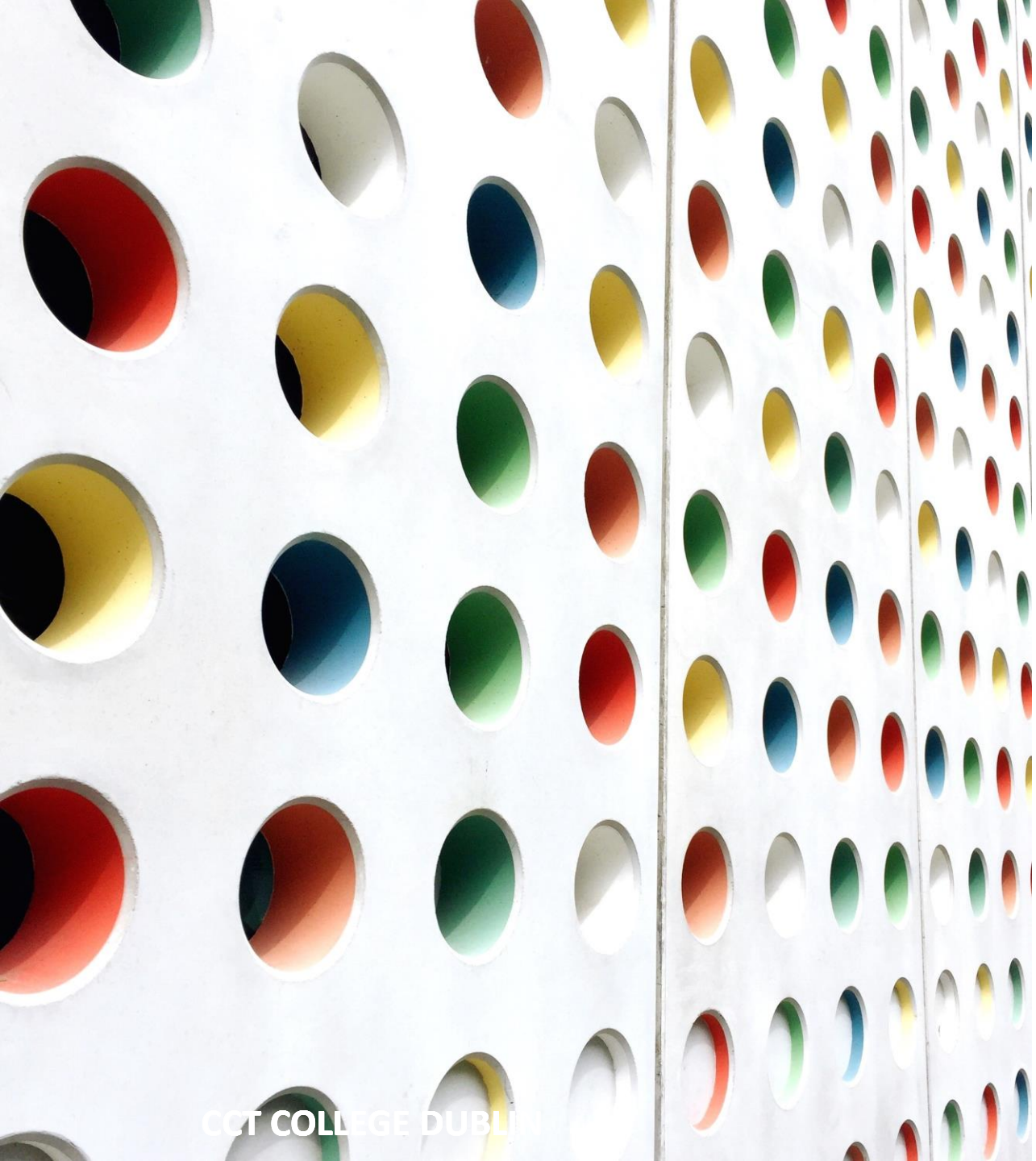


SOURCE CONTROL MANAGEMENT (SCM)

Lecturer: Dr. Muhammad Iqbal

Email: miqbal@cct.ie

CCT College Dublin

OUTLINE

- Source control management (scm)
- Why scm? And WHAT DO YOU MANAGE?
- Scm types
- GIT: CONCEPTS, BRANCHES, TAGS, MERGE VS REBASE and MERGE & CONFLICTS
- Project management
- SCM lab using Git and Github

SOURCE CONTROL MANAGEMENT (SCM)

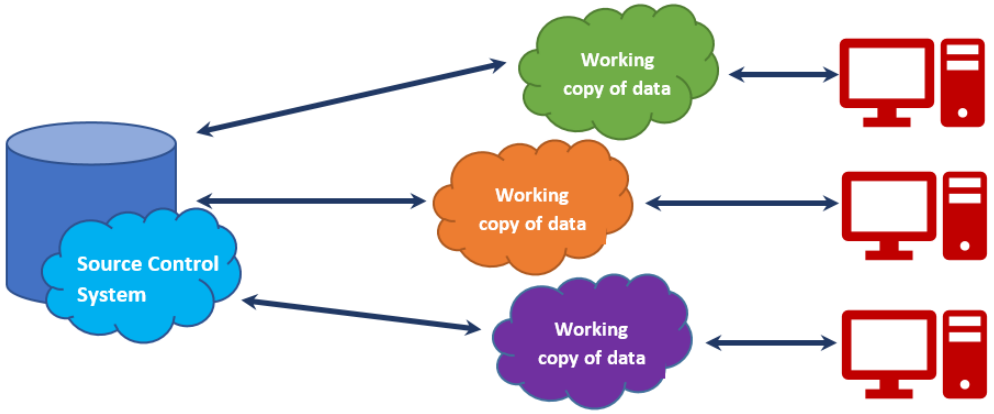
- Source control management **(SCM)** refers to tools that help you to keep track of your code with a complete history of changes.
- Source Control Management **(SCM)** is better known as **Revision Control Management**.

SOURCE CONTROL MANAGEMENT (SCM)

- **SCM** is necessary to keep projects under control, especially on multi-projects spanning different environments.
- A good **SCM** system should manage the following
  - Versioning and chronological changes
  - Comparing and merging
  - Restoring to checkpoint

SOURCE CONTROL MANAGEMENT (SCM)

- A source control system is a component of a source repository and version management system. If you are building **enterprise-level applications** in a distributed environment, you will want to keep your source code in a safe vault with easy-to-manage, easy-to-integrate, and easy-to-create versions of each check-in.
- The source control repository helps you to manage your code. There are plenty of source control repositories available in the market. Some of the most common repositories are Git, TFS, and SVN.



SOURCE CONTROL MANAGEMENT (SCM)

- In general, repositories are being hosted on on-premise environments or in a cloud-hosted environment such as Microsoft Team Services, Team Foundation Server, GitHub, Bitbucket, and so on, and developers connect to the remote repository to clone the entire changeset on their system to perform changes on their code.
- When the changes are over and well tested, they push those changes to the remote.
- We discuss the following core points to use a source control repository such as Git and Github Project using Github environment.

WHY SCM?

- How to share the work?
- How to keep track of changes and its authors?

HISTORY

- The concept started with engineering
  - **Drawing and project offices evolved a robust system for dealing with engineering processes, and thus were born the fledgling change control and release systems.**
- Modern implementation in software development since ~1980
- Git started in 2005

WHAT DO YOU MANAGE?

- Code, or any text based projects (like all the configuration needed for the project)
- Manage project versions
  - Global project version (tags)
  - Each modification is a «version»
- Change requests

SCM TYPES

- **CENTRALIZED**
  - has single central repository
  - single point of failure
  - requires a network to inspect a history
- **Examples:** Tortoise SVN

SCM TYPES

- **DISTRIBUTED**
  - local repository with full history
  - independent work
  - **Examples:** Git, Mercurial SCM

GIT: VIEWING

- We look at how you can view the state of a project to see what's going on, so you can find areas that could be improved or so you can be familiar with the interface when you're ready to contribute.
- **Bootstrap** is a repository that allows developers to quickly develop attractive web applications. We use the popular Bootstrap open source project as an example.
  - Viewing the **README.md** File
  - Viewing the Commit History
  - Viewing Pull Requests

https://github.com/twbs/bootstrap

GIT: VIEWING

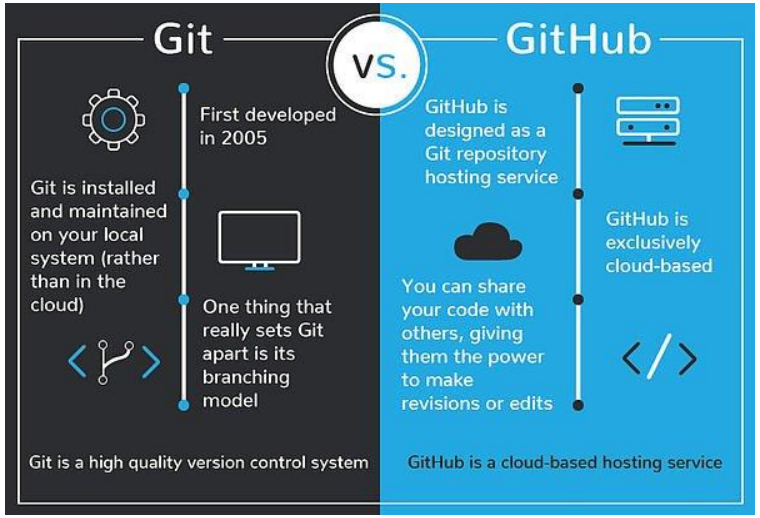
- We can understand from Bootstrap repository to view the state of a project and to gain an insight what's going on, so you can find areas that could be improved or so you can be familiar with the interface when you're ready to contribute.
- **Bootstrap** is a repository that allows developers to quickly develop attractive web applications. We use the popular Bootstrap open source project as an example.
    - Viewing the **README.md** File
    - Viewing the Commit History
    - Viewing Pull Requests

https://github.com/twbs/bootstrap

GIT AND GITHUB

- **Git:** Git is a version control system for tracking changes (SCM) in computer files and coordinating work on those files among multiple people. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

- **GitHub:** GitHub is a web-based Git repository hosting service, which offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features.



GIT AND GITHUB

- If you need to use Microsoft Dynamics NAV or Microsoft Dynamics 365 Business Central with SCM integration, Git is the best solution (free, open source and natively integrated with VS Code).
- Git is an open source distributed version control engine. VS Code supports Git natively. Some Git commands are as follows:
  - git remote add origin master <repo url>
  - git push -u origin master
  - git push -u add master
- The Git local repository commits to GitHub, pushes the content of the local repository into the branch called master.

GIT: PROJECT MANAGEMENT

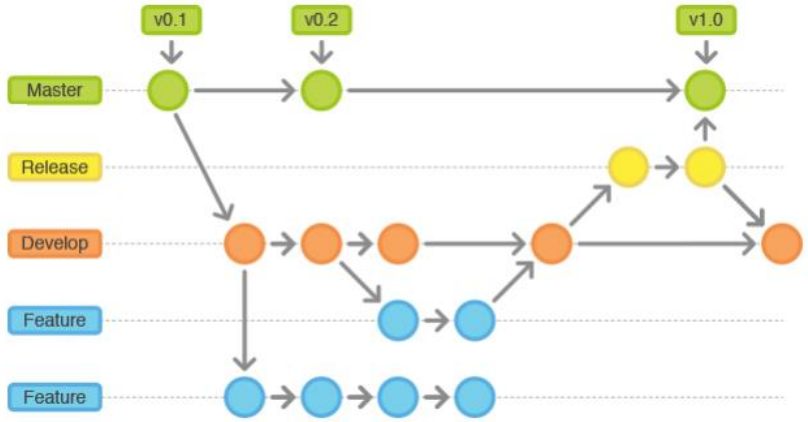
- For project management with GitHub, you can get better understanding for software development even if you're not writing code.
- Project management on **GitHub** typically starts in the form of simple task items that need to be worked on with **GitHub** Issues, organizing GitHub issues by applying labels to them and giving them deadlines with milestones.
- Finally, if your work involves project managers or more teams, you may decide to collect your issues and pull requests onto boards with GitHub Projects.
- GitHub Issues provide a lightweight, easy-to-use tool for managing outstanding work—whether it's bugs that need to be fixed or new features that need to be built.
- Generally, when starting a new project, someone may begin by managing both bugs and features using GitHub Issues.
- Later they may move to another tool like ZenHub, Waffle.io, or JIRA if they need features that GitHub Issues does not provide.

GIT: CONCEPTS

- Learn git - https://git-scm.com/doc
- **Git Project**
- Each set of changes to files are **commits**
- Each commit has a **parent commit** (except for the first one)
- The current commit is the **HEAD**
- The git project is kept in a remote location - **repository**
- Retrieving changes from the repo is **pulling**
- Sending changes to the repo is **pushing**
- When choosing what to commit, files are **staged**

GIT: BRANCHES

- **Branches** - independent set of commits
- So far all the commits are on the master branch
- Branch can be an abstraction for the line of development
  - master is the stable, tested version of the project
  - develop is the place where features come together
  - feature is the branch for single feature
- Branch can be temporary:
  - To have a change request
  - To develop an independent feature



- Create branches or switch between them with `git branch`

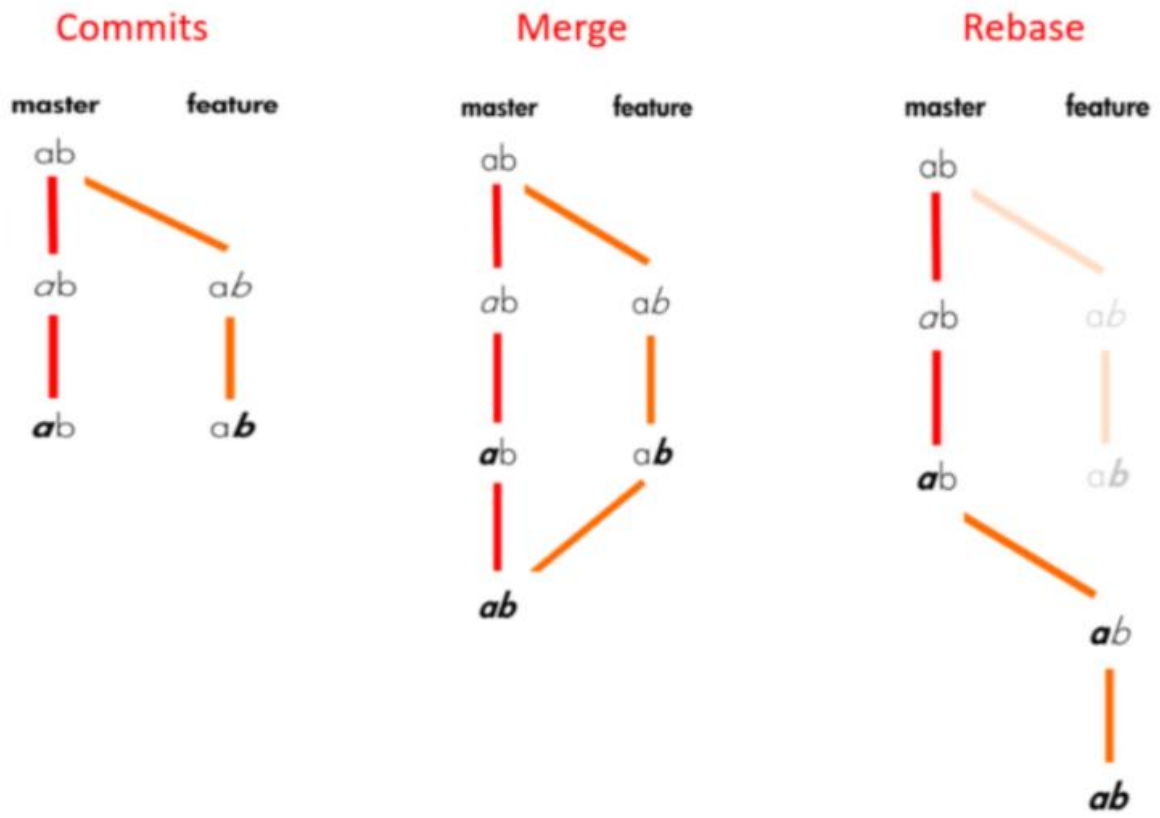
GIT: TAGS

- So far:
- changes are managed, not the project
- all the commits come one after the other
- the only version available is the **commit id**
- **Tagging** versions:
  - Tag a **commit** to be a particular software version
  - Easier to identify the versions
  - You have to **push** the tags

GIT: MERGE & CONFLICTS

- Branches can be merged to have all the changes in only one branch.
- When multiple changes are in conflict with each other, the merge fails:
  `Automatic merge failed; fix conflicts and then commit the result.`
- **Solutions:**
  - Cherrypick the changes (can be quite long)
  - Abandon your changes (and put them back later)
  - Work on the latest commit (git pull) and have a good task repartition

GIT: MERGE VS REBASE



- Read this article on Medium.

OPEN SOURCE PROJECT MANAGEMENT

- Code managed by git, how can people contribute?
- **Fork** the project on your account
- Work on a change on your fork
- Ask for your changes to be added to the official code - pull request

PROJECT MANAGEMENT WITH GITLAB

- Provides from basic issue tracking to scrum and kanban style
- Track and manage issues
- Issue boards to visualize work
- Agile project management
- DevOps pipeline traceability
- Read more about GitLab project management.

SCM LAB

- **OBJECTIVES**

1. Perform the Github Desktop Tutorial
2. Create a repository and clone it on your computer
3. Create a branch and navigate between branches
4. Modify a file in the repository and push the modifications
5. Manage conflicts

BEFORE STARTING

1. Install Git on your Operating system

2. Create a Github account: https://github.com/

3. Install Github Desktop from this url: https://desktop.github.com/

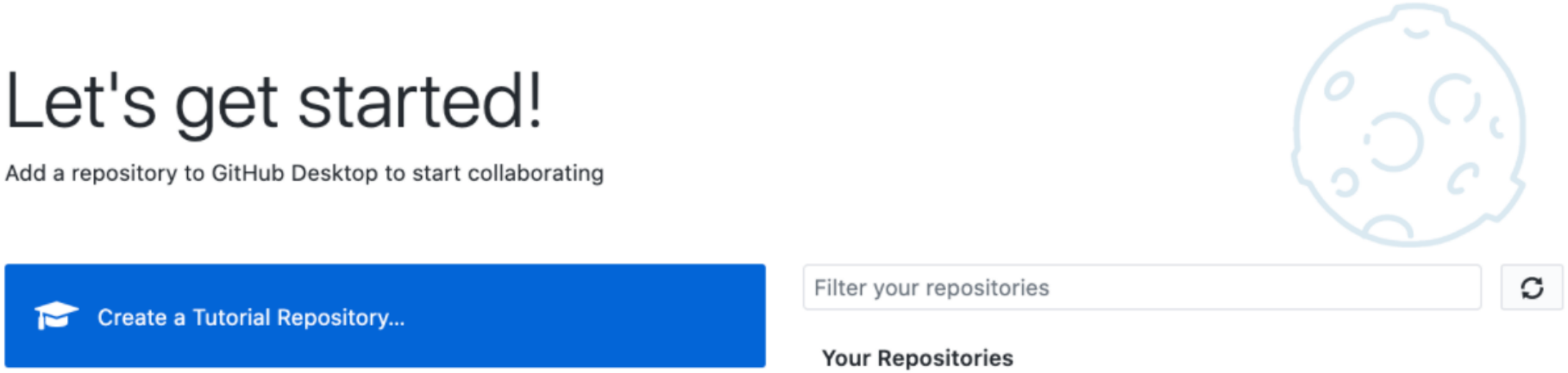
4. Install an IDE or text editor
   (ex: https://code.visualstudio.com/ or https://atom.io/)

5. Create an account on Github if you do not have already.

1. PERFORM THE GITHUB DESKTOP TUTORIAL

1. Launch Github Desktop

2. Start the Github Desktop Tutorial by clicking "Create a Tutorial Repository ..." and follow the instructions



2. CREATE A REPOSITORY AND CLONE IT ON YOUR COMPUTER

- **A single member of the group creates a repository**

1. Log into Github.com
2. Navigate to the "**Your repositories**" page
3. Click on "**New**"
    1. Choose a name
    2. Chose the "**Public**" option
    3. Check the box "**Add a README file**"
    4. Check the box "**Add .ignore**" with the "**Node**" template
4. Navigate to the created repository, then to the **Settings → Manage Access** page
5. Click on "**Invite a collaborator**" and add the other members of your group

2. CREATE A REPOSITORY AND CLONE IT ON YOUR COMPUTER

- All group members clone the repository

    6. Open Github Desktop

    7. Click on "Clone a repository from the Internet"

    8. GitHub.com → Filter your repositories

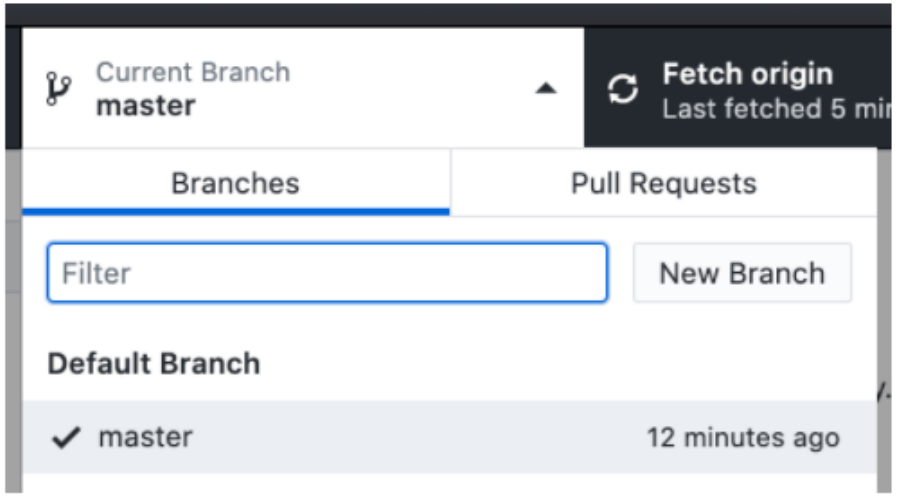
    9. Click on “Clone”

- You now have a copy of the local repository

3. CREATE A BRANCH AND NAVIGATE BETWEEN BRANCHES

- **A single member of the group creates the develop branch:**

1. In Github Desktop: Current branch → New branch



2. Choose the name of the branch (develop) and "Create branch"

- You can now navigate (= checkout) between the master and develop branches

4. MODIFY A FILE IN THE REPOSITORY AND PUSH THE MODIFICATIONS

- A single member of the group modifies the "README.md" file and pushes to the develop branch:

1. Modify the "**README.md**" file entirely:
    - Open the repository in your IDE
    - Replace the entire contents of the "**README.md**" file
2. In Github Desktop:
    - Choose a commit message ("Summary")
    - Click on "**Commit to develop**"
    - Use "**Publish branch**" to push your changes to the remote repository

4. MODIFY A FILE IN THE REPOSITORY AND PUSH THE MODIFICATIONS

- The other members of the group get the changes

3. Use "Fetch origin" to synchronize your local repository with remote changes

4. Navigate to the develop branch and watch your files change in your IDE

5. MANAGE CONFLICTS

- A conflict occurs when a part of a file has been modified on 2 branches which must be merged (**merge**). We will create a conflict to see how to resolve it.
- Both **2 members** of the group create a branch from develop and modify the same part (same lines) of the **README.md** file:
    1. Create a branch dev-firstname "**based on ... develop**"
    2. Edit the **README.md** file
    3. Commit changes
    4. Push your changes

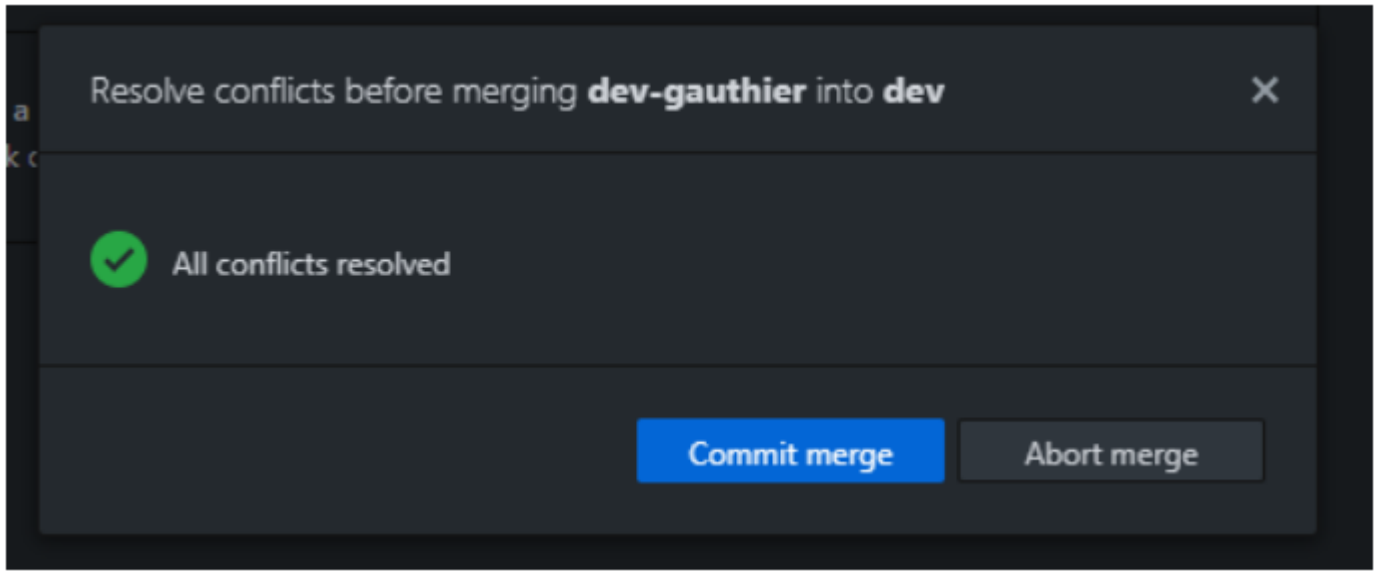
5. MANAGE CONFLICTS

- Alternately, each member merges the new branches into develop

5. Checkout the develop branch
6. Click on "Merge into current branch ..."
7. Select branch
8. Push your changes
9. **The 2nd member** of the group will encounter a conflict
    - Open your IDE as suggested by Github Desktop
    - The conflict materialize as follows:

5. MANAGE CONFLICTS

- To resolve the conflict, keep either the HEAD part, or the dev-firstname part and delete the other lines (including the ==== and >>>> lines)
- Once the lines are deleted you can commit the merge from Github Desktop:



6. REDO THE LAB USING COMMAND LINE INTERFACE (CLI)

- The goal is to redo the same actions as before with the command line. For this you must have a terminal and GIT installed by default on Linux or MacOS.
- For the installation:
  - Windows: https://gitforwindows.org/
  - Linux: https://git-scm.com/download/linux
  - MacOS: https://git-scm.com/download/mac
- You can now open Terminal (on Linux or MacOS) or Git Bash (on Windows).

6. REDO THE LAB USING COMMAND LINE INTERFACE (CLI)

- You can find a detailed list of the different GIT CLI commands:
- https://gist.github.com/aquelito/8596717
- https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf
- https://www.geeksforgeeks.org/difference-between-git-and-github/

BONUS TASKS

1. Add multiple remotes (reference here)
2. Squashing commits (reference here and here)
3. Reverting history (reference here and here)
4. Rebasing (reference here)
5. Cherry-picking (reference here and here)
6. Learn Monorepos in Git (reference here)