**Developing for the Agy App**

There are a few tools that we use to create the development pipeline for the agy app. The first and most important is git.

<https://git-scm.com/>

You’ll need to download and install git. Most of what we’ll be doing will be through a PowerShell but I recommend also installing the Git GUI as well as it makes some of the steps a lot easier.

Another very important thing for the agy app is .NET Core

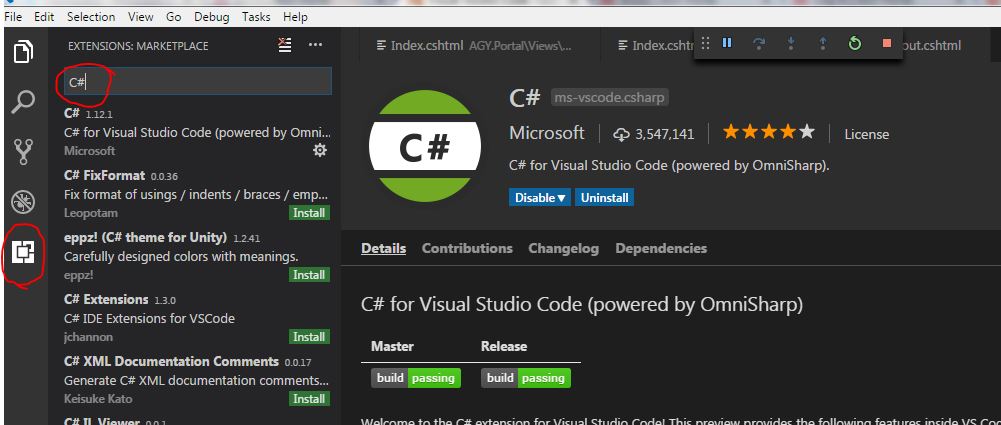
<https://www.microsoft.com/net/core#windowscmd>

From this link you can download the .NET Core SDK so that you have access to all the dotnet commands.

Visual Studio Code is what I have been using for this app. If you don’t already have it you can download it here.

<https://code.visualstudio.com/>

Visual Studio Code has an extension for C# for debugging so you will want to grab that.



All you do is open Code and select the bottom tab on the left and then type C# into the search bar. You should be able to click an install button and it will download and install the extension.

You’ll probably want to restart VSCode is not prompted to already

Once you have Git, .NET Core, and Visual Studio updated you can pull the repository.

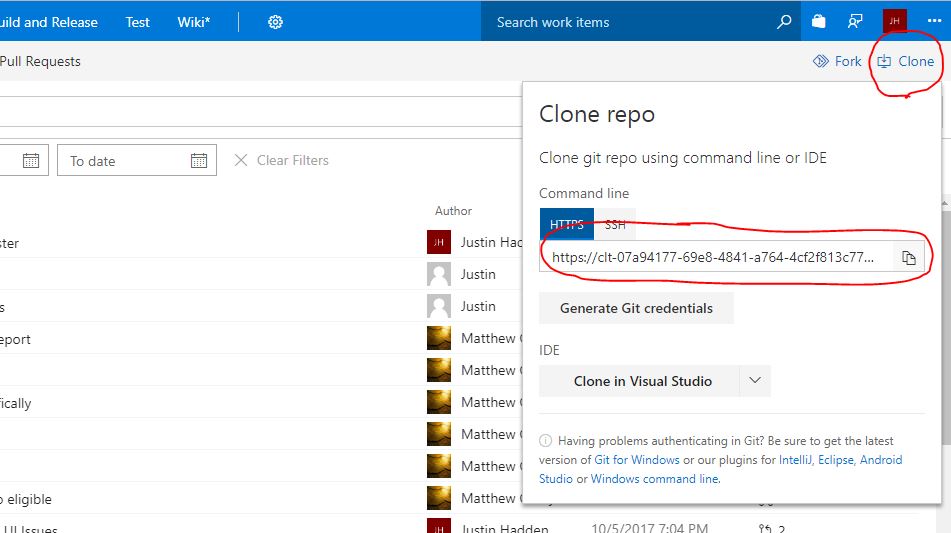
**Visual Studio Team Services**

To get access to the code you will need to have a Microsoft account and get me to add it to the project. Once you are a member of the project you can create branches, watch issues, and make pull requests to be deployed to the Azure portal.

We need to get set up with teamservices before we can continue.

**Using Git to get the AgyApp repository**

To get the code repository all you need to do is pull it with git. The link that you need to clone the repository is accessed by clicking the clone button in the top right corner of the AgyApp Project Page.



1. Create a folder where you want to store the EVOSS code
2. Open the folder with Visual Studio Code
3. Once in the folder with VSCode look for the terminal at the bottom of the screen. If you do not see the terminal press Ctrl + `.
4. Once in the terminal type: git init
5. Now we need to create a name for the remote repository. Type: git remote add origin <Insert Clone Link Here>
6. Now that we have the remote we reference it with ‘origin’ because that’s what we named it
7. Type: git pull origin master 🡨 Pulls the master branch from the remote repository
8. You will be prompted to give credentials which are just your teamservices credentials.

You now have all the code.

To make sure that your commits are not going into the origin master the first thing you should do is create your own branch git checkout -b <your name> Now that you have your own branch, anything you commit will just be to your branch.

To commit changes you’ve made to the code:

1. First see if you have anything to commit: git status
2. If you see files flagged as unstaged you can either add them individually or add the whole project. To add the whole project type: git add -A To add files individually type: git add <file path>
3. Things that you “add” are considered staged. Type: git commit –m “<write a comment>” to commit anything currently staged
4. Your branch has now committed whatever files you added. To push them to teamservices we’ll use the remote again. Type: git push origin <your branch name> This creates a new branch on teamservices next to the master. This way we can manage code more easily and then merge the branches into master when we think we have a good bit of code.

**Using Visual Studio Code with .NET Core**

The first time that you open the EVOSS folder with visual studio code you will be prompted to create a debug file so that Visual Code can run the application. It will create a .vscode folder at the top of all the other folders and contain 2 .json files. These tell Visual Code where to look for the csproj.

Next, we need to make sure that we have all of the applications dependencies.

1. In the terminal, make sure you are in the AGY.Portal folder. If not then type: cd agy.portal
2. type: dotnet restore

That should update all dependencies allowing you to run the application with F5.

**Other things that could help**

* Something to access the azure database with to make sure things are going in correctly
  + SSM
  + Visual Studio Server Manager
* Notepad++ to handle any merge conflicts from git
* Gitlab could be useful for handling issues. You could have the code repo on gitlab and handle all the issues, pull requests, and development branches and then push to either a Teamservices development branch or a release branch. Teamservices would then automatically deploy the app.
* Currently we use bower to handle frontend dependencies. You’ll need npm to get bower. \*If you don’t add anymore frontend dependencies then you will not be required to use bower.

**Useful Tutorials**

To learn more about using VSCode and .NET Core MVC: <https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app-xplat/start-mvc>