- Downloading Git on Windows https://git-scm.com/download/win
- Installing on Debian-based distribution, such as Ubuntu `sudo apt install git-all`
- install git on respective OS from https://git-scm.com
- Start a new repository.

```
git init
```

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

- Check Your git Settings

```
git config --list
```

#You can also check what Git thinks a specific key's value is by typing git config :

- To get the specific config value that is currently set

```
git config user.name
```

Working with GIT Local

- Go to the directory where you want to initialize as git repo

```
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
```

- 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
- 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
```

> The above command will open a text editor to enter the commit message

OR

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

- Viewing the Commit History

```
git log
```

- To view the only last two entries

```
git log -p 2
```

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

- Limiting Log Output

```
git log --since=2.weeks
```

- To change the commit message of an existing commit

```
git commit -m '1st commit'
git add forgotten_file
git commit --amend -m "an updated commit message"
git log
```

- Unstaging a Staged File

```
git add *
git reset file1 file2
```

- Generating an SSH Key

```
ssh-keygen -t rsa -C "github-sign-in-emailid@gmail.com"
```

- 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 browser
- Push from existing repository

```
git remote add origin git@github.com:githubusername/git-practical.git
```

- check remote repo

```
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 GH repo

```
git pull origin master
```

- Push changes in GH

```
git push origin master
```

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

```
git checkout e4b71efa7f76c0fc0875e0562d5fb6d7dadbff9c test.txt
OR
git show e4b71efa7f76c0fc0875e0562d5fb6d7dadbff9c:test.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.

- List Git's Remotes

```
git remote -v
```

- 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

- 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
```