Level 1

1. git help
2. git help command name
3. git help config
4. git config –global user.name “kshitijag0209”
5. git config –global user.email Kshitij.agrawal0209@gmail.com
6. git config – global color.ui true

 # Start A Repo:-

1. git init // LOCAL SERVER
2. git status
3. git add . / git add * / git add –all / git add file or folder name
4. git commit -m “message”

 # Git History:-

1. git log

 # git diff → To check difference

git diff filename /// NOTE:- File should be in staging area

SUMMARY:-

1. **git init** → Create Repo
2. **git status** → inspect content
3. **git add** → add files to directory
4. **git diff** → show difference b/w the working directory & staging area
5. **git commit** → permanently stores files changes from staging area in repository
6. **git log** → show a list of all pervious commits

HOW TO BACK TRACK:-

1. **HEAD COMMIT:-** In git, the commit you are currently on is known as HEAD COMMIT. In many cases, the most recently made commit is HEAD commit.

To see the HEAD commit, enter

➔ git show HEAD

To revert back;

➔ git checkout HEAD filename

will restore the file in your working directory to look exactly as it did when you loast made a commit.

EXAMPLE:- git checkout HEAD filename.txt

Add Multiple File → git add filename_1 filename_2

To Unstage file from staging area use → git reset HEAD filename

2. RESET:-

    1. **First** → git log
    2. **Use** → git reset SHAS
       Enter the command to reset to a previous commit, using the first 7 characters of one of the past commits SHAS in your git log.

git reset commit_SHA

Example → git reset 56f7cd8

SUMMARY:-

1. **Git checkout HEAD filename** → Discard changes In working directory.
2. **Git reset HEAD filename** → Unstage file changes in staging area.
3. **Git reset SHA** → can be used to a previous commit in your history.

GIT BRANCHING:-

1. **Git branch** → check what branch you are currently on.
2. **New branch create** → git branch fencing.
3. **To switch to other branch use** → git checkout <branchname>
4. **If you want to include changes in fencing branch to master branch** → git merge branch_name.

   Example:- If, I want to merge 'skills' branch to master use → git merge skills
5. **Git merge conflicts** →
   git checkout master
   git merge fencing
   /// gives error in that different content file.
   Git asks which version to keep the master version or fencing version.
   - Change the version then add * then commit.
6. Delete branch -d branch_name

SUMMARY:-

1. **Git branch** → list all branches
2. **Git branch branch _name** → create a new branch
3. **Git checkout branch _name** → used to switch to other branch
4. **Git merge branch_name** → used to join file changes from one branch to other.
5. **Git branch -d branch_name** → delete the branch specified

CLONE

1. Git clone remot_location clone_name
2. List of git projects's remote → git remote -v
3. Git fetch → git fetch
   This command help to keep update
4. After git fetch we merge local master to remote master → git merge origin/master

SUMMARY:-

1. **Git clone** → create a local copy of remote
2. **Git remote -v** → list a git project's remote
3. **Git fetch** → fetch work from the reote into the local copy.
4. **Git merge origin/master** → merge origin/master into local master.
5. **Git push origin branch_name** → pushes a local branch to origin remote
   .

REBASE

- git rebase origin/master
  It merges requested branch (origin/master) & apply commits that you have made locally to the top of the history without creating merging commit.