### GIT Whitesheet

#### 1. Initialize Repository

git init # Initializes a new Git repository

#### 2. Set Config Variables

git config --global user.name "Your Name" # Set global username

git config --global user.email "your-email@example.com" # Set global email

#### 3. Check Configurations

git config --list # List all Git configuration settings

#### 4. Add Files to Staging Area

git add <file> # Adds a specific file to the staging area

git add <file1> <file2> # Adds multiple specific files to the staging area

git add . # Adds all modified and new files to the staging area

git add -u # Stages modified and deleted files (does not add new untracked files)

git add -A # Stages all changes, including new, modified, and deleted files

#### 5. Commit Changes

git commit -m "Your commit message" # Commit staged changes with a message

git commit -am "Your commit message" # Commit changes, including tracked files (no need to add first)

git commit --amend # Amend the previous commit (useful for fixing mistakes in the last commit)

git commit --amend --no-edit # Amend the previous commit without changing the commit message

git commit -v # Commit with a verbose output, showing diffs

git commit --dry-run # Shows what would be committed without actually making the commit

#### 6. Remove Files

git rm <file\_name> # Removes file from both working directory and staging area

git mv <old\_file> <new\_file> # Renames a file (moves it within the repo)

#### 7. Removing Files from Staging Area Only

git reset <file> # Unstages a file from the staging area, but leaves it in the working directory

#### 8. Check Status

git status # Displays the status of the working directory and staging area

#### 9. View Commit History

git log # Shows the commit history

git log --oneline # Shows a condensed commit history with one line per commit

git log --graph --oneline --decorate # Shows a visual representation of the branch history

#### 10. Check Last N Commits

git log -n <number> # Shows the last N commits (replace <number> with the desired number)

git log -<number> # Alternative to show the last N commits (replace <number> with the desired number)

git log -1 # Shows the last commit

git log --oneline -n <number> # Shows the last N commits in a single-line format

git log --oneline -<number> # Shows the last N commits in a single-line format

git log --oneline --graph --decorate -n <number> # Shows the last N commits with a graphical view

#### 11. Branching and Merging

git branch # List all branches in the repository

git branch <branch\_name> # Create a new branch

git checkout <branch\_name> # Switch to another branch

git checkout -b <branch\_name> # Create and switch to a new branch

git merge <branch\_name> # Merge a branch into the current branch

#### 12. View Differences

git diff # Shows differences between working directory and staging area

git diff --staged # Shows differences between staging area and the last commit

git diff <branch1>..<branch2> # Shows differences between two branches

#### 13. Reset Commands

git reset --hard # Resets the working directory and staging area to the last commit (warning: this deletes changes!)

git reset --soft HEAD~1 # Undo the last commit but keep the changes staged

#### 14. Pulling and Pushing

git pull # Fetches changes from the remote repository and merges them into the current branch

git push # Pushes changes from the local repository to the remote repository

#### 15. Cloning a Repository

git clone <repo\_url> # Clones an existing repository from a URL

#### 16. View Remote Repositories

git remote -v # Shows the remotes associated with your repository

git remote add <name> <url> # Adds a new remote repository

git remote remove <name> # Removes a remote repository

#### 17. Stashing Changes

git stash # Stashes changes in the working directory (temporarily saves changes)

git stash pop # Applies the most recent stash and removes it from the stash list

git stash list # Lists all stashed changes

#### 18. Tagging

git tag <tag\_name> # Creates a tag pointing to the current commit

git tag -a <tag\_name> -m "Tag message" # Creates an annotated tag

git push origin <tag\_name> # Pushes a tag to the remote repository

#### 19. Reverting Back to a Particular Commit

##### Revert Changes from a Commit

git revert <commit\_hash> # Creates a new commit that undoes the changes from the specified commit (safe, keeps history)

##### Checkout a Specific Commit (Detached HEAD)

git checkout <commit\_hash> # Checks out a specific commit temporarily (detached HEAD)

git checkout -b <new\_branch\_name> <commit\_hash> # Creates a new branch from a specific commit

##### Reset to a Specific Commit (Destructive)

git reset --soft <commit\_hash> # Resets to a commit but keeps changes staged

git reset --mixed <commit\_hash> # Resets to a commit and unstages changes (keeps them in working directory)

git reset --hard <commit\_hash> # Resets to a commit and discards all changes (dangerous)

#### 20. Removing Untracked Files

##### Remove All Untracked Files

git clean -f # Removes all untracked files in the working directory

##### Remove All Untracked Files and Directories

git clean -fd # Removes untracked files and directories

##### Remove a Specific Untracked File

git clean -f <file\_name> # Removes a specific untracked file

##### Check What Will Be Removed (Dry Run)

git clean -n # Lists untracked files that would be removed, without actually removing them

##### Advanced GIT

##### Interactive Rebase

#### Amending the Last Commit

# Modify the Commit Message

git commit --amend # Opens the default editor to modify the commit message

# Add Changes to the Last Commit

git add <file1> <file2> # Stage the missing changes

git commit --amend # Combine staged changes with the last commit and modify the message if needed

# Amend Commit Without Changing the Commit Message

git commit --amend --no-edit # Adds staged changes to the last commit without modifying the message

# Force Push the Amended Commit (if already pushed)

git push --force # Force push to the remote repository after amending the commit (use with caution)

git push --force-with-lease # Force push with additional safety checks to avoid overwriting others' work

git rebase -i HEAD~3 # Start interactive rebase for the last commit

git rebase --continue # Continue rebase after resolving conflicts or editing commits

git rebase --abort # Abort rebase and return to original state

git rebase --skip # Skip the current commit during rebase

##### Cherry- pick

git cherry-pick <commit\_hash>