

Sign up for our free weekly [Web Developer Newsletter](#).



[articles](#) [Q&A](#) [forums](#) [lounge](#)

Search for articles, questions, tips



Create Your Own Private NuGet Server in Windows Azure



Alex Sanséau, 5 Feb 2015

CPOL

Rate this:



4.92 (18 votes)

As an introduction to Windows Azure, this article will guide you step by step to create your own private NuGet server in Windows Azure.

Introduction

Whether you are part of a large development team or working alone on your own side project, you'll probably want to re-use code across multiple projects.

One neat way of doing it, is to create packages, to store them in a repository and to reference them whenever needed. NuGet packages also allow you to have multiple versions of a same package and to specify any eventual dependencies. If you regularly use NuGet Package Manager from Visual Studio, then you are already very familiar with the concept.

This article will show you how you can create your own private NuGet server and how to host it in Windows Azure.

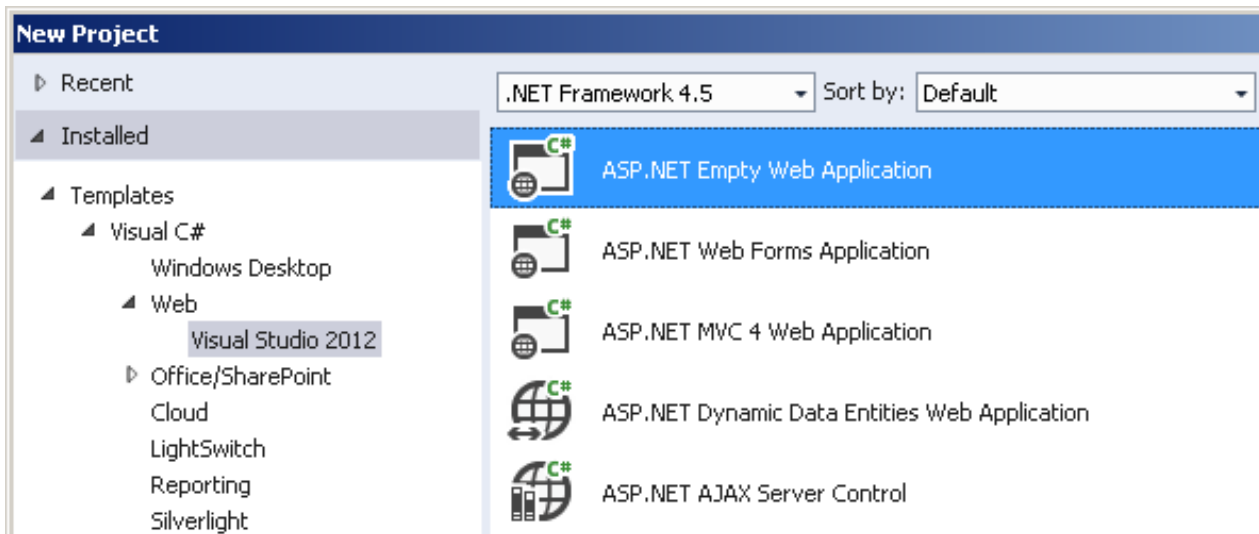
Pre-requisites

Do I need to mention you need Visual Studio? Maybe not, but I've just done it! Any recent edition should do (screenshots for this article are taken from Visual Studio 2013).

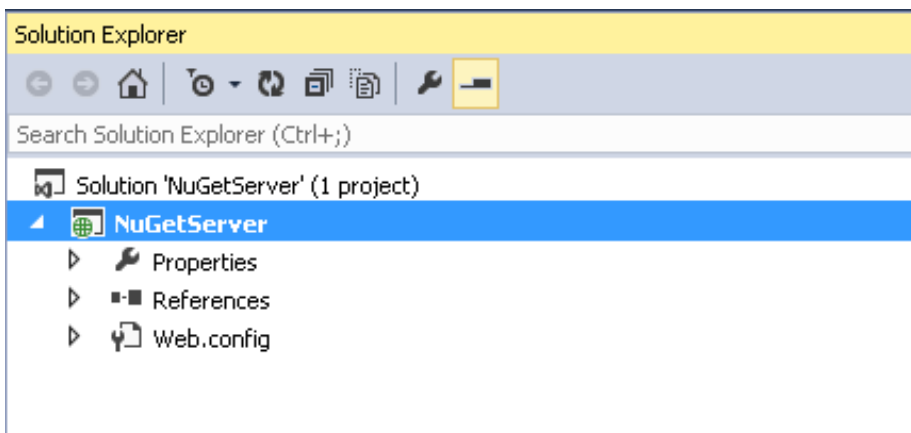
And obviously, you'll also need a valid subscription to Windows Azure. If you don't have one yet, you can sign up for a free 30-day trial [here](#).

Create a New NuGet Server

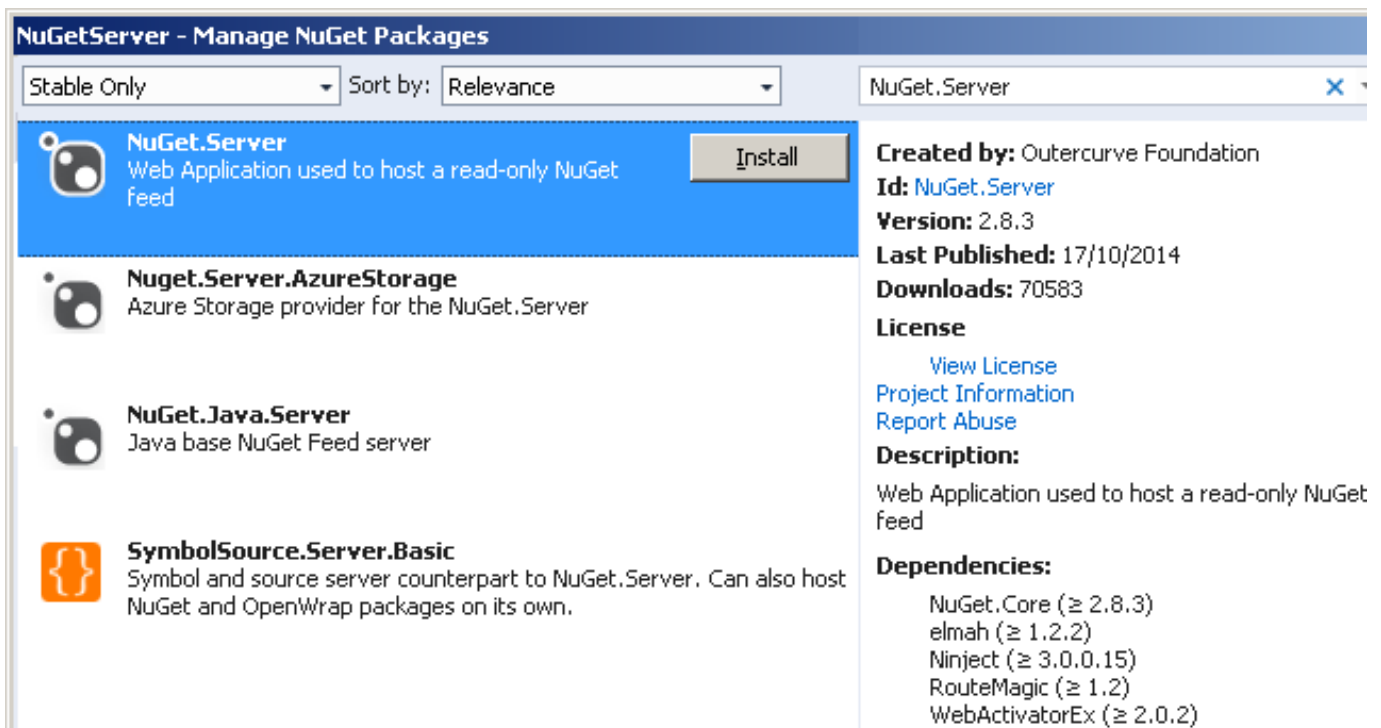
Open Visual Studio. Go to **File > New > Project...** to create a new project. From the templates, select **ASP.NET Empty Web Application**, as follows:



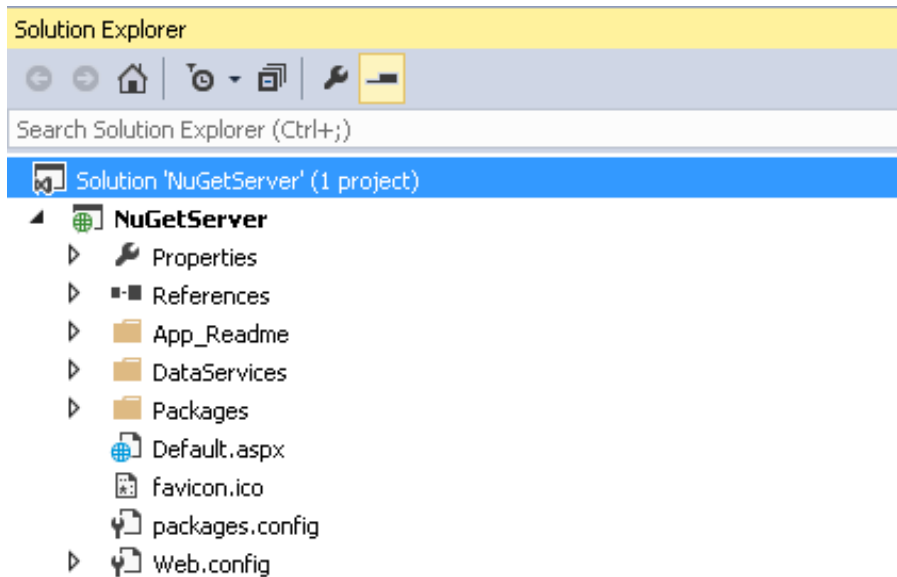
The **Solution Explorer** should show an empty solution, as follows:



Now, we'll use the official NuGet repository to add **NuGet.Server** to our project. Right-click on **References** and select **Manage NuGet Packages...**. Search for **NuGet.Server** and click **Install**.



Now, the solution should look like this:



Press F6 and ensure the solution builds successfully.

We're almost ready. There's just one more thing to do before we can run and deploy our code, which is to review our configuration. Open up **web.config** and review the following settings.

Hide Copy Code

```
<!--
    Determines if an Api Key is required to push/delete packages from the server.
-->
<add key="requireApiKey" value="true"/>
<!--
    Set the value here to allow people to push/delete packages from the server.
    NOTE: This is a shared key (password) for all users.
-->
<add key="apiKey" value="Xx5TJbXBe9jEvAfGxV7x"/>
<!--
    Change the path to the packages folder. Default is ~/Packages.
    This can be a virtual or physical path.
-->
<add key="packagesPath" value=""/>
```

To ensure only authorized users will be able to add and delete packages, set **requireApiKey** to **true** and set **apiKey** to the specific key you want to use. (Please make sure to use a different key than the one given in this example). Leave **packagesPath** to its default value.

Press F5 to run the server. You should see the following home page:

You are running NuGet.Server v2.8.50926.602

Click [here](#) to view your packages.

Repository URLs

In the package manager settings, add the following URL to the list of Package Sources:

http://localhost:52001/nuget

Use the command below to push packages to this feed using the nuget command line tool (nuget.exe).

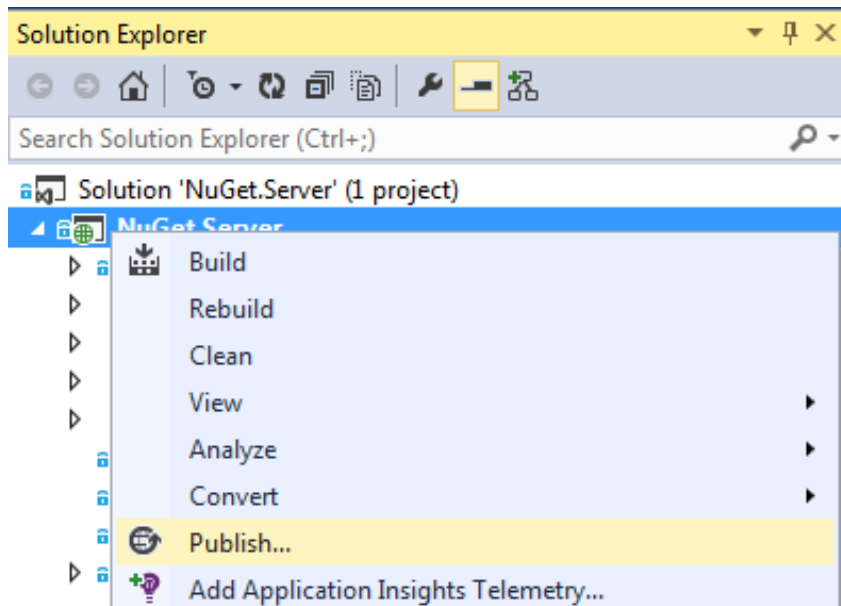
nuget push {package file} -s http://localhost:52001/ {apikey}

To add packages to the feed put package files (.nupkg files) in the folder "C:\Code\NuGetServer\NuGetServer\Packages".

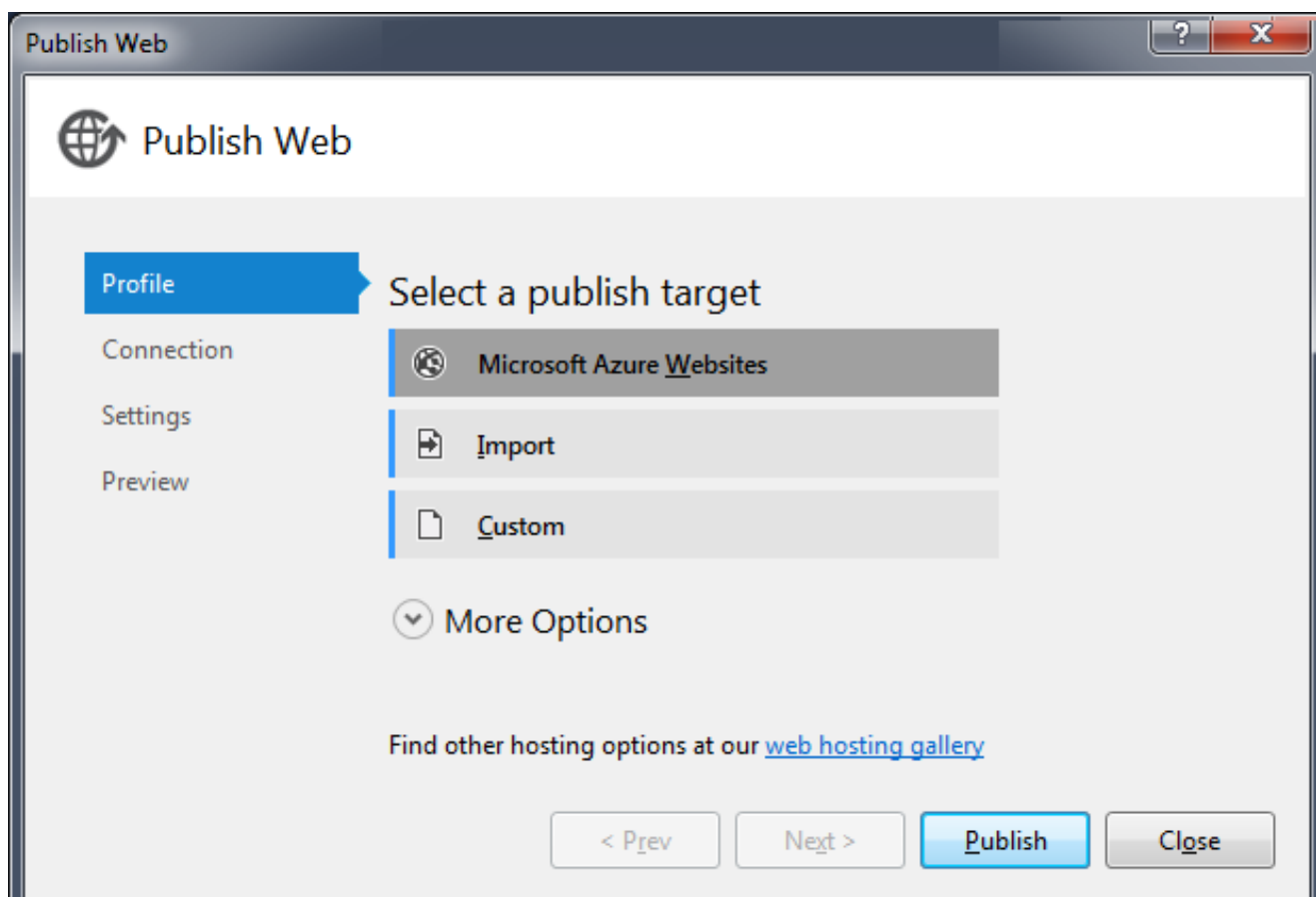
Now that we've got a NuGet server up and running locally, let's deploy it to Windows Azure.

Deploy to Windows Azure

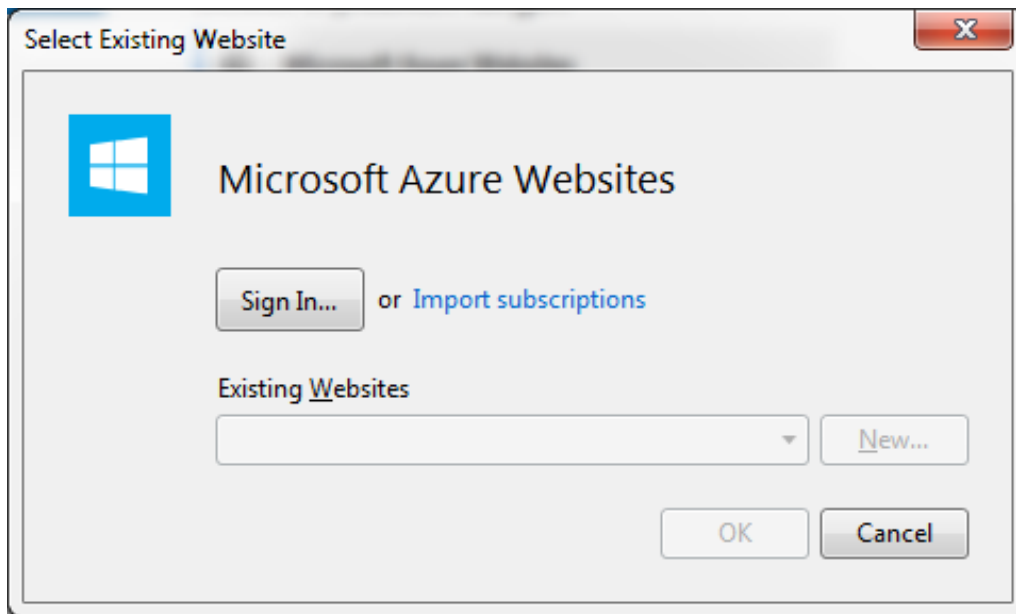
The integration with Windows Azure from Visual Studio is excellent and you can create your new website and deploy to it directly from Visual Studio. In the **Solution Explorer**, right click on **NuGetServer** project and click on **Publish...**



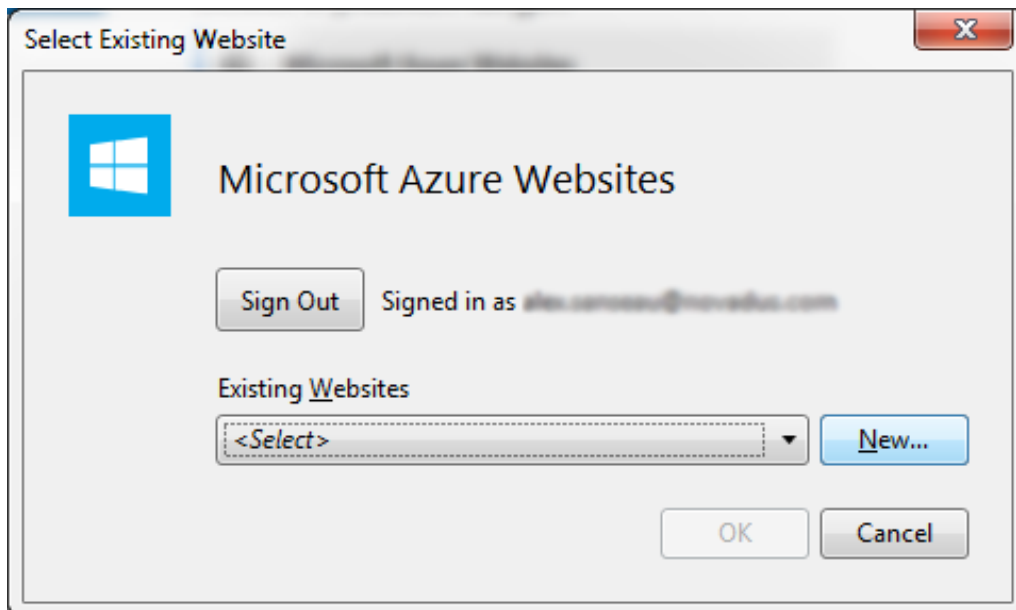
From the **Profile** section, select **Microsoft Azure Websites**.



Click **sign-in** and authenticate with the email address used when you signed up for Windows Azure.

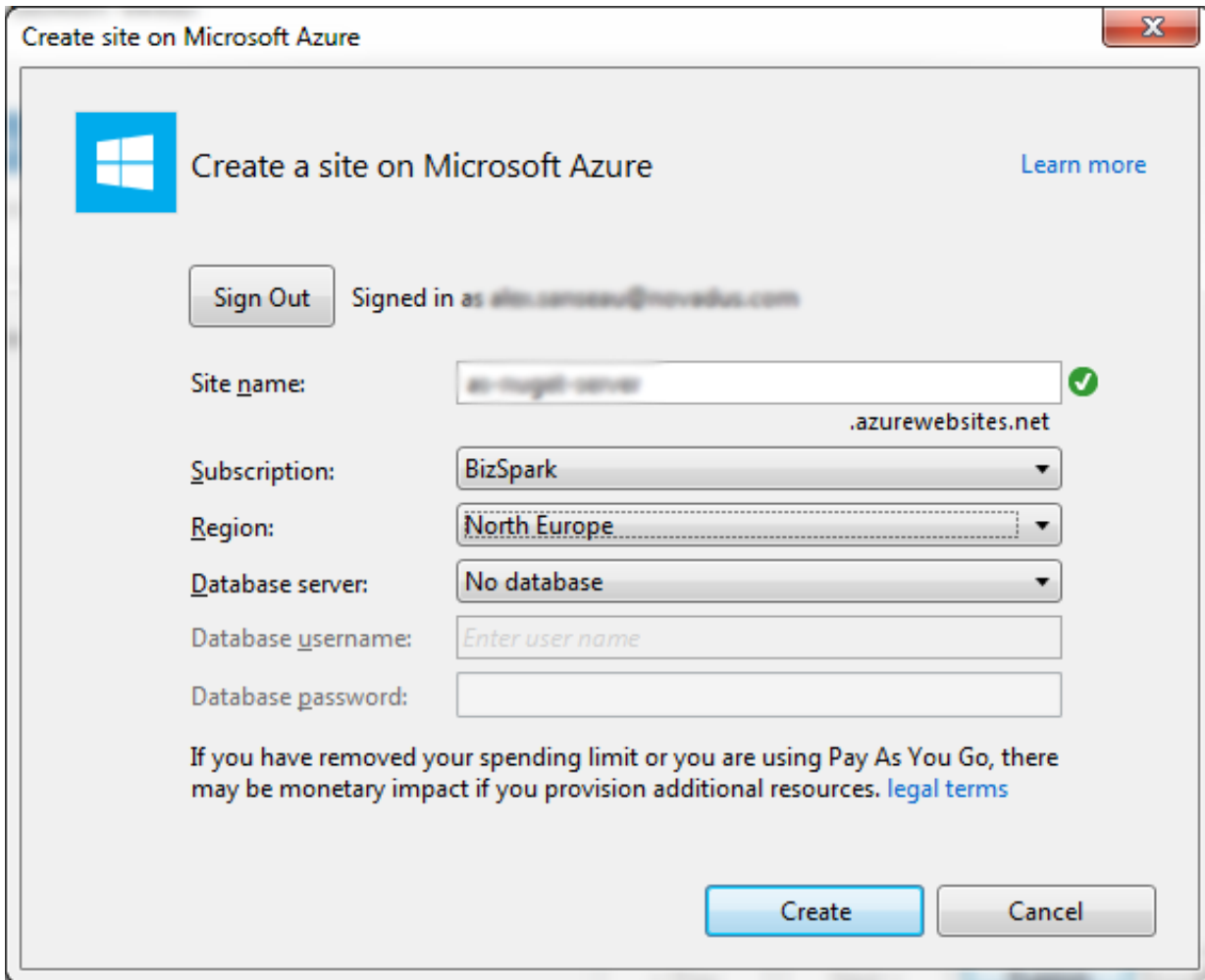


Now that you are signed in, click **New...** to create a new website:



Chose a **site name** that has not be taken already. I'm going to use **code-project-nugetserver** for the remainder of this article. The Nuget server will then be available at: **<https://code-project-nugetserver.azurewebsites.net>**. (of course, you should use a different name as a reader may have already used it).

Unless you have multiple subscriptions, keep the default one. Chose a region that will be close to your users in order to improve performance. There is no need for a database for this project.



Create site on Microsoft Azure

Create a site on Microsoft Azure [Learn more](#)

Sign Out Signed in as alan.sarasa@novadus.com

Site name: as-nuget-server ☒ .azurewebsites.net

Subscription: BizSpark

Region: North Europe

Database server: No database

Database username: Enter user name

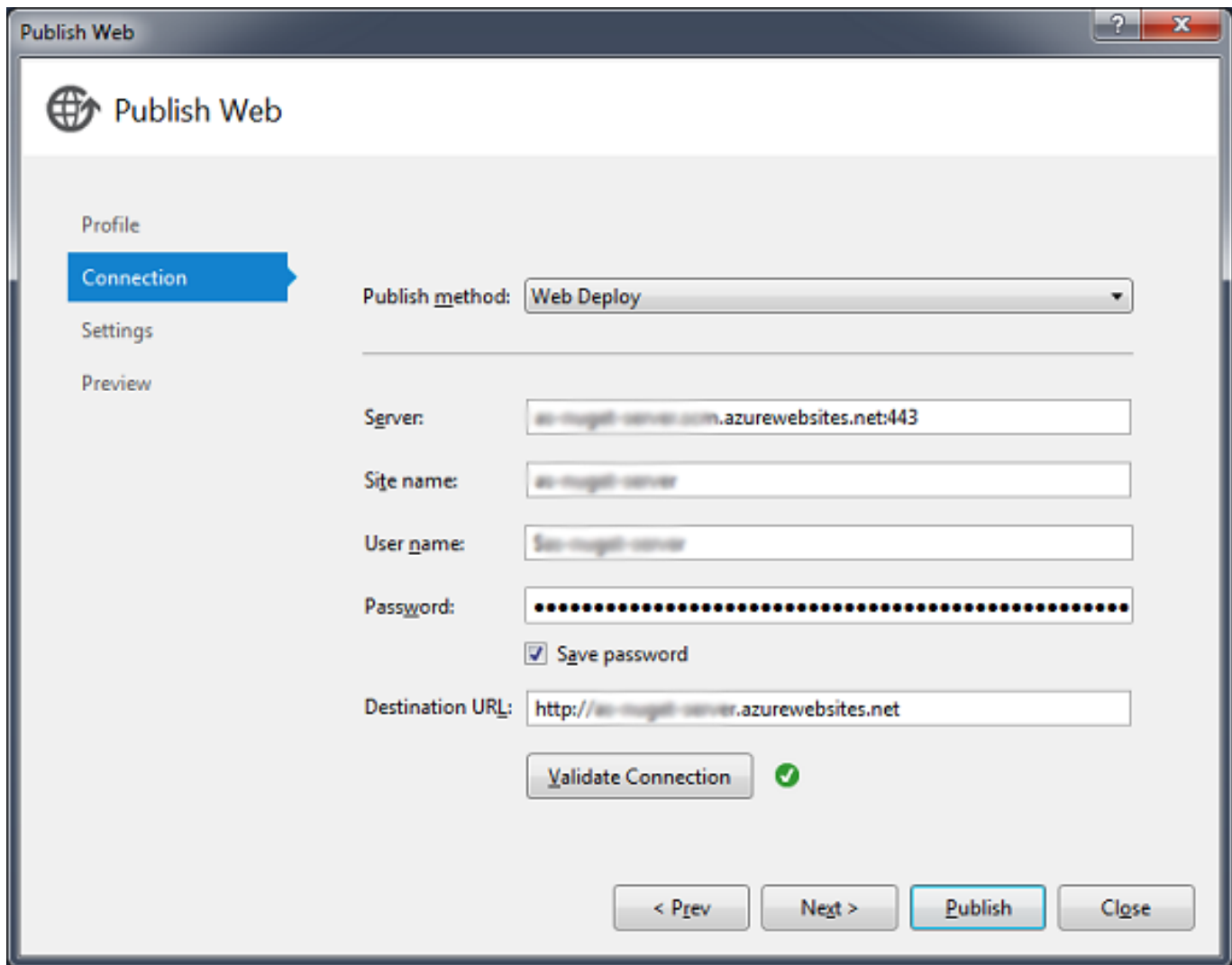
Database password:

If you have removed your spending limit or you are using Pay As You Go, there may be monetary impact if you provision additional resources. [legal terms](#)

Create Cancel

Now, click **Create** to create the website in Azure. Once it's created, you'll be shown the final screen before publication. It contains connection details, such as the address of the site, the username and the password. All fields should be correctly pre-populated and you can click **Validate Connection** to ensure the details are correct.

When you are ready, click **Publish** to publish the NuGet server to the website.



Publish Web

Profile
Connection
Settings
Preview

Publish method: Web Deploy


Server: as-nuget-server.azurewebsites.net:443

Site name: as-nuget-server

User name: as-nuget-server

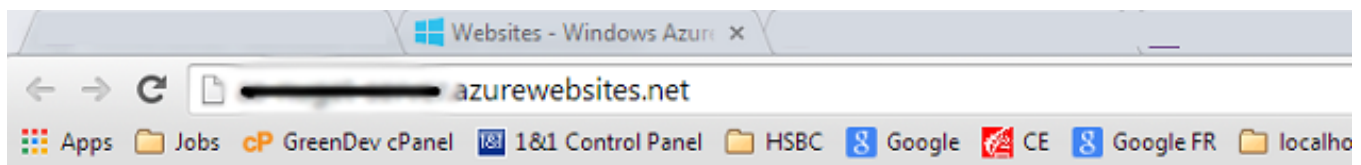
Password:
☒ Save password

Destination URL: http://as-nuget-server.azurewebsites.net

Validate Connection 

< Prev Next > **Publish** Close

Now go to <https://code-project-nugetserver.azurewebsites.net> (remember it was recommended you chose a different name, so the first part of the URL should be different for you) to ensure the server is up and running. You should see something like:



You are running NuGet.Server v2.8.50320.36

Click [here](#) to view your packages.

Repository URLs

In the package manager settings, add the following URL to the list of Package Sources:

`http://as-nuget-server.azurewebsites.net/nuget`

Use the command below to push packages to this feed using the nuget command line tool (nuget.exe).

`nuget push {package file} -s http://as-nuget-server.azurewebsites.net/ {apikey}`

Congratulations, you've got your NuGet server up and running!

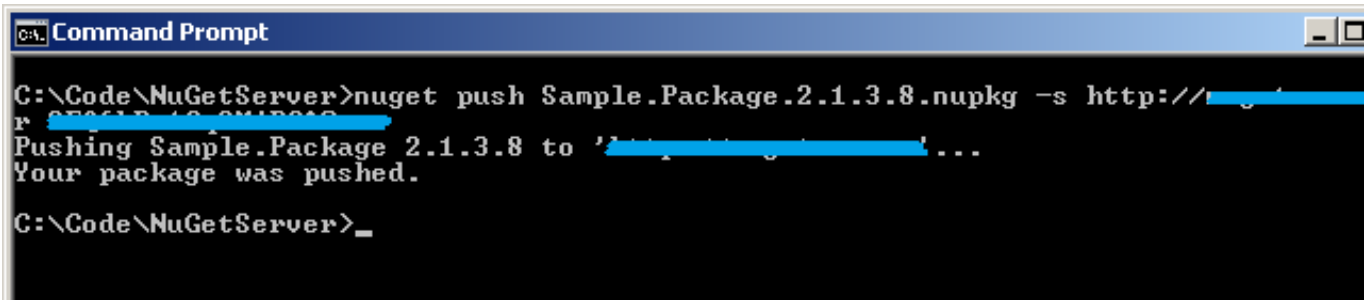
Publish your First Package

Download **NuGet.exe** from <http://nuget.codeplex.com/releases/view/121838> (or check for the latest version) and make sure it's in your path so you can run it from the command line. Indeed, **NuGet.exe** is a command line tool which allows you to create and to publish packages. For the purpose of this article, I've attached a sample package (**Sample.Package.2.1.3.8.nupkg**), and if you want to read more about how to create NuGet packages, there is an excellent documentation on nuget.org.

You are now ready to publish your first package. From the command line, run:

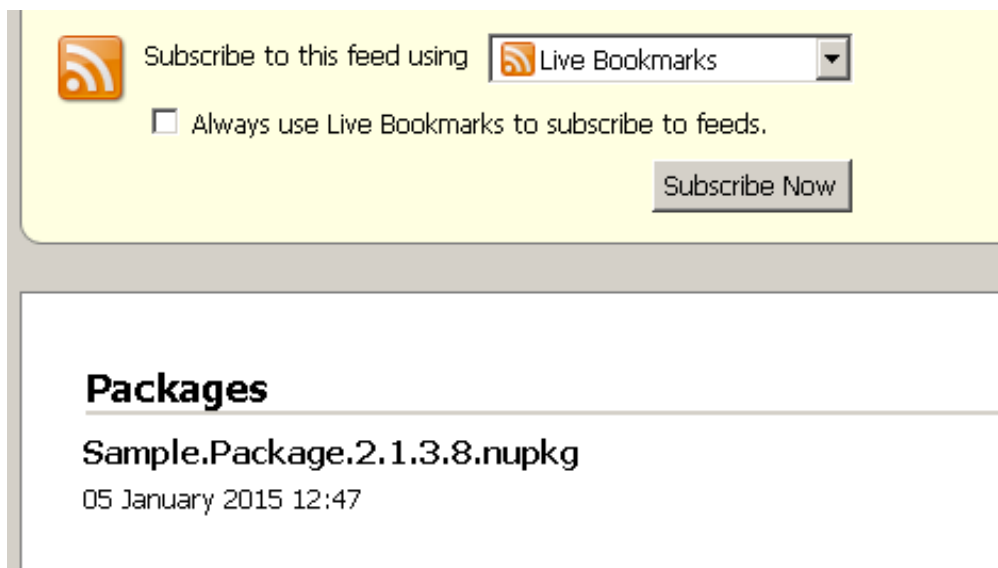
[Hide](#) [Copy Code](#)

```
nuget push Sample.Package.2.1.3.8.nupkg -s  
https://code-project-nugetserver.azurewebsites.net Xx5TJbXBe9jEvAfGxV7x
```



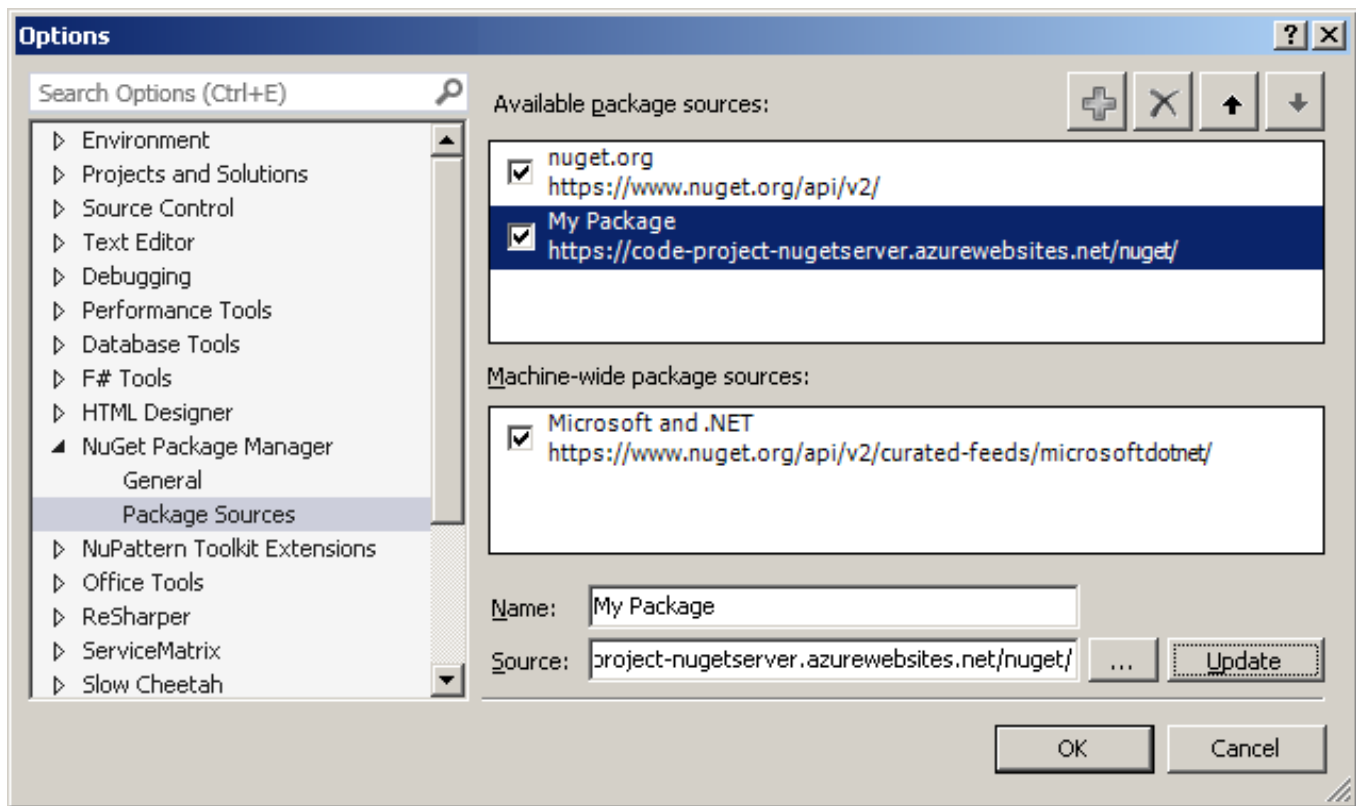
```
C:\Code\NuGetServer>nuget push Sample.Package.2.1.3.8.nupkg -s http://...  
Pushing Sample.Package 2.1.3.8 to '...' ...  
Your package was pushed.  
C:\Code\NuGetServer>
```

You can double check your packages has been successfully published by going to <https://code-project-nugetserver.azurewebsites.net/nuget/Packages>. Ensure **Sample.Package.2.1.3.8.nupkg** is listed in the page:

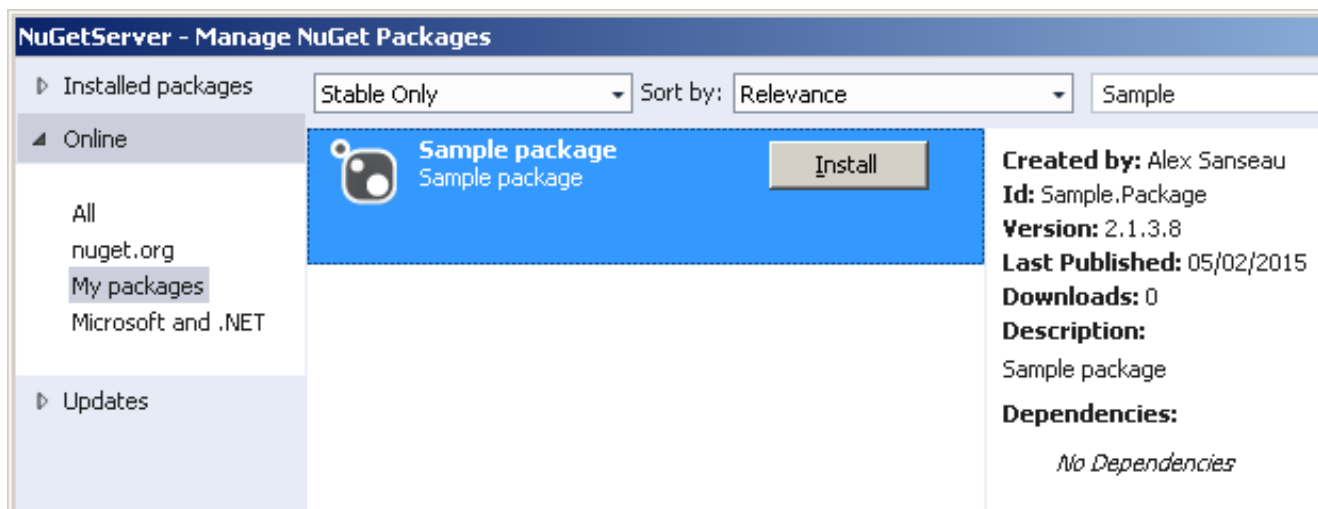


Download Your Own Packages from Visual Studio

Now that you have your own NuGet Server, you can configure Visual Studio Package Manager to serve your own packages. In **Visual Studio**, go to **Tools > Options > NuGet Package Manager > Packages Sources**. Click on the + sign (top right corner) to add a new source. Give it a name in the **Name** field, for instance **My Packages** and set the **Source** to <https://code-project-nugetserver.azurewebsites.net/nuget/>.



Now, you can add your own packages to your projects. For a given project, right-click on **References** and select **Manage NuGet Packages...**. On the left hand-side, select **My Packages**, and **Sample.Package.2.1.3.8.nupkg** should be listed:



You now have a mature development environment where you can publish packages for re-use across multiple projects.

Points of Interest

In this article, I've covered in detail how to create your own private NuGet server and how to publish it to Windows Azure. This is quite simple to set up and your server should be running from the cloud in less than 20 minutes.

Bundle up your code in packages and publish them to share with your development team or for yourself to re-use in future projects. You can also use NuGet packages for deployment purpose: I will cover how to deploy code with Octopus Deploy in a future article.

One of the great advantages of Windows Azure is scalability and I will cover in detail in a separate article how you can set up your website to automatically scale up or down based on your server workload.

Gotcha: If you don't want to store the NuGet packages in the default folder, make sure you specify a relative path. Your server is running in IIS alongside a number of other websites. The security model in Azure makes sure that each website runs

in isolation and therefore has only got access to the root folder of the website.

Please let me know how you get on with your NuGet server. Do you use it at work? Or do you use it at home for a side project? Do you use it for deployments? Have you encounter any limitations or do you have any feature requests?

Thanks for your feedback and happy coding!

History

- [05/02/2015]: First publication

License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOL\)](#)

Share



EMAIL

TWITTER

About the Author



Alex Sanséau

 Technical Lead
United Kingdom 

Alex is an experienced .NET professional with strong technical skills and management experience. Alex is specialised in Microsoft stack and web technology (C#, MVC, Azure, T-SQL, CSS, jQuery...) and he's passionate about software development best practices.

You may also be interested in...

[An Overview of the NuGet Ecosystem](#)

[AvePoint & Amazon Web Services: SharePoint in AWS](#)



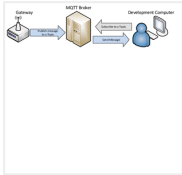
[Azure WebState](#)



[Intel IoT Gateway: Windows 10 Getting Started Guide](#)



[IOBOT : Build your own Azure powered Robots](#)



[Intel® IoT Gateways: Publishing Data to an MQTT Broker Using Python](#)

Comments and Discussions

You must Sign In to use this message board.		
Search Comments <input type="text"/>		<input type="button" value="Go"/>
		First Prev Next
Are packages really persisted?		1
Liero_ 7-Dec-15 23:35		
Download count		1
walterhuang 5-Nov-15 21:40		
Great Article		2
Tim Corey 18-Jun-15 8:31		
Security?		4
Member 4315982 11-Jun-15 19:35		
Deploying to azure and using these packages		2
Stuart Dobson 24-May-15 13:55		
Where is the downloadable sample?		3
emmag 30-Apr-15 19:23		
Growing interest in private NuGet servers		1
Member 11444684 11-Feb-15 4:13		
Don't use this if you need to host hundreds of packages		3
Xavier Decoster 8-Feb-15 7:43		
My vote of 5		2
GastonV 6-Feb-15 10:35		
How about database?		3

Grzegorz Mrozik 5-Feb-15 21:16[Refresh](#)**1**

 General  News  Suggestion  Question  Bug  Answer  Joke  Praise  Rant  Admin

[Permalink](#) | [Advertise](#) | [Privacy](#) | [Terms of Use](#) | [Mobile](#)
Web03 | 2.8.160115.1 | Last Updated 5 Feb 2015

Select Language ▼

Layout: [fixed](#) | [fluid](#)

Article Copyright 2015 by Alex Sanséau
Everything else Copyright © [CodeProject](#), 1999-2016