We are going to practice working with Git. You will need to coordinate with your team.

Part 1: Pulling the repository to your hard drive

Have one member of your team take care of this step. After they initialize the repository, the rest of the teammates can clone the repository as normal.

This applies only to BitBucket. If you're using GitHub, skip ahead to Part 2.

On your team's BitBucket page, you should see a link saying “I'm starting from scratch”. Click this link, and then it will give you a list of commands to execute (though we only need the second two.)

```
git init
git remote add origin https://rjmfff@bitbucket.org/rjmfff/another-sample.git
```

Create a folder somewhere on your hard drive for the project. Navigate to that folder using Git Bash. You can either use the right-click context menu (in Windows Explorer),

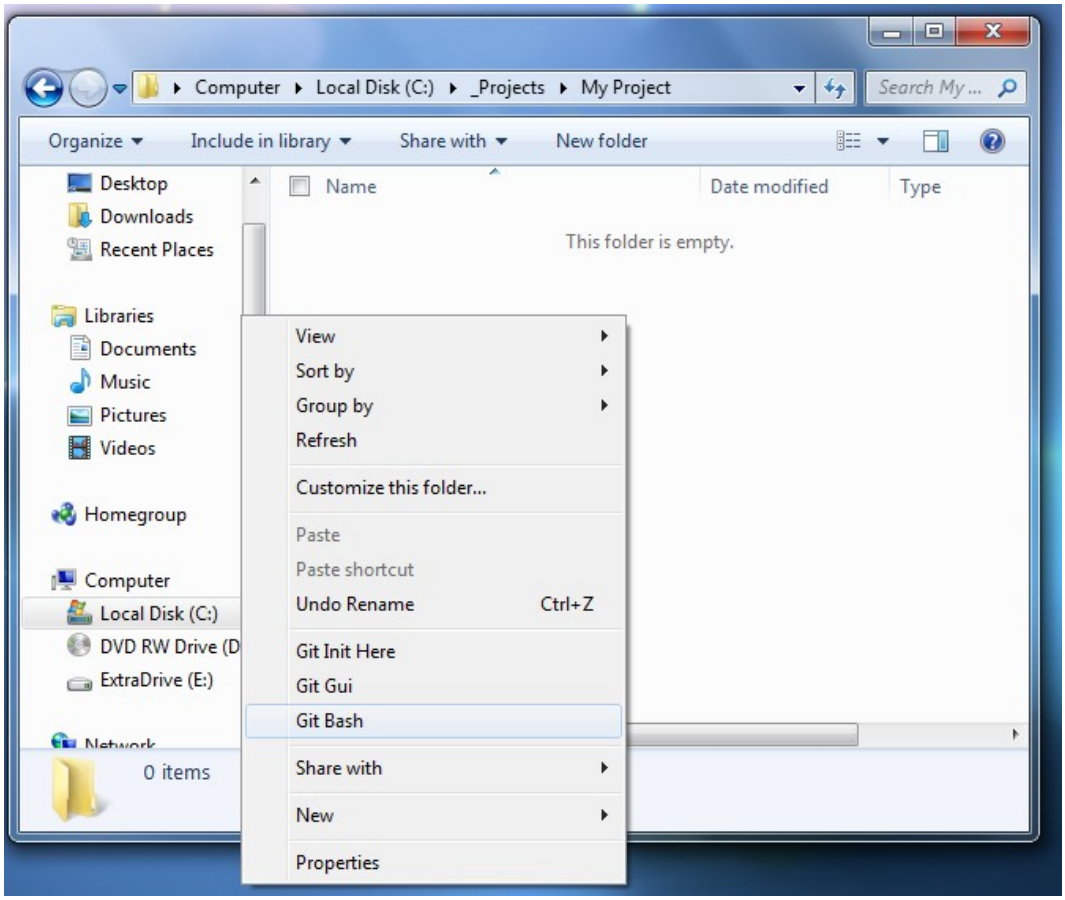


Figure 1. Right-click menu in Windows Explorer

or navigate with the **cd** command (in Git Bash):

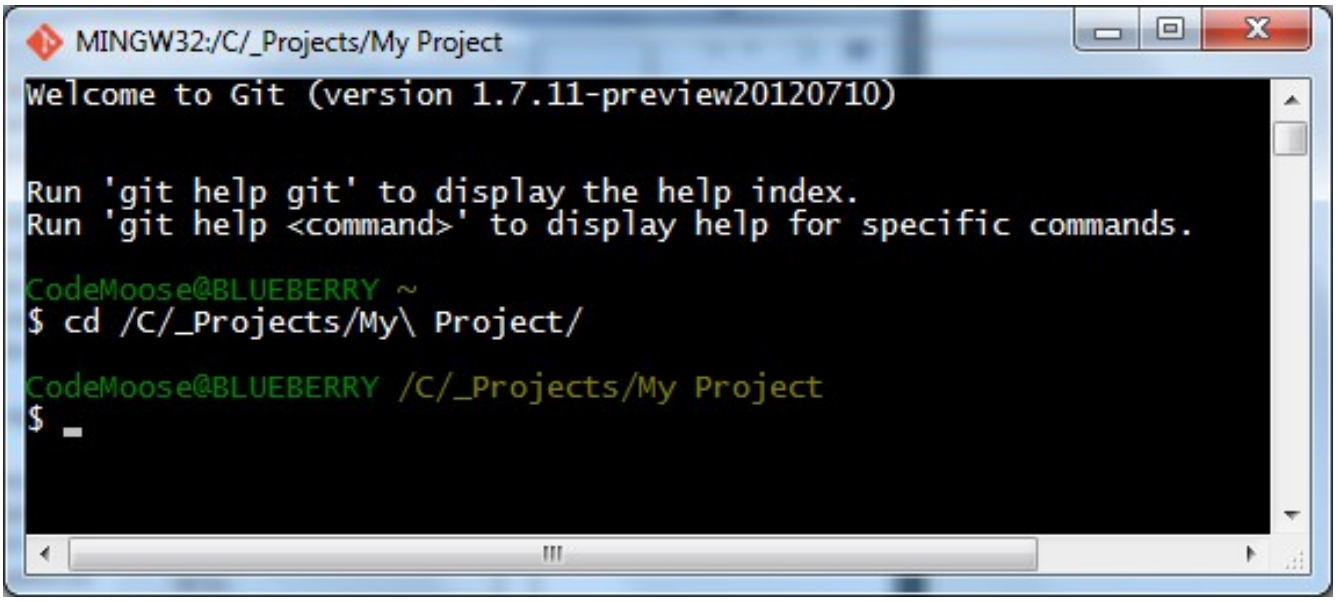


Figure 2. Navigating the hard drive within Git Bash

Note that you need to use /C/ instead of C: from within Git Bash. If your folder has a space, use a \ slash immediately before the space. You can also use the tab key to autocomplete path names.

If you're on Linux or Mac, you should be able to use Git directly from the terminal.

Once you've navigated here, use the git init command, and then the git remote command with the repo URL given on the BitBucket page.

```
git init
git remote add origin https://rjmfff@bitbucket.org/rjmfff/another-sample.git
```

After this is done, create a new file inside this folder. This can just be a text file for now.

Open the text file, enter some text, and then save it.

Back in the Git Bash window, type

```
git add [filename]
git commit -m "First Commit"
git push -u origin master
```

(Each line is a separate command. Hit enter after each command.)

Part 2: Cloning the Repository

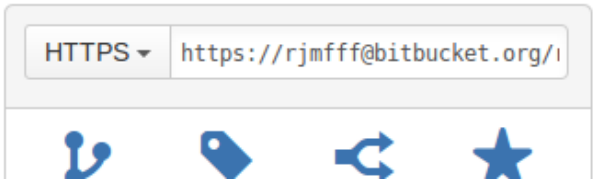
If your team is using BitBucket, one of your teammates should have gone through Part 1 to initialize the repository. Once that is done, the rest of the teammates need to follow this part.

If your team is using GitHub, start from this step.

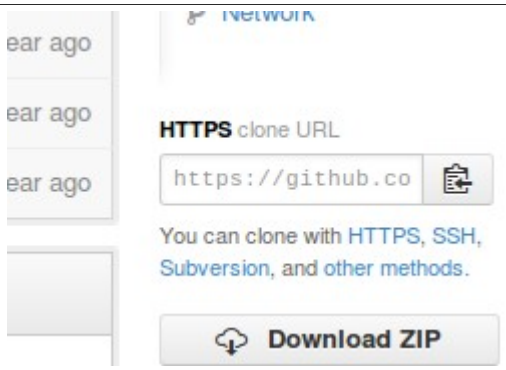
Create a folder on your hard drive where you want to store your project files.

Open the directory in Git Bash, either with the right-click menu (see fig. 1) or via the **cd** command in Git Bash (see fig. 2).

You will also need to go to your repository's webpage and look for the repository URL.



Repository URL location in BitBucket



Repository URL location in GitHub

Copy the repository URL. Then, in Git Bash, enter:

```
git clone [repository URL]
```

The file that your teammate created (BitBucket), or README.md (GitHub) will now be within that folder.

Part 3: Adding a File

Each teammate should do this separately.

Within your project folder, create a text file and name it after yourself. You can add text to it if you would like.

In Git Bash, enter:

```
git status
```

You will see a list of any files that are new or have had changes made to them. Then enter:

```
git add [your filename]
```

Enter

```
git status
```

again and you will see a list of files that will be committed, as well as any files that have changed but have not been **added** to the commit.

Then, commit and push your file:

```
git commit -m "Added my file"
  git push -u origin master
```

Your file and your commit history should now show up in the repository.

Part 4: Editing a File

Each teammate should do this separately.

After everybody in your team has completed part 3, use the following command in Git Bash:

```
git pull
```

This will pull everybody's files down to your hard drive.

In each team member's file (including your own), open it up and append something to the end of the file, save, and close it.

Have each team member go one at a time:

Add all the files using

```
git add [filename1] [filename2]
```

or

```
git add *.txt
```

You can use wildcards in your add commands.

Then commit:

```
git commit –m “[my name] modifications”
```

And push:

```
git push –u origin master
```

Anyone who goes after the first person will have their git push command fail, and it will ask you to use git pull to get the latest files.

After attempting the git push, use:

```
git pull
```

And Git will try to auto-merge your files. If the merge succeeds, it commit those files as a merge and you can proceed to push again.

If the auto-merge fails, you need to open up the file(s) that caused the merge conflict and manually choose what goes where.

Once you're done, commit (if needed) and push again.

Part 5: Removing a File

Finally, we can remove files using the git rm command:

```
git rm [my file name]
```

This will remove a file from your hard drive, and from the repository next time you push your changes.

If you rename a file, git will see the original name as a “deleted file” and the new name as a “new file” when you enter `git status`. However, if you **rm** the old filename and **add** the new filename, in the commit it will show up as a rename.