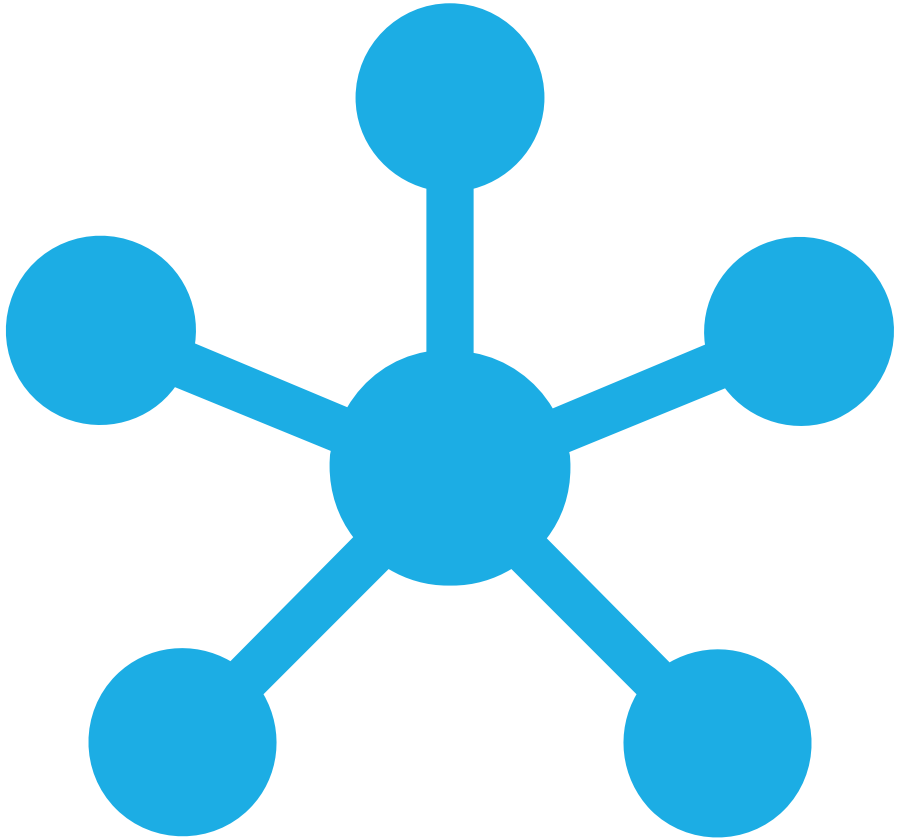
GITHUB



INTRODUCTION

We have talked about using **git** as a tool that we can use on our local computers to add some fancy features to our projects in terms of managing them.

We have learnt some simple commands that allow us to work safer as an individual who only ever needs to work on their local computer.

The problem with this simple model of work is that it really doesn't transfer to real world projects too well. In the real world you might have multiple team members all working on the same code base, from multiple computers all over the world.

To address this much more real world scenario, we introduce **GitHub.**

GITHUB

GitHub adds cloud based collaboration to our repositories. This means that anyone with permission can contribute to a repository. They can make code changes, suggest features and many other useful things for teams.

GitHub is not just for teams however, it is also a useful tool for you as an individual. You can use GitHub as a cloud based backup for your projects so that if anything ever happens to your local computer, you can still access your code.

You can also use GitHub as a means of transporting your code between different computers. For example, you might work on your projects on your computer, but then need to move the finished project onto your web server. GitHub is one way of doing this!

GitHub is also where you will find many **open source** projects hosted. Open source meaning that the code is visible to anyone who wants to look or even contribute to!

GITHUB ACCOUNT

To start, we need a free account. Go and sign up on https://github.com/.

If you already have one, great! One step done. Make sure you are just using a free account! The free tier comes with more than enough for our needs. The paid plans are more for large organizations who need much more from GitHub.

DIFFERENT REPOSITORY TERMS

When we introduce GitHub, we now have to make some distinctions. When working with our local computer with a repository, this is called a **local repository.**

When we start working with GitHub, we no longer have only a **local repository.** We introduce another repository hosted on GitHub called the **remote repository.**

Our goal when first learning about how git and GitHub interact with each other is simply to make sure that our **local repository** matches our **remote repository.**

We will get into more advanced workflows later that have to do with things like branching.

SIMPLE GOALS AND MORE COMMANDS

To start our GitHub adventures we have simple goals. Make sure the GitHub repository is a mirror of our local git repository. We achieve this by adding some new commands to our toolbelt:

- **git remote <add | remove> <URL>**
  - This command will connect or disconnect our local repository to our remote repository on GitHub. We get the URL from GitHub. Make sure the URL used here is the HTTPS **not the SSH**.
- **git branch –M main**
  - This command sets our local repository to be on the same default branch our remote repository starts on. For now, just know you need to do this once at the initial setup of a project.
- **git push –u origin main**
  - This command is ran once during the connection of our local repository to our remote repository. We will go into details of this after we learn about branching.
- **git push**
  - This command will take our code that has been committed locally, and *push* it up to our remote repository. You run this when you are happy with your local changes and want them backed up.

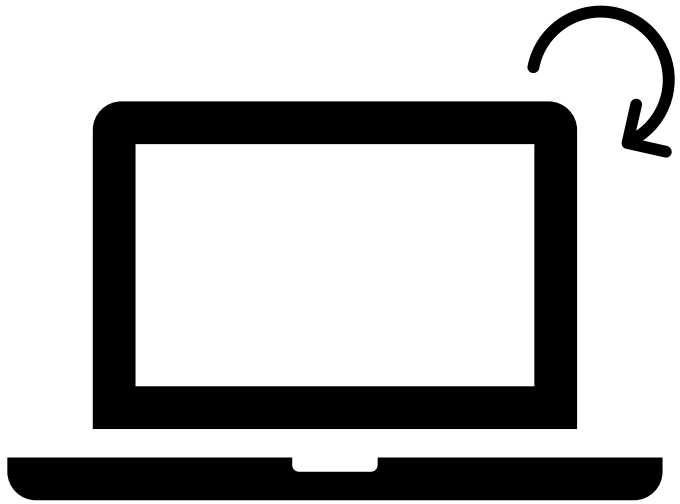
DIFFERENT WORKFLOWS

Git and GitHub can interact with each other through many different workflows. One thing that we must keep in mind is **git is not GitHub**. They interact with each other, but they are not the same thing!

Git is used locally on a computer when you are doing project work. When GitHub is used properly, it is a way to distribute and organize the work that we do with git locally.

In most of the workflows you will encounter during this course, GitHub will really just be a mirror of your local git work stored in the cloud used to move code around to your cloud servers.

WORKING ALONE WITH NO REMOTE REPOSITORY

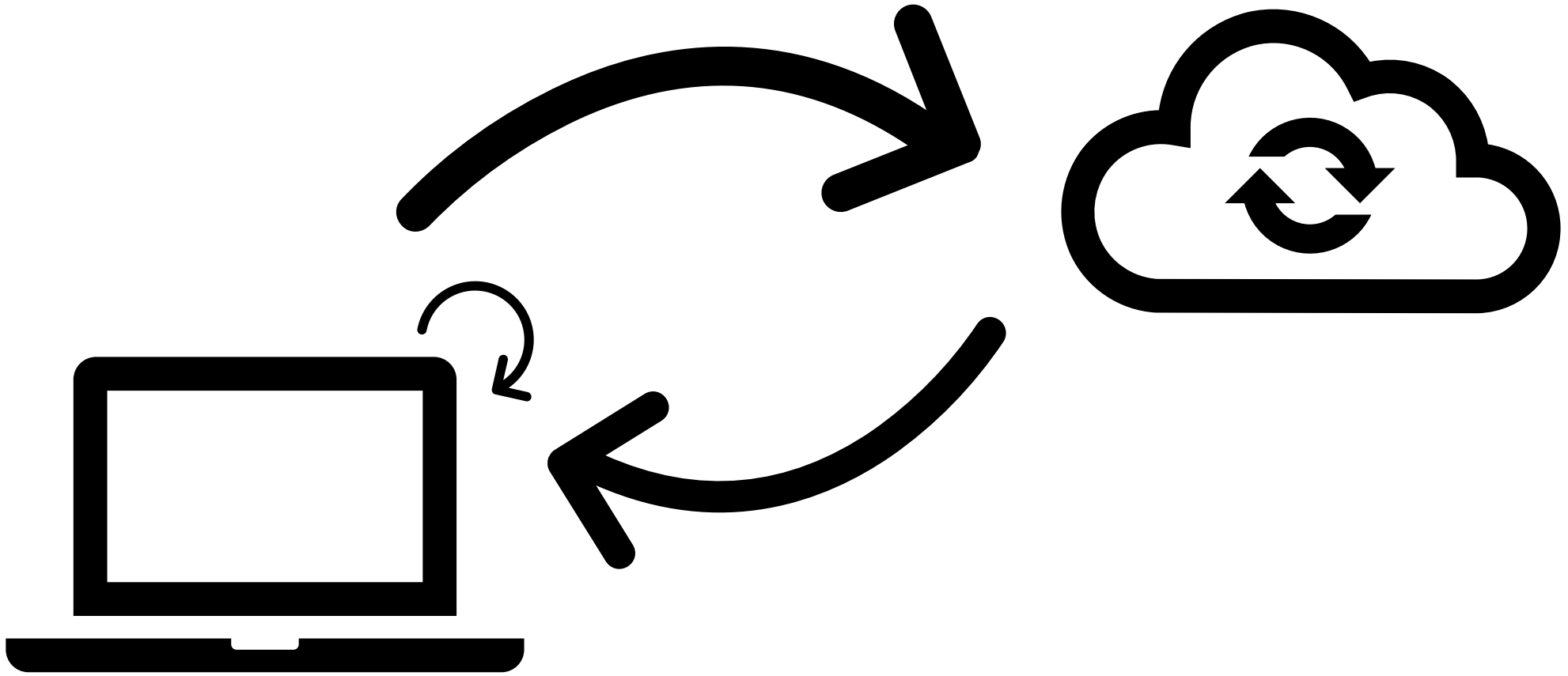


WORKING ALONE WITH NO REMOTE WORKFLOW

Because this is the most basic situation, it also has a very easy workflow.

1. Create the project folder and open it with VSCode
2. Initialize the project folder as a repo using **git init**
3. When you are happy with any amount of work run the commands:
    1. git add –A
    2. git commit –m “Descriptive message about your work”

WORKING ALONE WITH A REMOTE REPOSITORY



WORKING ALONE WITH A REMOTE REPOSITORY

Because this is going to be the main workflow we will use in the course, let's take this workflow and use it to introduce some new GitHub topics.

Before we can even think about starting to code up our projects, we need to be starting from a solid foundation where our repositories are correctly set up and linked. This means our local repository can talk to our remote repository and update the code up in GitHub.

Keep in mind throughout these steps our goal is to make sure our GitHub repository mirrors our local git repository. Meaning the work we do locally is sent up to GitHub. Please keep in mind that at any point when trying to push code up to GitHub, you may be asked for your username and password!

WORKING ALONE WITH A REMOTE WORKFLOW

Having a remote repository starts to add some complexity, luckily not too much.

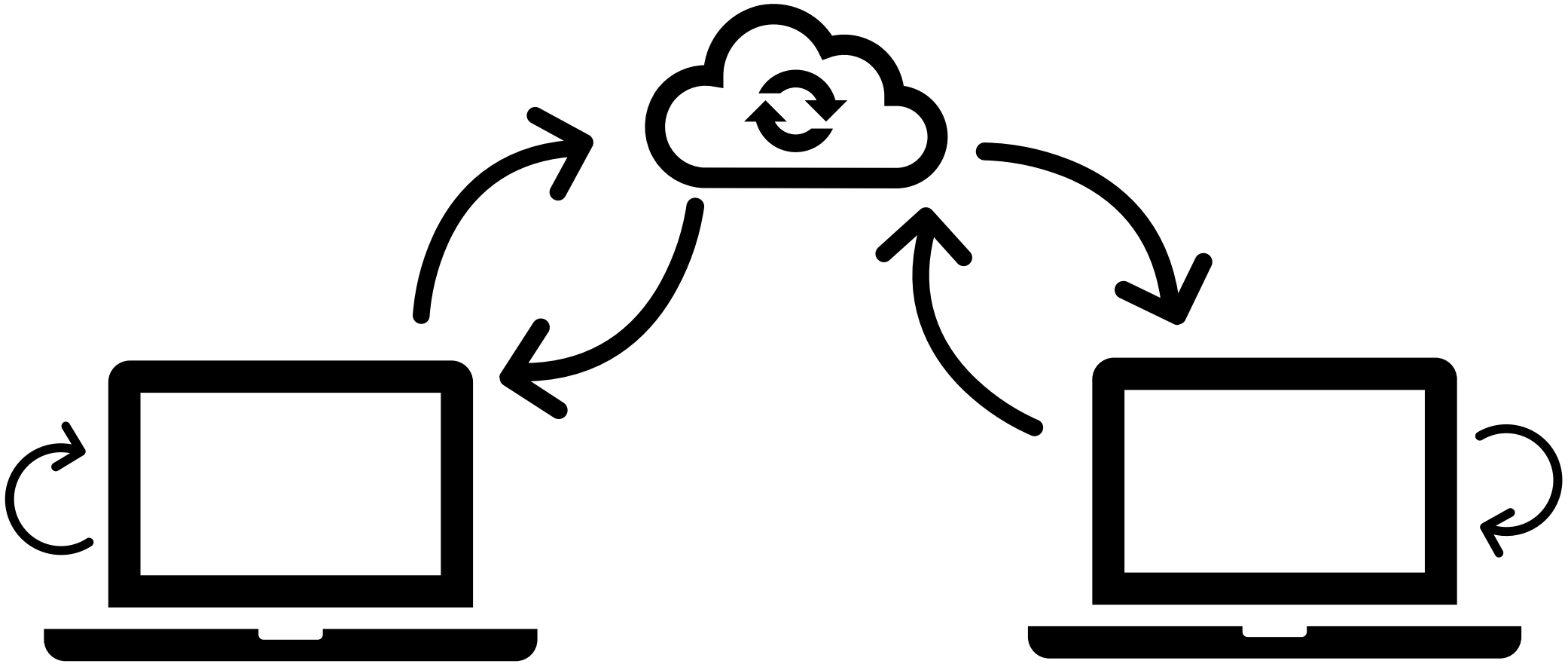
1. Create the project folder (either with mkdir, or any other tool you have to create projects) and open it with VSCode
2. Initialize the project folder as a repo using **git init**
3. Create a new repository on GitHub, make sure it is empty
4. In your local repository, create a file called **README.md**
5. Open this file and give a description of the project in plain text using your own wording (make sure you save).
6. Add and commit this file.
7. Run the commands given to you in the GitHub repository in your local repository. They will look like:
    1. **git remote add origin <URL>**
    2. **git branch –M main**
    3. **git push –u origin main**
8. Code the project! Any time you are happy with work, run the commands:
    1. git add –A
    2. git commit -m “Message about the work you did”
    3. git push

KNOWLEDGE CHECK

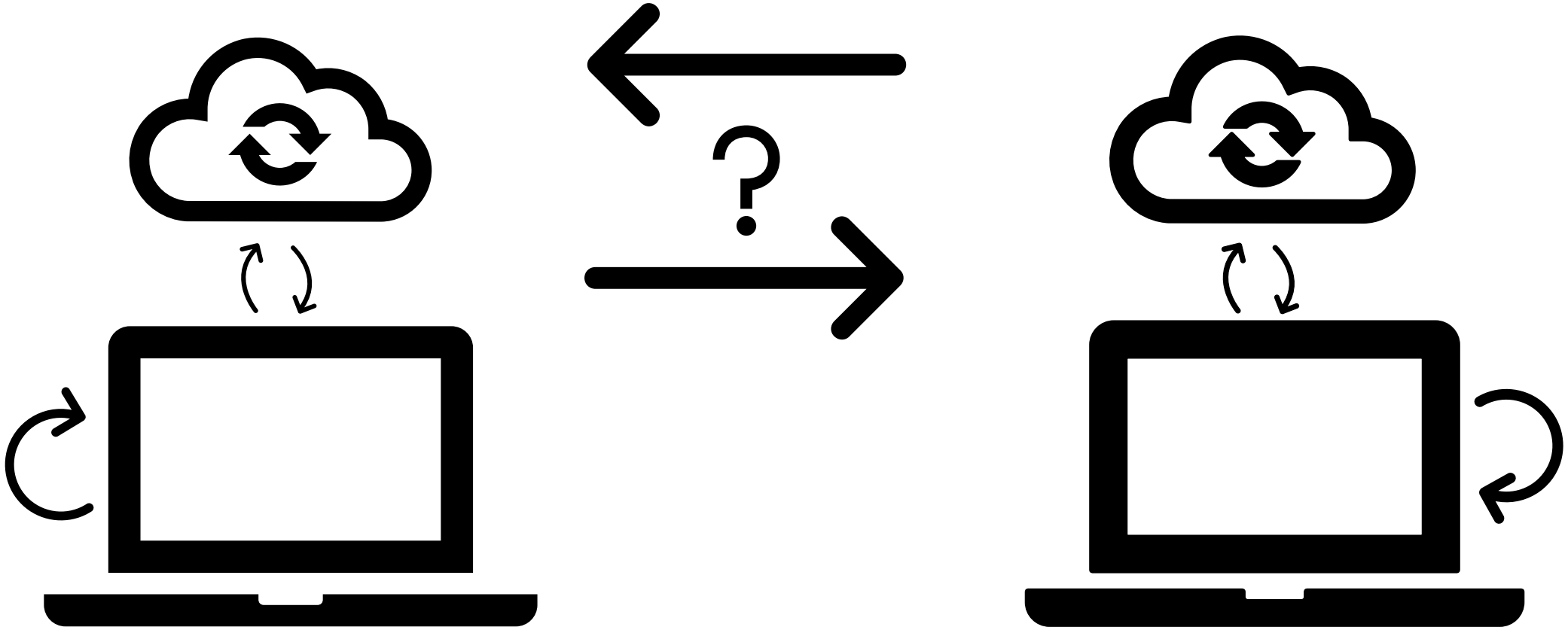
Let's see how our handle on setting up our Git and GitHub repos is.

1. If you haven't already, create a directory called **Scratch inside your InnoTech** directory.
2. Either using mkdir or your projectStarter script, create a folder called **GitHubCheck inside the Scratch folder.**
3. Open GitHubCheck with VSCode.
4. Follow the working alone with a remote workflow to get everything using Git and connected to GitHub.
    1. When it comes to the coding step, just add one line to an index.html

WORKING TOGETHER WITH ONE REMOTE REPOSITORY



WORKING TOGETHER WITH MULTIPLE REMOTE REPOSITORIES



People often get intimidated by Git and GitHub working together because of fundamental misunderstandings of the technology and what it is trying to do.

You will see new errors when working with these technologies. The reason we introduce this stuff so early is that you see the errors and learn how to fix them!