Course: DevOps Name: Billipati Sai Teja

Module: GIT Mail-ID: ([BILLIPATISAITEJA@GMAIL.COM](mailto:BILLIPATISAITEJA@GMAIL.COM))

Topic: Git Commands Batch no: 115

Trainer Name: Mr. Madhukar sir Assignment no: 01

Date of submission: 08th – Nov – 2023

--------------------------- Git Commands ---------------------------

* First, we have to download Git Bash and setup the environment.

We should create the GitHub user account after creating account store the user id and email id.

---------------------------> user id: Saiteja074 (my id)

---------------------------> user email: billipatisaiteja@gmail.Com (mail id)

Then open the git bash in system.

user creation in the gitbash

>>> git config --global user.name "Saiteja074" ---------------------------> It is used to connect the GitHub account git bash by the username.

>>> git config --global user.email "billipatisaiteja@gmail.Com" ---------------------------> It is used to connect the GitHub account gitbash by the email id.

>>> git config --list ---------------------------> The command shows list of the accounts that connect to gitbash.

Then it takes permission to the GitHub account.

Then the credentials has been stored in the git bash.

Note : This process can be done only one time to setup the account.

--------------------------- Stages of the Git ---------------------------

* There are 3 main stages in the git, they are

1. Working area

2. Staging area

3. Local repository

4. Central repository (or) Remote repository

--------------------------- Identify the git repository ---------------------------

* There are 2 identifications to know there are git repositories, there are:

1. Main

2. Master

--------------------------- Git Basic Commands ---------------------------

* git init 🡪 Initializes a new Git repository. Until you run this command inside a repository or directory, it’s just a regular folder. Only after you input this does it accept further Git commands.
* git clone 🡪 Makes a Git repository copy from a remote source. Also adds the original location as a remote so you can fetch from it again and push to it if you have permissions.
* git add Adds 🡪 files changes in your working directory to your index.
* git rm 🡪 Removes files from your index and you’re working directory so they will not be tracked.
* git commit 🡪 Takes all of the changes written in the index, creates a new commit object pointing to it and sets the branch to point to that new commit.
* git status 🡪 Shows you the status of changes as untracked, modified, or staged.
* git branch 🡪 Lists all of the branches in your repository. This also tells you what branch you're currently in.
* git checkout 🡪 Lets you navigate between the branches created by git branch.
* git merge 🡪 Merges lines of development together. This command is typically used to combine changes made on two distinct branches.

git merge branch name

* git pull 🡪 Fetches the files from the remote repository and merges it with your local one. This command is equal to the git fetch and the git merge sequence.

git pull origin master

* git push 🡪 Pushes all the modified local objects to the remote repository and advances its branches.
* git stash 🡪 Temporarily saves changes that you don't want to commit immediately. You can apply the changes later.
* git diff 🡪 Used to make a diff file with the changes between two paths or files.
* git grep 🡪 Lets you search through your trees of content for words and phrases.

git grep "Hello"

* git reset 🡪 Resets your index and working directory to the state of your last commit.

git reset --hard HEAD

* git log 🡪 Shows a listing of commits on a branch including the corresponding details.
* git branch 🡪 List all of the branches in your repo. Add a <branch> argument to

create a new branch with the name <branch>

* git checkout -b <branch>
* Create and check out a new branch named <branch>. Drop the -b flag to checkout an existing branch.
* Git pull 🡪 We are pull from the remote repository to local repository.
* git revert 🡪 Creates a new commit that undoes all of the changes made in a previous commit, then apply it to the current branch.

git revert commitID

------------------------- END -------------------------