Lab 3 – Feature branching with Git Bash

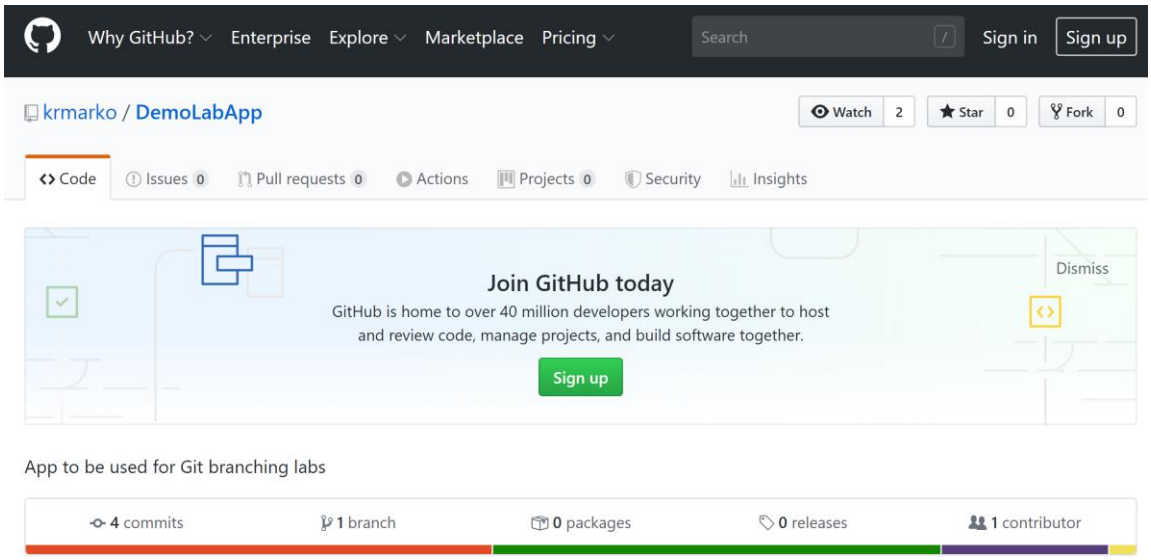
Objectives:

After completing this lab, you will be able to work with a remote repository:

- Fork and clone a remote repository
- Create local feature branches to makes changes and push them up to the remote repository

Fork a Git Repository

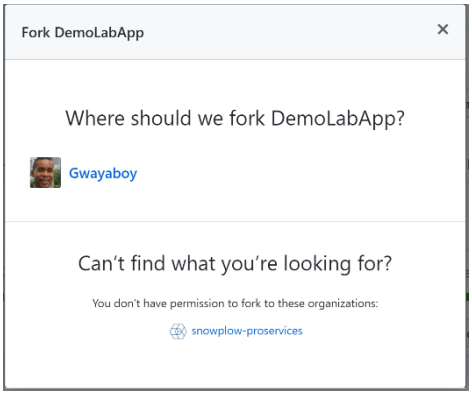
1) Open your favourite web browser and navigate to https://github.com/krmarko/DemoLabApp.



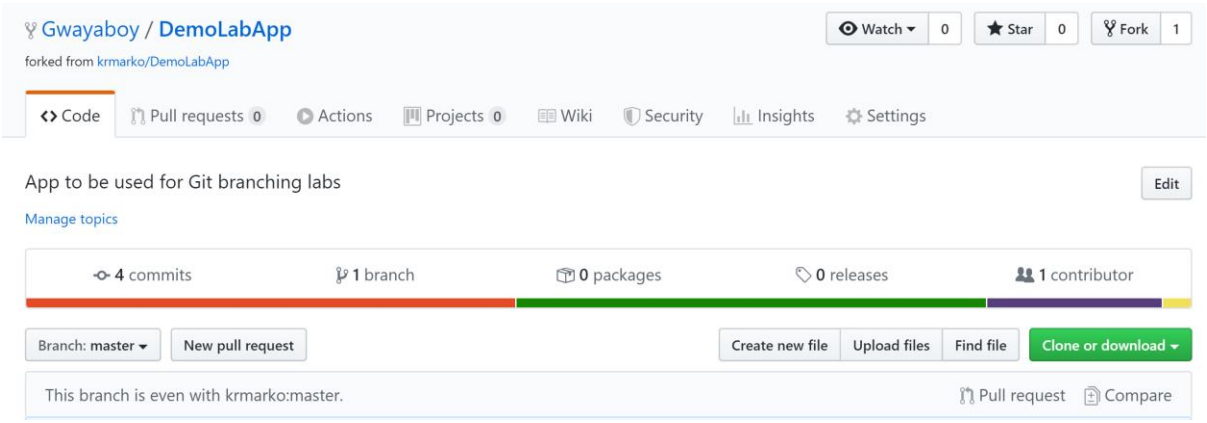
2) Ensure you are signed into your GitHub account.

   If you are not yet signed in click on the "Sign in" button on the top right to authenticate with your personal GitHub account. If you haven't signed up yet, please create one here

3) Click on the Fork button on the top right corner and confirm that you want to fork DemoLapApp into your personal GitHub account.



4) You should then be redirected to the forked repository within your GitHub account



Clone Remote Git Repository

Open a Git bash command prompt and navigate to a folder on your local drive that will contain the DemoLabApp project we will be working on.

For example, I am setting my current directory to my local c:\dev\ folder as below:

```
frtheola@MININT-N4S2CM2 MINGW64 ~
$ cd /c/dev/

frtheola@MININT-N4S2CM2 MINGW64 /c/dev
$
```

If you wish to have an identical setting as above, please execute the following set of commands in Git Bash, `cd /c` then `mkdir dev` and finally `cd dev`

1) Either way, once you've changed to the desired directory, then clone your remote repository using the command below:

```
git clone https://github.com/<yourGitHubUserName>/DemoLabApp.git
```

Please replace *<yourGitHubUserName>* with your GitHub username which you can find in the URL.

In my case that would be https://github.com/Gwayaboy/DemoLabApp.git

if the command was successfully run, you should then see something similar as below. Git is extremely good at compressing which greatly reduce the payload over the wire.

```
frtheola@MININT-N4S2CM2 MINGW64 ~/source/repos
$ git clone https://github.com/Gwayaboy/DemoLabApp.git
Cloning into 'DemoLabApp'...
remote: Enumerating objects: 84, done.
remote: Counting objects: 100% (84/84), done.
remote: Compressing objects: 100% (65/65), done.
remote: Total 84 (delta 14), reused 79 (delta 13), pack-reused 0
Receiving objects: 100% (84/84), 681.43 KiB | 252.00 KiB/s, done.
Resolving deltas: 100% (14/14), done.
```

2) Navigate to the DemoLabApp subfolder with

```
cd DemoLabApp
```

Please also note that when cloning a remote git repository, it automatically sets your origin alias from the provided URL (https://github.com/*<YourGitHubUserName>*/DemoLabApp.git)

You can verify this by executing the following command.

```
git remote get-url origin
```

```
frtheola@MININT-N4S2CM2 MINGW64 /c/dev/DemoLabApp (master)
$ git remote get-url origin
https://github.com/Gwayaboy/DemoLabApp.git
```

Create a new feature branch

We are going to create a local feature branch so that we can make our changes while protecting the integrity of our main or master branch.

1) Create a feature branch using the first part of your email address. For example, mine would be frtheola_myNewFeatureA

```
git branch <emailHost>_myNewFeatureA
```

2) Then switch to the newly created branch with the following command:

```
git checkout <emailHost>_myNewFeatureA
```

You should have something similar once you've executed the commands above:

```
frtheola@MININT-N4S2CM2 MINGW64 /c/dev/DemoLabApp (master)
$ git branch frtheola_myNewFeature

frtheola@MININT-N4S2CM2 MINGW64 /c/dev/DemoLabApp (master)
$ git checkout frtheola_myNewFeature
Switched to branch 'frtheola_myNewFeature'

frtheola@MININT-N4S2CM2 MINGW64 /c/dev/DemoLabApp (frtheola_myNewFeature)
$
```

Please note that you can create and switch to a new branch in a single command line:

```
git checkout -b <emailHost>_myNewFeatureA
```

and you will get the following message and prompt:

```
frtheola@MININT-N4S2CM2 MINGW64 /c/dev/DemoLabApp (master)
$ git checkout -b frtheola_myNewFeature
Switched to a new branch 'frtheola_myNewFeature'

frtheola@MININT-N4S2CM2 MINGW64 /c/dev/DemoLabApp (frtheola_myNewFeature)
$
```

Branching is a core, cheap and optimised feature of git, using short-lived feature branches will help with better continuous integration, multiple team members to work simultaneously while protecting the master branch. We'll discuss later considerations around merging branches back to master.

Make changes

We are going to add a new about.cshtml razor view, add a link to the to it in the top menu in our new feature branch

1) Navigate to the Views/Shared folder

```
cd Views/Shared/
```

2) Then create a new empty about.cshtml file

```
touch about.cshtml
```

Please note that this command wouldn't work outside of Git bash in a classic or PowerShell prompt.

3) Get back to the root of the DemoLapApp project directory by executing the following command **twice**: (also note space between cd and ..)

```
cd ..
```

Your current directory should now be at the root project folder

```
frtheola@MININT-N4S2CM2 MINGW64 /c/dev/DemoLabApp/Views/Shared (frtheola_myNewFe
ature)
$ cd ..

frtheola@MININT-N4S2CM2 MINGW64 /c/dev/DemoLabApp/Views (frtheola_myNewFeature)
$ cd ..

frtheola@MININT-N4S2CM2 MINGW64 /c/dev/DemoLabApp (frtheola_myNewFeature)
$
```

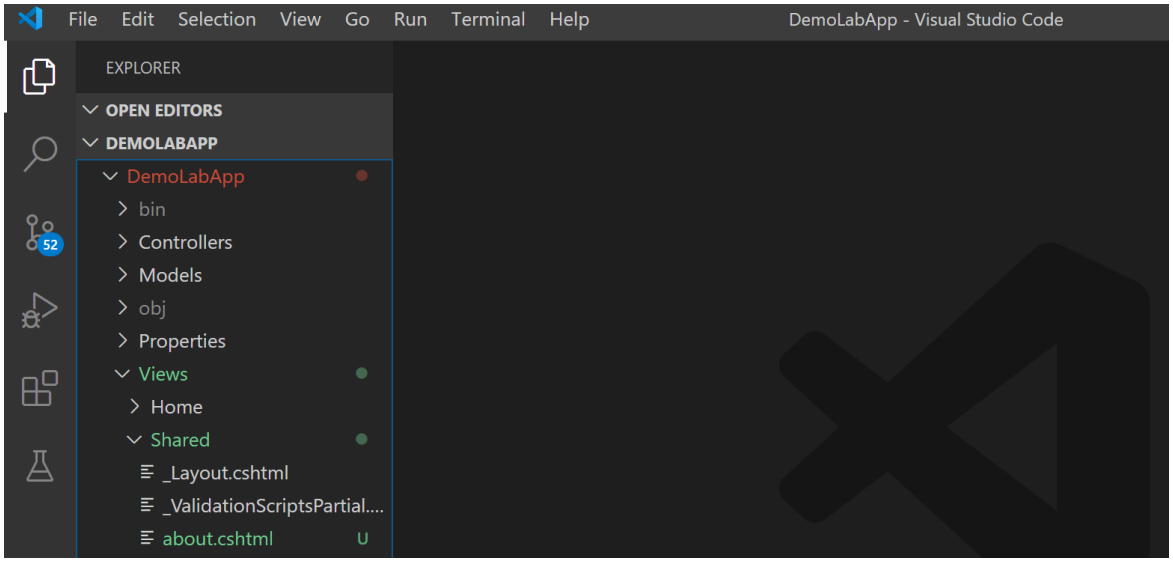
Please note you can achieve changing to the root project directory by specifying the full path to the DemoLabApp directory as follow:

```
cd /c/dev/DemoLabApp
```

4) Then open Visual Studio Code using the current directory project folder

```
code .
```

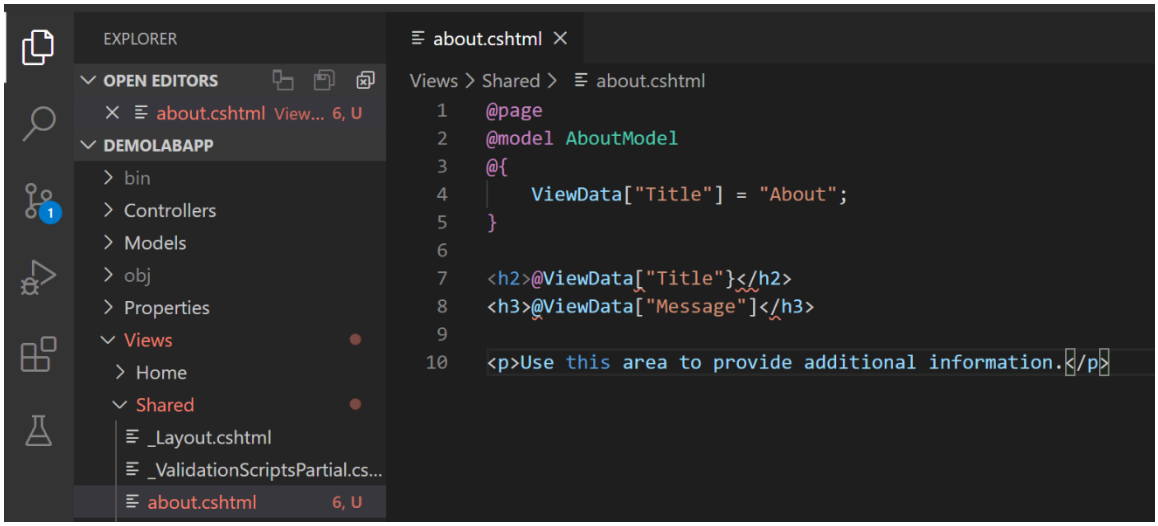
Visual studio code will then highlight untracked/modified new files in green and mark them with **U** (as untracked change) such as about.cshtml



5) Select about.cshtml file in the left project tree pan and then please copy/paste and save the following content:

```
@page
@model AboutModel
@{
    ViewData["Title"] = "About";
}

<h2>@ViewData["Title"}</h2>
<h3>@ViewData["Message"]</h3>

<p>Use this area to provide additional information.</p>
```

Please notice that once the file is saved Visual Studio Code will highlight in red the new added and mention 6 new lines were inserted.



6) Next, we are going to amend the _Layout.cshtml - *responsible for rendering for all pages a top menu* – by adding a new about menu item link. In Visual Studio code please select the _Layout.cshtml file on the left pan within the project explorer tree.
After adding the 3rd list item (li) in the unordered list (ul) the HTML top menu code should look as below:

```
<div class="navbar-collapse collapse d-sm-inline-flex flex-sm-row-reverse">
  <ul class="navbar-nav flex-grow-1">
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
    </li>
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
    </li>
    <li class="nav-item">
          <a class="nav-link text-dark" asp-area="" asp-controller="About" asp-action="about">About</a>
    </li>
  </ul>
</div>
```

7) Save the file and let's finally amend the HomeController to serve the appropriate view. Under the **Controllers** folder, select the HomeController.cs file in the project tree explorer After the Privacy action (ignore code before and after), add the About action method as follow:

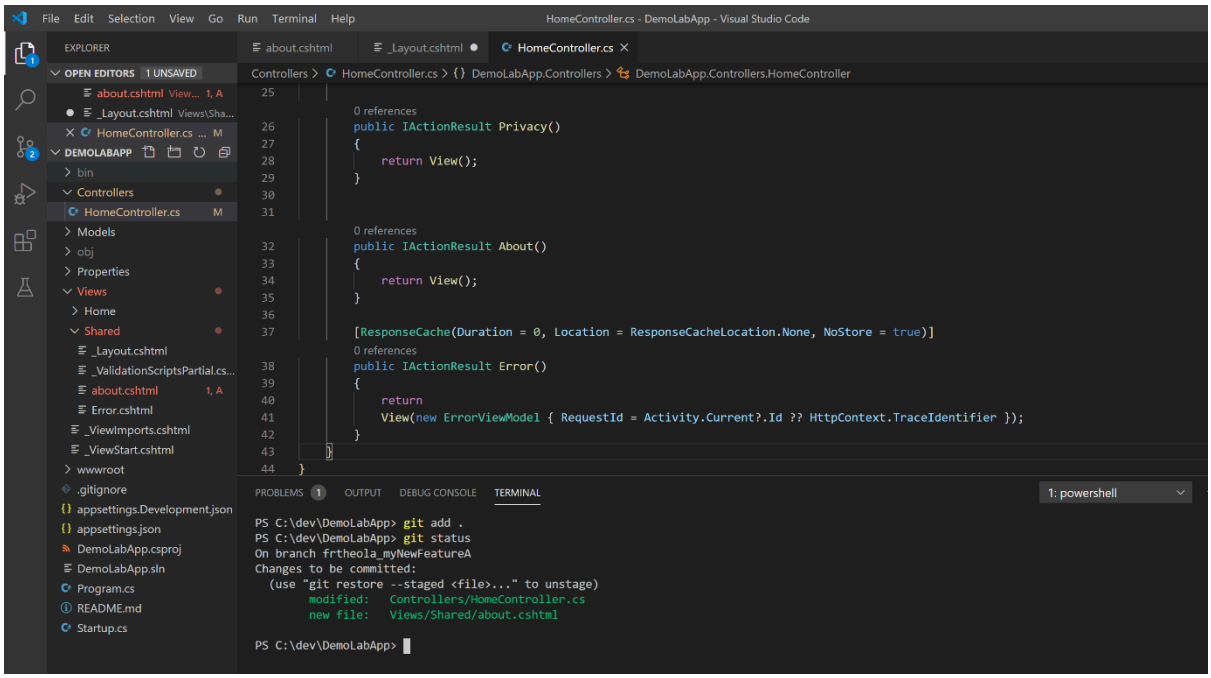
```
public class HomeController : Controller

{

      #region some code before…
      public IActionResult Privacy()
      {
          return View();
      }
      public IActionResult About()
      {
          return View();
      }
      #region more code after…
}
```

After saving the file please notice again how Visual Studio Code is keeping track of the changes on the HomeController by highlighting orange the modified file and marking with **M**

8) Now let's go back to Git Bash to add all our changes to staging.

```
git add .
```

Please note that Visual Studio Code also has an integrated PowerShell terminal prompt to execute git commands which will be accessible if C:\Program Files\Git\bin path is set in your environment variables or if you've installed appropriate VS Code git extensions like GitLens.



Whether you've used Visual Studio Code's terminal prompt or Git Bash if you execute the `git status` command, you should see 1 modified and 1 new file added to staging.

```
frtheola@MININT-N4S2CM2 MINGW64 /c/dev/DemoLabApp (frtheola_myNewFeatureA)
$ git status
On branch frtheola_myNewFeatureA
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Controllers/HomeController.cs
        new file:   Views/Shared/about.cshtml


frtheola@MININT-N4S2CM2 MINGW64 /c/dev/DemoLabApp (frtheola_myNewFeatureA)
```

9) Now let's commit our staged changes

```
git commit -m "added 'About' functionality"
```

Please notice that you must provide with a comment when committing any staged changes. Also, it is recommended to have meaningful and functionality centric comments rather than a mere description of the technical components or files added/deleted/amended.

10) Finally let's push our committed changes and our new feature branch up to our remote repository (remember your feature branch name will be something like *<emailHost>_myNewFeatureA)*.

```
git push origin <yourNewFeatureBranchName>
```

```
frtheola@MININT-N4S2CM2 MINGW64 /c/dev/DemoLabApp (frtheola_myNewFeatureA)
$ git push origin frtheola_myNewFeatureA
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 707 bytes | 353.00 KiB/s, done.
Total 7 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
remote:
remote: Create a pull request for 'frtheola_myNewFeatureA' on GitHub by visiting:
remote:      https://github.com/Gwayaboy/DemoLabApp/pull/new/frtheola_myNewFeatureA
remote:
To https://github.com/Gwayaboy/DemoLabApp.git
 * [new branch]      frtheola_myNewFeatureA -> frtheola_myNewFeatureA
```