We need tools

Sublime Text

A text editor. Your new companion, day & night.

Terminal (Bash)

Don't fear the command line.

Git & GitHub

Version Control. Collaboration.

Your turn!

Go to Setup

Different names

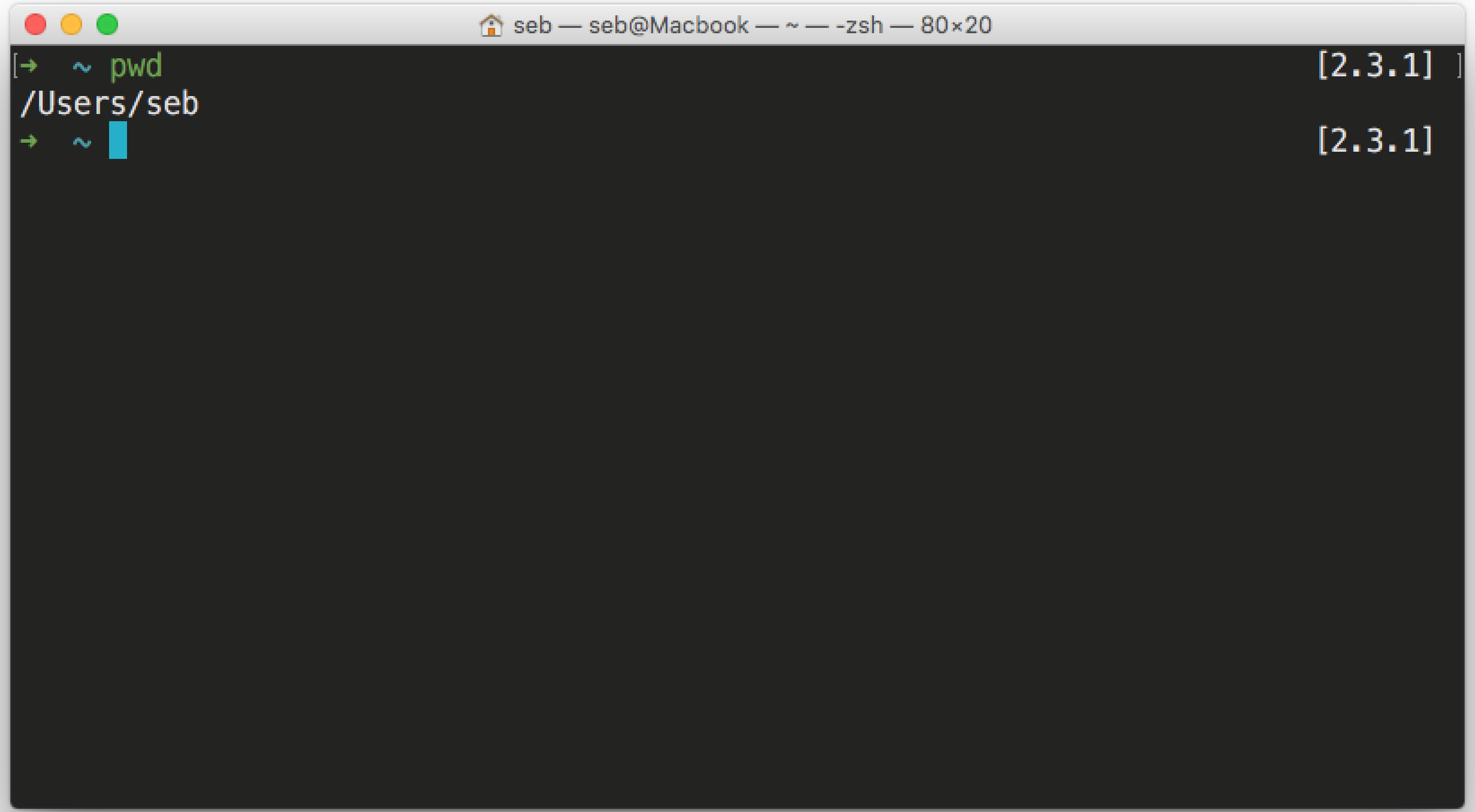
- Command prompt
- Console
- Terminal

Basic commands

Where am I ?

1. Look for the directory name before the prompt, after `➜`
2. Or print the path of the current directory

```
pwd
```

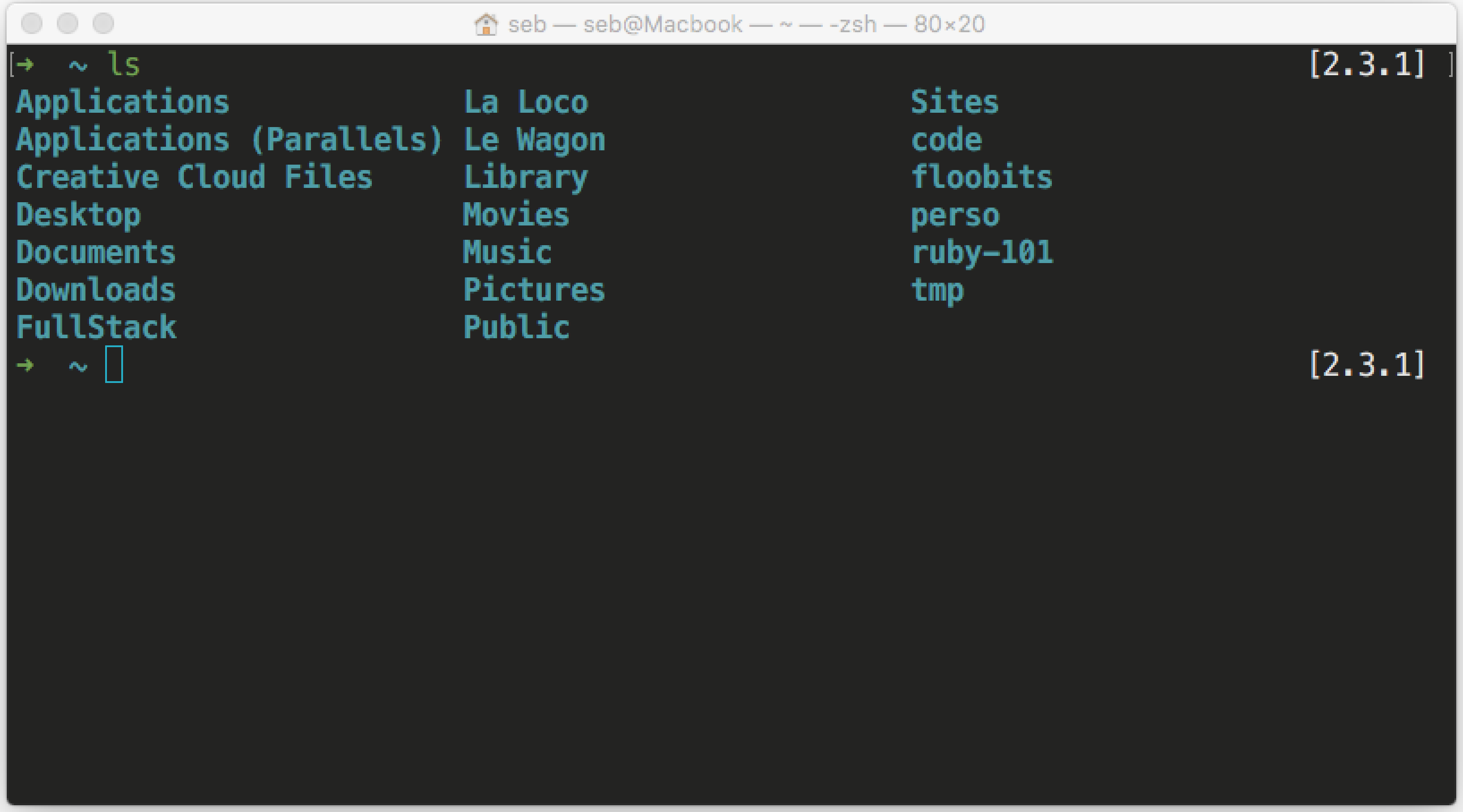


```
➜  ~ pwd                                                          [2.3.1]
/Users/seb
➜  ~                                                              [2.3.1]
```

That's your `$HOME` directory.

Where can I go ?

ls (or **ll**, an alias of `ls -lh`)

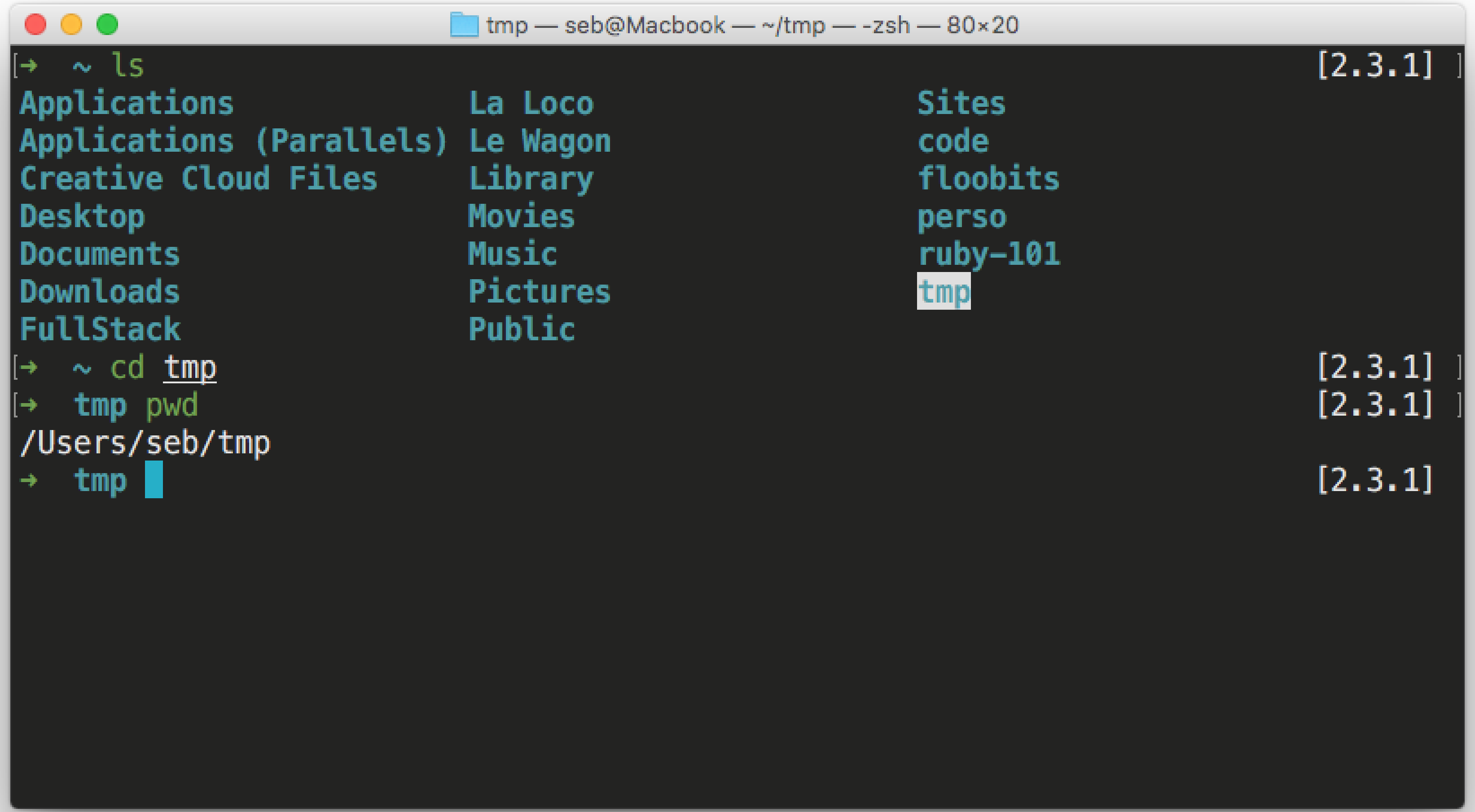


seb — seb@Macbook — ~ — -zsh — 80×20

```
➜  ~ ls                                                           [2.3.1]
Applications                La Loco                 Sites
Applications (Parallels) Le Wagon                code
Creative Cloud Files        Library                 floobits
Desktop                     Movies                  perso
Documents                   Music                   ruby-101
Downloads                   Pictures                tmp
FullStack                   Public
➜  ~                                                              [2.3.1]
```

Let's go there

cd <FOLDER_NAME>

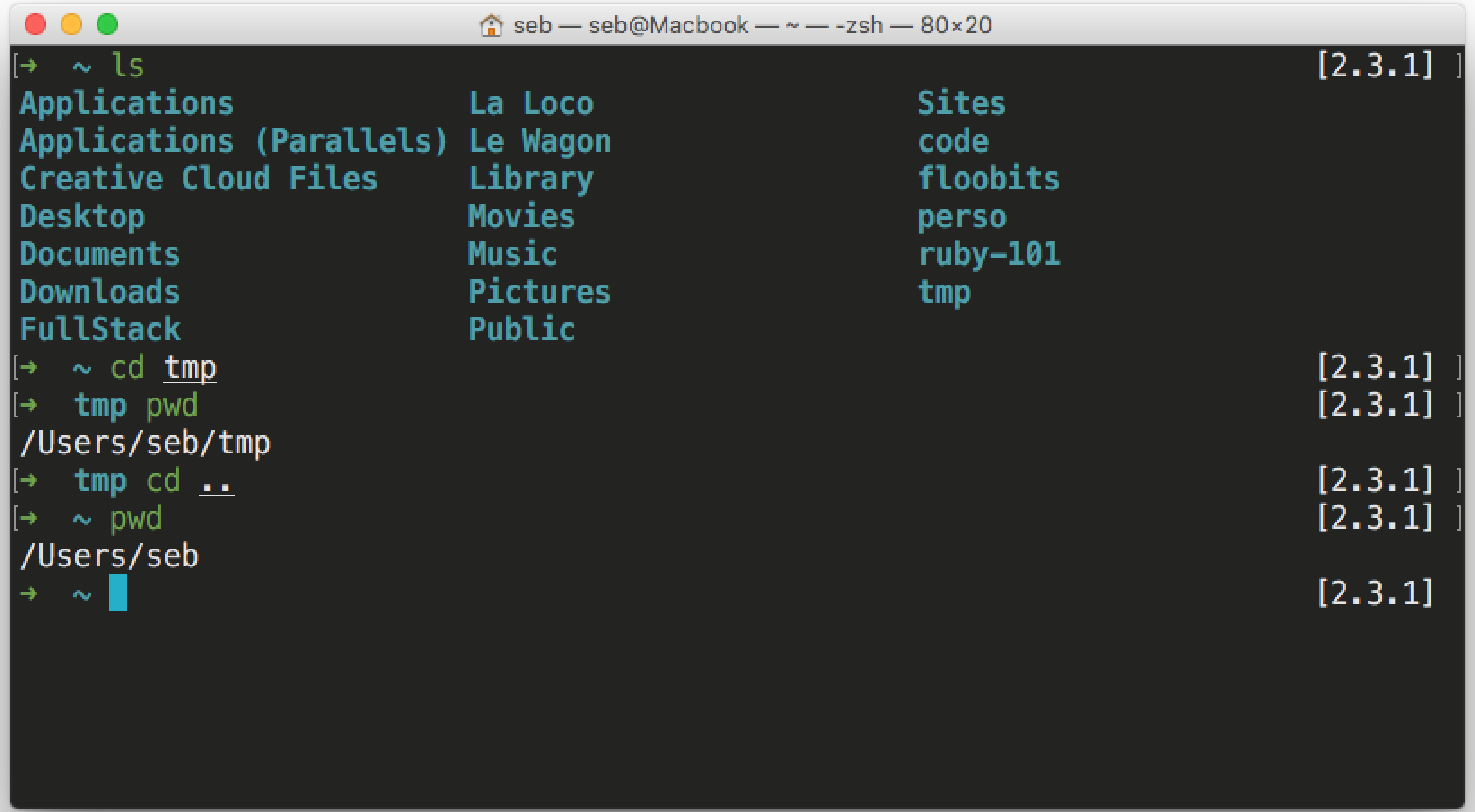


tmp — seb@Macbook — ~/tmp — -zsh — 80×20

```
➜  ~ ls                                                               [2.3.1]
Applications             La Loco                  Sites
Applications (Parallels) Le Wagon                 code
Creative Cloud Files     Library                  floobits
Desktop                  Movies                   perso
Documents                Music                    ruby-101
Downloads                Pictures                 tmp
FullStack                Public
➜  ~ cd tmp                                                           [2.3.1]
➜  tmp pwd                                                            [2.3.1]
/Users/seb/tmp
➜  tmp                                                                [2.3.1]
```

How can I go up?

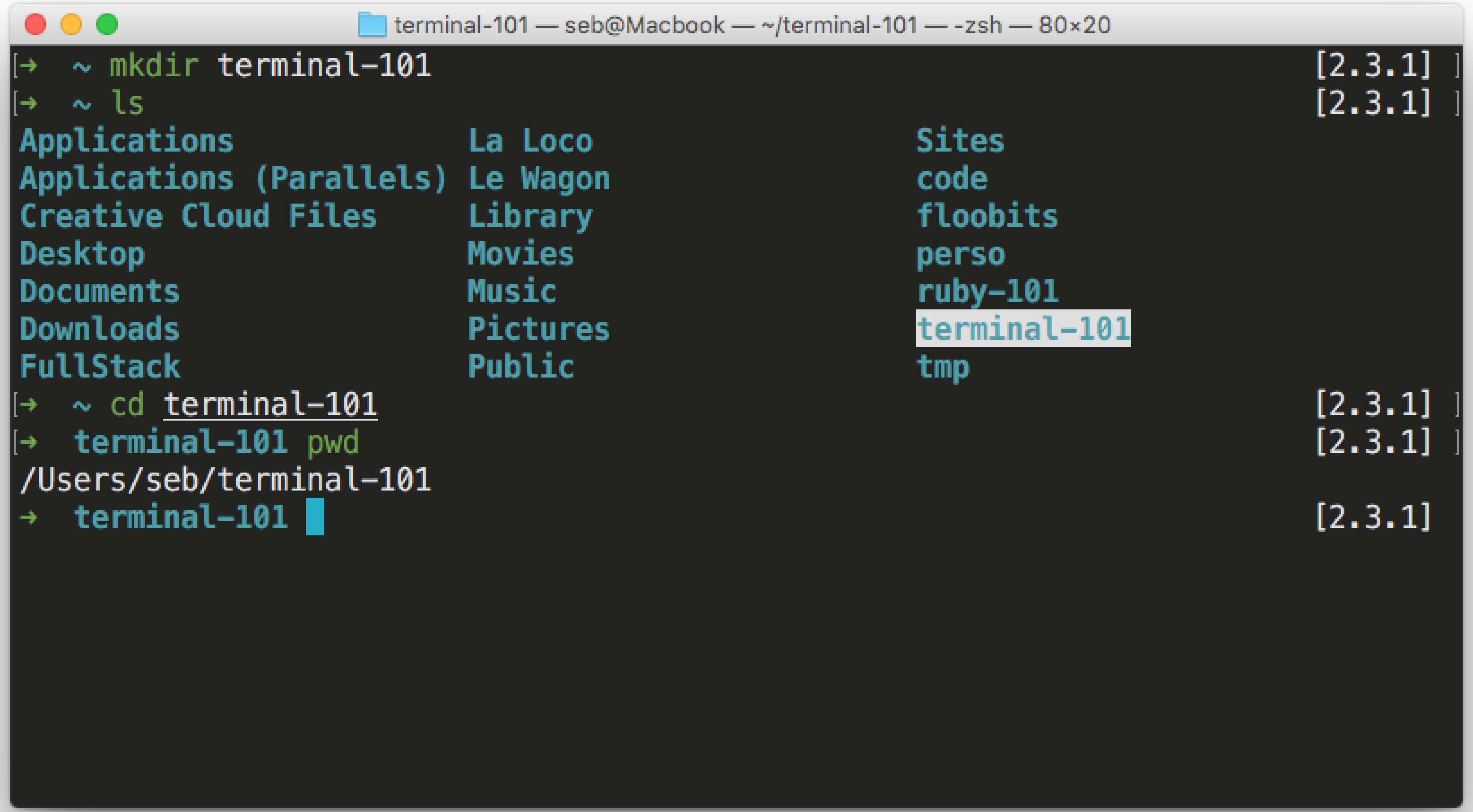
cd ..



```
➜  ~ ls                                                          [2.3.1]
Applications                La Loco                  Sites
Applications (Parallels)    Le Wagon                 code
Creative Cloud Files        Library                  floobits
Desktop                     Movies                   perso
Documents                   Music                    ruby-101
Downloads                   Pictures                 tmp
FullStack                   Public
➜  ~ cd tmp                                                      [2.3.1]
➜  tmp pwd                                                       [2.3.1]
/Users/seb/tmp
➜  tmp cd ..                                                     [2.3.1]
➜  ~ pwd                                                         [2.3.1]
/Users/seb
➜  ~                                                             [2.3.1]
```

Let's create a directory

mkdir <NEW_FOLDER>

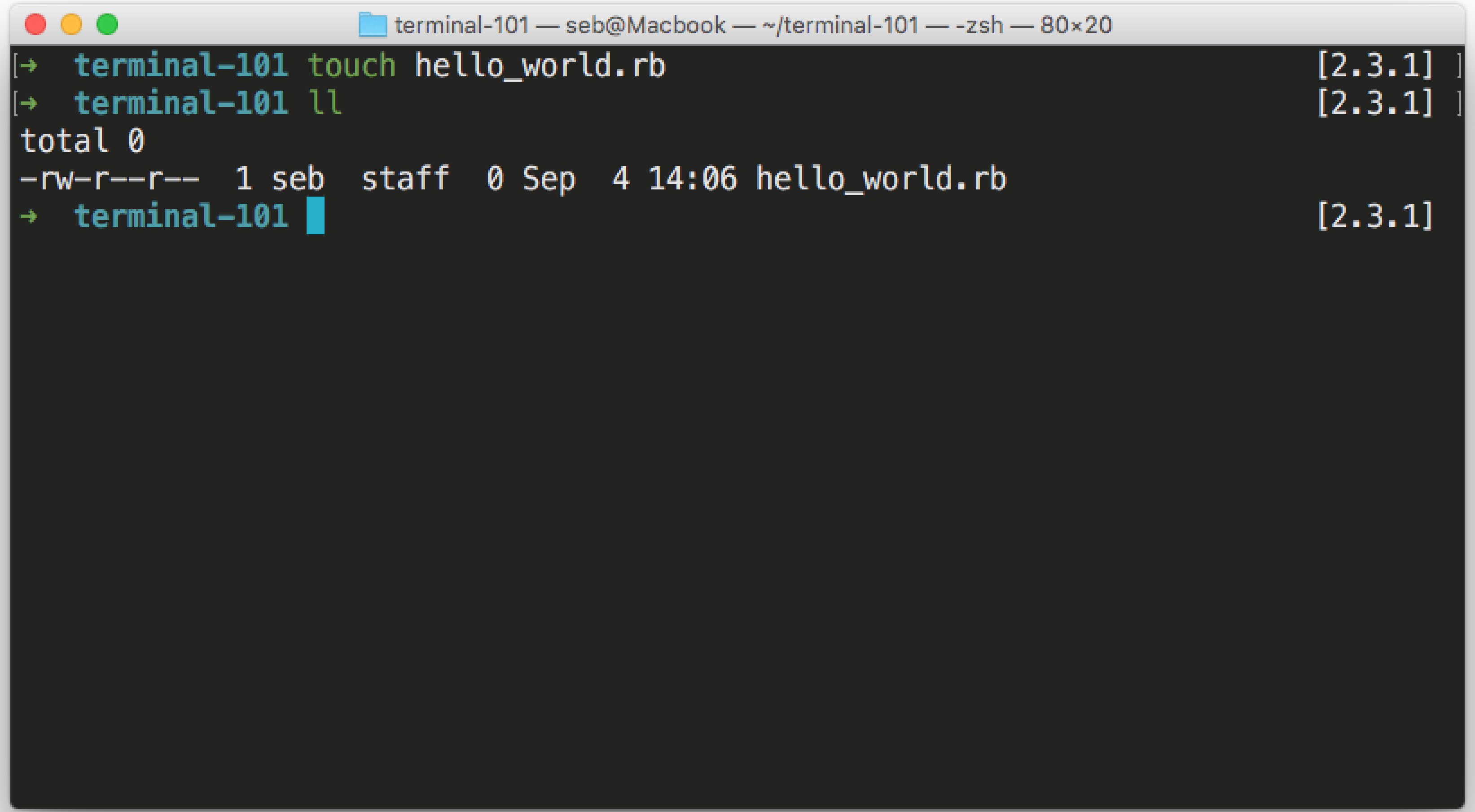


terminal-101 — seb@Macbook — ~/terminal-101 — -zsh — 80×20

```
[→  ~ mkdir terminal-101                                              [2.3.1] ]
[→  ~ ls                                                              [2.3.1] ]
Applications             La Loco                  Sites
Applications (Parallels) Le Wagon                 code
Creative Cloud Files     Library                  floobits
Desktop                  Movies                   perso
Documents                Music                    ruby-101
Downloads                Pictures                 terminal-101
FullStack                Public                   tmp
[→  ~ cd terminal-101                                                 [2.3.1] ]
[→  terminal-101 pwd                                                  [2.3.1] ]
/Users/seb/terminal-101
 →  terminal-101                                                      [2.3.1]
```

Let's create a file

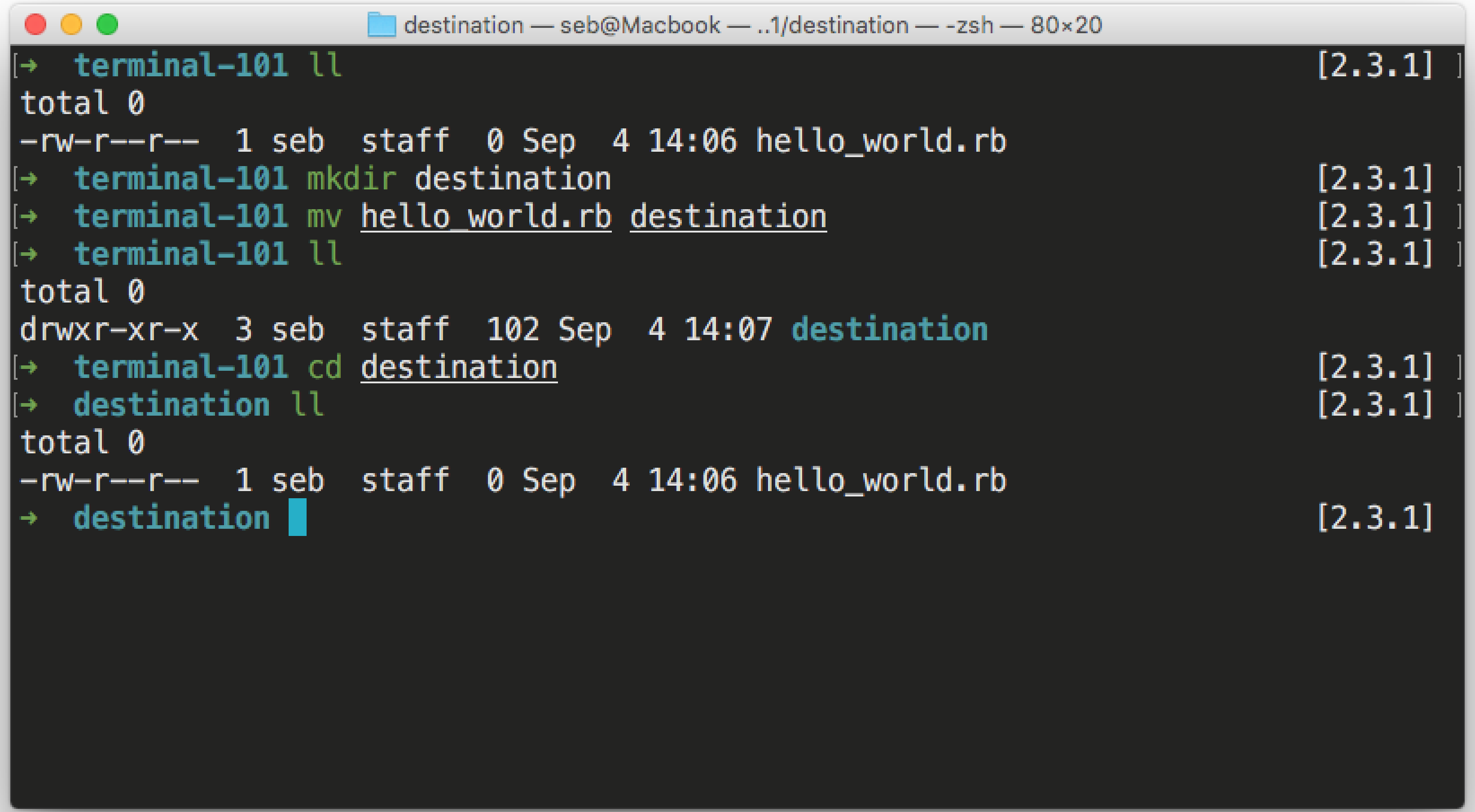
touch <FILE_NAME>



```
➜  terminal-101 touch hello_world.rb                                    [2.3.1]
➜  terminal-101 ll                                                      [2.3.1]
total 0
-rw-r--r--  1 seb  staff  0 Sep  4 14:06 hello_world.rb
➜  terminal-101                                                         [2.3.1]
```

Let's move a file (or directory)

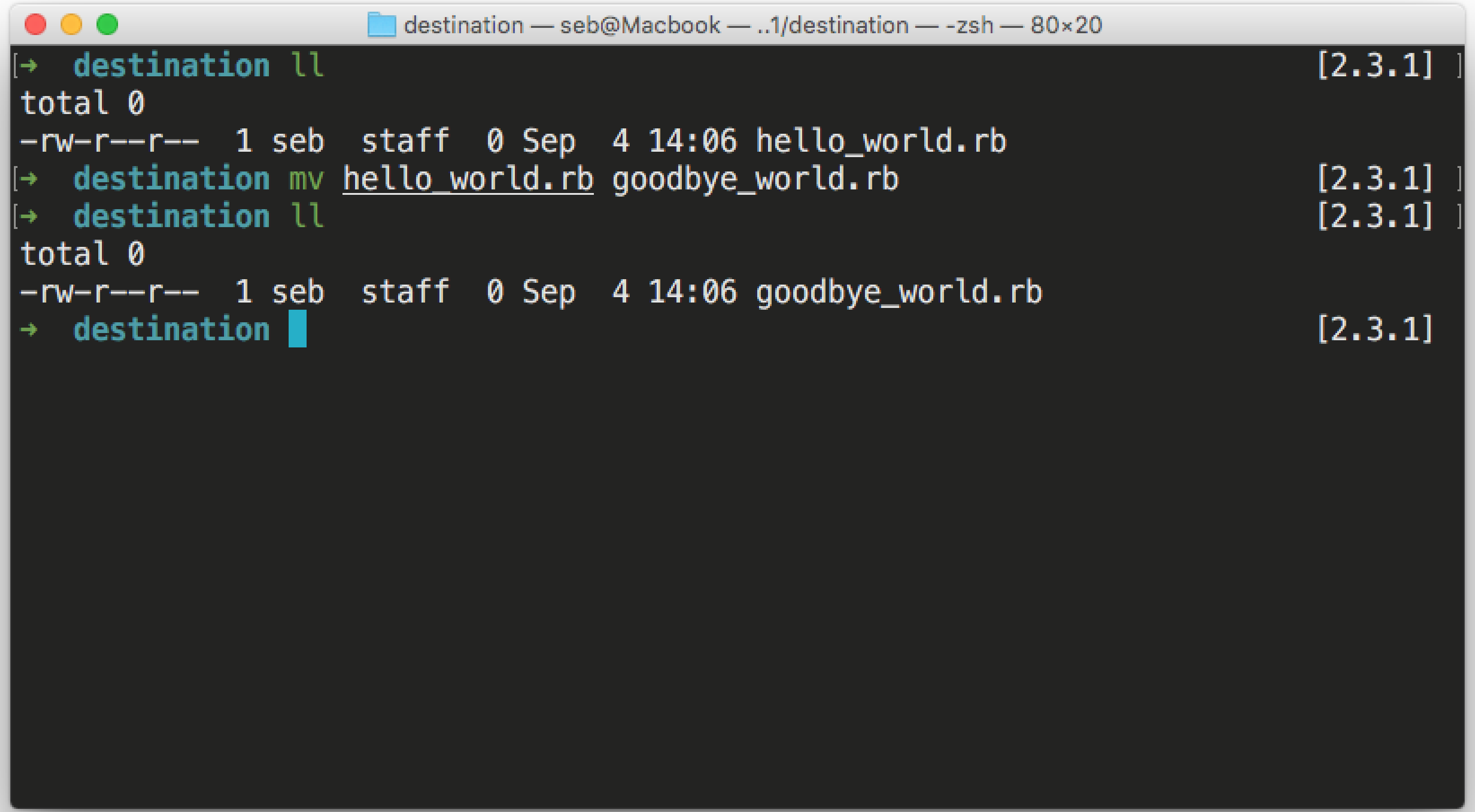
mv <FILE_NAME> <FOLDER_NAME>



```
➜  terminal-101 ll                                                        [2.3.1]
total 0
-rw-r--r--  1 seb  staff  0 Sep  4 14:06 hello_world.rb
➜  terminal-101 mkdir destination                                         [2.3.1]
➜  terminal-101 mv hello_world.rb destination                             [2.3.1]
➜  terminal-101 ll                                                        [2.3.1]
total 0
drwxr-xr-x  3 seb  staff  102 Sep  4 14:07 destination
➜  terminal-101 cd destination                                            [2.3.1]
➜  destination ll                                                         [2.3.1]
total 0
-rw-r--r--  1 seb  staff  0 Sep  4 14:06 hello_world.rb
➜  destination                                                            [2.3.1]
```

Let's rename a file (or directory)

mv <FILE_NAME> <NEW_FILENAME>

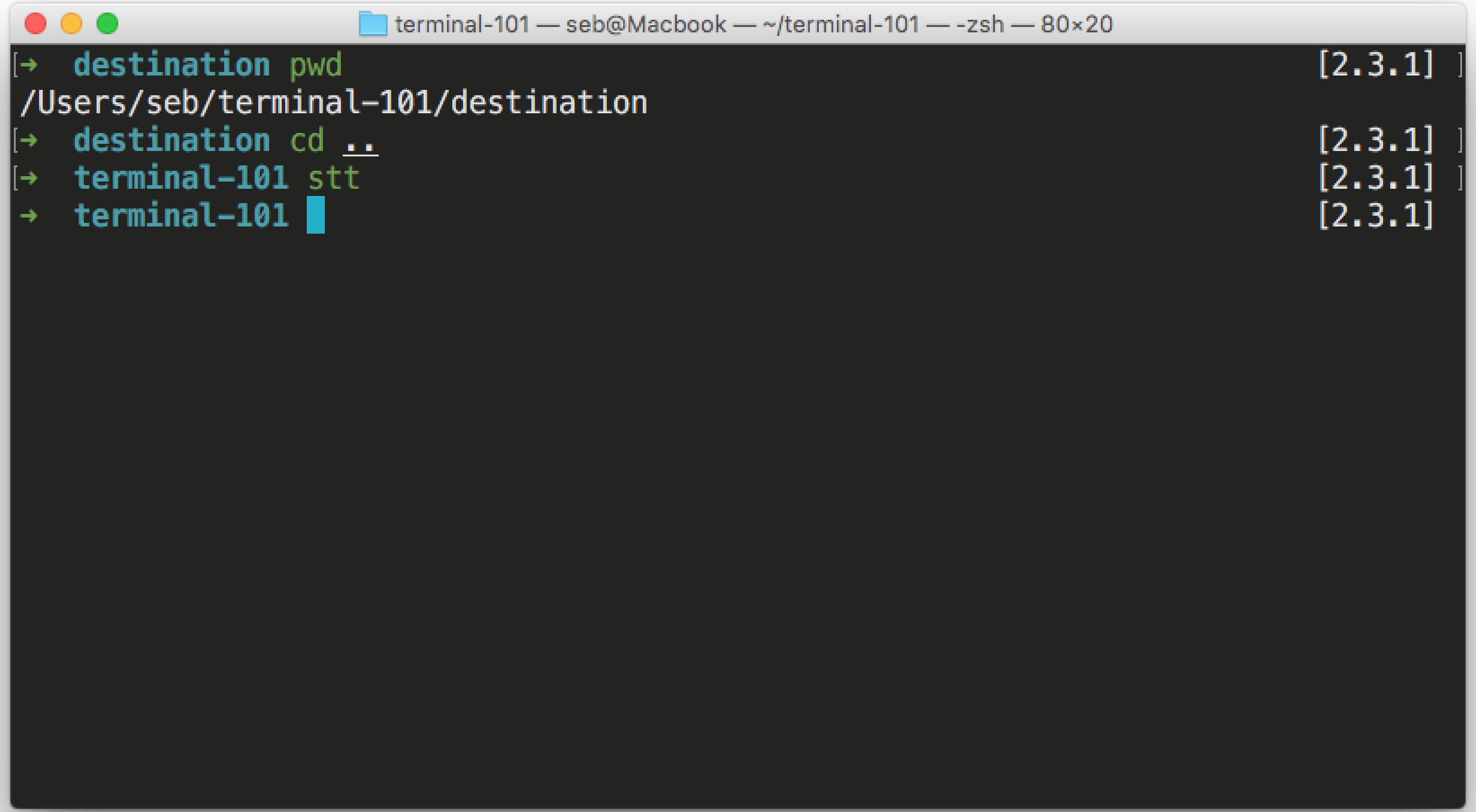


destination — seb@Macbook — ..1/destination — -zsh — 80×20

```
→  destination ll                                                    [2.3.1]
total 0
-rw-r--r--  1 seb  staff  0 Sep  4 14:06 hello_world.rb
→  destination mv hello_world.rb goodbye_world.rb                    [2.3.1]
→  destination ll                                                    [2.3.1]
total 0
-rw-r--r--  1 seb  staff  0 Sep  4 14:06 goodbye_world.rb
→  destination                                                       [2.3.1]
```

Open current directory in Sublime Text

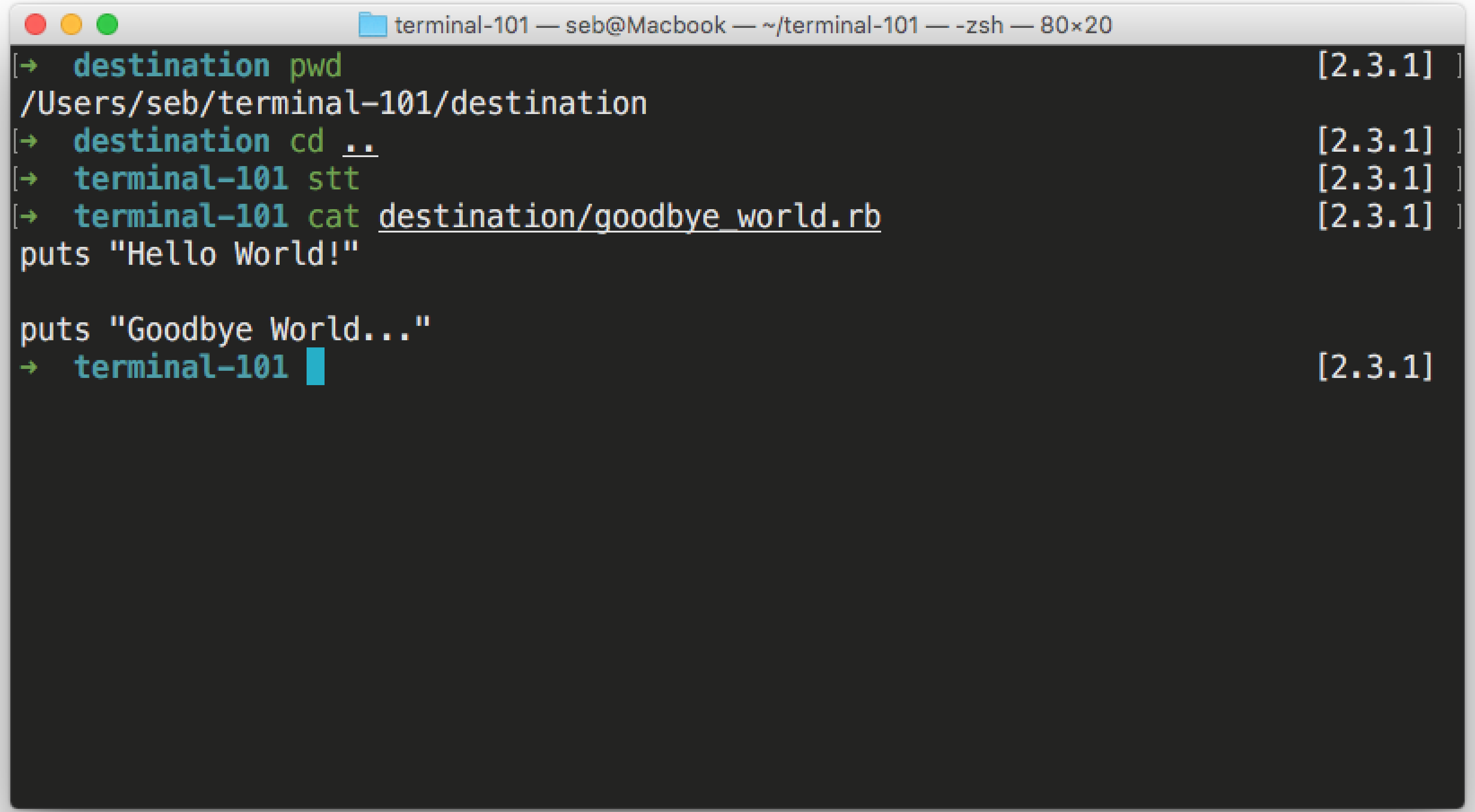
Open your current directory in Sublime with **stt**


terminal-101 — seb@Macbook — ~/terminal-101 — -zsh — 80×20

```
➜  destination pwd                                                      [2.3.1]
/Users/seb/terminal-101/destination
➜  destination cd ..                                                    [2.3.1]
➜  terminal-101 stt                                                     [2.3.1]
➜  terminal-101                                                         [2.3.1]
```

Let's view the content of a text file

cat <FILE_NAME>


terminal-101 — seb@Macbook — ~/terminal-101 — -zsh — 80×20

```
→  destination pwd                                                    [2.3.1]
/Users/seb/terminal-101/destination
→  destination cd ..                                                  [2.3.1]
→  terminal-101 stt                                                   [2.3.1]
→  terminal-101 cat destination/goodbye_world.rb                      [2.3.1]
puts "Hello World!"

puts "Goodbye World..."
→  terminal-101                                                       [2.3.1]
```

And many more!

Cheatsheet

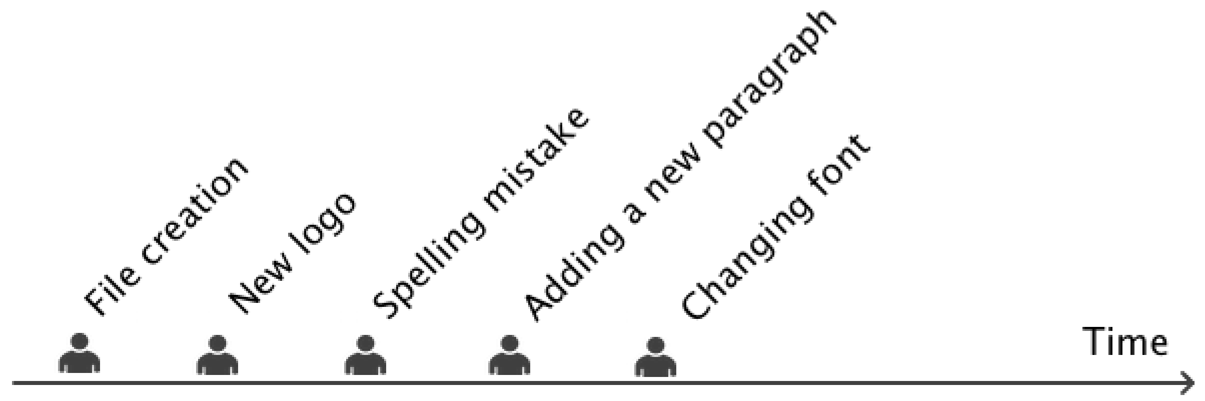
We are knowledge Workers

We create and edit **files** (text, images, etc.)

Everyday workflow

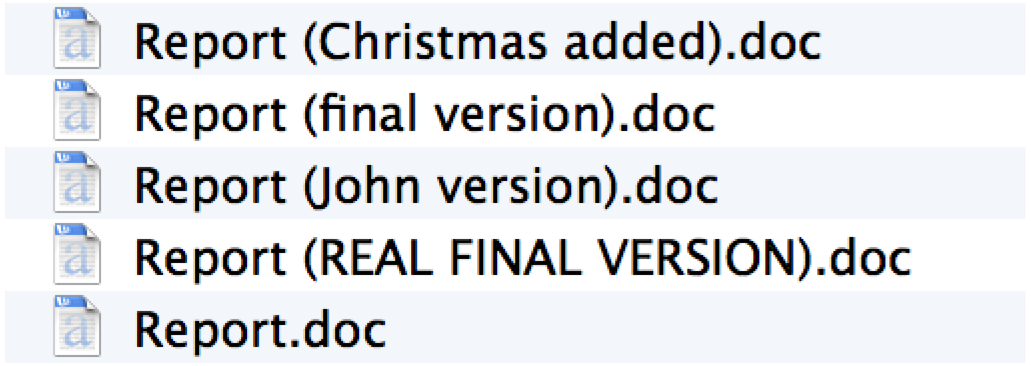
1. Create a file
2. Save it
3. Edit it
4. Save it again
5. etc.

File life


File creation
New logo
Spelling mistake
Adding a new paragraph
Changing font
Time

Manual Version Control

How most people keep track of different versions of a file



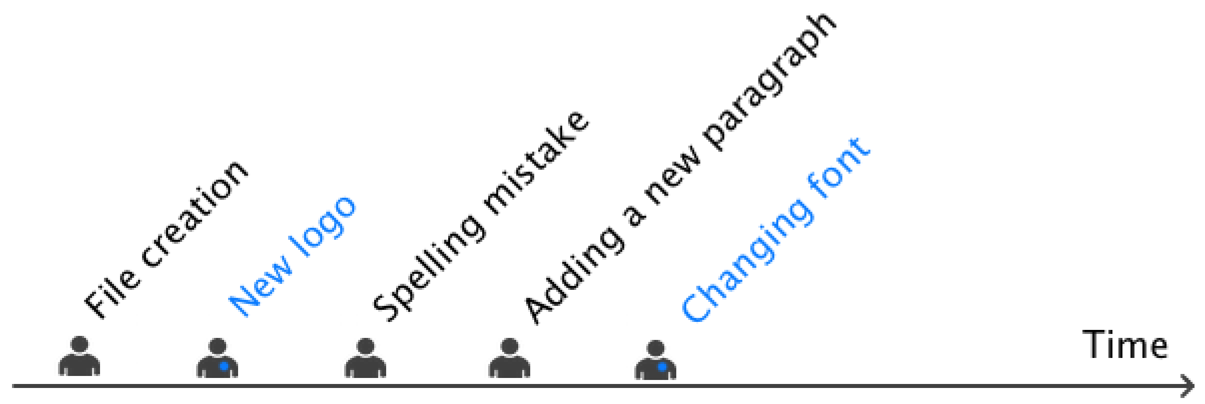
Report (Christmas added).doc
Report (final version).doc
Report (John version).doc
Report (REAL FINAL VERSION).doc
Report.doc

Can we automate this?

For each document version, we need to know:

1. **When** the file was modified
2. **What** changed
3. **Why** it was modified

There's more: Teams



That's one more question:

For each document version, we need to know:

1. When the file was modified
2. What changed
3. Why it was modified
4. **Who** did the change

In a nutshell

We want a tool which:

- tracks document versions
- keeps an history of document changes
- foster team work

That would be

git

Git basic commands

Starting

```
# From existing repository (on GitHub for instance)
git clone <github_ssh_clone_url>

# Or from scratch
mkdir new_project
cd new_project
git init
```

Status

git can tell you if your folder
has some modified files (dirty)

```
git status
```

Commit

A `commit` (a snapshot of the folder) is a 3-step job.

```
# First check which files have been modified
git status

# Then, add the ones you want to the staging area.
git add <file_1_which_has_been_modified>
git add <file_2_which_has_been_modified>

# You can review your staging area
git status

# Take a snapshot of what is in the staging area.
git commit --message "A meaningful message about this change"
```

Diff

If `git status` tells you something changed, you can inspect exactly what changed:

```
git diff
git diff <a_specific_file_or_folder>
```

Log

Show commit history with:

```
git log

# More fancy command in your ~/.gitconfig
git lg
```

Live-code: git init

Let's create a project and start tracking it

```
mkdir -p ~/code/$GITHUB_USERNAME/git-101
cd ~/code/$GITHUB_USERNAME/git-101
git init
ls -a # it has created a .git hidden folder
```

Live-code: first commit

Let's create an `index.html` file and code some basic HTML content

```
touch index.html
stt
# code some basic HTML content
```

Time to commit our work

```
git status # file not staged
git add index.html
git status # file staged, ready to commit
git commit -m "Basic HTML content for home page"
git status
```

Live-code: second commit

Let's add an image in our project

```
curl https://raw.githubusercontent.com/lewagon/karr-images/master/white_logo_red_circle.png > logo.png
stt # add <img src="logo.png"> to your HTML
```

Time to commit our work

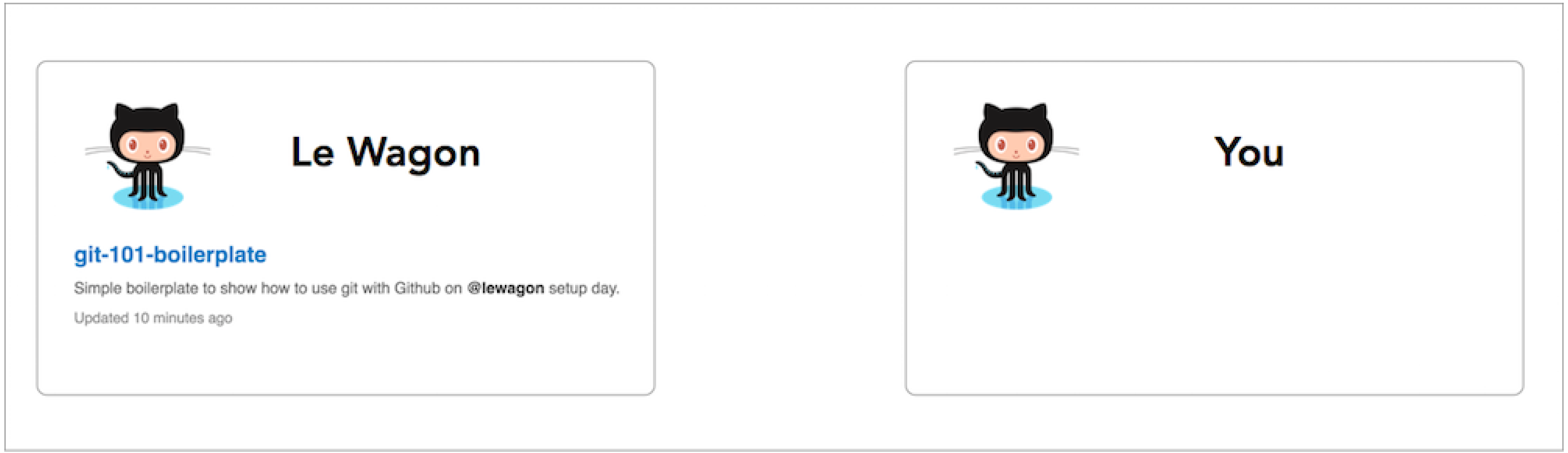
```
git status
git diff index.html # what has changed?
git add index.html
git add logo.png
git status
git commit -m "Adding logo to home page"
git status
git log # check commits history
```

Now push it up

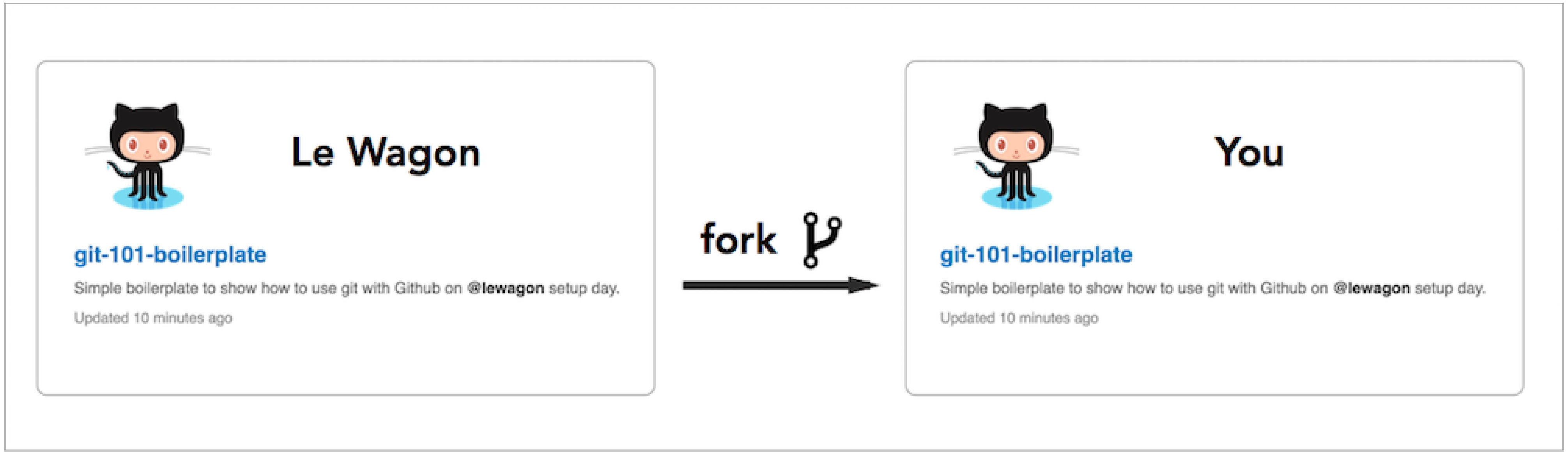
```
hub create
hub browse # can you see your new repo on Github?
git status # check again - anything to be pushed?
git push origin master
```

Remote

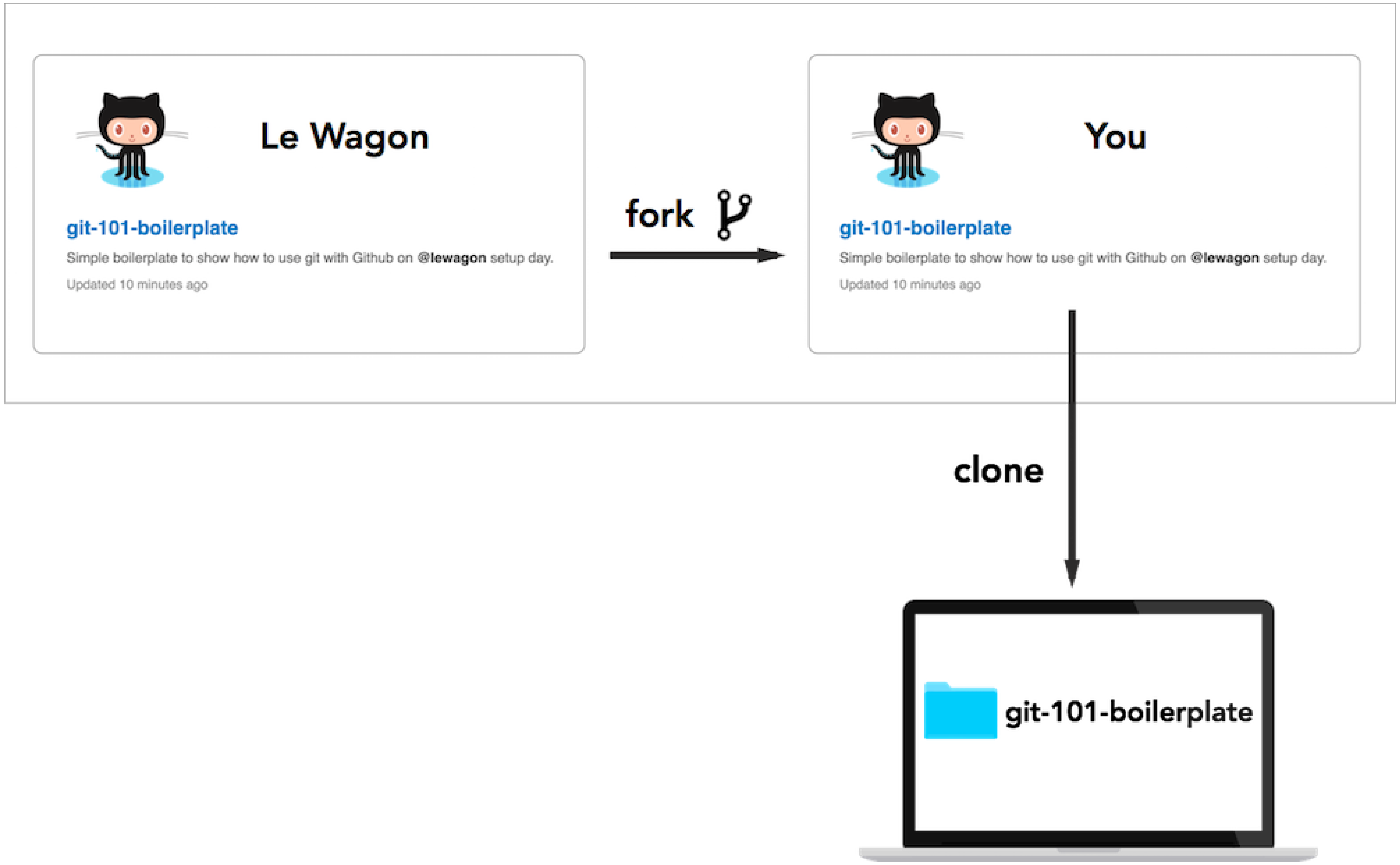
Fork and clone


Le Wagon
git-101-boilerplate
Simple boilerplate to show how to use git with Github on @lewagon setup day.
Updated 10 minutes ago
You

Fork and clone


Le Wagon
git-101-boilerplate
Simple boilerplate to show how to use git with Github on @lewagon setup day.
Updated 10 minutes ago
fork
You
git-101-boilerplate
Simple boilerplate to show how to use git with Github on @lewagon setup day.
Updated 10 minutes ago

Fork and clone


Le Wagon
git-101-boilerplate
Simple boilerplate to show how to use git with Github on @lewagon setup day.
Updated 10 minutes ago
fork
You
git-101-boilerplate
Simple boilerplate to show how to use git with Github on @lewagon setup day.
Updated 10 minutes ago
clone
git-101-boilerplate

Pushing the changes

Once you've committed your work, push it to Github.

```
# Generic command
git push <remote> <branch>

# What we'll use
git push origin master
```

Live-code: creating an new repo

1. Let's make a new repo called git-101-practice (like you can for every exercise)
2. Then let's put stuff into it

```
cd ~/code/$GITHUB_USERNAME
mkdir git-101-practice
cd git-101-practice
git init
git status # it's already tracked by git
```

3. Then make it sync remotely, by adding the github as a remote repo called `origin`

```
git remote add origin git@github.com:lewagon/git-101-practice.git
```

Live-code: commit and push

Let's make a change, commit **and push**

```
stt # change the HTML code
git add index.html
git commit -m "adding some custom text"
git status
git push origin master # Pushing on Github
```

Check that **project was updated on Github**.

Git advanced

In the next few weeks, we'll see how git can help us with

- Solving conflicts
- collaboration (using branches)
- production deployment (using multiple remotes)

Learn.lewagon.com Demo

- Navigation
- Lectures
- Classmates
- Buddies
- Exercises