

# *Git vs. GitHub and Version Control, an Introduction*



# *Git vs. GitHub and Version Control, an Introduction*

*First things first — Git is not GitHub. This is a common mistake people make.*



# ***Which of the following best summarizes the difference?***

- A.** GitHub is the open-source version of Git*
- B.** Git is a version control system. GitHub is an online service that interfaces well with Git*
- C.** Git is the command line version of GitHub*
- D.** Git is primarily used for storage whereas GitHub is used for projects*

# *Git*

- *A program you run from the command line*
- *A distributed version control system*
- *Programmers use Git so they can keep a history of all changes made to their code.*

# *GitHub*

- *A hosting service for Git repositories*
- *A web interface to explore Git repositories*

# ***What is GitHub Enterprise (GHE)?***

- *Professional application of Github*
- *Where GitHub is a public ‘Social Network’ for programming and programmers. Github Enterprise is a private, professional application of GitHub.*
- *Both use Git*

# ***‘Git’ on GitHub and GitHub Enterprise:***

- *For those of you that have not already set up your GitHub and GitHub Enterprise accounts, lets take a minute to do that now.*

*GitHub*

*GitHub Enterprise for General Assembly*

# *What about Commands?*

- *You can take a look at the list of available git commands by running (there are a lot of commands:*

*\$ git help -a*

- *We only really need about 10 git commands, probably less.*

# Code-Along (20 min)?

- *Create a directory on your Desktop*

```
mgalarnyk@55Y4KC2 MINGW64 ~  
$ cd Desktop  
  
mgalarnyk@55Y4KC2 MINGW64 ~/Desktop  
$ mkdir hello-world
```

- *You can place this directory under Git revision control*

```
mgalarnyk@55Y4KC2 MINGW64 ~/Desktop  
$ cd hello-world  
  
mgalarnyk@55Y4KC2 MINGW64 ~/Desktop/hello-world  
$ git init  
Initialized empty Git repository in C:/Users/mgalarnyk/Desktop/hello-world/.git/
```



# *The .git folder*

- *We can look at the contents of the empty folder*

```
mgalarnyk@55Y4KC2 MINGW64 ~/Desktop/hello-world (master)
$ ls -A
.git/
```

- *Notice there is a hidden folder called .git*
  - *This is where all the information about your repository is stored. This is no need to make any changes to this folder.*
- You can control all of the Git flow using git commands*

# Adding a File

- Create a new file

*\$ touch a.txt*

- Show the *untracked* files using

*\$ git status*

- We can use the *git add* command to add the file to the temporary staging area (git calls it the “index”)

*\$ git add .*

```
mgalarnyk@55Y4KC2 MINGW64 ~/Desktop/hello-world (master)
$ touch b.txt

mgalarnyk@55Y4KC2 MINGW64 ~/Desktop/hello-world (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        b.txt

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

mgalarnyk@55Y4KC2 MINGW64 ~/Desktop/hello-world (master)
$ git add .
```

# *Commit*

- *To permanently store the contents of the index of the repository, (i.e. commit these changes to the “HEAD”), you need to run the following command:*

*\$ git commit -m “Good practice to say what you did”*

```
mgalarnyk@55Y4KC2 MINGW64 ~/Desktop/hello-world (master)
$ git commit -m "Added a.txt because it was needed for the unit test"
```

# *Checking the Log*

- *If we want to view the commit history, we can run:*

*\$ git log*

```
mgalarnyk@55Y4KC2 MINGW64 ~/Desktop/hello-world (master)
$ git log
commit 3a8be2850e0b3aa873f84e005854b207b7bc1001 (HEAD -> master)
Author: Michael Galarnyk <mgalarnyk@daymon.com>
Date: Thu Aug 3 12:52:39 2017 -0700

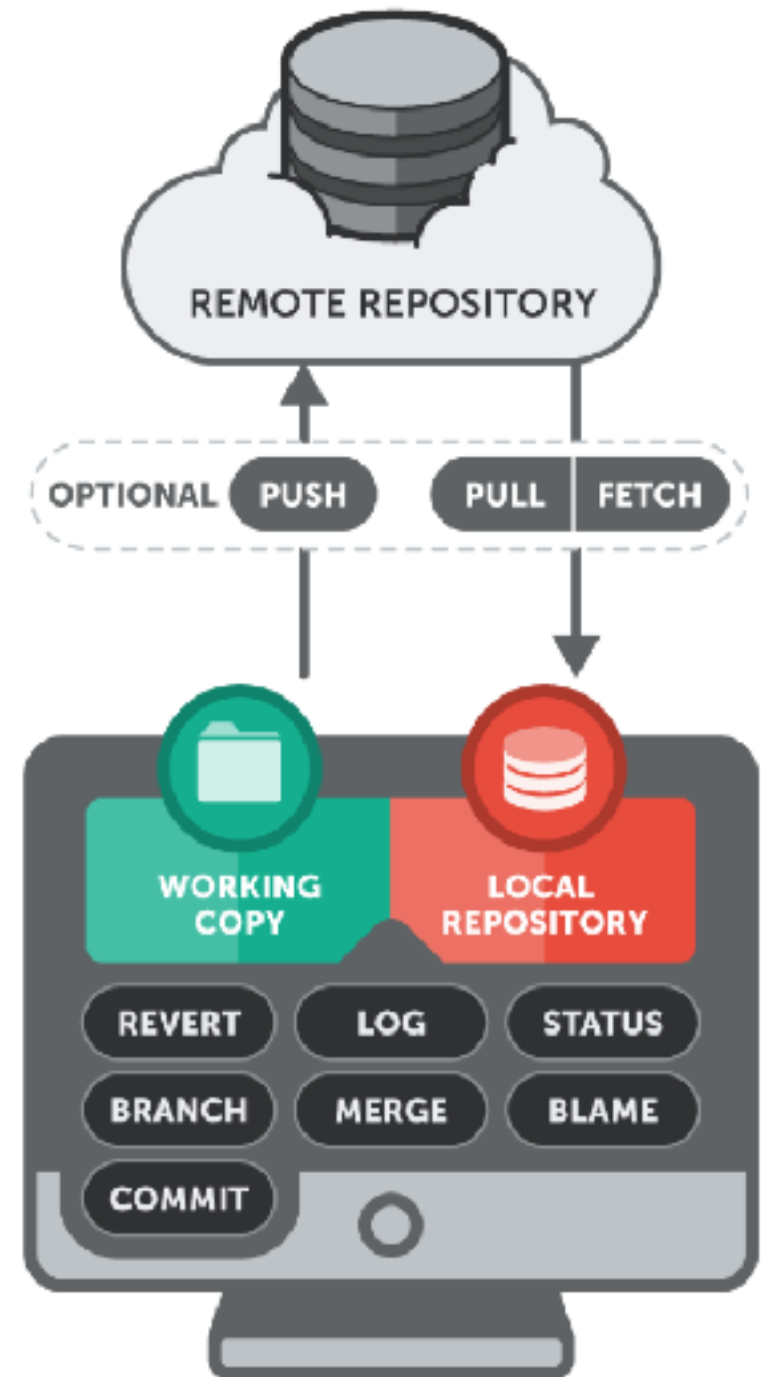
    Added a.txt because it was needed for the unit test
```

# ***Making and Cloning Repositories: Code-Along (20 min)***

*Typically we have two repository locations:*

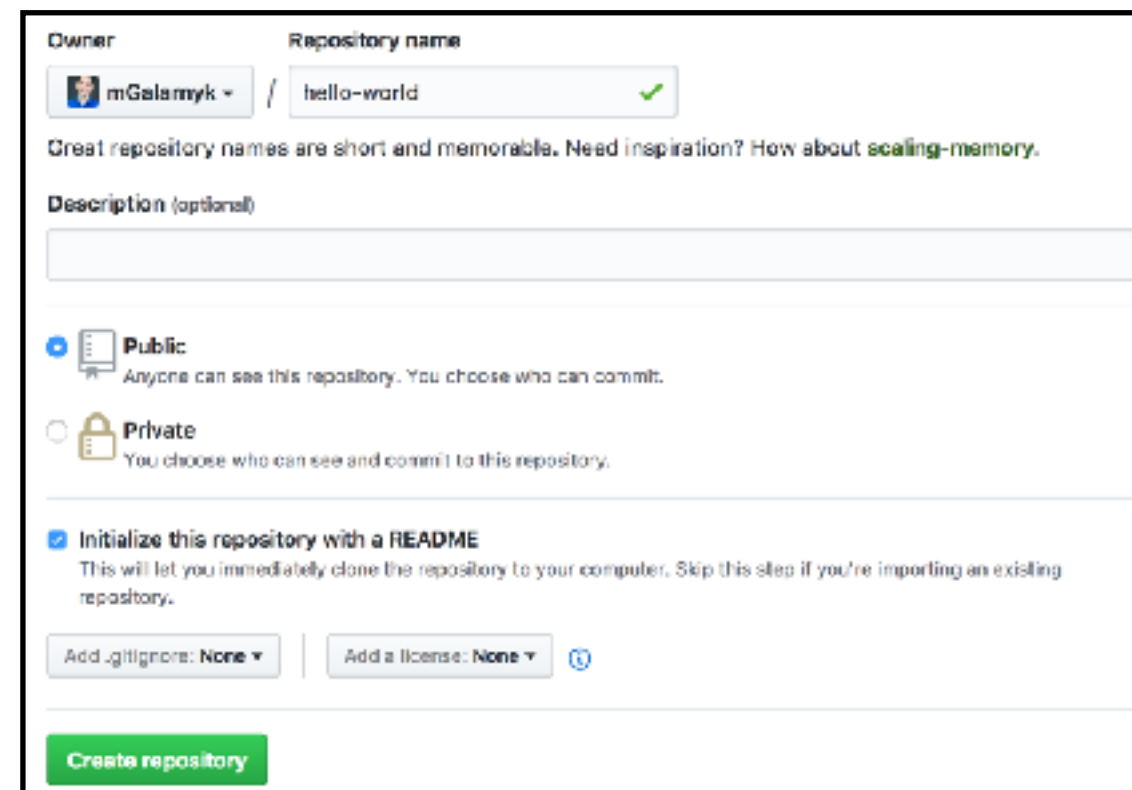
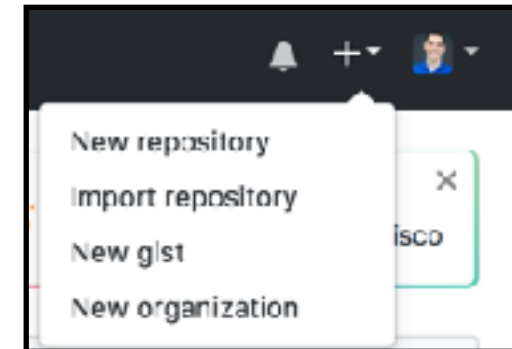
## ***Remote and Local***

- **Remote:** *Repositories that are NOT stored in our current location/machine. Usually where we store the repo.*
- **Local:** *Repositories that are stored on our current machine. Usually where we work on the repo*



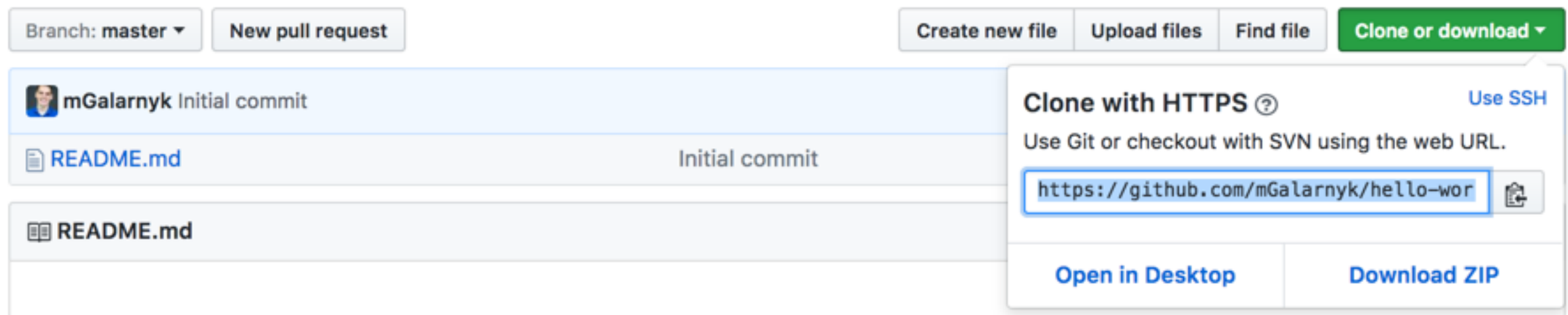
# Making a Repo

1. Go to your GitHub account
  2. On the right-hand side, hit the + button for New Repository.
  3. Name your repository hello-world
  4. **Initialize this repository with a README.**  
(Now we can git pull)
  5. Click the big, green Create Repository button.
- We now need to connect our local Git repository with our newly created remote repository on GitHub. We have to add a “remote” repository, an address where we can send our local files to be stored.

A screenshot of the GitHub 'Create new repository' form. The form has two main sections: 'Owner' and 'Repository name'. The 'Owner' section shows a dropdown menu with 'mGalamyk' selected. The 'Repository name' section shows a text input with 'hello-world' and a green checkmark. Below these sections, there is a text input for 'Description (optional)'. The 'Public' option is selected, with a description: 'Anyone can see this repository. You choose who can commit.' The 'Private' option is also visible, with a description: 'You choose who can see and commit to this repository.' Below the visibility options, there is a checkbox for 'Initialize this repository with a README', which is checked. A description for this checkbox says: 'This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.' At the bottom, there are two dropdown menus: 'Add .gitignore: None' and 'Add a license: None'. A large green 'Create repository' button is at the bottom right.

# *Making a Repo*

On the right-hand side of your GitHub there should be a green 'Clone or download' button. This button should reveal a tiny window with a URL. Copy the provided URL, which is the path to this remote repo.



*Next, change directories to hello-world then type*

```
mgalarnyk@55Y4KC2 MINGW64 ~/Desktop/hello-world (master)
$ git remote add origin https://github.com/mGalarnyk/hello-world.git
```



# *Pushing to GitHub*

*In order to send files from our local machine to our remote repository on GitHub, we need to use the command `git push`. However, you also need to add the name of the remote repo — in this case, we call it `origin` — and the name of the branch, in this case `master`*

*`git push origin master`*

*(this won't work **yet**)*

```
$ git push origin master
To https://github.com/mGalarnyk/hello-world.git
 ! [rejected]          master -> master (fetch first)
error: failed to push some refs to 'https://github.com/mGalarnyk/hello-world.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```



# *Pulling from GitHub*

*Just as we added the README.md in our repo, we need to first pull that file from our local repository to check that we don't have a "conflict".*

*git pull origin master —allow-unrelated-histories*

*git status*

*git add .*

*git commit -m "Added a README.md"*

*Note: 99.9% of the time, the command **git pull origin master** without the —allow-unrelated-histories is enough.*

# *Pulling from GitHub*

*Anyone notice something  
odd/wrong here?*

```
mgalarnyk@55Y4KC2 MINGW64 ~/Desktop/hello-world (master)
$ git pull origin master --allow-unrelated-histories
From https://github.com/mGalarynk/hello-world
 * branch                master      -> FETCH_HEAD
Merge made by the 'recursive' strategy.
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

mgalarnyk@55Y4KC2 MINGW64 ~/Desktop/hello-world (master)
$ git status
On branch master
nothing to commit, working tree clean

mgalarnyk@55Y4KC2 MINGW64 ~/Desktop/hello-world (master)
$ git add .

mgalarnyk@55Y4KC2 MINGW64 ~/Desktop/hello-world (master)
$ git commit -m "Added a README.md"
On branch master
nothing to commit, working tree clean

mgalarnyk@55Y4KC2 MINGW64 ~/Desktop/hello-world (master)
$ git push origin master
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 540 bytes | 0 bytes/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/mGalarynk/hello-world.git
 cef9be8..33b4a6c  master -> master
```

# Checking the Log Part 2

```
mgalarnyk@55Y4KC2 MINGW64 ~/Desktop/hello-world (master)
$ git log
commit 33b4a6c476cf999e7e9f906229288c6c6272b084 (HEAD -> master, origin/master)
Merge: 3a8be28 cef9be8
Author: Michael Galarnyk <mgalarnyk@daymon.com>
Date: Thu Aug 3 14:01:56 2017 -0700

    Merge branch 'master' of https://github.com/mGalarnyk/hello-world

commit cef9be8c77b70a69e70134a27f3aafac6508fef6
Author: Michael Galarnyk <mGalarnyk@users.noreply.github.com>
Date: Thu Aug 3 13:28:12 2017 -0700

    Initial commit

commit 3a8be2850e0b3aa873f84e005854b207b7bc1001
Author: Michael Galarnyk <mgalarnyk@daymon.com>
Date: Thu Aug 3 12:52:39 2017 -0700

    Added a.txt because it was needed for the unit test
```

*Notice, we are missing a commit message.*

*Refresh your GitHub web page and files should appear.*

The screenshot shows the GitHub interface for a repository named 'hello-world' by user 'mGalarnyk'. The top navigation bar includes links for 'Pull requests', 'Issues', 'Marketplace', and 'Gist'. The repository header shows 'Unwatch' (1), 'Star' (0), and 'Fork' (0). Below the header, tabs for 'Code', 'Issues' (0), 'Pull requests' (0), 'Projects' (0), 'Wiki', 'Settings', and 'Insights' are visible. A message states 'No description, website, or topics provided.' with an 'Add topics' link and an 'Edit' button. A summary bar indicates '3 commits', '1 branch', '0 releases', and '1 contributor'. Action buttons include 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The commit history shows a merge by Michael Galarnyk and two file additions: 'README.md' and 'a.txt'.

File	Commit Message	Time
Michael Galarnyk	Merge branch 'master' of <a href="https://github.com/mGalarnyk/hello-world">https://github.com/mGalarnyk/hello-world</a>	Latest commit 33b4a6c 27 minutes ago
README.md	Initial commit	an hour ago
a.txt	Added a.txt because it was needed for the unit test	2 hours ago

*Using the git remote add method is useful for taking an existing **local** repository and creating a new **remote** one*

# ***Independent Practice***

*Repeat the steps above, starting with the creation of your remote repo, to create a local and remote repository with your GitHub Enterprise account.*

# *Cloning your First Repository*

## **Cloning your first repository**

Alternatively, you can clone this repo from your GitHub link to have it automatically configured in a new directory.

```
git clone https://github.com/github-username/  
hello-world.git
```

Now that everyone has a repository on GitHub, let's clone one!

Cloning allows you to make a local copy of a remote repository.

Navigate back to your Desktop, and **delete your hello-world repository**.

```
cd ~/Desktop  
rm -rf hello-world
```

# *Cloning your First Repository*

Now, ask the person sitting next to you for his or her GitHub name and navigate to that repository on GitHub.

## **Clone that repo!**

To retrieve the contents of the repo, all you need to do is:

```
$ git clone git@github.com:<partners-github-username>/hello-world.git
```

Git should reply:

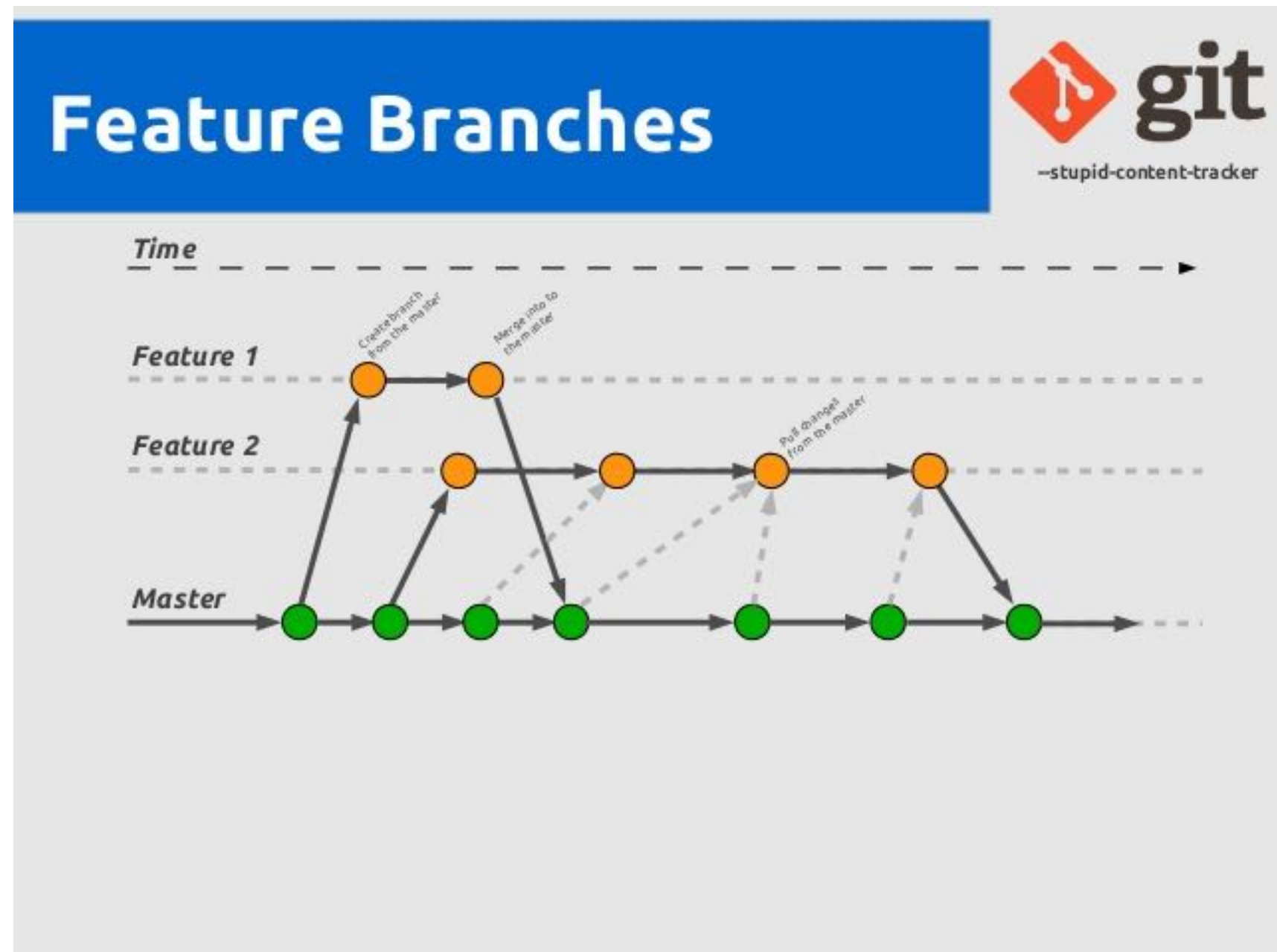
```
Cloning into 'hello-world'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 3 (delta 0),
pack-reused 0
Receiving objects: 100% (3/3), done.
Checking connectivity... done.
```

You've now cloned your first repository!

# What is a Pull Request?

Often times when we are working on a project with others, we all can't work on the same repo at the same time so we have things called branches.

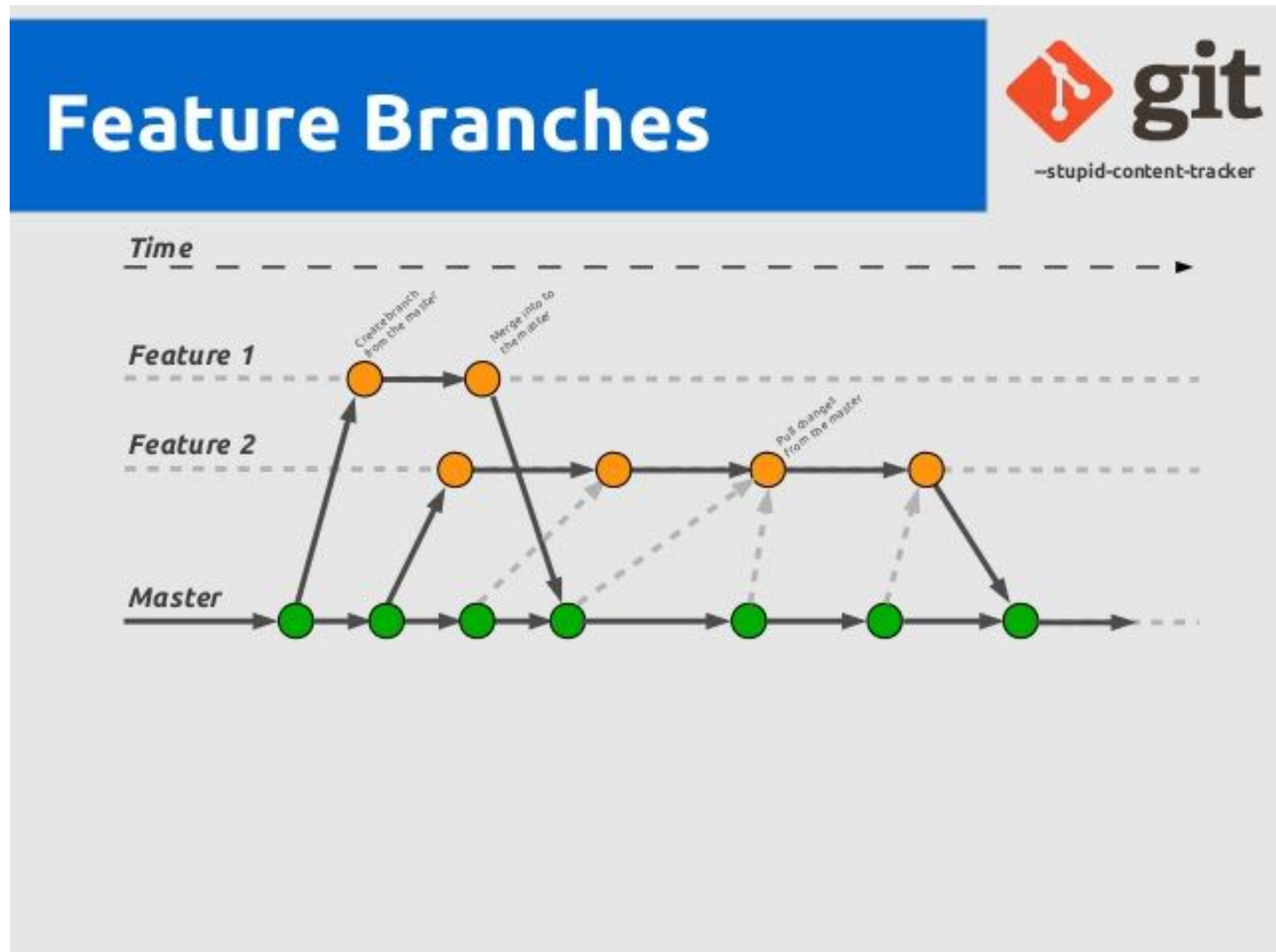
With branches, we can create a branch off of the main repo or “Master Branch”, perform our work whether that be additions or alterations and then merge the changes we made on our branch back to the master.






# What is a Pull Request?

- Pull requests are as a means of queuing and validating merges.
- Requests go through an administrator who can review your changes and additions and approve or deny your request.



# *Create a Pull Request on GitHub: Code-Along (15 min)*

Before you can open a pull request, you must create a branch in your local repository, commit to it, and push that branch to a repository or fork on GitHub.

1. Visit the repository you've pushed to.
2. Click "Branch:master" on the left-hand side and create a branch called 'branch-edits' by typing in the box and hitting enter. This should put you on your new branches page.
3. Make sure you're on your new branch and in GitHub create a new '.md' or '.txt' file using the 'Create new file' button. Call it what you like and place some text in it.
4. Click the "Compare & pull request" button in the repository . 
5. You'll land right onto the compare page. Next, you can click "Edit" at the top to pick a new branch to merge to using the "Head Branch" dropdown menu.
6. Select the target branch that your branch should be merged to using the "Base Branch" dropdown menu.
7. Review your proposed change.
8. Hit "Click to create a pull request" for this comparison.
9. Enter a title and description for your pull request.
10. Click "Send pull request."

As you are working on your own repo in your own GitHub you will be able to merge the branch at your leisure.

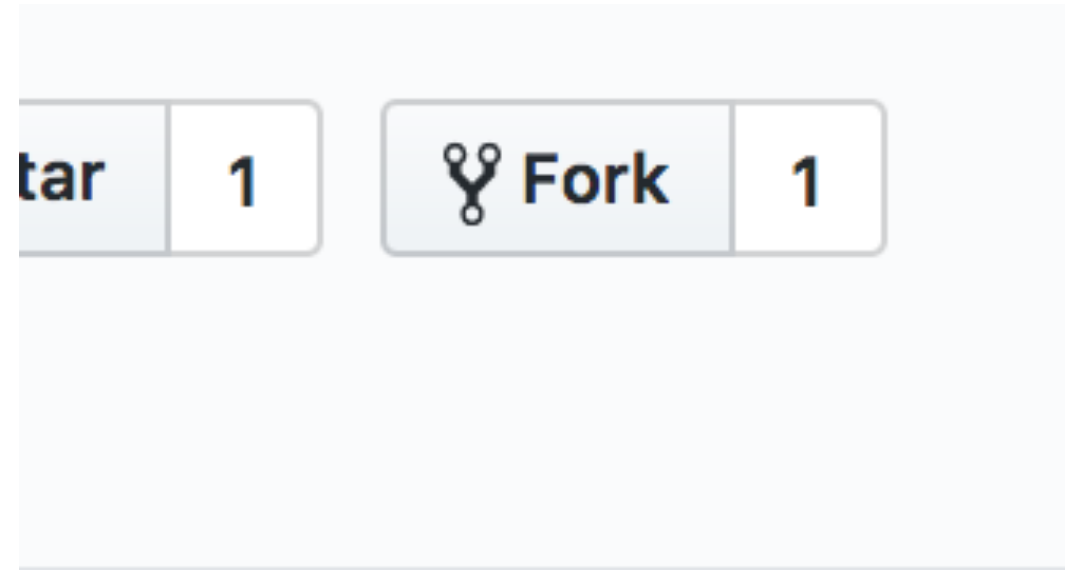
# ***Why is it called a Pull Request?***

*Even though we are trying to **push** our information into the master, it is called a pull request because we are making a request to the administrator to **pull** our branch into the master.*

# *How we will use GitHub?*

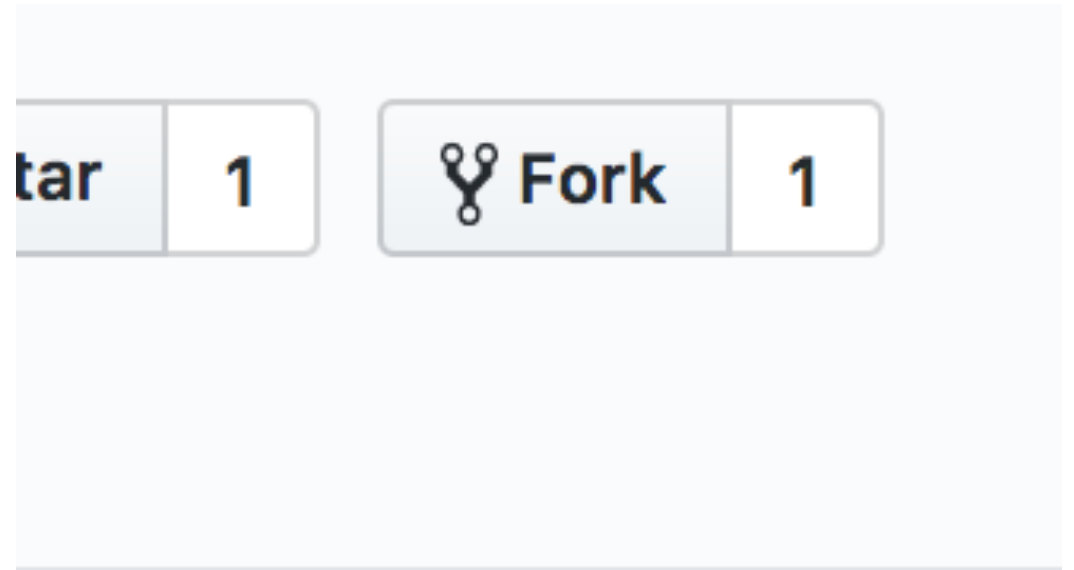
*We will have a central class repository where all the lesson materials are housed.*

*You will create a **Forked** version of the master repo which will be yours to do with as you please.*



# *What is a Fork?*

- *Forked are personal copies of repositories*
- *If the original is updated, you can **pull** those changes to your copy via*  
*\$ git pull upstream master*



# ***Assess: Independent Practice (10 min)***

- *With a partner, take turns explaining the following commands pretending they are unfamiliar with Git and this is their first exposure to it.*

*init, add, commit, push, pull, and clone*

# *Lesson Review: Git and GitHub*

Do you feel comfortable with Git, GitHub and GitHub Enterprise?  
As we'll be using them in a lot of our coursework, let's make sure you are!

We understand that Git and GitHub can be difficult to mentally grasp initially, so if you have any questions about the following or other Git aspects please ask now.

- Basic Git commands, `init`, `add`, `commit`, `push`, `pull`, and `clone`
- Local and Remote repositories
- Cloning, Branching, Merging and Pull Requests