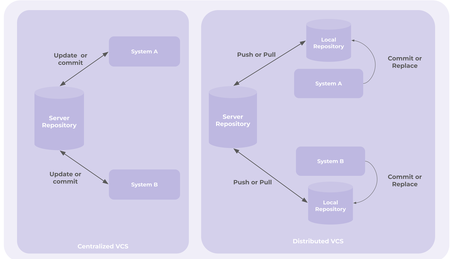
GIT Cheat Sheet

Section 01: Introduction

What is Version Control?

In layman's terms, version control, also known as source control, is the practice of tracking and managing changes to software code.
Technically, version control is the management of changes made to the code, documents, programs, and other information. The changes are referred to as versions.

There are **two** types of version control systems:
- Centralized Version Control System (CVCS)
- Distributed Version Control System (DVCS)



Our DevOps tool, Git, falls under the category of Distributed VCS.

What is Git?

Git is a free and open-source **DevOps** tool used as a version control system. It was created by Linus Torvalds, the creator of the Linux Operating System Kernel in 2005.
It is a repository used to manage a project's version or set of files as they change over time. Here, multiple developers can contribute to the code simultaneously. It also supports a non-linear way of development where you can make multiple branches for each new feature.
According to a survey by **Stack Overflow,** 93% of the developers are using Git as their Version Control Tool.

Install Git and GUIs

Git can be installed on operating systems like Windows, Mac, and Linux. By default, Git is pre-installed on Mac and Linux machines!

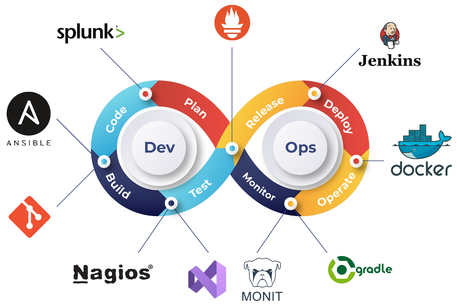
Download Git for Mac: https://git-scm.com/download/mac

Download Git for Windows: https://git-scm.com/download/win

Download Git for Linux: https://git-scm.com/download/linux

Section 02: Understanding Git

Role of Git in DevOps

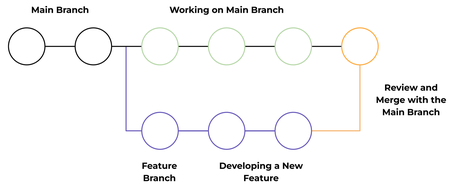


Key Terminologies in Git

- **Local repository:** It is a local directory that contains the code files for the project.
- **Remote repository:** It is an online version of the local repository that is hosted on services like GitHub, GitLab, and BitBucket.
- **Cloning:** The act of making a clone or copy of a repository in a new directory.
- **Commit:** It is a snapshot of the project's staged changes.
- **Branch:** A branch is a copy of the project used for working in an isolated environment without affecting the main project.
- **Git merge:** The process of combining two branches in Git.
- **.gitignore file:** It is a file that specifies intentionally untracked files that Git should ignore.
- **Staging area:** A cache that holds changes that will go into your next commit.
- **Git stash:** Another type of cache that holds unwanted changes you may want to return to later.
- **Commit ID or hash:** A unique identifier for each commit, used for switching to different save points.
- **HEAD (always capitalized letters):** A reference name for the latest commit, to save you from having to type commit IDs.

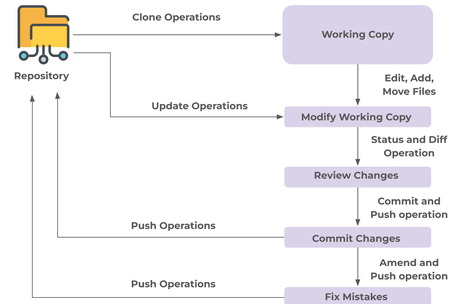
Branches in Git

Branches are special copies of the code base that allow you to work on new features in an isolated environment. The changes made in a branch won't affect the main branch, which is also the main project development environment.



Git Life Cycle

- **Step 01 - Cloning Repository:** We are cloning the code that resides in the remote repository to make a local repository on our system
- **Step 02 - Adding Changes:** We perform multiple operations or changes like editing a file, adding new code, deleting previous code, moving files, etc.
- **Step 03 - Pulling Changes from Remote Repository:** Then we pull the files again from the remote repository to check if someone has made some recent changes or not.
- **Step 04 - Reviewing the Code:** We Review the changes, move the code to the staging area, and remove any conflict present in the code.
- **Step 05 - Committing Changes:** We finally commit the changes with a proper commit message.



Section 03: Git Commands

Git configuration

Commands	Description
git config --global user.name "[firstname lastname]"	Set the global configuration for the Git user's name.
git config --global user.email "[valid-email]"	Set the global configuration for the Git email.
git config --global color.ui auto	Enable automatic coloring for Git output, i.e., Git will automatically colorize its output when it's going to a terminal.
git config --list	Displays the current configuration settings for Git on your system.

Setup and Initialization

Commands	Description
git init	Initialize an existing directory.
git clone [url]	Retrieve the repository from the hosted location through the URL.

Staging and Snapshots

Commands	Description
git status	To know the status
git diff	Displays what has been changed but not staged yet.
git add [file]	To add a file
git add [filename] [2nd filename] [3rd filename]	To add multiple files
git reset [file]	Used to unstage a file while keeping the changes in the working directory.
git commit -m "[descriptive message]"	Commit your staged content
git diff --staged	Displays what has been staged but not committed yet.
git commit --amend -m "new_message"	To amend the last commit or the last message

Branch and Merge

Commands	Description
git branch	To list branches
git branch -a	To list all the branches
git branch [branch name]	To create a new branch
git branch -d [branch name]	To delete a branch
git push origin –delete [branchName]	To delete a remote branch
git checkout -b [branch name]	To create and switch to a new branch
git checkout -b [branch name] origin/[branch name]	To clone and switch to a remote branch
git checkout [branch name]	To switch to a branch
git checkout –	To switch to the branch last checked out
git checkout — [file-name.txt]	Used to discard any changes made to a file
git merge [branch name]	To merge a branch into an active branch

Inspect and Compare

Commands	Description
git log	Show the commit history
git diff branchB...branchA	Show the difference between branchA and branchB
git log branchB..branchA	Show the commits on branchA that are not there on branchB
git log --follow [file]	Show the commits that changed the file
git show [SHA]	Show any object in a human-readable format

Share and Update

Commands	Description
git push origin [branch name]	To push a branch to a remote repository
git push -u origin [branch name]	To push the changes made to a remote repository (-u remember the branch for the next use)
git push origin –delete [branch name]	To delete a remote branch
git pull	To update a local repository to the newest commit
git pull origin [branch name]	To pull the changes from a remote repository
git remote add origin ssh://git@github.com/[username]/[repository-name].git	To add a remote repository
git remote set-url origin ssh://git@github.com/[username]/[repository-name].git	To set a repository's origin branch to SSH
git fetch [alias]	Fetch all the branches from that hosted remote repository.

Rewrite History

Commands	Description
git rebase [branch]	Used to integrate changes from one branch into another branch.
git reset --hard [commit]	Used to reset the current branch to a specific commit

Temporary Commit

Commands	Description
git stash	To stash the changes in a dirty working directory
git stash pop	Write working from the top of the stash stack.
git stash list	List the stack-order of stashed file changes.
git stash drop	Discard the changes from the top of the stash stack.
git stash clear	To remove all the stashed entries

Section 04: Summary

How do Git Commands Work?