Git Workshop Activity

Activity for the SUNY Oswego Computer Science Association's Git Workshop. Written by Christopher Wells, and released under CC0 license.

Installing Git

The process for installing Git varies based on the operating system you are using.

If you are using Linux or Mac, the process is fairly simple, you should be able to install Git through your package manager. Then you will be able to use Git directly from your terminal.

If you are using Windows, you should download and install the Git Bash program, which is a command line that allows you to use Git. You can get Git Bash from the following webpage:

> https://git-scm.com/downloads

Once you have Git Bash installed you can run it and it will act as a command line that has Git support.

Setting up Git

Once you have Git installed, you will need to set it up with your name and email address, so that it can properly attribute your work to you. In order to do this, you must run the following commands, substituting in your name and email address.

```
git config --global user.name "YOUR NAME"
git config --global user.email "YOUR EMAIL ADDRESS"
```

For example, if my name was Jane Doe and my email was jdoe@email.com then I would run the following commands:

```
git config --global user.name "Jane Doe"
git config --global user.email "jdoe@email.com"
```

Creating a GitHub Repository

The first thing you need to do before you can create a GitHub project repository is that you need to create a GitHub account. If you already have an account on GitHub, then you can skip this next set of steps.

Creating a GitHub Account

Creating a GitHub account is fairly straightforward. Simply go to the GitHub webpage:

> https://github.com/

Right on the main page there is a section where you can create an account by giving your desired account name, email, and password.

Creating a Project Repository

Once you have created your account on GitHub, you can now create project repositories. To create a test repository you can click on the + icon in the top right corner of the page and select “New Repository” from the dropdown menu.

Once you are at the “New Repository” page you can give your repository a name, for the test repository you can just call it “first-repository”. You can also give your repository a description, set whether you want it to be public or private, and choose what files it will have initially.

For now, you can just give the repository a quick description and leave the default settings. Then click on the “Create repository” button.

Cloning a Repository to your Machine

So now that you have created a repository on GitHub, you’ll want to be able to copy that repository to your computer so that you can add stuff to it. You can do this using Git through your terminal (Linux + Mac) or through Git Bash (Windows).

On the page for your repository you should see a section where it gives you a link like the following. This link is what you will use to copy your repository to your machine.

> https://github.com/ExcaliburZero/test-repository.git
>
> Yours will look something like this ^
> But be sure to use the link that shows up for you

To copy your repository to your machine, copy the link that is given on your repository page. Go into terminal or Git Bash and navigate to the directory where you want to put the copy of your repository and then run the following command.

```
git clone "REPOSITORY LINK HERE"
```

For example, if the link to my repository were https://github.com/ExcaliburZero/test-repository.git then I would run the following command.

```
git clone "https://github.com/ExcaliburZero/test-repository.git"
```

Once you run the git clone command, you should now have a copy of your repository on your computer. However, the repository is currently empty so all you will have is an empty folder.

Adding Files to your Repository

To add a file to your repository on GitHub you will first need to add it to your local copy of the repository. So create a file named “README.md” in your repository directory with the following contents.

Since you have cloned the repository to your computer, it is now present as a directory on your file system. You can add files to it by just saving a file into that directory using a program like Notepad or Gedit.

```
# First Repository
This is my first repository
```

Once you have created that file, navigate into the repository directory in your terminal / Git Bash and run the following command.

```
git status
```

You should now be able to see that Git has noticed that you have added the “README.md” file to the directory and note that it is a new file.

Next you will want to let Git know that you want the new file to be tracked in the repository. You can do this by running the following command.

```
git add README.md
```

This command will tell Git that you want it to prepare to save the changes you have made to the file to the repository history. If you run the git status command again, you can see that the file has been “staged”.

Once a file has been staged, you can save the changes you made to it to the repository history by “committing” it. You can commit the changes by running the following command.

```
git commit -m "DESCRIBE CHANGES HERE"
```

For example, if I wanted to commit the added “README.md” file I would run the following.

```
git commit -m "Added readme file"
```

Now if you run the git status command again, you can see that the “README.md” file is no longer listed, as the changes to it have been committed.

If you run the git log command, you can see the information on the commits in the repository including the commit you just made for adding the readme file.

```
git log
```

Pushing Changes to GitHub

Once you have made changes to your local copy of the repository and have committed those changes, you can push those changes up to GitHub.

To push the changes to GitHub you can use the git push command. In the git push command you must specify two things, the remote you want to push to and the branch you want to push.

```
git push REMOTE BRANCH
```

In this case, since we have not changed anything with the remotes or branches, we can just use the defaults. The default remote is “origin” and the default branch is “master”. So we can just run the following command.

```
git push origin master
```

Git will prompt you to enter your username and password for GitHub, and once you have done so successfully it will push the local changes to your GitHub repository.

Now if you go back to the page for your GitHub repository you will see the “README.md” file that you committed and pushed.

Collaborating With GitHub

One of the nice things about GitHub is that it makes it easier for you to collaborate on project with others. To demonstrate this, let's try making a contribution to a project. First, go to the following link.

> https://github.com/ExcaliburZero/test-repository

This is a simple test project that you can try making a contribution to to learn the process.

Forking a Repository

In order to contribute to the repository, the first thing you need to do is "fork" it, which creates a copy of the repository on your GitHub account. You can fork the repository by clicking on the "Fork" button in the top right corner.

Once you have forked the repository, you should now have a copy of it on your own GitHub account. By creating a fork, you can work on the project without needing to get permission from the repository owner to work with their repository directly.

Since you now have a fork of the repository you can clone your forked repository down to your computer. First, make sure that you are on the forked repository and not the original one, then click on the "Clone or download" button and copy the link the is shown.

Then with the link to the fork you can use the git clone command to clone it to your computer.

```
git clone "FORK LINK"
```

For example, if the url for my fork were https://github.com/ExcaliburZero/test-repository.git then I would run.

```
git clone "https://github.com/ExcaliburZero/test-repository.git"
```

Then you should have a local copy of the forked repository.

Making Changes to the Fork

Now that you have a local copy of your fork you can start making changes to it.

Previously, when we were making changes to our own repository we just used the default branch. However, when you are working on a group project it is general practice to use branches.

To create a new branch based off of the current branch you can run the following command, substituting in the name of the new branch to create.

```
git checkout -b BRANCH_NAME
```

For example, if I wanted to name my new branch “add-info” I would run the following command.

```
git checkout -b add-info
```

Once you create your new branch you can start making your changes. You can confirm that you have created and switched into your new branch by running the following command, and you should see the master branch and your new branch listed.

```
git branch
```

For now you should edit the “README.md” file and add your name to “Participants” section.

```
## Participants
* Jane Doe
```

Once you have made the change, add the changes and commit them by running the following commands.

```
git add README.md
git commit -m "Added name to participants"
```

Once you have added and committed your changes, you can push them up to your fork.

In this case we are using the same origin remote representing your fork, however this time we are using a different branch than master.

First, run the git branch command to see what branch you are currently on, then run the git push command using “origin” for your remote and the branch you are currently on.

```
git branch
git push REMOTE BRANCH
```

For example, if your new branch is “add-info”, then you would run the following command.

```
git push origin add-info
```

Now your changes should be pushed up to your fork.

Making a Pull Request

Now that you have made your changes to the fork, you can request them to be pulled into the original project. To do this you need to create a “pull request”, which is a request to the original project owner to have them incorporate your code into the project.

To create a pull request, go to your forked repository on GitHub and click on the “branches” section to see your repository branches. Then find the branch that you made the changes in and click the “New Pull Request” button.

Then you will be taken to the pull request creation page. All of the default settings should be good, you will just need to give the pull request a name and a description.

Once you have given the pull request a name and description, you can click on the “Create pull request” button to submit your pull request.

Now you have made your changes to the project and submitted them for approval. You did it!