Git and GitHub Datasheet

What is Git:

- Git is aversion control system used for tracking changes in computer files.
- It is generally used for source code management in software development.
- Git is used to tracking changes in the source code.
- The distributed version control tool is used for source code management.

Why Git:

- Over 70% of developers use Git
- Developers can work together from anywhere in the world.
- Developers can see the full history of the project.
- Developers can revert to earlier versions of a project.

Features of Git:

- When a file is changed, added or deleted, it is considered modified
- You select the modified files you want to stage
- The staged files are committed, which prompts Git to store a permanent snapshot of the files.
- Git allows you to see the full history of every commit.
- You can revert back to any previous commit.

What is GitHub:

- Git is not the same as GitHub.
- GitHub makes tools that use Git.
- GitHub is the largest host of source code in the world, and has been owned by Microsoft since 2018.

Configuring git for the First time:

```
$ Git Config --global user.name "<Enter your username here>"
```

```
$ Git Config --global user. Email "<Enter your email here>"
```

General Git Features:

Initializing Git:

```
Git init
```

Git now Know that it should watch the folder you initiated it on. Git creates a hidden folder to keep track of changes.

Staging files /Adding files to Git repo:

Staged files are files that are ready to be committed to the repository you are working on.

When you first add files to an empty repository, they are all untracked. To get Git to track them, you need to stage them, or add them to the staging environment.

```
$ Git add <filename with extension>
```

Staging all files in a folder:

```
$ Git add --all
```

```
$ Git add -A
```

```
$ Git add .
```

Making a Commit:

Adding commits keep track of our progress and changes as we work. Git Considers each commit change point or “save point”. It is a point in the project you Can go back to if you find a bug, or want to make a changes.

When we commit, we should always include a message.

```
$ Git Commit –m <”Enter your message here”>
```

Commit without Stage:

Sometimes, when you make small changes, using the staging environment seems like a waste of time. It is possible to commit changes directly, skipping the staging environment.

```
$ Git Commit -a –m <”Enter your message here”>
```

Status of files and log:

```
$ Git status
```

File status in a more compact way:

```
$ Git status --short
```

Log of a File:

```
$ Git log
```

```
$ Git log --oneline
```

Git Help:

If you are having trouble remembering commands or options for commands, you can use Git help.

See all the available options for the specific command –

```
$ Git <command>  -help
```

See all possible commands -

```
$ Git help --all
```

If you find yourself stuck in the list view , SHIFT + G to jump the end of the list ,then q to exit the view.

Git Branching:

In Git, a branch is a new/ separate version of the main repository. Branches allow you to work on different parts of a project without impacting the main branch. when the work is complete, a branch can be merged with the main project.

We can even switch between branches and work on different projects without them interfering with each other.

Making a new Git Branch:

```
$ Git branch <name of branch>
```

Checking all available Branches:

```
$ Git branch
```

Switching to other Branches :

```
$ Git checkout <branch name>
```

Making a new branch and directly switching to it:

```
$ Git checkout -b <branch name>
```

Deleting a Branch :

```
$ Git checkout -d <branch name>
```

Merging two Branches:

Its preferred to changes / switch to master branch before any branch needs to be merged with it .

```
$ Git merge <branch name>
```

Push local repo to GitHub:

To connect to the local repository to remote repository.

```
$ Git remote add origin <paste URL here>
```

Pushing local repo to GitHub after doing the above process at least once:

First Commit all the changes. Then push all the changes to our remote origin i.e. remote repo on GitHub.

```
$ Git push origin
```

Pull local repo from GitHub:

Git pull is used to pull all changes from a remote repository into the branch we are working on. It is a combination of fetch and merge. Use it to update your local Git.

```
$ Git pull origin
```