Getting Started

In Phase 3, your team will work together on a shared Git repository to complete each week's assignments. Only one of you needs to follow steps 1-9 below to create your team's git repository. The remaining team members should complete steps 10-13. (Note: All team members will need to execute step 5 upon cloning the repository, and before they begin working on this phase of the project)

1. Create a new directory on your computer named **capstone**. We recommend putting this in an easy to access location on your computer (i.e. `/Users/<username>/Documents/computing/capstone`)

   `mkdir capstone && cd capstone`

2. In your newly created directory, initialize a git repository. If you have not already, you will need to install git.

   `git init --initial-branch=dev`

   Here we specify dev as the initial or default branch. We use dev to align the terminology between deployments/environments and the branch names in our source code repositories.

3. Open this directory in Visual Studio Code.

4. Create a file in this directory named `README.md`. In that file, add the following pieces of information and save the file.

   i. A Team Name

   ii. Your name

5. Create a Python Virtual Environment. In the integrated VSCode terminal window run:

   `python -m venv .venv`

   Once created, you can **activate** this virtual environment as follows:

   (Linux/Mac) `source .venv/bin/activate`

   (Windows) `.venv\Scripts\activate`

6. Create a new file in your directory named `.gitignore`. In this file, add the following line:

   ```
   .venv
   ```

   This tells git to recursively ignore any files in the .venv directory.

7. Commit these files to your git repository. Before doing so, check the status of changes in your working directory.

   `git status`

   You should see something like the following in your terminal. If not, revisit the previous steps.

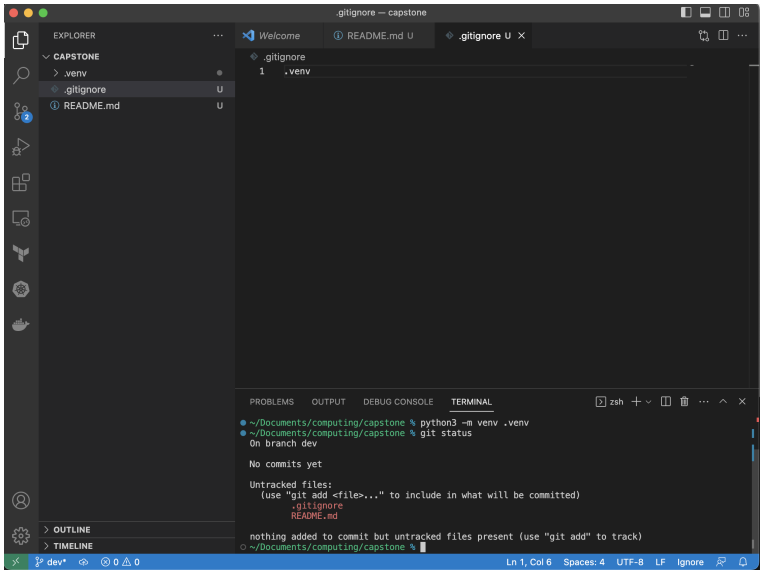

Figure 3.1: git status output after creating new files

If your output looked like the screenshot above, run the following to "stage" your changes.

```
git add README.md
```

```
git add .gitignore
```

Now that you have added the file to your 'staged' changes, when you run `git status` again you will see a different outcome, verify that now:

```
git status
```

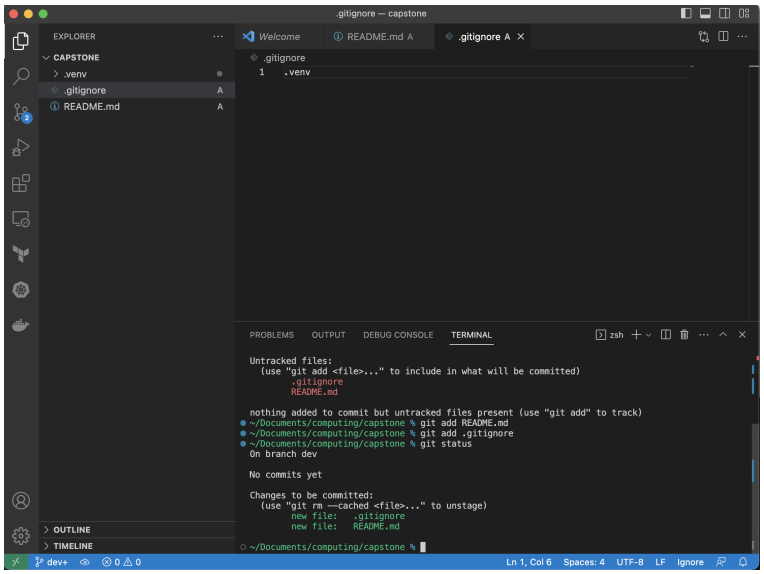

Figure 3.2: git status output after staging your work

If your output looked like the image above, commit your changes.

```
git commit
```

Note: if you have not used git before, this command will fail and you will need to configure your settings as follows before you re-run the above command.

```
git config --global user.name <Your GitHub username here>
```

```
git config --global user.email <Your GitHub email here>
```

Note: A commit message or "gist" should follow the following format. First, there should be a short, single line summary which will serve as the commit title. Then, after an empty line, a sequence of short descriptions should describe every change (there should only be a small number included in each commit) included in the commit. Example below.

```
Initialize Project

Created our team's initial README.md
```

As you can see, this commit message is succinct, but clearly describes what was changed in the repository in this commit. Beyond serving as an esoteric reference of past work, this makes reviewing different versions of work and tracking down future bugs faster and more reliable on your team.

8. Create a remote repository. While you would typically do this yourself (and select the repository location best suited for your team's access and development needs) we have created a remote repository for you in the AFC-AI2C GitHub. Insert the link below where specified in the below command. This command specifies a remote repository that your newly created git repository will 'track' at the specified URL.

   ```
   git remote add origin <remote repo link here>
   ```

   i. `https://github.com/AFC-AI2C/capstone-team-1.git` (CW2 Gonzalez, SPC Lacovara, SPC Fortuna)
   ii. `https://github.com/AFC-AI2C/capstone-team-2.git` (CW2 Vickers, SFC Schulze, SPC Morales)
   iii. `https://github.com/AFC-AI2C/capstone-team-3.git` (SFC Thames, SSG Malagon, SPC Tarila)
   iv. `https://github.com/AFC-AI2C/capstone-team-4.git` (1LT Cheng, SSG Nguyen, SSG Mangual)
   v. `https://github.com/AFC-AI2C/capstone-team-5.git` (1LT Owens, SFC Thompson, SSG Kuewa)
   vi. `https://github.com/AFC-AI2C/capstone-team-6.git` (SFC Belyayev, SSG Shields, SSG Prickett)

vii. `https://github.com/AFC-AI2C/capstone-team-7.git` (1LT Robinson, SSG Ballew)

9. Push to the remote repository. This command pushes your changes to the specified remote repository, setting the local branch called `dev` to "track" the remote branch with the same name in the remote repository named `origin`.

   ```
   git push --set-upstream origin dev
   ```

10. Now that you have a repository setup, each additional team member needs to setup the directory by "cloning" this repository as follows.

    In a shell, command prompt, or terminal, navigate to the directory you want to store Phase 3 work in (suggested: `/Users/<username>/Documents/computing`) and run the command below. You can find the repo link in your team's link above under the green "Clone" button.

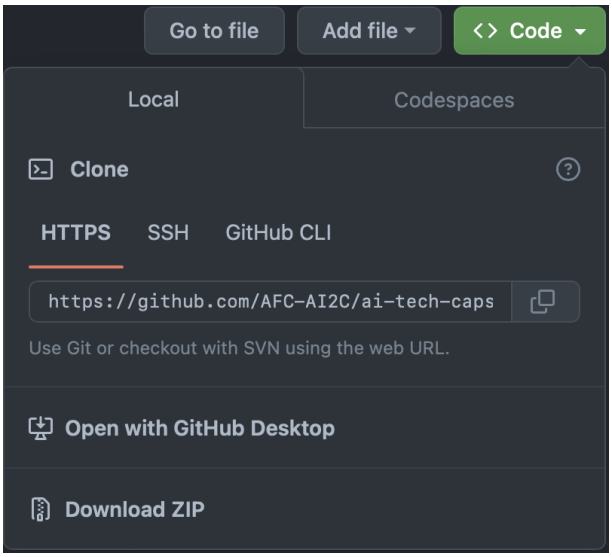

Figure 3.3: GitHub repository clone button

   ```
   git clone <remote repo link here> capstone
   ```

   This will create a directory called "capstone" that will store the contents of your team's remote repository. Open this directory in VSCode.

11. Create a new branch. Open the integrated terminal in VSCode and run:

    ```
    git checkout -b add-<your name>
    ```

    You have now created a place you can make changes to your team's source code safely and in parallel with your other teammates.

12. Open the README.md file and add your name to the list of names. Save the file.

13. Run the following sequence of commands to stage, commit, and push your work.

    i. `git status`
    ii. `git add README.md`
    iii. `git commit`
    iv. `git push -u origin add-<your name>`

14. Open your team's repository in GitHub and create a pull request from your branch to dev.

15. After another team member reviews your work, merge your work into dev. Once you, or anyone else on your team starts to work on the project again, start first by running `git pull` to fetch the latest changes to your base branch, dev, before you create and checkout a new branch.

Best Practices for working with Git

1. Commit and push your changes whenever you either a: finish a major change/feature or b: finish working for the day.
2. Have a team mate to review your work; another set of eyes and a different perspective are always helpful.
3. Be explicit in actions. Only add files you are sure you want to add. Use `git status` to check yourself and your work before you commit.