## Darshan Nikam Date: 01/04/2024

### **Git**

Git is a distributed version control system that is widely used for tracking changes in source code during software development. It was created by Linus Torvalds in 2005 to manage the development of the Linux kernel, but it has since become one of the most popular version control systems for all kinds of projects.

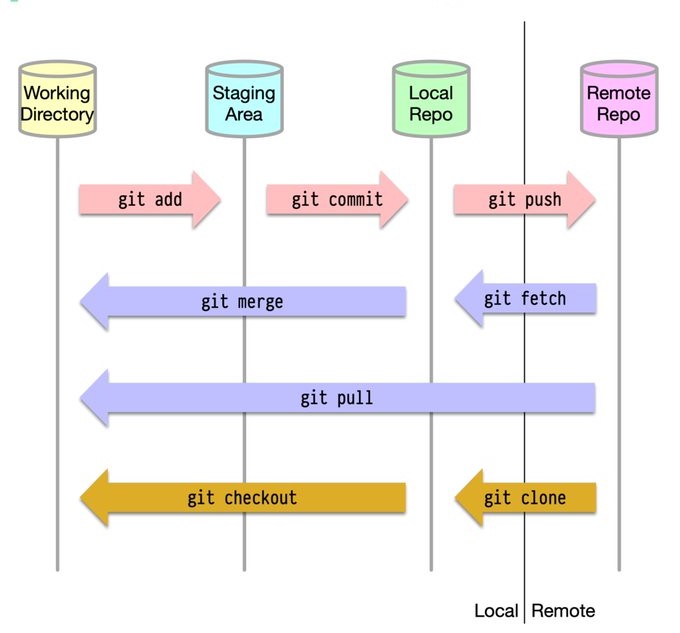
Git allows multiple developers to collaborate on a project by providing mechanisms for managing different versions of files, tracking changes, and coordinating work among team members. It provides features such as branching, merging, and tagging, which enable developers to work on separate features or fixes independently and then integrate their changes seamlessly.

Key concepts in Git include:

* Repository: A repository, or repo, is a collection of files and their revision history. Git repositories can be local (on your computer) or remote (on a server), and they contain all the files and metadata associated with a project.
* Staging Area (Index): The staging area is an intermediate step in the Git workflow where you can prepare changes before committing them to the repository. Files are added to the staging area using the “git add” command, allowing you to selectively include specific changes in your commits. This helps to organize and review changes before they become part of the commit history.
* Commit: A commit is a snapshot of changes to the files in a repository at a specific point in time. Commits include a message describing the changes made and can be thought of as checkpoints in the project's history.
* Branch: A branch is a parallel version of the repository's code. Branches allow developers to work on different features or fixes independently without affecting the main codebase. Changes made in one branch can later be merged into another branch.
* Merge: Merging is the process of combining the changes from one branch into another. This is typically done when a feature or fix is complete and ready to be integrated into the main codebase.
* Pull Request: In a collaborative development environment, a pull request is a request to merge changes from one branch into another. Pull requests allow developers to review code, discuss changes, and ensure that the integration of changes is done smoothly.
* Remote: A remote is a repository hosted on a server, such as GitHub, GitLab, or Bitbucket. Remotes allow multiple developers to collaborate on a project by providing a central location for sharing code and coordinating work.
* Clone: Cloning creates a local copy of a remote repository on your computer. This allows you to work on the code locally and synchronize changes with the remote repository.
* Fetch: Fetching retrieves changes from a remote repository and brings them into your local repository, but it doesn't automatically merge them with your current branch.
* Pull: Pulling is a combination of fetching changes from a remote repository and merging them with your current branch. It's often used to update your local repository with changes from the remote
* Push: Pushing sends your local commits to a remote repository, updating the remote with your changes.
* Conflict: A conflict occurs when Git is unable to automatically merge changes from different branches. Conflicts require manual resolution by the user.

Git is a powerful tool for managing software development projects, and its flexibility and robustness make it an essential part of modern software development workflows.

Git Workflow Diagram



* Important Git Commands:
* ***git init: Initializes a new Git repository in the current directory, creating a .git subdirectory that contains all the necessary repository files.***
* ***git config: Sets configuration values for your user, repository, or globally.***

***Syntax: git config --global user.name "Your Name" 🡪 Sets your name globally.***

***git config --global user.email*** [***your@email.com***](mailto:your@email.com) ***🡪 Sets your email globally.***

* ***git add: Adds changes in the working directory to the staging area for the next commit.***

***Syntax: git add <file(s)>***

***git add . to add all changes.***

* ***git status: Displays the status of the working directory, including untracked files, modified files, and files staged for commit.***
* ***git commit: Records changes in the staging area to the repository's history with a descriptive commit message.***

***Syntax: git commit -m "Your commit message"***

* ***git log: Displays the commit history of the current branch, showing commit messages, authors, dates, and commit hashes.***
* ***git diff: Shows the differences between the working directory, staging area, and the last commit.***

***Syntax: git diff <file(s)>***

* ***git log: Displays the commit history of the current branch, showing commit messages, authors, dates, and commit hashes.***
* ***git clone: Clones a remote repository into a new directory on your local machine.***

***Syntax: git clone <repository\_URL>***

* ***git branch: Lists, creates, or deletes branches.***

***Syntax: git branch 🡪 Lists all local branches.***

***git branch -a 🡪 Lists all local and Remote branches.***

***git branch <branch\_name>*** 🡪 ***Creates a new branch with the specified name.***

***git branch -d <branch\_name> 🡪 Deletes the specified branch.***

* ***git checkout: Switches branches or restores working tree files.***

***Syntax: git checkout <branch\_name> 🡪 Switches to the specified branch.***

***git checkout -b <new\_branch\_name> 🡪 Creates a new branch and switches to it in one step.***

* ***git merge: Combines the specified branch's history into the current branch.***

***Syntax: git merge <branch\_name>***

* ***git pull: Fetches changes from a remote repository and integrates them into the current branch.***
* ***Syntax: git pull <remote> <branch>***
* ***git push: Sends local commits to a remote repository.***

***Syntax: git push <remote> <branch>***

* ***git remote: Manages connections to remote repositories.***

***Syntax: git remote add <name> <url>: Adds a remote repository.***

***git remote -v: Lists all remote repositories.***

* ***git stash: Temporarily shelves changes so you can work on something else.***

***Syntax: git stash 🡪 Stashes changes.***

***git stash apply 🡪 Applies the most recently stashed changes.***

* ***git reset: Resets the current HEAD to a specified state (commit).***

***Syntax: git reset <commit>***

* ***git rebase: Applies commits from one branch onto another.***

***Syntax: git rebase <branch\_name>***

* ***git tag: Creates, lists, deletes, or verifies tags.***

***Syntax: git tag <tag\_name> 🡪 Creates a new tag at the current commit.***

***git tag -l 🡪 Lists all tags.***

***Practice: Setting Up Git Workflow Between AWS Instances and GitHub***

***In this practice, a Git workflow will be established between two AWS EC2 instances and a GitHub repository. Git will be installed on the instances, with one instance using a personal access token for authentication and the other using SSH keys. Code changes will be made, committed, and pushed to the GitHub repository from one instance, and then pulled from the other instance. This setup enables collaboration across environments using Git and GitHub.***

1. ***Launch 2 Instances A and B in different Regions. And SSH Into each Instance.***

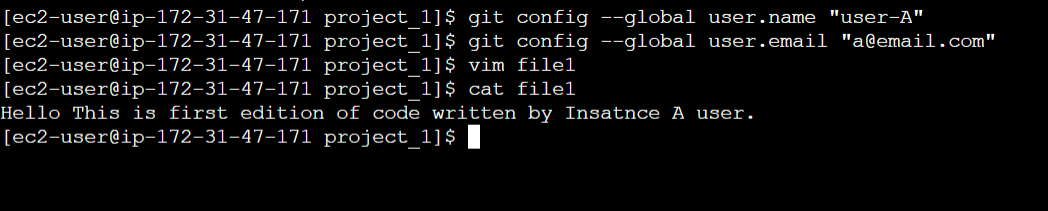
***(Allow SSH and HTTP RuleI in Inbound traffic)***

1. ***Run the following commands in both Instances separately.***

* ***sudo yum install git -y 🡪 To install Git.***
* ***mkdir project\_1 🡪 To Create a new directory for the repository.***
* ***cd project\_1 🡪 Change directory to Project\_1***
* ***git init 🡪 to initialize a new Git repository in a directory. It sets up the necessary files and data structures for version controlling your project. After initializing, you can add files, stage them for commit, and start tracking changes. This command is the first step in using Git for version control in your project.***
* ***git config --global user.name "Your Name" 🡪 To configure Git with your username.***
* ***git config --global user.email "abc@email.com" 🡪 To configure Git with your email address.***

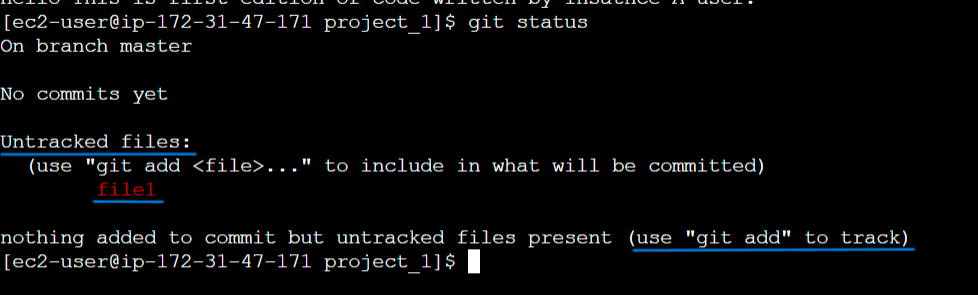
***Here, we have completed the setup: Git installed, a new project directory created, Git initialized, and user identity configured.***

1. ***Now create a file in the Local Repository of Instance A and write sample code.***

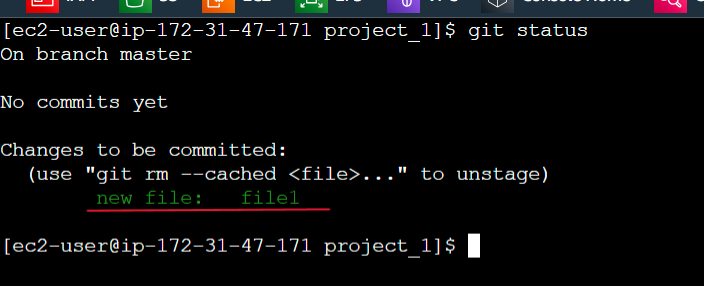


1. ***Now run the following commands:***

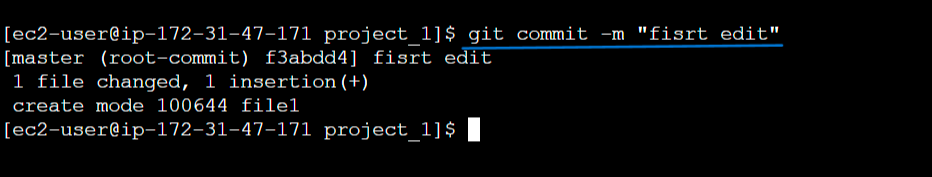
* ***git status 🡪 To display the current status of the Git repository***



* ***git add file\_name 🡪 To the working directory change to the staging area.***



* ***git commit -m “your\_commit\_message” 🡪 To save staged changes to the local repository with a commit message.***



1. ***Now we need to establish a connection between our local Git repository and a remote repository hosted on GitHub.***

***Run the following command. (Using GitHub repo URL)***

* ***git remote add origin https://github.com/username/repository.git***
* ***git branch -m main***
* ***git push origin main***

***After running the git push command git prompt for user name and Personal Access Token***

* ***A***
* ***A***