College of Engineering
Computer Science & Eng. Dept.
Course: COE420
Software Engineering Lab
Date: 2nd February 2020
Location: EB2-125

Course Instructor: Dr. Raafat Aburukba
Email: raburukba@aus.edu
Office: ESB-2052
Lab Instructor: Ms Hend ElGhazaly
Email: helghazaly@aus.edu
Office: ESB-1043 B

Lab 1: Version Control

Objectives:

- Introduction to version control methodology in GitHub.
- Using GitHub to version control the software under production.
- Creating GitHub accounts, organization, repository.
- Pushing versions to the remote directories using Git Bash.
- Pulling the versions from the remote directory using Git Bash.
- Reporting and fixing issues using GitHub.

Before you start:

- Agree on who will be *member_A, member_B, member_C, member_D,* in your team.
- Download the Java program from iLearn (for exercises 2 to 5).
- Download the report template from iLearn. Paste the screenshots as you are doing the lab exercises to avoid missing any required screenshots.

You should work **individually** on the lab but **submit only one group report** with all the screenshots.

Hand in: ***One team member needs to*** upload **one group** report on iLearn (following the template provided) that includes:

- Cover page with names and IDs of each team member
- The link to the GitHub repository created.
- Screenshots from Git Bash and GitHub for each exercise

Due Date: Before the next lab session (5% per day will be applied on late submissions).

Resources:

1. Gitbook: https://git-scm.com/book/en/v2
2. GitHub Tutorial-Video: https://www.youtube.com/watch?v=mYjZtU1-u9Y
3. Linux Torvald on Git: https://www.youtube.com/watch?v=4XpnKHJAok8

Introduction:

Why version control?

- Help a software team manage changes to source code over time.
- Help keep track of every modification to the code in a special kind of database
- Help developers to turn back to the earlier versions of the code to help fix the mistakes while minimizing disruption to all.
- Helps teams work on different components (tasks) of an application concurrently and allows integration of these components.
- Help a team to work on fixing the bug in an older version and another team to work on new feature of the application.
- Helps tracking changes made by the contributors and record date and related notes of the changes.

What is GitHub?

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. To use the GitHub platform, you must have an account on GitHub.

Note: Essential tutorials for using GitHub for source control management is available on website www.GitHub.com

What is Git?

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Merits of Git SCM (Source Control Management)

- Speed
- Simple design
- Strong support for non-linear development (thousands of parallel branches)
- Fully distributed
- Able to handle large projects like the Linux kernel efficiently (speed and data size)

Terminology:

- **Organization:** it is a shared account where the team members can collaborate. It allows the members of the organization to clone or download the content from (master or branches) repositories of the organization.

- **Repository:** a project folder which contains all the project files and each file's history.

Exercise 1: Create accounts, organization and repository on GitHub (all team members)

- Each team member needs to create an account on *GitHub*.
- Only one team member needs to create an organization and a repository.

member_A is the owner of the repository and needs to invite his/her team members. *member_B, member_C and member_D need* to join the organization to work on the software part of the project. The team members need accounts on GitHub.

Sample organization name: COE420-Hend-S20
Sample Repository Name: COE420Lab1
Team Members: Hend and Mohamed
Users accounts names on GitHub:
Hend: helghazaly
Mohamed: melghazaly

a. *Create an account on GitHub (all team members)*

Browse *www.github.com/join* and create an account.

Note that the username and organization's name have to be unique (not used before).

b. *Create a new Organization (member_A only)*

While creating an account, check the box that asks *if you would like to create an organization account* and fill the details (e.g: organization name: COE420-Hend-S20). If you have not checked the option to create an organization account at sign up:

Click on the + icon at the top right and select *New Organization*. Select the "***Team for Open Source***" plan and fill the details.

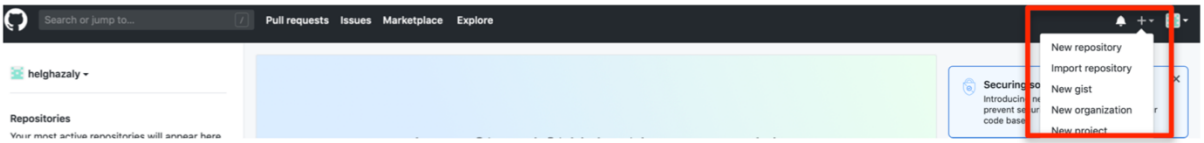


Figure 1: Creating an organization

c. *Invite organization members (member_A only)*

- Enter the username/Full name/email/ name of their GitHub account.
- You can add any number of users.
- Make the other members also owners of the organization (e.g. *member_B: Mohamed* also owner *COE420- Hend-S20*) (Figure 2).

An invitation will then be sent to the team members. Each team member has to accept the invitation by opening their email and accepting the invitation to join the organization.

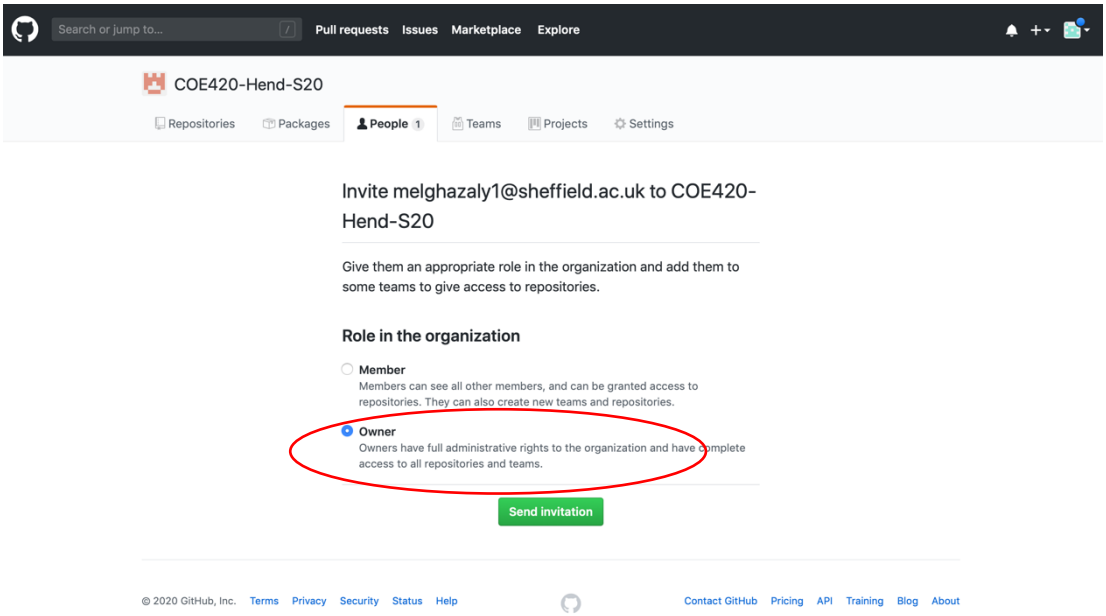


Figure 2: Invite team members to join the organization and make them owners

d. ***Create a new repository within the organization created (member_A only)***

1. Go to your organization and click on "***Create a new repository***" or in the upper right corner click + and then select ***New repository***
2. Ensure that the owner selected is the name of the organization (*e.g. COE420-Hend-S20*)
3. Name your repository ***COE420Lab1*** and write *a* ***short Description***. The name should be unique to that particular member's account.
4. Set it to ***Public***
5. Click on *"**Create repository**"* button

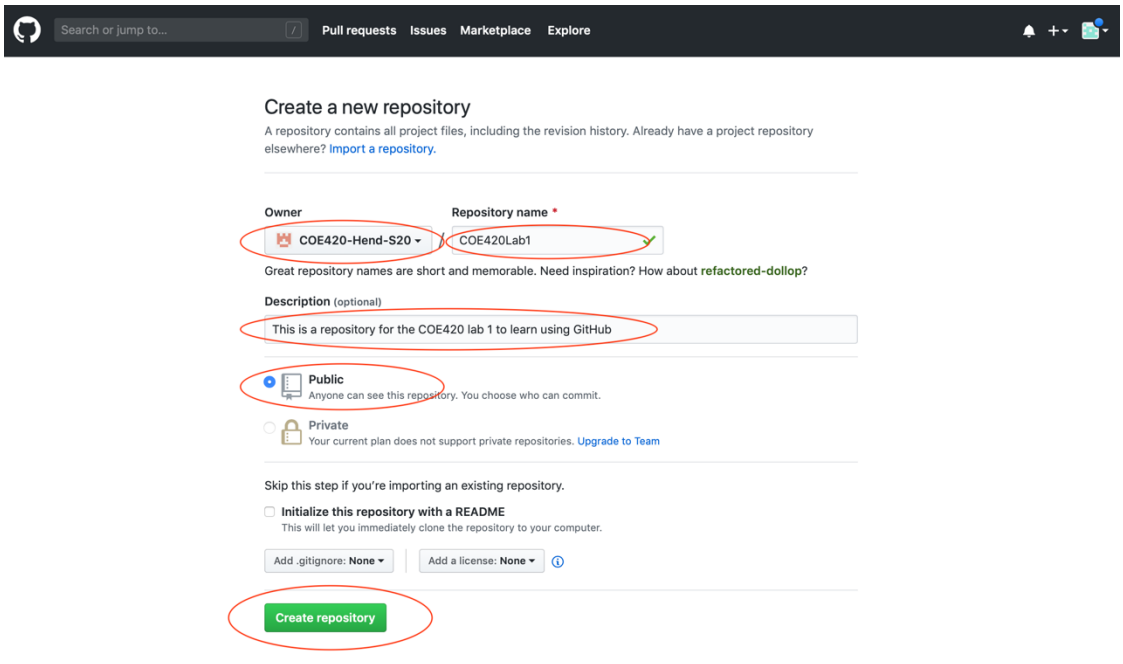


Figure 3: Create a repository within the organization

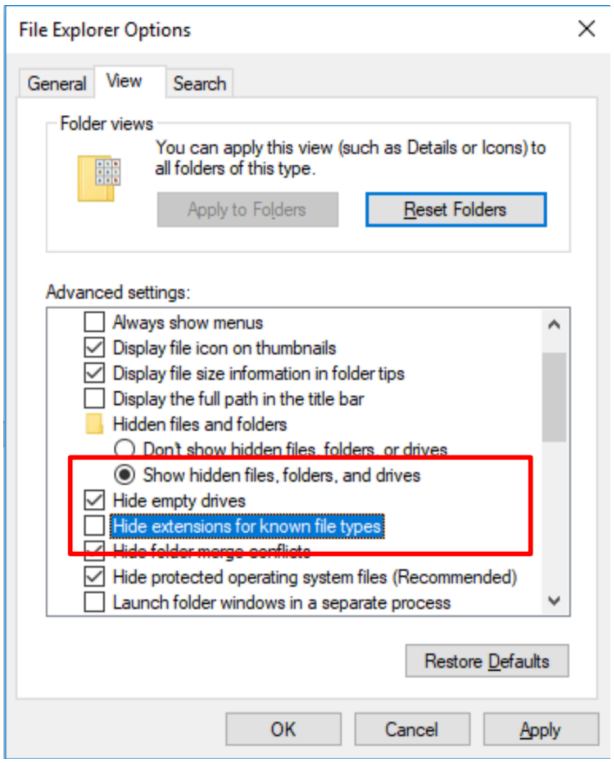
Exercise 2*: Pushing initial commit using *Git Bash (member_A only)
Download the java file provided on iLearn (Calculations.java)

member_A will do the following:

- Compile, run and commit the java program (Calculations.java).
- Create a Git local repository by name *COE420Lab1* using *Git Bash* and *Git* commands.
- Push the local repository *COE420Lab1* to the remote repository (e.g: *COE420-Hend-S20/ COE420Lab1).*

Steps for *member_A* (e.g. Hend on her work station)

1. (Optional - if needed) Select *Start -> Control Panel ->File Explorer Options -> View*
   a. *Select* (the radio button) "Show hidden Files, folders, or drives"
   b. *Uncheck* "Hide extensions for known file types"



2. Open the Java file (*Calculations.java*) in Eclipse (*Eclipse Java 2019-06*) and save it by name *COE420Lab1* in *U:/member a* drive
3. Open *Git Bash* (command line)
4. Change directory `cd U:/`
5. Create local repository *COE420Lab1* and confirm that *COE420Lab1* is master.

```
$git init
```

6. Create your identity. The first thing you should do when you install Git is to set your user name and e-mail address. This is important because every Git commit uses this information, and it's immutably baked into the commits you start creating.

```
$git config --global user.name helghazaly
$git config --global user.email helghazaly@aus.edu
```

Modify the commands with your details

7. Run the command to get the status of the repository and observe the output. The files shown in red color indicate that they are not in *staging* area.

```
$git status
```

8. Place all the files in *staging* area and know the status of the project. The files shown in green are all in the staging area.

```
$git add .
$git status
```

9. Place all the files in the local repository and check commit details

```
$git commit  -m  " Initial Commit"
$git log
```

10. Push the local repository content to the remote repository
    a. Create a short name to the remote repository

```
$ git remote add origin https://github.com/COE420-Hend-S20/COE420Lab1.git
```

Modify the link to the name of your organization and repository

    b. Push the local repository (master by default) to the remote repository using the short name created.

```
$ git push origin master
```

(Since you are pushing this to remote repository, a *sign up* window pops up and you need to sign in into your account to the local repository to the remote repository that you created)

- **Add in the report screenshot from Git Bash showing the commands were run and screenshot from GitHub showing the commit made**

Exercise 3: Pushing changes via Git Bash (member_B, member_C, member_D)

The following steps need to be done by the rest of the team members (add screenshots from Git Bash in the report):

1. Each member needs to clone the repository *COE420Lab1* in the folder *U:/member_**n***

   ```
   $cd U:/
   $git clone https://github.com/COE420-Hend-S20/COE420Lab1.git
   ```

   Modify the link to the name of your organization and repository

2. Complete the code in *Calcualtions.java* according to your member number (e.g *member_B* implement the *subtraction*) and push changes made back to the repository:
   a. Open Eclipse
   b. Open projects from File System... and select *Member_B* folder
   c. Add the feature (s*ubtraction*) to the Java program
   d. Save file

3. Push the local repository after adding the feature (e.g *subtraction by member_b*)

   ```
   $cd COE420Lab1
   $git add .
   $git commit –m "Subtraction implemented"
   $git push –u origin master
   ```

   Sign in to your GitHub account if the window pops up

Exercise 4: Getting Updated Version – all team members

a. Each team member should do this step before pushing their changes. *Member_A* should also update his/her local repository with the features added by the other team members:

```
$cd /u/member_a/COE420Lab1
$git pull origin master
```

b. Check if *member_A* can see the feature added by the team members

- Open Eclipse
- Open projects from File System... and select *member a* folder
- Check if the features added is seen by *member a*

Exercise 5: Reporting issues – all team members

There are 2 issues in *Calculations.java*. Let *member_A and member_B* identify the *bugs* in the code and post the issues on *GitHub*. Then let *member_C and member_D* fix these *bugs* and close the issues.

- **Add in the report a screenshot from the issues in GitHub showing the comments and that they are closed**

Appendix – Some useful commands

Git Commands:

`git config –global user.name user_name` : Create global identity
`git config –global user.email user_email` : Create global identity
(This info is attached with all commits)
`gitinit`: Creates local repository
`git add .` : Moves all file from working directory to stage area
`git commit –m "Commit message"` : Moves all files from stage area to local repository
`git remote add` *`origin URL_to_remote_repository :`* Create a short name (origin) to URL
`git push –u origin master` : Push master to the remote repository
`git fetch origin` : fetch from the remote repository to the local repository
`git merge origin/master` : Merge the recent fetch with the local repository
`git pull origin` : fetch and merge with local master
`git status`: Lists paths in the working directory that that are not tracked by git.
`git log`: Used to view commit history

Unix Commands:

```
$ls -> List all files in the present working directory
$cd path -> Change director to path
$cd .. Move up by one directory
$pwd -> Show present working directory.
```