

# git and Github

# git

- git is a *version control* system
- Commonly used by large code development projects to track and commit changes/additions to the codebase.
- Written by Linus Torvalds to manage the Linux kernel project
- Other version control systems
  - cvs -- **C**oncurrent **V**ersioning **S**ystem
  - svn -- **S**ubversion
  - hg -- Mercurial

# Creating a repository

In the directory that contains the files you want to track, type:

```
> git init
```

That's it!

# Adding files to the repository

git doesn't assume that all files in the directory should be committed to the repository, you must add them. To get a listing of both tracked and untracked files.

```
> git status
```

To add a file, (e.g. `file.txt`) to the repository index

```
> git add file.txt
```

To add all files in a directory.

```
> git add .
```

# Committing

Once you have added files to the index and your ready to save a “snapshot” of your repository, perform a commit

```
> git commit -m 'a commit message'
```

Leave off the `-m` and your `$EDITOR` will open for you to include a longer commit message. To commit all modified files already tracked in the repository without explicitly using `git add` use the `-a` option

```
> git commit -a -m 'a commit message'
```

# Good commit messages

## Long(er) commit message example

Short (50 chars or less) summary of changes.

More detailed explanatory text, if necessary. Wrap it at 72 characters. The first line is treated as the subject of an email and the rest of the text as the body. The blank line separating the summary from the body is critical.

Write your commit message in the present tense: "Fix bug" and not "Fixed bug."

Further paragraphs come after blank lines.

- \* Bullet points are okay, too
- \* Typically a hyphen or asterisk is used for the bullet, preceded by a single space, with blank lines in between

# Excluding files

Put files you don't want to appear in the status listing in a file named `.gitignore`

## Example `.gitignore` file

```
# Ignore emacs backup files:  
*~
```

```
# Ignore everything in the docs directory:  
docs
```

# Removing files from git

To remove a file (e.g., `file.txt`) from your repository use

```
> git rm file.txt
```

This removes `file.txt` completely from your disk, same as regular UNIX `rm`

If you only want to remove a file from the git repository, but leave it in your working directory use:

```
> git rm --cached file.txt
```



# Going back in time (reverting)

To go back to the previous commit

```
> git revert HEAD
```

To go back three commits

```
> git revert HEAD~3
```

To go back to a specific commit

```
> git revert daa8d81f
```

The `daa8d81f` is a hash string that identifies the commit, it can be seen with `git log`

# Going back while trashing changes

To permanently trash your changes and get back the most recent commit

```
> git reset --hard HEAD
```

To get back a previous commit and trash all commits that happened after

```
> git reset --hard daa8d81f
```

If you only want to reset one file (e.g., file.txt)

```
> git checkout file.txt
```

# Cloning remote repositories

To clone remote (not local to your machine) repositories and create a local working copy for your own modification use `git clone`, e.g.,

```
> git clone git://github.com/johntfoster/dotvim.git ~/.vim
```

Will clone my dotvim repository to a folder in `~` named `.vim` This repository becomes local, i.e. you can make changes and local commits.

To stay "in sync"

```
> git pull origin master
```

`origin` is the default remote repo name and `master` is the default branch name.

# Pushing to remote repositories

To push changes to a remote repository, any local changes must be first committed locally as usual:

```
git commit -am "A commit message."
```

The push to the remote repository with

```
git push origin master
```

# Advanced features

- Branching
- Merging

# Github

- <https://github.com>
- Cloud based remote repository server
  - Unlimited public repositories
  - Private repositories available (educational plans for free)
  - Teams/Organizations
  - Integrated "Pull request" and code review system
- Similar services
  - <https://bitbucket.org>
  - <https://gitlab.com>