Table of contents

Installation of Git

- Windows
  - Downloading Git on Windows https://git-scm.com/download/win
- Debian - Ubuntu
  - Installing on Debian-based distribution, such as Ubuntu `sudo apt install git-all`
- Install git on respective OS from https://git-scm.com
- Download Github for Desktop from here

> Execute all below commands in `Git Bash` shell

Git Local Operations

Create Local Repository, adding files and commit changes

Initialize a Git Repository

- Run `git --version` to check git version.
- Start a new repository. Check Your git Settings
- Create a new Project Folder and initialize git inside it.

```
git --version
mkdir git-practical
cd git-practical
git init
ls -altR  # List all the files recursively
git config --list
```

- The `git init` command creates an empty Git repository - basically a `.git` directory with subdirectories for `objects, refs/heads, refs/tags, and template files`.
- An initial `HEAD` file that references the `HEAD of the master branch` is also created.

- There is `global` and `local` `git config` that can be used. If you have multiple git servers, you need to config your git as per git directory bases. Or you can use global config.
- For local git repo config, use:

```
git config user.name "Yourname"
git config user.email "name@example.com"
```

- This will result in [user] section added to .git/config file:

```
cat .git/config

[user]
name = Yourname
email = name@example.com
```

- You can use global config also if you only have one git server.

```
git config --global user.email "name@example.com"
git config --global user.name "Yourname"
```

- Then the [user] section will be present at ~/.gitconfig, with the same content as in the .git/config file.
- To get the specific config value that is currently set

```
git config user.name
git config user.email
```

Working with GIT Local Repo

- Go to the directory where you want to initialize as git repo
- List all the files in the current directory

```
ls -al
echo 'This is Repo Created for Devops Demo' > README
git status
```

- To track the file, add this file into staging area

```
git add README
git status
```

- This command compares your staged changes to your last commit:

```
git diff --staged
```

- To commit a change, there should be a commit message provided.

```
git commit -m "changes made in the particular file"
```

- Make some more changes in the file:

```
echo 'This is content written after 1st Commit' >> README
git status
```

- If you edit a file that is already staged, it will appear in `"Changes to be committed:"` and `"Changes not staged for commit:"`
- Viewing Your Staged and Unstaged Changes
- make some changes in the file that is already staged, below command will show the differences

```
git diff
```

- If you want to see what you've staged that will go into your next commit, you can use
- Viewing the Commit History and view commit details in one line

```
git log
git log --oneline
```

- To view the only last two entries

```
git log -p -2
```

- More common options for `git log`

```
git log --stat
git log --pretty=oneline
```

- To change the commit message of an existing commit

```
echo "this is testing file" >> newfile.txt
git add newfile.txt
git commit -m '1st commit'
git log -p -2
git commit --amend -m "an updated commit message"
git log -p -2
```

- Unstaging a Staged File

```
echo "adding some content to unstage changes" >> newfile.txt
git add *
git reset newfile.txt
```

Github Account Remote Repository.

- Sign Up into www.github.com and verify email id.
- Connection to your GitHub Account using SSH
- Generating an SSH Key

```
ssh-keygen -t rsa -C "<GITHUB_EMAIL_ID>@gmail.com"
```

- Above command will create a Public and Private Key Pair.
- Add the Public Key file content into your Github Account Settings under: `Settings` > `SSH and GPG keys.` > `Add SSH key` > Paste the Public Key Content > Save
- Use your actual email address in the example above.
- Verify SSH authentication

```
ssh -T git@github.com
```

- Above command uses ssh to connect to GitHub over the SSH protocol.
- Create a empty repository in github from Github UI in browser
- Push from existing repository

```
git remote add origin git@github.com:<GITHUB_USERNAME>/git-practical.git
```

- The git remote command lists the names of all the remote repositories and the -v parameter (verbose) will display the full URL of the remote repository for each remote name listed

```
git remote -v
```

- First time push command use below -u parameter

```
git push -u origin master
```

- Make some changes to a file locally and push it to GH

```
git add filename
git status
git commit -m "Message for the commit"
```

- Before pushing any changes, make sure your local files are in sync with GitHub repo

```
git pull origin master
```

- Push changes in GitHub

```
git push origin master
```

- To get or display the content of the file as per particular commit

```
git show e4b71efa7f76c0fc0875e0562d5fb6d7dadbff9c:newfile.txt
```

> AWS CodeCommit is a version control service hosted by Amazon Web Services that you can use to privately store and manage assets

Reference information

- Creating a remote repository reference

```
git remote add remote-name remote-repository-location
```

- Using git remote add command allows us to associate a remote repository. Normally, you want to paste in the full URL for the remote repository given to you by your Git host (GitHub). By convention, the first or primary remote repository is named origin.

- Send Changes to Remote

```
git push -u remote-name branch-name

git push remote-name branch-name
```

- The git push sends all your local changes (commits) on branch `branch-name` to the remote named `remote-name`.
- The `-u` parameter is needed the first time you push a branch to the remote.
- Receive Changes from Remote

```
git pull remote-name branch-name
```

- The git pull receives all your remote changes (commits) from the remote named remote-name and on branch branch-name.

> Getting help from git

```
git help <verb>
git help config
git add -h
```