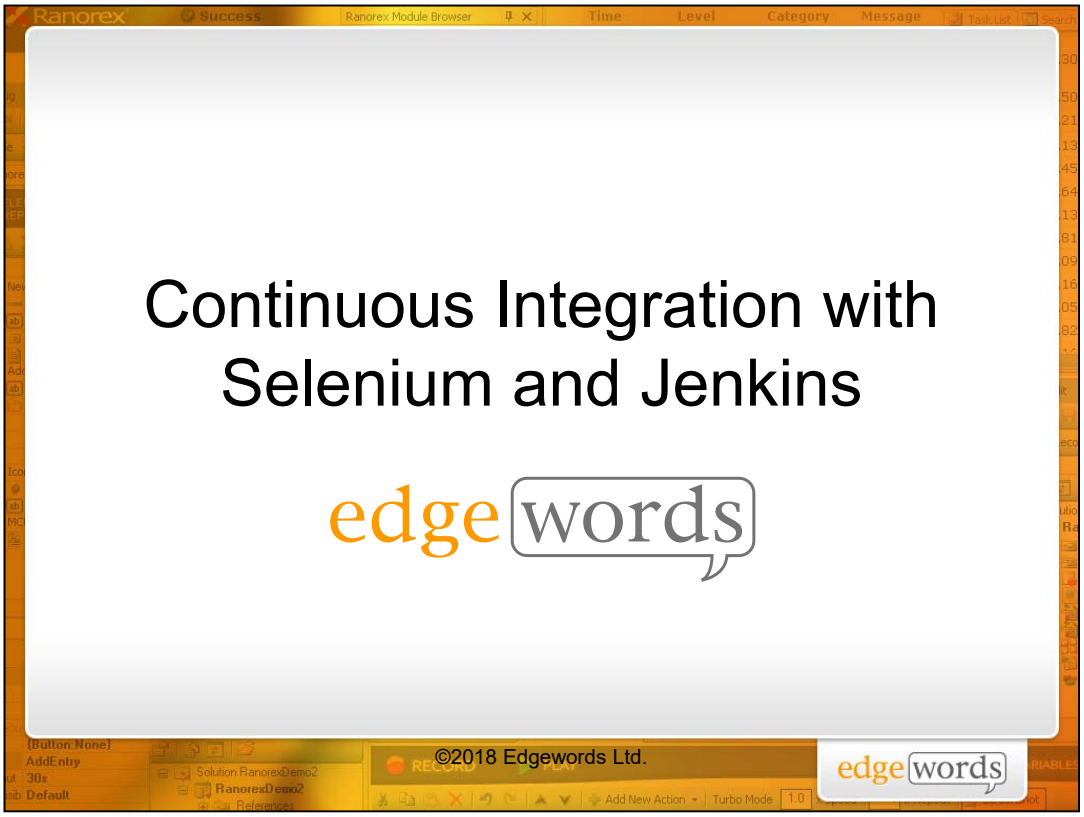
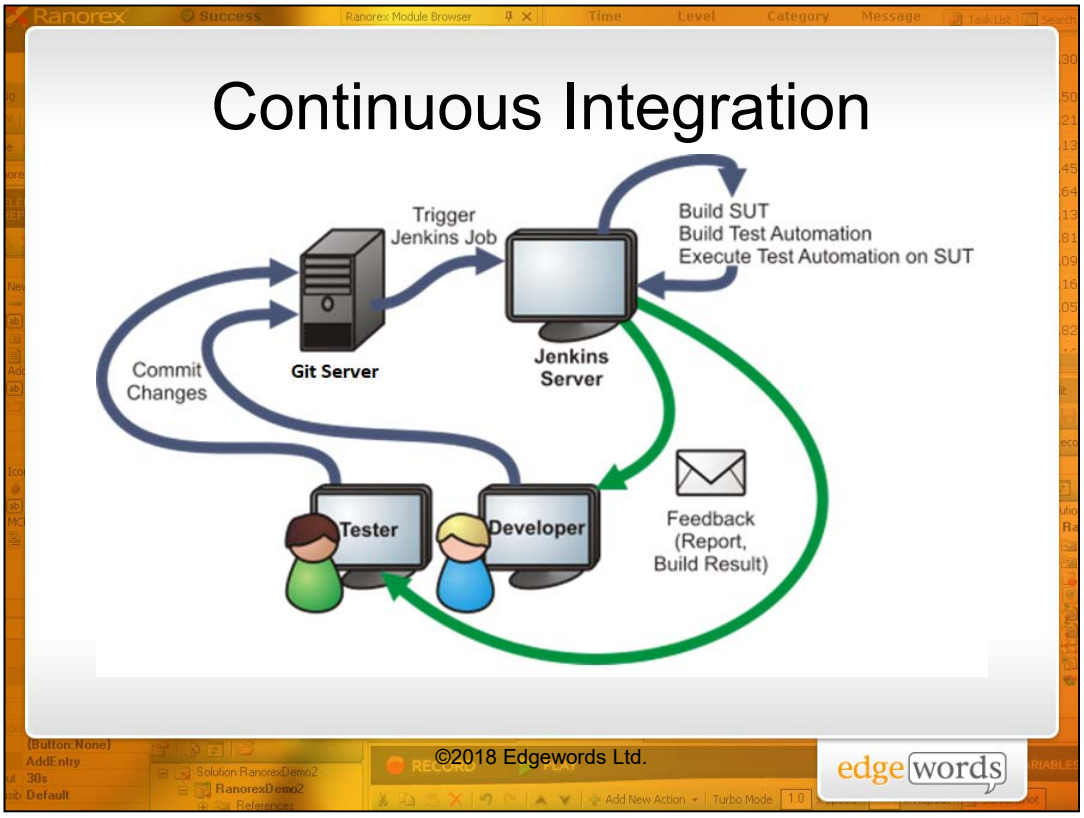
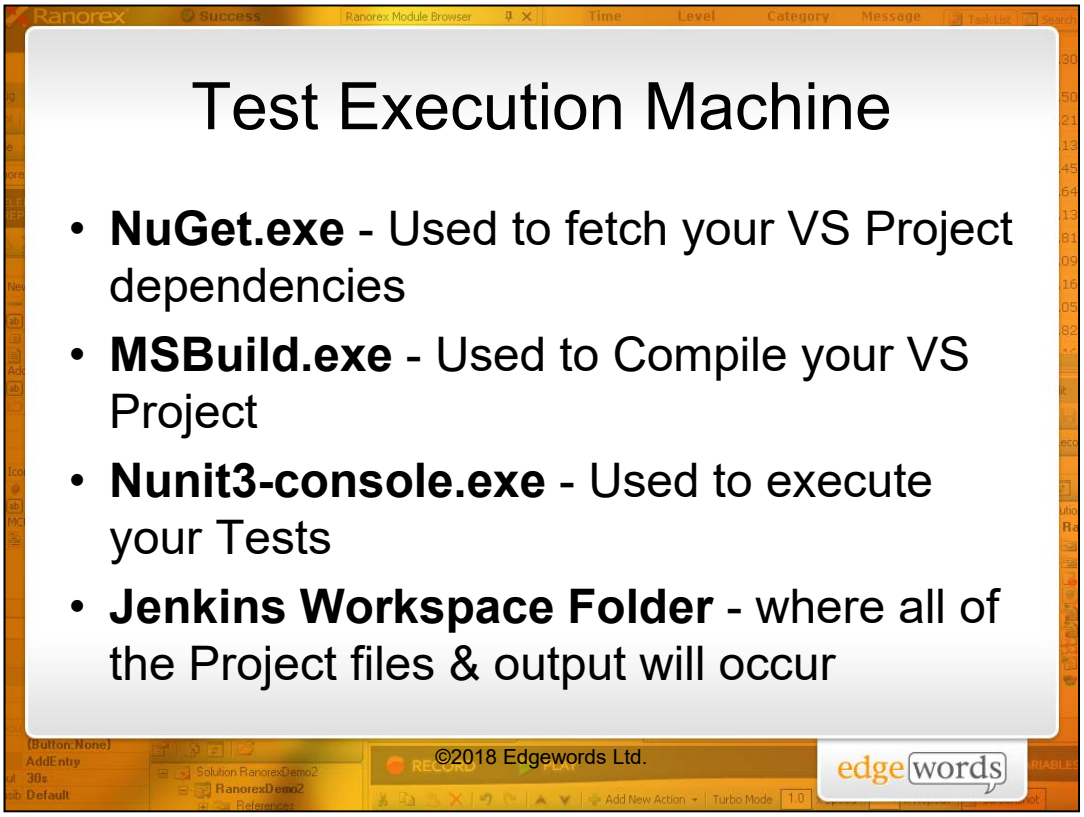


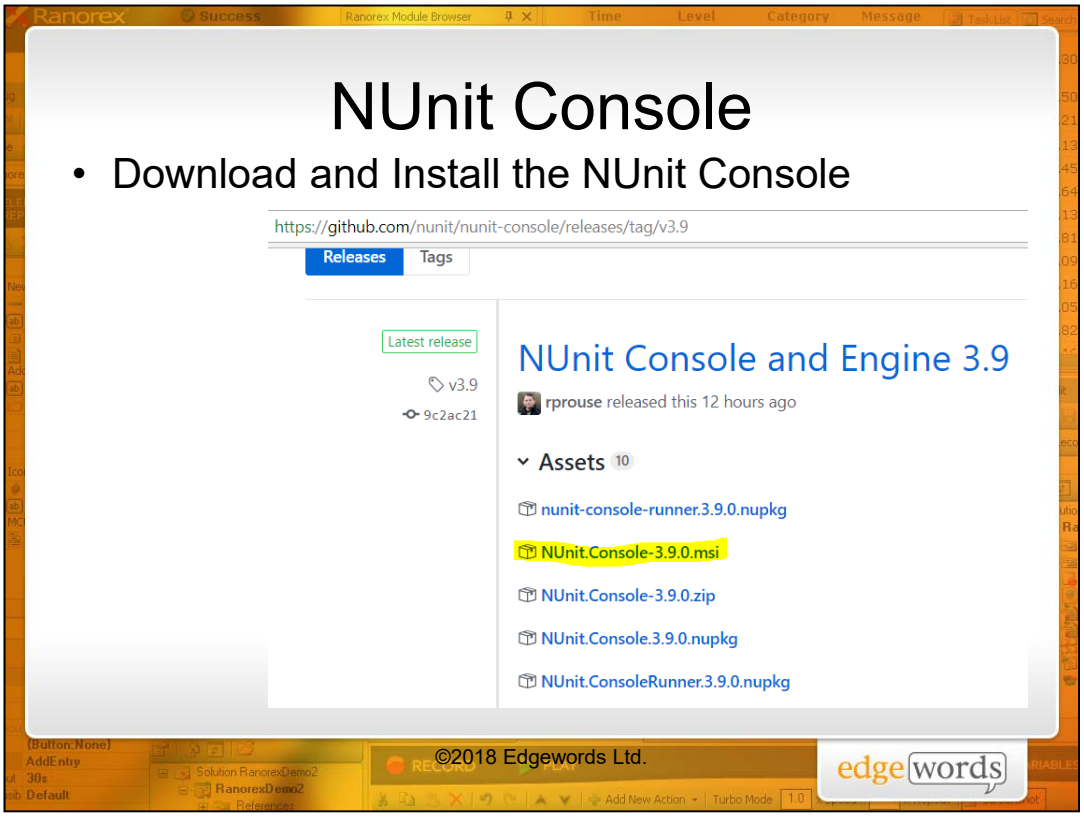


Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, and then automated testing allowing teams to detect problems early.

So for example, if we are delivering .NET applications and using WebDriver/NUnit to test them; the process maybe that the developer code is changed, this is committed to a Git Server. Then a C.I. tool such as Jenkins polls the Git server for changes, if detected, it compiles the new code, compiles our WebDriver tests, invokes the new version of the application, then executes our WebDriver Tests, and finally emails the test results to us. All unattended & automated.



Download NuGet.exe and add this to your system path
https://www.nuget.org/downloads



https://github.com/nunit/nunit-console/releases

Download NUnit Console msi installer, and install the package.

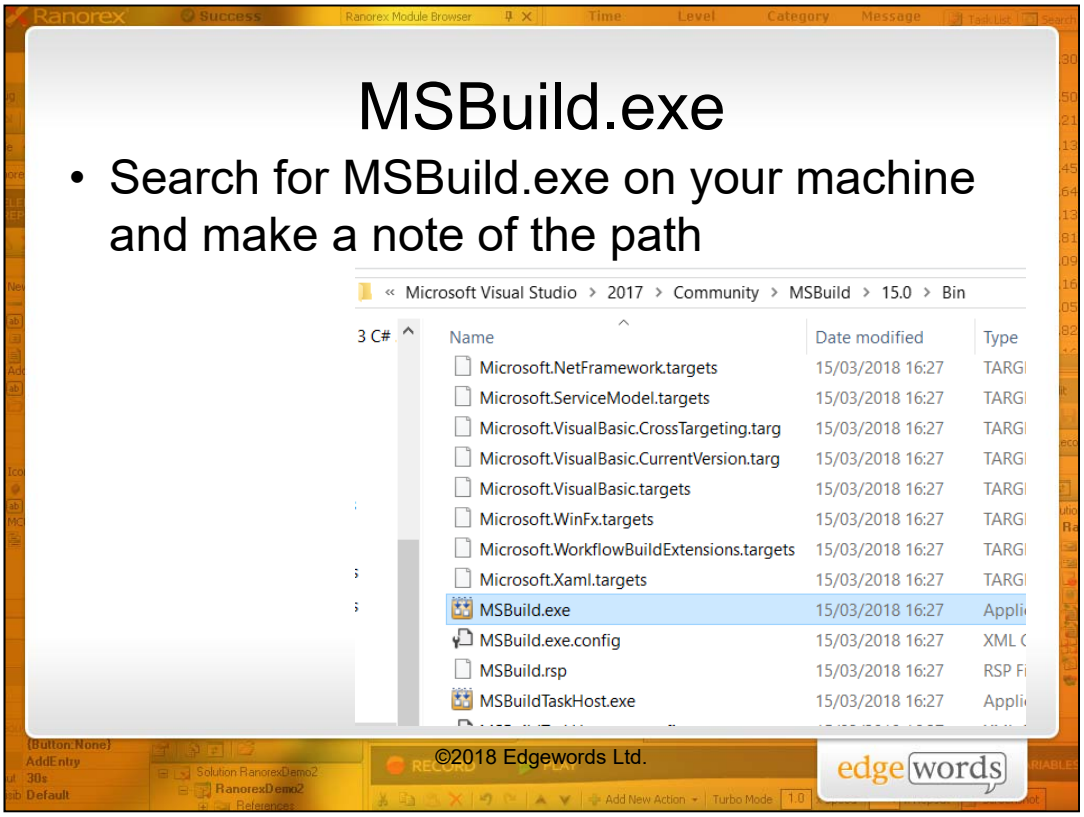
Ensure that the path to nunit3-console.exe is added to your system path

There are two options for Assemblies through NuGet that allow us powerful command line execution:

- NUnit.Console
- NUnit.ConsoleRunner

Either option will work with Jenkins, however Console comes with many useful extra tools for reporting formats

So install NUnit.Console NuGet package to your VS Project.



If you have VS installed then one way to find out where MSBuild.exe is installed on your machine is to click the Windows Start button and type ‘developer command prompt’

And look at the path it is set to.

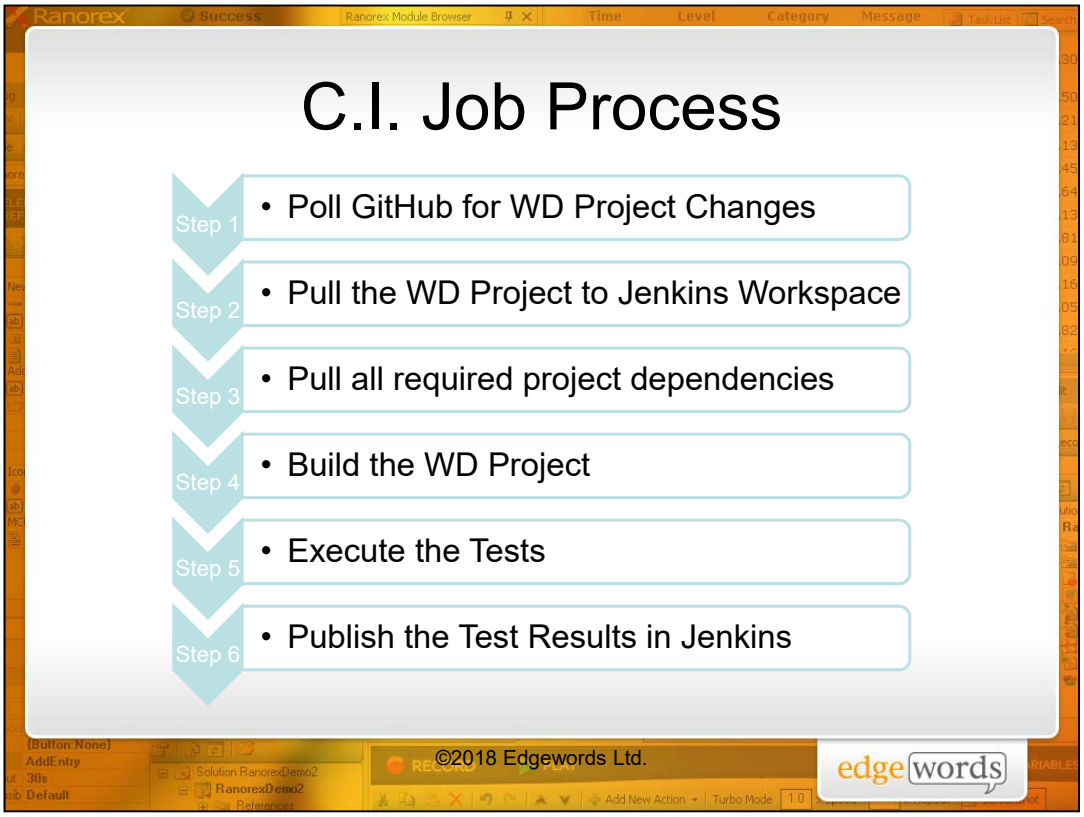
The path for .NET framework installation is usually

C:\Windows\Microsoft.NET\Framework[64 or empty][framework_version]

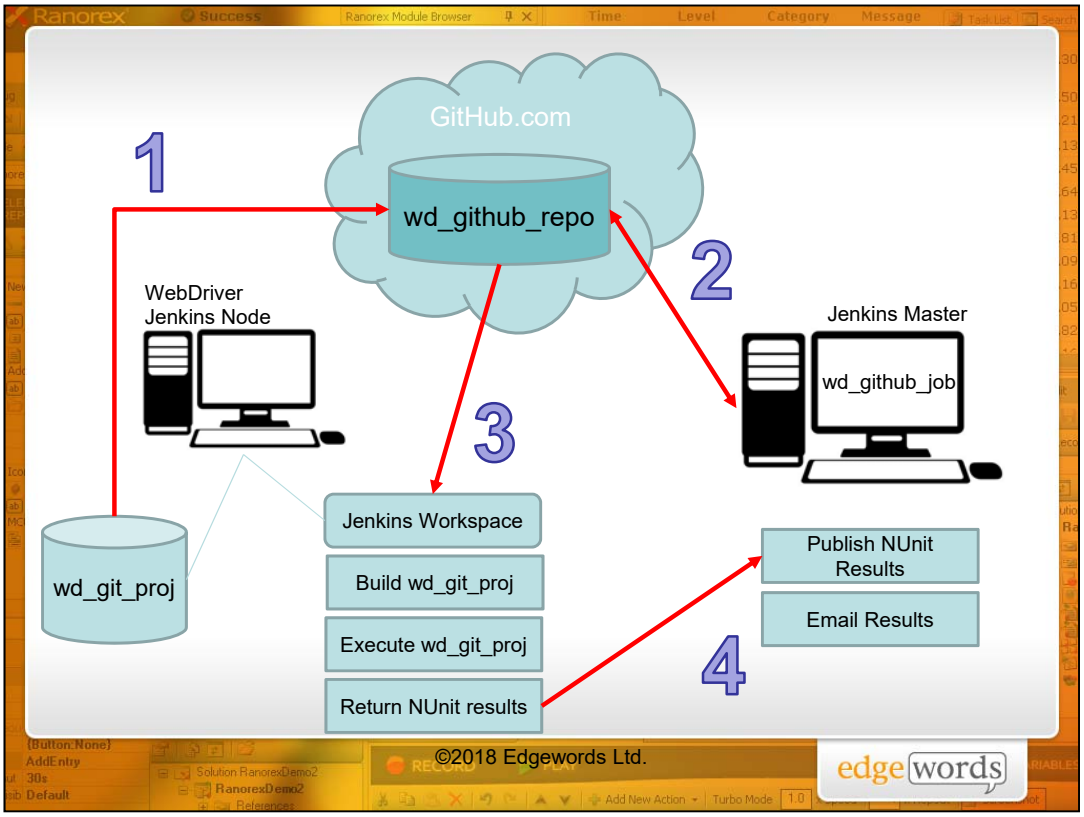
The path when Visual Studio is installed is

C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\MSBuild\15.0\Bin

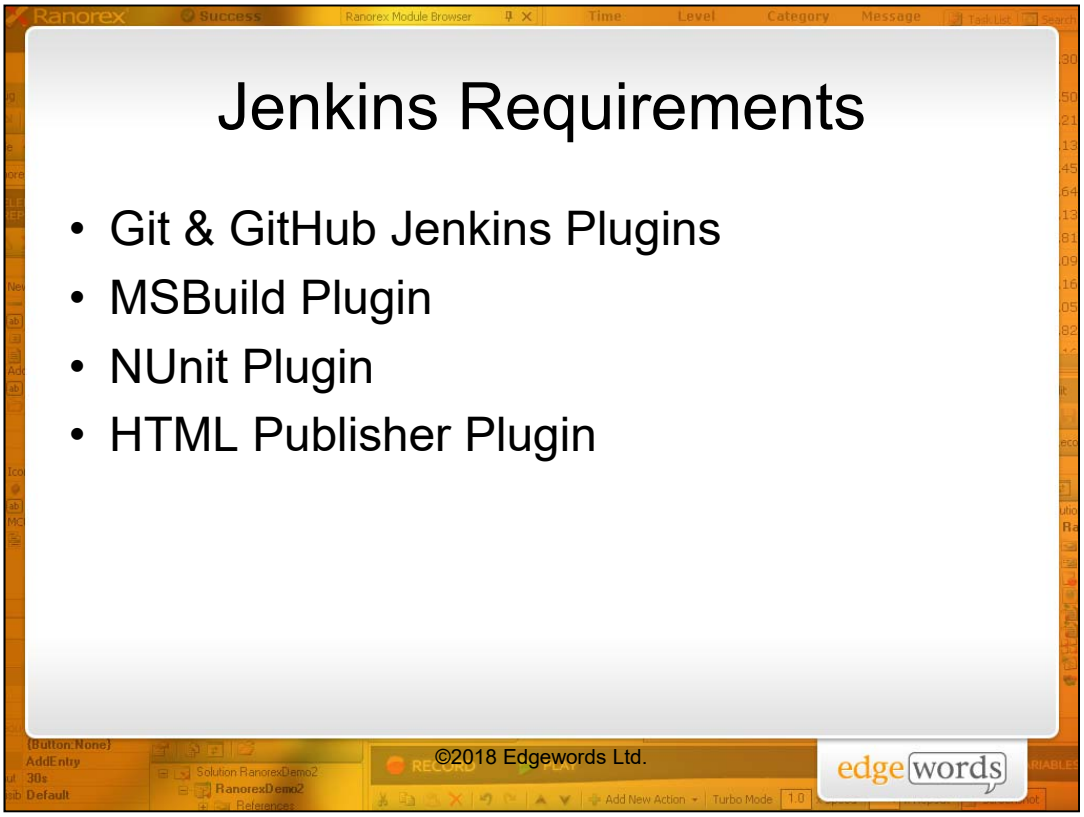
Again you could add this to your system path if you want



In this example we are going to create a Jenkins Job that performs a number of steps.

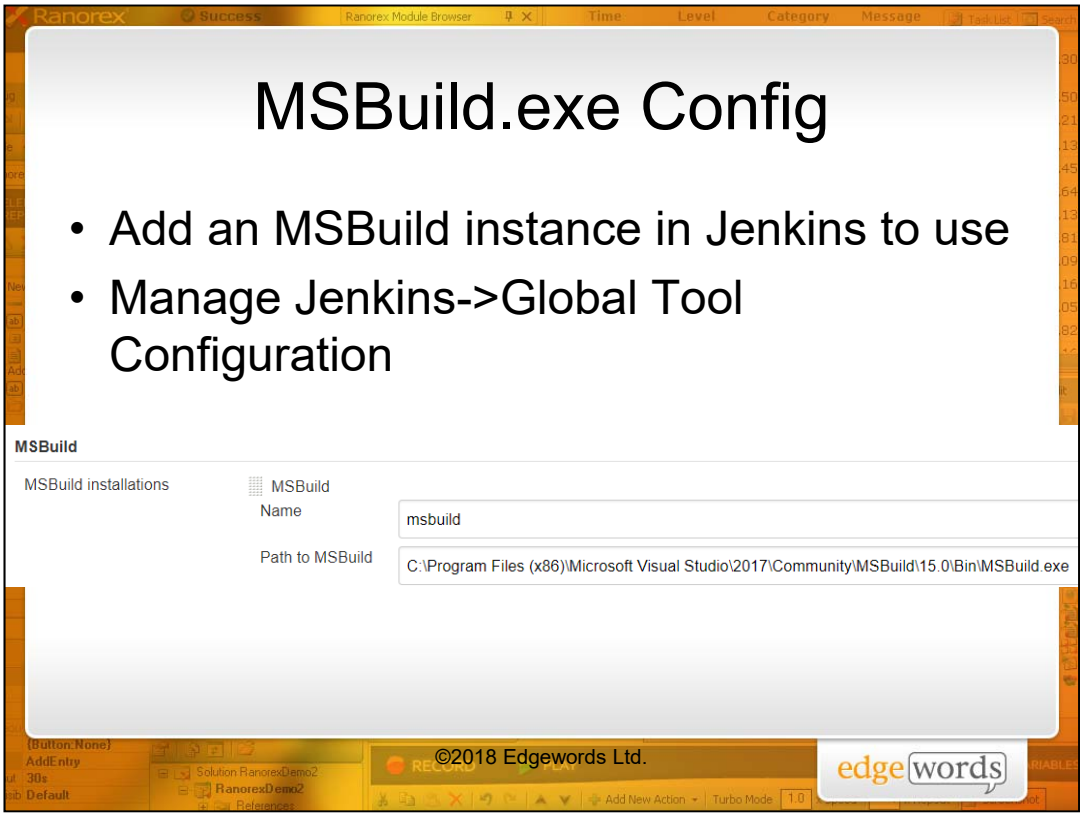


1. Push your WebDriver Project up to GitHub
2. Jenkins Polls the GitHub Repo for changes
3. If there is a change, then Jenkins Pulls the Project from GitHub down to the local Jenkins Workspace, Our Jenkins Job then performs the build steps:
    1. Build the .NET project in the Jenkins Workspace using MSBuild.exe
    2. Execute the Compiled Project using NUnit.Console
    3. return the NUnit Test Results to Jenkins
4. Jenkins Publishes the Test Results, and Emails them to interested parties



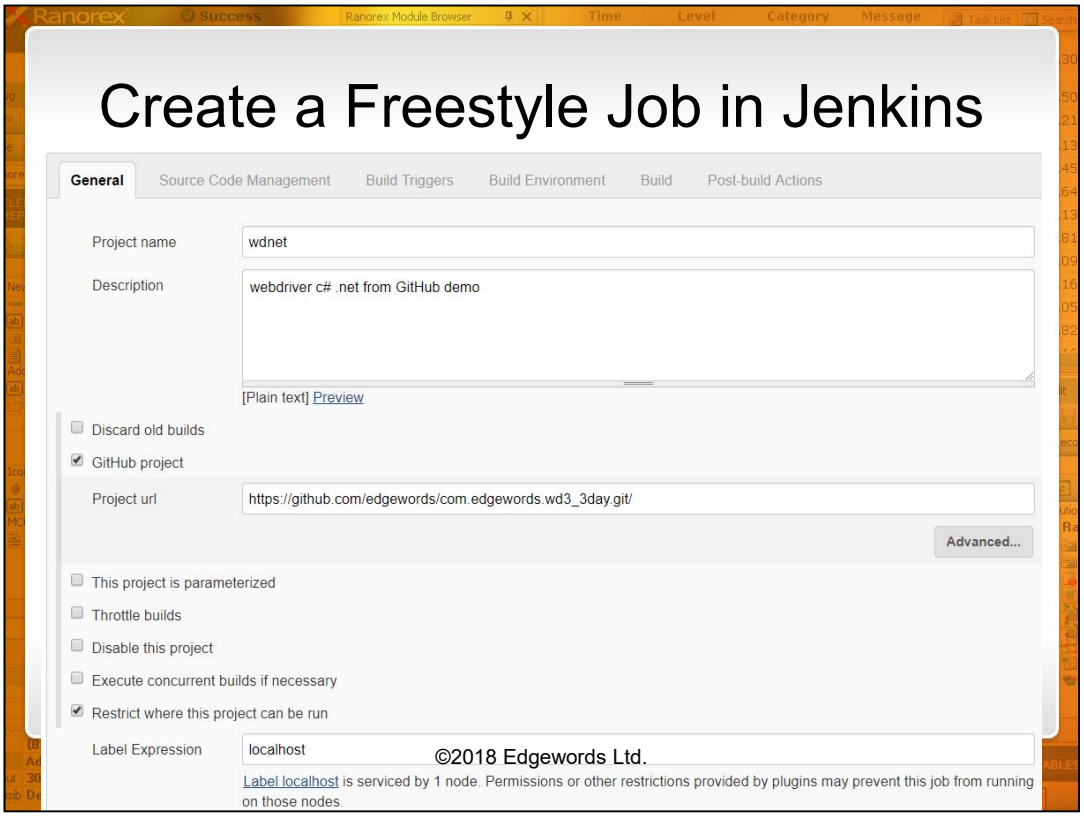
Install all of the Jenkins Plugins above.

To do this in Jenkins go to Manage Jenkins->Manage Plugins, click on the available tab, and then search for the names of the plugins to install. Select them and click install without restart

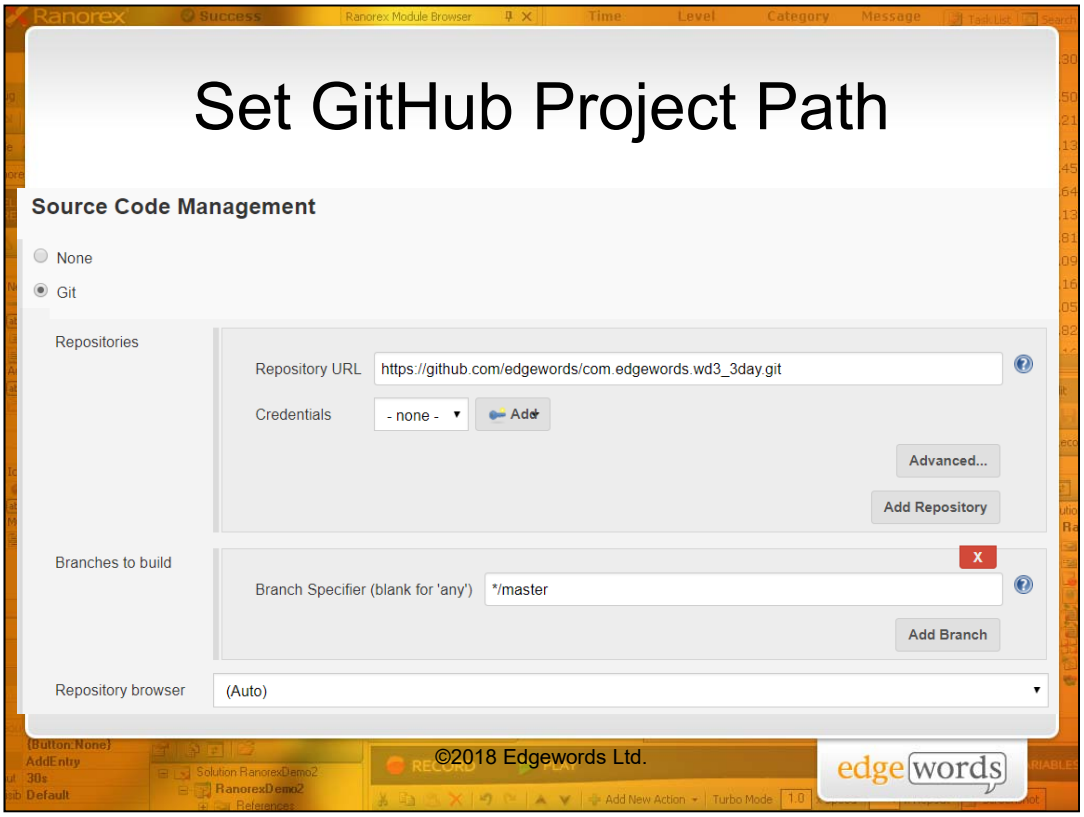


You need to tell Jenkins where MSBuild.exe is on the Test machine.

You can then override this for each Node, by going into the Node configuration, checking the Tool Locations checkbox, from the drop-down that then appears, choose MSBuild and then set the path to MSBuild.exe for that particular Node

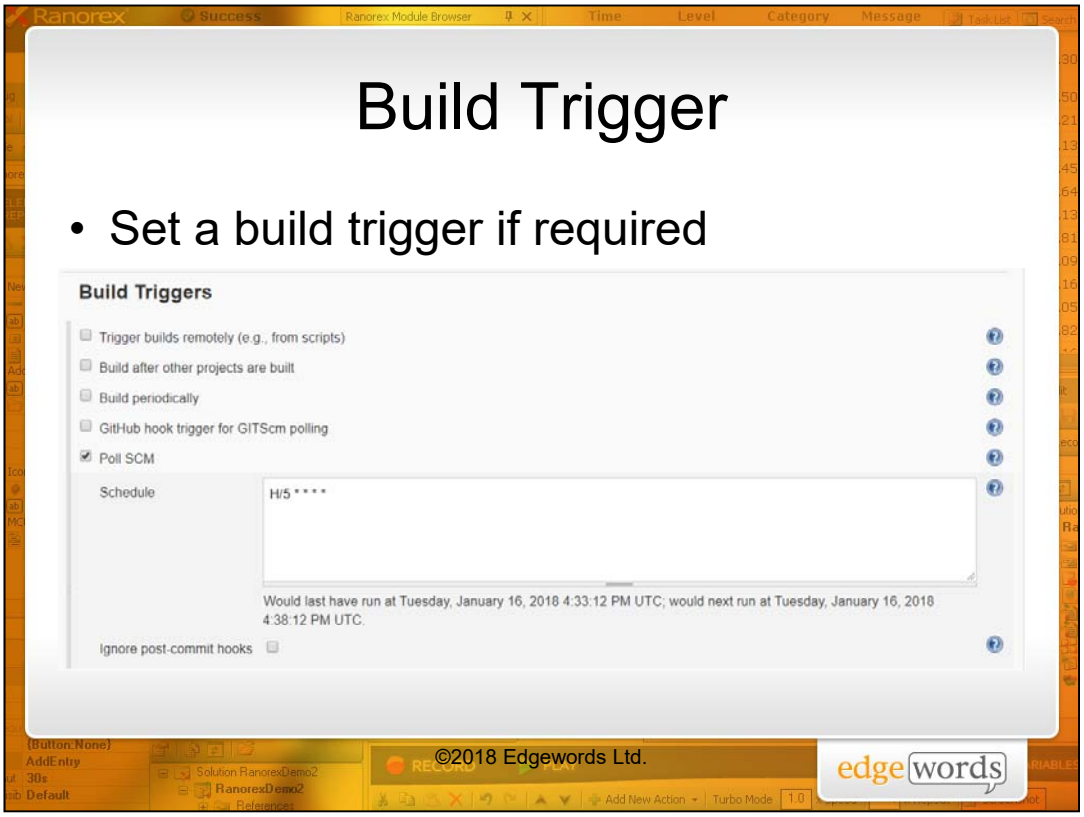


Now configure a new Freestyle Job in Jenkins,.

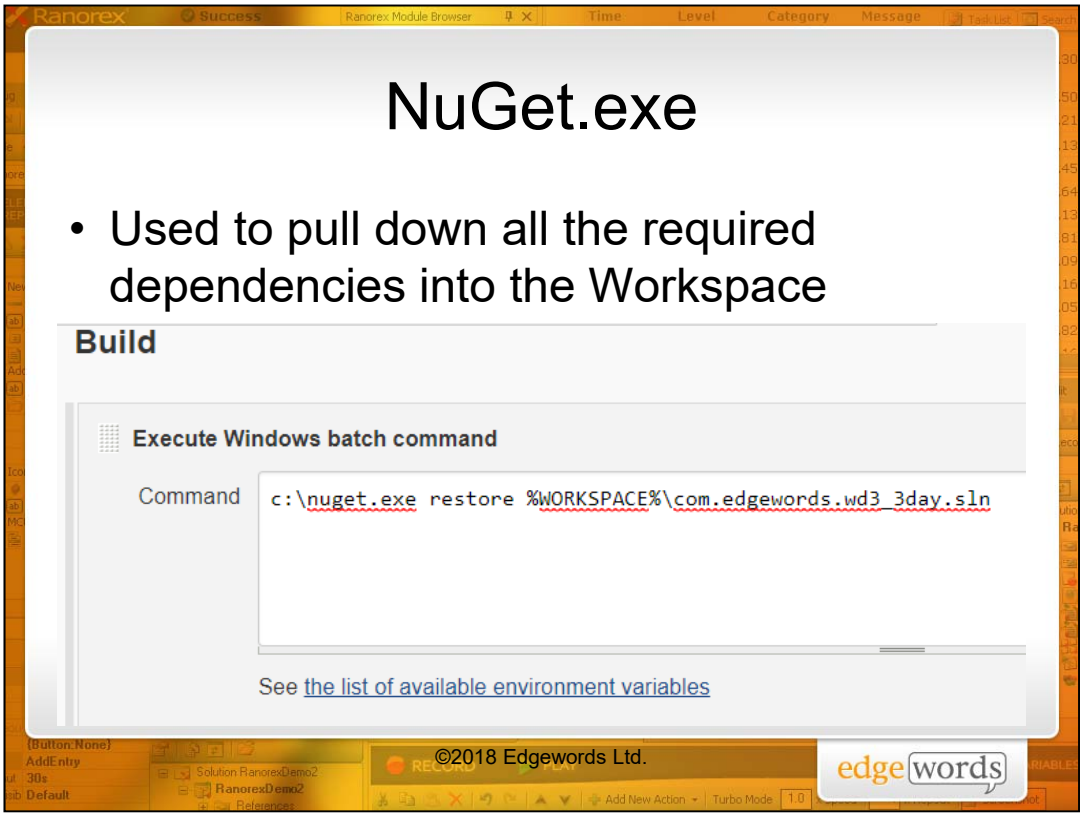


Provide the Git Hub repository URL that we wish to monitor (if local then use file protocol: file:///c:/my/path/to/project)

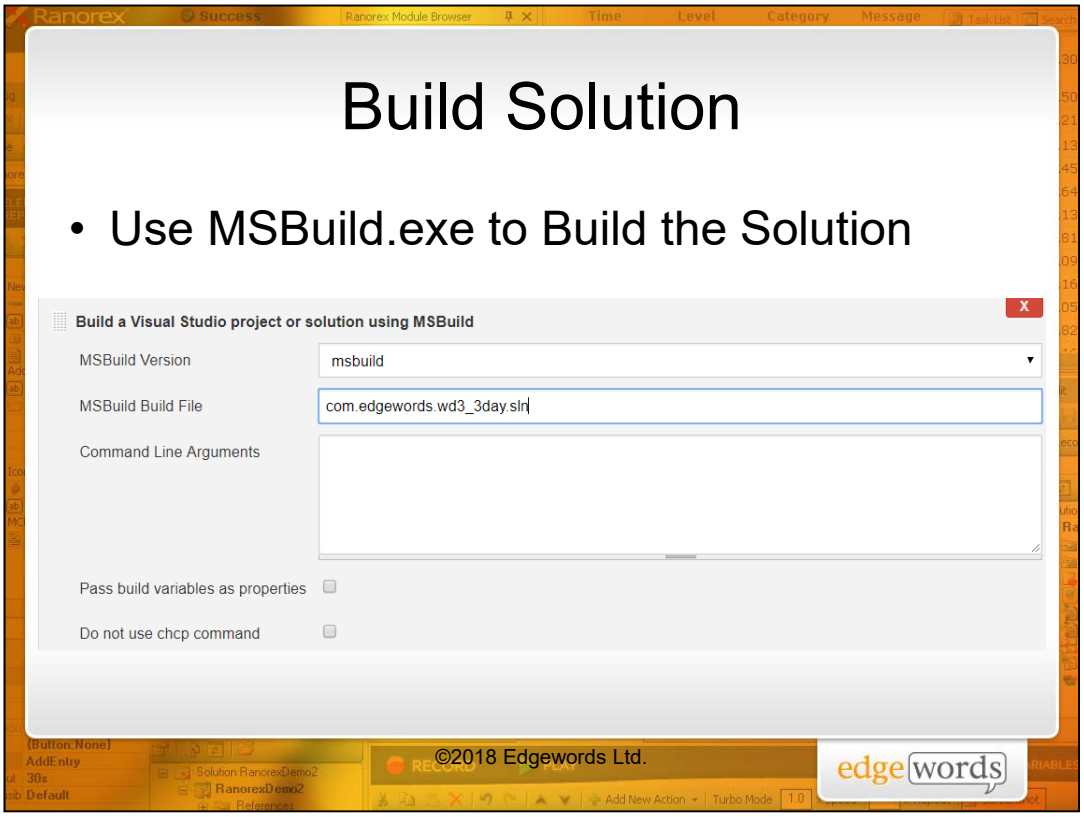
This will automatically pull the GitHub repository to your Jenkins workspace



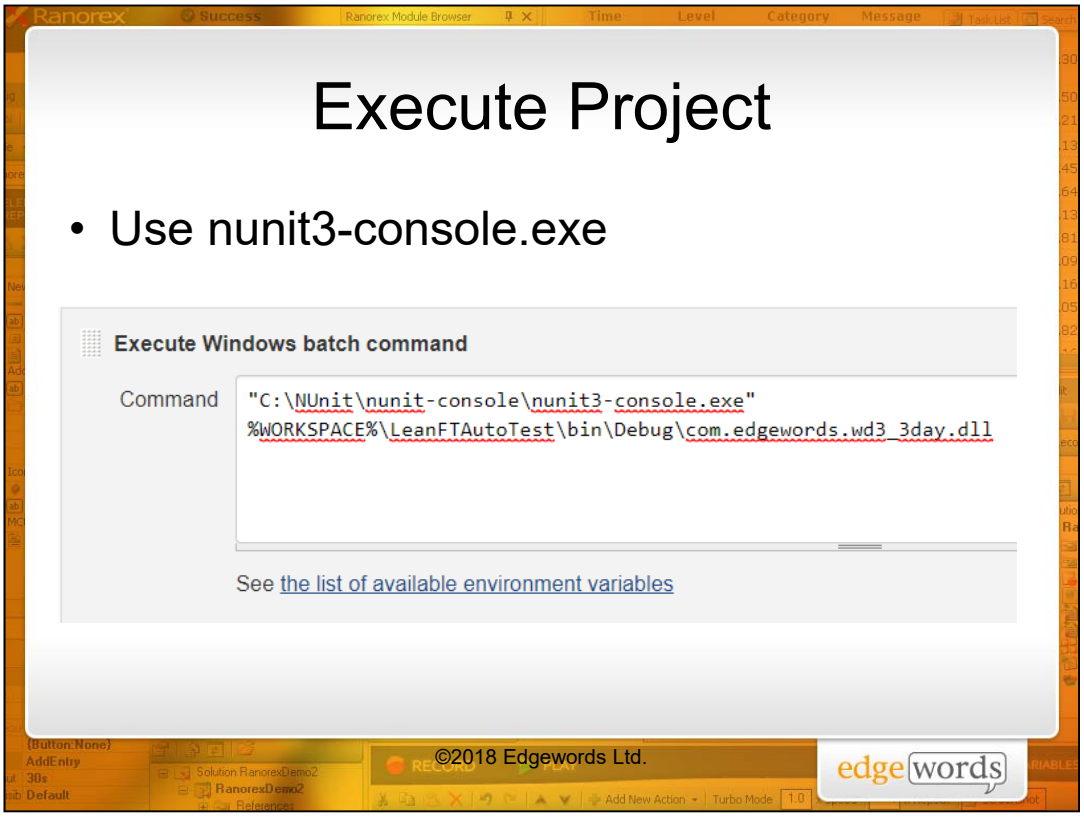
In the example above, this tells Jenkins to Poll the GitHub Repository every 5 minutes for changes



Ensure that NuGet.exe is installed on the build machine. This is used to download and .NET project dependencies required for the project into the Jenkins Workspace, so we can then build the Project successfully

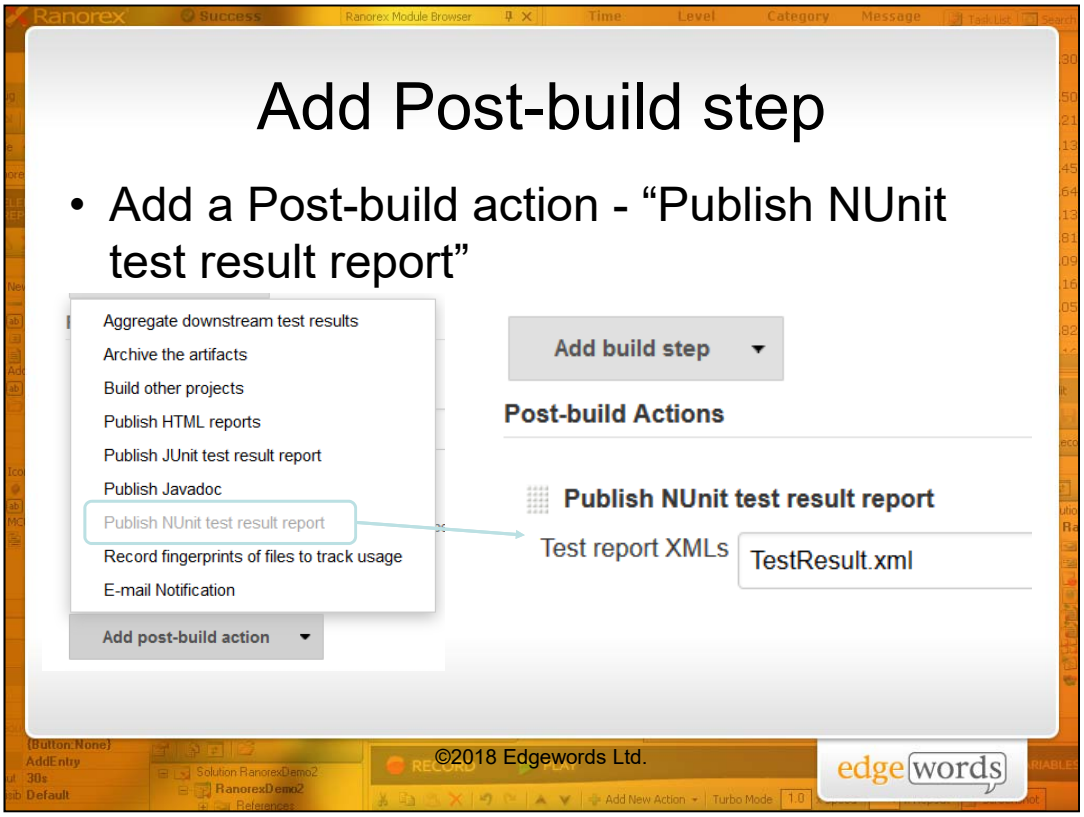


Now we use MSBuild.exe to build our Visual Studio Project



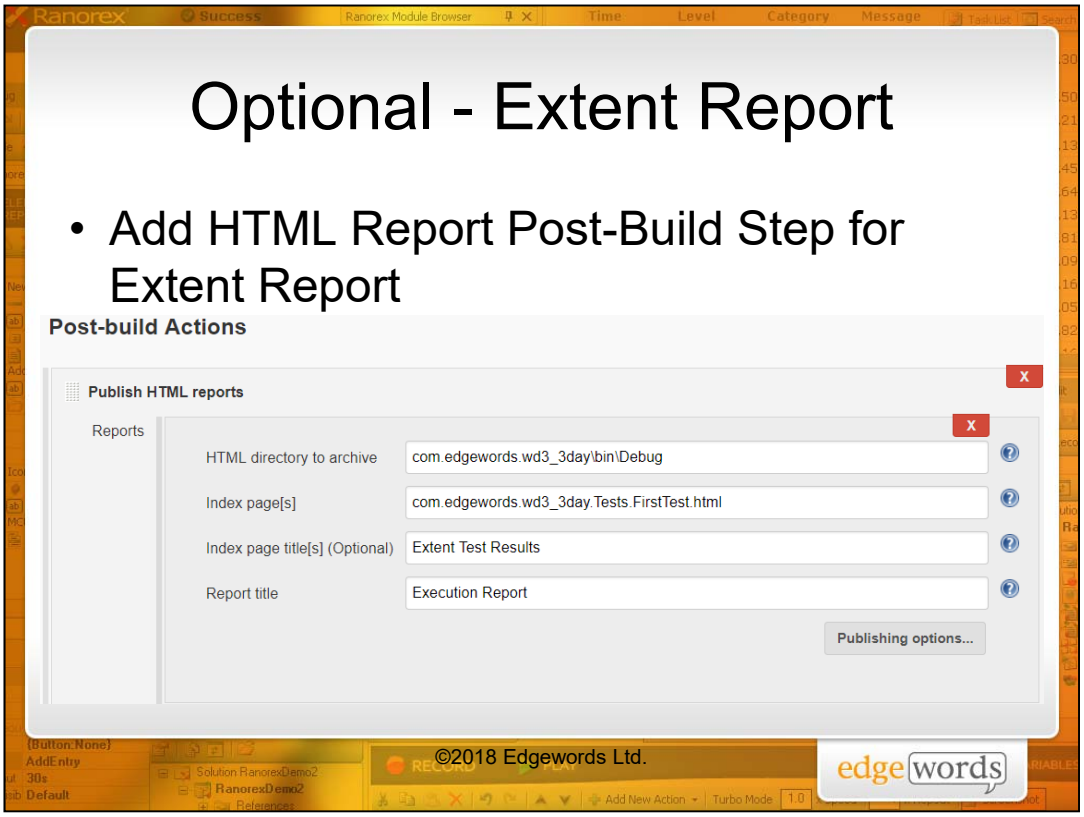
We then use nunit3-console.exe to execute the Test Suite.

When we built the project, the output was a .DLL assembly which we pass to MSBuild.exe



Finally add a Post-build step - “Publish NUnit test result report” and put the name of the Test Results .xml file (TestResult.xml is the default)

The “Publish NUnit test result report” will only appear in the actions drop-down if you have correctly installed the NUnit Jenkins plugin.



If you are using Extent Reports to create HTML results, then as long as you have installed the Jenkins HTML Publisher plug-in, you can publish your extent report.

NOTE: Due to security in Jenkins, if html reports are not showing up, try this fix:

To view html results you need to override a security setting in Jenkins. To do this, go to Manage Jenkins->Script console and type in the following command:

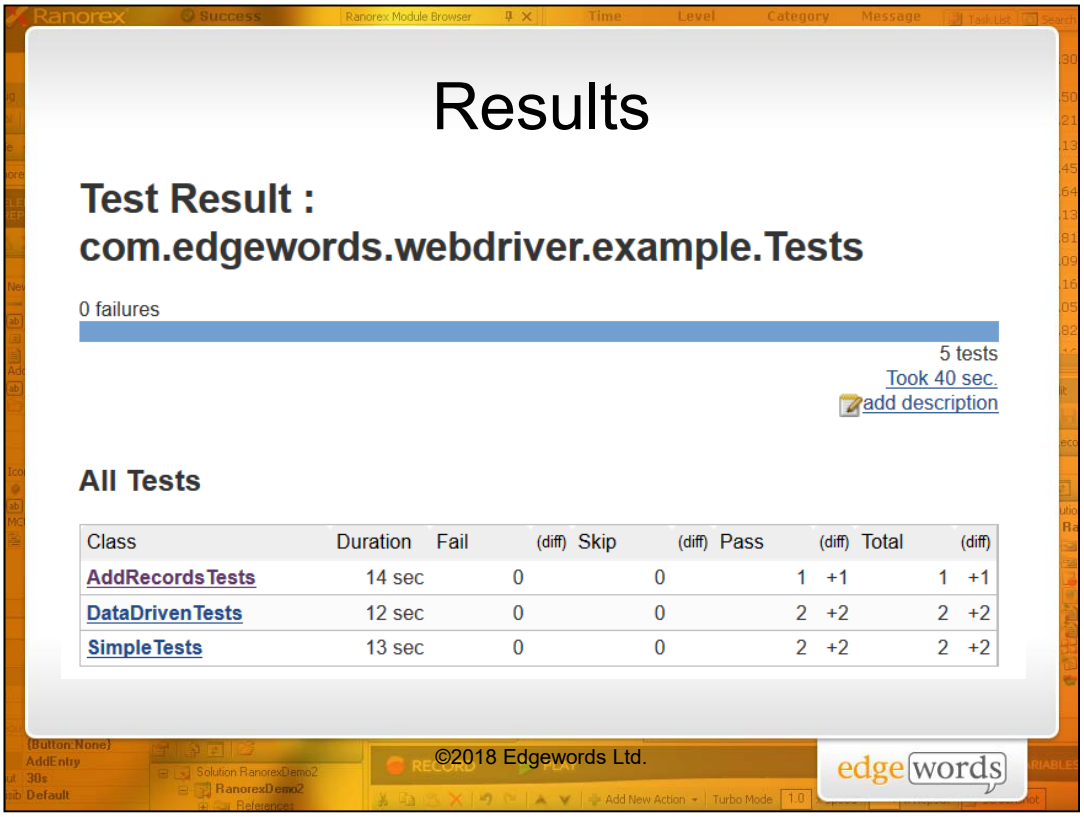
System.setProperty("hudson.model.DirectoryBrowserSupport.CSP", "")

And click run

Save the Jenkins Job, and click the Build Now button to test the job.

Once executed, from the Build History, click the last results link and there should be an option - “Test Result”

Select this to view the published NUnit results.



The Test Results will be displayed, and you can click the links to drill down through the report!