**TASKS:**

**TASK 1:**

* git push without using remote aliases:

**Command:**

git push <remote-url> <branch-name>

**Note:**

Remote-url: repository url where you are going to push.

Branch-name: mention branch name used for that remote-url i.e., main or master.

* using username and password in git push command

**Command:**

**git push** [**https://<username>:<password>@repo-url**](https://%3cusername%3e:%3cpassword%3e@repo-url) **<branch-name>**

* using access-token pushed file to remote location

**Command:**

**git push** [**https://<username>:<access-token>@repo-url**](https://%3cusername%3e:%3caccess-token%3e@repo-url) **<branch-name>**

**Note:**

* To give access token first create access token in github
* Next,select the permissions for that access token which u have created.
* Then, take that token and use it while pushing to remote location.

**TASK 2:**

* Changing commit msg from recent commit msg to new commit msg in central repository(github).

**Command:**

* --amend is used to change the recent commit message and Add new commit message by using below command.

git commit –amend -m “pass new commit message”

* Forcebly push to remote repository using below command.

git push –force <remot-name> <branch\_name>

**TASK 3:**

* Changed commit msg from old commit msg to new commit msg in local repository.

**Command:**

git commit –amend -m “new commit message”

**TASK 4:**

* If there are multiple commit msgs,select the one commit msg which u are going to change.

**Command:**

* Command for changing multiple commit msgs.

git rebase -i –root

* Command for changing specific commit msgs.

git rebase -i HEAD~3

* Changing commit msg from old commit msg to new commit msg inplace of pick use reword
* reword means use commit and edit the existing commit message.

**TASK 5:**

* push some code in git after three push rollback the first push.

**Command:**

git revert HEAD

**TASK 6:**

**HEAD:**

* It is used to checks the last commit in the current branch.
* It can be understood as current branch.
* It is reference to current commit.

**.gitignore:**

* Create .gitignore file in git repo
* It will ignore the files which are untracked.
* And place the file which you want to ignore in .gitignore file.

**git checkout:**

* It is used to switch current active branch.

**Command:**

* git checkout <branch-name>

using git checkout we can create new branch

* git checkout -b <branch-name>

if we use commit id while creating branch the output it displays was the head position will be moved to the current used commitid.

* git checkout -b <branch-name> <commit-id>

**git reset:**

* git reset command allows you to RESET your current head to a specified state.

**Commands:**

* git reset HEAD filename(staging area to workspace)
* git reset --soft <previouscommitid>(local repo to staging area)
* git reset –mixed <previoscommitid>(local repo to workspace)
* git reset –hard commitid (head will be moved to current commitid)

**git revert:**

* The git revert command is used to apply revert operation from existing commit msg to new commit message.

**Commands:**

git revert commitid

**git merge:**

* git merge command is used to merge data between to branches.
* To merge data from main to master first checkout to master and merge.
* Below command is used to merge data from one branch to another branch.

**Commands:**

git merge <branchname>

pushing to central repository

git push [https://username:<access-token>@repo-url](https://username:%3caccess-token%3e@repo-url) <branchname> --force

**How to fix merge conflicts**

**Commands:**

* First checkout to branch1

git checkout <branchname>

* Create file ,add file and commit file

touch filename

git add filename

git commit -m “pass message”

* First checkout to branch2

git checkout <branchname>

* Create file ,add file and commit file

touch filename

git add filename

git commit -m “pass message”

* Checkout to one branch1 and pass file to another branch

git merge branch1

* Add data in file1 in branch1, add and commit.
* Next add data in file1 in branch2 ,add and commit.
* Checkout to branch2 type the below command

git merge branch1

* We will get below errors
* CONFLICT (add/add): Merge conflict in file1
* Auto-merging file1
* Automatic merge failed; fix conflicts and then commit the result.
* To solve this goto the file1 remove the >>>>>,====,<<<<< symbols and then add file ,commit file the conflicts will be resolved

git add filename

git commit -m “message”

**rebase git merge:**

* create file1, file4 in branch1
* add and commit file
* git checkout branch2
* create file5 in branch2
* add and commit file
* git merge branch1