## Guided Practice: Let's use Git (15 mins)

Let's make our first Git repo. Try this with me!

First, create a directory on your Desktop:

```bash

$ cd ~/Desktop

$ mkdir hello-world

$ cd hello-world

```

You can place this directory under Git revision control using the command:

```bash

$ git init

```

Git will reply:

```bash

Initialized empty Git repository in <location>

```

> Check: Did you get this message?

You've now initialized the working directory.

#### The .git folder

If we look at the contents of this empty folder using:

```bash

ls -a

```

We should see that there is now a hidden folder called `.git` this is where all of the information about your repository is stored. There is no need for you to make any changes to this folder. You can control all the git flow using `git` commands.

#### Add a file

Let's create a new file:

```bash

$ touch a.txt

```

If we run `git status` we should get:

```bash

On branch master

Initial commit

Untracked files:

(use "git add <file>..." to include in what will be committed)

a.txt

nothing added to commit but untracked files present (use "git add" to track)

```

This means that there is a new \*\*untracked\*\* file. Next, tell Git to take a snapshot of the contents of all files under the current directory (note the .)

```bash

$ git add .

```

This snapshot is now stored in a temporary staging area which Git calls the "index".

#### Commit

To permanently store the contents of the index in the repository, (commit these changes to the HEAD), you need to run:

```bash

$ git commit -m "Please remember this file at this time"

```

You should now get:

```bash

[master (root-commit) b4faebd] Please remember this file at this time

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 a.txt

```

> Check: Did you get this message?

#### Checking the log

If we want to view the commit history, we can run:

```bash

git log

```

You should see:

```bash

\* b4faebd (HEAD -> master) Please remember this file at this time

```

> Check: Did you get the message above?

#### Make changes to the file

Now let's open a.txt in a text editor.

Inside the file, write something.

```

This next line is dependent on config of the VM - currently it will not show git dif

If you press `return` in the terminal, you will now see that you have untracked changes.

```

Running `git status` again will show you that a.txt has been \*\*modified\*\*.

#### Revert to a previous commit

Let's now make a second commit.

```bash

$ git add .

$ git commit -m "Second commit"

```

Checking `git log` will show you 2 commits with different ids:

```bash

\* 6e78569 (HEAD, master) Second commit

\* b4faebd Please remember this file at this time

```

We can revert the file back to the first commit using it's specific commit id with:

```bash

$ git reset --soft b4faebd

```

This will do a soft reset, where the changes in the file we made are still there - the changes are staged but not committed anymore.

If we want to revert the file back and disregard any changes (dangerous!), we can use:

```bash

$ git reset --hard b4faebd

```