GIT – VERSION CONTROL SYSTEM

JOHN BRYCE
Leading in IT Education
a matrix company

1. Motivation
2. What is VCS / SCM?
3. What is Git?
4. Terminology: Repository & Commit
5. How Git Works?
6. Basic Git workflow
7. Installing Git
8. Git Bash CLI
9. First-Time Git Setup
10. Getting a Git Repository
11. Recording Changes
12. Committing Changes
13. Useful Git Commands
14. Remote Repositories
15. GitHub & Bitbucket
16. Clone from GitHub

Motivation

Scenario 1:

Say you're working on your software project in class and then you need to continue working on it at home (on a different computer) – **What do you do?**

Scenario 2:

Say you're working on the same project with one partner or more, or you're a software developer in a R&D team – **How do you share your work with others? How do you get updates made by them?**

Scenario 3:

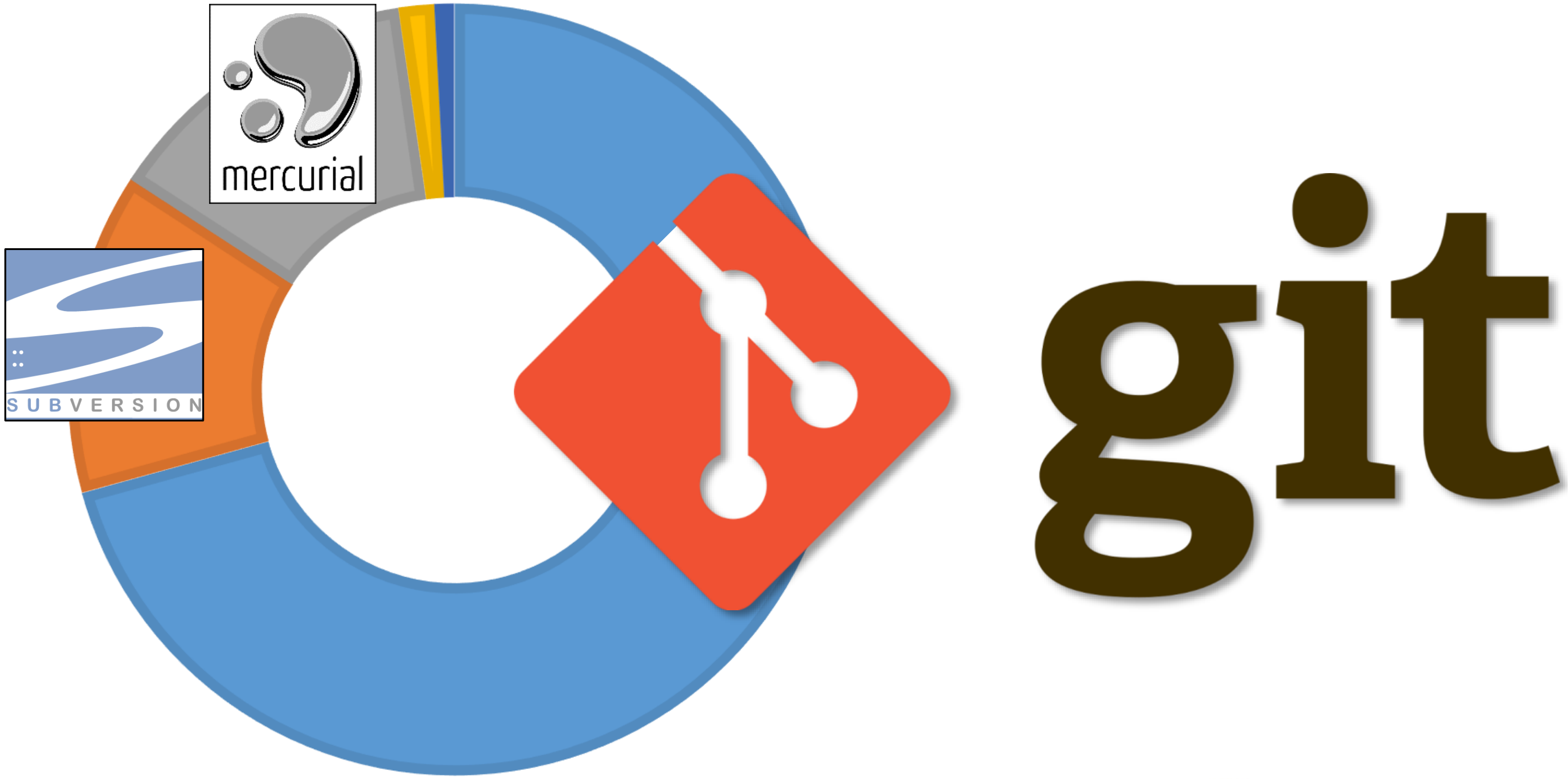
Say you're working on your program, and you discover that you've made a critical mistake and need to revert back and cancel all changes you've done over the past hour / day – **How do your recover your code?**

What is VCS / SCM?

- A **Version Control System - VCS** (also known as **source control management - SCM**) is a system that records changes to a file or set of files over time
- It's especially useful for version controlling of software source code files, but can be used with almost any type of computer files
- A VCS allows you to:
  - Return back to a specific version
  - Revert specific files back to a previous state
  - Revert the entire project back to a previous state
  - Compare changes over time
  - See who last modified something that might be causing a problem
  - See who introduced an issue and when
- If you screw things up or lose files, you can easily recover!

What is Git?

- Git is a free and open source version control system.
- Git was created by Linus Torvalds in 2005 for development of the Linux kernel.
- In 2019, Git is the most popular VCS by a large margin.
  Other popular VCSs: Apache Subversion (SVN), Mercurial



Terminology: Repository & Commit

Repository

In version control systems, a repository is data structure that stores metadata for a set of files or directory structure. The metadata includes a **historical record of changes** in the repository, and **a set of "commit" objects**.

Commit

- An action which adds the latest changes to the source code to the repository, making these changes part of the head revision of the repository.
- Each time we perform a commit, a new commit object is created in the repository.
- Commit objects are kept in the repository indefinitely.
- When other users do an **update** or a **checkout** from the repository, they will receive the latest committed version (unless they specify they wish to retrieve an older version of the source code).

How Git Works?

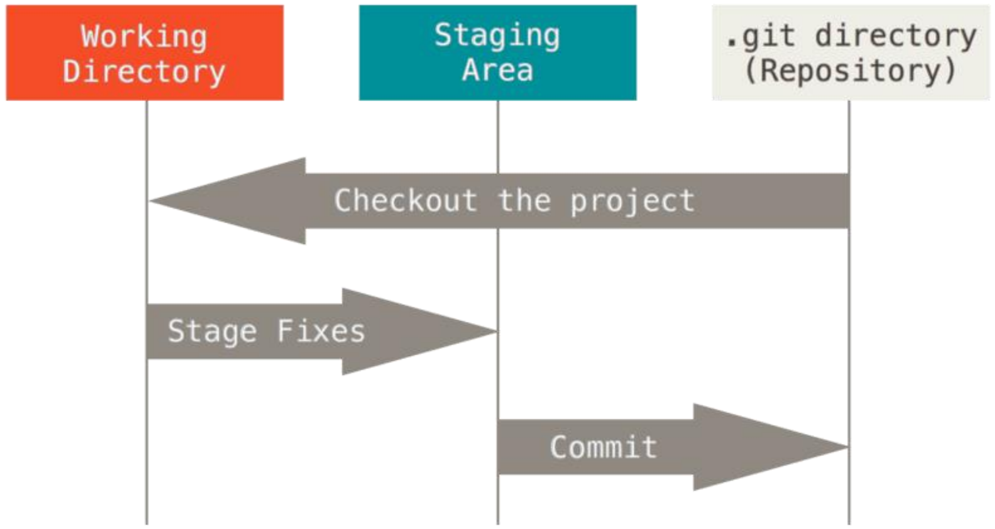
Git thinks of its data like a series of **snapshots** of a miniature file system. Each time we do a commit a new “snapshot” of the current state of our project is created.

In Git, files can be in one of three states:

- **Committed** - the data is safely stored in the local database
- **Modified** - the file was changed but it was not committed to the database yet
- **Staged** - a modified file is marked to go into the next commit (snapshot)

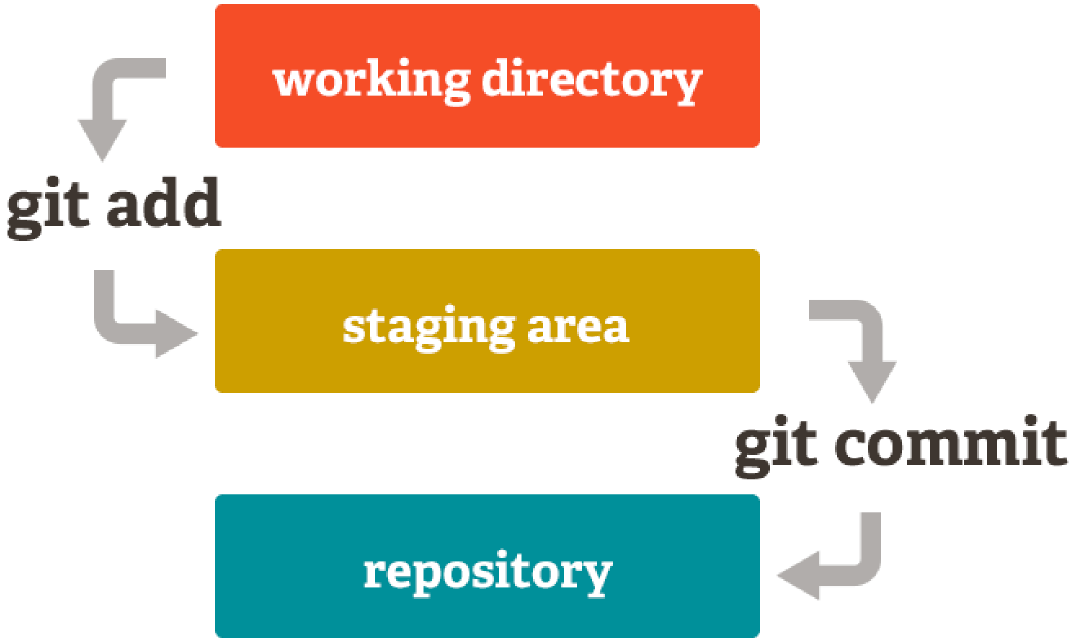
Three main sections of a Git project:

- **Git directory** - where Git stores the metadata and object database for your project
- **The working tree** - a single checkout of one version of the project
- **The staging area** - a file that stores information about what will go into the next commit



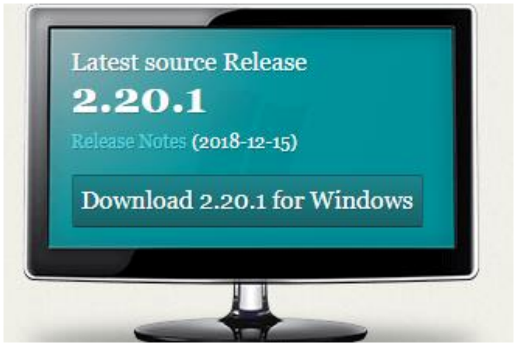
Basic Git workflow

1. Modify files in the working tree
2. Selectively stage just those changes you want to be part of your next commit - adds only those changes to the staging area
3. Do a commit, which takes the files as they are in the staging area and stores that snapshot permanently in the Git directory

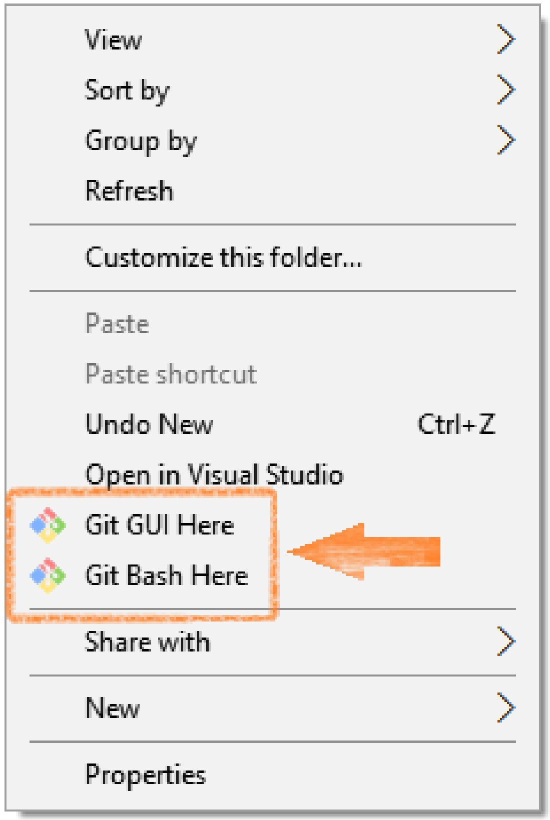


Installing Git

- Git installer can be downloaded from Git's official website: https://git-scm.com/
- After downloading the installer run it and accept all defaults by clicking "Next" throughout the installation steps.



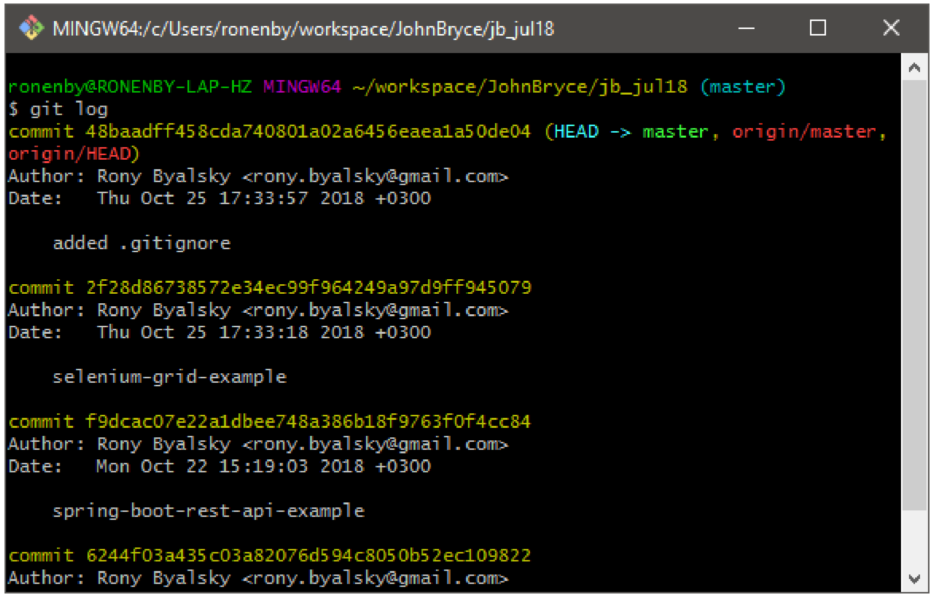
- To verify Git was properly installed on Windows, open any folder in the file explorer and right-click anywhere to open the menu. You should see 2 new entries in the menu: "Git GUI Here" & "Git Bash Here"
- Verify that **Git Bash** opens properly. This is the tool that we'll use the most for the rest of this course.



Git Bash CLI

- There are many ways to use Git: via different GUI utilities or from the command-line.
- It's strongly recommended to learn to use Git from the command line as that is the common method for using Git, regardless of your operating system and IDE of choice.

- If you know how to run the command-line version, you can probably also figure out how to run the GUI version.
- Git for Windows provides a BASH emulation used to run Git from the command line. This powerful tool allows you to run Git commands, as well as many Linux-like command-line utilities which don't exist on Windows natively.



```
ronenby@RONENBY-LAP-HZ MINGW64 ~/workspace/JohnBryce/jb_jul18 (master)
$ git log
commit 48baadff458cda740801a02a6456eaea1a50de04 (HEAD -> master, origin/master,
origin/HEAD)
Author: Rony Byalsky <rony.byalsky@gmail.com>
Date:   Thu Oct 25 17:33:57 2018 +0300

    added .gitignore

commit 2f28d86738572e34ec99f964249a97d9ff945079
Author: Rony Byalsky <rony.byalsky@gmail.com>
Date:   Thu Oct 25 17:33:18 2018 +0300

    selenium-grid-example

commit f9dcac07e22a1dbee748a386b18f9763f0f4cc84
Author: Rony Byalsky <rony.byalsky@gmail.com>
Date:   Mon Oct 22 15:19:03 2018 +0300

    spring-boot-rest-api-example

commit 6244f03a435c03a82076d594c8050b52ec109822
Author: Rony Byalsky <rony.byalsky@gmail.com>
```

Tip: If you're not familiar with the Linux shell (command-line), it's strongly recommended to invest some time to learn the basics.
Linux commands come in handy in numerous software development scenarios.

First-Time Git Setup

NOTE: All following commands and examples are intended to be executed from command-line. On Windows, use Git Bash as explained in the previous slides.

The first thing you should do after installing Git is to set your user name and email address. This is important because every Git commit uses this information.

(This should be done only once on any given computer):

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

If you want to check your configuration settings, use:

```
$ git config --list
```

Getting a Git Repository

It's possible to obtain a Git repository in one of two ways:

1. **Initialize a Repository in an Existing Directory:**
   Open Git Bash in a directory that is currently not under version control and you want to start controlling with Git, and type:

   ```
   $ git init
   ```

   This creates a new subdirectory named .git that contains all of your necessary repository files — a Git repository skeleton. At this point, nothing in your project is tracked yet.

2. **Clone an existing Git repository from elsewhere:**

   ```
   $ git clone <url>
   ```

   Executing this command receives a full copy of nearly all data that the server has. Every version of every file for the history of the project is pulled down.

Recording Changes

Each file in the working directory can be in one of two states:

- **Tracked files** - Files that were in the last snapshot and Git knows about them. Tracked files can be in one of the following states: *unmodified*, *modified*, or *staged*
- **Untracked files** - Any files in your working directory that were not in your last snapshot and are not in your staging area.

As you edit files, Git sees them as modified (changed since the last commit). We need to selectively stage modified files and then commit those changes.

```
$ git status
```

- The main tool to determine which files are in which state

```
$ git add <file(s)>
```

- A multipurpose command: use it to begin tracking new files, or to stage a modified file. Think of it as: “add precisely this content to the next commit”.

Committing Changes

After setting up the staging area the way we want it (by using **git add**), we can commit our changes:

Every time we perform a commit, we're recording a snapshot of our project that we can revert to or compare to later.

```
$ git commit -m "<your_commit_message>"
```

NOTE: The commit message is mandatory. You can't do a commit without specifying a commit message.
Try to provide an informative message, with a description of what has been changed or added as part of this commit.

REMEMBER: Any files you have created or modified that you haven't run **git add** on since you edited them — won't go into this commit.

Useful Git Commands

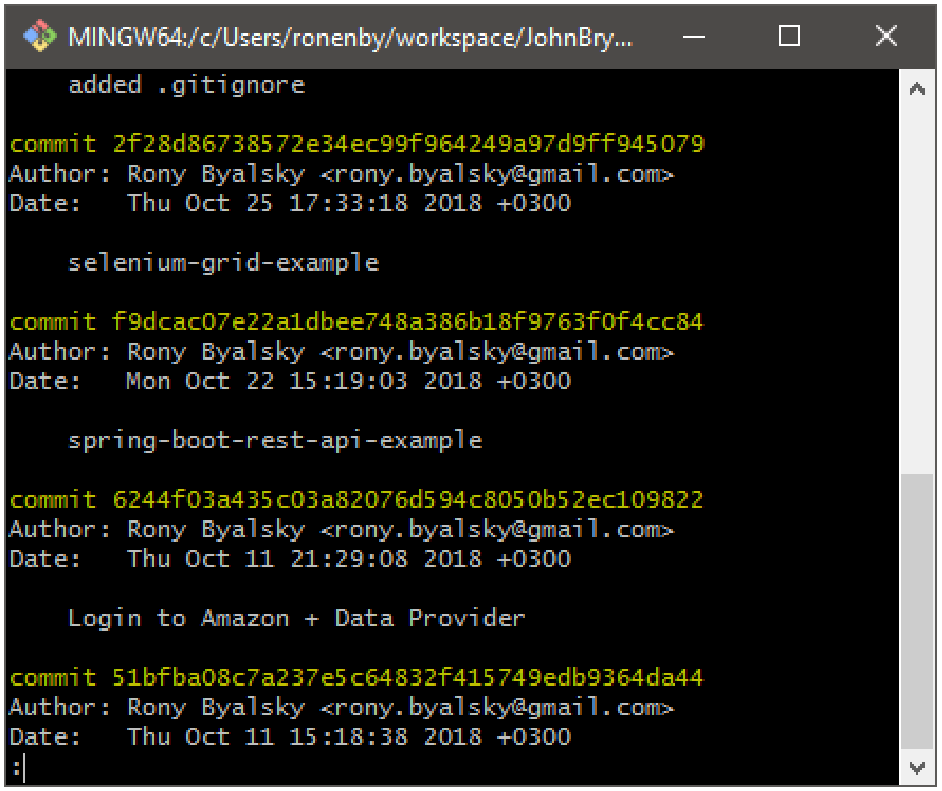
`$ git diff` - Shows the exact lines added and removed compared to the last commit.

`$ git log`

- Shows the history of commits

`.gitignore (file)`

- Use this file to list all files that you want Git to ignore (don't track changes and even don't show the file as being untracked).


MINGW64:/c/Users/ronenby/workspace/JohnBry...

```
    added .gitignore

commit 2f28d86738572e34ec99f964249a97d9ff945079
Author: Rony Byalsky <rony.byalsky@gmail.com>
Date:   Thu Oct 25 17:33:18 2018 +0300

    selenium-grid-example

commit f9dcac07e22a1dbee748a386b18f9763f0f4cc84
Author: Rony Byalsky <rony.byalsky@gmail.com>
Date:   Mon Oct 22 15:19:03 2018 +0300

    spring-boot-rest-api-example

commit 6244f03a435c03a82076d594c8050b52ec109822
Author: Rony Byalsky <rony.byalsky@gmail.com>
Date:   Thu Oct 11 21:29:08 2018 +0300

    Login to Amazon + Data Provider

commit 51bfba08c7a237e5c64832f415749edb9364da44
Author: Rony Byalsky <rony.byalsky@gmail.com>
Date:   Thu Oct 11 15:18:38 2018 +0300
:
```

Remote Repositories

- Remote repositories are versions of your project that are hosted on the Internet or network somewhere.
- Collaborating with others involves managing these remote repositories and **pushing** and **pulling** data to and from them when you need to share work.

```
$ git pull
```

- Incorporate changes from a remote repository into the current branch. Shorthand for **git fetch** followed by **git merge FETCH_HEAD**

```
$ git push
```

- Use this command when you want to share your project and push your latest commits to the remote repository.

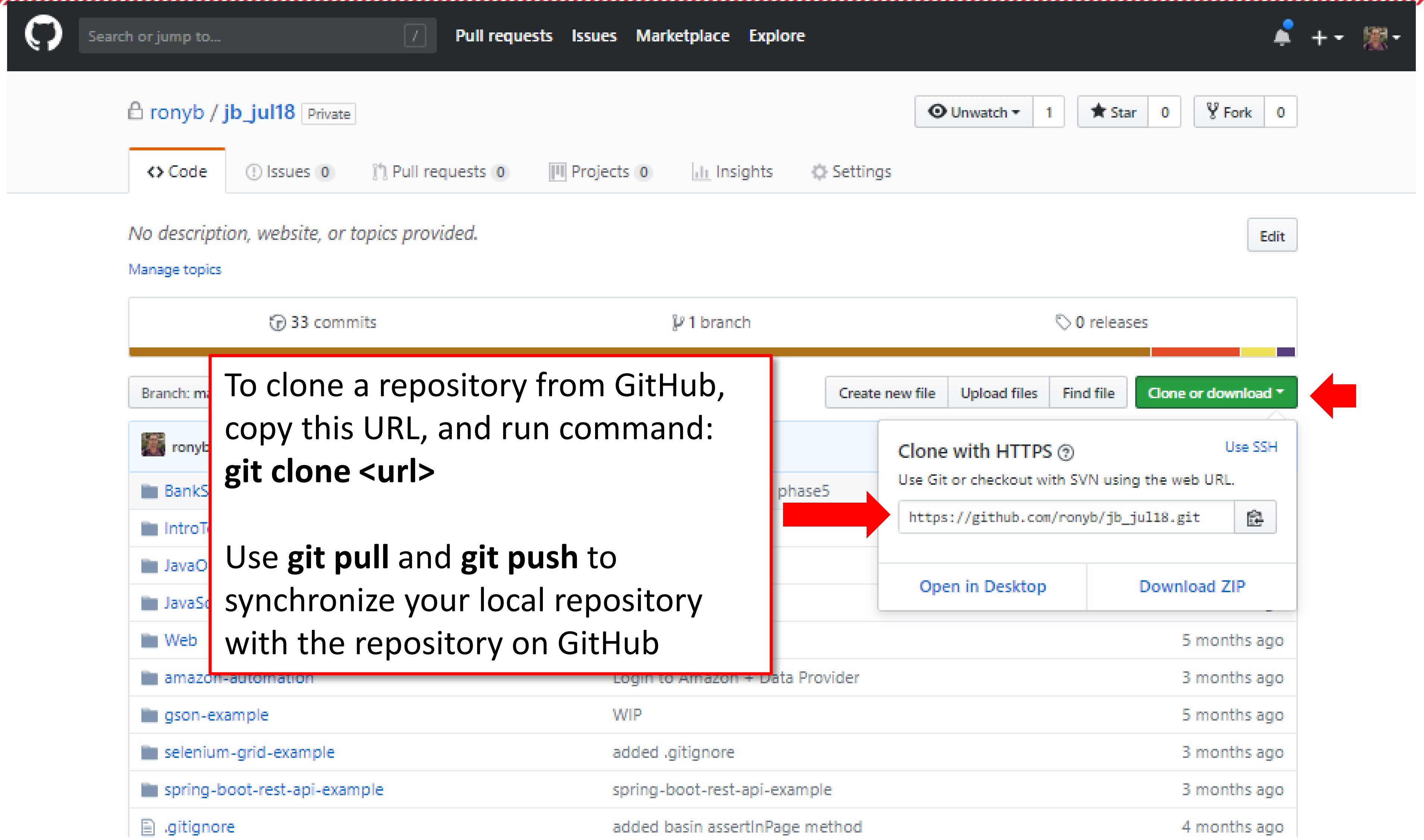
GitHub & Bitbucket

- **GitHub** is a **web-based** hosting service for version control using Git.
- It's the largest host of source code in the world: having over 28 million users and 57 million repositories (as of June 2018)
- Offers all of the functionality of Git, as well as adding its own **collaboration features:**
  - access control
  - bug tracking
  - feature requests
  - task management
  - code review
  - wikis
- Offers unlimited private repositories to all plans, including free accounts.

ATLASSIAN
Bitbucket

- **Bitbucket** is another popular hosting service owned by Atlassian (the owner of Jira), offering feature similar to GitHub.

Clone from GitHub



No description, website, or topics provided.

Use Git or checkout with SVN using the web URL.

```
https://github.com/ronyb/jb_jul18.git
```

BankS	phase5	
IntroT		
JavaO		
JavaS		
Web		5 months ago
amazon-automation	Login to Amazon + Data Provider	3 months ago
gson-example	WIP	5 months ago
selenium-grid-example	added .gitignore	3 months ago
spring-boot-rest-api-example	spring-boot-rest-api-example	3 months ago
.gitignore	added basin assertInPage method	4 months ago

To clone a repository from GitHub, copy this URL, and run command:
git clone <url>

Use **git pull** and **git push** to synchronize your local repository with the repository on GitHub

JOHN BRYCE
Leading in IT Education
a matrix company

Thank You!