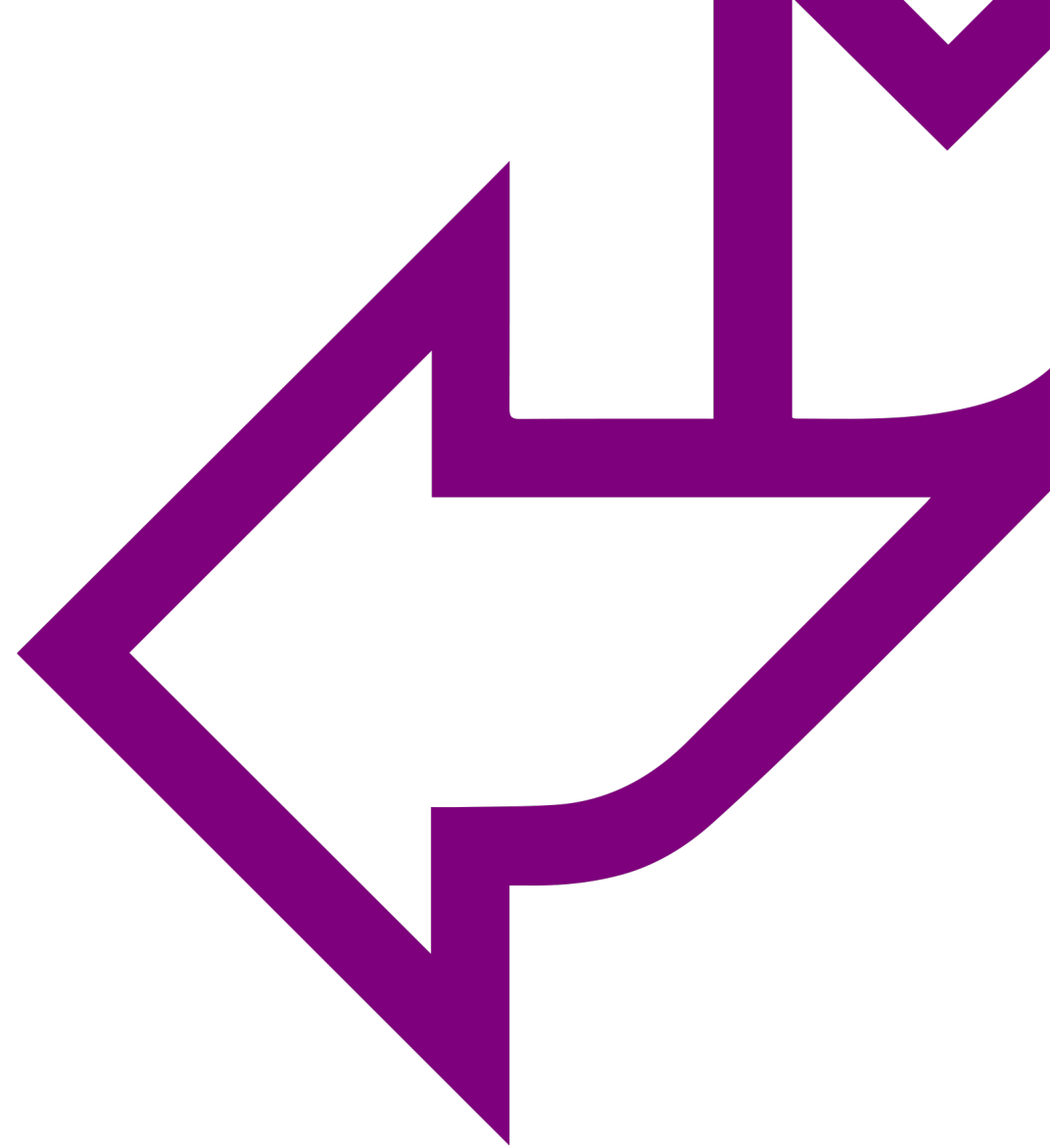




# What is Version Control?

Module 5 – Version Control with GIT





# WHAT IS VERSION CONTROL?

**Version control** (also known as **source control** or **revision control**) is the practice of tracking and managing changes to code

Source control is vital for managing software development projects

Being able to track changes to code allows developers to:

- centralise all code changes and additions into one code repository
- collaborate effectively and work in teams
- control the integration of new code into the codebase
- track changes from the entire team over the full lifetime of the project
- revert code back to previous versions



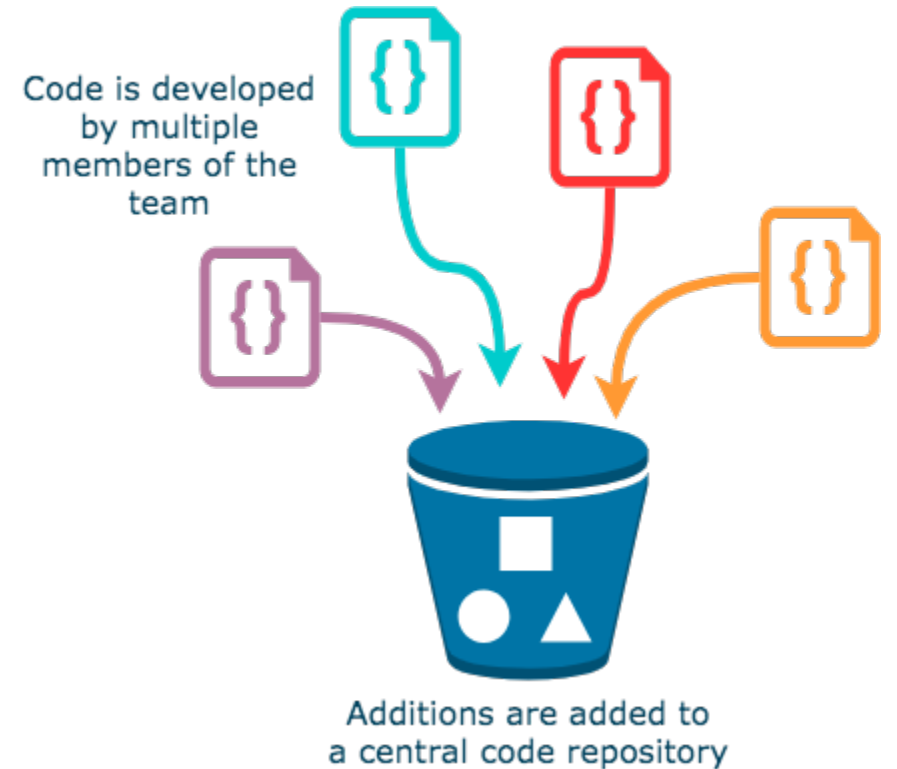


# Repositories

Version control stores code in a **repository**. This essentially functions as a big bucket for all your code to live in!

Having one place where your code resides means that versioning stays consistent, regardless of who's working on it.

This is vital when developers are working on the same project in tandem with one another – if there was no repository, there would be no cohesion or central codebase.



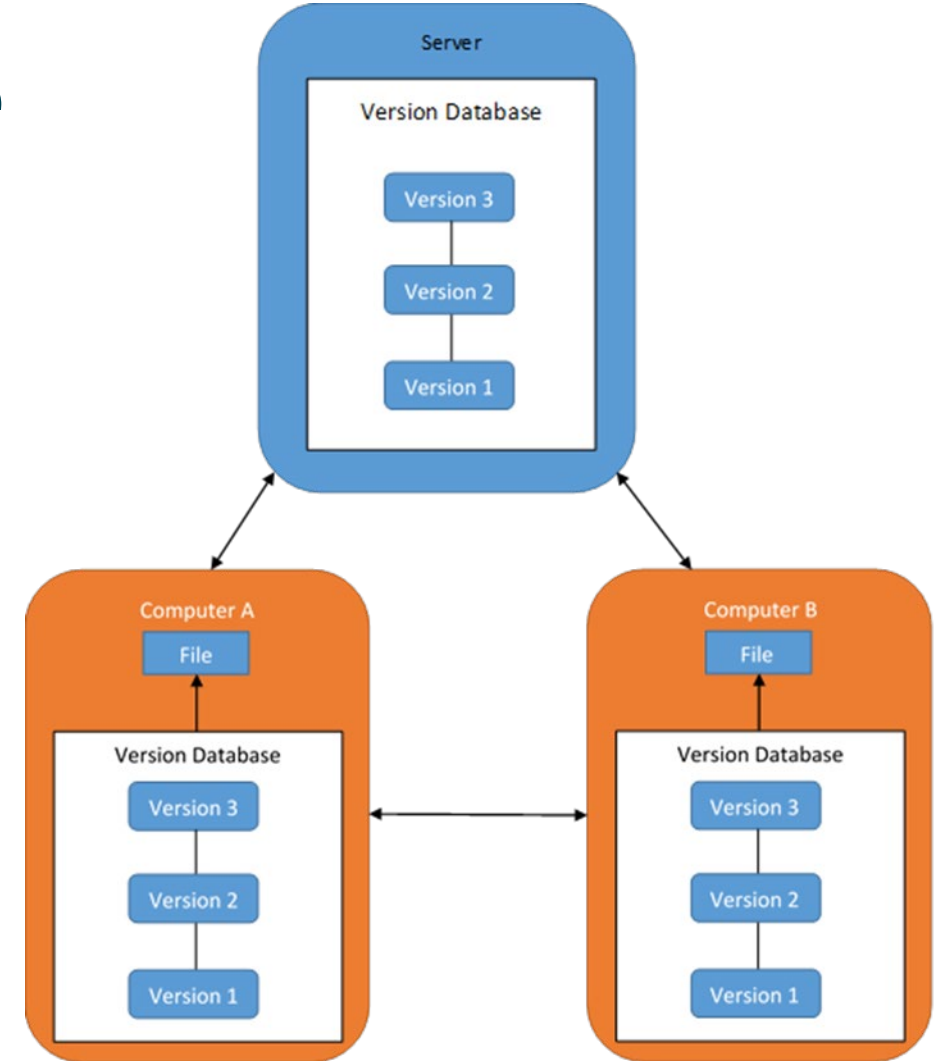


# Version control structure

In a version control system, all work is held on a server, but the repository is mirrored locally when accessed

Several remote repositories are created, which essentially creates backups of the central repo!

With this, there is no **single point of failure**





# INTRODUCING GIT VERSION CONTROL

Git is a version control system used by thousands around the world, and is arguably one of the most popular version control system (VCS) tools in software development

It can be used stand-alone or through inclusion within an IDE such as Eclipse, Visual Studio, or VS Code

During this module, we will be using the Git CLI (Git Bash) and a web-based UI for its repository (GitHub)





# REPOSITORY HOSTING SERVICES

There are several web-based version control repository hosting services, most of which either utilise or are wholly based upon Git, such as:

- **GitHub (this is what we will use in this module!)**
- GitLab
- Bitbucket

Many cloud providers have their own code repository offerings which offer simple and powerful integration with other cloud services:

- AWS CodeCommit
- Azure Repos (as part of Azure DevOps)
- Google Cloud Source Repositories



# INSTALLING GIT AND GIT BASH

If you don't already have Git downloaded, you can download and install the version of Git required for your operating system here:

<https://git-scm.com/download>



# QA Repositories: local and remote

## Local repositories:

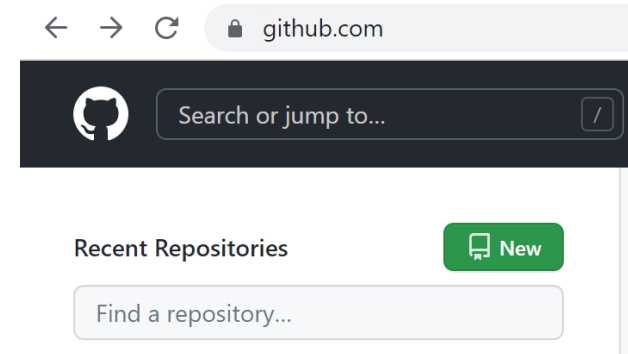
- Reside on a developer's machine
- Are accessible by that one developer
- Are usually linked to a remote repository to enable **pushing** and **pulling** of team members' code changes
- Are created using the **git init** command

```
victo@DESKTOP-5A2PA0H MINGW64 /c/git_demo
$ git init
Initialized empty Git repository in C:/git_demo/.git/

victo@DESKTOP-5A2PA0H MINGW64 /c/git_demo (main)
$
```

## Remote repositories:

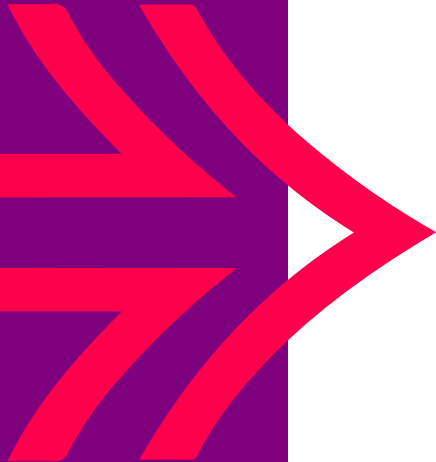
- Reside on a repository hosted Git Server
- Are accessible to all team members who can **git clone** the repo
- Enable developers to **push** their local code to this remote repository and **pull** down their team's code to their local repository
- Created on a Git Server using a GUI







# GITHUB: SIGNING UP FOR AN ACCOUNT



Sign-up for a GitHub account: <https://github.com/join>

Join GitHub

## Create your account

Username \*

Email address \*

Password \*

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.  
[Learn more.](#)


Email preferences

☐ Send me occasional product updates, announcements, and offers.

Verify your account

Please solve this puzzle so we know you are a real person

Verify



Create account



# Configuring Git locally

Once you've made an account, you then need to configure Git locally

Git needs to know who you are to enable you to commit code to repositories. Use the **git config** commands as follows:

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

To confirm you have configured your user.name:

```
victo@DESKTOP-5A2PA0H MINGW64 /c/git_demo (main)  
$ git config user.name  
Victoria
```





# CLONING A REPOSITORY



The **git clone** allows the cloning of a repository to a newly created directory

- Once the clone is created locally, **git fetch** can be executed to get updates for the remote branches
- This lets us keep our local branch up to date with changes on the remote

Use **git clone** to copy and link to a remote repo:

```
# git clone https://github.com/velocity1b/2022
```



```
victo@DESKTOP-5A2PA0H MINGW64 /c/avanade/velocity1b
$ git clone https://github.com/velocity1b/2022
Cloning into '2022'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5 (delta 0), reused 5 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```



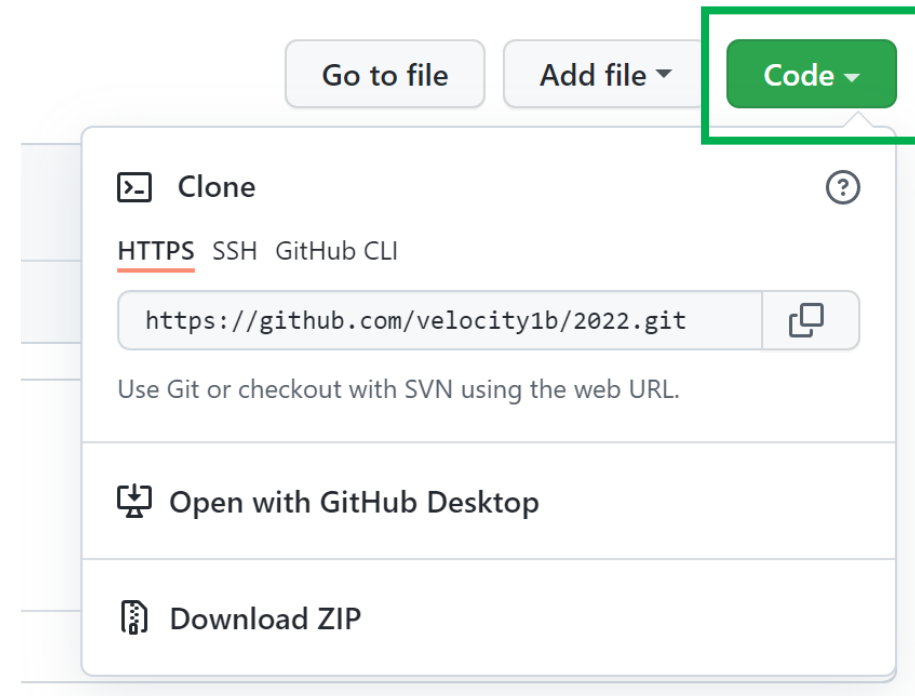
# CLONING A REPOSITORY: STEP 1



In order to clone a repository, you need to know the URL of the repository you want to clone.

To get the URL of the repository you want to clone, you need to go to that repository in the VCS (Version Control System) you are using, such as: GitHub.

1. In GitHub, you can find the URL by selecting the green **Code** button within the repo.





# CLONING A REPOSITORY: STEP 2

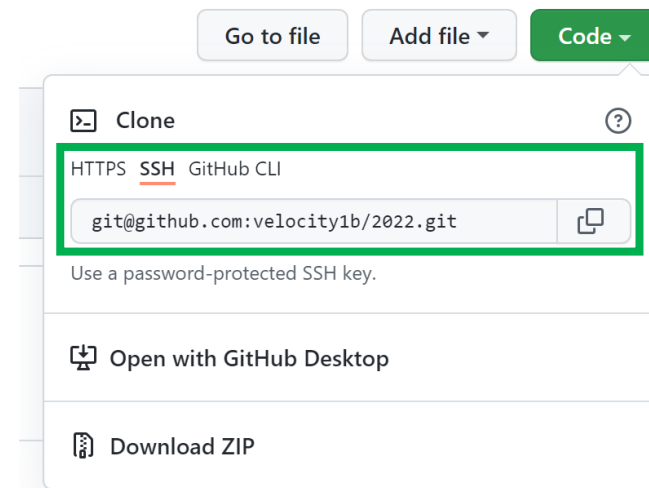
When you select the green Code button, you will have the option of three types of URLs to copy in order to clone a repository; HTTPS, SSH, and GitHub CLI

2. Select the SSH tab on the pop-up and copy the URL. The SSH URL should be in the format of:

```
git@github.com:[username]/[repository_name].git
```

For example:

```
git@github.com:velocity1b/2022.git
```





# CLONING A REPOSITORY: STEP 3

3. Open Git Bash and navigate to the directory where you want to save the cloned repository.

- You may want to create a new directory using **mkdir** and change into that directory using the **cd** command

In this example, a new directory has been created and the remote repository has been cloned into it:

```
git clone git@github.com:[username]/[repository_name].git
```

```
victo@DESKTOP-5A2PA0H MINGW64 /c/avanade  
$ mkdir velocity  
  
victo@DESKTOP-5A2PA0H MINGW64 /c/avanade  
$ cd velocity/
```

```
victo@DESKTOP-5A2PA0H MINGW64 /c/avanade/velocity  
$ git clone git@github.com:velocity1b/2022.git  
Cloning into '2022'...  
Enter passphrase for key '/c/Users/victo/.ssh/id_rsa':  
remote: Enumerating objects: 3, done.  
remote: Counting objects: 100% (3/3), done.  
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0  
Receiving objects: 100% (3/3), done.
```



# CLONING A REPOSITORY: STEP 4

4. Once the repository has been cloned locally, change to the project directory with the **cd** command into the local copy of the repository

Run the command **git branch** to check that the active branch has been checked out.

For example:

```
victo@DESKTOP-5A2PA0H MINGW64 /c/avanade/velocity
$ dir
2022

victo@DESKTOP-5A2PA0H MINGW64 /c/avanade/velocity
$ cd 2022

victo@DESKTOP-5A2PA0H MINGW64 /c/avanade/velocity/2022 (main)
$ git branch
* main
```



# **Exercise 1: Getting Started**

- **Create a Git account**
- **Create a new Git repository**
- **Clone the repository**





**END OF SECTION**

