1. Create “Class Library” in Visual Studio.
2. Add code you want to be in your library.
3. Copy the following from NugetTemplate folder to root folder of your project:

Folder: “Scripts”

Files: ”.gitignore”, “.travis.yml” , “LICENCE”, “README.md”

1. Create new empty repository on github. Note repository’s URL for <projectUrl> in the next step.
2. Create YourProjectName.nuspec file in the root folder and add the following:

<?xml version="1.0"?>

<package >

<metadata>

<!-- The unique identifier for the package. This is the package name that is shown when packages are listed using the Package Manager Console. These are also used when installing a package using the Install-Package command within the Package Manager Console. Package IDs may not contain any spaces or characters that are invalid in an URL. In general, they follow the same rules as .NET namespaces do. So Foo.Bar is a valid ID, Foo! and Foo Bar are not. -->

<id>YourProjectNameWithNoSpaces</id>

<version>1.0.0.0</version>

<title>Your Project Name</title>

<authors>First name Last name</authors>

<owners>Intergen Ltd</owners>

<licenseUrl>http://www.apache.org/licenses/LICENSE-2.0</licenseUrl>

<projectUrl>YourGithubURL</projectUrl>

<iconUrl>https://avatars2.githubusercontent.com/u/17167744</iconUrl>

<requireLicenseAcceptance>false</requireLicenseAcceptance>

<description>Description of the library</description>

<releaseNotes>Updates in this build, can be left blank</releaseNotes>

<copyright>Copyright 2016</copyright>

<tags>Add tag, can be left blank</tags>

</metadata>

<files>

<file src=" YourProjectName/bin/Release/\*.dll" target="lib" />

<file src=" YourProjectName/bin/Release/\*.pdb" target="lib" />

</files>

</package>

1. Replace everything highlighted in red and green. You might need to add dependencies if your project requires them. Make sure to read Licence file and sign it at the end (line 189) or pick a licence that suits your needs.

Intergen use: Do not modify lines highlighted in green. Licence is already signed.

1. Follow these instructions to push local repository to git repository:

<https://help.github.com/articles/adding-an-existing-project-to-github-using-the-command-line/>

**git init**

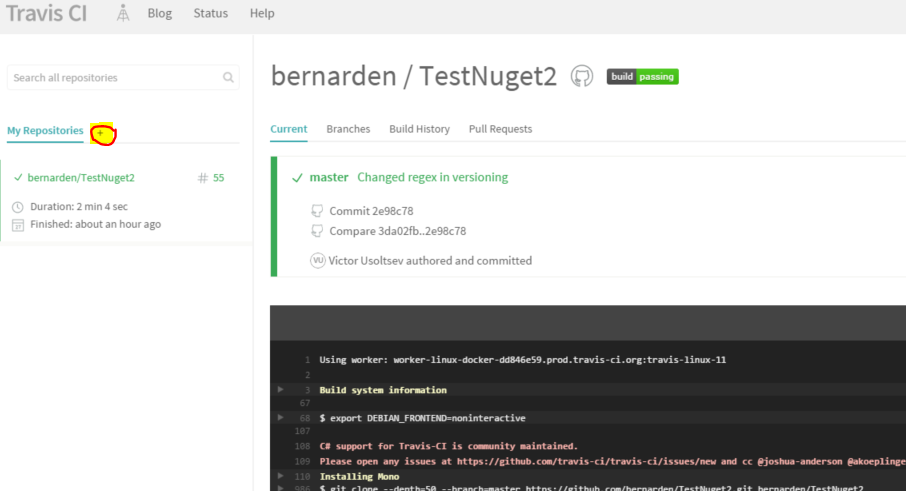
**git add .**

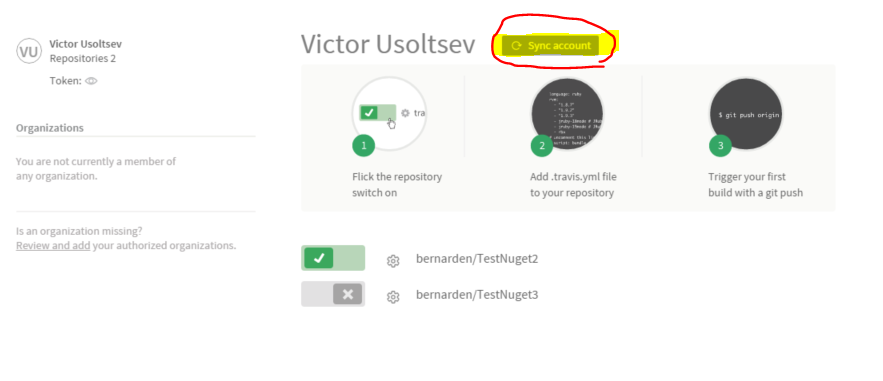
**git commit -m "Created project"**

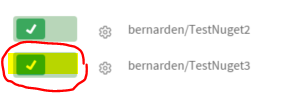
**git remote add origin YourGithubURL**

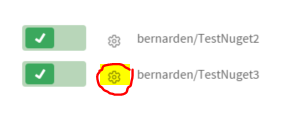
**git push origin master**

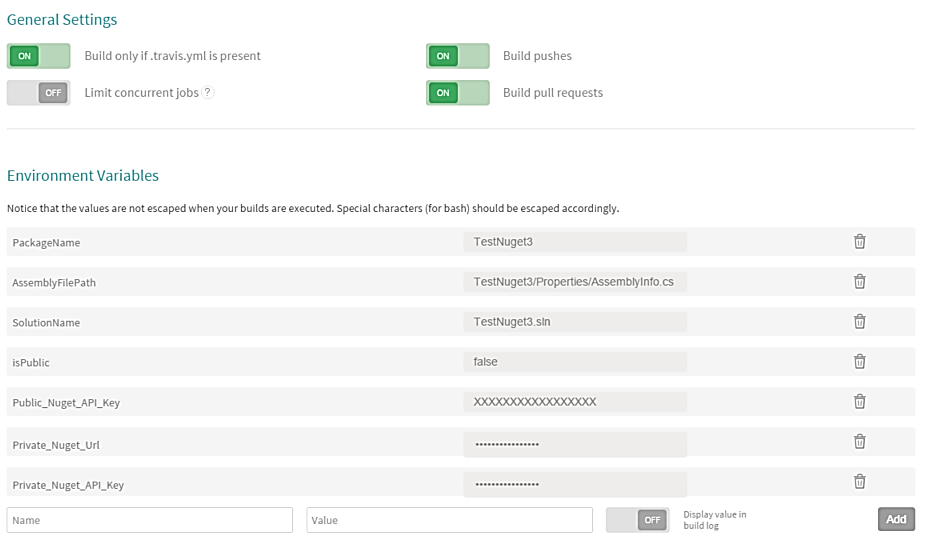
1. Login in <https://travis-ci.org/> with GitHub credentials. (allow it to access your repositories)
2. Sync your repositories on travis-ci and turn the switch on







1. Open settings for the new project then set switches and environment variables as follows:



|  |  |  |
| --- | --- | --- |
| Name | Value | isVisible |
| PackageName | YourProjectName | Yes |
| AssemblyFilePath | YourProjectName/Properties/AssemblyInfo.cs | Yes |
| SolutionName | YourProjectName.sln | Yes |
| isPublic | false | Yes |
| Public\_Nuget\_API\_Key | YourNuget.OrgAPIKey | No |
| Private\_Nuget\_Url | PrivateURL | No |
| Private\_Nuget\_API\_Key | PrivateAPIKey | No |

Note: Highlighted text is project and/or user specific.

Note: isPublic = false will only publish to private server, isPublic = true will publish to private and public servers. Modify scripts for different behaviour.

1. Update your project’s readme and push it to the repository.

(Any push to the repository will start a new build on travis-ci)

1. Done. Your NuGet should be published and your build state should be green.

All future build versions (major and minor) should only be adjusted in AssemblyInfo.cs

Build number is set by Travis-CI automatically.