# Top Git Commands

Stan 2021

# Why is Git different?

"Git doesn’t think of or store its data this way. Instead, Git thinks of its data more like a series of snapshots of a miniature filesystem. With Git, every time you commit, or save the state of your project, Git basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot. To be efficient, if files have not changed, Git doesn’t store the file again, just a link to the previous identical file it has already stored. Git thinks about its data more like a stream of snapshots." git-scm

# Configuration

- This command sets the author name and email address respectively to be used with your commits.
  e.g. your GitHub account email address.

```
$ git config --global user.name "John Doe"
$ git config --global user.email "johndoe@example.com"
$ git config --global core.editor vim # or nano or emacs but don't cry after that
```

# SSH (use it!!!)

- Generating a new pair of SSH keys
- Adding public key to you GitHub account
- For GitHub fans: GitHub CLI

```
# start the ssh-agent in the background
$ eval `ssh-agent -s`
> Agent pid 59566

$ ssh-add ~/.ssh/$privateKey
Enter passphrase for ~/.ssh/$privateKey
# Great! You won't have to type your passphrase again until you close this terminal
```

# Initialize a new repository

```
$ mkdir $localRepoName
$ cd $localRepoName
$ git init

# if you created an empty repo **with the same name** on GitHub:

$ vim README.md
$ git add README.md
$ git commit -m "Initial commit" # commit local changes

$ git remote add origin git@github.com:$userName/$localRepoName.git
$ git branch -M main # 'main' was master before
$ git push -u origin main # push files to GitHub
```

# Clone a repository

```
$ git clone https://github.com/$userName/$repoName.git
```

- Remember you can use SSH
- Remember to use the SSH agent

# Add files

```
$ git add <files> # <files> will be tracked by git
$ git add . # when using a .gitignore
```

- You can use a `.gitignore` that should be commited before adding specific files

```
# .gitignore
# ignore objects files
*.o
# ignore the 'test' folder and everythin inside
test/
```

# Commit changes

```
$ git commit -m "message"

# or if you don't mind
$ git commit -a
```

# Push to a server

```
$ git push <optional $branch> # that can be `git push origin $branch`
```

# Tag a commit

```
$ git commit -m "this commit will be tagged"
$ git tag "version-0.0"
$ git push --tags

# On GitHub, a tag initialises a release, very useful.
# Using DevOps, a tag can launch a pipeline (trigger on a specific tag).
```

# Pull

```
$ git pull # sync with the server
```

# Compare files

```
$ git diff # This command shows the file differences which are not yet staged.

# show differences between branches
$ git diff $firstBranch $secondBranch
```

# Status of the repository

```
$ git status # shows changed files and untracked files

# useful for git reset --hard
# show the history of the commits for the current branch, and their ID
$ git log
commit e45f8399cf1092db2e04e51e914f399c569b5eb0 (HEAD -> main, origin/main, origin/HEAD)
Author: Firstname NAME <email@email.com>
Date:   Tue Mar 16 08:36:26 2021 +0100

    Create README.md

commit 1b093c745d7a55a8cbfaa5d0f58a874944936863 ----------> $commitID
Author: Firstname NAME <email@email.com>
Date:   Tue Mar 16 08:34:37 2021 +0100

    initial commit
```

# Rollback changes

```
$ git reset $file

#  This command discards all history
# and goes back to the specified commit

$ git reset --hard $commitID # you'll have to merge after that
```

# Remove files

```
$ git rm <files>
```

# Use branches

```
$ git branch $branchName
$ git checkout $branchName # switches to this branch
# or in one command
$ git checkout -b $branchName # creates and switches

$ git branch -d $branchName # deletes a branch
```

# Merge

```
$ git merge $branchName

# or
$ git pull

----<error message>----

$ git merge $branchName # e.g. main
# you will have to resolve conflicts using git diff
```

# #RTFM!

- git-scm documentation