

The image shows a screenshot of the Ranorex software interface. The main window displays a presentation slide with the following content:

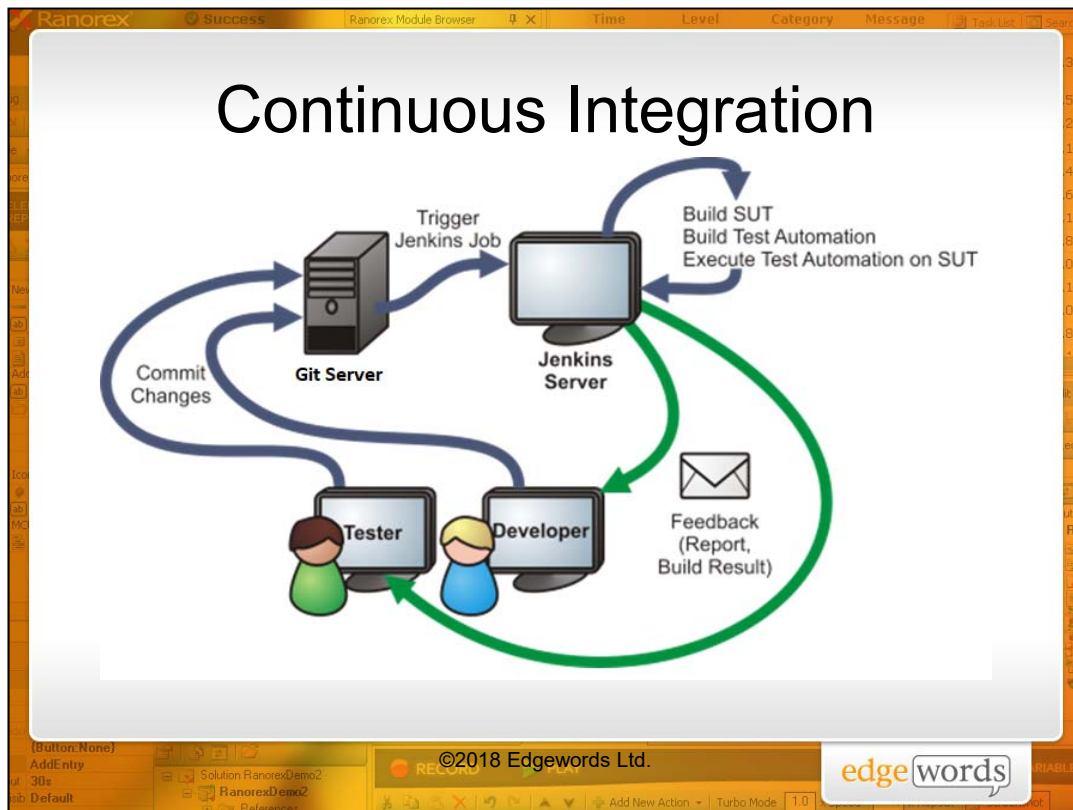
Continuous Integration with Selenium and Jenkins

edge words

©2018 Edgewords Ltd.

edge words

The interface includes a top status bar with 'Ranorex', 'Success', 'Ranorex Module Browser', 'Time', 'Level', 'Category', and 'Message'. The bottom status bar shows 'Add Entry', '30s', 'Default', 'Solution RanorexDemo2', 'RanorexDemo2', 'References', 'Add New Action', 'Turbo Mode', and '1.0'. The 'edge words' logo is also present in the bottom right corner of the slide area.



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.

Test Execution Machine

- **NuGet.exe** - Used to fetch your VS Project dependencies
- **MSBuild.exe** - Used to Compile your VS Project
- **Nunit3-console.exe** - Used to execute your Tests
- **Jenkins Workspace Folder** - where all of the Project files & output will occur

©2018 Edgewords Ltd.

edgewords

The screenshot shows a web browser window displaying the NuGet website. The page title is "NuGet.exe". The browser address bar shows "https://www.nuget.org/downloads". The page content includes a search bar, navigation links for "Packages", "Upload", and "Statistics", and a list of available NuGet distributions. The "Windows x86 Commandline" distribution is highlighted, showing "nuget.exe - recommended latest v4.7.1" and "nuget.exe v4.8.1".

NuGet.exe

- Download NuGet.exe and add to your System path

Available NuGet Distributions

Windows x86 Commandline

nuget.exe - recommended latest v4.7.1

nuget.exe v4.8.1

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

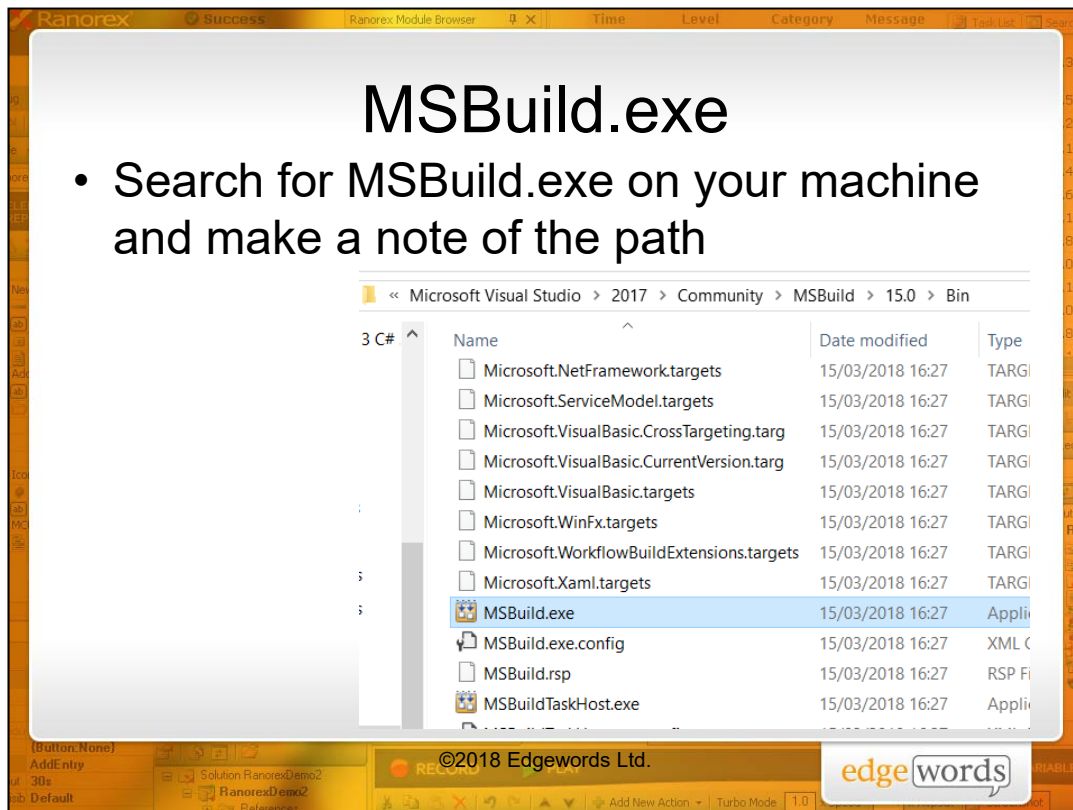


There are two options for Assemblies through NuGet that allow us powerful command line execution:

- 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

Ranorex Success Ranorex Module Browser X Time Level Category Message Test Log

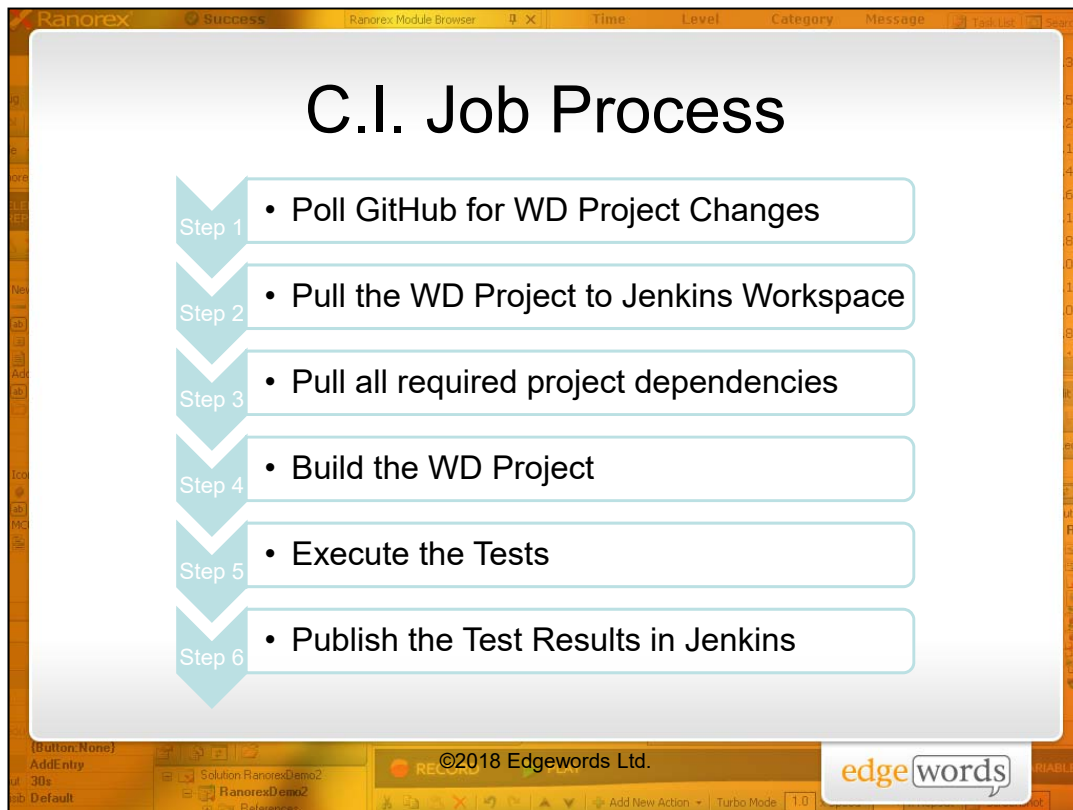
Part 2
JENKINS INTEGRATION

(button:None) AddEntry 30s Default Solution RanorexDemo2 RanorexDemo2 References ©2018 Edgewords Ltd. edgewords

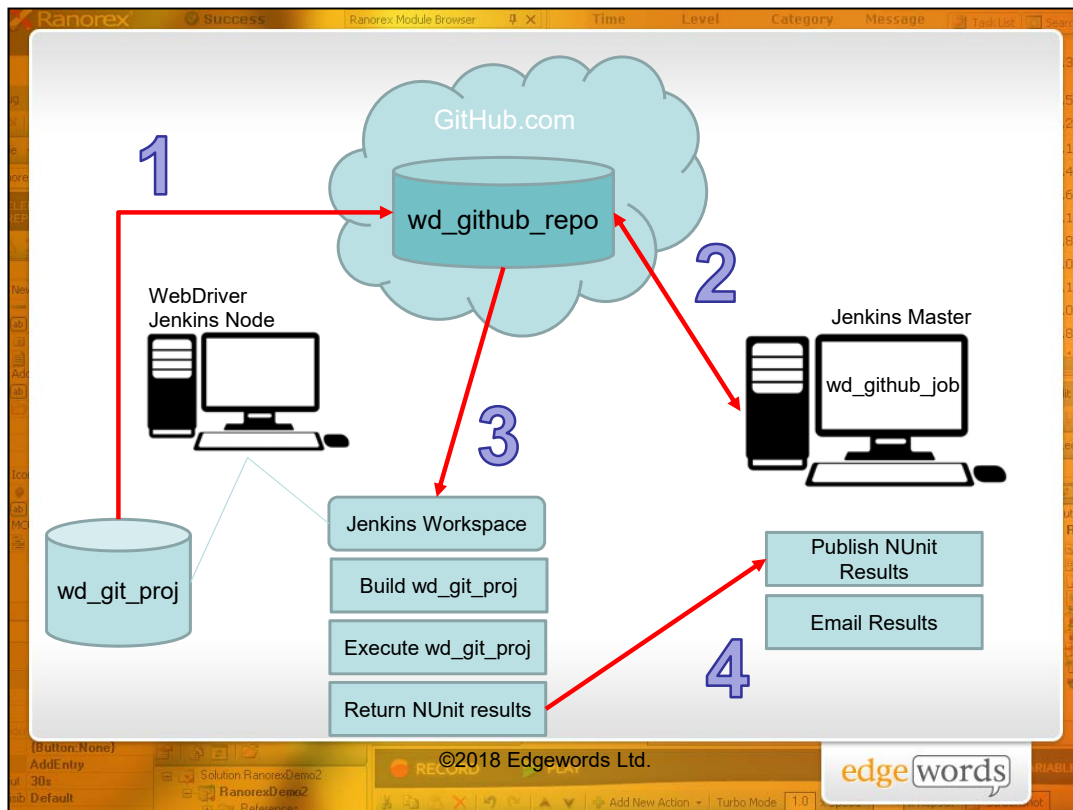
30 50 21 13 45 64 15 81 09 16 05 82 3.7 4.0 4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.0 5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 6.0 6.1 6.2 6.3 6.4 6.5 6.6 6.7 6.8 6.9 7.0 7.1 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9 8.0 8.1 8.2 8.3 8.4 8.5 8.6 8.7 8.8 8.9 9.0 9.1 9.2 9.3 9.4 9.5 9.6 9.7 9.8 9.9 10.0

RIABLES

Add New Action Turbo Mode 1.0



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

The image shows a screenshot of a presentation slide within the Ranorex software interface. The slide has a white background and a black border. At the top, the title 'Jenkins Requirements' is displayed in a large, bold, black font. Below the title, there is a bulleted list of four items: 'Git & GitHub Jenkins Plugins', 'MSBuild Plugin', 'JUnit Plugin', and 'HTML Publisher Plugin'. The slide is set against a background of the Ranorex application window, which includes a top status bar with 'Success' and 'Ranorex Module Browser' tabs, and a bottom toolbar with various icons and the text '©2018 Edgewords Ltd.' and 'edgewords' logo.

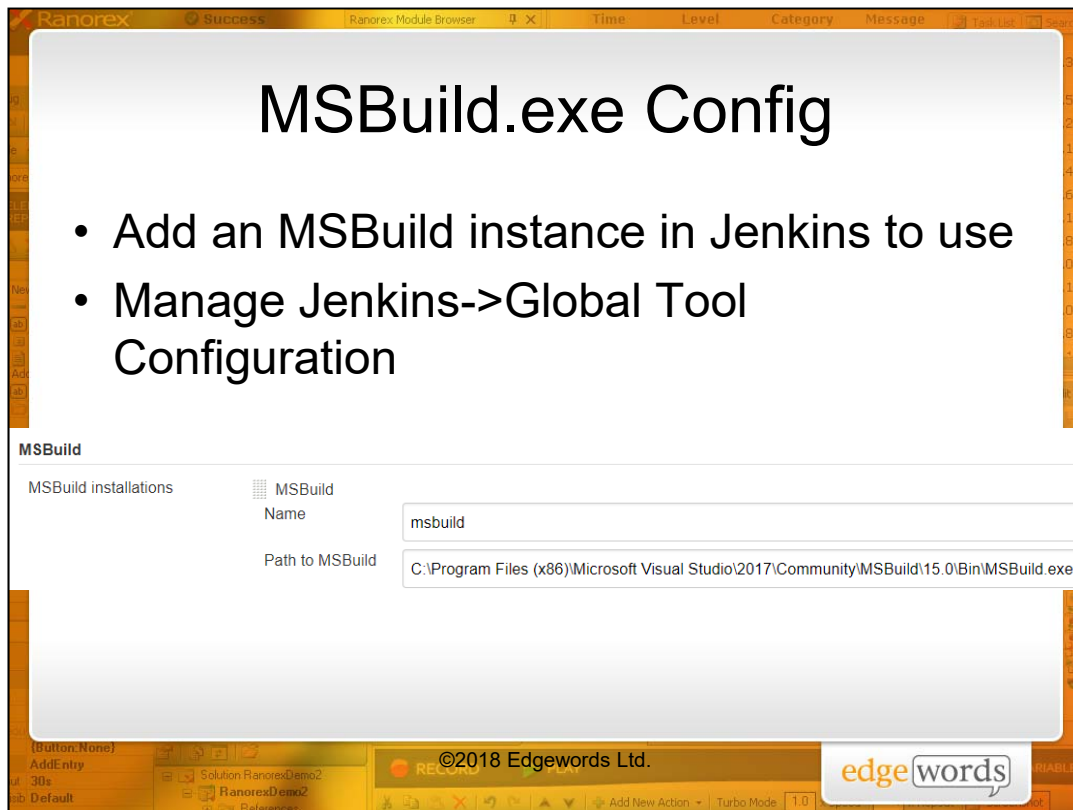
Jenkins Requirements

- Git & GitHub Jenkins Plugins
- MSBuild Plugin
- NUnit Plugin
- HTML Publisher Plugin

©2018 Edgewords Ltd. edgewords

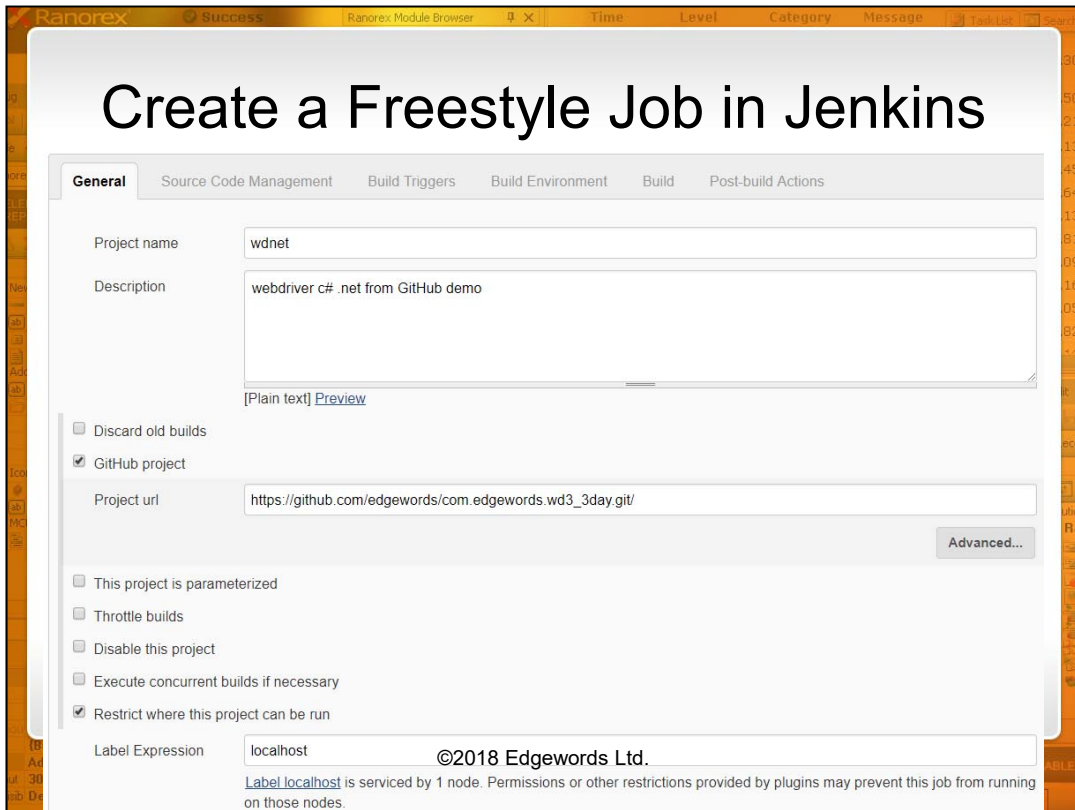
Install all of the Jenkins Plugins above.

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

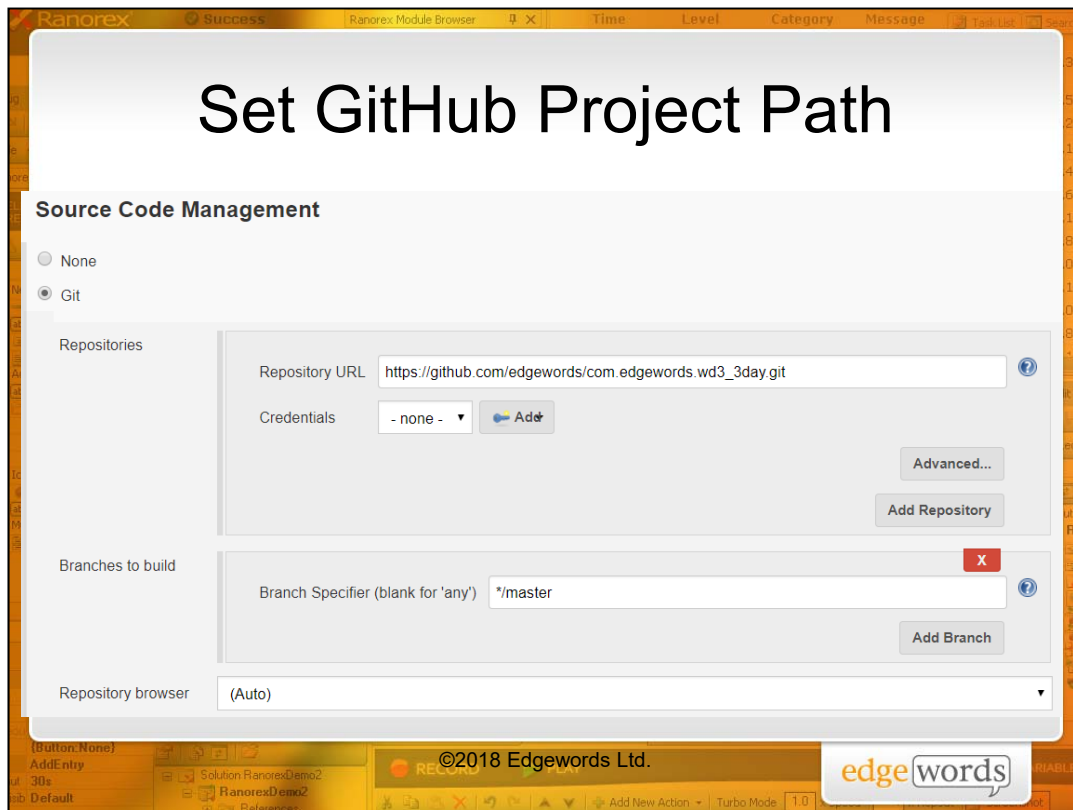


You need to tell Jenkins where MSBuild.exe is on the Test machine.

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

Build Trigger

- Set a build trigger if required

Build Triggers

- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built
- Build periodically
- GitHub hook trigger for GITScm polling
- Poll SCM

Schedule:

Would last have run at Tuesday, January 16, 2018 4:33:12 PM UTC; would next run at Tuesday, January 16, 2018 4:38:12 PM UTC.

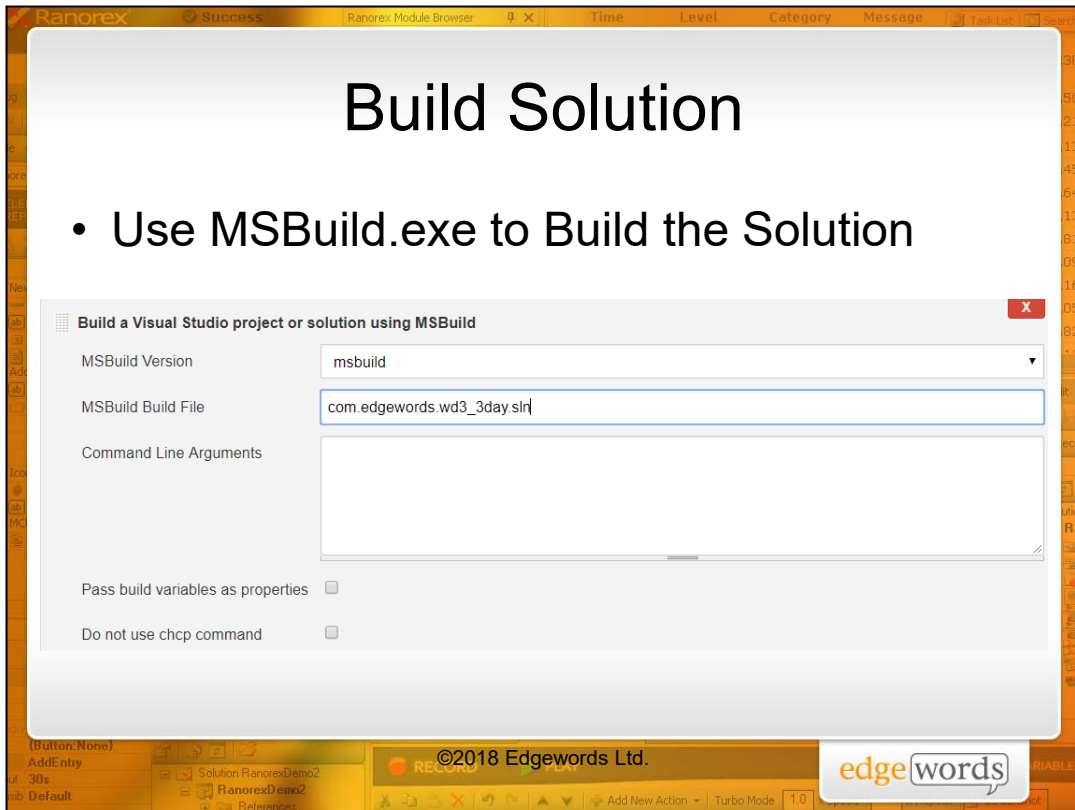
Ignore post-commit hooks

©2018 Edgewords Ltd. edge words

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

The screenshot shows a Ranorex build configuration window. At the top, the title is "NuGet.exe". Below it, a bullet point states: "Used to pull down all the required dependencies into the Workspace". The main configuration area is titled "Build" and contains a section for "Execute Windows batch command". Inside this section, there is a "Command" field with the following text: `c:\nuget.exe restore %WORKSPACE%\com.edgewords.wd3_3day.sln`. Below the command field, there is a link that says "See the list of available environment variables". The bottom of the window shows a footer with "©2018 Edgewords Ltd." and the "edgewords" logo.

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

Execute Project

- Use `nunit3-console.exe`

Execute Windows batch command

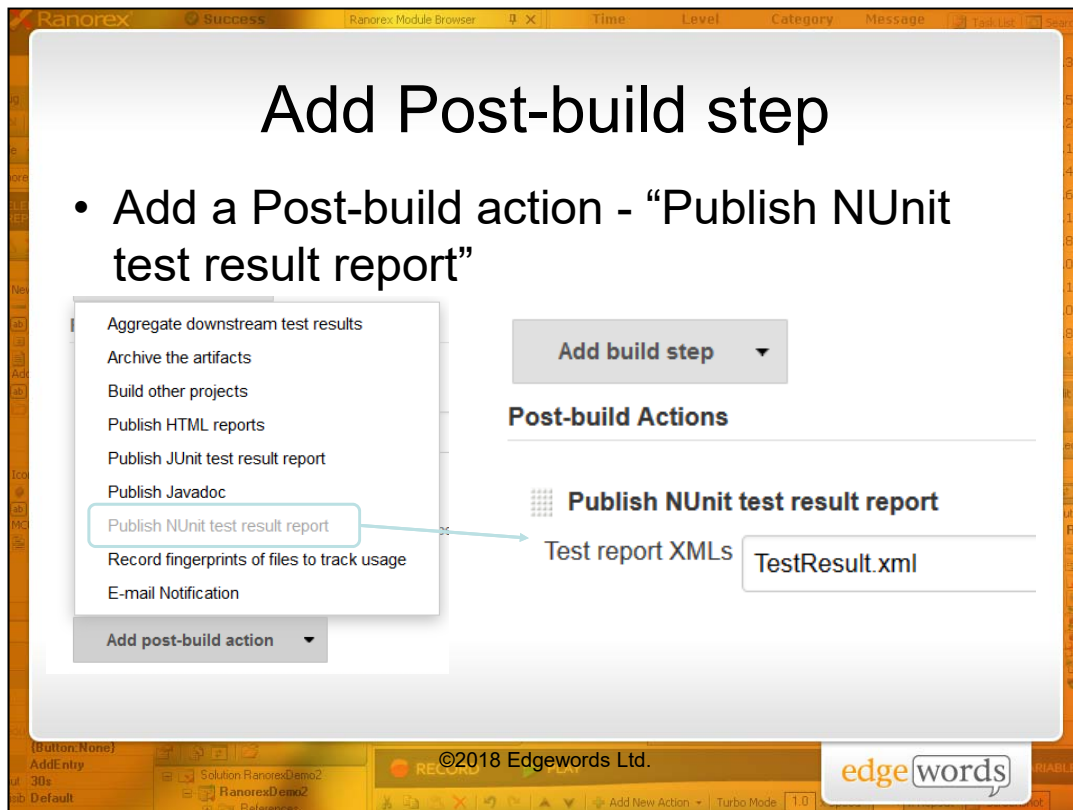
Command `"C:\NUnit\nunit-console\nunit3-console.exe"`
`%WORKSPACE%\LeanFTAutoTest\bin\Debug\com.edgewords.wd3_3day.dll`

See [the list of available environment variables](#)

©2018 Edgewords Ltd.

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.

Optional - Extent Report

- Add HTML Report Post-Build Step for Extent Report

Post-build Actions

Publish HTML reports

Reports

HTML directory to archive	<input type="text" value="com.edgewords.wd3_3day\bin\Debug"/>
Index page[s]	<input type="text" value="com.edgewords.wd3_3day.Tests.FirstTest.html"/>
Index page title[s] (Optional)	<input type="text" value="Extent Test Results"/>
Report title	<input type="text" value="Execution Report"/>

Publishing options...

©2018 Edgewords Ltd. edge words

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.

Results

Test Result :
com.edgewords.webdriver.example.Tests

0 failures

5 tests
Took 40 sec.
[add description](#)

All Tests

Class	Duration	Fail	(diff) Skip	(diff) Pass	(diff) Total	(diff)
AddRecordsTests	14 sec	0	0	1 +1	1 +1	
DataDrivenTests	12 sec	0	0	2 +2	2 +2	
SimpleTests	13 sec	0	0	2 +2	2 +2	

©2018 Edgewords Ltd.

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