# Version control using Git

https://github.com/nih-fmrif/git-training

## Session 1

12/02/2019

Data Science and Sharing Team
NIMH

# The problem...

Scenario 1 - Managing your own code

Day 1: You write, troubleshoot, and successfully verify that step 1 of the analysis works as expected. File saved. You go home.

Day 2: You clean up step 1 a bit  and then get started working on step 2. At the end of the day, file saved. You go home.

A few days later: You run your code and SURPRISE, it fails. You find the location of the error, but you swore that you solved this issue on day 1. Maybe something happened on day 2 that broke the code from day 1?

You wish you had  a time machine…

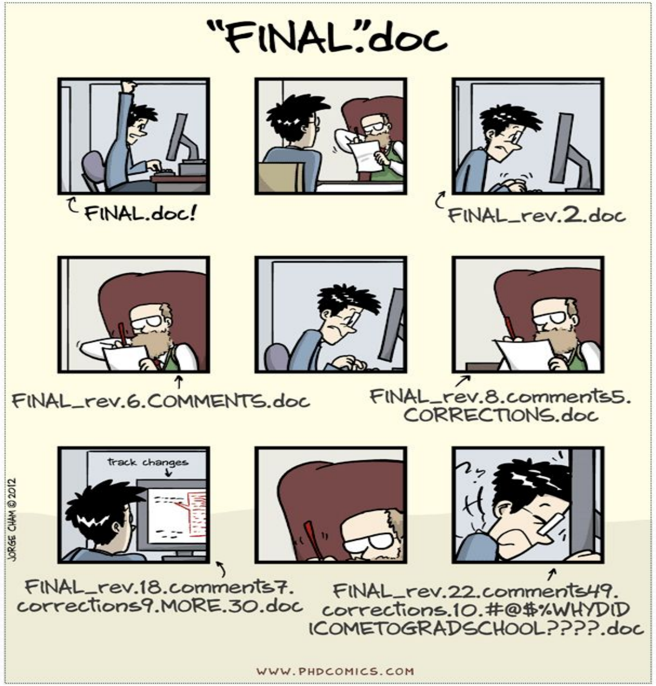
# The problem...

Scenario 2 - Collaboration with 5 other people...

How do you share your changes with others?

How do you manage comments and different versions of a document / code / etc…?

How do you name files??



# The solution… version control!

- Record a file's history
- Restore to previous versions
- Effectively an unlimited ‘undo’
- Collaborate with others in parallel
- Side-effect: backup!

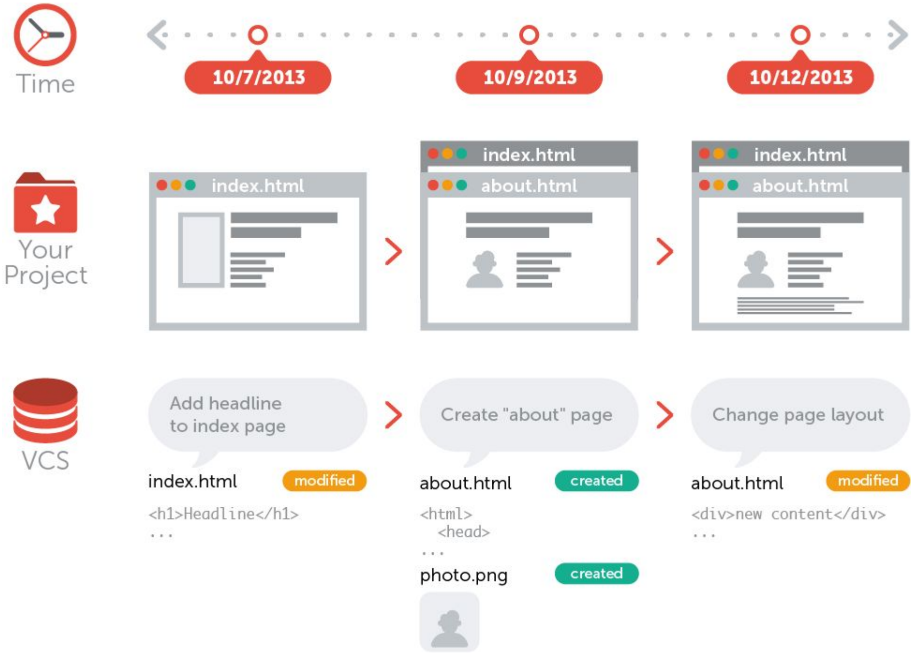


Image from: git-tower.com

# What is Git?

- Git is a free and open source distributed version control system to handle everything from small to very large projects with speed and efficiency
  - https://git-scm.com/

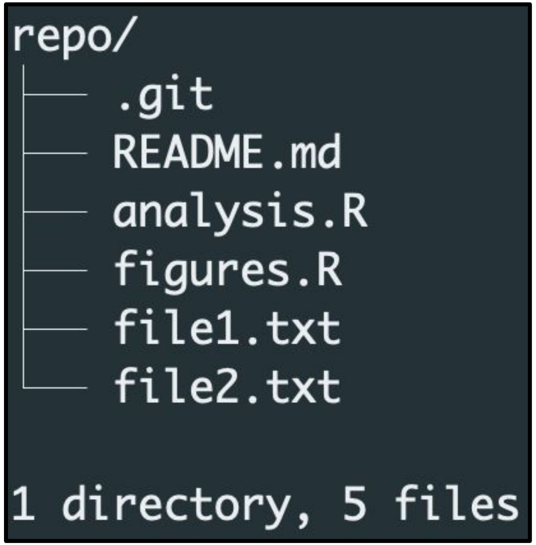
- Git services include:
  - Github -- https://github.com/
  - Bitbucket -- https://bitbucket.org/
  - Gitlab -- https://about.gitlab.com/

Other version control systems include:
Mercurial, Bazaar, Subversion, & CVS

# What is a repository?

- A kind of database that stores all snapshots of your project
- Stored in a magical folder (.git) in your project’s root directory
- Other than magic, what is contained within .git?
    - History of snapshots
        - Objects (eg, files)
        - Trees (eg, directories)
        - Metadata
            - What changes were made
    - Metadata (eg, configuration info)
    - Current changes in preparation for the next snapshot

- HEAD - current branch



```
repo/
├── .git
├── README.md
├── analysis.R
├── figures.R
├── file1.txt
└── file2.txt

1 directory, 5 files
```

# How does one repository?

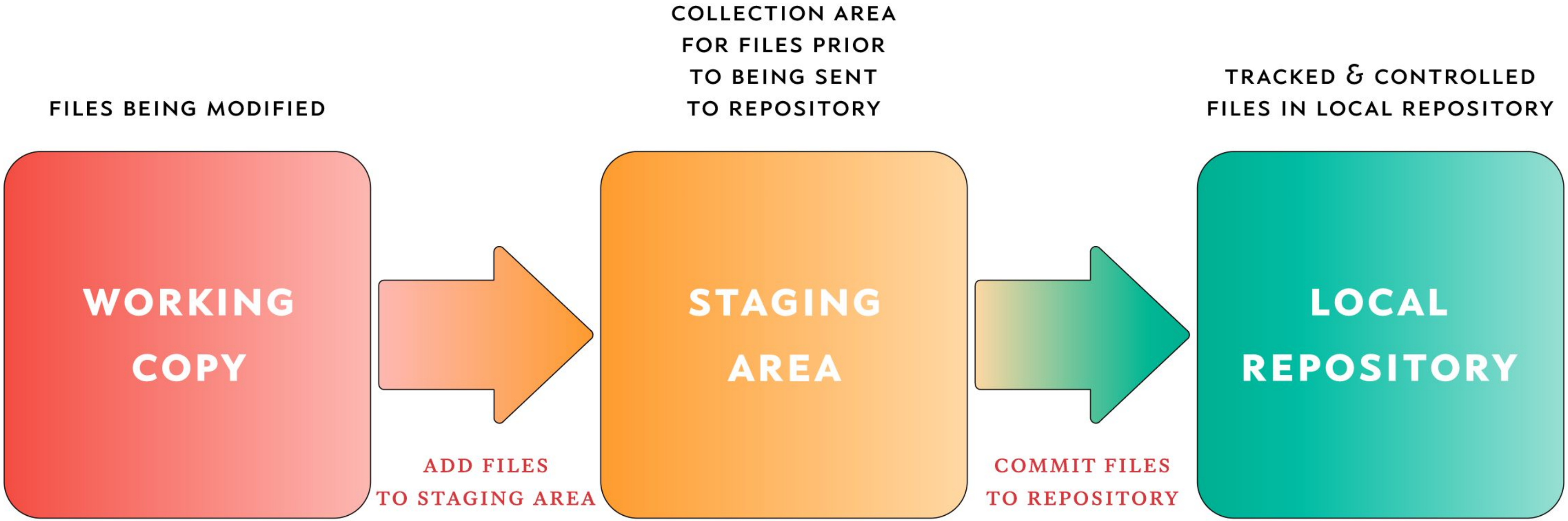

FILES BEING MODIFIED
COLLECTION AREA FOR FILES PRIOR TO BEING SENT TO REPOSITORY
TRACKED & CONTROLLED FILES IN LOCAL REPOSITORY
WORKING COPY
STAGING AREA
LOCAL REPOSITORY
ADD FILES TO STAGING AREA
COMMIT FILES TO REPOSITORY

Image from: [http://practicalseries.com/1002-vcs/02-02-concept.html](http://practicalseries.com/1002-vcs/02-02-concept.html)

# How does one repository?

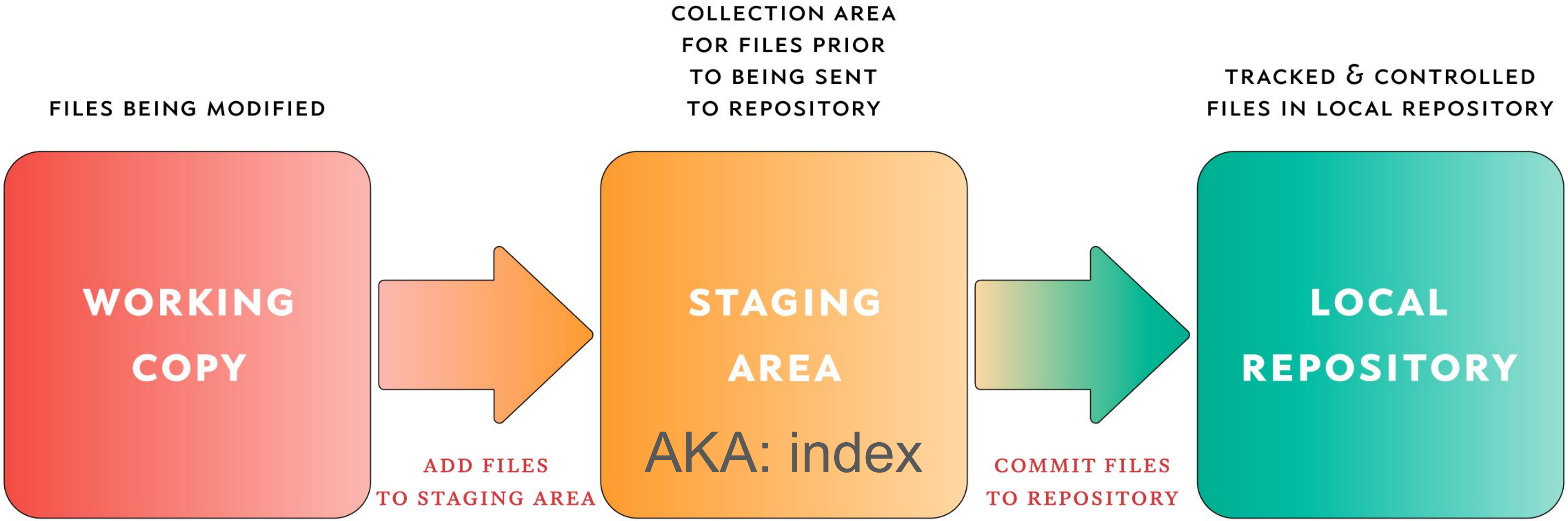

FILES BEING MODIFIED
WORKING COPY
ADD FILES TO STAGING AREA
COLLECTION AREA FOR FILES PRIOR TO BEING SENT TO REPOSITORY
STAGING AREA
AKA: index
COMMIT FILES TO REPOSITORY
TRACKED & CONTROLLED FILES IN LOCAL REPOSITORY
LOCAL REPOSITORY

Image from: http://practicalseries.com/1002-vcs/02-02-concept.html

# How does one repository?

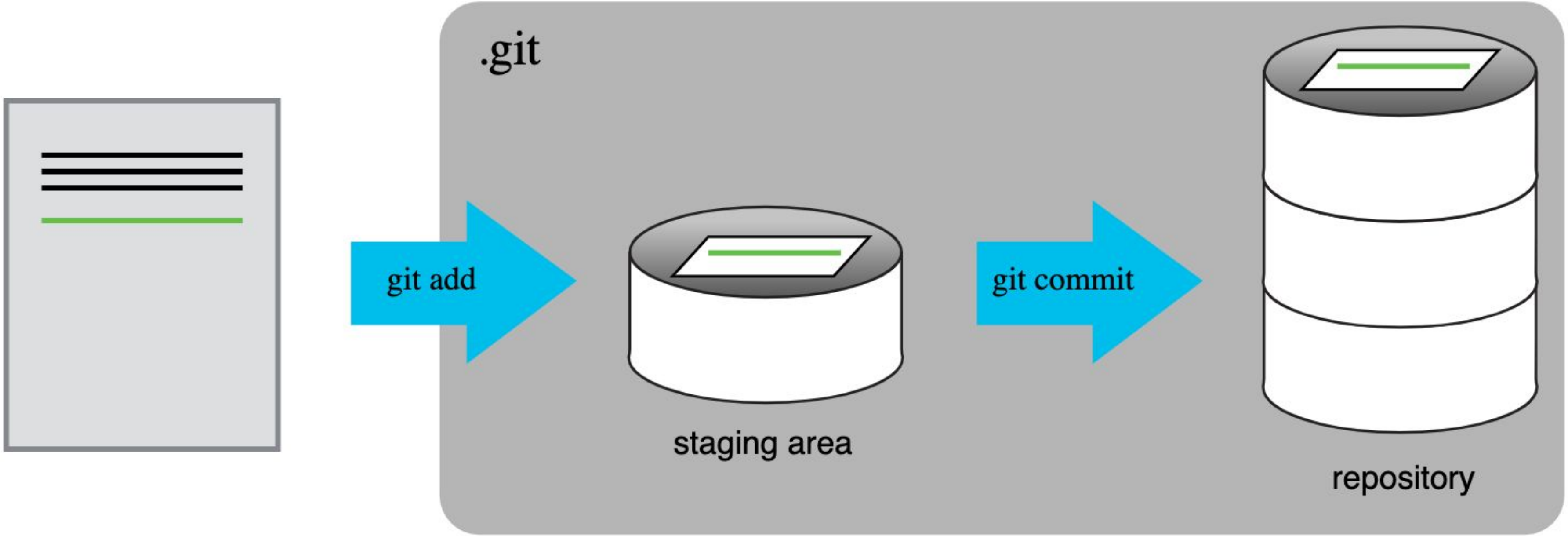
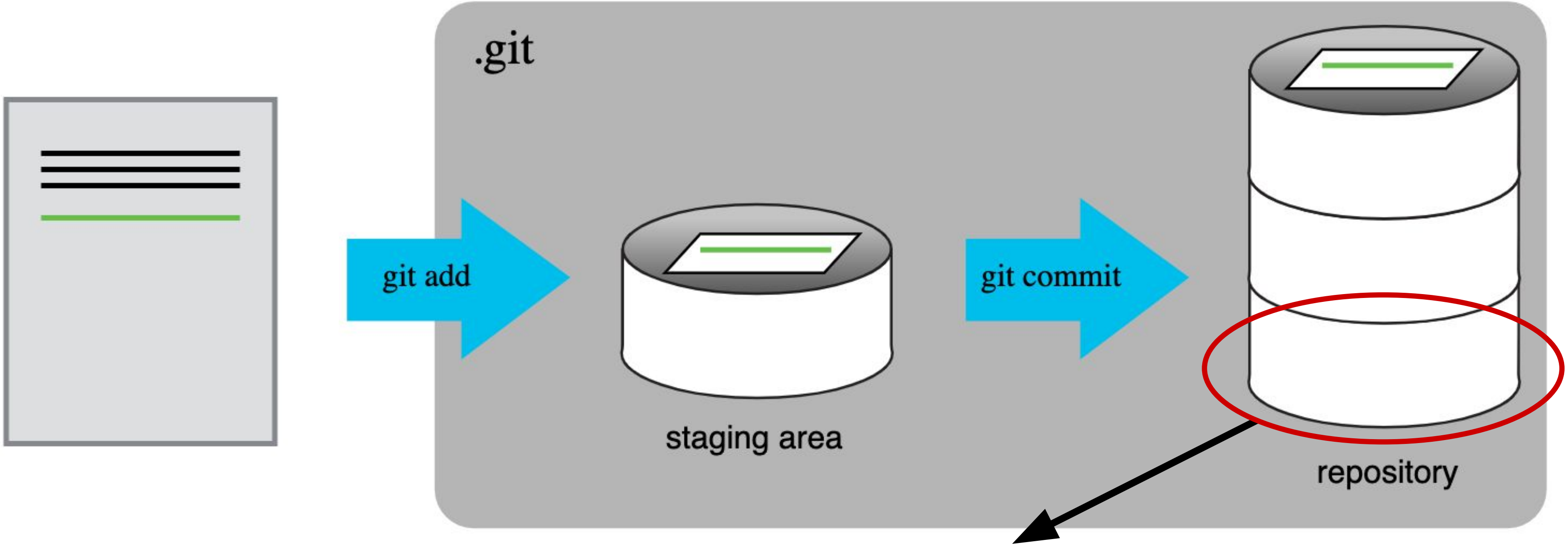


Image from: http://swcarpentry.github.io/git-novice/04-changes/index.html

# How does one repository?

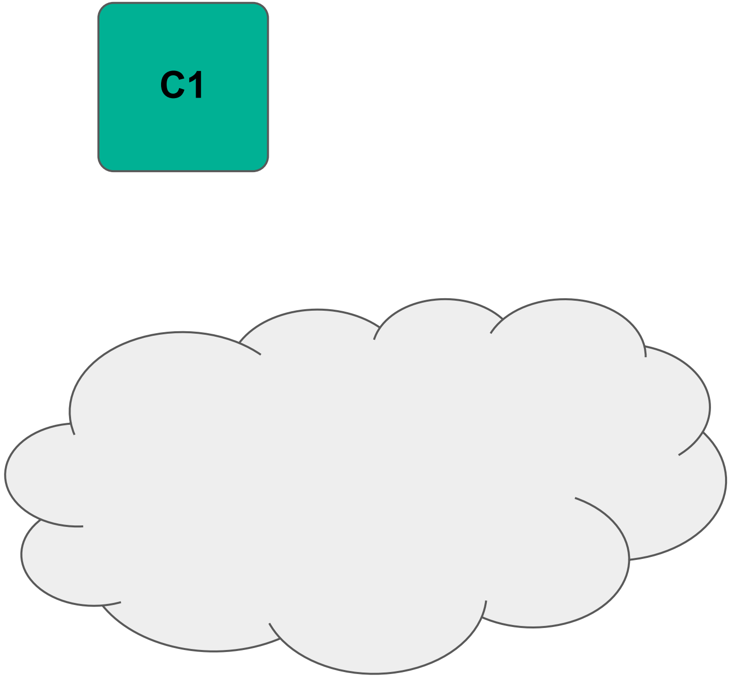


**Each snapshot is typically referred to as a “commit”**

Image from: http://swcarpentry.github.io/git-novice/04-changes/index.html

# Where does a repository live?

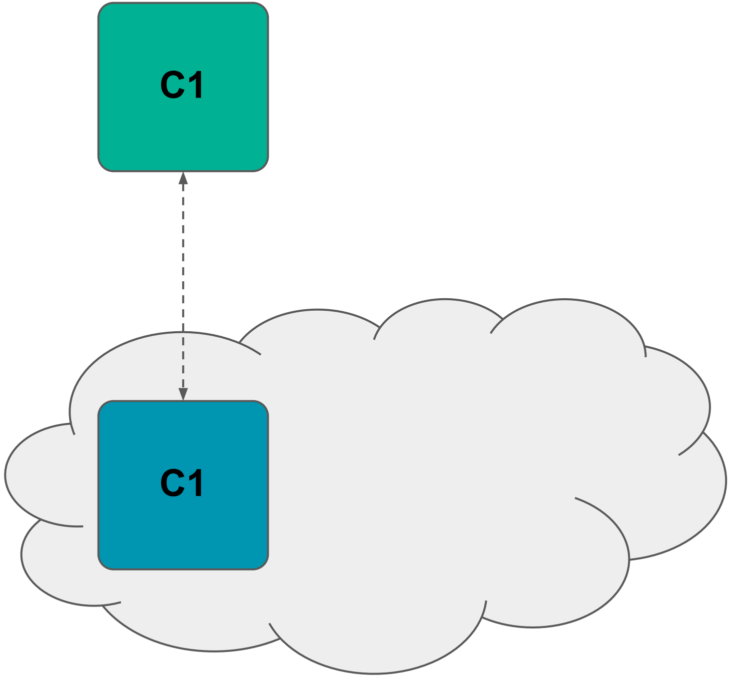
Local
(eg, your computer)



C1

Remote
(eg, GitHub)

# Where does a repository live?



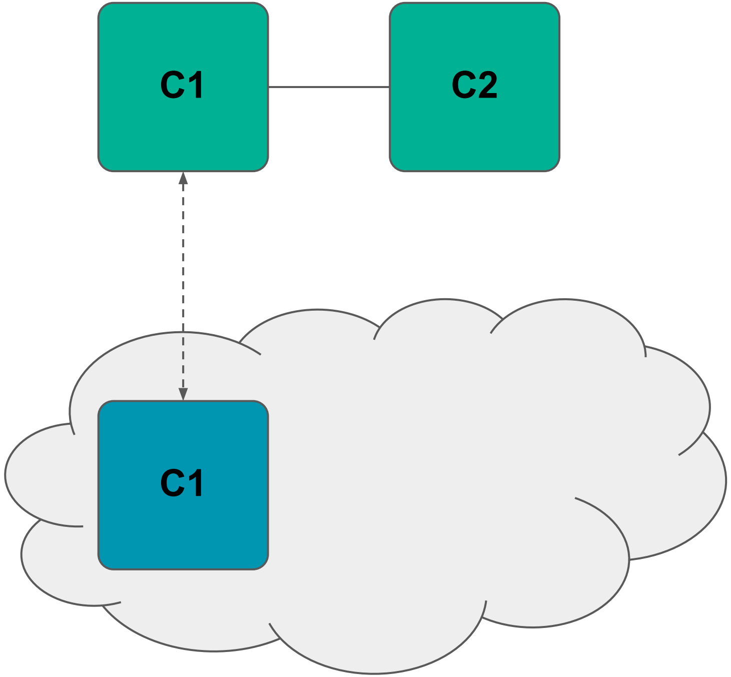
Local
(eg, your computer)

Remote
(eg, GitHub)

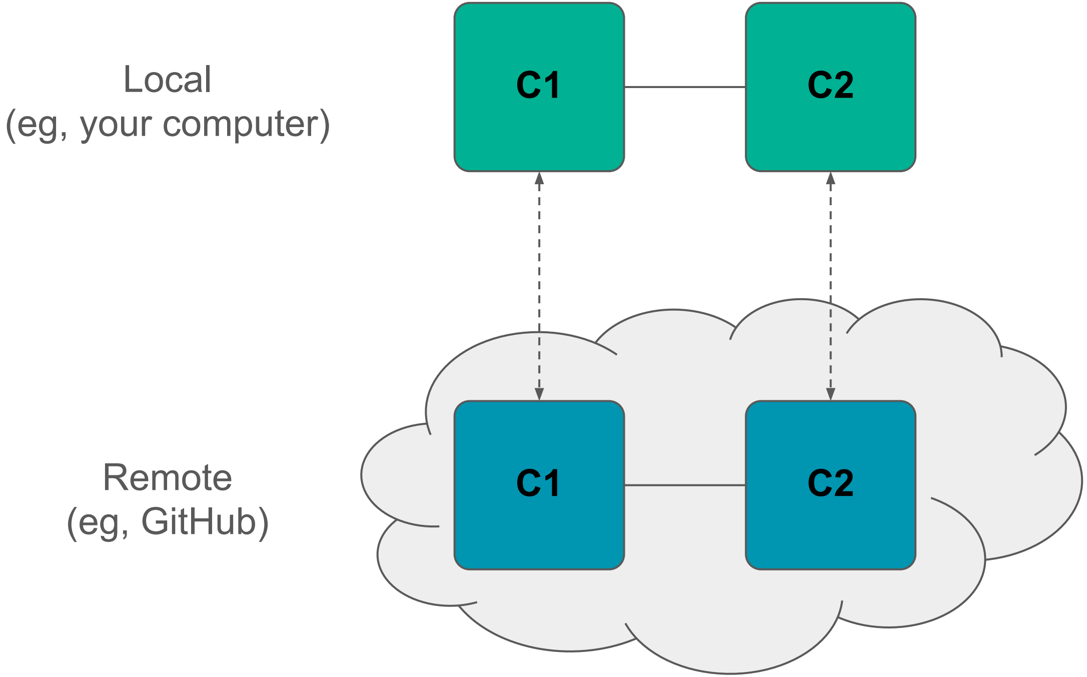
# Where does a repository live?

Local
(eg, your computer)

Remote
(eg, GitHub)



# Where does a repository live?



# Git commands

FINE! I NEED TO CLONE THE BRANCHES IN YOUR GIT REPO TO REVIEW YOUR CODE.
?
AAAARGH!!
GEEKHERO 2008 - SALVATORE IOVENE
35

```
git <verb> [options]
```

Image from: http://www.geekherocomic.com/2008/08/15/git-clone/index.html

# Git commands: init

Create a new local repository

- Execute within the project’s root directory
- Creates a .git directory

```
git init
```

# Git commands: status

Check the status of your working tree

- Shows you files that you have changed relative to your last commit

```
git status
```

# Git commands: add

## Add file(s) to the staging area

- Save changes before you add
- Can add multiple files
- Refrain from wildcards (*)

```
git add <file(s)>
```

# Git commands: commit

Commit changes in staging area to local repository

- Takes a snapshot and assigns a commit ID
- Make sure to add an informative message

```
git commit -m “message”
```

# Git commands: diff

See the differences between your working tree and previous commit

- Can also see differences relative to:
    - Previous commit
    - Staging area

```
git diff [options]
```

# Git commands: remote

Manage remote repositories

- Use git commands on a remote repo
- Use `git config` to configure user name, git client, etc...

```
git remote <verb> [options]
```

# Git commands: push

Push local changes to remote repository

- Origin - remote repo from clone
- Master - master branch name

```
git push <remote> <branch>
```

# Git commands: log

See the history of changes to a repository

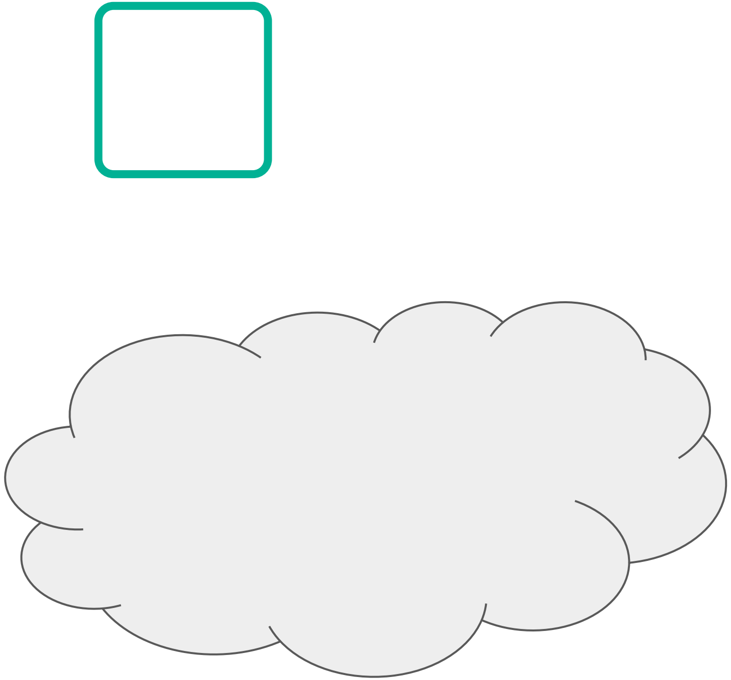
- Date
- Commit ID and message
- Who made the change

```
git log
```

# Where does a repository live?

Local
(eg, your computer)

Remote
(eg, GitHub)

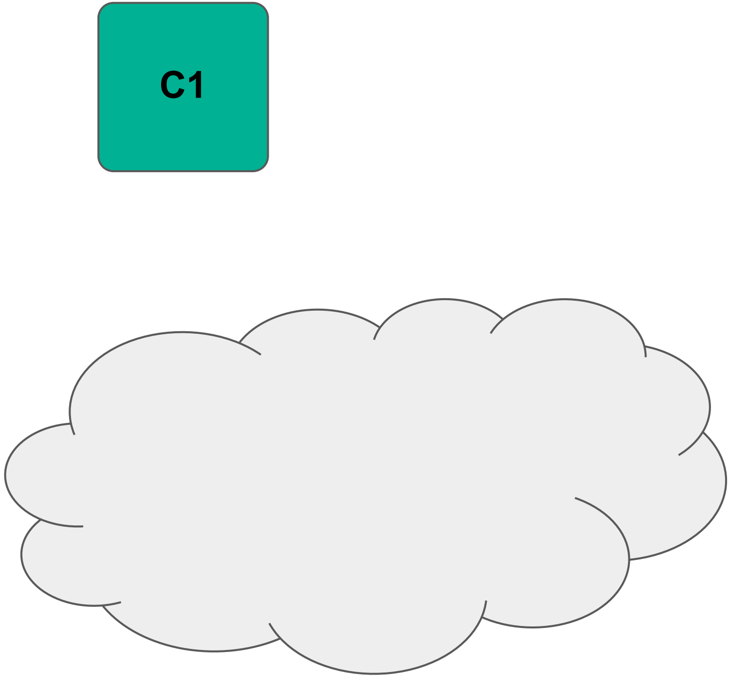


Git command

```
git init
```

# Where does a repository live?

Local
(eg, your computer)



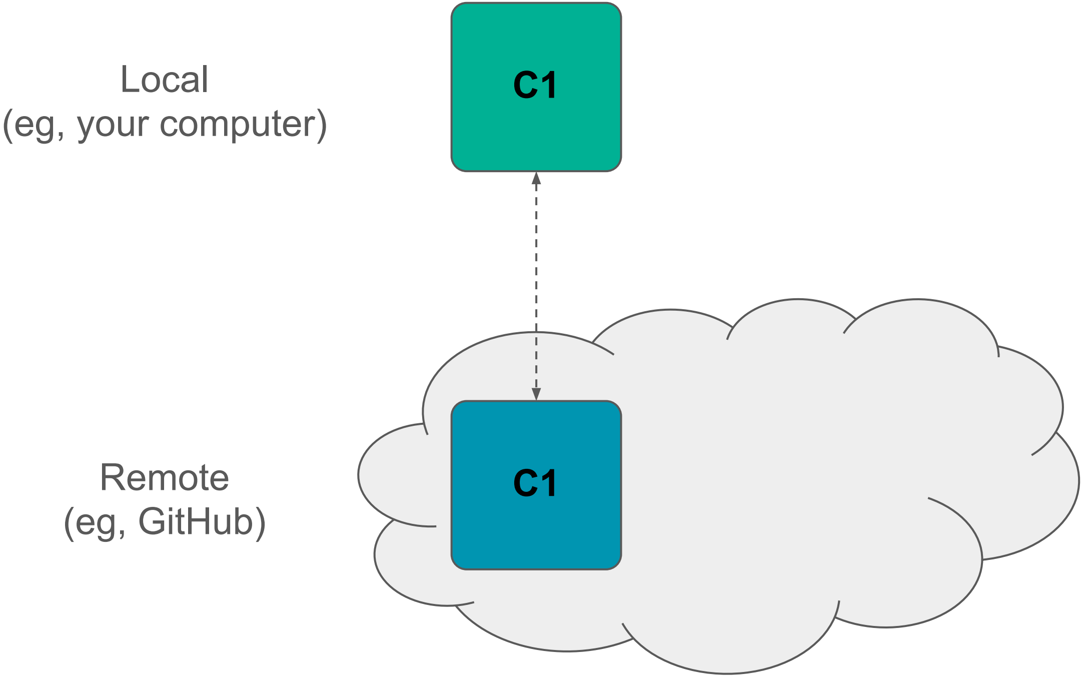
C1

Remote
(eg, GitHub)

<u>Git command</u>

```
git add file.txt
git commit -m "message"
```

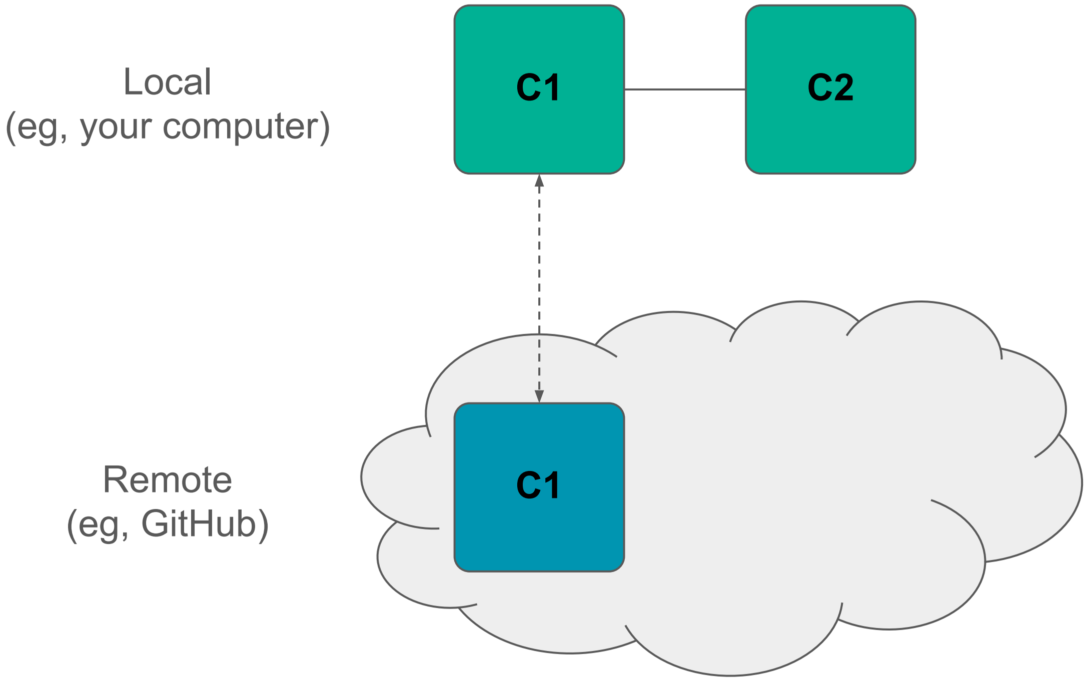
# Where does a repository live?



<u>Git command</u>

```
git push origin master
```

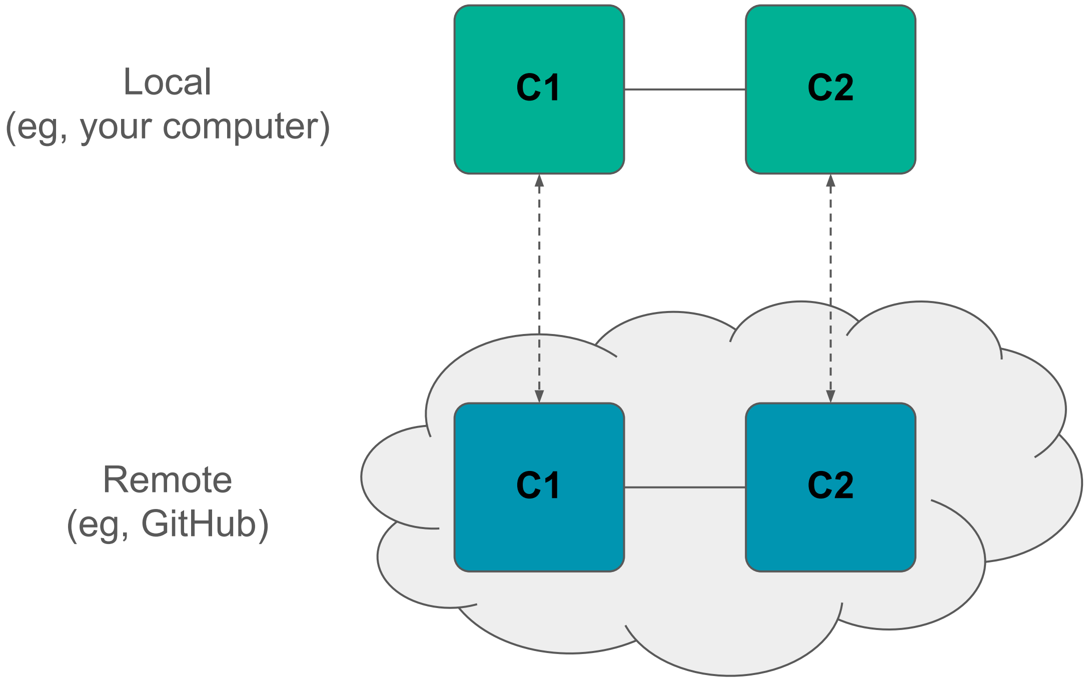
# Where does a repository live?



<u>Git command</u>

```
git add file.txt
git commit -m "message"
```

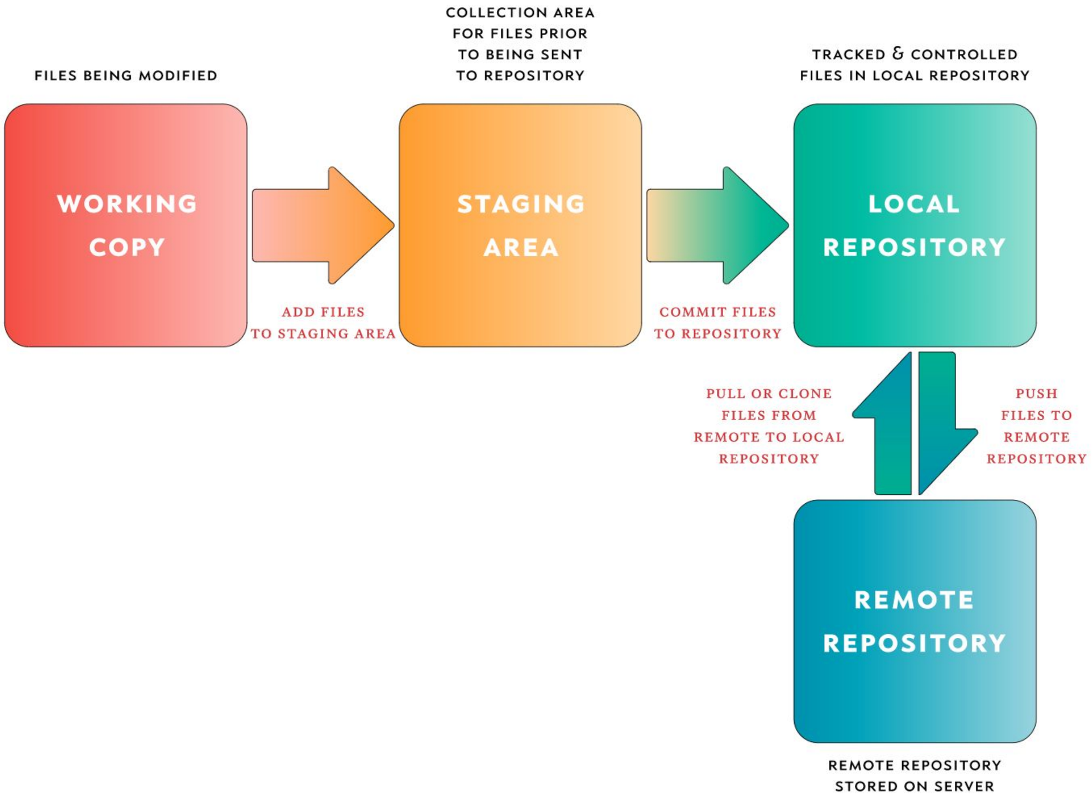
# Where does a repository live?



<u>Git command</u>

```
git push origin master
```

# Git workflow


FILES BEING MODIFIED
WORKING COPY
ADD FILES TO STAGING AREA
COLLECTION AREA FOR FILES PRIOR TO BEING SENT TO REPOSITORY
STAGING AREA
COMMIT FILES TO REPOSITORY
TRACKED & CONTROLLED FILES IN LOCAL REPOSITORY
LOCAL REPOSITORY
PULL OR CLONE FILES FROM REMOTE TO LOCAL REPOSITORY
PUSH FILES TO REMOTE REPOSITORY
REMOTE REPOSITORY
REMOTE REPOSITORY STORED ON SERVER

# Example: create local repository

| | |
|---|---|
| ● Navigate to desktop | `cd ~/Desktop` |
| ● Make directory | `mkdir sample` |
| ● Navigate to directory | `cd sample` |
| ● View all contents | `ls -a` |

What are the contents of the `sample` directory?

# Example: create local repository

- Create repository `git init`
- View all contents `ls -a`

How have the contents of the `sample` directory changed?

# Example: using a local repository

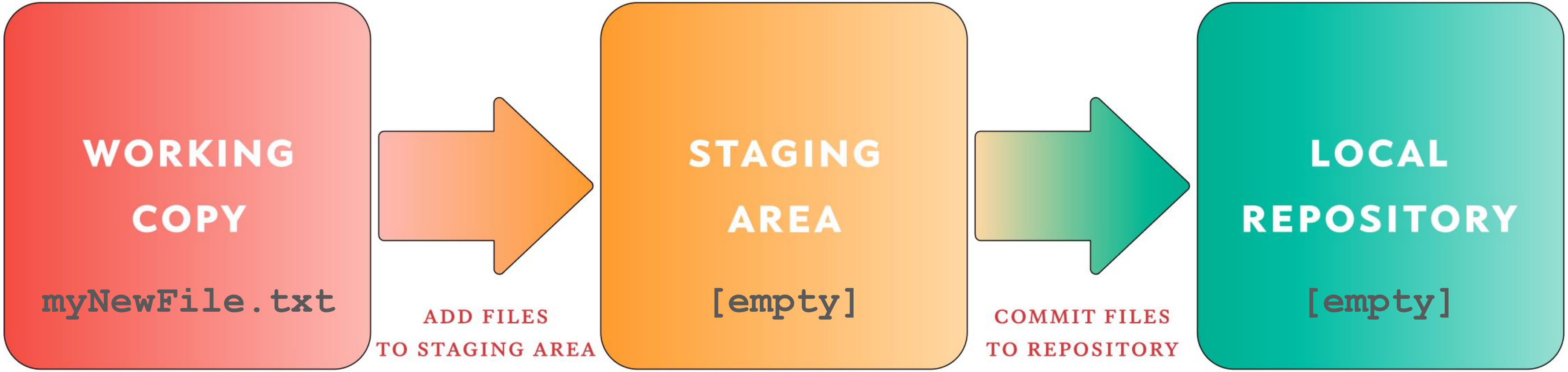
- Create new file
- Add a line to file
- Check the status

```
touch myNewFile.txt
echo ‘Hello, world!’ > myNewFile.txt
git status
```

# Example: using a local repository

- Create new file
- Add a line to file
- Check the status

```
touch myNewFile.txt
echo 'Hello, world!' > myNewFile.txt
git status
```



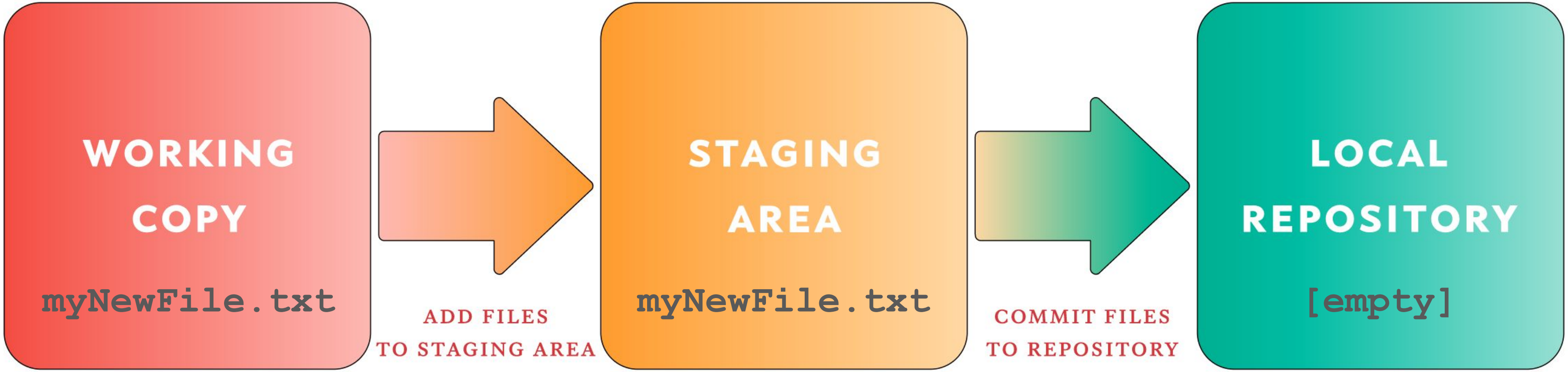
# Example: using a local repository

| | |
|---|---|
| ● Add file to staging area | `git add myNewFile.txt` |
| ● Check the status | `git status` |

# Example: using a local repository

- Add file to staging area
- Check the status

```
git add myNewFile.txt
git status
```



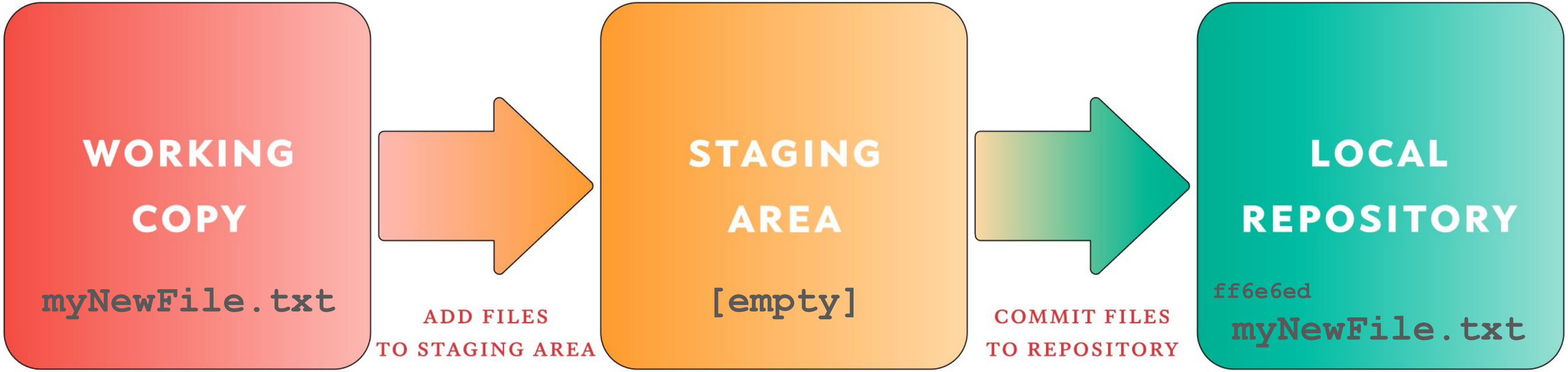
# Example: using a local repository

- Commit the changes
- Check the status

```
git commit -m “test file commit”
          git status
```

# Example: using a local repository

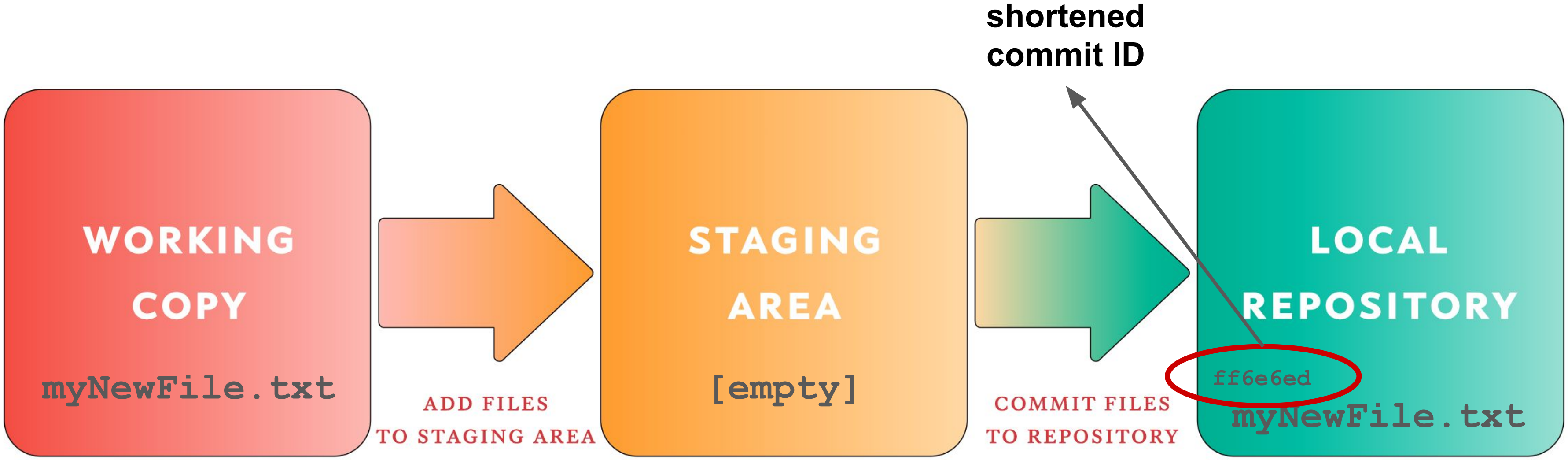
- Commit the changes `git commit -m “test file commit”`
- Check the status `git status`



# Example: using a local repository

- Commit the changes
- Check the status

```
git commit -m “test file commit”
git status
```



# Example: seeing the differences

- Open the file
- Add 2 lines
- View the differences

```
vi myNewFile.txt
# add two lines
git diff
```

# Example: seeing the differences

- Open the file
- Add 2 lines
- View the differences

- View the status
- Add the changes to staging area
- Commit changes

```
vi myNewFile.txt
# add two lines
git diff

git status
git add myNewFile.txt

git commit -m “adding two lines”
```

# Example: adding repository to remote

- On GitHub
  - Create new repository
  - Copy HTTPS link
  - Push repo from command line

```
git remote add origin https://    # copy the HTTPS link
git remote -v                     # view remote location
git push -u origin master         # push upstream
git log                           # view the log
```

See changes on GitHub!

# README.md

- Simple and brief documentation for others to understand your project
- README and wikis are the two ways to document your work on GitHub
- README is generally in the root directory, though all directories can have one
- ‘.md’ stands for markdown
- Markdown lightweight markup language with plain text formatting syntax

https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

# Exercise

- Create a directory called `first-repo`
- Make it a git repository
- Create two files: `sample.txt` and `README.md`
- Make a couple changes to each file
- Track and commit both files
- Create repository on GitHub
- Push local changes to remote repository
- Create a new directory called `session_1`

# Exercise

- Create a directory called `first-repo`
- Make it a git repository
- Create two files: `sample.txt` and `README.md`
- Make a couple changes to each file
- Track and commit both files
- Create repository on GitHub
- Push local changes to remote repository
- Create a new directory called `session_1`

Do you think we need to track this sub-directory separately?