<https://www.hostinger.com/tutorials/basic-git-commands>

initiates a new Git repository within a directory.

# git init

# touch file1.yml

stage file changes

# git add .

Unstage changes

# git rm --cache file1.yml

commit message for the changes, making them part of your project’s history:

# git commit -m “messege”

displays valuable insights into your files’ modifications and staging status.

# git status

list of commit history

# git log

to discard changes in working directory

# git restore file1.yml

compare changes between your working directory and the most recent commit

# git diff

To compare changes between two commits

# Git diff commit1 commit2

removes files from your working directory and stages the removal for the next commit.removes files from your working directory and stages the removal for the next commit.

# git rm file-1

To restore it here is different mechanism

# git restore --staged file-1 >> restore from staged

# git restore file-1

rename and move files within your working directory

# git mv file-1 file-2

configures various aspects of Git, including user information and preferences.

# git config --global user.email “your\_mail”

list all existing branches

# git branch

To create a Git branch named “non-prod”

# git branch non-prod

To switch to branch

# git checkout non-prod

To rename branch

# git branch -m branch-name new-branch-name

To discard change in working directory

# git checkout – file1.yml

Use git reset to move the branch pointer back to the commit before the merge. Or back to specific commit

# git reset –hard <commit-id>

apply specific commits from one branch to another without merging an entire branch.

# git cherry-pick <commit-id>

apply changes from one Git branch to another by moving or combining commits. It helps maintain a cleaner commit history:

# git rebase <from-branch>

To store temporary changes that are not yet ready to be committed,

# git stash

To list them

# git

creates a copy of a remote repository on your local machine.

# git clone

sends your local Git branch commits to a remote repository, updating it with your latest changes.

For example, you want to push changes from the local repository called “main” to the remote repository named “origin”:

# git push origin main

push an existing repository from the command line

# git remote add origin <url>.git

# git push origin main

To retrieve new commits from a remote repository without automatically merging them into your current branch

# git fetch

identifying bugs or issues in your project’s history. To start the bisecting process

# git bisect start

Git will automatically navigate you through commits to find the problematic ones using the following:

# git bisect run <test-script>

command determines the author and the most recent change to each file line:

# git blame file1.txt

command logs Git branch changes. It allows you to track your repository’s timeline, even when commits are deleted or lost:

# git reflog

The git clean command in Git is used to remove untracked files and directories from your working directory. Untracked files are files that are not part of the Git repository and have not been added to the version control system.

# git clean

# git clean -d

Assuming you are currently on the branch where you want to merge changes (let's call it branchA), and you want to merge changes from another branch (let's call it branchB):

1. **Checkout the Target Branch (**branchA**):** Make sure you are on the branch where you want to merge changes.

git checkout branchA

1. **Merge the Other Branch (**branchB**) into the Target Branch:** Use the git merge command to merge changes from branchB into branchA.

git merge branchB