# GIT – Set Up New Repository

When setting up a new Git repository for a Visual Studio solution, the easiest seems to be the use of Visual Studio’s Git user interface. Refer to the Pluralsight course “Using Git For Source Control In Visual Studio 2019”, or refer to my notes at **sncole\source\GitRepositories\WebAppCourseNotes\** **UsingGitForSourceControlInVisualStudio2019\** **UsingGit4SrcCtlInVS2019.docx**. In particular refer to the section titled Create a Local Git Repository.

For a non-Visual-Studio Git repository there are 2 options: (1) set up a folder for the repository; this folder can contain subfolders; specify the files to be tracked; commit to git; (2) clone a git repository from elsewhere; at this stage of my experience (8/6/2020) I don’t yet how to clone.

To build the repository from scratch, create a folder – optionally with subfolders - that contain one or more files to be tracked.

Using Git Bash

* Navigate to the top folder containing the project; then issue command

$ git init

* Type one or more commands of the form

$ git add *filename* (‘\*’ denoting wild-card is permitted)

This “stages” the file (or files) in anticipation of making it (or them) “committed”.

* Type the command

$ git commit –m ‘Initial project version’

This changes the status of all staged files in the repository to “committed”. ‘Initial project version’ is a comment indicating the 1st step of tracking; you are welcome to change the wording.

* Files that have been added (even when not yet committed) are deemed to be “tracked” by Git. If you want git to list the files that are being tracked

$ git ls-tree –r master - -name-only

Using Git GUI

* Start GitGui; git responds with a dialog titled Git Gui.
* Click **Create New Repository**; git responds with a dialog subtitled “Create New Repository”
* Click the **Browse** command button; git responds with a file browser.
* Browse to the top folder containing the project; then click the **Select Folder** command button; git responds by copying the directory of the repository into the **Directory:** text box.
* Click the **Create** command button; git responds by displaying a dialog with an upper section “Unstaged Changes” and a lower section “Staged Changes (Will Commit)”. The number of files listed under “Unstaged Changes” may be quite large, and it may include files that you will not want git to process (e.g. binary files, library files, . . .). If this is the case, refer to the description of .gitignore below.
* Click the file icon for one or more files that appear under “Unstaged Changes”. git responds by moving these names to the lower section.
* If you stage a file by mistake, you can move it back to the upper section “Unstaged Changes” by clicking its file icon.
* When you are satisfied that you have selected the desired set of staged files, (1) type an appropriate message into the **Initial Commit Message** text area – a message such as ‘Initial project version’, then (2) click the **Commit** command button.

All files in the repository are eligible to be tracked. If your repository contains a large number of files, it might be appropriate to use .gitignore. “.gitignore” is the name of a text file that lists the files that you do not want to track. .gitignore can be placed in any folder in the repository, and the documentation You can have more than one instance of .gitignore (in the root folder and in subfolders). At this point I find it convenient to have .gitignore in the root folder of the repository. An example of a .gitignore file is the one used in a repository for a Visual Studio solution that contains 2 projects – OdeToFood.Data and OdeToFood.Web . . .

\*packages/

\*.vs/

\*OdeToFood.Data/bin/

\*OdeToFood.Data/obj/

\*OdeToFood.Web/bin/

\*OdeToFood.Web/obj/

I don’t want to perform revision control for any file in a folder whose directory name ends with “packages”. I don’t want to perform revision control for any file in a folder whose directory name ends with “.vs”. I don’t want to perform revision control for “obj” or “bin” files for either of my projects.

Add Repository to Github

The following instructions are for initializing the repository in Github.

* Navigate to the Github website. Log in.
* There is a “+” icon in the upper-right corner of the web page. Click the down-arrow to the right of this icon. Github displays a popup menu.
* Click **New repository** in the popup menu. Github displays a web page titled **Create a new repository**. By default the **Owner** text box contains my user ID.
* Type a name in the **Repository name** text box. I find it convenient to type the name of the root folder of the repository in my home computer.
* Click the **Create reposity** command button.

Next we need to connect the repository on the local computer to the Github repository. Be prepared to type the SSH key passphrase.

* Start Git Bash. Navigate to the root folder of the repository.
* Type the following commands.

$ git remote add origin git@github.com:*GitHubUserID*/*GitHubRepositoryName*.git

$ git push –u origin master