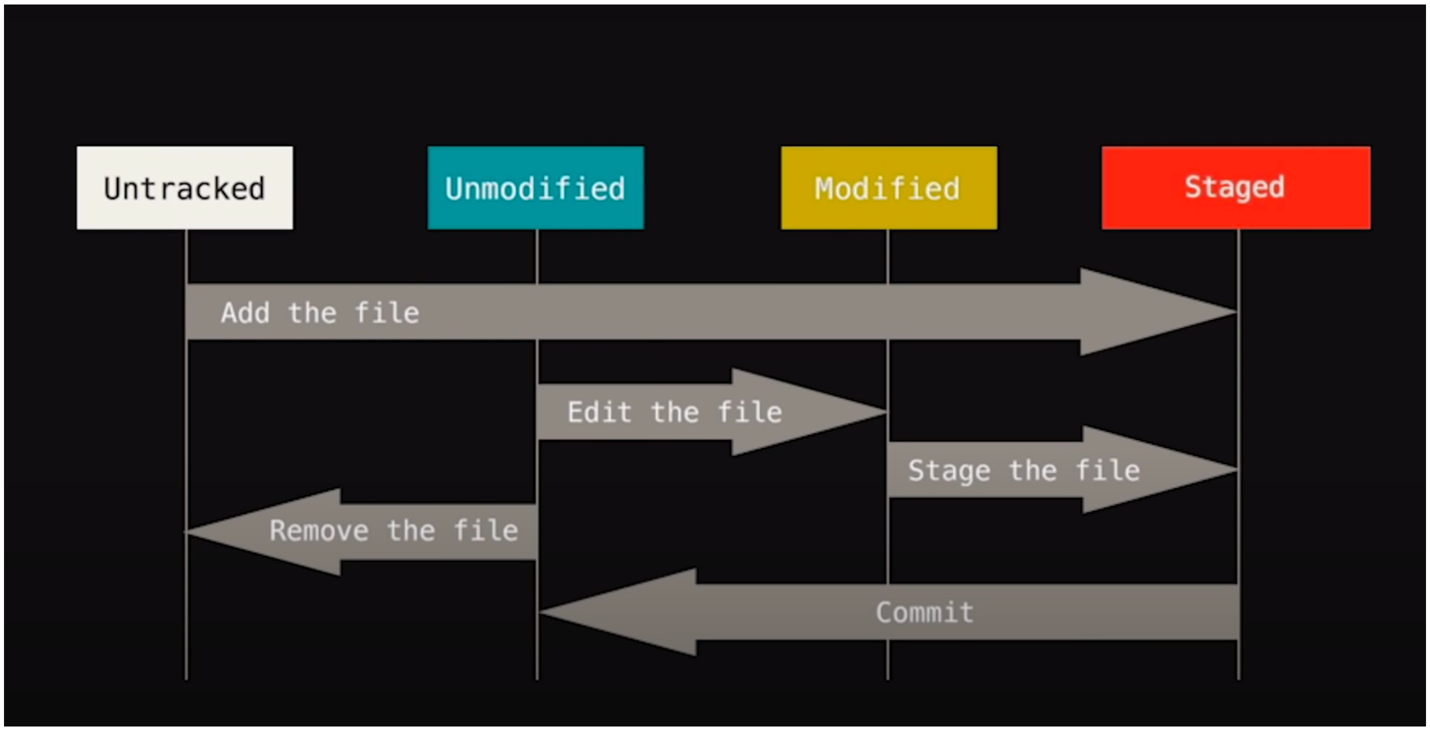
Git & Github

Git is an open-source, version control tool created in 2005 by developers working on the Linux operating system.

GitHub is a company founded in 2008 that makes tools which integrate with git.

```
git --version #check version
git status #Gives info on branch and commit suggestions
```

Git Basics #basics



Staging area is used to decides which files are needed for a snapshot (to be in github repository)

`.gitignore`

To ignore a file, create `.gitignore` and add the name of the file in it In `.gitignore` file -

```
/mylogs.log
*.log
ignore/
```

ignore file in the same root folder as .gitignore ignore all log files ignore `ignore` folder , `/` represents a folder

Git config

```
git config --global user.email #set email
git config --global user.name #set name
git config --global --list # list configurations
```

Git init

```
git init # initializing empty repository (Go to the folder which is to be made a repository)
```

Git add

```
git add <file_name> #Adding a file to staging area
git add -A #All file to staging area
```

After modify you have to again write `git add <file_name>` to add that file to staging area

Git commit

```
git commit #commit files to git(Get a snapshot of files)
git commit -m "<message>" #add a commit message
git commit -a -m "<message>" #commit all files(only works when file are first added using git add)
```

Git restore(sometimes called git checkout)

```
git restore <file_name> #Used to get the previous version of a file in the working directory (replace working file with
staged file)
git restore -W <folder_name> #restore the specified working directory with staged are
git restore -f
```

Git log

```
git log #see all the commits made
git log -p -<no_of_commits> # see specific number of commit
```

Git diff

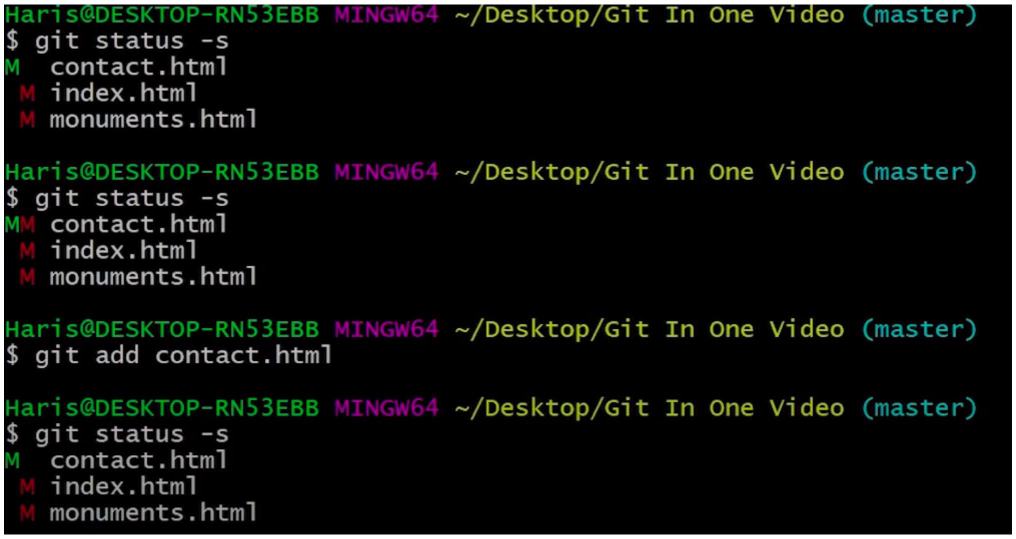
```
git diff #compares the working area with staging area
git diff --staged #compares staging area with commit
```

Git rm

```
git rm <file_name> #removes the file from working as staging area
git rm --cached <file_name> #removes file only from the staging area, does not remove file from working area
```

Git status

```
git status -s #detailed status
```



```
Haris@DESKTOP-RN53EBB MINGW64 ~/Desktop/Git In One Video (master)
$ git status -s
M  contact.html
 M index.html
 M monuments.html

Haris@DESKTOP-RN53EBB MINGW64 ~/Desktop/Git In One Video (master)
$ git status -s
MM contact.html
 M index.html
 M monuments.html

Haris@DESKTOP-RN53EBB MINGW64 ~/Desktop/Git In One Video (master)
$ git add contact.html

Haris@DESKTOP-RN53EBB MINGW64 ~/Desktop/Git In One Video (master)
$ git status -s
M  contact.html
 M index.html
 M monuments.html
```

green **M** indicates file modified in staging are red **M** indicates file modified in working area

Git branch

```
git branch # show current branch
```

Git checkout

```
git checkout <branch> #change branch
git checkout -b <branch> #create and change branch
```

Git remote

```
git remote set-url <variable_name> <ssh_url> # set remote url

git remote -v #see remote url
```

Setting up ssh-agent for no passphrase

```
eval "$(ssh-agent -s)" # start ssh-agent
ssh-add ~/.ssh/id_ed25519 # add ssh key -> add passphrase
git push <push_repo> <local_repo>

# close ssh-agent after use
eval "$(ssh-agent -k)"
```

Git pull

- Similar to git clone
- Can be run multiple times

Forks and Pull Requests

Basic Steps to upload an folder to github

```
# Existing local repo
git init
git remote -v # check for fetch and push locations
git remote add <name_url_branch> <url> # ssh url for the repo
git pull <name_url> --allow-unrelated-histories <name_url_branch> <local_branch>
git branch #info on local branch
git push -u <name_url_branch> <local_branch> #push files to <name_url_branch> repository within github
```

Info

Pull is required before push