Shivaraja.D.M
DevOps

Git & GitHub Commands Cheat Sheet

Git Configuration

Command	Description
git config --global user.name "Your Name"	Set global username
git config --global user.email "your@email.com"	Set global email
git config --list	List all configurations

Creating Repositories

Command	Description
git init	Initialize a new local repository
git clone <repo-url>	Clone an existing repository

Basic Snapshotting

Command	Description
git status	Show modified files in working directory
git add <file>	Stage a file
git add .	Stage all changes
git commit -m "message"	Commit staged changes
git commit -am "message"	Add & commit tracked files

Branching & Merging

Command	Description
git branch	List branches
git branch <name>	Create new branch
git checkout <name>	Switch to branch
git checkout -b <name>	Create & switch to branch
git merge <branch>	Merge branch into current
git branch -d <name>	Delete branch

Sharing & Updating Projects

Command	Description

git remote add origin <url>	Add remote repository
git remote -v	List remote URLs
git fetch origin	Fetch changes from remote
git pull origin <branch>	Pull changes from remote
git push origin <branch>	Push changes to remote

Inspection & Comparison

Command	Description
git log	View commit history
git log --oneline	Compact commit history
git diff	Show changes between commits
git show <commit>	Show details about commit

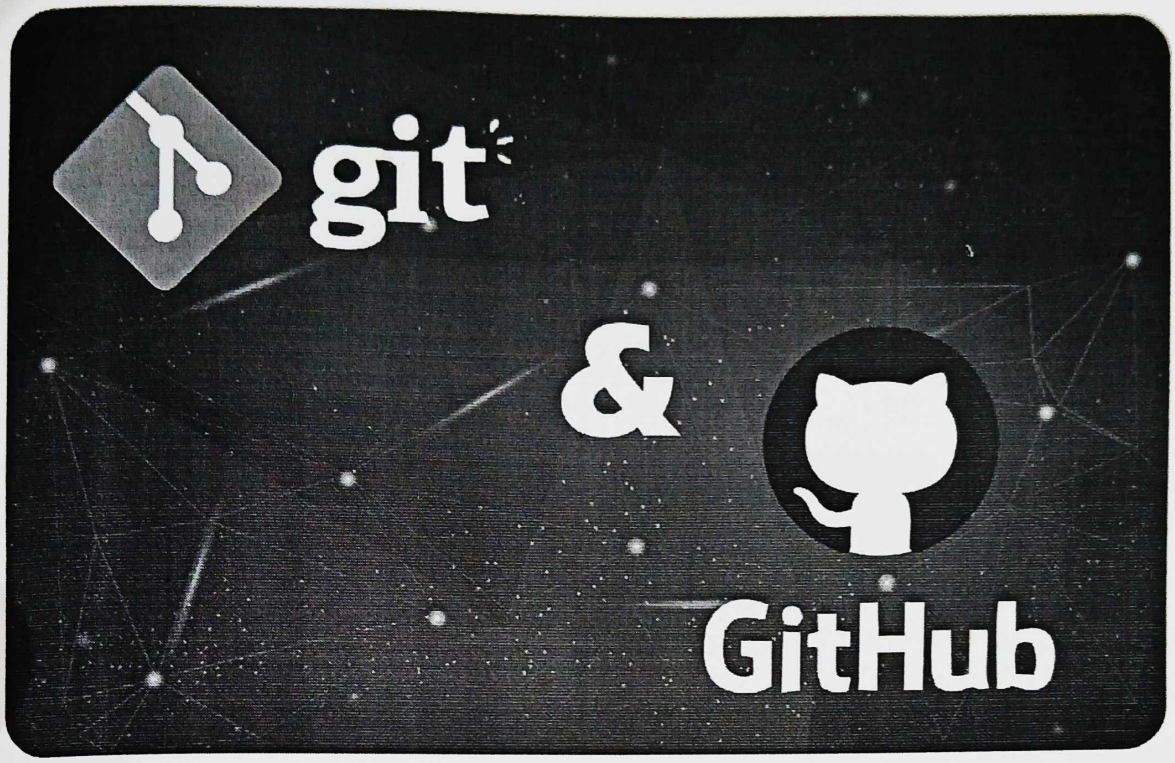
Undoing Changes

Command	Description
git restore <file>	Discard changes in file
git reset --hard <commit>	Reset to specific commit
git revert <commit>	Create new commit that undoes changes

GitHub Specific

Command	Description
gh auth login	Login to GitHub CLI
gh repo create	Create new GitHub repo
gh repo clone <repo>	Clone repo via GitHub CLI
gh issue list	List GitHub issues
gh pr create	Create a pull request

Global Information Tracker



Mastering Git: Commands and Workflows

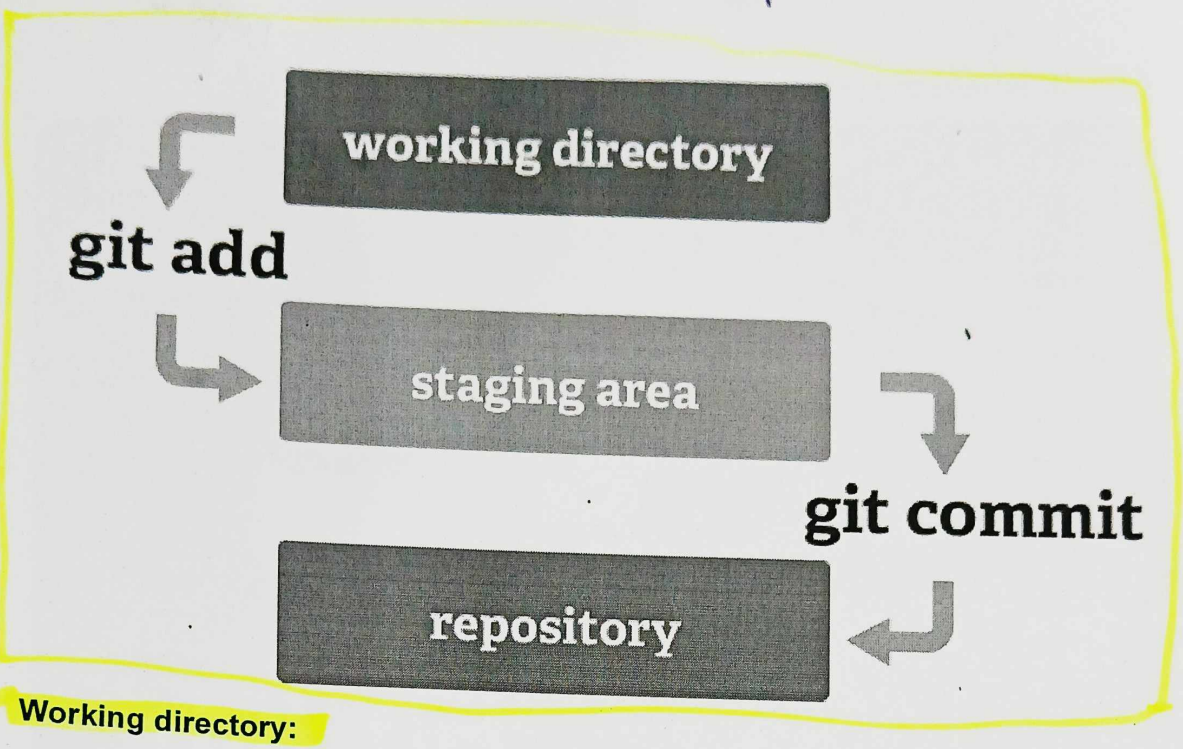
GIT:

Git stands for global information. It is nothing but a version control system (VCS) or source code management (SCM).

- Git is used to track the files.
- It will maintain multiple versions of the same file.
- It is platform-independent.
- It is free and open-source.
- They can handle large projects efficiently.
- It is 3rd generation of VCS.
- Written on C programming it.
- It came on the year 2025

GIT STAGES:

Git workflow typically involves the following stages



Working directory:

You modify files here (create, edit, delete).

Example : You're working on a web project with HTML, CSS, and JavaScript files. You add a new feature in script.js, but Git doesn't track this change until you explicitly stage it with git add script.js.

Staging area:

It is default area where you prepare changes before committing

Example: You've updated style.css and index.html. Instead of committing these files directly, you add them to the staging area using git add style.css index.html. This ensures only the selected changes are part of the next commit.

Repository:

The repository is your project's database. It contains the complete history of your changes.

Example: After staging your changes, you commit them to the repository using git commit -m "Added responsive design". This creates a checkpoint you can revert to if needed.

Types of repositories

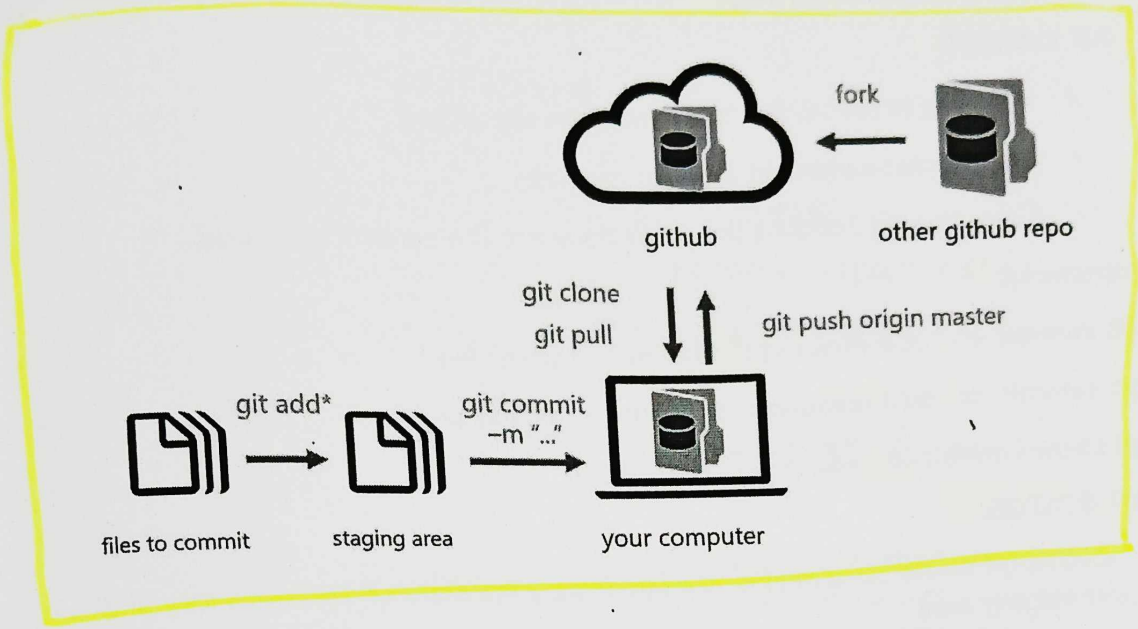
Local repository:

The local repository is everything inside your .git directory. Mainly what you will see in your Local Repository are all of your checkpoints or commits. This area stores the complete history of your project. (so don't delete it).

Remote repository:

The remote repository is a Git repository that is stored on some remote computer.

GIT WORKFLOW:



GIT Alternative:

- GitLab: A Git-based platform with CI/CD features.
- SVN: A centralized version control system.
- Bitbucket: Supports Git and Mercurial repositories.

INSTALL GIT:

command to install git : yum install git -y (yum: yellowdog updater modifier

To check the git version: **git --version**

To get empty repo : **git init .**

GIT ADD:

- Git add command is a straightforward command. It adds files to the staging area.
- We can add single or multiple files at once in the staging area.
- Every time we add or update any file in our project, it is required to forward updates to the staging area.
- The staging and committing are co-related to each other

commands:

To track the file: **git add file_name**

To track the multiple files : **git add aws azure gcp**

all regular files : **git add ***

including hidden files: **git add .**

GIT COMMIT:

- It is used to record the changes in the repository.
- It is the next command after the git add.
- Every commit contains the index data and the commit message

command:

git commit -m "add message" filename: commit file

git commit -m "add message" filename . : all files commit

git show commit-id : see tracking file data

GIT STATUS:

- The git status command is used to display the state of the repository and staging area.
- It allows us to see the tracked, untracked files and changes.
- This command will not show any commit records or information.

Command: git status

GIT LOG:

this command is the Git version control system that shows a history of commits made in a Git repository. It displays a list of past changes, including information like the commit message, author, date, and a unique identifier for each commit.

commands

git log : used to see the history of the git .

git log --oneline :used to see only commit ID's and messages

git log --pretty=oneline : Used to get only commit ID's and messages

git log -1 : used to see the latest commit

git log -3 : used to see latest 3 commits

git log --follow --all filename : used to see the no of commits for a single file

git log --graph --oneline --all : see all the history in graph

page 1

git config user.email "xyz@gmail.com" : change author email-id

git config user.name **"username"** : change author

GIT AMEND:

Git command used to make changes to the most recent Git commit. It allows you to edit the commit message, add additional changes, or both.

commands:

git commit --amend -m "message" : used to change the commit message for a latest commit

git commit --amend --author "username " : used to change the author of latest commit

git commit --amend --no-edit : used to commit the changes with previous commit

GIT RESET:

Git that allows you to move the HEAD and the current branch pointer to a specific commit, effectively "resetting" your project's state. It's commonly used to undo changes or to un-stage commits.

COMMANDS:

git reset --hard HEAD~1 : used to delete the latest commit along with the changes

git reset --hard HEAD~3 : used to delete the latest 3 commits along with the changes

git reset --soft HEAD~1 : used to delete only commits but not actions/changes

git reset --soft HEAD~3 : used to delete only latest commits but not actions/changes

GIT IGNORE:

- It will be useful when you don't want to track some specific files then we use a file called **.gitignore .**
- create some text files and create a directory with "jpg" files

command: vim .gitignore

git rm --cached filename : remove tracked and un-tracked file

GIT BRANCHES:

- A branch represents an independent line of development.
- The git branch command lets you create, list, rename, and delete branches.
- The default branch name in Git is master.

- allows you to work on different features or changes to your code independently, without affecting the main or other branches.
- It's a way to organize and manage your code changes, making it easier to collaborate and maintain your project.

COMMANDS:

git branch : used to see the list of branches

git branch branch-name : to create a branch

git checkout branch-name : to switch one branch to another

git checkout -b branch-name : used to create and switch a branch at a time

git branch -m old-branch new-branch : used to rename a branch

git branch -d branch-name : to delete a branch

git branch branch-name deleted-branch-id : Used to get deleted branch id

git branch -D branch-name : to delete a branch forcefully

The -d option will delete the branch only if it has already been pushed and merged with the remote branch. Use -D instead if you want to force the branch to be deleted, even if it hasn't been pushed or merged yet. The branch is now deleted locally.

Now all the things you have done is on your local system.

GIT MERGE:

Git merge is a command used in the Git version control system to combine changes from one branch.

To merge: git merge branch_name

GIT MERGE CONFLICTS:

GIT makes merging super easy!

CONFLICTS generally arise when two people two people have changed the same lines in a file (or) if one developer deleted a file while another developer is working on the same file!

In this situation git cannot determine what is correct!

Important

Lets understand in a simple way!

cat>file1 : hai all

add & commit

git checkout -b branch1

cat>file1 : 1234

add & commit

git checkout master

cat>>file1 : abcd

add & commit

git merge branch1 : remove it

git diff file1 vim file1

git add

git commit -m "final commits"

NOTE: Don't give file name on commit

Identify Merge Conflicts:

see the file in master branch then you will see both the data in a single file including branch names that are dividing with conflict messages

Resolve Conflicts:

open file in VIM EDITOR and delete all the conflict dividers and save it!

add git to that file and commit it with the command (git commit -m "merged and resolved the conflict issue in abc.txt")

GIT REBASE:

Git rebase is a Git command used to incorporate changes from one branch into another. It allows you to re-organize and streamline the commit history by moving or combining commits from one branch onto another.

Command: git rebase-branch

Advanced Git Features

1. Git Revert:

This command undoes changes by creating a new commit.

- **Command:** git revert commit_id
- **Scenario:** A recent commit introduced a bug. Use git revert to roll back the changes while preserving the commit history.

2. Git Cherry-Pick:

Select specific commits to apply to another branch.

- **Command:** git cherry-pick commit_id

- **Scenario:** A hotfix made on the hotfix branch needs to be applied to master. Use git cherry-pick to apply the fix without merging the entire branch.

3. Git Stash:

Using the git stash command, developers can temporarily save changes made in the working directory. It allows them to quickly switch contexts when they are not quite ready to commit changes. And it allows them to more easily switch between branches.

Generally, the stash's meaning is "store something safely in a hidden place."

COMMANDS:

git stash : to delete the changes temporarily

git stash save "message" : to save the stash along with the message

git stash apply : to get back the data again

git stash list : to get the list of stashes

git stash clear : to clear all stashes

git stash pop : to delete the first stash

git stash drop : used to delete the latest stash

git stash drop stash@{2} : used to delete a particular stash

GIT TAGS:

tags are markers or labels that you can place on specific commits (versions) in a Git repository. These tags provide a way to give meaningful names to important points in the project's history, such as software releases or significant milestones.

COMMANDS:

git tag : to see the list of tag

git tag tagname : to create normal tag

git tag -a -m "message" tagname : to create annotated tag

git tag -d tagname : to delete a tag

git show tagname : To see the details of the tag

GIT-HUB

- GitHub is a web-based platform used for version control.
- It simplifies the process of working with other people and makes it easy to collaborate on projects.

- Team members can work on files and easily merge their changes in with the master branch of the project.

COMMANDS:

git remote add origin repo-url : link local-repo to central-repo

git remote -v : used to get the linked repo in github

git push -u origin branch-name : push the code from local to central

git push -u origin branch-1 branch-2 : used to push the code to multiple branches

git push -u origin --all : used to push the code to all branches at a time

git clone repo-url : used to get the code from central to local

git pull origin branch : used to get the chances from central to local

git fetch branch-name : used to fetch the data from central to local

git fetch --all : used to fetch the changes from all branches in github

git merge origin/branch : used to merge the changes from central to local

git push -u origin --delete branch-name : used to delete the github branch from local

git remote rm origin : used to unlink the github-repo

git remote rename old -link new-link : used to change the repo

page ①

■ Docker Cheatsheet

Basics

Description	Command
Check version	docker --version
Run hello-world	docker run hello-world
List running containers	docker ps
List all containers	docker ps -a
List images	docker images

Images

Description	Command
Pull an image	docker pull <image>
Build from Dockerfile	docker build -t <name> .
Remove image	docker rmi <image_id>

Containers

Description	Command
Run a container	docker run -it <image> bash
Start a stopped container	docker start <id>
Stop a container	docker stop <id>
Remove a container	docker rm <id>
Logs of a container	docker logs <id>

Volumes

Description	Command
List volumes	docker volume ls
Create volume	docker volume create <name>
Mount volume	docker run -v <volume>:/path <image>

Networking

Description	Command
List networks	docker network ls

**********Docker commands**********

>> docker -v
to check the version of docker installed

>> docker images
to list/display the images available

>> docker pull nginx
to pull an image (this pulls an image with default tag "latest")

>> docker pull nginx:1.22
to pull an image, this pulls an image with the specific tag "1.22"

>> docker container ls
to list/display the containers "which are running/active"

>> docker container ls -a
to list/display all the containers along with stopped/exited

>> docker ps
to list/display the containers "which are running/active"

>> docker ps -a
to list/display all the containers along with stopped/exited

>> docker run -d -p 8080:80 --name my-web nginx
to run a docker image
-d stands/used for detached mode
-p 8080:80 , to specify to map local port 8080 with port 80 of container
--name my-web, to specify a name to the container (my-web is the container name in this example)

>> docker stop container_name
to stop a container

>> docker start container_name
to start a container

>> docker rm container_name
to remove/delete stopped container

>> docker rm -f container_name
to forcefully remove/delete a container

>> docker rmi image_name
to delete an image

pag(3)

Create network	docker network create <name>
Connect container to network	docker network connect <network> <container>