## Objectives

After completing this tutorial you will be able to...

* Configure git.
* Make and commit changes to a repository.
* Add new files to a repository.
* Remove a file from a repository.
* Modify a file and commit the change.
* Stage changes for commit.
* Unstage changes for commit.
* Explain the purpose of the stage/cache/index.
* Inspect the state of a repository.
* Undo a commit.

Throughout the tutorial you will be asked many questions. The goal is to become familiar with basic git commands and to produce notes that you can refer back to later. Don’t worry too much about accuracy, for each question, provide your best answer and move on to the next question.

## Setup

### Git

Download and install git for your operating system:

* <https://git-scm.com/downloads>

Starting a terminal:

* **Windows**:
  + git-bash.exe (Linux style commands)
  + git-cmd.exe (Windows style commands)
* **Mac OSX:** Finder -> Applications -> Utilities -> Terminal.app
* **Linux:** will vary depending on your window manager

## Help

Run the following commands.

git help  
git help -ag  
git help init

1. What does git help do?
2. What does -ag cause git help to do?
3. What does git help [command] do?

## Identify yourself

Run the following commands, replacing BOGUS NAME and BOGUS@EMAIL with your name and email.

git config --global user.name 'BOGUS NAME'  
git config --global user.email 'BOGUS@EMAIL'

1. What are these commands doing?
2. What is the purpose of --global?

## Create repository

mkdir project  
cd project  
git init

1. What was created by git init?
2. By default any file that starts with . is hidden. How do you display a hidden file?
3. What would happen if you delete .git?
4. You find an old project on your hard drive. You do not remember if it is a under version control by git. How can you find out?

## Basic commands

Use a plain text editor to create names.txt inside the project folder. Put the names of your team in the file. Save and exit.

Run git status before and after each of these commands.

git add names.txt  
git commit –m *“*Add our names.*”*  
git log

1. What kind of information does git status report?
2. What does git add names.txt do?
3. What does git commit -m "Add our names." do?
4. What does git log do?

Use a plain text editor to create the following files:

* birthdays.txt - Put your birthdays in this file.
* movies.txt - Put the last movie each of you watched.

Run git status before and after each of these commands.

git add .  
git commit

Commit will open the vim editor; write a multi-line commit message, save and quit (press **esc** and then type **:wq**).

git log

1. What does git add . do? What do you think . means?
2. What does git commit (without -m) do?
3. If you want to write a more detailed commit message (which is good practice) what command would you use?

## Stage/Cache/Index

Do the following:

* Modify names.txt so that names are listed in *Last, First* format, one per line.
* Modify movies.txt so they are in reverse alphabetical order by title.
* Create a new file foods.txt that contains your favorite foods (one for each team member).

Run the following commands:

git add names.txt  
git status

1. Below write each file name under the state that its changes are currently in. Compose a definition for each state.

* **Staged**
* **Unstaged**
* **Untracked**

1. If you run git commit what changes will be committed (***don't do it***)?
2. What command do you run to stage changes?
3. What command do you run to unstage changes?

Run the following commands:

git diff  
git diff --cached

1. What does git diff display?
2. What does git diff --cached display?
3. Formulate a sequence of commands to unstage changes to names.txt, and stage the changes to movies.txt. Execute your commands and confirm they worked.
4. Edit movies.txt, change any one of the movies, and save it. Then run git status. What do you observe? Explain what you think is going on.
5. Delete names.txt. Then run git status. What do you observe? Explain what you think is going on.
6. Rename movies.txt to last-movies. Run git status. Observe and explain.
7. Formulate a sequence of commands to stage all changes including the untracked file and commit (with any reasonable message you like). Execute them.
8. In git vernacular, index, cache, and stage all refer to the same thing. What does it hold?
9. Why have a stage? Why not just commit all changes since the last commit?

## Undo

Run the following commands:

git log  
git status  
git reset --soft "HEAD^"  
git log  
git status

1. What does git reset --soft "HEAD^" do?

Run the following commands:

git commit –m "Redo."  
git log   
git status   
git reset --hard "HEAD^"   
git log   
git status

1. What does git reset --hard "HEAD^" do?
2. What is the difference between --hard and --soft?
3. What do you think HEAD^1 means?
4. What do you think HEAD means?

## Helpful resources

* <https://git-scm.com/doc>
* <https://www.atlassian.com/git/tutorials/>
* <https://training.github.com/kit/downloads/github-git-cheat-sheet.pdf>