GitHub Club
Git Cheat Sheet

GIT-GitHub Club
GANDHINAGAR INSTITUTE OF TECHNOLOGY

1. CONFIGURATION / INITIALISATION

```
$ git config --global user.name <name>
```

Sets the name you want attached to your commit transactions

```
$ git config --global user.email <valid email>
```

Sets the email you want attached to your commit transactions

2. CREATING A REPOSITORY

```
$ git init <directory>
```

Create empty Git repository in specified directory
Run with no arguments to initialize the current directory as a git repository

```
$ git add
```

Stage all *changed* files, ready for commit

```
$ git add --all
```

Stage *all* files, ready for commit

```
$ git clone <url>
```

Clone (download) a repository that already exists on GitHub, including all of the files, branches and commits

3. CHECK STATUS

```
$ git status
```

List which files are staged, unstaged, and untracked

```
$ git log
```

Display the entire commit history using the default format

```
$ git log -p <file/directory>
```

Display change history for file/directory including diffs

```
$ git diff
```

Show changes to the files not yet staged

```
$ git show
```

Show one or more objects (blobs, trees, tags and commits)

4. REMOTE REPOSITORIES

```
$ git remote
```

Manage the set of repositories ("remotes") whose branches you track

5. BRANCHING

```
$ git branch
```

List all local branches

```
$ git branch <branch-name>
```

Create a new branch

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

Switch to the specified branch and update the working directory

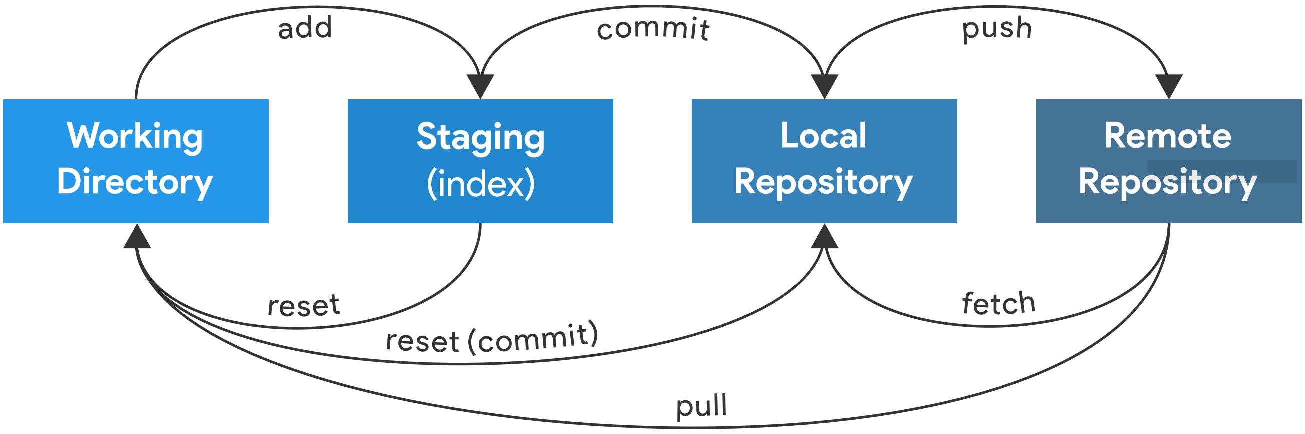
```
$ git checkout -b <branch-name>
```

Create a new branch, switch to it and update the working directory

6. COMMIT

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

Commit all staged files to versioned history



Working Directory
Staging (index)
Local Repository
Remote Repository
add
commit
push
reset
reset (commit)
fetch
pull

Workflow in Repositories

Git Cheat Sheet

7. MERGE & SYNCHRONIZE CHANGES

```
$ git merge <brach-name>
```

Join the specified branch into your current branch

```
$ git push
```

Upload all local branch commits to remote repository on GitHub
Set its copy as an upstream

```
$ git pull <remote>
```

Fetch changes from the remote repository on GitHub and merge current branch with its upstream

```
$ git fetch <remote>
```

Fetch changes from the remote repository on GitHub, but not update tracking branches

8. REVERT & RESET

```
$ git reset <commit>
```

Undoes all commits *after* the specified commit, preserving changes locally

```
$ git reset --hard <commit>
```

Discard all history and change back to the specified commit

9. TAG

```
$ git tag
```

List all tags

```
$ git tag <commit>
```

Tag the specified commit

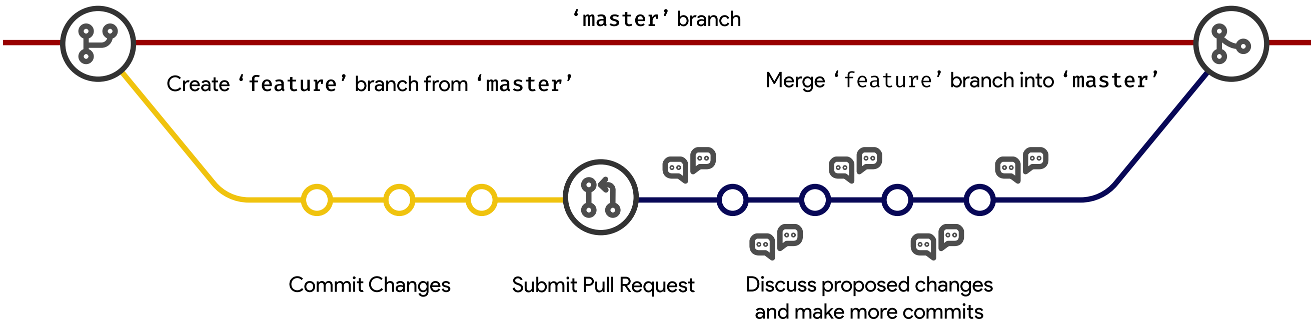
IGNORING FILES

Sometimes it is a good idea to exclude directories from being tracked with Git. This is typically done using a special file named `.gitignore` Git will ignore tracking of the directory (along with its children directories) where the `.gitignore` file is placed.

HELP

```
$ git --help
```

Display help information about Git, show the synopsis of the git command and a list of the most commonly used Git commands


'master' branch
Create 'feature' branch from 'master'
Merge 'feature' branch into 'master'
Commit Changes
Submit Pull Request
Discuss proposed changes and make more commits

GitHub Workflow

Glossary

`git`: an open source, distributed version-control system

`GitHub`: a platform for hosting and collaborating on Git repositories

`commit`: an individual change to a file or a set of files which is identified by SHA

`branch`: a parallel version of a repository which is contained within the repository but does not affect the primary or master branch

`clone`: a version of a repository, including all commits and branches

`remote`: a common repository on GitHub that all team members can use to exchange their changes

`fork`: a personal copy of a repository on GitHub owned by a different user

`pull request`: proposed changes to a repository submitted by a user and accepted or rejected by a repository's collaborators

`HEAD`: a defined commit of a branch, usually the most recent commit at the tip of the branch; the HEAD pointer can be moved to different branches, tags, or commits when using `git checkout`

"WHERE SUCCESS IS A TRADITION"