Git Cheat Sheet

What is Git?

Git is a distributed source control system

How is it different from Clearcase?

In Clearcase, to perform any operation, you have to be connected to Siemens network.

Not in Git. In Git, almost all the operations can be done offline. These include checkout (of a file), commit, branching and merging.

The operations that require network access are push, pull and fetch.

Basic Git commands

Although almost all the operations can be done in Visual Studio Team Explorer, it is helpful to know the underlying Git commands and get a hang of running them in Git Bash, so that you can work with Git in absence of any UI tool.

Initilalize or clone a Git Repo

The most basic command to get started with Git is

```
git init
```

This initializes a Git repository in the folder where it is run.

To get a repo from an external source, e.g. Github or Azure Devops, we need to do what is known as ***clone*** using the command

```
git clone <repository url>
```

Add and Commit files

After running `git init` in a non-empty folder, git doesn't add the existing files to source control automatically. As of now they are in ***untracked*** state.

To make Git track these files, run the command

```
git add <filename>
```

To add all the files in the folder, run

```
git add *
```

or

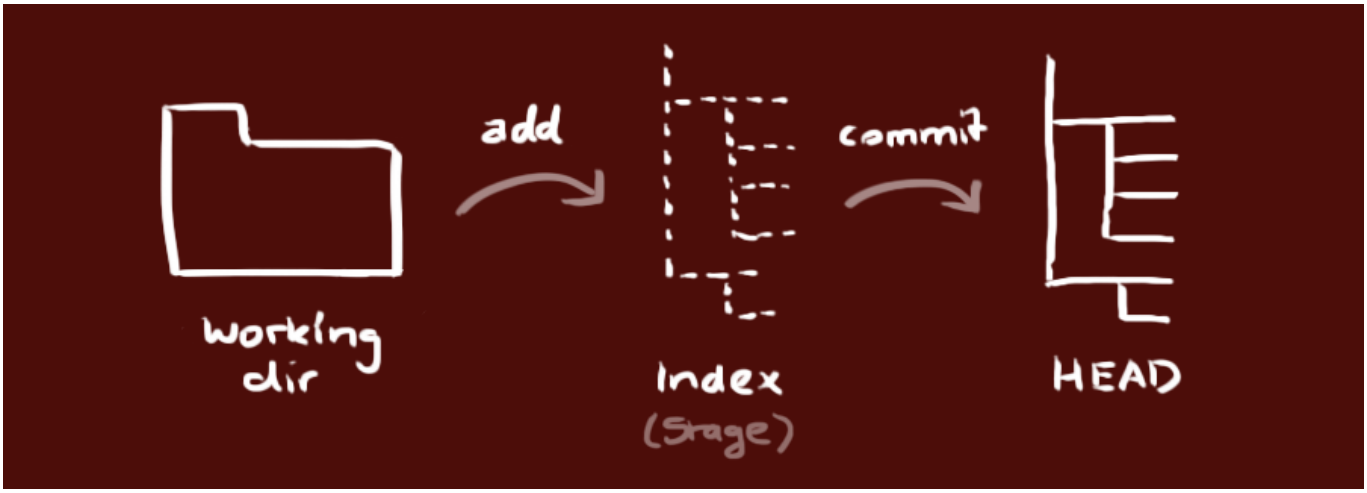
```
git add .
```

This adds all the untracked files to what is know as ***staging area*** or ***index***.

The files are ready to be committed now. To commit them, run

```
git commit -m <commit message>
```

Your local repository consists of three "trees" maintained by git. the first one is your ***Working Directory*** which holds the actual files. The second one is the ***Index*** which acts as a staging area and finally the ***HEAD*** which points to the last commit you've made.



Branching

Whenever you start working on a defect or feature, you need to create a new branch. This is similar to the Clearcase branching workflow.

Typically, a new branch is created from the master branch.

To create a new branch,

1. Make sure you are on master branch by running

   ```
   git checkout master
   ```

2. Run

   ```
   git checkout -b <branch name>
   ```

Now, you are on the newly created branch.

Merging

After you are done with your changes on your topic branch, it's time to merge the changes onto the master branch. To do this,

1. Be sure to be on the destination branch (in this case, master branch). Run

```
git checkout master
```

2. Run

```
git merge <topic branch name>
```

Remote operations

After making changes in your branches locally, you need to send you changes to a remote repository so that other members of your tem can access those changes.

Start by adding a remote repo to your local repo by running

```
git remote add <remote name> <URL of remote>
```

By convention, the remote name is "origin", although it can be anything you wish.

Note: This applies only if you have created a repo from scratch. If you have cloned the repo, the remote is automatically set for you.

Before pushing changes to remote, you need to first get any existing commits that your teammate may have pushed on the branch.

This operation is known as ***pull***.

If you just want to get changes from remote, but not merge them into your local repo, run

```
git fetch <remote name>
```

e.g.

```
git fetch origin
```

To pull changes into a branch, run

```
git pull <remote name> <branch name>
```

e.g.

```
git pull origin master
```

This command merges the changes to your local branch. You need to resolve **merge conflicts** if there are any.

An easy way to remember the difference between **fetch** and **pull** is

Pull = Fetch + Merge

After pulling the changes from remote, you can then push the changes that you have made by running

```
git push origin <branch name>
```

e.g.

```
git push origin master
```

Workflow to send merge to master in DIGSI

A typical workflow in DIGSI would involve:

1. Creating a topic branch (defect or feature) for your requirement.
2. Implementing your changes in your branch
3. Merging these changes to master.

A few points to note:

1. ALWAYS make a topic branch for your changes. Never make changes directly in the master branch.
2. The master branch will be protected by branch policies in Azure Devops to prevent you from pushing any changes directly to remote master. To merge changes to master, you need to raise a **Pull Request** from your branch to master.

Let's assume you have created a branch called *feature1* to implement your changes.

For clarity, here are the branches you will work with:

Local	Remote
master	master
feature1	feature1

The workflow is:

1. Pull changes from *remote* master to **local** master.
2. Merge changes from **local** master to **local** feature1. This is equivalent to running Merge Manager with latest label in Clearcase. You need to resolve any merge conflicts here.
3. Push changes from **local** feature1 to *remote* feature1.
4. Raise Pull Request to merge *remote* feature1 to *remote* master.

Miscellaneous Git commands

These are some other useful Git commands.

1. `git status` tells you the status of your working directory, i.e. how many files are modified, untracked or staged.
2. `git log` gives you the history of all the commits made till now.

   Bonus: `git log --graph` shows a nice ASCII graph, helping you visualize the merging of different branches.
3. `git diff` shows the differences in modified files vs. staged files.

`git diff --staged` shows the differences between staged files and committed files.

4. Adding the `--help` flag after any command will open the documentation of that command in a web browser.

   e.g. `git commit --help`

A good resource to learn more about Git is the official Git Book.