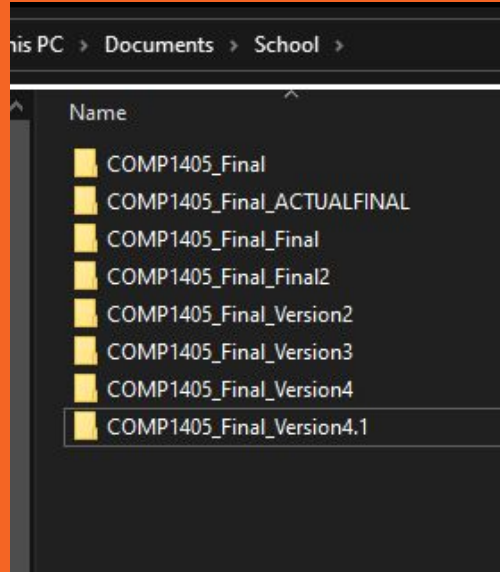

Git Workshop

CCSS - Shirley Zhan

Does your work look like this?



Use Git



- Git is a distributed **version control** system that tracks changes to files over time

- Git stores the entire history of a project, enabling easy rollback to previous states(no need to scramble and delete/edit code)

- It facilitates branching and merging, allowing developers to work on features independently and merge changes back into the main codebase

THE WITCHER WILD HUNT

v 4.00

- CONTINUE
- NEW GAME
- LOAD GAME
- OPTIONS
- MY REWARDS

La Cage au Fou - 15.12.2022 09:50:29



Autosave Slot

La Cage au Fou - 15.12.2022 09:28:05

Autosave Slot

La Cage au Fou - 15.12.2022 08:50:49

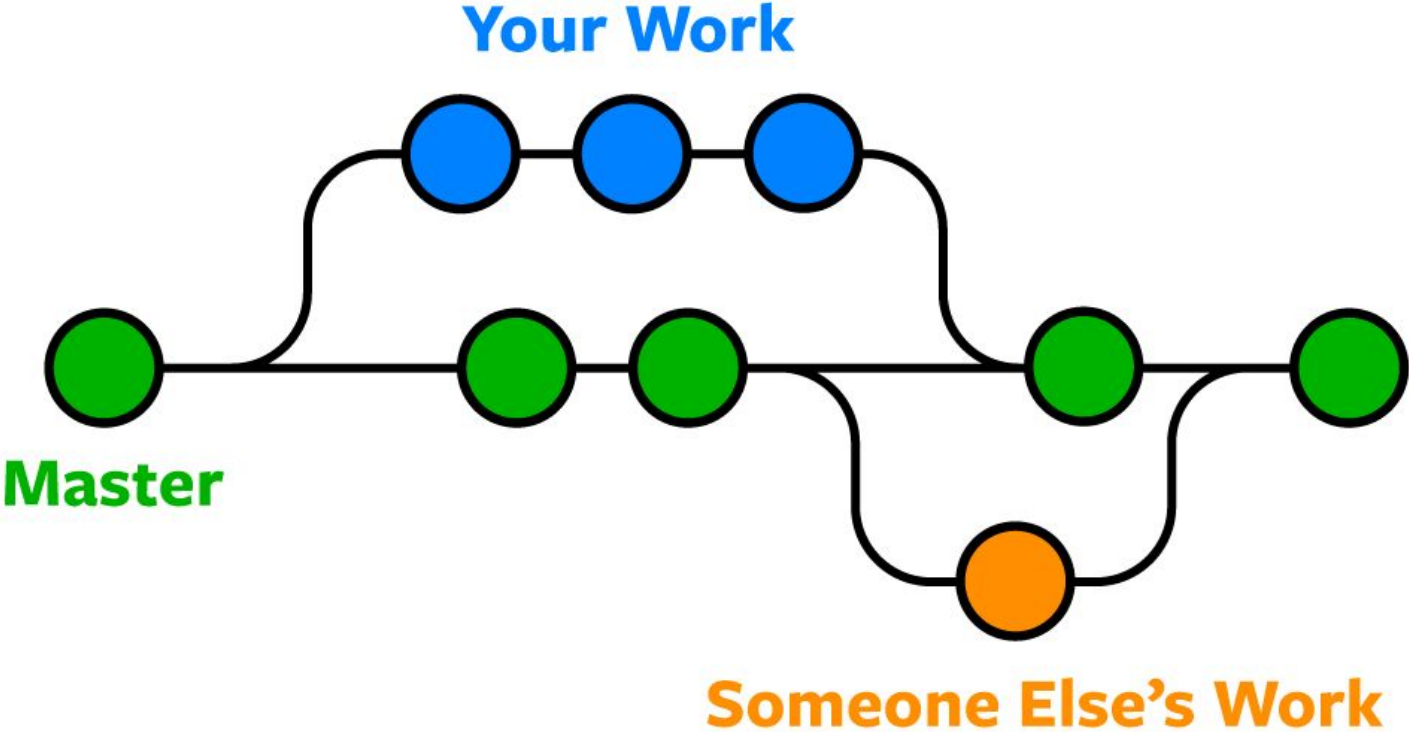


Manual Save Slot



- Load game
- Delete saved game
- Navigation
- Cross Progression
- Back

Crucial for collaboration in a team

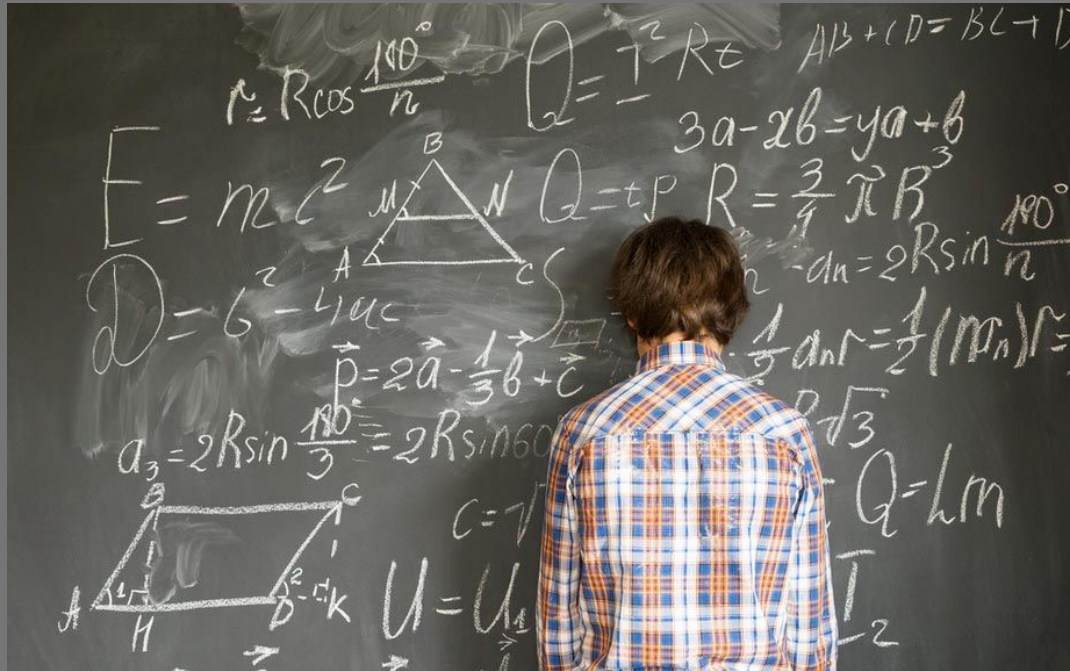


Common Git Commands



- `$git config`
- `$git init`
- `$git clone <path>`
- `$git add <file_name>`
- `$git commit`
- `$git status`
- `$git remote`
- `$git checkout <branch_name>`
- `$git branch`
- `$git push`
- `$git pull`
- `$git merge <branch_name>`
- `$git diff`
- `$git reset`
- `$git revert`
- `$git tag`
- `$git log`

This sounds so complicated...



First, some setup...

Install git, to check if it's installed, run **git -version**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS C:\Users\shirl\Documents\School\Git Workshop\git-workshop> git --version
git version 2.36.0.windows.1
PS C:\Users\shirl\Documents\School\Git Workshop\git-workshop> |
```

Now set up your name and email(identifiers)

```
C:\Users\shirl>git config --global user.name "Shirley Zhan"
C:\Users\shirl>git config --global user.email "shirleyzhan3@gmail.com"
C:\Users\shirl>
```


First, some setup...

Now let's set up **GitHub**, got to **GitHub** and make an account



Let's create our first project

First let's use `mkdir` to create a directory

```
C:\Users\shirl\Documents\CCSS>mkdir GitWorkshop  
C:\Users\shirl\Documents\CCSS>
```

First let's use `mkdir` to create a directory, then run `git init` to create a **git repository**. This step is crucial in running all git commands

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git init  
Initialized empty Git repository in C:/Users/shirl/Documents/CCSS/GitWorkshop/.git/  
C:\Users\shirl\Documents\CCSS\GitWorkshop>
```

Let's make a change

First by making a new txt file. Use the command `echo` to create a file and write to it

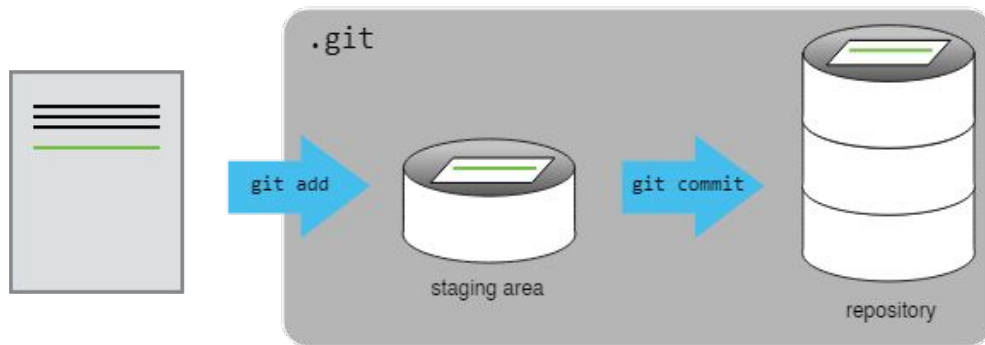
```
C:\Users\shirl\Documents\CCSS\GitWorkshop>echo This is a Git Workshop > Workshop  
.txt  
  
C:\Users\shirl\Documents\CCSS\GitWorkshop>
```

Make your first commit!

Use `git add <file_name>` to stage your files

Staging is like setting up your code to be committed. It's like the step in your mail sending where you put the letter in the envelope.

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git add Workshop.txt  
C:\Users\shirl\Documents\CCSS\GitWorkshop>
```



Make your first commit!

Now use `git commit -m 'message'` to commit the change. You have to write a message so make sure it's a cool one!

Committing is officially sending out the mail. It's saving the current progress that you've made. You would usually write a message because you want other people to know what your progress is.

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git commit -m "This is my first commit :)"
[master (root-commit) 519f41e] This is my first commit:)
1 file changed, 1 insertion(+)
create mode 100644 Workshop.txt

C:\Users\shirl\Documents\CCSS\GitWorkshop>
```

Look at what happened...

Use `git log` to see what happened(or use an extension)

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git log
commit 519f41e3bf069c7ede4e3677d23b1280fd60e784 (HEAD -> master)
Author: Shirley <shirleyzhan3@gmail.com>
Date:   Mon Sep 11 22:27:57 2023 -0400

    This is my first commit:)

C:\Users\shirl\Documents\CCSS\GitWorkshop>
```

Now let's make a change to a file

Add a line to your text file and save it. Use `git diff` to see what changes were made

☰ Workshop.txt

```
1 This is a Git Workshop
2 The contents of this workshop will help people learn about git
3
```

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git diff
diff --git a/Workshop.txt b/Workshop.txt
index 95cdc66..171f021 100644
--- a/Workshop.txt
+++ b/Workshop.txt
@@ -1,2 @@
-This is a Git Workshop
+This is a Git Workshop
+The contents of this workshop will help people learn about git

C:\Users\shirl\Documents\CCSS\GitWorkshop>
```

Let's make another change

Use `git add` to stage your changed file. Use `git status` to see what your current changes look like

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git add Workshop.txt

C:\Users\shirl\Documents\CCSS\GitWorkshop>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   Workshop.txt

C:\Users\shirl\Documents\CCSS\GitWorkshop>_
```


Let's try unstaging our changes

Use `git reset HEAD <file_name>` to remove files from the staging area.

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git reset HEAD Workshop.txt
```

```
Unstaged changes after reset:
```

```
M      Workshop.txt
```

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git status
```

```
On branch 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:   Workshop.txt
```

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

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>_
```

Now let's commit that change

Use `git commit` to commit the change.

Not how the message this time is different from when we initially committed. This is because we modified a file instead of creating a new one.




```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git commit -m "Added description to Workshop.txt"
[master bf85883] Added description to Workshop.txt
 1 file changed, 2 insertions(+), 1 deletion(-)

C:\Users\shirl\Documents\CCSS\GitWorkshop>_
```

Let's add our project to Github

Navigate to github and create a new repo


Owner **Repository name**


PUBLIC   ben ▾ / iOSApp 

Great repository names are short and memorable. Need inspiration? How about **drunken-dubstep**.

Description (optional)

iOS project for our mobile group

 **Public**
Anyone can see this repository. You choose who can commit.

 **Private**
You choose who can see and commit to this repository.

Initialize this repository with a README
This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: **None** ▾ | Add a license: **None** ▾ ⓘ

Create repository

Let's add our project to Github

Copy the link in the page and run the commands to add your project to github. Use **git remote** to access github and **git push** to add your local changes to github

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git remote add origin https://github.com/shirleyzhan00/GitWorkshop.git
```

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git push -u origin master
```

```
Enumerating objects: 6, done.
```

```
Counting objects: 100% (6/6), done.
```

```
Delta compression using up to 8 threads
```

```
Compressing objects: 100% (3/3), done.
```

```
Writing objects: 100% (6/6), 554 bytes | 554.00 KiB/s, done.
```

```
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
```

```
To https://github.com/shirleyzhan00/GitWorkshop.git
```

```
* [new branch]      master -> master
```

```
branch 'master' set up to track 'origin/master'.
```

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>
```

Let's push a change to github!

Use `git push origin <branch_name>` to push your commits

Pushing allows our **local** commits to be synced with the **remote**. You do not need to push after every commit. Push as often as you need

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git add .

C:\Users\shirl\Documents\CCSS\GitWorkshop>git commit -m "Adding another file"
[master 96402fa] Adding another file
 1 file changed, 5 insertions(+)
 create mode 100644 Test.txt

C:\Users\shirl\Documents\CCSS\GitWorkshop>git push origin master
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% (7/7), 758 bytes | 379.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/shirleyzhan00/GitWorkshop.git
   bf85883..96402fa  master -> master

C:\Users\shirl\Documents\CCSS\GitWorkshop>_
```

Let's make a new branch

We currently just have main

In large projects, usually there are multiple branches for each feature which are separate from each other

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git branch
* master
C:\Users\shirl\Documents\CCSS\GitWorkshop>_
```

Let's make a new branch

Use `git checkout -b <branch_name>` to create a new branch and switch to it

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git checkout -b Feature-Branch-Shirley
Switched to a new branch 'Feature-Branch-Shirley'
C:\Users\shirl\Documents\CCSS\GitWorkshop>
```

Let's make some changes to new branch

For simplicity's sake, let's just create a bunch of text files. Use a **bat(multiple commands)** file to make life easier for yourselves :))



*Untitled - Notepad

File Edit Format View Help

```
echo This is file 1 > Data1.txt
echo This is file 2 > Data2.txt
echo This is file 3 > Data3.txt
echo This is file 4 > Data4.txt
echo This is file 5 > Data5.txt
echo This is file 6 |> Data6.txt
```

File name: command.bat

Save as type: All Files (*.*)

^ Hide Folders

Encoding: UTF-8

Save

Cancel

Let's make some changes to new branch

Run the bat file and remove it so it's not in our commit

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>command.bat  
  
C:\Users\shirl\Documents\CCSS\GitWorkshop>echo This is file 1 1>Data1.txt  
C:\Users\shirl\Documents\CCSS\GitWorkshop>echo This is file 2 1>Data2.txt  
C:\Users\shirl\Documents\CCSS\GitWorkshop>echo This is file 3 1>Data3.txt  
C:\Users\shirl\Documents\CCSS\GitWorkshop>echo This is file 4 1>Data4.txt  
C:\Users\shirl\Documents\CCSS\GitWorkshop>echo This is file 5 1>Data5.txt  
C:\Users\shirl\Documents\CCSS\GitWorkshop>echo This is file 6 1>Data6.txt  
C:\Users\shirl\Documents\CCSS\GitWorkshop>del /f command.bat  
C:\Users\shirl\Documents\CCSS\GitWorkshop>
```

Let's make some changes to new branch

Let's stage the changes

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git add .
```

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git status
```

```
On branch Feature-Branch-Shirley
```

```
Changes to be committed:
```

```
(use "git restore --staged <file>..." to unstage)
```

```
new file:   Data1.txt
```

```
new file:   Data2.txt
```

```
new file:   Data3.txt
```

```
new file:   Data4.txt
```

```
new file:   Data5.txt
```

```
new file:   Data6.txt
```

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>
```

Git Stash

If you want to switch branches but don't want to commit yet, **stash** those changes and **pop** them later back. If you had switched without stashing or committing, you would have lost the changes you made.

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git stash
Saved working directory and index state WIP on Feature-Branch-Shirley: 96402fa A
dding another file

C:\Users\shirl\Documents\CCSS\GitWorkshop>_
```

Merging

Let's work in the master branch and create another file(note that it's the same as a file in another branch...)

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>echo This is a conflict > Data1.txt
C:\Users\shirl\Documents\CCSS\GitWorkshop>git add .
C:\Users\shirl\Documents\CCSS\GitWorkshop>git commit -m "Adding a new text file"

[master 17ad3c2] Adding a new text file
 1 file changed, 1 insertion(+)
 create mode 100644 Data1.txt
C:\Users\shirl\Documents\CCSS\GitWorkshop>_
```

Merging

Use `git stash pop` and get back all the changes we've made

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git stash pop
On branch Feature-Branch-Shirley
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   Data1.txt
    new file:   Data2.txt
    new file:   Data3.txt
    new file:   Data4.txt
    new file:   Data5.txt
    new file:   Data6.txt

Dropped refs/stash@{0} (247041f432667ff1619decd5dcc325d4bc162040)

C:\Users\shirl\Documents\CCSS\GitWorkshop>
```

Merging

Let's say we're done with the feature in the feature branch and we want to merge. Switch to the branch you want to **merge into** and type `git merge <feature_branch_name_you_want_to_merge>`

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git merge Feature-Branch-Shirley
Auto-merging Data1.txt
CONFLICT (add/add): Merge conflict in Data1.txt
Automatic merge failed; fix conflicts and then commit the result.

C:\Users\shirl\Documents\CCSS\GitWorkshop>
```

Merge Conflicts

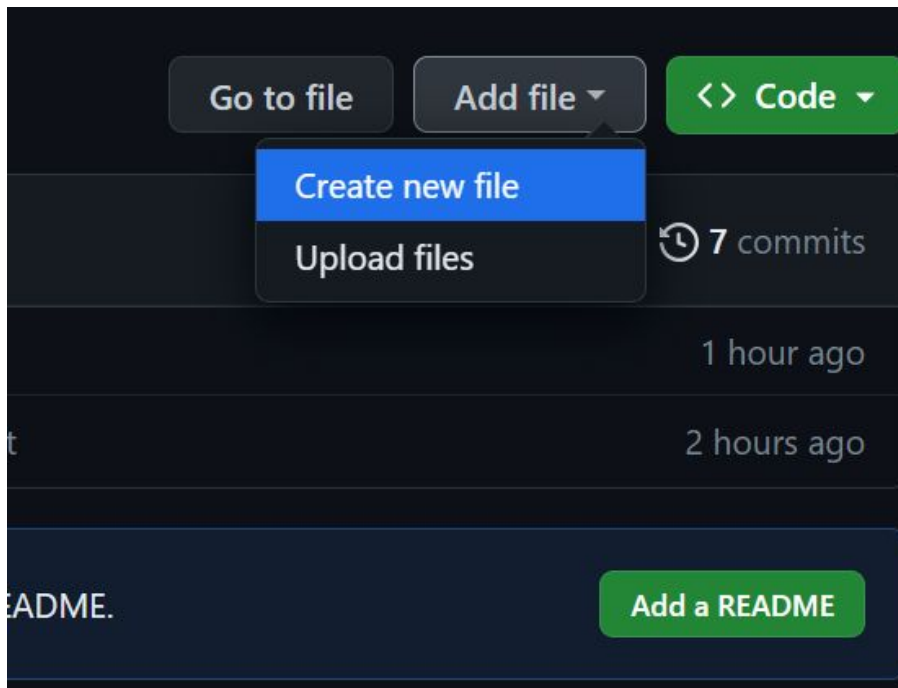
Merge conflicts happen when the same file is worked on by 2 different branches. You usually need to work with the other person working on the other branch to resolve it.

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>more Data1.txt
<<<<<<< HEAD
This is a conflict
=====
This is file 1
>>>>>>> Feature-Branch-Shirley

C:\Users\shirl\Documents\CCSS\GitWorkshop>
```

Pulling Changes

Sometimes changes are pushed to the remote and your branch might not have it yet. Let's simulate this by creating a file in github directly



Pulling Changes

Do `git pull` to get those changes.

```
C:\Users\shirl\Documents\CCSS\GitWorkshop>git pull
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (7/7), 1.90 KiB | 102.00 KiB/s, done.
From https://github.com/shirleyzhan00/GitWorkshop
   96402fa..832146a  master    -> origin/master
Merge made by the 'ort' strategy.
 NewFile | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 NewFile

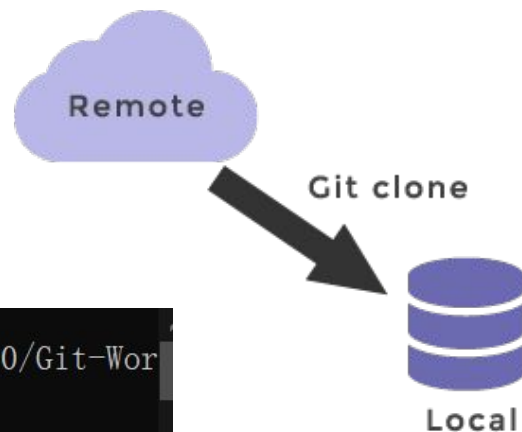
C:\Users\shirl\Documents\CCSS\GitWorkshop>_
```

Let's clone your first project!

Usually at work, you would need to **clone** a project and get a local copy of it. The command you would use is **git clone**

```
C:\Users\shirl\Documents\CCSS>git clone https://github.com/shirleyzhan00/Git-Workshop.git
Cloning into 'Git-Workshop'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 18 (delta 4), reused 11 (delta 2), pack-reused 0
Receiving objects: 100% (18/18), done.
Resolving deltas: 100% (4/4), done.

C:\Users\shirl\Documents\CCSS>
```



New Branches

Most, if not all projects on github don't let you directly commit to the main branch. You need to create a new branch and create a pull request to merge with the main branch.

```
C:\Users\shirl\Documents\CCSS\Git-Workshop>echo test > text.txt
C:\Users\shirl\Documents\CCSS\Git-Workshop>git add .
C:\Users\shirl\Documents\CCSS\Git-Workshop>git commit -m "done"
[main a38014f] done
 1 file changed, 1 insertion(+)
 create mode 100644 text.txt
C:\Users\shirl\Documents\CCSS\Git-Workshop>git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 309 bytes | 309.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Bypassed rule violations for refs/heads/main:
remote:
remote: - Changes must be made through a pull request.
remote:
To https://github.com/shirleyzhan00/Git-Workshop.git
 e138676..a38014f main -> main
C:\Users\shirl\Documents\CCSS\Git-Workshop>
```

Challenge!

The demo.py is a simple calculator. Create a **new branch** and add some changes to it. You can add a new feature or just add a text file.

```
C:\Users\shirl\Documents\CCSS\Git-Workshop>python demo.py
Simple Python Calculator
Operations:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exponent
Enter operation 1, 2, 3, 4, or 5 or 'exit' to quit: _
```

Pull Request

When you're ready, use the command `git push --set-upstream origin <branch_name>` to push your branch. Make a pull request so the owner can see it and approve it

```
C:\Users\shirl\Documents\CCSS\Git-Workshop>git push --set-upstream origin test
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 322 bytes | 322.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'test' on GitHub by visiting:
remote:   https://github.com/shirleyzhan00/Git-Workshop/pull/new/test
remote:
To https://github.com/shirleyzhan00/Git-Workshop.git
 * [new branch]      test -> test
branch 'test' set up to track 'origin/test'.

C:\Users\shirl\Documents\CCSS\Git-Workshop>_
```