ARBEITSKREIS
OPEN SOURCE SOFTWARE

Lars Quentin

Introduction to Git

How to share code and collaborate with others!

Table of contents

1 Motivation

2 Theory

3 Getting Started

4 GitHub GUI

5 Advanced

6 Conclusion

Goals

- Understanding git and its usages
- How to
  - Start a local repository
  - Manage and Commit updates
  - Sync between local and remote
  - Manage branches and merge conflicts
- Be able to use git enough for the next project

Original Problem

How did people work together on code before?

- How to make sure they weren't interefering each other:
  1. Sending updated source code archives

Original Problem

How did people work together on code before?

- How to make sure they weren't interefering each other:
  1. Sending updated source code archives
  2. Shared Directory and file locks

Original Problem

How did people work together on code before?

- How to make sure they weren't interefering each other:
  1. Sending updated source code archives
  2. Shared Directory and file locks
  3. Shared Directory and luck

Original Problem

How did people work together on code before?

- How to make sure they weren't interefering each other:
  1. Sending updated source code archives
  2. Shared Directory and file locks
  3. Shared Directory and luck
- Code Backups were done manually

Original Problem

How did people work together on code before?

- How to make sure they weren't interefering each other:
  1. Sending updated source code archives
  2. Shared Directory and file locks
  3. Shared Directory and luck
- Code Backups were done manually
- Problems with that approach:
  - If shared directory, they can overwrite it accidentally

Original Problem

How did people work together on code before?

- How to make sure they weren't interefering each other:
  1. Sending updated source code archives
  2. Shared Directory and file locks
  3. Shared Directory and luck
- Code Backups were done manually
- Problems with that approach:
  - If shared directory, they can overwrite it accidentally
  - Local versons were vastly different, hard to merge together

Original Problem

How did people work together on code before?

- How to make sure they weren't interefering each other:
  1. Sending updated source code archives
  2. Shared Directory and file locks
  3. Shared Directory and luck
- Code Backups were done manually
- Problems with that approach:
  - If shared directory, they can overwrite it accidentally
  - Local versons were vastly different, hard to merge together
  - Everything relied on a lot of communication and manual work.

Solution: Git

- Git is a **distributed version control system** (**VCS**)
- Initially developed for the Linux kernel
- Bundles set of changes into named updates, called **commits**
- People can create their own updates, **branching** out
- Allows for huge collaboration
  - Linux has over 1400 contributors! [1]

Figure: Git Logo [2]

Git is not GitHub

Figure: GitHub Logo [3]

- Git is a program for versioning
- GitHub is a website that hosts Git projects
- Git is not made by GitHub
- Analogy: E-Mail
  - Outlook (the program) is an Email-Client
  - Google Mail is a Email hoster
  - Outlook is not made by Google!

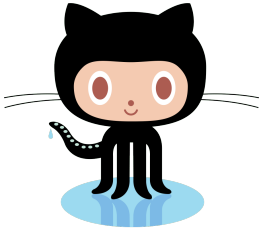


Figure: GitHub Mascot: Octocat [4]

Why to use Git

- Collaborate with others

Why to use Git

- Collaborate with others
- Manage multiple, actively worked on versions
  - 2 people can't trivially write in the same file!

Why to use Git

- Collaborate with others
- Manage multiple, actively worked on versions
  - 2 people can't trivially write in the same file!
- Version your code, take more risks, roll back mistakes!

Why to use Git

- Collaborate with others
- Manage multiple, actively worked on versions
  - 2 people can't trivially write in the same file!
- Version your code, take more risks, roll back mistakes!
- Make your code more discoverable
  - It's common to Google: "`<MY PROBLEM> github`"
  - Better discoverability than personal website

Why to use Git

- Collaborate with others
- Manage multiple, actively worked on versions
  - 2 people can't trivially write in the same file!
- Version your code, take more risks, roll back mistakes!
- Make your code more discoverable
  - It's common to Google: "`<MY PROBLEM> github`"
  - Better discoverability than personal website
- Use GitHub/Gitlab as an portfolio

How does Git work?

- Git projects are called **repositories** or **repos**
- There are 2 ways to create a Git repository
  - **Initialize** a new folder (Create)
  - **Clone** an existing repo (Download)
- This means that it is **local** on your device
  - Just a normal folder you can work in with any tools!
- Once you finished something, you can bundle it into an update
  - A so-called **commit**

What does **Distributed** mean?

- Once you **initialize** or **clone** the repo, it is local on the device.
  - You do *not* work on the remote server!

What does **Distributed** mean?

- Once you **initialize** or **clone** the repo, it is local on the device.
  - You do *not* work on the remote server!
- Every developer has its local version
  - It doesn't change automatically!

What does **Distributed** mean?

- Once you **initialize** or **clone** the repo, it is local on the device.
  - You do *not* work on the remote server!
- Every developer has its local version
  - It doesn't change automatically!
- Instead, one can manually
  - **Pull** the newest commits from the server
  - **Push** the local commits to the server

What does **Distributed** mean?

- Once you **initialize** or **clone** the repo, it is local on the device.
  - You do *not* work on the remote server!
- Every developer has its local version
  - It doesn't change automatically!
- Instead, one can manually
  - **Pull** the newest commits from the server
  - **Push** the local commits to the server

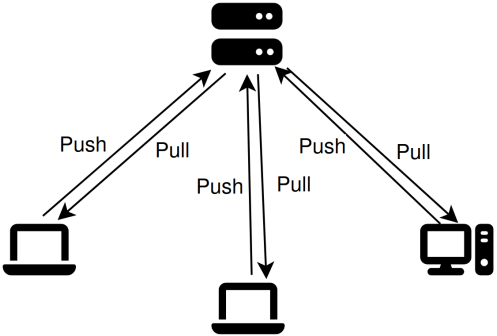


Figure: Every computer has a local version.

About Commits

Git is (mainly) for text files!

- Because it tracks changes line by line

About Commits

Git is (mainly) for text files!

- Because it tracks changes line by line
- The following are **NOT** text files:
  - Word files (`.docx`)

About Commits

Git is (mainly) for text files!

- Because it tracks changes line by line
- The following are **NOT** text files:
  - Word files (`.docx`)
  - PDFs

About Commits

Git is (mainly) for text files!

- Because it tracks changes line by line
- The following are **NOT** text files:
  - Word files (`.docx`)
  - PDFs
  - Audio, Video, Pictures...

About Commits

Git is (mainly) for text files!

- Because it tracks changes line by line
- The following are **NOT** text files:
  - Word files (`.docx`)
  - PDFs
  - Audio, Video, Pictures...
- Non text files can be put into git
  - Fullly replaced everytime!

About Commits

Git is (mainly) for text files!

- Because it tracks changes line by line
- The following are **NOT** text files:
  - Word files (`.docx`)
  - PDFs
  - Audio, Video, Pictures...
- Non text files can be put into git
  - Fullly replaced everytime!
- A commit is the **difference** in lines
  - Called a **diff**

About Commits

Git is (mainly) for text files!

- Because it tracks changes line by line
- The following are **NOT** text files:
  - Word files (`.docx`)
  - PDFs
  - Audio, Video, Pictures...
- Non text files can be put into git
  - Fullly replaced everytime!
- A commit is the **difference** in lines
  - Called a **diff**

```
> git diff
diff --git a/example.md b/example.md
index 02e444f..75a137d 100644
--- a/example.md
+++ b/example.md
@@ -1,10 +1,10 @@
 This is an example document

 This line was not touched
-This line was deleted
 Another unchanged line
-THIS line WAS changed
+This line was changed
+This line was added
 ```
 def addition(a,b):
-   return a-b # Wrong!
+   return a+b # Correct!
 ```
```

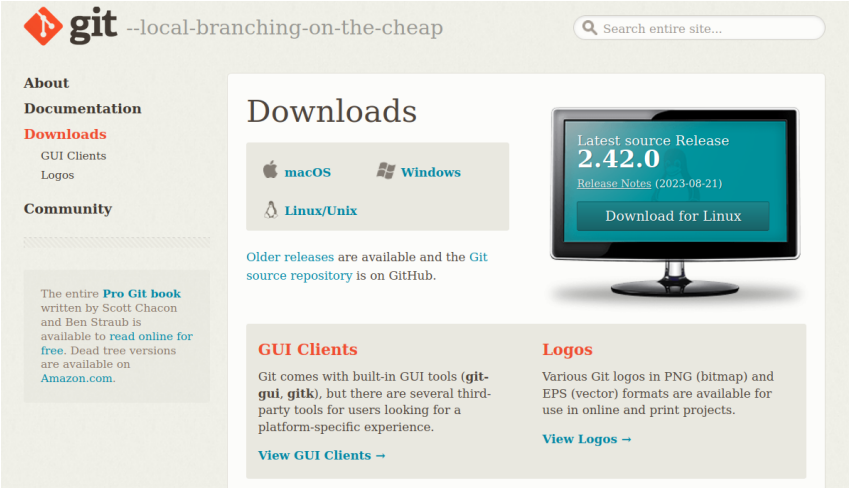
Figure: Red is deleted, green is added

About Commits (cont.)

When should you commit

- If you can describe what you have done.
  - Think of an experiment log.
  - "I am currently filling the 41st ml into this flask!"
- Why do we commit:
  - Better understanding for others
  - Better understanding for our future self

Installing Git

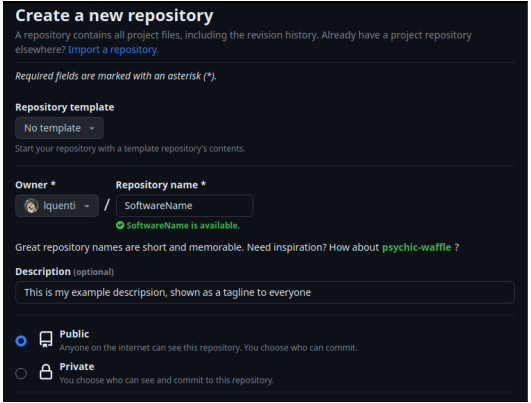


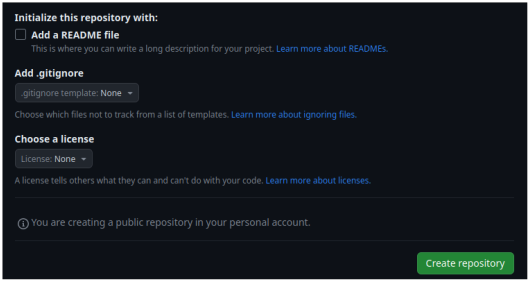
Initial configuration

Before starting, we have to do the following:

- Check whether it is installed
- Set an author and Email adress
- Configure an SSH key (CLI only)

Create Repository (GitHub)





Link: `https://github.com/new`

Starting a local repository

```
lquenti@lquenti-Latitude-7420:~/example$ ls
code.c  data.csv  File1.md  LICENSE
```

Starting a local repository

```
lquenti@lquenti-Latitude-7420:~/example$ ls
code.c  data.csv  File.md  LICENSE
lquenti@lquenti-Latitude-7420:~/example$ git init
Initialized empty Git repository in /home/lquenti/example/.git/
```

Starting a local repository

```
lquenti@lquenti-Latitude-7420:~/example$ ls
code.c  data.csv  File1.md  LICENSE
lquenti@lquenti-Latitude-7420:~/example$ git init
Initialized empty Git repository in /home/lquenti/example/.git/
lquenti@lquenti-Latitude-7420:~/example$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        File1.md
        LICENSE
        code.c
        data.csv

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

Starting a local repository

```
lquenti@lquenti-Latitude-7420:~/example$ ls
code.c  data.csv  File a.md  LICENSE
lquenti@lquenti-Latitude-7420:~/example$ git init
Initialized empty Git repository in /home/lquenti/example/.git/
lquenti@lquenti-Latitude-7420:~/example$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        File a.md
        LICENSE
        code.c
        data.csv

nothing added to commit but untracked files present (use "git add" to track)
lquenti@lquenti-Latitude-7420:~/example$ git add code.c
```

Starting a local repository

```
lquenti@lquenti-Latitude-7420:~/example$ ls
code.c  data.csv  Filea.md  LICENSE
lquenti@lquenti-Latitude-7420:~/example$ git init
Initialized empty Git repository in /home/lquenti/example/.git/
lquenti@lquenti-Latitude-7420:~/example$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Filea.md
        LICENSE
        code.c
        data.csv

nothing added to commit but untracked files present (use "git add" to track)
lquenti@lquenti-Latitude-7420:~/example$ git add code.c
lquenti@lquenti-Latitude-7420:~/example$ git add data.csv
```

Starting a local repository

```
lquenti@lquenti-Latitude-7420:~/example$ ls
code.c  data.csv  File a.md  LICENSE
lquenti@lquenti-Latitude-7420:~/example$ git init
Initialized empty Git repository in /home/lquenti/example/.git/
lquenti@lquenti-Latitude-7420:~/example$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        File a.md
        LICENSE
        code.c
        data.csv

nothing added to commit but untracked files present (use "git add" to track)
lquenti@lquenti-Latitude-7420:~/example$ git add code.c
lquenti@lquenti-Latitude-7420:~/example$ git add data.csv
```

```
lquenti@lquenti-Latitude-7420:~/example$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   code.c
        new file:   data.csv

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

Starting a local repository

```
lquenti@lquenti-Latitude-7420:~/example$ ls
code.c  data.csv  File a.md  LICENSE
lquenti@lquenti-Latitude-7420:~/example$ git init
Initialized empty Git repository in /home/lquenti/example/.git/
lquenti@lquenti-Latitude-7420:~/example$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        File a.md
        LICENSE
        code.c
        data.csv

nothing added to commit but untracked files present (use "git add" to track)
lquenti@lquenti-Latitude-7420:~/example$ git add code.c
lquenti@lquenti-Latitude-7420:~/example$ git add data.csv
```

```
lquenti@lquenti-Latitude-7420:~/example$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   code.c
        new file:   data.csv

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

lquenti@lquenti-Latitude-7420:~/example$ git rm --cached code.c
rm 'code.c'
```

Starting a local repository

```
lquenti@lquenti-Latitude-7420:~/example$ ls
code.c  data.csv  File a.md  LICENSE
lquenti@lquenti-Latitude-7420:~/example$ git init
Initialized empty Git repository in /home/lquenti/example/.git/
lquenti@lquenti-Latitude-7420:~/example$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        File a.md
        LICENSE
        code.c
        data.csv

nothing added to commit but untracked files present (use "git add" to track)
lquenti@lquenti-Latitude-7420:~/example$ git add code.c
lquenti@lquenti-Latitude-7420:~/example$ git add data.csv
```

```
lquenti@lquenti-Latitude-7420:~/example$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   code.c
        new file:   data.csv

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

lquenti@lquenti-Latitude-7420:~/example$ git rm --cached code.c
rm 'code.c'
lquenti@lquenti-Latitude-7420:~/example$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   data.csv

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        File a.md
        LICENSE
        code.c
```

Creating a commit

```
lquenti@lquenti-Latitude-7420:~/example$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   data.csv

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Filea.md
        LICENSE
        code.c
```

Creating a commit

```
lquenti@lquenti-Latitude-7420:~/example$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   data.csv

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

lquenti@lquenti-Latitude-7420:~/example$ git commit -m "Adding CSV data"
[main (root-commit) ed02d32] Adding CSV data
 1 file changed, 11 insertions(+)
 create mode 100644 data.csv
```

Creating a commit

```
lquenti@lquenti-Latitude-7420:~/example$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   data.csv

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

lquenti@lquenti-Latitude-7420:~/example$ git commit -m "Adding CSV data"
[main (root-commit) ed02d32] Adding CSV data
 1 file changed, 11 insertions(+)
 create mode 100644 data.csv
lquenti@lquenti-Latitude-7420:~/example$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Filea.md
        LICENSE
        code.c

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

Creating a commit

```
lquenti@lquenti-Latitude-7420:~/example$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   data.csv

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

lquenti@lquenti-Latitude-7420:~/example$ git commit -m "Adding CSV data"
[main (root-commit) ed02d32] Adding CSV data
 1 file changed, 11 insertions(+)
 create mode 100644 data.csv
lquenti@lquenti-Latitude-7420:~/example$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Filea.md
        LICENSE
        code.c

nothing added to commit but untracked files present (use "git add" to track)
lquenti@lquenti-Latitude-7420:~/example$ git log --oneline
ed02d32 (HEAD -> main) Adding CSV data
```

Push and pull updates

```
128 lquenti@lquenti-Latitude-7420:~/example$ git log --oneline
ed02d32 (HEAD -> main) Adding CSV data
```

Creating a commit

```
128 lquenti@lquenti-Latitude-7420:~/example$ git log --oneline
ed02d32 (HEAD -> main) Adding CSV data
lquenti@lquenti-Latitude-7420:~/example$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 360 bytes | 360.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:lquenti/example.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Creating a commit

```
128 lquenti@lquenti-Latitude-7420:~/example$ git log --oneline
ed02d32 (HEAD -> main) Adding CSV data
lquenti@lquenti-Latitude-7420:~/example$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 360 bytes | 360.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:lquenti/example.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
lquenti@lquenti-Latitude-7420:~/example$ # Some change from someone else
```

Creating a commit

```
128 lquenti@lquenti-Latitude-7420:~/example$ git log --oneline
ed02d32 (HEAD -> main) Adding CSV data
lquenti@lquenti-Latitude-7420:~/example$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 360 bytes | 360.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:lquenti/example.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
lquenti@lquenti-Latitude-7420:~/example$ # Some change from someone else
lquenti@lquenti-Latitude-7420:~/example$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 696 bytes | 696.00 KiB/s, done.
From github.com:lquenti/example
   ed02d32..72c5177  main       -> origin/main
Updating ed02d32..72c5177
Fast-forward
 README.md | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 README.md
```

Creating a commit

```
128 lquenti@lquenti-Latitude-7420:~/example$ git log --oneline
ed02d32 (HEAD -> main) Adding CSV data
lquenti@lquenti-Latitude-7420:~/example$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 360 bytes | 360.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:lquenti/example.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
lquenti@lquenti-Latitude-7420:~/example$ # Some change from someone else
lquenti@lquenti-Latitude-7420:~/example$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 696 bytes | 696.00 KiB/s, done.
From github.com:lquenti/example
   ed02d32..72c5177  main       -> origin/main
Updating ed02d32..72c5177
Fast-forward
 README.md | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 README.md
lquenti@lquenti-Latitude-7420:~/example$ git log --oneline
72c5177 (HEAD -> main, origin/main) Create README.md
ed02d32 Adding CSV data
```

Cloning a remote repository

```
lquenti@lquenti-Latitude-7420:~$ git clone git@github.com:torvalds/linux.git
Cloning into 'linux'...
remote: Enumerating objects: 9719232, done.
remote: Counting objects: 100% (238/238), done.
remote: Compressing objects: 100% (143/143), done.
remote: Total 9719232 (delta 160), reused 129 (delta 95), pack-reused 9718994
Receiving objects: 100% (9719232/9719232), 4.47 GiB | 12.45 MiB/s, done.
Resolving deltas: 100% (7949442/7949442), done.
Updating files: 100% (81756/81756), done.
```

Cloning a remote repository

```
lquenti@lquenti-Latitude-7420:~$ git clone git@github.com:torvalds/linux.git
Cloning into 'linux'...
remote: Enumerating objects: 9719232, done.
remote: Counting objects: 100% (238/238), done.
remote: Compressing objects: 100% (143/143), done.
remote: Total 9719232 (delta 160), reused 129 (delta 95), pack-reused 9718994
Receiving objects: 100% (9719232/9719232), 4.47 GiB | 12.45 MiB/s, done.
Resolving deltas: 100% (7949442/7949442), done.
Updating files: 100% (81756/81756), done.
lquenti@lquenti-Latitude-7420:~$ cd linux/
CREDITS  LICENSES  Kbuild         README    MAINTAINERS  certs  crypto   fs    include  io
COPYING  Kconfig   Documentation  Makefile  arch         block  drivers  init  ipc      ke
```

Cloning a remote repository

```
lquenti@lquenti-Latitude-7420:~$ git clone git@github.com:torvalds/linux.git
Cloning into 'linux'...
remote: Enumerating objects: 9719232, done.
remote: Counting objects: 100% (238/238), done.
remote: Compressing objects: 100% (143/143), done.
remote: Total 9719232 (delta 160), reused 129 (delta 95), pack-reused 9718994
Receiving objects: 100% (9719232/9719232), 4.47 GiB | 12.45 MiB/s, done.
Resolving deltas: 100% (7949442/7949442), done.
Updating files: 100% (81756/81756), done.
lquenti@lquenti-Latitude-7420:~$ cd linux/
CREDITS  LICENSES  Kbuild         README    MAINTAINERS  certs  crypto   fs    include  io
COPYING  Kconfig   Documentation  Makefile  arch         block  drivers  init  ipc      ke
lquenti@lquenti-Latitude-7420:~/linux$ vim ./fs/read_write.c
```

Cloning a remote repository

```
lquenti@lquenti-Latitude-7420:~$ git clone git@github.com:torvalds/linux.git
Cloning into 'linux'...
remote: Enumerating objects: 9719232, done.
remote: Counting objects: 100% (238/238), done.
remote: Compressing objects: 100% (143/143), done.
remote: Total 9719232 (delta 160), reused 129 (delta 95), pack-reused 9718994
Receiving objects: 100% (9719232/9719232), 4.47 GiB | 12.45 MiB/s, done.
Resolving deltas: 100% (7949442/7949442), done.
Updating files: 100% (81756/81756), done.
lquenti@lquenti-Latitude-7420:~$ cd linux/
CREDITS  LICENSES  Kbuild         README    MAINTAINERS  certs  crypto   fs    include  io
COPYING  Kconfig   Documentation  Makefile  arch         block  drivers  init  ipc      ke
lquenti@lquenti-Latitude-7420:~/linux$ vim ./fs/read_write.c
lquenti@lquenti-Latitude-7420:~/linux$ git status
On branch master
Your branch is up to date with 'origin/master'.

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:   fs/read_write.c

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

Cloning a remote repository

```
lquenti@lquenti-Latitude-7420:~$ git clone git@github.com:torvalds/linux.git
Cloning into 'linux'...
remote: Enumerating objects: 9719232, done.
remote: Counting objects: 100% (238/238), done.
remote: Compressing objects: 100% (143/143), done.
remote: Total 9719232 (delta 160), reused 129 (delta 95), pack-reused 9718994
Receiving objects: 100% (9719232/9719232), 4.47 GiB | 12.45 MiB/s, done.
Resolving deltas: 100% (7949442/7949442), done.
Updating files: 100% (81756/81756), done.
lquenti@lquenti-Latitude-7420:~$ cd linux/
CREDITS  LICENSES  Kbuild         README    MAINTAINERS  certs  crypto   fs    include  io
COPYING  Kconfig   Documentation  Makefile  arch         block  drivers  init  ipc      ke
lquenti@lquenti-Latitude-7420:~/linux$ vim ./fs/read_write.c
lquenti@lquenti-Latitude-7420:~/linux$ git status
On branch master
Your branch is up to date with 'origin/master'.

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:   fs/read_write.c

no changes added to commit (use "git add" and/or "git commit -a")
lquenti@lquenti-Latitude-7420:~/linux$ git diff
diff --git a/fs/read_write.c b/fs/read_write.c
index 4771701c896b..3bd3097d6df0 100644
--- a/fs/read_write.c
+++ b/fs/read_write.c
@@ -563,6 +563,9 @@ EXPORT_SYMBOL(kernel_write);

 ssize_t vfs_write(struct file *file, const char __user *buf, size_t count, loff_t *pos)
 {
+  /* This is the write function.
+   * I have nothing to add here tbh
+   */
```

Introduction to Git GUIs

Use a Git GUI?

- Pro
  - Flatter learning curve
  - Visual representation
  - Less memorization
- Contra
  - Less powerful
  - Slower for advanced tasks
  - Abstraction based vendor lock-in

While it is worthwhile to learn git, a GUI can help initially!

Introduction to Git GUIs

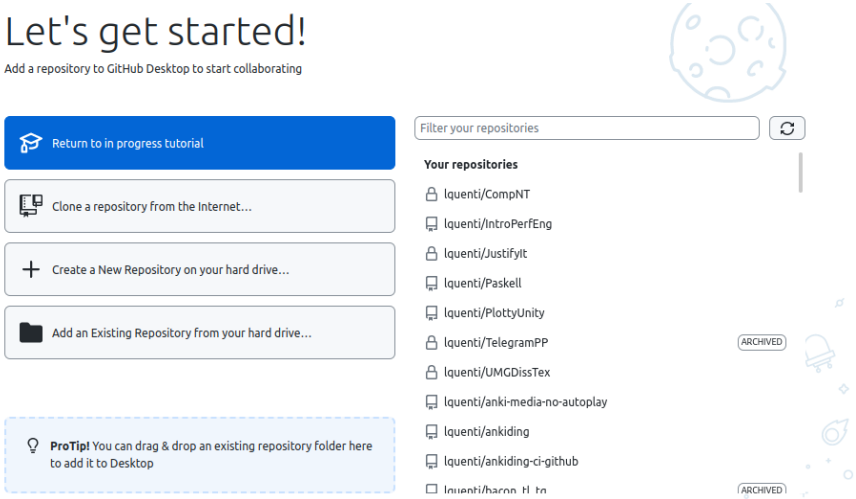
Use a Git GUI?

- Pro
  - Flatter learning curve
  - Visual representation
  - Less memorization
- Contra
  - Less powerful
  - Slower for advanced tasks
  - Abstraction based vendor lock-in

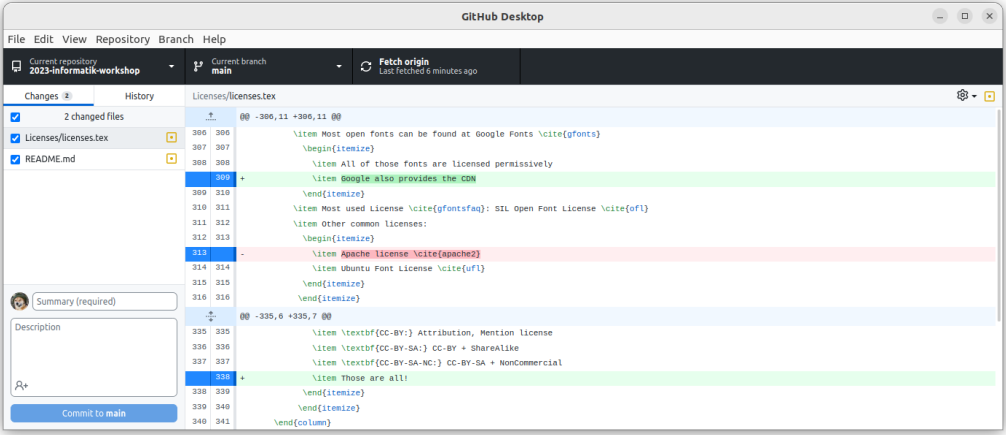
While it is worthwhile to learn git, a GUI can help initially!

GitHub GUI

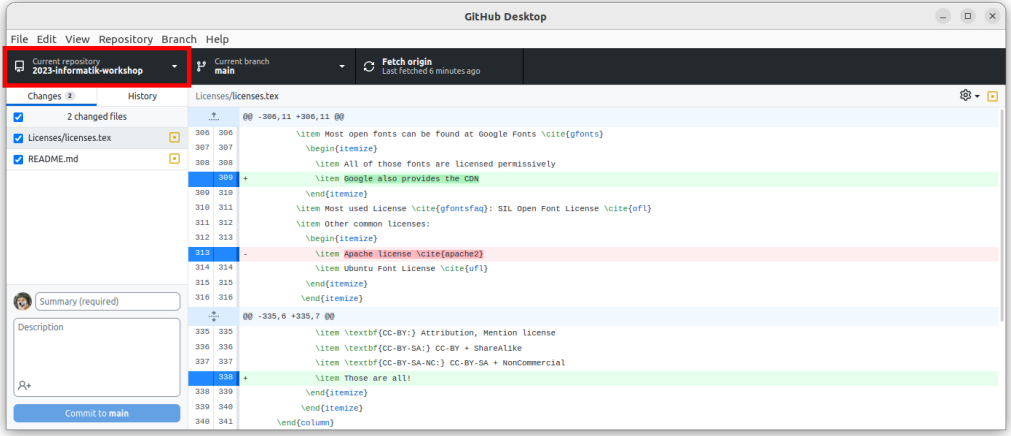
- Supports many git features
- Only good “multiplatform” standalone GUI client
  - Linux community-maintained
- Supports Non-Git GitHub features
  - Including CI/CD
- Syntax Highlighted Diffs
- Git Branch visualization


Let's get started!
Add a repository to GitHub Desktop to start collaborating
Return to in progress tutorial
Clone a repository from the Internet...
Create a New Repository on your hard drive...
Add an Existing Repository from your hard drive...
ProTip! You can drag & drop an existing repository folder here to add it to Desktop
Filter your repositories
Your repositories
lquenti/CompNT
lquenti/IntroPerfEng
lquenti/JustifyIt
lquenti/Paskell
lquenti/PlottyUnity
lquenti/TelegramPP
ARCHIVED
lquenti/UMGDissTex
lquenti/anki-media-no-autoplay
lquenti/ankiding
lquenti/ankiding-ci-github
ARCHIVED

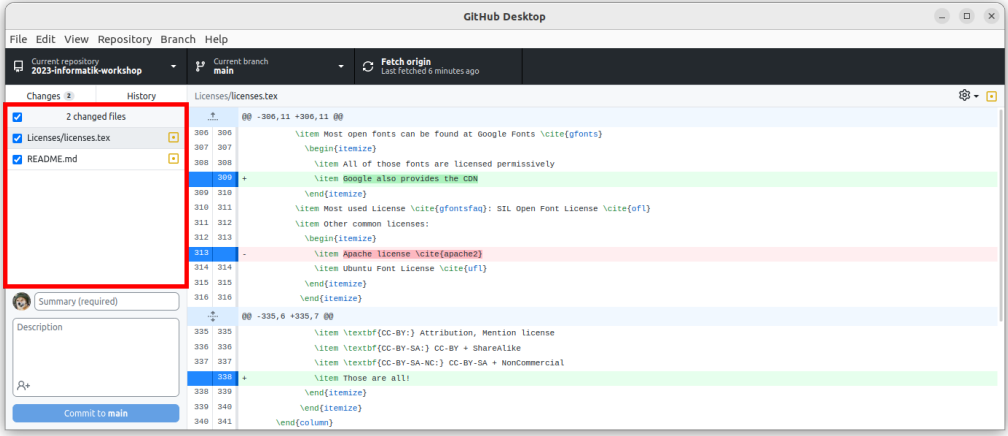
In a repository



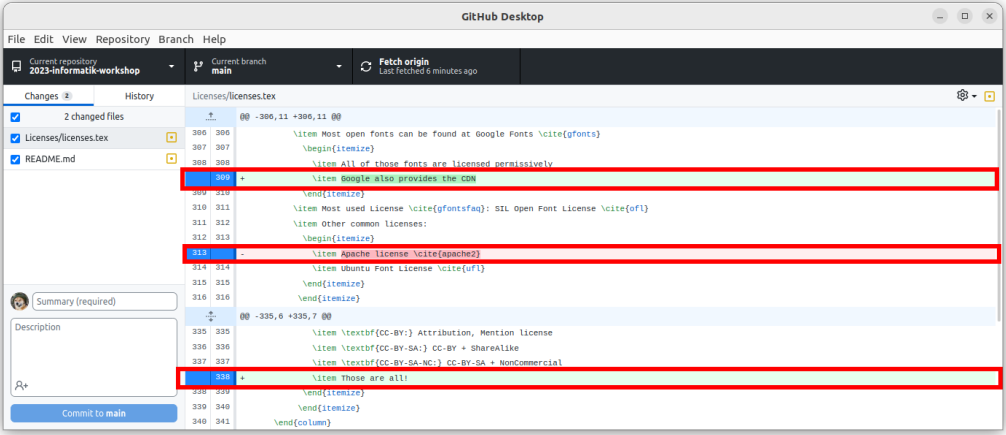
In a repository



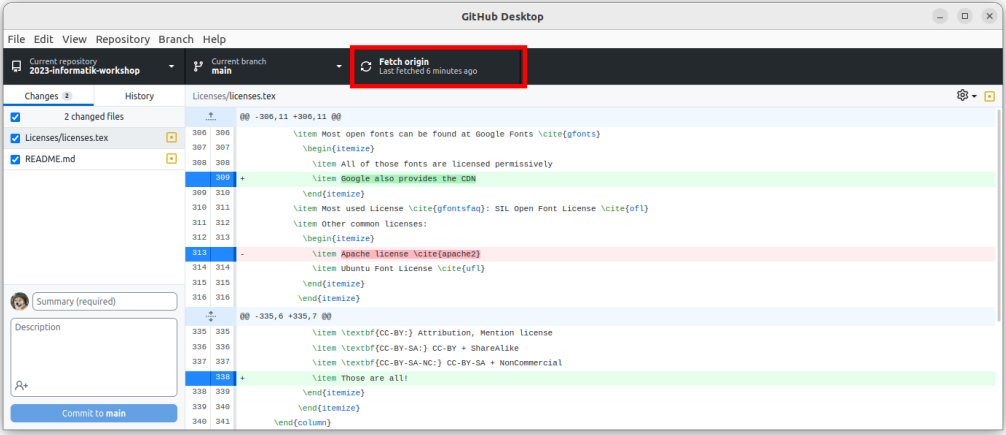
In a repository



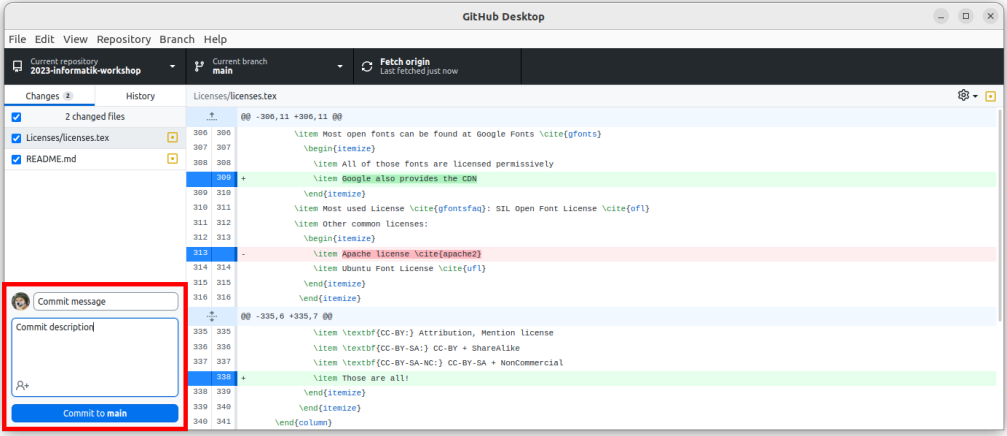
In a repository



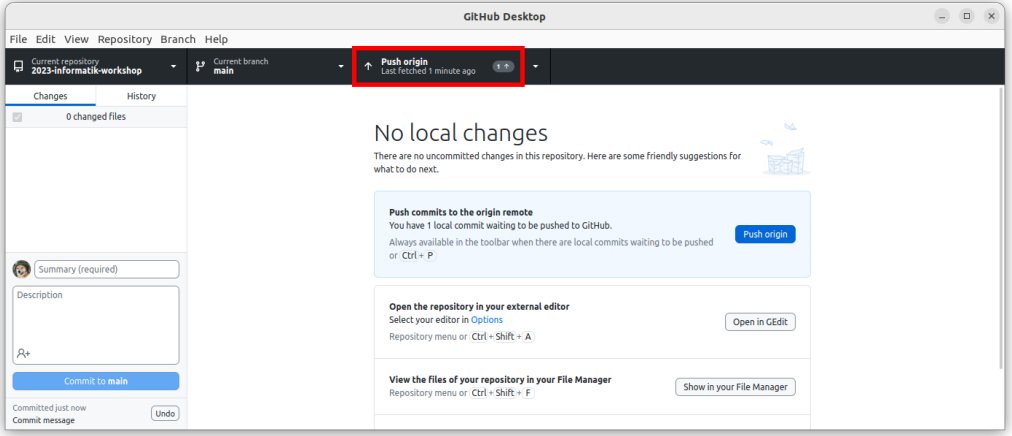
In a repository



Committing


GitHub Desktop
File Edit View Repository Branch Help
Current repository
2023-informatik-workshop
Current branch
main
Fetch origin
Last fetched just now
Changes 2
History
2 changed files
Licenses/licenses.tex
README.md
Commit message
Commit description
Commit to main
Licenses/licenses.tex
@@ -306,11 +306,11 @@
\item Most open fonts can be found at Google Fonts \cite{gfonts}
\begin{itemize}
\item All of those fonts are licensed permissively
+ \item Google also provides the CDN
\end{itemize}
\item Most used License \cite{gfontsfaq}: SIL Open Font License \cite{ofl}
\item Other common licenses:
\begin{itemize}
- \item Apache license \cite{apache2}
\item Ubuntu Font License \cite{ufl}
\end{itemize}
\end{itemize}
@@ -335,6 +335,7 @@
\item \textbf{CC-BY:} Attribution, Mention license
\item \textbf{CC-BY-SA:} CC-BY + ShareAlike
\item \textbf{CC-BY-SA-NC:} CC-BY-SA + NonCommercial
+ \item Those are all!
\end{itemize}
\end{itemize}
\end{column}

Pushing a Commit



Problem: Merge Conflicts

- Alice and Bob pull a project and work on it

Problem: Merge Conflicts

- Alice and Bob pull a project and work on it
- Alice changes `src/foo.c`

Problem: Merge Conflicts

- Alice and Bob pull a project and work on it
- Alice changes `src/foo.c`
- Alice commits and pushes her update

Problem: Merge Conflicts

- Alice and Bob pull a project and work on it
- Alice changes `src/foo.c`
- Alice commits and pushes her update
- Bob changes `src/foo.c`

Problem: Merge Conflicts

- Alice and Bob pull a project and work on it
- Alice changes `src/foo.c`
- Alice commits and pushes her update
- Bob changes `src/foo.c`
- Bob also commits and pushes his update

Problem: Merge Conflicts

- Alice and Bob pull a project and work on it
- Alice changes `src/foo.c`
- Alice commits and pushes her update
- Bob changes `src/foo.c`
- Bob also commits and pushes his update
- But Bob's version doesn't have Alice's update!

Solution: Branching

- Everybody uses their own **branch**
  - Often around a *feature*
- Everybody can work without problems on their own
- Branches then can get **merged** when done
- Extreme example: Linux Kernel 66-way merge

Solution: Branching

- Everybody uses their own **branch**
  - Often around a *feature*
- Everybody can work without problems on their own
- Branches then can get **merged** when done
- Extreme example: Linux Kernel 66-way merge

Advanced Features

Gitignore

- Often, temporary files are generated
  - log files
  - `node_modules`
  - `.DS_Store`, ...
- A **`.gitignore`** can list files to ignore by git
- GitHub provides templates for many languages!
  - https://github.com/github/gitignore

Advanced Features

Gitignore

- Often, temporary files are generated
  - log files
  - `node_modules`
  - `.DS_Store`, ...
- A **`.gitignore`** can list files to ignore by git
- GitHub provides templates for many languages!
  - https://github.com/github/gitignore

git blame

- Found a bug? Find out who did it.
- Maps each line to
  - The commit it was added
  - The author
- Support for most text editors!

Advanced Features: `git bisect`

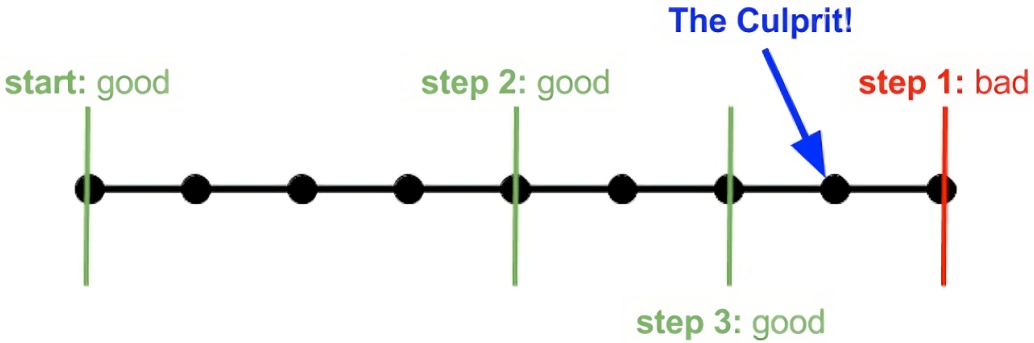


Figure: An example `bisect` [5]

Conclusion

Summary

- Git is a useful tool for code
- Starting a repo: `init` and `clone`
- Creating an update: `add`, `rm`, `commit`
- Updates: `push` and `pull`
- Branches and merges for collaboration

For many common problems:

`https://ohshitgit.com/`

References

Linux Contributor. SUSE Defines. URL: `https://www.suse.com/suse-defines/definition/linux-contributor/` (visited on 09/14/2023).

Git. URL: `https://git-scm.com/` (visited on 09/14/2023).

GitHub Logos and Usage. GitHub. URL: `https://github.com` (visited on 09/14/2023).

GitHub Octodex. URL: `https://octodex.github.com/` (visited on 09/14/2023).

Who Broke My Test? A Git Bisect Tutorial. Sumo Logic. URL: `https://www.sumologic.com/blog/who-broke-my-test-a-git-bisect-tutorial/` (visited on 09/25/2023).

Acknowledgements

- Icons provided by `font-awesome` under the OFL license