

# **GitHub and Repository Management**

**Student Workbook 3c – Command Line and Git**

Version 7.0 Y

# Table of Contents

<b>Module 1 GitHub - Collaborating with Others</b> .....	<b>3</b>
Section 1-1 Migrating a Local Repository to GitHub.....	4
Migrating a Local Repository .....	5
Creating an EMPTY Remote Repository .....	6
Configuring the Remote Repository Locally: <code>git remote</code> .....	8
Pushing to the Remote Repository: <code>git push</code> .....	10
Pulling from the Remote Repository: <code>git pull</code> .....	11
Working with Other Branches and a Remote Repo .....	12
Exercises – Existing Project Push.....	13
Section 1-2 README.md - How to Create Good Markup Files.....	15
README.md.....	16
Markdown Language.....	17
Basic Markdown Rules .....	18
Example.....	20
A Good README.....	21
Exercises – README.md .....	23

# **Module 1**

## **GitHub - Collaborating with Others**

## Section 1–1

# Migrating a Local Repository to GitHub

# Migrating a Local Repository

---

- **Often you will have a project that was started before the GitHub repository was created**
  - `git clone` will not allow you to clone a new repository into an existing folder
- **You can create a new EMPTY repository on GitHub, and then configure your project to point to the empty repo**
  - There cannot be any files in the repo when you create it
    - \* Not even a `README.md` or a `.gitignore`

# Creating an EMPTY Remote Repository

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

The screenshot shows the GitHub 'Create a new repository' form. The 'Owner' is 'gdzierzon' and the 'Repository name' is 'empty-project', which is marked as available. The 'Description' field is empty. The 'Public' option is selected. Under 'Initialize this repository with:', the 'Add a README file' checkbox is unchecked. The '.gitignore' template is set to 'None'. The license is set to 'None'. A large red circle with the text 'Don't add ANY files to this' has three lines pointing to the 'Add a README file' checkbox, the '.gitignore template: None' dropdown, and the 'License: None' dropdown. At the bottom, there is a blue 'Create repository' button and a note: 'You are creating a public repository in your personal account.'

Owner \* / Repository name \*

gdzierzon / empty-project ✓

Great repository names are empty-project is available. Need inspiration? How about shiny-octo-carnival?

Description (optional)

☒ Public  
Anyone on the internet can see this repository. You choose who can commit.

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

Initialize this repository with:  
Skip this step if you're importing an existing repository.

☐ Add a README file  
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore  
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None

Choose a license  
A license tells others what they can and can't do with your code. [Learn more.](#)

License: None



① You are creating a public repository in your personal account.

Create repository

- After clicking the **New** button, you can:
    - Name the repo
    - Set the visibility of the repo (public / private)
    - DO NOT add a README file or gitignore file
    - DO NOT specify licensing
    - Accept the default main branch name as 'main' or change it to something else
- \* Until recently, it defaulted to 'master'

- There may be small differences on your internal GitHub
- After creating an empty repo - GitHub displays instructions on how to point to the new repo

Quick setup — if you've done this kind of thing before


 Set up in Desktop or ☐ HTTPS ☐ SSH  

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

---

...or create a new repository on the command line


```
echo "# empty-project" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/gdzierzon/empty-project.git
git push -u origin main
```



---

...or push an existing repository from the command line

```
git remote add origin https://github.com/gdzierzon/empty-project.git
git branch -M main
git push -u origin main
```



---

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

# Configuring the Remote Repository Locally:

## git remote

- If your local repo was not cloned from a GitHub repo it will not know where to push or pull
  - But you can configure your repo to point to any remote git repository
- You can use the `git remote -v` command to see if it is pointing to a remote repository

### Finding out about the remote repo

```
$ git remote -v
```

If your repo was cloned from a remote it should look similar to this

```
MINGW64:/c/LearnToCode/Workbook1/Hello-World
grego@Shadow MINGW64 /c/LearnToCode/workbook1/Hello-world (main)
$ git remote -v
origin https://github.com/gdzierzon/Hello-world.git (fetch)
origin https://github.com/gdzierzon/Hello-world.git (push)
```

If you created the repo using `git init` instead of cloning it, your screen may look similar to this

```
MINGW64:/c/LearnToCode/Workbook1/Hello-World
grego@Shadow MINGW64 /c/LearnToCode/workbook1/Hello-world (main)
$ git remote -v

grego@Shadow MINGW64 /c/LearnToCode/workbook1/Hello-world (main)
```

- If you do not see an origin listed as a remote, your repo cannot be pushed to a remote server
  - You can still configure your repo to add an **origin** link



- If you created the repo on your local computer first, you should use the `git remote add` command to create and configure the remote repo

- You may be prompted for GitHub credentials

#### Creating a remote repo using Git

```
$ git remote add origin https://github.com/gdzierzon/empty-project.git
```

- Notice here we are specifically giving the remote the name 'origin'

# Pushing to the Remote Repository:

`git push`

---

- The `git push` command says "push the commits in the local branch to the remote branch"
  - Once executed, commits since your last push will be available on GitHub (the remote repo)

## Initial push

```
$ git push -u origin main
```

- Use the `-u` flag the FIRST time you push any new branch to create an *upstream* tracking connection
  - `origin main` is the name of the remote and the name of the new branch
- **Development is a constant cycle of:**
  - making local changes
  - committing to the local repo
  - pushing changes to the remote repo
  - pulling changes from the remote repo
- **So, why do you pull changes?**
  - Because others might be working in the same project and have made pushes that you want to refresh in your own local repo

# Pulling from the Remote Repository:

## `git pull`

---

- The `git pull` command that says "pull the commits in the remote branch to the local branch"
  - When executed, any commits made to the remote branch since your last pull become available in your local copy of the repository

### Example pull

```
$ git checkout main  
$ git pull origin main
```

- It's a good practice to make sure you are in the right branch before you execute the pull command

# Working with Other Branches and a Remote Repo

---

- **When you create a new branch locally and decide to push it to the remote repo, you will need to use the -u flag on the initial push**
  - Afterwards, the push command is simple

- **A log of a programmer's work might be:**

```
$ git checkout main > checkout the latest main
$ git pull origin main

$ git checkout -b MyNewFeature1 > create a branch off main
> make changes <

$ git status
$ git add .
$ git commit -m "Some changes occurred"
$ git status

> check everything <

$ git push -u origin MyNewFeature1 > create a remote tracking
                                   > branch
> make more changes <

$ git status
$ git add .
$ git commit -m "More changes occurred"
$ git status

> check everything <

$ git push origin MyNewFeature1 > push additional changes
```

- **Alternatively, you may not want to push the branch to the remote**
  - You might work in the branch locally and then merge to main
  - Then, push main to the remote

# Exercises – Existing Project Push

---

In this exercise you will create a new remote GitHub repository for your LearnToCode workbook exercises. You will then see how to convert an existing project folder as a git repository and push your work to GitHub.

## EXERCISE 1

**Step 1:** Log into GitHub and create a new empty repository.

Go to your GitHub profile, click the Repositories tab, then click the **New** button

Name the repo `command-line` and let everything else can stay at the default values (so **do not add readme!**), then click "Create Repository".

**Step 2:** Configure the remote repository

While still in GitHub, copy the HTTPS URL of your new repository. It will resemble:

**`https://github.com/your-github-username/command-line.git`**

Navigate to your `C:/pluralsight/command-line` folder in GitBash.

Initialize this directory as a git repo.

```
$ git init
```

Verify that you do not currently have any remote repository linked (there should be no origins)

```
$ git remote -v
```

Add a new remote to your `command-line` repo and push all of your existing code to the remote

```
$ git remote add origin https://github.com/<your-github-username>/command-line.git  
$ git remote -v  
$ git status  
$ git push -u origin main
```

## Section 1–2

### README .md - How to Create Good Markup Files

# README .md

---

- **A README .md file is a text file that describes the project contained in a repo**
  - It resides at the root of the project
  - The Git cloud services GitHub, GitLab, and Bitbucket look for your README and display its information along with the list of files and directories in your project
- **The README contains information such as:**
  - what the project is used for
  - the technologies used in the project
  - how to collaborate with you (if appropriate)
  - license information (if any)
- **It helps the audience understand how to install and use your project**
- **Many people get into the habit of making it the first file you create in a new project**



# Markdown Language

---

- **Most READMEs are often written using the Markdown language**
  - The wiki for markdown is here:  
<https://en.wikipedia.org/wiki/Markdown>
- **It lets you to add some lightweight formatting without using a sophisticated editor**
  - Remember, the README file is a plain text file
- **A quick Google search will lead you to lots of good examples of READMEs and markdown syntax, however the following pages will demonstrate some examples**

# Basic Markdown Rules

---

- The examples below show different ways to use markdown in a README file

There are two ways to create a BIG heading:

**# This is a Big Heading**

This is also a Big Heading  
=====

**## This is a Sub-heading**

This is also a Sub-heading  
-----

Paragraphs are just a bunch of sentences separated by a blank line. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam.

See, this is another paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam.

Sometimes, you want to force a line break. In this case, put two spaces at the end of a line and do you see the line break?

You can add a horizontal rule like this:

---

You can also set the font on text. For example,  
*\_these words are italicized\_*,  
on some systems *\*these worlds might be italicized\**,  
**\*\*these words are bolded\*\***,  
and finally, ``the words use a monospace font``.

You can have bulleted lists:

- \* HTML
- \* CSS
- \* JavaScript

You also can have numbered lists:

1. HTML

2. CSS
3. JavaScript

You can add links. For more information, click [[here](https://en.wikipedia.org/wiki/Markdown)](<https://en.wikipedia.org/wiki/Markdown>)

If you want to add formatted code to markdown, use backticks before and after it like this:

```
...
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <!-- body markup goes here -->
  </body>
</html>
...
```

And to add an image, try this:

![[Image](images/readme-images/dana.jpg "icon")]

- **To see the markdown above rendered, look at the next page**

# Example

---

## 🔗 This is a Big Heading

---

## 🔗 This is also a Big Heading

---

### 🔗 This is a Sub-heading

---

### 🔗 This is also a Sub-heading

---

Paragraphs are just a bunch of sentences separated by a blank line. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam.

See, this is another paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam.

Sometimes, you want to force a line break. In this case, put two spaces at the end of a line and do you see the line break?

You can add a horizontal rule like this:

---

You can also set the font on text. For example, *these words are italicized*, on some systems *these words might be italicized*, **these words are bolded**, and finally, `the words use a monospace font`.

You can have bulleted lists:

- HTML
- CSS
- JavaScript

You also can have numbered lists:

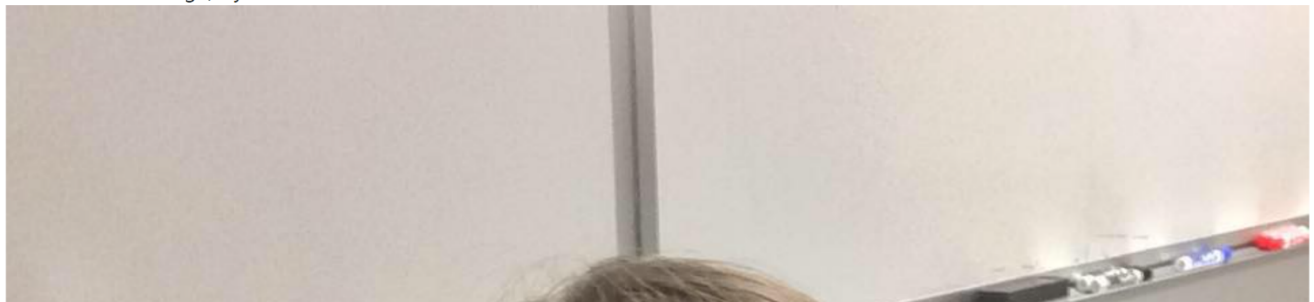
1. HTML
2. CSS
3. JavaScript

You can add links. For more information, click [here](#)

If you want to add formatted code to markdown, use backticks before and after it like this:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <!-- body markup goes here -->
  </body>
</html>
```

And to add an image, try this:



# A Good README

---

- **Good READMEs are like good books -- everyone has different opinions**
  - <https://www.makeareadme.com/>
- **Minimally, you should include:**
  - Name
  - Description
    - \* Consider adding a list of features section
- **You may also want to add screenshots or even a video of your user interface**
- **If your project requires non-obvious installation steps, you will want to include them**
  - You may even include a list of system or software requirements
- **You may include code examples of how to use or extend the software if it is a framework**
- **If you provide support, tell people where they can go to for help**
  - It might be an email address, a web site, a chat room, or even a place to report issues
- **If you are open to contributions, describe your requirements are for accepting them**

- **Perhaps even include an authors and acknowledgment section!**
- **Finally, for open source projects, describe how it is licensed**
  - Types of licenses include MIT, Apache, and BSD
  - <https://snyk.io/blog/mit-apache-bsd-fairest-of-them-all/>

# Exercises – README.md

---

## EXERCISE 1

In this exercise you will add a `README.md` file to your `command-line` repository.

**Step 1:** Open your `command-line` folder in IntelliJ.

Open the `README.md` file

**Step 2:** Modify your `README` file to create a basic resume that resembles the following:

### **YOUR NAME**

Contact details go here.

A summary of you and your skills goes here

### **Highlights**

- Bulleted list of skills

### **Work Experience**

#### **Job Title**

#### **Company**

Date Range

- Bulleted list of responsibilities

#### **Job Title**

#### **Company**

Date Range

- Bulleted list of responsibilities

**Step 3:** Stage, commit and push your changes. Then navigate to your repo on [GitHub.com](https://github.com) and verify that your resume appears on the homepage of the repo.





