Branch: master

Find file | Copy path

vivadata-student-003 / curriculum / 00-Setup / 00-Lectures / 00-Terminal-and-Git.ipynb

nmviva Initialize repository

13cb83a on 31 Mar

1 contributor

Raw | Blame | History

316 lines (315 sloc) 10.1 KB

Terminal and Git

Photo by Markus Spiske (https://unsplash.com/photos/iar-afB0QQw)

I. Terminal commands

Find more command lines on : https://www.learnenough.com/command-line-tutorial/frontmatter (https://www.learnenough.com/command-line-tutorial/frontmatter)

I.1. Running a terminal

Command	Description	Example
`man <command>`	Display manual page for command	`$ man echo`
`^C`	Get out of trouble	`$ tail ^C`
`clear` or `^L`	Clear screen	`$ clear`
`exit` or `^D`	Exit terminal	`$ exit`
`which`	Locate a program on the path	`$ which curl`
`echo $message`	Display variables	`$ echo "Hello"`
`ping <url>`	Ping a server URL	`$ ping vivadata.org`
`ps`	Show processes	`$ ps aux`
`top`	Show processes (sorted)	`$ top`
`kill -<level> <pid>`	Kill a process	`$ kill -15 24601`
`pkill -<level> -f <name>`	Kill matching processes	`$ pkill -15 -f spring`
`history`	History of commands in a particular terminal shell	`history`

I.2. Directories

I.2.1. Manipulating

Command	Description	Example

`mkdir <name>`	Make directory with name	`$ mkdir foo`
`pwd`	Print working directory	`$ pwd`
`cd <dir>`	Change to	`$ cd foo/`
`cd ~/<dir>`	cd relative to home	`$ cd ~/foo/`
`cd`	Change to home directory	`$ cd`
`cd -`	Change to previous directory	`$ cd && pwd && cd -`
`.`	The current directory	`$ cp ~/foo.txt .`
`..`	One directory up	`$ cd ..`
`cp -r <old> <new>`	Copy recursively	`$ cp -r ~/foo .`
`rmdir <dir>`	Remove (empty) dir	`$ rmdir foo/`
`rm -rf <dir>`	Remove dir & contents	`$ rm -rf foo/`

Be careful : the command **`rm -rf /`** is **very dangerous** and should **never** be used (even as a joke).

I.2.2. Inspecting

Command	Description	Example
`ls`	List directory or file	`$ ls hello.txt`
`ls -l`	List long form	`$ ls -l hello.txt`
`ls -a`	List all (including hidden)	`$ ls -a`
`find`	Find files & directories	`$ find . -name foo*.*`
`grep -ri <string> <dir>`	Grep recursively (case-insensitive)	`$ grep -ri foo bar/`
`atom <dir>`	Open directory in Atom	`atom .`
`open <dir>`	Open directory in file browser	`open .`

I.3. Files

I.3.1. Manipulating

Command	Description	Example
`touch <file>`	Create an empty file	`$ touch foo`
`curl`	Download data with URLs	`$ curl -O vivadata.org`
`>`	Redirect output to filename	`$ echo foo > foo.txt`
`>>`	Append output to filename	`$ echo bar >> foo.txt`
`mv <name> <dir>`	Move file to directory	`$ mv foo bar`
`mv <old> <new>`	Rename file from old to new	`$ mv foo bar`
`cp <old> <new>`	Copy old to new	`$ cp foo bar`
`rm <file>`	Remove (delete) file	`$ rm foo`
`rm -f <file>`	Force-remove file	`$ rm -f bar`

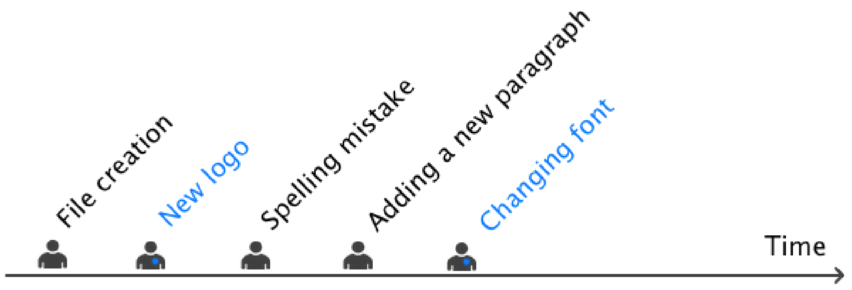
I.3.2. Inspecting

Command	Description	Example
`cat <file>`	Print contents of file to screen	`$ cat hello.txt`
`diff <f1> <f2>`	Diff files 1 & 2	`$ diff foo.txt bar.txt`
`head <file>`	Display first part of file	`$ head foo`
`tail <file>`	Display last part of file	`$ tail bar`
`wc <file>`	Count lines, words, bytes	`$ wc foo`
`less <file>`	View file contents interactively	`$ less foo`
`grep <string> <file>`	Find string in file	`$ grep foo bar.txt`
`grep -i <string> <file>`	Find case-insensitively	`$ grep -i foo bar.txt`

II. Git and Github

II.1. The problem

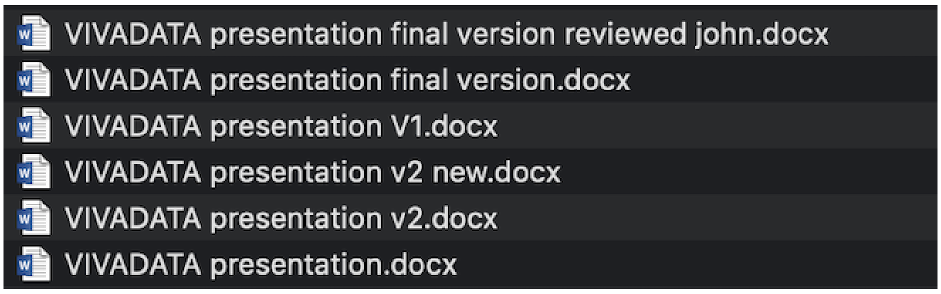
A file has a life cycle



We want to keep track of different versions

- **When** the file was modified ?
- **What** has changed ?
- **Why** it has been modified ?
- **Who** did the change ?

Tracking manually is painful and inefficient



VIVADATA presentation final version reviewed john.docx
VIVADATA presentation final version.docx
VIVADATA presentation V1.docx
VIVADATA presentation v2 new.docx
VIVADATA presentation v2.docx
VIVADATA presentation.docx

How can we automate this ?

II.2. Local work with Git

Most useful commands

Command	Description	Example
`git init`	Start tracking changes in directory (initialised git repository)	
`git clone <github_url>`	Retrieve existing repository (on GitHub for instance)	`git clone git@github.com:username/existing_project.git`
`git status`	Check if the repository has some modified files	
`git add <file>`	Track the changes (we can add multiple files)	`git add .`
`git commit`	Save the changes	`git commit -m "Describe the changes"`
`git reset <file>`	Remove file from tracking (to discard changes before committing)	
`git diff <file>`	Inspect what has changed in a specifid file	

`git log`	Display commit history

How to use it ?

Start a new project

```
mkdir -p ~/code/YOUR_GITHUB_USERNAME/sample_directory && cd $_  # Create a new folder
git init # Initialise a git repository
ls -al #  A .git hidden folder has automatically been created
```

Create a file and commit changes

```
touch README.md # Create a README.md file
code .  # Open README file in VSCode editor and write something
git status  # The file is not tracked
git add README.md # Track the file
git status  # The file is now tracked
git commit -m "Create README" # Save changes
git status # All changes are saved
```

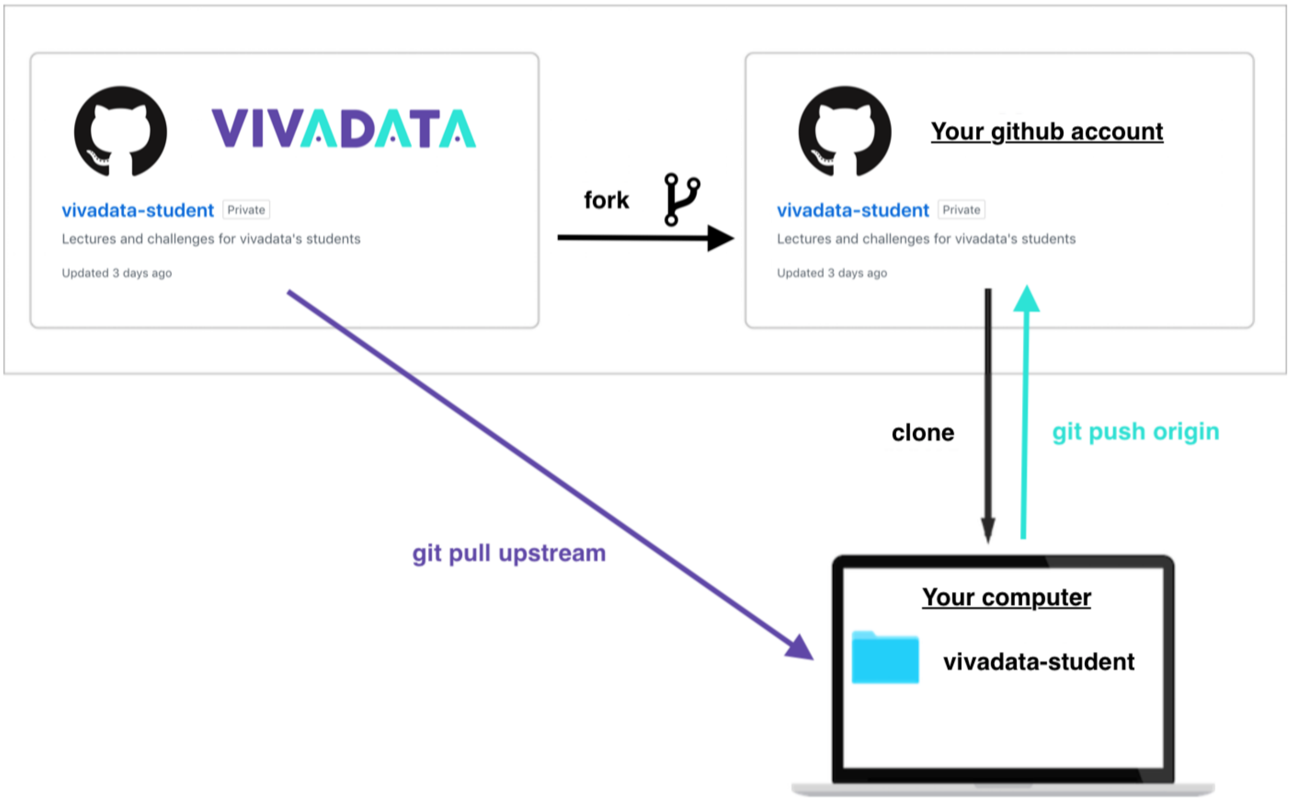
Modify an existing file and commit changes

```
curl https://giphy.com/gifs/Bunim-MurrayProductions-life-kim-kardashian-e-dhjJRAs2ZbWtrYvSUT > kimk.gif # Save
 a gif from the internet
code .  # Edit the README file and add kimk.gif
git status  # The changes are not tracked
git diff README.md  # Check what has changed
git add README.md # Track the changes in README.md
git add kimk.gif # Track the new file kimk.gif
git status # All changes are now tracked
git commit -m "Add gif to README.md" # Save changes
git status # All changes are saved
git log  # Check commits history
```

II.3 Remote work with Github

GitHub

How does it work ?



How to use it ?

On the first day :

1. Fork a repository on your Github account: https://github.com/vivadata/vivadata-student (https://github.com/vivadata/vivadata-student)
2. Clone the forked repository on your computer

```
cd ~/code/YOUR_GITHUB_USERNAME
git clone git@github.com:YOUR_GITHUB_USERNAME/vivadata-student.git
git status  # It is already tracked by git
```

1. Create the remotes

```
# Create the upstream remote
git remote add upstream git@github.com:vivadata/vivadata-student.git

# Create the origin remote
```