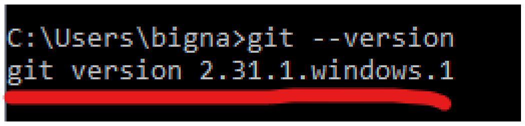
Git Introduction

Goal

Introduce the software tool Git and familiarize the reader with its core functionality.

Prerequisite

Id	Item	Detail

1	Install Git on your computer	You can check if Git is already installed by typing into a Command Prompt or Terminal Shell
		`git --version`
		
		If installed, you will receive verification of the version:
		
		`C:\Users\bigna>git --version`
		`git version 2.31.1.windows.1`
		
		If you are using an older version, you can update by following these instructions
		
		Otherwise, if git is not a recognized command, you need to Install Git

Introduction

Git is an essential tool for developing software applications. It is used to manage source code repositories, track changes, and collaborate within a team.

When developing software features, it is often necessary to change multiple files within a project directory. For instance, one may create a new class as a separate file and also update the **‘main’** method to instantiate the new class. Git’s core function is tracking changes to entire directories, allowing the developer to capture specific directory states, which are delineated by **commits.** When a developer is satisfied with code changes, they can **commit** those changes along with a **commit message** describing the updates to the git repository, thereby locking in the current version (or state) of each of those files.

Git also allows for multiple code variants to coexist within a code base with the concept of **branches**. Branches are similar to tree branches, which start at a common base and then split and extend independently. The primary ‘released’ code typically resides in a **main** branch, which is analogous to a tree trunk. New software features are developed on branches from the master in a secluded environment that won’t disrupt the master. If feature development is successful, it can be **merged** into the master branch and included in the ‘released’ version.

Exercises

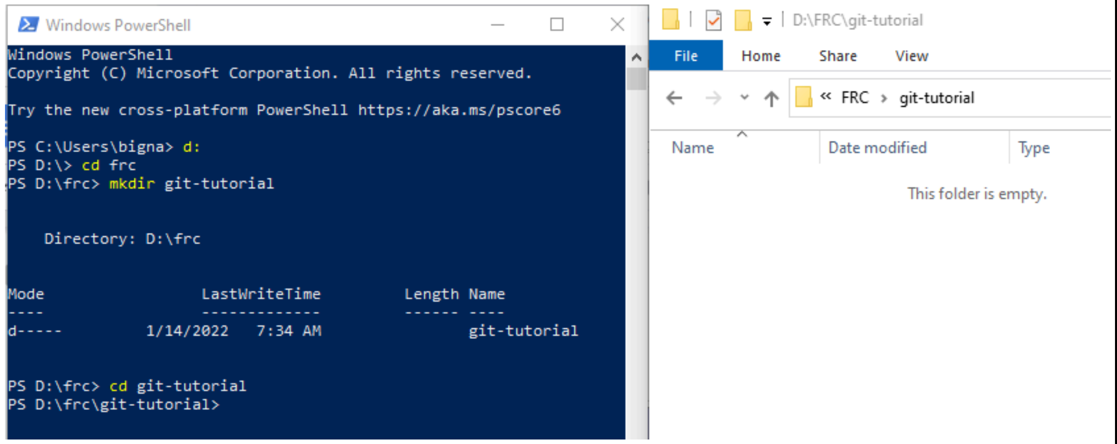
Git Essentials

This first set of exercises introduces the user to core concepts of Git.

Note that Git is different than Github.
Git is a utility for managing source code.
GitHub is a cloud storage location for code repositories.
GitHub is not required for this portion.

In this exercise, a local repository will be created and maintained through a series of commits.
Core concepts include:

- Creating a repo
- Adding files
- Issue a commit with message
- **Add** files to **tracked** list and issuing further commits
- Navigating commits via **checkout**
- Creating feature branches
- Navigating between branches
- Merging branches

Step	Detail
1	Open Powershell / Terminal Verify Git is installed and recognized on your machine (see prerequisites) Create a directory for your repository Navigate to that directory in both shell and file explorer / finder Note: For shell, use the ‘change directory’ command cd as shown below Notice how the shell prompt states the current working directory ‘D:\frc\git-tutorial’ 

2	In Powershell, create a new Git repository and list all files by typing:
	`git init`
	`ls -h`

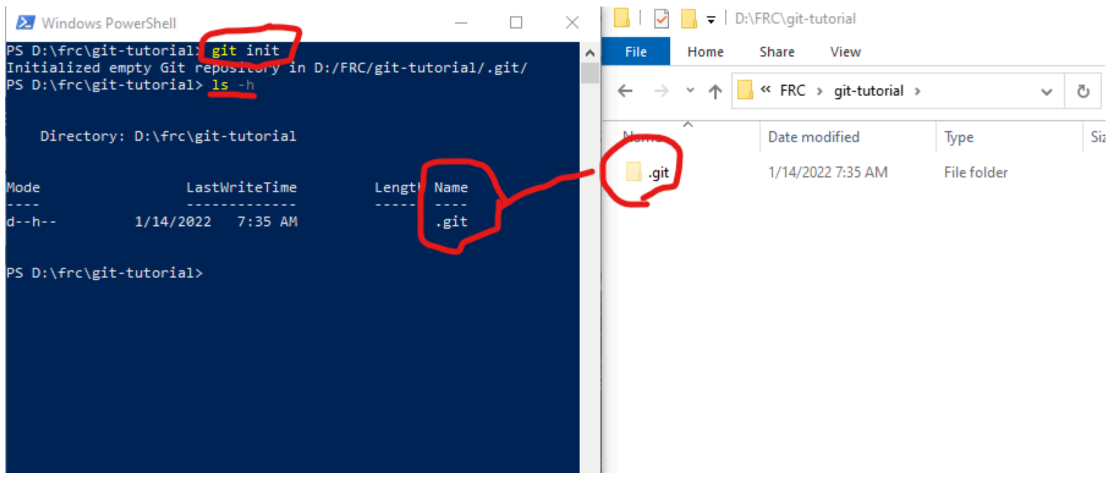
Screenshot in step 1:

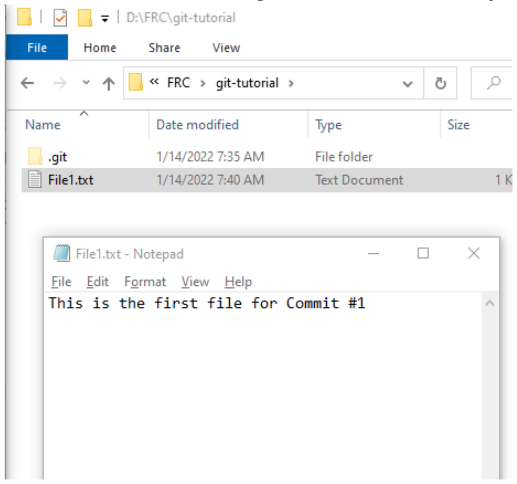
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\bigna> d:
PS D:\> cd frc
PS D:\frc> mkdir git-tutorial


    Directory: D:\frc


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         1/14/2022   7:34 AM                git-tutorial


PS D:\frc> cd git-tutorial
PS D:\frc\git-tutorial>
```

Step 2 commands:

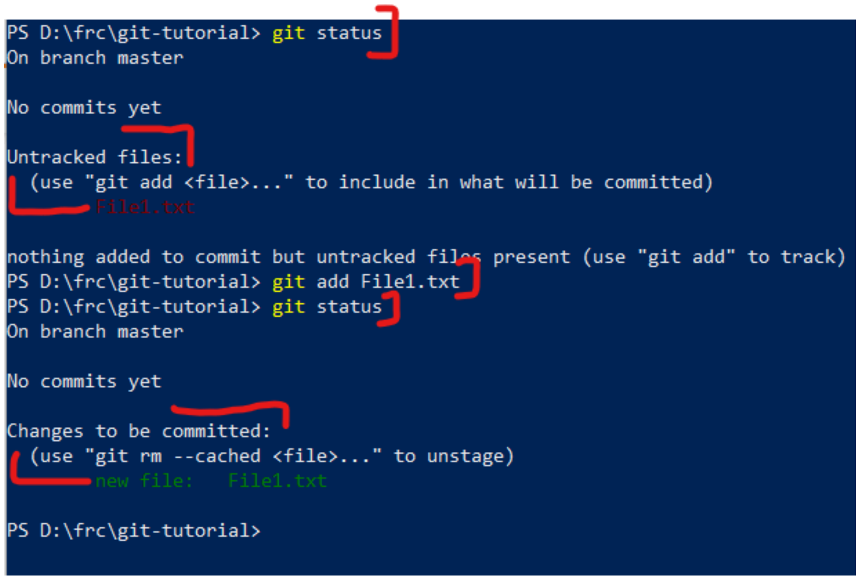
```
git init
ls -h
```

Step	Detail

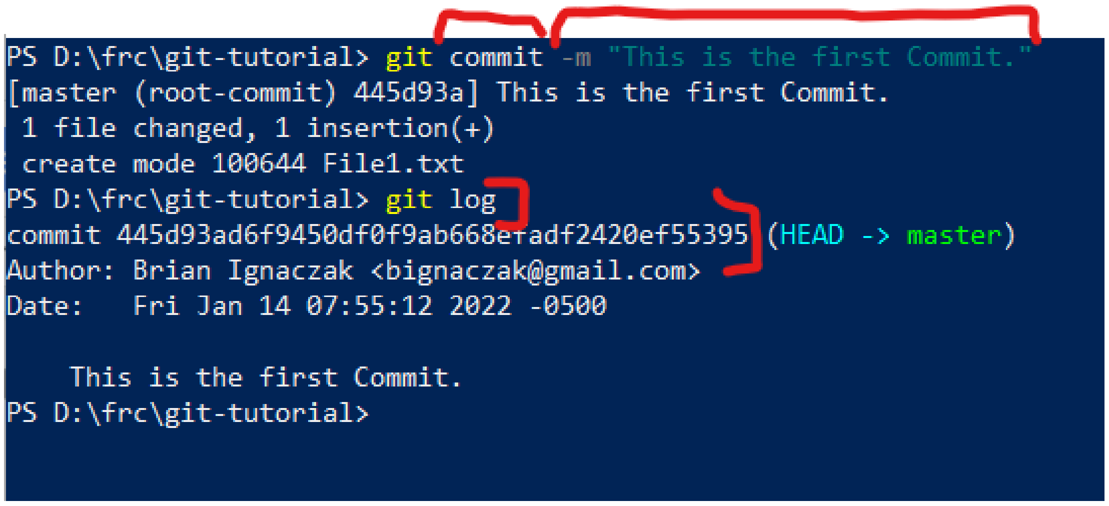
	
	`PS D:\frc\git-tutorial> git init`
	`Initialized empty Git repository in D:/FRC/git-tutorial/.git/`
	`PS D:\frc\git-tutorial> ls -h`
	
	`    Directory: D:\frc\git-tutorial`
	
	`Mode                 LastWriteTime         Lengt Name`
	`----                 -------------         ----- ----`
	`d--h--         1/14/2022   7:35 AM               .git`
	
	`PS D:\frc\git-tutorial>`
	
	D:\FRC\git-tutorial
	Date modified: 1/14/2022 7:35 AM, Type: File folder, Name: .git
	
	You should see a hidden directory named .git. The content inside that directory is managed, don’t alter that directory.

3	Create a text file in git-tutorial directory, add content to the file, and save.  D:\FRC\git-tutorial .git — 1/14/2022 7:35 AM — File folder File1.txt — 1/14/2022 7:40 AM — Text Document — 1 K File1.txt - Notepad `This is the first file for Commit #1`

4	In PowerShell, add the file as a tracked file.
	● First, check the status to see what files are tracked. Note that the created file is “**Untracked**”, meaning it won’t be included in a commit.
	● Then **add** the file created in the previous step to the list of **tracked** files
	● Then verify status has changed
	
	`git status`
	`git add File1.txt`
	`git status`

Step	Detail
	
5	In Powershell, issue a commit and add a commit message

```
git commit -m "This is the first commit."
git status
```

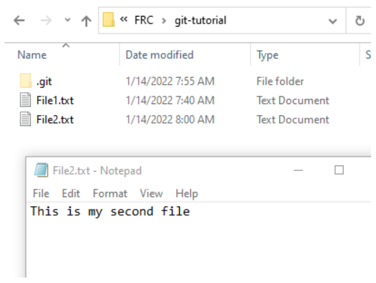
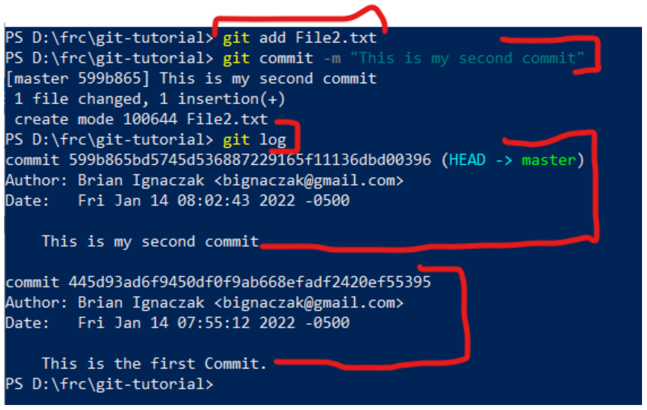


```
PS D:\frc\git-tutorial> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        File1.txt

nothing added to commit but untracked files present (use "git add" to track)
PS D:\frc\git-tutorial> git add File1.txt
PS D:\frc\git-tutorial> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   File1.txt

PS D:\frc\git-tutorial>
```

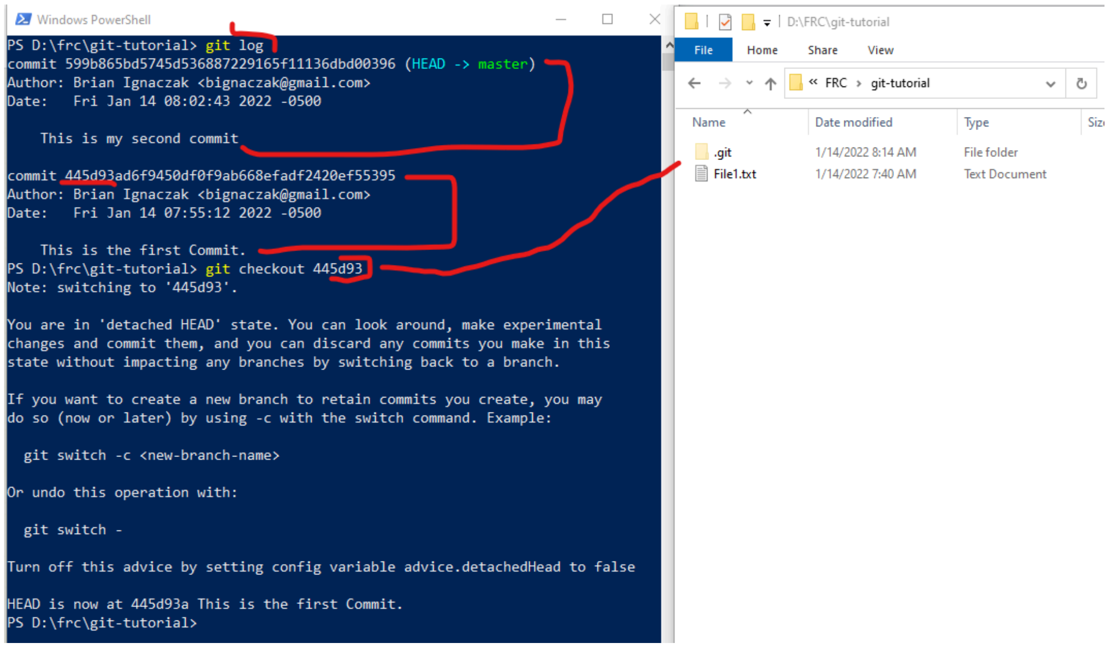
```
PS D:\frc\git-tutorial> git commit -m "This is the first Commit."
[master (root-commit) 445d93a] This is the first Commit.
 1 file changed, 1 insertion(+)
 create mode 100644 File1.txt
PS D:\frc\git-tutorial> git log
commit 445d93ad6f9450df0f9ab668efadf2420ef55395 (HEAD -> master)
Author: Brian Ignaczak <bignaczak@gmail.com>
Date:   Fri Jan 14 07:55:12 2022 -0500

    This is the first Commit.
PS D:\frc\git-tutorial>
```

Note that the long id assigned to the commit: 445d96…

Step	Detail

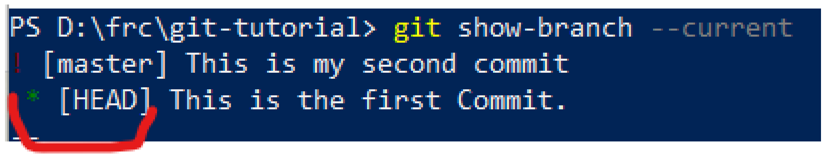
6	Repeat steps 3-5 to generate a second file and issue a second commit.
	● Add a file with some content and save it
	● Include the file in tracked changes in the git repository
	● Issue a commit and include a message
	
	 « FRC › git-tutorial
	Name
	Date modified
	Type
	File2.txt - Notepad
	File Edit Format View Help
	.git 1/14/2022 7:55 AM File folder
	File1.txt 1/14/2022 7:40 AM Text Document
	File2.txt 1/14/2022 8:00 AM Text Document
	This is my second file
	
	`PS D:\frc\git-tutorial> git add File2.txt`
	`PS D:\frc\git-tutorial> git commit -m "This is my second commit"`
	`[master 599b865] This is my second commit`
	` 1 file changed, 1 insertion(+)`
	` create mode 100644 File2.txt`
	`PS D:\frc\git-tutorial> git log`
	`commit 599b865bd5745d536887229165f11136dbd00396 (HEAD -> master)`
	`Author: Brian Ignaczak <bignaczak@gmail.com>`
	`Date:   Fri Jan 14 08:02:43 2022 -0500`
	
	`    This is my second commit`
	
	`commit 445d93ad6f9450df0f9ab668efadf2420ef55395`
	`Author: Brian Ignaczak <bignaczak@gmail.com>`
	`Date:   Fri Jan 14 07:55:12 2022 -0500`
	
	`    This is the first Commit.`
	`PS D:\frc\git-tutorial>`
	
	Note how the git log shows 2 commits with unique commit ids

7	You can visit different states of the repo by navigating commits, also known as moving the **head**. Note that you only need to include a few characters of the commit id, enough to uniquely identify it Also note that the state of directory in File Explorer has reverted back before the second file was created.

Step	Detail
	
8	Navigate the head back to the latest commit of master branch by checking it out. ● Show where the head is currently pointing `git show-branch --current`

```
PS D:\frc\git-tutorial> git log
commit 599b865bd5745d536887229165f11136dbd00396 (HEAD -> master)
Author: Brian Ignaczak <bignaczak@gmail.com>
Date:   Fri Jan 14 08:02:43 2022 -0500

    This is my second commit

commit 445d93ad6f9450df0f9ab668efadf2420ef55395
Author: Brian Ignaczak <bignaczak@gmail.com>
Date:   Fri Jan 14 07:55:12 2022 -0500

    This is the first Commit.
PS D:\frc\git-tutorial> git checkout 445d93
Note: switching to '445d93'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 445d93a This is the first Commit.
PS D:\frc\git-tutorial>
```

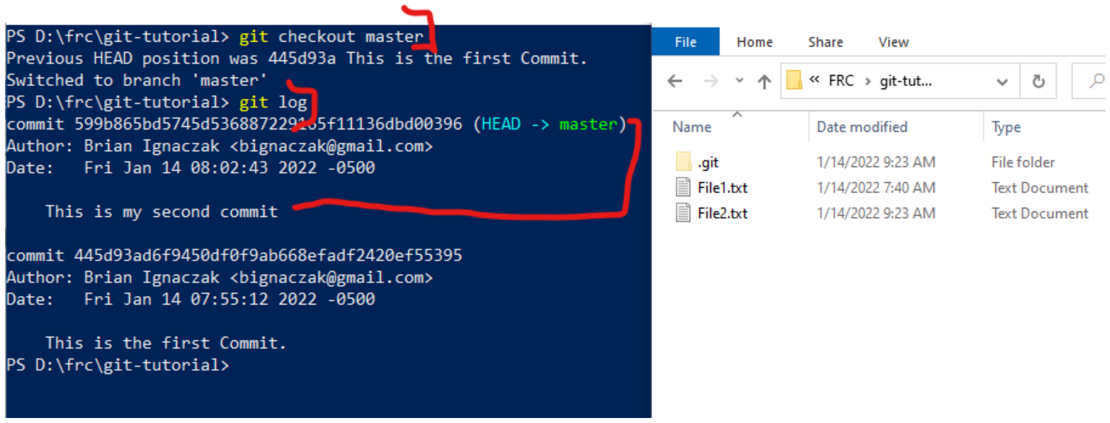
Step	Detail



```
PS D:\frc\git-tutorial> git show-branch --current
! [master] This is my second commit
 * [HEAD] This is the first Commit.
```

- Move to the latest commit in the master branch

```
git checkout master
git log
```



```
PS D:\frc\git-tutorial> git checkout master
Previous HEAD position was 445d93a This is the first Commit.
Switched to branch 'master'
PS D:\frc\git-tutorial> git log
commit 599b865bd5745d536887229165f11136dbd00396 (HEAD -> master)
Author: Brian Ignaczak <bignaczak@gmail.com>
Date:   Fri Jan 14 08:02:43 2022 -0500

    This is my second commit

commit 445d93ad6f9450df0f9ab668efadf2420ef55395
Author: Brian Ignaczak <bignaczak@gmail.com>
Date:   Fri Jan 14 07:55:12 2022 -0500

    This is the first Commit.
PS D:\frc\git-tutorial>
```

Name	Date modified	Type
.git	1/14/2022 9:23 AM	File folder
File1.txt	1/14/2022 7:40 AM	Text Document
File2.txt	1/14/2022 9:23 AM	Text Document

Note that the second file is restored

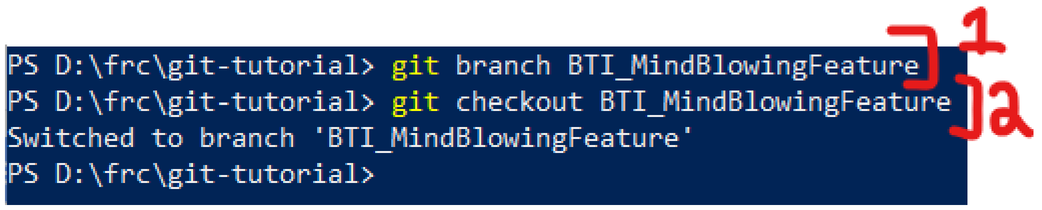
9

New software features are developed on branches.

1. Create a new branch. For name, use template { *Initials* }_{ *Feature* }
   Example: **BTI_MindBlowingFeature**
2. Checkout the new branch

Step	Detail

```
git branch <feature-name>
git checkout <feature-name>
```



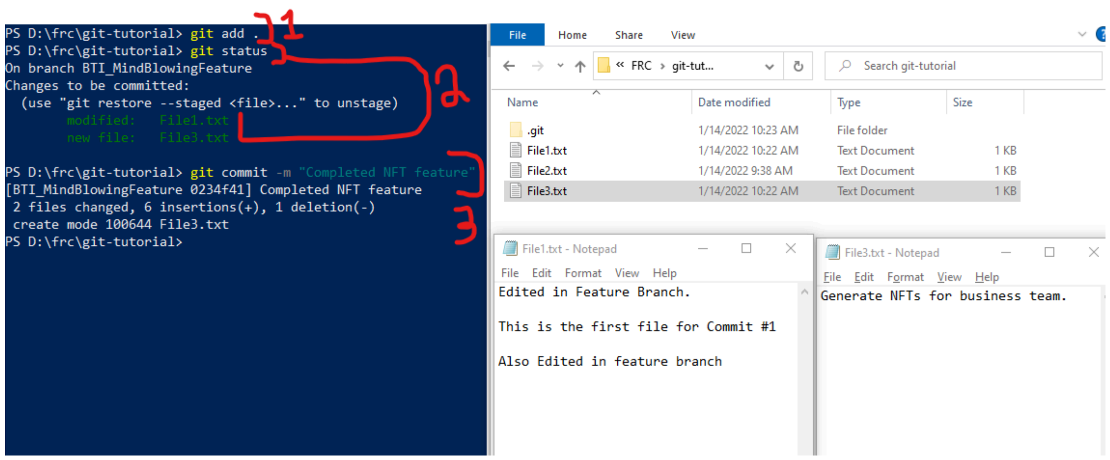
```
PS D:\frc\git-tutorial> git branch BTI_MindBlowingFeature
PS D:\frc\git-tutorial> git checkout BTI_MindBlowingFeature
Switched to branch 'BTI_MindBlowingFeature'
PS D:\frc\git-tutorial>
```

10

Do some work in the repository. Add a new file and commit it.

1. Create a **new file** and **edit File1.txt**. Add the new file to the tracked list. Use a period as a wildcard character to track all modified files
2. Verify the status of tracked files
3. Commit changes to the feature branch
4. Close any open text files

```
git add .
git status
git commit -m "Completed NFT feature"
```



```
PS D:\frc\git-tutorial> git add .
PS D:\frc\git-tutorial> git status
On branch BTI_MindBlowingFeature
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   File1.txt
        new file:   File3.txt

PS D:\frc\git-tutorial> git commit -m "Completed NFT feature"
[BTI_MindBlowingFeature 0234f41] Completed NFT feature
 2 files changed, 6 insertions(+), 1 deletion(-)
 create mode 100644 File3.txt
PS D:\frc\git-tutorial>
```

Name	Date modified	Type	Size
.git	1/14/2022 10:23 AM	File folder	
File1.txt	1/14/2022 10:22 AM	Text Document	1 KB
File2.txt	1/14/2022 9:38 AM	Text Document	1 KB
File3.txt	1/14/2022 10:22 AM	Text Document	1 KB

```
Edited in Feature Branch.

This is the first file for Commit #1

Also Edited in feature branch
```

```
Generate NFTs for business team.
```

When working with the team's repository, you should **always direct your commits to feature branches, never the master branch**.

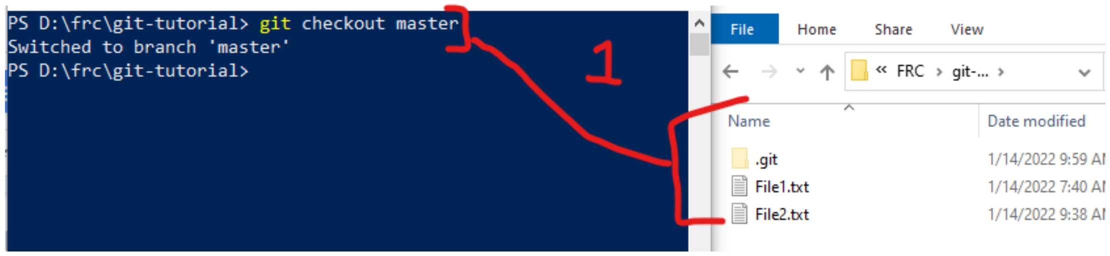
11

Navigate back to the master branch and merge the new feature branch

1. Checkout the master branch
2. Merge the feature branch

Step	Detail

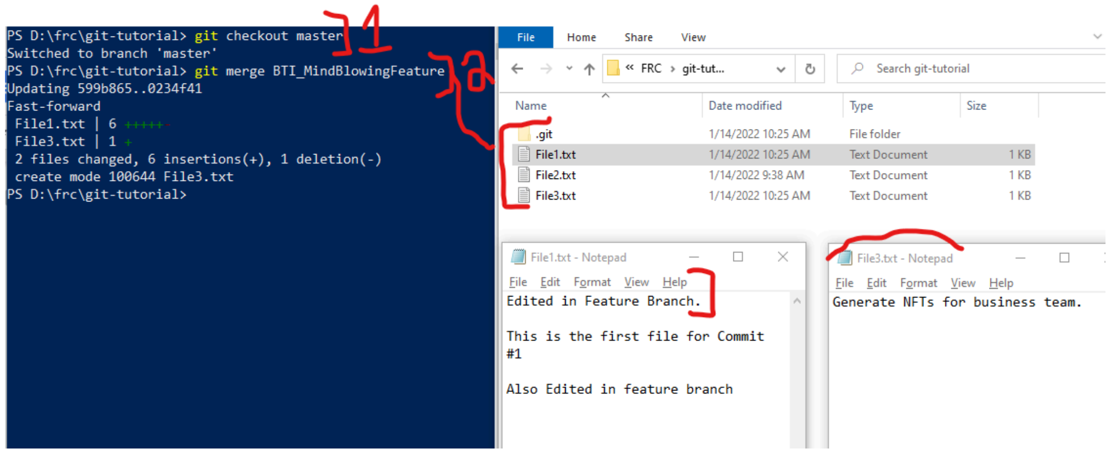
```
git checkout master
```



```
PS D:\frc\git-tutorial> git checkout master
Switched to branch 'master'
PS D:\frc\git-tutorial>
```

Note: state of the repo reverted back before 3rd file created

```
git merge BTI_MindBlowingFeature
```



```
PS D:\frc\git-tutorial> git checkout master
Switched to branch 'master'
PS D:\frc\git-tutorial> git merge BTI_MindBlowingFeature
Updating 599b865..0234f41
Fast-forward
 File1.txt | 6 +++++-
 File3.txt | 1 +
 2 files changed, 6 insertions(+), 1 deletion(-)
 create mode 100644 File3.txt
PS D:\frc\git-tutorial>
```

Name	Date modified	Type	Size
.git	1/14/2022 10:25 AM	File folder	
File1.txt	1/14/2022 10:25 AM	Text Document	1 KB
File2.txt	1/14/2022 9:38 AM	Text Document	1 KB
File3.txt	1/14/2022 10:25 AM	Text Document	1 KB

File1.txt - Notepad

```
Edited in Feature Branch.

This is the first file for Commit
#1

Also Edited in feature branch
```

File3.txt - Notepad

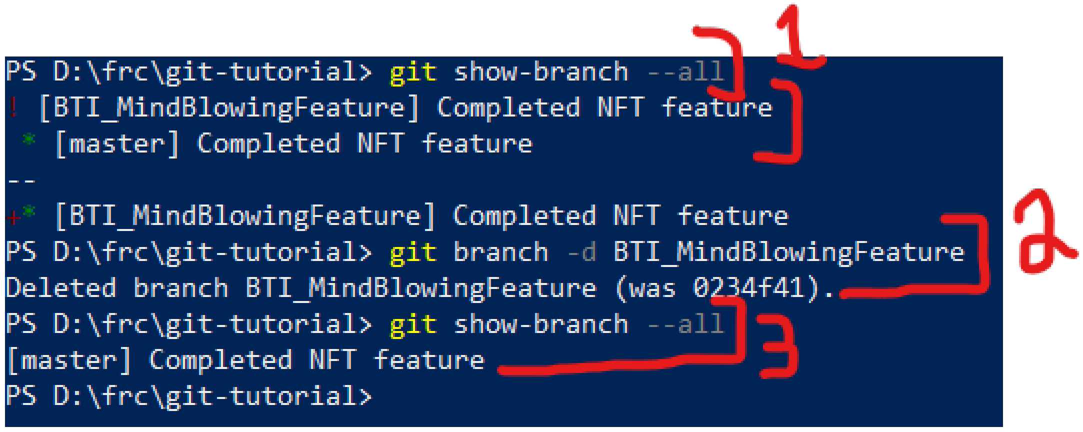
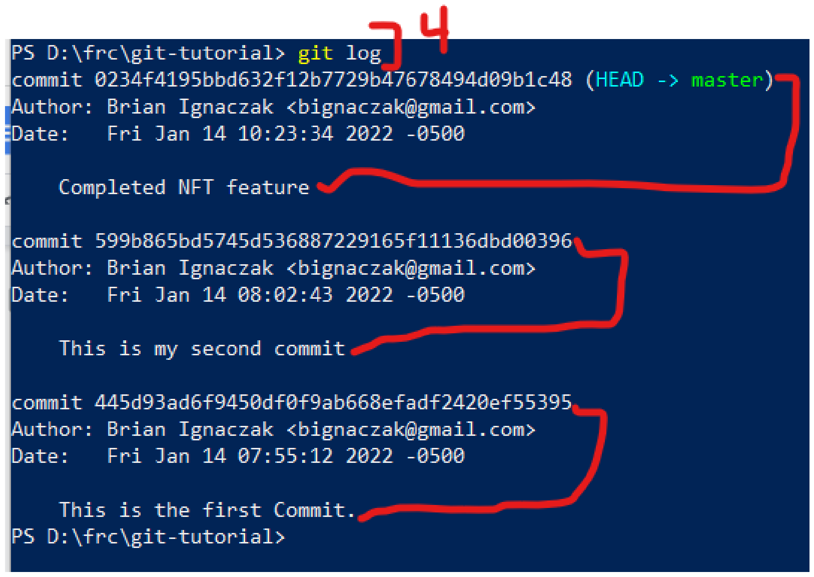
```
Generate NFTs for business team.
```

12

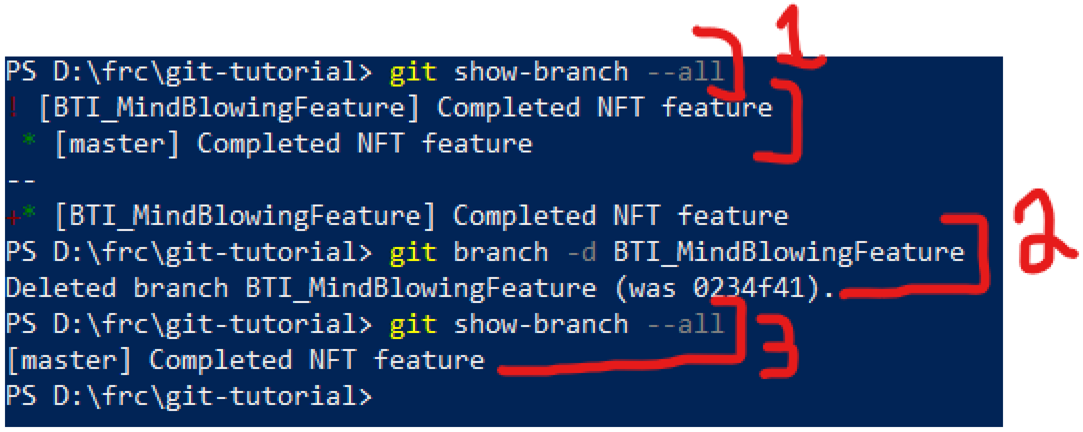
Now that the changes have been merged into the master branch, the feature branch can be deleted. Note that the commits have been incorporated into the master branch.

1. Show all branches
2. Delete the feature branch

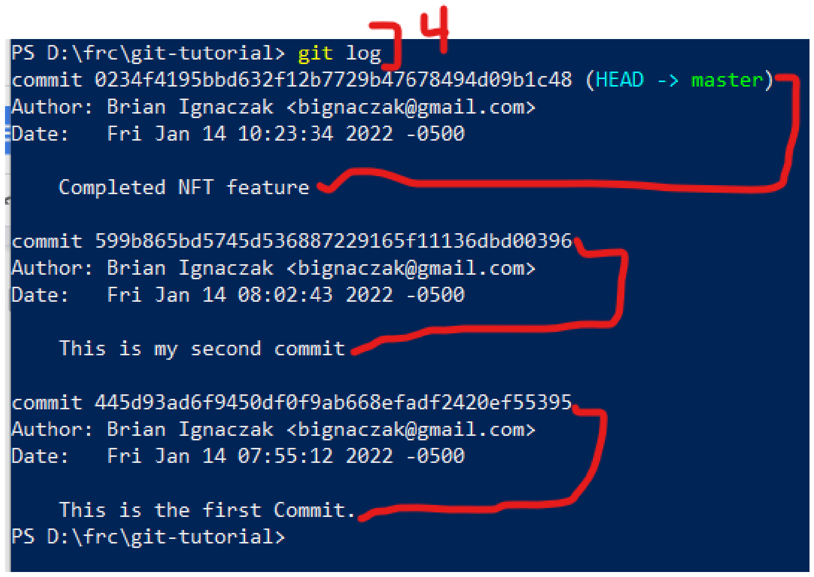
Step	Detail

	3. Verify the feature branch is gone
	4. Review the commit history of master branch
	```
	git show-branch --all
	git branch -d <feature-name>
	git show-branch --all
	git log
	```
	
	

```
git show-branch --all
git branch -d <feature-name>
git show-branch --all
git log
```



```
PS D:\frc\git-tutorial> git show-branch --all
! [BTI_MindBlowingFeature] Completed NFT feature
 * [master] Completed NFT feature
--
+* [BTI_MindBlowingFeature] Completed NFT feature
PS D:\frc\git-tutorial> git branch -d BTI_MindBlowingFeature
Deleted branch BTI_MindBlowingFeature (was 0234f41).
PS D:\frc\git-tutorial> git show-branch --all
[master] Completed NFT feature
PS D:\frc\git-tutorial>
```



```
PS D:\frc\git-tutorial> git log
commit 0234f4195bbd632f12b7729b47678494d09b1c48 (HEAD -> master)
Author: Brian Ignaczak <bignaczak@gmail.com>
Date:   Fri Jan 14 10:23:34 2022 -0500

    Completed NFT feature

commit 599b865bd5745d536887229165f11136dbd00396
Author: Brian Ignaczak <bignaczak@gmail.com>
Date:   Fri Jan 14 08:02:43 2022 -0500

    This is my second commit

commit 445d93ad6f9450df0f9ab668efadf2420ef55395
Author: Brian Ignaczak <bignaczak@gmail.com>
Date:   Fri Jan 14 07:55:12 2022 -0500

    This is the first Commit.
PS D:\frc\git-tutorial>
```

Glossary

Term	Definition

Git	Locally installed software program that tracks changes in files
GitHub	A website that acts as a distribution center for git files
Repository	A collection of files and directories that comprise the code base of a software application
Commit	An incremental unit of change in a repository, similar to clicking “save” on a file.
Commit Message	A message submitted with a commit to describe the changes it contains.
Branch	Often referred to as a feature branch, it is a separate track of code changes used to develop and test new software features. Branch is analogous to tree branches. This concept allows for concurrent development of features within a team.
Checkout	The process of navigating to a specified branch or commit. For instance, to move from the master to a feature branch, the user must checkout the feature branch
Merge	A process of integrating code from one branch into another branch. This typically occurs when the development of a feature branch is complete and it is brought into the master branch
Remote	The version of the repository that is stored in Github (or other server)
Local	The version of the repository that is stored on the user’s local computer.
Clone	The process of copying a remote repository and creating a local repository on user’s computer
Push	The process of uploading commits from a local repository onto Github.
Untracked	Files not not slated for inclusion in commit
Tracked	Files slated for inclusion in commit
Add	The process of adding files to the the tracked list
Reset	The process of removing files from the tracked list
Head	The current commit being viewed on the checked-out branch.