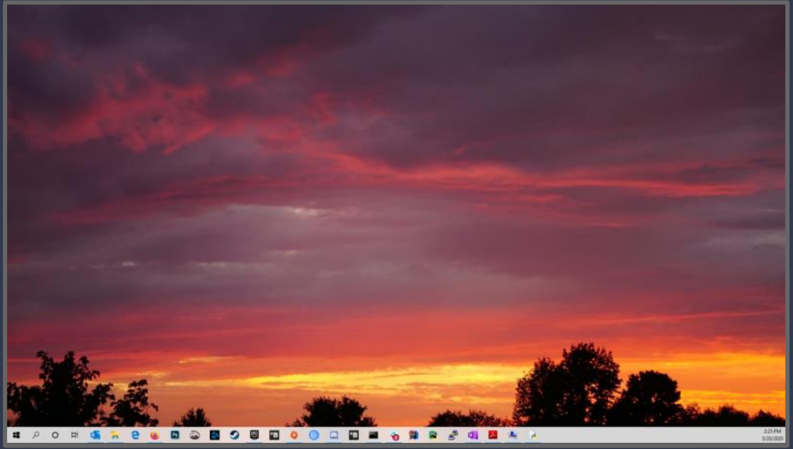
CSEC/SWEN-123
Software Development & Problem Solving

Unit 1.2: Git Basics & Environment Variables

RIT | Golisano College of Computing and Information Sciences
Department of Computing Security

Windows 10 Computer Literacy



If you’d like to use a non-Windows computer for your out of class work, you will need to do some extra learning on your own.

Many of the commands you will use work much the same in Linux or OSX. Others you may easily Google.

- The goal of this week’s lectures is to establish a common, minimum level of computer literacy amongst the students in the class.
- We will explore:
  - The Command Line ✓
  - The File System ✓
  - Version Control (with Git) ➤
  - Environment Variables ➤
  - More Advanced Git
  - Scripting
- Today we will learn about ***version control software*** (VCS) - Specifically, you will begin learning how to use Git. You will learn how to:
  - Initialize a repository.
  - Add new/changed files to your repository.
  - Commit your changes.
  - Connect your repository to GitHub.
  - Push your changes to GitHub.
- We will also learn about creating and modifying the ***system environment variables*** in Windows 10.

What is Version Control?

- A **version control system** (VCS) is used to manage a persistent, ***versioned*** backup of your code.
  - If stored on a different computer (e.g. www.github.com), the backup provides a secure duplicate of your code that can be recovered in the event of a disaster such as a lost or stolen laptop.
  - The backup is also ***versioned*** meaning that you can revert to an older version of the code if/when necessary.
- Directories and files are tracked in a ***repository***.
  - Each developer has their own workspace, i.e. their local file system on a laptop of a desktop.
  - Some developers have more than one, e.g. one on a laptop and a second one on a desktop PC.
  - Teams working together share the same ***remote repository***., e.g. on GitHub.
- Developers can:
  - Make changes to the workspace including **adding**, **removing**, **modifying**, or **moving** files (including directories).
  - **Commit** changes from the local workspace to the remote repository.
  - See the **history** of changes made to the repository.

There are many alternative implementations of source control software including CVS, SVN, Microsoft TFS, and many others.

This semester, we will be using Git, which is used throughout industry.

The first step is to make sure that Git is installed and configured properly. We can do this using the `git --version` command.

```
PS C:\Users\ron> git --version
git version 2.26.2.windows.1
```

If installed correctly, you should see output similar to the above.

1.2.1

Git Version

Verify that Git is installed and configured properly.

- Launch the command prompt and use `git --version` to verify that Git is installed and configured properly. You should see output similar to the example below.

```
PS C:\Users\ron> git --version
git version 2.26.2.windows.1
```

Creating a New Local Repository

- Before you can do anything else, you need to create a **local repository** in which to store your files (code, images, PDF, etc.) on your local file system.
- There are multiple ways to create an empty Git repository:
  - Create the repository on GitHub and download it to your computer.
  - Use a tool like GitHub desktop or Tortoise Git.
  - Create the repository as part of a project in an integrated development environment like Pycharm or VS Code.
  - Create the repository from the command prompt using the `git init` command.
- This semester we will start by using the command prompt.

The `git init` command will make a new, empty repository in the directory from which it is executed.

```
PS C:\Users\ron\Workspace> git init
Initialized empty Git repository in
C:/Users/ron/Workspace/.git/
```

Once initialized, you can begin to start creating files and directories to track in the repository.

You will also be able to add, modify, move, or delete files in the repository using Git commands.

1.2.2

Create a Local Repository

Use the `git init` command to create a git repository in your local file system.

- Create the following directory structure inside your user directory: `SoftDevI\Unit01\Day02`
  - e.g. `C:\Users\ron\SoftDevI\Unit01\Day02`
- Change into the `Day02` directory.
- Run the git init command to create a new, empty repository.
  - You should see output similar to that shown below.

```
PS C:\Users\ron\SoftDevI\UNit01\Day02> git init
Initialized empty Git repository in
C:/Users/ron/SoftDevI/Unit01/Day02/.git
```

- Try running the `ls -force` command inside the directory. What do you see?

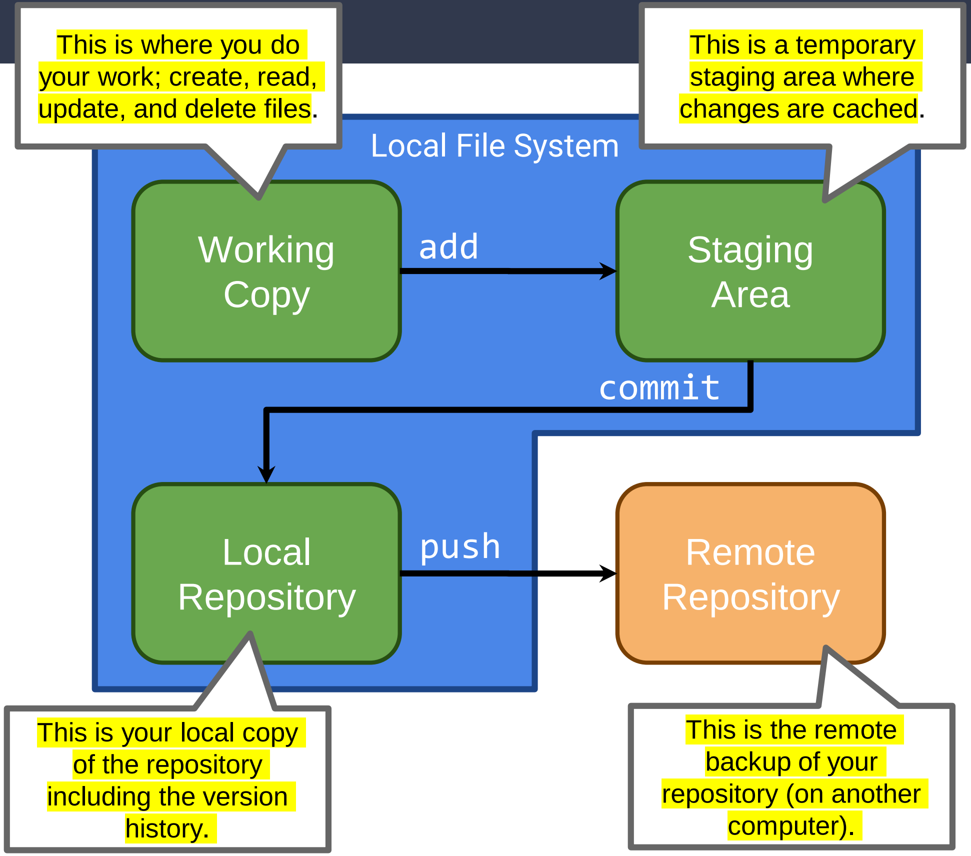
When Git creates a repository, it creates a hidden directory with lots of files inside it.

You can use `ls -Force` to see the directory. You can `cd` into it as well, e.g. `cd .git`

Be careful not to modify any of the files inside - doing so can corrupt your repository!

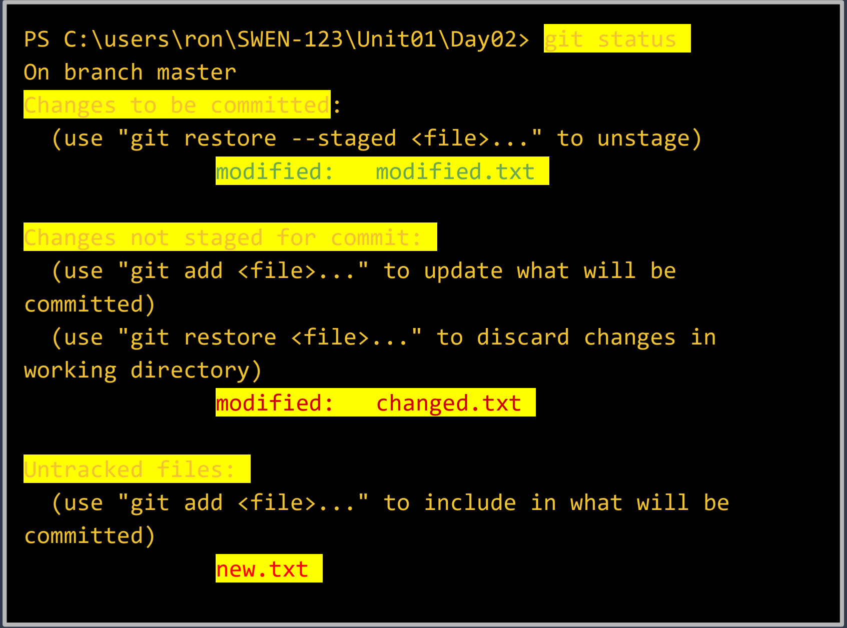
How Git Works

- As you work with Git, the files in your project will move through four different stages.
  - The **working copy** is the directory on your file system were you create, update, and delete your files.
  - The **staging area** is where you temporarily cache files that you would like to commit to your local repository.
  - The **local repository** is a copy of your repository in the local file system, complete with version history for all of your files.
  - The **remote** repository is a backup of your repository on a separate computer (server).
- You must execute separate Git commands to move your files from one stage to the next.


This is where you do your work; create, read, update, and delete files.
This is a temporary staging area where changes are cached.
Local File System
Working Copy
add
Staging Area
commit
Local Repository
push
Remote Repository
This is your local copy of the repository including the version history.
This is the remote backup of your repository (on another computer).

Git Status

- The `git status` command can be used to check the status of all of the files in your repository.
- The output is color coded depending on the status of each file.
  - **Untracked files** are files that have never been added to the repository and appear in **red**.
  - **Modified files** are existing files already being tracked by the repository have not been staged. They also appear in **red**.
  - **Modified files** are existing files that have been staged appear in **green**.
- Depending on the current state of your project, you may see different files in different states, e.g. untracked files, modified files, and staged files.
- If there are no changes in your workspace, you will see fairly benign output from Git, e.g.:
  - `No commits yet`
  - `nothing to commit, working tree clean`



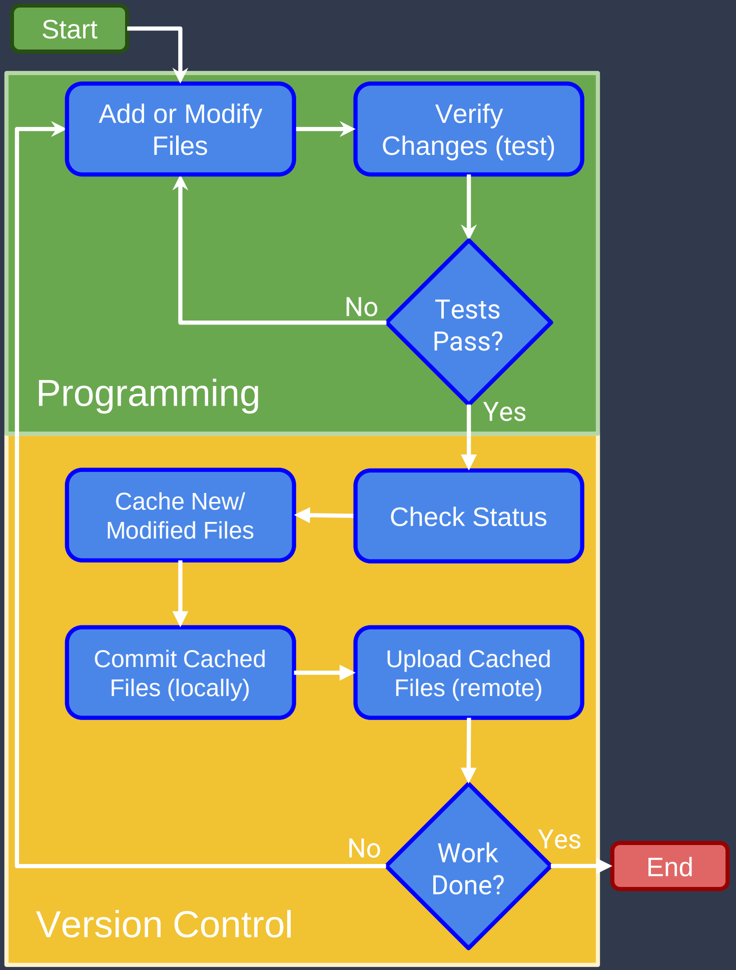
```
PS C:\users\ron\SWEN-123\Unit01\Day02> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
              modified:   modified.txt

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:   changed.txt

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

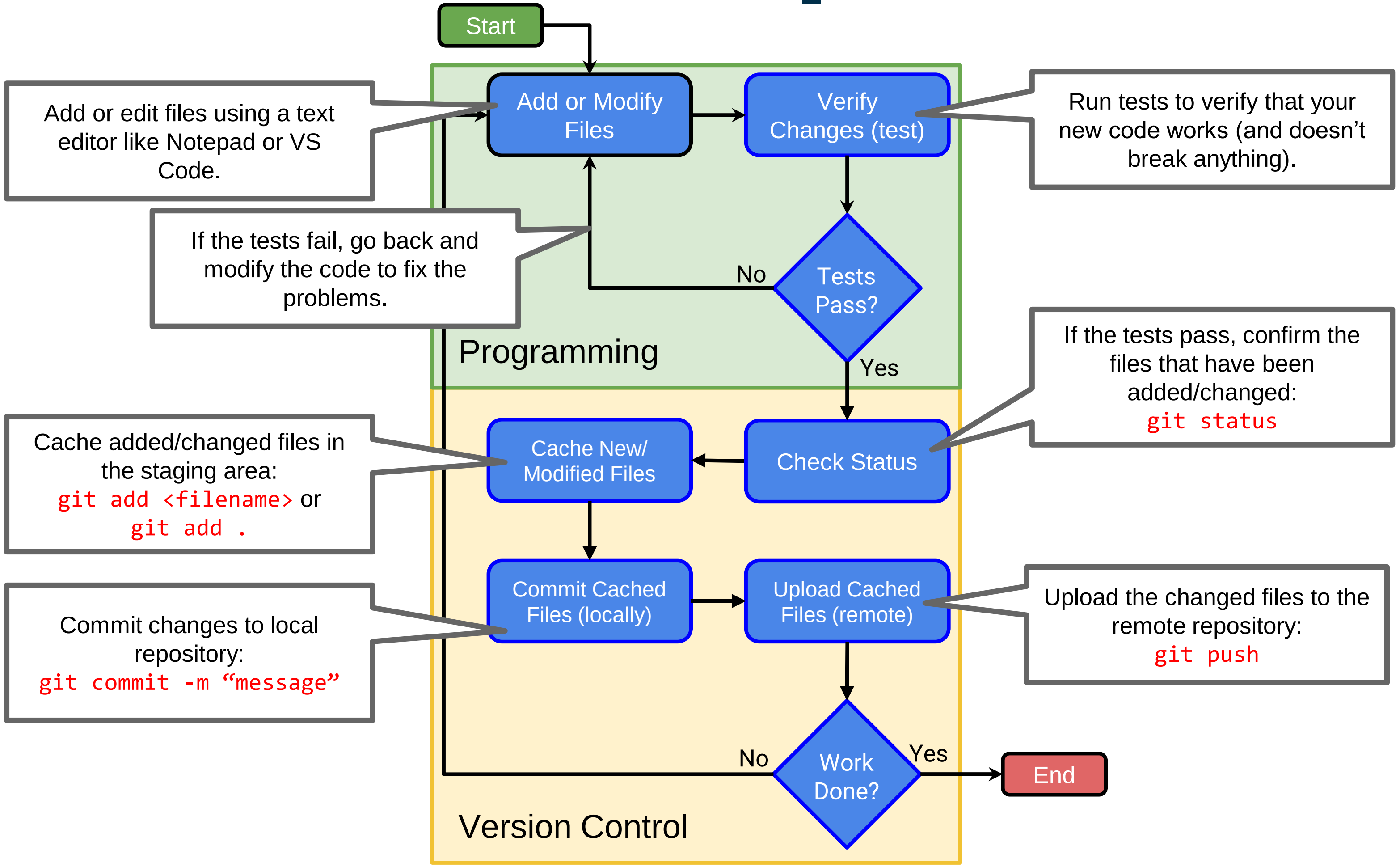
The output from git status also offers helpful advice regarding common actions that you may want to perform on each file, e.g. "`use git add <file>...`" for untracked files.

Development Workflow



- A **version control system** (VCS) is an excellent way to back up your files and preserve previous versions (your version history).
- But a VCS is only as good as the process that a software developer practices when using it.
- A VCS is more effective when:
  - A small, incremental change is made - when programming, this should only be a few lines of code!
  - The change is verified via testing.
  - The change is added to the repository.
  - Repeat every few minutes!
- Unfortunately, it is often the case that inexperienced developers get ***stuck*** in the programming loop and only seldomly commit. This causes lots of potential problems, including:
  - Increased odds that a disaster will result in significant lost work.
  - Harder to move between computers.
  - Harder to roll back specific changes.
  - Too few commits obfuscate the history of changes.

A Closer Look at the Development Workflow



Add or edit files using a text editor like Notepad or VS Code.

If the tests fail, go back and modify the code to fix the problems.

Run tests to verify that your new code works (and doesn't break anything).

If the tests pass, confirm the files that have been added/changed: `git status`

Cache added/changed files in the staging area: `git add <filename>` or `git add .`

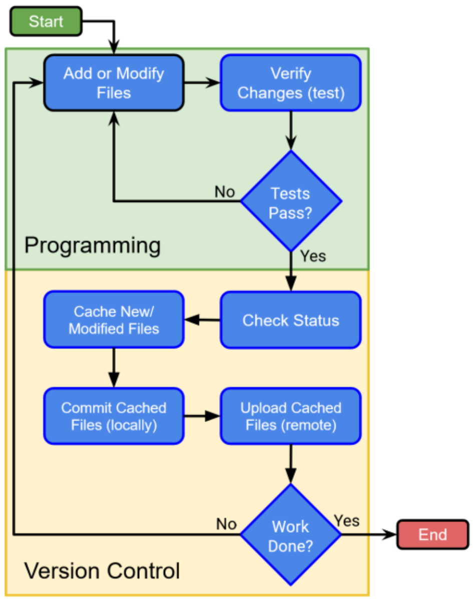
Commit changes to local repository: `git commit -m “message”`

Upload the changed files to the remote repository: `git push`

1.2.3

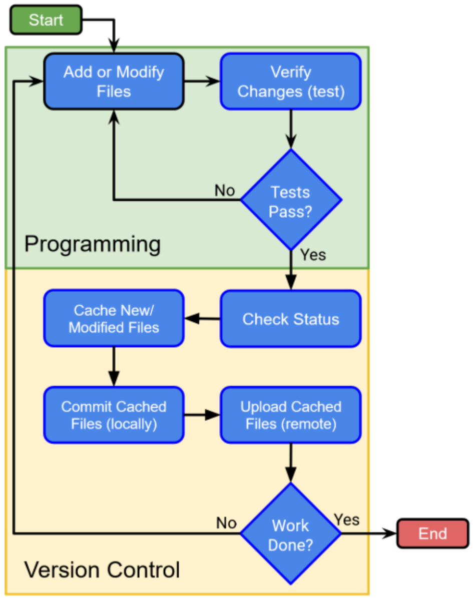
Adding Files

Create a new file and commit it to your repository.



- If necessary, launch a command prompt and navigate to your `Day02` folder.
- Use the `git status` command.
  - You should see that there is **nothing to commit**.
- Use Notepad to create a new file named "`me.txt`" with your first name, last name, email address, and today's date each on a separate line.
  - Don't forget to save (***CTRL-S***)!
- Use `git status` again to see the status the new file.
  - You should see that the file is **untracked**.
- Use the `git add` command to stage the file, e.g. `git add me.txt`
- Use `git status` again to see that the status of the file has changed.
  - You should see that the file is **staged** ("`Changes to be committed`").
- Use `git commit` to commit the file into your local repository.
  - Specify a comment using the `-m` option, e.g. `git commit -m "initial commit"`
  - If prompted to do so, enter your GitHub email and/or username.
- Use `git status` one last time.
  - You should see that there is **nothing to commit**.

1.2.4



Modify a File

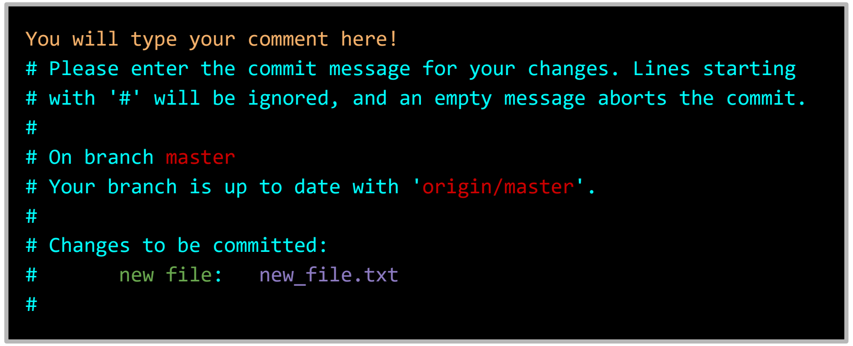
Make a change to your file and practice the Git workflow.

- Edit your "`me.txt`" file and add the day of the week on a separate line at the end of the file.
- Practice the Git workflow to properly commit the file to your local repository.
  - Status
  - Add
  - Commit (with comment)

Oops! I forgot to use `-m`!

- In some version control systems, adding a comment each time you commit to the repository is optional, however, Git ***requires*** that there is a comment with every single commit to the repository.
  - This is a good thing! When used properly the commit comment can be used to quickly find a specific change.
- The fastest and easiest way to do this is by using the `-m` option when executing the commit to specify the command.
  - e.g. `git commit -m "clever comment"`
- But what happens when you forget to use the `-m` option?

If you forget to use `-m`, Git will open a text editor into which you may type a comment. By default, this editor will be **vim**, which is installed with Git.



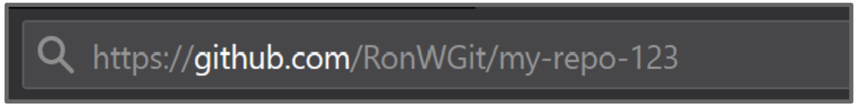
```
You will type your comment here!
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Your branch is up to date with 'origin/master'.
#
# Changes to be committed:
#       new file:   new_file.txt
#
```

While very powerful, vim is not the most user friendly text editor. You will need to type `i` (to insert), type your comment on the first line, then press `ESC` and type `:x` into the prompt at the bottom of the window to save.

Pushing to GitHub

- As stated previously, keeping a versioned backup of your project on your local file system has lots of advantages!
- However, ensuring that your code is backed up to a remote server has all of the same advantages, plus several more:
  - Secure backup in the event of a disaster (e.g. lost, stolen, broken computer).
  - View your files and version history in a browser.
  - More easily share you code.
  - Work from anywhere by downloading your project to any computer.
- You must first use the `git remote` command to connect your local repository to the remote.
- You must then must use `git push` to upload your code (and version history).

Begin by opening your repository in a browser, and copying the URL from the address bar.



https://github.com/RonWGit/my-repo-123

Next, use `git remote add origin <URL>` to configure your local repository to use your remote repository.

```
PS C:\Users\ron\Repo> git remote add origin
    https://github.com/RonWGit/my-repo-123
```

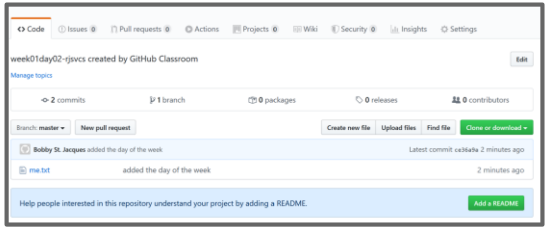
Then use `git push` to upload your changes.

```
PS C:\Users\ron\Repo> git push -u origin master
...lots of output
```

Push to GitHub

Configure your local repository to use your assignment repository as the remote repository and push your files to GitHub

- Copy the URL to your assignment repository from your browser.
- If necessary, launch a command prompt and navigate to your `Day02` directory.
- Use the `git remote add origin <URL>` command to connect your local repository to your assignment repository.
  - Replace `<URL>` with the URL to your assignment repository.
  - This will connect your local repository to the one at the specified `<URL>`.
- Use the `git push -u origin master` command to push your files to your assignment repository on GitHub.
  - You only need to use `-u origin master` the very first time that you push.
  - This will push your code to the `master` branch on the `origin` (your remote repository)
- Open your repository in your browser. Refresh if necessary. You should see that your files have been uploaded to GitHub.



week01day02-rjsvcs created by GitHub Classroom

2 commits | 1 branch | 0 packages | 0 releases | 0 contributors

Bobby St. Jacques added the day of the week — Latest commit ce36a9a 2 minutes ago

me.txt — added the day of the week — 2 minutes ago

Help people interested in this repository understand your project by adding a README.

GitHub & GitHub Classroom

- So far you have only backed up versions of your files on your local computer.
- There are still lots of benefits to doing this!
  - You still have a **local backup**.
  - You can revert to previous **versions**.
  - You can see the **history** of changes to your project.
- However, you do not have **disaster recovery**; if your computer is broken, lost, or stolen, you will lose all of your work.
- VCS (like Git) works best when you routinely move your latest changes to a separate computer. There are lots of advantages:
  - A safe, secure backup of your code.
  - More easily **share** your code with others.
  - Download your code on multiple computers.
  - View your code and version history in a browser.
- GitHub provides a free service that allows you to move your changes to their servers in "the cloud."

GitHub Classroom is integrated into GitHub. It makes it easy for instructors to make assignments that include separate, private repositories for every student.

You should have already accepted your first GitHub Classroom assignment for your homework. As part of this, you will have linked your GitHub account to your name in the roster.

We will now use another GitHub Classroom assignment to teach you how to connect your local repository to a remote repository on GitHub.

Cloning from GitHub

`git init != git clone`

It is important to note the differences between `git init` and `git clone`.

The `git init` command creates a **brand new**, ***empty*** repository in the local file system on your current computer.

You can then use `git remote` to connect it to a remote repository and push your code.

On the other hand, you can use `git clone` to download an **existing** remote repository to your local file system (even if the repository is empty).

This is especially useful when the remote repository already contains some code, and you won't need to use `git remote` to connect your local repository (when you clone, it is automatically connected).

- You do not always start a new project by creating a new, empty repository on your local computer.
- It is often the case that you would like to download an existing repository from GitHub to the file system of the computer on which you are working.
- For example:
  - You may use a lab computer to start your assignment in class, and you want to continue working on your laptop or PC in your dorm room.
  - A GitHub Classroom assignment may include **starter code** that you need to download before you can start working.
- Downloading a repository from GitHub into a new workspace on your computer is called **cloning**.
  - You use the `git clone <URL>` command to do this.
  - Make sure to use replace `<URL>` with the location of the repository that you want to clone, e.g. `https://www.github.com/RonWGit/my-repo-123`
- **Note:** `git clone` will only download changes that have been uploaded to the remote repository.
  - If you forget to push, you will lose your changes (or in the very least they will be stuck on the other PC).

1.2.7

Clone from GitHub

Create a new directory and clone your GitHub repository into it.

- If necessary, launch a command prompt and navigate to your `Unit01` directory.
- Use the `git clone <URL>` command to clone your repository into the directory.
  - Be sure to replace `<URL>` with the URL to your assignment repository.
- Verify that your files have been created including:
  - A subdirectory named the same as your repository.
    - e.g. `assignment-1-1-2-ronwgit`
  - The “`me.txt`” file inside the directory; open the file in Notepad to verify its contents.
  - The hidden `.git` folder (`ls -Force`).

1.2.8

Add Another File

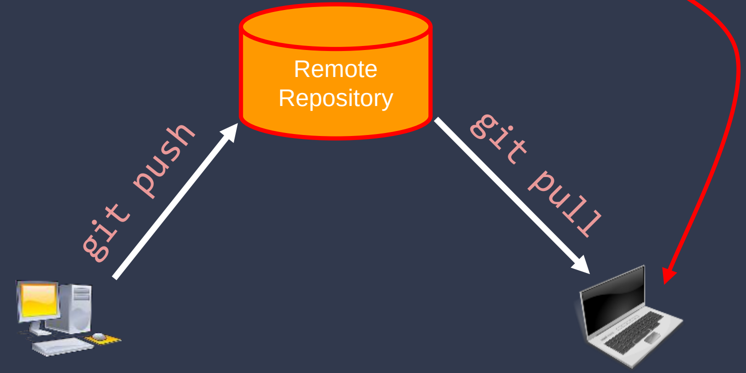
Add a new file to your cloned repository and push it to the remote repository.

- If necessary, launch a command prompt and navigate to the repository that your cloned from GitHub in the previous activity.
  - e.g. `SoftDevI\Unit01\assignment-1-1-2-ronwgit`
- Use Notepad to create a new file called "`primes.txt`". Add the first 10 prime numbers to the file, each on its own line.
  - Hint: 1 is not prime.
- Use the Git workflow to push the file to your remote repository.
  - `status`, `add`, `commit`, `push`.
- Open your repository in a browser and verify that the `primes.txt` file has has been uploaded.
- Now navigate to your `Day02` directory and list the files. What do you see?
  - e.g. `cd ..\Day02`

Pulling from Git

- There may be more than one local repository connected to a single remote repository.
  - This is the case when you are working on a team; every team member has their own local repository.
  - It is also the case when you are working alone, but use more than one computer, e.g. a lab computer in class and a PC in your dorm room.
- The remote repository always has the **most recent changes** that were pushed from any of the local repositories that are connected to it.
  - Conversely, it will **not** have changes in any local repository that have not been pushed.
  - This is **one** very good reason to follow the Git workflow, where changes are pushed every few minutes!
- This means that some of the local repositories may be **out of sync** with the remote repository.
  - Meaning that the local repository is **outdated** and doesn't have the most recent version of the project.
- In order to download the most recent changes from the remote repository, you will need to execute a **pull** using the `git pull` command.

When working from multiple computers (either by yourself or on a team) one or more local repositories will become **out of sync**.



For this reason, when switching from one computer to another, the very first thing you will do is a `git pull` to ensure that you have the most recent changes.

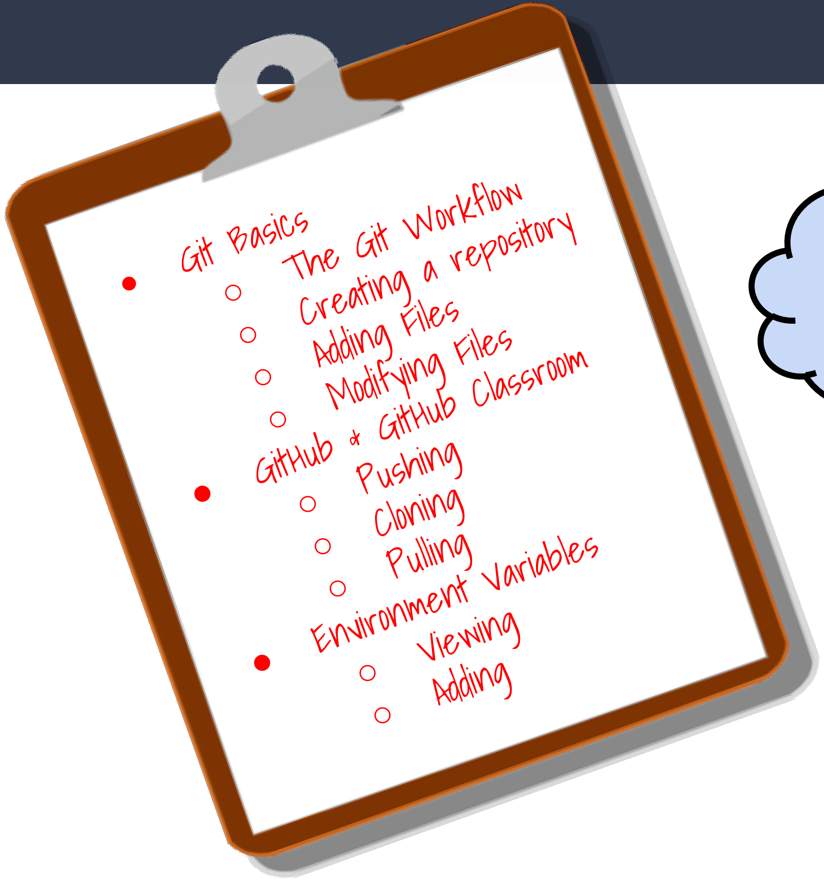
1.2.9

Pull from GitHub

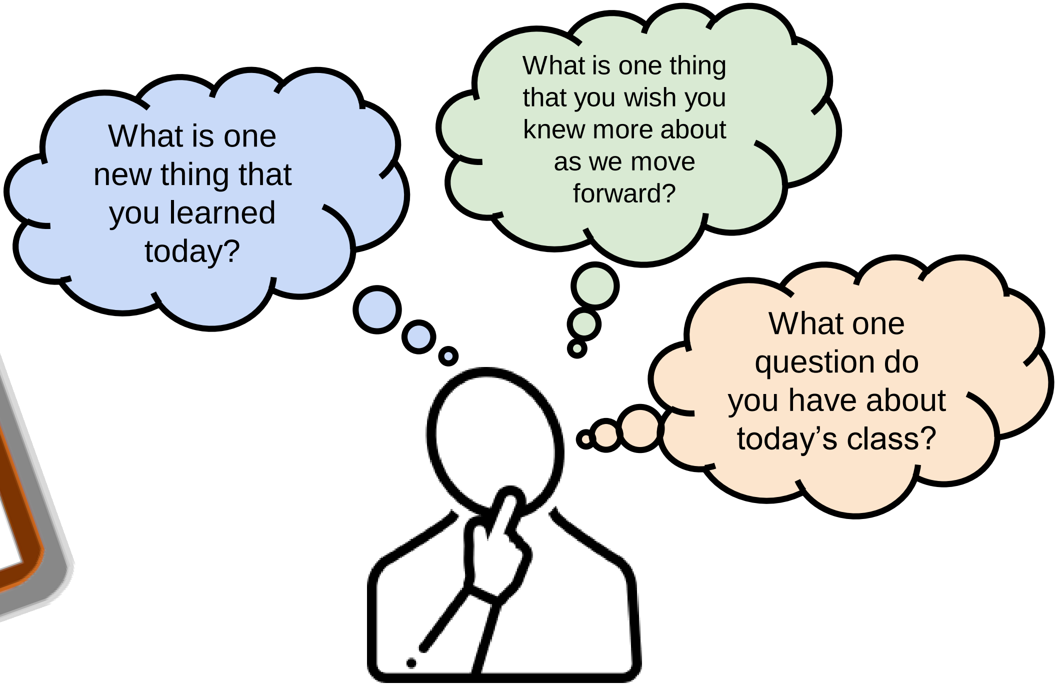
Your `Day02` repository is now **out of sync** with the remote repository on GitHub. Pull the most recent changes from GitHub into your `Day02` repository.

- If necessary, launch a command prompt and navigate to your `Day02` directory.
  - e.g. `SoftDevI\Unit01\Day02`
  - List the files in the directory to verify that "`primes.txt`" is not present.
- Use the `git pull` command to pull the latest version of the files in the repository into this local copy.
  - List the files in the directory to verify that "`primes.txt`" is now present.

Summary & Reflection



- Git Basics
  - The Git Workflow
  - Creating a repository
  - Adding Files
  - Modifying Files
- GitHub & GitHub Classroom
  - Pushing
  - Cloning
  - Pulling
- Environment Variables
  - Viewing
  - Adding



What is one new thing that you learned today?

What is one thing that you wish you knew more about as we move forward?

What one question do you have about today's class?

Please answer the questions above in your notes for today.

In case of fire

1. git commit
2. git push
3. leave building