# Getting Started with GitHub on Windows 10 Using Git Bash

Andy Malinsky

14 February 2019

# Contents

# 1 Introduction

## 1.1 What is Git and Why Use It?

Software development projects, especially in industry, are usually maintained by a team of developers. Therefore, it is crucial that each team member is able to contribute to the source code without experiencing overriding conflicts. This is where Git comes in. **Git** is a distributed version control system. This means that Git can track the changes made to a code base and maintain a history of previous versions of a project. This allows for a more efficient collaborative experience between developers working on the same project in parallel. Git distributes copies of the code between a local machine and a remote server, from file storage locations known as repositories. Local machines are the individual computers of each developer, while remote servers could include cloud based platforms such as GitHub. Developers make changes to the code on their local machine, and then share it with collaborators by pushing their changes to the remote server.

## 1.2 What is GitHub?

**GitHub** is a cloud-based web platform that optimizes version control and collaboration using Git. It provides a user interface for interacting with Git projects, and acts as a remote server for source code. In addition, many developers use GitHub as a software project portfolio, which they can share with potential employers and collaborators.

## 1.3 What is Git Bash?

Git commands are ultimately executed at the command line, and were originally designed for complementary Unix environments, such as Linux or macOS. Windows has a built-in command prompt, but it is not a Unix terminal environment. Although the default Windows command prompt is capable of running Git commands, the use of common Unix commands are not recognized, such as 'ls' (discussed later). **Git Bash** addresses this as it provides an emulation of the default Unix shell, a command line user interface, known as Bash.

## 1.4 Requirements

The purpose of this user manual is to demonstrate the process of using Git Bash and GitHub for an individual user experience. Topics and Git commands pertaining to collaboration between multiple users are not covered. It is assumed that you have Windows 10 installed, and that you are familiar with navigating the Windows File Explorer, launching Desktop applications, and running downloaded files from the Internet. This guide is for Git version 2.20.1 or greater.

# 2 Setting up the Environment

## 2.1 Download and Install Git

In order to run Git commands using Git Bash you need to install Git first.

1. Open an Internet browser and navigate to the following link: https://git-scm.com/downloads
2. Click on *Windows* to download the latest source release.
3. Run the downloaded file.
4. Click *Next* for every default selection.
5. Click *Install*.

## 2.2 Create a GitHub Account

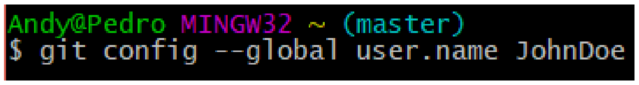
Now you are ready to create a GitHub account.

1. Open an Internet browser and navigate to the following link: https://github.com/join?source=header-home
2. Follow the steps to create your free personal account.
3. Once you have created an account, sign in to GitHub.

### 2.3 Configure Your Git Identity

Now that you have Git installed and a GitHub account created, the next step is to set your username and email address for Git. Note, you must press Enter to run each command in the terminal. You may refer to the Figure captions before typing a command.

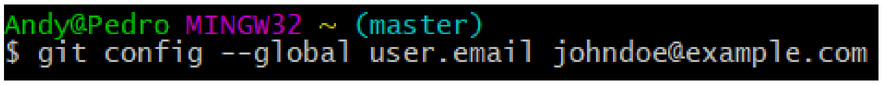
1. Launch Git Bash from the Windows Start menu.
2. Type in the following command to set your username:



```
Andy@Pedro MINGW32 ~ (master)
$ git config --global user.name JohnDoe
```

Figure 1: The value after *user.name* should match your GitHub username.

3. Type in the following command to set your email:



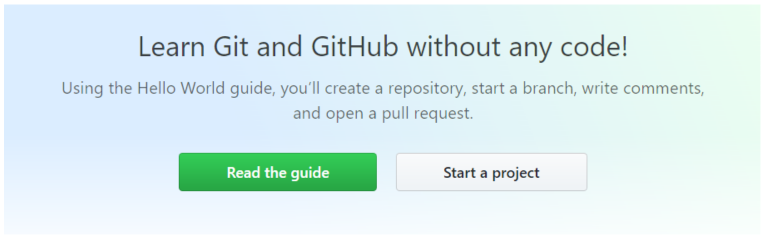
```
Andy@Pedro MINGW32 ~ (master)
$ git config --global user.email johndoe@example.com
```

Figure 2: The value after *user.email* should match the email associated with your GitHub account.

# 3 Starting a GitHub Project

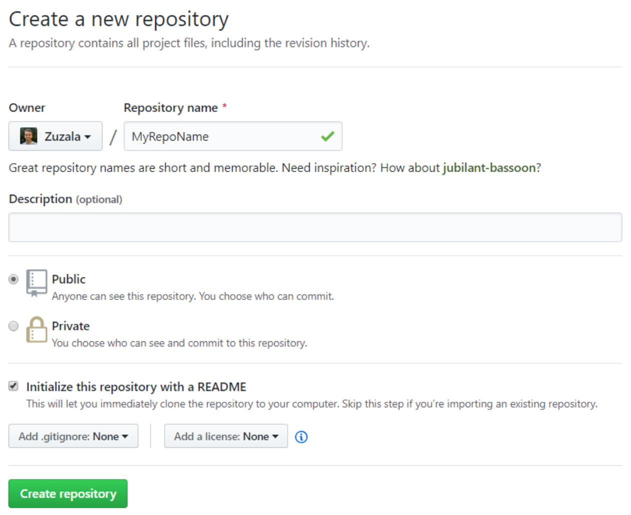
### 3.1 Create a New Repository

Once you are logged in to GitHub you should be able to click the *Start a project* button.
Next, you will be able to create a new repository. You can always add or edit a Description later. Make sure to check the box to *Initialize this repository with a README* to simplify the process. A README file is simply a form of documentation markup that provides information about the project. Once you have filled out the fields click *Create repository*.


Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

Figure 3: GitHub welcome screen. Find *Start a project* button.


Create a new repository

A repository contains all project files, including the revision history.

Owner: Zuzala / Repository name *: MyRepoName

Great repository names are short and memorable. Need inspiration? How about **jubilant-bassoon**?

Description (optional)

[✓ 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 let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

Figure 4: Create a new repository form. First time users may need to verify their email address.

## 3.2 Clone Your Repository

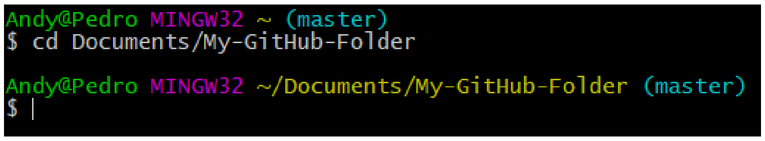
Now that you have created a remote location for your repository, you want to have a copy of it on your local machine to make edits on.

1. Using File Explorer, create a new folder where you will keep your GitHub projects.

2. Next, you want to launch Git Bash and set the current working directory to your GitHub project folder. Here are two ways to do this:

### 3.2.1 Navigating the Command Line

In Git Bash, you want to use the **cd** command to change directories. If you created your GitHub folder in Documents, the command would be the following:



```
Andy@Pedro MINGW32 ~ (master)
$ cd Documents/My-GitHub-Folder

Andy@Pedro MINGW32 ~/Documents/My-GitHub-Folder (master)
$ |
```

Figure 5: Notice the current directory changed.

To navigate up, or back, one directory use the **cd ..** command. Keep this window open as you continue.

### 3.2.2 Launch Git Bash from Current Directory Shortcut

Another way of changing the working directory is through File Explorer. Open your desired location, right-click, and select *Git Bash Here*.

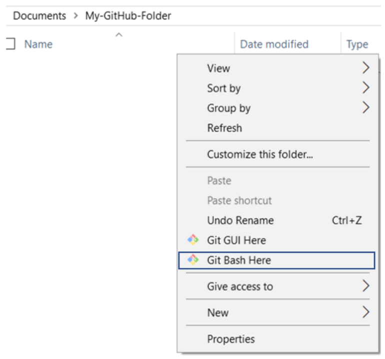


Figure 6: This will launch Git Bash with the current directory set to this folder path.

Once you have Git Bash set to your GitHub folder, the next step is to clone your remote repository.

1. Within your repository on GitHub, click the *Clone or download button.* Then, copy your project's link, as seen in the figure below.

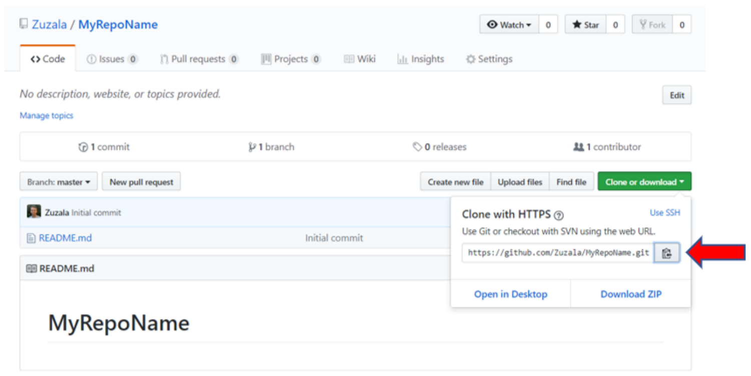
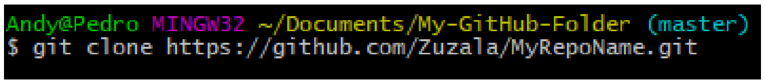


Figure 7: You can click the clipboard button on the right to copy the link.

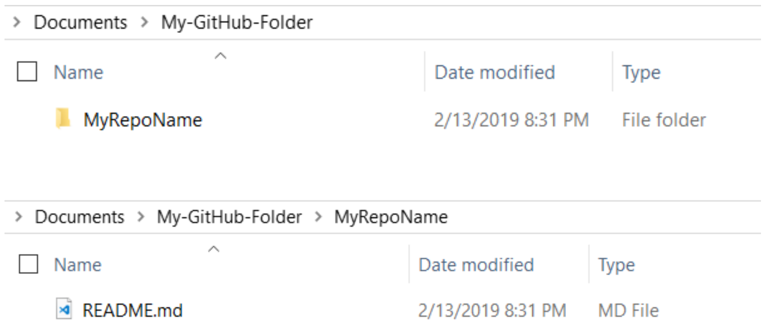
2. In Git Bash, type the following command:



```
Andy@Pedro MINGW32 ~/Documents/My-GitHub-Folder (master)
$ git clone https://github.com/Zuzala/MyRepoName.git
```

Figure 8: You can right-click and select Paste or press Shift+Ins.

3. Check your folder in File Explorer.



> Documents > My-GitHub-Folder

| Name | Date modified | Type |
|---|---|---|
| MyRepoName | 2/13/2019 8:31 PM | File folder |

> Documents > My-GitHub-Folder > MyRepoName

| Name | Date modified | Type |
|---|---|---|
| README.md | 2/13/2019 8:31 PM | MD File |

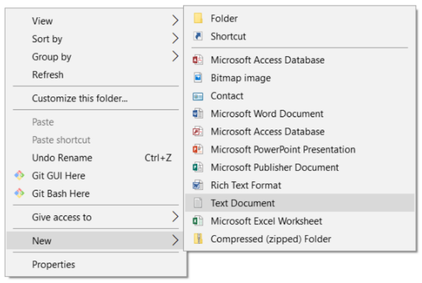
Figure 9: You will now find a local copy of the repository in your folder.

Now you want to set the current working directory to the cloned folder either by using the **cd** command in Git Bash or by using the right-click shortcut.

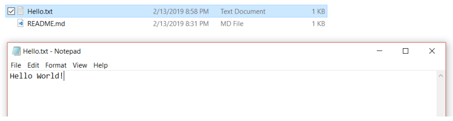
## 3.3 Tracking and Uploading Changes

Now you are free to make any changes to your project from your local repository. GitHub supports a wide range of file types, so you can add or create different types of files and folders. The final part of this manual will demonstrate how to upload changes from your local machine to your remote GitHub repository. The three main steps in this process that will be covered are the Add, Commit, and Push commands.

1. You can start by creating a new file in your cloned folder. You may use the following figure as an example of creating a simple text file.



(a) Right-click and select Text Document.



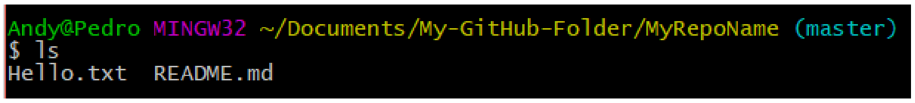
```
Hello.txt    2/13/2019 8:58 PM   Text Document   1 KB
README.md    2/13/2019 8:31 PM   MD File         1 KB
```

```
Hello World!
```

(b) Now you should have at least two files in your folder.

Figure 10: In the background, Git is tracking your changes.

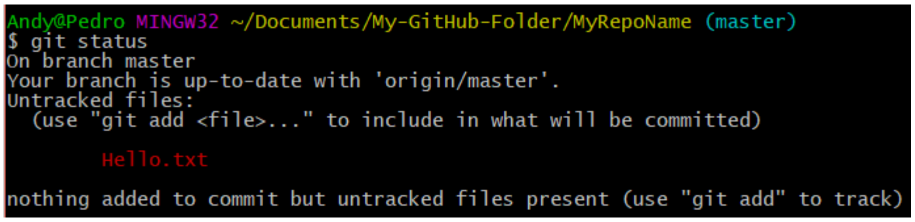
2. You can check what files are present in your folder by typing **ls** in Git Bash.



```
Andy@Pedro MINGW32 ~/Documents/My-GitHub-Folder/MyRepoName (master)
$ ls
Hello.txt  README.md
```

Figure 11: You will notice all files in the current folder are listed.

3. You can also check what files were tracked by Git by typing the command **git status**.



```
Andy@Pedro MINGW32 ~/Documents/My-GitHub-Folder/MyRepoName (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Hello.txt

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

Figure 12: You will notice all edited files are listed in red. This is because they are currently "untracked".

4. **Add:** Git uses a concept of a staging environment to prepare tracked files to be uploaded. To add files to the staging environment type the following command:

   **git add .**

   To track an individual file, for example, you can type **git add Hello.txt**. You may run **git status** again and notice newly tracked files listed in green.

5. **Commit:** The next step is to "commit" your changes. Git adds the staged files to a commit, a record of changes that is placed among other previous commits, adhering to a version control system. To create a commit type the following command:

   **git commit -m "Your message about the commit"**

   You should add a short but meaningful message that describes the changes you made. One example could be "Added text file".

6. **Push:** The last step is to push your commit by simply typing the following command:

   **git push**

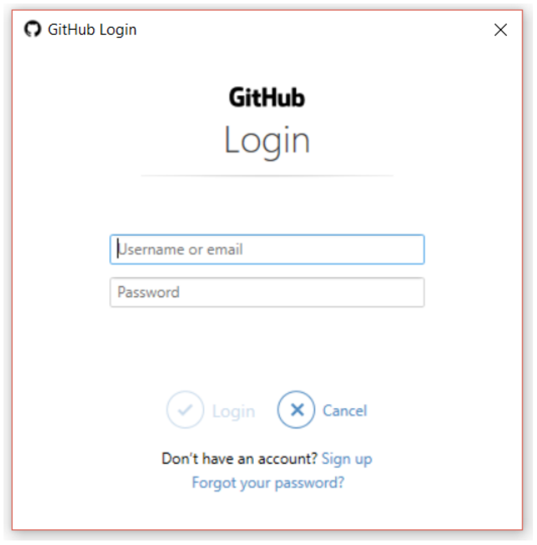


Figure 13: You may be asked to enter your GitHub login credentials for security reasons.

7. Once you have pushed, refresh your browser to view your changes on GitHub.

# Index