
IT-566
Computer Scripting
Techniques

Week 2

Talking Points

- Development Environment Setup and Configuration (Loose Ends)
- Project Organization & Layout
- Bash Build Script
- Git Development Workflow

Development Environment

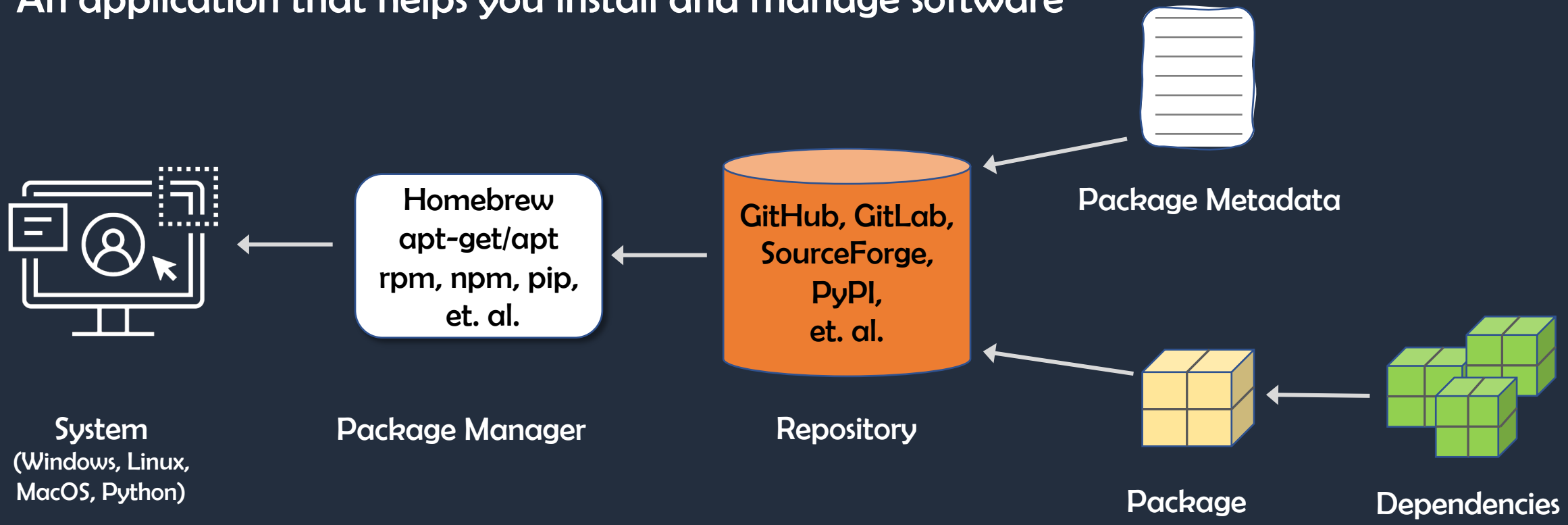
Setup and Configuration
(Loose Ends)

- Package Managers
- `tree` Command
- Configure ssh for GitHub
- `pipenv`
  Setup, Usage & Why You Need It
- `.bash_profile`
  Helpful Settings

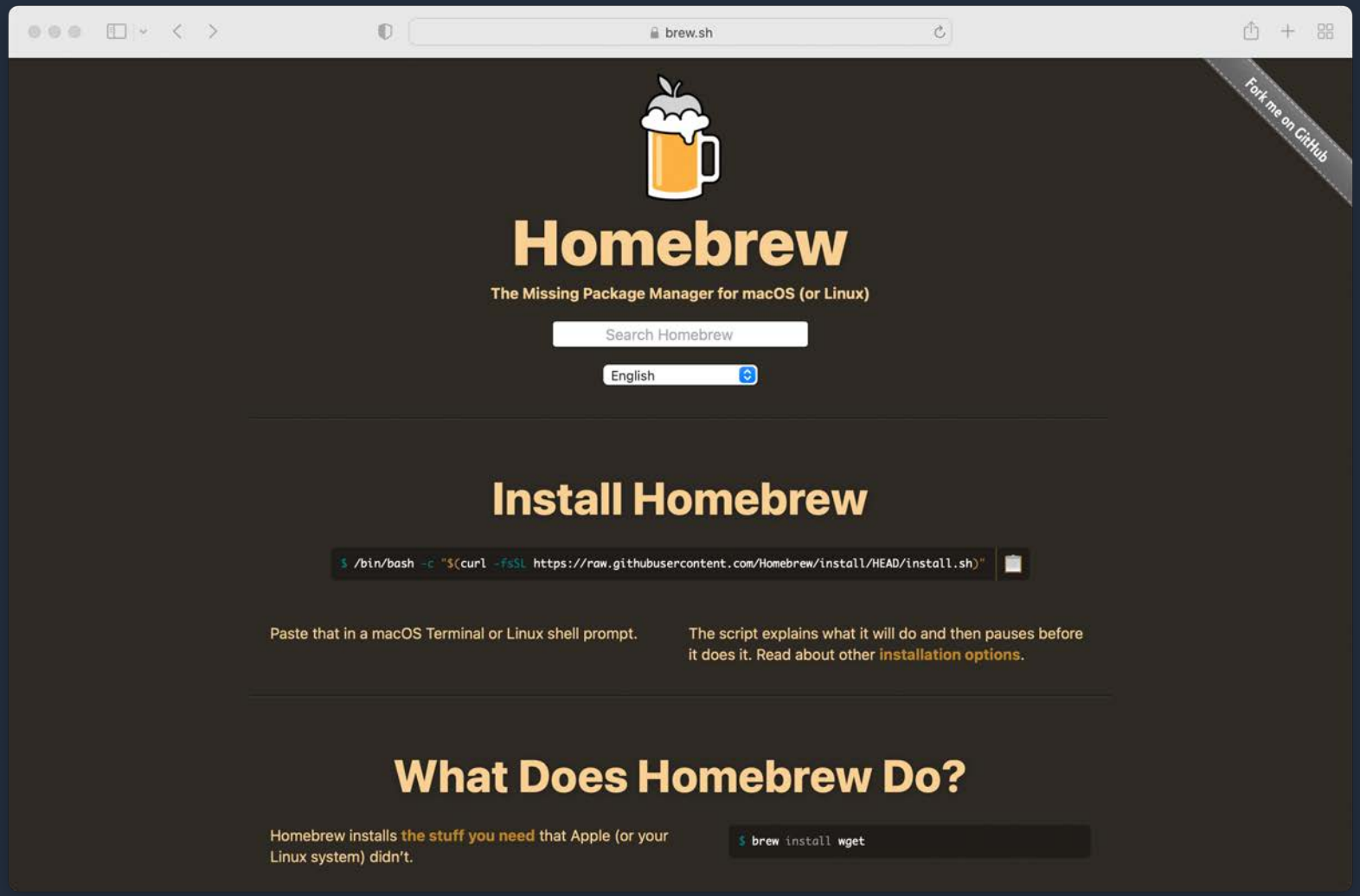
Package Managers

What Is A Package Manager?

- An application that helps you install and manage software



Homebrew MacOS (And Linux) Package Manager



- Package Manager for MacOS
  - x86 & arm64
- Simplifies Software Installation
  - Especially Unix utils
  - Developer tools
- Linux, too
  - x86 (no arm64 binaries)

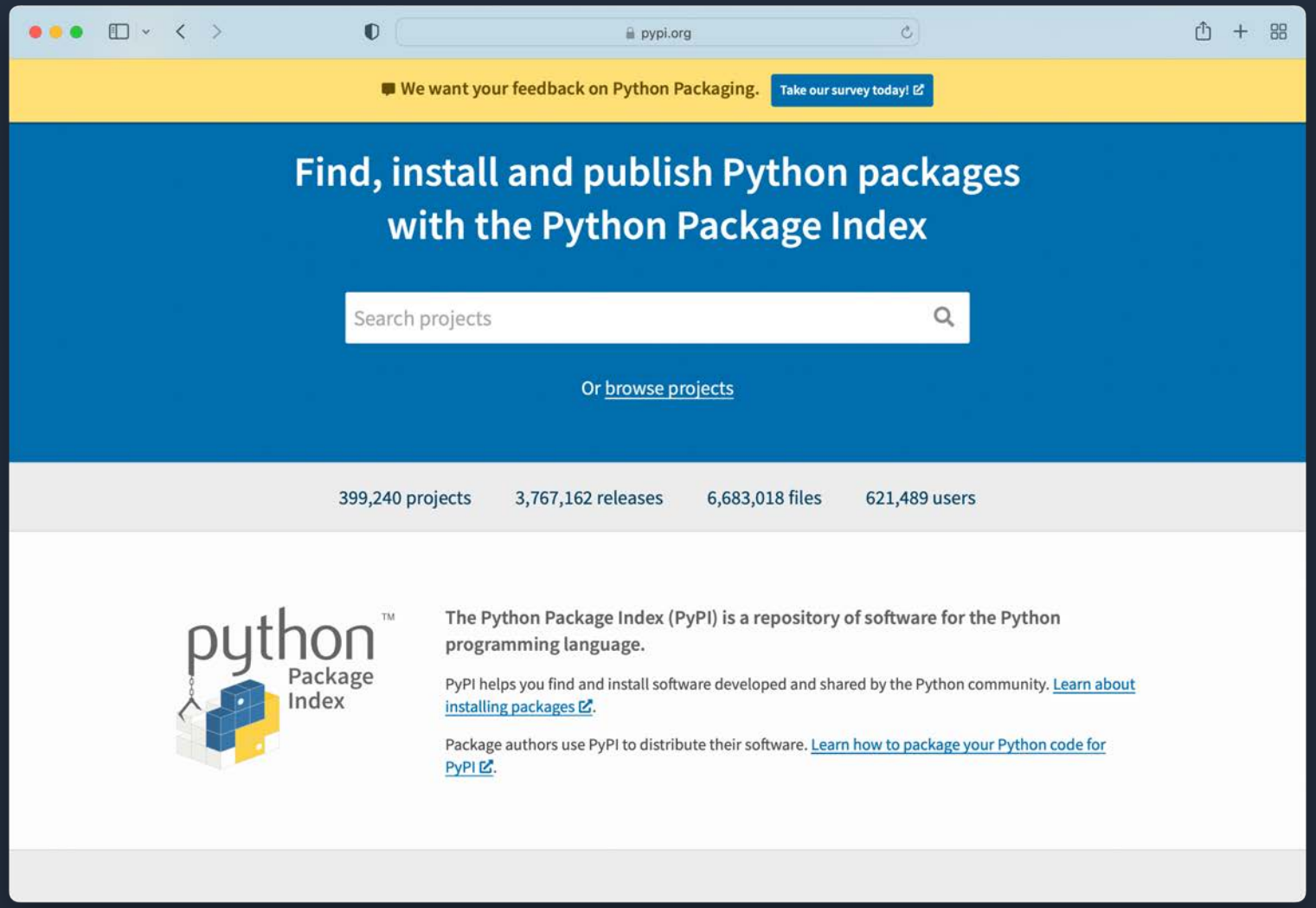
Linux Package Managers

Depends on Distribution

apt-get / apt

rpm

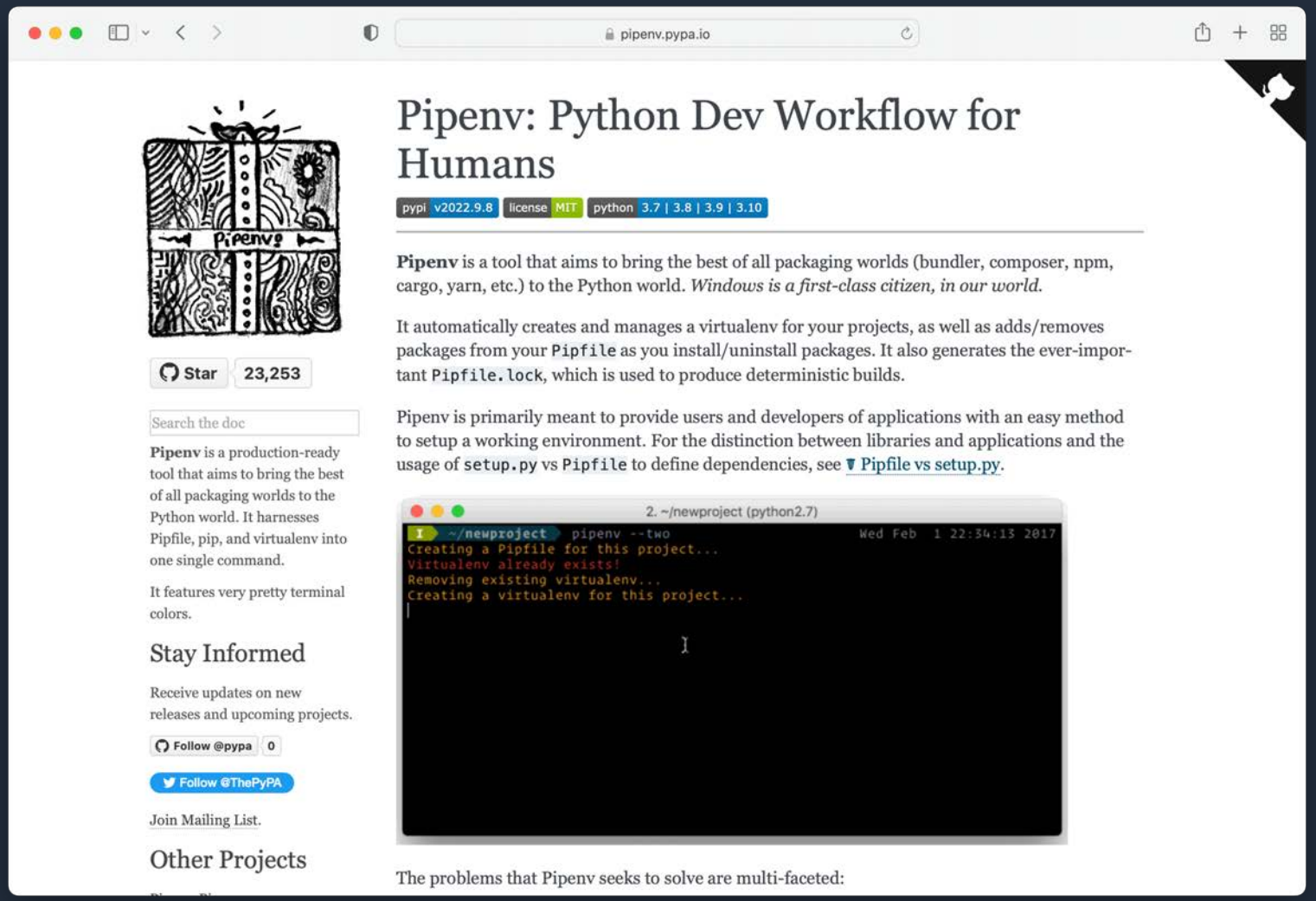
Python Package Manager — pip



- pip or pip3
  - pip for Python 2
  - pip3 for Python 3
  - Generally...
- PyPI
  - Python Package Index

pipenv Setup, Usage, & Why You Need It

Pipenv — Why You Need It



- Create and Manage Python Virtual Environments
- Why?
  - You may have multiple python projects based on different Python versions
  - Helps manage and avoid python package dependency conflicts
  - Enables deterministic builds
- With very few exceptions, avoid installing Python packages globally
- Create a new venv with pipenv and install there

pipenv – Configuration and Usage

vi

1 .bash_profile

```
# aliases
alias dir='ls -al'
alias cls='clear'
alias first-vpc='cd ~/dev/dev.aws/projects/first-vpc'
alias it590='cd ~/dev/dev.classes/it-590-aws'
alias it566='cd ~/dev/dev.classes/it-566-computer-scripting'
alias gasson='cd ~/dev/dev.gasson/marketpredictor'

test -e "${HOME}/.iterm2_shell_integration.bash" && source "${HOME}/.iterm2_shell_integration.bash"

# prompt
PS1="\n\[\033[35m\]\$(/bin/date)\n\[\033[32m\]\w \[\033[1;33m\]\$(__git_ps1
'(%s)')\n\[$(iterm2_prompt_mark)\]\[\033[1;32m\][\!:\#]\[\033[1;33m\] \u@\h $ "

test -e "${HOME}/.iterm2_shell_integration.bash" && source "${HOME}/.iterm2_shell_integration.bash"

# setup AWS cli environment
. ~/bin/aws/set-aws.sh

export PIPENV_VENV_IN_PROJECT="true"
export PATH
eval "$(/opt/homebrew/bin/brew shellenv)"
```

NORMAL .bash_profile 87% 21:1

- export the environment variable: PIPENV_VENV_IN_PROJECT="true"
- Restart shell

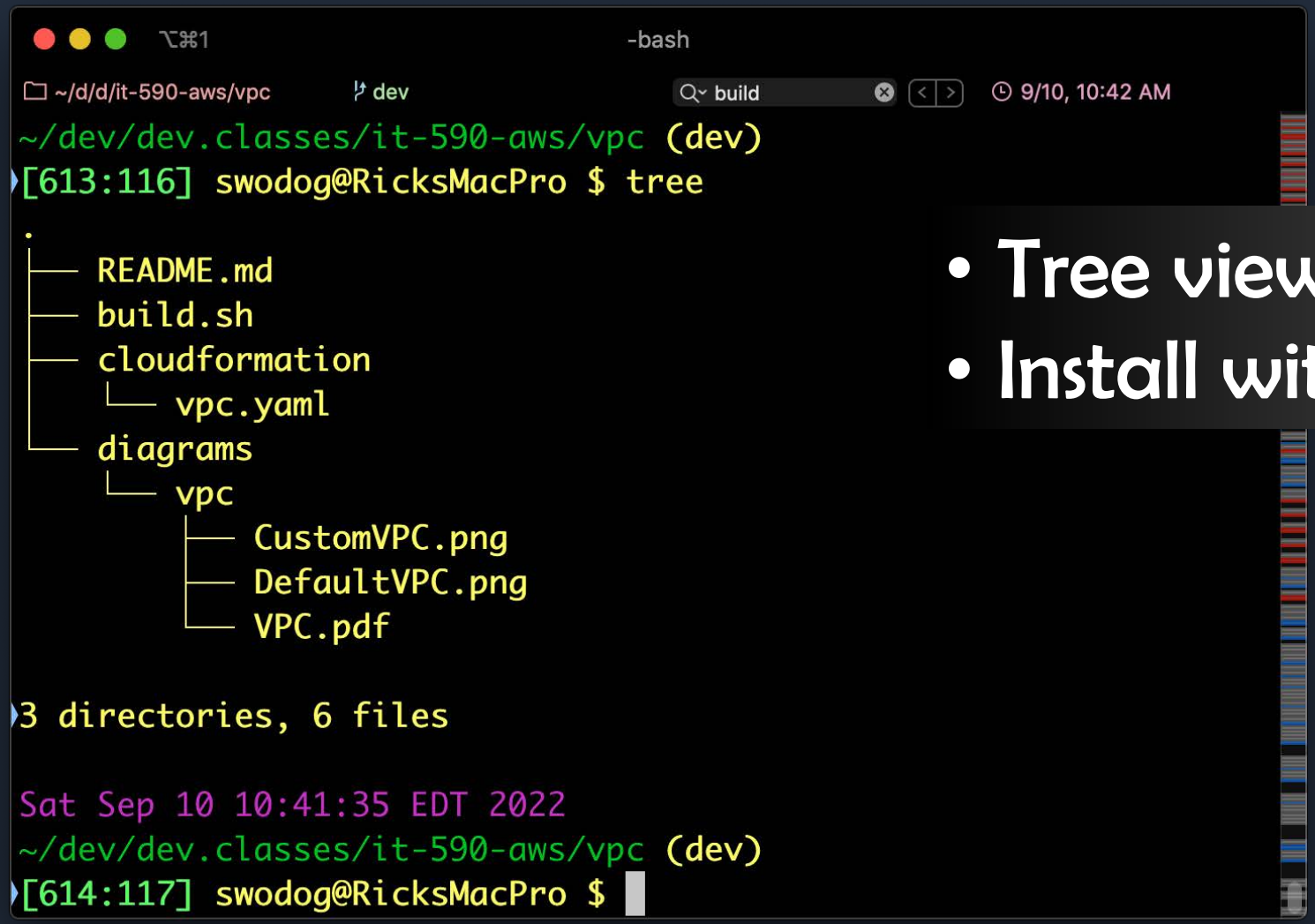
Usage

- pipenv -- python 3
- or be more specific
  - pipenv --python 3.6
  - pipenv --python 3.10
  - etc....
- Install a package
  - pipenv install requests
- This will create a Pipfile
- ...and a .venv directory


Oh Yeah?
Show me!

`tree` Command

tree Command (MacOS)



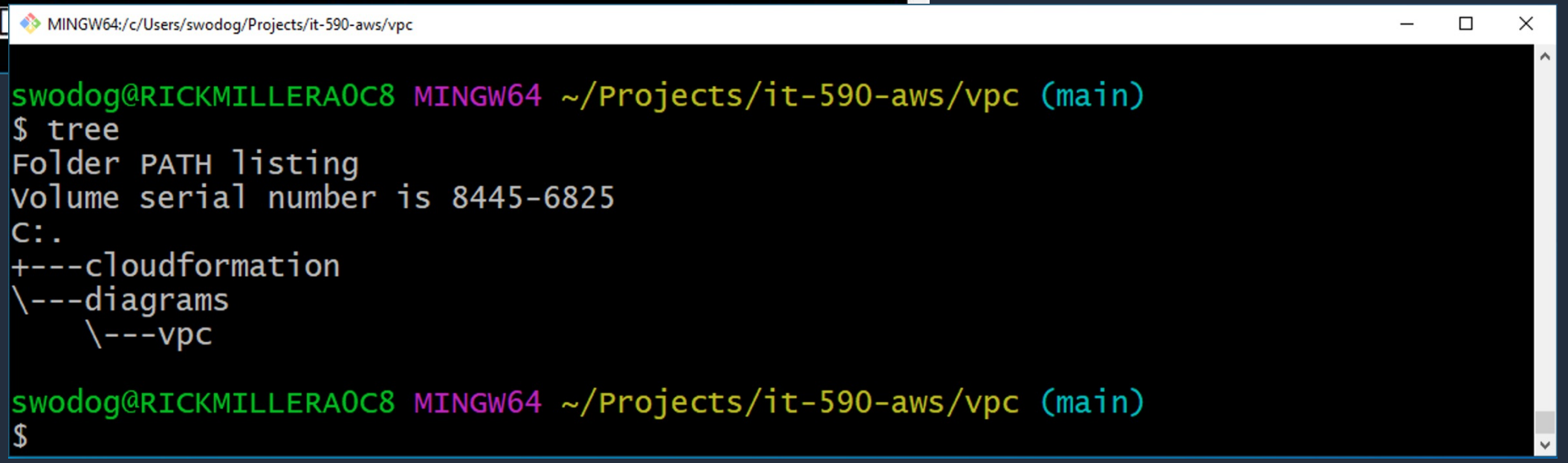
- Tree view of directory structure
- Install with brew

tree Command (Windows)

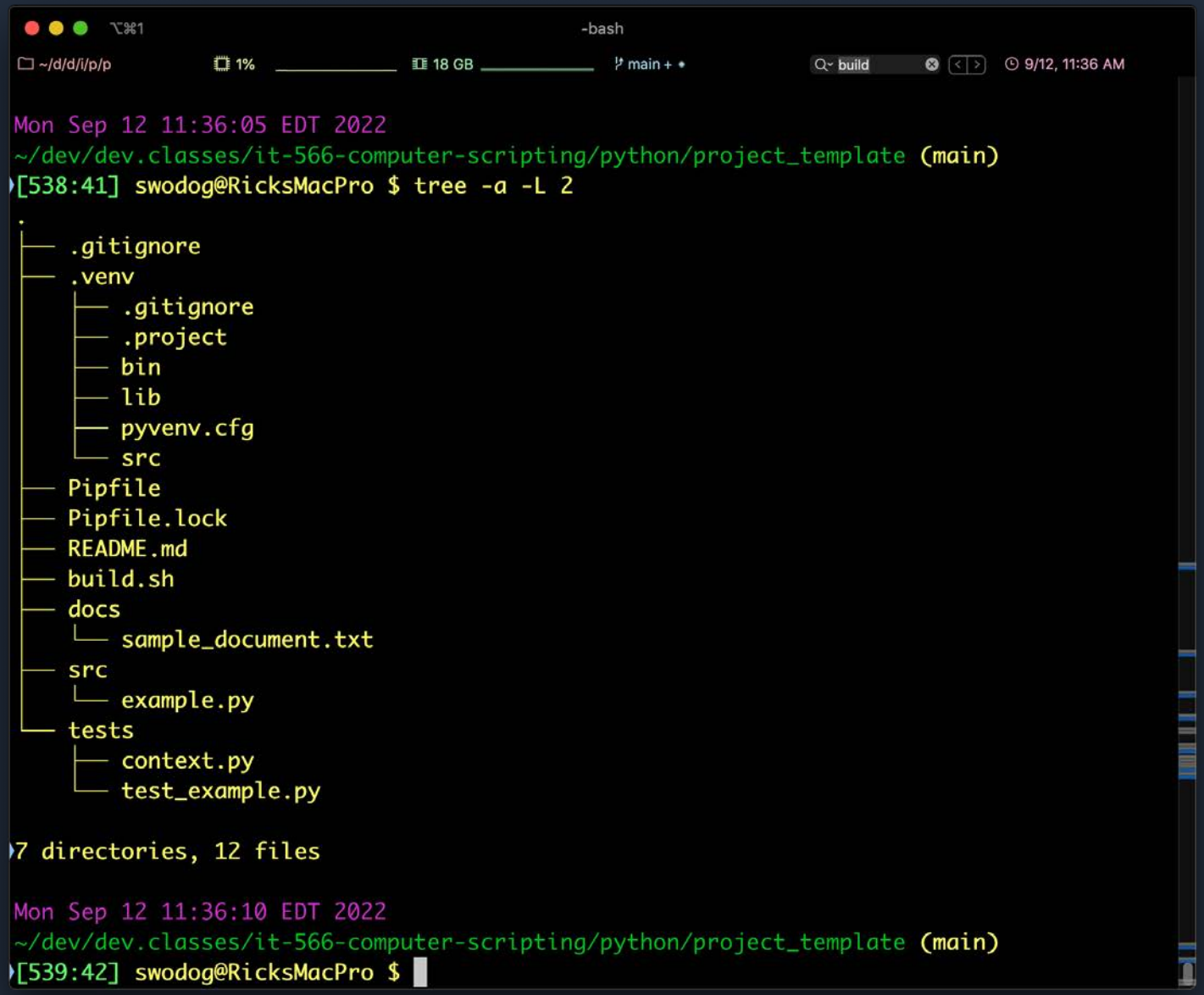
- Comes with Windows
  - Must configure to use in git bash

Add alias to tree command in .bash_profile

Relaunch Git Bash



tree Command (Show Hidden Files)



tree -a -L 2


Really?
Show me!

.bash_profile Helpful Settings

.bash_profile Helpful Settings

vi

1 .bash_profile

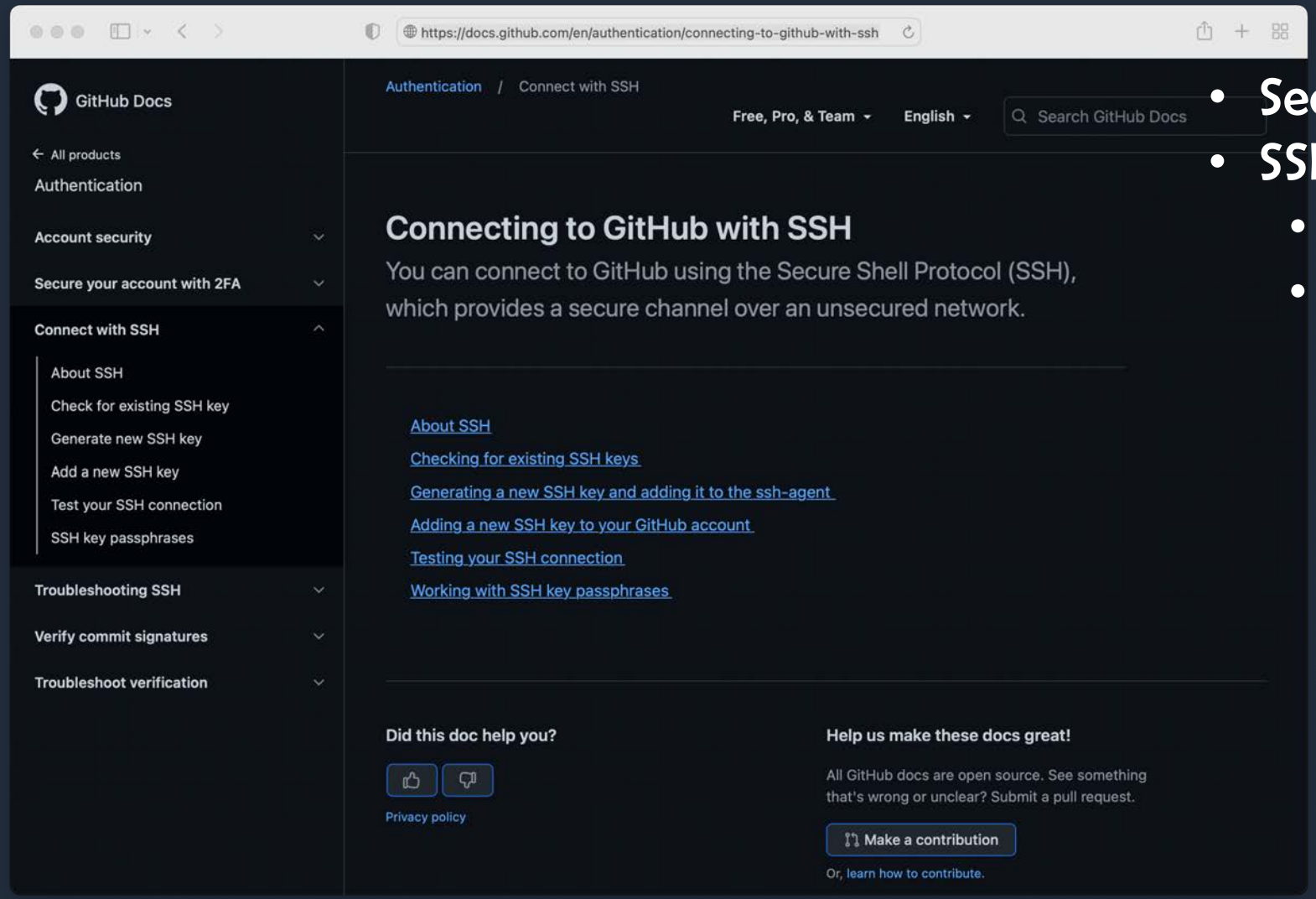
```
# aliases
alias dir='ls -al'
alias cls='clear'
alias first-vpc='cd ~/dev/dev.aws/projects/first-vpc'
alias it590='cd ~/dev/dev.classes/it-590-aws'
alias it566='cd ~/dev/dev.classes/it-566-computer-scripting'
alias gasson='cd ~/dev/dev.gasson/marketpredictor'

test -e "${HOME}/.iterm2_shell_integration.bash" && source "${HOME}/.iterm2_shell_integration.bash"

# prompt
PS1="\n\[\033[35m\]\$(/bin/date)\n\[\033[32m\]\w \[\033[1;33m\]\$(__git_ps1
'(%s)')\n\[$(iterm2_prompt_mark)\]\[\033[1;32m\][\!:\#]\[\033[1;33m\] \u@\h $ "

test -e "${HOME}/.iterm2_shell_integration.bash" && source "${HOME}/.iterm2_shell_integration.bash"

# setup AWS cli environment
. ~/bin/aws/set-aws.sh

export PIPENV_VENV_IN_PROJECT="true"
export PATH
eval "$(/opt/homebrew/bin/brew shellenv)"
```

NORMAL .bash_profile 87% 21:1

- Command Aliases
- Environment Variables
- Fancy Prompt
- Links to other shell scripts
- Much, much, more...

Configure SSH for GitHub

Configure SSH for GitHub



- Secure Repository Connection
- SSH Key Generation and Installation
  - Can be confusing
  - Let's walk through

GitHub SSH Key Generation and Configuration

Project Organization & Layout

Directory Structure & Project Artifacts

Python Modules & Packages


There's Bill, I'll ask him…
Hey Bill, me and Beth are having relationship issues. What do you think I should do?
Fix your Python project structure!

Project Organization

- Project Organization Refers to Several Issues
  - Directory Structure and Layout
  - Source Code Organization
    - Module and Package Organization
    - Application Architecture
- Today — Focus on Directory Structure and Project Artifacts
- Later Today — Focus on Application Architecture
  - …and how to organize source code into modules and packages

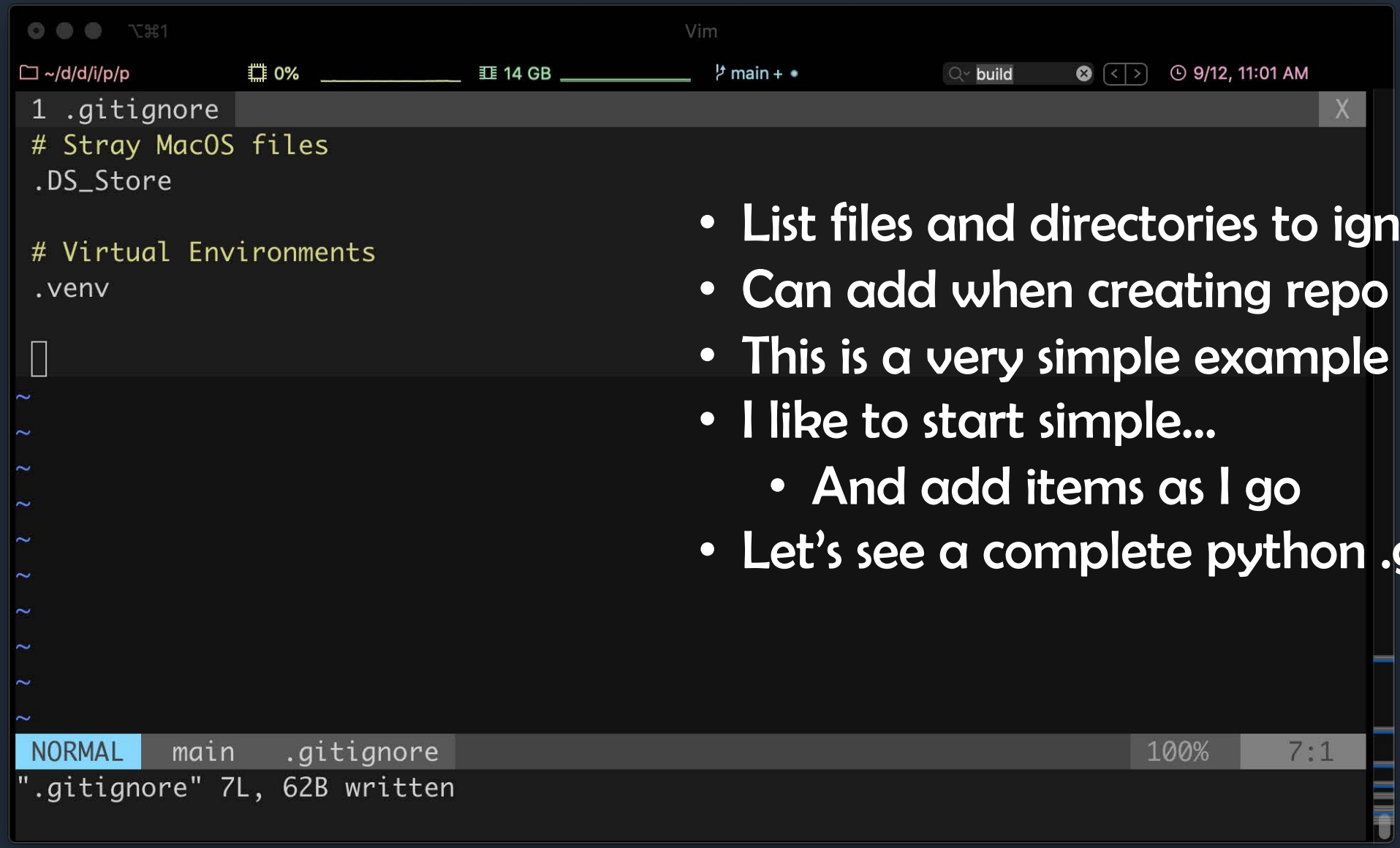
Directory Structure & Project Artifacts

Directory Structure and Project Artifacts

```
⌥⌘1                                -bash
~/d/d/i/p/p   1%   16 GB   main +   build   9/12, 10:23 AM

Mon Sep 12 10:14:04 EDT 2022
~/dev/dev.classes/it-566-computer-scripting/python/project_template (main)
[526:29] swodog@RicksMacPro $ tree -a -L 1
.
├── .gitignore
├── .venv
├── Pipfile
├── Pipfile.lock
├── README.md
├── build.sh
├── docs
├── src
└── tests

4 directories, 5 files

Mon Sep 12 10:14:22 EDT 2022
~/dev/dev.classes/it-566-computer-scripting/python/project_template (main)
[527:30] swodog@RicksMacPro $
```

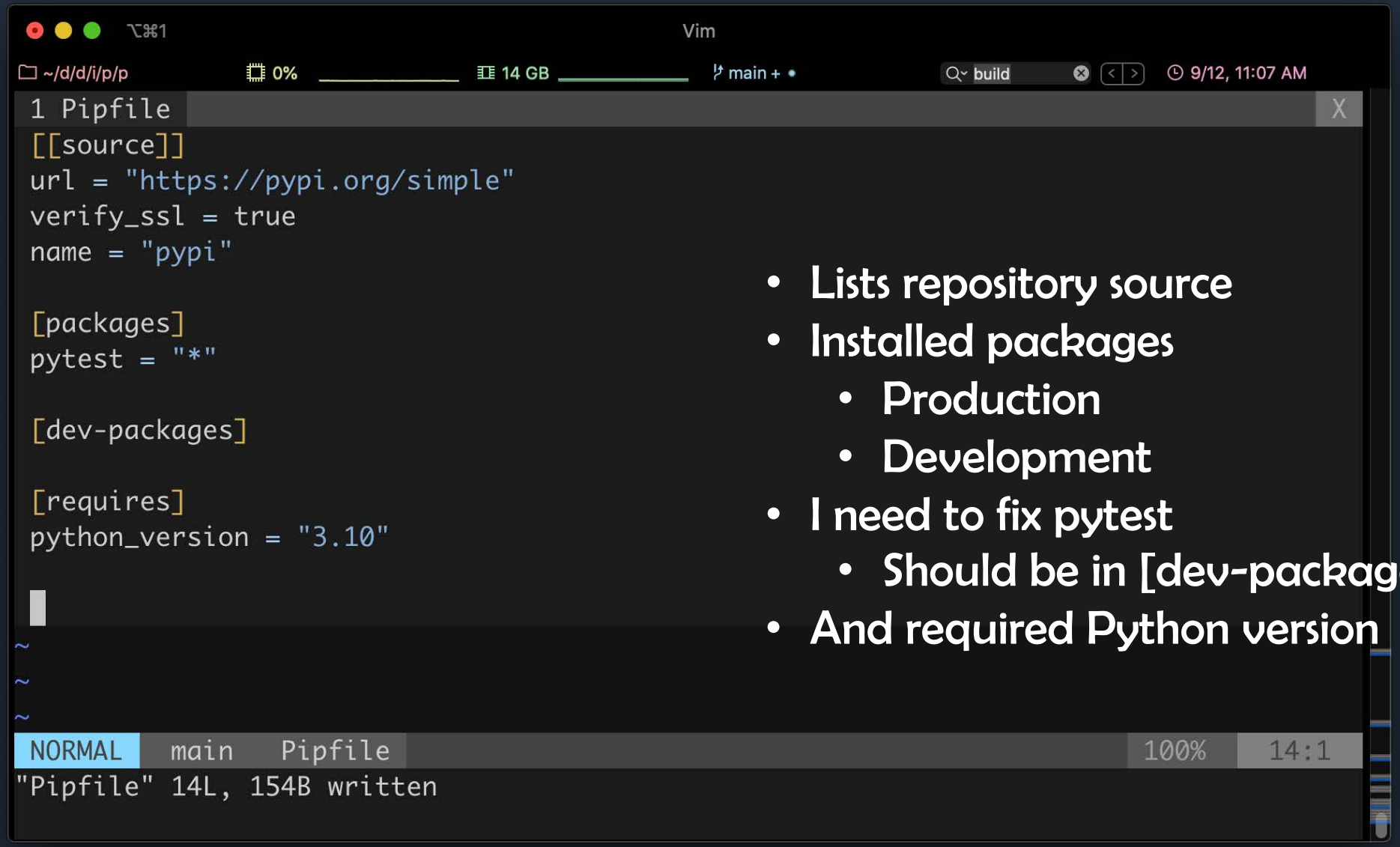
.gitignore — List of files not to add to repo
.venv — Virtual Environment created by pipenv
Pipfile — List of required project packages
Pipfile.lock — List of specific package versions
README.md — Markdown document displayed by GitHub
build.sh — Bash build file
docs — Documentation directory
src — Python source code directory
tests — Unit and integration tests directory

.gitignore



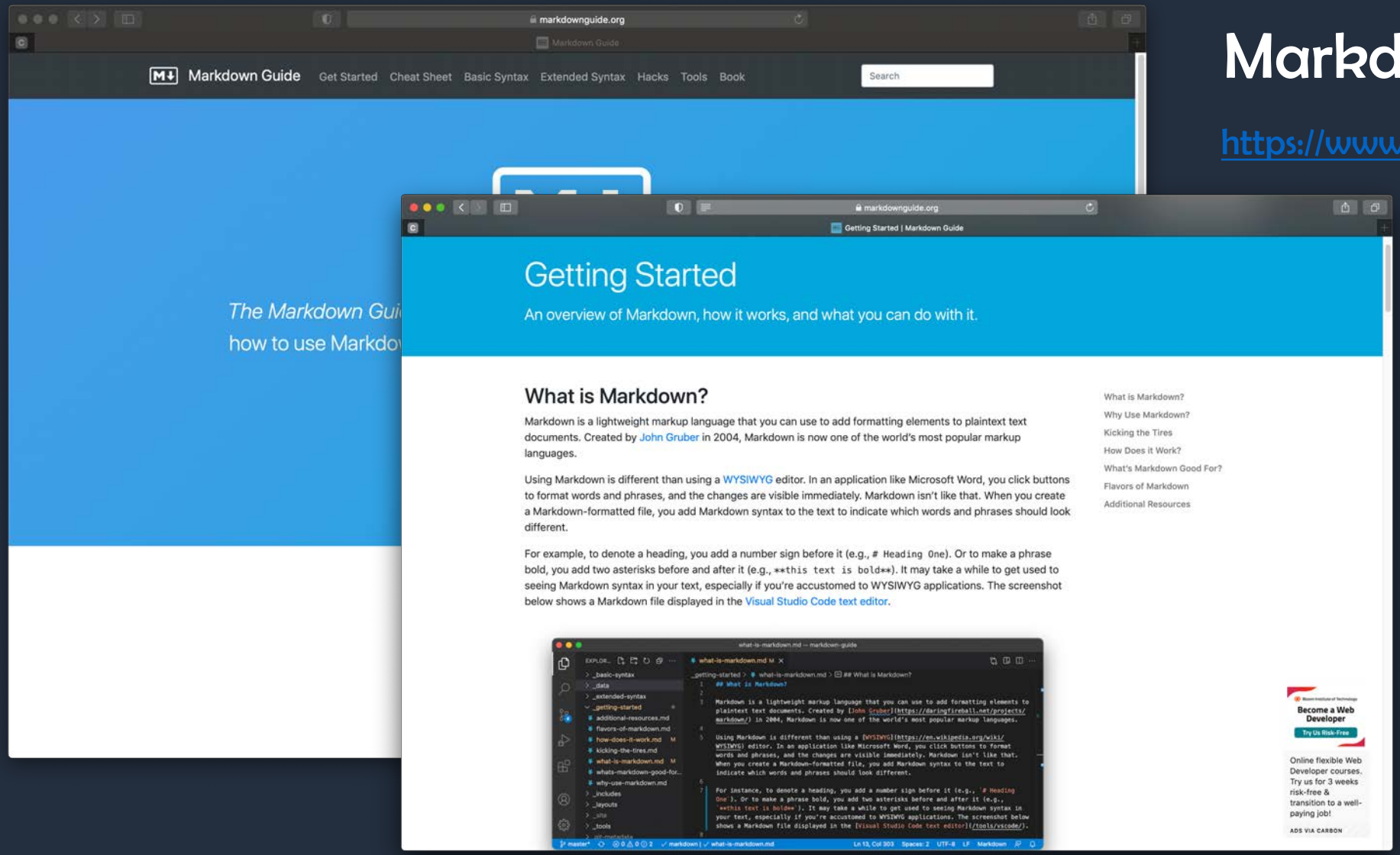
- List files and directories to ignore
- Can add when creating repo in GitHub
- This is a very simple example
- I like to start simple…
  - And add items as I go
- Let's see a complete python .gitignore

Pipfile



- Lists repository source
- Installed packages
  - Production
  - Development
- I need to fix pytest
  - Should be in [dev-packages] section
- And required Python version

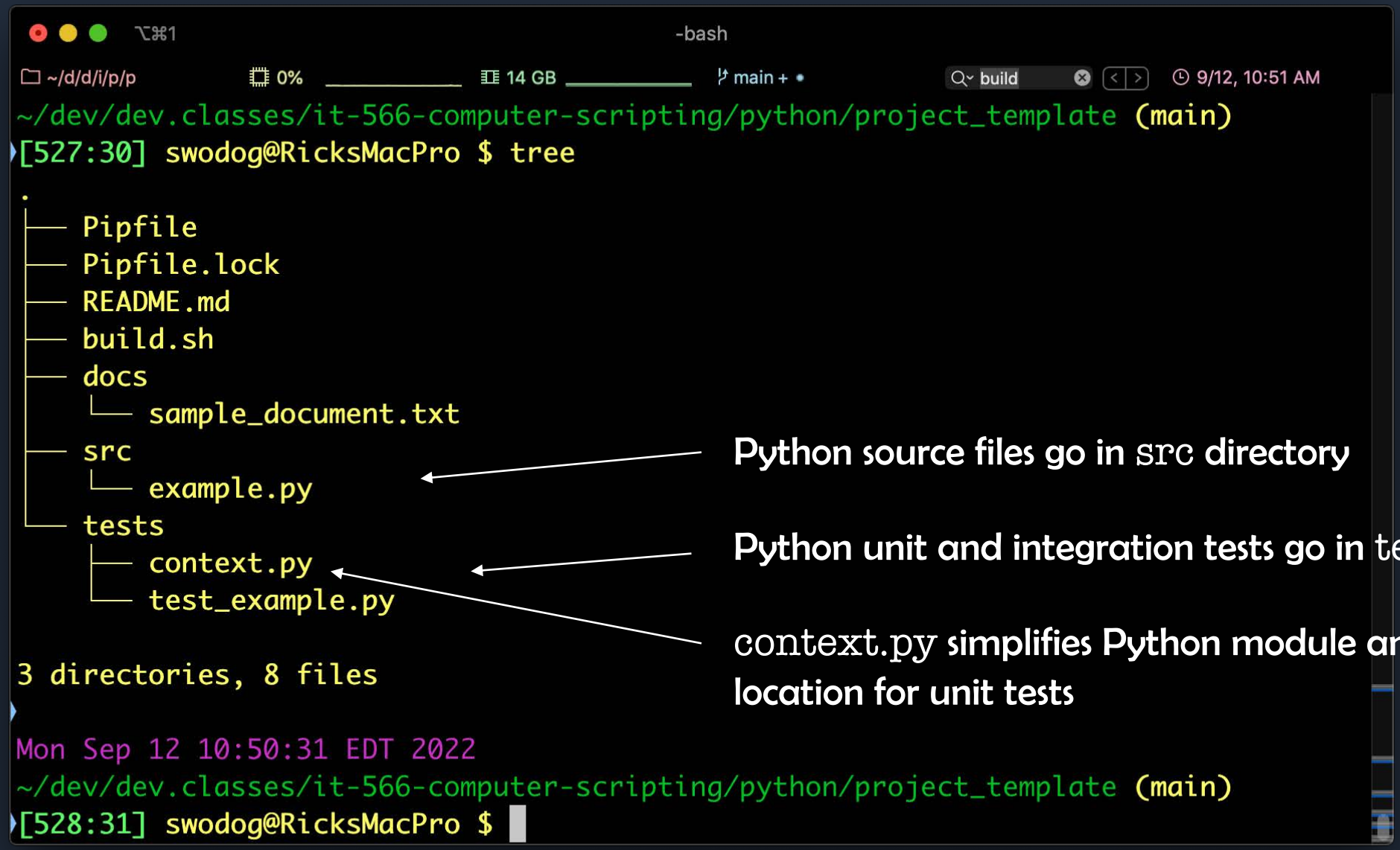
README.md



Markdown File

https://www.markdownguide.org

Directory Structure and Project Artifacts



Python source files go in `src` directory

Python unit and integration tests go in `tests` folder

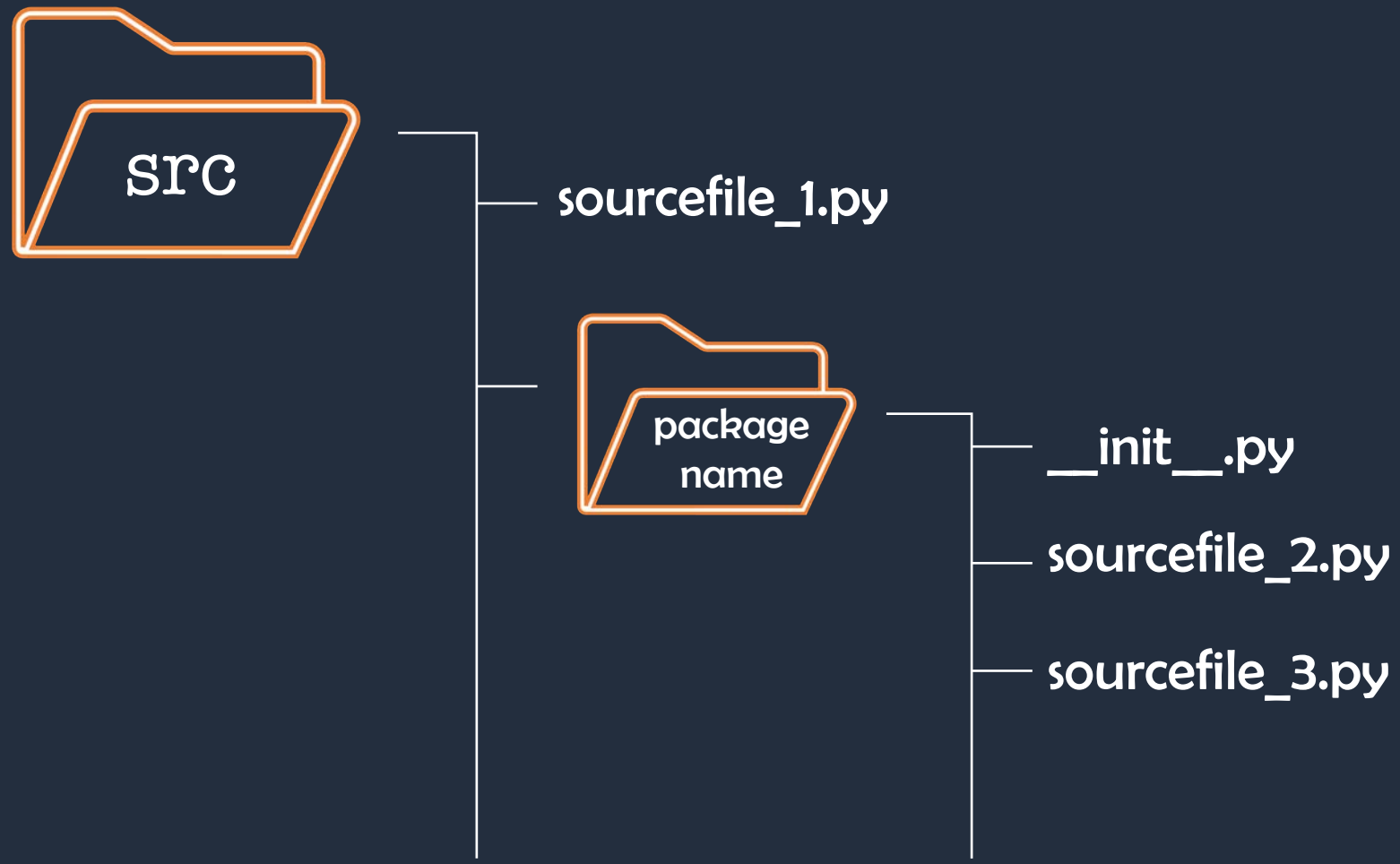
`context.py` simplifies Python module and package location for unit tests


Maybe they won't notice...
Hey, not so fast. What about the Pipfile.lock?

Python Modules and Packages

Python Modules and Packages

https://docs.python.org/3/reference/import.html#namespace-packages



Bash Build Script

Bash Build Script (build.sh)

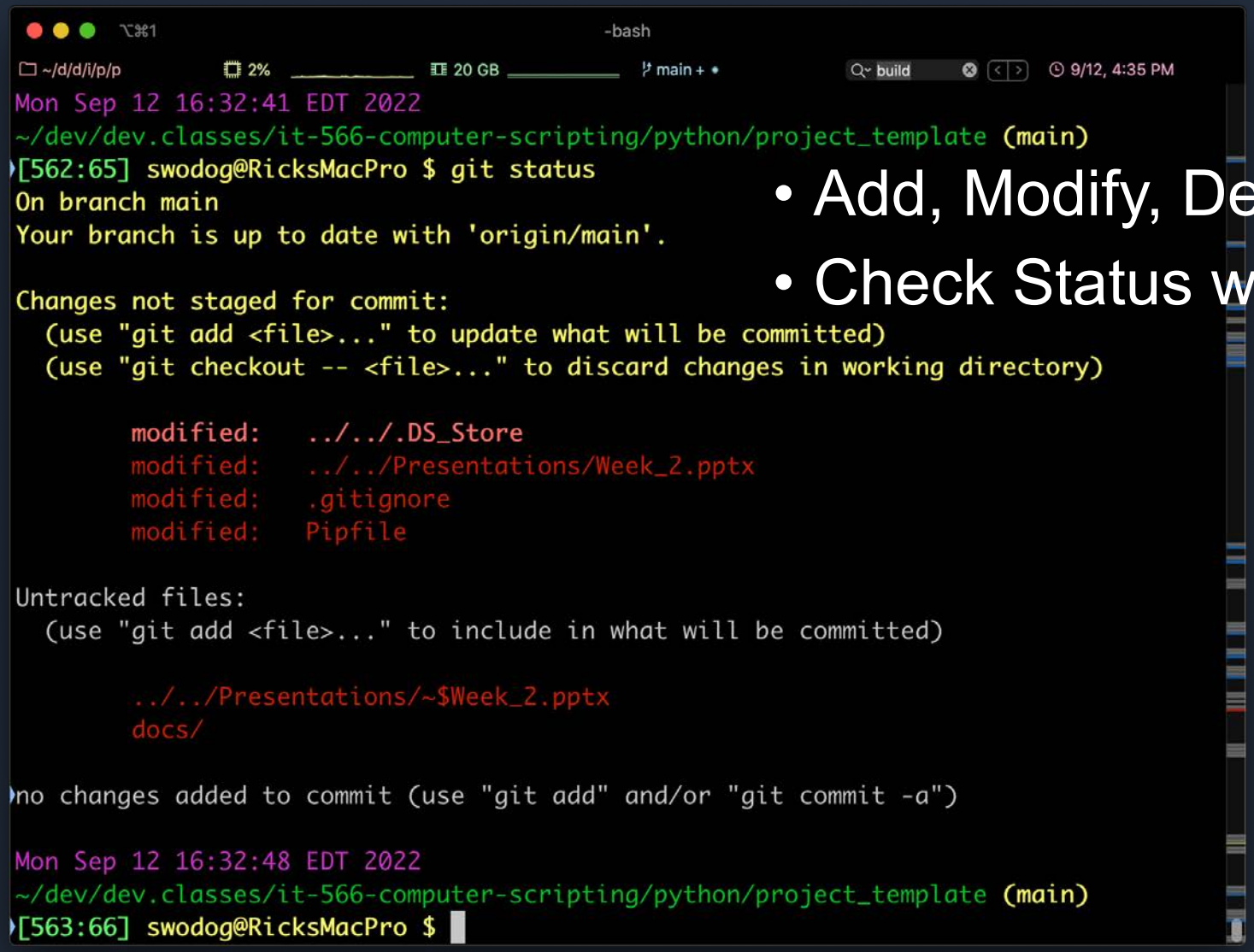
- Simplifies Project Management
- Consolidate Complex Commands
- Let’s take a look…

Git Development Workflow

Git Development Workflow

- Initial Activities
  - Create GitHub Repository
  - Add README.md and .gitignore
  - You will either start with a main or a master branch
- Clone The Repository Locally
  - git clone git@github.com:*repository_name.git*
  - Example: git clone git@github.com:pulpfreepress/it-566-computer-scripting.git

Git Development Workflow



- Add, Modify, Delete Files and Directories
- Check Status with git status

Followed by:

git add .
git commit –m “commit message”
git push


Would you like to see a demo?
Yes darling!