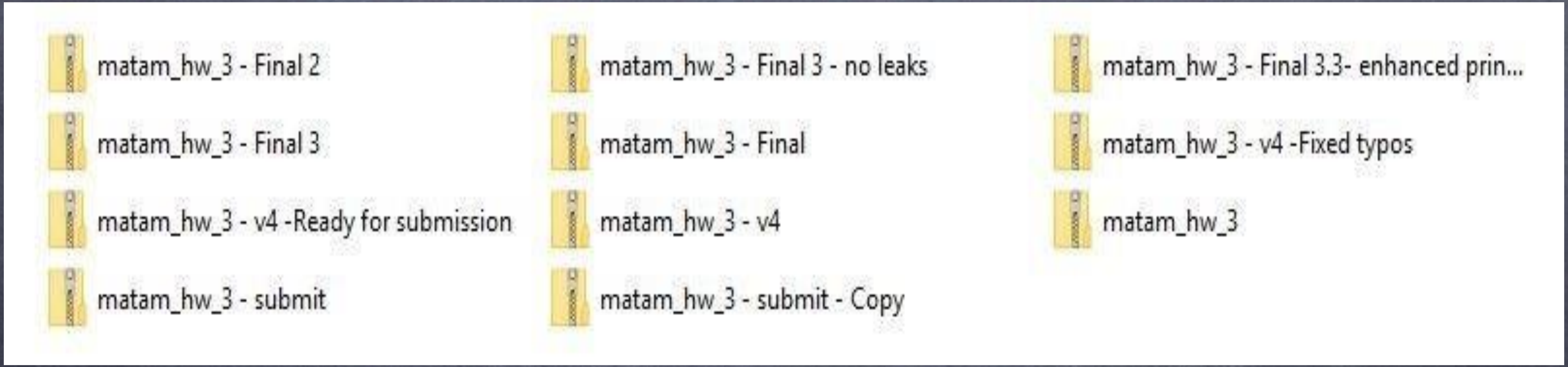
Introduction to git

ITI - Assiut Branch
Eng. Sara Ali

Why git is useful?

Use cases



matam_hw_3 - Final 2
matam_hw_3 - Final 3
matam_hw_3 - v4 -Ready for submission
matam_hw_3 - submit
matam_hw_3 - Final 3 - no leaks
matam_hw_3 - Final
matam_hw_3 - v4
matam_hw_3 - submit - Copy
matam_hw_3 - Final 3.3- enhanced prin...
matam_hw_3 - v4 -Fixed typos
matam_hw_3

Use cases

Consider this scenario:

- You have a homework submission for today and the assignment is ready for submission
- While testing it you discovered a minor bug and decided to fix it

Use cases

- After attempting to do so, you accidentally changed a working code and got yourself in a big mess
- You no longer remember what was and what wasn’t there
- It is 23:58 PM

Now you realize that **CTRL + Z** won’t solve your problems

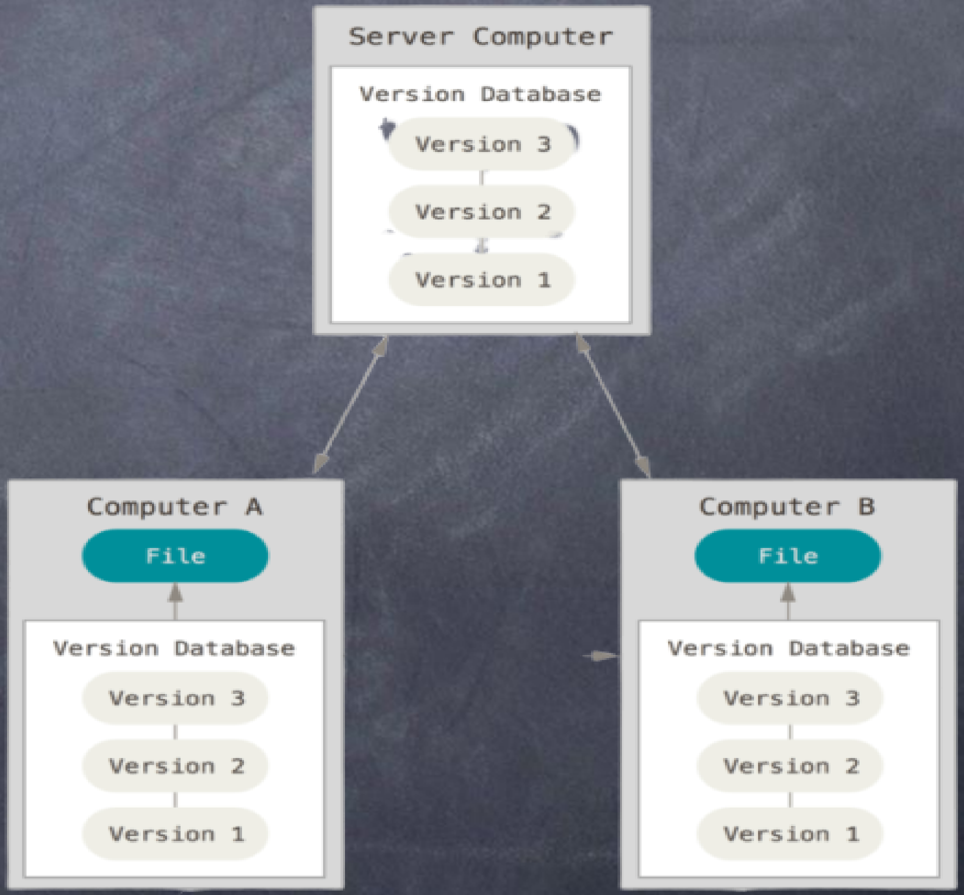
What is git?

What is git?

- Open source project originally developed in 2005 by Linus Torvalds
- A command line utility
- You can imagine **git** as something that sits on top of your file system and manipulates files.
- A distributed version control system - **DVCS**

What is “distributed version control system” ?

- **Version control system** is a system that records changes to a file or set of files over time so that you can recall specific versions later
- **Distributed** means that there is no main server and all of the full history of the project is available once you cloned the project.



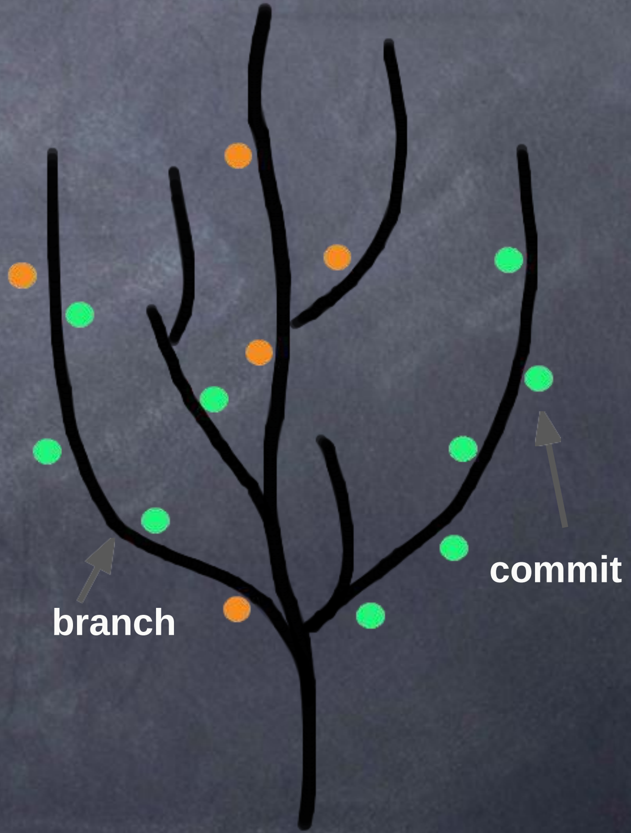
A brief history

- In 2002, the Linux kernel project began using a **DVCS** called BitKeeper
- In 2005, the commercial company that developed BitKeeper broke down, and the tool’s free-of-charge status was revoked
- This prompted the Linux development community (and in particular Linus Torvalds, the creator of Linux) to develop their own tool - **git**

Demo

git

- You can imagine **git** as something that sits on top of your file system and manipulates files.
- This “something” is a **tree** structure where each **commit** creates a new node in that tree.
- Nearly all **git** commands actually serve to navigate on this tree and to manipulate it accordingly.

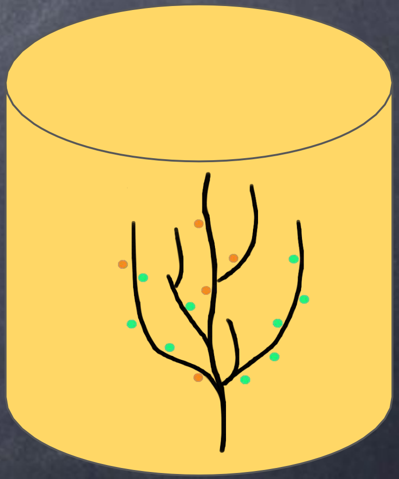


git repository

git repository

- The purpose of **git** is to manage a project, or a set of files, as they change over time. **Git** stores this information in a data structure called a **repository**

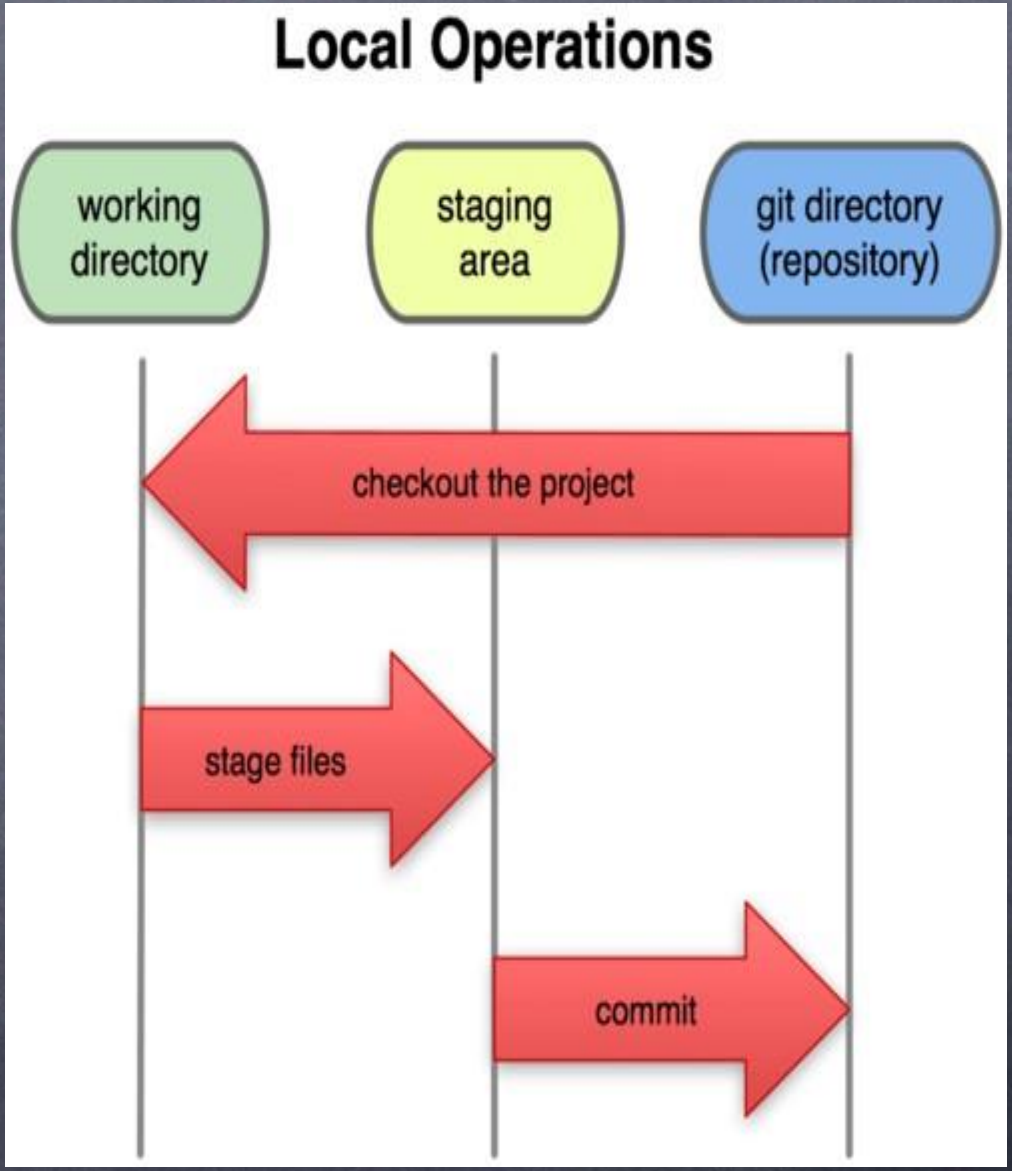
- A **git repository** contains, mainly:
  - A set of **commits**



Three states

Three main states of a Git:

- Working Directory
- Staging Area
- Git Directory
  (repository)



Working directory - Single checkout of one version of the project.

The files in the directory are pulled out of the compressed database in the Git directory and placed on disk for you to edit and use.

Staging area - It is a simple file, generally present in your Git directory, that stores information about what will go into your next commit.

Repository - Stores the metadata and object database for your project (while cloning the repository)

Installing Git

-Before you start using Git, you have to make it available on your computer.

https://git-scm.com/downloads

-If you already have Git installed, you can get the latest development version via Git itself:

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

First-Time Git Setup

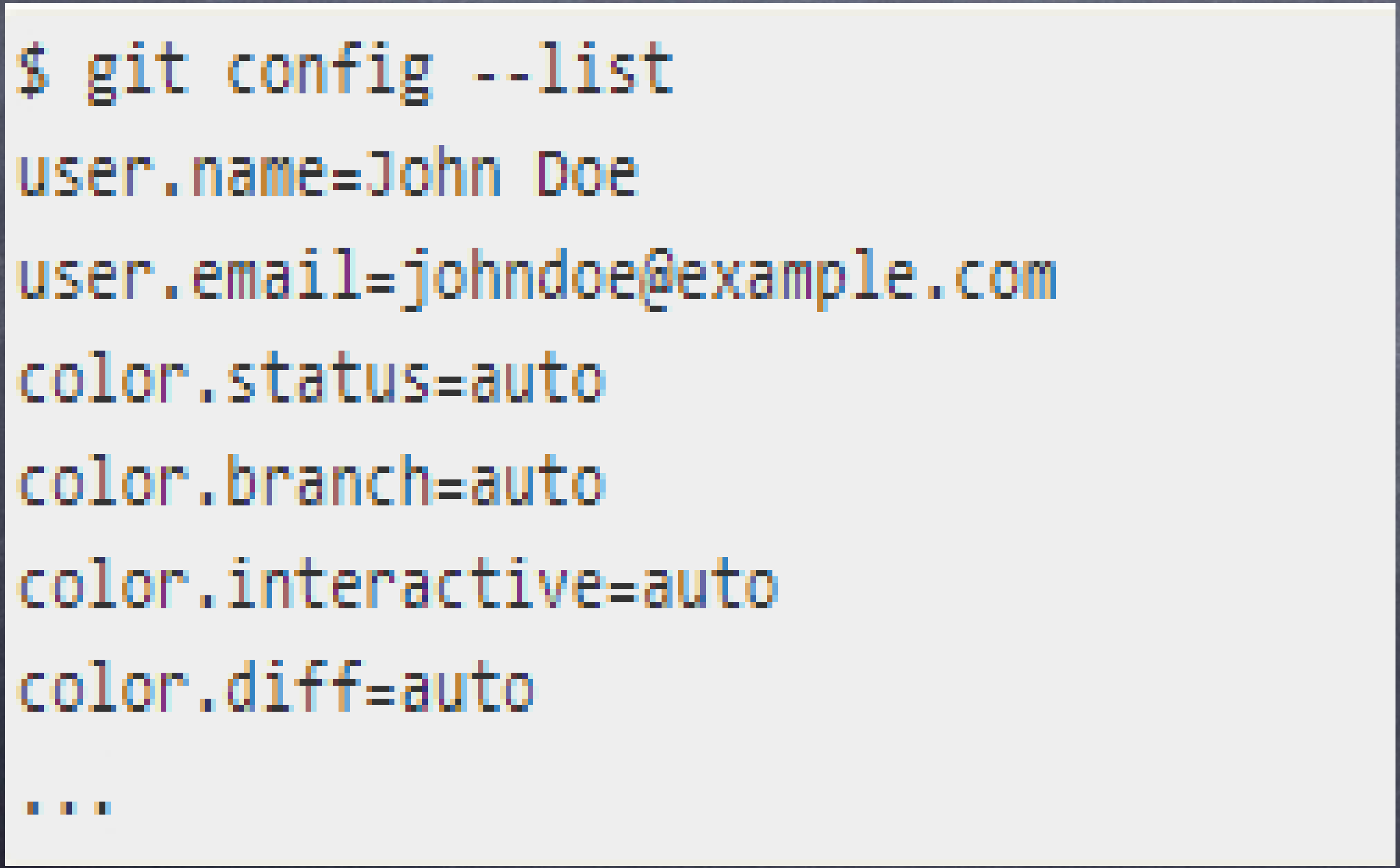
- **Your Identity:**

The first thing you should do when you install Git is to set your user name and email address. This is important because every Git commit uses this information, and it’s immutably baked into the commits you start creating:

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

- **Checking Your Settings:**

If you want to check your configuration settings, you can use the **git config --list** command to list all the settings Git can find at that point:



```
$ git config --list
user.name=John Doe
user.email=johndoe@example.com
color.status=auto
color.branch=auto
color.interactive=auto
color.diff=auto
...
```

git commands

git commands

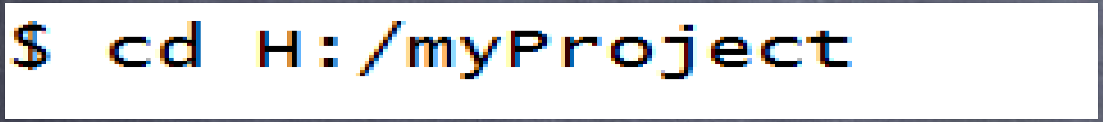
- For most of the basic interactions with git you’ll mainly use **7 commands** that we’ll cover here

git commands

- git init
- git status
- git add
- git commit
- git diff
- git log
- git clone

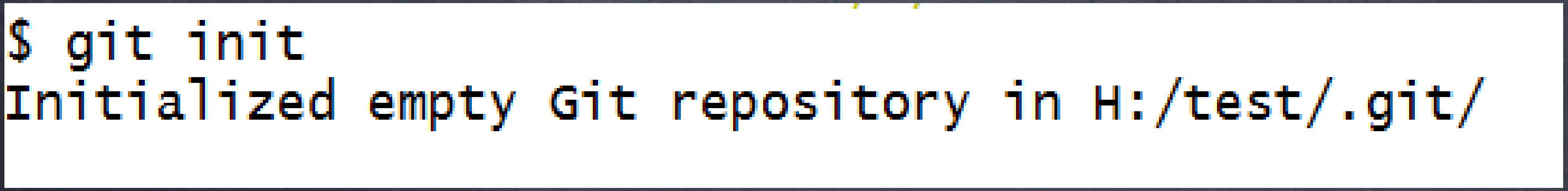
git init

- Creates a new **git repository**
- Can be used to convert an existing, unversioned project to a **git repository** or initialize a new **empty repository**
- If you have a project directory that is currently not under version control and you want to start controlling it with Git, you first need to go to that project's directory.



```
$ cd H:/myProject
```

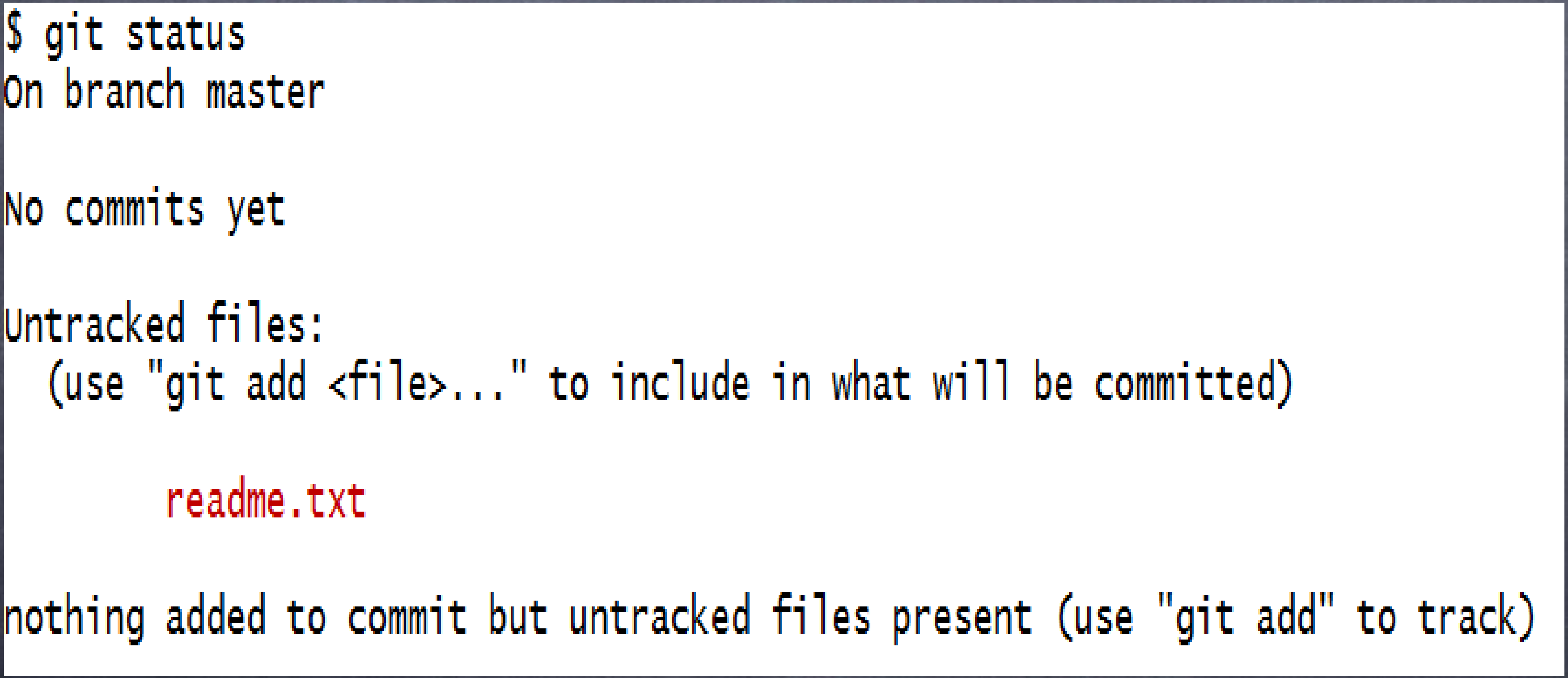
and type: git init



```
$ git init
Initialized empty Git repository in H:/test/.git/
```

git status

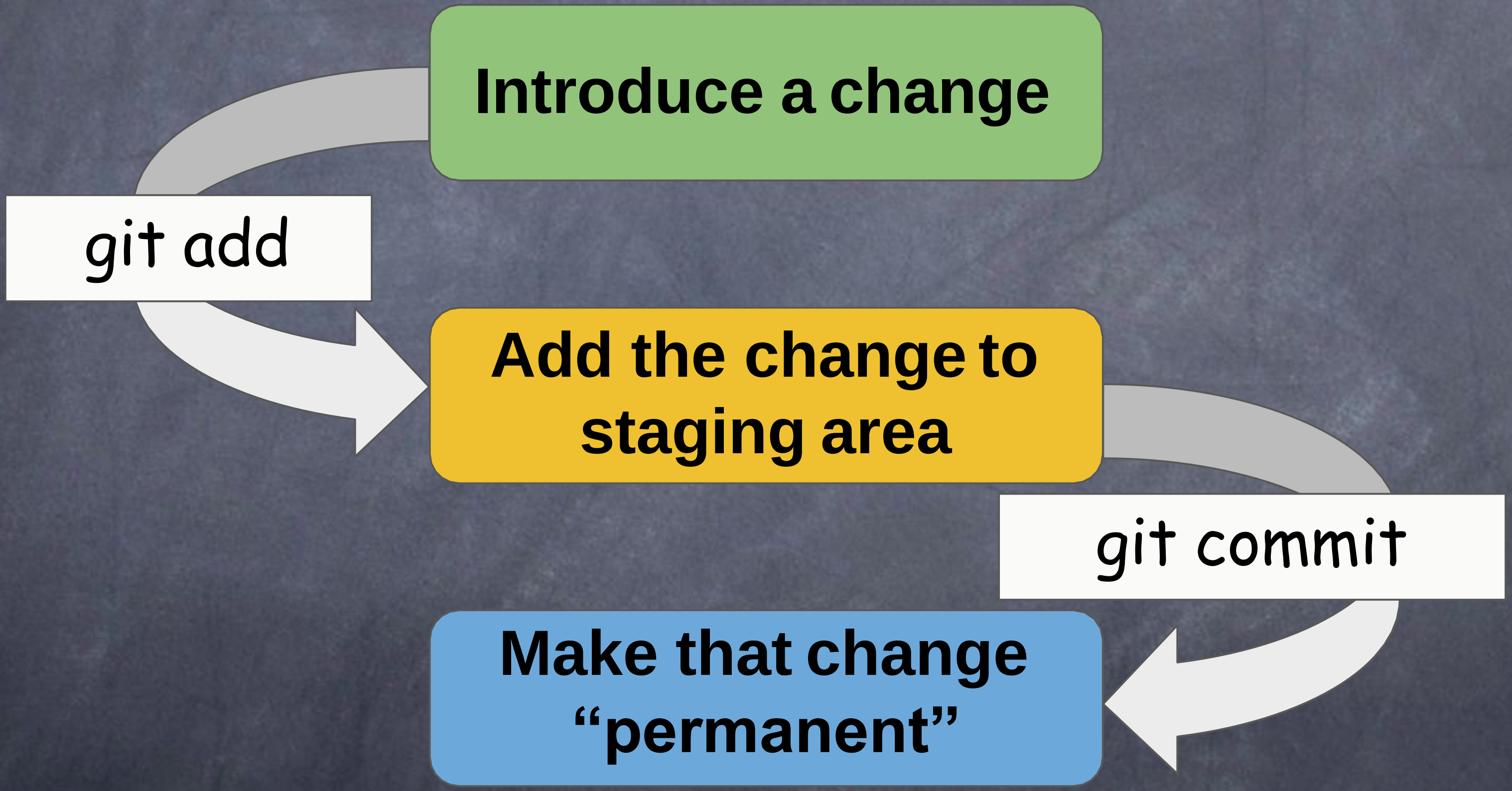
- Displays the file names that has been modified, added and untracked



```
$ git status
On branch master

No commits yet

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

        readme.txt

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

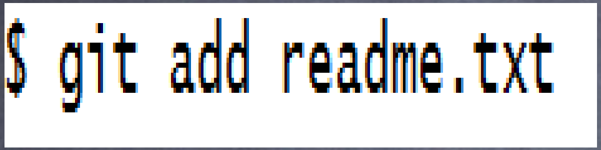
git add

- Adds changes:



git add (cont.)

$ git add file1 file2



```
$ git add readme.txt
```

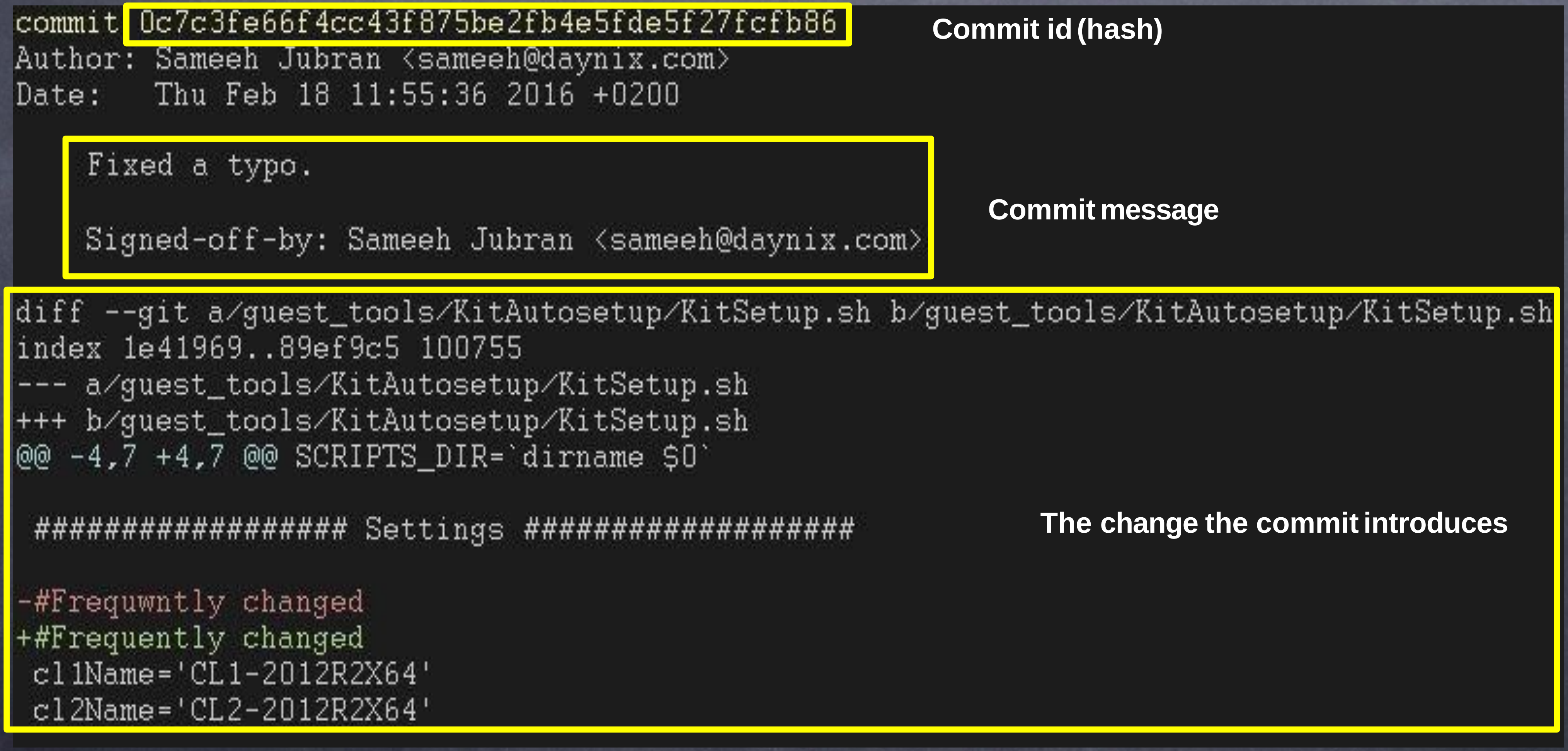
- To add all files

```
$ git add .
```

Commit

- A **commit** object mainly contains three things:
  - A set of **changes** the **commit** introduces
  - **Commit message** describing the changes
  - A **hash**, a 40-character string that uniquely identifies the commit object

Commit

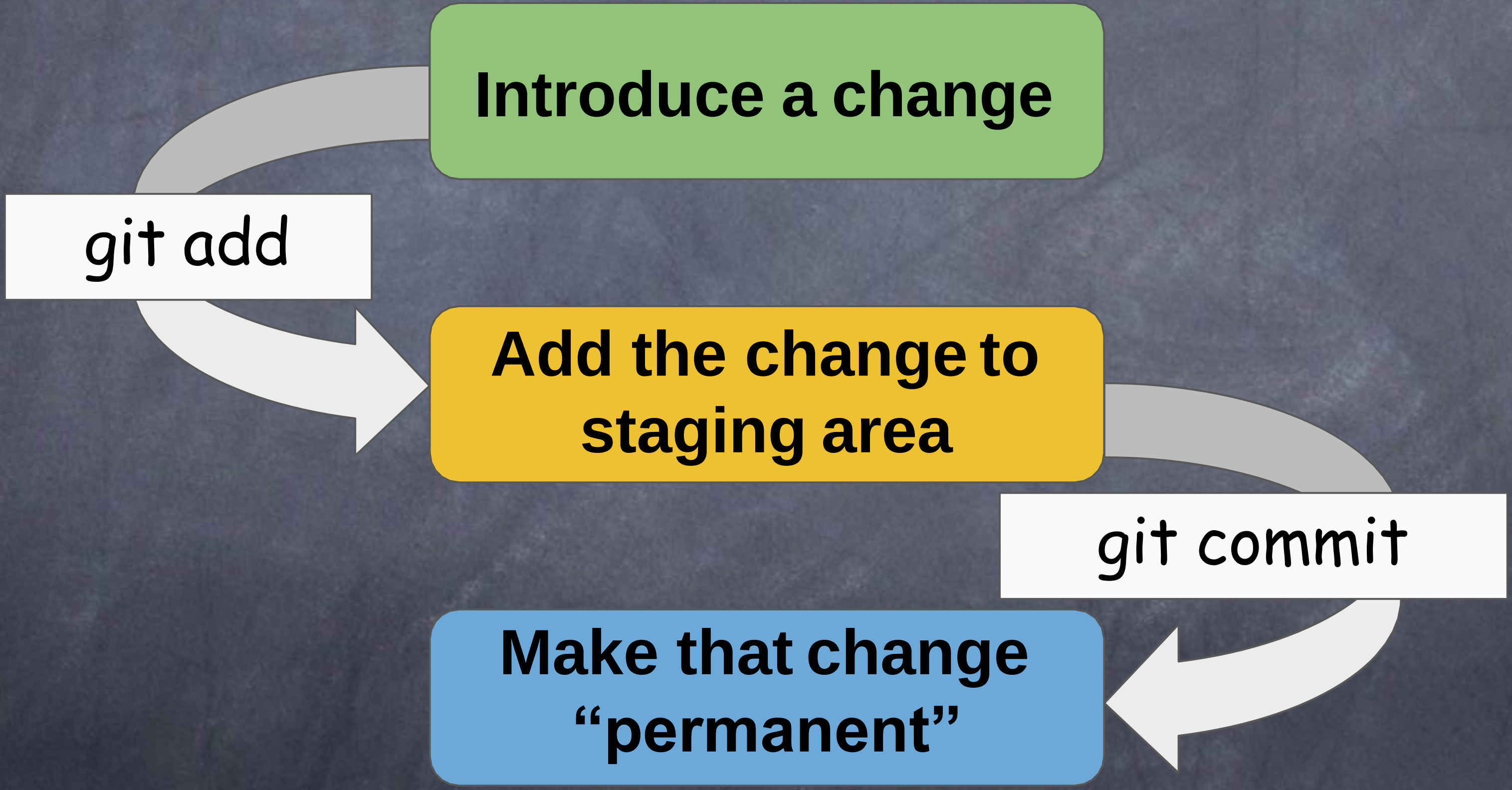


```
commit 0c7c3fe66f4cc43f875be2fb4e5fde5f27fcfb86
Author: Sameeh Jubran <sameeh@daynix.com>
Date:   Thu Feb 18 11:55:36 2016 +0200

    Fixed a typo.

    Signed-off-by: Sameeh Jubran <sameeh@daynix.com>

diff --git a/guest_tools/KitAutosetup/KitSetup.sh b/guest_tools/KitAutosetup/KitSetup.sh
index 1e41969..89ef9c5 100755
--- a/guest_tools/KitAutosetup/KitSetup.sh
+++ b/guest_tools/KitAutosetup/KitSetup.sh
@@ -4,7 +4,7 @@ SCRIPTS_DIR=`dirname $0`

 ################## Settings ###################

-#Frequwntly changed
+#Frequently changed
 cl1Name='CL1-2012R2X64'
 cl2Name='CL2-2012R2X64'
```

Let’s create a commit

git commit

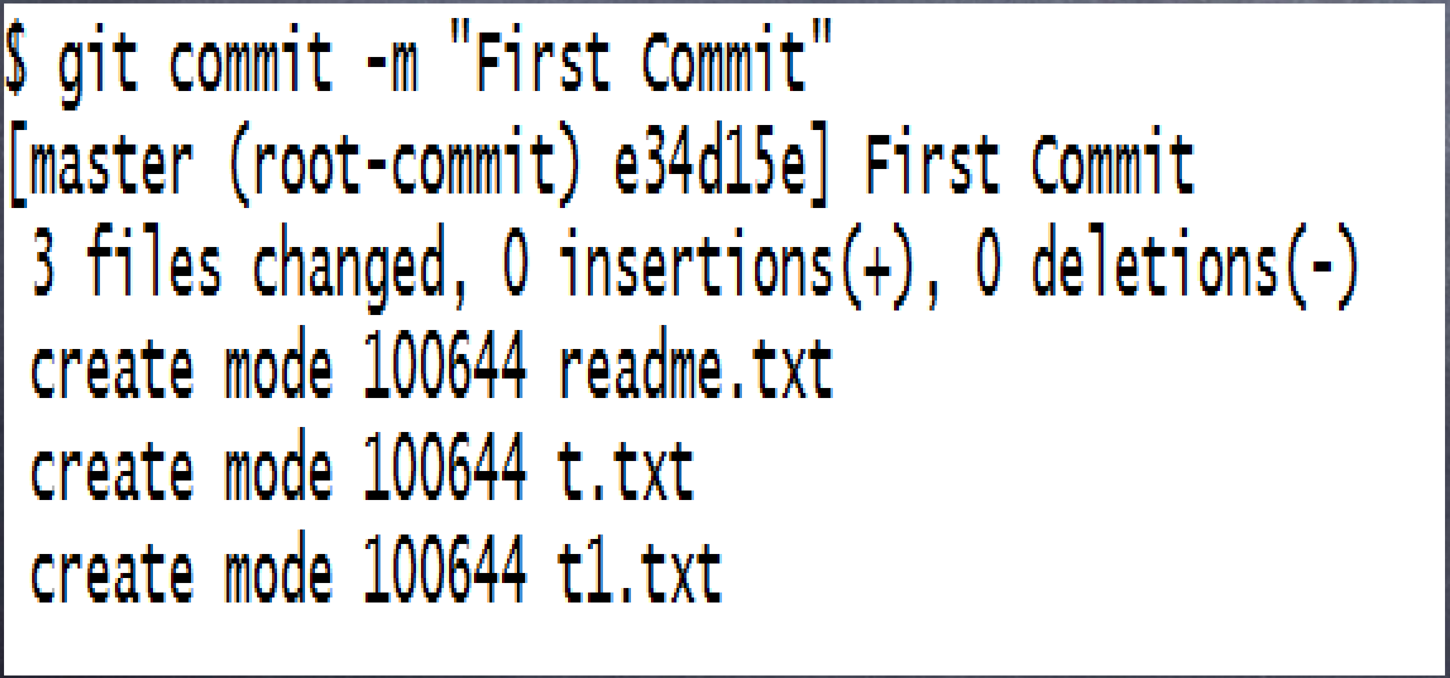
- Creates a **commit** out of the changes that had been added



git commit (cont.)

- Commit takes files from the Staging area and saves them in the repository.

- To make a commit:
  git commit –m "Your commit message"



```
$ git commit -m "First Commit"
[master (root-commit) e34d15e] First Commit
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 readme.txt
 create mode 100644 t.txt
 create mode 100644 t1.txt
```

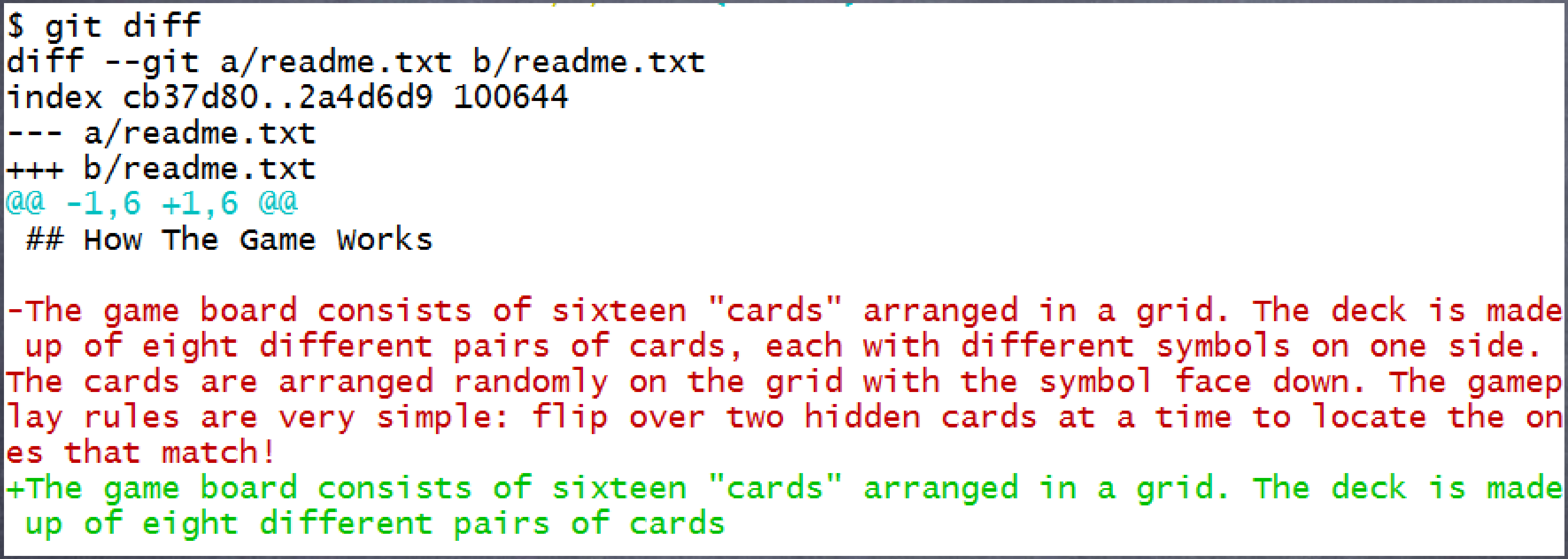
git commit (cont.)

- To skip the staging area during commits

  Command : git commit –a –m “Comments”

git diff

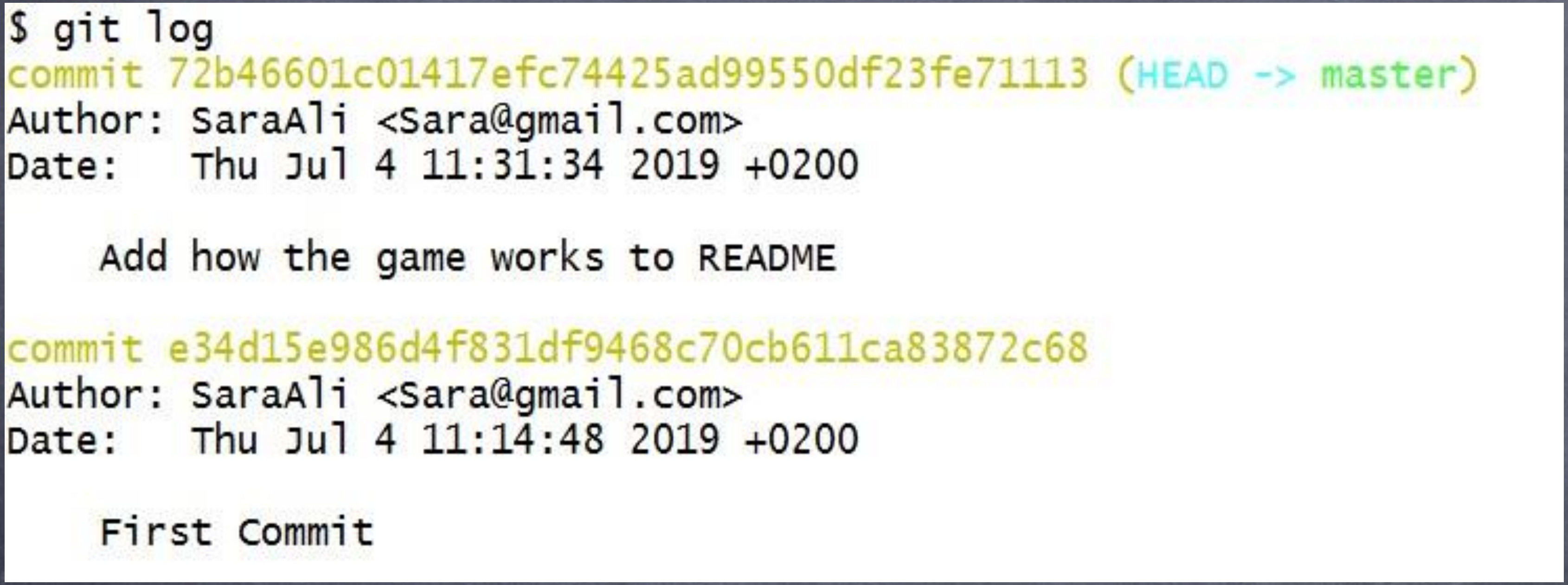
- Displays the change that was introduced: **git diff**



```
$ git diff
diff --git a/readme.txt b/readme.txt
index cb37d80..2a4d6d9 100644
--- a/readme.txt
+++ b/readme.txt
@@ -1,6 +1,6 @@
 ## How The Game Works

-The game board consists of sixteen "cards" arranged in a grid. The deck is made
 up of eight different pairs of cards, each with different symbols on one side.
The cards are arranged randomly on the grid with the symbol face down. The gamep
lay rules are very simple: flip over two hidden cards at a time to locate the on
es that match!
+The game board consists of sixteen "cards" arranged in a grid. The deck is made
 up of eight different pairs of cards
```

git log

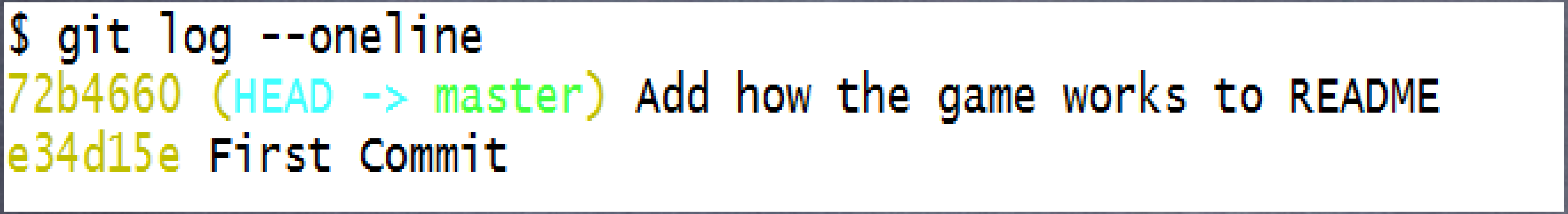
Shows the **commit** logs



```
$ git log
commit 72b46601c01417efc74425ad99550df23fe71113 (HEAD -> master)
Author: SaraAli <Sara@gmail.com>
Date:   Thu Jul 4 11:31:34 2019 +0200

    Add how the game works to README

commit e34d15e986d4f831df9468c70cb611ca83872c68
Author: SaraAli <Sara@gmail.com>
Date:   Thu Jul 4 11:14:48 2019 +0200

    First Commit
```

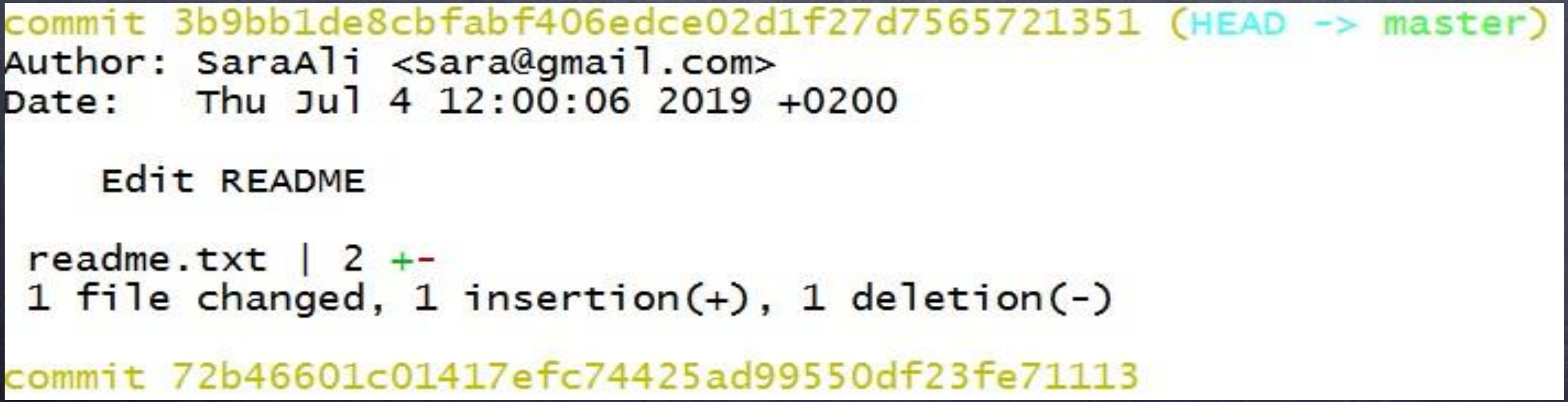
git log (cont.)

- The --oneline option prints each commit on a single line, which is useful if you're looking at a lot of commits.
  $ **git log –oneline**



```
$ git log --oneline
72b4660 (HEAD -> master) Add how the game works to README
e34d15e First Commit
```

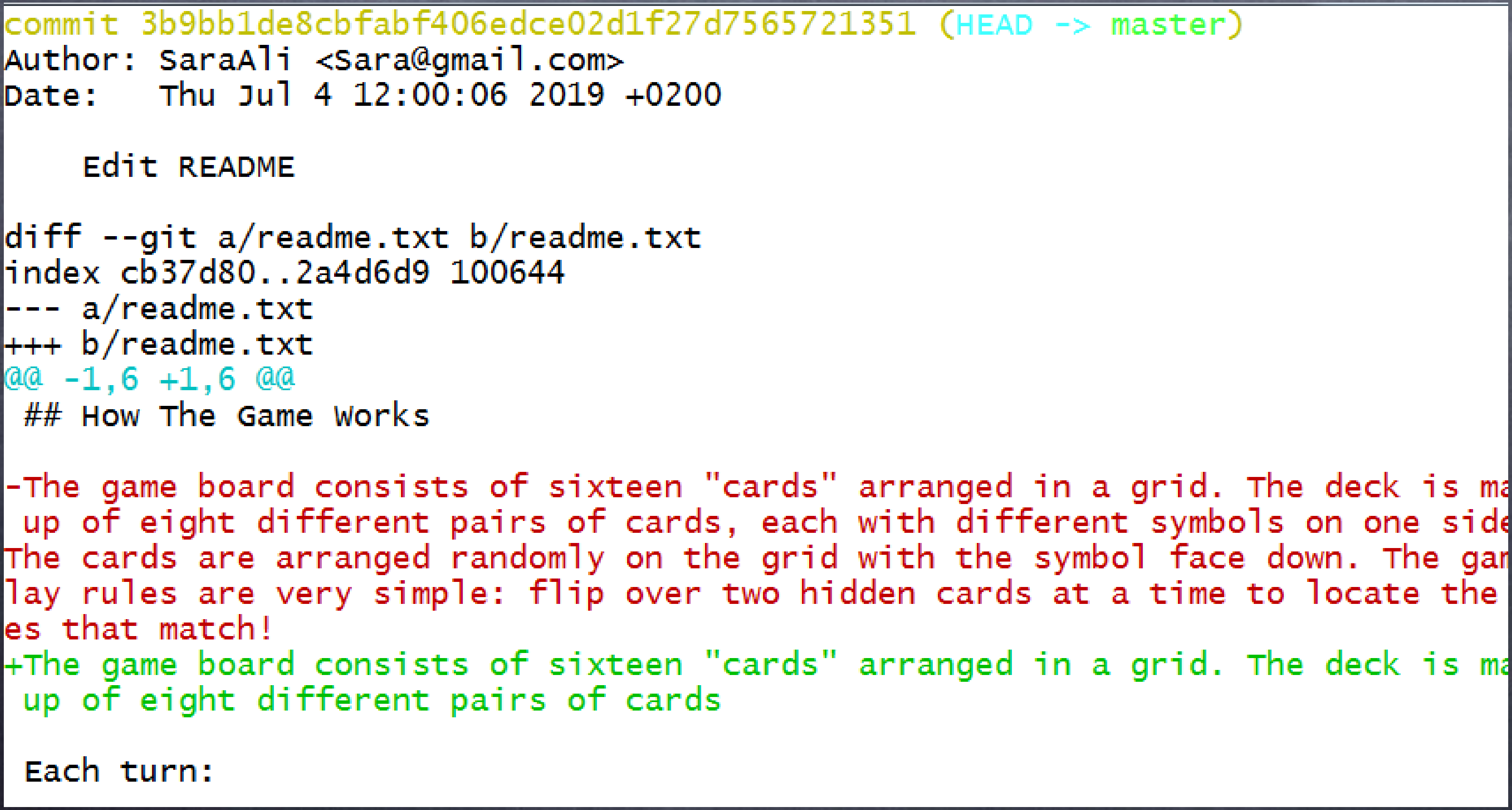
•display the files that have been changed in the commit, as well as the number of lines that have been added or deleted: $ **git log --stat**



```
commit 3b9bb1de8cbfabf406edce02d1f27d7565721351 (HEAD -> master)
Author: SaraAli <Sara@gmail.com>
Date:   Thu Jul 4 12:00:06 2019 +0200

    Edit README

 readme.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

commit 72b46601c01417efc74425ad99550df23fe71113
```

git log (cont.)

- The git log command has a flag that can be used to display the actual changes made to a file
  $ git log -p



```
commit 3b9bb1de8cbfabf406edce02d1f27d7565721351 (HEAD -> master)
Author: SaraAli <Sara@gmail.com>
Date:   Thu Jul 4 12:00:06 2019 +0200

    Edit README

diff --git a/readme.txt b/readme.txt
index cb37d80..2a4d6d9 100644
--- a/readme.txt
+++ b/readme.txt
@@ -1,6 +1,6 @@
 ## How The Game Works

-The game board consists of sixteen "cards" arranged in a grid. The deck is ma
 up of eight different pairs of cards, each with different symbols on one side
The cards are arranged randomly on the grid with the symbol face down. The gam
lay rules are very simple: flip over two hidden cards at a time to locate the
es that match!
+The game board consists of sixteen "cards" arranged in a grid. The deck is ma
 up of eight different pairs of cards

 Each turn:
```

git clone

- Copies an existing **git repository**

```
$ git clone https://github.com/schacon/simplegit-progit
```

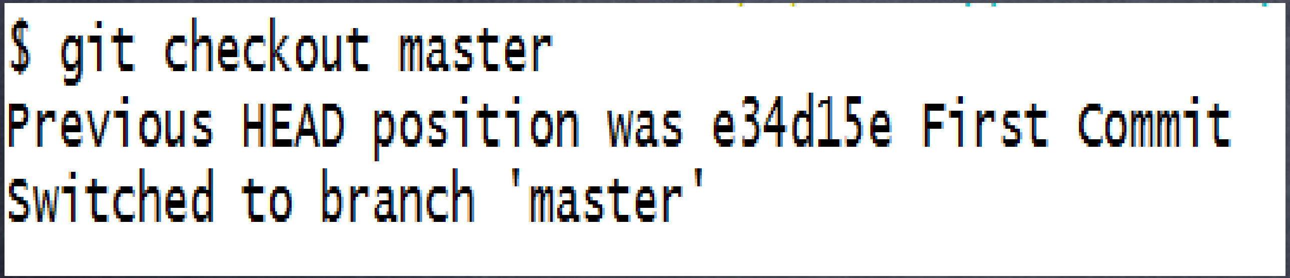
Bonus command: git checkout

- Checking out a commit makes the entire working directory match that commit

  $ **git checkout commitID**

- To checkout master to get the last commit

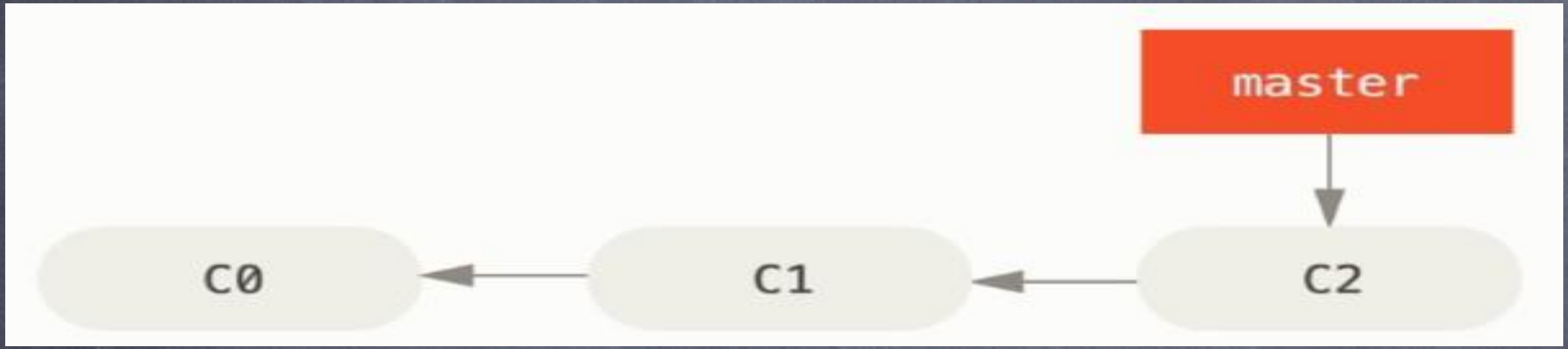
  $ **git checkout master**



```
$ git checkout master
Previous HEAD position was e34d15e First Commit
Switched to branch 'master'
```

Branching and Merging

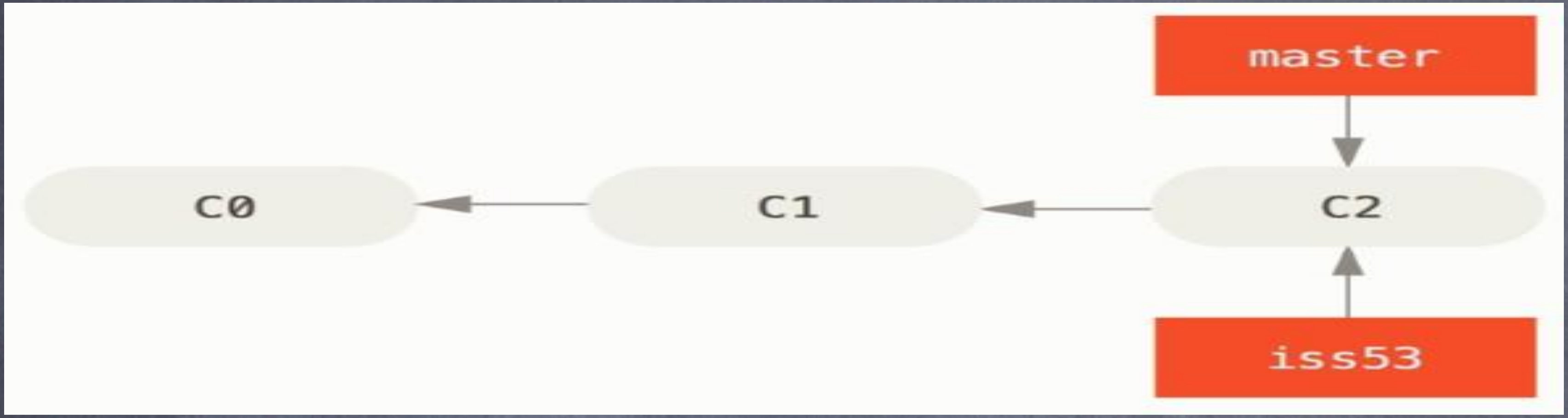
First, let’s say you’re working on your project and have a couple of commits already on the masterbranch.



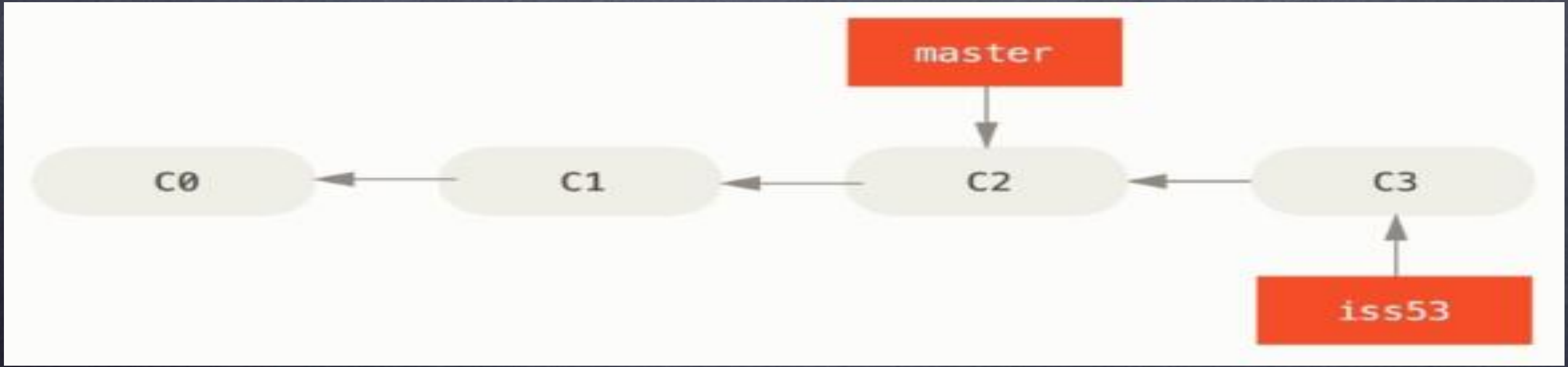
You’ve decided that you’re going to work on issue #53 in whatever issue-tracking system your company uses.

- To create a new branch: $ **git branch iss53**
- To switch to it: $ **git checkout iss53**

Branching and Merging(cont.)



You work on your website and do some commits. Doing so moves the iss53 branch forward, because you have it checked out (that is, your HEAD is pointing to it):

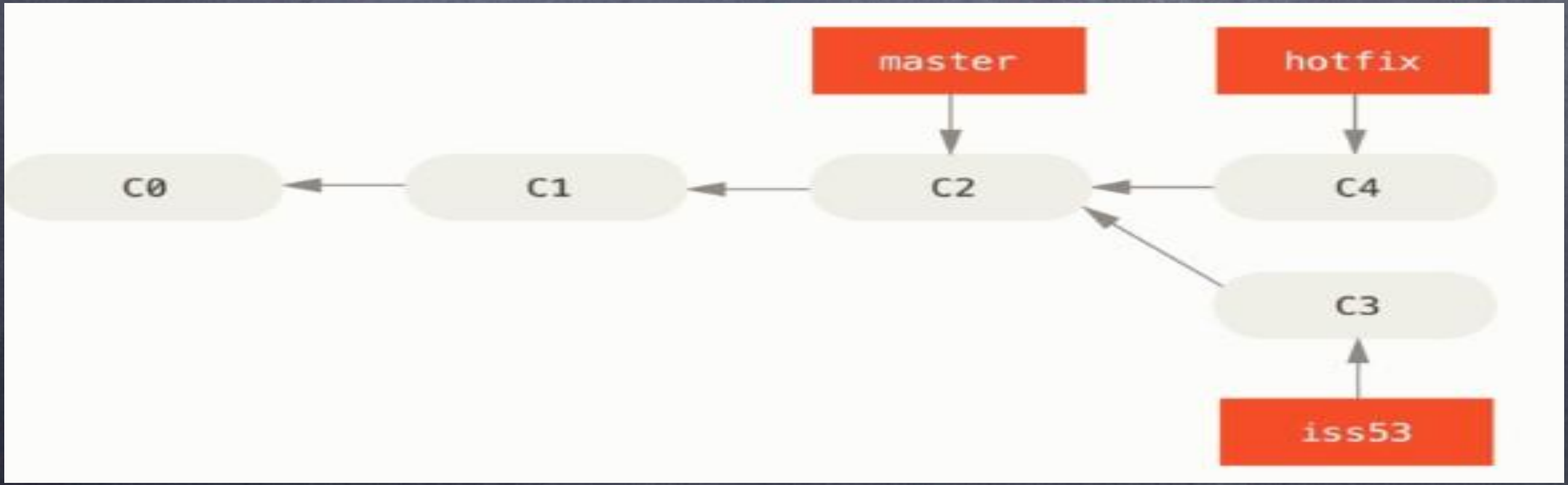


Branching and Merging(cont.)

- let’s assume you’ve committed all your changes, so you can switch back to your master branch:

```
$ git checkout master
```

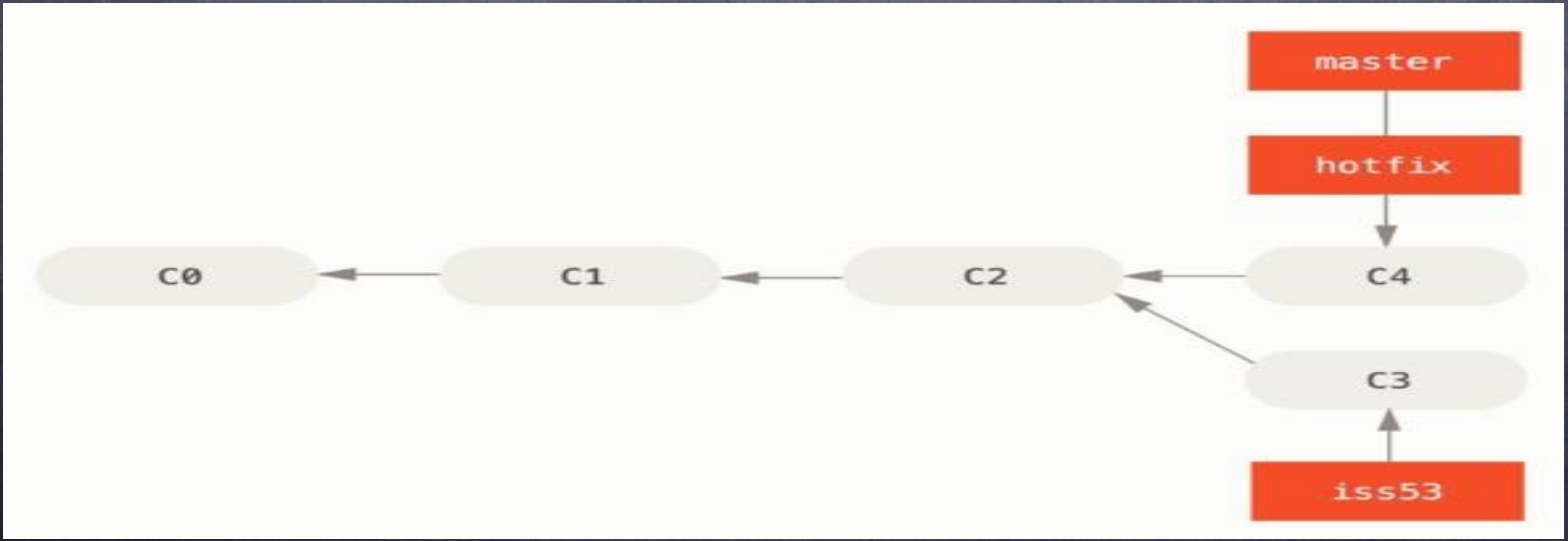
- Let’s create a hotfix branch on which to work until it’s completed:



Branching and Merging(cont.)

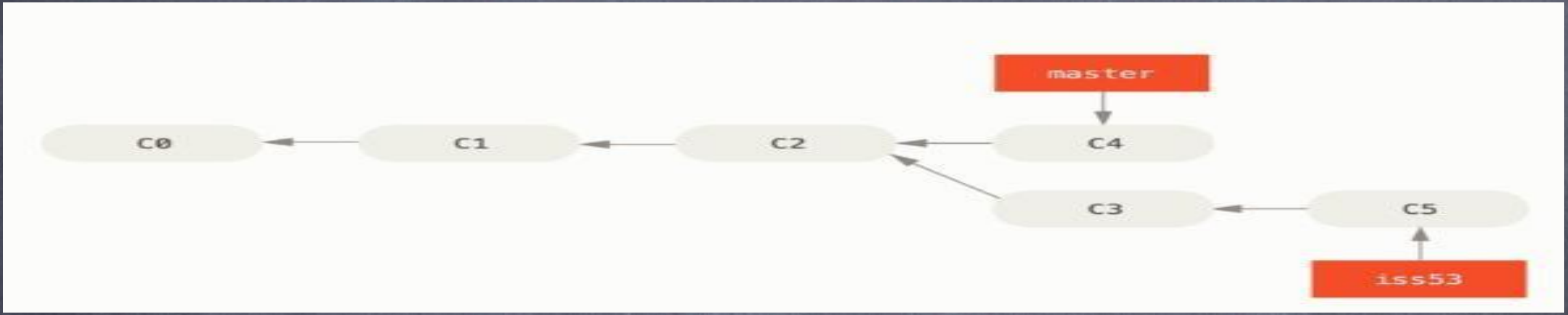
You can run your tests, make sure the hotfix is what you want, and finally merge the hotfix branch back into your master branch to deploy to production. You do this with the git merge command:

```
$ git checkout master
$ git merge hotfix
```

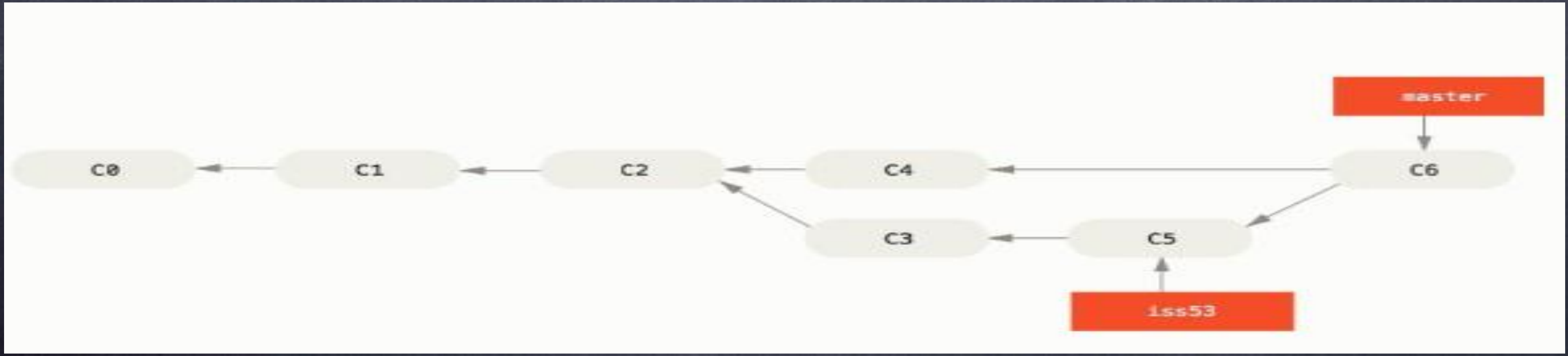


Branching and Merging(cont.)

Now you can switch back to your work-in-progress branch on issue #53 and continue working on it.



Suppose you’ve decided that your issue #53 work is complete and ready to be merged into your master branch



Branching and Merging(cont.)

- To delete the "iss53" branch

```
$ git branch -d iss53
```

- To force delete the " iss53 " branch

```
$ git branch -D iss53
```

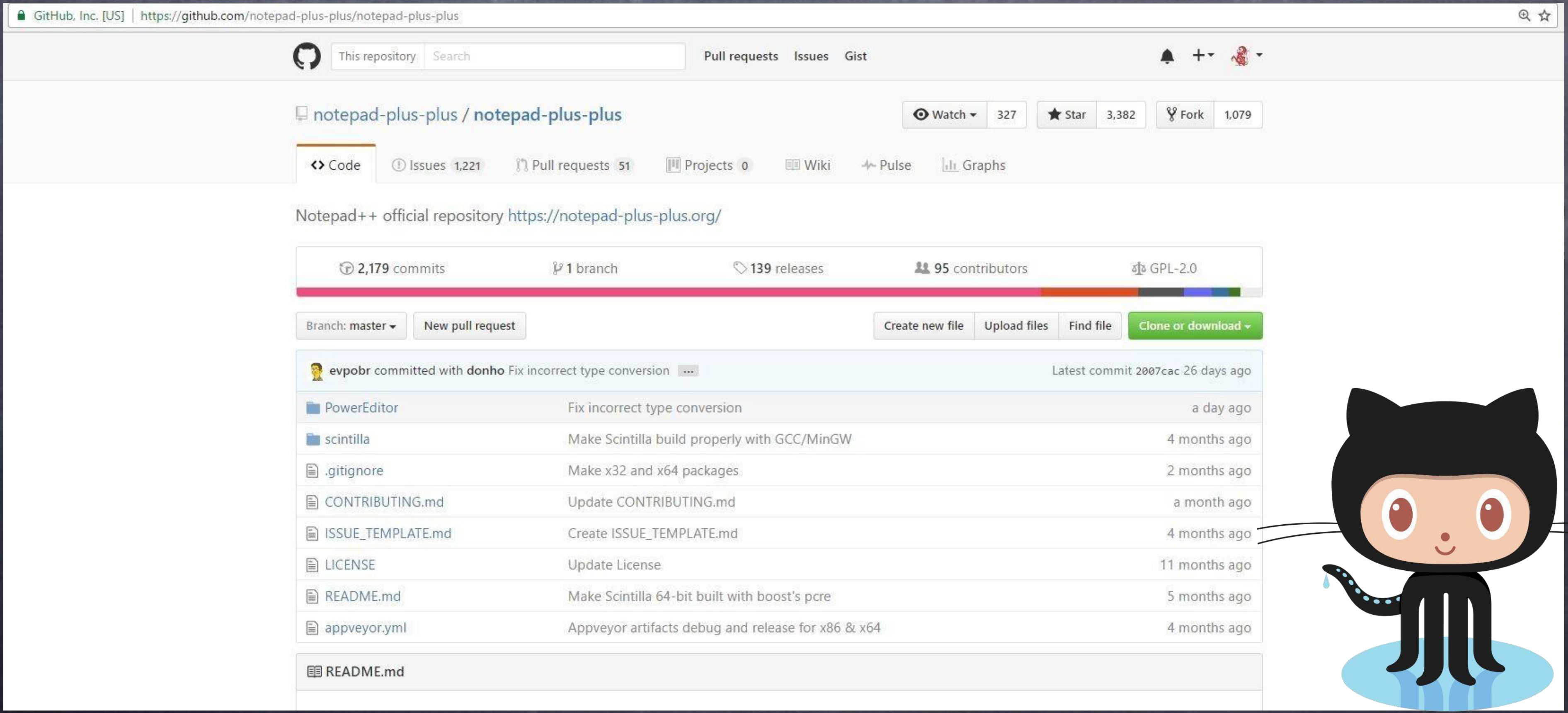
- To display all commits as graph

```
$ git log --oneline --graph --decorate --all
```

Github

Github

- GitHub is a web-based Git repository hosting service
- A remote repository is a repository that's just like the one you're using but it's just stored at a different location


GitHub, Inc. [US] | https://github.com/notepad-plus-plus/notepad-plus-plus

This repository | Search | Pull requests | Issues | Gist

notepad-plus-plus / notepad-plus-plus

Watch 327 | Star 3,382 | Fork 1,079

Code | Issues 1,221 | Pull requests 51 | Projects 0 | Wiki | Pulse | Graphs

Notepad++ official repository https://notepad-plus-plus.org/

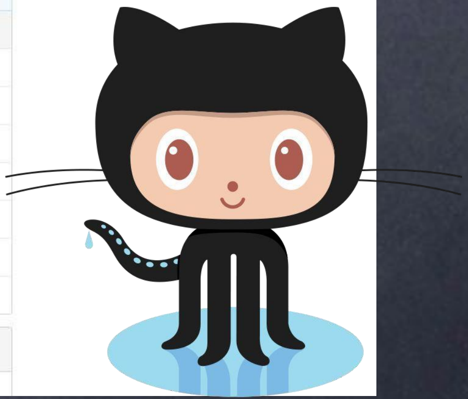
2,179 commits | 1 branch | 139 releases | 95 contributors | GPL-2.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

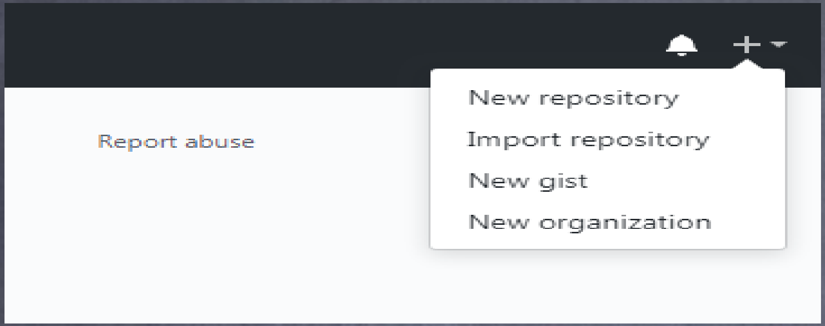
evpobr committed with donho Fix incorrect type conversion ... Latest commit 2007cac 26 days ago

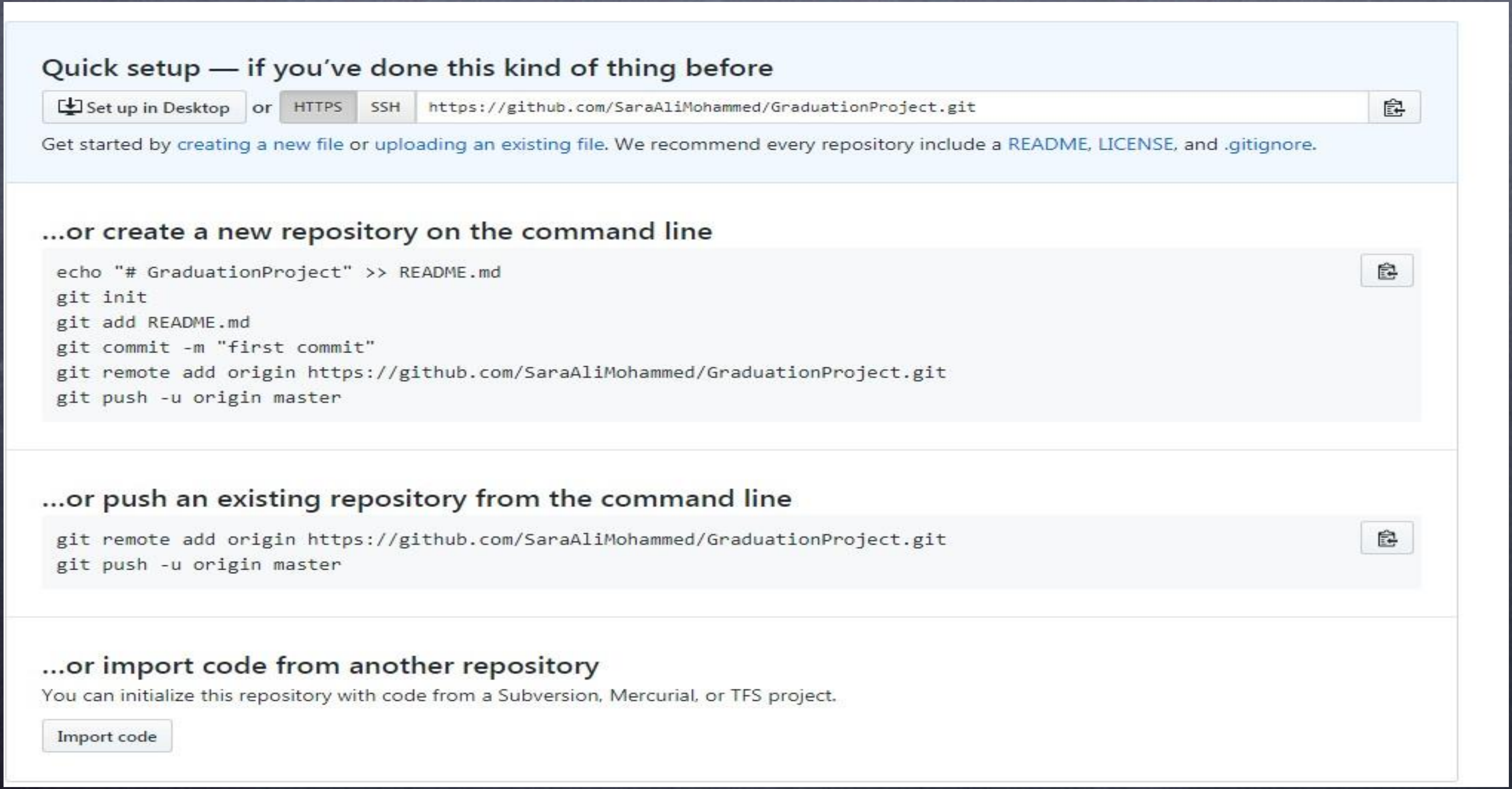
PowerEditor	Fix incorrect type conversion	a day ago
scintilla	Make Scintilla build properly with GCC/MinGW	4 months ago
.gitignore	Make x32 and x64 packages	2 months ago
CONTRIBUTING.md	Update CONTRIBUTING.md	a month ago
ISSUE_TEMPLATE.md	Create ISSUE_TEMPLATE.md	4 months ago
LICENSE	Update License	11 months ago
README.md	Make Scintilla 64-bit built with boost's pcre	5 months ago
appveyor.yml	Appveyor artifacts debug and release for x86 & x64	4 months ago

README.md



Github





Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/SaraAliMohammed/GraduationProject.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# GraduationProject" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/SaraAliMohammed/GraduationProject.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/SaraAliMohammed/GraduationProject.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

Github

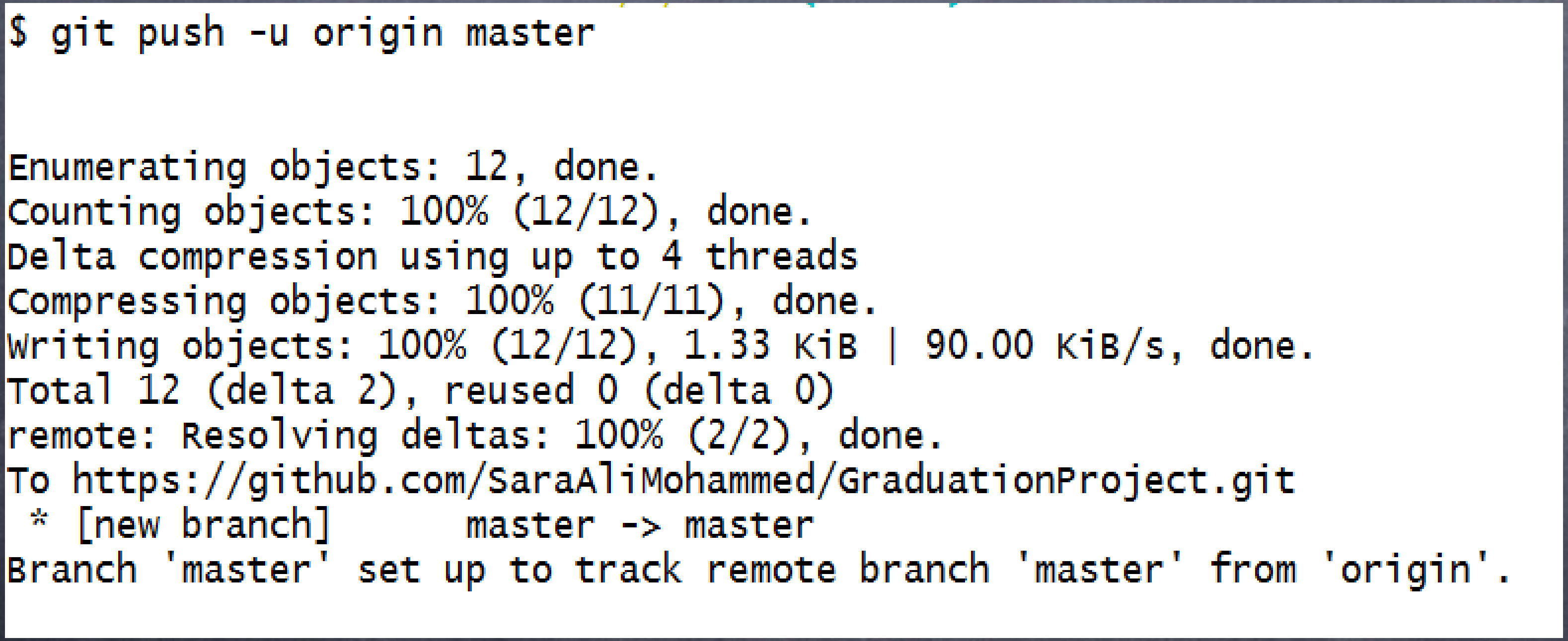
- To manage a remote repository, use the git remote command:
  $ git remote

- To add a connection to a new remote repository:
  $ git remote add origin linkOnGithub

- To see the details about a connection to a remote:
  $ git remote –v

Github

- The git push command is used to send commits from a local repository to a remote repository.
  $ **git push -u origin master**



```
$ git push -u origin master


Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 4 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (12/12), 1.33 KiB | 90.00 KiB/s, done.
Total 12 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/SaraAliMohammed/GraduationProject.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Github

- If there are changes in a remote repository that you'd like to include in your local repository:

```
$ git pull origin master
```

Q&A