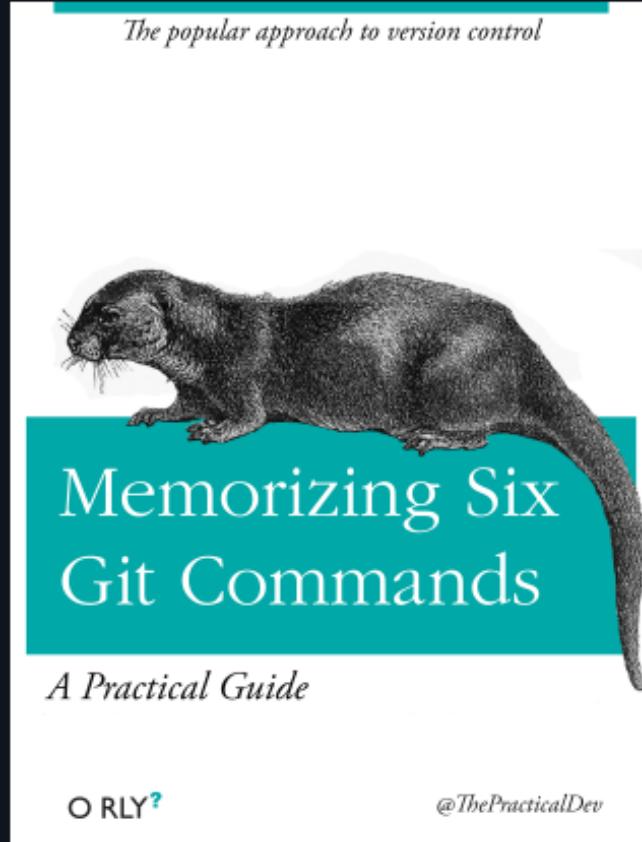


■ Git crash course

Mitsiu Alejandro
Carreño Sarabia



Agenda

- └─ Installation
- └─ Get help
- └─ Git workflow
- └─ Cheatsheet
- └─ Remote

 Installation

■ Installation <https://git-scm.com/install/>

Important settings:

Components -> Default

Editor -> Visual Studio Code as Git's default editor

Initial branch -> Override the default branch "main"

Path env -> Git from the command line & 3rd party

SSH -> Use bundled ssh

Https -> Native Windows Secure Channel

Line endings -> Checkout windows, commit unix

Git bash -> MinTTY

Git pull -> Fast forward or Merge

Credentials -> Git Credential Manager

Catching -> Enable catching

Verify the installation with:

```
git --version
```



Get help



Get help

This crash course will cover the most basic commands for submitting your exercises/homework. If you only take a single concept out of this slides is to learn how to get help on any git command.

The general formula for any git command is: `git <command> <args>`
any question you can do
`git <command> --help` o go to <https://git-scm.com/docs>

■ Get help

How did i learn the general formula?

```
$ git --help
usage: git [-v | --version] [-h | --help] [-C <path>] [-c
<name>=<value>]
          [--exec-path[=<path>]] [--man-path] [--info-path]
          [--no-replace-objects] [--bare]
          [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
          <command> [<args>]
```

Where:

- [] = optional argument or command
- | = other name for a given argument or command
- <> = placeholder

■ Understanding commands

```
$ git --help  
usage: git [-v | --version]
```

This two commands are equal

```
git -v  
git --version
```

[finished]

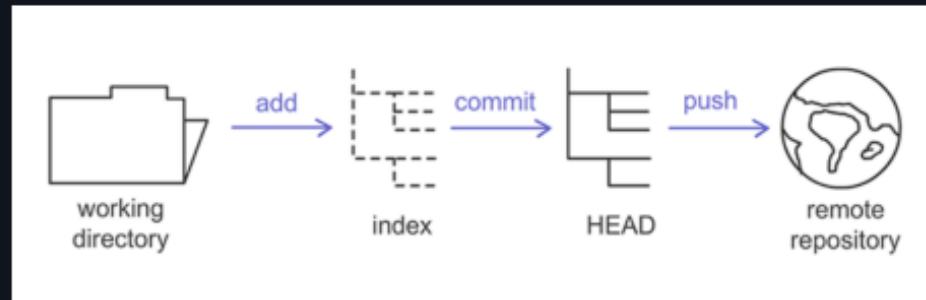
```
git version 2.52.0  
git version 2.52.0
```

Initial Setup

You must define your user and email with the following commands:

```
git config --global user.name "<Your username>"  
git config --global user.email "<Your email@alumnos.upa.edu.mx>"
```

Git workflow



Get your first repo

You can create a local copy of a remote repo with:

```
$ git clone <url>
```

This command will create a new folder in the current directory. Enter the new folder with `cd`

Git status

You can know the state of your repo with:

```
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

This output singals that no changes has been performed.

Let's modify `src/Helper.elm` and add "double x = 2 * x" at the bottom.

Check changes

```
$ git diff
diff --git a/src/Helper.elm b/src/Helper.elm
index 55edd56..5a1070a 100644
--- a/src/Helper.elm
+++ b/src/Helper.elm
@@ -1 +1,2 @@
 module Helper exposing (..)
+double x = 2 * x
```

- Green lines starting with + are new lines
- Red lines starting with - are deleted lines

Git state (Read your outputs)

- **Working directory** => The stage to **create, modify or delete** files. Git won't tracked this changes.

```
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working
     directory)
      modified:   src/Helper.elm

no changes added to commit (use "git add" and/or "git commit -a")
```

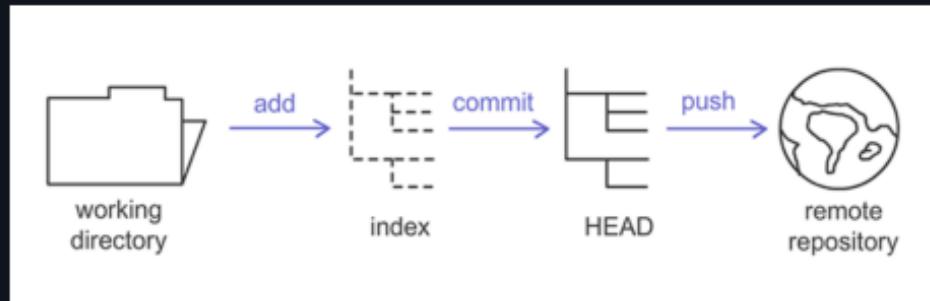
From working dir to index/stage (Read your outputs)

- **Index/Staging** => At this stage git start to track files and modifications locally

```
$ git add src/Helper.elm

$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   src/Helper.elm
```

From index/stage to HEAD (Read your outputs)



- **Commit** => Command to capture the current state in (index/stage)

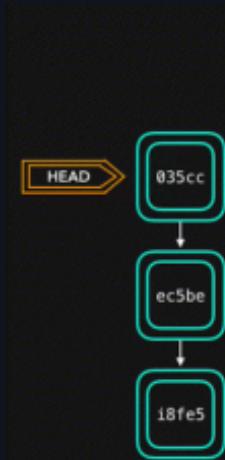
We must provide a simple description of the work to commit.

```
$ git commit -m'Complete double function'
```

This command creates a snapshot of the changes indexed/staged.
All work in the working directory remains unchanged.

Adding commits

- **Commit** => Command to capture the current state in (index/stage)
- **HEAD** => Pointer to the latest commit

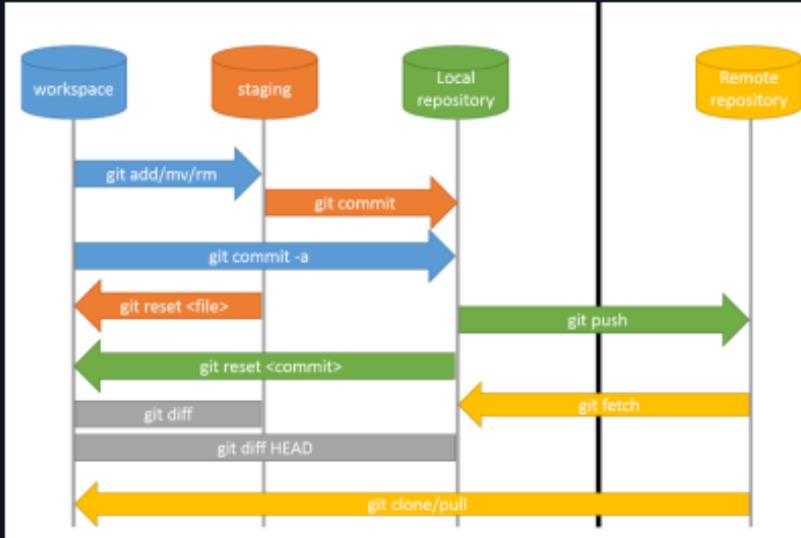


From HEAD to remote

- **Remote repo** When we are done with our local changes we can push them into our remote repo

```
$ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 520 bytes | 520.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:MACS-KINCAID/E1-elm-functions.git
  ecf4186..dd574be  main -> main
```

Cheatsheet



101 Git commands

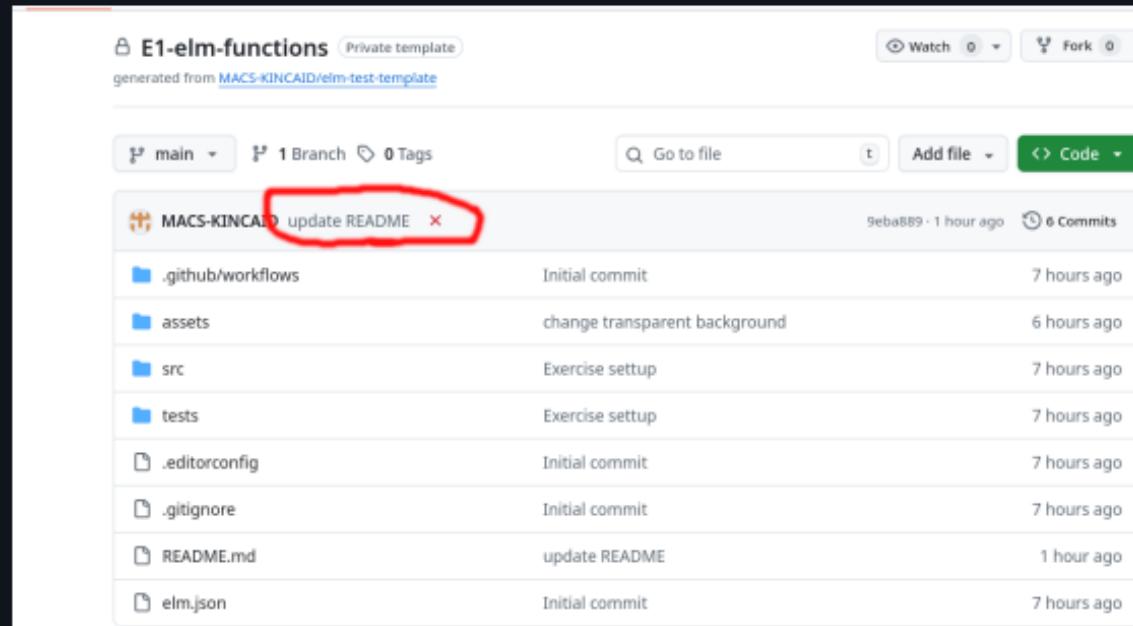
```
1 # Clone remote repo  
2 git clone <URL>  
3  
4 # Get current branch  
  status  
5 git status  
6  
7 # See changes in working  
  directory  
8 git diff
```

```
1 # Start tracking changes into files  
2 git add <file1> <file2>  
3  
4 # Create a snapshot with the content of  
  stage  
5 git commit -m'<Descrip de cambios>'  
6  
7 # Download remote changes  
8 git pull origin main  
9  
10 # Update remote changes with local  
   changes  
11 git push origin main
```

*origin => remote name

CI tests

On each push a set of test start to evaluate the quality of your code.
For more details click on the

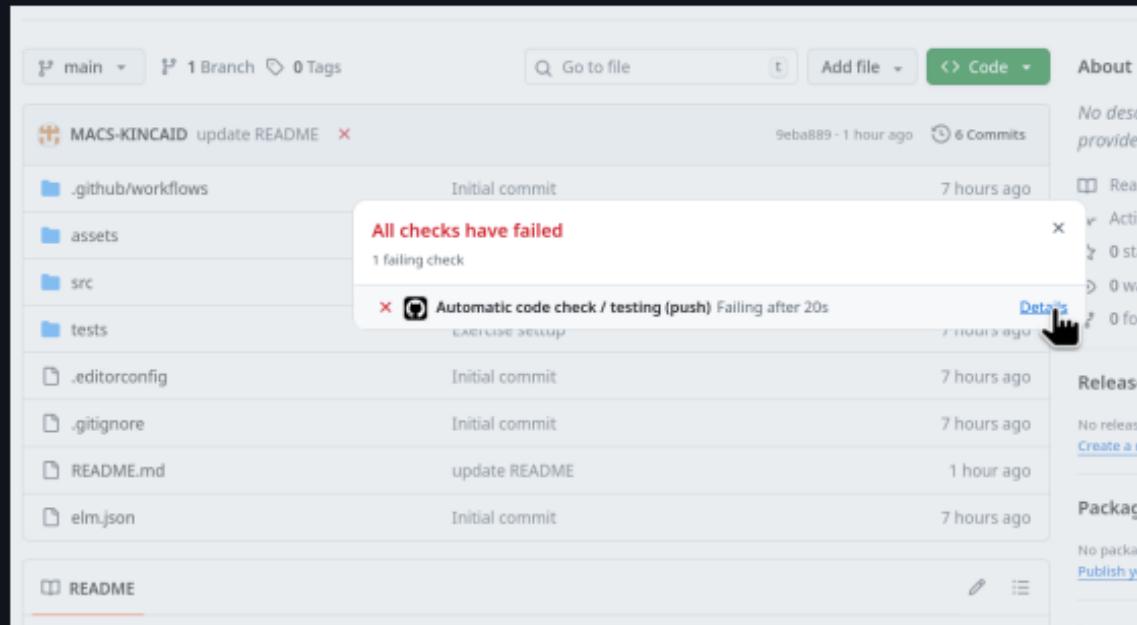


The screenshot shows a GitHub repository named "E1-elm-functions". The repository is private and was generated from "MACS-KINCAID/elm-test-template". It has 1 branch and 0 tags. The commit history is as follows:

Commit	Message	Time
.github/workflows	Initial commit	7 hours ago
assets	change transparent background	6 hours ago
src	Exercise setup	7 hours ago
tests	Exercise setup	7 hours ago
.editorconfig	Initial commit	7 hours ago
.gitignore	Initial commit	7 hours ago
README.md	update README	1 hour ago
elm.json	Initial commit	7 hours ago

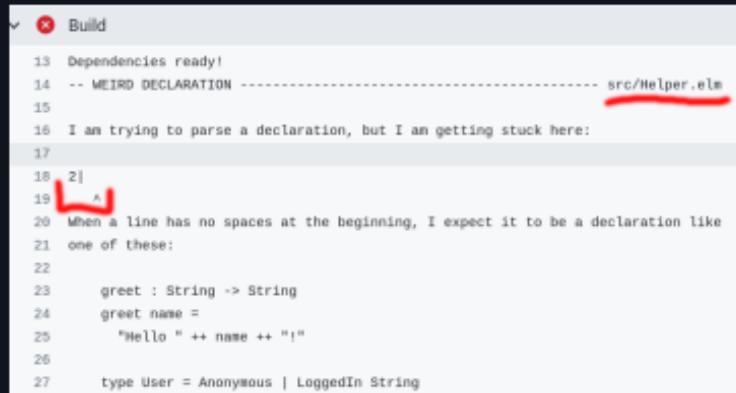
CI tests

Click on "details"



Fail detail

My example is pretty basic, my file is mostly empty, that's the error:



The screenshot shows a terminal window titled "Build" with the following output:

```
13 Dependencies ready!
14 -- MEIRD DECLARATION ----- src/Helper.elm
15
16 I am trying to parse a declaration, but I am getting stuck here:
17
18 2|
19  ^-----^
20 When a line has no spaces at the beginning, I expect it to be a declaration like
21 one of these:
22
23     greet : String -> String
24     greet name =
25         "Hello " ++ name ++ "!""
26
27 type User = Anonymous | LoggedIn String
```

A red box highlights the file path "src/Helper.elm". A red bracket highlights the first two lines of the error message, and a red arrow points to the first character of the second line.

CI commands

These are the four commands that are evaluated, you can run them locally. To solve the errors faster.

```
# Linux & Windows
# Check if our code complies the standarized elm rules:
elm-format src/ --validate
# Apply the format rules and modify our files:
elm-format src/
```

Verify our code compiles and is valid:

```
# Linux
elm make src/*
# Windows
elm make src/<file>.elm
```

CI commands

Check we provide all annotations:

```
# Linux
elm-review --template jfmengels/elm-review-common/example --rules
NoMissingTypeAnnotation,NoMissingTypeAnnotationInLetIn
# Windows Powershell
elm-review --template jfmengels/elm-review-common/example --rules
"NoMissingTypeAnnotation,NoMissingTypeAnnotationInLetIn"
# Windows cmd
elm-review --template jfmengels/elm-review-common/example --rules
NoMissingTypeAnnotation,NoMissingTypeAnnotationInLetIn
```

Check unit testing if present:

```
elm-test
```



CI commands

The commands are also in your class exercise/homework readme.
This is your first class exercise/homework.