GIT & GITHUB

2110215 PROG METH

WHAT IS GIT?

GIT IS A

VERSION CONTROL SYSTEM

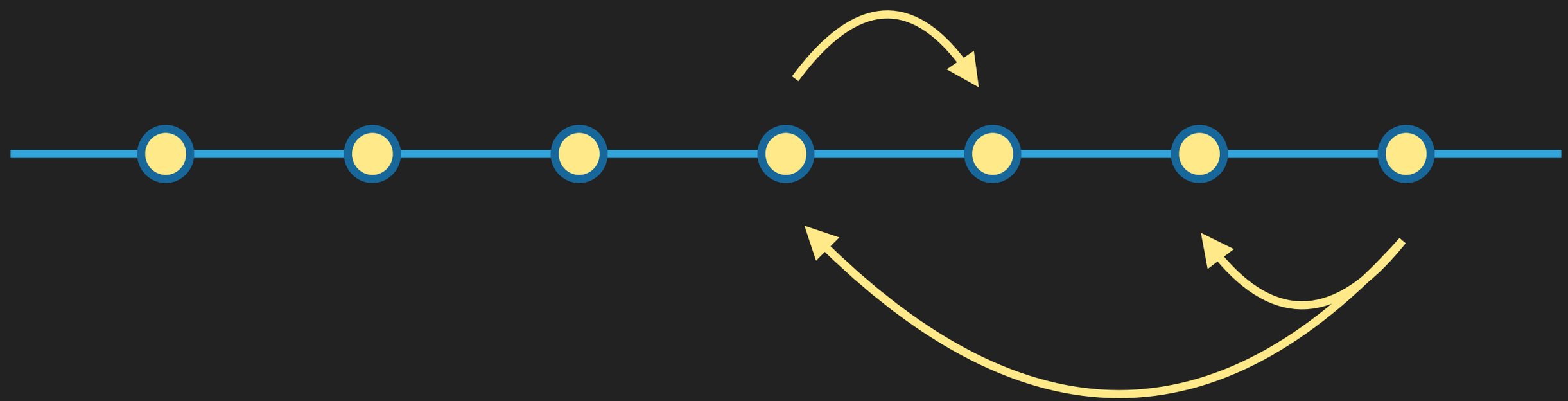
So what is this "Version Control"?

WHAT IS "VERSION CONTROL"?

- Imagine that you are playing a Pokemon game
- What would you do every time before quitting the game?
- You would create a Save point
- So that you can continue your adventure from where you left off the last time

WHAT IS "VERSION CONTROL"?

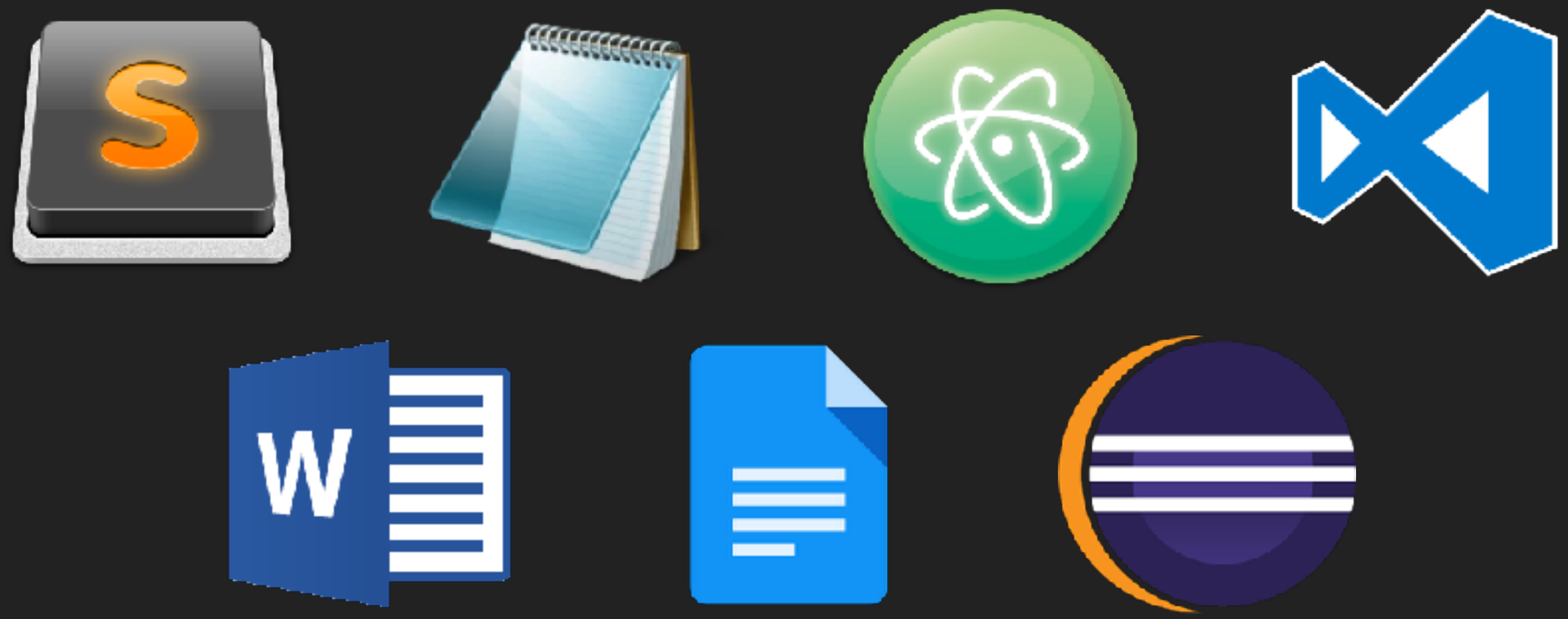
- Sometime, you might even go back and replay from the last save point again if you are unsatisfied with something.
- For example, you might unintentionally defeat a legendary Pokemon and lose a chance to catch it



In **Version Control**, you create and manage save points so that you can easily go back and forth to any of these save points whenever you want.

VERSION CONTROL IN DAILY USE

- Without notice, you have been using this feature for a long time in one certain kind of application
- That application is Text Editor/Code Editor



VERSION CONTROL IN DAILY USE

- Open up you favorite text editor/code editor
- Type some content
  (how about "version control is dull")
- Change one of the words
  (e.g. change "dull" to "life-changing awesome")
- Now (here it come…!)
  press cmd + z or ctrl + z

WOW!

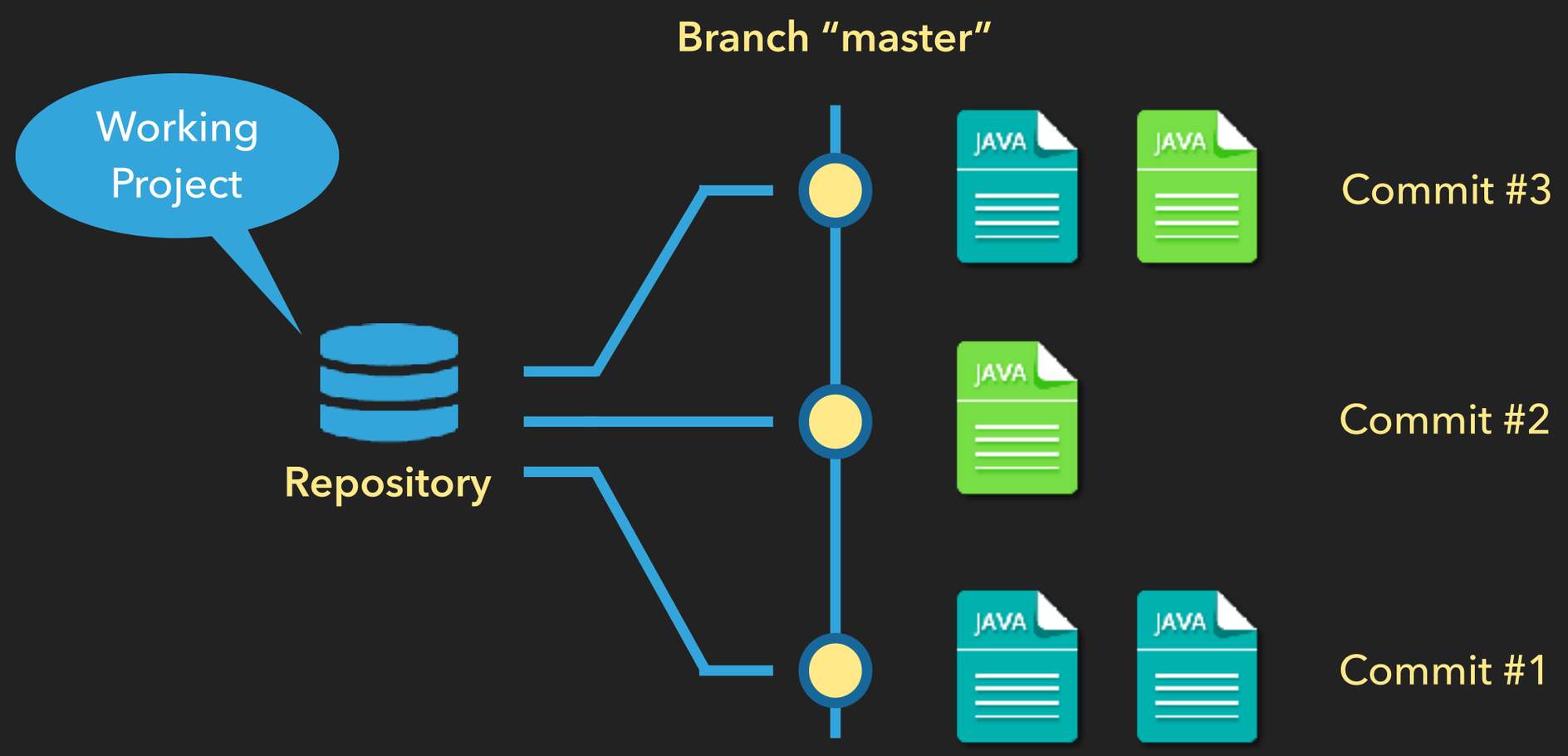
VERSION CONTROL IN ACTION!

WHAT IS GIT?

WHAT IS GIT?

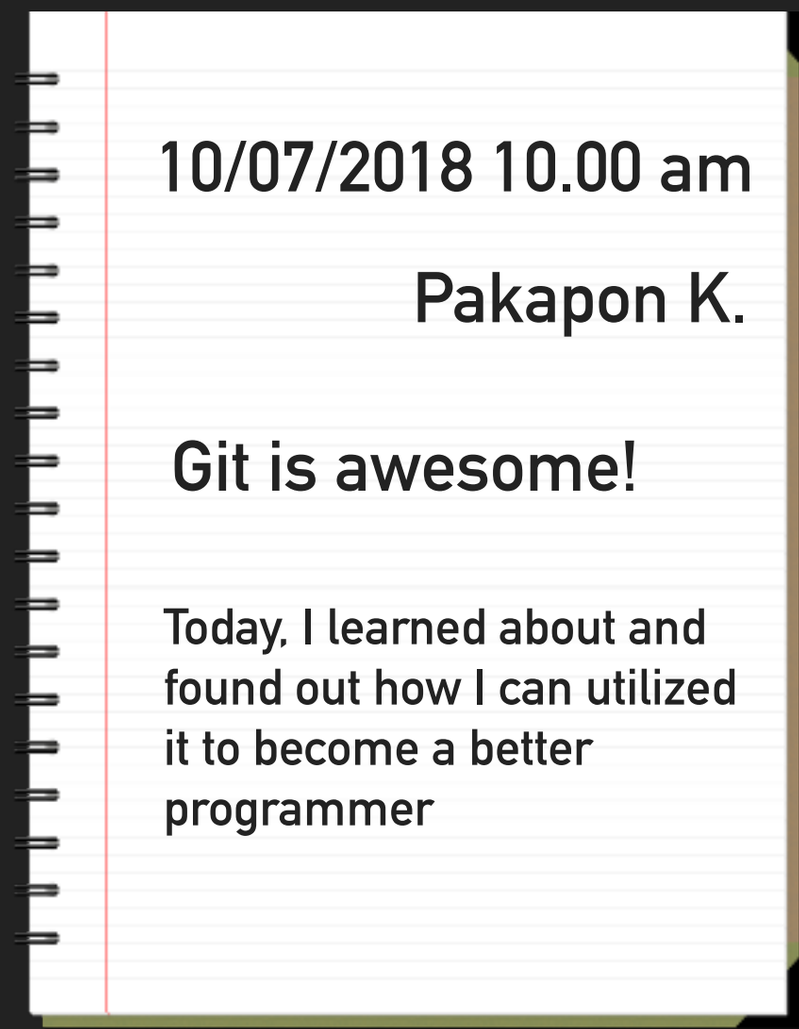
- Git is a version control system that **HELP** developers create and manage different versions of their projects by keeping track of changes in the form of a history log.
- There are 3 terms that you should be familiar with.
  - 1. Repository 
  - 2. Commit 
  - 3. Branch 

WHAT IS GIT?


Branch "master"
Working Project
Repository
JAVA
JAVA
Commit #3
JAVA
Commit #2
JAVA
JAVA
Commit #1

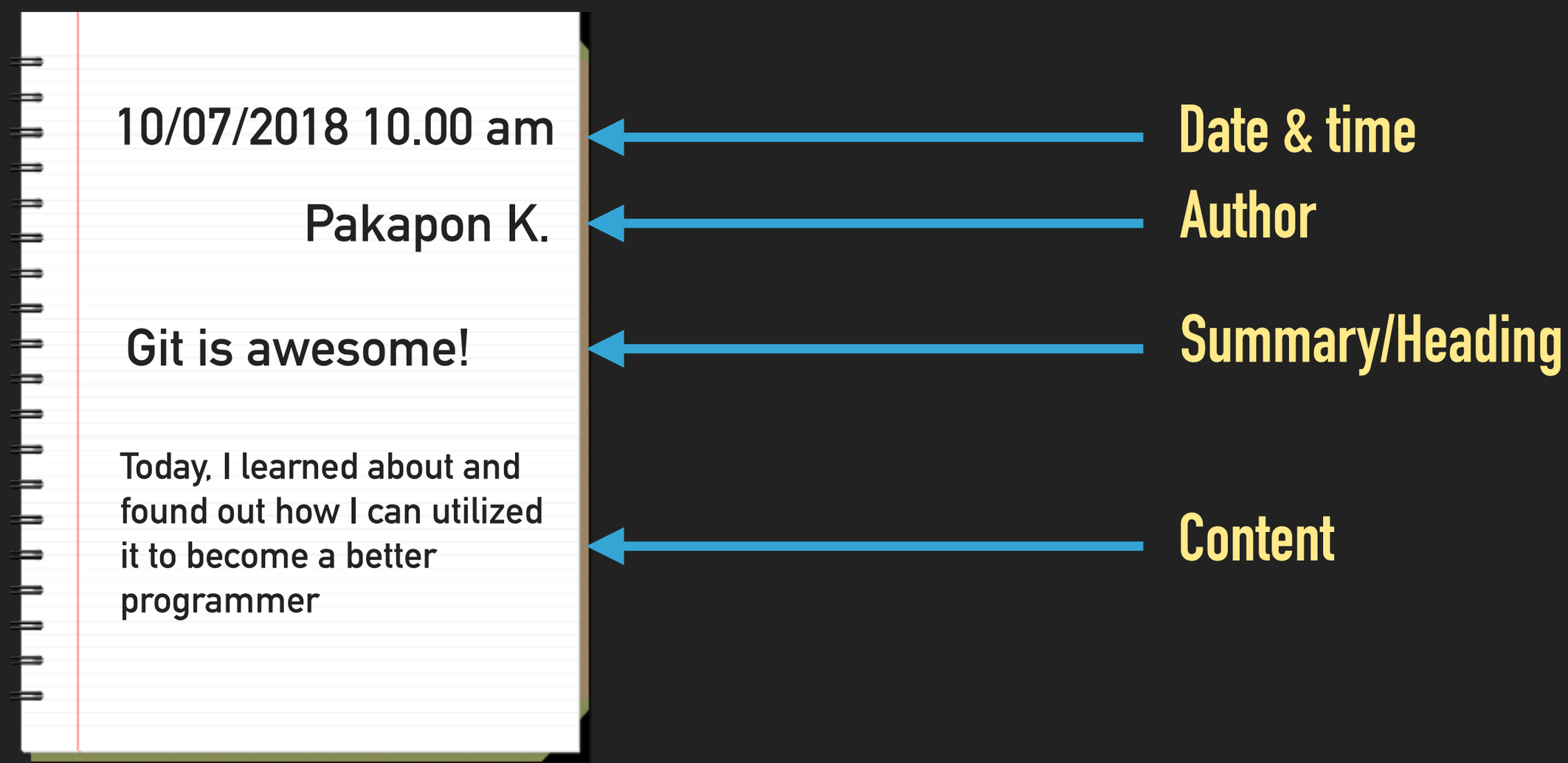
WHAT IS GIT?

- Repository is a notebook where you group the changes of each file into logs. Each log is written on a separated page.



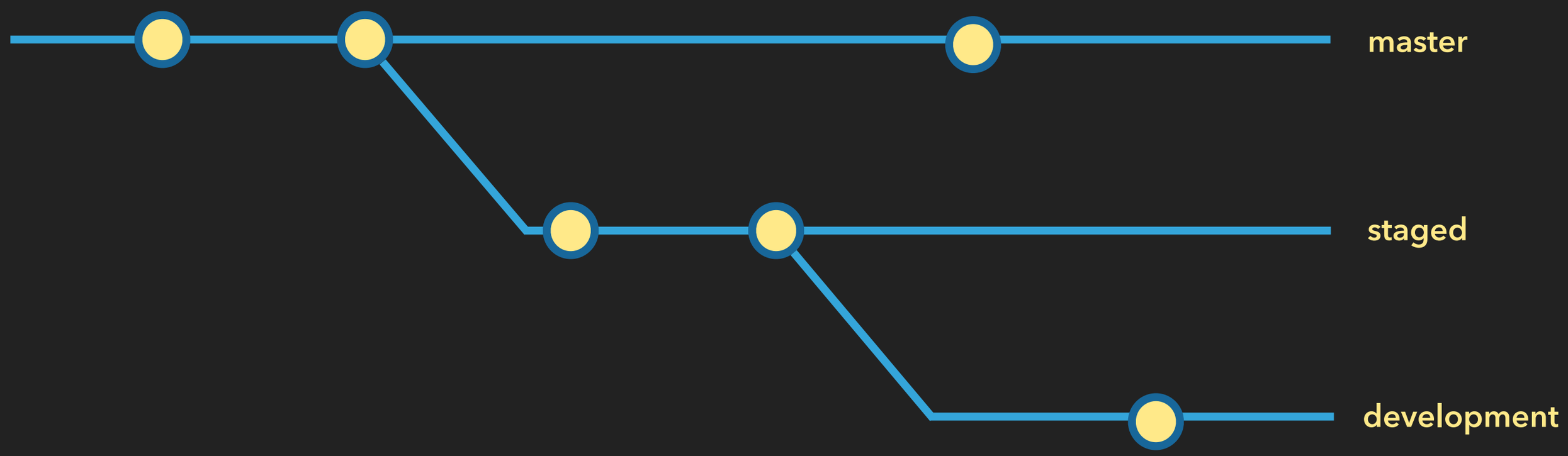
WHAT IS GIT?

- Commit is a log that written on each page of the notebook



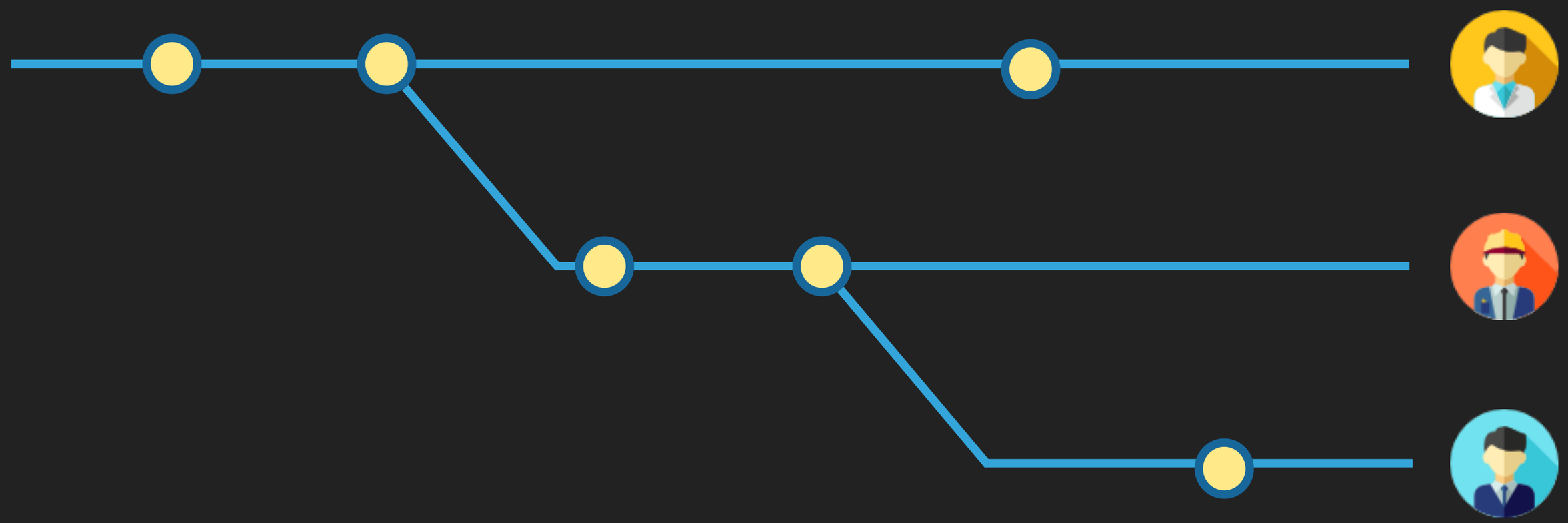
WHAT IS GIT?

- Branch is a sequence of commit that git need to follow in order to reconstruct your project.
  - There could be multiple branches inside git repository



WHAT IS GIT?

- Branch is a sequence of commit that git need to follow in order to reconstruct your project.
  - You can think of it as a way you must follow to get a different job.



Why do we need Git anyway?

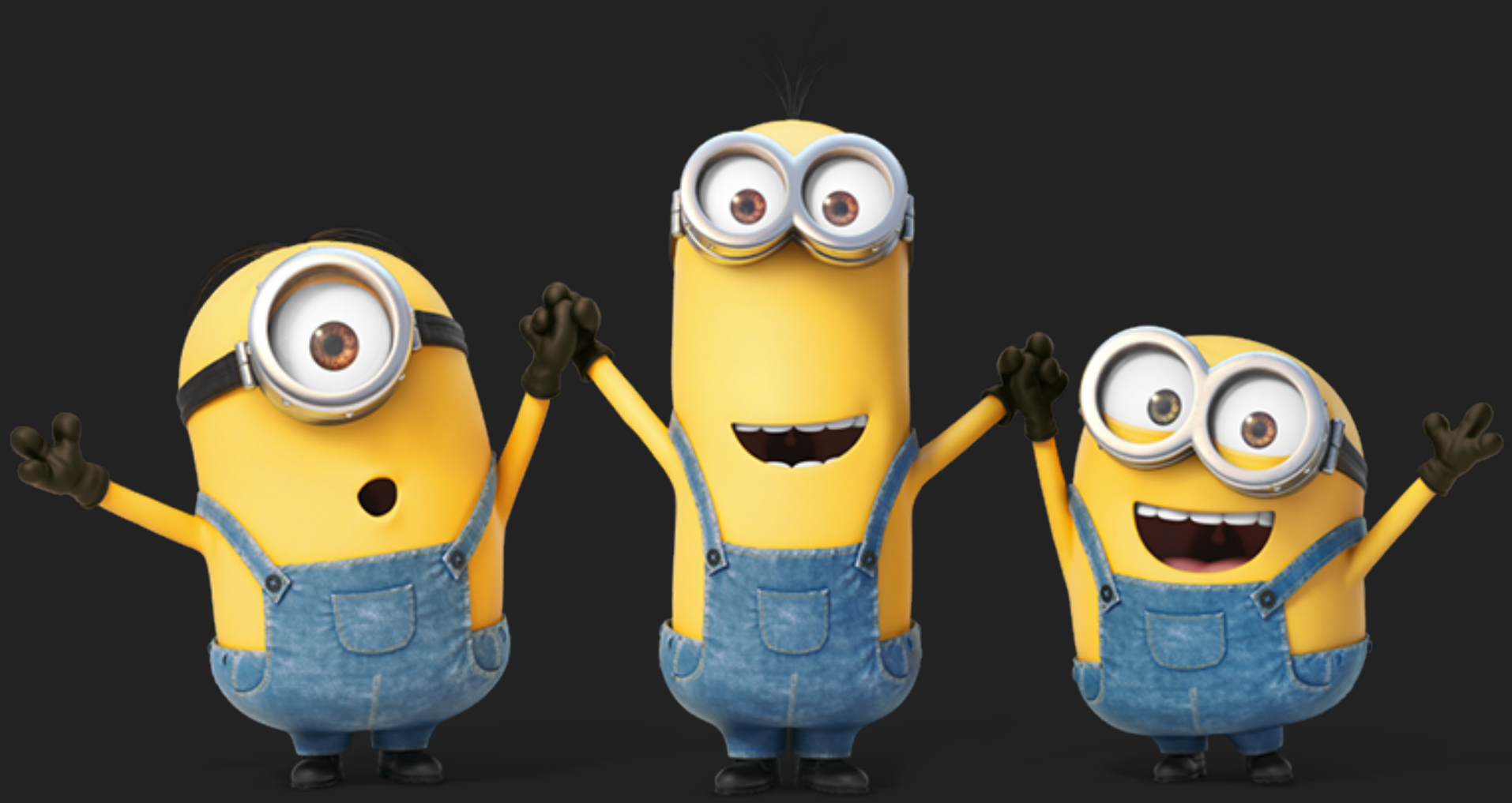
Have you ever done this before?

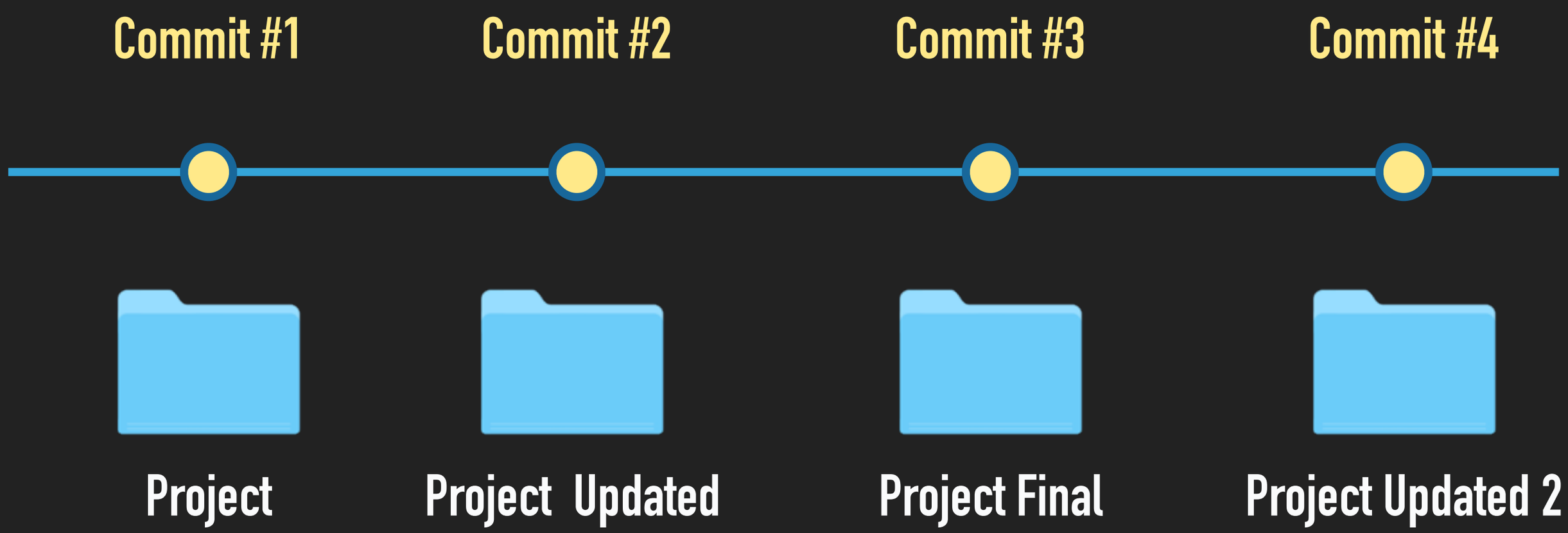
Project | Project Updated | Project Final | Project Updated 2

Which one is the latest ?

How are they different from each other ?

Git come into your rescue




Commit #1
Commit #2
Commit #3
Commit #4
Project
Project Updated
Project Final
Project Updated 2

GIT WORKFLOW

GIT WORKFLOW

- Assume that you have an existing Java project which consist of two classes - A and B
- You want to start using version control in this project so you create a new git repository

A

B

GIT WORKFLOW

- In a git repository, there are three main distinct areas that you need to know about.

1. Working Directory - the root directory of the project on your computer
2. Staging Index - where you place all changes in files that are going to be committed
3. Repository - where git store all the commits

Repository

This is where git store each commit of your project

Working Directory

The root directory of the project on your computer

Staging Index

The selected changes in files that are going to be committed

Repository

Working Directory

A

B

Staging Index

Repository

Working Directory

A

B

Staging Index

Repository

Working Directory

A

B

Staging Index

GIT WORKFLOW

- Next, Let's assume that you make some changes to Class B by adding a new method named "hello"

YOU

Add "hello" method

B

Repository

Working Directory

A

B

Staging Index

Repository

Working Directory

A

B

Staging Index

Repository

Working Directory

A

B

Staging Index

GIT WORKFLOW

- Now, you would be able to summarize the git workflow into three simple steps:

1. Add a new file or modify an existing file
2. Move the changes to the Staging Index
3. Commit the changes to the repository

GIT WORKFLOW

- Okay then, how about this scenario
- Assume that you add new method “defaultGreet” to Class A

YOU

Add “defaultGreet” method

A

Repository

Working Directory

A

B

Staging Index

GIT WORKFLOW

- However, right before you commit, you decide to modify class A again by adding a new method "greetByName"

YOU

Add "greetByName" method

A

Repository

Working Directory

A

B

Staging Index


Repository
Working Directory
Staging Index
A
B
What would happen
if you commit right now?

Repository

Working Directory

A

B

Staging Index

Repository

Working Directory

A

B

Staging Index

Repository

Working Directory

A

B

Staging Index

GIT WORKFLOW

- So now you have all these changes saved as commits in your repository
- How do you access these commits?

GIT WORKFLOW

- When each commit is made, git creates an ID for it
- The ID for the commit is its SHA (Secure Hash Algorithm)

Repository

6a1fd02

2631df1

b1f87e2

Repository

6a1fd02

2631df1

b1f87e2

Working Directory

A

B

Staging Index

WHAT IS GITHUB?

WHAT IS GITHUB?

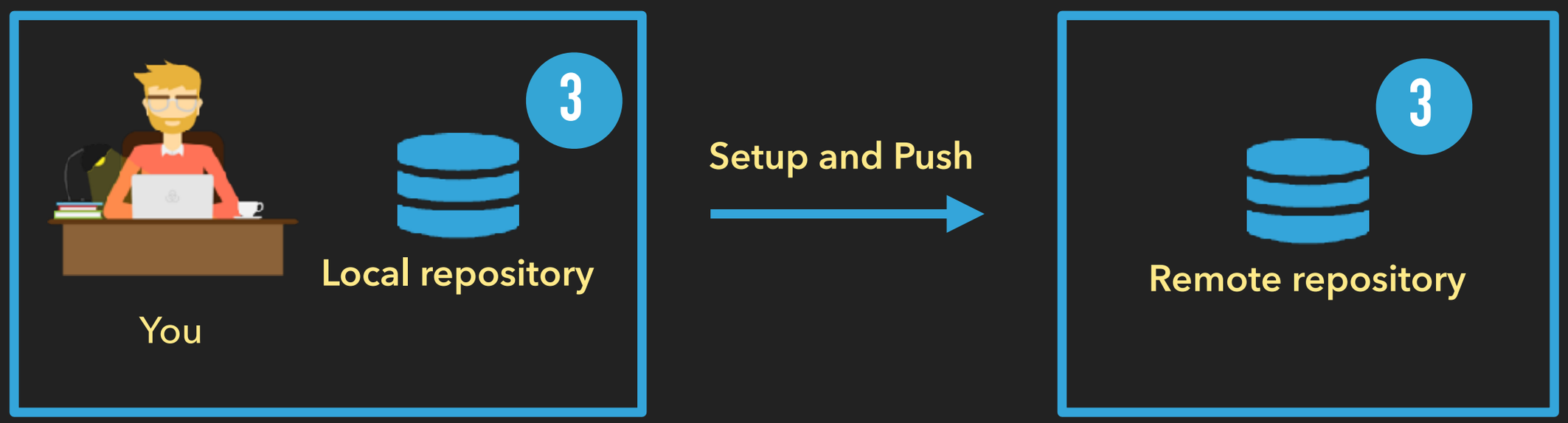
- In addition to managing the repository on your local computer, Git also provide abilities to:
  - Setup a remote repository
  - Push commits from a local repository to a remote repository
  - Pull commits from a remote repository to a local repository
  - Create new repository by cloning from an existing repository

WHAT IS GITHUB?

- These 4 features allow developers to collaborate on the same project easier by using the remote repository as a medium
  - Whenever they create a new commit, they also push it to the remote repository
  - Then, the others pull a new commit from the remote repository to their local repository
- Let's take a quick look on how the process might look like

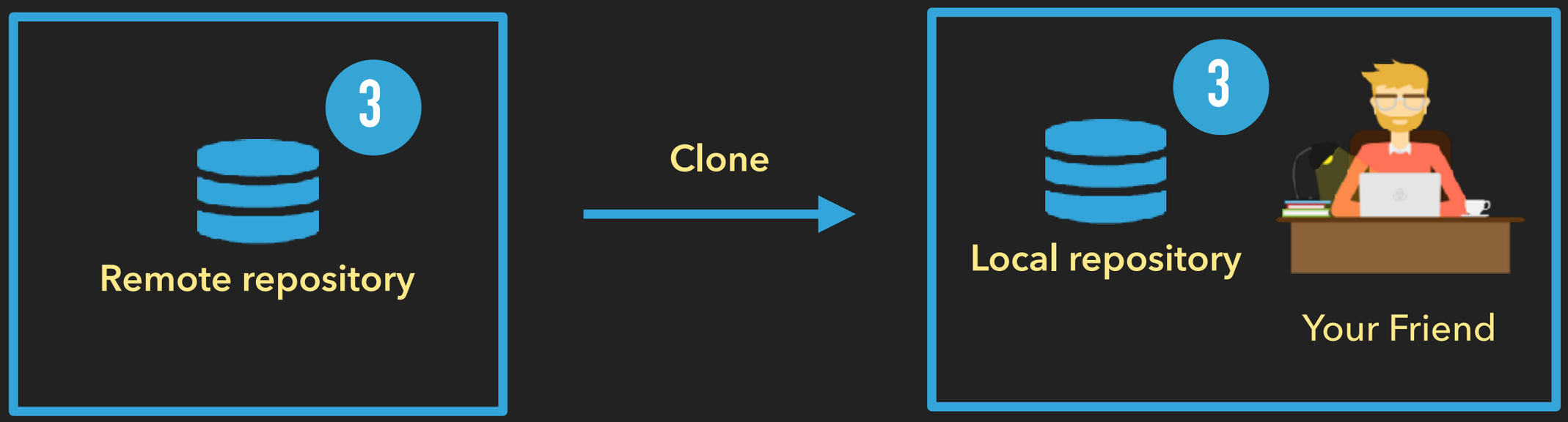
WHAT IS GITHUB?

- Assume that you have an existing repository with three commits
- First, you setup an remote repository and push the commits from your local repository to it so they are synced



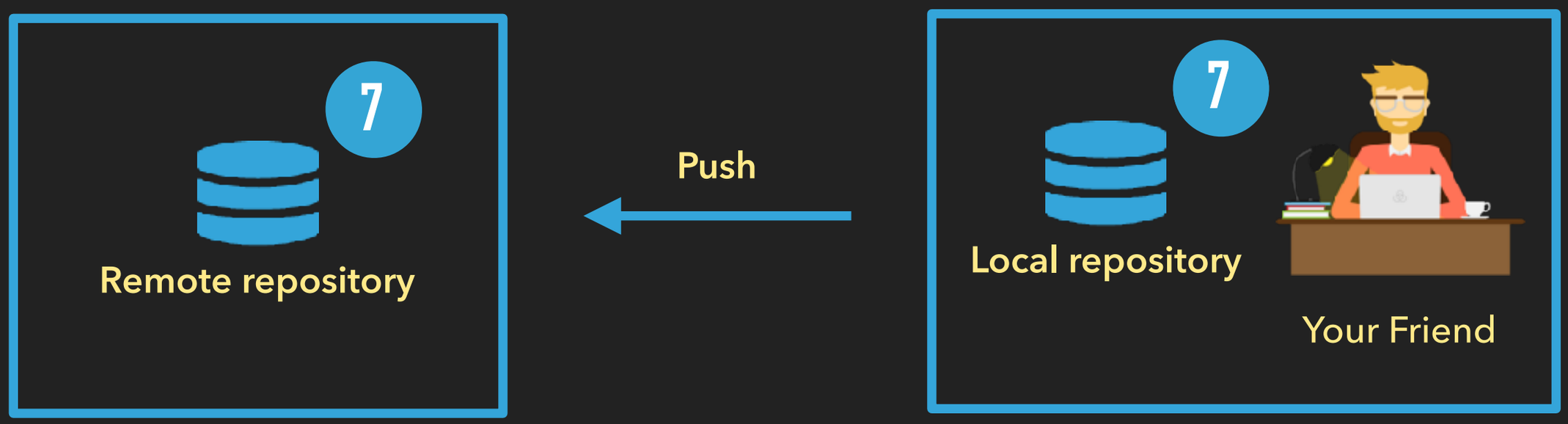
WHAT IS GITHUB?

- Then, your friend also want to help work on the same project
- The first thing he has to do is creating a local repository by cloning from the remote repository



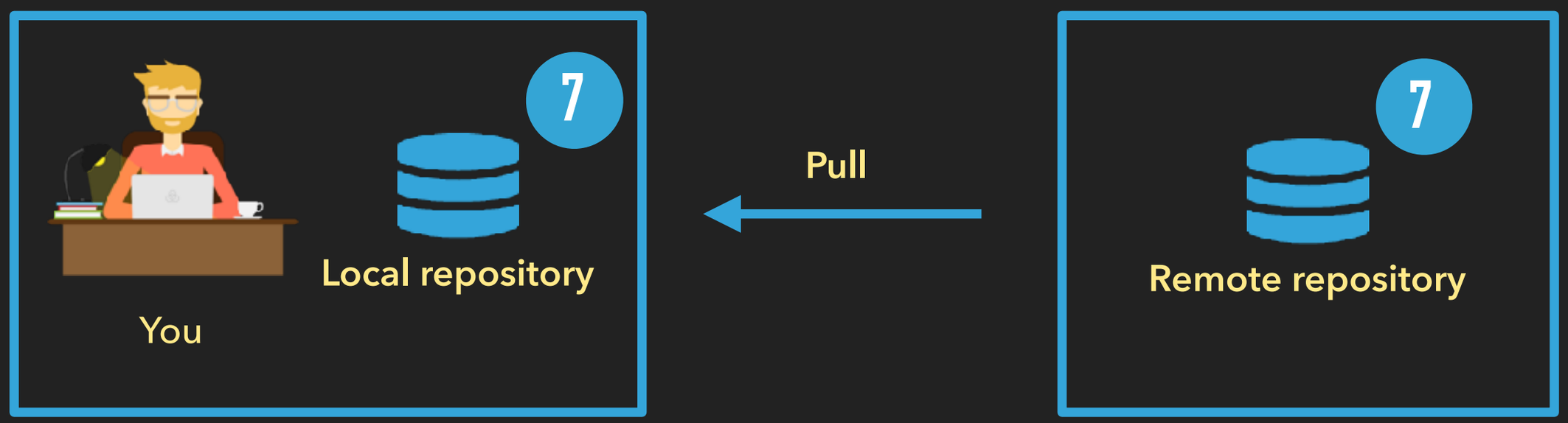
WHAT IS GITHUB?

- Later, your friend create 4 new commits, now, there are 7 commits in his local repository
- To keep the remote repository synced, he push the new commits from his local repository to the remote repository



WHAT IS GITHUB?

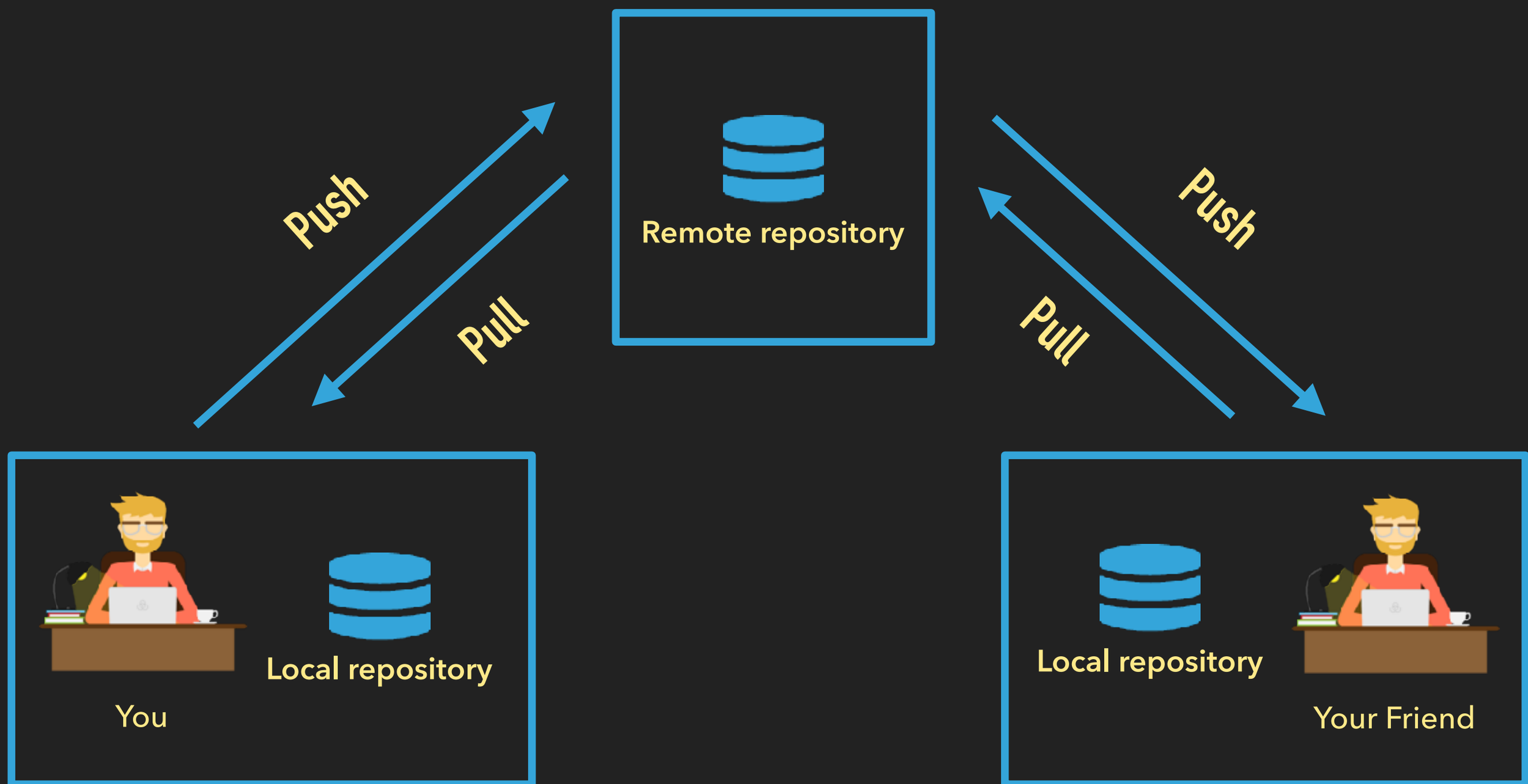
- Finally, you notice that there are 4 new commits on the remote repository
- So you pull these new commits from the remote repository to your local repository



WHAT IS GITHUB?

- In conclusion, you and your friend just have to keep push and pull the commits to ensure that both of you always work on the same version of the project as much as possible

WHAT IS GITHUB?



WHAT IS GITHUB?

- Okay, then, what is GitHub?
- GitHub is just a service that let you host your remote repository on their server
  - You can think of it like Google Drive or DropBox
- It also provides a few features to help you collaborate easier with any persons (e.g., fork, issues and pull requests)

GIT TUTORIAL

GIT TUTORIAL

- This tutorial will guide you through the basic git shell commands for working with Git
- There are 5 topics covered in this tutorial

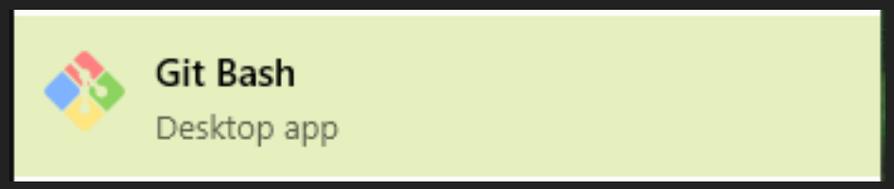
1. Create a git repository
2. .gitignore
3. Add Commits to a repository
4. Setup a remote repository
5. Review a repository history

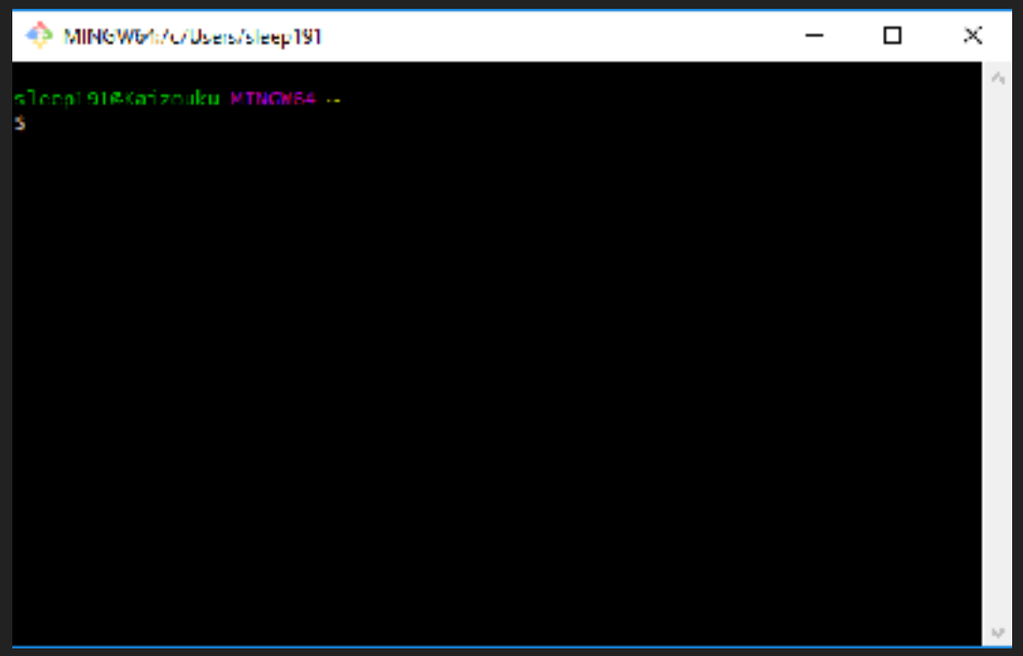
GIT TUTORIAL

- Before going through the tutorial, please ensure that you have Git installed on your computer and you have a GitHub account
  - For Windows user, you have to install Git Bash https://git-scm.com/downloads
- If you don't have Sublime Text or Visual Studio Code on your computer, please also install either one.
  - https://www.sublimetext.com/3
  - https://code.visualstudio.com/

GIT TUTORIAL

- In the next slides, run each of the displaying commands on your command line interface to make sure everything is set up.
- For Windows user, open Git Bash that installed previously





GIT TUTORIAL

```
# sets up Git with your name
git config --global user.name "<Your-Full-Name>"

# sets up Git with your email
git config --global user.email "<your-email-address>"

# makes sure that Git output is colored
git config --global color.ui auto

# displays the original state in a conflict
git config --global merge.conflictstyle diff3

git config --list
```

GIT TUTORIAL

Sublime Text (Windows user)

```
# if you use Sublime Text 2
git config --global core.editor "'C:\Program Files\Sublime Text 2\
sublime_text.exe' -n -w"
```

or

```
# if you use Sublime Text 3
git config --global core.editor "'C:\Program Files\Sublime Text 3\
sublime_text.exe' -n -w"
```

GIT TUTORIAL

Sublime Text (Mac/Linux user)

```
# if you use Sublime Text 2
git config --global core.editor "'/Applications/Sublime Text 2.app/
Contents/SharedSupport/bin/subl' -n -w"
```

or

```
# if you use Sublime Text 3
git config --global core.editor "'/Applications/Sublime Text.app/
Contents/SharedSupport/bin/subl' -n -w"
```

GIT TUTORIAL

Visual Studio Code (Windows user)

```
# if you use Visual Studio Code
git config --global core.editor "'C:\Program Files\Microsoft VS Code\Code.exe' -n -w"
```

Visual Studio Code (Mac/Linux user)

```
# if you use Visual Studio Code
git config --global core.editor "'/Applications/Visual Studio Code.app/Contents/Resources/app/bin/code' --wait"
```

GIT TUTORIAL

- In this tutorial, you are going to create a Java Project named "Git_Tutorial" and initialise a git repository for this project
- Then, implement a Rectangle class inside package "rect"
  - Two private integer fields - width and height
  - Getter methods for both fields
  - One constructor which receive width and height as parameters
    - if the given width or height is less than one, set it to one
  - "getArea" method which return area of the rectangle

GIT TUTORIAL

- While implementing Rectangle class, you are also going to test it with a provided JUnit test case “TestRectangle”
- Finally, after passing the test case, set up a remote repository and push all commits from your local repository to the remote repository.

CREATE A GIT REPOSITORY

CREATE A GIT REPOSITORY

- git init
  - create a brand new repository from the scratch
- git clone
  - Copy existing repos from somewhere else to your local computer
- git status
  - Check the status of the repo

git init

git clone

git status

CREATE A GIT REPOSITORY

- First, you are going to create a brand new repository using git init
- Open Eclipse and create a new Java project "Git_Tutorial"
- Open your terminal then go to the project directory
  - For Windows user, open Git Bash
  - Use *cd* command to change directory

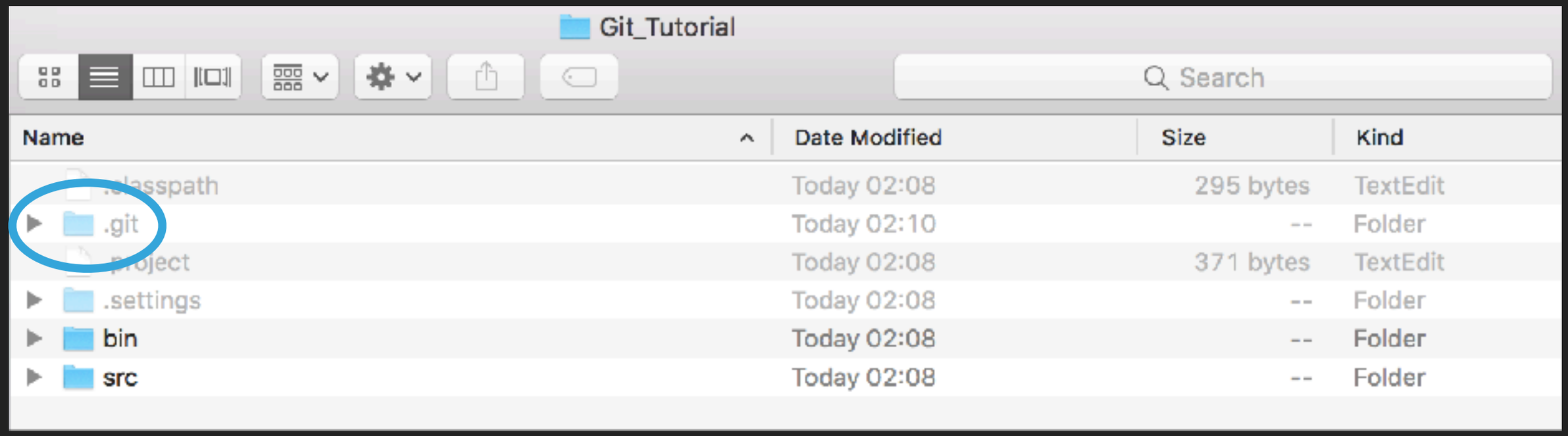
CREATE A GIT REPOSITORY

- At the project directory, run git init command
- If you execute the command successfully, git will inform you that an empty Git repository is initialized

```
~/Documents/2110215/2018-1/Git_Tutorial  git init
Initialized empty Git repository in /Users/Pakapon/Documents/2110215/2018-1/Git_
Tutorial/.git/
~/Documents/2110215/2018-1/Git_Tutorial  master
```

CREATE A GIT REPOSITORY

- Take a look at the project directory again, you will see .git directory which Git created when you run git init command
  - Note that .git directory is a hidden folder



CREATE A GIT REPOSITORY

- .git directory contains all the necessary files and directories for Git to keep track of everything in your project directory
- All of the commits are also stored in .git directory
- So .git directory is actually a repository

CREATE A GIT REPOSITORY

- Next command is git clone
- To use this command you have to pass the path to the Git repository that you want to clone

```
# Clone existing repository
git clone “<path-to-repository>”
```

- In this tutorial, you are going to clone the complete version of “Git_Tutorial” repository from GitHub
- Path to repository: https://github.com/2110215-ProgMeth/Git_Tutorial_Complete.git

CREATE A GIT REPOSITORY

- Open your terminal then go to workspace directory of Eclipse
- Run git clone https://github.com/2110215-ProgMeth/Git_Tutorial_Complete.git

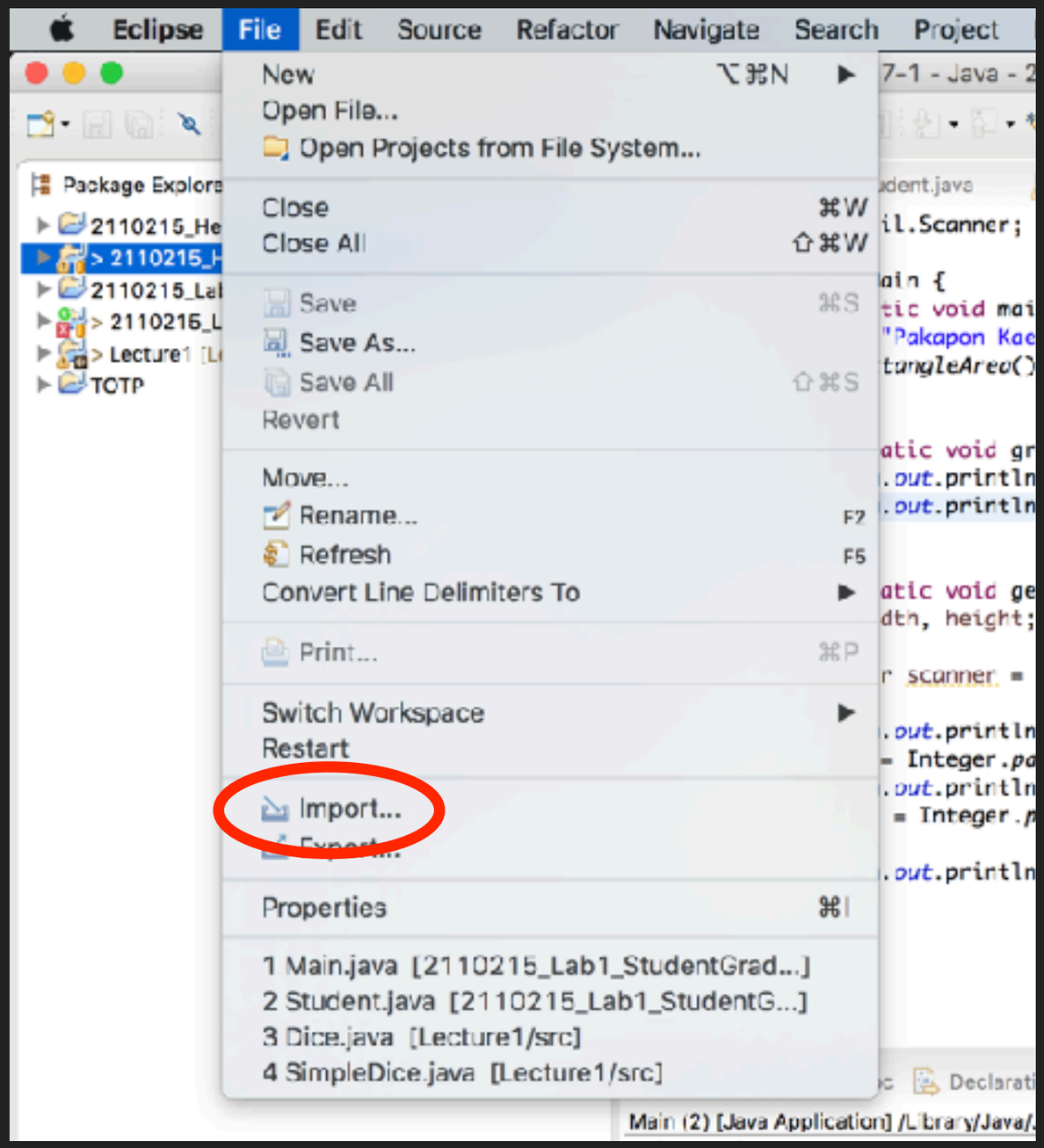
```
~/Documents/2110215/2018-1 git clone https://github.com/2110215-ProgMeth/Git_
Tutorial_Complete.git

Cloning into 'Git_Tutorial_Complete'...
remote: Counting objects: 53, done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 53 (delta 17), reused 53 (delta 17), pack-reused 0
Unpacking objects: 100% (53/53), done.
~/Documents/2110215/2018-1
```

CREATE A GIT REPOSITORY

- When Git complete the cloning process, import the project to Eclipse

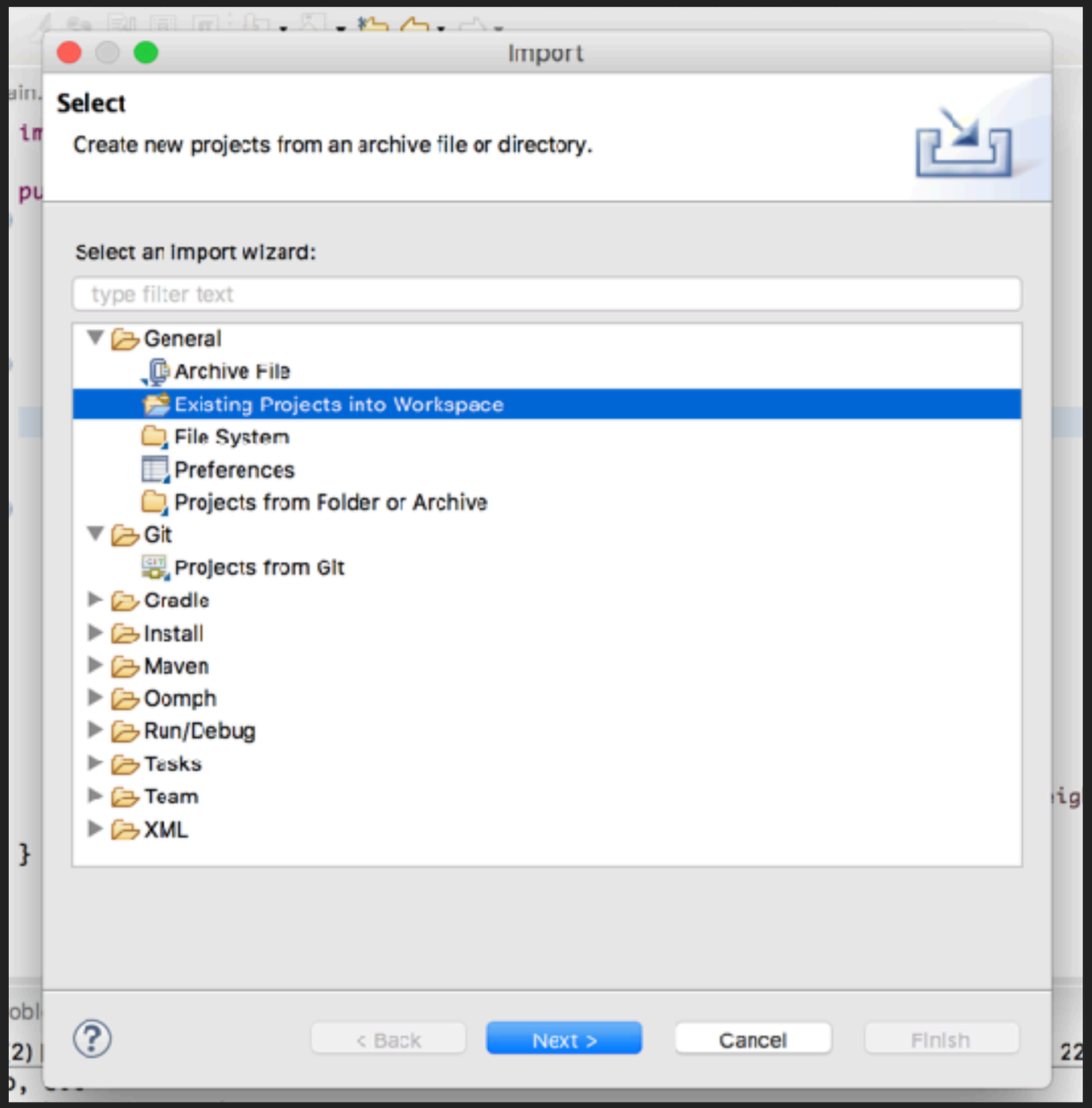
1. Menu File -> Import ....
2. Choose General -> Existing Projects into workspace then click Next
3. Click Browse... then select the project directory (the one that you just cloned)
4. Click Finish

CREATE A GIT REPOSITORY



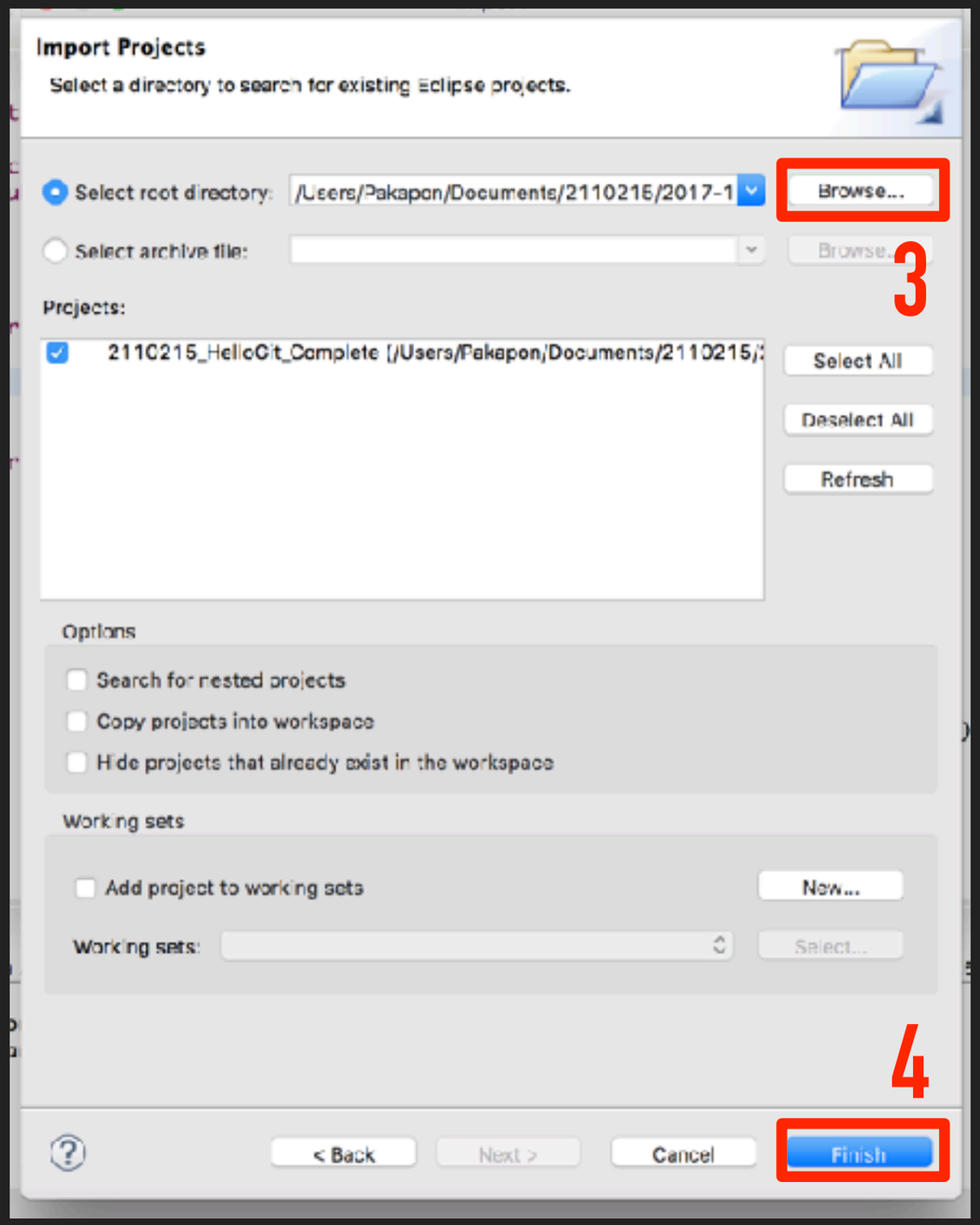
1. Menu File -> Import ....

CREATE A GIT REPOSITORY



2. Choose General -> Existing Projects into workspace then click Next

CREATE A GIT REPOSITORY



3. Click Browse… then select the project directory

4. Click Finish

CREATE A GIT REPOSITORY

- Finally the last command in this section: git status
- This is one of the most common uses command when you interact with Git
- It provides useful information about the current status of your repository
  - List of new files or modified file in repository
  - Changes that in Staging Index and ready to be committed

CREATE A GIT REPOSITORY

- Let's try running **git status** on Git_Tutorial repository (the one that you initialized using **git init**)

```
~/Documents/2110215/2018-1/Git_Tutorial  master  git status
On branch master

No commits yet

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

        .classpath
        .project
        .settings/

nothing added to commit but untracked files present (use "git add" to track)
~/Documents/2110215/2018-1/Git_Tutorial  master
```

This area tells us that this is an empty repository

This area tells us about the new files in repository which haven't been committed to repository yet
It even tells a command for moving this changes to Staging Index

CREATE A GIT REPOSITORY

- As you going through this tutorial, you will learn more about the information that git status could provide

END OF TOPIC!

LET'S GO TO NEXT ONE!

.GITIGNORE

.GITIGNORE

- Let's take a look at the result of git status again
- Git found new files, which Eclipse auto-generated when you initialize the project, that haven't been committed yet

```
~/Documents/2110215/2018-1/Git_Tutorial  master  git status
On branch master

No commits yet

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

        .classpath
        .project
        .settings/

nothing added to commit but untracked files present (use "git add" to track)
~/Documents/2110215/2018-1/Git_Tutorial  master
```

.GITIGNORE

- Let's take a look at the result of git status again
- Git found new files, which Eclipse auto-generated when you initialize the project, that haven't been committed yet

```
~/Documents/2110215/2018-1/Git_Tutorial  master  git status
On branch master

No commits yet

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

        .classpath
        .project
        .settings/

nothing added to commit but untracked files present (use "git add" to track)
~/Documents/2110215/2018-1/Git_Tutorial  master
```

Should we commit these files?

.GITIGNORE

- There are some kind of files that should NOT be committed to the repository
  - Mostly, They are files that generated by tool or command and you can not distinguish the different between each of them. e.g., .class and JAR files
- Especially, when you are working with an IDE, there are a lot of files that could be auto-generated behind the scene without you notice

.GITIGNORE

- So we create .gitignore file, place it at the root of project directory to tell Git that the files listed on .gitignore do not need to be tracked
- But how can we knows which files should be listed inside .gitignore

.GITIGNORE

- Fortunately, GitHub already created a repository collecting all .gitignore templates for most projects

https://github.com/github/gitignore

- You just have to select the right one for your projects
- In this case, we need .gitignore for Eclipse project
  - Here is the link: https://github.com/github/gitignore/blob/master/Global/Eclipse.gitignore

.GITIGNORE

- Let's add this .gitignore at the root of project directory
- Then run git status again

```
~/Documents/2110215/2018-1/Git_Tutorial  master  git status
On branch master

No commits yet

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

        .classpath
        .gitignore
        .project

nothing added to commit but untracked files present (use "git add" to track)
~/Documents/2110215/2018-1/Git_Tutorial  master
```

.GITIGNORE

- Good! now you don't have to worry about accidentally committing something you should not

END OF TOPIC!

LET'S GO TO NEXT ONE!

ADD COMMITS TO A REPOSITORY

ADD COMMITS TO A REPOSITORY

- git add
  - Add changes in files from the working directory to the staging index
- git commit
  - Take changes in files from the staging index and save them in the repository

git add

git commit

ADD COMMITS TO A REPOSITORY

- Before start implementing the project, Let's make two commits
- One for .gitignore and another for .classpath and .project

```
~/Documents/2110215/2018-1/Git_Tutorial  master  git status
On branch master

No commits yet

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

        .classpath
        .gitignore
        .project

nothing added to commit but untracked files present (use "git add" to track)
~/Documents/2110215/2018-1/Git_Tutorial  master
```

ADD COMMITS TO A REPOSITORY

- Let's start with .gitignore
- First you have to move .gitignore to the staging index using git add command

```
# Move files from the working directory to the staging index
git add <file1> <file2> … <fileN>
```

- Run git add .gitignore

ADD COMMITS TO A REPOSITORY

- Run git status and you will see that .gitignore is moved to the staging index ready to be committed
- Now you just have to create a commit

```
~/Documents/2110215/2018-1/Git_Tutorial  master +  git status
On branch master

No commits yet

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

        new file:   .gitignore

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

        .classpath
        .project

~/Documents/2110215/2018-1/Git_Tutorial  master +
```

This area represents the staging index where all the files that are going to be committed is located

ADD COMMITS TO A REPOSITORY

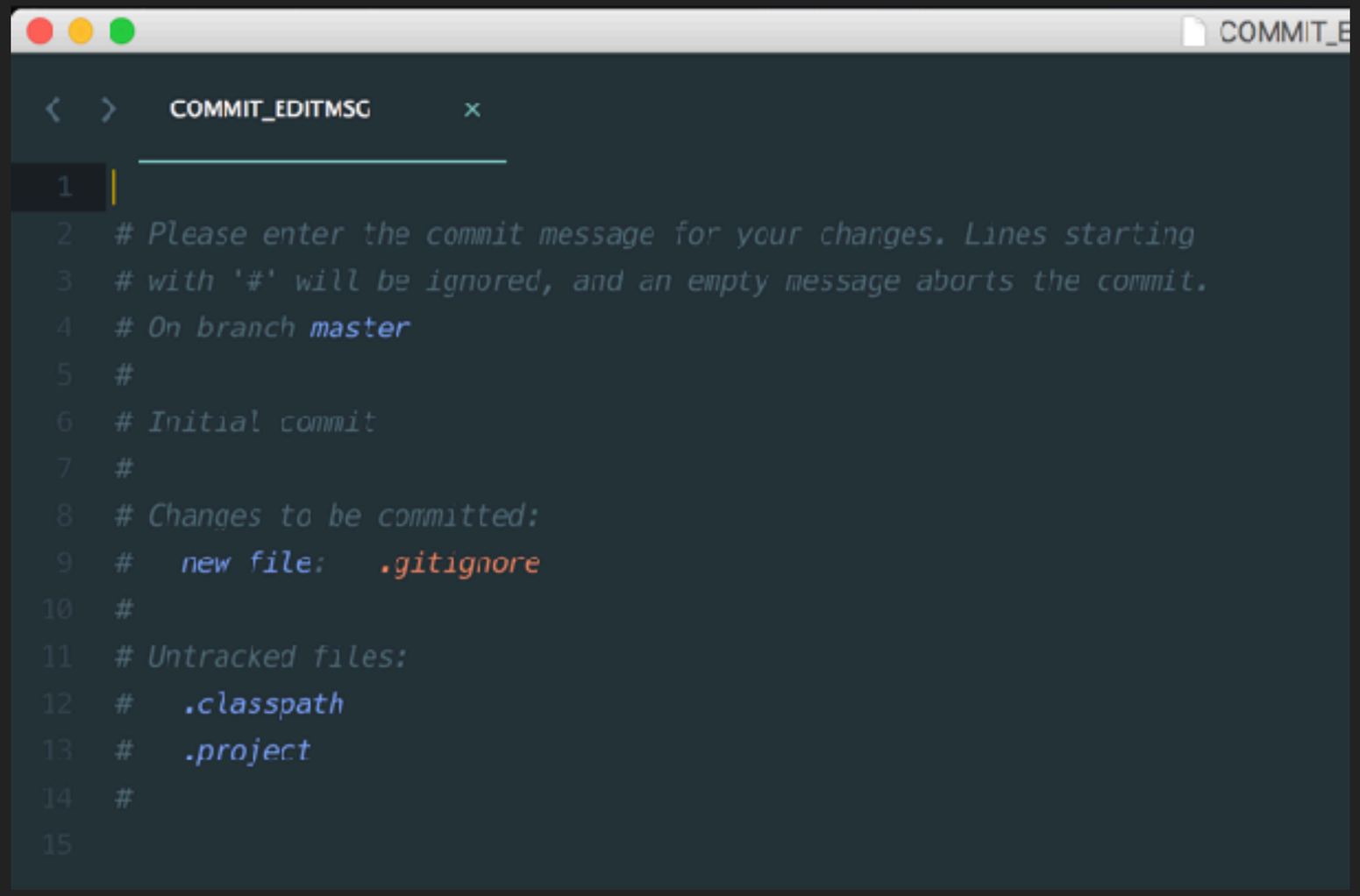
- Every commit need a message described what changes in that commit.
- The message should be easy to understand by anyone
- Here is the recommended structure of Git commit message

```
Summarize changes in around 50 characters or less

More detailed explanatory text, if necessary.
```

- If you want to know more about how to write a Git commit message
  - Read this blog: https://chris.beams.io/posts/git-commit/

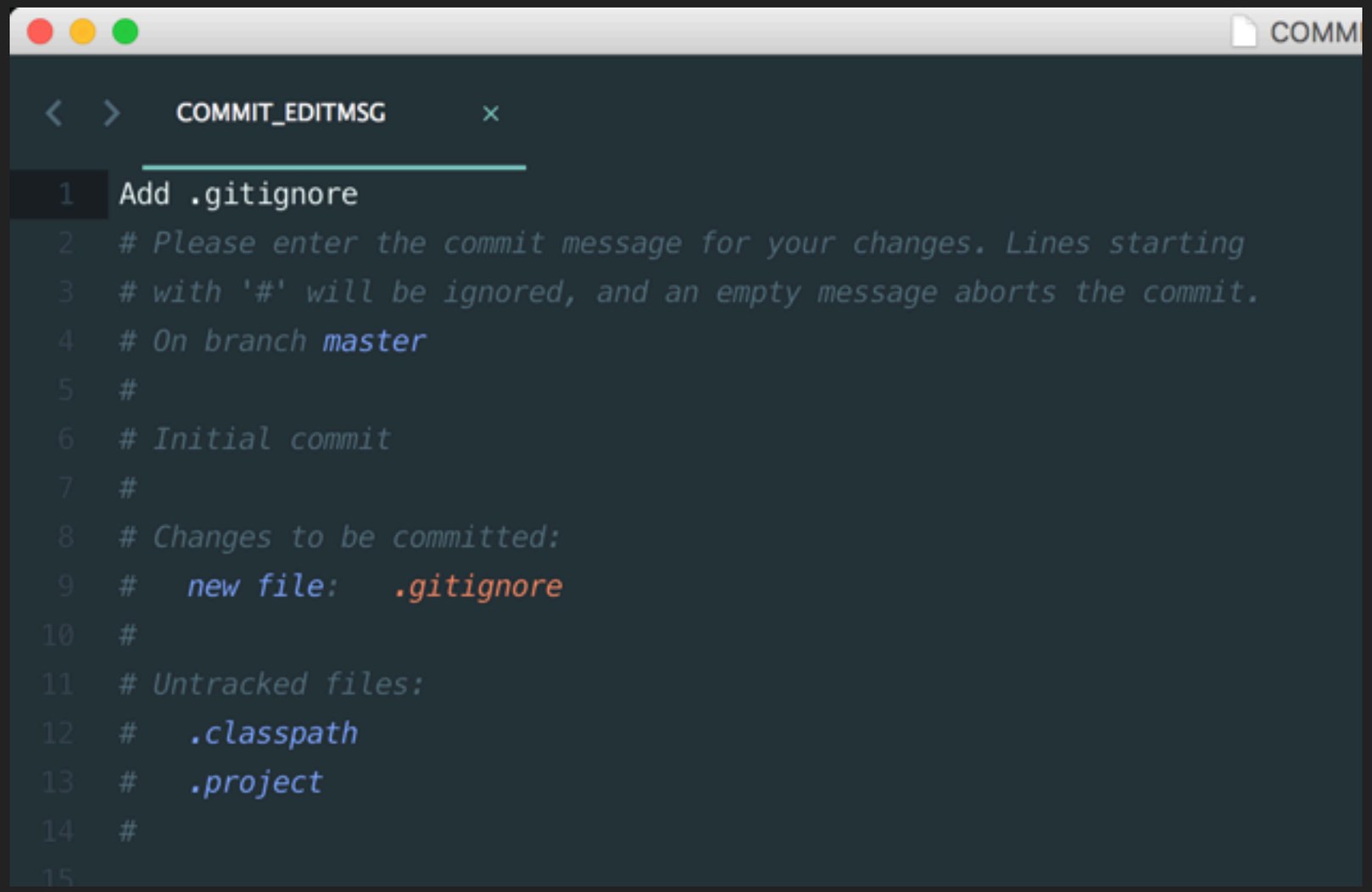
ADD COMMITS TO A REPOSITORY

- Run git commit and then Sublime Text installed on your computer should be opened



ADD COMMITS TO A REPOSITORY

- Write a Git commit message (How about "Add .gitignore")



ADD COMMITS TO A REPOSITORY

- Save then close the tab on Sublime Text.
- Back in your terminal, you will see that the commit is successfully created

```
✘ ~/Documents/2110215/2017-1/2110215_HelloGit  master +  git commit
[master (root-commit) 07657ca] Add .gitignore
 1 file changed, 49 insertions(+)
 create mode 100644 .gitignore
```

- Next, let's create a commit for .classpath and .project

ADD COMMITS TO A REPOSITORY

- Run git add .classpath .project
  - You can also run git add . - this will all new or modified files to the staging index
- Create new commit using git commit command
- How about using "Add Eclipse project files" as a Git commit message for this time

```
~/Documents/2110215/2017-1/2110215_HelloGit  master +  git commit
[master 1145aff] Add Eclipse project files
 2 files changed, 23 insertions(+)
 create mode 100644 .classpath
 create mode 100644 .project
```

ADD COMMITS TO A REPOSITORY

- Finally, before you start coding, let's add "TestRectangle", a JUnit test case for Rectangle, into this project
  - Copy the test case from "Git_Tutorial_Complete" project
  - Create a package "test.grader" inside "Git_Tutorial" project then add the test case to it
- Let's create a new commit for this change

ADD COMMITS TO A REPOSITORY

- Now, let's implement our Rectangle class
- When you pass one of the test case, try to create a new commit by yourself

GOOD LUCK

SET UP A REMOTE REPOSITORY

SET UP A REPOSITORY

- git remote add
  - Tell git to recognize a remote repository that you might push commits to or pull commits from in the future
- git push
  - Push new commits from local repository to the remote repository

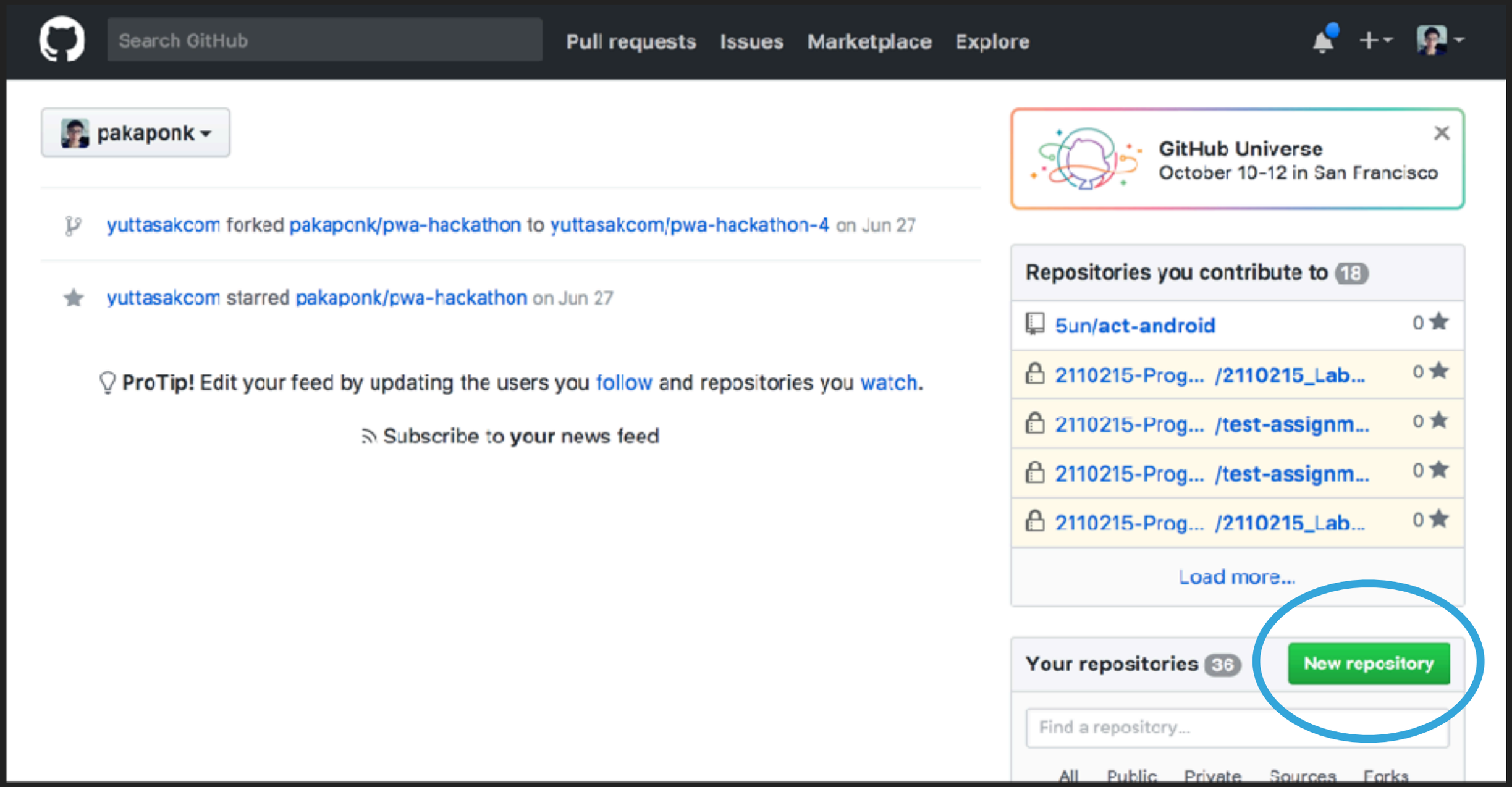
git remote add

git push

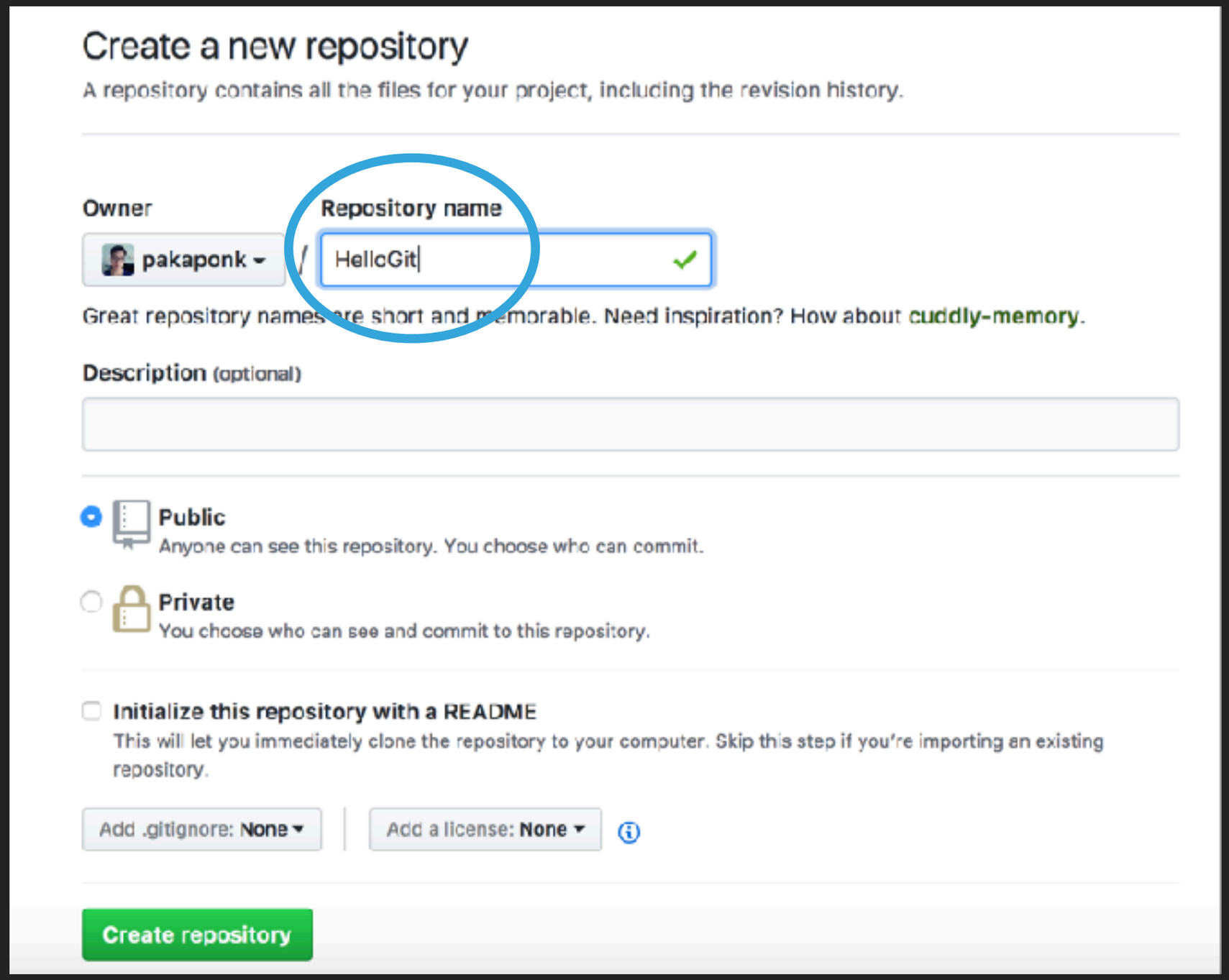
SETUP A REMOTE REPOSITORY

- Okay, Let's setup a remote repository for the project
- In general, you can go to GitHub then use your account to create a new remote repository

SETUP A REMOTE REPOSITORY

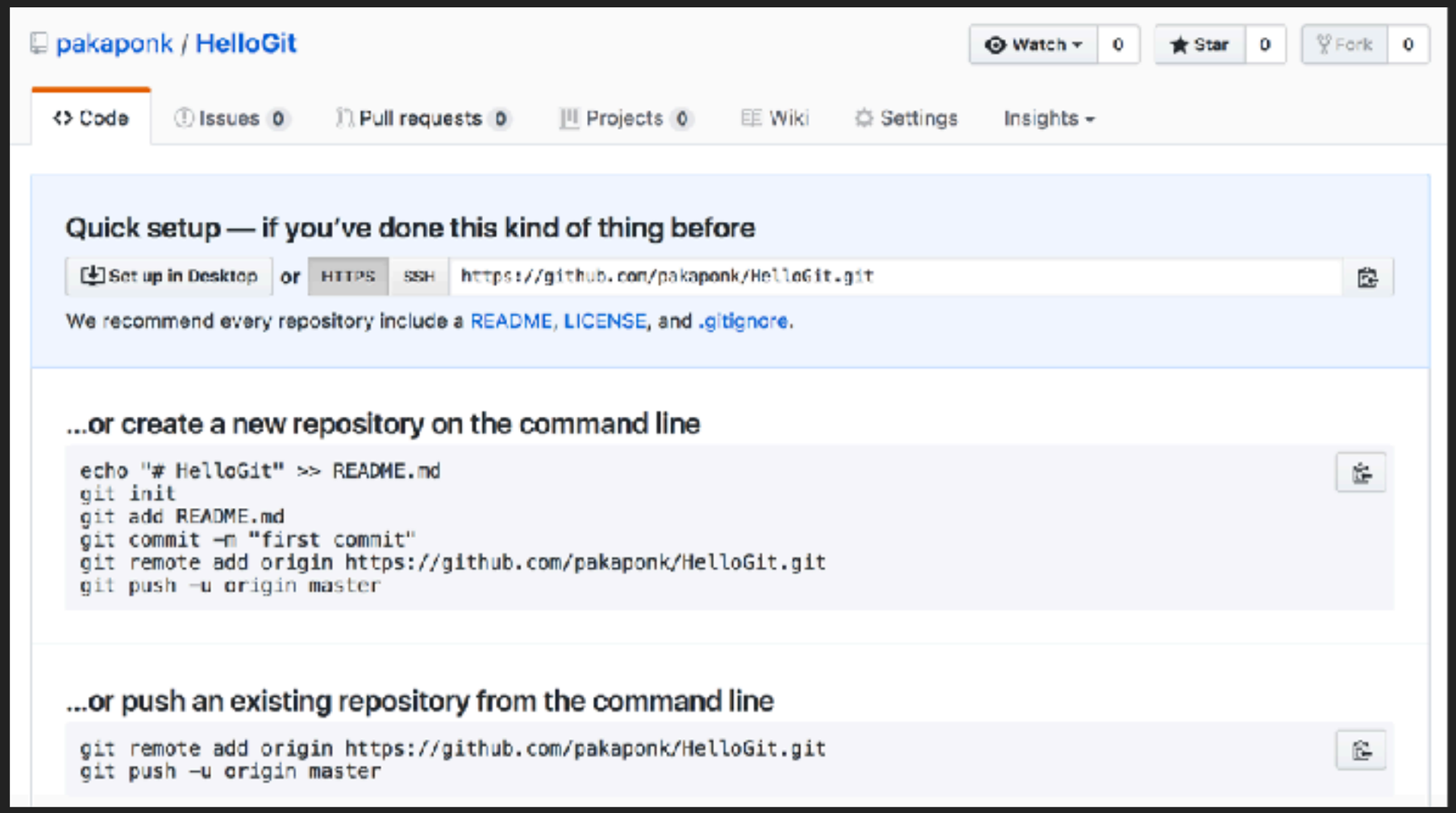


SETUP A REMOTE REPOSITORY



SETUP A REMOTE REPOSITORY

- When you successfully create a new repository on GitHub, you should be able to see the following page



SETUP A REMOTE REPOSITORY

- However, in this class, you are going to create a remote repository by clicking the special link we provided for each lab
- For this tutorial, please use this link:
  - https://classroom.github.com/a/3ixXhXOC

SETUP A REMOTE REPOSITORY

- Copy the command in the circle

- Back to the terminal, paste the command and run it

```
~/Documents/2110215/2017-1/2110215_HelloGit  master  git remote add origin https://github.com/
pakaponk/HelloGit.git
~/Documents/2110215/2017-1/2110215_HelloGit  master  git push -u origin master
Counting objects: 16, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (14/14), done.
Writing objects: 100% (16/16), 2.34 KiB | 0 bytes/s, done.
Total 16 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/pakaponk/HelloGit.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
~/Documents/2110215/2017-1/2110215_HelloGit  master
```

SETUP A REMOTE REPOSITORY

- git remote add is a command for setting up new remote repository which take two arguments

  1. the alias for the remote repository
  2. path to the remote repository

```
# Set up new remote repository
git remote add <name> <path-to-remote-repository>
```

SETUP A REMOTE REPOSITORY

- In this case, when I run:
  - git remote add origin https://github.com/2110215-ProgMeth/git-tutorial-pakaponk
  - I set up new remote repository for my repository called "origin"
- Note that "origin" is a default alias for a remote repository when you create new local repository by cloning it from an existing remote repository

SETUP A REMOTE REPOSITORY

- git push is a command for push new commits to the remote repository
- In this case when I run:
  - git push -u origin master
  - Git is going to push new commits from branch master on the local repository to “origin” (the alias for the remote repository set up earlier)
  - -u is an option which make Git create a link between branch master on the local repository and branch master on the origin
  - So that future push/pull will automatically associate between these two branches

SETUP A REMOTE REPOSITORY

- If you run git status, Git will tell you that both branch master on local repository and remote repository is synced

```
~/Documents/2110215/2017-1/2110215_HelloGit  master  git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean
~/Documents/2110215/2017-1/2110215_HelloGit  master
```

END OF TOPIC!

LET'S GO TO THE LAST ONE!

REVIEW A REPOSITORY HISTORY

REVIEW A REPOSITORY HISTORY

- git log
  - Display information about the existing commits
- git show
  - Displays information about the given commit

git log

git show

REVIEW A REPOSITORY HISTORY

- If you want to view all the existing repository, try git log
- This command will list all the existing commits from the most recent one to the oldest one (Top to bottom)

```
commit 2951d43d111027f05ce2568e51f29de1b4228840
Author: Pakapon Kaewmark <sleep191@gmail.com>
Date:   Wed Aug 23 04:05:33 2017 +0700

    Add getRectangleArea() for calculate an area of specified rectangle

    This method receive width and height of the rectangle from user input

commit d5fcb17cc6af08db4289b9f3409a3d24b1d6212d
Author: Pakapon Kaewmark <sleep191@gmail.com>
Date:   Wed Aug 23 04:01:38 2017 +0700

    Add new class Main with greet() method for display greeting message

commit 1145affdfd11c8b0f9a363d4ef2b86c94d3be4b2
Author: Pakapon Kaewmark <sleep191@gmail.com>
Date:   Wed Aug 23 03:11:02 2017 +0700

    Add Eclipse project files

commit 07657caf184d673956648b937b5d023949bf4d88
Author: Pakapon Kaewmark <sleep191@gmail.com>
Date:   Wed Aug 23 02:19:48 2017 +0700

    Add .gitignore
(END)
```

SETUP A REMOTE REPOSITORY

- Each commit provides information about:
  1. its SHA
  2. the person who created this commit
  3. the date when this commit is created
  4. its commit message

```
commit 2951d43d111027f05ce2568e51f29de1b4228840
Author: Pakapon Kaewmark <sleep191@gmail.com>
Date:   Wed Aug 23 04:05:33 2017 +0700

    Add getRectangleArea() for calculate an area of specified rectangle

    This method receive width and height of the rectangle from user input
```

REVIEW A REPOSITORY HISTORY

- If you want to see the detail of the specific commit, try git show
- To use this command, you have to know the SHA of the commit

```
# Display information about specified commit
git show <commit>
```

- For example: git show 2951d43d111027f05ce2568e51f29de1b4228840

REVIEW A REPOSITORY HISTORY

- In addition to the information that you can view from git log command
- You can also view the changes made in that commit

```
commit 2951d43d111027f05ce2568e51f29de1b4228840
Author: Pakapon Kaewmark <sleep191@gmail.com>
Date:   Wed Aug 23 04:05:33 2017 +0700

    Add getRectangleArea() for calculate an area of specified rectangle

    This method receive width and height of the rectangle from user input

diff --git a/src/Main.java b/src/Main.java
index 8cc53da..6f31836 100644
--- a/src/Main.java
+++ b/src/Main.java
@@ -1,3 +1,4 @@
+import java.util.Scanner;

 public class Main {
        public static void main(String[] args) {
@@ -8,4 +9,17 @@ public class Main {
                System.out.println("Hello, Git");
                System.out.println("I am " + name );
        }
+
+       private static void getRectangleArea(){
+               int width, height;
+
+               Scanner scanner = new Scanner(System.in);
+
:
```

END!

RESOURCE

- More Tutorial
  - https://try.github.io/
  - https://www.udacity.com/course/version-control-with-git--ud123
- You can try to use Git GUI such as GitHub Desktop or SourceTree
  - https://desktop.github.com/
  - https://www.sourcetreeapp.com/