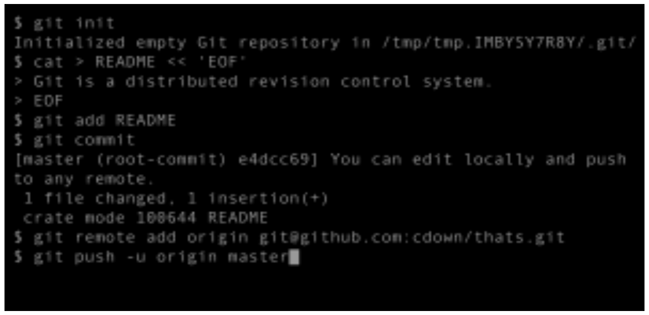
git basics

Nicole Sultanum

JSC 270



```
$ git init
Initialized empty Git repository in /tmp/tmp.IMBY5Y7R8Y/.git/
$ cat > README << 'EOF'
> Git is a distributed revision control system.
> EOF
$ git add README
$ git commit
[master (root-commit) e4dcc69] You can edit locally and push
to any remote.
 1 file changed, 1 insertion(+)
 crate mode 100644 README
$ git remote add origin git@github.com:cdown/thats.git
$ git push -u origin master
```

https://git-scm.com/

Git is a **version control system**

- It allows you to have multiple collaborators working on the same project at the same time
  - Various permission levels
  - Branches – i.e., versions of a project that can be worked on in parallel
- Also useful for one-person projects, to keep track of different versions of a code base
- Can be used to track 'binary' files, but is really meant to keep track of 'text' files – more specifically code files

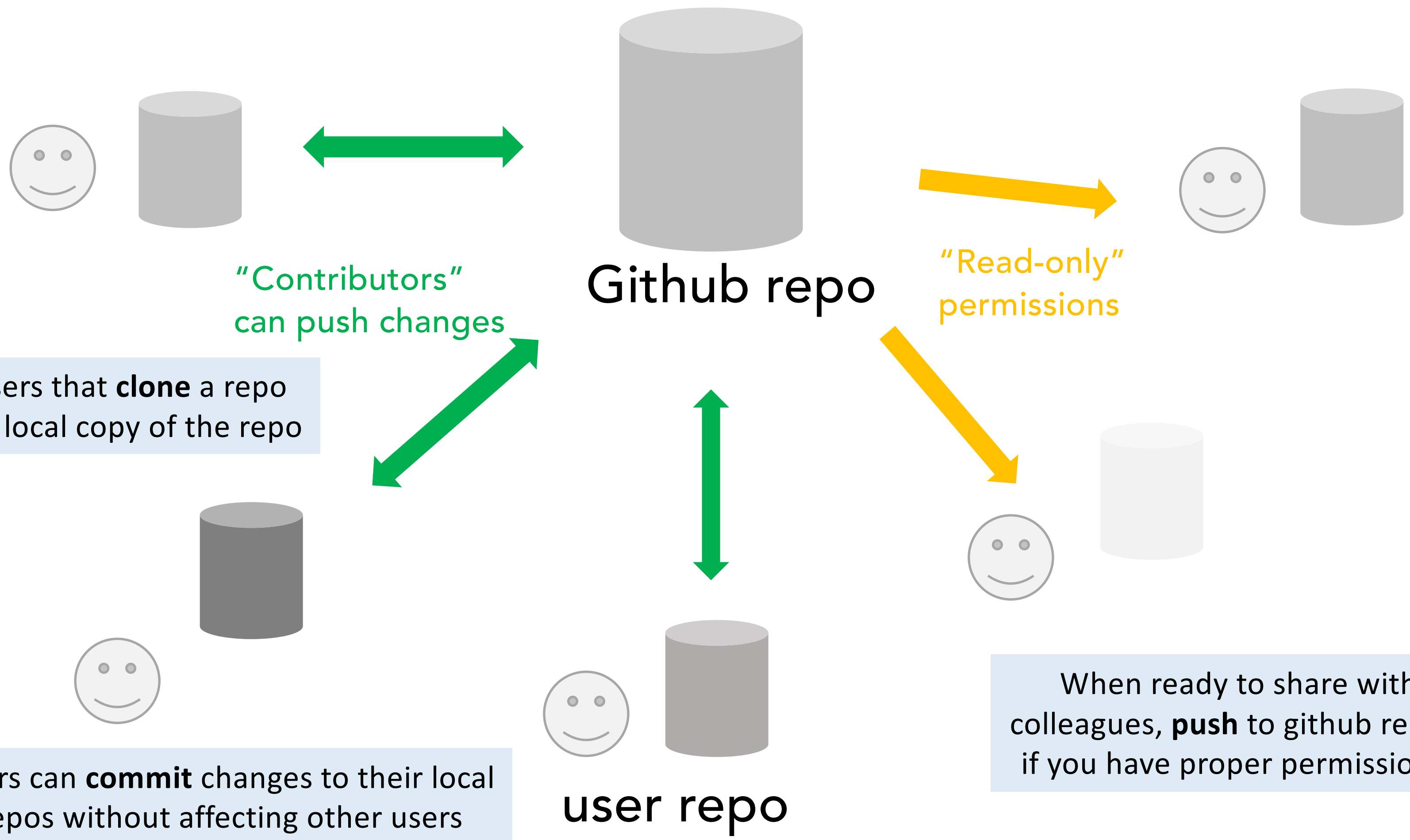
What is *github*?

http://github.com/

Github is an **online service to host git repositories**

- For teams: a centralized source to maintain a shared codebase
- For one-person projects: maintain repository backups and facilitate working on a project from multiple machines
  - Although you can always create and maintain your own local git repository on your machine ☺
- Every contributor of a repository gets a copy of the repo to work independently
- Many other similar services, e.g. *Bitbucket* and *Gitlab*

The typical (and very simplified) anatomy of a *github* based project



All users that **clone** a repo get a local copy of the repo

Users can **commit** changes to their local repos without affecting other users

When ready to share with colleagues, **push** to github repo… if you have proper permissions!

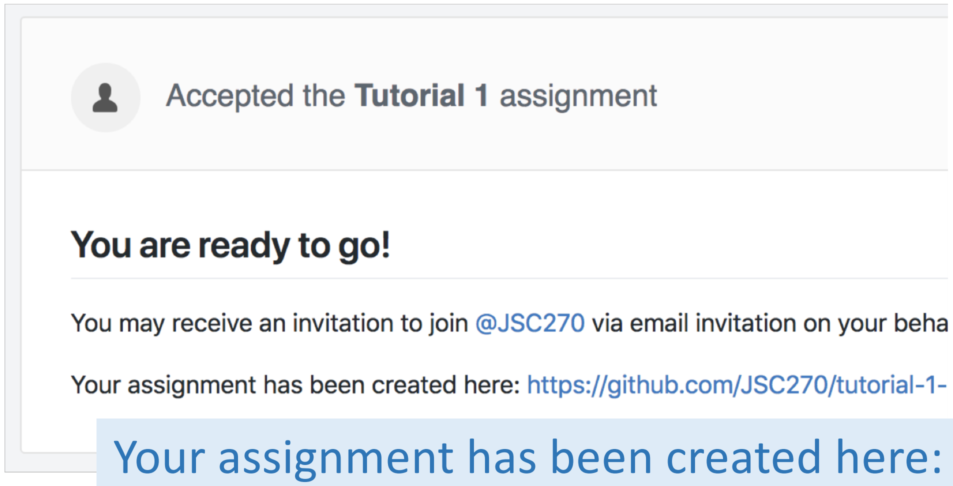
How to use *Github Classroom*

How to *clone* a shared github repo into a local repo @ teach.cs

1. Navigate to the Tutorial 1.0 repository in **Github Classroom** and accept the assignment

classroom.github.com

https://classroom.github.com/a/Wwxq8088



Accepted the **Tutorial 1** assignment

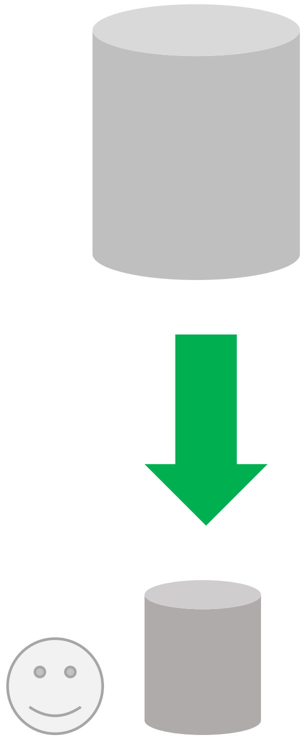
You are ready to go!

You may receive an invitation to join @JSC270 via email invitation on your beha

Your assignment has been created here: https://github.com/JSC270/tutorial-1-

Your assignment has been created here: https://github.com/JSC270/tutorial-1-0<your-git-username>

2. Copy the link!



For Mac/Linux

How to *clone*
a shared github repo
into a local repo
@ teach.cs

3. Open the terminal, and log into your teach.cs account

```
ssh -l <your_username> teach.cs.toronto.edu
```

Access to your remote workspace at teach.cs. Param:

-l ***login*** (for more param options, check **ssh -h**)

4. (***Optional***) Create a folder to store course tutorials/assignments

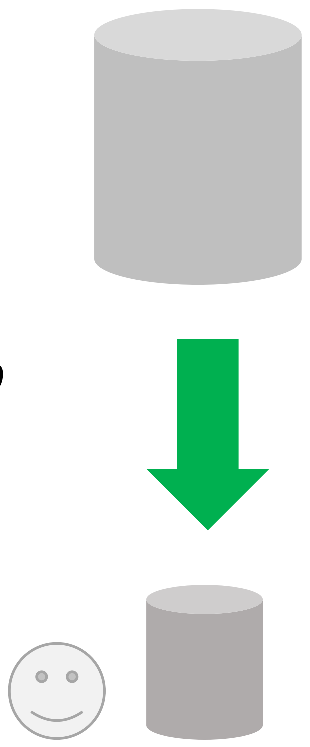
```
mkdir jsc270
```

*Creates a new folder in the current directory called **jsc270***

```
cd jsc270
```

Navigates to the newly created folder

Hint: Prior to creating the folder, make sure you have already navigated to the directory you want that folder in.



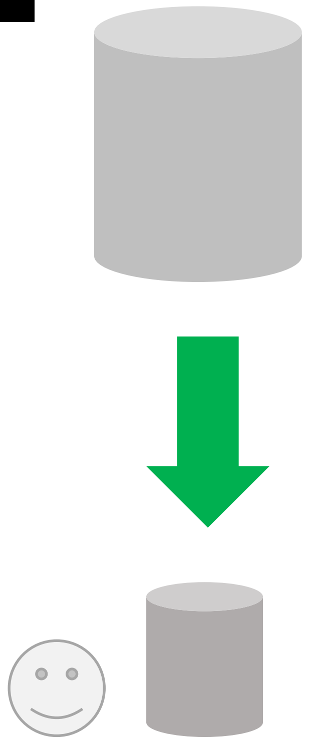
For Mac/Linux

How to *clone* a shared github repo into a local repo @ teach.cs

5. Clone the repository

```
git clone https://github.com/JSC270/tutorial-1-0-<your-username>
```

```
Cloning into 'tutorial-1-0-<your-username>'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
```



6. Navigate to the repo folder and start working!

```
cd tutorial-1-0-<your-username>
```

```
jupyter notebook
```

Creates a jupyter notebook (or restarts, if it already exists) in the current folder

Useful commands:

```
mv <name_before> <name_after>
```

Renames <name_before> file or folder to <name_after>

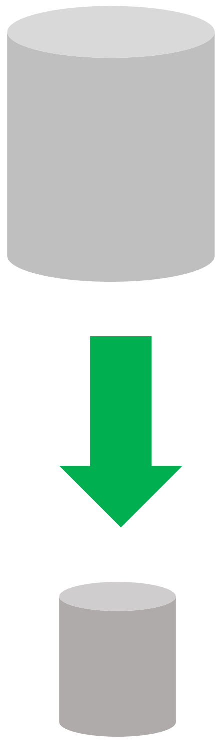
```
cd ..
```

Navigates to the parent folder

```
ls
```

Lists files and folders in the current folder

How to *clone* a shared github repo into a local repo @ teach.cs



How to *modify* The local repo

1. Add files to be tracked

```
git add <my_notebook>.ipynb
```

```
A  "<my_notebook>.ipynb"
```

You can also 'add' more than one file at a time:

```
git add <file1> <file2> <file3> ...
```

2. Commit to save a set of changes on tracked files

```
git commit -a -m "<commit messsage>"
```

-a	**auto stage all modified files**
-m	**commit message**

Commits ➔ save points
If that's a state you might want to revert back to if you mess up in the near future, then commit.

You should always write a little something to describe what changes you made at the time

Files that were not 'added' to the commit (via **git add**) will not be saved. Use –a to "***stage***" all modified (tracked) files, or git add manually if you want to control which files are ***staged*** in that commit.

```
M "<my_notebook>.ipynb"
```

Unstaged

```
M "<my_notebook>.ipynb"
```

Staged

Useful commands:

How to *modify*
The local repo

```
git status
```

Show status of repo, including modified and untracked files

```
git status -s
```

A summarized version of the above

```
git checkout <filename>
```

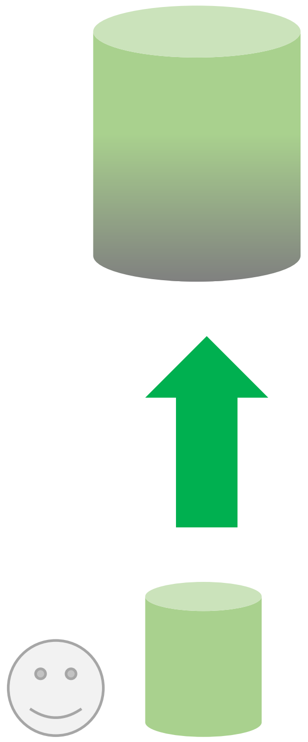
Loads most recent committed version – i.e., discard ***local uncommited changes****.*
Use with caution!

Working with the central repo

Push your local changes to the central repository

```
git push
```

Push your local changes to the repo from where you cloned your local repo, into a default "master" branch



Working with collaborators

Retrieve changes from your collaborators, merge, then commit

```
git pull
```

Pulls your collaborators' changes to your local repo, and tries to merge them with your local changes

If two or more collaborators change the same parts in a file, **conflicts** will show up. You need to fix these before continuing. Git will modify these files to indicate where conflicts appeared.

```
git commit -a -m "<merge msg>"
```

Create a "checkpoint" for your local repo plus your collaborator's changes

```
git push
```

Save the merged version to the central repo