Introduction to Git and GitHub

https://kau-github-workshop.github.io

Khalid Alharbi, Ph.D.

Outline

- Install *git*
- Configuring *git*
- Accepting the invite to the assignment
- Getting started with *git*

Getting Started with *Git*

1. Install git

```
https://gitforwindows.org
```

2. Set up git for the first time

```
$  git config
```

3. Clone your remote git repository

```
$  git clone
```

4. Add files to git

```
$  git add
```

5. Commit files

```
$  git commit
```

6. Push to a remote repository

```
$  git push origin main
```

Version Control

- Every software project results in artifacts
  - Requirement documentation, designs, prototypes, and source code files
- The source code is the most precious asset whose value must be protected
- Source code files need to be kept safe with a record of historical changes to the code and their authors
  - Seamless collaboration between multiple team members
  - Allow team members to work faster without stepping on each others' work
  - If a mistake is made, we can go back, compare the changes, fix the error, or revert (undo) the changes
- Version control is the practice of tracking and managing changes to software code over time
  - Version control protects source code from human errors and unintended consequences

Why Version Control

- Save and **track** your progress in **snapshots**
  - “Snapshot” is a way to record a set of changes to the source code.
- **Revert** back to a previous **snapshot**
  - Made a mistake or not happy with a change, undo the changes by going back to the previous snapshot.
  - Made progress and changes to the source code, crate another snapshot.
- Work on a separate version of the main code base in a new **branch**
  - Trying to fix a bug or add a new feature to the main code, create a new **branch**, test, and **merge** with the main branch.
- Work with a distributed team of developers
  - Track contribution and and gain insights on your project.
- Adopting version control is the de facto standard for Agile software development

Git

- Git is the most popular version control system
- Originally developed in 2005 by Linus Torvalds, the creator of Linux.
- Highly optimized for distributed workflow.
  - Committing new changes, branching, merging, comparing past versions
- Integrity and security is core component of Git
  - File contents, directory relationships, versions, tags, commits, branches, are all protected with a cryptographically secure hashing algorithm (SHA1)
- Flexible for both complex and linear workflow in small and large projects
- Git is the de facto standard for version control systems and every Software Engineer is expected to know how to use it.

TL;DR

You can think of *git* as:

1. a tool to keep track of different versions of a document
2. a powerful collaboration tool

Install Git

- Go get Git!
  - On Windows:
    - Go to https://gitforwindows.org/
    - Or go to https://git-scm.com/
  - On macOS:
    - Install Xcode Command Line Tools
    - Open the Terminal app and run `xcode-select --install`
    - Or go to https://developer.apple.com/downloads/index.action
    - Or go to https://git-scm.com/
  - On Linux:
    - Debian / Ubuntu (apt-get) `sudo apt update && sudo apt install git`
    - Fedora (dnf/yum) `sudo dnf install git` or `sudo yum install git`

Verify the Installation

- Open up the terminal application:
  - On Windows: Navigate using the Start button, and you should see **Git Bash** listed under the Git menu option.
  - On macOS: open **Terminal.app**
  - On Linux: open **Terminal** app
- Run the following command:

```
$  git ––version
```

  - If you have installed git successfully, you should see a version number instead of an error message (e.g., `git version 2.39.0`).

Tools you need

- You’re going to need a text editor and if you’re writing Java code you will also need an IDE to improve your productivity.
- Text editor:
  - Download and install VS Code: https://code.visualstudio.com/

- IDE:
  - Download and install IntelliJ IDEA (Community Edition): https://www.jetbrains.com/idea/download

GitHub

- GitHub is a web-based Git repository hosting service for software development, version control, and collaboration.
- You're going to need a GitHub account. Create one at https://github.com
- GitHub repositories can be accessed either over HTTPS or SSH
- **HTTPS**: generate a personal access token to authenticate with GitHub from the command line.
  - Repo's HTTPS URL: https://github.com/user/repo.git
  - See https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token
- **SSH**: We can also work with all repositories on GitHub over SSH. However, firewalls and proxies might block access to SSH port 22 as this the case with University network.
  - Repo's SSH URL: git@github.com:user/repo.git

Quick tour of the command line

- You’re going to be using command line when using git.
  - On Windows
    - Start menu -> type “**Git Bash**” into the search bar.
    - You may also use the modern terminal application from Microsoft Store called Terminal.
    - Or use the good old command prompt (cmd.exe) but you will be limited to Windows specific commands.
  - On macOS
    - In Finder -> Go menu -> Utilities -> Terminal.
  - On Linux
    - I’m not worried about you; let’s be real, you know what you’re doing 😁

Most common commands

Command	meaning	Example
cd	Stands for “change directory”. It is used to change the current working directory (i.e., navigate to another folder).	`cd /home/Documents`
pwd	Stands for “print working directory.” It displays the path of the directory you’re in.	`pwd`
ls	Stands for “list”. It lists the contents of the directory you’re currently in.	`ls`
mkdir	Stands for “make directory”. It creates directories	`mkdir myfiles`
cp	Stands for “copy”. It copies the contents of the source file to the target file.	`cp file1.java file1_copy.java`

Navigating directories

- To navigate the file system, you will run the `cd` command from the Terminal.
- There are two special objects inside every directory:
  - **A single dot** `.` represents the directory you are in
  - **A double dot** `..` represents the parent directory.
- The `pwd` command displays the absolute path of the directory your in

```
$  cd /Users/khalid/learning-git

$  pwd
/Users/khalid/learning-git

$  cd ..

$  pwd
/Users/khalid/
```

Listing directories

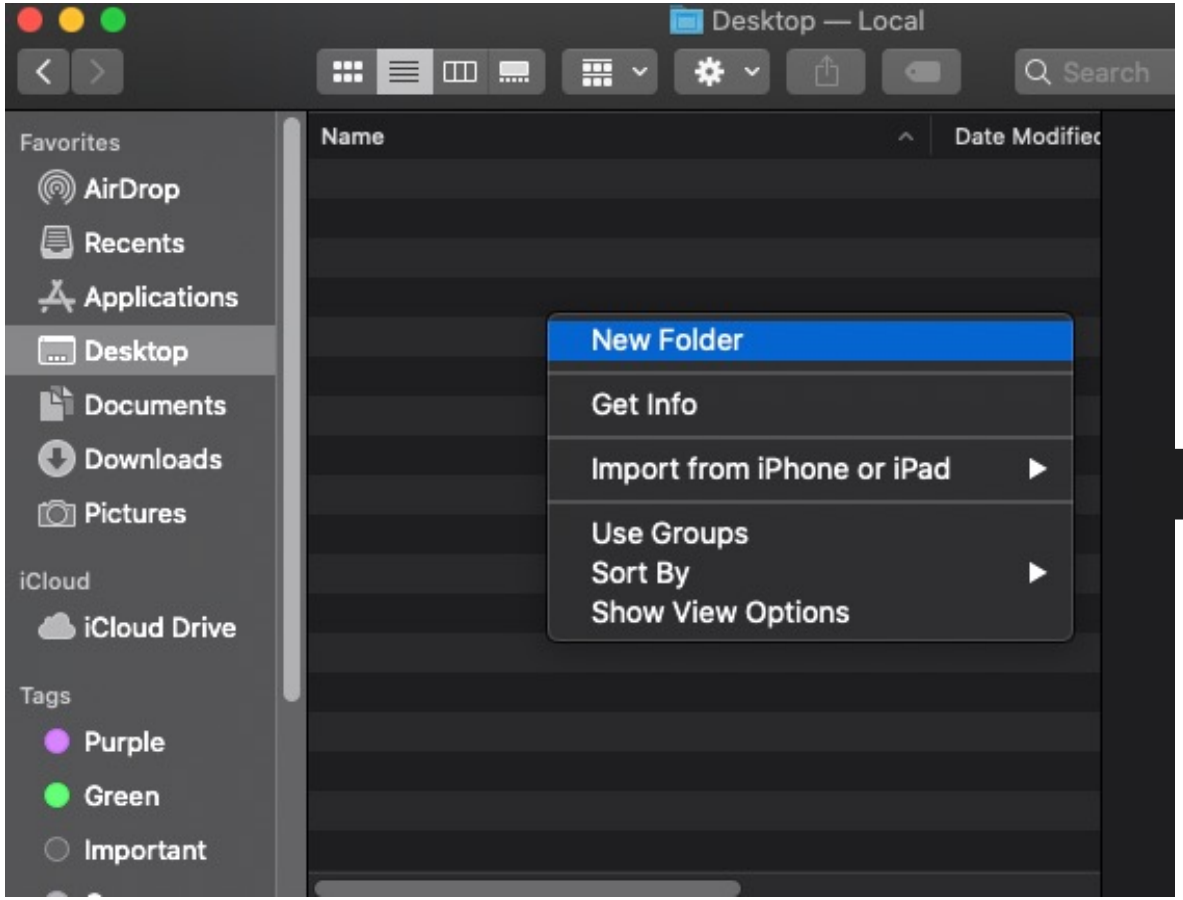
- To list the content of a directory, you will run the `ls` command
- The `ls` command may take a few optional command options:
  - `-a` to include all hidden files and directories
  - `-l` to list in long format that includes file sizes, ownership, and permissions

```
$  ls
pom.xml src

$  ls -a
.       ..      .git    pom.xml src

$  ls -a
total 8
-rw-r--r--  1 khalid  staff  2636 Jan 16 19:07 pom.xml
drwxr-xr-x  4 khalid  staff   128 Jan 16 19:07 src
```

Making Directories




```
$  mkdir my-java-project
```

Getting help in the command line

- All command line tools and utilities come with a manual
- The manual describes in detail how to use the tool with all valid commands and options.
- To access the manual, you either run the *man* program for system tools such ls as in `$ man ls` or you may use the tool name followed by `–help` for all third-party tools such as git.
- To read the help page of git, run:

```
$ git --help
```

Getting Started with *Git*

1. Set up git for the first time

```
$ git config
```

2. Create your first git repository

```
$ git init
```

3. Add files to git

```
$ git add
```

4. Commit files

```
$ git commit
```

5. Push to a remote repository

```
$ git push
```

Setting up Git for the first time

```
$ git config
```

- Let’s set our username and email to associate commits with an identify.
- To set your commit username, run:

```
$ git config --global user.name “Khalid Alharbi”
```

- To set your commit email, run:

```
$ git config --global user.email “khalid@example.com”
```

  - You may only use email addresses connected to your GitHub account.
  - If you would like to keep your email address, then you need to enable email privacy on your GitHub account settings and use the noreply email address in the form of `<username>@users.noreply.github.com`

Authenticating with *Git*

- You can connect to GitHub using either HTTPS or SSH
- HTTPS is recommended when you're in a restricted network where SSH may be blocked
- When using git over HTTPS, git will ask for your GitHub username and password
- Password-based authentication has been removed in favor of personal access tokens
- Git on Windows uses Git Credential Manager, which will open a link on your browser to authenticate with GitHub.
- On macOS, git will ask you for a password but again you can't use your GitHub password, so you need to generate an app specific password called "Personal Access Token" as per the steps below:

On your GitHub.com, click on your profile **->** settings **->** Developer settings **->** Personal access tokens **->** Fine-grained tokens **->** Generate new token

  - Give it a name (e.g., my laptop) and an expiration date.
  - Under "Repository access", select "All repositories"
  - Under Permissions, scroll to "Contents" and select "Access: Read and write"
  - Click "Generate token"
  - Copy the token as this will be used whenever git asks for your password.

- You may also use Git Credential Manager (GCM) or a git credential helper to cache the token securely.

Setting up and authenticating

Setting up git and authenticating with *GitHub* is only done once on a new machine and does not need to be repeated for every project.

Create a sample project

- Create a directory named *app*
- Create a file named *Main.java* inside *app/* with the following content

```
class Main {

  public static void main(String[] args) {

    int first = 10;
    int second = 20;

    // add two numbers
    int sum = first + second;
    System.out.println(first + " + " + second + " = "  + sum);
  }
}
```

Create your first git repository

```
$  git init
```

- A git repository is nothing but a folder managed by git with a hidden folder named *.git*
- Go to the directory where your app is stored at using cd
- To create a git repository, run the following command inside the project directory:

```
$  git init
```

Add and Commit

```
$  git add
$  git commit
```

- When running the git add command, Git stores a copy of that file in the index

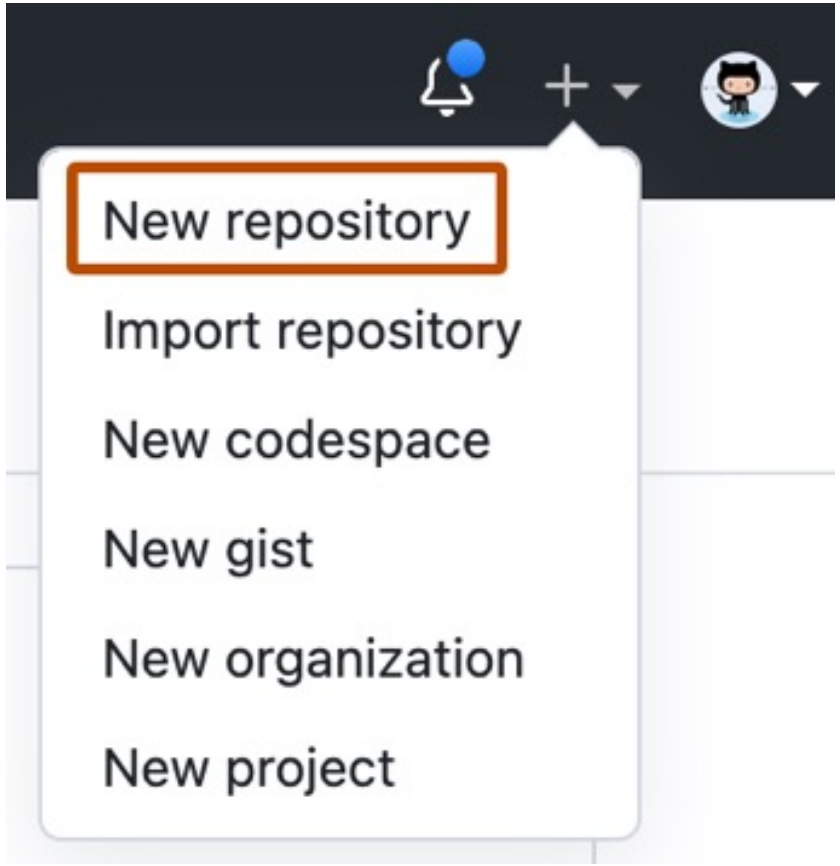
```
$  git add Main.java
$  git commit –m ‘added main class’
```

That was your first commit and snapshot!

Congratulations on your first commit!

Create a remote repository on GitHub

- That was all done locally, so let's create put our project on GitHub.
- Create a remote repository on GitHub by clicking on the "+" button:
- Give it a name (e.g., "github-workshop-example")
  - Do NOT check the boxes for adding README, license, etc.



Add the remote repository on GitHub

```
$  git remote add <name> <URL>
```

- Wi will need to tell our local repository that we have a remote repository on GitHub

```
$  git remote add origin https://github.com/OWNER/REPOSITORY.git
```

- Replace OWNER with your username on GitHub and REPOSITORY with the name of the repository you just created on GitHub.
- origin is just a shorthand name for the URL of our remote repository
  - Anytime we need to upload or download from our remote repo on GitHub, we will use *origin* instead of the long URL
  - By convention, the default remote repository is called "origin", but our local git repository can work with several remotes (with different names) at the same time.

Push: Upload local to remote

```
$  git push <remote_name> <branch>
```

- The *git push* command is used to upload local repository content to a remote repository. Pushing is how you transfer commits from your local repository to a remote repo.

```
$  git push origin main
```

Coming up next

- Complete the challenges in the workshop app, Git-it
- Undoing changes
- Branching
- Merging
- Resolving Conflict
- How to collaborate on GitHub