hello

welcome to **“Dive In with Git”**

Kumar **Ashwin**

SICSR ACM Students Chapter

$ whoami

Kumar **Ashwin**

- Security Consultation Intern @ Payatu
- Have been involved in tech communities like null and Developers Circle: Pune

- 0xCardinal.com
- **0xCardinal** on social platforms

Agenda

- What is Version Control?
- How git came into existence?
- Basics/Types of version control systems.
- Break – 5 mins
- Concepts involved in Git
- **Hands-On**
- Break – 5 mins
- Collaboration in Git
  - **Demo**
- Bonus Stuff
- Resources
- QnA

By the end of workshop,

- You will be able to create your project repositories and understand how to version control them.
- You will be able to contribute to other repositories or any Open Source Projects.

Let's start

What is Version Control?

Version control is "**practice of tracking and managing changes to software code**“ or any thing as such.

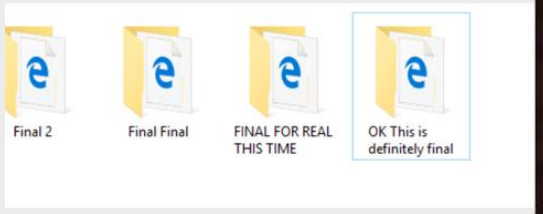
- Source Code Management
- Revision Control
- Branching and Merging

Types

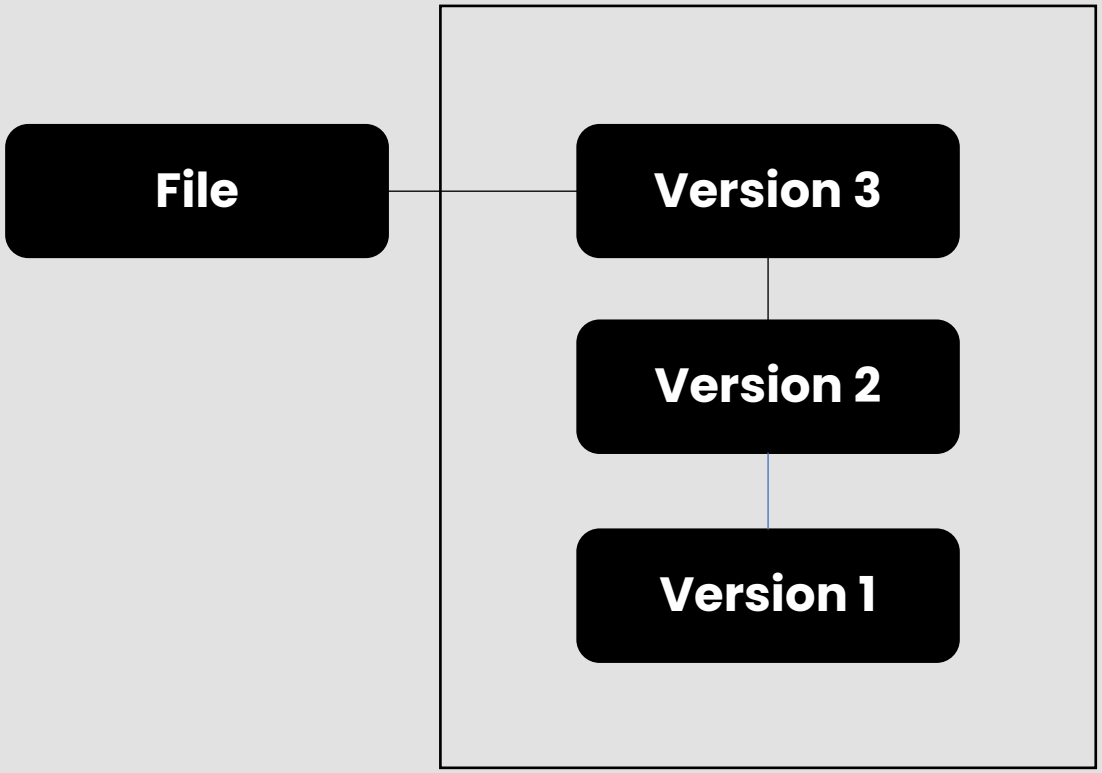
Distributed ·····

Centralized ·····

Local ·····



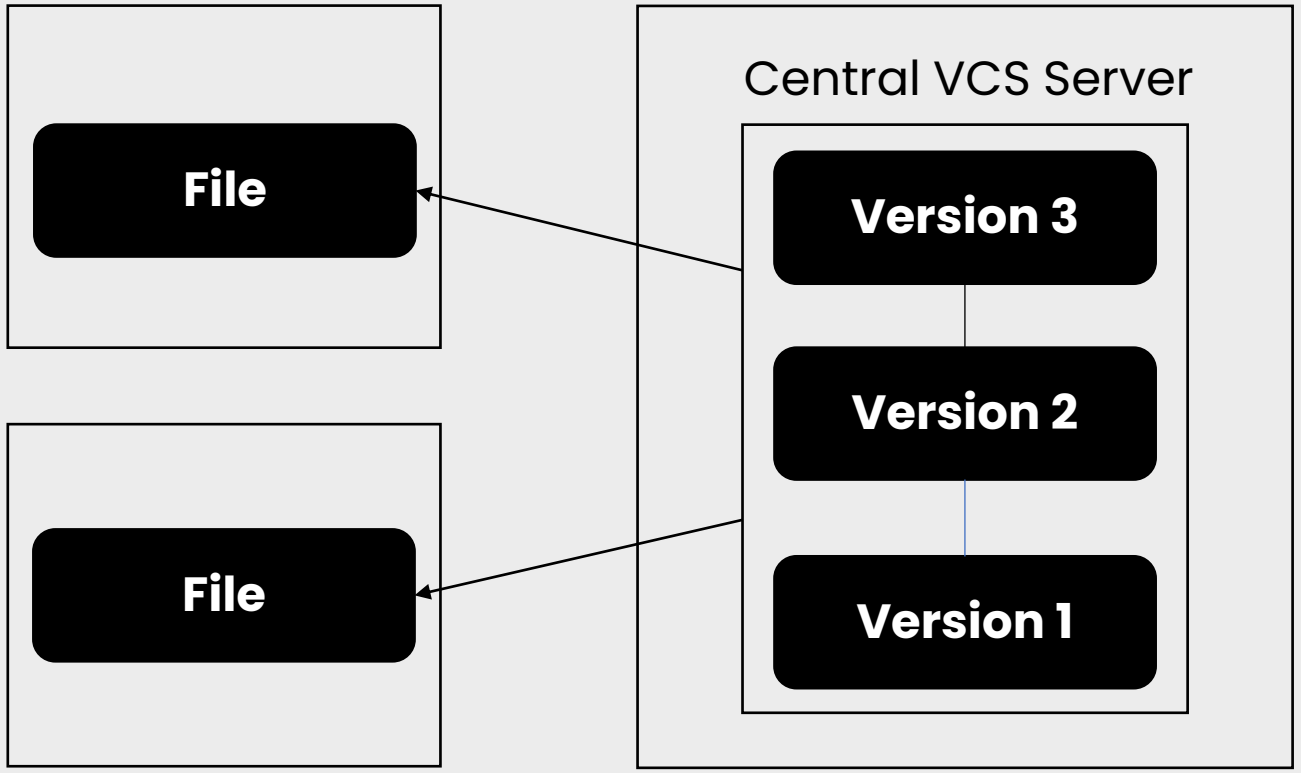
Types – 1/3 - **Local**



- Bad for collaboration
- Error prone

E.g., Local File System, etc.

Types – 2/3 - **Centralized**



"single server that contains all the versioned files"

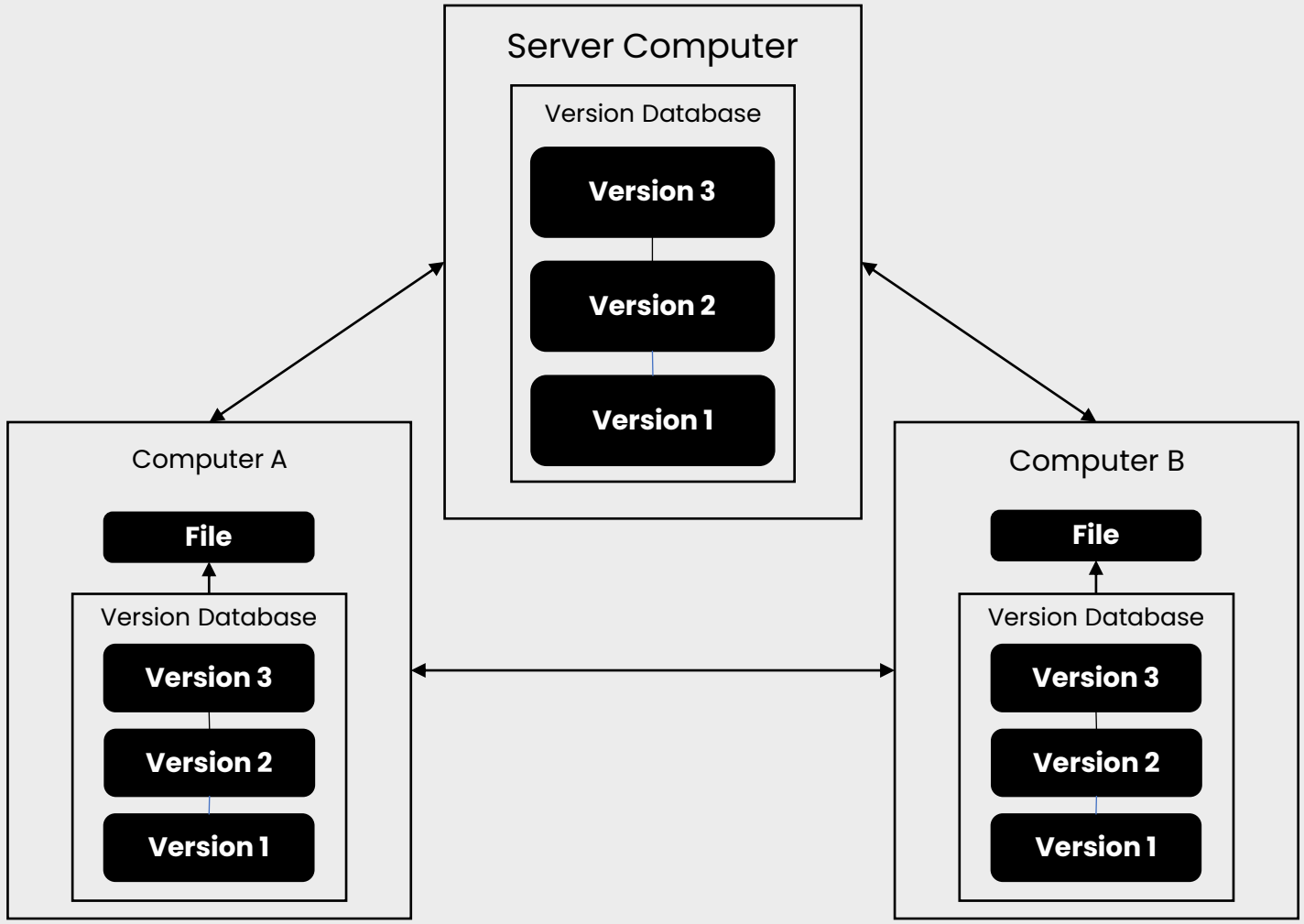
+ Fine grained control over repositories
+ Better for sharing and collaboration

but

- Server should always be kept in check, because if server goes down, workflow goes down

For E.g., SVN, etc.

Types – 3/3 - **Distributed**



+ every clone is a copy of the complete repository with all of its history.
+ Better for sharing and collaboration

For E.g., **Git**, Mercurial etc.

Why Git?

Because it is **“widely acceptable”**

How git came into existence?

Linux Kernel Community's Frustration with available VCS.

Linus Torvald – Creator of Git

Git (and its competitors) is sometimes categorized as a **version control system (VCS)**, sometimes a **source code management system (SCM),** and sometimes a **revision control system (RCS).** Torvalds thinks **life's too short** to distinguish between the definitions, so we won't.

Fun Fact | *In 2020, **Linux Kernel**'s Git Repo surpassed **1 million** commits.*

Starting with Git

Kumar Ashwin

Ashwin is a meticulous newbie in the world of cyber-security, crashing in with a brain focusing on capturing the flags. The young enthusiast believes in gaining knowledge by sharing it, simplicity being the key to interact with like-minded individuals. He contributes to communitites such as null and Developer Circle: Pune. He has been leading/managing null Study Groups for six different domains and have been working with Payatu as an intern for cyber security. He holds industry certifications such as CEH Practicals and MTA: Security Fundamentals. He often writes about his experience on his blog and his social footprints can be tracked on twitter mostly. He is surely the social being you will find at a snooker table dominated by a geeky conversation!

Setting up a repository

What is a GIT repository?

It is the project directory having a **`.git`** folder in it.

- Track and stores the information about all the past changes and commit along with all the configuration details.

```
My-awesome-website/
├── .git/
└── index.html
```

Setting up a repository

How to set-up a repository?

```
$ git init
```

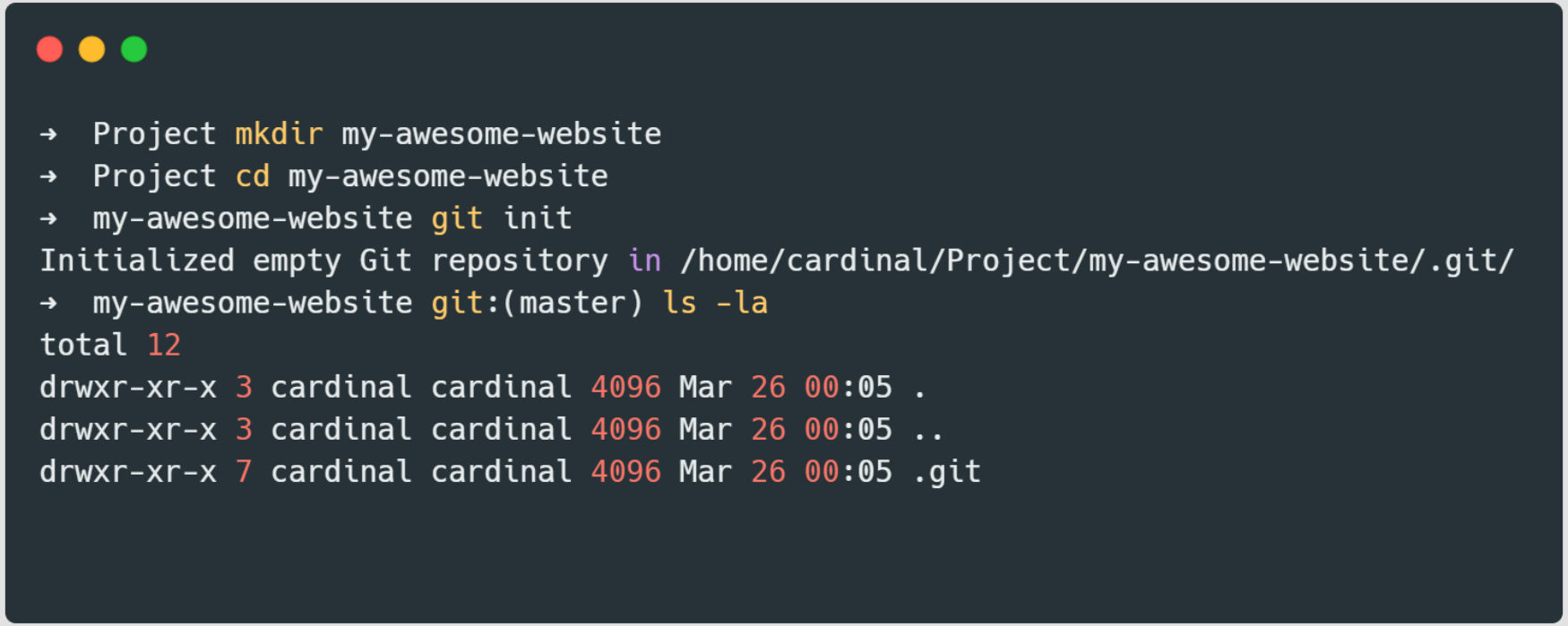
- Initializes a git repository.
- Creates a `.git` directory/folder inside the project directory, that will track all the changes made in the project directory.

```
$ git clone <repo-url>
```

- Clones the remote already existing repository.
- It already has a `.git` folder in it.

Setting up a repository

How to set-up a repository?



Adding data to the repository

- Create a folder named **`images/`** inside the project folder
- Inside the **`images/`** put in an image of yours and rename it to **`profile.jpg`**
- Create a file name **`index.html`** inside the project folder
- And copy the template and paste it in the **`index.html`** file

`Template Link:` https://0xcardinal.com/git-workshop/profile-template

```
My-awesome-website/
├── .git/
├── images/
│   └── profile.jpg
└── index.html
```

Inspecting the repository

`$ git status`

- Displays the state of the working directory and the stating area.

```
→  my-awesome-website git:(master) X git status
On branch master

No commits yet

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

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

Saving changes in the repository

`$ git add`

- Adds a change in the working directory to the staging area.
- Staging area, is a place where the files are kept that are to be included in the next commit.
- Why git add?
  - Helps in making atomic commits
  - It easy to track down bugs and revert changes with minimal impact on the rest of the project.

`$ git commit`

- Captures a snapshot of the project's currently staged changes.
- Equivalent to saving a file.

> *The commands: **`git add`**, **`git status`**, and **`git commit`** are all used in combination to save a snapshot of a Git project's current state.*

Saving changes in the repository

```
➜  my-awesome-website git:(master) ✗ git add .
➜  my-awesome-website git:(master) ✗ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   images/profile.jpg
        new file:   index.html

➜  my-awesome-website git:(master) ✗ git commit -m "Adds image folder and index.html"
[master (root-commit) 53f172e] Adds image folder and index.html
 2 files changed, 42 insertions(+)
 create mode 100644 images/profile.jpg
 create mode 100644 index.html
➜  my-awesome-website git:(master) git status
On branch master
nothing to commit, working tree clean
```

Commit Messages

- Be as verbose as possible in the commit message.
- Use imperative mood - spoken or written as if giving a command or instruction
- Make sense of the commit message.

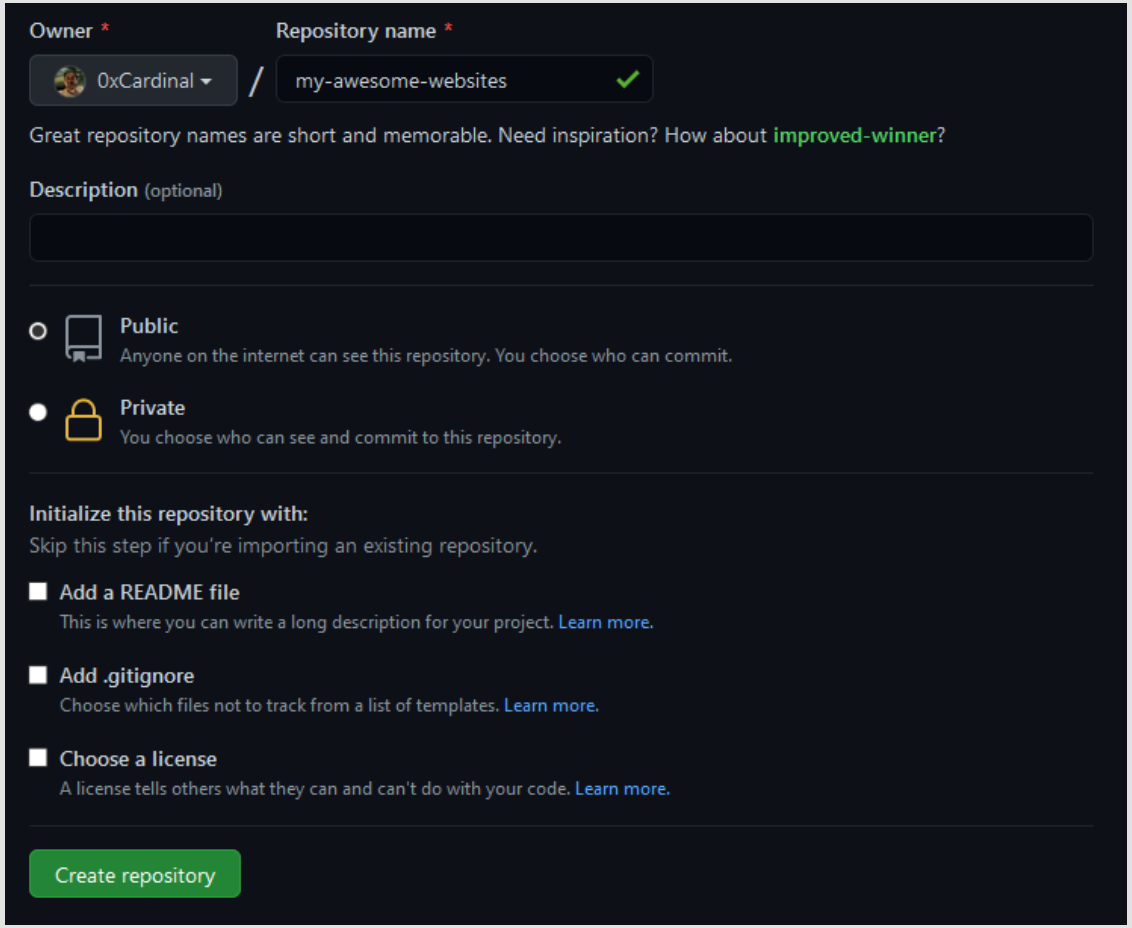
```
→  my-awesome-website git:(master) ✗ git log --oneline
093401b (HEAD -> master) Second Commit
53f172e Initial Commit
```

```
→  my-awesome-website git:(master) ✗ git log --oneline
093401b (HEAD -> master) Add PNG profile picture
53f172e Adds image folder and index.html
```

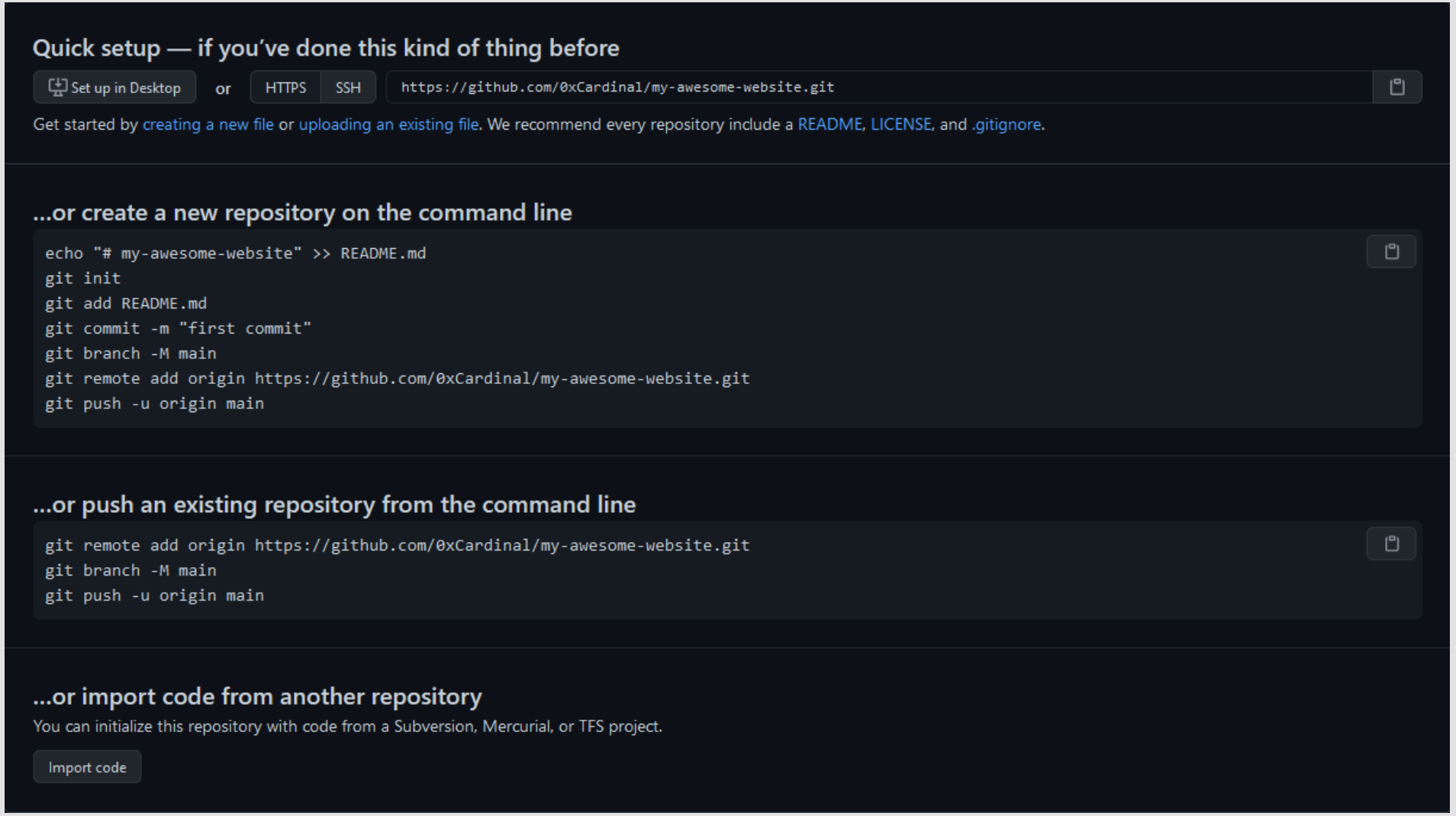
Saving changes in the repository

We are done with making changes locally and now we need to push it to a remote repository.

GitHub comes to rescue.



Saving changes in the repository



Connecting local and remote

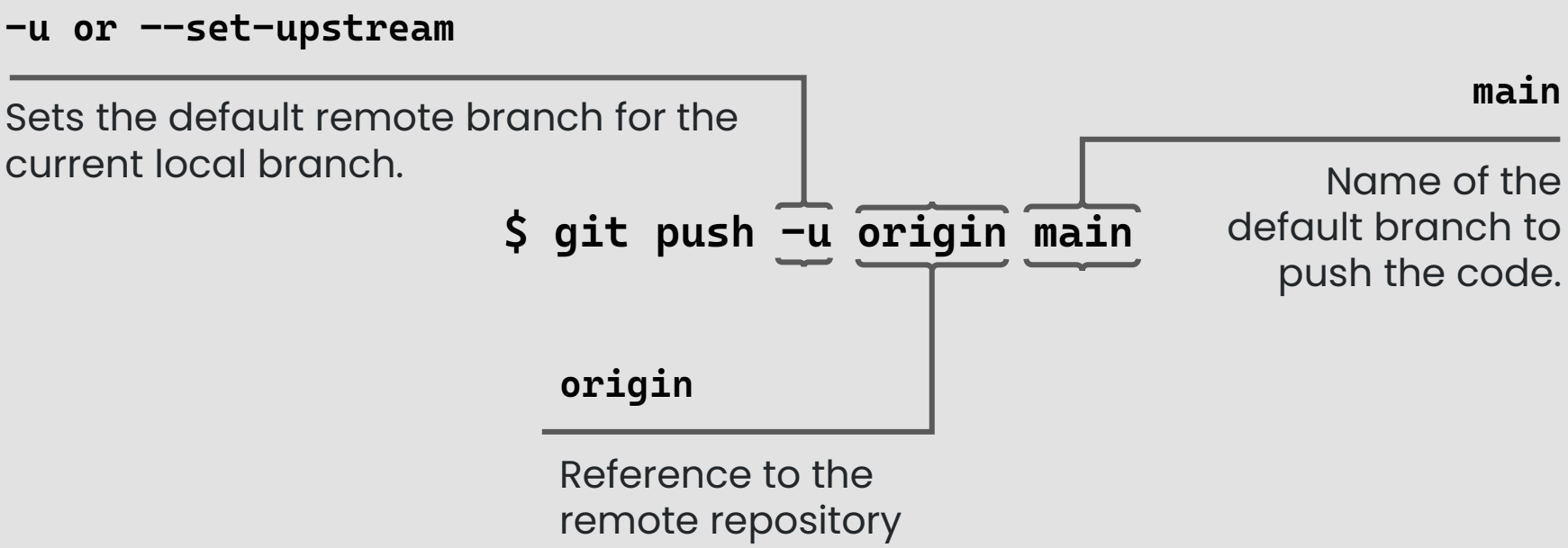
- The **`git remote`** command lets you create, view, and delete connections to other repositories.
- Remote connections are more like **bookmarks rather than direct links into other repositories**.

```
$ git remote add <name> <url>

$ git remote add origin https://github.com/0xCardinal/my-awesome-website.git
```

Pushing to the remote repository

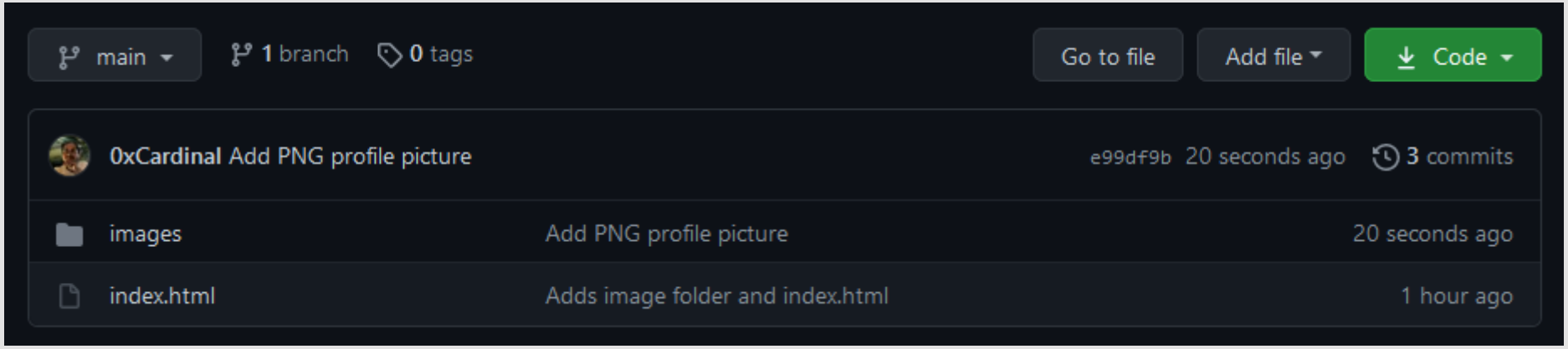
- **`git push`** is most commonly used to publish an upload local changes to a central repository. After a local repository has been modified a push is executed to share the modifications with remote team members.



Pushing to the remote repository

```
→  my-awesome-website git:(master) git remote add origin https://github.com/0xCardinal/my-awesome-website.git
→  my-awesome-website git:(master) git branch -M main
→  my-awesome-website git:(main) git push -u origin main
Username for 'https://github.com': 0xCardinal
Password for 'https://0xCardinal@github.com':
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (8/8), 1.32 KiB | 192.00 KiB/s, done.
Total 8 (delta 0), reused 0 (delta 0)
To https://github.com/0xCardinal/my-awesome-website.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Pushing to the remote repository



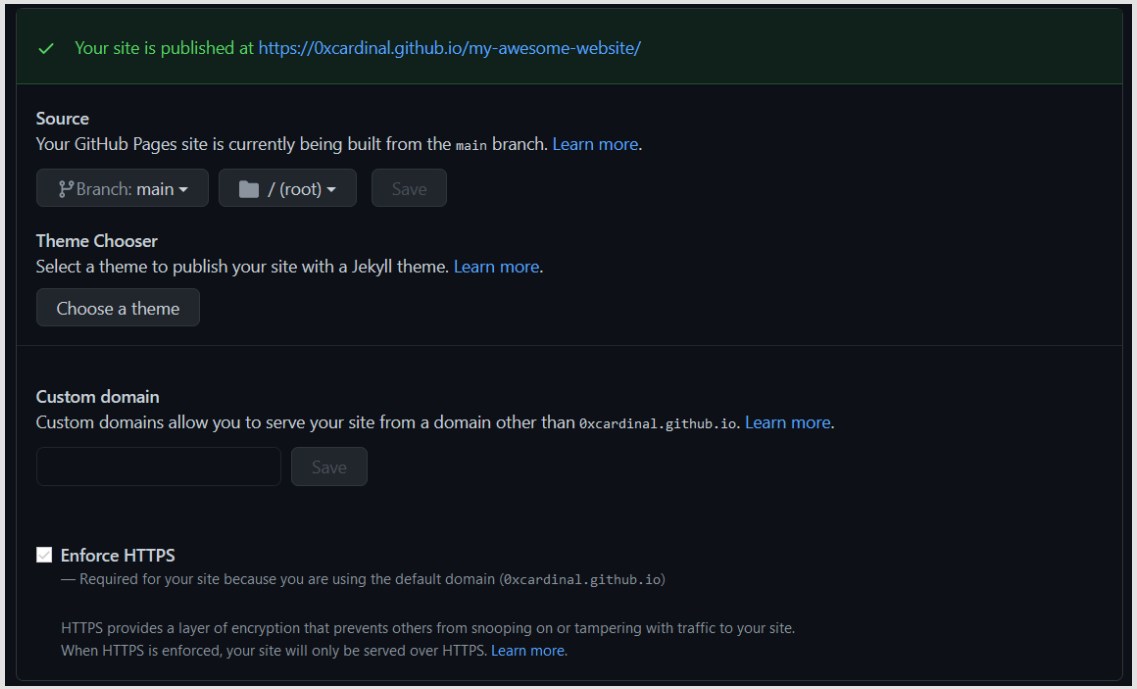
Hosting using GitHub

Using [GitHub Pages]

- GitHub allows the repository owners to host their website using GitHub pages.
- It looks for files such as **`index.html`** or **`index.md`** and it takes that as the initial point to host the repository as a webpage.

To enable GitHub Pages
Go to
`Settings` > `Scroll Down to GitHub Pages`

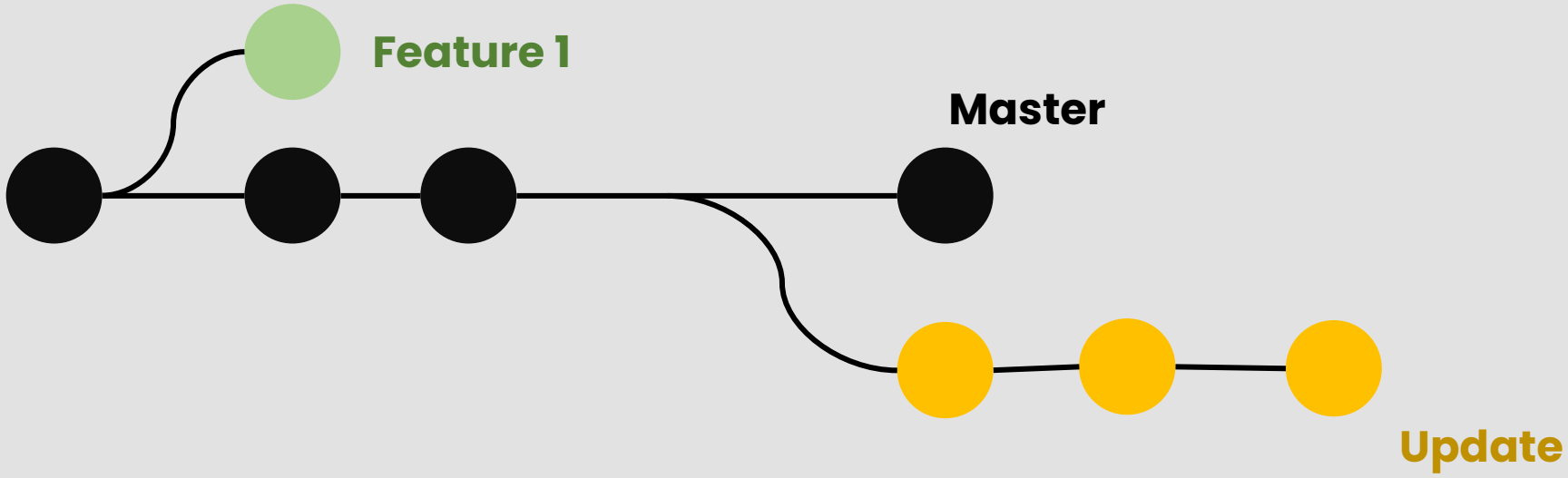
`http://username.github.io/repo-name`



Collaboration in Git

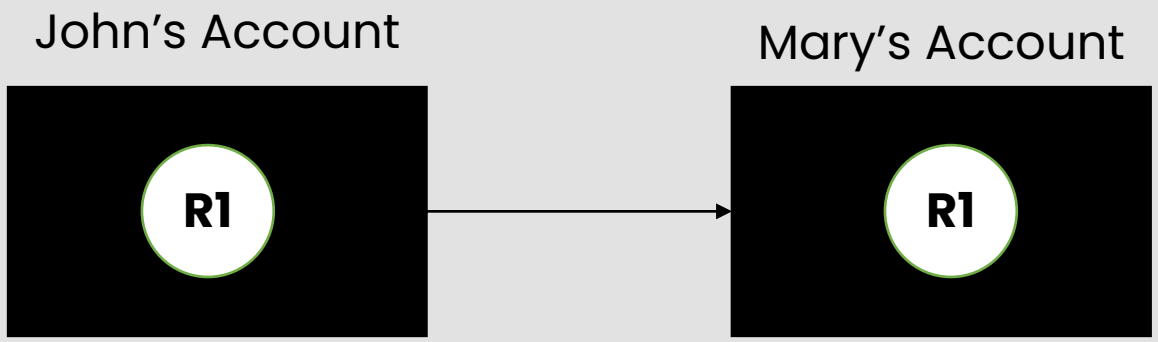
Branching

- A branch shows an independent line of development.
- Branches stays in the same repository.



Forking

- A fork is a copy of a repository.
- Forking a repository allows you to freely experiment with changes without affecting the original project.



Collaboration in Git Demo

Contributing to Other Repositories

Markdown

Bonus

Markdown is a lightweight and easy-to-use syntax for styling all forms of writing.

Why Markdown?

- Easy to use (See syntax)
- Widely acceptable – can be used to make websites as well.
    - Example – Hugo, Gatsby, etc. Work using Markdown format posts.

`README.md` uses Markdown Formatting.

License

Bonus

For your repository to truly be open source, you'll need to license it so that others are free to use, change, and distribute the software.

(Read more)

How to choose a license?

- https://opensource.guide/legal/#which-open-source-license-is-appropriate-for-my-project
- https://choosealicense.com/

Depreciation Notice

“Basic authentication using just a password to Git is deprecated, and will soon no longer work.”

(Read More)

Resources

- https://letmegooglethat.com/?q=All+about+git
- https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet
- https://www.atlassian.com/git/tutorials
- https://education.github.com/pack

Contact me @ ashwin@0xcardinal.com

Thank You :)

Kumar **Ashwin** | 0xcardinal.com