The Terminal

Introduction to the *command line*

written by **Reindert-Jan Ekker**
presented by **Roy Bakker**

Overview

- Introduction
- Navigating the filesystem with **cd** and **ls**
- Reading files with **cat** and **less**
- Create, copy and delete: **mkdir**, **cp**, **rm**
- Nice tricks: history and completion
- Advanced: flags, wildcards

The Terminal

Power at your fingertips

Run programs using **text commands**

Why?

- It’s super **powerful**
- **Repeat** and **edit** previous actions
- Works very well over a **network**

Let’s dive right in!

Please start your own terminal and type along.

But.. before we start

- The **Windows** terminal is different
- We use **Linux** on **Vagrant**
- So we all have the same environment

A Word of Warning

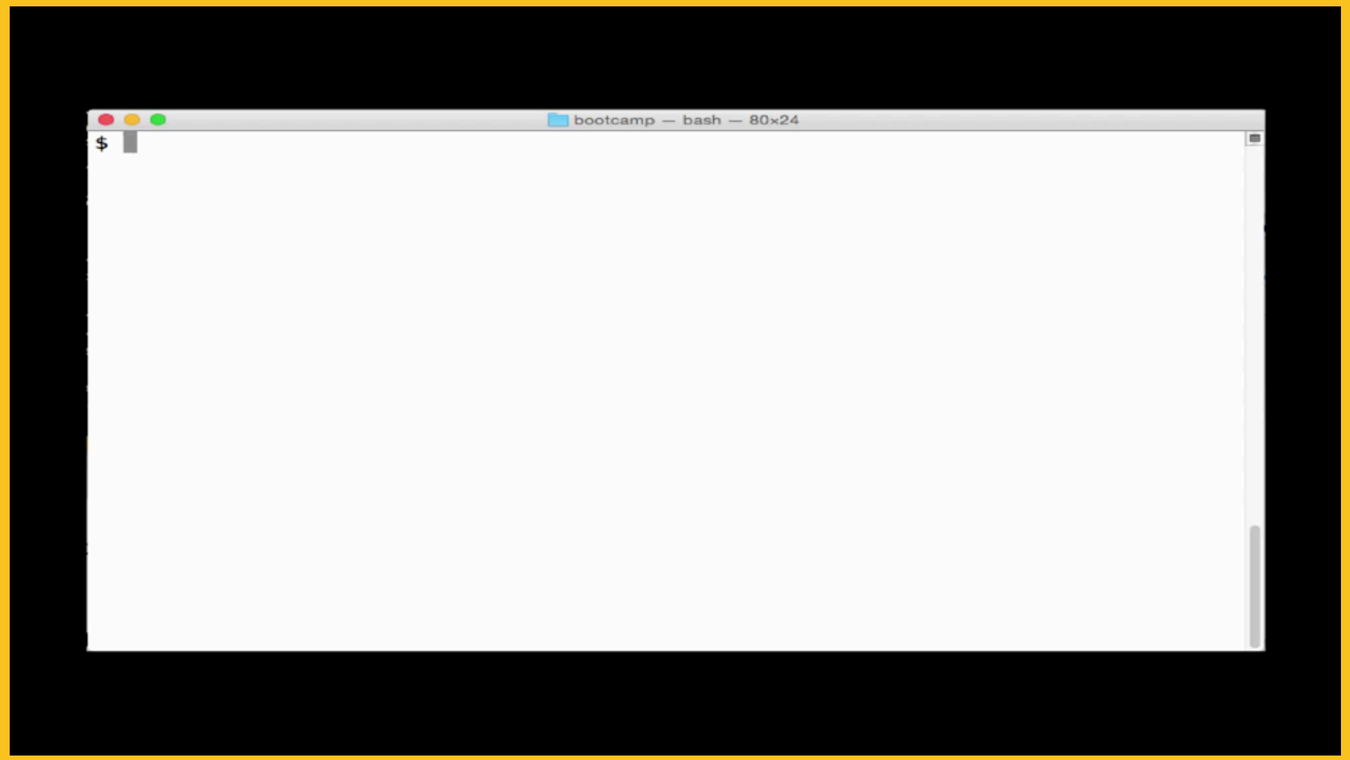
- The terminal is a **powertool**
- There is no **undo**
- **Read and re-read** what you type

Starting Vagrant

```
$ cd devbootcamp
$ vagrant up
Bringing machine up …
…
…

$ vagrant ssh
Welcome to Ubuntu 14.04.1 LTS
…
…

vagrant@developmentbootcamp $
```

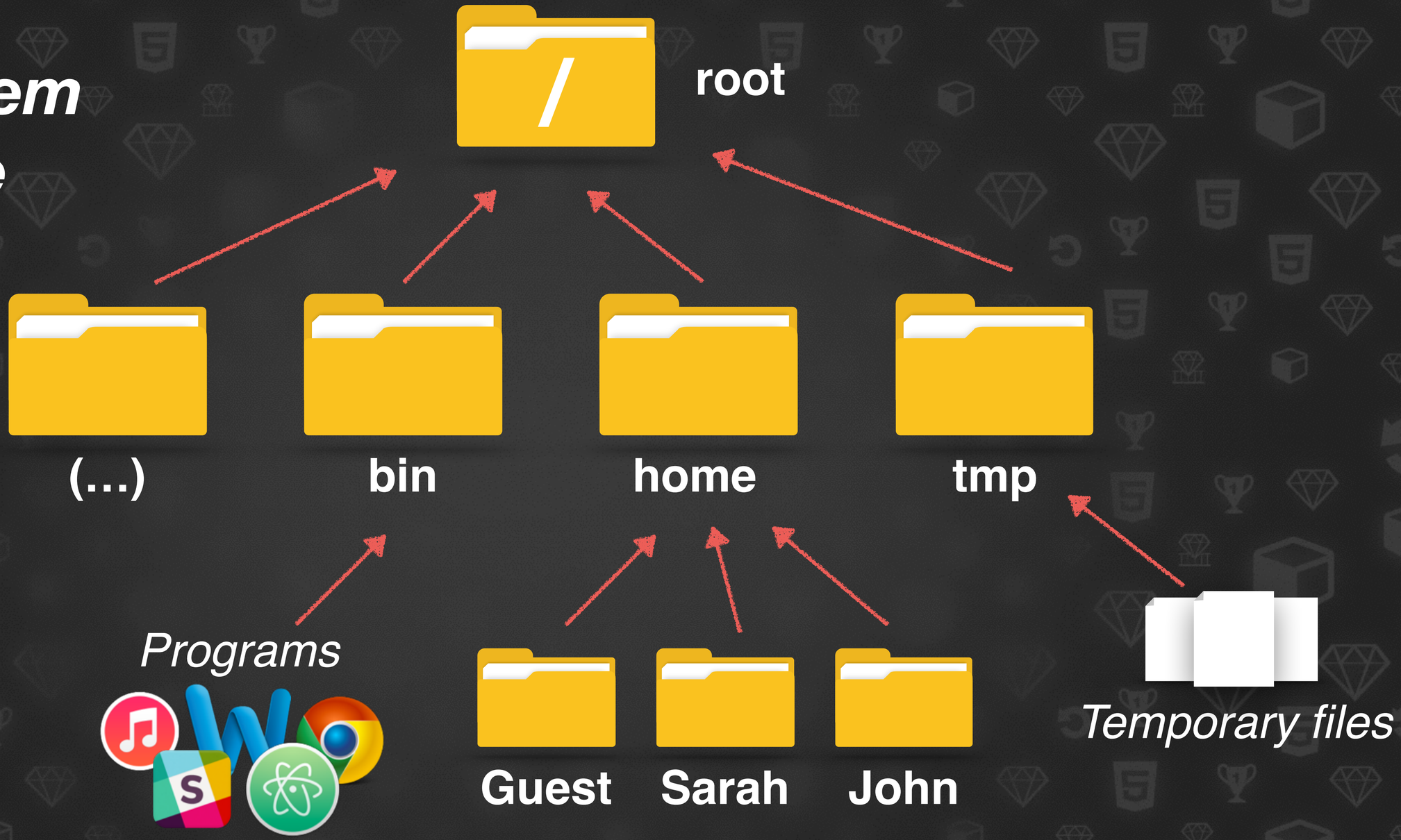


Directories

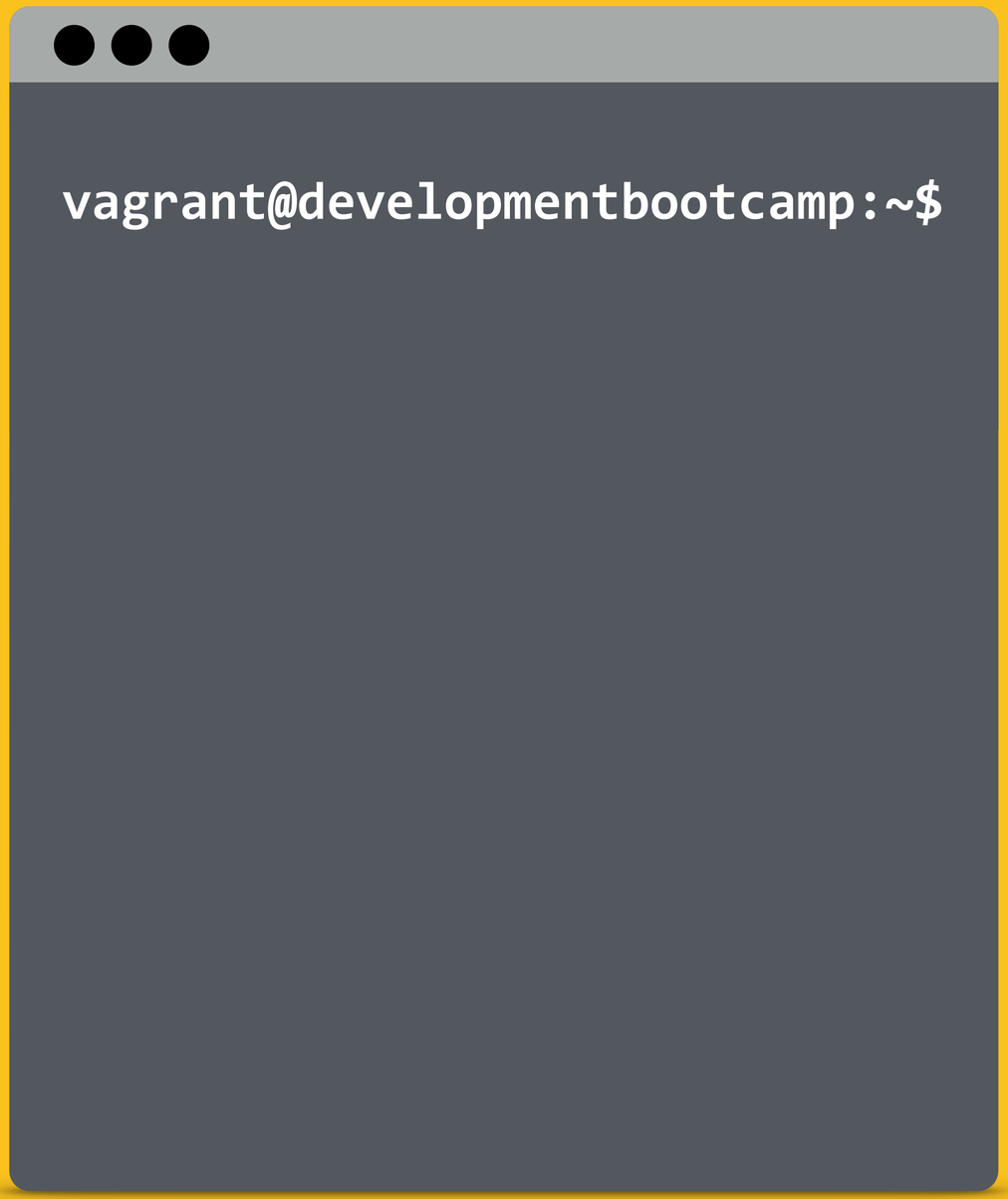
Navigating the Filesystem

Directories or Folders give structure to the filesystem
Using the **prompt** and the **cd** and **ls** commands to navigate

The Filesystem is a Tree



The Prompt



- The text ending with $ is called the prompt
- When you see it, the terminal is waiting for you to type a command
- It shows:
  - who you are
  - on which machine you are logged in
  - where you are (current working directory)

Command: ls

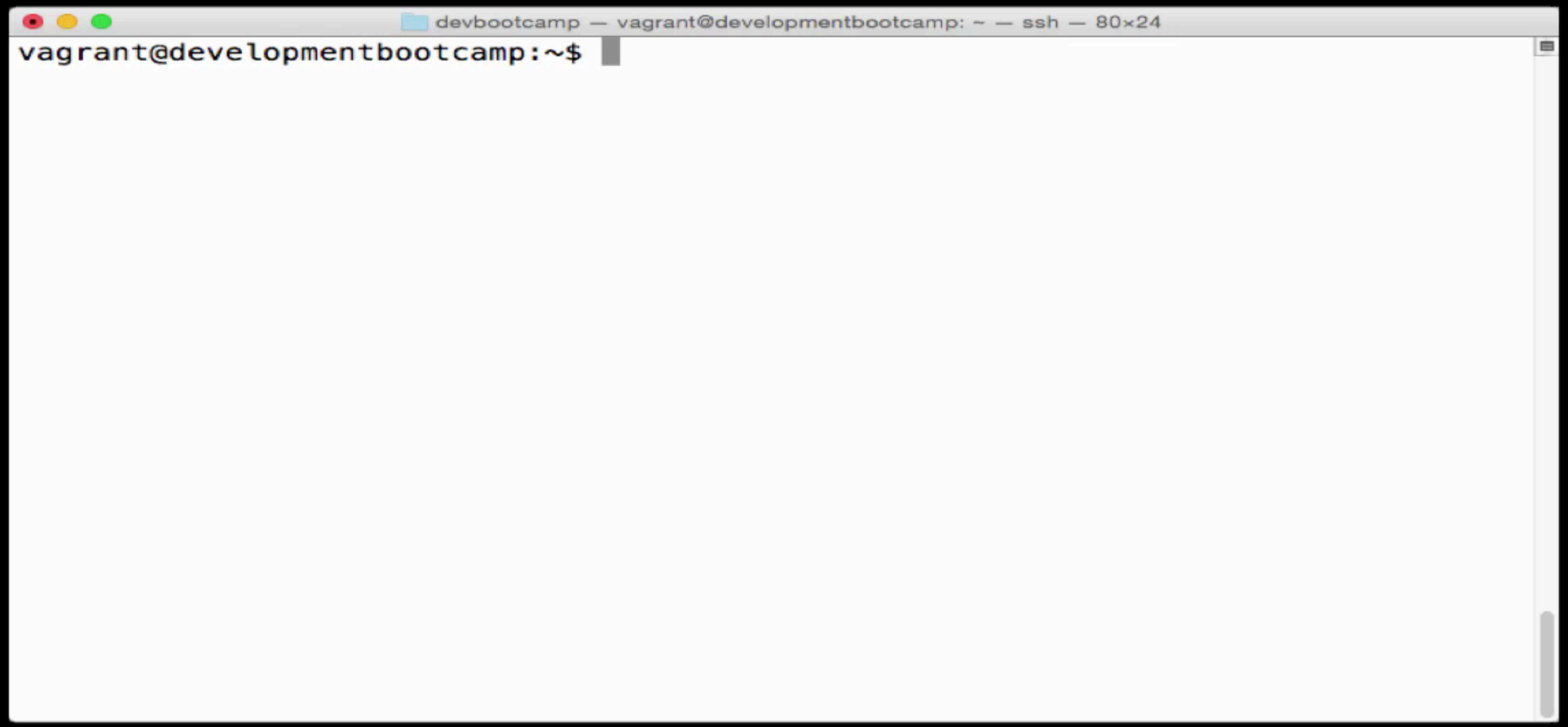
Short for **list**

Lists the **contents** of a **directory**

Command: cd

Short for **change directory**
Will move into another directory
Watch the **prompt** change..

Navigation


devbootcamp — vagrant@developmentbootcamp: ~ — ssh — 80×24
vagrant@developmentbootcamp:~$

About commands

The first word is the **program** to run
All words after that are **arguments**
These tell the program what to do

About cd

With an argument: go to that directory
No argument: go **home** (/home/vagrant)
With two dots as argument: go to **parent**
The **~** symbol is short for **home**

What do you think?

- What does **ls** do with no argument?
- What will it do **with** an argument?
- What will it do with **more** arguments?
- What if the argument is a directory that **doesn’t exist**?
- What if you pass “..” or “~” as argument?

Files

Read and edit

Reading files with **cat** and **less**

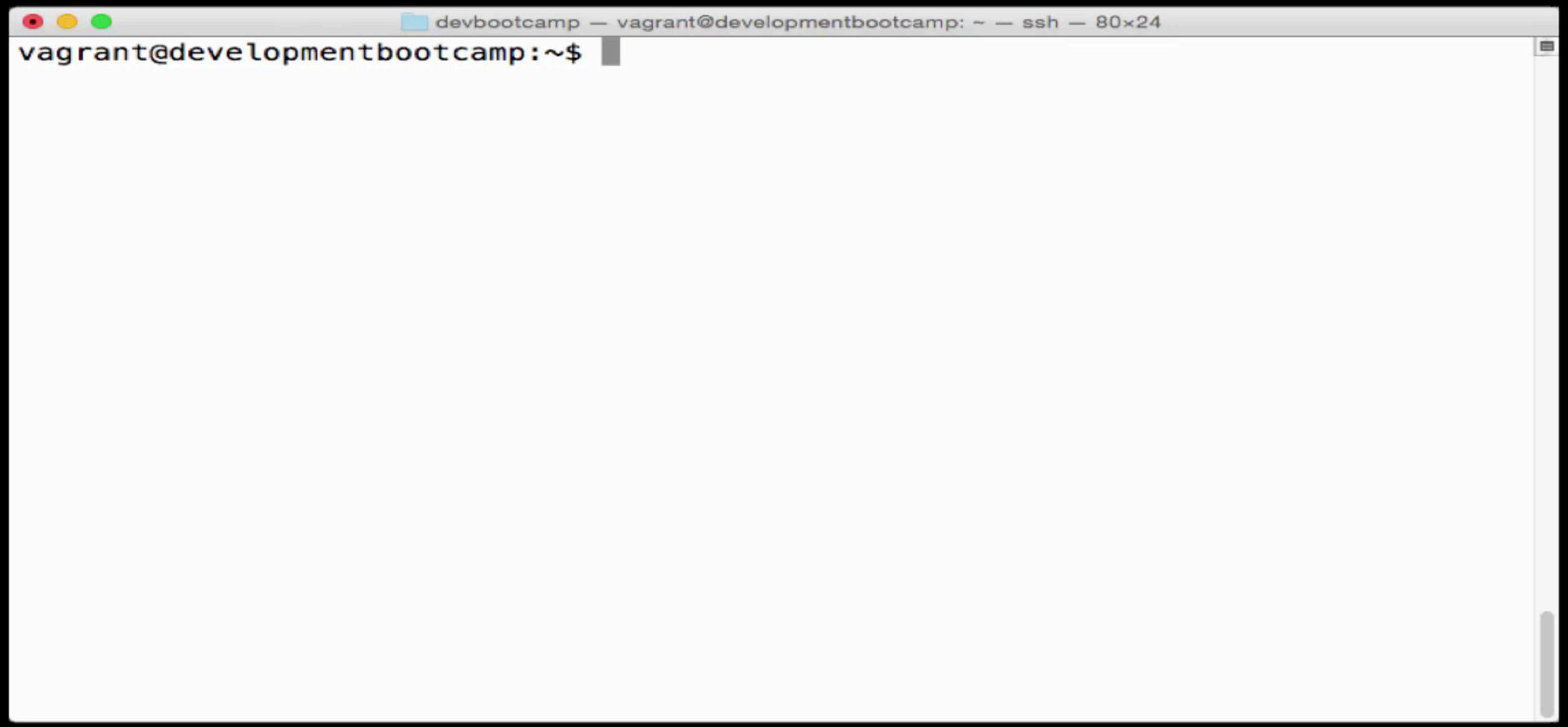
Vagrant and Shared Files

The **devbootcamp** dir on your host system
/vagrant on the vagrant Linux machine

Commands: cat and less

Both display contents of a **file** (not a dir)
Need name of the file as an **argument**
cat shows the entire contents at once
less does pagination

Navigation


devbootcamp — vagrant@developmentbootcamp: ~ — ssh — 80×24
vagrant@developmentbootcamp:~$

Create, Copy, Delete, Move

Creating directories with **mkdir**, copying with **cp**
Deleting with **rm** and **rmdir**, moving with **mv**

Directories

Create a directory with **mkdir**
Remove it with **rmdir**
NB: rmdir only works if dir is **empty**

Copying

Copy a file with **cp**

Two arguments: **cp source dest**

First argument is file to be copied

Second is destination

Moving Files

mv source dest

Works just like cp

Will silently overwrite dest

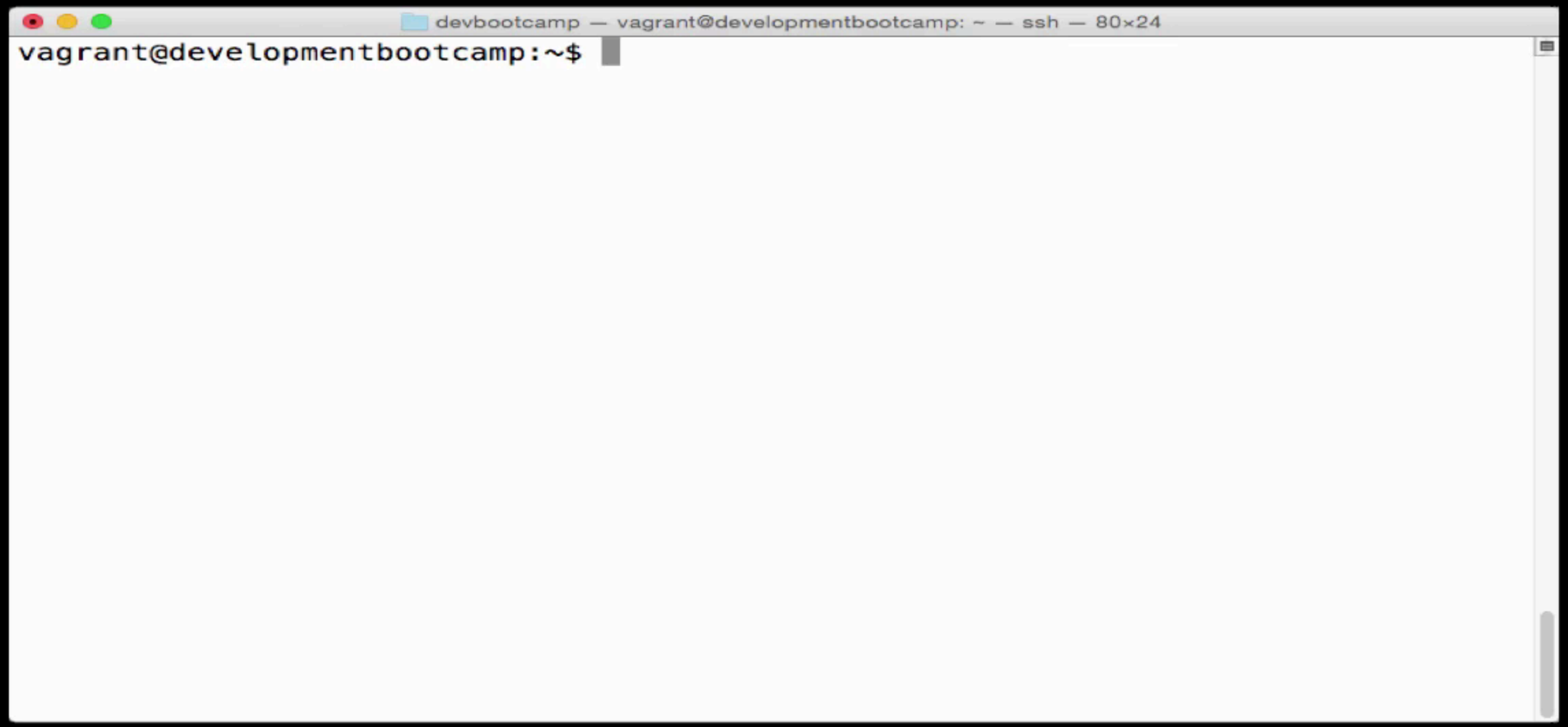
You can move a directory too

Deleting Files

Delete a file with **rm**

You will get no warning!

Navigation


devbootcamp — vagrant@developmentbootcamp: ~ — ssh — 80×24
vagrant@developmentbootcamp:~$

Question

Consider command: **cp p.jpg x**
What will happen when x is a **directory**?
What will happen when x is a **file**?
What will happen when x does not exist?

Two Handy Tricks

That will save you a lot of typing

History: it knows what you've been doing
Completion: it can predict what you want (kind of)

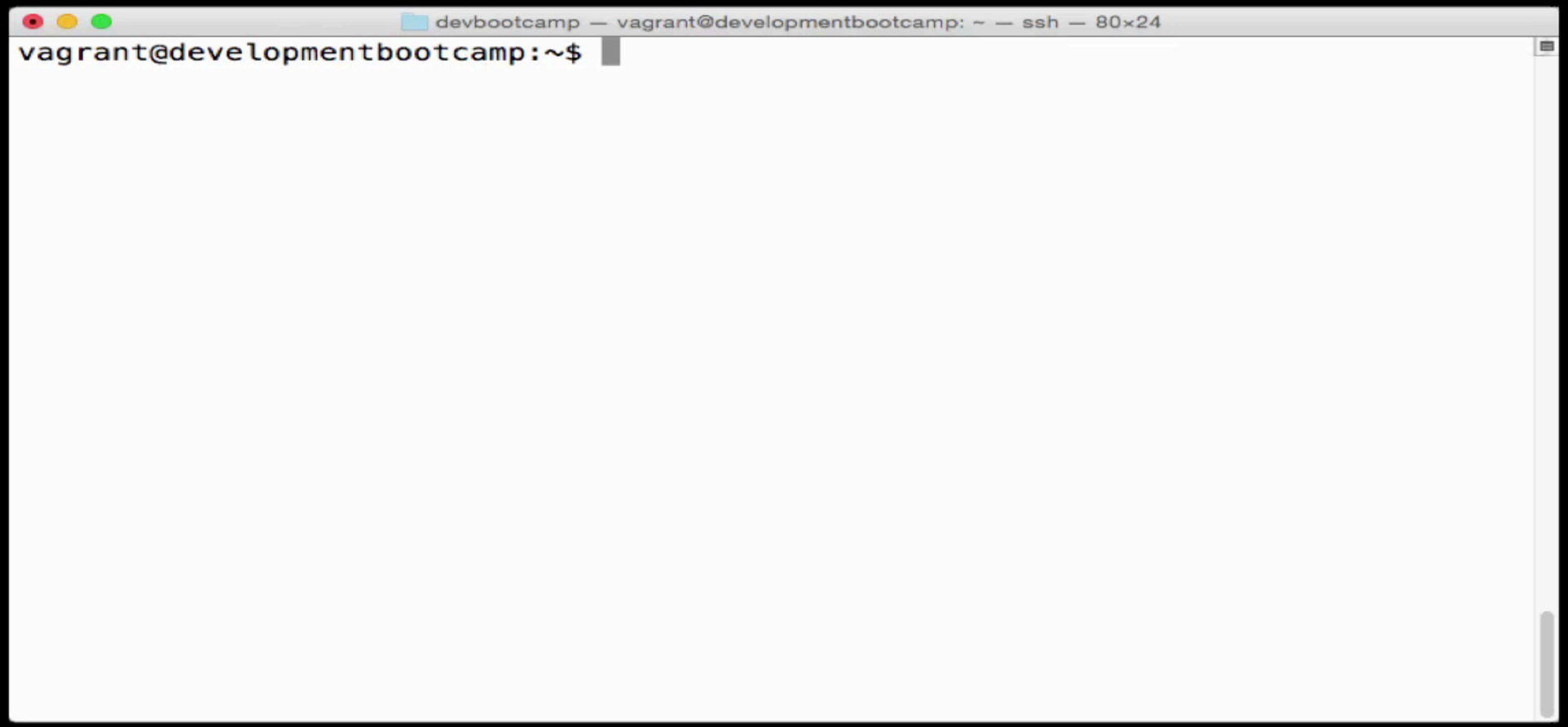
History

To recall your previous commands
Use the **up** and **down** arrow keys
You can edit the commands and re-run them

Completion

To complete file and directory names
Just type a bit of the filename
Press **TAB**

Navigation


devbootcamp — vagrant@developmentbootcamp: ~ — ssh — 80×24
vagrant@developmentbootcamp:~$

Slightly Advanced Stuff

Flags and Wildcards: do more with the same command

Flags

Arguments that start with a **dash** (-)
Usually consist of a single letter
Alter the behaviour of a command
There are many of them
Use **man** to learn about them

Some examples

- **ls -l** : detailed listing
- **ls -a** : show all files (incl. hidden)
- **rm -i** : ask for confirmation
- **rm -r** : delete a directory with all contents
- **cp -R** : copy a directory with all contents

Wildcards

Special combinations of characters
Get replaced by lists of files
Replacement happens **before** command
is executed

Wildcards: examples

- * : matches all files in the directory
- *.txt : matches all files ending in .txt

No demo?

Try it yourself! Have fun :)

Overview

`cd dir`	move into dir
`ls dir`	list contents of dir
`ls -a dir`	include hidden files
`ls -l dir`	“long format”

`cp src dest`	copy a file
`cp -R src dest`	copy a dir
`mv src dest`	move file/dir
`rm file`	delete a file

`cat file`	Show file contents
`less file`	Paginate contents

history Use up/down arrows
completion Use TAB

`mkdir dir`	create a new dir
`rmdir dir`	remove empty dir
`rm -r dir`	remove non-empty

wildcard: *, *.txt, a*
Will be replaced by matching filenames

Lunch Time!

Git/GitHub

Introduction to *version control*

written by **Reindert-Jan Ekker**
presented by **Roy Bakker**

Version Control

Total Recall without Arnold

Version Control. What is it and why do you want it?

What is it for?

- Safely **store** your code on a server
- Easily roll back your **changes**
- **Collaborate** with other people

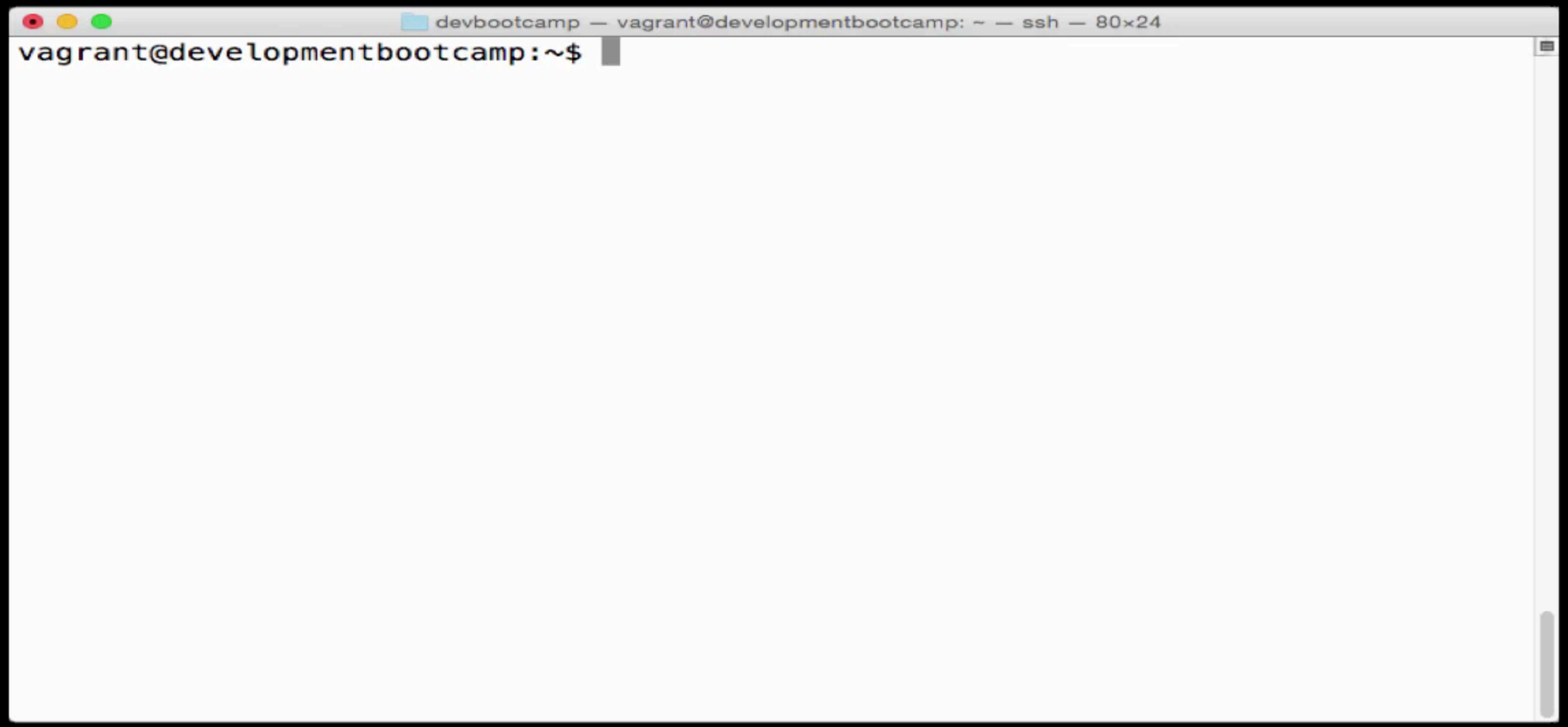
Git Repository

A repository holds your files and their version history
After making changes, you **commit** them to the repository

Sharing

After committing, you **push** your changes to the GitHub server
To get changes someone else made, you **pull**

Navigation


devbootcamp — vagrant@developmentbootcamp: ~ — ssh — 80×24
vagrant@developmentbootcamp:~$

Workshop

- Create a GitHub account
- Install GitHub Desktop
- Create a repo
- Share your work with others