

**Description:** Azure is an ever expanding set of cloud services engineered by Microsoft. Microsoft's technologies, Infrastructure or its development platforms are being widely used around the globe. This post explains how to successfully deploy a project to Azure environment using Visual Studio Team Services (VSTS).

# Azure Deployment using Visual Studio Team Services (VSTS) and .NET Core

## What is Microsoft Azure

Microsoft Azure is the latest technology in web hosting in web development and people don't forget it's not only about web hosting. A lot of features are coming with the package such as Azure cosmos DB, Load Balancing, Web services, Performance monitoring and Cost Management of the account you have purchased.

Microsoft Azure has been around for a few years now and you've probably heard of it. It's the Microsoft cloud platform that competes directly with Amazon's AWS (Amazon Web Services) and Google's Google Cloud. If you don't really know what "the cloud" is, I suggest you really look into it as I believe it's the future.

It's free and you only need a credit card and a Microsoft email account like Hotmail or Outlook. Beware, Azure is free for now. You get free credit which is valid for one month. Some services cost a monthly fee, for others, you pay per use. Just having an empty Azure account is always free. In this post, we're going to host a default web application in Azure using an App Service.

## Visual Studio Team Services (VSTS)

VSTS is Microsoft's cloud version of Team Foundation Service (TFS), which is their tool for continuous integration and deployment, development, measurement, source control, and Agile tools like SCRUM and Kanban boards. And also it VSTS is an extension of the Microsoft Visual Studio Architecture that allows it to encompass developers and/or development teams. Visual Studio is a software development environment built on the .NET framework that is designed for managing projects and development work in the variety of languages including Visual C# .NET, Visual C++.NET, Visual Basic .NET, Visual J# .NET and ASP.NET.

Just like with Azure, it's free and you need a Microsoft email address, like Hotmail or Outlook. For the free version of VSTS, you get quite a bit of functionality, like 240 free build minutes per month and unlimited free private Git repositories. With an MSDN subscription, you get some extra functionality, but the free tier is good enough for this post.

.NET development in Azure with Visual Studio Team Services | T168

[https://www.youtube.com/watch?v=wkVElc\\_sk88](https://www.youtube.com/watch?v=wkVElc_sk88)

## .NET Core

For the last few years, Microsoft has focused on .NET Core, an open source multi-platform subset of the .NET Framework. While it still has some issues, even in the latest .NET Core 2.1 release. It's a bit different than the full .NET Framework, but not a lot. You may have to download and install the latest SDK, which can all be found on the .NET Core GitHub page. Visual Studio support for .NET Core starts from Visual Studio 2017, so be sure you're up to date, otherwise you can probably follow along with a regular .NET Framework ASP.NET Web Application.

## Creating a .NET Core Project

So let's put these tools together. We start by creating a new (private) project in VSTS, which is pretty simple. In fact, you probably already created one when you created a VSTS account. I kept the default, MyFirstProject. When you browse to the Code tab of your project in VSTS, you can initialize your repository with a *README.md* file and a *.gitignore* file (choose Visual Studio).



### *Initialize a repository in VSTS.*

You can copy the link or clone directly into Visual Studio on the next page in the right upper corner under the “**Clone**” button.

Once you have the repository locally, we can start by adding some code. Since we're focusing on deploying to Azure using VSTS, it doesn't really matter what code you have, so just create an ASP.NET Core Web Application in Visual Studio 2017. You can pick a regular Web Application (which will use Razor View Pages by default) or a Model-View-Controller project, whichever you fancy. Once the default web application template loads, you can commit and push it to your Git repository and we can start.

## Azure App Services

For the next step, we're going to mess around in Azure a bit. Log in to your Azure portal and find your App Services. Simply click the "Add" button, pick a "Web App" and hit "Create". On the next page, you can set some settings for your web app, such as the URL at which your app will be hosted (which always ends at .azurewebsites.net).

### ***New Azure Web App***

You'll also have to create a resource group or pick an existing one. A resource group is simply a way to group certain resources for a quick overview. For example, you can have a separate resource group for dev, test, acceptance and production or for your customer portal and your internal tools, it's all up to you.

The most important setting here is the App Service Plan/Location. This is basically a server that you pay for. A new plan is generated for you, but you can create your own as well. You have different price tiers with different hardware specifications and Azure functionality, like CPU, memory, custom domains, staging slots, and daily backups.

The location of your plan determines where the server is physically located. For me, in the Netherlands, West Europe is ideal because I know it's in Amsterdam. Just go with the default Standard S1 plan. Even if you don't have free credits, you can delete it after you've read this blog post and it will only cost you like \$0.03.

## **Setting Up Deployment from Azure**

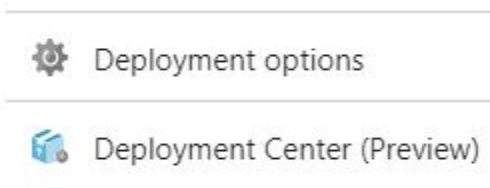
For now, we're staying in Azure. You've probably seen the "**Build and Release**" button in VSTS. That's where you can create new build and deployment pipelines.

A build pipeline can automatically build and test your software after a push to Git. This includes restoring NuGet packages, minifying and bundling any HTML, CSS and JavaScript you might have, executing task runners and basically anything you'd ever need to make your software build and run. When your latest commit is built and tested, VSTS can create an artifact for publication.

The deployment pipeline can download the artifact from a build pipeline and put it in Azure (or on your own on-premise servers). The trigger for a deployment is usually that a build has succeeded, but can also be a push to Git or a manual trigger.

When you're building and testing your software automatically, this is called Continuous Integration (or CI) and when your software is also automatically deployed, this is called Continuous Deployment (or CD). There's something in between, which holds off the deployment until a user explicitly presses a button and gives the "okay" sign, this is called Continuous Delivery. When you're doing both CI and Continuous Delivery and/or Deployment, you're doing "CI/CD".

Let's set this up with pretty much a single button click. You've got two options in Azure. In your Web App, either configure deployment from the "**Deployment options**" menu item or use the newer "**Deployment Center (Preview)**" which, as the name suggests, is still in preview.



### ***Deployment options in Azure***

## **Deployment Options**

Click the "**Deployment options**" menu item and simply fill in the options you want to use.

[Home](#) > [App Services](#) > [imunique](#) > Deployment option

## Deployment option

Set up deployment option


\* Choose Source  
Visual Studio Team Services

\* Choose your account  
sanderrossel


\* Choose a project  
MyFirstProject/MyFirstProject

\* Choose branch  
master

Performance Test  
Not Configured





Click here for help if you cannot see your VSTS projects.



### ***Deployment settings in Azure***

If you now browse to the Deployment options in your Web App, you'll see that Azure now shows all your commits. If you push a new commit to your repository, you can even see the build status in Azure because Azure linked it to your VSTS account.

WED 08/08

	Added a .NET Core web project. VSTS Building 9:13 PM
	Added README.md, .gitignore (VisualStudio) files VSTS Active 9:11 PM

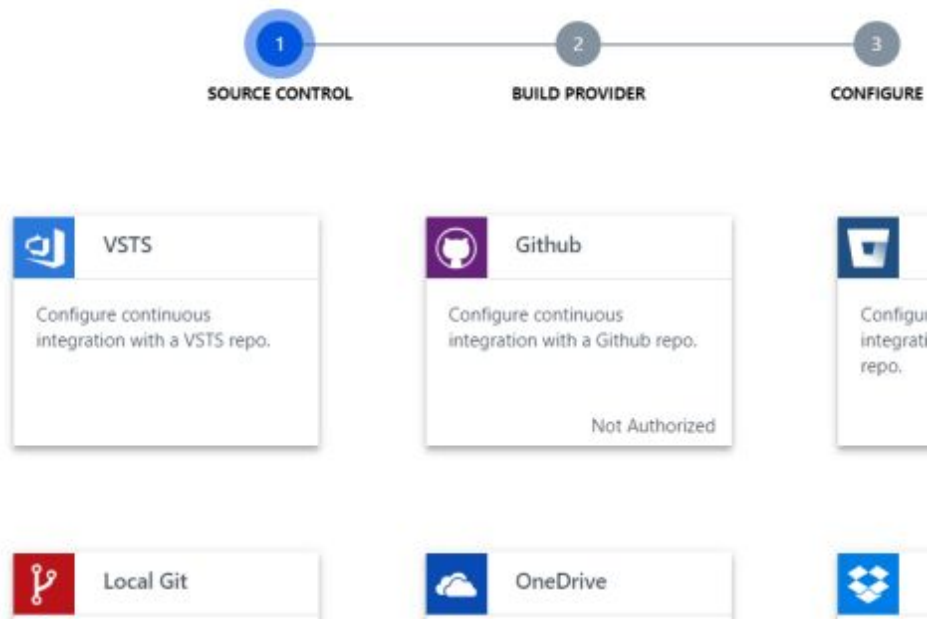
### ***Continuous deployment in Azure***

When deployment is done, browse to your Azure website and you can see your website is up and running. Try changing something on the homepage, push it to Git, wait a few minutes, and see your website get updated.

### **Deployment Center (Preview)**

It seems like Microsoft is planning to replace the “Deployment options” with the “Deployment Center (Preview)”. If you have already set up CI/CD with the Deployment options, you’ll see your commits in here too. For the purpose of this blog, we’re going to **“Disconnect”** (button in the top menu).

Now go to the Deployment Center (Preview) and you should see various source control options, like VSTS, Github and Bitbucket. It’s a little more visually appealing than most Azure panels. So pick VSTS and continue. Next, you have to pick a build provider.



### ***Deployment center in Azure***

In the next panel, you can pick your VSTS account and repository. Unfortunately, at the time of writing this blog, there seems to be a bug here which prevents me from picking an account. It worked in the past and the Azure team said they have fixed the problem in an upcoming release.

Anyway, if you continue, Azure creates a build pipeline and a deployment pipeline in VSTS. It gives you complete control over your builds and deployments as you can tweak them however you like. At the same time, the Deployment Center now shows you which builds have been deployed.

### **Pipelines in VSTS**

So browse to your VSTS and check out the build. The user interface isn't always intuitive, but you should be able to find it. You can edit the build and click around a bit. You shouldn't need to change it, but you can.

... > imunique - CI

Tasks Variables Triggers Options Retention

## Process

Build process



### Get sources

MyFirstProject

master

## Phase 1

Run on agent



### Restore

.NET Core



### Build

.NET Core



### Test

.NET Core



### Publish

.NET Core



### Publish Artifact

Publish Build Artifacts

## VSTS build

The deployment is a little different. There's just one build for every piece of code, but there can be multiple deployments, for example, for your entire DTAP street (dev, test, acceptance, and production). Azure created a release with a single environment, "**Dev**".





### VSTS release

You can change the deployment in the “**Tasks**” for that environment (either click the Dev tile or select Tasks in the upper menu). The tasks look kind of similar to the VSTS build and can be different for each environment. Mostly, your tasks will be more or less the same and you’ll only have some values that differ per environment. You can use “**Variables**” for these different values, but that’s out of the scope of this post.

It’s also pretty easy to clone an entire environment (including its tasks and variables). Just hover over the tile of the environment you’d like to clone and a “**Clone**” button will show up. Click it and you get a new environment that’s exactly like the one you cloned. Further setup of builds and releases is out of scope of this post, but you can click around if you like.

When you’re done playing around, simply delete the release and the build pipelines (in that order). Also make sure you disconnect the deployment in the Azure Deployment Center.

## Deployment from Visual Studio 2017

Finally, let’s switch back to Visual Studio. You can do the exact same thing in Visual Studio. Right click on the project you want to release (which is your web project) and choose “**Overview**” in the drop down menu. In the page that opens, go to “**Publish**” (in the menu on the left).

Here you can publish or set up your CI/CD. Click the “**Start**” button under “**Continuous Delivery**” and you can once again choose a VSTS subscription and an App Service (it creates a new App Service by default, but you can also select an existing one). Simply clicking the “**Ok**” button will have the same outcome as what we did in the Deployment Center. It will create a new build and deployment pipeline that is also visible in your Deployment Center. Pretty cool! Going back to the Publish page in Visual Studio, you can also just publish directly. This is pretty useful if you want to release something to Azure *right now*. Simply click the “**Start**” button under “**Publish**”. You can now either create a new Azure App Service or pick an existing one. Just follow the wizard and your application will be published to Azure directly, without a build or

deployment pipeline. Be careful though, your local build will be deployed, which will often be different than a build from the build server.

## Reference

<https://www.codeproject.com/Articles/1255892/Azure-Deployment-using-Visual-Studio-Team-Services>

<https://searchwindevelopment.techtarget.com/definition/Visual-Studio-Team-System>