Version Control and Git

Kevin Bonham, PhD

2 Oct, 2018

Learning Objectives

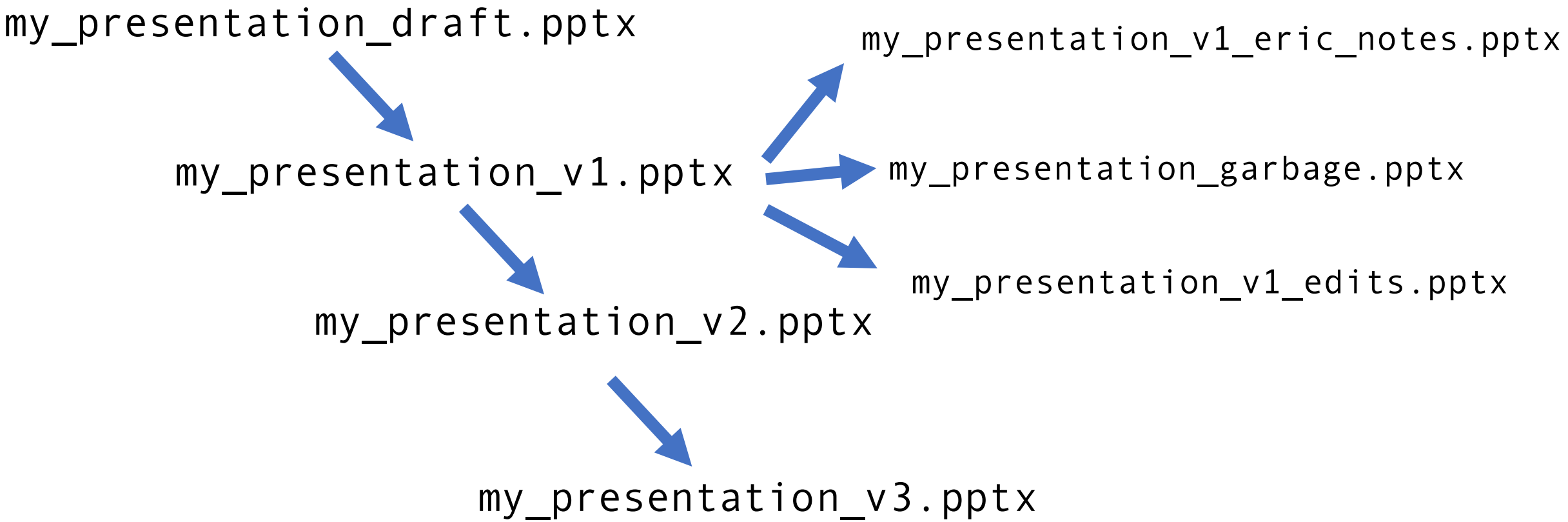
After completing this lesson, you will be able to:

- Recognize git jargon (repo, commit, stage, push, pull, branch etc.)
- Initialize and commit to a local git repository
- Use git branches to make changes to working code without losing the current state
- Clone a remote repository, make a change, and then start a pull request.

But first!

Version Control

Version control is a system for tracking the state of files and / or folders



What if you have multiple files?

```
my_app/
    ├ run.py ----------
    |
    |
    |
    ├ my_functions.py  -
    |
    |
    ├ my_variables.py  -
```

```
from my_functions import print_name
from my_variables import my_name

print_name(my_name)
```

```
def print_name(a_name):
    print(“My name is”, a_name)
```

```
my_name = "Kevin"
```

What if you have multiple files?

```
my_app/
    ├ run.py ----------
    |
    |
    |
    ├ my_functions.py  -
    |
    |
    ├ my_variables.py  -
```

```
from my_functions import print_name
from my_variables import user_name

print_name(user_name)
```

```
def print_name(a_name):
    print("Your name is", a_name)
```

```
user_name = input("What is your name? ")
```

What if you have multiple files?

```
my_app/
    ├ run_v2.py ----------
    |
    |
    |
    ├ my_functions_v2.py -
    |
    |
    ├ my_variables_v2.py -
```

```
from my_functions_v2 import print_name
from my_variables_v2 import user_name

print_name(user_name)
```

```
def print_name(a_name):
    print(“Your name is”, a_name)
```

```
user_name = input("What is your name? ")
```

What if you're collaborating?

```
my_app/
    ├ run.py ----------
    |
    |
    |
    ├ my_functions.py  -
    |
    |
    ├ my_variables.py  -
```

```
from my_functions import print_name
from my_variables import my_name

print_name(my_name)
```

```
def print_name(a_name):
    print("My name is", a_name)
```

```
my_name = "Eric"
```

Why not use web drive like gdrive or dropbox?

- Changes to files are often co-dependent
  - simultaneous coding is… problematic
- Explicit versioning
  - Changes can “break” code in ways text can’t be broken
- Ability to take multiple development paths

Version Control Systems (VCS)

A version control system:

- **Explicitly** tracks the state of files and folders
- Keeps a record of changes to files and folders
  - Without requiring name changes
- Enables moving forward and backward in time

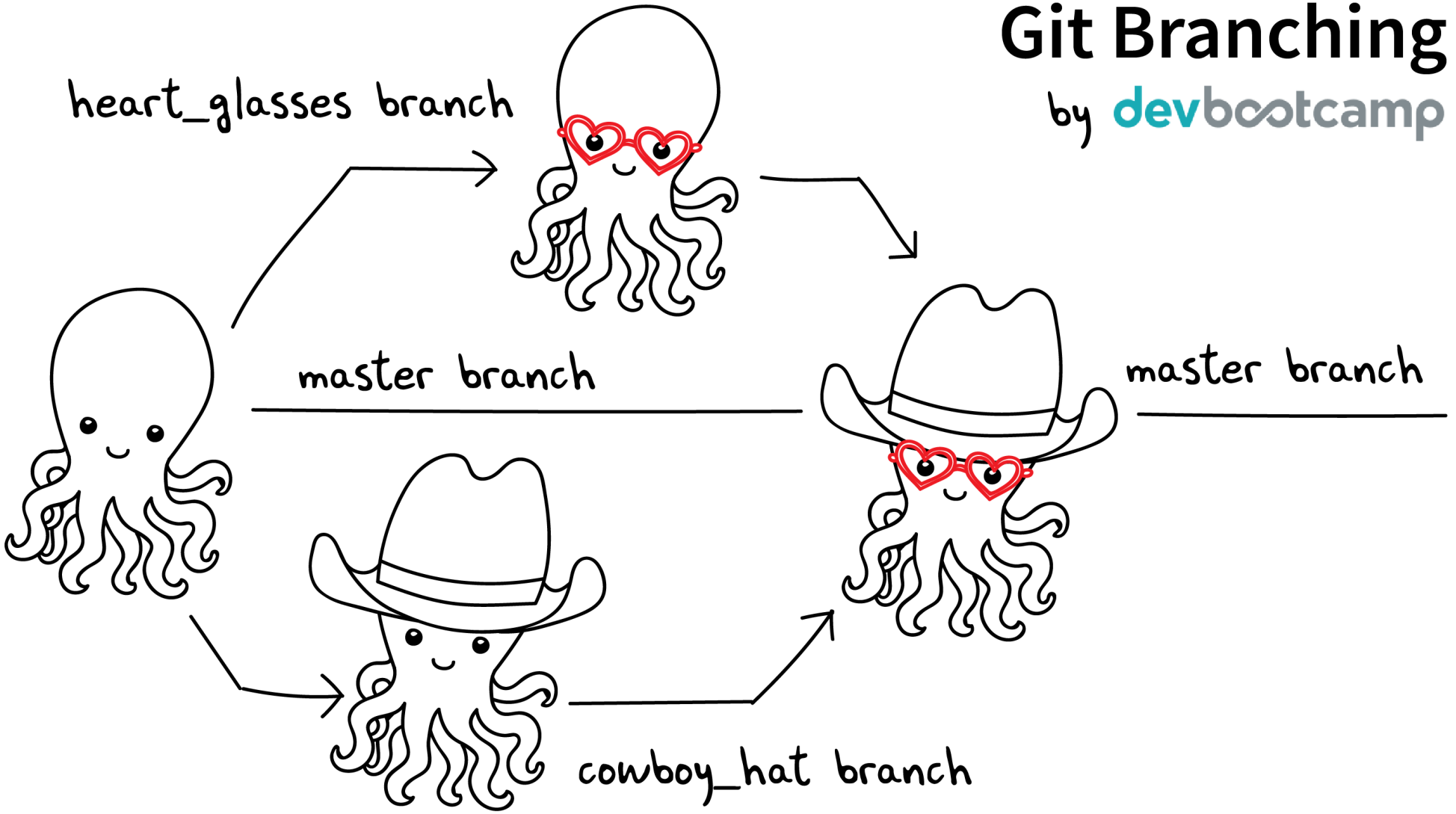
git is a Distributed Version Control System (DVCS)

Git has a lot of jargon

- **Repository (repo):** a git-enabled folder
  - Sometimes refers to all locations (remotes and locals) of a particular fork
- **Commit**: an snapshot of code
- **Branch:** a particular history of commits (multiple branches can exist in the same repo)
- **Merge:** When two branches are joined together and any conflicts are dealt with

Git Branching

by devbootcamp



git ≠ github

- Github (and gitlab and bitbucket) is a place for hosting a "remote" repository
  - Also adds collaboration features
- git is powerful even if used only on your local machine
  - But… get free private repos on github with your .edu address (or use gitlab / bitbucket)

Install git

- **Mac:**
  - https://brew.sh/
  - Copy install code (on the webpage)
    - `/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)”`
  - `$ brew install git`

- **Windows:**
  - https://gitforwindows.org/

Clone the exercise repo

```
$ git clone https://github.com/kescobo/bst273_lecture09.git

  Cloning into 'bst273_lecture09'...
  remote: Enumerating objects: 11, done.remote: Counting objects:
  100% (11/11), done.
  remote: Compressing objects: 100% (9/9), done.
  remote: Total 11 (delta 1), reused 11 (delta 1), pack-reused 0
  Unpacking objects: 100% (11/11), done.
```