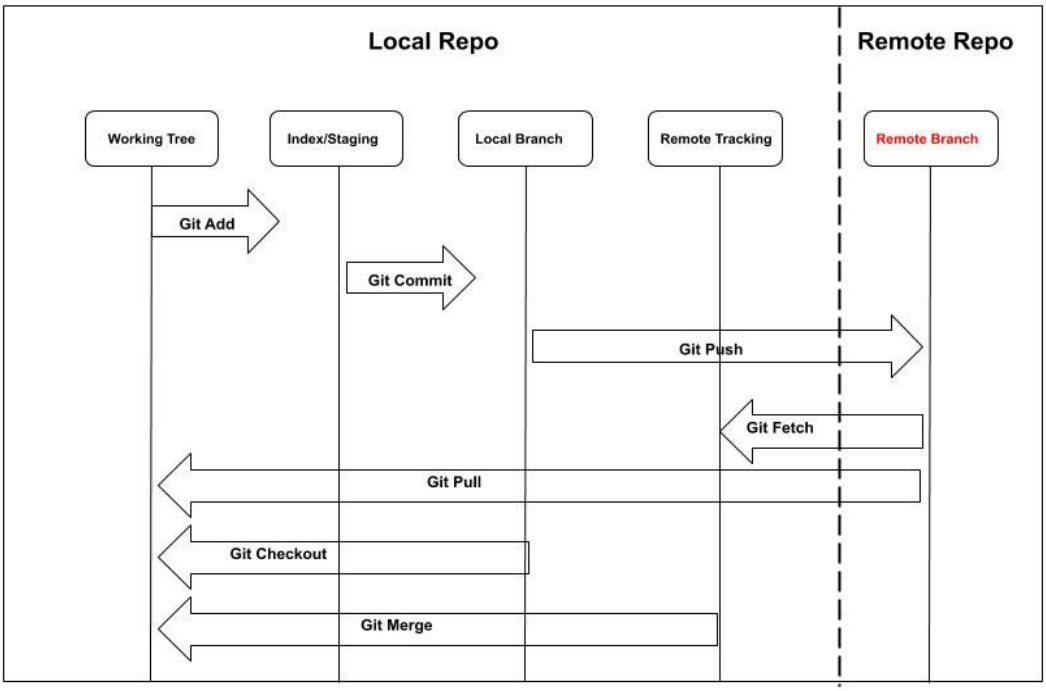
Day 6 Learn Some DevOps

DevOps Ecosystem: Build Tools- GIT & Jenkins

GIT

- **What Is VCS?**
    - A version control system (VCS) is a system that allows multiple users to manage multiple revisions of the same unit of information. It is a snapshot of your project over time.
- **Why VCS?**
    - VCS's are important because they provide a centralized repository, revision history, branching & tagging, automatic backups and they decrease the learning curve.
- **VCS Tools**
    - Tortoise SVN
    - Perforce
    - Mercurial
    - GIT
- **What Is GIT**
    - GIT is a free open source version control system used in software development with an emphasis on speed, data integrity and support for distributed, non-linear workflows.
    - Git stores information in a data structure called a repository. It allows for individuals on teams to work on various branches separate from the master branch so that changes and updates can be made without affecting the master branch and merged into the master branch once they are approved.

Workflow Diagram



- **Why GIT?**
    - **Distributed Nature**
        - Every user has a complete copy of the repository data stored locally
        - It allows for full functionality when disconnected from the network
        - Due to being distributed, you do not have to give the commit access to team members to use and access the versioning features.
    - **Branch Handling**
        - In GIT, every developer working directory is its own branch
        - GIT tracks the project revision of the branch

        - Records branch and merges events

    - **Better Merging**

        - Two GIT users can merge changes with each other and then push the changes to the remote repo
        - GIT automatically starts the next merge at the last merge
        - In GIT, you decide when to merge what from whom

    - **GIT Plugins**

        - GitFlow- This plugin permits to manage branching model in GIT
        - EGit- Implements Eclipse tool on top of Java implementation of GIT
        - GIT-Client- Allows use of GIT as a build SCM. Interaction with the GIT runtime is performed by the use of the GIT-Client plugin

- **GIT File Workflow**

    - GIT stores data as tree objects where a root node (master/1st branch) points to the next node (node A/1st revision) which points to node B (2nd revision). These objects are related or linked to one another as they change as opposed to stacked.

- **GIT Commands**

    - **Important GIT Commands When Working In Local Repository:**

        - **Initialization:** Create a new local repository `git init`

        - **Add Files:**
            - Add one file for staging `git add <filename>`
            - Add multiple files for staging `git add.`

        - **Status:** Check the status of a repo `git status`

        - **Committ:**
            - Commit changes in the HEAD of the local repo `git commit -m “commit message”`
            - Commit changes without staging the file `git commit -a -m “commit message”`

- **History:** See the commit history `git log`
- **Branch:**
  - Create a new branch `git branch <branchname>`
  - Switch from one branch to another `git checkout <branchname>`
  - Switch from any branch to master `git checkout master`
- **Merge:** Merge changes to the repo `git merge <SourceBranch> <DestinationBranch>>`
- **Stash:** Save local modifications (without commit) and revert to the working directory to match the HEAD commit `git stash`

Jenkins

- **What Is Continuous Integration (CI)?**
  - TEXT
- **Why Continuous Integration (CI)?**
  - TEXT
- **CI Tools**
  - GitLab
  - Travis CI
  - TeamCity
  - Jenkins
- **Introduction to Jenkins**
  - Jenkins is an open source CI tool written in Java used for testing and reporting on isolated changes in larger code bases in real time. The software enables developers to find and solve defects in a code base rapidly and to automate testing of their builds.
  - **Benefits**
    - Open source & easy to install on different OS's
    - Faster release cycles

    - Extensive plugin support
    - Ability to display results in graphical and tabular format
    - Shell and Windows command execution in prebuild

- **Plugin Management**

    - Jenkins support plugins which allow Jenkins to be extended to meet specific needs of individual projects.

    - **Plugin Categorization:**

        - Test
        - Reports
        - Notification
        - Deployment
        - Compile

- **Examples Of Delivery Pipelines**

    - **Scenario 1:** Create a simple delivery pipeline to compile and test a “Hello World” application

        - **Solution:** Compile > QA Test

    - **Scenario 2:** When the product is developed & delivered, create a delivery pipeline to promote the version of software through quality gates & automated steps

        - **Solution:** Test > Release > Deploy To Test > Deploy To Pre-Load > Deploy To Prod

- **Planning The Delivery Pipeline**

    - **Scenario 3:**

    - The product is developed & tested, a small change is made in the code and checked to SCM that triggers UT
    - On success, automatically trigger acceptance test & produce code coverage and static analysis report
    - If the acceptance test succeeds, the system triggers the deployment of project to an integration environment
    - Invoke integration test suite & regression test suite
    - If pass the system triggers UAT and performance test

- On success, automatically promote the project to release repo and notify the developer regarding results.

    - **Solution:** Unit Test > Acceptance Test > Code Coverage & Static Analysis > Deploy To Integration > Integration Test > Regression Test > UAT & Performance Test > Alert & Reports Notes > Deploy To Prod

- **Basic Tags in pom.xml**

- POM is an acronym for Project Object Model. It contains project & config information for the maeven to build the project dependencies, build directory, test directory, plugin, goals, etc.
- Maeven reads the pom.xml file then executes the goals

Tags	Description
project	Root element of the pom.xml file
modelVersion	Specifies the modelVersion and should be set to 4.0.0
groupid	Id for the project group & is unique amongst an organization or project
artifactid	Id of the artifact (project)
version	Specifies the version of the artifact under a given group