

CREATE-BEST

Module 1 – Source Control (Git and GitHub) and Open source licensing.

By: Miguel Garzón

<http://www.site.uottawa.ca/~mgarz042/>

Course Goals and Overview

- **Purpose:**
 - Introduction
 - Inspirations for Further Learning
- **Overview:**
 - Key Concepts
 - Installation and Setup
 - GitHub Introduction
 - Basic Git WorkFlow
 - Utility Commands

Lesson 1: The Basics

Why Git?



- **Distributed** source control system
 - Most operation in Git are local
 - Only a few commands required a network connection
- Massively scales
- **Open Source**
- Developed for Linux project requirements
 - 15 million lines of code
 - 12000 worldwide developers contributing to project
 - Dozens of active branches
- **Fast, FREE** and **Open** Source!
- Very active community!

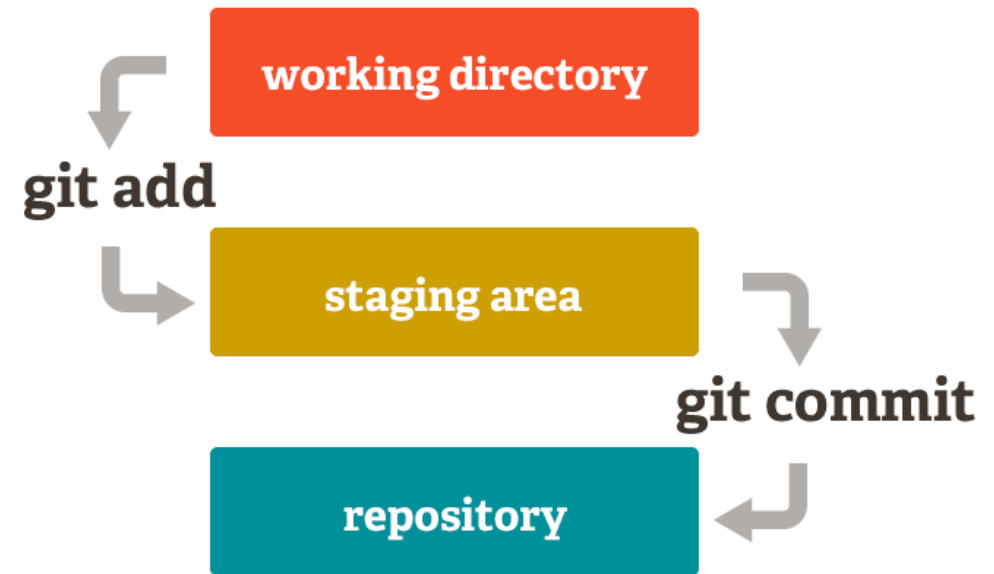
Core Concepts

- **Repository contains:**

- Files
- History
- Special configurations

- **Three (**local**) States of Git**

- **Working directory** is the directory or folder on your computer that holds all the project or application files.
- **Staging Area** (holding area or Git Index): Queues up all changes for the next commit
 - Files in the staging area can be moved in and out
- **Repository** – Git Repository (.git folder)



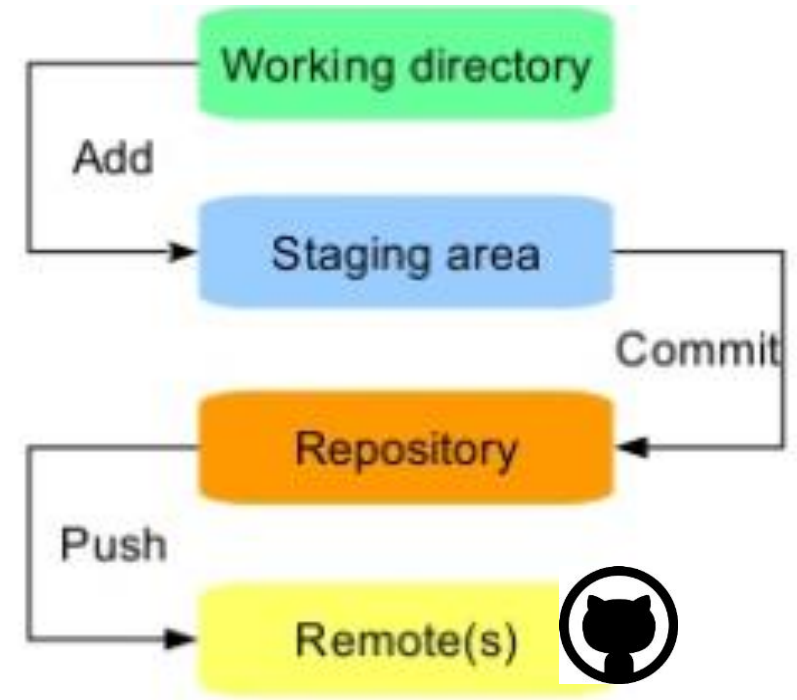
Core Concepts (cont'd)

- **Remote Repository:**

- The remote repository is just another repository with its own three states
- Last step in the basic Git Workflow

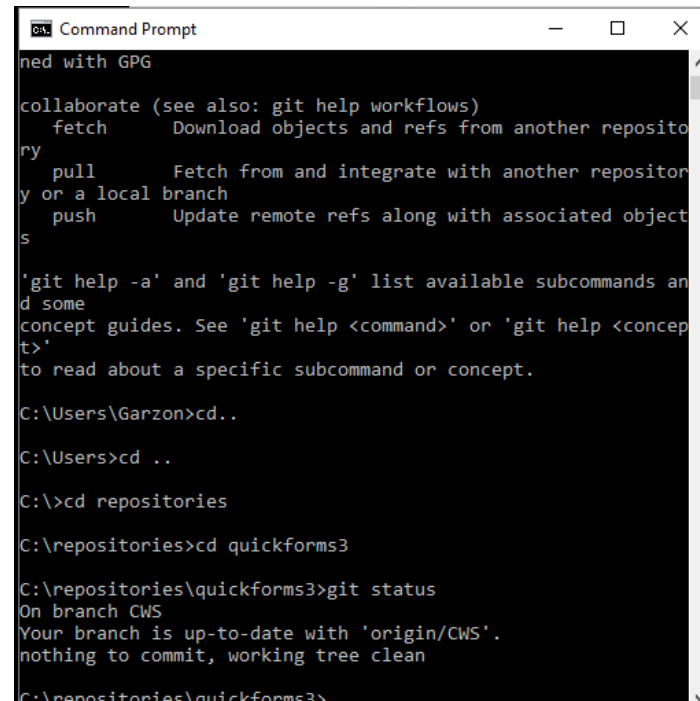
- **Master Branch**

- Times that contain your changes
- Git provides a default branch named **master**



Git - Command Line

- Git was originally designed as a command line tool, and graphical support came later.
- Features make it onto the command line before they are integrated into a graphical client
- Documentation
- More power!
- **Consistency:**
 - **Terminal** on Mac/Linux
 - **Git Bash** on Windows



```
Command Prompt
ned with GPG

collaborate (see also; git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository
            or a local branch
  push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.

C:\Users\Garzon>cd..

C:\Users>cd ..

C:\>cd repositories

C:\repositories>cd quickforms3

C:\repositories\quickforms3>git status
On branch CWS
Your branch is up-to-date with 'origin/CWS'.
nothing to commit, working tree clean

C:\repositories\quickforms3>
```

Git Installation

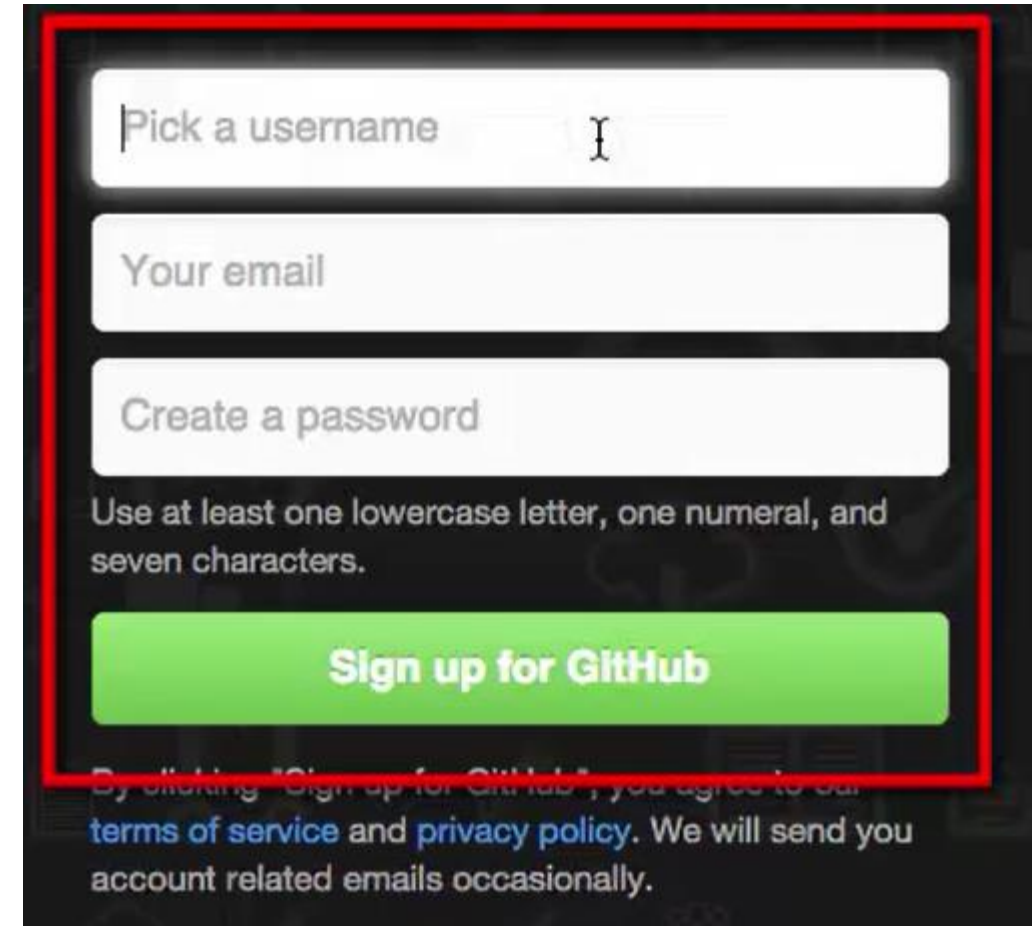
- Windows
 - <https://git-scm.com/download/windows>
- Mac OS X
 - <https://git-scm.com/download/mac>
 - Or type git version on the terminal

Git Commands will be the same!

Test your installation by typing: **git version**

GitHub – First Repository

- Navigate to GitHub.com
 - You can sign in or sign up.
 - Select the 'Unlimited public repositories for free' plan.
- Create a public repository:
 - **Name:** **createbest-github**
 - **Description:** A simple demo to show the basic Git Workflow
 - Select the option **'Initialize this repository with a README'**

A screenshot of the GitHub sign-up form. The form is set against a dark background and is enclosed in a red rectangular border. It contains three white input fields: 'Pick a username' with a cursor, 'Your email', and 'Create a password'. Below the password field is a text requirement: 'Use at least one lowercase letter, one numeral, and seven characters.' At the bottom of the form is a prominent green button with the text 'Sign up for GitHub'. Below the button, there is a line of text: 'By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We will send you account related emails occasionally.'

Prepare our local system

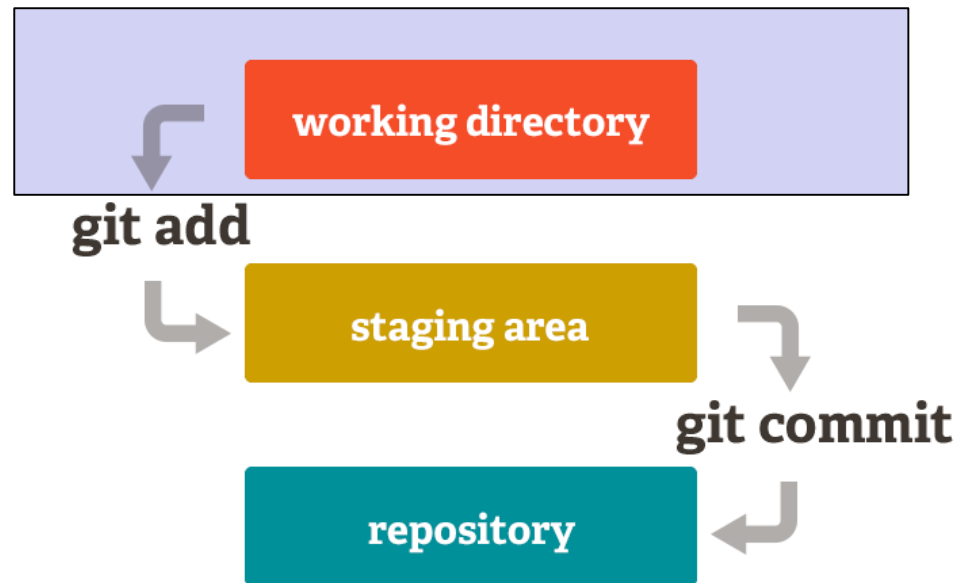
- Create a folder called **projects** using the command line
 - Place yourself under c:\
 - Type: `mkdir projects`
- Git requires two pieces of information before we proceed
 - Name: `git config --global user.name "Miguel Garzón"`
 - Email: `git config --global user.email mgarzon@uottawa.ca`

Cloning a Repository

- Using the command line, place yourself inside the folder projects (c:/projects)
- Type:
git clone https://github.com/mgarzon/createbest-github.git
- Confirm that a folder named 'createbest-github' has been created and that you have the README file
- Type
git status
and pay attention to the information displayed.

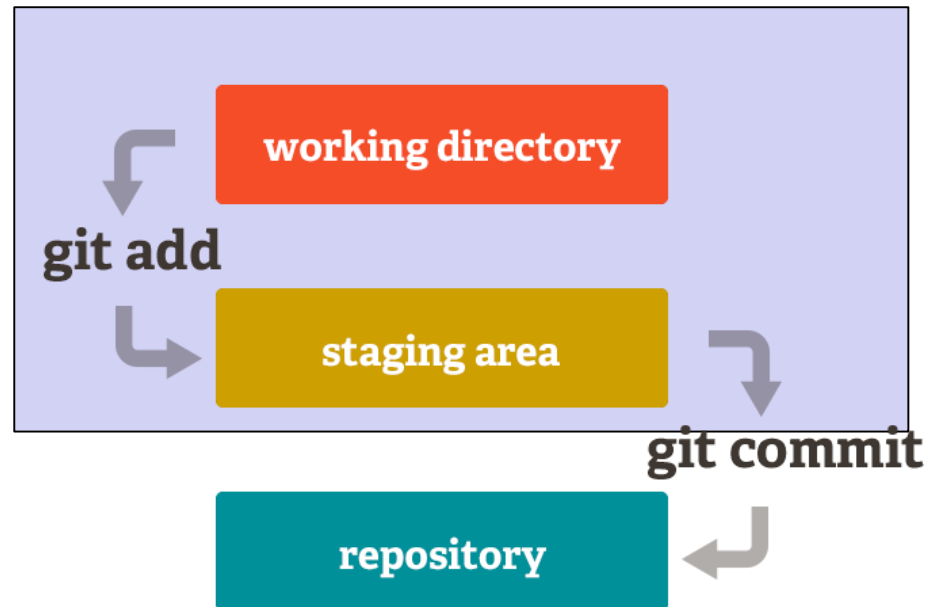
Our First commit

1. Add the file [game.js](#) to your **working repository** (C:\projects\createbest-github)
2. Type ***git status*** in the command line. What do you see?



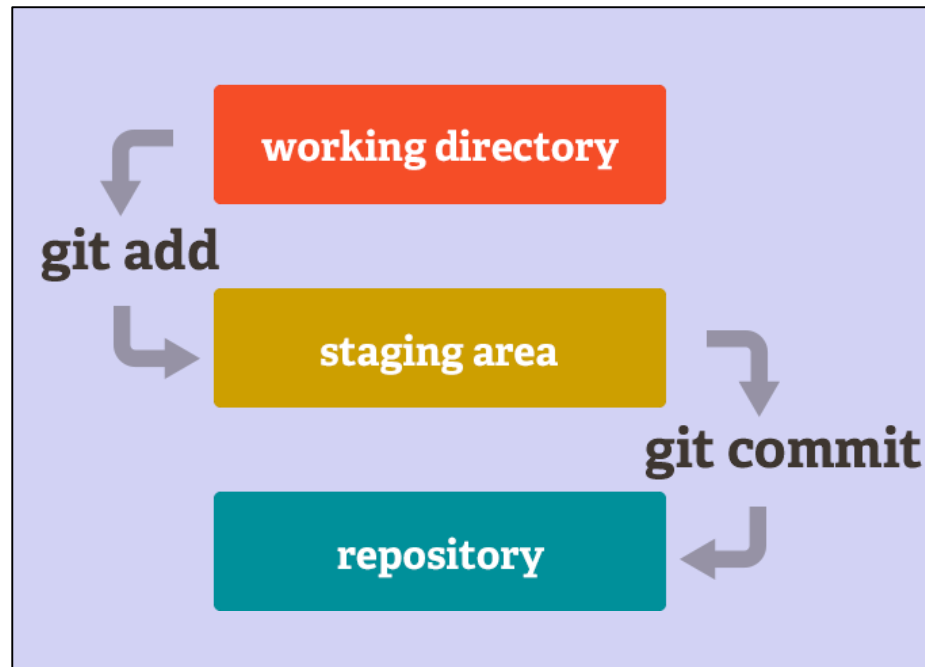
Our First commit (cont'd)

1. Type ***git add game.js***
2. Then type ***git status*** again. What do you see now?



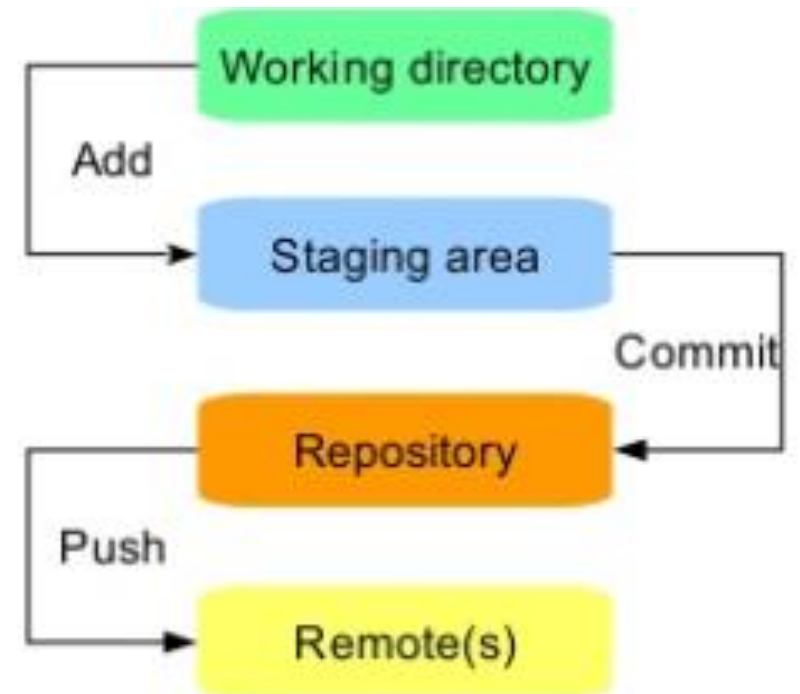
Our First commit (cont'd)

1. Type ***git commit -m "first commit"***
2. Then type ***git status*** again. What do you see now?



Publishing changes to GitHub

- Your previous commit is still a **local** command
- Type ***git push origin master***
 - **origin**: refers to the GitHub copy of our repository
 - **master**: refers to our default and only branch of the repository
- The git push command will prompt you for your GitHub **username** and **password**.
- **Check** your results.



Make Modifications to Files

- **Modify** the Readme.md by adding the following text: « Welcome to our Project ».
- **Modify** the game.js as follows:
 - Key code 37 should be 'right'.
- **Add, Commit** and **Push** your changes. Commit message should be 'Second commit'.

Commit multiple files at the same time:

git add .

git commit -m "second commit"

git push origin master

Comparing two commits

The screenshot shows a GitHub interface for the repository 'mgarzon / createbest-github'. At the top, there are navigation links for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Settings', and 'Insights'. The repository has 1 watch, 0 stars, and 0 forks. The current view is for a 'second commit' on the 'master' branch, committed by 'mgarzon' 9 minutes ago. The commit message is 'second commit'. The comparison shows 2 changed files: 'README.md' and 'game.js'. The 'README.md' file shows a diff with 4 additions and 2 deletions. The 'game.js' file shows a diff with 2 additions and 5 deletions. The diff for 'game.js' shows a change in the 'KEY_CODES' object, adding 'right' and removing 'left'.

mgarzon / createbest-github

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

second commit

master

mgarzon committed 9 minutes ago

1 parent 61dc059 commit 142f2af6715e09bcbb52d5ffc481b13c45999b7b

Showing 2 changed files with 4 additions and 2 deletions.

Unified Split

4 README.md

```
... @@ -1,3 @@
1 -# createbest-github
1 +# createbest-github
2 +
3 +WELCOME TO THIS PROJECT!
```

2 game.js

```
@@ -5,7 +5,7 @@
5 5
6 6 KEY_CODES = {
7 7   32: 'space',
8 - 37: 'left',
8 + 37: 'right',
9 9   38: 'up',
10 10  39: 'right',
11 11  40: 'down',
```

Review

- Git Introduction
- GitHub Introduction
- **Basic Workflow:**
 - Clone
 - Add/Edit Files
 - Staging Area
 - Commit
 - Pull & Push

Lesson 2: Intermediate Concepts

The Repository (local)

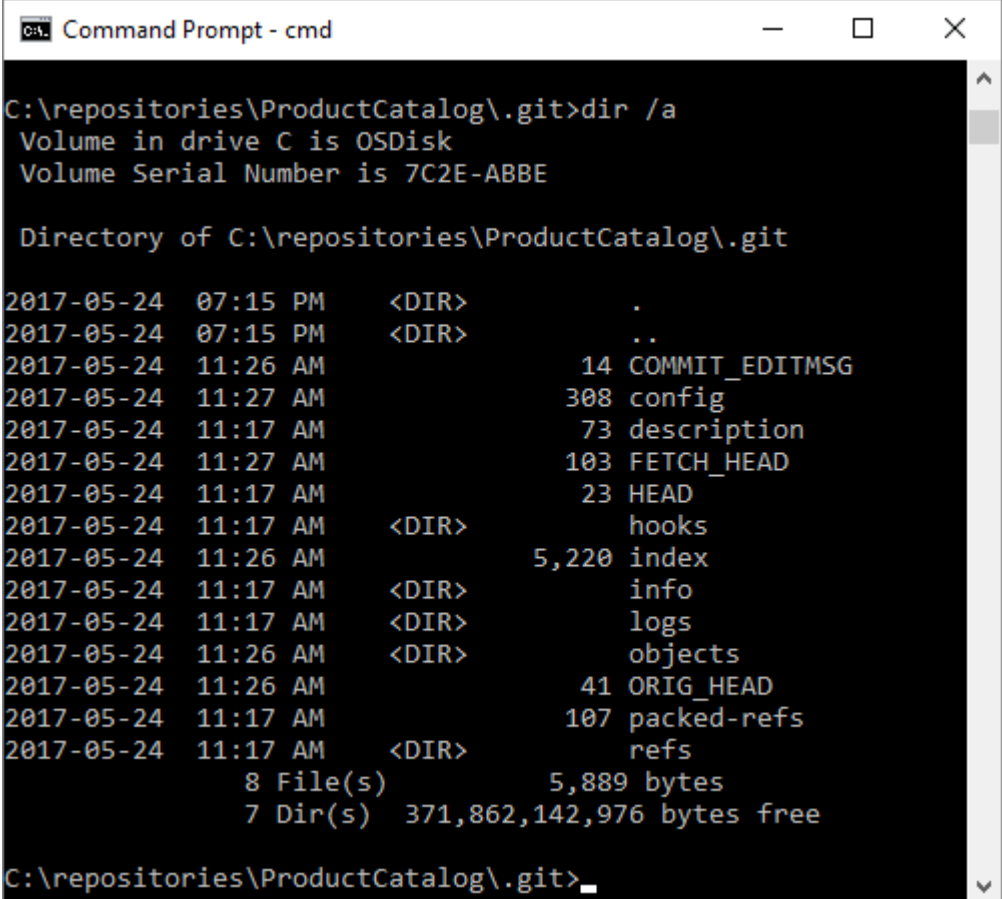
- The local repository is simply the **.git** Folder.

- Let's play with it:
 - Delete it by typing

rmdir .git /s

or

rm -rf .git (linux)



```
Command Prompt - cmd

C:\repositories\ProductCatalog\.git>dir /a
Volume in drive C is OSDisk
Volume Serial Number is 7C2E-ABBE

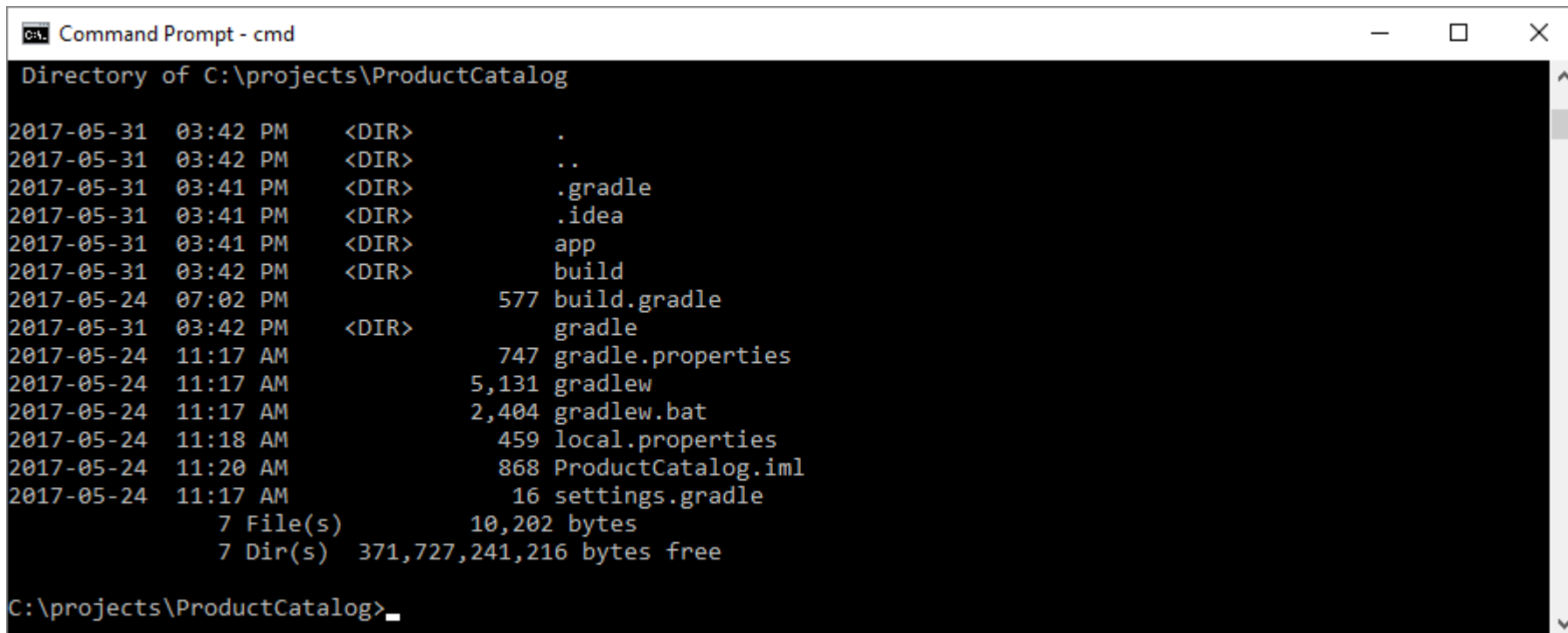
Directory of C:\repositories\ProductCatalog\.git

2017-05-24  07:15 PM    <DIR>          .
2017-05-24  07:15 PM    <DIR>          ..
2017-05-24  11:26 AM             14 COMMIT_EDITMSG
2017-05-24  11:27 AM            308 config
2017-05-24  11:17 AM             73 description
2017-05-24  11:27 AM            103 FETCH_HEAD
2017-05-24  11:17 AM             23 HEAD
2017-05-24  11:17 AM    <DIR>          hooks
2017-05-24  11:26 AM          5,220 index
2017-05-24  11:17 AM    <DIR>          info
2017-05-24  11:17 AM    <DIR>          logs
2017-05-24  11:26 AM    <DIR>          objects
2017-05-24  11:26 AM             41 ORIG_HEAD
2017-05-24  11:17 AM            107 packed-refs
2017-05-24  11:17 AM    <DIR>          refs
                        8 File(s)          5,889 bytes
                        7 Dir(s)  371,862,142,976 bytes free

C:\repositories\ProductCatalog\.git>
```

Starting With an Existing Project

1. Remember our ProductCatalog Project. We will start with this existing project.
2. **Move** this project to the *projects* folder we created before

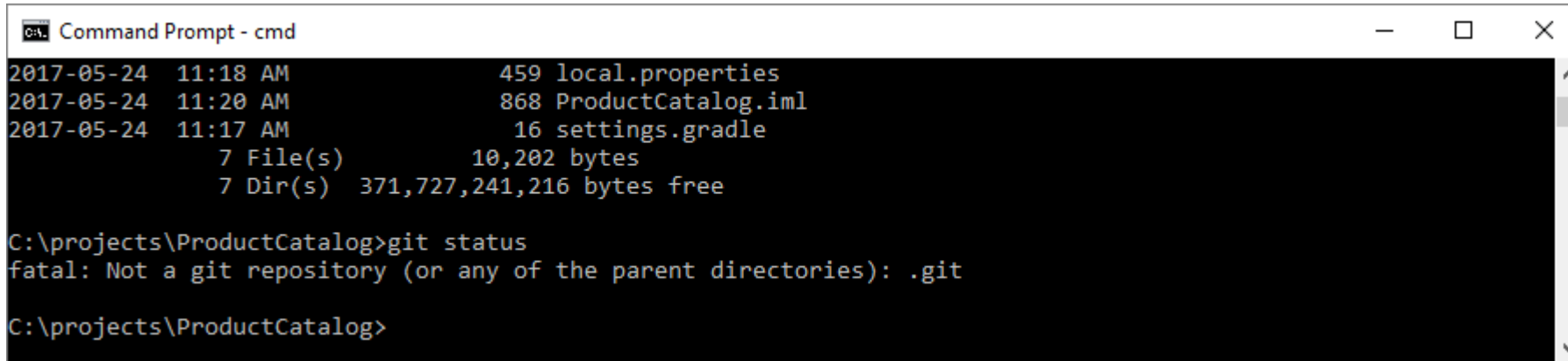


```
Command Prompt - cmd
Directory of C:\projects\ProductCatalog

2017-05-31  03:42 PM    <DIR>          .
2017-05-31  03:42 PM    <DIR>          ..
2017-05-31  03:41 PM    <DIR>          .gradle
2017-05-31  03:41 PM    <DIR>          .idea
2017-05-31  03:41 PM    <DIR>          app
2017-05-31  03:42 PM    <DIR>          build
2017-05-24  07:02 PM      577 build.gradle
2017-05-31  03:42 PM    <DIR>          gradle
2017-05-24  11:17 AM      747 gradle.properties
2017-05-24  11:17 AM    5,131 gradlew
2017-05-24  11:17 AM    2,404 gradlew.bat
2017-05-24  11:18 AM      459 local.properties
2017-05-24  11:20 AM      868 ProductCatalog.iml
2017-05-24  11:17 AM       16 settings.gradle
                7 File(s)      10,202 bytes
                7 Dir(s)  371,727,241,216 bytes free

C:\projects\ProductCatalog>
```

Starting With an Existing Project (2)



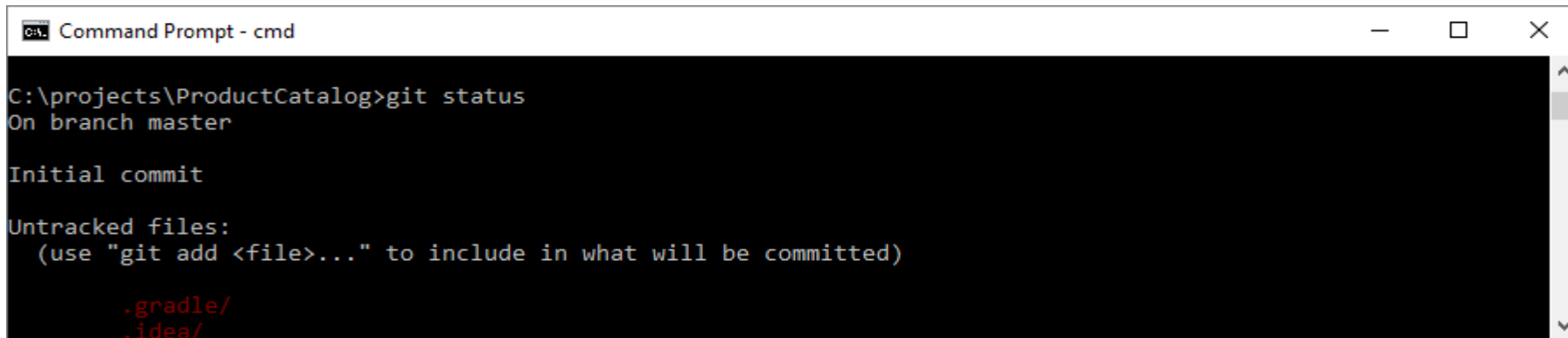
```
Command Prompt - cmd
2017-05-24 11:18 AM          459 local.properties
2017-05-24 11:20 AM          868 ProductCatalog.iml
2017-05-24 11:17 AM           16 settings.gradle
          7 File(s)         10,202 bytes
          7 Dir(s)  371,727,241,216 bytes free

C:\projects\ProductCatalog>git status
fatal: Not a git repository (or any of the parent directories): .git

C:\projects\ProductCatalog>
```

Starting With an Existing Project (3)

1. To place this folder under version control with Git, we use the **init** command.
git init .
 - At this point, Git has initialized an empty Git repository in the current folder.
2. Verify that a .git folder exists in the current folder. Type **dir /a** (or **ls -al**)
3. Type *git status* to check the current state of the (local) repository



```
Command Prompt - cmd

C:\projects\ProductCatalog>git status
On branch master

Initial commit

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

        .gradle/
        .idea/
```

Commit Untracked Files

- Before committing our changes let's create a file called **LICENSE.MD**

Commit all files at the same time:

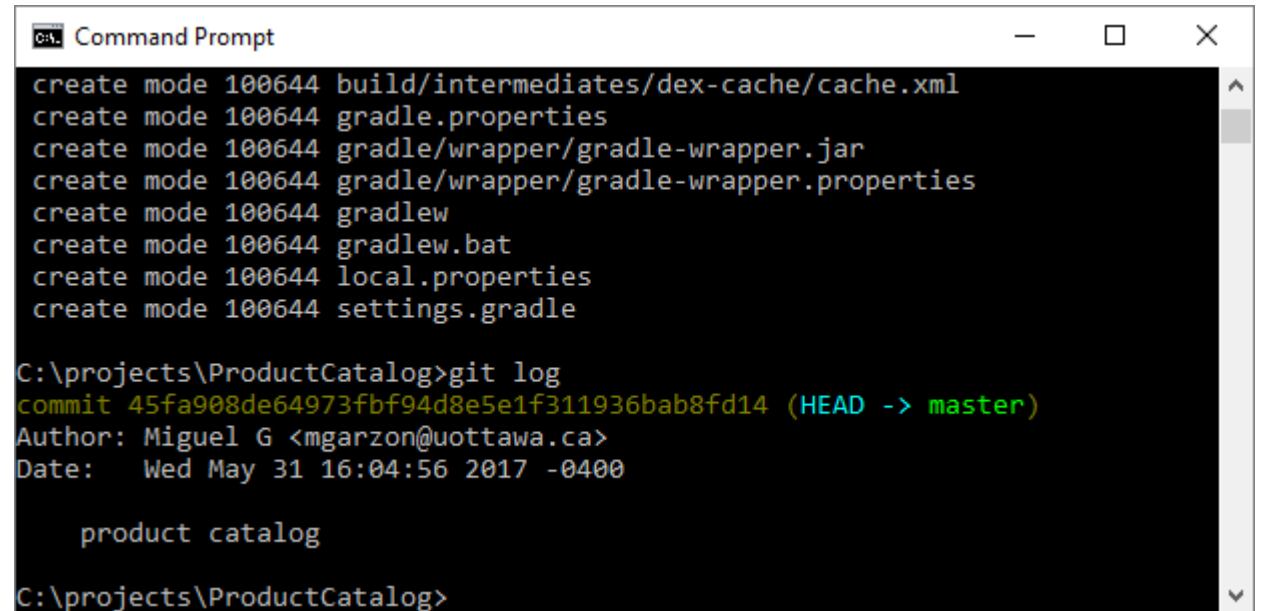
git add .

git commit -m "product catalog app"

Don't push it! We still want to play with the staging area!

Logs!

- Use the ***git log*** to see the list of commits made locally. The log contains:
 - A long identifier (SHA-1 identifier)
 - Author
 - Timestamp
 - Commit message
- ***git show*** gives you more details
Try it!



```
C:\ Command Prompt
create mode 100644 build/intermediates/dex-cache/cache.xml
create mode 100644 gradle.properties
create mode 100644 gradle/wrapper/gradle-wrapper.jar
create mode 100644 gradle/wrapper/gradle-wrapper.properties
create mode 100644 gradlew
create mode 100644 gradlew.bat
create mode 100644 local.properties
create mode 100644 settings.gradle

C:\projects\ProductCatalog>git log
commit 45fa908de64973fbf94d8e5e1f311936bab8fd14 (HEAD -> master)
Author: Miguel G <mgarzon@uottawa.ca>
Date:   Wed May 31 16:04:56 2017 -0400

    product catalog

C:\projects\ProductCatalog>
```

Making changes to Staging area that will be reverted

- **Make** any changes to the file **build.gradle**
- **Add** the changes to the staging area

git add build.gradle

- **Unstage** your changes 😊 (Unstage the file)

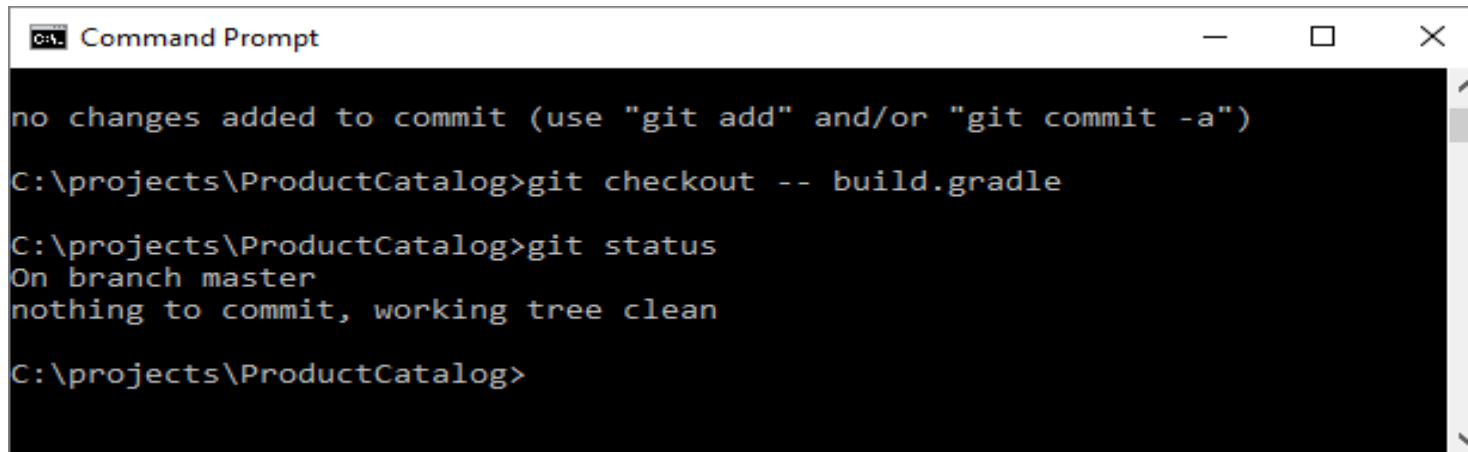
git reset HEAD build.gradle

Making changes to Staging area that will be reverted

Were the changes reverted?

Reverting Changes

- The changes were **unstaged** but not reverted. To revert:
 - Type *git checkout -- build.gradle*
- Check the contents of the file now. What happened?



```
Command Prompt

no changes added to commit (use "git add" and/or "git commit -a")

C:\projects\ProductCatalog>git checkout -- build.gradle

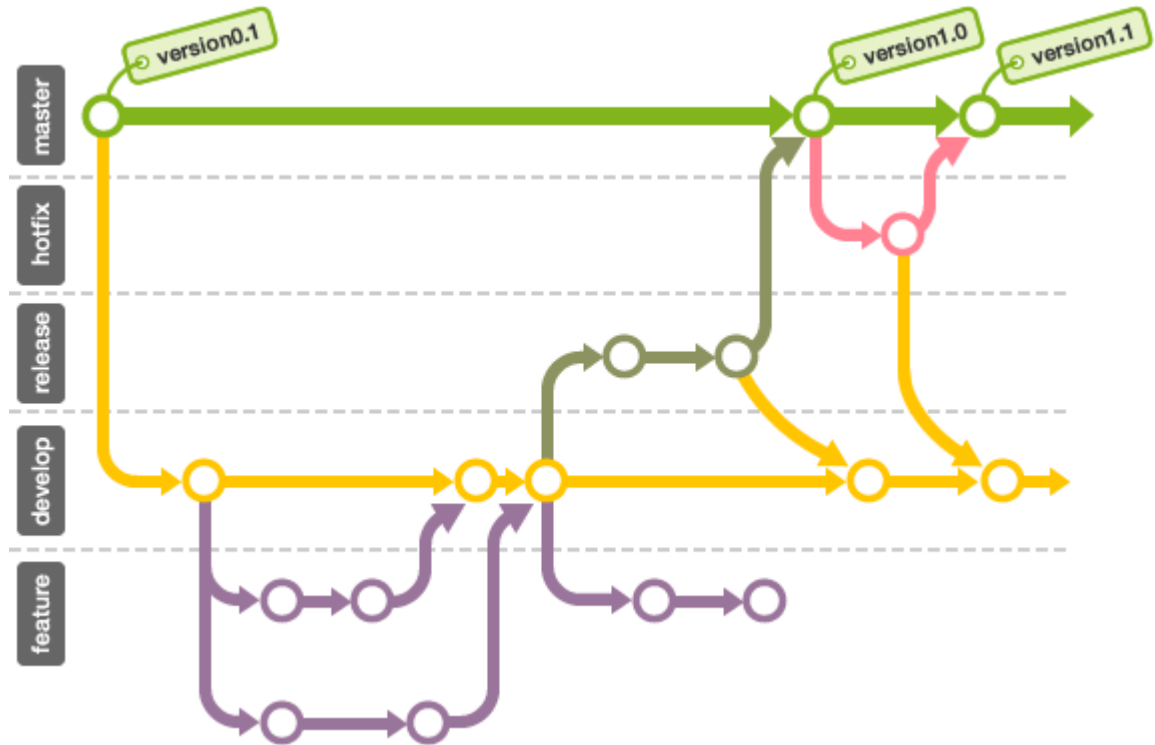
C:\projects\ProductCatalog>git status
On branch master
nothing to commit, working tree clean

C:\projects\ProductCatalog>
```

Lesson 2: Advanced Concepts

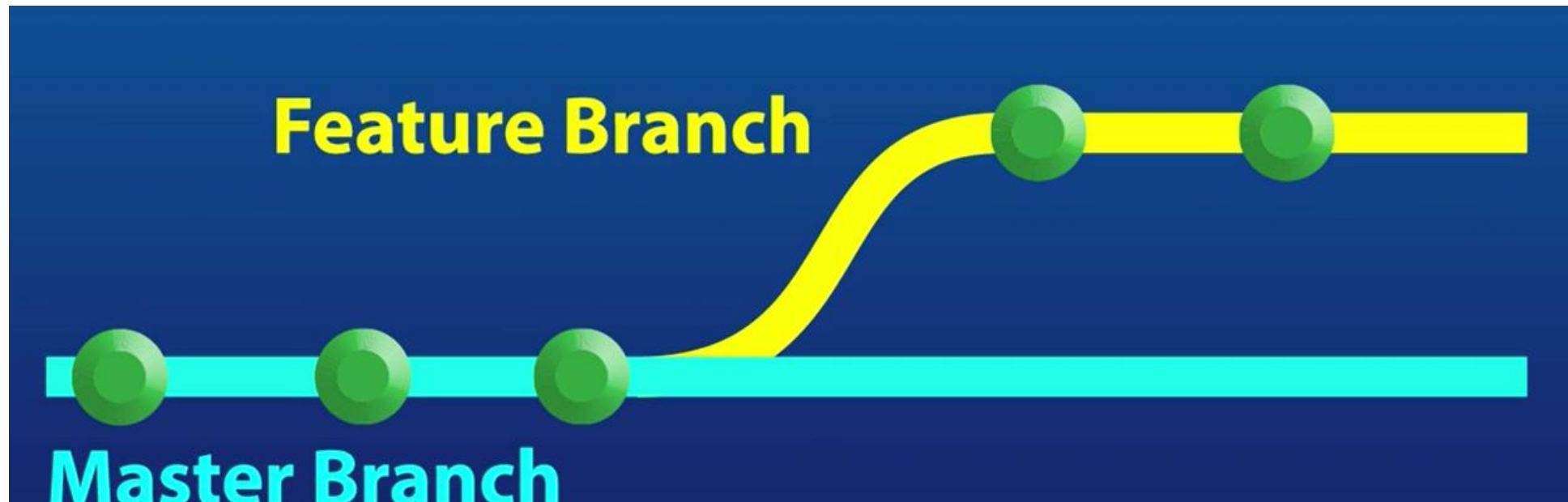
Branches

- A branch is a timeline of commits
- Branch names are labels!
 - Deletion removes label only



Creating a Branch

- We will rejoin the Master Branch later.



Merging Scenarios

1. Fast-Forward:

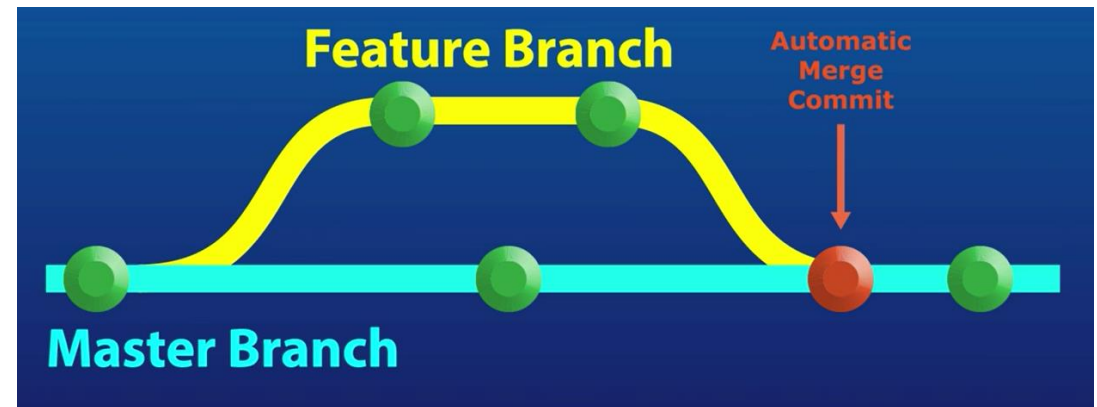
- Simplest case
- No additional work has been detected on the Master (parent) Branch



Merging Scenarios (cont'd)

2. Automated

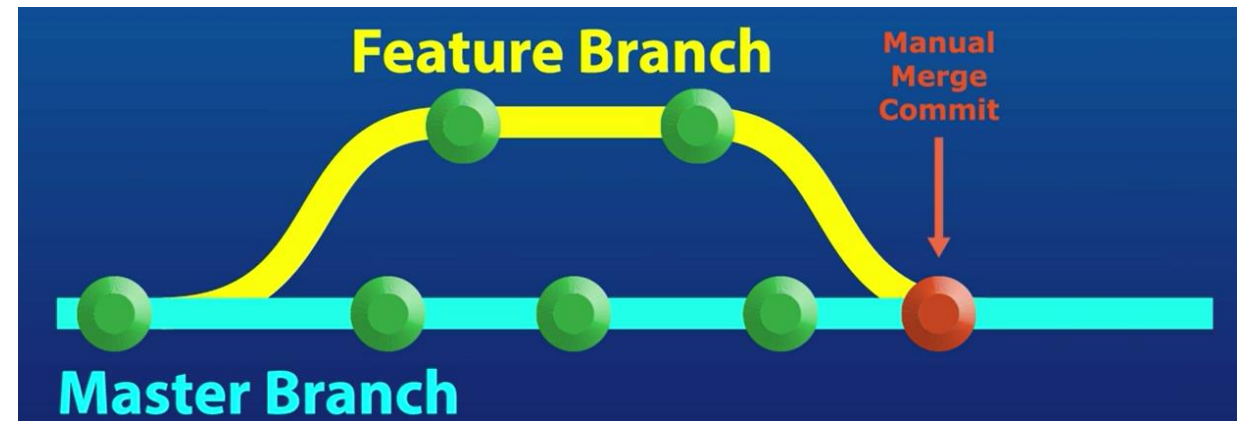
- Non-Conflicting states
- Preserves both Timelines



Merging Scenarios (cont'd)

3. Manual

- Conflicting Merge State
- Git is unable to automatically resolve any conflicts
- Merge conflicts must be resolved before doing the commit



Branching

- Type ***git branch*** to see the list of branches
- To **create** the branch and **switch** to that branch, type:

git checkout -b devbranch

- Now, modify the README file and add, commit and push it.

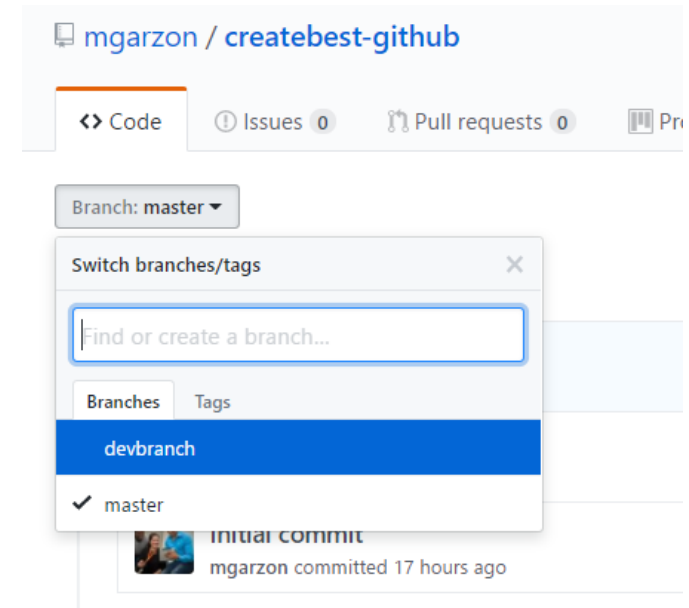
git add .

git commit -m "committing to another branch"

git push origin devbranch

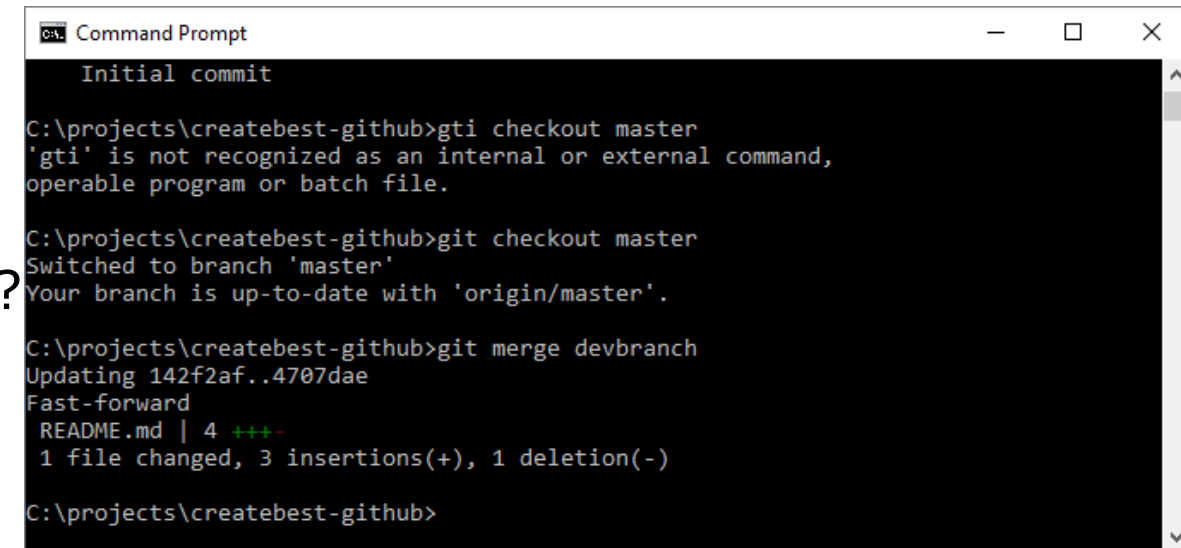
- You can compare the branches by typing:

git diff master devbranch



Merging

- To integrate the changes into the master branch:
 1. **Switch** to my parent branch (master): *git checkout master*
 2. Type *git log -all*. What do you see?
 3. Merge : *git merge devbranch*
 4. Type *git log -all*. What do you see now?
 5. Now delete the branch
git branch -d devbranch



```
Initial commit

C:\projects\createbest-github>gti checkout master
'gti' is not recognized as an internal or external command,
operable program or batch file.

C:\projects\createbest-github>git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.

C:\projects\createbest-github>git merge devbranch
Updating 142f2af..4707dae
Fast-forward
 README.md | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)

C:\projects\createbest-github>
```

Lesson 4: Open Source Licenses

Open Source Licenses

- Open source licenses are licenses that comply with the [Open Source Definition](#) — in brief, they allow software to be freely used, modified, and shared.



I want it simple and permissive.

The **MIT License** is a permissive license that is short and to the point. It lets people do anything they want with your code as long as they provide attribution back to you and don't hold you liable.

jQuery, **.NET Core**, and **Rails** use the MIT License.



I'm concerned about patents.

The **Apache License 2.0** is a permissive license similar to the MIT License, but also provides an express grant of patent rights from contributors to users.

Android, **Apache**, and **Swift** use the Apache License 2.0.



I care about sharing improvements.

The **GNU GPLv3** is a copyleft license that requires anyone who distributes your code or a derivative work to make the source available under the same terms, and also provides an express grant of patent rights from contributors to users.

Bash, **GIMP**, and **Privacy Badger** use the GNU GPLv3.

Open source licenses

- MIT: <https://choosealicense.com/licenses/mit/>
- Apache: <https://choosealicense.com/licenses/apache-2.0/>
- GNU: <https://choosealicense.com/licenses/gpl-3.0/>

How to apply the license?

Create a text file (typically named LICENSE or LICENSE.txt) in the root of your source code and copy the text of the license into the file.

More info: <https://choosealicense.com/licenses/>

THANK YOU!