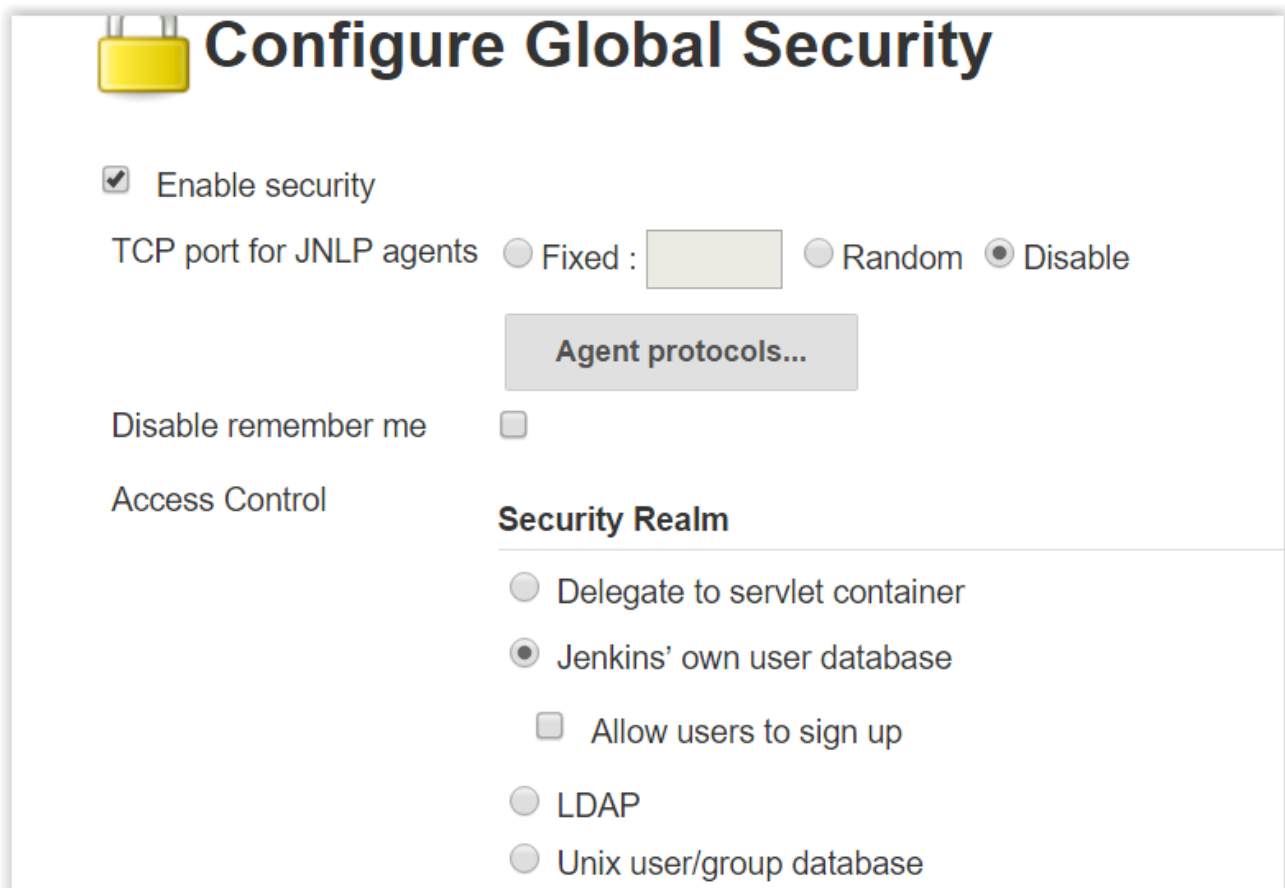



## Exercise 3 Jenkins - Quick Guide

1. Add windows agent to Jenkins like in the Demo and install it as a service:
2. Go to windows and make a new folder c:\jenkins
3. Also need to install git in windows
4. Install jre : <http://download.oracle.com/otn-pub/java/jdk/8u121-b13/e9e7ea248e2c4826b92b3f075a80e441/jre-8u121-windows-x64.exe>
5. Now open Jenkins window
6. Go to manage Jenkins
7. Go to configure global security



 **Configure Global Security**

☒ Enable security

TCP port for JNLP agents ☐ Fixed :  ☐ Random ☒ Disable

**Agent protocols...**

Disable remember me ☐

Access Control

**Security Realm**

☐ Delegate to servlet container

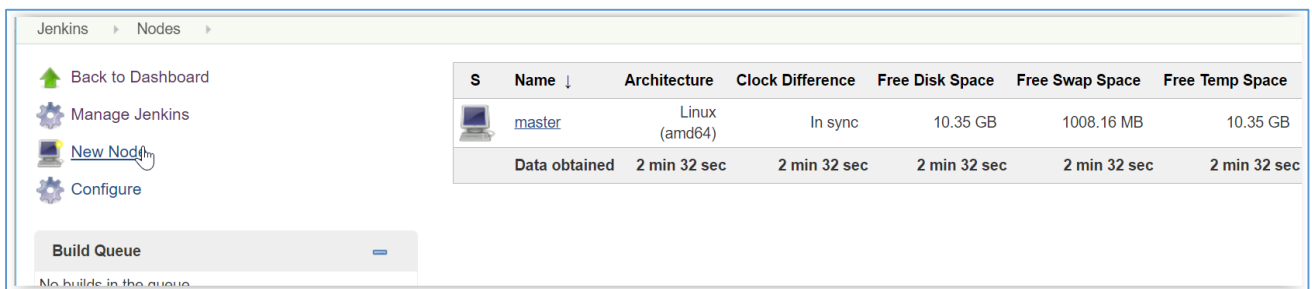
☒ Jenkins' own user database

☐ Allow users to sign up

☐ LDAP

☐ Unix user/group database

8. Change TCP port for JNLP agents to random
9. And click save
10. Go to manage nodes



Jenkins > Nodes

- Back to Dashboard
- Manage Jenkins
- New Node**
- Configure

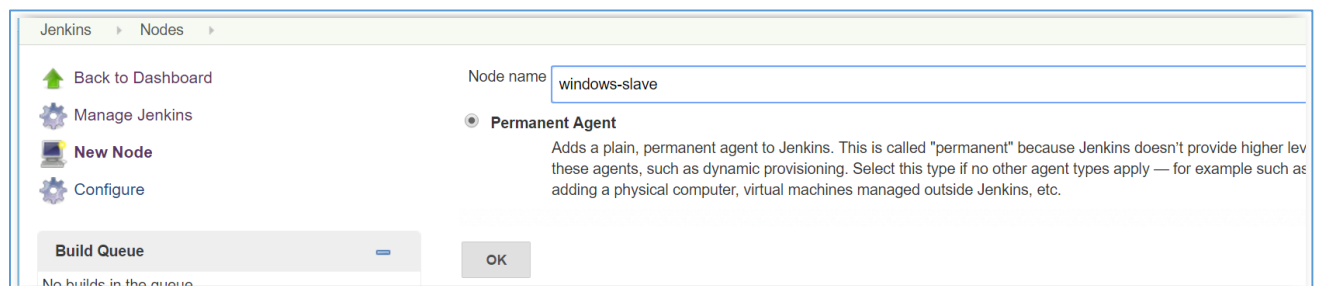
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space
	master	Linux (amd64)	In sync	10.35 GB	1008.16 MB	10.35 GB
	Data obtained	2 min 32 sec	2 min 32 sec	2 min 32 sec	2 min 32 sec	2 min 32 sec

Build Queue

No builds in the queue.

11. Click new node

12. Give it a name and mark permanent agent



Jenkins > Nodes

- Back to Dashboard
- Manage Jenkins
- New Node**
- Configure

Node name: windows-slave

☒ **Permanent Agent**

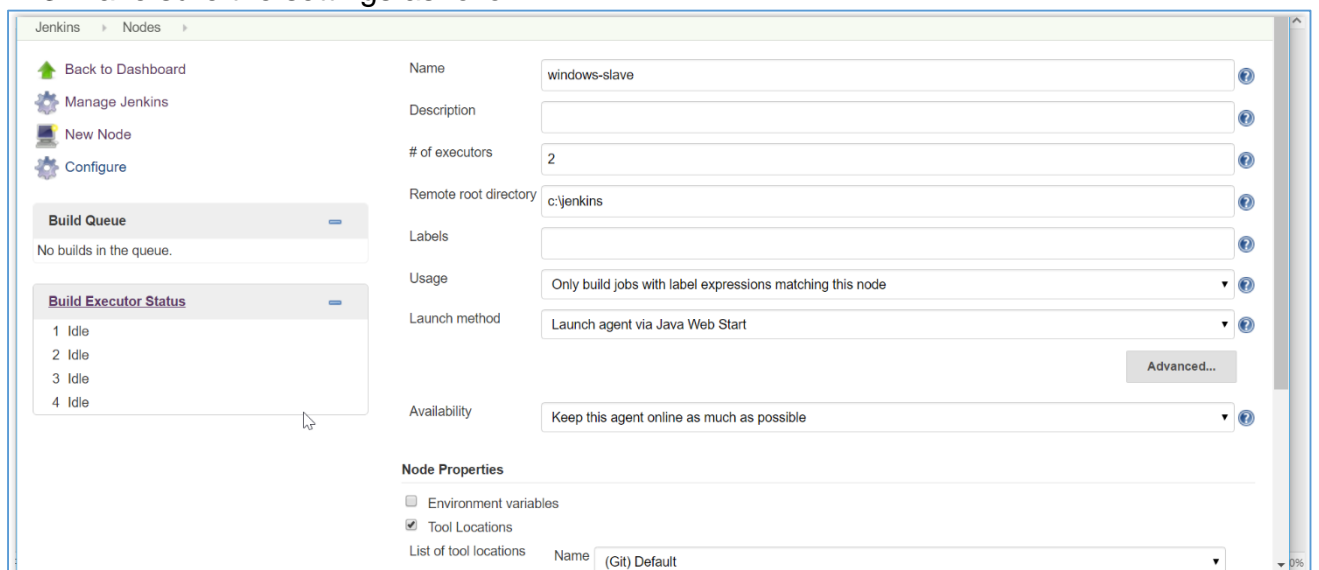
Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as adding a physical computer, virtual machines managed outside Jenkins, etc.

Build Queue

No builds in the queue.

OK

13. Make sure the settings as follow:



Jenkins > Nodes

- Back to Dashboard
- Manage Jenkins
- New Node**
- Configure

Build Queue

No builds in the queue.

**Build Executor Status**

- 1 Idle
- 2 Idle
- 3 Idle
- 4 Idle

Name: windows-slave

Description:

# of executors: 2

Remote root directory: c:\jenkins

Labels:

Usage: Only build jobs with label expressions matching this node

Launch method: Launch agent via Java Web Start

Availability: Keep this agent online as much as possible

Advanced...

Node Properties

- ☐ Environment variables
- ☒ Tool Locations

List of tool locations: Name (Git) Default

14. Insert the path to git bin folder

15. And click save
16. Now click on the new node and launch agent
17. Now in the small window where it says connected: go file and click install as a service
18. Open Jenkins in your browser
19. Navigate to "Manage Jenkins"
20. Navigate to "Manage Plugins"
21. Install the MSBuild Plugin for Jenkins:

Updates	Available	Installed	Advanced
Enabled		Name ↓	Version
<input checked="" type="checkbox"/>		<a href="#">Ant Plugin</a> This plugin adds <a href="#">Apache Ant</a> support to Jenkins.	<a href="#">1.2</a>
<input checked="" type="checkbox"/>		<a href="#">Credentials Plugin</a> This plugin allows you to store credentials in Jenkins.	<a href="#">1.24</a>
<input checked="" type="checkbox"/>		<a href="#">CVS Plug-in</a> Integrates Jenkins with CVS version control system using a modified version of the Netbeans cvsclient.	<a href="#">2.11</a>
<input checked="" type="checkbox"/>		<a href="#">External Monitor Job Type Plugin</a> Adds the ability to monitor the result of externally executed jobs.	<a href="#">1.4</a>
<input checked="" type="checkbox"/>		<a href="#">Git client plugin</a> Shared library plugin for other Git related Jenkins plugins.	<a href="#">1.19.0</a>
<input checked="" type="checkbox"/>		<a href="#">Git plugin</a> This plugin integrates <a href="#">GIT</a> with Jenkins.	<a href="#">2.4.0</a>
<input checked="" type="checkbox"/>		<a href="#">Javadoc Plugin</a> This plugin adds Javadoc support to Jenkins.	<a href="#">1.1</a>
<input checked="" type="checkbox"/>		<a href="#">JUnit Plugin</a> Allows JUnit-format test results to be published.	<a href="#">1.2-beta-4</a>
<input checked="" type="checkbox"/>		<a href="#">LDAP Plugin</a> Adds LDAP authentication to Jenkins	<a href="#">1.11</a>
<input checked="" type="checkbox"/>		<a href="#">Mailer Plugin</a> This plugin allows you to configure email notifications. This is a break-out of the original core based email component.	<a href="#">1.16</a>
<input checked="" type="checkbox"/>		<a href="#">Matrix Authorization Strategy Plugin</a> Offers matrix-based security authorization strategies (global and per-project).	<a href="#">1.1</a>
<input checked="" type="checkbox"/>		<a href="#">Matrix Project Plugin</a> Multi-configuration (matrix) project type.	<a href="#">1.4.1</a>
<input checked="" type="checkbox"/>		<a href="#">Maven Integration plugin</a> Jenkins plugin for building Maven 2/3 jobs via a special project type.	<a href="#">2.7.1</a>
<input checked="" type="checkbox"/>		<a href="#">MSBuild Plugin</a> This plugin makes it possible to build a Visual Studio project ( .proj ) and solution files ( .sln )	<a href="#">1.24</a>

22. Jenkins now understands MSBuild files

23. but we still need to configure the MSBuild Plugin with a path to the **msbuild.exe** we would like to use

## MSBuild configuration

When the MSBuild plugin was installed it added its own configuration options to the Jenkins global configuration page.

1. Click "Manage Jenkins"
2. Click "Configure System"
3. Scroll down the list until you find "MSBuild"
4. Click the "MSBuild installations..." button
5. Click "Add MSBuild"
6. Give the new MSBuild configuration a name like "MSBuild-default".
7. In the path field, insert the fully qualified path to **msbuild.exe**  
On my server the path is: **C:\Program Files (x86)\MSBuild14.0\Bin\msbuild.exe**, but this can be different on your system.
8. Click save

The screenshot shows the Jenkins 'MSBuild' configuration page. On the left, there's a sidebar with 'MSBuild' and 'MSBuild installations'. The main area shows a table with one configuration named 'MSBuild-default'. The 'Path to MSBuild' field is filled with 'C:\Program Files (x86)\MSBuild14.0\Bin\msbuild.exe'. Below this, a yellow warning icon and text indicate that the path is not a directory on the Jenkins master. At the bottom, there are buttons for 'Add MSBuild' and 'Delete MSBuild'.

### MSBuild installation

MSBuild is installed with Visual Studio, it's the build system that Visual Studio uses when you select "build" or hit "F5".

It's not always feasible or even possible to install Visual Studio on your build machine. This could be due to license and security issues etc.

To accommodate this Microsoft has released a separate package called: "Microsoft Build Tools 2015" that contains all you need for using MSBuild.

Direct download: <https://www.microsoft.com/en-us/download/details.aspx?id=48159>

After successful installation you have MSBuild available on the build server and with that you get the path value for step 8, above.

With this step done Jenkins is ready to build and deploy with MSBuild and Git.

## Create a new Jenkins build project

Project name

Description

[Plain text] [Preview](#)

1. Select "new" job
2. Select "Freestyle project".

- ☐ Discard Old Builds
- ☐ This build is parameterized
- ☐ Disable Build (No new builds will be executed until the project is re-enabled.)
- ☐ Execute concurrent builds if necessary

### Advanced Project Options

#### Source Code Management

- ☒ None
- ☐ CVS
- ☐ CVS Projectset
- ☐ Git
- ☐ Subversion

#### Build Triggers

- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ Poll SCM

#### Build

Add build step ▼

#### Post-build Actions

Add post-build action ▼

Save

Apply

In the advanced project option click on run only on specific node label : windows (or your windows agent slave name label)

Next, expand the “Source Code Management” region by selecting “Git”.

Fork my github repo:

<https://github.com/Lidorlg/jenkins-ci-template.git>

and use your own url in git:

The screenshot shows the 'Source Code Management' configuration section in Jenkins. The 'Git' option is selected under 'Source Code Management'. The 'Repositories' section is expanded, showing a 'Repository URL' field with a red error message 'Please enter Git repository.' and a 'Credentials' dropdown menu set to '- none -'. There are 'Add Repository' and 'Delete Repository' buttons. The 'Branches to build' section shows a 'Branch Specifier (blank for 'any')' set to '\*/master' with 'Add Branch' and 'Delete Branch' buttons. The 'Repository browser' is set to '(Auto)'. At the bottom, there is an 'Additional Behaviours' section with an 'Add' button.

## *Testing the Git configuration*

1. Navigate to the “Project” page
2. Click the “Build Now” to start a “Build”.

**Project Windows Service deployment**

Back to Dashboard

Status

Changes

Workspace

Build Now

Delete Project

Configure

Workspace

Recent Changes

**Build History** trend

find

#1 Jan 5, 2016 10:04 AM

RSS for all RSS for failures

**Permalinks**

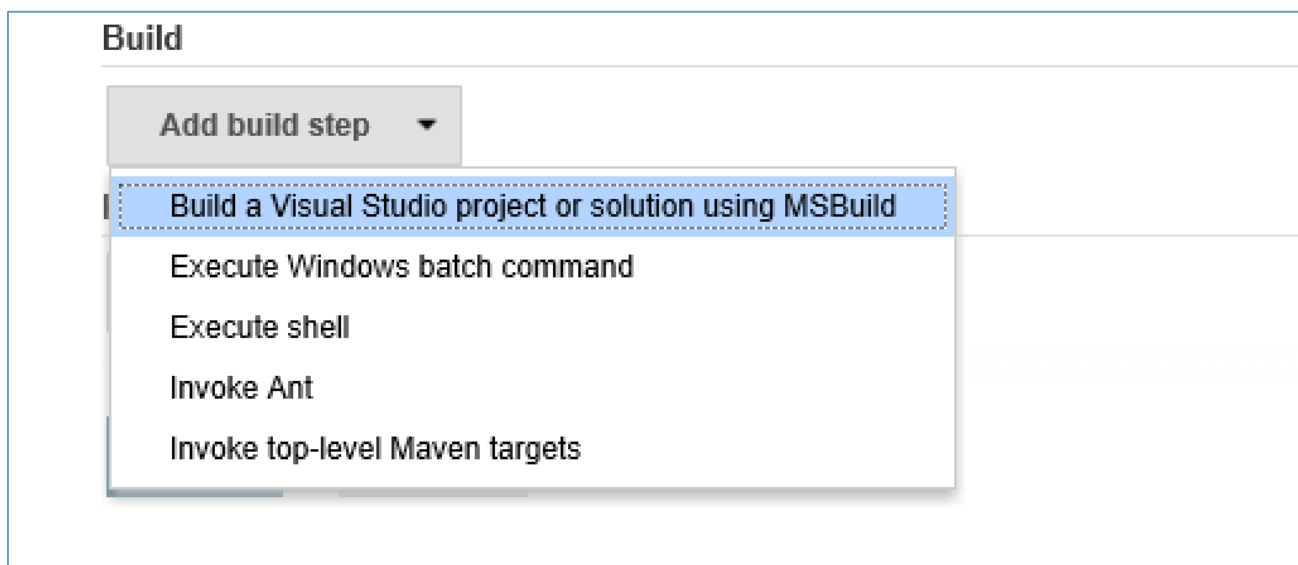
- [Last build \(#1\). 1 min 0 sec ago](#)
- [Last stable build \(#1\). 1 min 0 sec ago](#)
- [Last successful build \(#1\). 1 min 0 sec ago](#)
- [Last completed build \(#1\). 1 min 0 sec ago](#)

3. In the “Build History” region you should now see a build in progress with the name “#1”.
4. If every completes as expected “SUCCESS” then the bubble maker stays blue, if not it goes red.

## *Building the source*

The next step is to compile and build the source code.

1. Navigate to the “project” page
2. Select “Configure”
3. Find the “Add build step”



Select the "Build a Visual Studio project or solution using MSBuild".

We need to configure a few values here:

1. First, select the MSBuild version (we configured this in a previous step).
2. Then give the path to the \*.sln or \*.proj file for your project.

*For the pre-cooked repository the path*

*is: [srcMyWindowsServiceMyWindowsServiceDeploy-Windows-Service-Via-MSBuild.proj](#)*

*Please note: We are not pointing to the solution file, rather we are pointing to a custom MSBuild file that is in the project. This MSBuild file is handling all steps involved with compiling and deploying the Windows Service.*

*If this step not working -> point to sln file*



The screenshot shows the 'Build' configuration page in Jenkins. It has a title bar 'Build' and a subtitle 'Build a Visual Studio project or solution using MSBuild'. There are four main input fields: 'MSBuild Version' with a dropdown menu showing 'MSBuild-default'; 'MSBuild Build File' with a text box containing 'src\MyWindowsService\MyWindowsService\Deploy-Windows-Service-Via-MSBuild.proj'; 'Command Line Arguments' with a large empty text area; and 'Pass build variables as properties' with an unchecked checkbox. At the bottom left, there is a button labeled 'Add build step' with a small downward arrow.

## *NuGet package restore*

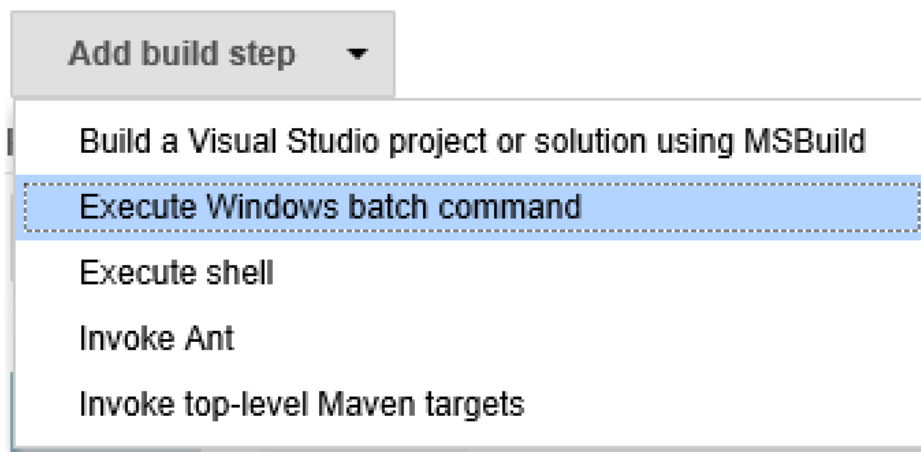
If we were to build the project right now it would fail due to missing NuGet packages. Therefore we need to restore the NuGet packages before attempting to build the source.

**nuget.exe** makes this task very easy, we simply need to fire this command on the solution file:

**nuget restore "path to \*.sln" file**

No surprise Jenkins can handle multiple build steps, so let's add a build step to enable NuGet restore.

1. Navigate to the "project" page.
2. Select "Configure".
3. Find the "Add build step".
4. Click "Add build step".



5. Select "Execute Windows batch command".
6. Re-arrange the build order by dragging the new build step to the top.

#### Build

##### Execute Windows batch command

Command `nuget restore src\MyWindowsService\`

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

##### Build a Visual Studio project or solution using MSBuild

MSBuild Version

MSBuild-default

MSBuild Build File

src\MyWindowsService\MyWindowsService\Deploy-Windows-Service-Via-MSBuild.proj

Command Line Arguments

Pass build variables as properties ☐

7. In the Command field insert:






```
nuget restore src\MyWindowsService
```

8. Click "Save".

We are now ready to build the project!

*The final test! Primetime!*

1. Navigate to the project page
2. Click the "Build now" link
3. Wait for the build to complete
4. Open "Windows Explorer" and navigate to **C:**
5. Confirm that the file "MyWindowsService.log" exists

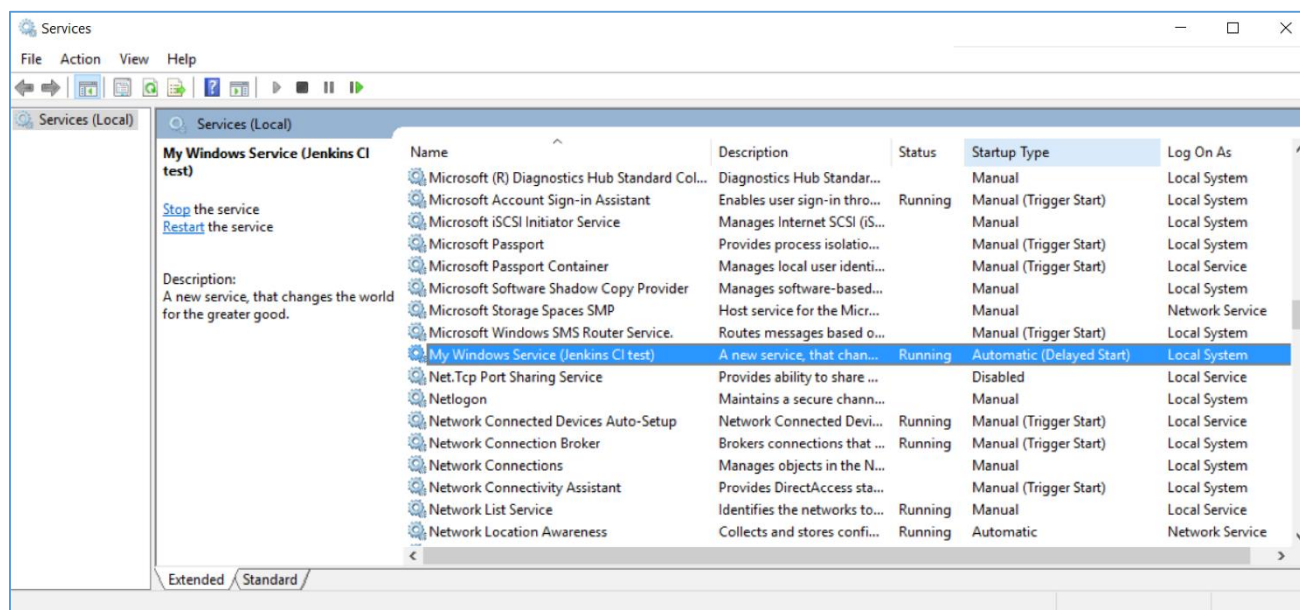
 Users	11/1/2015 11:21 AM	File folder	
 Windows	11/9/2015 3:50 PM	File folder	
 WindowsAzure	11/1/2015 11:06 AM	File folder	
 MyWindowsService.log	1/5/2016 3:03 PM	Text Document	1 KB
 test_test	1/5/2016 9:22 PM	Text Document	1 KB

6. Open the file and read the log content.

This file was create by our newly installed Windows Service!

7. Check that the Service is installed and running:

- A. On Windows open "Services" management window
- B. Scroll down to you find the service "My Windows Service (...)"



**Congratulations! You have now successfully configured Jenkins to download, compile and deploy a .NET Windows Service! Automation ROCKS!**