# GIT - Real example

**GITHUB - Create a new repo**

Go to website *github.com/new*

Create a new repo (*myPI*), mark it public so that others will be able to see it.

Click Initialise this repository with a README

Click create repository.

Under the **CODE** icon you will find the URL for the repo. This can be used in the command line window to checkout the code.

For *myPI* the repo link is [*https://github.com/breskit/myPI.git*](https://github.com/breskit/myPI.git)

On PC, create a workspace where the checked out project can live.

D:\Users\tbreski\OneDrive - Capgemini\Desktop\GITHUB

Clone & checkout the code from GitHub using the repo URL

**git clone** [*https://github.com/breskit/myPI.git*](https://github.com/breskit/myPI.git)

This creates a new folder (*myPI*) where the code is checked out to.

Add the PI calculation file into the folder.

Add the file to the local GIT structure.

**git add** calculatePI.py

Now commit the file into the project.

**git commit** -m “Added in PI calculation file stolen from the interweb”

Next, push changes for the local branch up to the web:

**git push -u origin master**

If you refresh the web page, you will now see the new python file included.

Our **master** branch is now ready.

# Create a new branch to work on

Now let’s create a new feature branch to work on where we can get the latest version of the project, make some changes, and commit this back to the main branch.

For now, we will create a **develop** branch that we will use to make changes on. Periodically we will merge the **develop** branch into the **master** branch and will continue to make further changes on the **develop** branch.

Let’s make the branch. From within the current project folder, create a new branch.

**git checkout -b develop**

A confirmation message is given that you are now switched to the new branch called **develop**.

At present, the only place this develop branch exists in on your local machine.

Next, let’s make sure we track changes to all files, so let’s add them into our branch.

**git add .**

Now, let’s make a simple change to the README.md file so that we can see a difference between the **master** and **develop** branches.

Edit the README.md file with VS Code and save your changes.

Using **git status** you will see that the file is now in a **modified** state. Now that we are done editing this file, we need to add the file into our staging area using **git add READM.md** or **git add .** to include all changed files into the staging area.

Once in the staging area we can commit the changes into the branch, adding a comment for what we have done.

**git commit -m “Updated README with additional details”**

The change is now committed, but it is only present on your local machine. What we need to do next is to ensure that the develop branch is updated on GitHub. At present, it doesn’t even show!

To add our branch into GitHub we need push it. To do this we use:

**git push -u origin develop**

On the GitHub portal, you can now open a pull request to have the changes merged into master.

Click on the Pull Request tab and open and submit a new pull request.

Once happy you can accept the pull request to merge the changes by clicking on **Merge pull request**, and then **Confirm**.

Now, on GitHub, you can check the versions of the files across the two different branches and they should be identical.

Next, let’s add a new file into our develop branch.

From VS Code, add a new file called **testing.txt**. Edit and save the file.

Using **git status** we can see that there is a file that is currently untracked and needs to be added, so that it is included in our next commit. Let’s add that using **git add testing.txt**

Next, we commit the change into the project using **git commit -m “Added new file”**

The final piece to ensure this is in GitHub is to push the file. This time we simply need to use **git push** as we have already linked our local project to that on the web.

Again, to ensure the master branch is updated, we need to create another pull request.

In GitHub make sure we select the **develop** branch and from there we can see the warning that we are ahead of **master** and we can create the pull request from there. Alternately we can click on the pull request tab, select **master** as **base** and **develop** as **compare** and can create the request this way.

Once again, accept and confirm the merge.

Go back to the main GitHub code page and you will now see the new file added.

# Working with other people’s projects

Start by navigating to GitHub for the project you wish to work with and create a fork of the project. This will clone a copy of that project within your own GitHub account.

From there, steps are like working with one of your own projects. You should clone this to your new local workspace, make the required changes, and push back to GitHub. The real fun starts when you wish to commit your own changes into the original project.

Having pushed your own changes back in to your cloned repo, from there you can raise a pull request. It will be up to the original author(s) to accept or reject the pull request you raise. Following that, you will see your changes accepted (or rejected!) into the project.