, NOT develop.
6. **Release branch** merges into **both master AND develop**.
7. Jenkins achieves CI using **plugins**.
8. Jenkins is **free and open-source**.
9. Jenkins default URL: **http://localhost:8080/**
10. **Master node** schedules; **Slave nodes** execute.
11. **Jenkinsfile** must be in **repository root**.
12. **Declarative pipeline** root block = pipeline {}.
13. **agent any** = run on any available agent.
14. **post** block runs after all stages (success/failure actions).
15. Jenkins cron: **0 and 7 both = Sunday**.
16. **Freestyle** = UI-configured, simple; **Pipeline** = code-based, flexible.
17. **SCM Polling** = Jenkins checks for changes; **Webhooks** = SCM notifies Jenkins.
18. Selenium WebDriver interacts **directly** with browser (no server).
19. Selenium RC used a **server as intermediary**.
20. Selenium does **NOT** support desktop applications.
21. git add . adds **all** files; git add <file> adds one file.
22. git commit -m → the **-m** flag writes commit message on command line.
23. After cloning, all files are **tracked and unmodified**.
24. **Develop branch** is also called the **"integration branch"**.
25. Token authentication replaced password for git push.