

Contents

[Cloud Services Documentation](#)

[Overview](#)

[What is Cloud Services?](#)

[Cloud service config files and packaging](#)

[Get Started](#)

[Example .NET Cloud Service](#)

[Example Python for Visual Studio Cloud Service](#)

[Set up a hybrid HPC cluster with Microsoft HPC Pack](#)

[How To](#)

[Plan](#)

[Virtual machine sizes](#)

[Updates](#)

[Develop](#)

[Create PHP web and worker roles](#)

[Build and deploy a Node.js application](#)

[Build a Node.js web application using Express](#)

[Storage and Visual Studio](#)

[Blob storage and connected services](#)

[Queue storage and connected services](#)

[Table storage and connected services](#)

[Configure traffic rules for a role](#)

[Handle Cloud Service lifecycle events](#)

[Socket.io \(Node.js\)](#)

[Use Twilio to make a phone call \(.NET\)](#)

[Configure start up tasks](#)

[Create startup tasks](#)

[Common startup tasks](#)

[Use a task to Install .NET on a Cloud Service role](#)

[Configure Remote Desktop](#)

[Portal](#)

[PowerShell](#)

[Visual Studio](#)

[Deploy](#)

[Create and deploy a cloud service in portal](#)

[Create an empty cloud service container in PowerShell](#)

[Configure a custom domain name](#)

[Connect to a custom Domain Controller](#)

[Manage service](#)

[Common management tasks](#)

[Configure Cloud Service](#)

[Manage a Cloud Service using Azure Automation](#)

[Configure automatic scaling](#)

[Use Python to manage Azure Resources](#)

[Mitigating speculative execution](#)

[Guest OS patches](#)

[Guest OS retirement](#)

[Retirement policy](#)

[Family 1 retirement notice](#)

[Guest OS release news](#)

[Cloud Services Role config XPath cheat sheet](#)

[Manage certificates](#)

[Cloud Services and management certificates](#)

[Configure SSL](#)

[Monitor](#)

[Monitor cloud service](#)

[Use performance counters](#)

[Test performance](#)

[Test with Visual Studio Profiler](#)

[Enable diagnostics](#)

[Azure PowerShell](#)

[.NET](#)

[Visual Studio](#)

[Store and view diagnostic data in Azure Storage](#)

[Trace Cloud Service with Diagnostics](#)

[Troubleshoot](#)

[Debug](#)

[Options for a Cloud Service](#)

[Local Cloud Service with Visual Studio](#)

[Published Cloud Service with Visual Studio](#)

[Cloud Service allocation failure](#)

[Common causes of Cloud Service roles recycling](#)

[Default TEMP folder size too small for role](#)

[Common deployment problems](#)

[Role failed to start](#)

[Recovery guidance](#)

[Cloud Services FAQ](#)

[Application and service availability FAQ](#)

[Configuration and management FAQ](#)

[Connectivity and networking FAQ](#)

[Deployment FAQ](#)

[Reference](#)

[Code samples](#)

[.csdef XML Schema](#)

[LoadBalancerProbe Schema](#)

[WebRole Schema](#)

[WorkerRole Schema](#)

[NetworkTrafficRules Schema](#)

[.cscfg XML Schema](#)

[Role Schema](#)

[NetworkConfiguration Schema](#)

[REST](#)

[Resources](#)

[Azure Roadmap](#)

[Learning path](#)

[MSDN forum](#)

[Pricing](#)

[Pricing calculator](#)

[Service updates](#)

[Videos](#)

Learn how to use Cloud Services to host and run highly available, scalable cloud applications and APIs. Tutorials, API references and other documentation show you how to manage virtual machine hosts and configure, patch, and install software.

Learn about Cloud Services

Cloud Services Video Library

Get Started with Cloud Services using .NET

Get Started with Cloud Services using Python

Reference

Languages

[.csdef XML Schema](#)

[LoadBalancerProbe Schema](#)

[WebRole Schema](#)

[WorkerRole Schema](#)

[NetworkTrafficRules Schema](#)

[.cscfg XML Schema](#)

[Role Schema](#)

[NetworkConfiguration Schema](#)

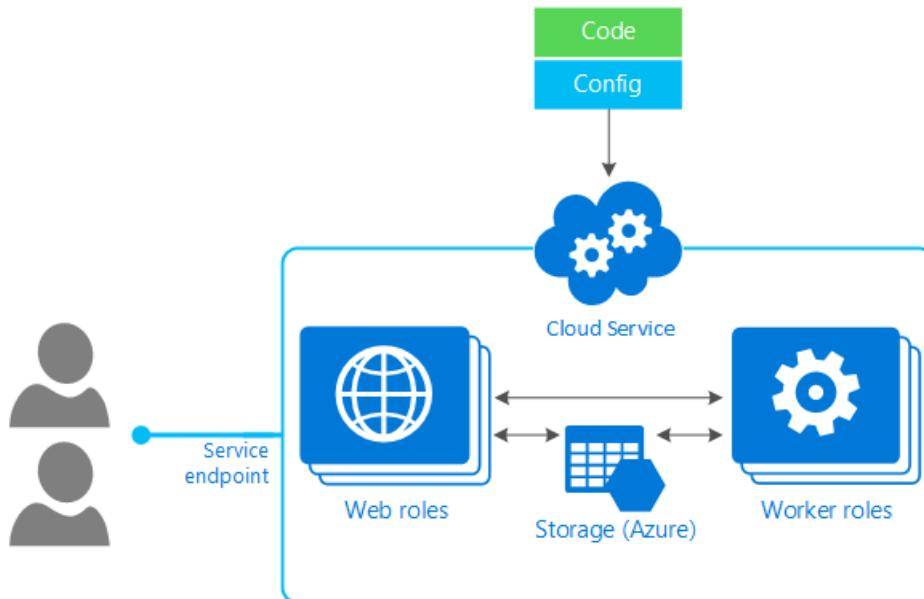
REST

[REST API](#)

Overview of Azure Cloud Services

12/18/2018 • 3 minutes to read • [Edit Online](#)

Azure Cloud Services is an example of a [platform as a service](#) (PaaS). Like [Azure App Service](#), this technology is designed to support applications that are scalable, reliable, and inexpensive to operate. In the same way that App Service is hosted on virtual machines (VMs), so too is Azure Cloud Services. However, you have more control over the VMs. You can install your own software on VMs that use Azure Cloud Services, and you can access them remotely.



More control also means less ease of use. Unless you need the additional control options, it's typically quicker and easier to get a web application up and running in the Web Apps feature of App Service compared to Azure Cloud Services.

There are two types of Azure Cloud Services roles. The only difference between the two is how your role is hosted on the VMs:

- **Web role:** Automatically deploys and hosts your app through IIS.
- **Worker role:** Does not use IIS, and runs your app standalone.

For example, a simple application might use just a single web role, serving a website. A more complex application might use a web role to handle incoming requests from users, and then pass those requests on to a worker role for processing. (This communication might use [Azure Service Bus](#) or [Azure Queue storage](#).)

As the preceding figure suggests, all the VMs in a single application run in the same cloud service. Users access the application through a single public IP address, with requests automatically load balanced across the application's VMs. The platform [scales and deploys](#) the VMs in an Azure Cloud Services application in a way that avoids a single point of hardware failure.

Even though applications run in VMs, it's important to understand that Azure Cloud Services provides PaaS, not infrastructure as a service (IaaS). Here's one way to think about it. With IaaS, such as Azure Virtual Machines, you first create and configure the environment your application runs in. Then you deploy your application into this environment. You're responsible for managing much of this world, by doing things such as deploying new patched versions of the operating system in each VM. In PaaS, by contrast, it's as if the environment already exists. All you have to do is deploy your application. Management of the platform it runs on, including deploying new versions of

the operating system, is handled for you.

Scaling and management

With Azure Cloud Services, you don't create virtual machines. Instead, you provide a configuration file that tells Azure how many of each you'd like, such as "three web role instances" and "two worker role instances." The platform then creates them for you. You still choose [what size](#) those backing VMs should be, but you don't explicitly create them yourself. If your application needs to handle a greater load, you can ask for more VMs, and Azure creates those instances. If the load decreases, you can shut down those instances and stop paying for them.

An Azure Cloud Services application is typically made available to users via a two-step process. A developer first [uploads the application](#) to the platform's staging area. When the developer is ready to make the application live, they use the Azure portal to swap staging with production. This [switch between staging and production](#) can be done with no downtime, which lets a running application be upgraded to a new version without disturbing its users.

Monitoring

Azure Cloud Services also provides monitoring. Like Virtual Machines, it detects a failed physical server and restarts the VMs that were running on that server on a new machine. But Azure Cloud Services also detects failed VMs and applications, not just hardware failures. Unlike Virtual Machines, it has an agent inside each web and worker role, and so it's able to start new VMs and application instances when failures occur.

The PaaS nature of Azure Cloud Services has other implications, too. One of the most important is that applications built on this technology should be written to run correctly when any web or worker role instance fails. To achieve this, an Azure Cloud Services application shouldn't maintain state in the file system of its own VMs. Unlike VMs created with Virtual Machines, writes made to Azure Cloud Services VMs aren't persistent. There's nothing like a Virtual Machines data disk. Instead, an Azure Cloud Services application should explicitly write all state to Azure SQL Database, blobs, tables, or some other external storage. Building applications this way makes them easier to scale and more resistant to failure, which are both important goals of Azure Cloud Services.

Next steps

- [Create a cloud service app in .NET](#)
- [Create a cloud service app in Node.js](#)
- [Create a cloud service app in PHP](#)
- [Create a cloud service app in Python](#)

What is the Cloud Service model and how do I package it?

11/7/2018 • 9 minutes to read • [Edit Online](#)

A cloud service is created from three components, the service definition (*.csdef*), the service config (*.cscfg*), and a service package (*.cspkg*). Both the **ServiceDefinition.csdef** and **ServiceConfig.cscfg** files are XML-based and describe the structure of the cloud service and how it's configured; collectively called the model. The **ServicePackage.cspkg** is a zip file that is generated from the **ServiceDefinition.csdef** and among other things, contains all the required binary-based dependencies. Azure creates a cloud service from both the **ServicePackage.cspkg** and the **ServiceConfig.cscfg**.

Once the cloud service is running in Azure, you can reconfigure it through the **ServiceConfig.cscfg** file, but you cannot alter the definition.

What would you like to know more about?

- I want to know more about the [ServiceDefinition.csdef](#) and [ServiceConfig.cscfg](#) files.
- I already know about that, give me [some examples](#) on what I can configure.
- I want to create the [ServicePackage.cspkg](#).
- I am using Visual Studio and I want to...
 - [Create a cloud service](#)
 - [Reconfigure an existing cloud service](#)
 - [Deploy a Cloud Service project](#)
 - [Remote desktop into a cloud service instance](#)

ServiceDefinition.csdef

The **ServiceDefinition.csdef** file specifies the settings that are used by Azure to configure a cloud service. The [Azure Service Definition Schema \(.csdef File\)](#) provides the allowable format for a service definition file. The following example shows the settings that can be defined for the Web and Worker roles:

```

<?xml version="1.0" encoding="utf-8"?>
<ServiceDefinition name="MyServiceName"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
  <WebRole name="WebRole1" vmsize="Medium">
    <Sites>
      <Site name="Web">
        <Bindings>
          <Binding name="HttpIn" endpointName="HttpIn" />
        </Bindings>
      </Site>
    </Sites>
    <Endpoints>
      <InputEndpoint name="HttpIn" protocol="http" port="80" />
      <InternalEndpoint name="InternalHttpIn" protocol="http" />
    </Endpoints>
    <Certificates>
      <Certificate name="Certificate1" storeLocation="LocalMachine" storeName="My" />
    </Certificates>
    <Imports>
      <Import moduleName="Connect" />
      <Import moduleName="Diagnostics" />
      <Import moduleName="RemoteAccess" />
      <Import moduleName="RemoteForwarder" />
    </Imports>
    <LocalResources>
      <LocalStorage name="localStoreOne" sizeInMB="10" />
      <LocalStorage name="localStoreTwo" sizeInMB="10" cleanOnRoleRecycle="false" />
    </LocalResources>
    <Startup>
      <Task commandLine="Startup.cmd" executionContext="limited" taskType="simple" />
    </Startup>
  </WebRole>

  <WorkerRole name="WorkerRole1">
    <ConfigurationSettings>
      <Setting name="DiagnosticsConnectionString" />
    </ConfigurationSettings>
    <Imports>
      <Import moduleName="RemoteAccess" />
      <Import moduleName="RemoteForwarder" />
    </Imports>
    <Endpoints>
      <InputEndpoint name="Endpoint1" protocol="tcp" port="10000" />
      <InternalEndpoint name="Endpoint2" protocol="tcp" />
    </Endpoints>
  </WorkerRole>
</ServiceDefinition>

```

You can refer to the [Service Definition Schema](#) for a better understanding of the XML schema used here, however, here is a quick explanation of some of the elements:

Sites

Contains the definitions for websites or web applications that are hosted in IIS7.

InputEndpoints

Contains the definitions for endpoints that are used to contact the cloud service.

InternalEndpoints

Contains the definitions for endpoints that are used by role instances to communicate with each other.

ConfigurationSettings

Contains the setting definitions for features of a specific role.

Certificates

Contains the definitions for certificates that are needed for a role. The previous code example shows a certificate that is used for the configuration of Azure Connect.

LocalResources

Contains the definitions for local storage resources. A local storage resource is a reserved directory on the file system of the virtual machine in which an instance of a role is running.

Imports

Contains the definitions for imported modules. The previous code example shows the modules for Remote Desktop Connection and Azure Connect.

Startup

Contains tasks that are run when the role starts. The tasks are defined in a .cmd or executable file.

ServiceConfiguration.cscfg

The configuration of the settings for your cloud service is determined by the values in the **ServiceConfiguration.cscfg** file. You specify the number of instances that you want to deploy for each role in this file. The values for the configuration settings that you defined in the service definition file are added to the service configuration file. The thumbprints for any management certificates that are associated with the cloud service are also added to the file. The [Azure Service Configuration Schema \(.cscfg File\)](#) provides the allowable format for a service configuration file.

The service configuration file is not packaged with the application, but is uploaded to Azure as a separate file and is used to configure the cloud service. You can upload a new service configuration file without redeploying your cloud service. The configuration values for the cloud service can be changed while the cloud service is running. The following example shows the configuration settings that can be defined for the Web and Worker roles:

```
<?xml version="1.0"?>
<ServiceConfiguration serviceName="MyServiceName"
  xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfiguration">
  <Role name="WebRole1">
    <Instances count="2" />
    <ConfigurationSettings>
      <Setting name="SettingName" value="SettingValue" />
    </ConfigurationSettings>

    <Certificates>
      <Certificate name="CertificateName" thumbprint="CertThumbprint" thumbprintAlgorithm="sha1" />
      <Certificate name="Microsoft.WindowsAzure.Plugins.RemoteAccess.PasswordEncryption"
        thumbprint="CertThumbprint" thumbprintAlgorithm="sha1" />
    </Certificates>
  </Role>
</ServiceConfiguration>
```

You can refer to the [Service Configuration Schema](#) for better understanding the XML schema used here, however, here is a quick explanation of the elements:

Instances

Configures the number of running instances for the role. To prevent your cloud service from potentially becoming unavailable during upgrades, it is recommended that you deploy more than one instance of your web-facing roles. By deploying more than one instance, you are adhering to the guidelines in the [Azure Compute Service Level Agreement \(SLA\)](#), which guarantees 99.95% external connectivity for Internet-facing roles when two or more role instances are deployed for a service.

ConfigurationSettings

Configures the settings for the running instances for a role. The name of the `<Setting>` elements must match the setting definitions in the service definition file.

Certificates

Configures the certificates that are used by the service. The previous code example shows how to define the certificate for the RemoteAccess module. The value of the *thumbprint* attribute must be set to the thumbprint of the certificate to use.

NOTE

The thumbprint for the certificate can be added to the configuration file by using a text editor. Or, the value can be added on the **Certificates** tab of the **Properties** page of the role in Visual Studio.

Defining ports for role instances

Azure allows only one entry point to a web role. Meaning that all traffic occurs through one IP address. You can configure your websites to share a port by configuring the host header to direct the request to the correct location. You can also configure your applications to listen to well-known ports on the IP address.

The following sample shows the configuration for a web role with a website and web application. The website is configured as the default entry location on port 80, and the web applications are configured to receive requests from an alternate host header that is called "mail.mysite.cloudapp.net".

```
<WebRole>
  <ConfigurationSettings>
    <Setting name="DiagnosticsConnectionString" />
  </ConfigurationSettings>
  <Endpoints>
    <InputEndpoint name="HttpIn" protocol="http" port="80" />
    <InputEndpoint name="Https" protocol="https" port="443" certificate="SSL"/>
    <InputEndpoint name="NetTcp" protocol="tcp" port="808" certificate="SSL"/>
  </Endpoints>
  <LocalResources>
    <LocalStorage name="Sites" cleanOnRoleRecycle="true" sizeInMB="100" />
  </LocalResources>
  <Site name="Mysite" packageDir="Sites\Mysite">
    <Bindings>
      <Binding name="http" endpointName="HttpIn" />
      <Binding name="https" endpointName="Https" />
      <Binding name="tcp" endpointName="NetTcp" />
    </Bindings>
  </Site>
  <Site name="MailSite" packageDir="MailSite">
    <Bindings>
      <Binding name="mail" endpointName="HttpIn" hostheader="mail.mysite.cloudapp.net" />
    </Bindings>
    <VirtualDirectory name="artifacts" />
    <VirtualApplication name="storageproxy">
      <VirtualDirectory name="packages" packageDir="Sites\storageProxy\packages"/>
    </VirtualApplication>
  </Site>
</WebRole>
```

Changing the configuration of a role

You can update the configuration of your cloud service while it is running in Azure, without taking the service offline. To change configuration information, you can either upload a new configuration file, or edit the configuration file in place and apply it to your running service. The following changes can be made to the configuration of a service:

- **Changing the values of configuration settings**

When a configuration setting changes, a role instance can choose to apply the change while the instance is

online, or to recycle the instance gracefully and apply the change while the instance is offline.

- **Changing the service topology of role instances**

Topology changes do not affect running instances, except where an instance is being removed. All remaining instances generally do not need to be recycled; however, you can choose to recycle role instances in response to a topology change.

- **Changing the certificate thumbprint**

You can only update a certificate when a role instance is offline. If a certificate is added, deleted, or changed while a role instance is online, Azure gracefully takes the instance offline to update the certificate and bring it back online after the change is complete.

Handling configuration changes with Service Runtime Events

The [Azure Runtime Library](#) includes the `Microsoft.WindowsAzure.ServiceRuntime` namespace, which provides classes for interacting with the Azure environment from a role. The `RoleEnvironment` class defines the following events that are raised before and after a configuration change:

- **Changing event**

This occurs before the configuration change is applied to a specified instance of a role giving you a chance to take down the role instances if necessary.

- **Changed event**

Occurs after the configuration change is applied to a specified instance of a role.

NOTE

Because certificate changes always take the instances of a role offline, they do not raise the `RoleEnvironment.Changing` or `RoleEnvironment.Changed` events.

ServicePackage.cspkg

To deploy an application as a cloud service in Azure, you must first package the application in the appropriate format. You can use the **CSPack** command-line tool (installed with the [Azure SDK](#)) to create the package file as an alternative to Visual Studio.

CSPack uses the contents of the service definition file and service configuration file to define the contents of the package. **CSPack** generates an application package file (.cspkg) that you can upload to Azure by using the [Azure portal](#). By default, the package is named `[ServiceDefinitionFileName].cspkg`, but you can specify a different name by using the `/out` option of **CSPack**.

CSPack is located at

```
C:\Program Files\Microsoft SDKs\Azure\.NET SDK\[sdk-version]\bin\
```

NOTE

`CSPack.exe` (on windows) is available by running the **Microsoft Azure Command Prompt** shortcut that is installed with the SDK.

Run the `CSPack.exe` program by itself to see documentation about all the possible switches and commands.

TIP

Run your cloud service locally in the **Microsoft Azure Compute Emulator**, use the `/copyonly` option. This option copies the binary files for the application to a directory layout from which they can be run in the compute emulator.

Example command to package a cloud service

The following example creates an application package that contains the information for a web role. The command specifies the service definition file to use, the directory where binary files can be found, and the name of the package file.

```
cspack [DirectoryName]\[ServiceDefinition]
/role:[RoleName];[RoleBinariesDirectory]
/sites:[RoleName];[VirtualPath];[PhysicalPath]
/out:[OutputFileName]
```

If the application contains both a web role and a worker role, the following command is used:

```
cspack [DirectoryName]\[ServiceDefinition]
/out:[OutputFileName]
/role:[RoleName];[RoleBinariesDirectory]
/sites:[RoleName];[VirtualPath];[PhysicalPath]
/role:[RoleName];[RoleBinariesDirectory];[RoleAssemblyName]
```

Where the variables are defined as follows:

VARIABLE	VALUE
[DirectoryName]	The subdirectory under the root project directory that contains the .csdef file of the Azure project.
[ServiceDefinition]	The name of the service definition file. By default, this file is named ServiceDefinition.csdef.
[OutputFileName]	The name for the generated package file. Typically, this is set to the name of the application. If no file name is specified, the application package is created as [ApplicationName].cspkg.
[RoleName]	The name of the role as defined in the service definition file.
[RoleBinariesDirectory]	The location of the binary files for the role.
[VirtualPath]	The physical directories for each virtual path defined in the Sites section of the service definition.
[PhysicalPath]	The physical directories of the contents for each virtual path defined in the site node of the service definition.
[RoleAssemblyName]	The name of the binary file for the role.

Next steps

I'm creating a cloud service package and I want to...

- [Setup remote desktop for a cloud service instance](#)
- [Deploy a Cloud Service project](#)

I am using Visual Studio and I want to...

- [Create a new cloud service](#)
- [Reconfigure an existing cloud service](#)
- [Deploy a Cloud Service project](#)

- Setup remote desktop for a cloud service instance

Get started with Azure Cloud Services and ASP.NET

12/20/2018 • 29 minutes to read • [Edit Online](#)

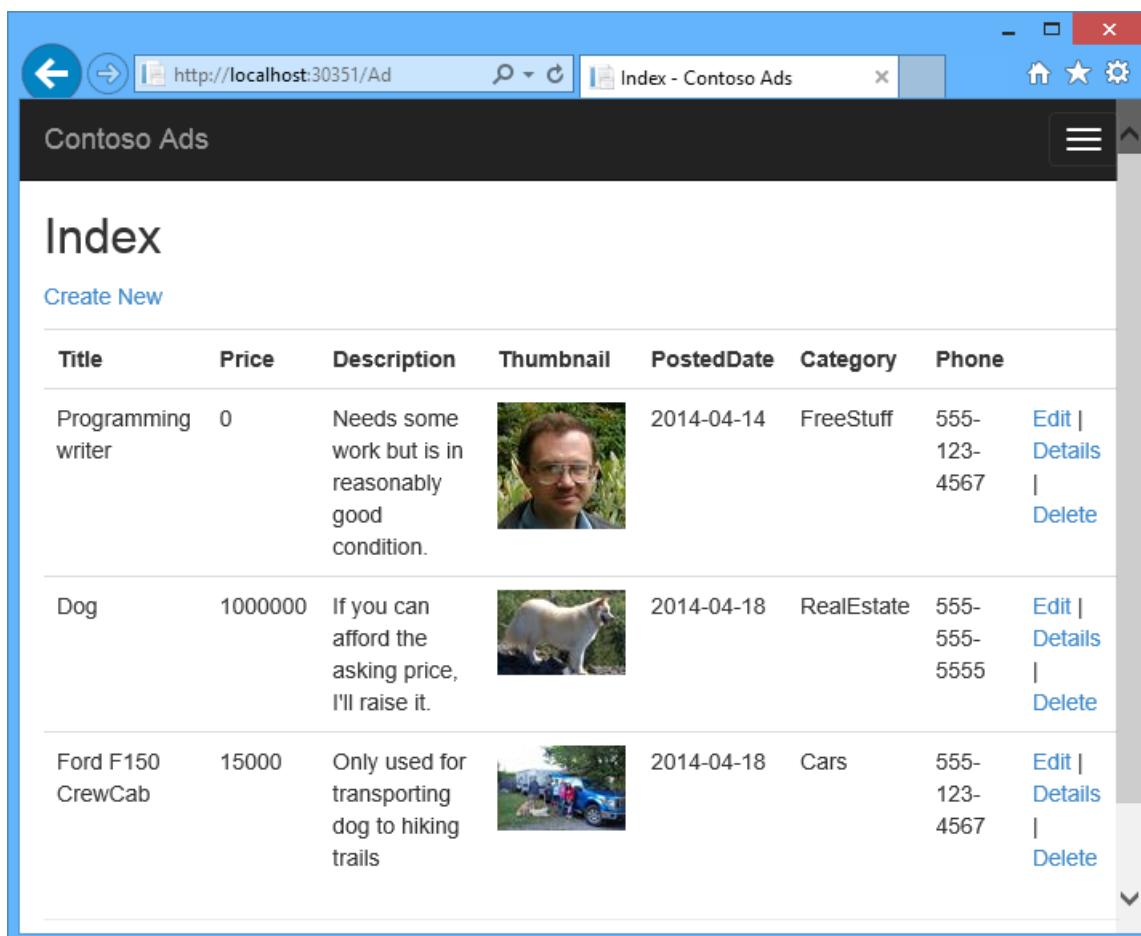
Overview

This tutorial shows how to create a multi-tier .NET application with an ASP.NET MVC front-end, and deploy it to an [Azure cloud service](#). The application uses [Azure SQL Database](#), the [Azure Blob service](#), and the [Azure Queue service](#). You can [download the Visual Studio project](#) from the MSDN Code Gallery.

The tutorial shows you how to build and run the application locally, how to deploy it to Azure and run in the cloud, and how to build it from scratch. You can start by building from scratch and then do the test and deploy steps afterward if you prefer.

Contoso Ads application

The application is an advertising bulletin board. Users create an ad by entering text and uploading an image. They can see a list of ads with thumbnail images, and they can see the full-size image when they select an ad to see the details.



The screenshot shows a web browser window with the URL <http://localhost:30351/Ad>. The title bar says "Index - Contoso Ads". The main content area has a header "Index" and a "Create New" button. Below is a table with the following data:

Title	Price	Description	Thumbnail	PostedDate	Category	Phone	
Programming writer	0	Needs some work but is in reasonably good condition.		2014-04-14	FreeStuff	555-123-4567	Edit Details Delete
Dog	1000000	If you can afford the asking price, I'll raise it.		2014-04-18	RealEstate	555-555-5555	Edit Details Delete
Ford F150 CrewCab	15000	Only used for transporting dog to hiking trails		2014-04-18	Cars	555-123-4567	Edit Details Delete

The application uses the [queue-centric work pattern](#) to off-load the CPU-intensive work of creating thumbnails to a back-end process.

Alternative architecture: App Service and WebJobs

This tutorial shows how to run both front-end and back-end in an Azure cloud service. An alternative is to run the front-end in [Azure App Service](#) and use the [WebJobs](#) feature for the back-end. For a tutorial that uses WebJobs,

see [Get Started with the Azure WebJobs SDK](#). For information about how to choose the services that best fit your scenario, see [Azure App Service, Cloud Services, and virtual machines comparison](#).

What you'll learn

- How to enable your machine for Azure development by installing the Azure SDK.
- How to create a Visual Studio cloud service project with an ASP.NET MVC web role and a worker role.
- How to test the cloud service project locally, using the Azure storage emulator.
- How to publish the cloud project to an Azure cloud service and test using an Azure storage account.
- How to upload files and store them in the Azure Blob service.
- How to use the Azure Queue service for communication between tiers.

Prerequisites

The tutorial assumes that you understand [basic concepts about Azure cloud services](#) such as *web role* and *worker role* terminology. It also assumes that you know how to work with [ASP.NET MVC](#) or [Web Forms](#) projects in Visual Studio. The sample application uses MVC, but most of the tutorial also applies to Web Forms.

You can run the app locally without an Azure subscription, but you'll need one to deploy the application to the cloud. If you don't have an account, you can [activate your MSDN subscriber benefits](#) or [sign up for a free trial](#).

The tutorial instructions work with either of the following products:

- Visual Studio 2013
- Visual Studio 2015
- Visual Studio 2017

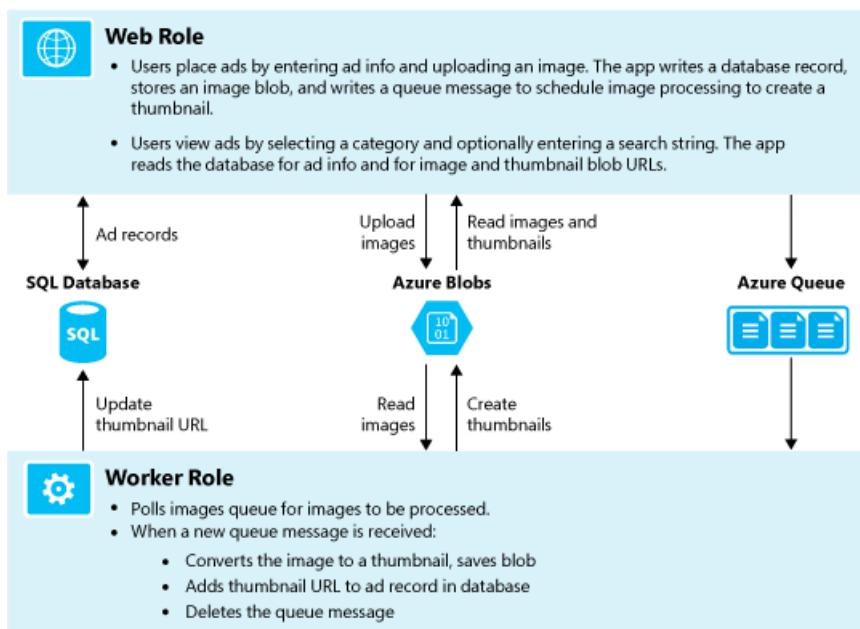
If you don't have one of these, Visual Studio may be installed automatically when you install the Azure SDK.

Application architecture

The app stores ads in a SQL database, using Entity Framework Code First to create the tables and access the data. For each ad, the database stores two URLs, one for the full-size image and one for the thumbnail.

	Name	Data Type	Allow Nulls
1	AdId	int	<input type="checkbox"/>
2	Title	nvarchar(100)	<input checked="" type="checkbox"/>
3	Price	int	<input type="checkbox"/>
4	Description	nvarchar(1000)	<input checked="" type="checkbox"/>
5	ImageURL	nvarchar(1000)	<input checked="" type="checkbox"/>
6	ThumbnailURL	nvarchar(1000)	<input checked="" type="checkbox"/>
7	PostedDate	datetime	<input type="checkbox"/>
8	Category	int	<input checked="" type="checkbox"/>
9	Phone	nvarchar(12)	<input checked="" type="checkbox"/>
10			<input type="checkbox"/>

When a user uploads an image, the front-end running in a web role stores the image in an [Azure blob](#), and it stores the ad information in the database with a URL that points to the blob. At the same time, it writes a message to an Azure queue. A back-end process running in a worker role periodically polls the queue for new messages. When a new message appears, the worker role creates a thumbnail for that image and updates the thumbnail URL database field for that ad. The following diagram shows how the parts of the application interact.



Set up the development environment

To start, set up your development environment with Visual Studio and the Azure SDK.

- Visual Studio 2017 includes the Azure SDK. If you're using VS2017, no additional setup is needed for the development environment.
- For Visual Studio 2015, click the following link to install the [Azure SDK for Visual Studio 2015](#).
- For Visual Studio 2013, click the following link to install the [Azure SDK for Visual Studio 2013](#).
- If you don't have Visual Studio installed, use the following to install [Visual Studio 2017](#) with the Azure SDK.

NOTE

Depending on the number of the SDK dependencies already on your machine, installing the SDK could take a long time, from several minutes to a half hour or more.

Download and run the completed solution

- Download and unzip the [completed solution](#).
- Start Visual Studio.
- From the **File** menu choose **Open Project**, navigate to where you downloaded the solution, and then open the solution file.
- Press CTRL+SHIFT+B to build the solution.

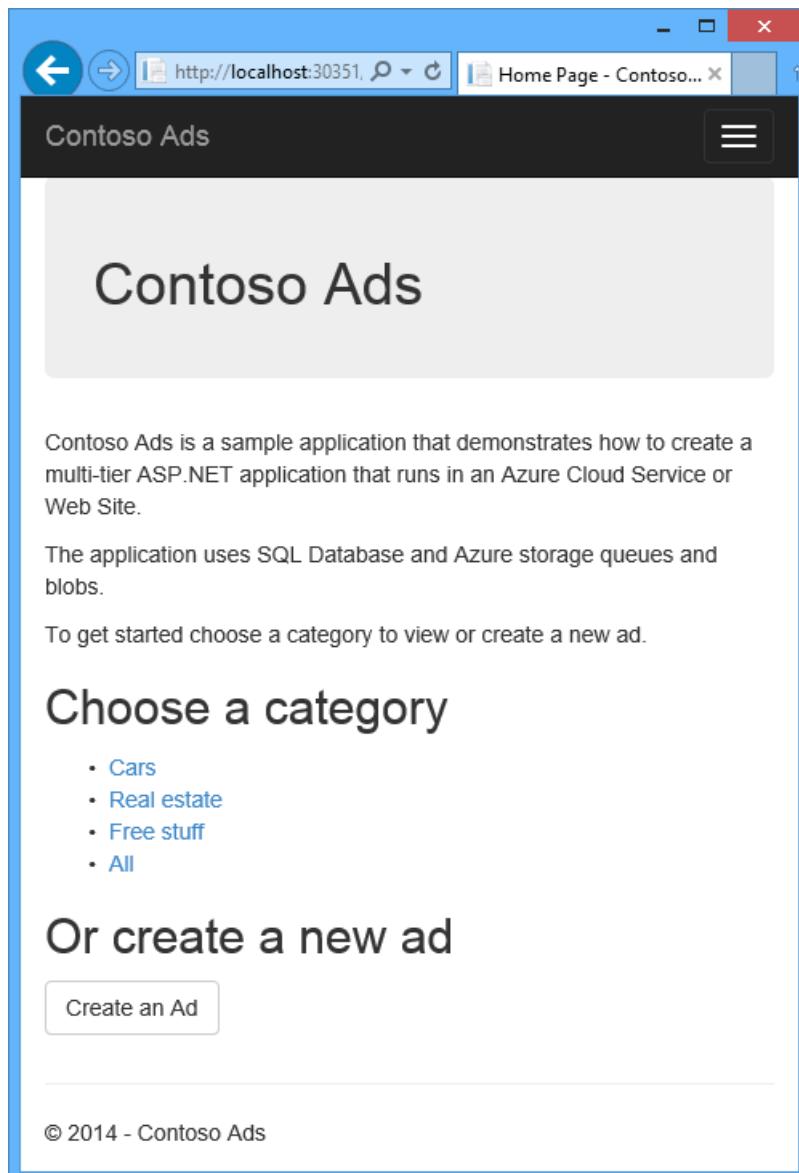
By default, Visual Studio automatically restores the NuGet package content, which was not included in the .zip file. If the packages don't restore, install them manually by going to the **Manage NuGet Packages for Solution** dialog box and clicking the **Restore** button at the top right.

- In **Solution Explorer**, make sure that **ContosoAdsCloudService** is selected as the startup project.
- If you're using Visual Studio 2015 or higher, change the SQL Server connection string in the application *Web.config* file of the ContosoAdsWeb project and in the *ServiceConfiguration.Local.cscfg* file of the ContosoAdsCloudService project. In each case, change "(localdb)\v11.0" to "(localdb)\MSSQLLocalDB".
- Press CTRL+F5 to run the application.

When you run a cloud service project locally, Visual Studio automatically invokes the Azure *compute emulator* and Azure *storage emulator*. The compute emulator uses your computer's resources to simulate

the web role and worker role environments. The storage emulator uses a [SQL Server Express LocalDB](#) database to simulate Azure cloud storage.

The first time you run a cloud service project, it takes a minute or so for the emulators to start up. When emulator startup is finished, the default browser opens to the application home page.



8. Click **Create an Ad**.
9. Enter some test data and select a *jpg* image to upload, and then click **Create**.

The screenshot shows a web browser window titled "Create - Contoso Ads". The main content area is titled "Create Ad". There are several input fields:

- Title:** Ford F150 CrewCab
- Price:** 15000
- Description:** Only used for transporting dog to
hiking trails
- Image file:** \\tdvkstra1\c\$\Users\tdvkst\ (Browse...)
- Category:** Cars
- Phone:** 555-0123

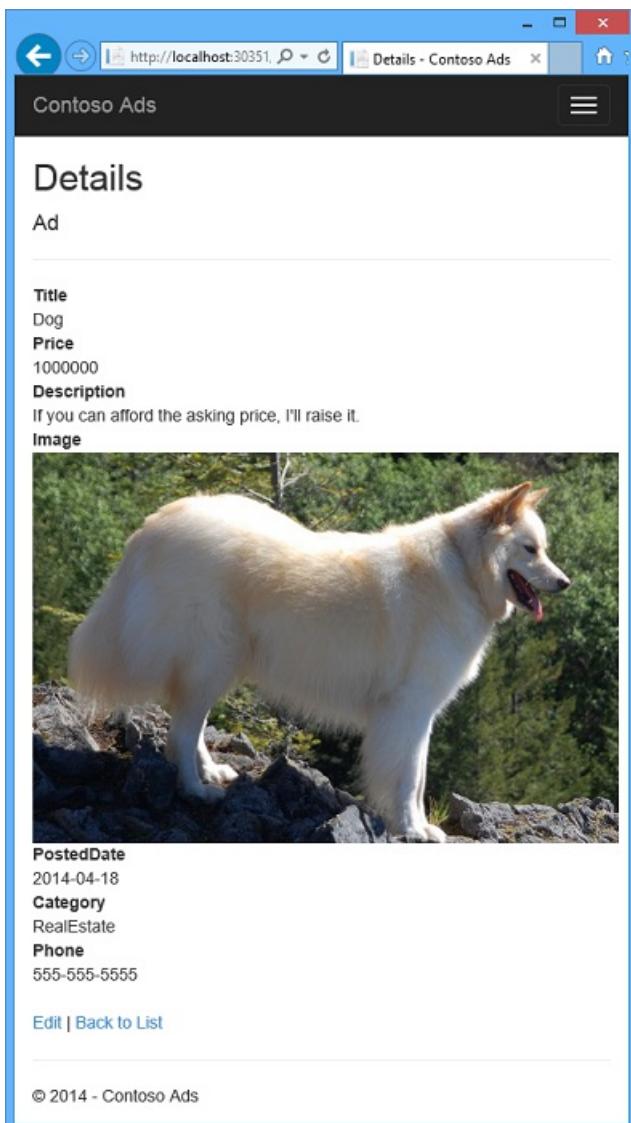
A "Create" button is at the bottom left, and a "Back to List" link is at the bottom center. The footer says "© 2014 - Contoso Ads".

The app goes to the Index page, but it doesn't show a thumbnail for the new ad because that processing hasn't happened yet.

10. Wait a moment and then refresh the Index page to see the thumbnail.

Title	Price	Description	Thumbnail	PostedDate	Category	Phone	
Programming writer	0	Needs some work but is in reasonably good condition.		2014-04-14	FreeStuff	555-123-4567	Edit Details Delete
Dog	1000000	If you can afford the asking price, I'll raise it.		2014-04-18	RealEstate	555-555-5555	Edit Details Delete
Ford F150 CrewCab	15000	Only used for transporting dog to hiking trails		2014-04-18	Cars	555-123-4567	Edit Details Delete

11. Click **Details** for your ad to see the full-size image.



You've been running the application entirely on your local computer, with no connection to the cloud. The storage emulator stores the queue and blob data in a SQL Server Express LocalDB database, and the application stores the ad data in another LocalDB database. Entity Framework Code First automatically created the ad database the first time the web app tried to access it.

In the following section you'll configure the solution to use Azure cloud resources for queues, blobs, and the application database when it runs in the cloud. If you wanted to continue to run locally but use cloud storage and database resources, you could do that. It's just a matter of setting connection strings, which you'll see how to do.

Deploy the application to Azure

You'll do the following steps to run the application in the cloud:

- Create an Azure cloud service.
- Create an Azure SQL database.
- Create an Azure storage account.
- Configure the solution to use your Azure SQL database when it runs in Azure.
- Configure the solution to use your Azure storage account when it runs in Azure.
- Deploy the project to your Azure cloud service.

Create an Azure cloud service

An Azure cloud service is the environment the application will run in.

1. In your browser, open the [Azure portal](#).

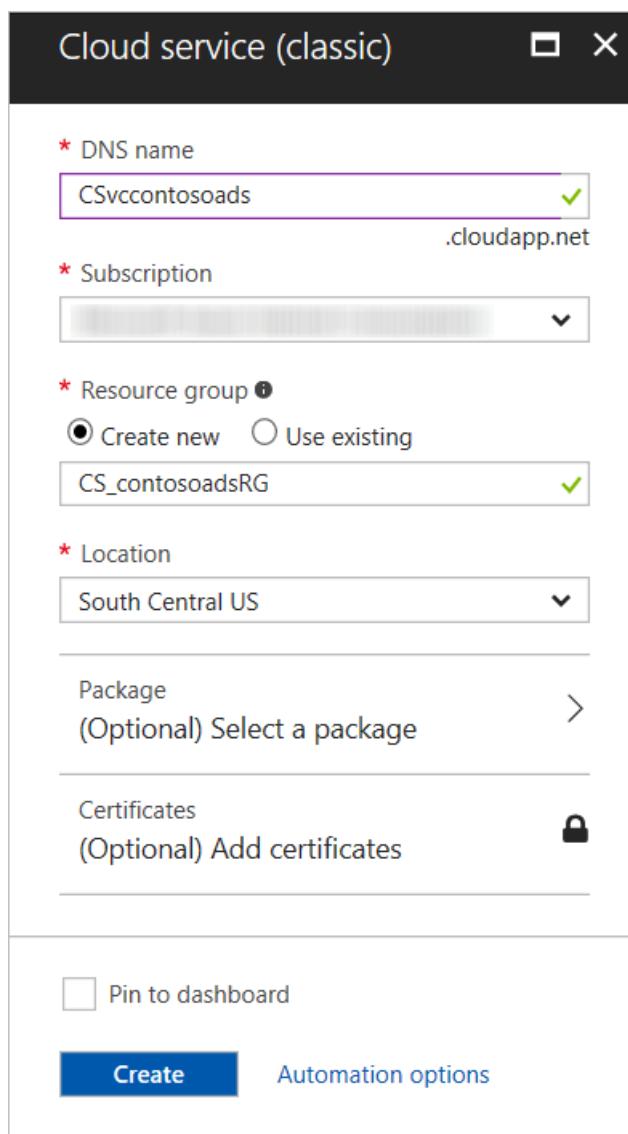
2. Click **Create a resource > Compute > Cloud Service**.
3. In the DNS name input box, enter a URL prefix for the cloud service.

This URL has to be unique. You'll get an error message if the prefix you choose is already in use.
4. Specify a new Resource group for the service. Click **Create new** and then type a name in the Resource group input box, such as CS_contosoadsRG.
5. Choose the region where you want to deploy the application.

This field specifies which datacenter your cloud service will be hosted in. For a production application, you'd choose the region closest to your customers. For this tutorial, choose the region closest to you.

6. Click **Create**.

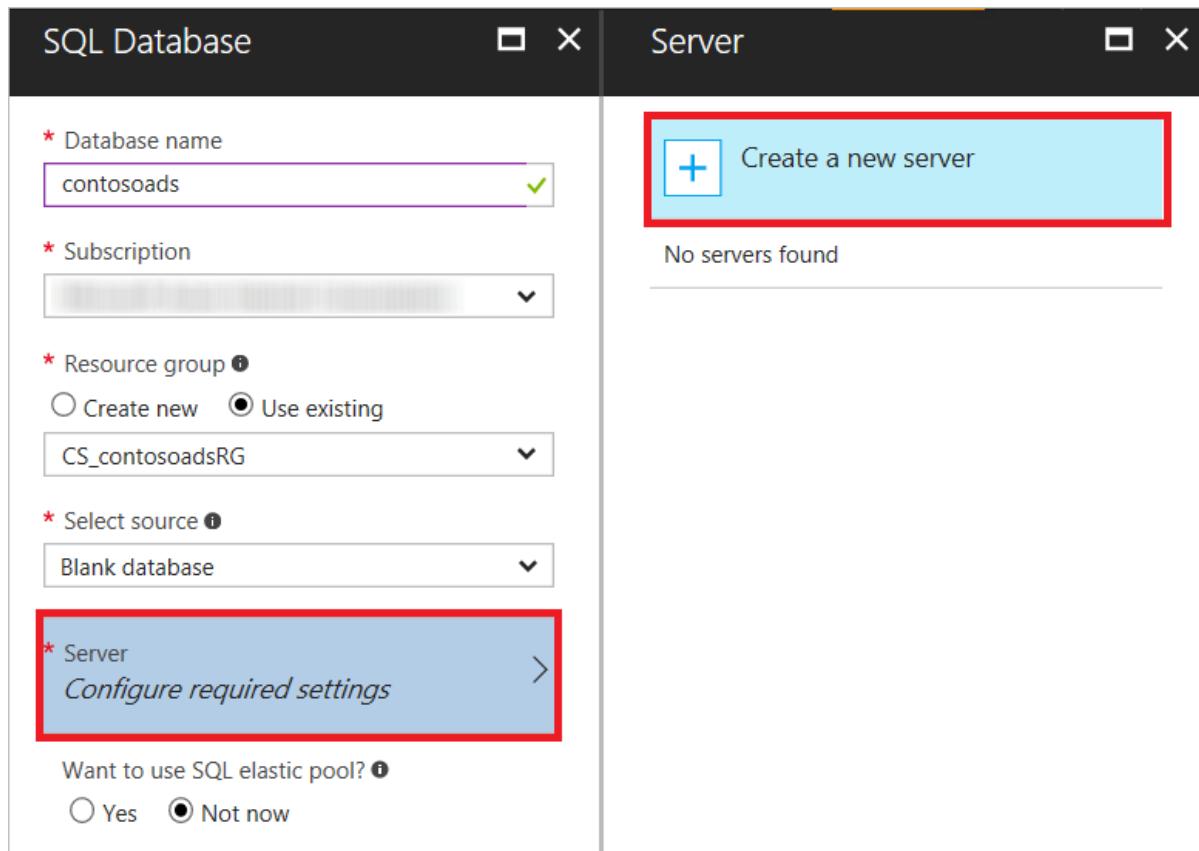
In the following image, a cloud service is created with the URL CSvccontosoads.cloudapp.net.



Create an Azure SQL database

When the app runs in the cloud, it will use a cloud-based database.

1. In the [Azure portal](#), click **Create a resource > Databases > SQL Database**.
2. In the **Database Name** box, enter *contosoads*.
3. In the **Resource group**, click **Use existing** and select the resource group used for the cloud service.
4. In the following image, click **Server - Configure required settings** and **Create a new server**.



Alternatively, if your subscription already has a server, you can select that server from the drop-down list.

5. In the **Server name** box, enter `csvcontosodbserver`.
6. Enter an administrator **Login Name** and **Password**.

If you selected **Create a new server**, you aren't entering an existing name and password here. You're entering a new name and password that you're defining now to use later when you access the database. If you selected a server that you created previously, you'll be prompted for the password to the administrative user account you already created.

7. Choose the same **Location** that you chose for the cloud service.

When the cloud service and database are in different datacenters (different regions), latency will increase and you will be charged for bandwidth outside the data center. Bandwidth within a data center is free.

8. Check **Allow azure services to access server**.
9. Click **Select** for the new server.

New server

* Server name
csvcontosodbserver .database.windows.net

* Server admin login
azureuser

* Password

* Confirm password

* Location
South Central US

Allow azure services to access server ?

Select

The dialog box has a dark header bar with 'New server' and standard window controls. Below is a light-colored form area with five input fields, each with a red asterisk indicating it's required. The first field contains 'csvcontosodbserver' followed by '.database.windows.net'. The second field contains 'azureuser'. The third and fourth fields are for password entry. The fifth field is a dropdown for location, showing 'South Central US'. At the bottom left is a checked checkbox for 'Allow azure services to access server' with a help icon. A large blue 'Select' button is at the bottom right.

10. Click **Create**.

Create an Azure storage account

An Azure storage account provides resources for storing queue and blob data in the cloud.

In a real-world application, you would typically create separate accounts for application data versus logging data, and separate accounts for test data versus production data. For this tutorial, you'll use just one account.

1. In the [Azure portal](#), click **Create a resource > Storage > Storage account - blob, file, table, queue**.
2. In the **Name** box, enter a URL prefix.

This prefix plus the text you see under the box will be the unique URL to your storage account. If the prefix you enter has already been used by someone else, you'll have to choose a different prefix.

3. Set the **Deployment model** to *Classic*.
4. Set the **Replication** drop-down list to **Locally redundant storage**.

When geo-replication is enabled for a storage account, the stored content is replicated to a secondary datacenter to enable failover if a major disaster occurs in the primary location. Geo-replication can incur additional costs. For test and development accounts, you generally don't want to pay for geo-replication. For more information, see [Create, manage, or delete a storage account](#).

5. In the **Resource group**, click **Use existing** and select the resource group used for the cloud service.
6. Set the **Location** drop-down list to the same region you chose for the cloud service.

When the cloud service and storage account are in different datacenters (different regions), latency will increase and you will be charged for bandwidth outside the data center. Bandwidth within a data center is free.

Azure affinity groups provide a mechanism to minimize the distance between resources in a data center, which can reduce latency. This tutorial does not use affinity groups. For more information, see [How to Create an Affinity Group in Azure](#).

7. Click **Create**.

Create storage account X

The cost of your storage account depends on the usage and the options you choose below.
[Learn more](#)

* Name *
csvcontosoads .core.windows.net

Deployment model *
[Resource manager](#) **Classic**

Account kind *
[General purpose](#)

Performance *
[Standard](#) [Premium](#)

Replication *
[Locally-redundant storage \(LRS\)](#)

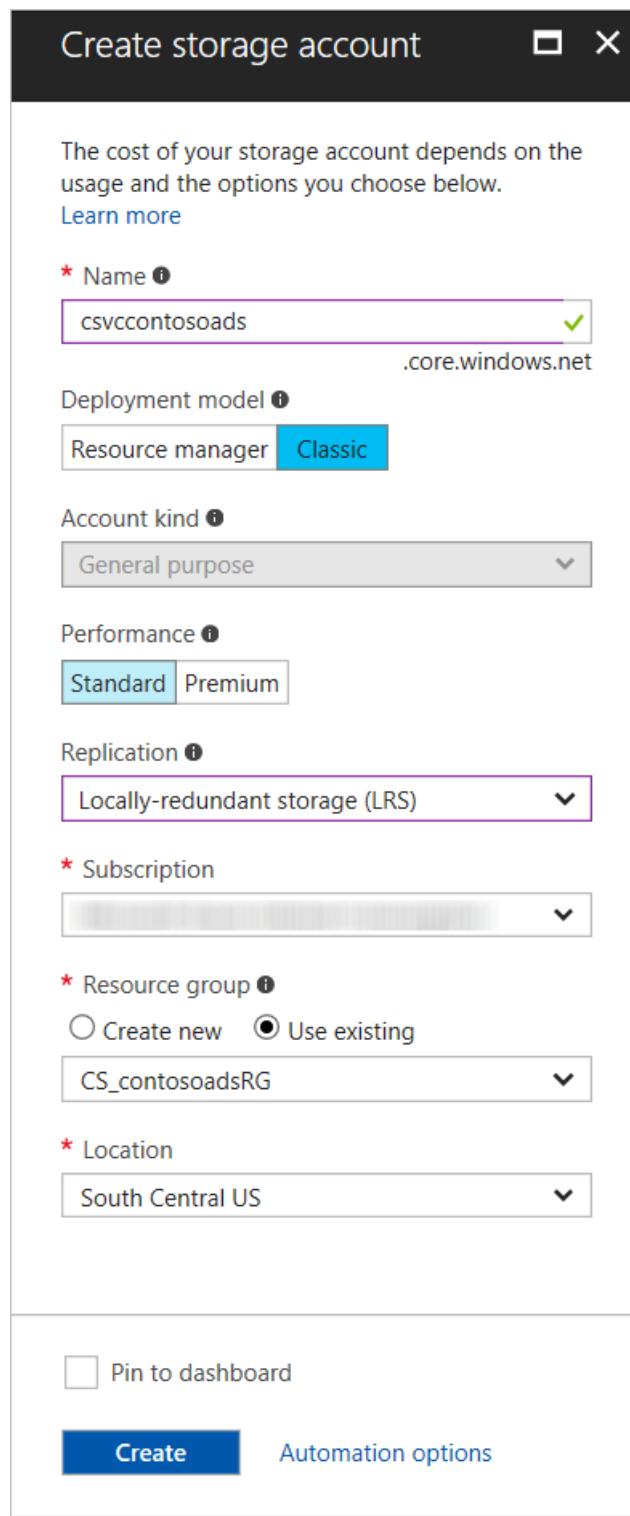
* Subscription
[\[Subscription\]](#)

* Resource group *
 Create new Use existing
[CS_contosoardsRG](#)

* Location
[South Central US](#)

Pin to dashboard

Create [Automation options](#)



In the image, a storage account is created with the URL `csvcontosoards.core.windows.net`.

Configure the solution to use your Azure SQL database when it runs in Azure

The web project and the worker role project each has its own database connection string, and each needs to point to the Azure SQL database when the app runs in Azure.

You'll use a [Web.config transform](#) for the web role and a cloud service environment setting for the worker role.

NOTE

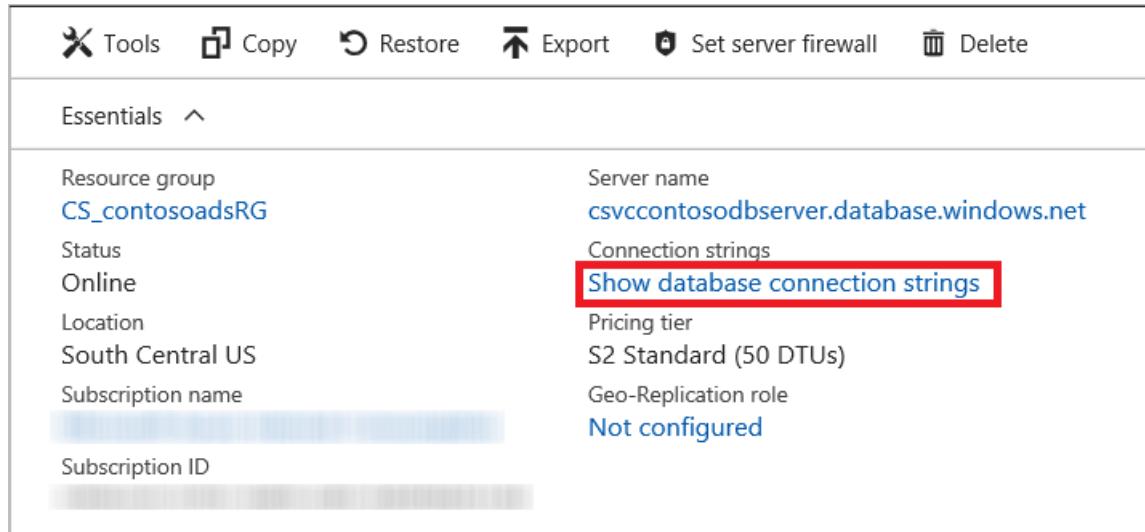
In this section and the next section, you store credentials in project files. [Don't store sensitive data in public source code repositories.](#)

1. In the ContosoAdsWeb project, open the *Web.Release.config* transform file for the application *Web.config* file, delete the comment block that contains a `<connectionStrings>` element, and paste the following code in its place.

```
<connectionStrings>
  <add name="ContosoAdsContext" connectionString="{connectionstring}"
    providerName="System.Data.SqlClient" xdt:Transform="SetAttributes" xdt:Locator="Match(name)"/>
</connectionStrings>
```

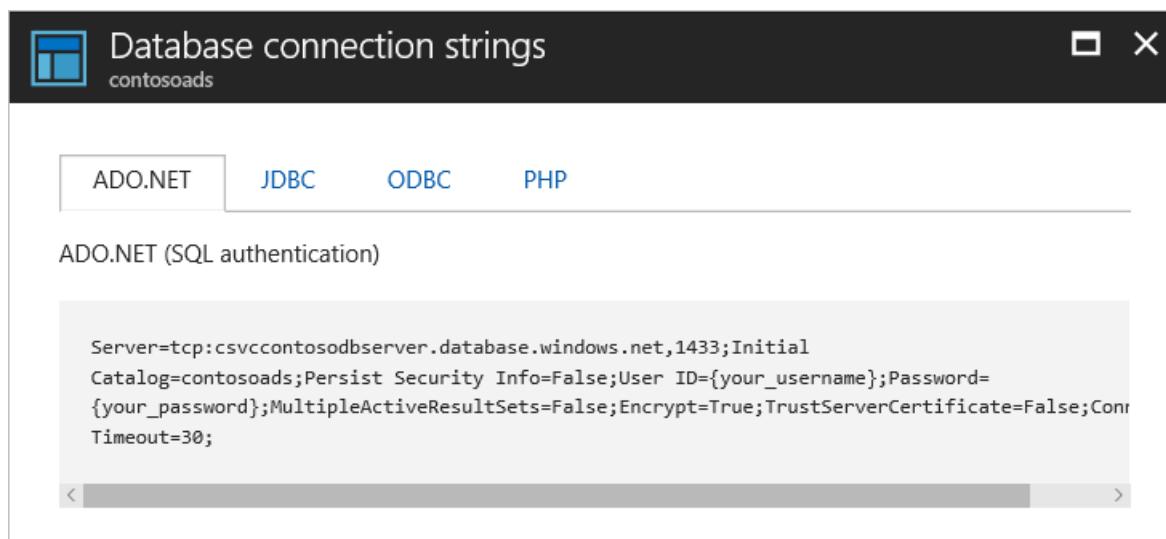
Leave the file open for editing.

2. In the [Azure portal](#), click **SQL Databases** in the left pane, click the database you created for this tutorial, and then click **Show connection strings**.



The screenshot shows the Azure portal's 'Essentials' blade for a database named 'CS_contosoadsRG'. The 'Connection strings' section is highlighted with a red box around the 'Show database connection strings' link. Other visible details include the server name 'csvccontosodbserver.database.windows.net', status 'Online', location 'South Central US', and geo-replication role 'Not configured'.

The portal displays connection strings, with a placeholder for the password.



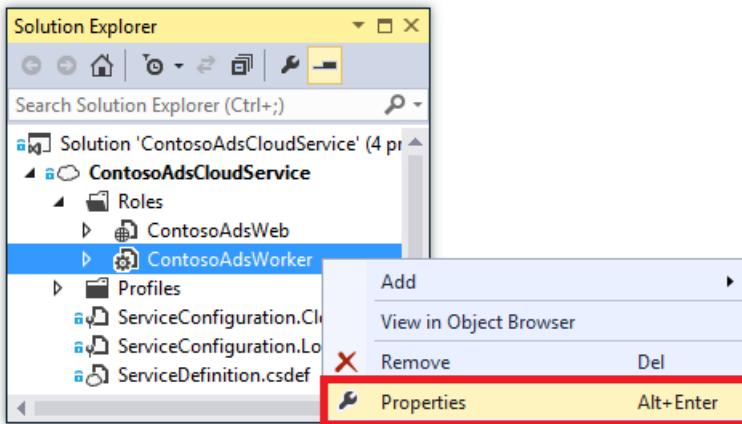
The screenshot shows the 'Database connection strings' dialog for the 'contosoads' database. The 'ADO.NET' tab is selected. The connection string is displayed as:

```
Server=tcp:csvccontosodbserver.database.windows.net,1433;Initial Catalog=contosoads;Persist Security Info=False;User ID={your_username};Password={your_password};MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Timeout=30;
```

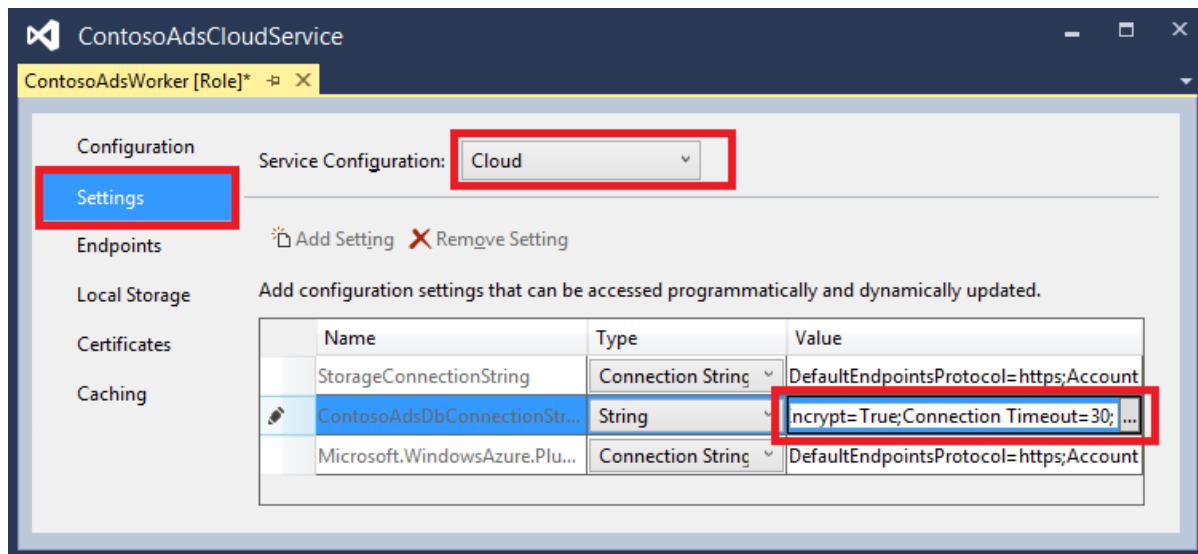
3. In the *Web.Release.config* transform file, delete `{connectionstring}` and paste in its place the ADO.NET connection string from the Azure portal.
4. In the connection string that you pasted into the *Web.Release.config* transform file, replace

{your_password_here} with the password you created for the new SQL database.

5. Save the file.
6. Select and copy the connection string (without the surrounding quotation marks) for use in the following steps for configuring the worker role project.
7. In **Solution Explorer**, under **Roles** in the cloud service project, right-click **ContosoAdsWorker** and then click **Properties**.



8. Click the **Settings** tab.
9. Change **Service Configuration** to **Cloud**.
10. Select the **Value** field for the **ContosoAdsDbConnectionString** setting, and then paste the connection string that you copied from the previous section of the tutorial.

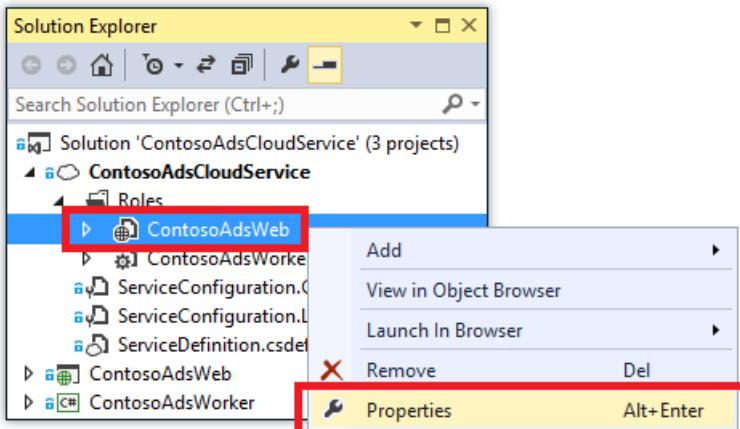


11. Save your changes.

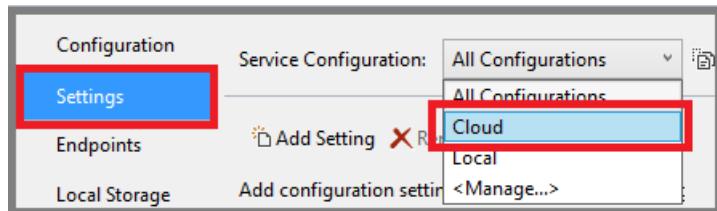
Configure the solution to use your Azure storage account when it runs in Azure

Azure storage account connection strings for both the web role project and the worker role project are stored in environment settings in the cloud service project. For each project, there is a separate set of settings to be used when the application runs locally and when it runs in the cloud. You'll update the cloud environment settings for both web and worker role projects.

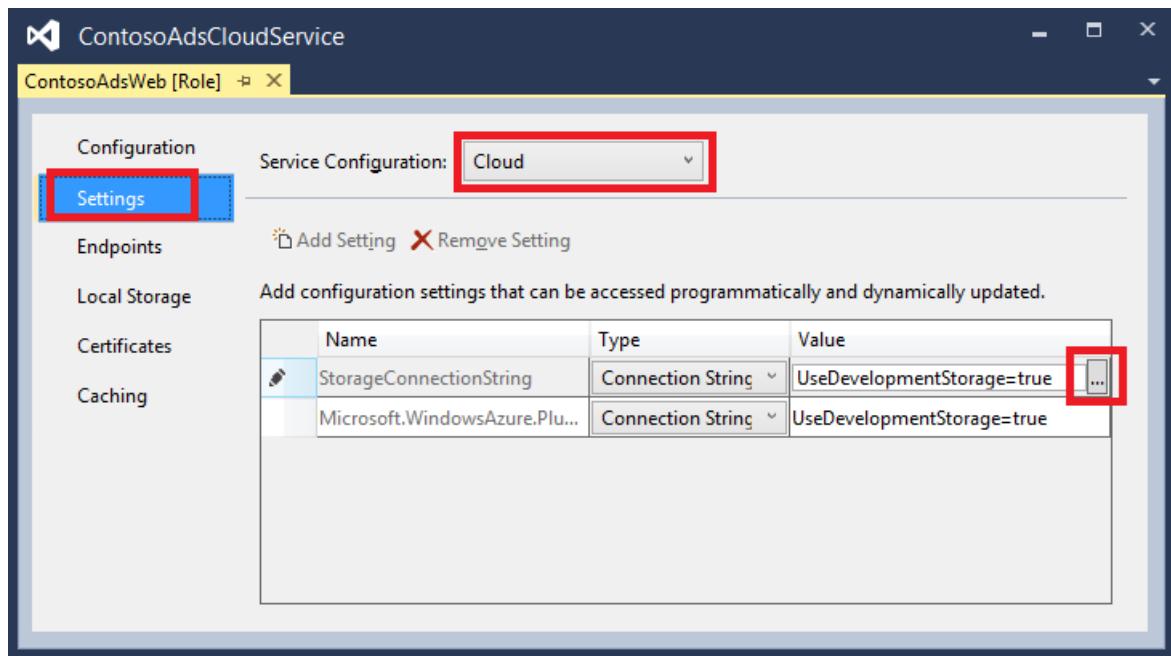
1. In **Solution Explorer**, right-click **ContosoAdsWeb** under **Roles** in the **ContosoAdsCloudService** project, and then click **Properties**.



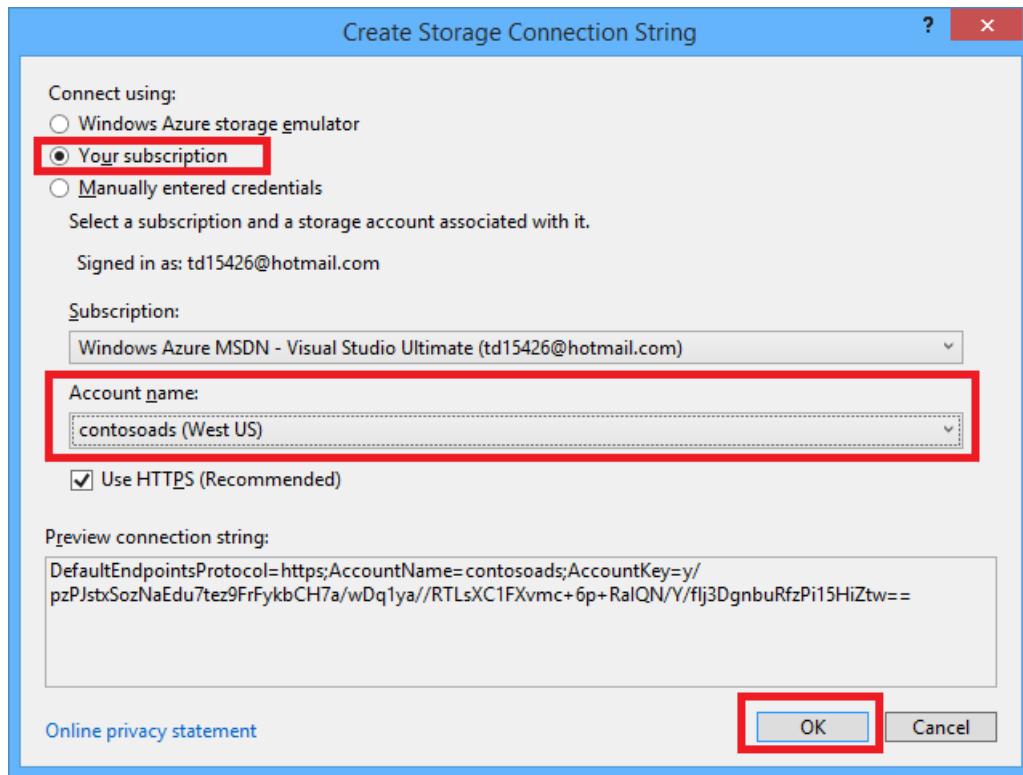
2. Click the **Settings** tab. In the **Service Configuration** drop-down box, choose **Cloud**.



3. Select the **StorageConnectionString** entry, and you'll see an ellipsis (...) button at the right end of the line. Click the ellipsis button to open the **Create Storage Account Connection String** dialog box.



4. In the **Create Storage Connection String** dialog box, click **Your subscription**, choose the storage account that you created earlier, and then click **OK**. If you're not already logged in, you'll be prompted for your Azure account credentials.



5. Save your changes.
6. Follow the same procedure that you used for the `StorageConnectionString` connection string to set the `Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString` connection string.

This connection string is used for logging.
7. Follow the same procedure that you used for the **ContosoAdsWeb** role to set both connection strings for the **ContosoAdsWorker** role. Don't forget to set **Service Configuration** to **Cloud**.

The role environment settings that you have configured using the Visual Studio UI are stored in the following files in the ContosoAdsCloudService project:

- `ServiceDefinition.csdef` - Defines the setting names.
- `ServiceConfiguration.Cloud.cscfg` - Provides values for when the app runs in the cloud.
- `ServiceConfiguration.Local.cscfg` - Provides values for when the app runs locally.

For example, the `ServiceDefinition.csdef` includes the following definitions:

```
<ConfigurationSettings>
    <Setting name="StorageConnectionString" />
    <Setting name="ContosoAdsDbConnectionString" />
</ConfigurationSettings>
```

And the `ServiceConfiguration.Cloud.cscfg` file includes the values you entered for those settings in Visual Studio.

```

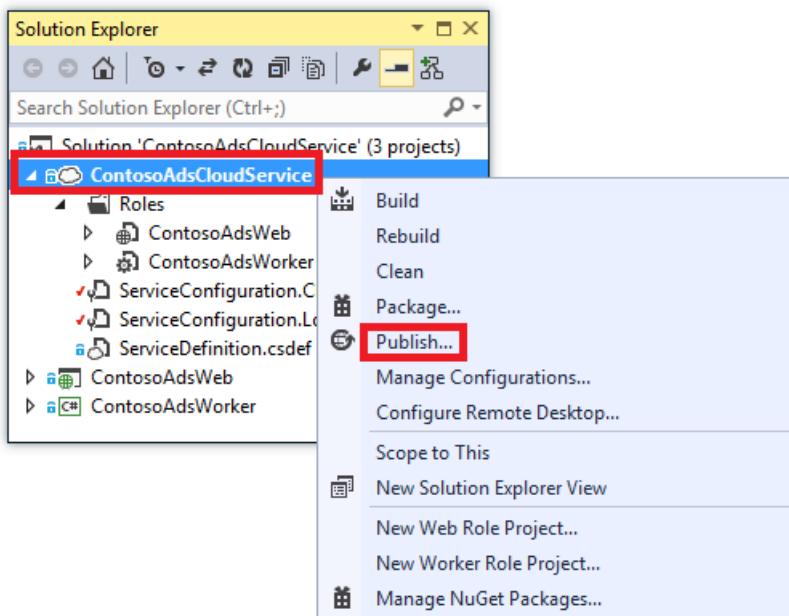
<Role name="ContosoAdsWorker">
  <Instances count="1" />
  <ConfigurationSettings>
    <Setting name="StorageConnectionString" value="{yourconnectionstring}" />
    <Setting name="ContosoAdsDbConnectionString" value="{yourconnectionstring}" />
    <!-- other settings not shown -->
  </ConfigurationSettings>
  <!-- other settings not shown -->
</Role>

```

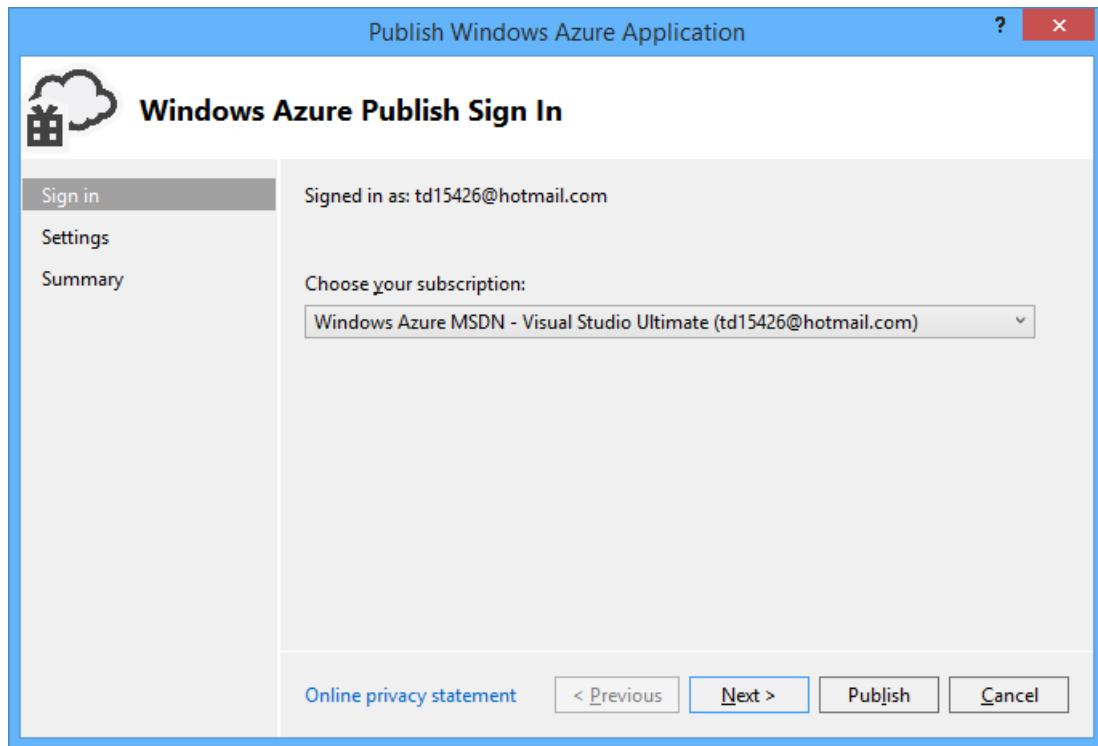
The `<Instances>` setting specifies the number of virtual machines that Azure will run the worker role code on. The [Next steps](#) section includes links to more information about scaling out a cloud service,

Deploy the project to Azure

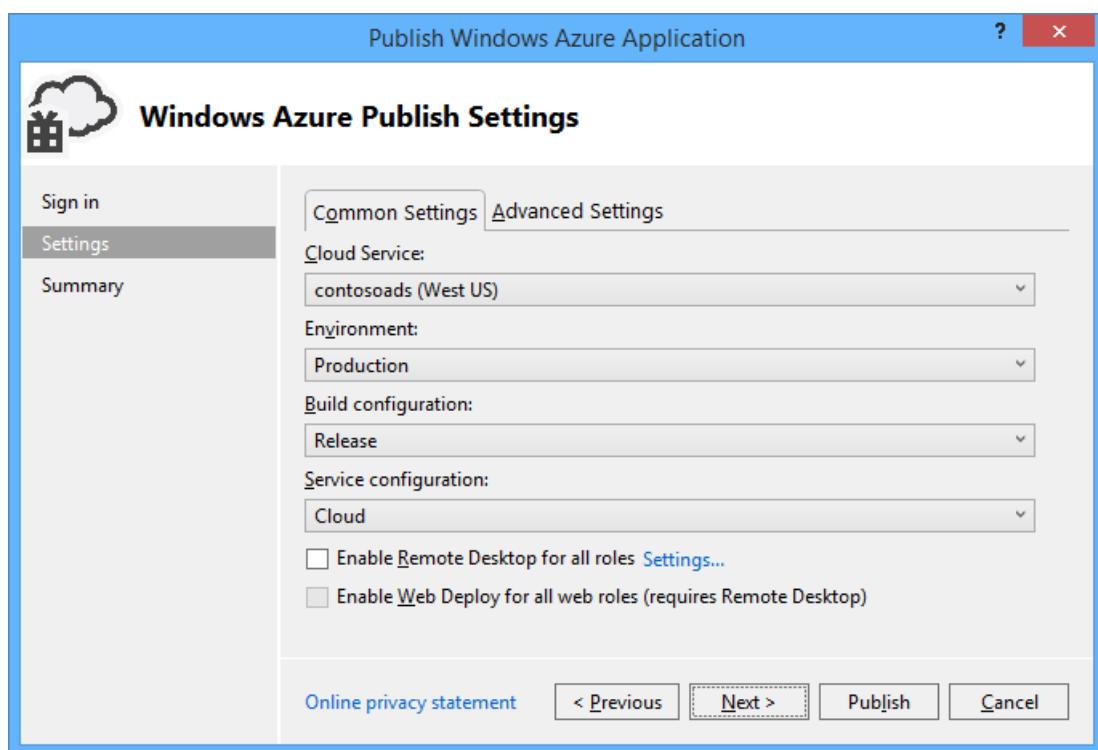
1. In **Solution Explorer**, right-click the **ContosoAdsCloudService** cloud project and then select **Publish**.



2. In the **Sign in** step of the **Publish Azure Application** wizard, click **Next**.

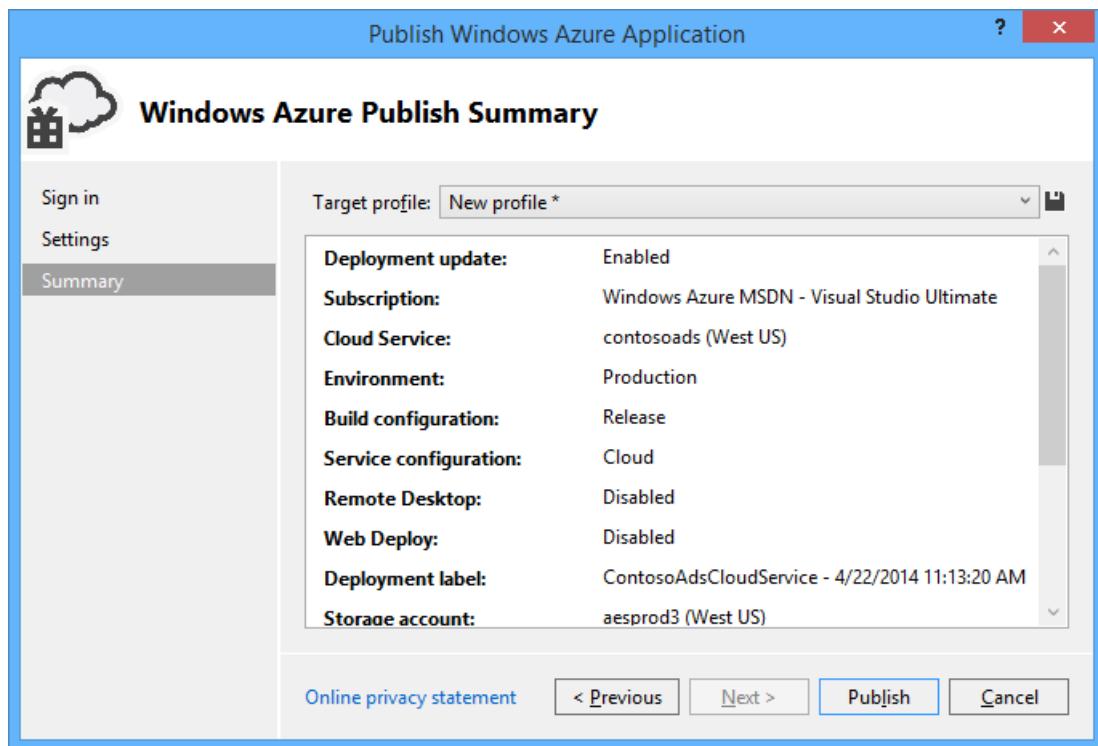


3. In the **Settings** step of the wizard, click **Next**.



The default settings in the **Advanced** tab are fine for this tutorial. For information about the advanced tab, see [Publish Azure Application Wizard](#).

4. In the **Summary** step, click **Publish**.



The **Azure Activity Log** window opens in Visual Studio.

5. Click the right arrow icon to expand the deployment details.

The deployment can take up to 5 minutes or more to complete.

The screenshot shows the 'Windows Azure Activity Log' window. It has tabs for Deployment, Storage, Log Requests, Virtual Machines, and Extensions. The Deployment tab is selected. A table lists one deployment entry: 'Deploying to contosoads - Production' with Status 'Completed' and Start Time '4/22/2014 6:32:00 PM'. Below the table, a 'History' pane shows log entries:
11:38:46 AM - Starting...
11:39:03 AM - Initializing...
11:39:03 AM - Created Website URL: http://contosoads.cloudapp.net/
11:39:03 AM - Complete.
A green play button icon is visible on the left.

6. When the deployment status is complete, click the **Web app URL** to start the application.
7. You can now test the app by creating, viewing, and editing some ads, as you did when you ran the application locally.

NOTE

When you're finished testing, delete or stop the cloud service. Even if you're not using the cloud service, it's accruing charges because virtual machine resources are reserved for it. And if you leave it running, anyone who finds your URL can create and view ads. In the [Azure portal](#), go to the **Overview** tab for your cloud service, and then click the **Delete** button at the top of the page. If you just want to temporarily prevent others from accessing the site, click **Stop** instead. In that case, charges will continue to accrue. You can follow a similar procedure to delete the SQL database and storage account when you no longer need them.

Create the application from scratch

If you haven't already downloaded [the completed application](#), do that now. You'll copy files from the downloaded

project into the new project.

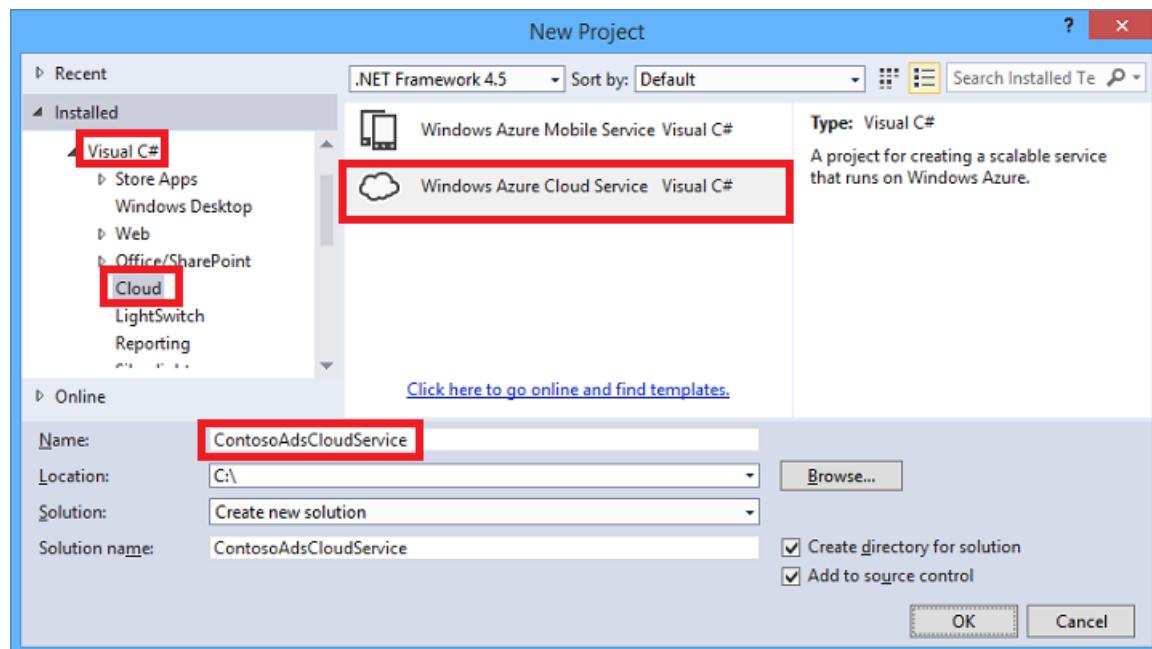
Creating the Contoso Ads application involves the following steps:

- Create a cloud service Visual Studio solution.
- Update and add NuGet packages.
- Set project references.
- Configure connection strings.
- Add code files.

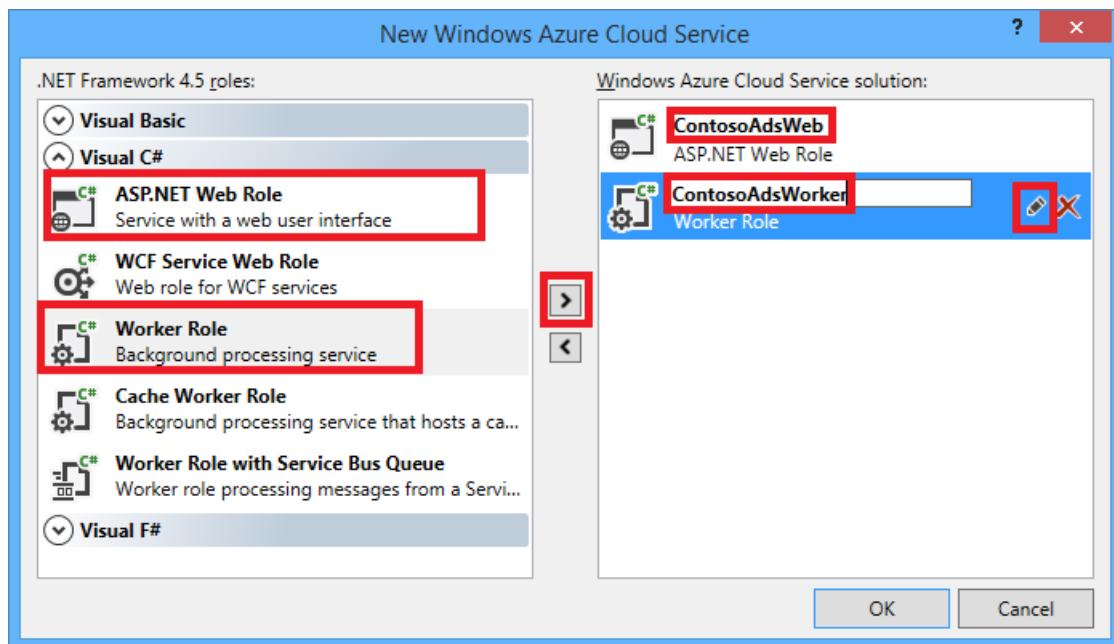
After the solution is created, you'll review the code that is unique to cloud service projects and Azure blobs and queues.

Create a cloud service Visual Studio solution

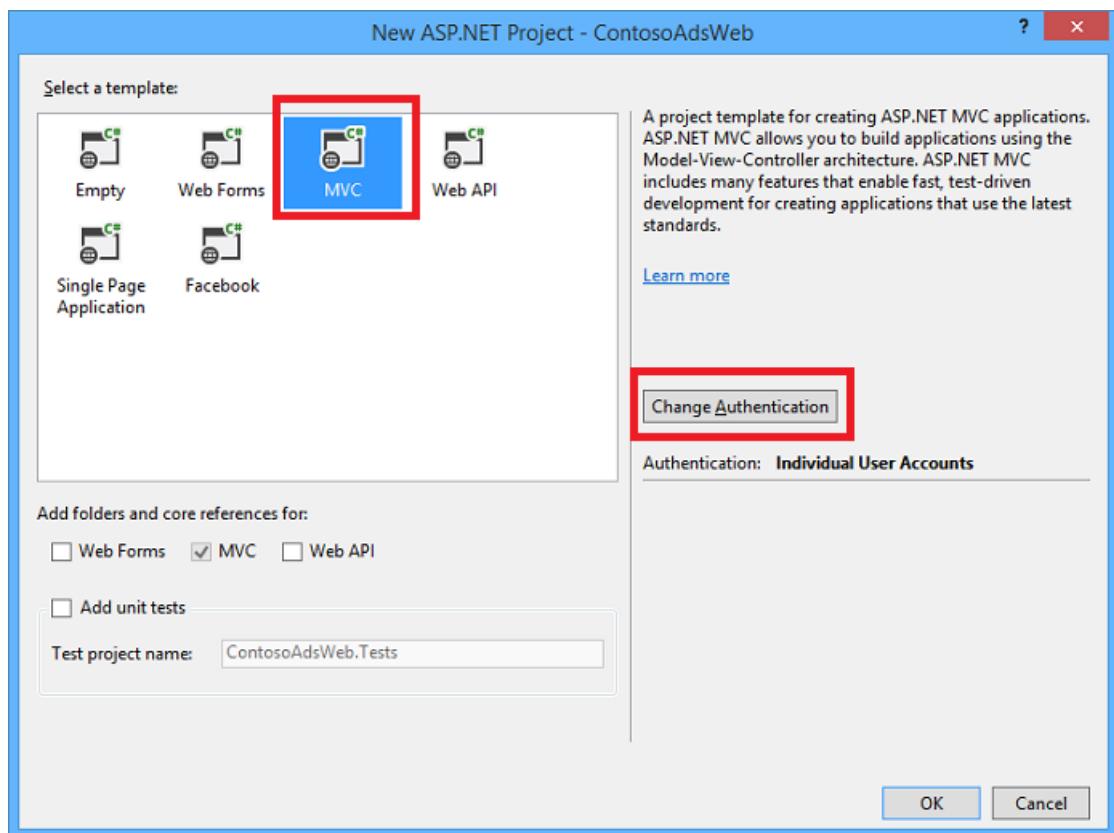
1. In Visual Studio, choose **New Project** from the **File** menu.
2. In the left pane of the **New Project** dialog box, expand **Visual C#** and choose **Cloud** templates, and then choose the **Azure Cloud Service** template.
3. Name the project and solution **ContosoAdsCloudService**, and then click **OK**.



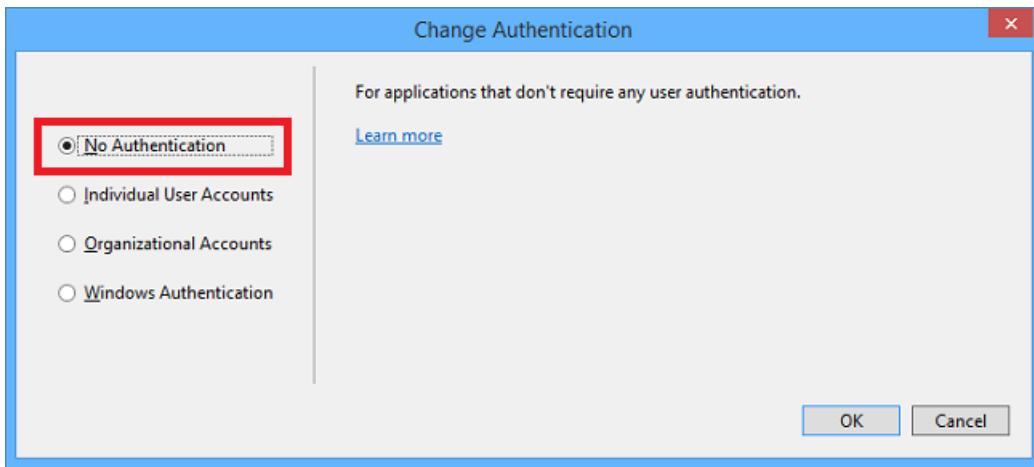
4. In the **New Azure Cloud Service** dialog box, add a web role and a worker role. Name the web role **ContosoAdsWeb**, and name the worker role **ContosoAdsWorker**. (Use the pencil icon in the right-hand pane to change the default names of the roles.)



5. When you see the **New ASP.NET Project** dialog box for the web role, choose the MVC template, and then click **Change Authentication**.



6. In the **Change Authentication** dialog box, choose **No Authentication**, and then click **OK**.



7. In the **New ASP.NET Project** dialog, click **OK**.
8. In **Solution Explorer**, right-click the solution (not one of the projects), and choose **Add - New Project**.
9. In the **Add New Project** dialog box, choose **Windows** under **Visual C#** in the left pane, and then click the **Class Library** template.
10. Name the project *ContosoAdsCommon*, and then click **OK**.

You need to reference the Entity Framework context and the data model from both web and worker role projects. As an alternative, you could define the EF-related classes in the web role project and reference that project from the worker role project. But in the alternative approach, your worker role project would have a reference to web assemblies that it doesn't need.

Update and add NuGet packages

1. Open the **Manage NuGet Packages** dialog box for the solution.
2. At the top of the window, select **Updates**.
3. Look for the *WindowsAzure.Storage* package, and if it's in the list, select it and select the web and worker projects to update it in, and then click **Update**.

The storage client library is updated more frequently than Visual Studio project templates, so you'll often find that the version in a newly-created project needs to be updated.

4. At the top of the window, select **Browse**.
5. Find the *EntityFramework* NuGet package, and install it in all three projects.
6. Find the *Microsoft.WindowsAzure.ConfigurationManager* NuGet package, and install it in the worker role project.

Set project references

1. In the ContosoAdsWeb project, set a reference to the ContosoAdsCommon project. Right-click the ContosoAdsWeb project, and then click **References - Add References**. In the **Reference Manager** dialog box, select **Solution - Projects** in the left pane, select **ContosoAdsCommon**, and then click **OK**.
2. In the ContosoAdsWorker project, set a reference to the ContosoAdsCommon project.

ContosoAdsCommon will contain the Entity Framework data model and context class, which will be used by both the front-end and back-end.

3. In the ContosoAdsWorker project, set a reference to `System.Drawing`.

This assembly is used by the back-end to convert images to thumbnails.

Configure connection strings

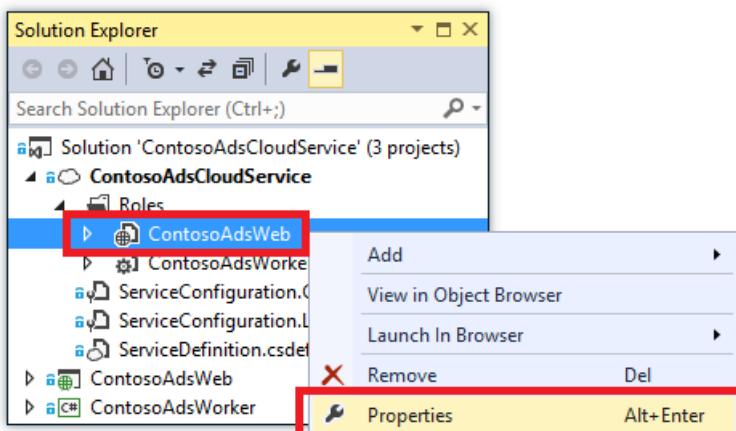
In this section, you configure Azure Storage and SQL connection strings for testing locally. The deployment instructions earlier in the tutorial explain how to set up the connection strings for when the app runs in the cloud.

1. In the ContosoAdsWeb project, open the application Web.config file, and insert the following `connectionStrings` element after the `configSections` element.

```
<connectionStrings>
    <add name="ContosoAdsContext" connectionString="Data Source=(localdb)\v11.0; Initial Catalog=ContosoAds; Integrated Security=True; MultipleActiveResultSets=True;" providerName="System.Data.SqlClient" />
</connectionStrings>
```

If you're using Visual Studio 2015 or higher, replace "v11.0" with "MSSQLLocalDB".

2. Save your changes.
3. In the ContosoAdsCloudService project, right-click ContosoAdsWeb under **Roles**, and then click **Properties**.



4. In the **ContosoAdsWeb [Role]** properties window, click the **Settings** tab, and then click **Add Setting**.
Leave **Service Configuration** set to **All Configurations**.
5. Add a setting named *StorageConnectionString*. Set **Type** to *ConnectionString*, and set **Value** to *UseDevelopmentStorage=true*.

A screenshot of the 'ContosoAdsWeb [Role]' properties window. The 'Settings' tab is selected. At the top, there is a dropdown labeled 'Service Configuration' with 'All Configurations' selected. Below the dropdown, there is a red box around the 'Add Setting' button. The main area shows a table of configuration settings:

Name	Type	Value
Microsoft.WindowsAzure.Pl...	Connection String	<Select Configuration>
StorageConnectionString	Connection String	UseDevelopmentStorage=true

6. Save your changes.
7. Follow the same procedure to add a storage connection string in the ContosoAdsWorker role properties.
8. Still in the **ContosoAdsWorker [Role]** properties window, add another connection string:

- Name: ContosoAdsDbConnectionString
- Type: String
- Value: Paste the same connection string you used for the web role project. (The following example is for Visual Studio 2013. Don't forget to change the Data Source if you copy this example and you're using Visual Studio 2015 or higher.)

```
Data Source=(localdb)\v11.0; Initial Catalog=ContosoAds; Integrated Security=True;  
MultipleActiveResultSets=True;
```

Add code files

In this section, you copy code files from the downloaded solution into the new solution. The following sections will show and explain key parts of this code.

To add files to a project or a folder, right-click the project or folder and click **Add - Existing Item**. Select the files you want and then click **Add**. If asked whether you want to replace existing files, click **Yes**.

1. In the ContosoAdsCommon project, delete the *Class1.cs* file and add in its place the *Ad.cs* and *ContosoAdscontext.cs* files from the downloaded project.
2. In the ContosoAdsWeb project, add the following files from the downloaded project.
 - *Global.asax.cs*.
 - In the *Views\Shared* folder: *_Layout.cshtml*.
 - In the *Views\Home* folder: *Index.cshtml*.
 - In the *Controllers* folder: *AdController.cs*.
 - In the *Views\Ad* folder (create the folder first): five *.cshtml* files.
3. In the ContosoAdsWorker project, add *WorkerRole.cs* from the downloaded project.

You can now build and run the application as instructed earlier in the tutorial, and the app will use local database and storage emulator resources.

The following sections explain the code related to working with the Azure environment, blobs, and queues. This tutorial does not explain how to create MVC controllers and views using scaffolding, how to write Entity Framework code that works with SQL Server databases, or the basics of asynchronous programming in ASP.NET 4.5. For information about these topics, see the following resources:

- [Get started with MVC 5](#)
- [Get started with EF 6 and MVC 5](#)
- [Introduction to asynchronous programming in .NET 4.5](#).

ContosoAdsCommon - Ad.cs

The *Ad.cs* file defines an enum for ad categories and a POCO entity class for ad information.

```

public enum Category
{
    Cars,
    [Display(Name="Real Estate")]
    RealEstate,
    [Display(Name = "Free Stuff")]
    FreeStuff
}

public class Ad
{
    public int AdId { get; set; }

    [StringLength(100)]
    public string Title { get; set; }

    public int Price { get; set; }

    [StringLength(1000)]
    [DataType(DataType.MultilineText)]
    public string Description { get; set; }

    [StringLength(1000)]
    [DisplayName("Full-size Image")]
    public string ImageURL { get; set; }

    [StringLength(1000)]
    [DisplayName("Thumbnail")]
    public string ThumbnailURL { get; set; }

    [DataType(DataType.Date)]
    [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
    public DateTime PostedDate { get; set; }

    public Category? Category { get; set; }
    [StringLength(12)]
    public string Phone { get; set; }
}

```

ContosoAdsCommon - ContosoAdsContext.cs

The ContosoAdsContext class specifies that the Ad class is used in a DbSet collection, which Entity Framework will store in a SQL database.

```

public class ContosoAdsContext : DbContext
{
    public ContosoAdsContext() : base("name=ContosoAdsContext")
    {
    }
    public ContosoAdsContext(string connString)
        : base(connString)
    {
    }
    public System.Data.Entity.DbSet<Ad> Ads { get; set; }
}

```

The class has two constructors. The first of them is used by the web project, and specifies the name of a connection string that is stored in the Web.config file. The second constructor enables you to pass in the actual connection string used by the worker role project, since it doesn't have a Web.config file. You saw earlier where this connection string was stored, and you'll see later how the code retrieves the connection string when it instantiates the DbContext class.

ContosoAdsWeb - Global.asax.cs

Code that is called from the `Application_Start` method creates an *images* blob container and an *images* queue if they don't already exist. This ensures that whenever you start using a new storage account, or start using the storage emulator on a new computer, the required blob container and queue will be created automatically.

The code gets access to the storage account by using the storage connection string from the *.cscfg* file.

```
var storageAccount = CloudStorageAccount.Parse  
    (RoleEnvironment.GetConfigurationSettingValue("StorageConnectionString"));
```

Then it gets a reference to the *images* blob container, creates the container if it doesn't already exist, and sets access permissions on the new container. By default, new containers only allow clients with storage account credentials to access blobs. The website needs the blobs to be public so that it can display images using URLs that point to the image blobs.

```
var blobClient = storageAccount.CreateCloudBlobClient();  
var imagesBlobContainer = blobClient.GetContainerReference("images");  
if (!imagesBlobContainer.CreateIfNotExists())  
{  
    imagesBlobContainer.SetPermissions(  
        new BlobContainerPermissions  
        {  
            PublicAccess = BlobContainerPublicAccessType.Blob  
        });  
}
```

Similar code gets a reference to the *images* queue and creates a new queue. In this case, no permissions change is needed.

```
CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();  
var imagesQueue = queueClient.GetQueueReference("images");  
imagesQueue.CreateIfNotExists();
```

ContosoAdsWeb - _Layout.cshtml

The *_Layout.cshtml* file sets the app name in the header and footer, and creates an "Ads" menu entry.

ContosoAdsWeb - Views\Home\Index.cshtml

The *Views\Home\Index.cshtml* file displays category links on the home page. The links pass the integer value of the `Category` enum in a querystring variable to the Ads Index page.

```
<li>@Html.ActionLink("Cars", "Index", "Ad", new { category = (int)Category.Cars }, null)</li>  
<li>@Html.ActionLink("Real estate", "Index", "Ad", new { category = (int)Category.RealEstate }, null)</li>  
<li>@Html.ActionLink("Free stuff", "Index", "Ad", new { category = (int)Category.FreeStuff }, null)</li>  
<li>@Html.ActionLink("All", "Index", "Ad", null, null)</li>
```

ContosoAdsWeb - AdController.cs

In the *AdController.cs* file, the constructor calls the `InitializeStorage` method to create Azure Storage Client Library objects that provide an API for working with blobs and queues.

Then the code gets a reference to the *images* blob container as you saw earlier in *Global.asax.cs*. While doing that it sets a default `retry policy` appropriate for a web app. The default exponential backoff retry policy could hang the web app for longer than a minute on repeated retries for a transient fault. The retry policy specified here waits three seconds after each try for up to three tries.

```
var blobClient = storageAccount.CreateCloudBlobClient();
blobClient.DefaultRequestOptions.RetryPolicy = new LinearRetry(TimeSpan.FromSeconds(3), 3);
imagesBlobContainer = blobClient.GetContainerReference("images");
```

Similar code gets a reference to the *images* queue.

```
CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();
queueClient.DefaultRequestOptions.RetryPolicy = new LinearRetry(TimeSpan.FromSeconds(3), 3);
imagesQueue = queueClient.GetQueueReference("images");
```

Most of the controller code is typical for working with an Entity Framework data model using a `DbContext` class. An exception is the `HttpPost` `Create` method, which uploads a file and saves it in blob storage. The model binder provides an `HttpPostedFileBase` object to the method.

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Create(
    [Bind(Include = "Title,Price,Description,Category,Phone")] Ad ad,
    HttpPostedFileBase imageFile)
```

If the user selected a file to upload, the code uploads the file, saves it in a blob, and updates the `Ad` database record with a URL that points to the blob.

```
if (imageFile != null && imageFile.ContentLength != 0)
{
    blob = await UploadAndSaveBlobAsync(imageFile);
    ad.ImageURL = blob.Uri.ToString();
}
```

The code that does the upload is in the `UploadAndSaveBlobAsync` method. It creates a GUID name for the blob, uploads and saves the file, and returns a reference to the saved blob.

```
private async Task<CloudBlockBlob> UploadAndSaveBlobAsync(HttpPostedFileBase imageFile)
{
    string blobName = Guid.NewGuid().ToString() + Path.GetExtension(imageFile.FileName);
    CloudBlockBlob imageBlob = imagesBlobContainer.GetBlockBlobReference(blobName);
    using (var fileStream = imageFile.InputStream)
    {
        await imageBlob.UploadFromStreamAsync(fileStream);
    }
    return imageBlob;
}
```

After the `HttpPost` `Create` method uploads a blob and updates the database, it creates a queue message to inform that back-end process that an image is ready for conversion to a thumbnail.

```
string queueMessageString = ad.AdId.ToString();
var queueMessage = new CloudQueueMessage(queueMessageString);
await queue.AddMessageAsync(queueMessage);
```

The code for the `HttpPost` `Edit` method is similar except that if the user selects a new image file any blobs that already exist must be deleted.

```

if (imageFile != null && imageFile.ContentLength != 0)
{
    await DeleteAdBlobsAsync(ad);
    imageBlob = await UploadAndSaveBlobAsync(imageFile);
    ad.ImageURL = imageBlob.Uri.ToString();
}

```

The next example shows the code that deletes blobs when you delete an ad.

```

private async Task DeleteAdBlobsAsync(Ad ad)
{
    if (!string.IsNullOrWhiteSpace(ad.ImageURL))
    {
        Uri blobUri = new Uri(ad.ImageURL);
        await DeleteAdBlobAsync(blobUri);
    }
    if (!string.IsNullOrWhiteSpace(ad.ThumbnailURL))
    {
        Uri blobUri = new Uri(ad.ThumbnailURL);
        await DeleteAdBlobAsync(blobUri);
    }
}
private static async Task DeleteAdBlobAsync(Uri blobUri)
{
    string blobName = blobUri.Segments[blobUri.Segments.Length - 1];
    CloudBlockBlob blobToDelete = imagesBlobContainer.GetBlockBlobReference(blobName);
    await blobToDelete.DeleteAsync();
}

```

ContosoAdsWeb - Views\Ad\Index.cshtml and Details.cshtml

The *Index.cshtml* file displays thumbnails with the other ad data.

```

```

The *Details.cshtml* file displays the full-size image.

```

```

ContosoAdsWeb - Views\Ad\Create.cshtml and Edit.cshtml

The *Create.cshtml* and *Edit.cshtml* files specify form encoding that enables the controller to get the `HttpPostedFileBase` object.

```
@using (Html.BeginForm("Create", "Ad", FormMethod.Post, new { enctype = "multipart/form-data" }))
```

An `<input>` element tells the browser to provide a file selection dialog.

```
<input type="file" name="imageFile" accept="image/*" class="form-control fileupload" />
```

ContosoAdsWorker - WorkerRole.cs - OnStart method

The Azure worker role environment calls the `OnStart` method in the `WorkerRole` class when the worker role is getting started, and it calls the `Run` method when the `OnStart` method finishes.

The `OnStart` method gets the database connection string from the `.cscfg` file and passes it to the Entity Framework `DbContext` class. The `SQLClient` provider is used by default, so the provider does not have to be specified.

```
var dbConnString = CloudConfigurationManager.GetSetting("ContosoAdsDbConnectionString");
db = new ContosoAdsContext(dbConnString);
```

After that, the method gets a reference to the storage account and creates the blob container and queue if they don't exist. The code for that is similar to what you already saw in the web role `Application_Start` method.

ContosoAdsWorker - WorkerRole.cs - Run method

The `Run` method is called when the `OnStart` method finishes its initialization work. The method executes an infinite loop that watches for new queue messages and processes them when they arrive.

```
public override void Run()
{
    CloudQueueMessage msg = null;

    while (true)
    {
        try
        {
            msg = this.imagesQueue.GetMessage();
            if (msg != null)
            {
                ProcessQueueMessage(msg);
            }
            else
            {
                System.Threading.Thread.Sleep(1000);
            }
        }
        catch (StorageException e)
        {
            if (msg != null && msg.DequeueCount > 5)
            {
                this.imagesQueue.DeleteMessage(msg);
            }
            System.Threading.Thread.Sleep(5000);
        }
    }
}
```

After each iteration of the loop, if no queue message was found, the program sleeps for a second. This prevents the worker role from incurring excessive CPU time and storage transaction costs. The Microsoft Customer Advisory Team tells a story about a developer who forgot to include this, deployed to production, and left for vacation. When he got back, his oversight cost more than the vacation.

Sometimes the content of a queue message causes an error in processing. This is called a *poison message*, and if you just logged an error and restarted the loop, you could endlessly try to process that message. Therefore the catch block includes an if statement that checks to see how many times the app has tried to process the current message, and if it has been more than 5 times, the message is deleted from the queue.

`ProcessQueueMessage` is called when a queue message is found.

```

private void ProcessQueueMessage(CloudQueueMessage msg)
{
    var adId = int.Parse(msg.AsString);
    Ad ad = db.Ads.Find(adId);
    if (ad == null)
    {
        throw new Exception(String.Format("AdId {0} not found, can't create thumbnail", adId.ToString()));
    }

    CloudBlockBlob inputBlob = this.imagesBlobContainer.GetBlockBlobReference(ad.ImageURL);

    string thumbnailName = Path.GetFileNameWithoutExtension(inputBlob.Name) + "thumb.jpg";
    CloudBlockBlob outputBlob = this.imagesBlobContainer.GetBlockBlobReference(thumbnailName);

    using (Stream input = inputBlob.OpenRead())
    using (Stream output = outputBlob.OpenWrite())
    {
        ConvertImageToThumbnailJPG(input, output);
        outputBlob.Properties.ContentType = "image/jpeg";
    }

    ad.ThumbnailURL = outputBlob.Uri.ToString();
    db.SaveChanges();

    this.imagesQueue.DeleteMessage(msg);
}

```

This code reads the database to get the image URL, converts the image to a thumbnail, saves the thumbnail in a blob, updates the database with the thumbnail blob URL, and deletes the queue message.

NOTE

The code in the `ConvertImageToThumbnailJPG` method uses classes in the `System.Drawing` namespace for simplicity. However, the classes in this namespace were designed for use with Windows Forms. They are not supported for use in a Windows or ASP.NET service. For more information about image-processing options, see [Dynamic Image Generation](#) and [Deep Inside Image Resizing](#).

Troubleshooting

In case something doesn't work while you're following the instructions in this tutorial, here are some common errors and how to resolve them.

ServiceRuntime.RoleEnvironmentException

The `RoleEnvironment` object is provided by Azure when you run an application in Azure or when you run locally using the Azure compute emulator. If you get this error when you're running locally, make sure that you have set the `ContosoAdsCloudService` project as the startup project. This sets up the project to run using the Azure compute emulator.

One of the things the application uses the Azure RoleEnvironment for is to get the connection string values that are stored in the `.cscfg` files, so another cause of this exception is a missing connection string. Make sure that you created the `StorageConnectionString` setting for both Cloud and Local configurations in the `ContosoAdsWeb` project, and that you created both connection strings for both configurations in the `ContosoAdsWorker` project. If you do a **Find All** search for `StorageConnectionString` in the entire solution, you should see it 9 times in 6 files.

Cannot override to port xxx. New port below minimum allowed value 8080 for protocol http

Try changing the port number used by the web project. Right-click the `ContosoAdsWeb` project, and then click **Properties**. Click the **Web** tab, and then change the port number in the **Project Url** setting.

For another alternative that might resolve the problem, see the following section.

Other errors when running locally

By default new cloud service projects use the Azure compute emulator express to simulate the Azure environment. This is a lightweight version of the full compute emulator, and under some conditions the full emulator will work when the express version does not.

To change the project to use the full emulator, right-click the ContosoAdsCloudService project, and then click **Properties**. In the **Properties** window click the **Web** tab, and then click the **Use Full Emulator** radio button.

In order to run the application with the full emulator, you have to open Visual Studio with administrator privileges.

Next steps

The Contoso Ads application has intentionally been kept simple for a getting-started tutorial. For example, it doesn't implement [dependency injection](#) or the [repository and unit of work patterns](#), it doesn't [use an interface for logging](#), it doesn't use [EF Code First Migrations](#) to manage data model changes or [EF Connection Resiliency](#) to manage transient network errors, and so forth.

Here are some cloud service sample applications that demonstrate more real-world coding practices, listed from less complex to more complex:

- [PhluffyFotos](#). Similar in concept to Contoso Ads but implements more features and more real-world coding practices.
- [Azure Cloud Service Multi-Tier Application with Tables, Queues, and Blobs](#). Introduces Azure Storage tables as well as blobs and queues. Based on an older version of the Azure SDK for .NET, will require some modifications to work with the current version.

For general information about developing for the cloud, see [Building Real-World Cloud Apps with Azure](#).

For a video introduction to Azure Storage best practices and patterns, see [Microsoft Azure Storage – What's New, Best Practices and Patterns](#).

For more information, see the following resources:

- [Azure Cloud Services Part 1: Introduction](#)
- [How to manage Cloud Services](#)
- [Azure Storage](#)
- [How to choose a cloud service provider](#)

Python web and worker roles with Python Tools for Visual Studio

12/18/2018 • 7 minutes to read • [Edit Online](#)

This article provides an overview of using Python web and worker roles using [Python Tools for Visual Studio](#). Learn how to use Visual Studio to create and deploy a basic Cloud Service that uses Python.

Prerequisites

- [Visual Studio 2013, 2015, or 2017](#)
- [Python Tools for Visual Studio \(PTVS\)](#)
- [Azure SDK Tools for VS 2013](#) or
[Azure SDK Tools for VS 2015](#) or
[Azure SDK Tools for VS 2017](#)
- [Python 2.7 32-bit](#) or [Python 3.5 32-bit](#)

NOTE

To complete this tutorial, you need an Azure account. You can [activate your Visual Studio subscriber benefits](#) or [sign up for a free trial](#).

What are Python web and worker roles?

Azure provides three compute models for running applications: [Web Apps feature in Azure App Service](#), [Azure Virtual Machines](#), and [Azure Cloud Services](#). All three models support Python. Cloud Services, which include web and worker roles, provide *Platform as a Service (PaaS)*. Within a cloud service, a web role provides a dedicated Internet Information Services (IIS) web server to host front end web applications, while a worker role can run asynchronous, long-running, or perpetual tasks independent of user interaction or input.

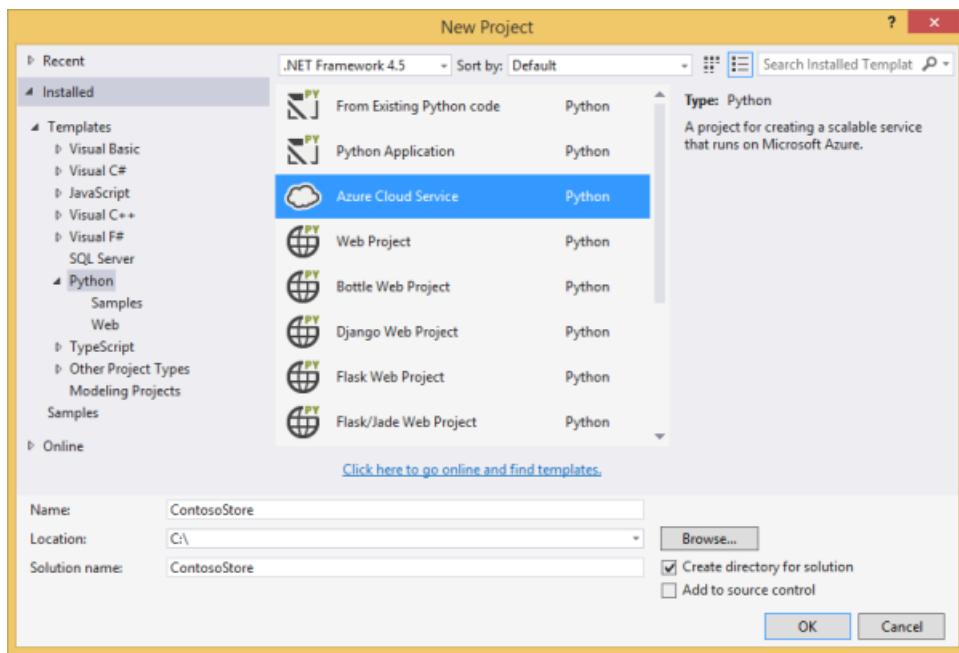
For more information, see [What is a Cloud Service?](#).

NOTE

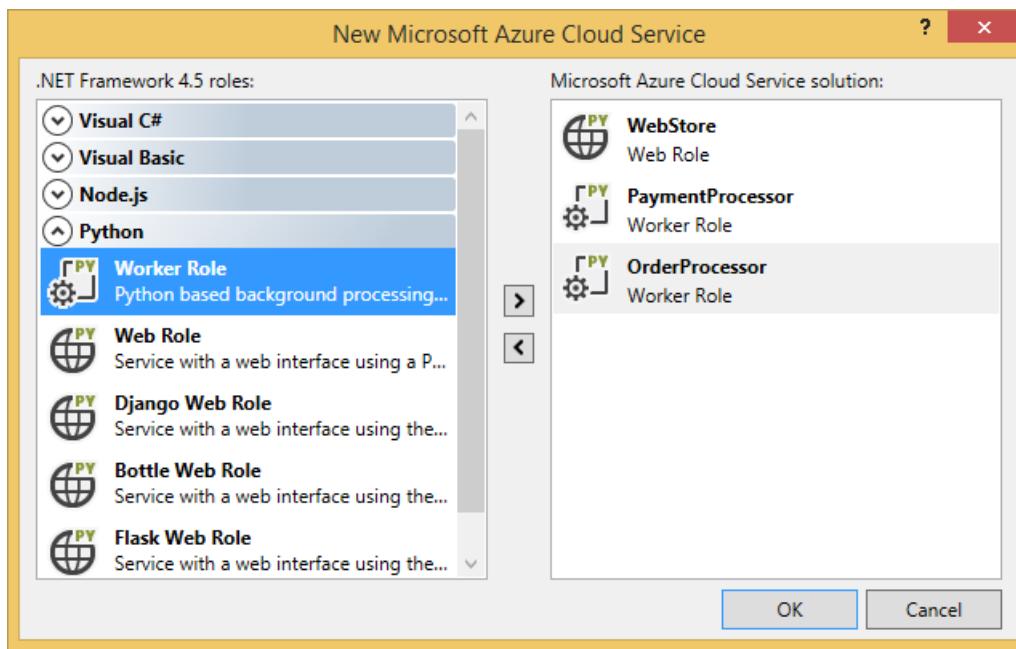
Looking to build a simple website? If your scenario involves just a simple website front end, consider using the lightweight Web Apps feature in Azure App Service. You can easily upgrade to a Cloud Service as your website grows and your requirements change. See the [Python Developer Center](#) for articles that cover development of the Web Apps feature in Azure App Service.

Project creation

In Visual Studio, you can select **Azure Cloud Service** in the **New Project** dialog box, under **Python**.



In the Azure Cloud Service wizard, you can create new web and worker roles.



The worker role template comes with boilerplate code to connect to an Azure storage account or Azure Service Bus.

The screenshot shows the Microsoft Visual Studio interface. The title bar says "ContosoStore - Microsoft Visual Studio". The menu bar includes FILE, EDIT, VIEW, PROJECT, BUILD, DEBUG, TEAM, TOOLS, TEST, ARCHITECTURE, ANALYZE, WINDOW, HELP, and Sign in. The toolbar has icons for file operations like Open, Save, and Print. The status bar at the bottom says "Ready".

The Solution Explorer window on the right shows a solution named "ContosoStore" containing four projects: Roles, OrderProcessor, PaymentProcessor, and WebStore. The "OrderProcessor" project is expanded, showing Python Environments (env (Python 3.4)), References, Search Paths, bin, requirements.txt, and worker.py.

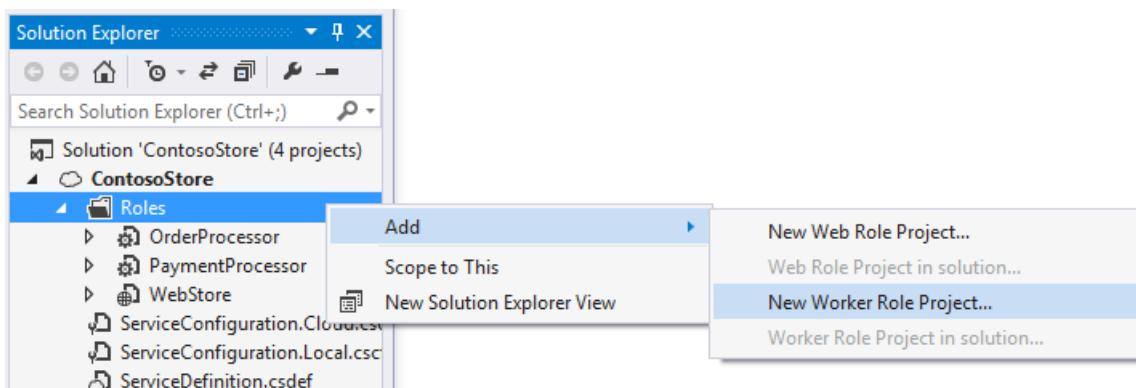
The code editor window on the left contains the "worker.py" file:

```

1 import os
2 from time import sleep
3
4 #
5 # The azure library provides access to services made available by the
6 # Microsoft Azure platform, such as storage and messaging.
7 #
8 # See http://go.microsoft.com/fwlink/?LinkId=254360 for documentation and
9 # example code.
10 #
11 from azure.servicebus import ServiceBusService
12 from azure.storage import CloudStorageAccount
13
14 #
15 # The CloudStorageAccount provides factory methods for the queue, table, a
16 # blob services.
17 #
18 # See http://go.microsoft.com/fwlink/?LinkId=246933 for Storage documentat
19 #
20 STORAGE_ACCOUNT_NAME = '__paste_your_storage_account_name_here__'
21 STORAGE_ACCOUNT_KEY = '__paste_your_storage_key_here__'
22
23 if os.environ.get('EMULATED', '').lower() == 'true':
24     # Running in the emulator, so use the development storage account
25     storage_account = CloudStorageAccount(None, None)
26 else:
27     storage_account = CloudStorageAccount(STORAGE_ACCOUNT_NAME, STORAGE_AC

```

You can add web or worker roles to an existing cloud service at any time. You can choose to add existing projects in your solution, or create new ones.



Your cloud service can contain roles implemented in different languages. For example, you can have a Python web role implemented using Django, with Python, or with C# worker roles. You can easily communicate between your roles using Service Bus queues or storage queues.

Install Python on the cloud service

WARNING

The setup scripts that are installed with Visual Studio (at the time this article was last updated) do not work. This section describes a workaround.

The main problem with the setup scripts is that they do not install python. First, define two [startup tasks](#) in the `ServiceDefinition.csdef` file. The first task (**PrepPython.ps1**) downloads and installs the Python runtime. The second task (**PipInstaller.ps1**) runs pip to install any dependencies you may have.

The following scripts were written targeting Python 3.5. If you want to use the version 2.x of python, set the **PYTHON2** variable file to **on** for the two startup tasks and the runtime task:

```
<Variable name="PYTHON2" value="on" /> .
```

```

<Startup>

  <Task executionContext="elevated" taskType="simple" commandLine="bin\ps.cmd PrepPython.ps1">
    <Environment>
      <Variable name="EMULATED">
        <RoleInstanceValue xpath="/RoleEnvironment/Deployment/@emulated" />
      </Variable>
      <Variable name="PYTHON2" value="off" />
    </Environment>
  </Task>

  <Task executionContext="elevated" taskType="simple" commandLine="bin\ps.cmd PipInstaller.ps1">
    <Environment>
      <Variable name="EMULATED">
        <RoleInstanceValue xpath="/RoleEnvironment/Deployment/@emulated" />
      </Variable>
      <Variable name="PYTHON2" value="off" />
    </Environment>
  </Task>
</Startup>

```

The **PYTHON2** and **PYPATH** variables must be added to the worker startup task. The **PYPATH** variable is only used if the **PYTHON2** variable is set to **on**.

```

<Runtime>
  <Environment>
    <Variable name="EMULATED">
      <RoleInstanceValue xpath="/RoleEnvironment/Deployment/@emulated" />
    </Variable>
    <Variable name="PYTHON2" value="off" />
    <Variable name="PYPATH" value="%SystemDrive%\Python27" />
  </Environment>
  <EntryPoint>
    <ProgramEntryPoint commandLine="bin\ps.cmd LaunchWorker.ps1" setReadyOnProcessStart="true" />
  </EntryPoint>
</Runtime>

```

Sample ServiceDefinition.csdef

```

<?xml version="1.0" encoding="utf-8"?>
<ServiceDefinition name="AzureCloudServicePython"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition" schemaVersion="2015-04.2.6">
  <WorkerRole name="WorkerRole1" vmsize="Small">
    <ConfigurationSettings>
      <Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString" />
      <Setting name="Python2" />
    </ConfigurationSettings>
    <Startup>
      <Task executionContext="elevated" taskType="simple" commandLine="bin\ps.cmd PrepPython.ps1">
        <Environment>
          <Variable name="EMULATED">
            <RoleInstanceValue xpath="/RoleEnvironment/Deployment/@emulated" />
          </Variable>
          <Variable name="PYTHON2" value="off" />
        </Environment>
      </Task>
      <Task executionContext="elevated" taskType="simple" commandLine="bin\ps.cmd PipInstaller.ps1">
        <Environment>
          <Variable name="EMULATED">
            <RoleInstanceValue xpath="/RoleEnvironment/Deployment/@emulated" />
          </Variable>
          <Variable name="PYTHON2" value="off" />
        </Environment>
      </Task>
    </Startup>
    <Runtime>
      <Environment>
        <Variable name="EMULATED">
          <RoleInstanceValue xpath="/RoleEnvironment/Deployment/@emulated" />
        </Variable>
        <Variable name="PYTHON2" value="off" />
        <Variable name="PYPATH" value="%SystemDrive%\Python27" />
      </Environment>
      <EntryPoint>
        <ProgramEntryPoint commandLine="bin\ps.cmd LaunchWorker.ps1" setReadyOnProcessStart="true" />
      </EntryPoint>
    </Runtime>
    <Imports>
      <Import moduleName="RemoteAccess" />
      <Import moduleName="RemoteForwarder" />
    </Imports>
  </WorkerRole>
</ServiceDefinition>

```

Next, create the **PrepPython.ps1** and **PipInstaller.ps1** files in the **./bin** folder of your role.

PrepPython.ps1

This script installs python. If the **PYTHON2** environment variable is set to **on**, then Python 2.7 is installed, otherwise Python 3.5 is installed.

```

[Net.ServicePointManager]::SecurityProtocol = "tls12, tls11, tls"
$is_emulated = $env:EMULATED -eq "true"
$is_python2 = $env:PYTHON2 -eq "on"
$nl = [Environment]::NewLine

if (-not $is_emulated){
    Write-Output "Checking if python is installed...$nl"
    if ($is_python2) {
        & "${env:SystemDrive}\Python27\python.exe" -V | Out-Null
    }
    else {
        py -V | Out-Null
    }

    if (-not $?) {

        $url = "https://www.python.org/ftp/python/3.5.2/python-3.5.2-amd64.exe"
        $outFile = "${env:TEMP}\python-3.5.2-amd64.exe"

        if ($is_python2) {
            $url = "https://www.python.org/ftp/python/2.7.12/python-2.7.12.amd64.msi"
            $outFile = "${env:TEMP}\python-2.7.12.amd64.msi"
        }

        Write-Output "Not found, downloading $url to $outFile$nl"
        Invoke-WebRequest $url -OutFile $outFile
        Write-Output "Installing$nl"

        if ($is_python2) {
            Start-Process msieexec.exe -ArgumentList "/q", "/i", "$outFile", "ALLUSERS=1" -Wait
        }
        else {
            Start-Process "$outFile" -ArgumentList "/quiet", "InstallAllUsers=1" -Wait
        }

        Write-Output "Done$nl"
    }
    else {
        Write-Output "Already installed"
    }
}

```

PipInstaller.ps1

This script calls up pip and installs all of the dependencies in the **requirements.txt** file. If the **PYTHON2** environment variable is set to **on**, then Python 2.7 is used, otherwise Python 3.5 is used.

```

$is_emulated = $env:EMULATED -eq "true"
$is_python2 = $env:PYTHON2 -eq "on"
$nl = [Environment]::NewLine

if (-not $is_emulated){
    Write-Output "Checking if requirements.txt exists$nl"
    if (Test-Path ..\requirements.txt) {
        Write-Output "Found. Processing pip$nl"

        if ($is_python2) {
            & "${env:SystemDrive}\Python27\python.exe" -m pip install -r ..\requirements.txt
        }
        else {
            py -m pip install -r ..\requirements.txt
        }

        Write-Output "Done$nl"
    }
    else {
        Write-Output "Not found$nl"
    }
}

```

Modify LaunchWorker.ps1

NOTE

In the case of a **worker role** project, **LauncherWorker.ps1** file is required to execute the startup file. In a **web role** project, the startup file is instead defined in the project properties.

The **bin\LaunchWorker.ps1** was originally created to do a lot of prep work but it doesn't really work. Replace the contents in that file with the following script.

This script calls the **worker.py** file from your python project. If the **PYTHON2** environment variable is set to **on**, then Python 2.7 is used, otherwise Python 3.5 is used.

```

$is_emulated = $env:EMULATED -eq "true"
$is_python2 = $env:PYTHON2 -eq "on"
$nl = [Environment]::NewLine

if (-not $is_emulated)
{
    Write-Output "Running worker.py$nl"

    if ($is_python2) {
        cd..
        iex "$env:PYPATH\python.exe worker.py"
    }
    else {
        cd..
        iex "py worker.py"
    }
}
else
{
    Write-Output "Running (EMULATED) worker.py$nl"

    # Customize to your local dev environment

    if ($is_python2) {
        cd..
        iex "$env:PYPATH\python.exe worker.py"
    }
    else {
        cd..
        iex "py worker.py"
    }
}

```

ps.cmd

The Visual Studio templates should have created a **ps.cmd** file in the **./bin** folder. This shell script calls out the PowerShell wrapper scripts above and provides logging based on the name of the PowerShell wrapper called. If this file wasn't created, here is what should be in it.

```

@echo off

cd /D %~dp0

if not exist "%DiagnosticStore%\LogFiles" mkdir "%DiagnosticStore%\LogFiles"
%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe -ExecutionPolicy Unrestricted -File %* >>
"%DiagnosticStore%\LogFiles\%~n1.txt" 2>> "%DiagnosticStore%\LogFiles\%~n1.err.txt"

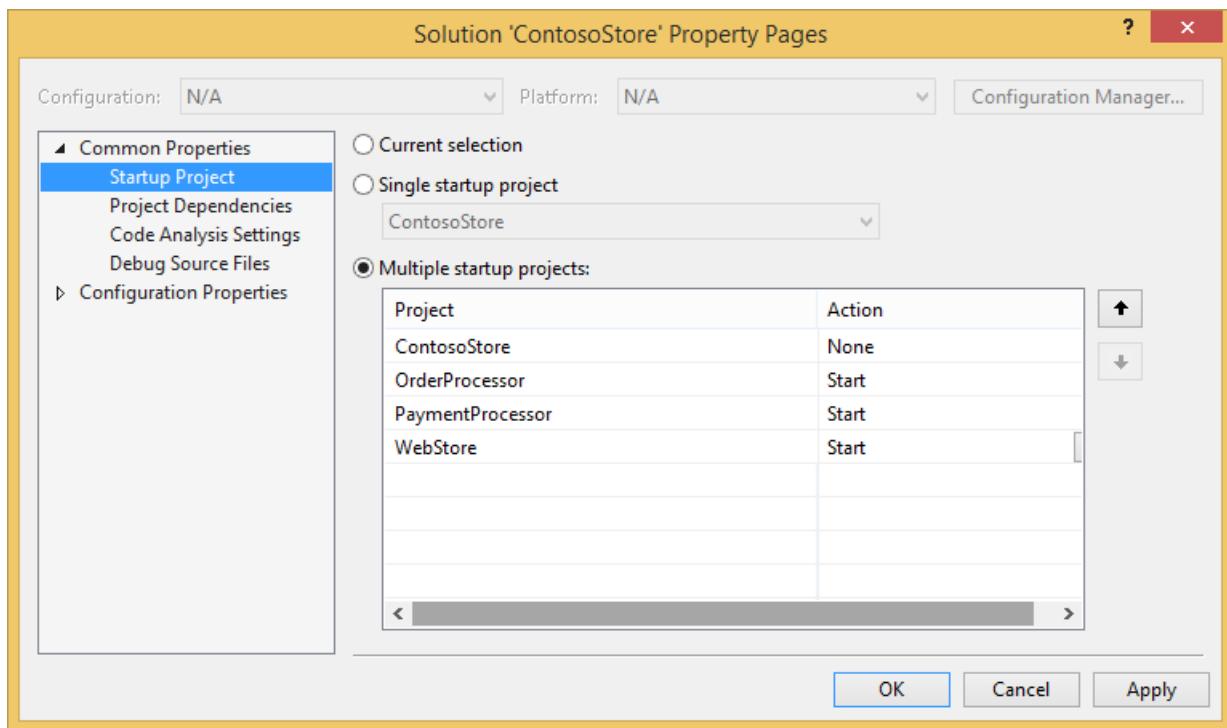
```

Run locally

If you set your cloud service project as the startup project and press F5, the cloud service runs in the local Azure emulator.

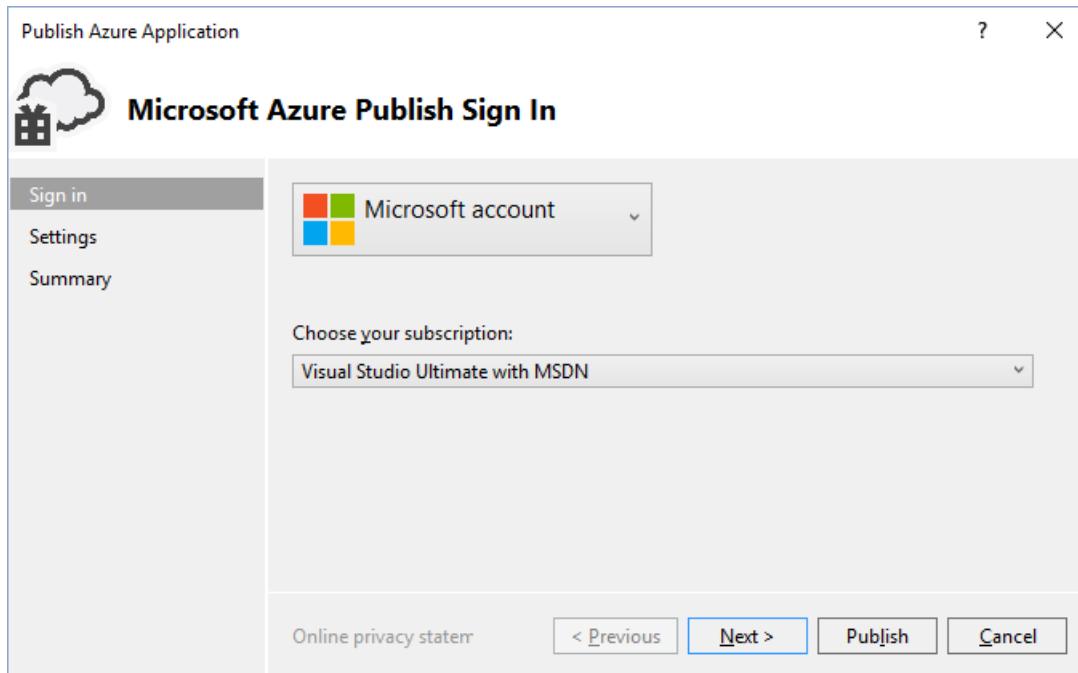
Although PTVS supports launching in the emulator, debugging (for example, breakpoints) does not work.

To debug your web and worker roles, you can set the role project as the startup project and debug that instead. You can also set multiple startup projects. Right-click the solution and then select **Set StartUp Projects**.



Publish to Azure

To publish, right-click the cloud service project in the solution and then select **Publish**.



Follow the wizard. If you need to, enable remote desktop. Remote desktop is helpful when you need to debug something.

When you are done configuring settings, click **Publish**.

Some progress appears in the output window, then you'll see the Microsoft Azure Activity Log window.

The screenshot shows the Microsoft Azure Activity Log interface. At the top, there are tabs for Deployment, Storage, Log Requests, Virtual Machines, and Extensions. Below the tabs is a table with columns for Description, Status, and Start Time (UTC). A single row is selected, showing a warning icon, the text "Deploying to contosostore2 - Production", a green status bar, and the start time "10/10/2014 5:12:40 PM". To the left of the table, there are sections for "Production" status (Validation warnings, Website URL, Deployment ID, Open in Server Explorer), each with a yellow warning icon. To the right of the table is a "History" pane displaying deployment logs:

```
10:12:40 AM - Preparing deployment for Contosostore - 10/10/2014 10:12:06 AM with Subscription ID [REDACTED] using Service Management URL 'https://management.core.windows.net/'... 10:12:41 AM - Connecting... 10:12:41 AM - Verifying storage account '[REDACTED]'... 10:12:41 AM - Uploading Package...
```

Deployment takes several minutes to complete, then your web and/or worker roles run on Azure!

Investigate logs

After the cloud service virtual machine starts up and installs Python, you can look at the logs to find any failure messages. These logs are located in the **C:\Resources\Directory\{role}\LogFiles** folder. **PrepPython.err.txt** has at least one error in it from when the script tries to detect if Python is installed and **PipInstaller.err.txt** may complain about an outdated version of pip.

Next steps

For more detailed information about working with web and worker roles in Python Tools for Visual Studio, see the PTVS documentation:

- [Cloud Service Projects](#)

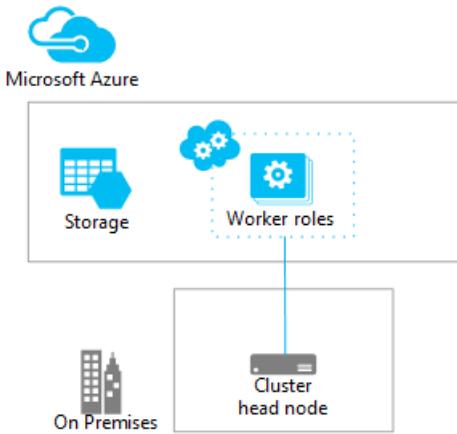
For more details about using Azure services from your web and worker roles, such as using Azure Storage or Service Bus, see the following articles:

- [Blob Service](#)
- [Table Service](#)
- [Queue Service](#)
- [Service Bus Queues](#)
- [Service Bus Topics](#)

Set up a hybrid high performance computing (HPC) cluster with Microsoft HPC Pack and on-demand Azure compute nodes

11/7/2018 • 9 minutes to read • [Edit Online](#)

Use Microsoft HPC Pack 2012 R2 and Azure to set up a small, hybrid high performance computing (HPC) cluster. The cluster shown in this article consists of an on-premises HPC Pack head node and some compute nodes you deploy on-demand in an Azure cloud service. You can then run compute jobs on the hybrid cluster.



This tutorial shows one approach, sometimes called cluster "burst to the cloud," to use scalable, on-demand Azure resources to run compute-intensive applications.

This tutorial assumes no prior experience with compute clusters or HPC Pack. It is intended only to help you deploy a hybrid compute cluster quickly for demonstration purposes. For considerations and steps to deploy a hybrid HPC Pack cluster at greater scale in a production environment, or to use HPC Pack 2016, see the [detailed guidance](#). For other scenarios with HPC Pack, including automated cluster deployment in Azure virtual machines, see [HPC cluster options with Microsoft HPC Pack in Azure](#).

Prerequisites

- **Azure subscription** - If you don't have an Azure subscription, you can create a [free account](#) in just a couple of minutes.
- **An on-premises computer running Windows Server 2012 R2 or Windows Server 2012** - Use this computer as the head node of the HPC cluster. If you aren't already running Windows Server, you can download and install an [evaluation version](#).
 - The computer must be joined to an Active Directory domain. For test purposes, you can configure the head node computer as a domain controller. To add the Active Directory Domain Services server role and promote the head node computer as a domain controller, see the documentation for Windows Server.
 - To support HPC Pack, the operating system must be installed in one of these languages: English, Japanese, or Chinese (Simplified).
 - Verify that important and critical updates are installed.
- **HPC Pack 2012 R2** - [Download](#) the installation package for the latest version free of charge and copy the files to the head node computer. Choose installation files in the same language as your installation of Windows Server.

NOTE

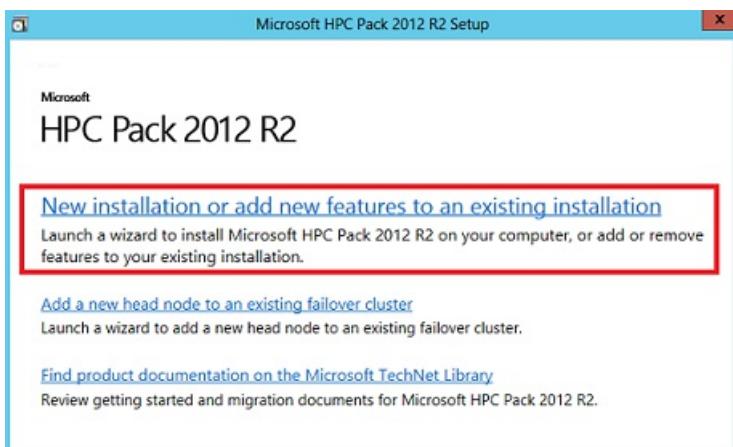
If you want to use HPC Pack 2016 instead of HPC Pack 2012 R2, additional configuration is needed. See the [detailed guidance](#).

- **Domain account** - This account must be configured with local Administrator permissions on the head node to install HPC Pack.
- **TCP connectivity on port 443** from the head node to Azure.

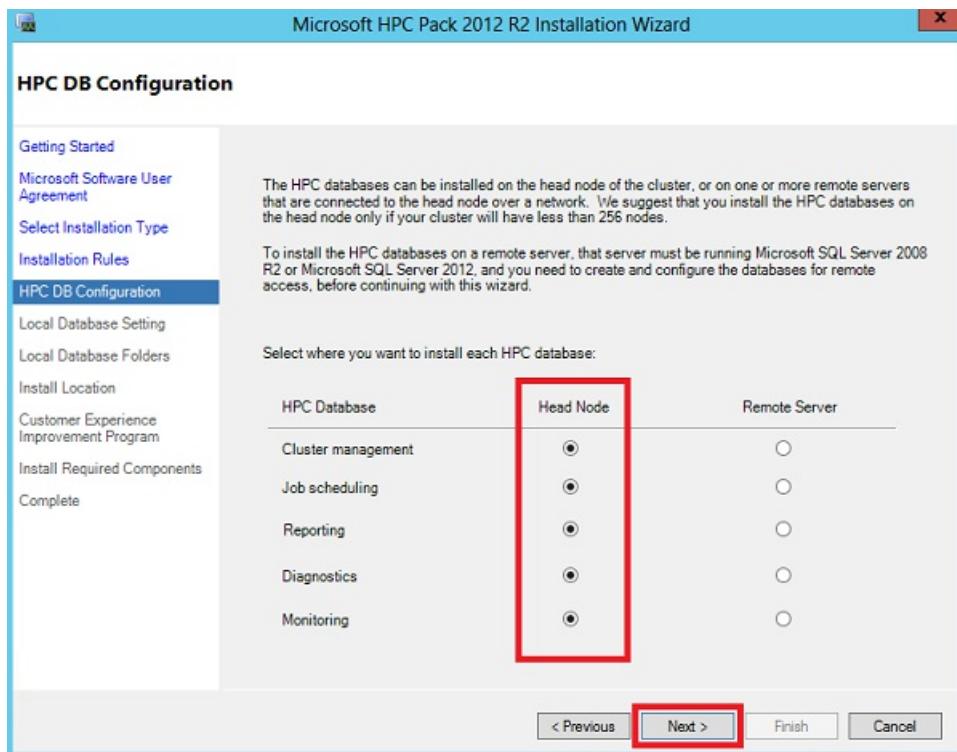
Install HPC Pack on the head node

You first install Microsoft HPC Pack on your on-premises computer running Windows Server. This computer becomes the head node of the cluster.

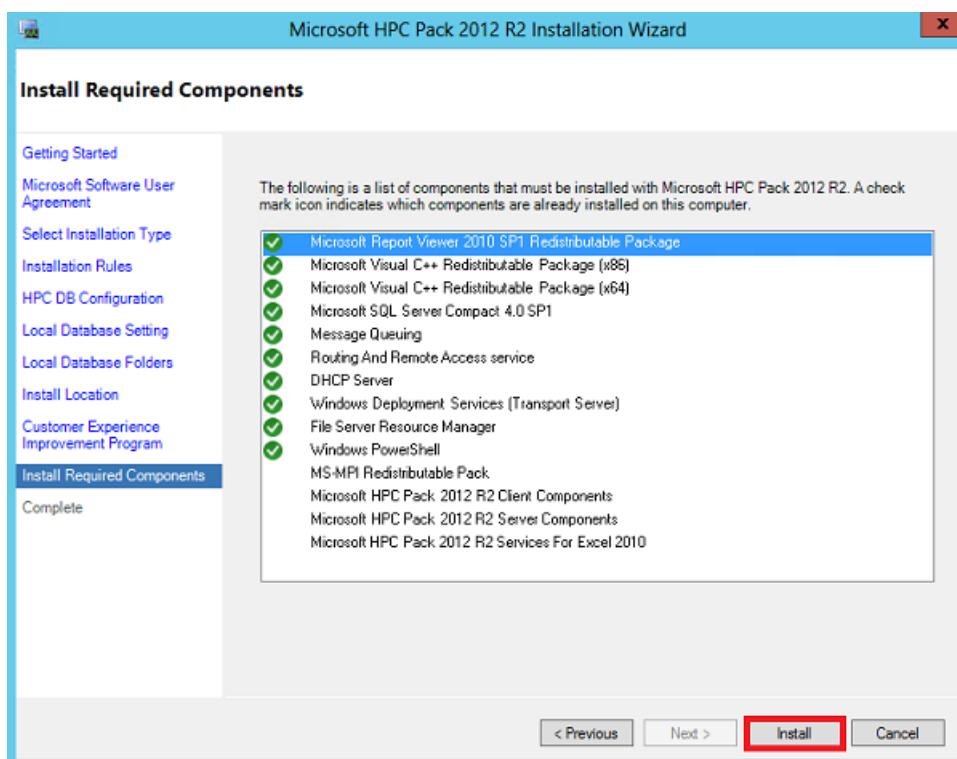
1. Log on to the head node by using a domain account that has local Administrator permissions.
2. Start the HPC Pack Installation Wizard by running Setup.exe from the HPC Pack installation files.
3. On the **HPC Pack 2012 R2 Setup** screen, click **New installation or add new features to an existing installation**.



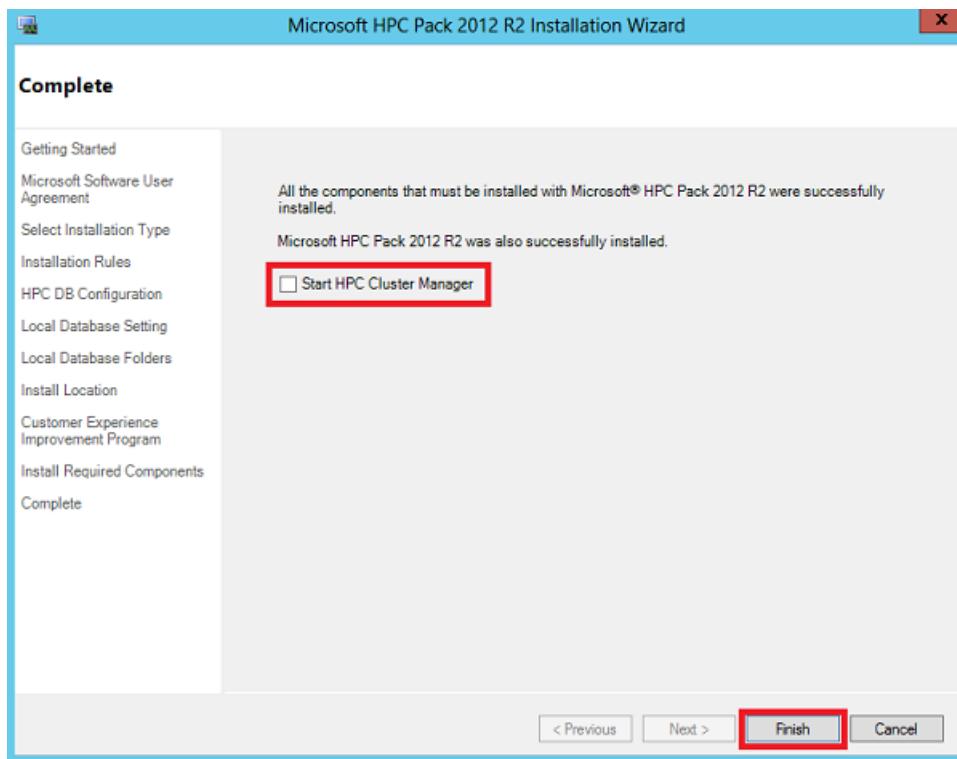
4. On the **Microsoft Software User Agreement** page, click **Next**.
5. On the **Select Installation Type** page, click **Create a new HPC cluster by creating a head node**, and then click **Next**.
6. The wizard runs several pre-installation tests. Click **Next** on the **Installation Rules** page if all tests pass. Otherwise, review the information provided and make any necessary changes in your environment. Then run the tests again or if necessary start the Installation Wizard again.
7. On the **HPC DB Configuration** page, make sure **Head Node** is selected for all HPC databases, and then click **Next**.



8. Accept default selections on the remaining pages of the wizard. On the **Install Required Components** page, click **Install**.



9. After the installation completes, uncheck **Start HPC Cluster Manager** and then click **Finish**. (You start HPC Cluster Manager in a later step.)



Prepare the Azure subscription

Perform the following steps in the [Azure portal](#) with your Azure subscription. After completing these steps, you can deploy Azure nodes from the on-premises head node.

NOTE

Also make a note of your Azure subscription ID, which you need later. Find the ID in **Subscriptions** in the portal.

Upload the default management certificate

HPC Pack installs a self-signed certificate on the head node, called the Default Microsoft HPC Azure Management certificate, that you can upload as an Azure management certificate. This certificate is provided for testing and proof-of-concept deployments to secure the connection between the head node and Azure.

1. From the head node computer, sign in to the [Azure portal](#).
2. Click **Subscriptions** > *your_subscription_name*.
3. Click **Management certificates** > **Upload**.
4. Browse on the head node for the file C:\Program Files\Microsoft HPC Pack 2012\Bin\hpccert.cer. Then, click **Upload**.

The **Default HPC Azure Management** certificate appears in the list of management certificates.

Create an Azure cloud service

NOTE

For best performance, create the cloud service and the storage account (in a later step) in the same geographic region.

1. In the portal, click **Cloud services (classic)** > **+Add**.
2. Type a DNS name for the service, choose a resource group and a location, and then click **Create**.

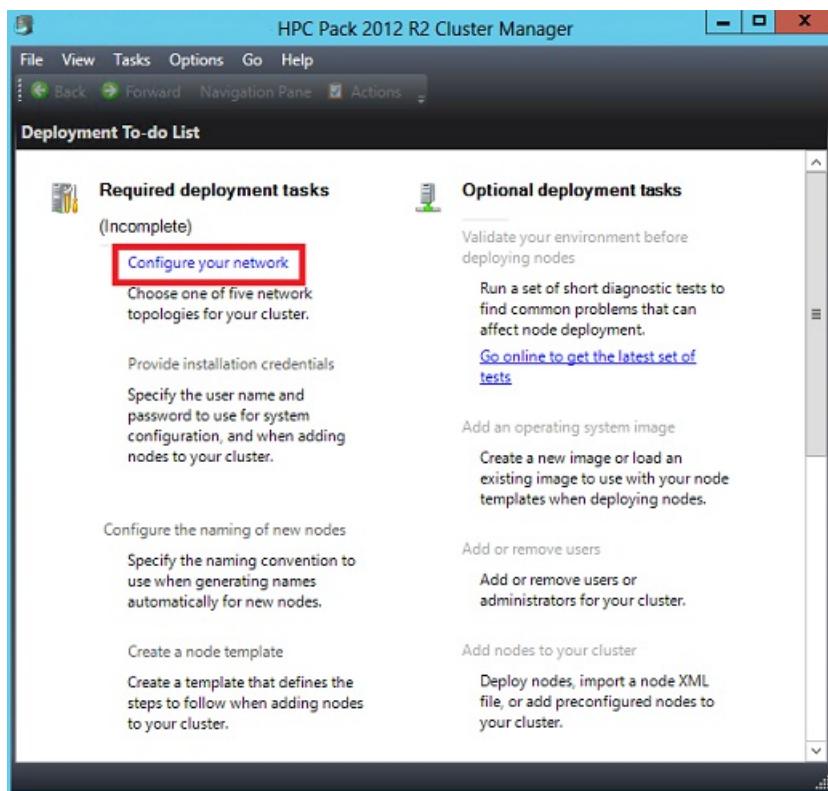
Create an Azure storage account

1. In the portal, click **Storage accounts (classic)** > **+Add**.
2. Type a name for the account, and select the **Classic** deployment model.
3. Choose a resource group and a location, and leave other settings at default values. Then click **Create**.

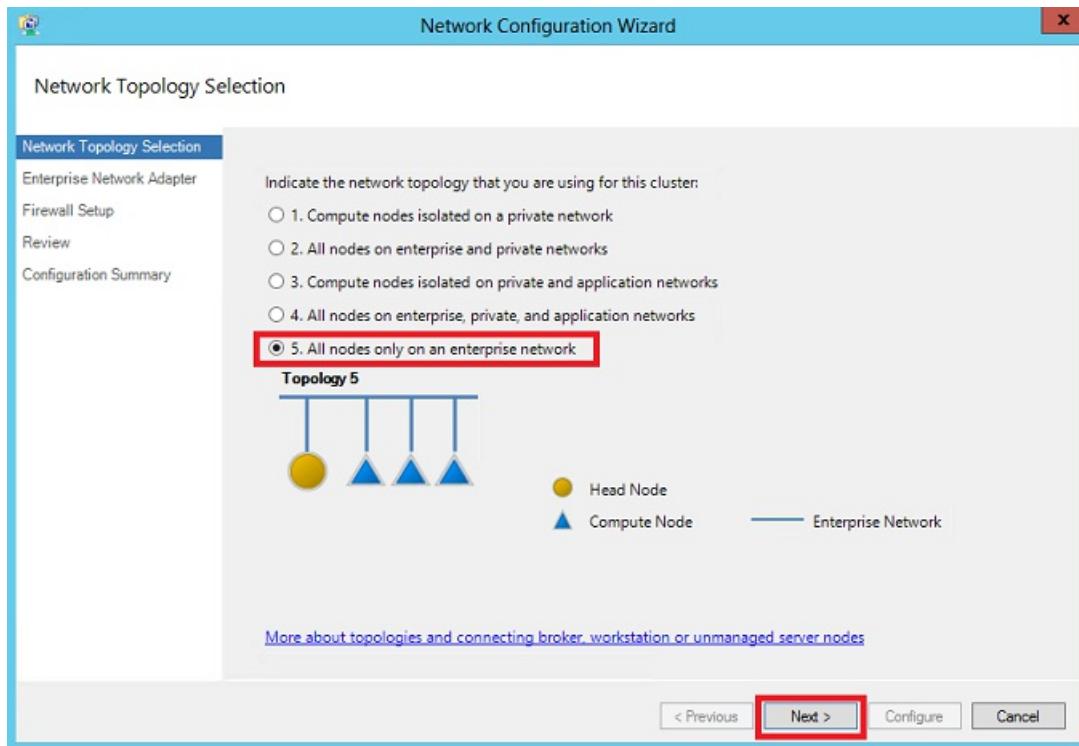
Configure the head node

To use HPC Cluster Manager to deploy Azure nodes and to submit jobs, first perform some required cluster configuration steps.

1. On the head node, start HPC Cluster Manager. If the **Select Head Node** dialog box appears, click **Local Computer**. The **Deployment To-do List** appears.
2. Under **Required deployment tasks**, click **Configure your network**.



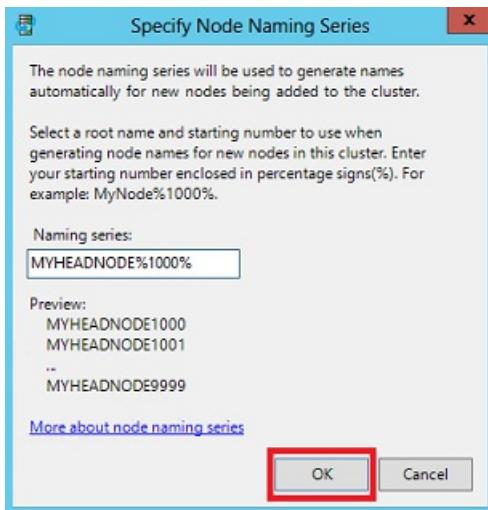
3. In the Network Configuration Wizard, select **All nodes only on an enterprise network** (Topology 5). This network configuration is the simplest for demonstration purposes.



4. Click **Next** to accept default values on the remaining pages of the wizard. Then, on the **Review** tab, click **Configure** to complete the network configuration.
5. In the **Deployment To-do List**, click **Provide installation credentials**.
6. In the **Installation Credentials** dialog box, type the credentials of the domain account that you used to install HPC Pack. Then click **OK**.

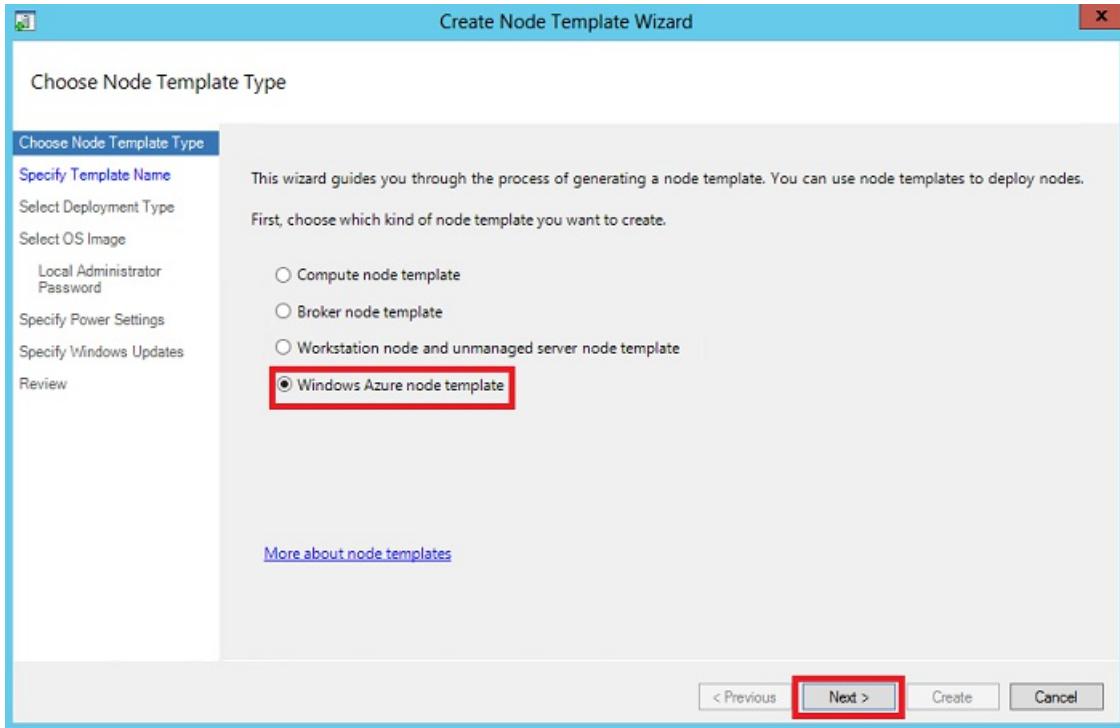


7. In the **Deployment To-do List**, click **Configure the naming of new nodes**.
8. In the **Specify Node Naming Series** dialog box, accept the default naming series and click **OK**. Complete this step even though the Azure nodes you add in this tutorial are named automatically.

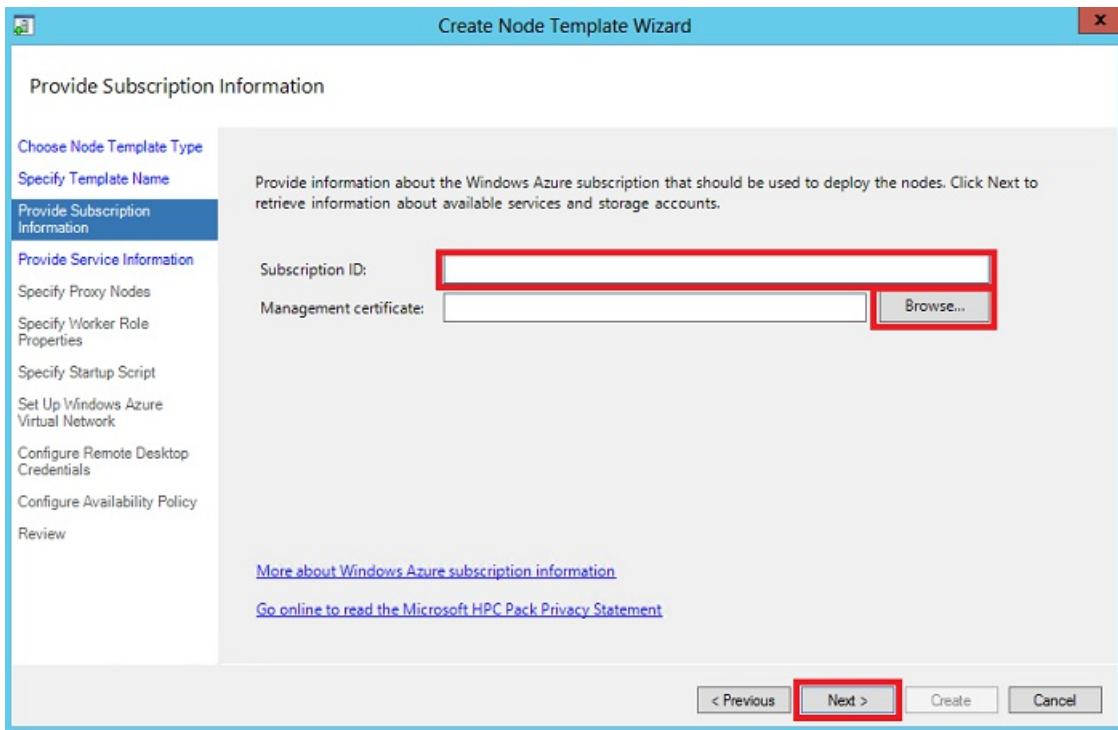


9. In the **Deployment To-do List**, click **Create a node template**. Later in the tutorial, you use the node template to add Azure nodes to the cluster.
10. In the Create Node Template Wizard, do the following:

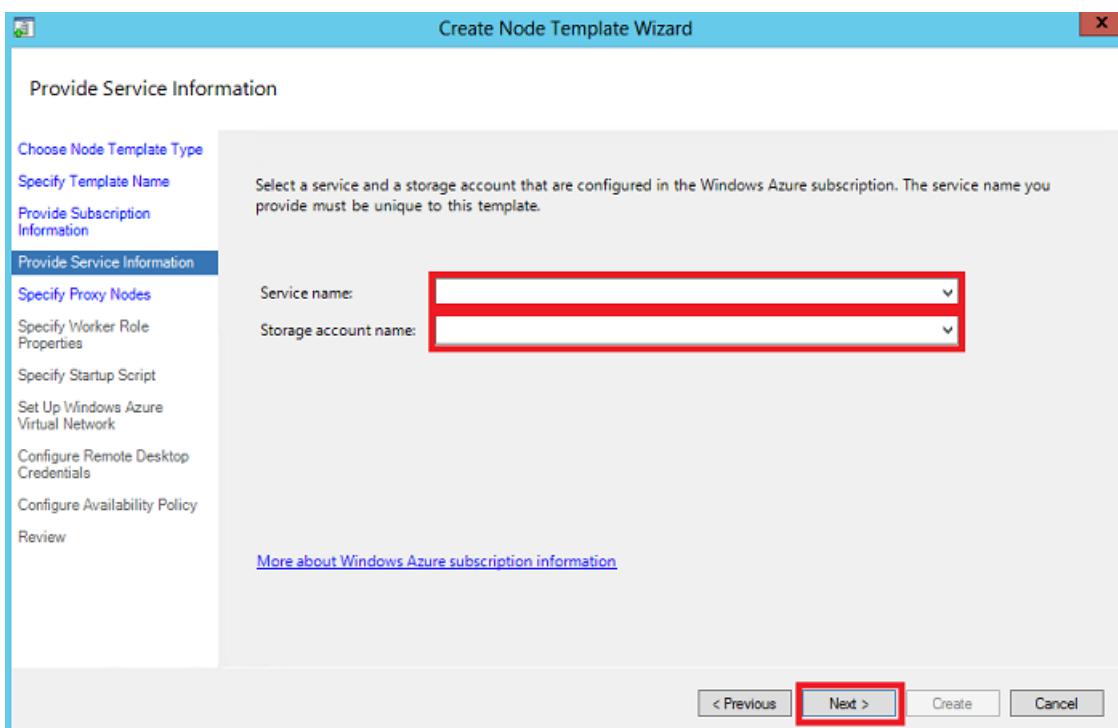
- a. On the **Choose Node Template Type** page, click **Windows Azure node template**, and then click **Next**.



- b. Click **Next** to accept the default template name.
- c. On the **Provide Subscription Information** page, enter your Azure subscription ID (available in your Azure account information). Then, in **Management certificate**, browse for **Default Microsoft HPC Azure Management**. Then click **Next**.



- d. On the **Provide Service Information** page, select the cloud service and the storage account that you created in a previous step. Then click **Next**.



- e. Click **Next** to accept default values on the remaining pages of the wizard. Then, on the **Review** tab, click **Create** to create the node template.

NOTE

By default, the Azure node template includes settings for you to start (provision) and stop the nodes manually, using HPC Cluster Manager. You can optionally configure a schedule to start and stop the Azure nodes automatically.

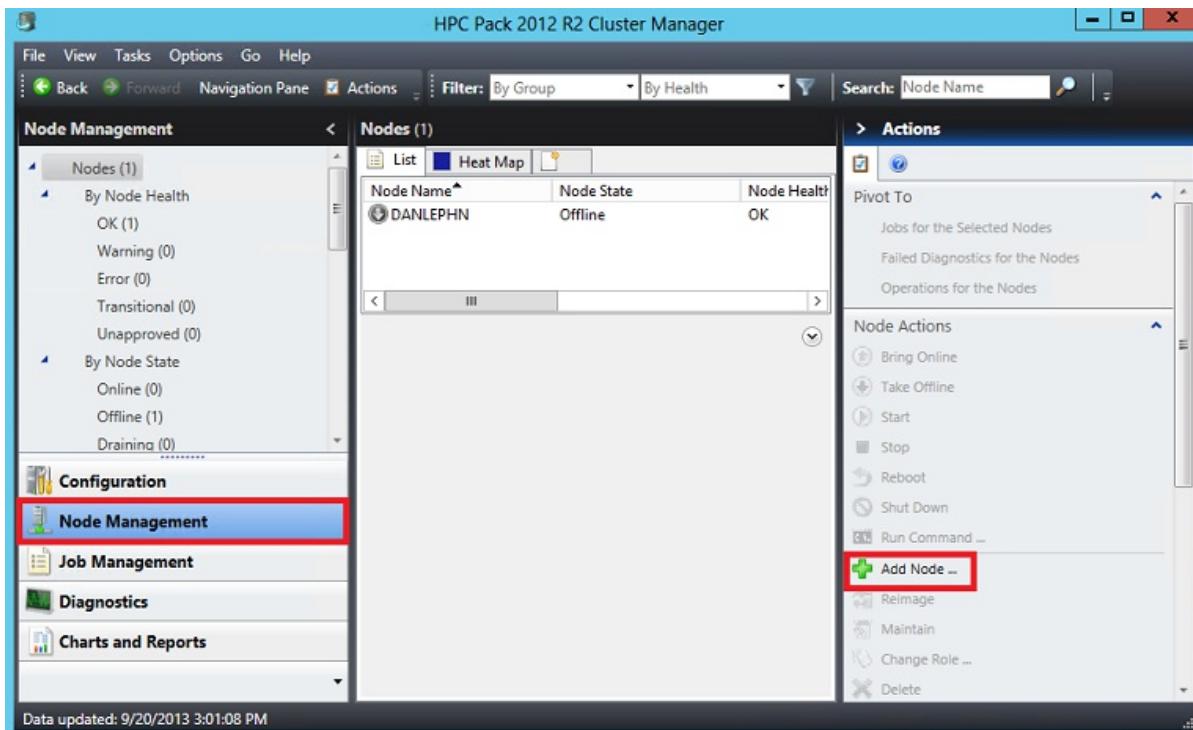
Add Azure nodes to the cluster

Now use the node template to add Azure nodes to the cluster. Adding the nodes to the cluster stores their

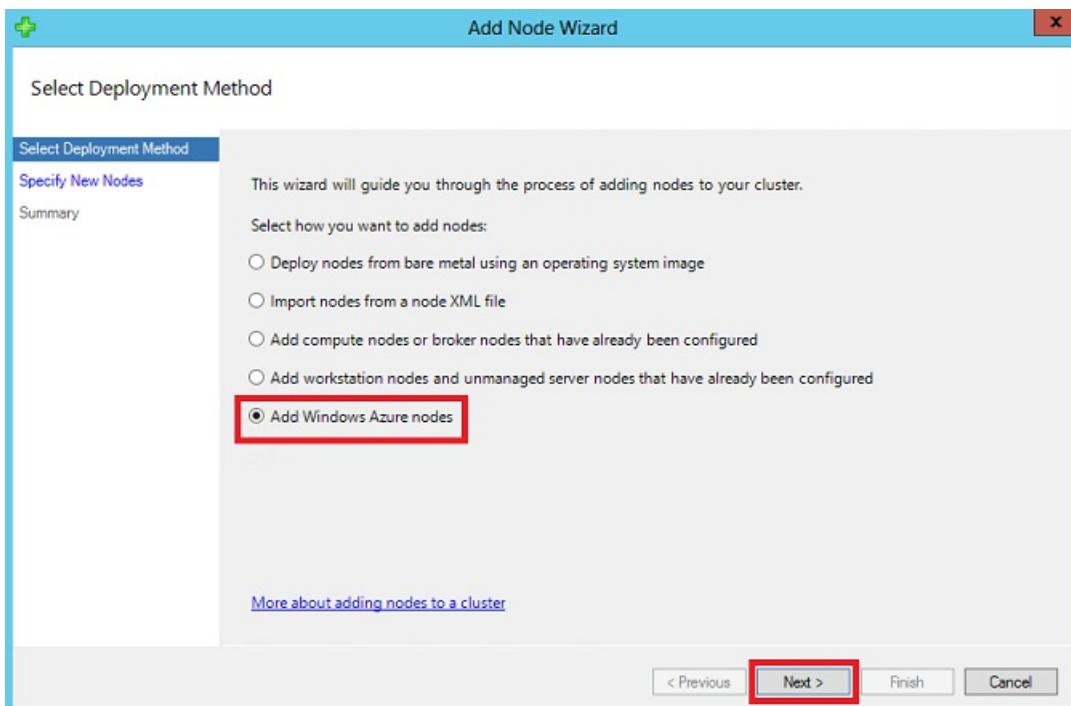
configuration information so that you can start (provision) them at any time in the cloud service. Your subscription only gets charged for Azure nodes after the instances are running in the cloud service.

Follow these steps to add two Small nodes.

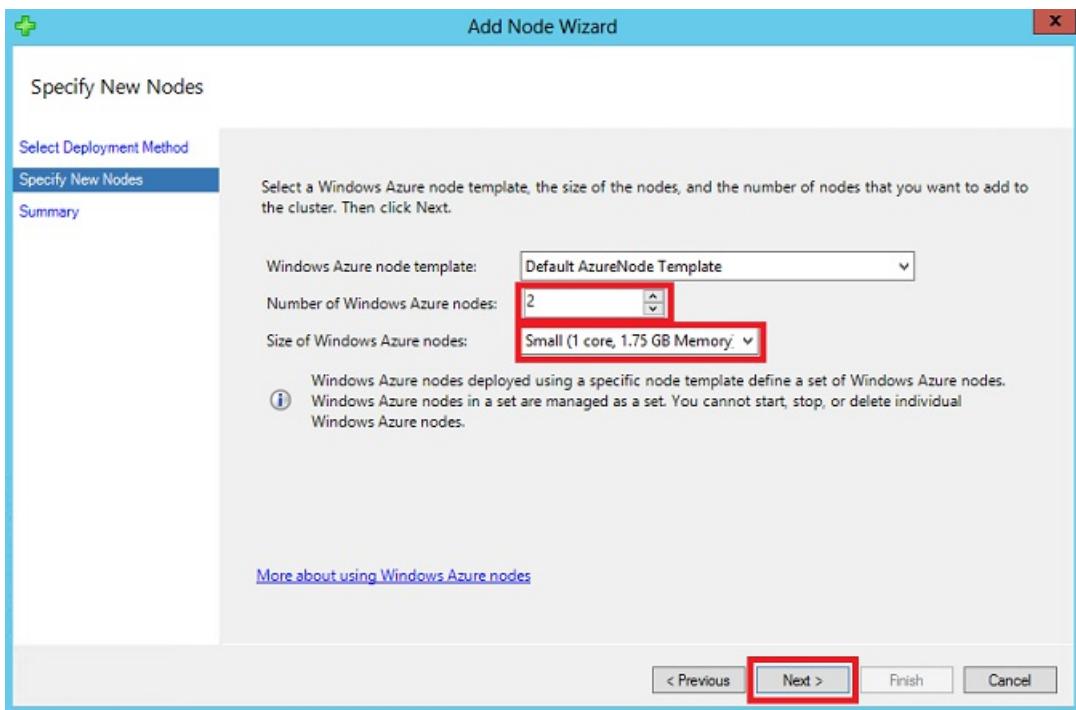
1. In HPC Cluster Manager, click **Node Management** (called **Resource Management** in current versions of HPC Pack) > **Add Node**.



2. In the Add Node Wizard, on the **Select Deployment Method** page, click **Add Windows Azure nodes**, and then click **Next**.



3. On the **Specify New Nodes** page, select the Azure node template you created previously (called by default **Default AzureNode Template**). Then specify **2** nodes of size **Small**, and then click **Next**.



4. On the **Completing the Add Node Wizard** page, click **Finish**.

Two Azure nodes, named **AzureCN-0001** and **AzureCN-0002**, now appear in HPC Cluster Manager. Both are in the **Not-Deployed** state.

The screenshot shows the 'HPC Pack 2012 R2 Cluster Manager' application window. The left sidebar has tabs for 'Node Management', 'Configuration', 'Job Management', 'Diagnostics', and 'Charts and Reports'. The 'Node Management' tab is currently selected. The main area displays a table titled 'Nodes (3)' with columns for 'Node Name', 'Node State', and 'Node Health'. Two nodes are listed: 'AzureCN-0001' and 'AzureCN-0002', both marked as 'Not-Deployed' and 'Unapproved'. A third node, 'DANLEPHN', is listed as 'Offline' with 'OK' health. To the right of the table is an 'Actions' panel with buttons for 'Pivot To', 'Node Actions' (including Bring Online, Take Offline, Start, Stop, Reboot, Shut Down, Run Command ...), and 'Add Node ...'. The status bar at the bottom indicates 'Data updated: 9/20/2013 3:07:17 PM'.

Start the Azure nodes

When you want to use the cluster resources in Azure, use HPC Cluster Manager to start (provision) the Azure nodes and bring them online.

1. In HPC Cluster Manager, click **Node Management** (called **Resource Management** in current versions of HPC Pack), and select the Azure nodes.
2. Click **Start**, and then click **OK**.

The screenshot shows the HPC Pack 2012 R2 Cluster Manager interface. In the 'Nodes (3)' list, 'AzureCN-0001' and 'AzureCN-0002' are highlighted with a red box and show 'Not-Deployed' status under 'Node State'. The 'Actions' pane on the right has a 'Start' button highlighted with a red box.

Node Name	Node State	Node Health
AzureCN-0001	Not-Deployed	Unapproved
AzureCN-0002	Not-Deployed	Unapproved
DANLEPHN	Offline	OK

The nodes transition to the **Provisioning** state. View the provisioning log to track the provisioning progress.

The screenshot shows the HPC Pack 2012 R2 Cluster Manager interface. In the 'Nodes (3)' list, 'AzureCN-0001' and 'AzureCN-0002' are highlighted with a red box and show 'Provisioning' status under 'Node State'. The 'Actions' pane on the right has a 'Start' button highlighted with a red box. A message in the center pane indicates provisioning started at 9/20/2013 3:18:09 PM.

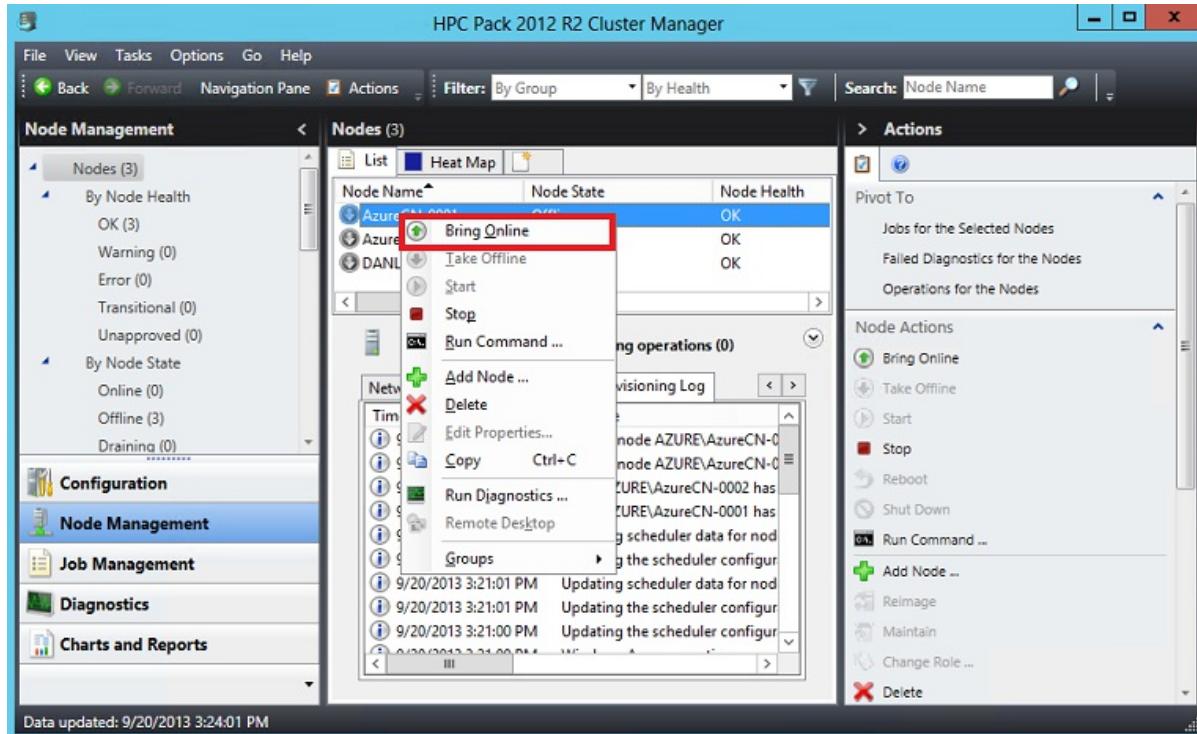
Node Name	Node State	Node Health
AzureCN-0001	Provisioning	Transitional
AzureCN-0002	Provisioning	Transitional
DANLEPHN	Offline	OK

3. After a few minutes, the Azure nodes finish provisioning and are in the **Offline** state. In this state, the role instances are running but cannot yet accept cluster jobs.
4. To confirm that the role instances are running, in the Azure portal, click **Cloud Services (classic) > your_cloud_service_name**.

You should see two **HpcWorkerRole** instances (nodes) running in the service. HPC Pack also automatically deploys two **HpcProxy** instances (size Medium) to handle communication between the head node and Azure.

NAME	STATUS	SIZE	UPDATE	FAULT
HpcProxy				
HpcProxy_IN_0	Running	Medium	0	0
HpcProxy_IN_1	Running	Medium	1	1
HpcWorkerRole1				
HpcWorkerRole1_IN_0	Running	Small	0	0
HpcWorkerRole1_IN_1	Running	Small	1	1

5. To bring the Azure nodes online to run cluster jobs, select the nodes, right-click, and then click **Bring Online**.



HPC Cluster Manager indicates that the nodes are in the **Online** state.

Run a command across the cluster

To check the installation, use the HPC Pack **clusrun** command to run a command or application on one or more cluster nodes. As a simple example, use **clusrun** to get the IP configuration of the Azure nodes.

1. On the head node, open a command prompt as an administrator.
2. Type the following command:

```
clusrun /nodes:azurecn* ipconfig
```

3. If prompted, enter your cluster administrator password. You should see command output similar to the following.

```
C:\Users\ClusterAdmin>clusrun /nodes:azurecn* ipconfig
----- AZURECN-0002 returns 0 -----
Windows IP Configuration

Ethernet adapter Ethernet:

  Connection-specific DNS Suffix  . : reddog.microsoft.com
  Link-local IPv6 Address . . . . . : fe80::bded:3bd4:ac84:7da2%12
    IPv4 Address . . . . . : 10.142.78.36
    Subnet Mask . . . . . : 255.255.254.0
    Default Gateway . . . . . : 10.142.78.1

Tunnel adapter Local Area Connection* 12:

  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix' . : reddog.microsoft.com

Tunnel adapter Local Area Connection* 11:

  Connection-specific DNS Suffix  . :
  IPv6 Address . . . . . : 2001:0:9d38:953c:345f:14c9:f571:b1db
  Link-local IPv6 Address . . . . . : fe80::345f:14c9:f571:b1db%20
  Default Gateway . . . . . : ::

----- AZURECN-0001 returns 0 -----

Windows IP Configuration

Ethernet adapter Ethernet:

  Connection-specific DNS Suffix  . : reddog.microsoft.com
  Link-local IPv6 Address . . . . . : fe80::31d9:9809:4672:f3e3%12
    IPv4 Address . . . . . : 10.142.106.26
    Subnet Mask . . . . . : 255.255.254.0
    Default Gateway . . . . . : 10.142.106.1

Tunnel adapter Local Area Connection* 11:

  Media State . . . . . : Media disconnected
  Connection-specific DNS Suffix' . : reddog.microsoft.com

Tunnel adapter Local Area Connection* 12:

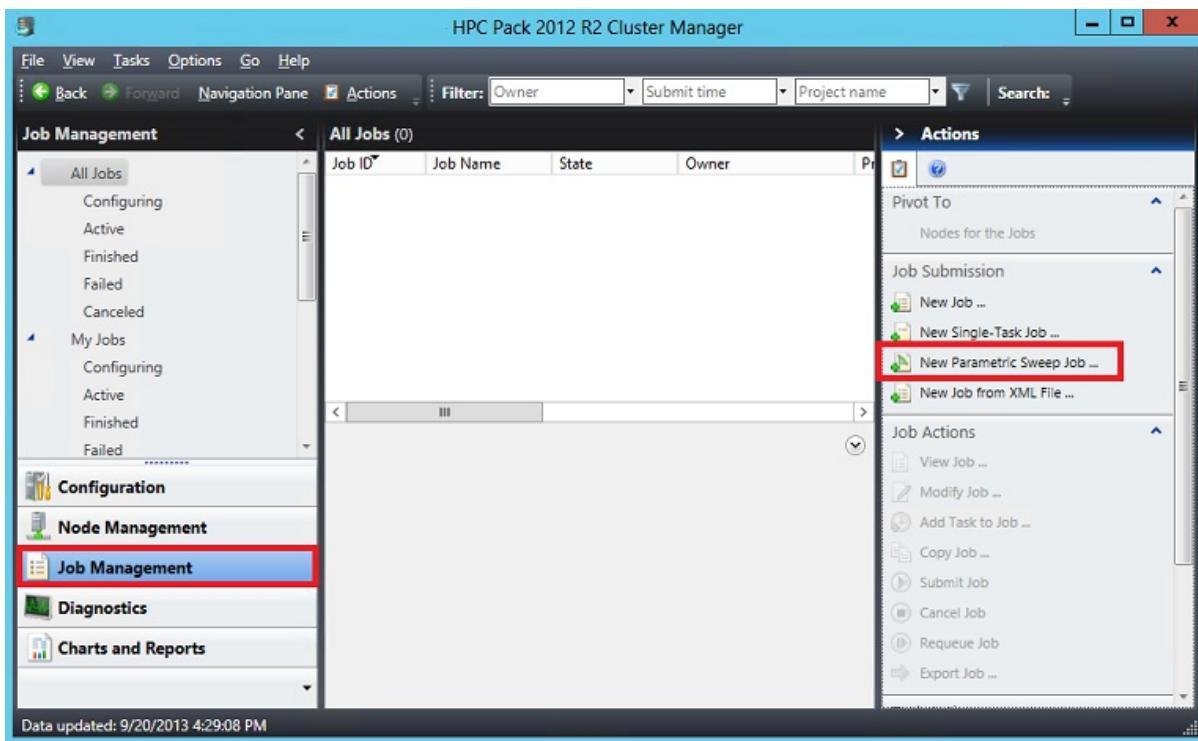
  Connection-specific DNS Suffix  . :
  IPv6 Address . . . . . : 2001:0:5ef5:79fd:3816:2359:f571:95e5
  Link-local IPv6 Address . . . . . : fe80::3816:2359:f571:95e5%20
  Default Gateway . . . . . : ::

----- Summary -----
2 Nodes succeeded
0 Nodes failed
```

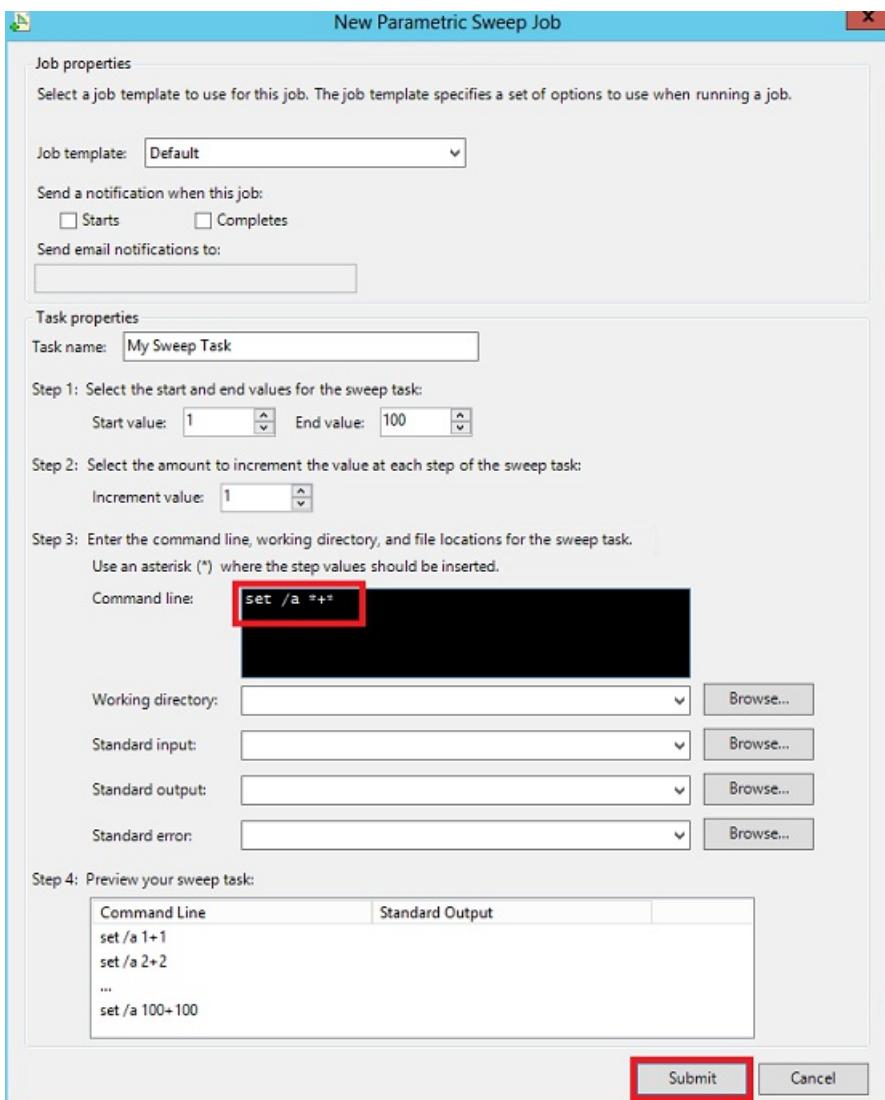
Run a test job

Now submit a test job that runs on the hybrid cluster. This example is a simple parametric sweep job (a type of intrinsically parallel computation). This example runs subtasks that add an integer to itself by using the **set/a** command. All the nodes in the cluster contribute to finishing the subtasks for integers from 1 to 100.

1. In HPC Cluster Manager, click **Job Management > New Parametric Sweep Job**.

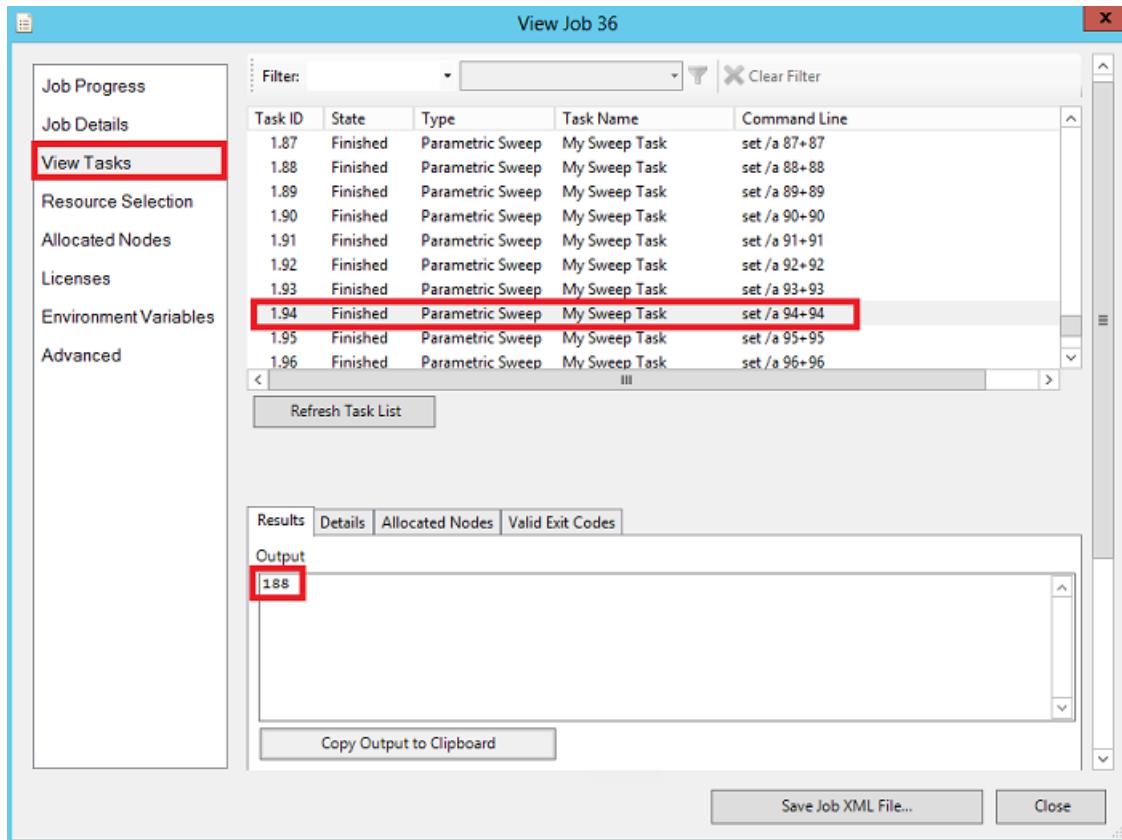


2. In the **New Parametric Sweep Job** dialog box, in **Command line**, type `set /a *+*` (overwriting the default command line that appears). Leave default values for the remaining settings, and then click **Submit** to submit the job.

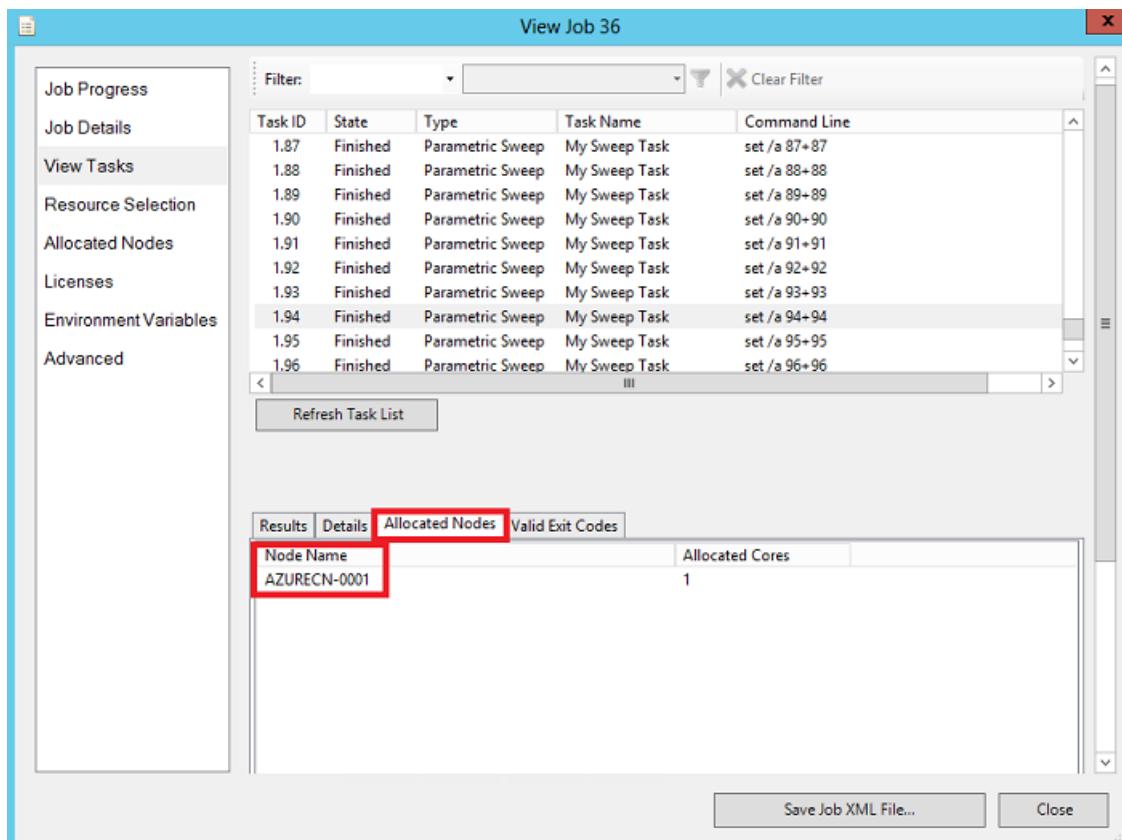


3. When the job is finished, double-click the **My Sweep Task** job.

4. Click **View Tasks**, and then click a subtask to view the calculated output of that subtask.



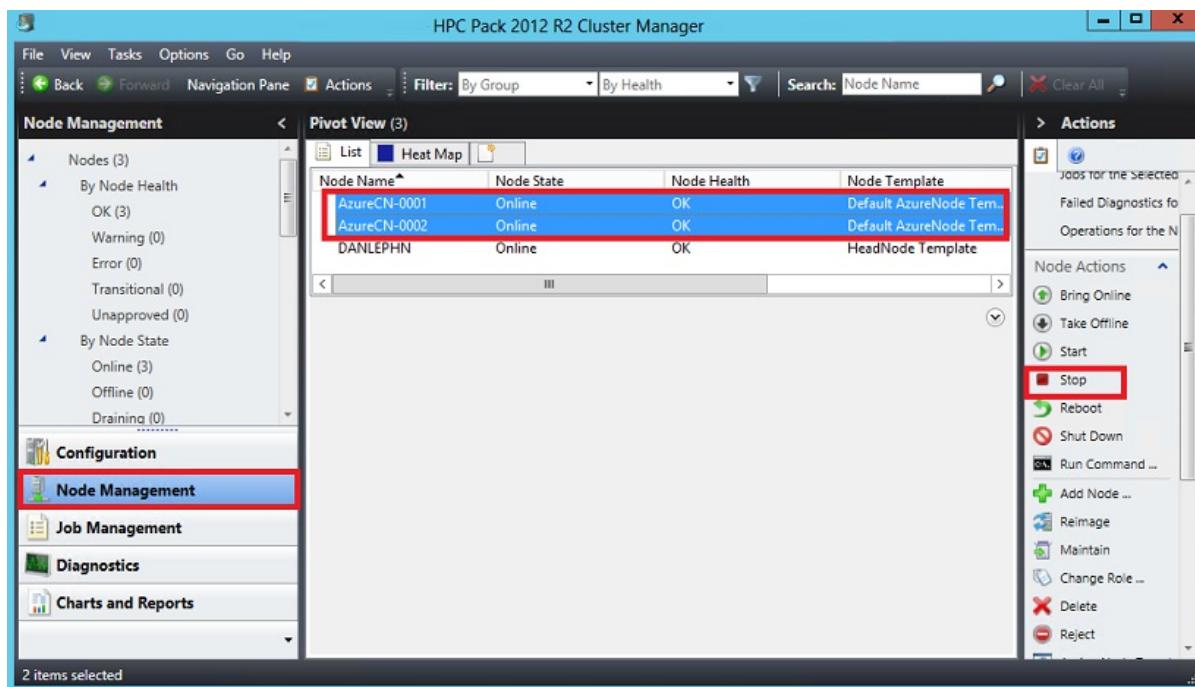
5. To see which node performed the calculation for that subtask, click **Allocated Nodes**. (Your cluster might show a different node name.)



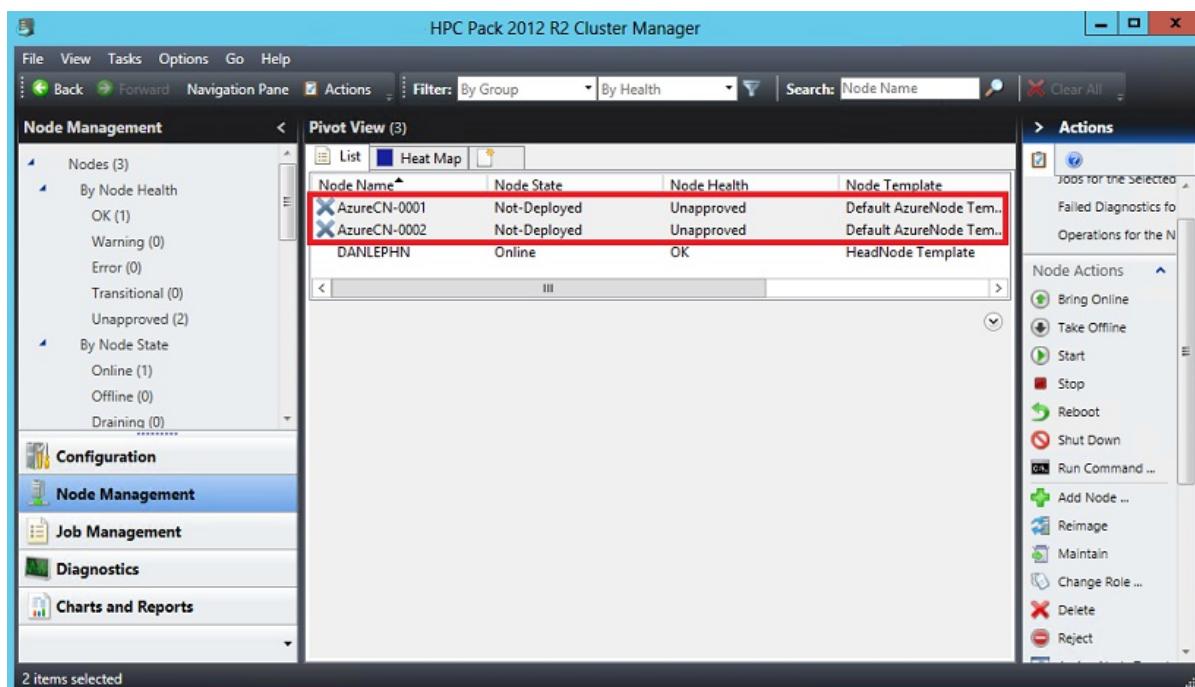
Stop the Azure nodes

After you try out the cluster, stop the Azure nodes to avoid unnecessary charges to your account. This step stops the cloud service and removes the Azure role instances.

1. In HPC Cluster Manager, in **Node Management** (called **Resource Management** in previous versions of HPC Pack), select both Azure nodes. Then, click **Stop**.



2. In the **Stop Windows Azure nodes** dialog box, click **Stop**.
3. The nodes transition to the **Stopping** state. After a few minutes, HPC Cluster Manager shows that the nodes are **Not-Deployed**.



4. To confirm that the role instances are no longer running in Azure, in the Azure portal, click **Cloud services (classic)** > *your_cloud_service_name*. No instances are deployed in the production environment.

This completes the tutorial.

Next steps

- Explore the documentation for [HPC Pack](#).
- To set up a hybrid HPC Pack cluster deployment at greater scale, see [Burst to Azure Worker Role Instances with Microsoft HPC Pack](#).

- For other ways to create an HPC Pack cluster in Azure, including using Azure Resource Manager templates, see [HPC cluster options with Microsoft HPC Pack in Azure](#).

Sizes for Cloud Services

10/25/2018 • 9 minutes to read • [Edit Online](#)

This topic describes the available sizes and options for Cloud Service role instances (web roles and worker roles). It also provides deployment considerations to be aware of when planning to use these resources. Each size has an ID that you put in your [service definition file](#). Prices for each size are available on the [Cloud Services Pricing](#) page.

NOTE

To see related Azure limits, see [Azure Subscription and Service Limits, Quotas, and Constraints](#)

Sizes for web and worker role instances

There are multiple standard sizes to choose from on Azure. Considerations for some of these sizes include:

- D-series VMs are designed to run applications that demand higher compute power and temporary disk performance. D-series VMs provide faster processors, a higher memory-to-core ratio, and a solid-state drive (SSD) for the temporary disk. For details, see the announcement on the Azure blog, [New D-Series Virtual Machine Sizes](#).
- Dv3-series, Dv2-series, a follow-on to the original D-series, features a more powerful CPU. The Dv2-series CPU is about 35% faster than the D-series CPU. It is based on the latest generation 2.4 GHz Intel Xeon® E5-2673 v3 (Haswell) processor, and with the Intel Turbo Boost Technology 2.0, can go up to 3.1 GHz. The Dv2-series has the same memory and disk configurations as the D-series.
- G-series VMs offer the most memory and run on hosts that have Intel Xeon E5 V3 family processors.
- The A-series VMs can be deployed on various hardware types and processors. The size is throttled, based on the hardware, to offer consistent processor performance for the running instance, regardless of the hardware it is deployed on. To determine the physical hardware on which this size is deployed, query the virtual hardware from within the Virtual Machine.
- The A0 size is over-subscribed on the physical hardware. For this specific size only, other customer deployments may impact the performance of your running workload. The relative performance is outlined below as the expected baseline, subject to an approximate variability of 15 percent.

The size of the virtual machine affects the pricing. The size also affects the processing, memory, and storage capacity of the virtual machine. Storage costs are calculated separately based on used pages in the storage account. For details, see [Cloud Services Pricing Details](#) and [Azure Storage Pricing](#).

The following considerations might help you decide on a size:

- The A8-A11 and H-series sizes are also known as *compute-intensive instances*. The hardware that runs these sizes is designed and optimized for compute-intensive and network-intensive applications, including high-performance computing (HPC) cluster applications, modeling, and simulations. The A8-A11 series uses Intel Xeon E5-2670 @ 2.6 GHZ and the H-series uses Intel Xeon E5-2667 v3 @ 3.2 GHz. For detailed information and considerations about using these sizes, see [High performance compute VM sizes](#).
- Dv3-series, Dv2-series, D-series, G-series, are ideal for applications that demand faster CPUs, better local disk performance, or have higher memory demands. They offer a powerful combination for many enterprise-grade applications.
- Some of the physical hosts in Azure data centers may not support larger virtual machine sizes, such as A5 – A11. As a result, you may see the error message **Failed to configure virtual machine {machine name}** or **Failed to create virtual machine {machine name}** when resizing an existing virtual machine to a new size;

creating a new virtual machine in a virtual network created before April 16, 2013; or adding a new virtual machine to an existing cloud service. See [Error: "Failed to configure virtual machine"](#) on the support forum for workarounds for each deployment scenario.

- Your subscription might also limit the number of cores you can deploy in certain size families. To increase a quota, contact Azure Support.

Performance considerations

We have created the concept of the Azure Compute Unit (ACU) to provide a way of comparing compute (CPU) performance across Azure SKUs and to identify which SKU is most likely to satisfy your performance needs. ACU is currently standardized on a Small (Standard_A1) VM being 100 and all other SKUs then represent approximately how much faster that SKU can run a standard benchmark.

IMPORTANT

The ACU is only a guideline. The results for your workload may vary.

SKU FAMILY	ACU/CORE
ExtraSmall	50
Small-ExtraLarge	100
A5-7	100
A8-A11	225*
A v2	100
D	160
D v2	160 - 190*
D v3	160 - 190*
E v3	160 - 190*
G	180 - 240*
H	290 - 300*

ACUs marked with a * use Intel® Turbo technology to increase CPU frequency and provide a performance boost. The amount of the boost can vary based on the VM size, workload, and other workloads running on the same host.

Size tables

The following tables show the sizes and the capacities they provide.

- Storage capacity is shown in units of GiB or 1024^3 bytes. When comparing disks measured in GB (1000^3 bytes) to disks measured in GiB (1024^3) remember that capacity numbers given in GiB may appear smaller.

For example, 1023 GiB = 1098.4 GB

- Disk throughput is measured in input/output operations per second (IOPS) and MBps where MBps = 10^6 bytes/sec.
- Data disks can operate in cached or uncached modes. For cached data disk operation, the host cache mode is set to **ReadOnly** or **ReadWrite**. For uncached data disk operation, the host cache mode is set to **None**.
- Maximum network bandwidth is the maximum aggregated bandwidth allocated and assigned per VM type. The maximum bandwidth provides guidance for selecting the right VM type to ensure adequate network capacity is available. When moving between Low, Moderate, High and Very High, the throughput increases accordingly. Actual network performance will depend on many factors including network and application loads, and application network settings.

A-series

SIZE	CPU CORES	MEMORY: GiB	TEMPORARY STORAGE: GiB	MAX NICs / NETWORK BANDWIDTH
ExtraSmall	1	0.768	20	1 / low
Small	1	1.75	225	1 / moderate
Medium	2	3.5	490	1 / moderate
Large	4	7	1000	2 / high
ExtraLarge	8	14	2040	4 / high
A5	2	14	490	1 / moderate
A6	4	28	1000	2 / high
A7	8	56	2040	4 / high

A-series - compute-intensive instances

For information and considerations about using these sizes, see [High performance compute VM sizes](#).

SIZE	CPU CORES	MEMORY: GiB	TEMPORARY STORAGE: GiB	MAX NICs / NETWORK BANDWIDTH
A8*	8	56	1817	2 / high
A9*	16	112	1817	4 / very high
A10	8	56	1817	2 / high
A11	16	112	1817	4 / very high

*RDMA capable

Av2-series

SIZE	CPU CORES	MEMORY: GIB	TEMPORARY STORAGE (SSD): GIB	MAX NICs / NETWORK BANDWIDTH
Standard_A1_v2	1	2	10	1 / moderate
Standard_A2_v2	2	4	20	2 / moderate
Standard_A4_v2	4	8	40	4 / high
Standard_A8_v2	8	16	80	8 / high
Standard_A2m_v2	2	16	20	2 / moderate
Standard_A4m_v2	4	32	40	4 / high
Standard_A8m_v2	8	64	80	8 / high

D-series

SIZE	CPU CORES	MEMORY: GIB	TEMPORARY STORAGE (SSD): GIB	MAX NICs / NETWORK BANDWIDTH
Standard_D1	1	3.5	50	1 / moderate
Standard_D2	2	7	100	2 / high
Standard_D3	4	14	200	4 / high
Standard_D4	8	28	400	8 / high
Standard_D11	2	14	100	2 / high
Standard_D12	4	28	200	4 / high
Standard_D13	8	56	400	8 / high
Standard_D14	16	112	800	8 / very high

Dv2-series

SIZE	CPU CORES	MEMORY: GIB	TEMPORARY STORAGE (SSD): GIB	MAX NICs / NETWORK BANDWIDTH
Standard_D1_v2	1	3.5	50	1 / moderate
Standard_D2_v2	2	7	100	2 / high
Standard_D3_v2	4	14	200	4 / high
Standard_D4_v2	8	28	400	8 / high
Standard_D5_v2	16	56	800	8 / extremely high

SIZE	CPU CORES	MEMORY: GIB	TEMPORARY STORAGE (SSD): GIB	MAX NICs / NETWORK BANDWIDTH
Standard_D11_v2	2	14	100	2 / high
Standard_D12_v2	4	28	200	4 / high
Standard_D13_v2	8	56	400	8 / high
Standard_D14_v2	16	112	800	8 / extremely high
Standard_D15_v2	20	140	1,000	8 / extremely high

Dv3-series

SIZE	CPU CORES	MEMORY: GIB	TEMPORARY STORAGE (SSD): GIB	MAX NICs / NETWORK BANDWIDTH
Standard_D2_v3	2	8	50	2 / moderate
Standard_D4_v3	4	16	100	2 / high
Standard_D8_v3	8	32	200	4 / high
Standard_D16_v3	16	64	400	8 / extremely high
Standard_D32_v3	32	128	800	8 / extremely high
Standard_D64_v3	64	256	1600	8 / extremely high

Ev3-series

SIZE	CPU CORES	MEMORY: GIB	TEMPORARY STORAGE (SSD): GIB	MAX NICs / NETWORK BANDWIDTH
Standard_E2_v3	2	16	50	2 / moderate
Standard_E4_v3	4	32	100	2 / high
Standard_E8_v3	8	64	200	4 / high
Standard_E16_v3	16	128	400	8 / extremely high
Standard_E32_v3	32	256	800	8 / extremely high
Standard_E64_v3	64	432	1600	8 / extremely high

G-series

SIZE	CPU CORES	MEMORY: GIB	TEMPORARY STORAGE (SSD): GIB	MAX NICs / NETWORK BANDWIDTH
Standard_G1	2	28	384	1 / high
Standard_G2	4	56	768	2 / high
Standard_G3	8	112	1,536	4 / very high
Standard_G4	16	224	3,072	8 / extremely high
Standard_G5	32	448	6,144	8 / extremely high

H-series

Azure H-series virtual machines are the next generation high performance computing VMs aimed at high end computational needs, like molecular modeling, and computational fluid dynamics. These 8 and 16 core VMs are built on the Intel Haswell E5-2667 V3 processor technology featuring DDR4 memory and local SSD-based storage.

In addition to the substantial CPU power, the H-series offers diverse options for low latency RDMA networking using FDR InfiniBand and several memory configurations to support memory intensive computational requirements.

SIZE	CPU CORES	MEMORY: GIB	TEMPORARY STORAGE (SSD): GIB	MAX NICs / NETWORK BANDWIDTH
Standard_H8	8	56	1000	8 / high
Standard_H16	16	112	2000	8 / very high
Standard_H8m	8	112	1000	8 / high
Standard_H16m	16	224	2000	8 / very high
Standard_H16r*	16	112	2000	8 / very high
Standard_H16mr*	16	224	2000	8 / very high

*RDMA capable

Configure sizes for Cloud Services

You can specify the Virtual Machine size of a role instance as part of the service model described by the [service definition file](#). The size of the role determines the number of CPU cores, the memory capacity, and the local file system size that is allocated to a running instance. Choose the role size based on your application's resource requirement.

Here is an example for setting the role size to be [Standard_D2](#) for a Web Role instance:

```
<WorkerRole name="Worker1" vmsize="Standard_D2">
...
</WorkerRole>
```

Changing the size of an existing role

As the nature of your workload changes or new VM sizes become available, you may want to change the size of your role. To do so, you must change the VM size in your service definition file (as shown above), repackage your Cloud Service, and deploy it. It is not possible to change VM sizes directly from the portal or PowerShell.

TIP

You may want to use different VM sizes for your role in different environments (eg. test vs production). One way to do this is to create multiple service definition (.csdef) files in your project, then create different cloud service packages per environment during your automated build using the CSPack tool. To learn more about the elements of a cloud services package and how to create them, see [What is the cloud services model and how do I package it?](#)

Get a list of sizes

You can use PowerShell or the REST API to get a list of sizes. The REST API is documented [here](#). The following code is a PowerShell command that will list all the sizes available for Cloud Services.

```
Get-AzureRoleSize | where SupportedByWebWorkerRoles -eq $true | select InstanceSize, RoleSizeLabel
```

Next steps

- Learn about [azure subscription and service limits, quotas, and constraints](#).
- Learn more [about high performance compute VM sizes](#) for HPC workloads.

How to update a cloud service

7/12/2018 • 11 minutes to read • [Edit Online](#)

Updating a cloud service, including both its roles and guest OS, is a three step process. First, the binaries and configuration files for the new cloud service or OS version must be uploaded. Next, Azure reserves compute and network resources for the cloud service based on the requirements of the new cloud service version. Finally, Azure performs a rolling upgrade to incrementally update the tenant to the new version or guest OS, while preserving your availability. This article discusses the details of this last step – the rolling upgrade.

Update an Azure Service

Azure organizes your role instances into logical groupings called upgrade domains (UD). Upgrade domains (UD) are logical sets of role instances that are updated as a group. Azure updates a cloud service one UD at a time, which allows instances in other UD's to continue serving traffic.

The default number of upgrade domains is 5. You can specify a different number of upgrade domains by including the `upgradeDomainCount` attribute in the service's definition file (.csdef). For more information about the `upgradeDomainCount` attribute, see [WebRole Schema](#) or [WorkerRole Schema](#).

When you perform an in-place update of one or more roles in your service, Azure updates sets of role instances according to the upgrade domain to which they belong. Azure updates all of the instances in a given upgrade domain – stopping them, updating them, bringing them back on-line – then moves onto the next domain. By stopping only the instances running in the current upgrade domain, Azure makes sure that an update occurs with the least possible impact to the running service. For more information, see [How the update proceeds](#) later in this article.

NOTE

While the terms **update** and **upgrade** have slightly different meaning in the context Azure, they can be used interchangeably for the processes and descriptions of the features in this document.

Your service must define at least two instances of a role for that role to be updated in-place without downtime. If the service consists of only one instance of one role, your service will be unavailable until the in-place update has finished.

This topic covers the following information about Azure updates:

- [Allowed service changes during an update](#)
- [How an upgrade proceeds](#)
- [Rollback of an update](#)
- [Initiating multiple mutating operations on an ongoing deployment](#)
- [Distribution of roles across upgrade domains](#)

Allowed service changes during an update

The following table shows the allowed changes to a service during an update:

CHANGES PERMITTED TO HOSTING, SERVICES, AND ROLES	IN-PLACE UPDATE	STAGED (VIP SWAP)	DELETE AND RE-DEPLOY
Operating system version	Yes	Yes	Yes
.NET trust level	Yes	Yes	Yes
Virtual machine size ¹	Yes ²	Yes	Yes
Local storage settings	Increase only ²	Yes	Yes
Add or remove roles in a service	Yes	Yes	Yes
Number of instances of a particular role	Yes	Yes	Yes
Number or type of endpoints for a service	Yes ²	No	Yes
Names and values of configuration settings	Yes	Yes	Yes
Values (but not names) of configuration settings	Yes	Yes	Yes
Add new certificates	Yes	Yes	Yes
Change existing certificates	Yes	Yes	Yes
Deploy new code	Yes	Yes	Yes

¹ Size change limited to the subset of sizes available for the cloud service.

² Requires Azure SDK 1.5 or later versions.

WARNING

Changing the virtual machine size will destroy local data.

The following items are not supported during an update:

- Changing the name of a role. Remove and then add the role with the new name.
- Changing of the Upgrade Domain count.
- Decreasing the size of the local resources.

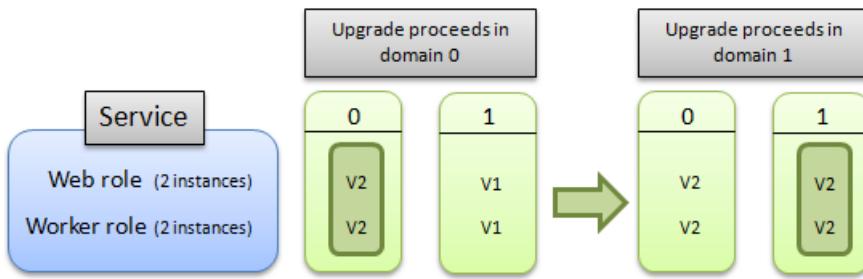
If you are making other updates to your service's definition, such as decreasing the size of local resource, you must perform a VIP swap update instead. For more information, see [Swap Deployment](#).

How an upgrade proceeds

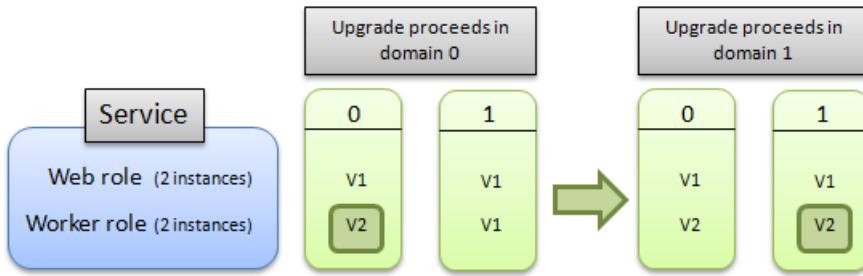
You can decide whether you want to update all of the roles in your service or a single role in the service. In either case, all instances of each role that is being upgraded and belong to the first upgrade domain are stopped, upgraded, and brought back online. Once they are back online, the instances in the second upgrade domain are

stopped, upgraded, and brought back online. A cloud service can have at most one upgrade active at a time. The upgrade is always performed against the latest version of the cloud service.

The following diagram illustrates how the upgrade proceeds if you are upgrading all of the roles in the service:



This next diagram illustrates how the update proceeds if you are upgrading only a single role:



During an automatic update, the Azure Fabric Controller periodically evaluates the health of the cloud service to determine when it's safe to walk the next UD. This health evaluation is performed on a per-role basis and considers only instances in the latest version (i.e. instances from UDs that have already been walked). It verifies that a minimum number of role instances, for each role, have achieved a satisfactory terminal state.

Role Instance Start Timeout

The Fabric Controller will wait 30 minutes for each role instance to reach a Started state. If the timeout duration elapses, the Fabric Controller will continue walking to the next role instance.

Impact to drive data during Cloud Service upgrades

When upgrading a service from a single instance to multiple instances your service will be brought down while the upgrade is performed due to the way Azure upgrades services. The service level agreement guaranteeing service availability only applies to services that are deployed with more than one instance. The following list describes how the data on each drive is affected by each Azure service upgrade scenario:

SCENARIO	C DRIVE	D DRIVE	E DRIVE
VM reboot	Preserved	Preserved	Preserved
Portal reboot	Preserved	Preserved	Destroyed
Portal reimage	Preserved	Destroyed	Destroyed
In-Place Upgrade	Preserved	Preserved	Destroyed
Node migration	Destroyed	Destroyed	Destroyed

Note that, in the above list, the E: drive represents the role's root drive, and should not be hard-coded. Instead, use the **%RoleRoot%** environment variable to represent the drive.

To minimize the downtime when upgrading a single-instance service, deploy a new multi-instance service to the staging server and perform a VIP swap.

Rollback of an update

Azure provides flexibility in managing services during an update by letting you initiate additional operations on a service, after the initial update request is accepted by the Azure Fabric Controller. A rollback can only be performed when an update (configuration change) or upgrade is in the **in progress** state on the deployment. An update or upgrade is considered to be in-progress as long as there is at least one instance of the service which has not yet been updated to the new version. To test whether a rollback is allowed, check the value of the RollbackAllowed flag, returned by [Get Deployment](#) and [Get Cloud Service Properties](#) operations, is set to true.

NOTE

It only makes sense to call Rollback on an **in-place** update or upgrade because VIP swap upgrades involve replacing one entire running instance of your service with another.

Rollback of an in-progress update has the following effects on the deployment:

- Any role instances which had not yet been updated or upgraded to the new version are not updated or upgraded, because those instances are already running the target version of the service.
- Any role instances which had already been updated or upgraded to the new version of the service package (*.cspkg) file or the service configuration (*.cscfg) file (or both files) are reverted to the pre-upgrade version of these files.

This functionality is provided by the following features:

- The [Rollback Update Or Upgrade](#) operation, which can be called on a configuration update (triggered by calling [Change Deployment Configuration](#)) or an upgrade (triggered by calling [Upgrade Deployment](#)) as long as there is at least one instance in the service which has not yet been updated to the new version.
- The Locked element and the RollbackAllowed element, which are returned as part of the response body of the [Get Deployment](#) and [Get Cloud Service Properties](#) operations:
 1. The Locked element allows you to detect when a mutating operation can be invoked on a given deployment.
 2. The RollbackAllowed element allows you to detect when the [Rollback Update Or Upgrade](#) operation can be called on a given deployment.

In order to perform a rollback, you do not have to check both the Locked and the RollbackAllowed elements. It suffices to confirm that RollbackAllowed is set to true. These elements are only returned if these methods are invoked by using the request header set to "x-ms-version: 2011-10-01" or a later version. For more information about versioning headers, see [Service Management Versioning](#).

There are some situations where a rollback of an update or upgrade is not supported, these are as follows:

- Reduction in local resources - If the update increases the local resources for a role the Azure platform does not allow rolling back.
- Quota limitations - If the update was a scale down operation you may no longer have sufficient compute quota to complete the rollback operation. Each Azure subscription has a quota associated with it that specifies the maximum number of cores which can be consumed by all hosted services that belong to that subscription. If performing a rollback of a given update would put your subscription over quota then that a rollback will not be enabled.
- Race condition - If the initial update has completed, a rollback is not possible.

An example of when the rollback of an update might be useful is if you are using the [Upgrade Deployment](#) operation in manual mode to control the rate at which a major in-place upgrade to your Azure hosted service is rolled out.

During the rollout of the upgrade you call [Upgrade Deployment](#) in manual mode and begin to walk upgrade domains. If at some point, as you monitor the upgrade, you note some role instances in the first upgrade domains that you examine have become unresponsive, you can call the [Rollback Update Or Upgrade](#) operation on the deployment, which will leave untouched the instances which had not yet been upgraded and rollback instances which had been upgraded to the previous service package and configuration.

Initiating multiple mutating operations on an ongoing deployment

In some cases you may want to initiate multiple simultaneous mutating operations on an ongoing deployment. For example, you may perform a service update and, while that update is being rolled out across your service, you want to make some change, e.g. to roll the update back, apply a different update, or even delete the deployment. A case in which this might be necessary is if a service upgrade contains buggy code which causes an upgraded role instance to repeatedly crash. In this case, the Azure Fabric Controller will not be able to make progress in applying that upgrade because an insufficient number of instances in the upgraded domain are healthy. This state is referred to as a *stuck deployment*. You can unstick the deployment by rolling back the update or applying a fresh update over top of the failing one.

Once the initial request to update or upgrade the service has been received by the Azure Fabric Controller, you can start subsequent mutating operations. That is, you do not have to wait for the initial operation to complete before you can start another mutating operation.

Initiating a second update operation while the first update is ongoing will perform similar to the rollback operation. If the second update is in automatic mode, the first upgrade domain will be upgraded immediately, possibly leading to instances from multiple upgrade domains being offline at the same point in time.

The mutating operations are as follows: [Change Deployment Configuration](#), [Upgrade Deployment](#), [Update Deployment Status](#), [Delete Deployment](#), and [Rollback Update Or Upgrade](#).

Two operations, [Get Deployment](#) and [Get Cloud Service Properties](#), return the Locked flag which can be examined to determine whether a mutating operation can be invoked on a given deployment.

In order to call the version of these methods which returns the Locked flag, you must set request header to "x-ms-version: 2011-10-01" or a later. For more information about versioning headers, see [Service Management Versioning](#).

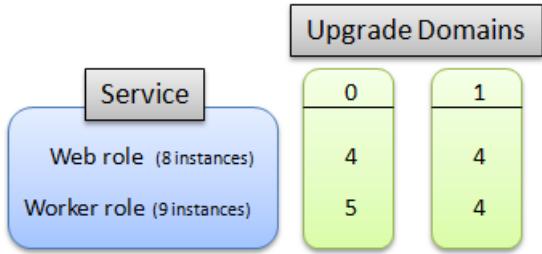
Distribution of roles across upgrade domains

Azure distributes instances of a role evenly across a set number of upgrade domains, which can be configured as part of the service definition (.csdef) file. The max number of upgrade domains is 20 and the default is 5. For more information about how to modify the service definition file, see [Azure Service Definition Schema \(.csdef File\)](#).

For example, if your role has ten instances, by default each upgrade domain contains two instances. If your role has 14 instances, then four of the upgrade domains contain three instances, and a fifth domain contains two.

Upgrade domains are identified with a zero-based index: the first upgrade domain has an ID of 0, and the second upgrade domain has an ID of 1, and so on.

The following diagram illustrates how a service that contains two roles are distributed when the service defines two upgrade domains. The service is running eight instances of the web role and nine instances of the worker role.



NOTE

Note that Azure controls how instances are allocated across upgrade domains. It's not possible to specify which instances are allocated to which domain.

Next steps

- [How to Manage Cloud Services](#)
- [How to Monitor Cloud Services](#)
- [How to Configure Cloud Services](#)

Create PHP web and worker roles

11/27/2018 • 8 minutes to read • [Edit Online](#)

Overview

This guide will show you how to create PHP web or worker roles in a Windows development environment, choose a specific version of PHP from the "built-in" versions available, change the PHP configuration, enable extensions, and finally, deploy to Azure. It also describes how to configure a web or worker role to use a PHP runtime (with custom configuration and extensions) that you provide.

Azure provides three compute models for running applications: Azure App Service, Azure Virtual Machines, and Azure Cloud Services. All three models support PHP. Cloud Services, which includes web and worker roles, provides *platform as a service (PaaS)*. Within a cloud service, a web role provides a dedicated Internet Information Services (IIS) web server to host front-end web applications. A worker role can run asynchronous, long-running or perpetual tasks independent of user interaction or input.

For more information about these options, see [Compute hosting options provided by Azure](#).

Download the Azure SDK for PHP

The [Azure SDK for PHP](#) consists of several components. This article will use two of them: Azure PowerShell and the Azure emulators. These two components can be installed via the Microsoft Web Platform Installer. For more information, see [How to install and configure Azure PowerShell](#).

Create a Cloud Services project

The first step in creating a PHP web or worker role is to create an Azure Service project. An Azure Service project serves as a logical container for web and worker roles, and it contains the project's [service definition \(.csdef\)](#) and [service configuration \(.cscfg\)](#) files.

To create a new Azure Service project, run Azure PowerShell as an administrator, and execute the following command:

```
PS C:\>New-AzureServiceProject myProject
```

This command will create a new directory (`myProject`) to which you can add web and worker roles.

Add PHP web or worker roles

To add a PHP web role to a project, run the following command from within the project's root directory:

```
PS C:\myProject> Add-AzurePHPWebRole roleName
```

For a worker role, use this command:

```
PS C:\myProject> Add-AzurePHPWorkerRole roleName
```

NOTE

The `roleName` parameter is optional. If it is omitted, the role name will be automatically generated. The first web role created will be `WebRole1`, the second will be `WebRole2`, and so on. The first worker role created will be `WorkerRole1`, the second will be `WorkerRole2`, and so on.

Specify the built-in PHP version

When you add a PHP web or worker role to a project, the project's configuration files are modified so that PHP will be installed on each web or worker instance of your application when it is deployed. To see the version of PHP that will be installed by default, run the following command:

```
PS C:\myProject> Get-AzureServiceProjectRoleRuntime
```

The output from the command above will look similar to what is shown below. In this example, the `IsDefault` flag is set to `true` for PHP 5.3.17, indicating that it will be the default PHP version installed.

Runtime Version	PackageUri	IsDefault
Node 0.6.17	http://nodertncu.blob.core...	False
Node 0.6.20	http://nodertncu.blob.core...	True
Node 0.8.4	http://nodertncu.blob.core...	False
IISNode 0.1.21	http://nodertncu.blob.core...	True
Cache 1.8.0	http://nodertncu.blob.core...	True
PHP 5.3.17	http://nodertncu.blob.core...	True
PHP 5.4.0	http://nodertncu.blob.core...	False

You can set the PHP runtime version to any of the PHP versions that are listed. For example, to set the PHP version (for a role with the name `roleName`) to 5.4.0, use the following command:

```
PS C:\myProject> Set-AzureServiceProjectRole roleName php 5.4.0
```

NOTE

Available PHP versions may change in the future.

Customize the built-in PHP runtime

You have complete control over the configuration of the PHP runtime that is installed when you follow the steps above, including modification of `php.ini` settings and enabling of extensions.

To customize the built-in PHP runtime, follow these steps:

1. Add a new folder, named `php`, to the `bin` directory of your web role. For a worker role, add it to the role's root directory.
2. In the `php` folder, create another folder called `ext`. Put any `.dll` extension files (e.g., `php_mongo.dll`) that you want to enable in this folder.
3. Add a `php.ini` file to the `php` folder. Enable any custom extensions and set any PHP directives in this file. For example, if you wanted to turn `display_errors` on and enable the `php_mongo.dll` extension, the contents of your `php.ini` file would be as follows:

```
display_errors=On  
extension=php_mongo.dll
```

NOTE

Any settings that you don't explicitly set in the `php.ini` file that you provide will automatically be set to their default values. However, keep in mind that you can add a complete `php.ini` file.

Use your own PHP runtime

In some cases, instead of selecting a built-in PHP runtime and configuring it as described above, you may want to provide your own PHP runtime. For example, you can use the same PHP runtime in a web or worker role that you use in your development environment. This makes it easier to ensure that the application will not change behavior in your production environment.

Configure a web role to use your own PHP runtime

To configure a web role to use a PHP runtime that you provide, follow these steps:

1. Create an Azure Service project and add a PHP web role as described previously in this topic.
2. Create a `php` folder in the `bin` folder that is in your web role's root directory, and then add your PHP runtime (all binaries, configuration files, subfolders, etc.) to the `php` folder.
3. (OPTIONAL) If your PHP runtime uses the [Microsoft Drivers for PHP for SQL Server](#), you will need to configure your web role to install [SQL Server Native Client 2012](#) when it is provisioned. To do this, add the [sqlncli.msi x64 installer](#) to the `bin` folder in your web role's root directory. The startup script described in the next step will silently run the installer when the role is provisioned. If your PHP runtime does not use the Microsoft Drivers for PHP for SQL Server, you can remove the following line from the script shown in the next step:

```
msiexec /i sqlncli.msi /qn IACCEPTSQLNLILICENSETERMS=YES
```

4. Define a startup task that configures [Internet Information Services \(IIS\)](#) to use your PHP runtime to handle requests for `.php` pages. To do this, open the `setup_web.cmd` file (in the `bin` file of your web role's root directory) in a text editor and replace its contents with the following script:

```

@ECHO ON
cd "%~dp0"

if "%EMULATED%"=="true" exit /b 0

msiexec /i sqlncli.msi /qn IACCEPTSQLNCLILICENSETERMS=YES

SET PHP_FULL_PATH=%~dp0php\php-cgi.exe
SET NEW_PATH=%PATH%;%RoleRoot%\base\x86

%WINDIR%\system32\inetsrv\appcmd.exe set config -section:system.webServer/fastCgi /+
[fullPath='%PHP_FULL_PATH%',maxInstances='12',idleTimeout='60000',activityTimeout='3600',requestTimeout=
'60000',instanceMaxRequests='10000',protocol='NamedPipe',flushNamedPipe='False']" /commit:apphost
%WINDIR%\system32\inetsrv\appcmd.exe set config -section:system.webServer/fastCgi /+
[fullPath='%PHP_FULL_PATH%'].environmentVariables.[name='PATH',value='%NEW_PATH%']" /commit:apphost
%WINDIR%\system32\inetsrv\appcmd.exe set config -section:system.webServer/fastCgi /+
[fullPath='%PHP_FULL_PATH%'].environmentVariables.[name='PHP_FCGI_MAX_REQUESTS',value='10000']"
/commit:apphost
%WINDIR%\system32\inetsrv\appcmd.exe set config -section:system.webServer/handlers /+
[name='PHP',path='*.php',verb='GET,HEAD,POST',modules='FastCgiModule',scriptProcessor='%PHP_FULL_PATH%',resourceType='Either',requireAccess='Script']" /commit:apphost
%WINDIR%\system32\inetsrv\appcmd.exe set config -section:system.webServer/fastCgi /+
[fullPath='%PHP_FULL_PATH%'].queueLength:50000"

```

5. Add your application files to your web role's root directory. This will be the web server's root directory.
6. Publish your application as described in the [Publish your application](#) section below.

NOTE

The `download.ps1` script (in the `bin` folder of the web role's root directory) can be deleted after you follow the steps described above for using your own PHP runtime.

Configure a worker role to use your own PHP runtime

To configure a worker role to use a PHP runtime that you provide, follow these steps:

1. Create an Azure Service project and add a PHP worker role as described previously in this topic.
2. Create a `php` folder in the worker role's root directory, and then add your PHP runtime (all binaries, configuration files, subfolders, etc.) to the `php` folder.
3. (OPTIONAL) If your PHP runtime uses [Microsoft Drivers for PHP for SQL Server](#), you will need to configure your worker role to install [SQL Server Native Client 2012](#) when it is provisioned. To do this, add the [sqIncli.msi x64 installer](#) to the worker role's root directory. The startup script described in the next step will silently run the installer when the role is provisioned. If your PHP runtime does not use the Microsoft Drivers for PHP for SQL Server, you can remove the following line from the script shown in the next step:

```
msiexec /i sqlncli.msi /qn IACCEPTSQLNCLILICENSETERMS=YES
```

4. Define a startup task that adds your `php.exe` executable to the worker role's PATH environment variable when the role is provisioned. To do this, open the `setup_worker.cmd` file (in the worker role's root directory) in a text editor and replace its contents with the following script:

```
@echo on

cd "%~dp0"

echo Granting permissions for Network Service to the web root directory...
icacls ..\ /grant "Network Service":(OI)(CI)W
if %ERRORLEVEL% neq 0 goto error
echo OK

if "%EMULATED%"=="true" exit /b 0

msiexec /i sqlncli.msi /qn IACCEPTSQLNCLILICENSETERMS=YES

setx Path "%PATH%;%~dp0php" /M

if %ERRORLEVEL% neq 0 goto error

echo SUCCESS
exit /b 0

:error

echo FAILED
exit /b -1
```

5. Add your application files to your worker role's root directory.
6. Publish your application as described in the [Publish your application](#) section below.

Run your application in the compute and storage emulators

The Azure emulators provide a local environment in which you can test your Azure application before you deploy it to the cloud. There are some differences between the emulators and the Azure environment. To understand this better, see [Use the Azure storage emulator for development and testing](#).

Note that you must have PHP installed locally to use the compute emulator. The compute emulator will use your local PHP installation to run your application.

To run your project in the emulators, execute the following command from your project's root directory:

```
PS C:\MyProject> Start-AzureEmulator
```

You will see output similar to this:

```
Creating local package...
Starting Emulator...
Role is running at http://127.0.0.1:81
Started
```

You can see your application running in the emulator by opening a web browser and browsing to the local address shown in the output (`http://127.0.0.1:81` in the example output above).

To stop the emulators, execute this command:

```
PS C:\MyProject> Stop-AzureEmulator
```

Publish your application

To publish your application, you need to first import your publish settings by using the [Import-AzurePublishSettingsFile](#) cmdlet. Then you can publish your application by using the [Publish-AzureServiceProject](#) cmdlet. For information about signing in, see [How to install and configure Azure PowerShell](#).

Next steps

For more information, see the [PHP Developer Center](#).

Build and deploy a Node.js application to an Azure Cloud Service

12/20/2018 • 5 minutes to read • [Edit Online](#)

This tutorial shows how to create a simple Node.js application running in an Azure Cloud Service. Cloud Services are the building blocks of scalable cloud applications in Azure. They allow the separation and independent management and scale-out of front-end and back-end components of your application. Cloud Services provide a robust dedicated virtual machine for hosting each role reliably.

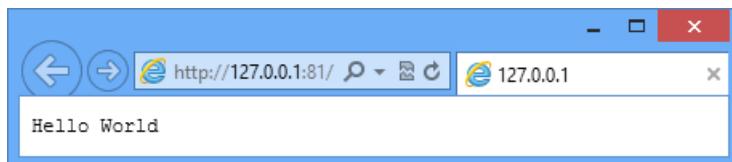
For more information on Cloud Services, and how they compare to Azure Websites and Virtual machines, see [Azure Websites, Cloud Services and Virtual Machines comparison](#).

TIP

Looking to build a simple website? If your scenario involves just a simple website front-end, consider [using a lightweight web app](#). You can easily upgrade to a Cloud Service as your web app grows and your requirements change.

By following this tutorial, you will build a simple web application hosted inside a web role. You will use the compute emulator to test your application locally, then deploy it using PowerShell command-line tools.

The application is a simple "hello world" application:



Prerequisites

NOTE

This tutorial uses Azure PowerShell, which requires Windows.

- Install and configure [Azure Powershell](#).
- Download and install the [Azure SDK for .NET 2.7](#). In the install setup, select:
 - MicrosoftAzureAuthoringTools
 - MicrosoftAzureComputeEmulator

Create an Azure Cloud Service project

Perform the following tasks to create a new Azure Cloud Service project, along with basic Node.js scaffolding:

1. Run **Windows PowerShell** as Administrator; from the **Start Menu** or **Start Screen**, search for **Windows PowerShell**.
2. [Connect PowerShell](#) to your subscription.
3. Enter the following PowerShell cmdlet to create to create the project:

```
New-AzureServiceProject helloworld
```

The screenshot shows a Windows Azure PowerShell window titled "Administrator: Windows Azure PowerShell". The command "new-azureserviceproject helloworld" is run, and the output indicates that a service has been created at "C:\node\helloworld". The window title bar also shows the command entered.

The **New-AzureServiceProject** cmdlet generates a basic structure for publishing a Node.js application to a Cloud Service. It contains configuration files necessary for publishing to Azure. The cmdlet also changes your working directory to the directory for the service.

The cmdlet creates the following files:

- **ServiceConfiguration.Cloud.cscfg, ServiceConfiguration.Local.cscfg** and **ServiceDefinition.csdef**: Azure-specific files necessary for publishing your application. For more information, see [Overview of Creating a Hosted Service for Azure](#).
- **deploymentSettings.json**: Stores local settings that are used by the Azure PowerShell deployment cmdlets.

4. Enter the following command to add a new web role:

```
Add-AzureNodeWebRole
```

The screenshot shows a Windows Azure PowerShell window titled "Administrator: Windows Azure PowerShell". The command "add-azurenodewebrole" is run, and the output indicates that a role has been created at "C:\node\helloworld\WebRole1". It also provides instructions to install the Windows Azure Client Library for Node.js. The window title bar shows the command entered.

The **Add-AzureNodeWebRole** cmdlet creates a basic Node.js application. It also modifies the **.cscfg** and **.csdef** files to add configuration entries for the new role.

NOTE

If you do not specify a role name, a default name is used. You can provide a name as the first cmdlet parameter:

```
Add-AzureNodeWebRole MyRole
```

The Node.js app is defined in the file **server.js**, located in the directory for the web role (**WebRole1** by default). Here is the code:

```
var http = require('http');
var port = process.env.port || 1337;
http.createServer(function (req, res) {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('Hello World\n');
}).listen(port);
```

This code is essentially the same as the "Hello World" sample on the [nodejs.org](#) website, except it uses the port number assigned by the cloud environment.

Deploy the application to Azure

NOTE

To complete this tutorial, you need an Azure account. You can [activate your MSDN subscriber benefits](#) or [sign up for a free account](#).

Download the Azure publishing settings

To deploy your application to Azure, you must first download the publishing settings for your Azure subscription.

1. Run the following Azure PowerShell cmdlet:

```
Get-AzurePublishSettingsFile
```

This will use your browser to navigate to the publish settings download page. You may be prompted to log in with a Microsoft Account. If so, use the account associated with your Azure subscription.

Save the downloaded profile to a file location you can easily access.

2. Run following cmdlet to import the publishing profile you downloaded:

```
Import-AzurePublishSettingsFile [path to file]
```

NOTE

After importing the publish settings, consider deleting the downloaded .publishSettings file, because it contains information that could allow someone to access your account.

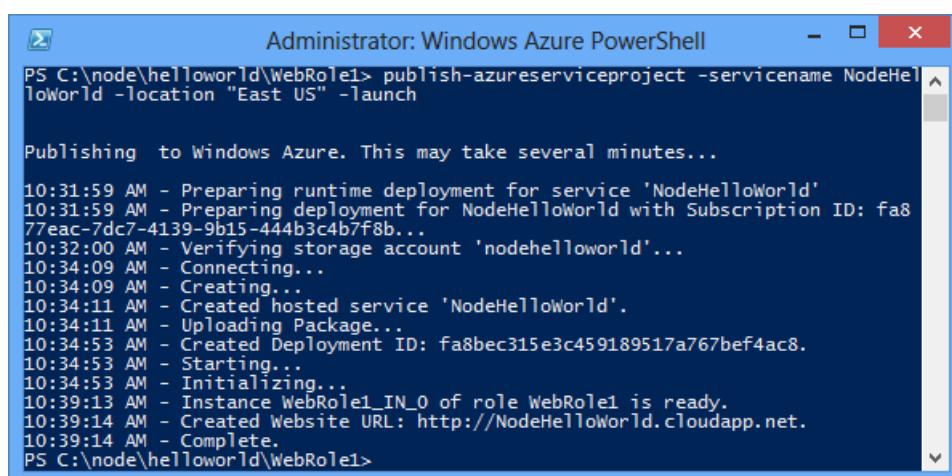
Publish the application

To publish, run the following commands:

```
$ServiceName = "NodeHelloWorld" + $(Get-Date -Format ('ddhhmm'))  
Publish-AzureServiceProject -ServiceName $ServiceName -Location "East US" -Launch
```

- **-ServiceName** specifies the name for the deployment. This must be a unique name, otherwise the publish process will fail. The **Get-Date** command tacks on a date/time string that should make the name unique.
- **-Location** specifies the datacenter that the application will be hosted in. To see a list of available datacenters, use the **Get-AzureLocation** cmdlet.
- **-Launch** opens a browser window and navigates to the hosted service after deployment has completed.

After publishing succeeds, you will see a response similar to the following:



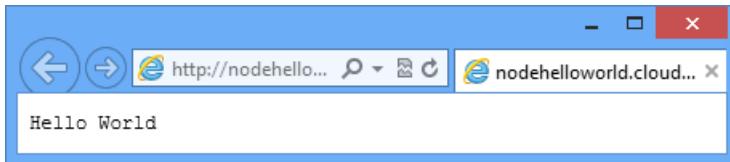
The screenshot shows a Windows Azure PowerShell window titled 'Administrator: Windows Azure PowerShell'. The command entered was 'publish-azureserviceproject -servicename NodeHelloWorld -location "East US" -launch'. The output shows the deployment process starting at 10:31:59 AM, including steps like preparing runtime deployment, preparing deployment for the service, verifying storage accounts, connecting, creating the hosted service, uploading the package, and finally starting and initializing the deployment. The URL 'http://NodeHelloWorld.cloudapp.net' is listed as created at 10:39:14 AM.

```
Administrator: Windows Azure PowerShell  
PS C:\node\helloworld\WebRole1> publish-azureserviceproject -servicename NodeHelloWorld -location "East US" -launch  
  
Publishing to Windows Azure. This may take several minutes...  
10:31:59 AM - Preparing runtime deployment for service 'NodeHelloWorld'  
10:31:59 AM - Preparing deployment for NodeHelloWorld with Subscription ID: fa877eac-7dc7-4139-9b15-444b3c4b7f8b...  
10:32:00 AM - Verifying storage account 'nodehelloworld'...  
10:34:09 AM - Connecting...  
10:34:09 AM - Creating...  
10:34:11 AM - Created hosted service 'NodeHelloWorld'.  
10:34:11 AM - Uploading Package...  
10:34:53 AM - Created Deployment ID: fa8bec315e3c459189517a767bef4ac8.  
10:34:53 AM - Starting...  
10:34:53 AM - Initializing...  
10:39:13 AM - Instance WebRole1_IN_0 of role WebRole1 is ready.  
10:39:14 AM - Created Website URL: http://NodeHelloWorld.cloudapp.net.  
10:39:14 AM - Complete.  
PS C:\node\helloworld\WebRole1>
```

NOTE

It can take several minutes for the application to deploy and become available when first published.

Once the deployment has completed, a browser window will open and navigate to the cloud service.



Your application is now running on Azure.

The **Publish-AzureServiceProject** cmdlet performs the following steps:

1. Creates a package to deploy. The package contains all the files in your application folder.
2. Creates a new **storage account** if one does not exist. The Azure storage account is used to store the application package during deployment. You can safely delete the storage account after deployment is done.
3. Creates a new **cloud service** if one does not already exist. A **cloud service** is the container in which your application is hosted when it is deployed to Azure. For more information, see [Overview of Creating a Hosted Service for Azure](#).
4. Publishes the deployment package to Azure.

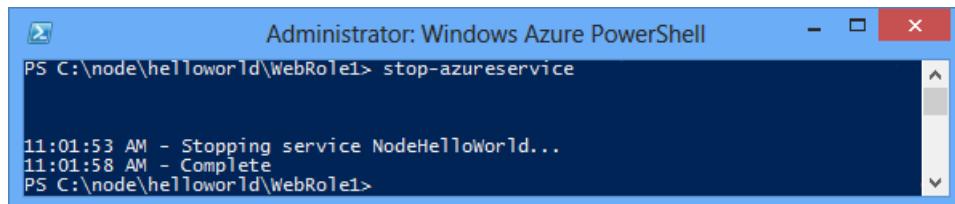
Stopping and deleting your application

After deploying your application, you may want to disable it so you can avoid extra costs. Azure bills web role instances per hour of server time consumed. Server time is consumed once your application is deployed, even if the instances are not running and are in the stopped state.

1. In the Windows PowerShell window, stop the service deployment created in the previous section with the following cmdlet:

```
Stop-AzureService
```

Stopping the service may take several minutes. When the service is stopped, you receive a message indicating that it has stopped.

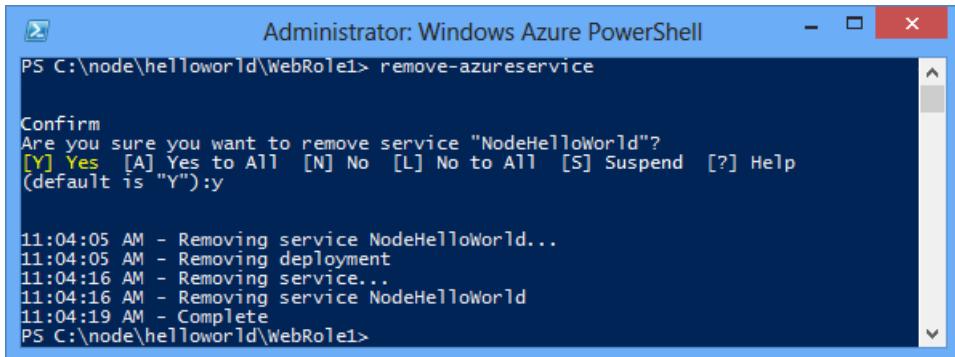


2. To delete the service, call the following cmdlet:

```
Remove-AzureService
```

When prompted, enter **Y** to delete the service.

Deleting the service may take several minutes. After the service has been deleted you receive a message indicating that the service was deleted.



A screenshot of the Windows Azure PowerShell window titled "Administrator: Windows Azure PowerShell". The command entered is "remove-azureservice". A confirmation dialog box is displayed, asking if the user wants to remove the service "NodeHelloWorld". The options are [Y] Yes, [A] Yes to All, [N] No, [L] No to All, [S] Suspend, and [?] Help. The default response is "y". The output shows the process of removing the service, deployment, and service again, concluding with a "Complete" message at 11:04:19 AM.

```
Administrator: Windows Azure PowerShell
PS C:\node\helloworld\WebRole1> remove-azureservice

Confirm
Are you sure you want to remove service "NodeHelloWorld"?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help
(default is "Y"):y

11:04:05 AM - Removing service NodeHelloWorld...
11:04:05 AM - Removing deployment
11:04:16 AM - Removing service...
11:04:16 AM - Removing service NodeHelloWorld
11:04:19 AM - Complete
PS C:\node\helloworld\WebRole1>
```

NOTE

Deleting the service does not delete the storage account that was created when the service was initially published, and you will continue to be billed for storage used. If nothing else is using the storage, you may want to delete it.

Next steps

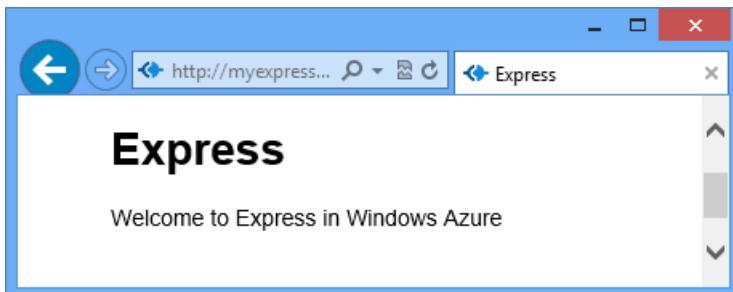
For more information, see the [Nodejs Developer Center](#).

Build and deploy a Node.js web application using Express on an Azure Cloud Services

9/13/2018 • 2 minutes to read • [Edit Online](#)

Node.js includes a minimal set of functionality in the core runtime. Developers often use 3rd party modules to provide additional functionality when developing a Node.js application. In this tutorial you will create a new application using the [Express](#) module, which provides an MVC framework for creating Node.js web applications.

A screenshot of the completed application is below:



Create a Cloud Service Project

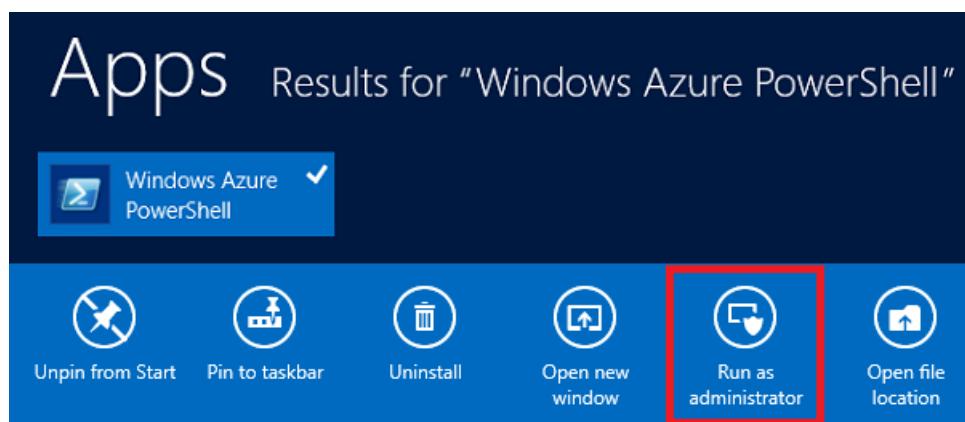
NOTE

To successfully complete this section, you must have a working installation of Node.js and the Azure SDK for Node.js for your platform.

- Install the Azure SDK for Node.js: [Windows installer](#) | [Mac installer](#) | [Linux download](#)
- If you are using Mac or Linux, install Node.js from <http://nodejs.org>. The Windows installer already includes Node.js.

Perform the following steps to create a new cloud service project named 'expressapp':

1. From the **Start Menu** or **Start Screen**, search for **Windows PowerShell**. Finally, right-click **Windows PowerShell** and select **Run As Administrator**.



2. Change directories to the **c:\node** directory and then enter the following commands to create a new solution named **expressapp** and a web role named **WebRole1**:

```
PS C:\node> New-AzureServiceProject expressapp
PS C:\Node\expressapp> Add-AzureNodeWebRole
PS C:\Node\expressapp> Set-AzureServiceProjectRole WebRole1 Node 0.10.21
```

NOTE

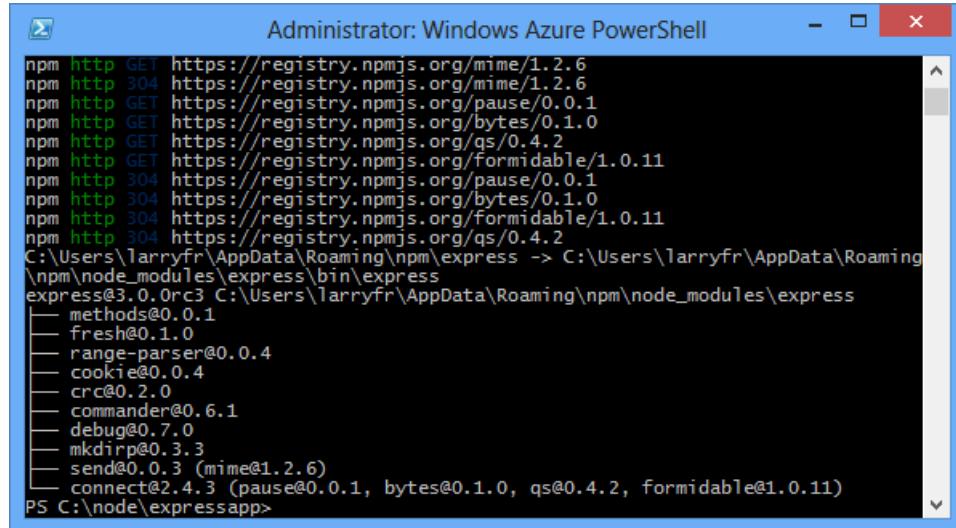
By default, **Add-AzureNodeWebRole** uses an older version of Node.js. The **Set-AzureServiceProjectRole** statement above instructs Azure to use v0.10.21 of Node. Note the parameters are case-sensitive. You can verify the correct version of Node.js has been selected by checking the **engines** property in **WebRole1\package.json**.

Install Express

1. Install the Express generator by issuing the following command:

```
PS C:\node\expressapp> npm install express-generator -g
```

The output of the npm command should look similar to the result below.

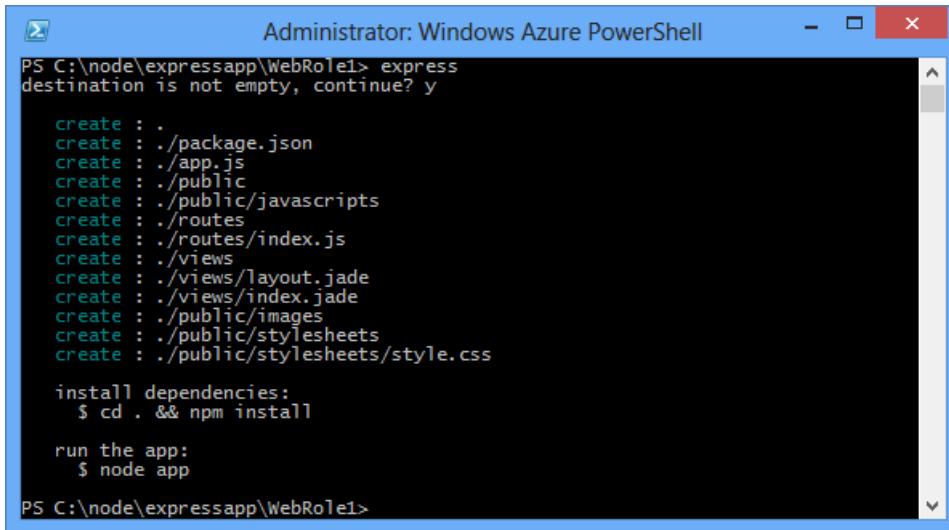


```
Administrator: Windows Azure PowerShell
npm http GET https://registry.npmjs.org/mime/1.2.6
npm http 304 https://registry.npmjs.org/mime/1.2.6
npm http GET https://registry.npmjs.org/pause/0.0.1
npm http GET https://registry.npmjs.org/bytes/0.1.0
npm http GET https://registry.npmjs.org/qs/0.4.2
npm http GET https://registry.npmjs.org/formidable/1.0.11
npm http 304 https://registry.npmjs.org/pause/0.0.1
npm http 304 https://registry.npmjs.org/bytes/0.1.0
npm http 304 https://registry.npmjs.org/formidable/1.0.11
npm http 304 https://registry.npmjs.org/qs/0.4.2
C:\Users\larryfr\AppData\Roaming\npm\express -> C:\Users\larryfr\AppData\Roaming\npm\node_modules\express\bin\express
express@3.0.0rc3 C:\Users\larryfr\AppData\Roaming\npm\node_modules\express
├── methods@0.0.1
└── fresh@0.1.0
  ├── range-parser@0.0.4
  ├── cookie@0.0.4
  ├── crc@0.2.0
  ├── commander@0.6.1
  ├── debug@0.7.0
  └── mkdirp@0.3.3
  └── send@0.0.3 (mime@1.2.6)
    └── connect@2.4.3 (pause@0.0.1, bytes@0.1.0, qs@0.4.2, formidable@1.0.11)
PS C:\node\expressapp>
```

2. Change directories to the **WebRole1** directory and use the express command to generate a new application:

```
PS C:\node\expressapp\WebRole1> express
```

You will be prompted to overwrite your earlier application. Enter **y** or **yes** to continue. Express will generate the app.js file and a folder structure for building your application.



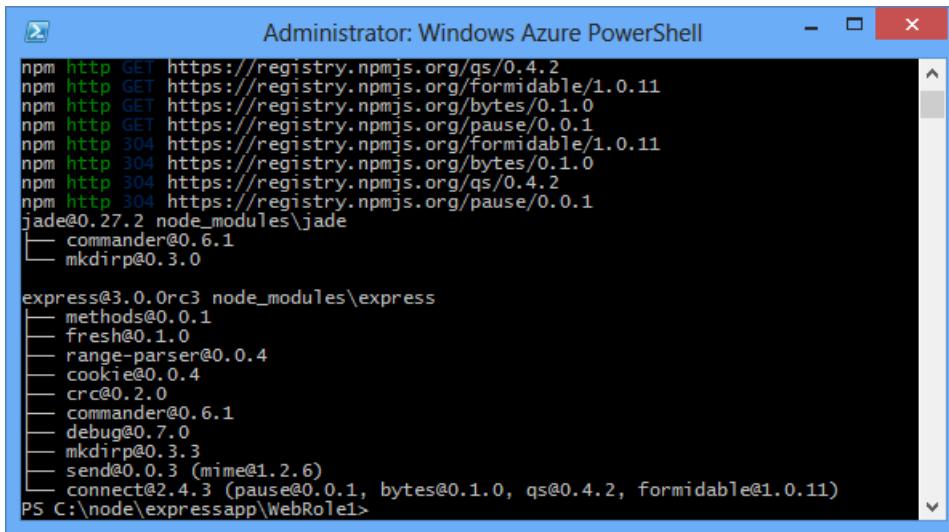
```
Administrator: Windows Azure PowerShell
PS C:\node\expressapp\WebRole1> express
destination is not empty, continue? y
  create : .
  create : ./package.json
  create : ./app.js
  create : ./public
  create : ./public/javascripts
  create : ./routes
  create : ./routes/index.js
  create : ./views
  create : ./views/layout.jade
  create : ./views/index.jade
  create : ./public/images
  create : ./public/stylesheets
  create : ./public/stylesheets/style.css

  install dependencies:
    $ cd . && npm install

  run the app:
    $ node app
PS C:\node\expressapp\WebRole1>
```

3. To install additional dependencies defined in the package.json file, enter the following command:

```
PS C:\node\expressapp\WebRole1> npm install
```



```
Administrator: Windows Azure PowerShell
npm http GET https://registry.npmjs.org/qs/0.4.2
npm http GET https://registry.npmjs.org/formidable/1.0.11
npm http GET https://registry.npmjs.org/bytes/0.1.0
npm http GET https://registry.npmjs.org/pause/0.0.1
npm http 304 https://registry.npmjs.org/formidable/1.0.11
npm http 304 https://registry.npmjs.org/bytes/0.1.0
npm http 304 https://registry.npmjs.org/qs/0.4.2
npm http 304 https://registry.npmjs.org/pause/0.0.1
jade@0.27.2 node_modules\jade
└── commander@0.6.1
    └── mkdirp@0.3.0

express@3.0.0rc3 node_modules\express
├── methods@0.0.1
├── fresh@0.1.0
├── range-parser@0.0.4
├── cookie@0.0.4
├── crc@0.2.0
└── commander@0.6.1
    ├── debug@0.7.0
    ├── mkdirp@0.3.3
    └── send@0.0.3 (mime@1.2.6)
    └── connect@2.4.3 (pause@0.0.1, bytes@0.1.0, qs@0.4.2, formidable@1.0.11)
PS C:\node\expressapp\WebRole1>
```

4. Use the following command to copy the **bin/www** file to **server.js**. This is so the cloud service can find the entry point for this application.

```
PS C:\node\expressapp\WebRole1> copy bin/www server.js
```

After this command completes, you should have a **server.js** file in the WebRole1 directory.

5. Modify the **server.js** to remove one of the '.' characters from the following line.

```
var app = require('../app');
```

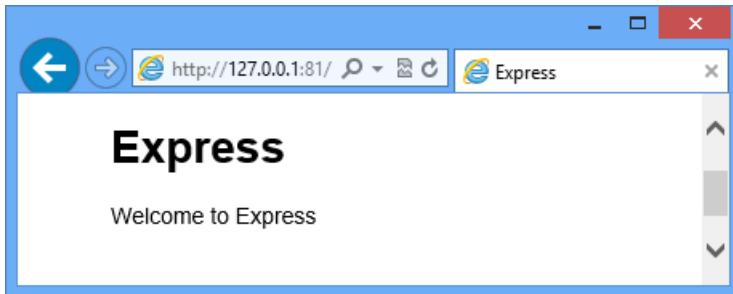
After making this modification, the line should appear as follows.

```
var app = require('./app');
```

This change is required since we moved the file (formerly **bin/www**) to the same directory as the app file being required. After making this change, save the **server.js** file.

6. Use the following command to run the application in the Azure emulator:

```
PS C:\node\expressapp\WebRole1> Start-AzureEmulator -launch
```

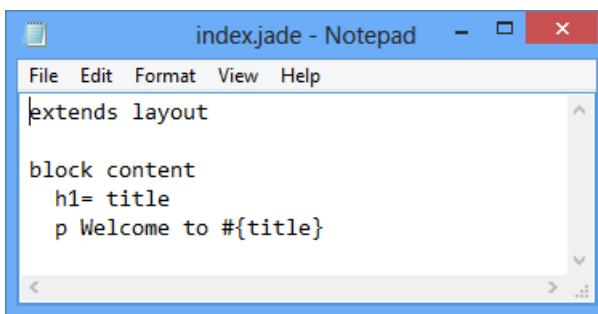


Modifying the View

Now modify the view to display the message "Welcome to Express in Azure".

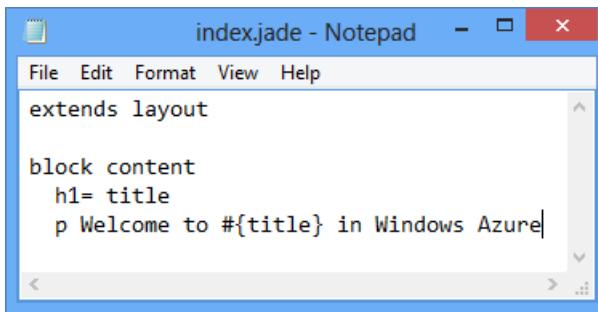
1. Enter the following command to open the index.jade file:

```
PS C:\node\expressapp\WebRole1> notepad views/index.jade
```



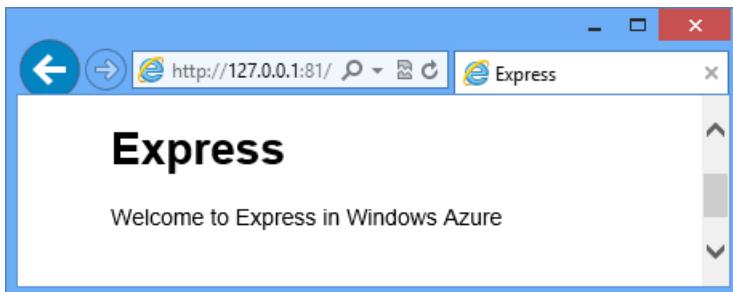
Jade is the default view engine used by Express applications. For more information on the Jade view engine, see <http://jade-lang.com>.

2. Modify the last line of text by appending **in Azure**.



3. Save the file and exit Notepad.

4. Refresh your browser and you will see your changes.



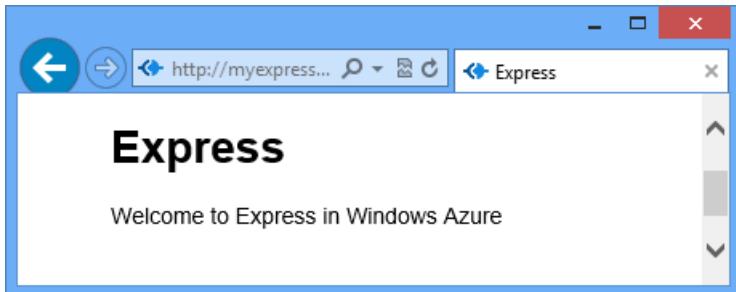
After testing the application, use the **Stop-AzureEmulator** cmdlet to stop the emulator.

Publishing the Application to Azure

In the Azure PowerShell window, use the **Publish-AzureServiceProject** cmdlet to deploy the application to a cloud service

```
PS C:\node\expressapp\WebRole1> Publish-AzureServiceProject -ServiceName myexpressapp -Location "East US" -Launch
```

Once the deployment operation completes, your browser will open and display the web page.



Next steps

For more information, see the [Node.js Developer Center](#).

Get started with Azure Blob Storage and Visual Studio connected services (cloud services projects)

11/7/2018 • 8 minutes to read • [Edit Online](#)

TIP

Manage Azure Blob storage resources with Azure Storage Explorer. [Azure Storage Explorer](#) is a free, standalone app from Microsoft that enables you to [manage Azure Blob storage resources](#). Using Azure Storage Explorer, you can visually create, read, update, and delete blob containers and blobs, as well as manage access to your blobs containers and blobs.

Overview

This article describes how to get started with Azure Blob Storage after you created or referenced an Azure Storage account by using the Visual Studio **Add Connected Services** dialog in a Visual Studio cloud services project. We'll show you how to access and create blob containers, and how to perform common tasks like uploading, listing, and downloading blobs. The samples are written in C# and use the [Microsoft Azure Storage Client Library for .NET](#).

Azure Blob Storage is a service for storing large amounts of unstructured data that can be accessed from anywhere in the world via HTTP or HTTPS. A single blob can be any size. Blobs can be things like images, audio and video files, raw data, and document files.

Just as files live in folders, storage blobs live in containers. After you have created a storage, you create one or more containers in the storage. For example, in a storage called "Scrapbook," you can create containers in the storage called "images" to store pictures and another called "audio" to store audio files. After you create the containers, you can upload individual blob files to them.

- For more information on programmatically manipulating blobs, see [Get started with Azure Blob storage using .NET](#).
- For general information about Azure Storage, see [Storage documentation](#).
- For general information about Azure Cloud Services, see [Cloud Services documentation](#).
- For more information about programming ASP.NET applications, see [ASP.NET](#).

Access blob containers in code

To programmatically access blobs in cloud service projects, you need to add the following items, if they're not already present.

1. Add the following code namespace declarations to the top of any C# file in which you wish to programmatically access Azure Storage.

```
using Microsoft.Framework.Configuration;
using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Blob;
using System.Threading.Tasks;
using LogLevel = Microsoft.Framework.Logging.LogLevel;
```

2. Get a **CloudStorageAccount** object that represents your storage account information. Use the following code to get the your storage connection string and storage account information from the Azure service

configuration.

```
CloudStorageAccount storageAccount = CloudStorageAccount.Parse(  
    CloudConfigurationManager.GetSetting("<storage account name>_AzureStorageConnectionString"));
```

3. Get a **CloudBlobClient** object to reference an existing container in your storage account.

```
// Create a blob client.  
CloudBlobClient blobClient = storageAccount.CreateCloudBlobClient();
```

4. Get a **CloudBlobContainer** object to reference a specific blob container.

```
// Get a reference to a container named "mycontainer."  
CloudBlobContainer container = blobClient.GetContainerReference("mycontainer");
```

NOTE

Use all of the code shown in the previous procedure in front of the code shown in the following sections.

Create a container in code

NOTE

Some APIs that perform calls out to Azure Storage in ASP.NET are asynchronous. See [Asynchronous programming with Async and Await](#) for more information. The code in the following example assumes that you are using async programming methods.

To create a container in your storage account, all you need to do is add a call to **CreateIfNotExistsAsync** as in the following code:

```
// If "mycontainer" doesn't exist, create it.  
await container.CreateIfNotExistsAsync();
```

To make the files within the container available to everyone, you can set the container to be public by using the following code.

```
await container.SetPermissionsAsync(new BlobContainerPermissions  
{  
    PublicAccess = BlobContainerPublicAccessType.Blob  
});
```

Anyone on the Internet can see blobs in a public container, but you can modify or delete them only if you have the appropriate access key.

Upload a blob into a container

Azure Storage supports block blobs and page blobs. In the majority of cases, block blob is the recommended type to use.

To upload a file to a block blob, get a container reference and use it to get a block blob reference. Once you have a blob reference, you can upload any stream of data to it by calling the **UploadFromStream** method. This operation creates the blob if it didn't previously exist, or overwrites it if it does exist. The following example shows how to

upload a blob into a container and assumes that the container was already created.

```
// Retrieve a reference to a blob named "myblob".
CloudBlockBlob blockBlob = container.GetBlockBlobReference("myblob");

// Create or overwrite the "myblob" blob with contents from a local file.
using (var fileStream = System.IO.File.OpenRead(@"path\myfile"))
{
    blockBlob.UploadFromStream(fileStream);
}
```

List the blobs in a container

To list the blobs in a container, first get a container reference. You can then use the container's **ListBlobs** method to retrieve the blobs and/or directories within it. To access the rich set of properties and methods for a returned **IListBlobItem**, you must cast it to a **CloudBlockBlob**, **CloudPageBlob**, or **CloudBlobDirectory** object. If the type is unknown, you can use a type check to determine which to cast it to. The following code demonstrates how to retrieve and output the URI of each item in the **photos** container:

```
// Loop over items within the container and output the length and URI.
foreach (IListBlobItem item in container.ListBlobs(null, false))
{
    if (item.GetType() == typeof(CloudBlockBlob))
    {
        CloudBlockBlob blob = (CloudBlockBlob)item;

        Console.WriteLine("Block blob of length {0}: {1}", blob.Properties.Length, blob.Uri);

    }
    else if (item.GetType() == typeof(CloudPageBlob))
    {
        CloudPageBlob pageBlob = (CloudPageBlob)item;

        Console.WriteLine("Page blob of length {0}: {1}", pageBlob.Properties.Length, pageBlob.Uri);

    }
    else if (item.GetType() == typeof(CloudBlobDirectory))
    {
        CloudBlobDirectory directory = (CloudBlobDirectory)item;

        Console.WriteLine("Directory: {0}", directory.Uri);
    }
}
```

As shown in the previous code sample, the blob service has the concept of directories within containers, as well. This is so that you can organize your blobs in a more folder-like structure. For example, consider the following set of block blobs in a container named **photos**:

```
photo1.jpg
2010/architecture/description.txt
2010/architecture/photo3.jpg
2010/architecture/photo4.jpg
2011/architecture/photo5.jpg
2011/architecture/photo6.jpg
2011/architecture/description.txt
2011/photo7.jpg
```

When you call **ListBlobs** on the container (as in the previous sample), the collection returned contains **CloudBlobDirectory** and **CloudBlockBlob** objects representing the directories and blobs contained at the top

level. Here is the resulting output:

```
Directory: https://<accountname>.blob.core.windows.net/photos/2010/
Directory: https://<accountname>.blob.core.windows.net/photos/2011/
Block blob of length 505623: https://<accountname>.blob.core.windows.net/photos/photo1.jpg
```

Optionally, you can set the **UseFlatBlobListing** parameter of the **ListBlobs** method to **true**. This results in every blob being returned as a **CloudBlockBlob**, regardless of directory. Here is the call to **ListBlobs**:

```
// Loop over items within the container and output the length and URI.
foreach (IListBlobItem item in container.ListBlobs(null, true))
{
    ...
}
```

and here are the results:

```
Block blob of length 4: https://<accountname>.blob.core.windows.net/photos/2010/architecture/description.txt
Block blob of length 314618: https://<accountname>.blob.core.windows.net/photos/2010/architecture/photo3.jpg
Block blob of length 522713: https://<accountname>.blob.core.windows.net/photos/2010/architecture/photo4.jpg
Block blob of length 4: https://<accountname>.blob.core.windows.net/photos/2011/architecture/description.txt
Block blob of length 419048: https://<accountname>.blob.core.windows.net/photos/2011/architecture/photo5.jpg
Block blob of length 506388: https://<accountname>.blob.core.windows.net/photos/2011/architecture/photo6.jpg
Block blob of length 399751: https://<accountname>.blob.core.windows.net/photos/2011/photo7.jpg
Block blob of length 505623: https://<accountname>.blob.core.windows.net/photos/photo1.jpg
```

For more information, see [CloudBlobContainer.ListBlobs](#).

Download blobs

To download blobs, first retrieve a blob reference and then call the **DownloadToStream** method. The following example uses the **DownloadToStream** method to transfer the blob contents to a stream object that you can then persist to a local file.

```
// Get a reference to a blob named "photo1.jpg".
CloudBlockBlob blockBlob = container.GetBlockBlobReference("photo1.jpg");

// Save blob contents to a file.
using (var fileStream = System.IO.File.OpenWrite(@"path\myfile"))
{
    blockBlob.DownloadToStream(fileStream);
}
```

You can also use the **DownloadToStream** method to download the contents of a blob as a text string.

```
// Get a reference to a blob named "myblob.txt"
CloudBlockBlob blockBlob2 = container.GetBlockBlobReference("myblob.txt");

string text;
using (var memoryStream = new MemoryStream())
{
    blockBlob2.DownloadToStream(memoryStream);
    text = System.Text.Encoding.UTF8.GetString(memoryStream.ToArray());
}
```

Delete blobs

To delete a blob, first get a blob reference and then call the **Delete** method.

```
// Get a reference to a blob named "myblob.txt".
CloudBlockBlob blockBlob = container.GetBlockBlobReference("myblob.txt");

// Delete the blob.
blockBlob.Delete();
```

List blobs in pages asynchronously

If you are listing a large number of blobs, or you want to control the number of results you return in one listing operation, you can list blobs in pages of results. This example shows how to return results in pages asynchronously, so that execution is not blocked while waiting to return a large set of results.

This example shows a flat blob listing, but you can also perform a hierarchical listing, by setting the **useFlatBlobListing** parameter of the **ListBlobsSegmentedAsync** method to **false**.

Because the sample method calls an asynchronous method, it must be prefaced with the **async** keyword, and it must return a **Task** object. The await keyword specified for the **ListBlobsSegmentedAsync** method suspends execution of the sample method until the listing task completes.

```
async public static Task ListBlobsSegmentedInFlatListing(CloudBlobContainer container)
{
    // List blobs to the console window, with paging.
    Console.WriteLine("List blobs in pages:");

    int i = 0;
    BlobContinuationToken continuationToken = null;
    BlobResultSegment resultSegment = null;

    // Call ListBlobsSegmentedAsync and enumerate the result segment returned, while the continuation token is
    // non-null.
    // When the continuation token is null, the last page has been returned and execution can exit the loop.
    do
    {
        // This overload allows control of the page size. You can return all remaining results by passing null
        // for the maxResults parameter,
        // or by calling a different overload.
        resultSegment = await container.ListBlobsSegmentedAsync("", true, BlobListingDetails.All, 10,
continuationToken, null, null);
        if (resultSegment.Results.Count<IListBlobItem>() > 0) { Console.WriteLine("Page {0}:", ++i); }
        foreach (var blobItem in resultSegment.Results)
        {
            Console.WriteLine("\t{0}", blobItem.StorageUri.PrimaryUri);
        }
        Console.WriteLine();

        //Get the continuation token.
        continuationToken = resultSegment.ContinuationToken;
    }
    while (continuationToken != null);
}
```

Next steps

Now that you've learned the basics of Azure Blob storage, follow these links to learn about more complex storage tasks.

- View the Blob service reference documentation in the [Azure Storage Client Library for .NET](#) reference for complete details about available APIs.

- To learn how to simplify the code you write to work with Azure Storage, check out [What is the Azure WebJobs SDK](#)
- View more feature guides to learn about additional options for storing data in Azure.
 - To work with Azure Storage Tables, see [Get Started with Azure Table storage using .NET](#).
 - To work with Azure Storage Queues, [Get started with Azure Queue storage using .NET](#).
 - To store relational data, see [Connect to SQL Database by using .NET \(C#\)](#).

Getting started with Azure Queue storage and Visual Studio connected services (cloud services projects)

8/17/2018 • 6 minutes to read • [Edit Online](#)

TIP

[Try the Microsoft Azure Storage Explorer](#)

[Microsoft Azure Storage Explorer](#) is a free, standalone app from Microsoft that enables you to work visually with Azure Storage data on Windows, macOS, and Linux.

Overview

This article describes how to get started using Azure Queue storage in Visual Studio after you have created or referenced an Azure storage account in a cloud services project by using the Visual Studio **Add Connected Services** dialog.

We'll show you how to create a queue in code. We'll also show you how to perform basic queue operations, such as adding, modifying, reading and removing queue messages. The samples are written in C# code and use the [Microsoft Azure Storage Client Library for .NET](#).

The **Add Connected Services** operation installs the appropriate NuGet packages to access Azure storage in your project and adds the connection string for the storage account to your project configuration files.

- See [Get started with Azure Queue storage using .NET](#) for more information on manipulating queues in code.
- See [Storage documentation](#) for general information about Azure Storage.
- See [Cloud Services documentation](#) for general information about Azure cloud services.
- See [ASP.NET](#) for more information about programming ASP.NET applications.

Azure Queue storage is a service for storing large numbers of messages that can be accessed from anywhere in the world via authenticated calls using HTTP or HTTPS. A single queue message can be up to 64 KB in size, and a queue can contain millions of messages, up to the total capacity limit of a storage account.

Access queues in code

To access queues in Visual Studio Cloud Services projects, you need to include the following items to any C# source file that access Azure Queue storage.

1. Make sure the namespace declarations at the top of the C# file include these **using** statements.

```
using Microsoft.Framework.Configuration;
using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Queue;
```

2. Get a **CloudStorageAccount** object that represents your storage account information. Use the following code to get the your storage connection string and storage account information from the Azure service configuration.

```
CloudStorageAccount storageAccount = CloudStorageAccount.Parse(  
    CloudConfigurationManager.GetSetting("<storage-account-name>_AzureStorageConnectionString"));
```

3. Get a **CloudQueueClient** object to reference the queue objects in your storage account.

```
// Create the queue client.  
CloudQueueClient queueClient = storageAccount.CreateCloudQueueClient();
```

4. Get a **CloudQueue** object to reference a specific queue.

```
// Get a reference to a queue named "messageQueue"  
CloudQueue messageQueue = queueClient.GetQueueReference("messageQueue");
```

NOTE: Use all of the above code in front of the code in the following samples.

Create a queue in code

To create the queue in code, just add a call to **CreateIfNotExists**.

```
// Create the CloudQueue if it does not exist  
messageQueue.CreateIfNotExists();
```

Add a message to a queue

To insert a message into an existing queue, create a new **CloudQueueMessage** object, then call the **AddMessage** method.

A **CloudQueueMessage** object can be created from either a string (in UTF-8 format) or a byte array.

Here is an example which inserts the message 'Hello, World'.

```
// Create a message and add it to the queue.  
CloudQueueMessage message = new CloudQueueMessage("Hello, World");  
messageQueue.AddMessage(message);
```

Read a message in a queue

You can peek at the message in the front of a queue without removing it from the queue by calling the **PeekMessage** method.

```
// Peek at the next message  
CloudQueueMessage peekedMessage = messageQueue.PeekMessage();
```

Read and remove a message in a queue

Your code can remove (de-queue) a message from a queue in two steps.

1. Call **GetMessage** to get the next message in a queue. A message returned from **GetMessage** becomes invisible to any other code reading messages from this queue. By default, this message stays invisible for 30 seconds.
2. To finish removing the message from the queue, call **DeleteMessage**.

This two-step process of removing a message assures that if your code fails to process a message due to hardware or software failure, another instance of your code can get the same message and try again. The following code calls **DeleteMessage** right after the message has been processed.

```
// Get the next message in the queue.  
CloudQueueMessage retrievedMessage = messageQueue.GetMessage();  
  
// Process the message in less than 30 seconds  
  
// Then delete the message.  
await messageQueue.DeleteMessage(retrievedMessage);
```

Use additional options to process and remove queue messages

There are two ways you can customize message retrieval from a queue.

- You can get a batch of messages (up to 32).
- You can set a longer or shorter invisibility timeout, allowing your code more or less time to fully process each message. The following code example uses the **GetMessages** method to get 20 messages in one call. Then it processes each message using a **foreach** loop. It also sets the invisibility timeout to five minutes for each message. Note that the 5 minutes starts for all messages at the same time, so after 5 minutes have passed since the call to **GetMessages**, any messages which have not been deleted will become visible again.

Here's an example:

```
foreach (CloudQueueMessage message in messageQueue.GetMessages(20, TimeSpan.FromMinutes(5)))  
{  
    // Process all messages in less than 5 minutes, deleting each message after processing.  
  
    // Then delete the message after processing  
    messageQueue.DeleteMessage(message);  
}
```

Get the queue length

You can get an estimate of the number of messages in a queue. The **FetchAttributes** method asks the Queue service to retrieve the queue attributes, including the message count. The **ApproximateMessageCount** property returns the last value retrieved by the **FetchAttributes** method, without calling the Queue service.

```
// Fetch the queue attributes.  
messageQueue.FetchAttributes();  
  
// Retrieve the cached approximate message count.  
int? cachedMessageCount = messageQueue.ApproximateMessageCount;  
  
// Display number of messages.  
Console.WriteLine("Number of messages in queue: " + cachedMessageCount);
```

Use the Async-Await Pattern with common Azure Queue APIs

This example shows how to use the Async-Await pattern with common Azure Queue APIs. The sample calls the async version of each of the given methods, this can be seen by the **Async** post-fix of each method. When an async method is used the async-await pattern suspends local execution until the call completes. This behavior allows the current thread to do other work which helps avoid performance bottlenecks and improves the overall

responsiveness of your application. For more details on using the Async-Await pattern in .NET see [Async and Await \(C# and Visual Basic\)](#)

```
// Create a message to put in the queue
CloudQueueMessage cloudQueueMessage = new CloudQueueMessage("My message");

// Add the message asynchronously
await messageQueue.AddMessageAsync(cloudQueueMessage);
Console.WriteLine("Message added");

// Async dequeue the message
CloudQueueMessage retrievedMessage = await messageQueue.GetMessageAsync();
Console.WriteLine("Retrieved message with content '{0}'", retrievedMessage.AsString);

// Delete the message asynchronously
await messageQueue.DeleteMessageAsync(retrievedMessage);
Console.WriteLine("Deleted message");
```

Delete a queue

To delete a queue and all the messages contained in it, call the **Delete** method on the queue object.

```
// Delete the queue.
messageQueue.Delete();
```

Next steps

Now that you've learned the basics of Azure queue storage, follow these links to learn about more complex storage tasks.

- View the Queue service reference documentation in the [Azure Storage Client Library for .NET](#) reference for complete details about available APIs.
- Learn more about using Queue storage at [Get started with Azure Queue storage using .NET](#)
- To learn how to simplify the code you write to work with Azure Storage, check out [What is the Azure WebJobs SDK](#)
- View more feature guides to learn about additional options for storing data in Azure.
 - To work with Azure Storage Tables, see [Get Started with Azure Table storage using .NET](#).
 - To work with Azure Storage Blobs, [Get started with Azure Blob storage using .NET](#).
 - To store relational data, see [Connect to SQL Database by using .NET \(C#\)](#).

Getting started with Azure table storage and Visual Studio connected services (cloud services projects)

11/7/2018 • 6 minutes to read • [Edit Online](#)

TIP

[Try the Microsoft Azure Storage Explorer](#)

[Microsoft Azure Storage Explorer](#) is a free, standalone app from Microsoft that enables you to work visually with Azure Storage data on Windows, macOS, and Linux.

Overview

This article describes how to get started using Azure table storage in Visual Studio after you have created or referenced an Azure storage account in a cloud services project by using the Visual Studio **Add Connected Services** dialog. The **Add Connected Services** operation installs the appropriate NuGet packages to access Azure storage in your project and adds the connection string for the storage account to your project configuration files.

The Azure Table storage service enables you to store large amounts of structured data. The service is a NoSQL datastore that accepts authenticated calls from inside and outside the Azure cloud. Azure tables are ideal for storing structured, non-relational data.

To get started, you first need to create a table in your storage account. We'll show you how to create an Azure table in code, and also how to perform basic table and entity operations, such as adding, modifying, reading and reading table entities. The samples are written in C# code and use the [Microsoft Azure Storage client library for .NET](#).

NOTE: Some of the APIs that perform calls out to Azure storage are asynchronous. See [Asynchronous programming with Async and Await](#) for more information. The code below assumes async programming methods are being used.

- See [Get started with Azure Table storage using .NET](#) for more information on programmatically manipulating tables.
- See [Storage documentation](#) for general information about Azure Storage.
- See [Cloud Services documentation](#) for general information about Azure cloud services.
- See [ASP.NET](#) for more information about programming ASP.NET applications.

Access tables in code

To access tables in cloud service projects, you need to include the following items to any C# source files that access Azure table storage.

1. Make sure the namespace declarations at the top of the C# file include these **using** statements.

```
using Microsoft.Framework.Configuration;
using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Table;
using System.Threading.Tasks;
using LogLevel = Microsoft.Framework.Logging.LogLevel;
```

- Get a **CloudStorageAccount** object that represents your storage account information. Use the following code to get the storage connection string and storage account information from the Azure service configuration.

```
CloudStorageAccount storageAccount = CloudStorageAccount.Parse(  
    CloudConfigurationManager.GetSetting("<storage account name>  
    _AzureStorageConnectionString"));
```

NOTE

Use all of the above code in front of the code in the following samples.

- Get a **CloudTableClient** object to reference the table objects in your storage account.

```
// Create the table client.  
CloudTableClient tableClient = storageAccount.CreateCloudTableClient();
```

- Get a **CloudTable** reference object to reference a specific table and entities.

```
// Get a reference to a table named "peopleTable".  
CloudTable peopleTable = tableClient.GetTableReference("peopleTable");
```

Create a table in code

To create the Azure table, just add a call to **CreateIfNotExistsAsync** to the after you get a **CloudTable** object as described in the "Access tables in code" section.

```
// Create the CloudTable if it does not exist.  
await peopleTable.CreateIfNotExistsAsync();
```

Add an entity to a table

To add an entity to a table, create a class that defines the properties of your entity. The following code defines an entity class called **CustomerEntity** that uses the customer's first name as the row key and the last name as the partition key.

```
public class CustomerEntity : TableEntity  
{  
    public CustomerEntity(string lastName, string firstName)  
    {  
        this.PartitionKey = lastName;  
        this.RowKey = firstName;  
    }  
  
    public CustomerEntity() {}  
  
    public string Email { get; set; }  
  
    public string PhoneNumber { get; set; }  
}
```

Table operations involving entities are done using the **CloudTable** object that you created earlier in "Access tables in code." The **TableOperation** object represents the operation to be done. The following code example shows how

to create a **CloudTable** object and a **CustomerEntity** object. To prepare the operation, a **TableOperation** is created to insert the customer entity into the table. Finally, the operation is executed by calling **CloudTable.ExecuteAsync**.

```
// Create a new customer entity.  
CustomerEntity customer1 = new CustomerEntity("Harp", "Walter");  
customer1.Email = "Walter@contoso.com";  
customer1.PhoneNumber = "425-555-0101";  
  
// Create the TableOperation that inserts the customer entity.  
TableOperation insertOperation = TableOperation.Insert(customer1);  
  
// Execute the insert operation.  
await peopleTable.ExecuteAsync(insertOperation);
```

Insert a batch of entities

You can insert multiple entities into a table in a single write operation. The following code example creates two entity objects ("Jeff Smith" and "Ben Smith"), adds them to a **TableBatchOperation** object using the Insert method, and then starts the operation by calling **CloudTable.ExecuteBatchAsync**.

```
// Create the batch operation.  
TableBatchOperation batchOperation = new TableBatchOperation();  
  
// Create a customer entity and add it to the table.  
CustomerEntity customer1 = new CustomerEntity("Smith", "Jeff");  
customer1.Email = "Jeff@contoso.com";  
customer1.PhoneNumber = "425-555-0104";  
  
// Create another customer entity and add it to the table.  
CustomerEntity customer2 = new CustomerEntity("Smith", "Ben");  
customer2.Email = "Ben@contoso.com";  
customer2.PhoneNumber = "425-555-0102";  
  
// Add both customer entities to the batch insert operation.  
batchOperation.Insert(customer1);  
batchOperation.Insert(customer2);  
  
// Execute the batch operation.  
await peopleTable.ExecuteBatchAsync(batchOperation);
```

Get all of the entities in a partition

To query a table for all of the entities in a partition, use a **TableQuery** object. The following code example specifies a filter for entities where 'Smith' is the partition key. This example prints the fields of each entity in the query results to the console.

```

// Construct the query operation for all customer entities where PartitionKey="Smith".
TableQuery<CustomerEntity> query = new TableQuery<CustomerEntity>()
    .Where(TableQuery.GenerateFilterCondition("PartitionKey", QueryComparisons.Equal, "Smith"));

// Print the fields for each customer.
TableContinuationToken token = null;
do
{
    TableQuerySegment<CustomerEntity> resultSegment = await peopleTable.ExecuteQuerySegmentedAsync(query,
token);
    token = resultSegment.ContinuationToken;

    foreach (CustomerEntity entity in resultSegment.Results)
    {
        Console.WriteLine("{0}, {1}\t{2}\t{3}", entity.PartitionKey, entity.RowKey,
entity.Email, entity.PhoneNumber);
    }
} while (token != null);

return View();

```

Get a single entity

You can write a query to get a single, specific entity. The following code uses a **TableOperation** object to specify a customer named 'Ben Smith'. This method returns just one entity, rather than a collection, and the returned value in **TableResult.Result** is a **CustomerEntity** object. Specifying both partition and row keys in a query is the fastest way to retrieve a single entity from the **Table** service.

```

// Create a retrieve operation that takes a customer entity.
TableOperation retrieveOperation = TableOperation.Retrieve<CustomerEntity>("Smith", "Ben");

// Execute the retrieve operation.
TableResult retrievedResult = await peopleTable.ExecuteAsync(retrieveOperation);

// Print the phone number of the result.
if (retrievedResult.Result != null)
    Console.WriteLine(((CustomerEntity)retrievedResult.Result).PhoneNumber);
else
    Console.WriteLine("The phone number could not be retrieved.");

```

Delete an entity

You can delete an entity after you find it. The following code looks for a customer entity named "Ben Smith", and if it finds it, it deletes it.

```

// Create a retrieve operation that expects a customer entity.
TableOperation retrieveOperation = TableOperation.Retrieve<CustomerEntity>("Smith", "Ben");

// Execute the operation.
TableResult retrievedResult = peopleTable.Execute(retrieveOperation);

// Assign the result to a CustomerEntity object.
CustomerEntity deleteEntity = (CustomerEntity)retrievedResult.Result;

// Create the Delete TableOperation and then execute it.
if (deleteEntity != null)
{
    TableOperation deleteOperation = TableOperation.Delete(deleteEntity);

    // Execute the operation.
    await peopleTable.ExecuteAsync(deleteOperation);

    Console.WriteLine("Entity deleted.");
}

else
    Console.WriteLine("Couldn't delete the entity.");

```

Next steps

Now that you've learned the basics of Azure Table storage, follow these links to learn about more complex storage tasks.

- View the Table Service reference documentation in the [Azure Storage Client Library for .NET](#) reference for complete details about available APIs.
- To learn how to simplify the code you write to work with Azure Storage, check out [What is the Azure WebJobs SDK](#)
- View more feature guides to learn about additional options for storing data in Azure.
 - To work with Azure Storage Blobs, see [Get Started with Azure Blob storage using .NET](#).
 - To work with Azure Storage Queues, [Get started with Azure Queue storage using .NET](#).
 - To store relational data, see [Connect to SQL Database by using .NET \(C#\)](#).

Enable communication for role instances in azure

7/12/2018 • 7 minutes to read • [Edit Online](#)

Cloud service roles communicate through internal and external connections. External connections are called **input endpoints** while internal connections are called **internal endpoints**. This topic describes how to modify the [service definition](#) to create endpoints.

Input endpoint

The input endpoint is used when you want to expose a port to the outside. You specify the protocol type and the port of the endpoint which then applies for both the external and internal ports for the endpoint. If you want, you can specify a different internal port for the endpoint with the [localPort](#) attribute.

The input endpoint can use the following protocols: **http, https, tcp, udp**.

To create an input endpoint, add the **InputEndpoint** child element to the **Endpoints** element of either a web or worker role.

```
<Endpoints>
  <InputEndpoint name="StandardWeb" protocol="http" port="80" localPort="80" />
</Endpoints>
```

Instance input endpoint

Instance input endpoints are similar to input endpoints but allows you map specific public-facing ports for each individual role instance by using port forwarding on the load balancer. You can specify a single public-facing port, or a range of ports.

The instance input endpoint can only use **tcp** or **udp** as the protocol.

To create an instance input endpoint, add the **InstanceInputEndpoint** child element to the **Endpoints** element of either a web or worker role.

```
<Endpoints>
  <InstanceInputEndpoint name="Endpoint2" protocol="tcp" localPort="10100">
    <AllocatePublicPortFrom>
      <FixedPortRange max="10109" min="10105" />
    </AllocatePublicPortFrom>
  </InstanceInputEndpoint>
</Endpoints>
```

Internal endpoint

Internal endpoints are available for instance-to-instance communication. The port is optional and if omitted, a dynamic port is assigned to the endpoint. A port range can be used. There is a limit of five internal endpoints per role.

The internal endpoint can use the following protocols: **http, tcp, udp, any**.

To create an internal input endpoint, add the **InternalEndpoint** child element to the **Endpoints** element of either a web or worker role.

```
<Endpoints>
  <InternalEndpoint name="Endpoint3" protocol="any" port="8999" />
</Endpoints>
```

You can also use a port range.

```
<Endpoints>
  <InternalEndpoint name="Endpoint3" protocol="any">
    <FixedPortRange max="8999" min="8995" />
  </InternalEndpoint>
</Endpoints>
```

Worker roles vs. Web roles

There is one minor difference with endpoints when working with both worker and web roles. The web role must have at minimum a single input endpoint using the **HTTP** protocol.

```
<Endpoints>
  <InputEndpoint name="StandardWeb" protocol="http" port="80" localPort="80" />
  <!-- more endpoints may be declared after the first InputEndPoint -->
</Endpoints>
```

Using the .NET SDK to access an endpoint

The Azure Managed Library provides methods for role instances to communicate at runtime. From code running within a role instance, you can retrieve information about the existence of other role instances and their endpoints, as well as information about the current role instance.

NOTE

You can only retrieve information about role instances that are running in your cloud service and that define at least one internal endpoint. You cannot obtain data about role instances running in a different service.

You can use the [Instances](#) property to retrieve instances of a role. First use the [CurrentRoleInstance](#) to return a reference to the current role instance, and then use the [Role](#) property to return a reference to the role itself.

When you connect to a role instance programmatically through the .NET SDK, it's relatively easy to access the endpoint information. For example, after you've already connected to a specific role environment, you can get the port of a specific endpoint with this code:

```
int port = RoleEnvironment.CurrentRoleInstance.InstanceEndpoints["StandardWeb"].IPEndpoint.Port;
```

The [Instances](#) property returns a collection of [RoleInstance](#) objects. This collection always contains the current instance. If the role does not define an internal endpoint, the collection includes the current instance but no other instances. The number of role instances in the collection will always be 1 in the case where no internal endpoint is defined for the role. If the role defines an internal endpoint, its instances are discoverable at runtime, and the number of instances in the collection will correspond to the number of instances specified for the role in the service configuration file.

NOTE

The Azure Managed Library does not provide a means of determining the health of other role instances, but you can implement such health assessments yourself if your service needs this functionality. You can use [Azure Diagnostics](#) to obtain information about running role instances.

To determine the port number for an internal endpoint on a role instance, you can use the **InstanceEndpoints** property to return a Dictionary object that contains endpoint names and their corresponding IP addresses and ports. The **IPEndpoint** property returns the IP address and port for a specified endpoint. The **PublicIPEndpoint** property returns the port for a load balanced endpoint. The IP address portion of the **PublicIPEndpoint** property is not used.

Here is an example that iterates role instances.

```
foreach (RoleInstance roleInst in RoleEnvironment.CurrentRoleInstance.Role.Instances)
{
    Trace.WriteLine("Instance ID: " + roleInst.Id);
    foreach (RoleInstanceEndpoint roleInstEndpoint in roleInst.InstanceEndpoints.Values)
    {
        Trace.WriteLine("Instance endpoint IP address and port: " + roleInstEndpoint.IPEndpoint);
    }
}
```

Here is an example of a worker role that gets the endpoint exposed through the service definition and starts listening for connections.

WARNING

This code will only work for a deployed service. When running in the Azure Compute Emulator, service configuration elements that create direct port endpoints (**InstanceInputEndpoint** elements) are ignored.

```
using System;
using System.Diagnostics;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.Diagnostics;
using Microsoft.WindowsAzure.ServiceRuntime;
using Microsoft.WindowsAzure.StorageClient;

namespace WorkerRole1
{
    public class WorkerRole : RoleEntryPoint
    {
        public override void Run()
        {
            try
            {
                // Initialize method-wide variables
                var epName = "Endpoint1";
                var roleInstance = RoleEnvironment.CurrentRoleInstance;

                // Identify direct communication port
                var myPublicEp = roleInstance.InstanceEndpoints[epName].PublicIPEndpoint;
                Trace.TraceInformation("IP:{0}, Port:{1}", myPublicEp.Address, myPublicEp.Port);

                // Identify public endpoint
                var myInternalEp = roleInstance.InstanceEndpoints[epName].IPEndpoint;
```

```

var myInternalEp = roleInstance.InstanceEndpoints[ephName].InternalEndpoints;

// Create socket listener
var listener = new Socket(
    myInternalEp.AddressFamily, SocketType.Stream, ProtocolType.Tcp);

// Bind socket listener to internal endpoint and listen
listener.Bind(myInternalEp);
listener.Listen(10);
Trace.TraceInformation("Listening on IP:{0},Port: {1}",
    myInternalEp.Address, myInternalEp.Port);

while (true)
{
    // Block the thread and wait for a client request
    Socket handler = listener.Accept();
    Trace.TraceInformation("Client request received.");

    // Define body of socket handler
    var handlerThread = new Thread(
        new ParameterizedThreadStart(h =>
    {
        var socket = h as Socket;
        Trace.TraceInformation("Local:{0} Remote{1}",
            socket.LocalEndPoint, socket.RemoteEndPoint);

        // Shut down and close socket
        socket.Shutdown(SocketShutdown.Both);
        socket.Close();
    })
    );

    // Start socket handler on new thread
    handlerThread.Start(handler);
}
}

catch (Exception e)
{
    Trace.TraceError("Caught exception in run. Details: {0}", e);
}
}

public override bool OnStart()
{
    // Set the maximum number of concurrent connections
    ServicePointManager.DefaultConnectionLimit = 12;

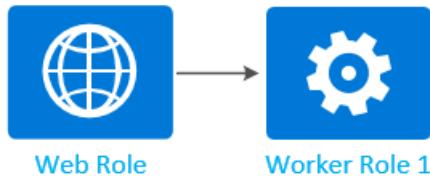
    // For information on handling configuration changes
    // see the MSDN topic at http://go.microsoft.com/fwlink/?LinkId=166357.
    return base.OnStart();
}
}
}
}

```

Network traffic rules to control role communication

After you define internal endpoints, you can add network traffic rules (based on the endpoints that you created) to control how role instances can communicate with each other. The following diagram shows some common scenarios for controlling role communication:

Scenario 1



Scenario 2



Scenario 3



Scenario 4



The following code example shows role definitions for the roles shown in the previous diagram. Each role definition includes at least one internal endpoint defined:

```
<ServiceDefinition name="MyService"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
<WebRole name="WebRole1" vmsize="Medium">
<Sites>
<Site name="Web">
<Bindings>
<Binding name="HttpIn" endpointName="HttpIn" />
</Bindings>
</Site>
</Sites>
<Endpoints>
<InputEndpoint name="HttpIn" protocol="http" port="80" />
<InternalEndpoint name="InternalTCP1" protocol="tcp" />
</Endpoints>
</WebRole>
<WorkerRole name="WorkerRole1">
<Endpoints>
<InternalEndpoint name="InternalTCP2" protocol="tcp" />
</Endpoints>
</WorkerRole>
<WorkerRole name="WorkerRole2">
<Endpoints>
<InternalEndpoint name="InternalTCP3" protocol="tcp" />
<InternalEndpoint name="InternalTCP4" protocol="tcp" />
</Endpoints>
</WorkerRole>
</ServiceDefinition>
```

NOTE

Restriction of communication between roles can occur with internal endpoints of both fixed and automatically assigned ports.

By default, after an internal endpoint is defined, communication can flow from any role to the internal endpoint of a role without any restrictions. To restrict communication, you must add a **NetworkTrafficRules** element to the **ServiceDefinition** element in the service definition file.

Scenario 1

Only allow network traffic from **WebRole1** to **WorkerRole1**.

```
<ServiceDefinition name="MyService"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
<NetworkTrafficRules>
<OnlyAllowTrafficTo>
<Destinations>
<RoleEndpoint endpointName="InternalTCP2" roleName="WorkerRole1"/>
</Destinations>
<AllowAllTraffic/>
<WhenSource matches="AnyRule">
<FromRole roleName="WebRole1"/>
</WhenSource>
</OnlyAllowTrafficTo>
</NetworkTrafficRules>
</ServiceDefinition>
```

Scenario 2

Only allows network traffic from **WebRole1** to **WorkerRole1** and **WorkerRole2**.

```
<ServiceDefinition name="MyService"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
<NetworkTrafficRules>
<OnlyAllowTrafficTo>
<Destinations>
<RoleEndpoint endpointName="InternalTCP2" roleName="WorkerRole1"/>
<RoleEndpoint endpointName="InternalTCP3" roleName="WorkerRole2"/>
</Destinations>
<WhenSource matches="AnyRule">
<FromRole roleName="WebRole1"/>
</WhenSource>
</OnlyAllowTrafficTo>
</NetworkTrafficRules>
</ServiceDefinition>
```

Scenario 3

Only allows network traffic from **WebRole1** to **WorkerRole1**, and **WorkerRole1** to **WorkerRole2**.

```
<ServiceDefinition name="MyService"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
<NetworkTrafficRules>
<OnlyAllowTrafficTo>
<Destinations>
<RoleEndpoint endpointName="InternalTCP2" roleName="WorkerRole1"/>
</Destinations>
<AllowAllTraffic/>
<WhenSource matches="AnyRule">
<FromRole roleName="WebRole1"/>
</WhenSource>
</OnlyAllowTrafficTo>
</NetworkTrafficRules>
<NetworkTrafficRules>
<OnlyAllowTrafficTo>
<Destinations>
<RoleEndpoint endpointName="InternalTCP3" roleName="WorkerRole2"/>
</Destinations>
<WhenSource matches="AnyRule">
<FromRole roleName="WorkerRole1"/>
</WhenSource>
</OnlyAllowTrafficTo>
</NetworkTrafficRules>
</ServiceDefinition>
```

Scenario 4

Only allows network traffic from **WebRole1** to **WorkerRole1**, **WebRole1** to **WorkerRole2**, and **WorkerRole1** to **WorkerRole2**.

```
<ServiceDefinition name="MyService"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
<NetworkTrafficRules>
<OnlyAllowTrafficTo>
<Destinations>
<RoleEndpoint endpointName="InternalTCP2" roleName="WorkerRole1"/>
</Destinations>
<AllowAllTraffic/>
<WhenSource matches="AnyRule">
<FromRole roleName="WebRole1"/>
</WhenSource>
</OnlyAllowTrafficTo>
</NetworkTrafficRules>
<NetworkTrafficRules>
<OnlyAllowTrafficTo >
<Destinations>
<RoleEndpoint endpointName="InternalTCP3" roleName="WorkerRole2"/>
</Destinations>
<AllowAllTraffic/>
<WhenSource matches="AnyRule">
<FromRole roleName="WorkerRole1"/>
</WhenSource>
</OnlyAllowTrafficTo>
</NetworkTrafficRules>
<NetworkTrafficRules>
<OnlyAllowTrafficTo >
<Destinations>
<RoleEndpoint endpointName="InternalTCP4" roleName="WorkerRole2"/>
</Destinations>
<AllowAllTraffic/>
<WhenSource matches="AnyRule">
<FromRole roleName="WebRole1"/>
</WhenSource>
</OnlyAllowTrafficTo>
</NetworkTrafficRules>
</ServiceDefinition>
```

An XML schema reference for the elements used above can be found [here](#).

Next steps

Read more about the Cloud Service [model](#).

Customize the Lifecycle of a Web or Worker role in .NET

7/12/2018 • 3 minutes to read • [Edit Online](#)

When you create a worker role, you extend the [RoleEntryPoint](#) class which provides methods for you to override that let you respond to lifecycle events. For web roles this class is optional, so you must use it to respond to lifecycle events.

Extend the RoleEntryPoint class

The [RoleEntryPoint](#) class includes methods that are called by Azure when it **starts**, **runs**, or **stops** a web or worker role. You can optionally override these methods to manage role initialization, role shutdown sequences, or the execution thread of the role.

When extending **RoleEntryPoint**, you should be aware of the following behaviors of the methods:

- The [OnStart](#) and [OnStop](#) methods return a boolean value, so it is possible to return **false** from these methods.

If your code returns **false**, the role process is abruptly terminated, without running any shutdown sequence you may have in place. In general, you should avoid returning **false** from the [OnStart](#) method.

- Any uncaught exception within an overload of a **RoleEntryPoint** method is treated as an unhandled exception.

If an exception occurs within one of the lifecycle methods, Azure will raise the [UnhandledException](#) event, and then the process is terminated. After your role has been taken offline, it will be restarted by Azure.

When an unhandled exception occurs, the [Stopping](#) event is not raised, and the [OnStop](#) method is not called.

If your role does not start, or is recycling between the initializing, busy, and stopping states, your code may be throwing an unhandled exception within one of the lifecycle events each time the role restarts. In this case, use the [UnhandledException](#) event to determine the cause of the exception and handle it appropriately. Your role may also be returning from the [Run](#) method, which causes the role to restart. For more information about deployment states, see [Common Issues Which Cause Roles to Recycle](#).

NOTE

If you are using the [Azure Tools for Microsoft Visual Studio](#) to develop your application, the role project templates automatically extend the **RoleEntryPoint** class for you, in the *WebRole.cs* and *WorkerRole.cs* files.

OnStart method

The **OnStart** method is called when your role instance is brought online by Azure. While the OnStart code is executing, the role instance is marked as **Busy** and no external traffic will be directed to it by the load balancer. You can override this method to perform initialization work, such as implementing event handlers and starting [Azure Diagnostics](#).

If **OnStart** returns **true**, the instance is successfully initialized and Azure calls the **RoleEntryPoint.Run** method. If **OnStart** returns **false**, the role terminates immediately, without executing any planned shutdown sequences.

The following code example shows how to override the **OnStart** method. This method configures and starts a diagnostic monitor when the role instance starts and sets up a transfer of logging data to a storage account:

```
public override bool OnStart()
{
    var config = DiagnosticMonitor.GetDefaultInitialConfiguration();

    config.DiagnosticInfrastructureLogs.ScheduledTransferLogLevelFilter = LogLevel.Error;
    config.DiagnosticInfrastructureLogs.ScheduledTransferPeriod = TimeSpan.FromMinutes(5);

    DiagnosticMonitor.Start("DiagnosticsConnectionString", config);

    return true;
}
```

OnStop method

The **OnStop** method is called after a role instance has been taken offline by Azure and before the process exits. You can override this method to call code required for your role instance to cleanly shut down.

IMPORTANT

Code running in the **OnStop** method has a limited time to finish when it is called for reasons other than a user-initiated shutdown. After this time elapses, the process is terminated, so you must make sure that code in the **OnStop** method can run quickly or tolerates not running to completion. The **OnStop** method is called after the **Stopping** event is raised.

Run method

You can override the **Run** method to implement a long-running thread for your role instance.

Overriding the **Run** method is not required; the default implementation starts a thread that sleeps forever. If you do override the **Run** method, your code should block indefinitely. If the **Run** method returns, the role is automatically gracefully recycled; in other words, Azure raises the **Stopping** event and calls the **OnStop** method so that your shutdown sequences may be executed before the role is taken offline.

Implementing the ASP.NET lifecycle methods for a web role

You can use the ASP.NET lifecycle methods, in addition to those provided by the **RoleEntryPoint** class, to manage initialization and shutdown sequences for a web role. This may be useful for compatibility purposes if you are porting an existing ASP.NET application to Azure. The ASP.NET lifecycle methods are called from within the **RoleEntryPoint** methods. The **Application_Start** method is called after the **RoleEntryPoint.OnStart** method finishes. The **Application_End** method is called before the **RoleEntryPoint.OnStop** method is called.

Next steps

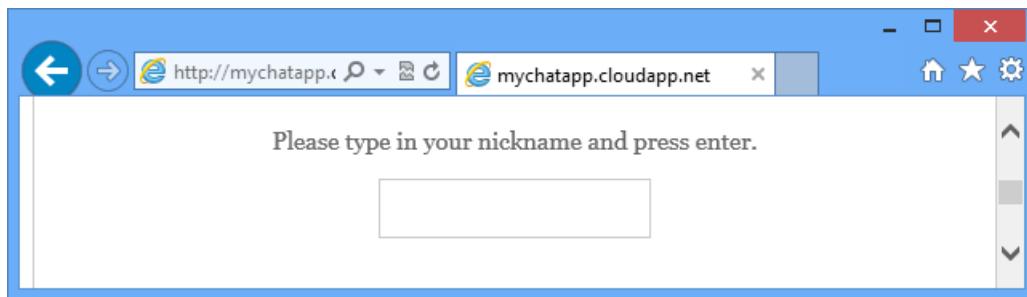
Learn how to [create a cloud service package](#).

Build a Node.js chat application with Socket.IO on an Azure Cloud Service

7/12/2018 • 4 minutes to read • [Edit Online](#)

Socket.IO provides realtime communication between your node.js server and clients. This tutorial walks you through hosting a socket.IO based chat application on Azure. For more information on Socket.IO, see [socket.io](#).

A screenshot of the completed application is below:



Prerequisites

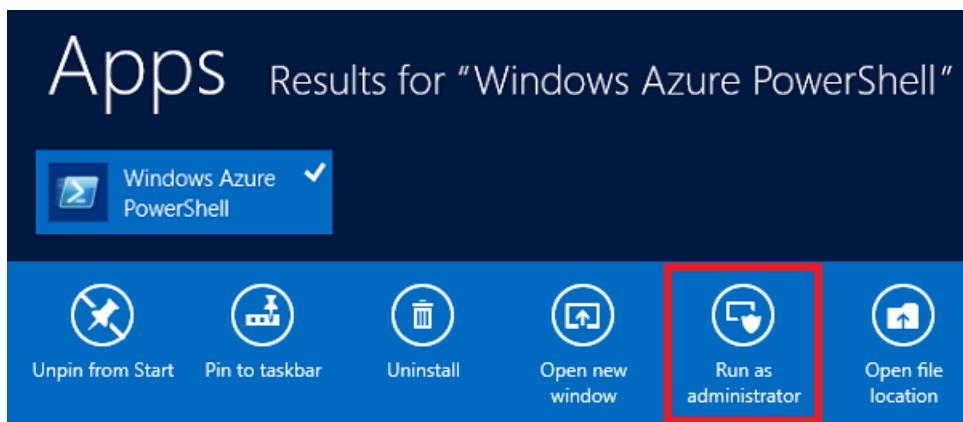
Ensure that the following products and versions are installed to successfully complete the example in this article:

- Install [Visual Studio](#)
- Install [Node.js](#)
- Install [Python version 2.7.10](#)

Create a Cloud Service Project

The following steps create the cloud service project that will host the Socket.IO application.

1. From the **Start Menu** or **Start Screen**, search for **Windows PowerShell**. Finally, right-click **Windows PowerShell** and select **Run As Administrator**.



2. Create a directory called **c:\node**.

```
PS C:\> md node
```

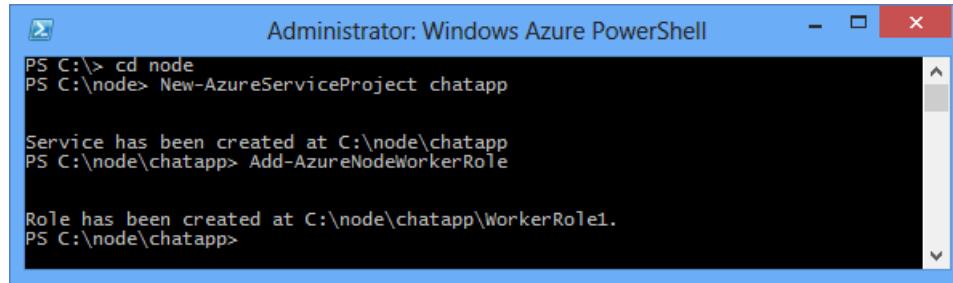
3. Change directories to the **c:\node** directory

```
PS C:\> cd node
```

4. Enter the following commands to create a new solution named **chatapp** and a worker role named **WorkerRole1**:

```
PS C:\node> New-AzureServiceProject chatapp
PS C:\Node> Add-AzureNodeWorkerRole
```

You will see the following response:



```
Administrator: Windows Azure PowerShell
PS C:\> cd node
PS C:\node> New-AzureServiceProject chatapp

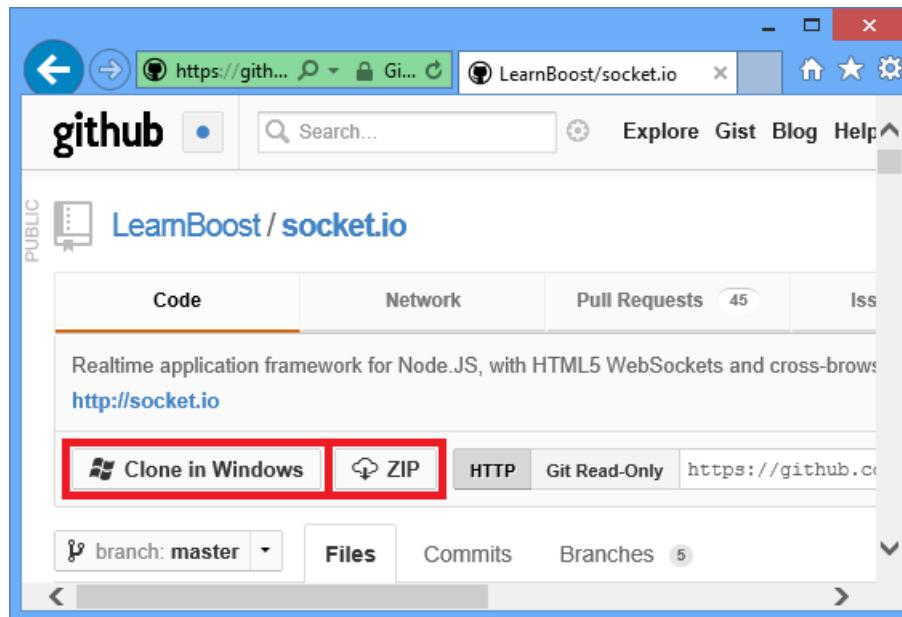
Service has been created at C:\node\chatapp
PS C:\node\chatapp> Add-AzureNodeWorkerRole

Role has been created at C:\node\chatapp\WorkerRole1.
PS C:\node\chatapp>
```

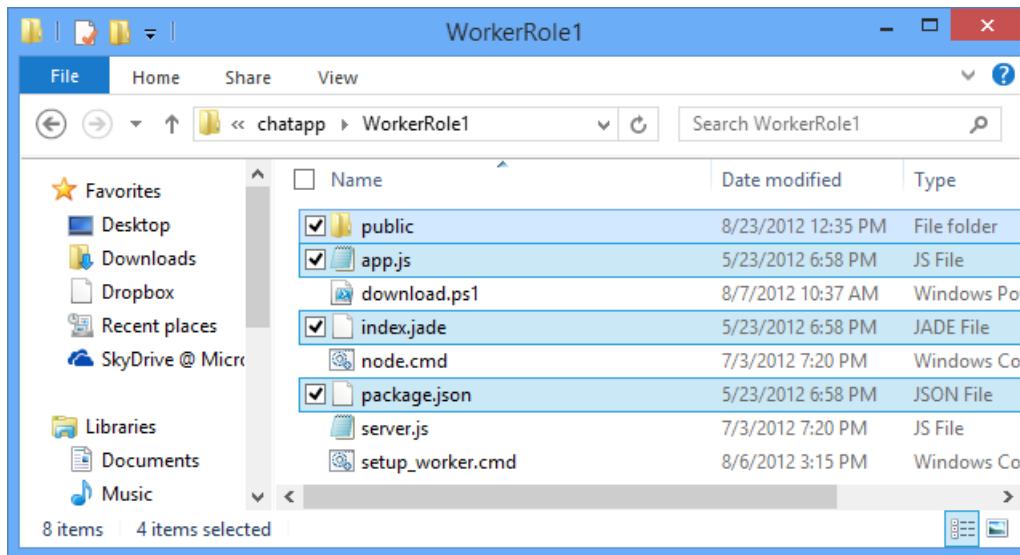
Download the Chat Example

For this project, we will use the chat example from the [Socket.IO GitHub repository](#). Perform the following steps to download the example and add it to the project you previously created.

1. Create a local copy of the repository by using the **Clone** button. You may also use the **ZIP** button to download the project.



2. Navigate the directory structure of the local repository until you arrive at the **examples\chat** directory. Copy the contents of this directory to the **C:\node\chatapp\WorkerRole1** directory created earlier.



The highlighted items in the screenshot above are the files copied from the **examples\chat** directory

3. In the **C:\node\chatapp\WorkerRole1** directory, delete the **server.js** file, and then rename the **app.js** file to **server.js**. This removes the default **server.js** file created previously by the **Add-AzureNodeWorkerRole** cmdlet and replaces it with the application file from the chat example.

Modify Server.js and Install Modules

Before testing the application in the Azure emulator, we must make some minor modifications. Perform the following steps to the server.js file:

1. Open the **server.js** file in Visual Studio or any text editor.
2. Find the **Module dependencies** section at the beginning of server.js and change the line containing **sio = require('../..//lib//socket.io')** to **sio = require('socket.io')** as shown below:

```
var express = require('express')
, stylus = require('stylus')
, nib = require('nib')
//, sio = require('..../..//lib//socket.io'); //Original
, sio = require('socket.io'); //Updated
var port = process.env.PORT || 3000; //Updated
```

3. To ensure the application listens on the correct port, open server.js in Notepad or your favorite editor, and then change the following line by replacing **3000** with **process.env.port** as shown below:

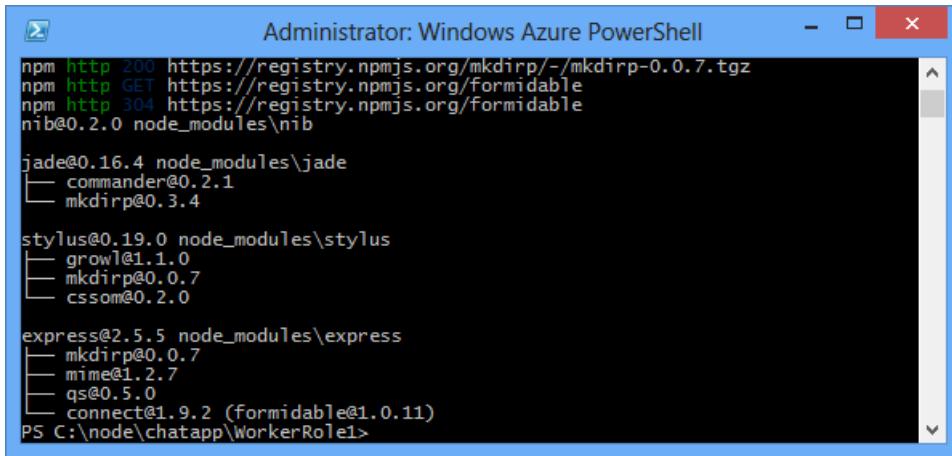
```
//app.listen(3000, function () { //Original
app.listen(process.env.port, function () { //Updated
  var addr = app.address();
  console.log('  app listening on http://'+addr.address+':'+addr.port);
});
```

After saving the changes to **server.js**, use the following steps to install required modules, and then test the application in the Azure emulator:

1. Using **Azure PowerShell**, change directories to the **C:\node\chatapp\WorkerRole1** directory and use the following command to install the modules required by this application:

```
PS C:\node\chatapp\WorkerRole1> npm install
```

This will install the modules listed in the package.json file. After the command completes, you should see output similar to the following:



A screenshot of the Windows Azure PowerShell window titled "Administrator: Windows Azure PowerShell". The command entered is "npm ls", which lists the package dependencies for the current project. The output shows the following tree structure:

```
npm http 200 https://registry.npmjs.org/mkdirp/-/mkdirp-0.0.7.tgz
npm http GET https://registry.npmjs.org/formidable
npm http 304 https://registry.npmjs.org/formidable
nib@0.2.0 node_modules\nib
├── commander@0.2.1
└── mkdirp@0.3.4

jade@0.16.4 node_modules\jade
├── commander@0.2.1
└── mkdirp@0.3.4

stylus@0.19.0 node_modules\stylus
├── growl@1.1.0
└── mkdirp@0.0.7

cssom@0.2.0

express@2.5.5 node_modules\express
├── mkdirp@0.0.7
├── mime@1.2.7
└── qs@0.5.0

connect@1.9.2 (formidable@1.0.11)
PS C:\node\chatapp\WorkerRole1>
```

2. Since this example was originally a part of the Socket.IO GitHub repository, and directly referenced the Socket.IO library by relative path, Socket.IO was not referenced in the package.json file, so we must install it by issuing the following command:

```
PS C:\node\chatapp\WorkerRole1> npm install socket.io --save
```

Test and Deploy

1. Launch the emulator by issuing the following command:

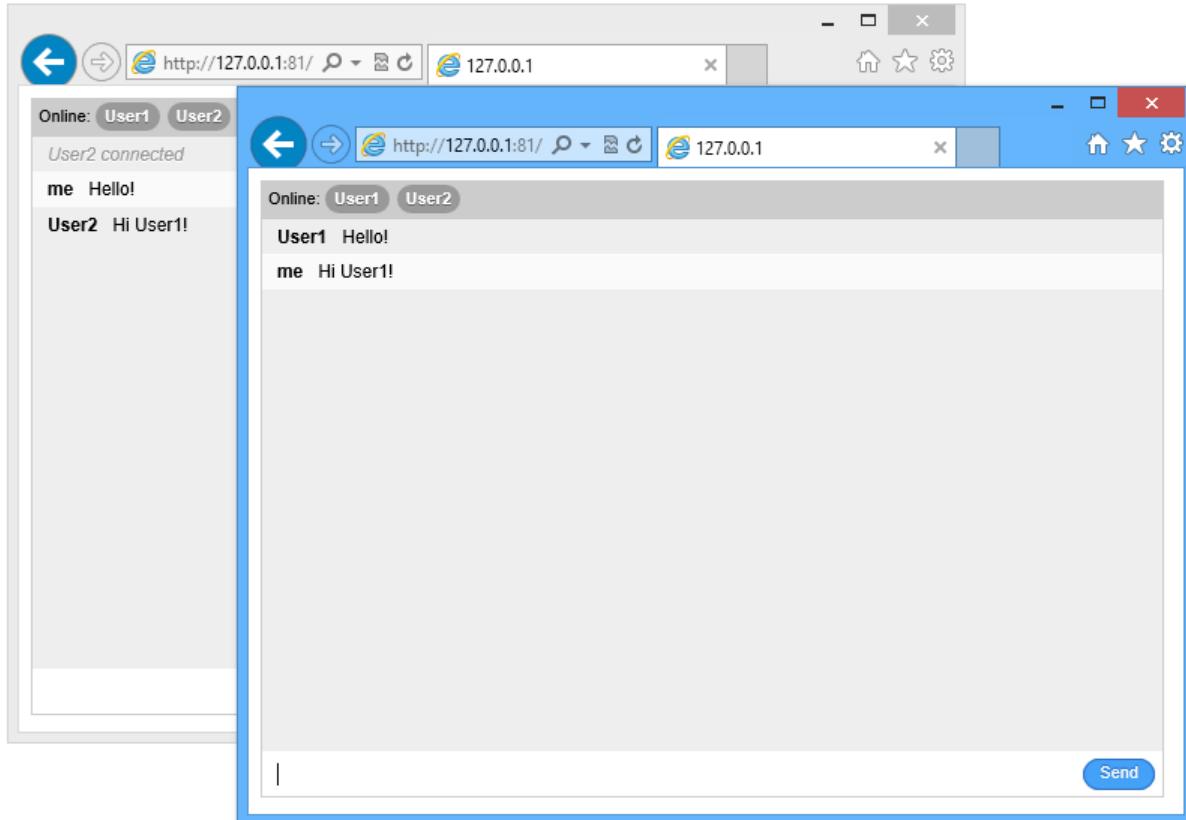
```
PS C:\node\chatapp\WorkerRole1> Start-AzureEmulator -Launch
```

NOTE

If you encounter issues with launching emulator, eg.: Start-AzureEmulator : An unexpected failure occurred. Details: Encountered an unexpected error The communication object, System.ServiceModel.Channels.ServiceChannel, cannot be used for communication because it is in the Faulted state.

reinstall AzureAuthoringTools v 2.7.1 and AzureComputeEmulator v 2.7 - make sure that version matches. >
>

2. Open a browser and navigate to <http://127.0.0.1>.
3. When the browser window opens, enter a nickname and then hit enter. This will allow you to post messages as a specific nickname. To test multi-user functionality, open additional browser windows using the same URL and enter different nicknames.



4. After testing the application, stop the emulator by issuing the following command:

```
PS C:\node\chatapp\WorkerRole1> Stop-AzureEmulator
```

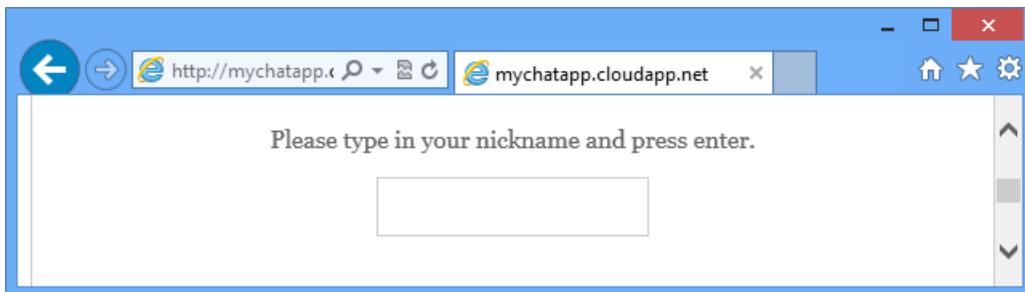
5. To deploy the application to Azure, use the **Publish-AzureServiceProject** cmdlet. For example:

```
PS C:\node\chatapp\WorkerRole1> Publish-AzureServiceProject -ServiceName mychatapp -Location "East US" -Launch
```

IMPORTANT

Be sure to use a unique name, otherwise the publish process will fail. After the deployment has completed, the browser will open and navigate to the deployed service.

If you receive an error stating that the provided subscription name doesn't exist in the imported publish profile, you must download and import the publishing profile for your subscription before deploying to Azure. See the [Deploying the Application to Azure](#) section of [Build and deploy a Node.js application to an Azure Cloud Service](#)



NOTE

If you receive an error stating that the provided subscription name doesn't exist in the imported publish profile, you must download and import the publishing profile for your subscription before deploying to Azure. See the [Deploying the Application to Azure](#) section of [Build and deploy a Node.js application to an Azure Cloud Service](#)

Your application is now running on Azure, and can relay chat messages between different clients using Socket.IO.

NOTE

For simplicity, this sample is limited to chatting between users connected to the same instance. This means that if the cloud service creates two worker role instances, users will only be able to chat with others connected to the same worker role instance. To scale the application to work with multiple role instances, you could use a technology like Service Bus to share the Socket.IO store state across instances. For examples, see the Service Bus Queues and Topics usage samples in the [Azure SDK for Node.js GitHub repository](#).

Next steps

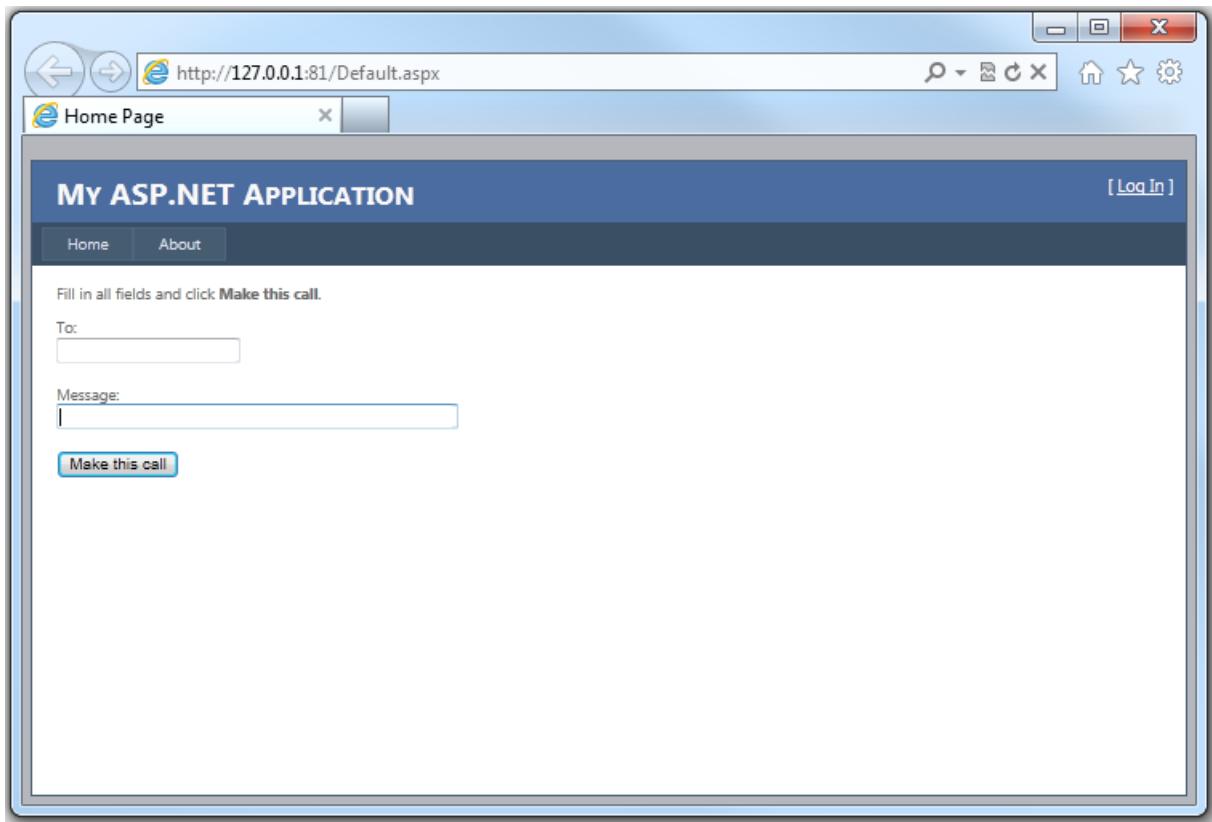
In this tutorial you learned how to create a basic chat application hosted in an Azure Cloud Service. To learn how to host this application in an Azure Website, see [Build a Node.js Chat Application with Socket.IO on an Azure Web Site](#).

For more information, see also the [Node.js Developer Center](#).

How to make a phone call using Twilio in a web role on Azure

11/27/2018 • 3 minutes to read • [Edit Online](#)

This guide demonstrates how to use Twilio to make a call from a web page hosted in Azure. The resulting application prompts the user to make a call with the given number and message, as shown in the following screenshot.



Prerequisites

You will need to do the following to use the code in this topic:

1. Acquire a Twilio account and authentication token from the [Twilio Console](#). To get started with Twilio, sign up at <https://www.twilio.com/try-twilio>. You can evaluate pricing at <https://www.twilio.com/pricing>. For information about the API provided by Twilio, see <https://www.twilio.com/voice/api>.
2. Add the *Twilio .NET library* to your web role. See **To add the Twilio libraries to your web role project**, later in this topic.

You should be familiar with creating a basic [Web Role on Azure](#).

How to: Create a web form for making a call

To add the Twilio libraries to your web role project:

1. Open your solution in Visual Studio.
2. Right-click **References**.
3. Click **Manage NuGet Packages**.

4. Click **Online**.
5. In the search online box, type *twilio*.
6. Click **Install** on the Twilio package.

The following code shows how to create a web form to retrieve user data for making a call. In this example, an ASP.NET Web Role named **TwilioCloud** is created.

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.master"
   AutoEventWireup="true" CodeBehind="Default.aspx.cs"
   Inherits="WebRole1._Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
  <div>
    <asp:BulletedList ID="varDisplay" runat="server" BulletStyle="NotSet">
    </asp:BulletedList>
  </div>
  <div>
    <p>Fill in all fields and click <b>Make this call</b>.</p>
    <div>
      To:<br /><asp:TextBox ID="toNumber" runat="server" /><br /><br />
      Message:<br /><asp:TextBox ID="message" runat="server" /><br /><br />
      <asp:Button ID="callpage" runat="server" Text="Make this call"
        onclick="callpage_Click" />
    </div>
  </div>
</asp:Content>
```

How to: Create the code to make the call

The following code, which is called when the user completes the form, creates the call message and generates the call. In this example, the code is run in the onclick event handler of the button on the form. (Use your Twilio account and authentication token instead of the placeholder values assigned to `accountSID` and `authToken` in the code below.)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Twilio;
using Twilio.Http;
using Twilio.Types;
using Twilio.Rest.Api.V2010;

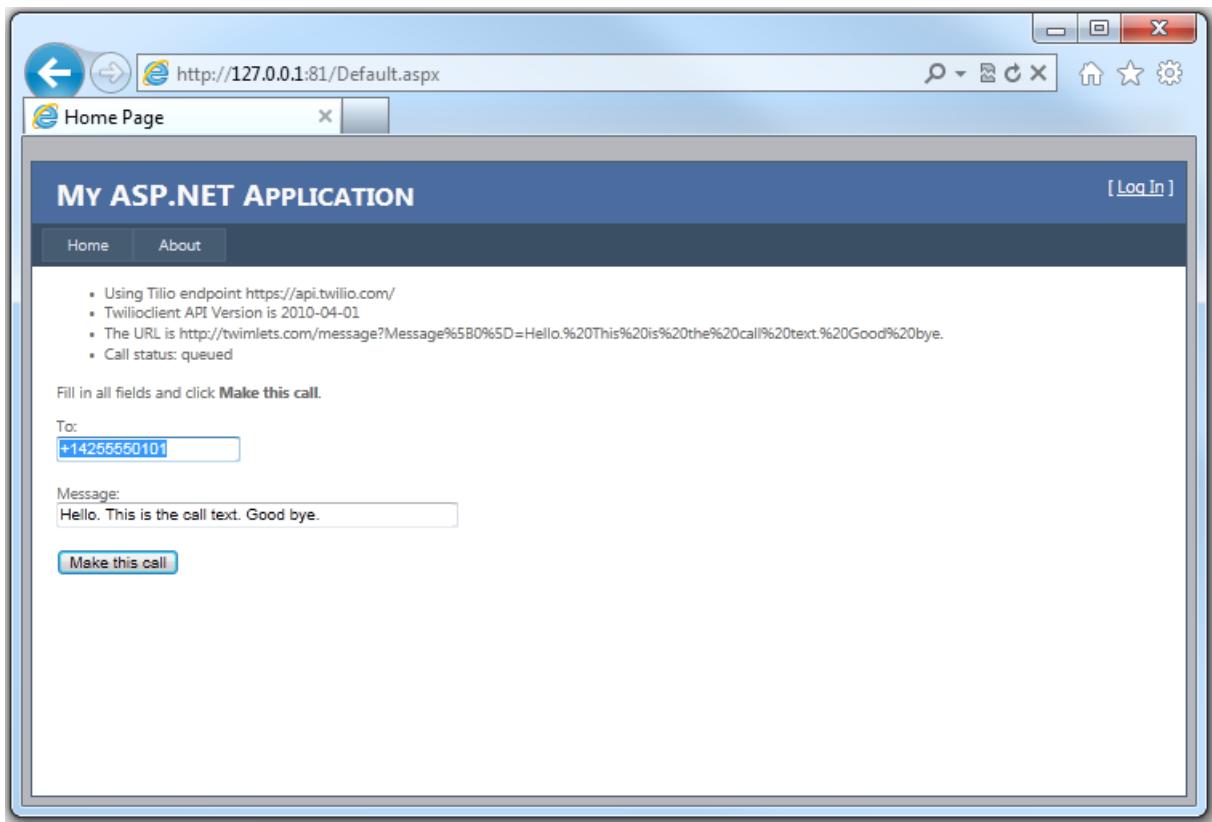
namespace WebRole1
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void callpage_Click(object sender, EventArgs e)
        {
            // Call porcessing happens here.

            // Use your account SID and authentication token instead of
            // the placeholders shown here.
            var accountSID = "ACXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
            var authToken = "your_auth_token";
        }
    }
}
```

The call is made, and the Twilio endpoint, API version, and the call status are displayed. The following screenshot shows output from a sample run.



More information about TwiML can be found at <https://www.twilio.com/docs/api/twiml>. More information about <Say> and other Twilio verbs can be found at <https://www.twilio.com/docs/api/twiml/say>.

Next steps

This code was provided to show you basic functionality using Twilio in an ASP.NET web role on Azure. Before deploying to Azure in production, you may want to add more error handling or other features. For example:

- Instead of using a web form, you could use Azure Blob storage or an Azure SQL Database instance to store phone numbers and call text. For information about using Blobs in Azure, see [How to use the Azure Blob storage service in .NET](#). For information about using SQL Database, see [How to use Azure SQL Database in .NET applications](#).
- You could use `RoleEnvironment.getConfigurationSettings` to retrieve the Twilio account ID and authentication token from your deployment's configuration settings, instead of hard-coding the values in your form. For information about the `RoleEnvironment` class, see [Microsoft.WindowsAzure.ServiceRuntime Namespace](#).
- Read the Twilio Security Guidelines at <https://www.twilio.com/docs/security>.
- Learn more about Twilio at <https://www.twilio.com/docs>.

See also

- [How to use Twilio for Voice and SMS capabilities from Azure](#)

How to configure and run startup tasks for a cloud service

7/12/2018 • 6 minutes to read • [Edit Online](#)

You can use startup tasks to perform operations before a role starts. Operations that you might want to perform include installing a component, registering COM components, setting registry keys, or starting a long running process.

NOTE

Startup tasks are not applicable to Virtual Machines, only to Cloud Service Web and Worker roles.

How startup tasks work

Startup tasks are actions that are taken before your roles begin and are defined in the [ServiceDefinition.csdef](#) file by using the [Task](#) element within the [Startup](#) element. Frequently startup tasks are batch files, but they can also be console applications, or batch files that start PowerShell scripts.

Environment variables pass information into a startup task, and local storage can be used to pass information out of a startup task. For example, an environment variable can specify the path to a program you want to install, and files can be written to local storage that can then be read later by your roles.

Your startup task can log information and errors to the directory specified by the **TEMP** environment variable. During the startup task, the **TEMP** environment variable resolves to the `C:\Resources\temp\[guid].[rolename]\RoleTemp` directory when running on the cloud.

Startup tasks can also be executed several times between reboots. For example, the startup task will be run each time the role recycles, and role recycles may not always include a reboot. Startup tasks should be written in a way that allows them to run several times without problems.

Startup tasks must end with an **errorlevel** (or exit code) of zero for the startup process to complete. If a startup task ends with a non-zero **errorlevel**, the role will not start.

Role startup order

The following lists the role startup procedure in Azure:

1. The instance is marked as **Starting** and does not receive traffic.
2. All startup tasks are executed according to their **taskType** attribute.
 - The **simple** tasks are executed synchronously, one at a time.
 - The **background** and **foreground** tasks are started asynchronously, parallel to the startup task.

WARNING

IIS may not be fully configured during the startup task stage in the startup process, so role-specific data may not be available. Startup tasks that require role-specific data should use [Microsoft.WindowsAzure.ServiceRuntime.RoleEntryPoint.OnStart](#).

3. The role host process is started and the site is created in IIS.

4. The [Microsoft.WindowsAzure.ServiceRuntime.RoleEntryPoint.OnStart](#) method is called.
5. The instance is marked as **Ready** and traffic is routed to the instance.
6. The [Microsoft.WindowsAzure.ServiceRuntime.RoleEntryPoint.Run](#) method is called.

Example of a startup task

Startup tasks are defined in the [ServiceDefinition.csdef](#) file, in the **Task** element. The **commandLine** attribute specifies the name and parameters of the startup batch file or console command, the **executionContext** attribute specifies the privilege level of the startup task, and the **taskType** attribute specifies how the task will be executed.

In this example, an environment variable, **MyVersionNumber**, is created for the startup task and set to the value "1.0.0.0".

ServiceDefinition.csdef:

```
<Startup>
  <Task commandLine="Startup.cmd" executionContext="limited" taskType="simple" >
    <Environment>
      <Variable name="MyVersionNumber" value="1.0.0.0" />
    </Environment>
  </Task>
</Startup>
```

In the following example, the **Startup.cmd** batch file writes the line "The current version is 1.0.0.0" to the `StartupLog.txt` file in the directory specified by the TEMP environment variable. The `EXIT /B 0` line ensures that the startup task ends with an **errorlevel** of zero.

```
ECHO The current version is %MyVersionNumber% >> "%TEMP%\StartupLog.txt" 2>&1
EXIT /B 0
```

NOTE

In Visual Studio, the **Copy to Output Directory** property for your startup batch file should be set to **Copy Always** to be sure that your startup batch file is properly deployed to your project on Azure (**approot\bin** for Web roles, and **approot** for worker roles).

Description of task attributes

The following describes the attributes of the **Task** element in the [ServiceDefinition.csdef](#) file:

commandLine - Specifies the command line for the startup task:

- The command, with optional command line parameters, which begins the startup task.
- Frequently this is the filename of a .cmd or .bat batch file.
- The task is relative to the AppRoot\Bin folder for the deployment. Environment variables are not expanded in determining the path and file of the task. If environment expansion is required, you can create a small .cmd script that calls your startup task.
- Can be a console application or a batch file that starts a [PowerShell script](#).

executionContext - Specifies the privilege level for the startup task. The privilege level can be limited or elevated:

- **limited**

The startup task runs with the same privileges as the role. When the **executionContext** attribute for the **Runtime** element is also **limited**, then user privileges are used.

- **elevated**

The startup task runs with administrator privileges. This allows startup tasks to install programs, make IIS configuration changes, perform registry changes, and other administrator level tasks, without increasing the privilege level of the role itself.

NOTE

The privilege level of a startup task does not need to be the same as the role itself.

taskType - Specifies the way a startup task is executed.

- **simple**

Tasks are executed synchronously, one at a time, in the order specified in the [ServiceDefinition.csdef](#) file.

When one **simple** startup task ends with an **errorlevel** of zero, the next **simple** startup task is executed. If there are no more **simple** startup tasks to execute, then the role itself will be started.

NOTE

If the **simple** task ends with a non-zero **errorlevel**, the instance will be blocked. Subsequent **simple** startup tasks, and the role itself, will not start.

To ensure that your batch file ends with an **errorlevel** of zero, execute the command `EXIT /B 0` at the end of your batch file process.

- **background**

Tasks are executed asynchronously, in parallel with the startup of the role.

- **foreground**

Tasks are executed asynchronously, in parallel with the startup of the role. The key difference between a **foreground** and a **background** task is that a **foreground** task prevents the role from recycling or shutting down until the task has ended. The **background** tasks do not have this restriction.

Environment variables

Environment variables are a way to pass information to a startup task. For example, you can put the path to a blob that contains a program to install, or port numbers that your role will use, or settings to control features of your startup task.

There are two kinds of environment variables for startup tasks; static environment variables and environment variables based on members of the [RoleEnvironment](#) class. Both are in the [Environment](#) section of the [ServiceDefinition.csdef](#) file, and both use the [Variable](#) element and **name** attribute.

Static environment variables uses the **value** attribute of the [Variable](#) element. The example above creates the environment variable **MyVersionNumber** which has a static value of "**1.0.0.0**". Another example would be to create a **StagingOrProduction** environment variable which you can manually set to values of "**staging**" or "**production**" to perform different startup actions based on the value of the **StagingOrProduction** environment variable.

Environment variables based on members of the [RoleEnvironment](#) class do not use the **value** attribute of the [Variable](#) element. Instead, the [RoleInstanceValue](#) child element, with the appropriate **XPath** attribute value, are used to create an environment variable based on a specific member of the [RoleEnvironment](#) class. Values for the **XPath** attribute to access various [RoleEnvironment](#) values can be found [here](#).

For example, to create an environment variable that is "**true**" when the instance is running in the compute emulator, and "**false**" when running in the cloud, use the following [Variable](#) and [RoleInstanceValue](#) elements:

```
<Startup>
  <Task commandLine="Startup.cmd" executionContext="limited" taskType="simple">
    <Environment>

      <!-- Create the environment variable that informs the startup task whether it is running
          in the Compute Emulator or in the cloud. "%ComputeEmulatorRunning%"=="true" when
          running in the Compute Emulator, "%ComputeEmulatorRunning%"=="false" when running
          in the cloud. -->

      <Variable name="ComputeEmulatorRunning">
        <RoleInstanceValue xpath="/RoleEnvironment/Deployment/@emulated" />
      </Variable>

    </Environment>
  </Task>
</Startup>
```

Next steps

Learn how to perform some [common startup tasks](#) with your Cloud Service.

[Package](#) your Cloud Service.

Common Cloud Service startup tasks

8/7/2018 • 16 minutes to read • [Edit Online](#)

This article provides some examples of common startup tasks you may want to perform in your cloud service. You can use startup tasks to perform operations before a role starts. Operations that you might want to perform include installing a component, registering COM components, setting registry keys, or starting a long running process.

See [this article](#) to understand how startup tasks work, and specifically how to create the entries that define a startup task.

NOTE

Startup tasks are not applicable to Virtual Machines, only to Cloud Service Web and Worker roles.

Define environment variables before a role starts

If you need environment variables defined for a specific task, use the [Environment](#) element inside the [Task](#) element.

```
<ServiceDefinition name="MyService"
  xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
  <WorkerRole name="WorkerRole1">
    ...
    <Startup>
      <Task commandLine="Startup.cmd" executionContext="limited" taskType="simple">
        <Environment>
          <Variable name="MyEnvironmentVariable" value="MyVariableValue" />
        </Environment>
      </Task>
    </Startup>
  </WorkerRole>
</ServiceDefinition>
```

Variables can also use a [valid Azure XPath](#) value to reference something about the deployment. Instead of using the `value` attribute, define a [RoleInstanceValue](#) child element.

```
<Variable name="PathToStartupStorage">
  <RoleInstanceValue
    xpath="/RoleEnvironment/CurrentInstance/LocalResources/LocalResource[@name='StartupLocalStorage']/@path" />
</Variable>
```

Configure IIS startup with AppCmd.exe

The [AppCmd.exe](#) command-line tool can be used to manage IIS settings at startup on Azure. [AppCmd.exe](#) provides convenient, command-line access to configuration settings for use in startup tasks on Azure. Using [AppCmd.exe](#), Website settings can be added, modified, or removed for applications and sites.

However, there are a few things to watch out for in the use of [AppCmd.exe](#) as a startup task:

- Startup tasks can be run more than once between reboots. For instance, when a role recycles.
- If a [AppCmd.exe](#) action is performed more than once, it may generate an error. For example, attempting to add

a section to *Web.config* twice could generate an error.

- Startup tasks fail if they return a non-zero exit code or **errorlevel**. For example, when *AppCmd.exe* generates an error.

It is a good practice to check the **errorlevel** after calling *AppCmd.exe*, which is easy to do if you wrap the call to *AppCmd.exe* with a *.cmd* file. If you detect a known **errorlevel** response, you can ignore it, or pass it back.

The errorlevel returned by *AppCmd.exe* are listed in the *winerror.h* file, and can also be seen on [MSDN](#).

Example of managing the error level

This example adds a compression section and a compression entry for JSON to the *Web.config* file, with error handling and logging.

The relevant sections of the *ServiceDefinition.csdef* file are shown here, which include setting the **executionContext** attribute to `elevated` to give *AppCmd.exe* sufficient permissions to change the settings in the *Web.config* file:

```
<ServiceDefinition name="MyService"
  xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
  <WorkerRole name="WorkerRole1">
    ...
    <Startup>
      <Task commandLine="Startup.cmd" executionContext="elevated" taskType="simple" />
    </Startup>
  </WorkerRole>
</ServiceDefinition>
```

The *Startup.cmd* batch file uses *AppCmd.exe* to add a compression section and a compression entry for JSON to the *Web.config* file. The expected **errorlevel** of 183 is set to zero using the **VERIFY.EXE** command-line program. Unexpected errorlevels are logged to *StartupErrorLog.txt*.

```

REM *** Add a compression section to the Web.config file. ***
%windir%\system32\inetsrv\appcmd set config /section:urlCompression /doDynamicCompression:True /commit:apphost
>> "%TEMP%\StartupLog.txt" 2>&1

REM ERRORLEVEL 183 occurs when trying to add a section that already exists. This error is expected if this
REM batch file were executed twice. This can occur and must be accounted for in a Azure startup
REM task. To handle this situation, set the ERRORLEVEL to zero by using the Verify command. The Verify
REM command will safely set the ERRORLEVEL to zero.
IF %ERRORLEVEL% EQU 183 DO VERIFY > NUL

REM If the ERRORLEVEL is not zero at this point, some other error occurred.
IF %ERRORLEVEL% NEQ 0 (
    ECHO Error adding a compression section to the Web.config file. >> "%TEMP%\StartupLog.txt" 2>&1
    GOTO ErrorExit
)

REM *** Add compression for json. ***
%windir%\system32\inetsrv\appcmd set config -section:system.webServer/httpCompression /+"dynamicTypes.
[mimeType='application/json; charset=utf-8',enabled='True']" /commit:apphost >> "%TEMP%\StartupLog.txt" 2>&1
IF %ERRORLEVEL% EQU 183 VERIFY > NUL
IF %ERRORLEVEL% NEQ 0 (
    ECHO Error adding the JSON compression type to the Web.config file. >> "%TEMP%\StartupLog.txt" 2>&1
    GOTO ErrorExit
)

REM *** Exit batch file. ***
EXIT /b 0

REM *** Log error and exit ***
:ErrorExit
REM Report the date, time, and ERRORLEVEL of the error.
DATE /T >> "%TEMP%\StartupLog.txt" 2>&1
TIME /T >> "%TEMP%\StartupLog.txt" 2>&1
ECHO An error occurred during startup. ERRORLEVEL = %ERRORLEVEL% >> "%TEMP%\StartupLog.txt" 2>&1
EXIT %ERRORLEVEL%

```

Add firewall rules

In Azure, there are effectively two firewalls. The first firewall controls connections between the virtual machine and the outside world. This firewall is controlled by the [EndPoints](#) element in the [ServiceDefinition.csdef](#) file.

The second firewall controls connections between the virtual machine and the processes within that virtual machine. This firewall can be controlled by the `netsh advfirewall firewall` command-line tool.

Azure creates firewall rules for the processes started within your roles. For example, when you start a service or program, Azure automatically creates the necessary firewall rules to allow that service to communicate with the Internet. However, if you create a service that is started by a process outside your role (like a COM+ service or a Windows Scheduled Task), you need to manually create a firewall rule to allow access to that service. These firewall rules can be created by using a startup task.

A startup task that creates a firewall rule must have an [executionContext](#) of **elevated**. Add the following startup task to the [ServiceDefinition.csdef](#) file.

```

<ServiceDefinition name="MyService"
  xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
  <WorkerRole name="WorkerRole1">
    ...
    <Startup>
      <Task commandLine="AddFirewallRules.cmd" executionContext="elevated" taskType="simple" />
    </Startup>
  </WorkerRole>
</ServiceDefinition>

```

To add the firewall rule, you must use the appropriate `netsh advfirewall firewall` commands in your startup batch file. In this example, the startup task requires security and encryption for TCP port 80.

```

REM Add a firewall rule in a startup task.

REM Add an inbound rule requiring security and encryption for TCP port 80 traffic.
netsh advfirewall firewall add rule name="Require Encryption for Inbound TCP/80" protocol=TCP dir=in
localport=80 security=authdynenc action=allow >> "%TEMP%\StartupLog.txt" 2>&1

REM If an error occurred, return the errorlevel.
EXIT /B %errorlevel%

```

Block a specific IP address

You can restrict an Azure web role access to a set of specified IP addresses by modifying your IIS **web.config** file. You also need to use a command file which unlocks the **ipSecurity** section of the **ApplicationHost.config** file.

To do unlock the **ipSecurity** section of the **ApplicationHost.config** file, create a command file that runs at role start. Create a folder at the root level of your web role called **startup** and, within this folder, create a batch file called **startup.cmd**. Add this file to your Visual Studio project and set the properties to **Copy Always** to ensure that it is included in your package.

Add the following startup task to the **ServiceDefinition.csdef** file.

```

<ServiceDefinition name="MyService"
  xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
  <WebRole name="WebRole1">
    ...
    <Startup>
      <Task commandLine="startup.cmd" executionContext="elevated" />
    </Startup>
  </WebRole>
</ServiceDefinition>

```

Add this command to the **startup.cmd** file:

```

@echo off
@echo Installing "IPv4 Address and Domain Restrictions" feature
powershell -ExecutionPolicy Unrestricted -command "Install-WindowsFeature Web-IP-Security"
@echo Unlocking configuration for "IPv4 Address and Domain Restrictions" feature
%windir%\system32\inetsrv\appcmd.exe unlock config -section:system.webServer/security/ipSecurity

```

This task causes the **startup.cmd** batch file to be run every time the web role is initialized, ensuring that the required **ipSecurity** section is unlocked.

Finally, modify the **system.webServer** section your web role's **web.config** file to add a list of IP addresses that are granted access, as shown in the following example:

This sample config **allows** all IPs to access the server except the two defined

```
<system.webServer>
  <security>
    <!--Unlisted IP addresses are granted access-->
    <ipSecurity>
      <!--The following IP addresses are denied access-->
      <add allowed="false" ipAddress="192.168.100.1" subnetMask="255.255.0.0" />
      <add allowed="false" ipAddress="192.168.100.2" subnetMask="255.255.0.0" />
    </ipSecurity>
  </security>
</system.webServer>
```

This sample config **denies** all IPs from accessing the server except for the two defined.

```
<system.webServer>
  <security>
    <!--Unlisted IP addresses are denied access-->
    <ipSecurity allowUnlisted="false">
      <!--The following IP addresses are granted access-->
      <add allowed="true" ipAddress="192.168.100.1" subnetMask="255.255.0.0" />
      <add allowed="true" ipAddress="192.168.100.2" subnetMask="255.255.0.0" />
    </ipSecurity>
  </security>
</system.webServer>
```

Create a PowerShell startup task

Windows PowerShell scripts cannot be called directly from the [ServiceDefinition.csdef](#) file, but they can be invoked from within a startup batch file.

PowerShell (by default) does not run unsigned scripts. Unless you sign your script, you need to configure PowerShell to run unsigned scripts. To run unsigned scripts, the **ExecutionPolicy** must be set to **Unrestricted**. The **ExecutionPolicy** setting that you use is based on the version of Windows PowerShell.

```
REM   Run an unsigned PowerShell script and log the output
PowerShell -ExecutionPolicy Unrestricted .\startup.ps1 >> "%TEMP%\StartupLog.txt" 2>&1

REM   If an error occurred, return the errorlevel.
EXIT /B %errorlevel%
```

If you're using a Guest OS that runs PowerShell 2.0 or 1.0 you can force version 2 to run, and if unavailable, use version 1.

```
REM   Attempt to set the execution policy by using PowerShell version 2.0 syntax.
PowerShell -Version 2.0 -ExecutionPolicy Unrestricted .\startup.ps1 >> "%TEMP%\StartupLog.txt" 2>&1

REM   If PowerShell version 2.0 isn't available. Set the execution policy by using the PowerShell
IF %ERRORLEVEL% EQU -393216 (
  PowerShell -Command "Set-ExecutionPolicy Unrestricted" >> "%TEMP%\StartupLog.txt" 2>&1
  PowerShell .\startup.ps1 >> "%TEMP%\StartupLog.txt" 2>&1
)

REM   If an error occurred, return the errorlevel.
EXIT /B %errorlevel%
```

Create files in local storage from a startup task

You can use a local storage resource to store files created by your startup task that is accessed later by your application.

To create the local storage resource, add a [LocalResources](#) section to the [ServiceDefinition.csdef](#) file and then add the [LocalStorage](#) child element. Give the local storage resource a unique name and an appropriate size for your startup task.

To use a local storage resource in your startup task, you need to create an environment variable to reference the local storage resource location. Then the startup task and the application are able to read and write files to the local storage resource.

The relevant sections of the **ServiceDefinition.csdef** file are shown here:

```
<ServiceDefinition name="MyService"
  xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
  <WorkerRole name="WorkerRole1">
    ...
    <LocalResources>
      <LocalStorage name="StartupLocalStorage" sizeInMB="5"/>
    </LocalResources>

    <Startup>
      <Task commandLine="Startup.cmd" executionContext="limited" taskType="simple">
        <Environment>
          <Variable name="PathToStartupStorage">
            <RoleInstanceValue
xpath="/RoleEnvironment/CurrentInstance/LocalResources/LocalResource[@name='StartupLocalStorage']/@path" />
          </Variable>
        </Environment>
      </Task>
    </Startup>
  </WorkerRole>
</ServiceDefinition>
```

As an example, this **Startup.cmd** batch file uses the **PathToStartupStorage** environment variable to create the file **MyTest.txt** on the local storage location.

```
REM Create a simple text file.

ECHO This text will go into the MyTest.txt file which will be in the    > "%PathToStartupStorage%\MyTest.txt"
ECHO path pointed to by the PathToStartupStorage environment variable. >> "%PathToStartupStorage%\MyTest.txt"
ECHO The contents of the PathToStartupStorage environment variable is    >> "%PathToStartupStorage%\MyTest.txt"
ECHO "%PathToStartupStorage%".                                >> "%PathToStartupStorage%\MyTest.txt"

REM Exit the batch file with ERRORLEVEL 0.

EXIT /b 0
```

You can access local storage folder from the Azure SDK by using the [GetLocalResource](#) method.

```
string localStoragePath =
Microsoft.WindowsAzure.ServiceRuntime.RoleEnvironment.GetLocalResource("StartupLocalStorage").RootPath;

string fileContent = System.IO.File.ReadAllText(System.IO.Path.Combine(localStoragePath, "MyTestFile.txt"));
```

Run in the emulator or cloud

You can have your startup task perform different steps when it is operating in the cloud compared to when it is in

the compute emulator. For example, you may want to use a fresh copy of your SQL data only when running in the emulator. Or you may want to do some performance optimizations for the cloud that you don't need to do when running in the emulator.

This ability to perform different actions on the compute emulator and the cloud can be accomplished by creating an environment variable in the [ServiceDefinition.csdef](#) file. You then test that environment variable for a value in your startup task.

To create the environment variable, add the [Variable/RoleInstanceValue](#) element and create an XPath value of `/RoleEnvironment/Deployment/@emulated`. The value of the **%ComputeEmulatorRunning%** environment variable is `true` when running on the compute emulator, and `false` when running on the cloud.

```
<ServiceDefinition name="MyService"
  xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition">
  <WorkerRole name="WorkerRole1">

    ...
    <Startup>
      <Task commandLine="Startup.cmd" executionContext="limited" taskType="simple">
        <Environment>
          <Variable name="ComputeEmulatorRunning">
            <RoleInstanceValue xpath="/RoleEnvironment/Deployment/@emulated" />
          </Variable>
        </Environment>
      </Task>
    </Startup>
  </WorkerRole>
</ServiceDefinition>
```

The task can now check the **%ComputeEmulatorRunning%** environment variable to perform different actions based on whether the role is running in the cloud or the emulator. Here is a .cmd shell script that checks for that environment variable.

```
REM   Check if this task is running on the compute emulator.

IF "%ComputeEmulatorRunning%" == "true" (
    REM   This task is running on the compute emulator. Perform tasks that must be run only in the compute
    emulator.
) ELSE (
    REM   This task is running on the cloud. Perform tasks that must be run only in the cloud.
)
```

Detect that your task has already run

The role may recycle without a reboot causing your startup tasks to run again. There is no flag to indicate that a task has already run on the hosting VM. You may have some tasks where it doesn't matter that they run multiple times. However, you may run into a situation where you need to prevent a task from running more than once.

The simplest way to detect that a task has already run is to create a file in the **%TEMP%** folder when the task is successful and look for it at the start of the task. Here is a sample cmd shell script that does that for you.

```

REM If Task1_Success.txt exists, then Application 1 is already installed.
IF EXIST "%PathToApp1Install%\Task1_Success.txt" (
    ECHO Application 1 is already installed. Exiting. >> "%TEMP%\StartupLog.txt" 2>&1
    GOTO Finish
)

REM Run your real exe task
ECHO Running XYZ >> "%TEMP%\StartupLog.txt" 2>&1
"%PathToApp1Install%\setup.exe" >> "%TEMP%\StartupLog.txt" 2>&1

IF %ERRORLEVEL% EQU 0 (
    REM The application installed without error. Create a file to indicate that the task
    REM does not need to be run again.

    ECHO This line will create a file to indicate that Application 1 installed correctly. >
    "%PathToApp1Install%\Task1_Success.txt"

) ELSE (
    REM An error occurred. Log the error and exit with the error code.

    DATE /T >> "%TEMP%\StartupLog.txt" 2>&1
    TIME /T >> "%TEMP%\StartupLog.txt" 2>&1
    ECHO An error occurred running task 1. Errorlevel = %ERRORLEVEL%. >> "%TEMP%\StartupLog.txt" 2>&1

    EXIT %ERRORLEVEL%
)

:Finish

REM Exit normally.
EXIT /B 0

```

Task best practices

Here are some best practices you should follow when configuring task for your web or worker role.

Always log startup activities

Visual Studio does not provide a debugger to step through batch files, so it's good to get as much data on the operation of batch files as possible. Logging the output of batch files, both **stdout** and **stderr**, can give you important information when trying to debug and fix batch files. To log both **stdout** and **stderr** to the StartupLog.txt file in the directory pointed to by the **%TEMP%** environment variable, add the text

`>> "%TEMP%\StartupLog.txt" 2>&1` to the end of specific lines you want to log. For example, to execute setup.exe in the **%PathToApp1Install%** directory:

```
"%PathToApp1Install%\setup.exe" >> "%TEMP%\StartupLog.txt" 2>&1
```

To simplify your xml, you can create a wrapper *cmd* file that calls all of your startup tasks along with logging and ensures each child-task shares the same environment variables.

You may find it annoying though to use `>> "%TEMP%\StartupLog.txt" 2>&1` on the end of each startup task. You can enforce task logging by creating a wrapper that handles logging for you. This wrapper calls the real batch file you want to run. Any output from the target batch file will be redirected to the *Startuplog.txt* file.

The following example shows how to redirect all output from a startup batch file. In this example, the ServerDefinition.csdef file creates a startup task that calls *logwrap.cmd*. *logwrap.cmd* calls *Startup2.cmd*, redirecting all output to **%TEMP%\StartupLog.txt**.

ServiceDefinition.cmd:

```
<Startup>
  <Task commandLine="logwrap.cmd startup2.cmd" executionContext="limited" taskType="simple" />
</Startup>
```

logwrap.cmd:

```
@ECHO OFF

REM logwrap.cmd calls passed in batch file, redirecting all output to the StartupLog.txt log file.

ECHO [%date% %time%] == START logwrap.cmd ===== >>
"%TEMP%\StartupLog.txt" 2>&1
ECHO [%date% %time%] Running %1 >> "%TEMP%\StartupLog.txt" 2>&1

REM Call the child command batch file, redirecting all output to the StartupLog.txt log file.
START /B /WAIT %1 >> "%TEMP%\StartupLog.txt" 2>&1

REM Log the completion of child command.
ECHO [%date% %time%] Done >> "%TEMP%\StartupLog.txt" 2>&1

IF %ERRORLEVEL% EQU 0 (

  REM No errors occurred. Exit logwrap.cmd normally.
  ECHO [%date% %time%] == END logwrap.cmd ===== >>
"%TEMP%\StartupLog.txt" 2>&1
  ECHO. >> "%TEMP%\StartupLog.txt" 2>&1
  EXIT /B 0

) ELSE (

  REM Log the error.
  ECHO [%date% %time%] An error occurred. The ERRORLEVEL = %ERRORLEVEL%. >> "%TEMP%\StartupLog.txt" 2>&1
  ECHO [%date% %time%] == END logwrap.cmd ===== >>
"%TEMP%\StartupLog.txt" 2>&1
  ECHO. >> "%TEMP%\StartupLog.txt" 2>&1
  EXIT /B %ERRORLEVEL%

)
```

Startup2.cmd:

```
@ECHO OFF

REM This is the batch file where the startup steps should be performed. Because of the
REM way Startup2.cmd was called, all commands and their outputs will be stored in the
REM StartupLog.txt file in the directory pointed to by the TEMP environment variable.

REM If an error occurs, the following command will pass the ERRORLEVEL back to the
REM calling batch file.

ECHO [%date% %time%] Some log information about this task
ECHO [%date% %time%] Some more log information about this task

EXIT %ERRORLEVEL%
```

Sample output in the **StartupLog.txt** file:

```
[Mon 10/17/2016 20:24:46.75] == START logwrap.cmd =====
[Mon 10/17/2016 20:24:46.75] Running command1.cmd
[Mon 10/17/2016 20:24:46.77] Some log information about this task
[Mon 10/17/2016 20:24:46.77] Some more log information about this task
[Mon 10/17/2016 20:24:46.77] Done
[Mon 10/17/2016 20:24:46.77] == END logwrap.cmd =====
```

TIP

The **StartupLog.txt** file is located in the `C:\Resources\temp\{role identifier}\RoleTemp` folder.

Set executionContext appropriately for startup tasks

Set privileges appropriately for the startup task. Sometimes startup tasks must run with elevated privileges even though the role runs with normal privileges.

The `executionContext` attribute sets the privilege level of the startup task. Using `executionContext="limited"` means the startup task has the same privilege level as the role. Using `executionContext="elevated"` means the startup task has administrator privileges, which allows the startup task to perform administrator tasks without giving administrator privileges to your role.

An example of a startup task that requires elevated privileges is a startup task that uses **AppCmd.exe** to configure IIS. **AppCmd.exe** requires `executionContext="elevated"`.

Use the appropriate taskType

The `taskType` attribute determines the way the startup task is executed. There are three values: **simple**, **background**, and **foreground**. The background and foreground tasks are started asynchronously, and then the simple tasks are executed synchronously one at a time.

With **simple** startup tasks, you can set the order in which the tasks run by the order in which the tasks are listed in the ServiceDefinition.csdef file. If a **simple** task ends with a non-zero exit code, then the startup procedure stops and the role does not start.

The difference between **background** startup tasks and **foreground** startup tasks is that **foreground** tasks keep the role running until the **foreground** task ends. This also means that if the **foreground** task hangs or crashes, the role will not recycle until the **foreground** task is forced closed. For this reason, **background** tasks are recommended for asynchronous startup tasks unless you need that feature of the **foreground** task.

End batch files with EXIT /B 0

The role will only start if the **errorlevel** from each of your simple startup task is zero. Not all programs set the **errorlevel** (exit code) correctly, so the batch file should end with an `EXIT /B 0` if everything ran correctly.

A missing `EXIT /B 0` at the end of a startup batch file is a common cause of roles that do not start.

NOTE

I've noticed that nested batch files sometimes hang when using the `/B` parameter. You may want to make sure that this hang problem does not happen if another batch file calls your current batch file, like if you use the **log wrapper**. You can omit the `/B` parameter in this case.

Expect startup tasks to run more than once

Not all role recycles include a reboot, but all role recycles include running all startup tasks. This means that startup tasks must be able to run multiple times between reboots without any problems. This is discussed in the [preceding section](#).

Use local storage to store files that must be accessed in the role

If you want to copy or create a file during your startup task that is then accessible to your role, then that file must be placed in local storage. See the [preceding section](#).

Next steps

[Review the cloud service model and package](#)

Learn more about how [Tasks](#) work.

[Create and deploy](#) your cloud service package.

Install .NET on Azure Cloud Services roles

11/7/2018 • 6 minutes to read • [Edit Online](#)

This article describes how to install versions of .NET Framework that don't come with the Azure Guest OS. You can use .NET on the Guest OS to configure your cloud service web and worker roles.

For example, you can install .NET 4.6.2 on the Guest OS family 4, which doesn't come with any release of .NET 4.6. (The Guest OS family 5 does come with .NET 4.6.) For the latest information on the Azure Guest OS releases, see the [Azure Guest OS release news](#).

IMPORTANT

The Azure SDK 2.9 contains a restriction on deploying .NET 4.6 on the Guest OS family 4 or earlier. A fix for the restriction is available on the [Microsoft Docs](#) site.

To install .NET on your web and worker roles, include the .NET web installer as part of your cloud service project. Start the installer as part of the role's startup tasks.

Add the .NET installer to your project

To download the web installer for the .NET Framework, choose the version that you want to install:

- [.NET 4.7.2 web installer](#)
- [.NET 4.6.2 web installer](#)

To add the installer for a *web* role:

1. In **Solution Explorer**, under **Roles** in your cloud service project, right-click your *web* role and select **Add > New Folder**. Create a folder named **bin**.
2. Right-click the bin folder and select **Add > Existing Item**. Select the .NET installer and add it to the bin folder.

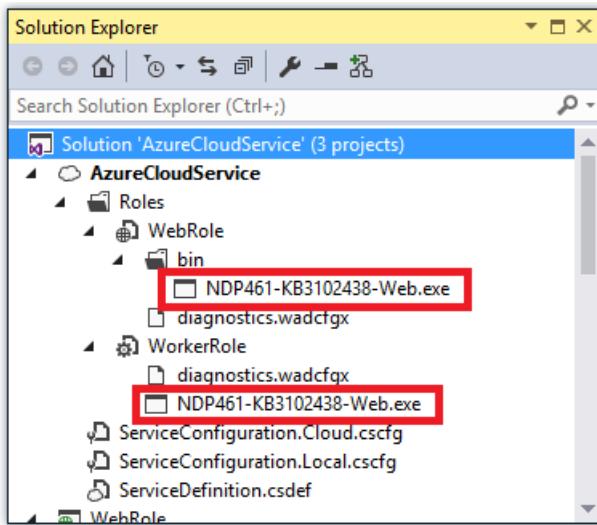
To add the installer for a *worker* role:

- Right-click your *worker* role and select **Add > Existing Item**. Select the .NET installer and add it to the role.

When files are added in this way to the role content folder, they're automatically added to your cloud service package. The files are then deployed to a consistent location on the virtual machine. Repeat this process for each web and worker role in your cloud service so that all roles have a copy of the installer.

NOTE

You should install .NET 4.6.2 on your cloud service role even if your application targets .NET 4.6. The Guest OS includes the Knowledge Base [update 3098779](#) and [update 3097997](#). Issues can occur when you run your .NET applications if .NET 4.6 is installed on top of the Knowledge Base updates. To avoid these issues, install .NET 4.6.2 rather than version 4.6. For more information, see the [Knowledge Base article 3118750](#) and [4340191](#).



Define startup tasks for your roles

You can use startup tasks to perform operations before a role starts. Installing the .NET Framework as part of the startup task ensures that the framework is installed before any application code is run. For more information on startup tasks, see [Run startup tasks in Azure](#).

1. Add the following content to the ServiceDefinition.csdef file under the **WebRole** or **WorkerRole** node for all roles:

```
<LocalResources>
    <LocalStorage name="NETFXInstall" sizeInMB="1024" cleanOnRoleRecycle="false" />
</LocalResources>
<Startup>
    <Task commandLine="install.cmd" executionContext="elevated" taskType="simple">
        <Environment>
            <Variable name="PathToNETFXInstall">
                <RoleInstanceValue
                    xpath="/RoleEnvironment/CurrentInstance/LocalResources/LocalResource[@name='NETFXInstall']/@path" />
            </Variable>
            <Variable name="ComputeEmulatorRunning">
                <RoleInstanceValue xpath="/RoleEnvironment/Deployment/@emulated" />
            </Variable>
        </Environment>
    </Task>
</Startup>
```

The preceding configuration runs the console command `install.cmd` with administrator privileges to install the .NET Framework. The configuration also creates a **LocalStorage** element named **NETFXInstall**. The startup script sets the temp folder to use this local storage resource.

IMPORTANT

To ensure correct installation of the framework, set the size of this resource to at least 1,024 MB.

For more information about startup tasks, see [Common Azure Cloud Services startup tasks](#).

2. Create a file named **install.cmd** and add the following install script to the file.

The script checks whether the specified version of the .NET Framework is already installed on the machine by querying the registry. If the .NET version is not installed, then the .NET web installer is opened. To help troubleshoot any issues, the script logs all activity to the file `startuptasklog-(current date and time).txt` that is stored in **InstallLogs** local storage.

IMPORTANT

Use a basic text editor like Windows Notepad to create the install.cmd file. If you use Visual Studio to create a text file and change the extension to .cmd, the file might still contain a UTF-8 byte order mark. This mark can cause an error when the first line of the script is run. To avoid this error, make the first line of the script a REM statement that can be skipped by the byte order processing.

```
REM Set the value of netfx to install appropriate .NET Framework.  
REM ***** To install .NET 4.5.2 set the variable netfx to "NDP452" *****  
REM ***** To install .NET 4.6 set the variable netfx to "NDP46" *****  
REM ***** To install .NET 4.6.1 set the variable netfx to "NDP461" *****  
http://go.microsoft.com/fwlink/?LinkId=671729  
REM ***** To install .NET 4.6.2 set the variable netfx to "NDP462" *****  
https://www.microsoft.com/download/details.aspx?id=53345  
REM ***** To install .NET 4.7 set the variable netfx to "NDP47" *****  
REM ***** To install .NET 4.7.1 set the variable netfx to "NDP471" *****  
http://go.microsoft.com/fwlink/?LinkId=852095  
REM ***** To install .NET 4.7.2 set the variable netfx to "NDP472" *****  
http://go.microsoft.com/fwlink/?LinkId=863262  
set netfx="NDP472"  
  
REM ***** Set script start timestamp *****  
set timehour=%time:~0,2%  
set timestamp=%date:~-4,4%%date:~-10,2%%date:~-7,2%-%timehour: =0%%time:~3,2%  
set "log=install.cmd started %timestamp%."  
  
REM ***** Exit script if running in Emulator *****  
if "%ComputeEmulatorRunning%"=="true" goto exit  
  
REM ***** Needed to correctly install .NET 4.6.1, otherwise you may see an out of disk space error *****  
set TMP=%PathToNETFXInstall%  
set TEMP=%PathToNETFXInstall%  
  
REM ***** Setup .NET filenames and registry keys *****  
if %netfx%=="NDP472" goto NDP472  
if %netfx%=="NDP471" goto NDP471  
if %netfx%=="NDP47" goto NDP47  
if %netfx%=="NDP462" goto NDP462  
if %netfx%=="NDP461" goto NDP461  
if %netfx%=="NDP46" goto NDP46  
  
set "netfxinstallfile=NDP452-KB2901954-Web.exe"  
set netfxregkey="0x5cbf5"  
goto logtimestamp  
  
:NDP46  
set "netfxinstallfile=NDP46-KB3045560-Web.exe"  
set netfxregkey="0x6004f"  
goto logtimestamp  
  
:NDP461  
set "netfxinstallfile=NDP461-KB3102438-Web.exe"  
set netfxregkey="0x6040e"  
goto logtimestamp  
  
:NDP462  
set "netfxinstallfile=NDP462-KB3151802-Web.exe"  
set netfxregkey="0x60632"  
goto logtimestamp  
  
:NDP47  
set "netfxinstallfile=NDP47-KB3186500-Web.exe"  
set netfxregkey="0x707FE"  
goto logtimestamp
```

```

:NDP471
set "netfxinstallfile=NDP471-KB4033344-Web.exe"
set netfxregkey="0x709fc"
goto logtimestamp

:NDP472
set "netfxinstallfile=NDP472-KB4054531-Web.exe"
set netfxregkey="0x70BF6"
goto logtimestamp

:logtimestamp
REM ***** Setup LogFile with timestamp *****
md "%PathToNETFXInstall%\log"
set startuptasklog="%PathToNETFXInstall%\log\startuptasklog-%timestamp%.txt"
set netfxinstallerlog="%PathToNETFXInstall%\log\NetFXInstallerLog-%timestamp%"
echo %log% >> %startuptasklog%
echo Logfile generated at: %startuptasklog% >> %startuptasklog%
echo TMP set to: %TMP% >> %startuptasklog%
echo TEMP set to: %TEMP% >> %startuptasklog%

REM ***** Check if .NET is installed *****
echo Checking if .NET (%netfx%) is installed >> %startuptasklog%
set /A netfxregkeydecimal=%netfxregkey%
set foundkey=0
FOR /F "usebackq skip=2 tokens=1,2*" %%A in (`reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full" /v Release 2>nul`) do @set /A foundkey=%%%C
echo Minimum required key: %netfxregkeydecimal% -- found key: %foundkey% >> %startuptasklog%
if %foundkey% GEQ %netfxregkeydecimal% goto installed

REM ***** Installing .NET *****
echo Installing .NET with commandline: start /wait %~dp0%netfxinstallfile% /q /serialdownload /log %netfxinstallerlog% /chainingpackage "CloudService Startup Task" >> %startuptasklog%
start /wait %~dp0%netfxinstallfile% /q /serialdownload /log %netfxinstallerlog% /chainingpackage "CloudService Startup Task" >> %startuptasklog% 2>&1
if %ERRORLEVEL%== 0 goto installed
    echo .NET installer exited with code %ERRORLEVEL% >> %startuptasklog%
    if %ERRORLEVEL%== 3010 goto restart
    if %ERRORLEVEL%== 1641 goto restart
    echo .NET (%netfx%) install failed with Error Code %ERRORLEVEL%. Further logs can be found in %netfxinstallerlog% >> %startuptasklog%

:restart
echo Restarting to complete .NET (%netfx%) installation >> %startuptasklog%
EXIT /B %ERRORLEVEL%

:installed
echo .NET (%netfx%) is installed >> %startuptasklog%

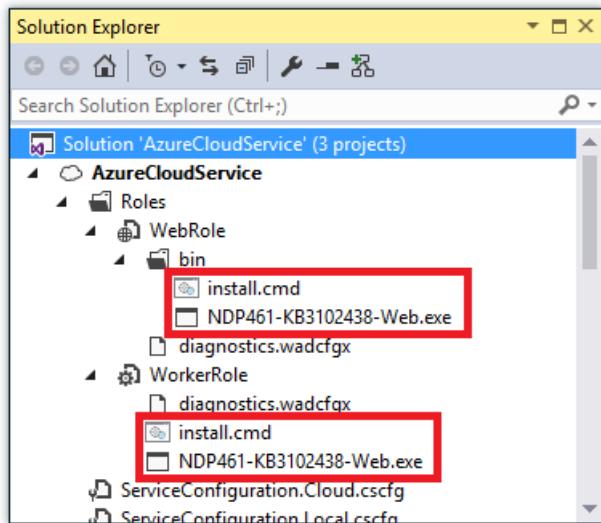
:end
echo install.cmd completed: %date:~-4,4%%date:~-10,2%%date:~-7,2%-%timehour: =0%%time:~3,2% >> %startuptasklog%

:exit
EXIT /B 0

```

3. Add the install.cmd file to each role by using **Add > Existing Item** in **Solution Explorer** as described earlier in this topic.

After this step is complete, all roles should have the .NET installer file and the install.cmd file.



Configure Diagnostics to transfer startup logs to Blob storage

To simplify troubleshooting installation issues, you can configure Azure Diagnostics to transfer any log files generated by the startup script or the .NET installer to Azure Blob storage. By using this approach, you can view the logs by downloading the log files from Blob storage rather than having to remote desktop into the role.

To configure Diagnostics, open the diagnostics.wadcfgx file and add the following content under the **Directories** node:

```
<DataSources>
  <DirectoryConfiguration containerName="netfx-install">
    <LocalResource name="NETFXInstall" relativePath="log"/>
  </DirectoryConfiguration>
</DataSources>
```

This XML configures Diagnostics to transfer the files in the log directory in the **NETFXInstall** resource to the Diagnostics storage account in the **netfx-install** blob container.

Deploy your cloud service

When you deploy your cloud service, the startup tasks install the .NET Framework if it's not already installed. Your cloud service roles are in the *busy* state while the framework is being installed. If the framework installation requires a restart, the service roles might also restart.

Additional resources

- [Installing the .NET Framework](#)
- [Determine which .NET Framework versions are installed](#)
- [Troubleshooting .NET Framework installations](#)

Enable Remote Desktop Connection for a Role in Azure Cloud Services

3/9/2018 • 2 minutes to read • [Edit Online](#)

Remote Desktop enables you to access the desktop of a role running in Azure. You can use a Remote Desktop connection to troubleshoot and diagnose problems with your application while it is running.

You can enable a Remote Desktop connection in your role during development by including the Remote Desktop modules in your service definition or you can choose to enable Remote Desktop through the Remote Desktop Extension. The preferred approach is to use the Remote Desktop extension as you can enable Remote Desktop even after the application is deployed without having to redeploy your application.

Configure Remote Desktop from the Azure portal

The Azure portal uses the Remote Desktop Extension approach so you can enable Remote Desktop even after the application is deployed. The **Remote Desktop** settings for your cloud service allows you to enable Remote Desktop, change the local Administrator account used to connect to the virtual machines, the certificate used in authentication and set the expiration date.

1. Click **Cloud Services**, select the name of the cloud service, and then select **Remote Desktop**.

The screenshot shows the Azure Cloud Service Overview page for a service named 'mytestcloudsvc - Staging'. The left sidebar contains several navigation items:

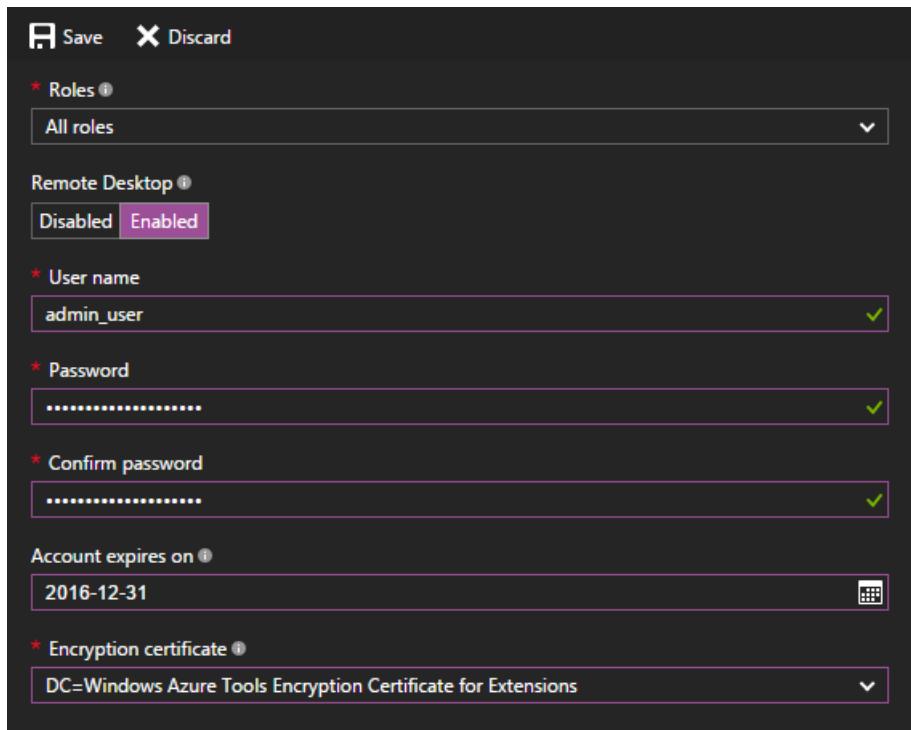
- Overview (selected)
- Activity log
- Access control (IAM)
- Diagnose and solve problems

Below the navigation is a section titled 'SETTINGS' containing the following options:

- Antimalware
- Certificates
- Configuration
- Extensions
- Instances
- Remote Desktop (highlighted with a red box)
- Scale
- Properties
- Locks

At the bottom of the sidebar is a 'SUPPORT + TROUBLESHOOTING' section with the 'New support request' option.

2. Choose whether you want to enable Remote Desktop for an individual role or for all roles, then change the value of the switcher to **Enabled**.
3. Fill in the required fields for user name, password, expiry, and certificate.



WARNING

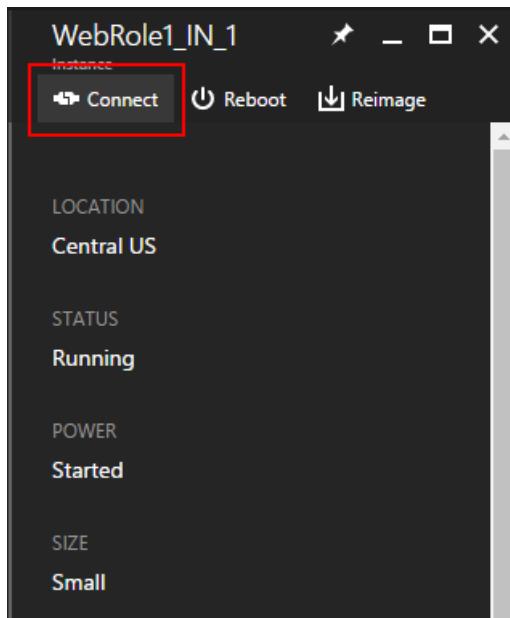
All role instances will be restarted when you first enable Remote Desktop and select **OK** (checkmark). To prevent a reboot, the certificate used to encrypt the password must be installed on the role. To prevent a restart, [upload a certificate for the cloud service](#) and then return to this dialog.

4. In **Roles**, select the role you want to update or select **All** for all roles.
5. When you finish your configuration updates, select **Save**. It will take a few moments before your role instances are ready to receive connections.

Remote into role instances

Once Remote Desktop is enabled on the roles, you can initiate a connection directly from the Azure portal:

1. Click **Instances** to open the **Instances** settings.
2. Select a role instance that has Remote Desktop configured.
3. Click **Connect** to download an RDP file for the role instance.



4. Click **Open** and then **Connect** to start the Remote Desktop connection.

NOTE

If your cloud service is sitting behind an NSG, you may need to create rules that allow traffic on ports **3389** and **20000**. Remote Desktop uses port **3389**. Cloud Service instances are load balanced, so you can't directly control which instance to connect to. The *RemoteForwarder* and *RemoteAccess* agents manage RDP traffic and allow the client to send an RDP cookie and specify an individual instance to connect to. The *RemoteForwarder* and *RemoteAccess* agents require that port **20000*** is open, which may be blocked if you have an NSG.

Additional resources

[How to Configure Cloud Services](#)

Enable Remote Desktop Connection for a Role in Azure Cloud Services using PowerShell

8/10/2018 • 4 minutes to read • [Edit Online](#)

Remote Desktop enables you to access the desktop of a role running in Azure. You can use a Remote Desktop connection to troubleshoot and diagnose problems with your application while it is running.

This article describes how to enable remote desktop on your Cloud Service Roles using PowerShell. See [How to install and configure Azure PowerShell](#) for the prerequisites needed for this article. PowerShell utilizes the Remote Desktop Extension so you can enable Remote Desktop after the application is deployed.

Configure Remote Desktop from PowerShell

The [Set-AzureServiceRemoteDesktopExtension](#) cmdlet allows you to enable Remote Desktop on specified roles or all roles of your cloud service deployment. The cmdlet lets you specify the Username and Password for the remote desktop user through the *Credential* parameter that accepts a **PSCredential** object.

If you are using PowerShell interactively, you can easily set the **PSCredential** object by calling the [Get-Credentials](#) cmdlet.

```
$remoteusercredentials = Get-Credential
```

This command displays a dialog box allowing you to enter the username and password for the remote user in a secure manner.

Since PowerShell helps in automation scenarios, you can also set up the **PSCredential** object in a way that doesn't require user interaction. First, you need to set up a secure password. You begin with specifying a plain text password convert it to a secure string using [ConvertTo-SecureString](#). Next you need to convert this secure string into an encrypted standard string using [ConvertFrom-SecureString](#). Now you can save this encrypted standard string to a file using [Set-Content](#).

You can also create a secure password file so that you don't have to type in the password every time. Also, a secure password file is better than a plain text file. Use the following PowerShell to create a secure password file:

```
ConvertTo-SecureString -String "Password123" -AsPlainText -Force | ConvertFrom-SecureString | Set-Content "password.txt"
```

IMPORTANT

When setting the password, make sure that you meet the [complexity requirements](#).

To create the credential object from the secure password file, you must read the file contents and convert them back to a secure string using [ConvertTo-SecureString](#).

The [Set-AzureServiceRemoteDesktopExtension](#) cmdlet also accepts an *Expiration* parameter, which specifies a **DateTime** at which the user account expires. For example, you could set the account to expire a few days from the current date and time.

This PowerShell example shows you how to set the Remote Desktop Extension on a cloud service:

```
$servicename = "cloudservice"
$username = "RemoteDesktopUser"
$securepassword = Get-Content -Path "password.txt" | ConvertTo-SecureString
$expiry = $(Get-Date).AddDays(1)
$credential = New-Object System.Management.Automation.PSCredential $username,$securepassword
Set-AzureServiceRemoteDesktopExtension -ServiceName $servicename -Credential $credential -Expiration $expiry
```

You can also optionally specify the deployment slot and roles that you want to enable remote desktop on. If these parameters are not specified, the cmdlet enables remote desktop on all roles in the **Production** deployment slot.

The Remote Desktop extension is associated with a deployment. If you create a new deployment for the service, you have to enable remote desktop on that deployment. If you always want to have remote desktop enabled, then you should consider integrating the PowerShell scripts into your deployment workflow.

Remote Desktop into a role instance

The [Get-AzureRemoteDesktopFile](#) cmdlet is used to remote desktop into a specific role instance of your cloud service. You can use the *LocalPath* parameter to download the RDP file locally. Or you can use the *Launch* parameter to directly launch the Remote Desktop Connection dialog to access the cloud service role instance.

```
Get-AzureRemoteDesktopFile -ServiceName $servicename -Name "WorkerRole1_IN_0" -Launch
```

Check if Remote Desktop extension is enabled on a service

The [Get-AzureServiceRemoteDesktopExtension](#) cmdlet displays that remote desktop is enabled or disabled on a service deployment. The cmdlet returns the username for the remote desktop user and the roles that the remote desktop extension is enabled for. By default, this happens on the deployment slot and you can choose to use the staging slot instead.

```
Get-AzureServiceRemoteDesktopExtension -ServiceName $servicename
```

Remove Remote Desktop extension from a service

If you have already enabled the remote desktop extension on a deployment, and need to update the remote desktop settings, first remove the extension. And enable it again with the new settings. For example, if you want to set a new password for the remote user account, or the account expired. Doing this is required on existing deployments that have the remote desktop extension enabled. For new deployments, you can simply apply the extension directly.

To remove the remote desktop extension from the deployment, you can use the [Remove-AzureServiceRemoteDesktopExtension](#) cmdlet. You can also optionally specify the deployment slot and role from which you want to remove the remote desktop extension.

```
Remove-AzureServiceRemoteDesktopExtension -ServiceName $servicename -UninstallConfiguration
```

NOTE

To completely remove the extension configuration, you should call the *remove* cmdlet with the **UninstallConfiguration** parameter.

The **UninstallConfiguration** parameter uninstalls any extension configuration that is applied to the service. Every extension configuration is associated with the service configuration. Calling the *remove* cmdlet without **UninstallConfiguration** disassociates the `[deployment]` from the extension configuration, thus effectively removing the extension. However, the extension configuration remains associated with the service.

Additional resources

[How to Configure Cloud Services](#)

Enable Remote Desktop Connection for a Role in Azure Cloud Services using Visual Studio

9/10/2018 • 6 minutes to read • [Edit Online](#)

Remote Desktop enables you to access the desktop of a role running in Azure. You can use a Remote Desktop connection to troubleshoot and diagnose problems with your application while it is running.

The publish wizard that Visual Studio provides for cloud services includes an option to enable Remote Desktop during the publishing process, using credentials that you provide. Using this option is suitable when using Visual Studio 2017 version 15.4 and earlier.

With Visual Studio 2017 version 15.5 and later, however, it's recommended that you avoid enabling Remote Desktop through the publish wizard unless you're working only as a single developer. For any situation in which the project might be opened by other developers, you instead enable Remote Desktop through the Azure portal, through PowerShell, or from a release pipeline in a continuous deployment workflow. This recommendation is due to a change in how Visual Studio communicates with Remote Desktop on the cloud service VM, as is explained in this article.

Configure Remote Desktop through Visual Studio 2017 version 15.4 and earlier

When using Visual Studio 2017 version 15.4 and earlier, you can use the **Enable Remote Desktop for all roles** option in the publish wizard. You can still use the wizard with Visual Studio 2017 version 15.5 and later, but don't use the Remote Desktop option.

1. In Visual Studio, start the publish wizard by right-clicking your cloud service project in Solution Explorer and choosing **Publish**.
2. Sign into your Azure subscription if needed and select **Next**.
3. On the **Settings** page, select **Enable Remote Desktop for all roles**, then select the **Settings...** link to open the **Remote Desktop Configuration** dialog box.
4. At the bottom of the dialog box, select **More Options**. This command displays a drop-down list in which you create or choose a certificate so that you can encrypt credentials information when connecting via remote desktop.

NOTE

The certificates that you need for a remote desktop connection are different from the certificates that you use for other Azure operations. The remote access certificate must have a private key.

5. Select a certificate from the list or choose <**Create...**>. If creating a new certificate, provide a friendly name for the new certificate when prompted and select **OK**. The new certificate appears in the drop-down list box.
6. Provide a user name and a password. You can't use an existing account. Don't use "Administrator" as the user name for the new account.
7. Choose a date on which the account will expire and after which Remote Desktop connections will be blocked.

8. After you've provided all the required information, select **OK**. Visual Studio adds the Remote Desktop settings to your project's `.cscfg` and `.csdef` files, including the password that's encrypted using the chosen certificate.
9. Complete any remaining steps using the **Next** button, then select **Publish** when you're ready to publish your cloud service. If you're not ready to publish, select **Cancel** and answer **Yes** when prompted to save changes. You can publish your cloud service later with these settings.

Configure Remote Desktop when using Visual Studio 2017 version 15.5 and later

With Visual Studio 2017 version 15.5 and later, you can still use the publish wizard with a cloud service project. You can also use the **Enable Remote Desktop for all roles** option if you're working only as a single developer.

If you're working as part of a team, you should instead enable remote desktop on the Azure cloud service by using either the [Azure portal](#) or [PowerShell](#).

This recommendation is due to a change in how Visual Studio 2017 version 15.5 and later communicates with the cloud service VM. When enabling Remote Desktop through the publish wizard, earlier versions of Visual Studio communicate with the VM through what's called the "RDP plugin." Visual Studio 2017 version 15.5 and later communicates instead using the "RDP extension" that is more secure and more flexible. This change also aligns with the fact that the Azure portal and PowerShell methods to enable Remote Desktop also use the RDP extension.

When Visual Studio communicates with the RDP extension, it transmits a plain text password over SSL. However, the project's configuration files store only an encrypted password, which can be decrypted into plain text only with the local certificate that was originally used to encrypt it.

If you deploy the cloud service project from the same development computer each time, then that local certificate is available. In this case, you can still use the **Enable Remote Desktop for all roles** option in the publish wizard.

If you or other developers want to deploy the cloud service project from different computers, however, then those other computers won't have the necessary certificate to decrypt the password. As a result, you see the following error message:

```
Applying remote desktop protocol (RDP) extension.  
Certificate with thumbprint [thumbprint] doesn't exist.
```

You could change the password every time you deploy the cloud service, but that action becomes inconvenient for everyone who needs to use Remote Desktop.

If you're sharing the project with a team, then, it's best to clear the option in the publish wizard and instead enable Remote Desktop directly through the [Azure portal](#) or by using [PowerShell](#).

Deploying from a build server with Visual Studio 2017 version 15.5 and later

You can deploy a cloud service project from a build server (for example, with Azure DevOps Services) on which Visual Studio 2017 version 15.5 or later is installed in the build agent. With this arrangement, deployment happens from the same computer on which the encryption certificate is available.

To use the RDP extension from Azure DevOps Services, include the following details in your build pipeline:

1. Include `/p:ForceRDPExtensionOverPlugin=true` in your MSBuild arguments to make sure the deployment works with the RDP extension rather than the RDP plugin. For example:

```
msbuild AzureCloudService5.ccproj /t:Publish /p:TargetProfile=Cloud /p:DebugType=None  
/p:SkipInvalidConfigurations=true /p:ForceRDPExtensionOverPlugin=true
```

2. After your build steps, add the **Azure Cloud Service Deployment** step and set its properties.
3. After the deployment step, add an **Azure Powershell** step, set its **Display name** property to "Azure Deployment: Enable RDP Extension" (or another suitable name), and select your appropriate Azure subscription.
4. Set **Script Type** to "Inline" and paste the code below into the **Inline Script** field. (You can also create a **.ps1** file in your project with this script, set **Script Type** to "Script File Path", and set **Script Path** to point to the file.)

```
Param(  
    [Parameter(Mandatory=$True)]  
    [string]$username,  
  
    [Parameter(Mandatory=$True)]  
    [string]$password,  
  
    [Parameter(Mandatory=$True)]  
    [string]$serviceName,  
  
    [Datetime]$expiry = ($(Get-Date).AddYears(1))  
)  
  
Write-Host "Service Name: $serviceName"  
Write-Host "User Name: $username"  
Write-Host "Expiry: $expiry"  
  
$securepassword = ConvertTo-SecureString -String $password -AsPlainText -Force  
$credential = New-Object System.Management.Automation.PSCredential $username,$securepassword  
  
# Try to remote existing RDP Extensions  
try  
{  
    $existingRDPExtension = Get-AzureServiceRemoteDesktopExtension -ServiceName $servicename  
    if ($existingRDPExtension -ne $null)  
    {  
        Remove-AzureServiceRemoteDesktopExtension -ServiceName $servicename -UninstallConfiguration  
    }  
}  
catch  
{  
}  
  
Set-AzureServiceRemoteDesktopExtension -ServiceName $servicename -Credential $credential -Expiration  
$expiry -Verbose
```

Connect to an Azure Role by using Remote Desktop

After you publish your cloud service on Azure and have enabled Remote Desktop, you can use Visual Studio Server Explorer to log into the cloud service VM:

1. In Server Explorer, expand the **Azure** node, and then expand the node for a cloud service and one of its roles to display a list of instances.
2. Right-click an instance node and select **Connect Using Remote Desktop**.
3. Enter the user name and password that you created previously. You are now logged into your remote session.

Additional resources

[How to Configure Cloud Services](#)

How to create and deploy a cloud service

9/10/2018 • 3 minutes to read • [Edit Online](#)

The Azure portal provides two ways for you to create and deploy a cloud service: *Quick Create* and *Custom Create*.

This article explains how to use the Quick Create method to create a new cloud service and then use **Upload** to upload and deploy a cloud service package in Azure. When you use this method, the Azure portal makes available convenient links for completing all requirements as you go. If you're ready to deploy your cloud service when you create it, you can do both at the same time using Custom Create.

NOTE

If you plan to publish your cloud service from Azure DevOps, use Quick Create, and then set up Azure DevOps publishing from the Azure Quickstart or the dashboard. For more information, see [Continuous Delivery to Azure by Using Azure DevOps](#), or see help for the **Quick Start** page.

Concepts

Three components are required to deploy an application as a cloud service in Azure:

- **Service Definition**

The cloud service definition file (.csdef) defines the service model, including the number of roles.

- **Service Configuration**

The cloud service configuration file (.cscfg) provides configuration settings for the cloud service and individual roles, including the number of role instances.

- **Service Package**

The service package (.cspkg) contains the application code and configurations and the service definition file.

You can learn more about these and how to create a package [here](#).

Prepare your app

Before you can deploy a cloud service, you must create the cloud service package (.cspkg) from your application code and a cloud service configuration file (.cscfg). The Azure SDK provides tools for preparing these required deployment files. You can install the SDK from the [Azure Downloads](#) page, in the language in which you prefer to develop your application code.

Three cloud service features require special configurations before you export a service package:

- If you want to deploy a cloud service that uses Secure Sockets Layer (SSL) for data encryption, [configure your application](#) for SSL.
- If you want to configure Remote Desktop connections to role instances, [configure the roles](#) for Remote Desktop.
- If you want to configure verbose monitoring for your cloud service, enable Azure Diagnostics for the cloud service. *Minimal monitoring* (the default monitoring level) uses performance counters gathered from the host operating systems for role instances (virtual machines). *Verbose monitoring* gathers additional metrics based on performance data within the role instances to enable closer analysis of issues that occur during application processing. To find out how to enable Azure Diagnostics, see [Enabling diagnostics in Azure](#).

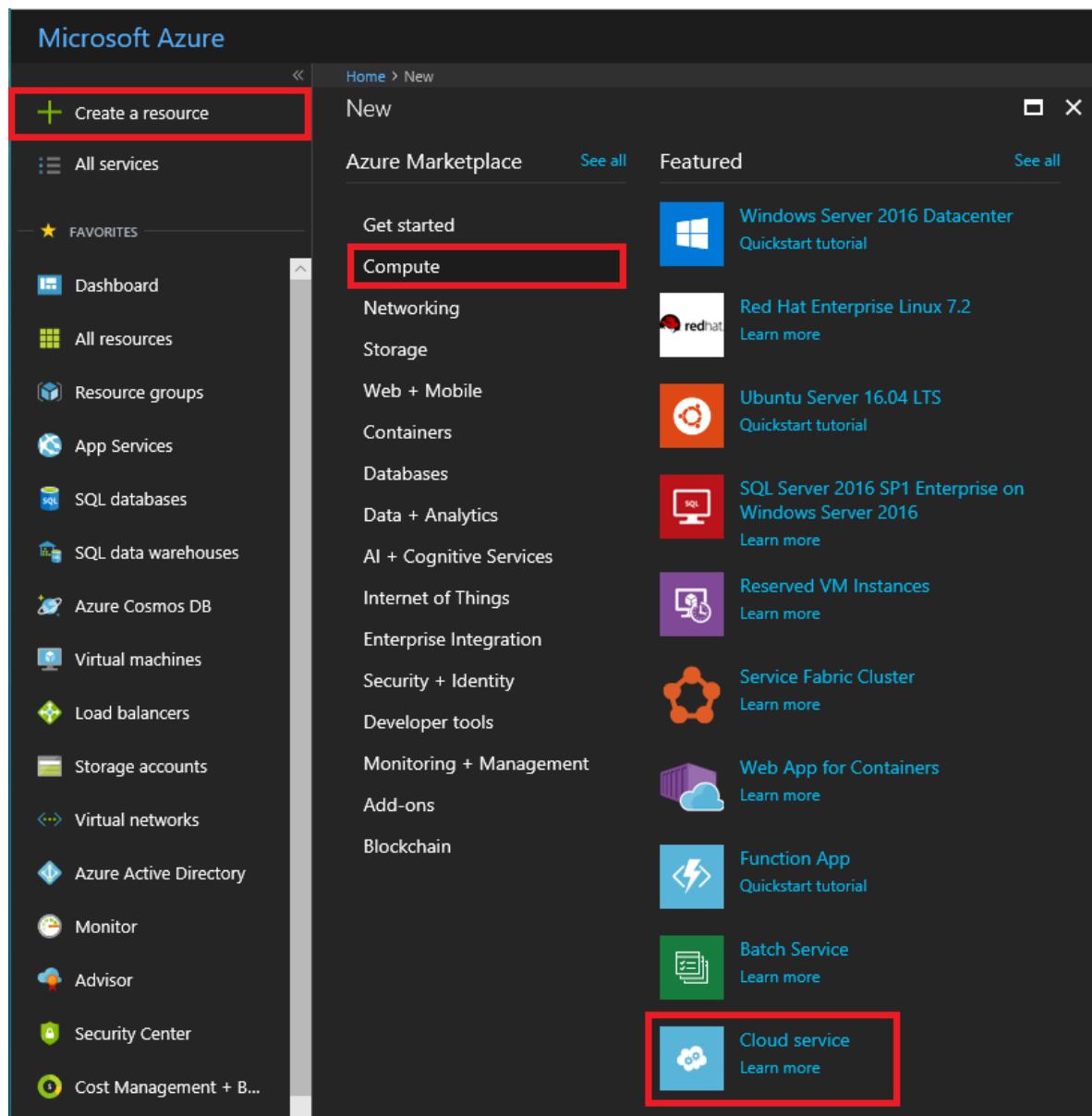
To create a cloud service with deployments of web roles or worker roles, you must [create the service package](#).

Before you begin

- If you haven't installed the Azure SDK, click **Install Azure SDK** to open the [Azure Downloads page](#), and then download the SDK for the language in which you prefer to develop your code. (You'll have an opportunity to do this later.)
- If any role instances require a certificate, create the certificates. Cloud services require a .pfx file with a private key. You can upload the certificates to Azure as you create and deploy the cloud service.

Create and deploy

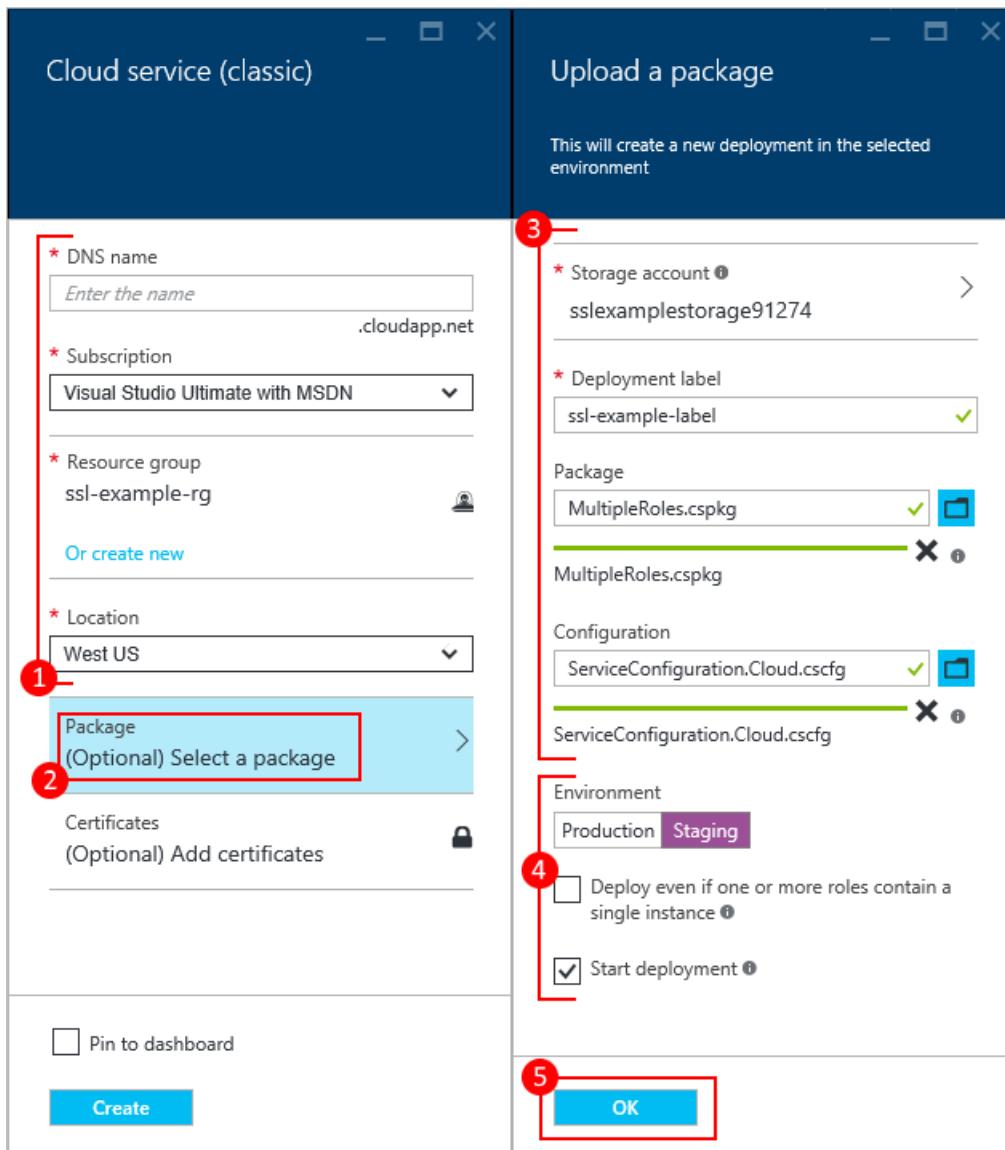
1. Log in to the [Azure portal](#).
2. Click **Create a resource > Compute**, and then scroll down to and click **Cloud Service**.



3. In the new **Cloud Service** pane, enter a value for the **DNS name**.
4. Create a new **Resource Group** or select an existing one.
5. Select a **Location**.
6. Click **Package**. This opens the **Upload a package** pane. Fill in the required fields. If any of your roles contain a single instance, ensure **Deploy even if one or more roles contain a single instance** is selected.
7. Make sure that **Start deployment** is selected.

8. Click **OK** which will close the **Upload a package** pane.

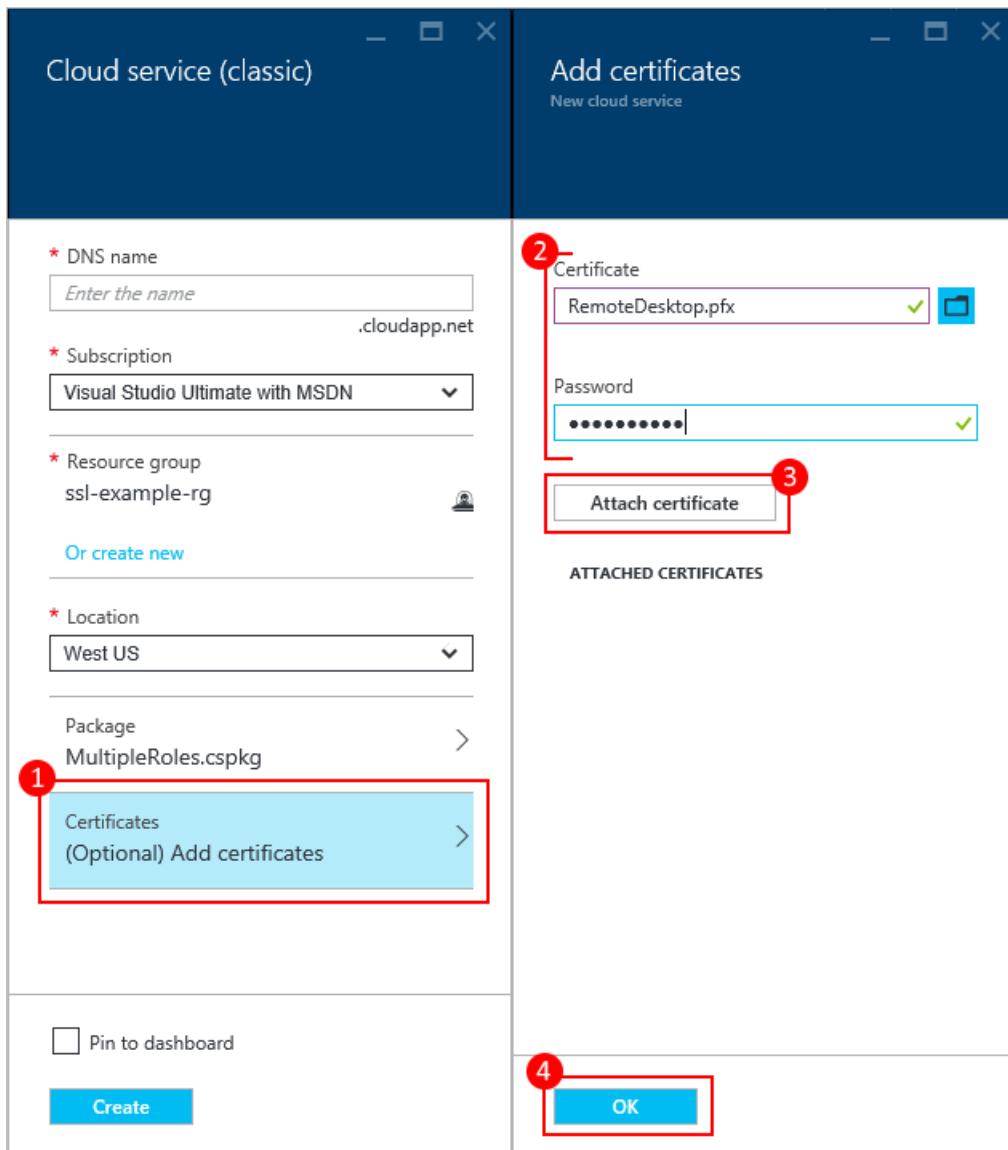
9. If you do not have any certificates to add, click **Create**.



Upload a certificate

If your deployment package was [configured to use certificates](#), you can upload the certificate now.

1. Select **Certificates**, and on the **Add certificates** pane, select the SSL certificate .pfx file, and then provide the **Password** for the certificate,
2. Click **Attach certificate**, and then click **OK** on the **Add certificates** pane.
3. Click **Create** on the **Cloud Service** pane. When the deployment has reached the **Ready** status, you can proceed to the next steps.



Verify your deployment completed successfully

1. Click the cloud service instance.

The status should show that the service is **Running**.

2. Under **Essentials**, click the **Site URL** to open your cloud service in a web browser.

freshnessreport - Staging
Cloud service (classic)

Settings Staging slot Update Stop Swap Delete

Essentials ^

Resource group
freshnessreport

Status
Running

Location
West US

Subscription name
Visual Studio Ultimate with MSDN

Subscription ID
c98 < guid >

Deployment name
3a2 < guid >

Deployment label
FreshnessReportCloudService - 4/12/2016...

Deployment ID
d5d b1. < guid >

Deployment label
a 12b

Site URL
http://b < guid > 9...

Public IP addresses
5 <IP> 6

Deployment label
FreshnessReportCloudService - 4/12/2016...

Deployment ID
d5d b1. < guid >

All settings →

Add tiles +

Roles and instances

Next steps

- General configuration of your cloud service.
- Configure a custom domain name.
- Manage your cloud service.
- Configure ssl certificates.

Use an Azure PowerShell command to create an empty cloud service container

11/7/2018 • 2 minutes to read • [Edit Online](#)

This article explains how to quickly create a Cloud Services container using Azure PowerShell cmdlets. Please follow the steps below:

1. Install the Microsoft Azure PowerShell cmdlet from the [Azure PowerShell downloads](#) page.
2. Open the PowerShell command prompt.
3. Use the [Add-AzureAccount](#) to sign in.

NOTE

For further instruction on installing the Azure PowerShell cmdlet and connecting to your Azure subscription, refer to [How to install and configure Azure PowerShell](#).

4. Use the **New-AzureService** cmdlet to create an empty Azure cloud service container.

```
New-AzureService [-ServiceName] <String> [-AffinityGroup] <String> [[-Label] <String>] [[-Description]
<String>] [[-ReverseDnsFqdn] <String>] [<CommonParameters>]
New-AzureService [-ServiceName] <String> [-Location] <String> [[-Label] <String>] [[-Description]
<String>] [[-ReverseDnsFqdn] <String>] [<CommonParameters>]
```

5. Follow this example to invoke the cmdlet:

```
New-AzureService -ServiceName "mytestcloudservice" -Location "Central US" -Label "mytestcloudservice"
```

For more information about creating the Azure cloud service, run:

```
Get-help New-AzureService
```

Next steps

- To manage the cloud service deployment, refer to the [Get-AzureService](#), [Remove-AzureService](#), and [Set-AzureService](#) commands. You may also refer to [How to configure cloud services](#) for further information.
- To publish your cloud service project to Azure, refer to the **PublishCloudService.ps1** code sample from [archived cloud services repository](#).

Configuring a custom domain name for an Azure cloud service

11/7/2018 • 6 minutes to read • [Edit Online](#)

When you create a Cloud Service, Azure assigns it to a subdomain of **cloudapp.net**. For example, if your Cloud Service is named "contoso", your users will be able to access your application on a URL like <http://contoso.cloudapp.net>. Azure also assigns a virtual IP address.

However, you can also expose your application on your own domain name, such as **contoso.com**. This article explains how to reserve or configure a custom domain name for Cloud Service web roles.

Do you already understand what CNAME and A records are? [Jump past the explanation](#).

NOTE

The procedures in this task apply to Azure Cloud Services. For App Services, see [Map an existing custom DNS name to Azure Web Apps](#). For storage accounts, see [Configure a custom domain name for your Azure Blob storage endpoint](#).

TIP

Get going faster--use the NEW Azure [guided walkthrough](#)! It makes associating a custom domain name AND securing communication (SSL) with Azure Cloud Services or Azure Websites a snap.

Understand CNAME and A records

CNAME (or alias records) and A records both allow you to associate a domain name with a specific server (or service in this case,) however they work differently. There are also some specific considerations when using A records with Azure Cloud services that you should consider before deciding which to use.

CNAME or Alias record

A CNAME record maps a *specific* domain, such as **contoso.com** or **www.contoso.com**, to a canonical domain name. In this case, the canonical domain name is the **[myapp].cloudapp.net** domain name of your Azure hosted application. Once created, the CNAME creates an alias for the **[myapp].cloudapp.net**. The CNAME entry will resolve to the IP address of your **[myapp].cloudapp.net** service automatically, so if the IP address of the cloud service changes, you do not have to take any action.

NOTE

Some domain registrars only allow you to map subdomains when using a CNAME record, such as www.contoso.com, and not root names, such as contoso.com. For more information on CNAME records, see the documentation provided by your registrar, [the Wikipedia entry on CNAME record](#), or the [IETF Domain Names - Implementation and Specification](#) document.

A record

An A record maps a domain, such as **contoso.com** or **www.contoso.com**, or a *wildcard domain* such as ***.contoso.com**, to an IP address. In the case of an Azure Cloud Service, the virtual IP of the service. So the main benefit of an A record over a CNAME record is that you can have one entry that uses a wildcard, such as ***.contoso.com**, which would handle requests for multiple sub-domains such as **mail.contoso.com**, **login.contoso.com**, or **www.contoso.com**.

NOTE

Since an A record is mapped to a static IP address, it cannot automatically resolve changes to the IP address of your Cloud Service. The IP address used by your Cloud Service is allocated the first time you deploy to an empty slot (either production or staging.) If you delete the deployment for the slot, the IP address is released by Azure and any future deployments to the slot may be given a new IP address.

Conveniently, the IP address of a given deployment slot (production or staging) is persisted when swapping between staging and production deployments or performing an in-place upgrade of an existing deployment. For more information on performing these actions, see [How to manage cloud services](#).

Add a CNAME record for your custom domain

To create a CNAME record, you must add a new entry in the DNS table for your custom domain by using the tools provided by your registrar. Each registrar has a similar but slightly different method of specifying a CNAME record, but the concepts are the same.

1. Use one of these methods to find the **.cloudapp.net** domain name assigned to your cloud service.

- Login to the [Azure portal](#), select your cloud service, look at the **Essentials** section and then find the **Site URL** entry.

The screenshot shows the Azure portal interface for managing a cloud service. The title bar says "TestCloudService - Staging" and "Cloud service (classic)". The main area is titled "Essentials". On the left, there's a sidebar with "Resource group: cloudservice1", "Status: Running", "Location: West US", "Subscription name: Visual Studio Ultimate with MSDN", and "Subscription ID: c98108fe-9999-9999-9999-594c31743d5d". The right side shows "Site URL: http://14b6a80c22e14bc99ede878f01a147...", "Public IP addresses: 13.00.000.000", "Deployment name: cc42eb1234eb45aaa2db590104232cab", "Deployment label: AzureCloudServicePython - 7/27/2016 4:21...", and "Deployment ID: 14b6a80c221145abc322178f01a147d0". A blue button at the bottom right says "All settings →".

OR

- Install and configure [Azure Powershell](#), and then use the following command:

```
Get-AzureDeployment -ServiceName yourservicename | Select Url
```

Save the domain name used in the URL returned by either method, as you will need it when creating a CNAME record.

2. Log on to your DNS registrar's website and go to the page for managing DNS. Look for links or areas of the site labeled as **Domain Name**, **DNS**, or **Name Server Management**.
3. Now find where you can select or enter CNAME's. You may have to select the record type from a drop down, or go to an advanced settings page. You should look for the words **CNAME**, **Alias**, or **Subdomains**.
4. You must also provide the domain or subdomain alias for the CNAME, such as **www** if you want to create an alias for **www.customdomain.com**. If you want to create an alias for the root domain, it may be listed as the '@' symbol in your registrar's DNS tools.
5. Then, you must provide a canonical host name, which is your application's **.cloudapp.net** domain in this case.

For example, the following CNAME record forwards all traffic from **www.contoso.com** to **contoso.cloudapp.net**, the custom domain name of your deployed application:

ALIAS/HOST NAME/SUBDOMAIN	CANONICAL DOMAIN
www	contoso.cloudapp.net

NOTE

A visitor of **www.contoso.com** will never see the true host (contoso.cloudapp.net), so the forwarding process is invisible to the end user.

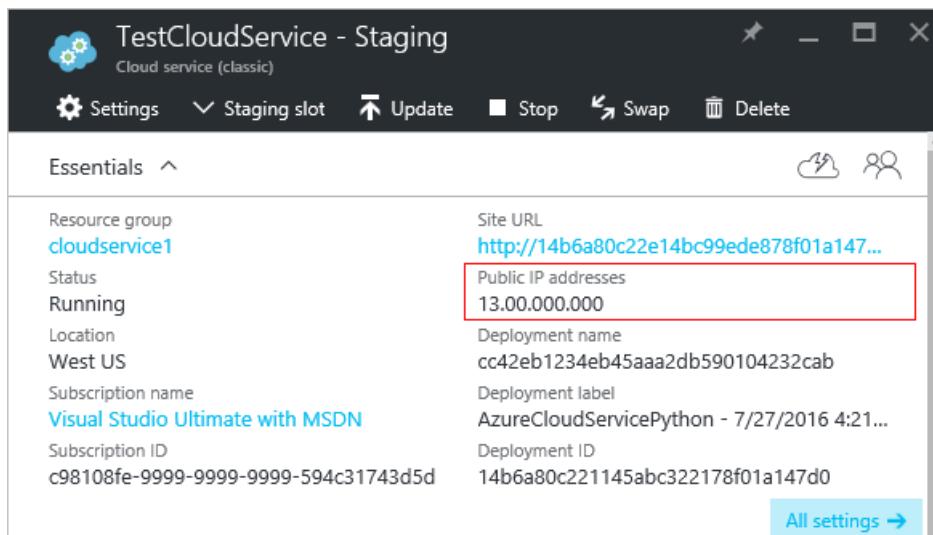
The example above only applies to traffic at the **www** subdomain. Since you cannot use wildcards with CNAME records, you must create one CNAME for each domain/subdomain. If you want to direct traffic from subdomains, such as **.contoso.com**, *to your cloudapp.net address, you can configure a **URL Redirect* or **URL Forward** entry in your DNS settings, or create an A record.*

Add an A record for your custom domain

To create an A record, you must first find the virtual IP address of your cloud service. Then add a new entry in the DNS table for your custom domain by using the tools provided by your registrar. Each registrar has a similar but slightly different method of specifying an A record, but the concepts are the same.

1. Use one of the following methods to get the IP address of your cloud service.

- Login to the [Azure portal](#), select your cloud service, look at the **Essentials** section and then find the **Public IP addresses** entry.



OR

- Install and configure [Azure Powershell](#), and then use the following command:

```
get-azurevm -servicename yourservicename | get-azureendpoint -VM ${_.VM} | select Vip
```

Save the IP address, as you will need it when creating an A record.

2. Log on to your DNS registrar's website and go to the page for managing DNS. Look for links or areas of the site labeled as **Domain Name, DNS, or Name Server Management**.
3. Now find where you can select or enter A record's. You may have to select the record type from a drop down, or go to an advanced settings page.

4. Select or enter the domain or subdomain that will use this A record. For example, select **www** if you want to create an alias for **www.customdomain.com**. If you want to create a wildcard entry for all subdomains, enter ******. ***This will cover all sub-domains such as *mail.customdomain.com, login.customdomain.com, and www.customdomain.com.***

If you want to create an A record for the root domain, it may be listed as the '@' symbol in your registrar's DNS tools.

5. Enter the IP address of your cloud service in the provided field. This associates the domain entry used in the A record with the IP address of your cloud service deployment.

For example, the following A record forwards all traffic from **contoso.com** to **137.135.70.239**, the IP address of your deployed application:

HOST NAME/SUBDOMAIN	IP ADDRESS
@	137.135.70.239

This example demonstrates creating an A record for the root domain. If you wish to create a wildcard entry to cover all subdomains, you would enter '*****' as the subdomain.

WARNING

IP addresses in Azure are dynamic by default. You will probably want to use a [reserved IP address](#) to ensure that your IP address does not change.

Next steps

- [How to Manage Cloud Services](#)
- [How to Map CDN Content to a Custom Domain](#)
- [General configuration of your cloud service.](#)
- Learn how to deploy a cloud service.
- Configure [ssl certificates](#).

Connecting Azure Cloud Services Roles to a custom AD Domain Controller hosted in Azure

11/7/2018 • 3 minutes to read • [Edit Online](#)

We will first set up a Virtual Network (VNet) in Azure. We will then add an Active Directory Domain Controller (hosted on an Azure Virtual Machine) to the VNet. Next, we will add existing cloud service roles to the pre-created VNet, then connect them to the Domain Controller.

Before we get started, couple of things to keep in mind:

1. This tutorial uses PowerShell, so make sure you have Azure PowerShell installed and ready to go. To get help with setting up Azure PowerShell, see [How to install and configure Azure PowerShell](#).
2. Your AD Domain Controller and Web/Worker Role instances need to be in the VNet.

Follow this step-by-step guide and if you run into any issues, leave us a comment at the end of the article. Someone will get back to you (yes, we do read comments).

The network that is referenced by the cloud service must be a **classic virtual network**.

Create a Virtual Network

You can create a Virtual Network in Azure using the Azure portal or PowerShell. For this tutorial, PowerShell is used. To create a virtual network using the Azure portal, see [Create a virtual network](#). The article covers creating a virtual network (Resource Manager), but you must create a virtual network (Classic) for cloud services. To do so, in the portal, select **Create a resource**, type *virtual network* in the **Search** box, and then press **Enter**. In the search results, under **Everything**, select **Virtual network**. Under **Select a deployment model**, select **Classic**, then select **Create**. You can then follow the steps in the article.

```
#Create Virtual Network

$vnetStr =
@"
<?xml version="1.0" encoding="utf-8"?>
<NetworkConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.microsoft.com/ServiceHosting/2011/07/NetworkConfiguration">
    <VirtualNetworkConfiguration>
        <VirtualNetworkSites>
            <VirtualNetworkSite name="[your-vnet-name]" Location="West US">
                <AddressSpace>
                    <AddressPrefix>[your-address-prefix]</AddressPrefix>
                </AddressSpace>
                <Subnets>
                    <Subnet name="[your-subnet-name]">
                        <AddressPrefix>[your-subnet-range]</AddressPrefix>
                    </Subnet>
                </Subnets>
            </VirtualNetworkSite>
        </VirtualNetworkSites>
    </VirtualNetworkConfiguration>
</NetworkConfiguration>
"@

$vnetConfigPath = "<path-to-vnet-config>"
Set-AzureVNetConfig -ConfigurationPath $vnetConfigPath
```

Create a Virtual Machine

Once you have completed setting up the Virtual Network, you will need to create an AD Domain Controller. For this tutorial, we will be setting up an AD Domain Controller on an Azure Virtual Machine.

To do this, create a virtual machine through PowerShell using the following commands:

```
# Initialize variables
# VNet and subnet must be classic virtual network resources, not Azure Resource Manager resources.

$vnetname = '<your-vnet-name>'
$subnetname = '<your-subnet-name>'
$vmsvc1 = '<your-hosted-service>'
$vm1 = '<your-vm-name>'
$username = '<your-username>'
$password = '<your-password>'
$affgrp = '<your-affgrp>'

# Create a VM and add it to the Virtual Network

New-AzureQuickVM -Windows -ServiceName $vmsvc1 -Name $vm1 -ImageName $imgname -AdminUsername $username -
Password $password -AffinityGroup $affgrp -SubnetNames $subnetname -VNetName $vnetname
```

Promote your Virtual Machine to a Domain Controller

To configure the Virtual Machine as an AD Domain Controller, you will need to log in to the VM and configure it.

To log in to the VM, you can get the RDP file through PowerShell, use the following commands:

```
# Get RDP file
Get-AzureRemoteDesktopFile -ServiceName $vmsvc1 -Name $vm1 -LocalPath <rdp-file-path>
```

Once you are signed in to the VM, set up your Virtual Machine as an AD Domain Controller by following the step-by-step guide on [How to set up your customer AD Domain Controller](#).

Add your Cloud Service to the Virtual Network

Next, you need to add your cloud service deployment to the new VNet. To do this, modify your cloud service cscfg by adding the relevant sections to your cscfg using Visual Studio or the editor of your choice.

```

<ServiceConfiguration serviceName="[hosted-service-name]"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfiguration" osFamily="[os-family]"
osVersion="*">
  <Role name="[role-name]">
    <Instances count="[number-of-instances]" />
  </Role>
  <NetworkConfiguration>

    <!--optional-->
    <Dns>
      <DnsServers><DnsServer name="[dns-server-name]" IPAddress="[ip-address]" /></DnsServers>
    </Dns>
    <!--optional-->

    <!--VNet settings
        VNet and subnet must be classic virtual network resources, not Azure Resource Manager resources.-->
    <VirtualNetworkSite name="[virtual-network-name]" />
    <AddressAssignments>
      <InstanceAddress roleName="[role-name]">
        <Subnets>
          <Subnet name="[subnet-name]" />
        </Subnets>
      </InstanceAddress>
    </AddressAssignments>
    <!--VNet settings-->

    </NetworkConfiguration>
  </ServiceConfiguration>

```

Next build your cloud services project and deploy it to Azure. To get help with deploying your cloud services package to Azure, see [How to Create and Deploy a Cloud Service](#)

Connect your web/worker roles to the domain

Once your cloud service project is deployed on Azure, connect your role instances to the custom AD domain using the AD Domain Extension. To add the AD Domain Extension to your existing cloud services deployment and join the custom domain, execute the following commands in PowerShell:

```

# Initialize domain variables

$domain = '<your-domain-name>'
$dmuser = '$domain\<your-username>'
$dmpswd = '<your-domain-password>'
$dmspwd = ConvertTo-SecureString $dmpswd -AsPlainText -Force
$dmcred = New-Object System.Management.Automation.PSCredential ($dmuser, $dmspwd)

# Add AD Domain Extension to the cloud service roles

Set-AzureServiceADDomainExtension -Service <your-cloud-service-hosted-service-name> -Role <your-role-name> -
Slot <staging-or-production> -DomainName $domain -Credential $dmcred -JoinOption 35

```

And that's it.

Your cloud services should be joined to your custom domain controller. If you would like to learn more about the different options available for how to configure AD Domain Extension, use the PowerShell help. A couple of examples follow:

```

help Set-AzureServiceADDomainExtension
help New-AzureServiceADDomainExtensionConfig

```

Manage Cloud Services in the Azure portal

8/10/2018 • 4 minutes to read • [Edit Online](#)

In the **Cloud Services** area of the Azure portal, you can:

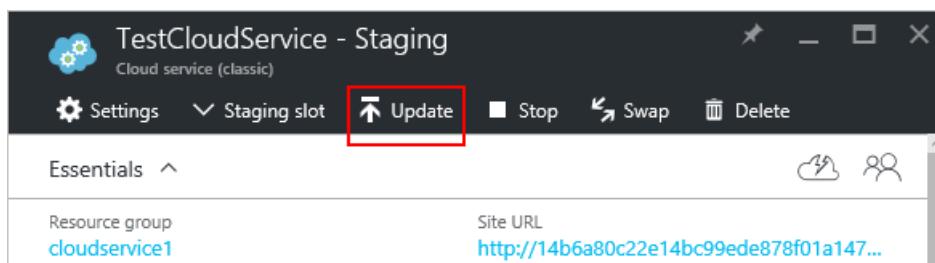
- Update a service role or a deployment.
- Promote a staged deployment to production.
- Link resources to your cloud service so that you can see the resource dependencies and scale the resources together.
- Delete a cloud service or a deployment.

For more information about how to scale your cloud service, see [Configure auto-scaling for a cloud service in the portal](#).

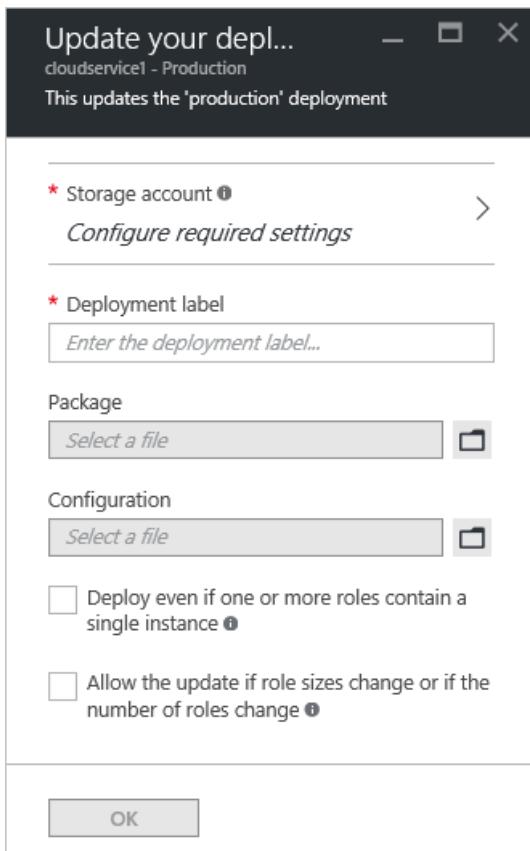
Update a cloud service role or deployment

If you need to update the application code for your cloud service, use **Update** on the cloud service blade. You can update a single role or all roles. To update, you can upload a new service package or service configuration file.

1. In the [Azure portal](#), select the cloud service you want to update. This step opens the cloud service instance blade.
2. On the blade, select **Update**.



3. Update the deployment with a new service package file (.cspkg) and service configuration file (.cscfg).



4. Optionally, update the storage account and the deployment label.
5. If any roles have only one role instance, select the **Deploy even if one or more roles contain a single instance** check box to enable the upgrade to proceed.

Azure can guarantee only 99.95 percent service availability during a cloud service update if each role has at least two role instances (virtual machines). With two role instances, one virtual machine processes client requests while the other is updated.

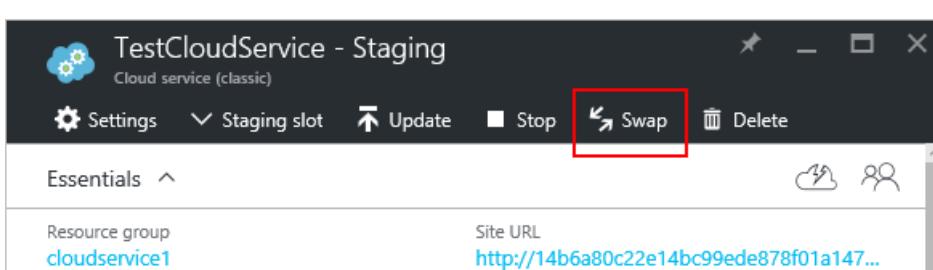
6. Select the **Start deployment** check box to apply the update after the upload of the package has finished.
7. Select **OK** to begin updating the service.

Swap deployments to promote a staged deployment to production

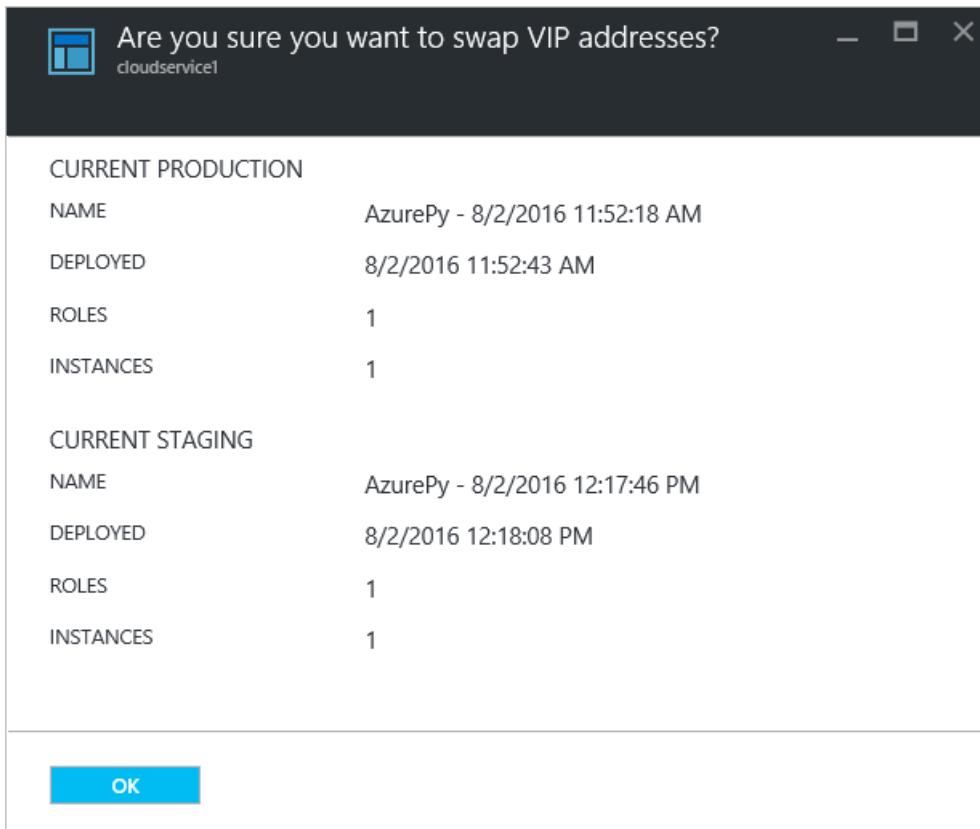
When you decide to deploy a new release of a cloud service, stage and test your new release in your cloud service staging environment. Use **Swap** to switch the URLs by which the two deployments are addressed and promote a new release to production.

You can swap deployments from the **Cloud Services** page or the dashboard.

1. In the [Azure portal](#), select the cloud service you want to update. This step opens the cloud service instance blade.
2. On the blade, select **Swap**.



3. The following confirmation prompt opens:



4. After you verify the deployment information, select **OK** to swap the deployments.

The deployment swap happens quickly because the only thing that changes is the virtual IP addresses (VIPs) for the deployments.

To save compute costs, you can delete the staging deployment after you verify that your production deployment is working as expected.

Common questions about swapping deployments

What are the prerequisites for swapping deployments?

There are two key prerequisites for a successful deployment swap:

- If you want to use a static IP address for your production slot, you must reserve one for your staging slot as well. Otherwise, the swap fails.
- All instances of your roles must be running before you can perform the swap. You can check the status of your instances on the **Overview** blade of the Azure portal. Alternatively, you can use the [Get-AzureRole](#) command in Windows PowerShell.

Note that guest OS updates and service healing operations also can cause deployment swaps to fail. For more information, see [Troubleshoot cloud service deployment problems](#).

Does a swap incur downtime for my application? How should I handle it?

As described in the previous section, a deployment swap is typically fast because it's just a configuration change in the Azure load balancer. In some cases, it can take 10 or more seconds and result in transient connection failures. To limit impact to your customers, consider implementing [client retry logic](#).

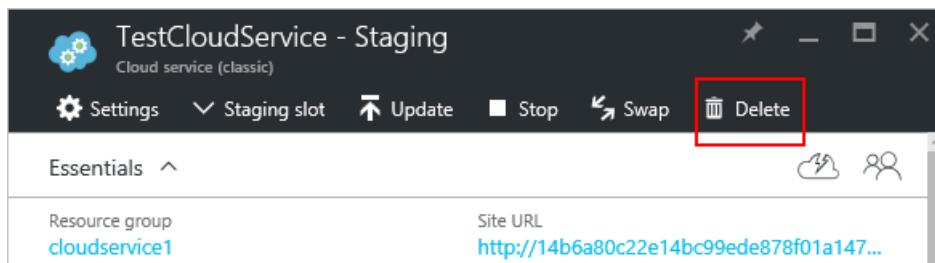
Delete deployments and a cloud service

Before you can delete a cloud service, you must delete each existing deployment.

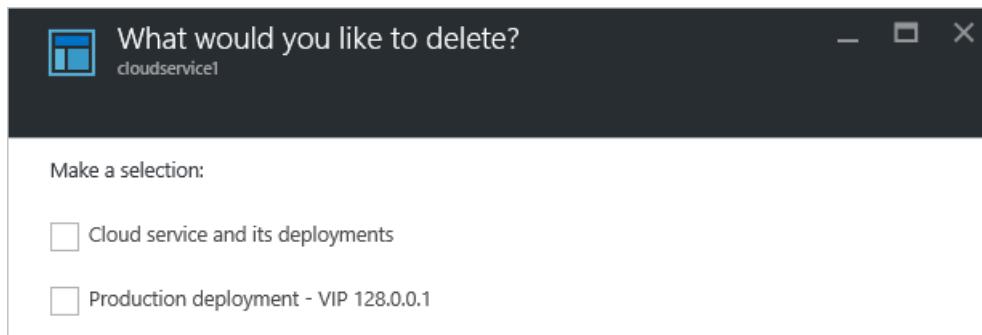
To save compute costs, you can delete the staging deployment after you verify that your production deployment is working as expected. You are billed for compute costs for deployed role instances that are stopped.

Use the following procedure to delete a deployment or your cloud service.

1. In the [Azure portal](#), select the cloud service you want to delete. This step opens the cloud service instance blade.
2. On the blade, select **Delete**.



3. To delete the entire cloud service, select the **Cloud service and its deployments** check box. Or you can choose either the **Production deployment** or the **Staging deployment** check box.



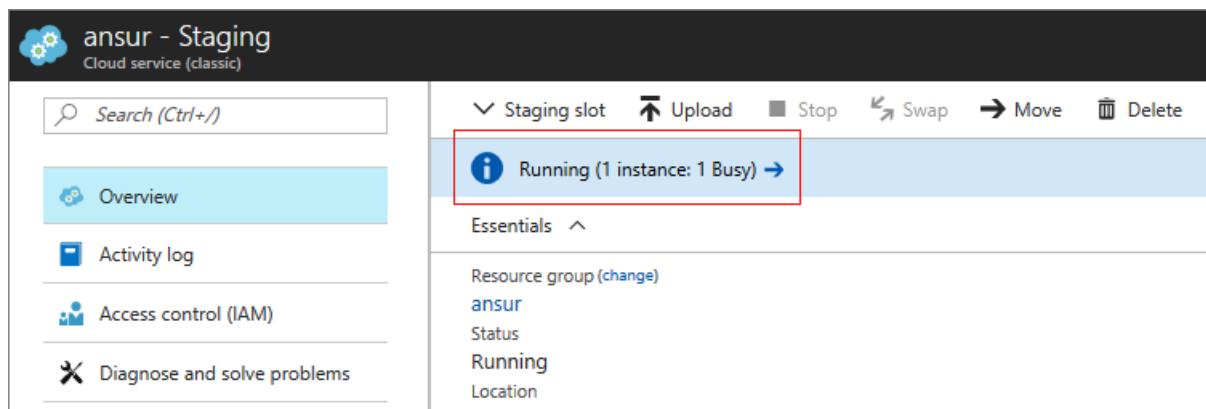
4. Select **Delete** at the bottom.
5. To delete the cloud service, select **Delete cloud service**. Then, at the confirmation prompt, select **Yes**.

NOTE

When a cloud service is deleted and verbose monitoring is configured, you must delete the data manually from your storage account. For information about where to find the metrics tables, see [Introduction to cloud service monitoring](#).

Find more information about failed deployments

The **Overview** blade has a status bar at the top. When you select the bar, a new blade opens and displays any error information. If the deployment doesn't contain any errors, the information blade is blank.



Next steps

- General configuration of your cloud service.
- Learn how to [deploy a cloud service](#).
- Configure a [custom domain name](#).
- Configure [SSL certificates](#).

How to Configure Cloud Services

7/12/2018 • 2 minutes to read • [Edit Online](#)

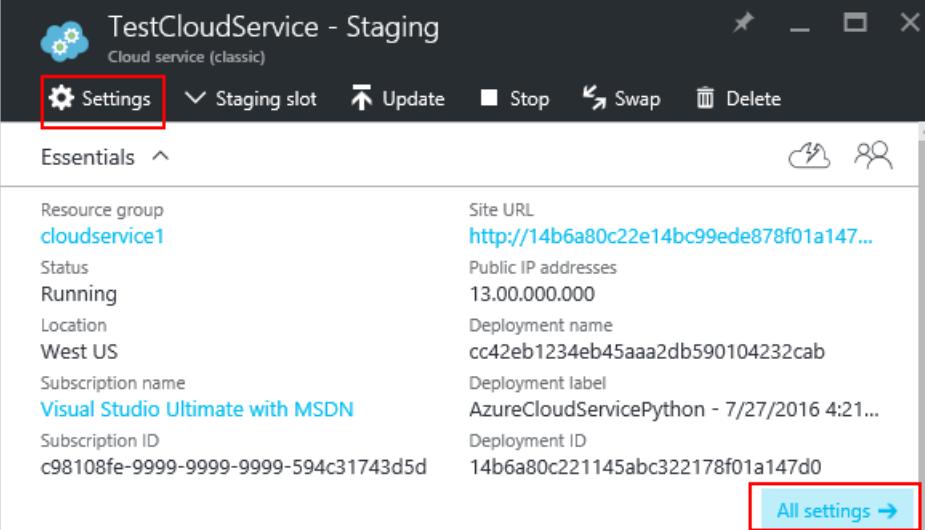
You can configure the most commonly used settings for a cloud service in the Azure portal. Or, if you like to update your configuration files directly, download a service configuration file to update, and then upload the updated file and update the cloud service with the configuration changes. Either way, the configuration updates are pushed out to all role instances.

You can also manage the instances of your cloud service roles, or remote desktop into them.

Azure can only ensure 99.95 percent service availability during the configuration updates if you have at least two role instances for every role. That enables one virtual machine to process client requests while the other is being updated. For more information, see [Service Level Agreements](#).

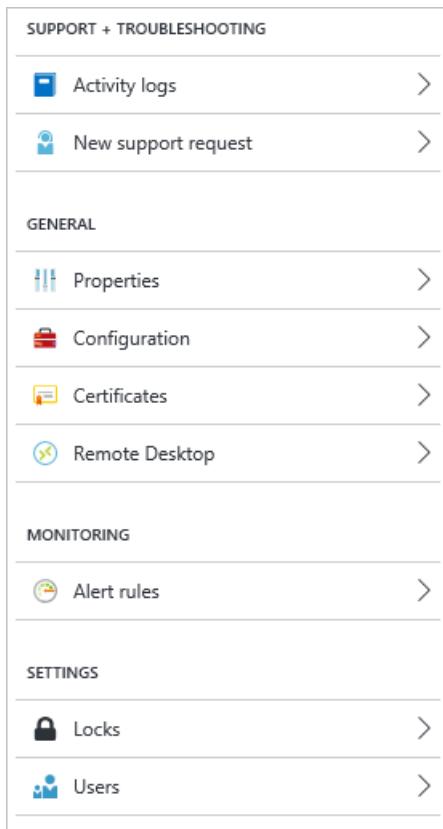
Change a cloud service

After opening the [Azure portal](#), navigate to your cloud service. From here, you manage many aspects of it.



The screenshot shows the Azure portal interface for a 'TestCloudService - Staging' cloud service. At the top, there's a navigation bar with icons for Settings, Staging slot, Update, Stop, Swap, and Delete. Below this is a section titled 'Essentials'. It lists various service details: Resource group (cloubservice1), Status (Running), Location (West US), Subscription name (Visual Studio Ultimate with MSDN), and Subscription ID (c98108fe-9999-9999-9999-594c31743d5d). To the right of these, there are fields for Site URL (http://14b6a80c22e14bc99ede878f01a147...), Public IP addresses (13.00.000.000), Deployment name (cc42eb1234eb45aaa2db590104232cab), Deployment label (AzureCloudServicePython - 7/27/2016 4:21...), and Deployment ID (14b6a80c221145abc322178f01a147d0). At the bottom right of this section is a blue button labeled 'All settings →' with a red box around it.

The **Settings** or **All settings** links will open up **Settings** where you can change the **Properties**, change the **Configuration**, manage the **Certificates**, set up **Alert rules**, and manage the **Users** who have access to this cloud service.



Manage Guest OS version

By default, Azure periodically updates your guest OS to the latest supported image within the OS family that you've specified in your service configuration (.cscfg), such as Windows Server 2016.

If you need to target a specific OS version, you can set it in **Configuration**.

The Configuration blade shows the following settings:

- Save, Discard, Download, Upload buttons
- * OS family: Windows Server 2016
- * OS version: Automatic
 - Windows Azure Guest OS 5.2 (Release 201610-02)
 - Windows Azure Guest OS 5.3 (Release 201611-01)** (selected)
 - Windows Azure Guest OS 5.4 (Release 201612-01)

IMPORTANT

Choosing a specific OS version disables automatic OS updates and makes patching your responsibility. You must ensure that your role instances are receiving updates or you may expose your application to security vulnerabilities.

Monitoring

You can add alerts to your cloud service. Click **Settings > Alert Rules > Add alert**.

The screenshot shows the Azure portal interface for managing settings. On the left, under 'SUPPORT + TROUBLESHOOTING', there are links for 'Activity logs' and 'New support request'. Under 'GENERAL', there are links for 'Properties', 'Configuration', 'Certificates', and 'Remote Desktop'. Under 'MONITORING', there is a link for 'Alert rules', which is highlighted with a red box. On the right, the 'Alert rules' section for 'TestCloudService/Staging/WorkerRole1' displays a message: 'You haven't created any alert rules.' A red box also highlights the '+ Add alert' button at the top of this section.

From here, you can set up an alert. With the **Metric** drop-down box, you can set up an alert for the following types of data.

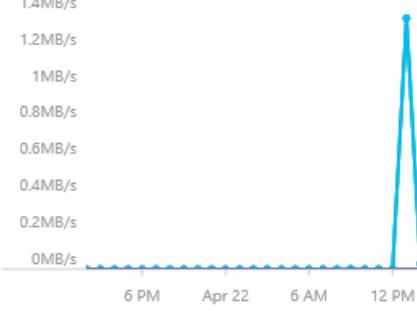
- Disk read
- Disk write
- Network in
- Network out
- CPU percentage

Add an alert rule

* Resource [?](#)
freshnessreport/Staging/WorkerRoleWit...

* Name [?](#)

Description

* Metric [?](#)
Disk read 

* Condition [?](#)
greater than 

* Threshold [?](#)
 bytes/second

* Period [?](#)
Over the last 5 minutes 

Email owners, contributors, and readers

Additional administrator email(s)

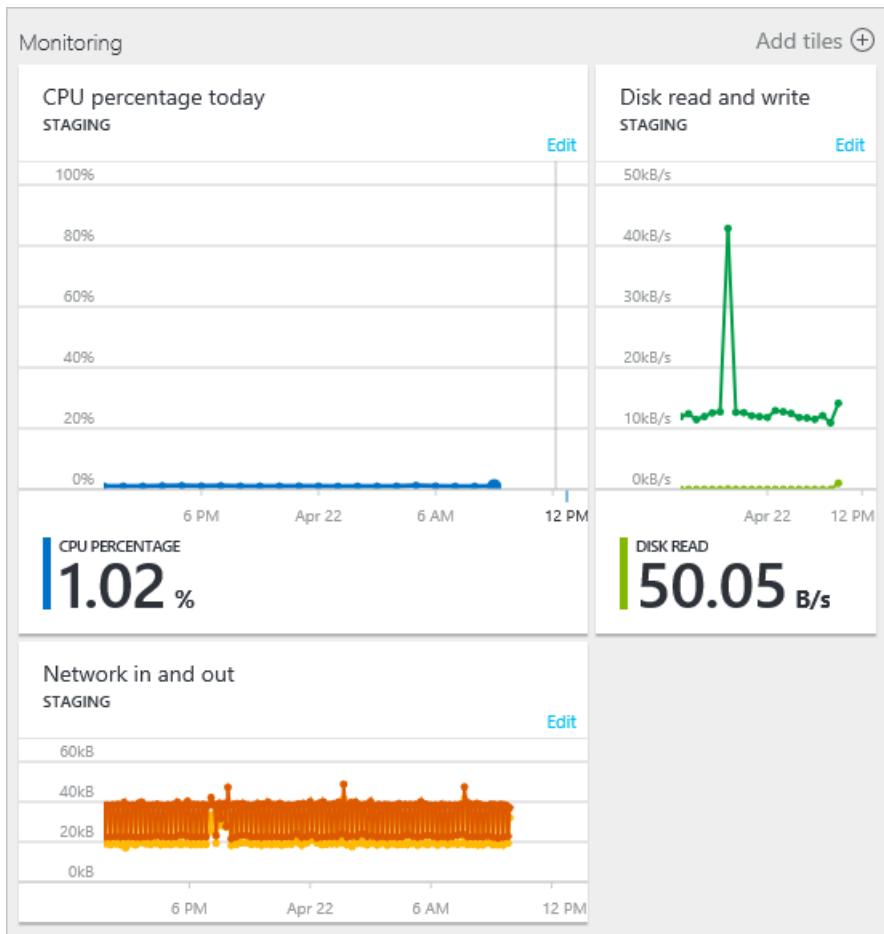
Webhook [?](#)

[Learn more about configuring webhooks](#)

OK

Configure monitoring from a metric tile

Instead of using **Settings > Alert Rules**, you can click on one of the metric tiles in the **Monitoring** section of the cloud service.



From here you can customize the chart used with the tile, or add an alert rule.

Reboot, reimagine, or remote desktop

You can set up remote desktop through the [Azure portal \(set up remote desktop\)](#), [PowerShell](#), or through [Visual Studio](#).

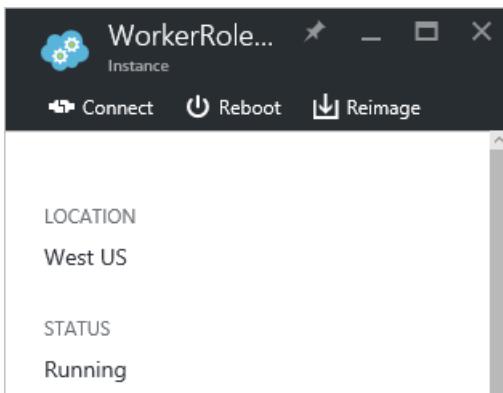
To reboot, reimagine, or remote into a Cloud Service, select the cloud service instance.

The screenshot shows the Azure Cloud Services Instances list. It displays the following information:

NAME	STATUS	SIZE	UPDATE	FAULT
WorkerRoleWithSBQueue1	Running	Small	0	0
WorkerRoleWithSBQue...	Running	Small	0	0

A red box highlights the first instance, "WorkerRoleWithSBQueue1".

You can then initiate a remote desktop connection, remotely reboot the instance, or remotely reimagine (start with a fresh image) the instance.



Reconfigure your .cscfg

You may need to reconfigure your cloud service through the [service config \(cscfg\)](#) file. First you need to download your .cscfg file, modify it, then upload it.

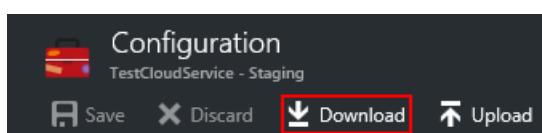
1. Click on the **Settings** icon or the **All settings** link to open up **Settings**.

A screenshot of the Azure portal showing the settings for a cloud service named 'TestCloudService - Staging'. The 'Settings' icon in the top navigation bar is highlighted with a red box. The 'All settings' link at the bottom right of the page is also highlighted with a red box.

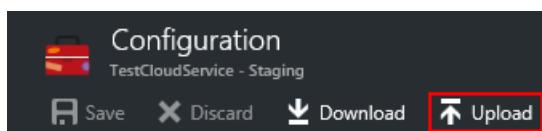
2. Click on the **Configuration** item.



3. Click on the **Download** button.



4. After you update the service configuration file, upload and apply the configuration updates:



5. Select the .cscfg file and click **OK**.

Next steps

- Learn how to [deploy a cloud service](#).
- Configure a custom domain name.
- [Manage your cloud service](#).
- Configure [ssl certificates](#).

Managing Azure Cloud Services using Azure Automation

6/27/2017 • 2 minutes to read • [Edit Online](#)

This guide will introduce you to the Azure Automation service, and how it can be used to simplify management of your Azure cloud services.

What is Azure Automation?

[Azure Automation](#) is an Azure service for simplifying cloud management through process automation. Using Azure Automation, long-running, manual, error-prone, and frequently repeated tasks can be automated to increase reliability, efficiency, and time to value for your organization.

Azure Automation provides a highly-reliable and highly-available workflow execution engine that scales to meet your needs as your organization grows. In Azure Automation, processes can be kicked off manually, by 3rd-party systems, or at scheduled intervals so that tasks happen exactly when needed.

Lower operational overhead and free up IT / DevOps staff to focus on work that adds business value by moving your cloud management tasks to be run automatically by Azure Automation.

How can Azure Automation help manage Azure cloud services?

Azure cloud services can be managed in Azure Automation by using the PowerShell cmdlets that are available in the [Azure PowerShell tools](#). Azure Automation has these cloud service PowerShell cmdlets available out of the box, so that you can perform all of your cloud service management tasks within the service. You can also pair these cmdlets in Azure Automation with the cmdlets for other Azure services, to automate complex tasks across Azure services and 3rd party systems.

Some example uses of Azure Automation to manage Azure Cloud Services include:

- [Continous deployment of a Cloud Service whenever cscfg or cspkg is updated in Azure Blob storage](#)
- [Rebooting Cloud Service instances in parallel, one upgrade domain at a time](#)

Next Steps

Now that you've learned the basics of Azure Automation and how it can be used to manage Azure cloud services, follow these links to learn more about Azure Automation.

- [Azure Automation Overview](#)
- [My first runbook](#)
- [Azure Automation learning map](#)

How to configure auto scaling for a Cloud Service in the portal

7/12/2018 • 3 minutes to read • [Edit Online](#)

Conditions can be set for a cloud service worker role that trigger a scale in or out operation. The conditions for the role can be based on the CPU, disk, or network load of the role. You can also set a condition based on a message queue or the metric of some other Azure resource associated with your subscription.

NOTE

This article focuses on Cloud Service web and worker roles. When you create a virtual machine (classic) directly, it is hosted in a cloud service. You can scale a standard virtual machine by associating it with an [availability set](#) and manually turn them on or off.

Considerations

You should consider the following information before you configure scaling for your application:

- Scaling is affected by core usage.

Larger role instances use more cores. You can scale an application only within the limit of cores for your subscription. For example, say your subscription has a limit of 20 cores. If you run an application with two medium-sized cloud services (a total of 4 cores), you can only scale up other cloud service deployments in your subscription by the remaining 16 cores. For more information about sizes, see [Cloud Service Sizes](#).

- You can scale based on a queue message threshold. For more information about how to use queues, see [How to use the Queue Storage Service](#).
- You can also scale other resources associated with your subscription.
- To enable high availability of your application, you should ensure that it is deployed with two or more role instances. For more information, see [Service Level Agreements](#).
- Auto Scale only happens when all the roles are in **Ready** state.

Where scale is located

After you select your cloud service, you should have the cloud service blade visible.

1. On the cloud service blade, on the **Roles and Instances** tile, select the name of the cloud service.

IMPORTANT: Make sure to click the cloud service role, not the role instance that is below the role.

Roles and instances					
NAME	STATUS	SIZE	UPDATE	FAULT	
▼ WorkerRoleWithSBQueue1					
WorkerRoleWithSBQue...	Running	Small	0	0	

2. Select the **scale** tile.

Automatic scale

You can configure scale settings for a role with either two modes **manual** or **automatic**. Manual is as you would expect, you set the absolute count of instances. Automatic however allows you to set rules that govern how and by how much you should scale.

Set the **Scale by** option to **schedule and performance rules**.

1. An existing profile.
2. Add a rule for the parent profile.
3. Add another profile.

Select **Add Profile**. The profile determines which mode you want to use for the scale: **always**, **recurrence**, **fixed date**.

After you have configured the profile and rules, select the **Save** icon at the top.

Profile

The profile sets minimum and maximum instances for the scale, and also when this scale range is active.

- **Always**

Always keep this range of instances available.

- **Recurrence**

Choose a set of days of the week to scale.

* Name
Default

Type
 always recurrence fixed date

Target range
1 1

* Days
7 selected

* Start time
12:00

* Time zone
(UTC-08:00) Pacific Time (US & Canada)

- **Fixed Date**

A fixed date range to scale the role.

* Name
Default

Type
 always recurrence fixed date

Target range
1 1

* Start time
2016-06-07 00:00:00

* End time
2016-06-07 23:59:00

* Time zone
(UTC-08:00) Pacific Time (US & Canada)

After you have configured the profile, select the **OK** button at the bottom of the profile blade.

Rule

Rules are added to a profile and represent a condition that triggers the scale.

The rule trigger is based on a metric of the cloud service (CPU usage, disk activity, or network activity) to which you can add a conditional value. Additionally you can have the trigger based on a message queue or the metric of some other Azure resource associated with your subscription.

* Resource
freshnessreport/Staging/WorkerRoleWithS... ▾

* Metric name
Please choose a metric ▾ !

* Operator
Greater than ▾

Threshold
5

Duration (minutes)
30 ✓

Time aggregation
Average ▾

* Action
increase count by ▾

Value
1

Cool down (minutes)
5

After you have configured the rule, select the **OK** button at the bottom of the rule blade.

Back to manual scale

Navigate to the [scale settings](#) and set the **Scale by** option to **an instance count that I enter manually**.

* Scale by ① an instance count that I enter manually ▾

Description Manual setup means that the number of instances you choose won't change, even if there are changes in load.

Instances ② ← ③ 1

This setting removes automated scaling from the role and then you can set the instance count directly.

1. The scale (manual or automated) option.
2. A role instance slider to set the instances to scale to.
3. Instances of the role to scale to.

After you have configured the scale settings, select the **Save** icon at the top.

Use service management from Python

9/5/2018 • 9 minutes to read • [Edit Online](#)

This guide shows you how to programmatically perform common service management tasks from Python. The **ServiceManagementService** class in the [Azure SDK for Python](#) supports programmatic access to much of the service management-related functionality that is available in the [Azure portal](#). You can use this functionality to create, update, and delete cloud services, deployments, data management services, and virtual machines. This functionality can be useful in building applications that need programmatic access to service management.

What is service management?

The Azure Service Management API provides programmatic access to much of the service management functionality available through the [Azure portal](#). You can use the Azure SDK for Python to manage your cloud services and storage accounts.

To use the Service Management API, you need to [create an Azure account](#).

Concepts

The Azure SDK for Python wraps the [Service Management API](#), which is a REST API. All API operations are performed over SSL and mutually authenticated by using X.509 v3 certificates. The management service can be accessed from within a service running in Azure. It also can be accessed directly over the Internet from any application that can send an HTTPS request and receive an HTTPS response.

Installation

All the features described in this article are available in the `azure-servicemanagement-legacy` package, which you can install by using pip. For more information about installation (for example, if you're new to Python), see [Install Python and the Azure SDK](#).

Connect to service management

To connect to the service management endpoint, you need your Azure subscription ID and a valid management certificate. You can obtain your subscription ID through the [Azure portal](#).

NOTE

You now can use certificates created with OpenSSL when running on Windows. Python 2.7.4 or later is required. We recommend that you use OpenSSL instead of .pfx, because support for .pfx certificates is likely to be removed in the future.

Management certificates on Windows/Mac/Linux (OpenSSL)

You can use [OpenSSL](#) to create your management certificate. You need to create two certificates, one for the server (a `.cer` file) and one for the client (a `.pem` file). To create the `.pem` file, execute:

```
openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout mycert.pem -out mycert.pem
```

To create the `.cer` certificate, execute:

```
openssl x509 -inform pem -in mycert.pem -outform der -out mycert.cer
```

For more information about Azure certificates, see [Certificates overview for Azure Cloud Services](#). For a complete description of OpenSSL parameters, see the documentation at <http://www.openssl.org/docs/apps/openssl.html>.

After you create these files, upload the `.cer` file to Azure. In the [Azure portal](#), on the **Settings** tab, select **Upload**. Note where you saved the `.pem` file.

After you obtain your subscription ID, create a certificate, and upload the `.cer` file to Azure, connect to the Azure management endpoint. Connect by passing the subscription ID and the path to the `.pem` file to **ServiceManagementService**.

```
from azure import *
from azure.servicemanagement import *

subscription_id = '<your_subscription_id>'
certificate_path = '<path_to_.pem_certificate>'

sms = ServiceManagementService(subscription_id, certificate_path)
```

In the preceding example, `sms` is a **ServiceManagementService** object. The **ServiceManagementService** class is the primary class used to manage Azure services.

Management certificates on Windows (MakeCert)

You can create a self-signed management certificate on your machine by using `makecert.exe`. Open a **Visual Studio command prompt** as an **administrator** and use the following command, replacing *AzureCertificate* with the certificate name you want to use:

```
makecert -sky exchange -r -n "CN=AzureCertificate" -pe -a sha1 -len 2048 -ss My "AzureCertificate.cer"
```

The command creates the `.cer` file and installs it in the **Personal** certificate store. For more information, see [Certificates overview for Azure Cloud Services](#).

After you create the certificate, upload the `.cer` file to Azure. In the [Azure portal](#), on the **Settings** tab, select **Upload**.

After you obtain your subscription ID, create a certificate, and upload the `.cer` file to Azure, connect to the Azure management endpoint. Connect by passing the subscription ID and the location of the certificate in your **Personal** certificate store to **ServiceManagementService** (again, replace *AzureCertificate* with the name of your certificate).

```
from azure import *
from azure.servicemanagement import *

subscription_id = '<your_subscription_id>'
certificate_path = 'CURRENT_USER\\my\\AzureCertificate'

sms = ServiceManagementService(subscription_id, certificate_path)
```

In the preceding example, `sms` is a **ServiceManagementService** object. The **ServiceManagementService** class is the primary class used to manage Azure services.

List available locations

To list the locations that are available for hosting services, use the **list_locations** method.

```
from azure import *
from azure.servicemanagement import *

sms = ServiceManagementService(subscription_id, certificate_path)

result = sms.list_locations()
for location in result:
    print(location.name)
```

When you create a cloud service or storage service, you need to provide a valid location. The **list_locations** method always returns an up-to-date list of the currently available locations. As of this writing, the available locations are:

- West Europe
- North Europe
- Southeast Asia
- East Asia
- Central US
- North Central US
- South Central US
- West US
- East US
- Japan East
- Japan West
- Brazil South
- Australia East
- Australia Southeast

Create a cloud service

When you create an application and run it in Azure, the code and configuration together are called an Azure [cloud service](#). (It was known as a *hosted service* in earlier Azure releases.) You can use the **create_hosted_service** method to create a new hosted service. Create the service by providing a hosted service name (which must be unique in Azure), a label (automatically encoded to base64), a description, and a location.

```
from azure import *
from azure.servicemanagement import *

sms = ServiceManagementService(subscription_id, certificate_path)

name = 'myhostedservice'
label = 'myhostedservice'
desc = 'my hosted service'
location = 'West US'

sms.create_hosted_service(name, label, desc, location)
```

You can list all the hosted services for your subscription with the **list_hosted_services** method.

```
result = sms.list_hosted_services()

for hosted_service in result:
    print('Service name: ' + hosted_service.service_name)
    print('Management URL: ' + hosted_service.url)
    print('Location: ' + hosted_service.hosted_service_properties.location)
    print('')
```

To get information about a particular hosted service, pass the hosted service name to the **get_hosted_service_properties** method.

```
hosted_service = sms.get_hosted_service_properties('myhostedservice')

print('Service name: ' + hosted_service.service_name)
print('Management URL: ' + hosted_service.url)
print('Location: ' + hosted_service.hosted_service_properties.location)
```

After you create a cloud service, deploy your code to the service with the **create_deployment** method.

Delete a cloud service

You can delete a cloud service by passing the service name to the **delete_hosted_service** method.

```
sms.delete_hosted_service('myhostedservice')
```

Before you can delete a service, all deployments for the service must first be deleted. For more information, see [Delete a deployment](#).

Delete a deployment

To delete a deployment, use the **delete_deployment** method. The following example shows how to delete a deployment named `v1`:

```
from azure import *
from azure.servicemanagement import *

sms = ServiceManagementService(subscription_id, certificate_path)

sms.delete_deployment('myhostedservice', 'v1')
```

Create a storage service

A [storage service](#) gives you access to Azure [blobs](#), [tables](#), and [queues](#). To create a storage service, you need a name for the service (between 3 and 24 lowercase characters and unique within Azure). You also need a description, a label (up to 100 characters, automatically encoded to base64), and a location. The following example shows how to create a storage service by specifying a location:

```
from azure import *
from azure.servicemanagement import *

sms = ServiceManagementService(subscription_id, certificate_path)

name = 'mystorageaccount'
label = 'mystorageaccount'
location = 'West US'
desc = 'My storage account description.'

result = sms.create_storage_account(name, desc, label, location=location)

operation_result = sms.get_operation_status(result.request_id)
print('Operation status: ' + operation_result.status)
```

In the preceding example, the status of the **create_storage_account** operation can be retrieved by passing the result returned by **create_storage_account** to the **get_operation_status** method.

You can list your storage accounts and their properties with the **list_storage_accounts** method.

```
from azure import *
from azure.servicemanagement import *

sms = ServiceManagementService(subscription_id, certificate_path)

result = sms.list_storage_accounts()
for account in result:
    print('Service name: ' + account.service_name)
    print('Location: ' + account.storage_service_properties.location)
    print('')
```

Delete a storage service

To delete a storage service, pass the storage service name to the **delete_storage_account** method. Deleting a storage service deletes all data stored in the service (blobs, tables, and queues).

```
from azure import *
from azure.servicemanagement import *

sms = ServiceManagementService(subscription_id, certificate_path)

sms.delete_storage_account('mystorageaccount')
```

List available operating systems

To list the operating systems that are available for hosting services, use the **list_operating_systems** method.

```
from azure import *
from azure.servicemanagement import *

sms = ServiceManagementService(subscription_id, certificate_path)

result = sms.list_operating_systems()

for os in result:
    print('OS: ' + os.label)
    print('Family: ' + os.family_label)
    print('Active: ' + str(os.is_active))
```

Alternatively, you can use the **list_operating_system_families** method, which groups the operating systems by family.

```
result = sms.list_operating_system_families()

for family in result:
    print('Family: ' + family.label)
    for os in family.operating_systems:
        if os.is_active:
            print('OS: ' + os.label)
            print('Version: ' + os.version)
    print()
```

Create an operating system image

To add an operating system image to the image repository, use the **add_os_image** method.

```
from azure import *
from azure.servicemanagement import *

sms = ServiceManagementService(subscription_id, certificate_path)

name = 'mycentos'
label = 'mycentos'
os = 'Linux' # Linux or Windows
media_link = 'url_to_storage_blob_for_source_image_vhd'

result = sms.add_os_image(label, media_link, name, os)

operation_result = sms.get_operation_status(result.request_id)
print('Operation status: ' + operation_result.status)
```

To list the operating system images that are available, use the **list_os_images** method. It includes all platform images and user images.

```
result = sms.list_os_images()

for image in result:
    print('Name: ' + image.name)
    print('Label: ' + image.label)
    print('OS: ' + image.os)
    print('Category: ' + image.category)
    print('Description: ' + image.description)
    print('Location: ' + image.location)
    print('Media link: ' + image.media_link)
    print()
```

Delete an operating system image

To delete a user image, use the **delete_os_image** method.

```

from azure import *
from azure.servicemanagement import *

sms = ServiceManagementService(subscription_id, certificate_path)

result = sms.delete_os_image('mycentos')

operation_result = sms.get_operation_status(result.request_id)
print('Operation status: ' + operation_result.status)

```

Create a virtual machine

To create a virtual machine, you first need to create a [cloud service](#). Then create the virtual machine deployment by using the **create_virtual_machine_deployment** method.

```

from azure import *
from azure.servicemanagement import *

sms = ServiceManagementService(subscription_id, certificate_path)

name = 'myvms'
location = 'West US'

#Set the location
sms.create_hosted_service(service_name=name,
    label=name,
    location=location)

# Name of an os image as returned by list_os_images
image_name = 'OpenLogic__OpenLogic-CentOS-62-20120531-en-us-30GB.vhd'

# Destination storage account container/blob where the VM disk
# will be created
media_link = 'url_to_target_storage_blob_for_vm_hd'

# Linux VM configuration, you can use WindowsConfigurationSet
# for a Windows VM instead
linux_config = LinuxConfigurationSet('myhostname', 'myuser', 'mypassword', True)

os_hd = OSVirtualHardDisk(image_name, media_link)

sms.create_virtual_machine_deployment(service_name=name,
    deployment_name=name,
    deployment_slot='production',
    label=name,
    role_name=name,
    system_config=linux_config,
    os_virtual_hard_disk=os_hd,
    role_size='Small')

```

Delete a virtual machine

To delete a virtual machine, you first delete the deployment by using the **delete_deployment** method.

```
from azure import *
from azure.servicemanagement import *

sms = ServiceManagementService(subscription_id, certificate_path)

sms.delete_deployment(service_name='myvm',
                      deployment_name='myvm')
```

The cloud service can then be deleted by using the **delete_hosted_service** method.

```
sms.delete_hosted_service(service_name='myvm')
```

Create a virtual machine from a captured virtual machine image

To capture a VM image, you first call the **capture_vm_image** method.

```
from azure import *
from azure.servicemanagement import *

sms = ServiceManagementService(subscription_id, certificate_path)

# replace the below three parameters with actual values
hosted_service_name = 'hs1'
deployment_name = 'dep1'
vm_name = 'vm1'

image_name = vm_name + 'image'
image = CaptureRoleAsVMImage ('Specialized',
    image_name,
    image_name + 'label',
    image_name + 'description',
    'english',
    'mygroup')

result = sms.capture_vm_image(
    hosted_service_name,
    deployment_name,
    vm_name,
    image
)
```

To make sure that you successfully captured the image, use the **list_vm_images** API. Make sure your image is displayed in the results.

```
images = sms.list_vm_images()
```

To finally create the virtual machine by using the captured image, use the **create_virtual_machine_deployment** method as before, but this time pass in the `vm_image_name` instead.

```
from azure import *
from azure.servicemanagement import *

sms = ServiceManagementService(subscription_id, certificate_path)

name = 'myvm'
location = 'West US'

#Set the location
sms.create_hosted_service(service_name=name,
    label=name,
    location=location)

sms.create_virtual_machine_deployment(service_name=name,
    deployment_name=name,
    deployment_slot='production',
    label=name,
    role_name=name,
    system_config=linux_config,
    os_virtual_hard_disk=None,
    role_size='Small',
    vm_image_name = image_name)
```

To learn more about how to capture a Linux virtual machine in the classic deployment model, see [Capture a Linux virtual machine](#).

To learn more about how to capture a Windows virtual machine in the classic deployment model, see [Capture a Windows virtual machine](#).

Next steps

Now that you've learned the basics of service management, you can access the [Complete API reference documentation for the Azure Python SDK](#) and perform complex tasks easily to manage your Python application.

For more information, see the [Python Developer Center](#).

Guidance for mitigating speculative execution side-channel vulnerabilities in Azure

5/22/2018 • 4 minutes to read • [Edit Online](#)

Last document update: 14 August 2018 10:00 AM PST.

The disclosure of a [new class of CPU vulnerabilities](#) known as speculative execution side-channel attacks has resulted in questions from customers seeking more clarity.

Microsoft has deployed mitigations across all our cloud services. The infrastructure that runs Azure and isolates customer workloads from each other is protected. This means that a potential attacker using the same infrastructure can't attack your application using these vulnerabilities.

Azure is using [memory preserving maintenance](#) whenever possible, to minimize customer impact and eliminate the need for reboots. Azure will continue utilizing these methods when making systemwide updates to the host and protect our customers.

More information about how security is integrated into every aspect of Azure is available on the [Azure Security Documentation](#) site.

NOTE

Since this document was first published, multiple variants of this vulnerability class have been disclosed. Microsoft continues to be heavily invested in protecting our customers and providing guidance. This page will be updated as we continue to release further fixes.

On August 14, 2018, the industry disclosed a new speculative execution side channel vulnerability known as [L1 Terminal Fault \(L1TF\)](#) which has been assigned multiple CVEs ([CVE-2018-3615](#), [CVE-2018-3620](#), and [CVE-2018-3646](#)). This vulnerability affects Intel® Core® processors and Intel® Xeon® processors. Microsoft has deployed mitigations across our cloud services which reinforce the isolation between customers. Please read below for additional guidance to protect against L1TF and previous vulnerabilities ([Spectre Variant 2 CVE-2017-5715](#) and [Meltdown Variant 3 CVE-2017-5754](#)).

Keeping your Operating Systems up-to-date

While an OS update is not required to isolate your applications running on Azure from other Azure customers, it is always a best practice to keep your software up-to-date. The latest Security Rollups for Windows contain mitigations for several speculative execution side channel vulnerabilities. Similarly, Linux distributions have released multiple updates to address these vulnerabilities. Here are our recommended actions to update your Operating System:

OFFERING	RECOMMENDED ACTION
Azure Cloud Services	Enable auto update or ensure you are running the newest Guest OS.
Azure Linux Virtual Machines	Install updates from your operating system provider. For more information, see Linux later in this document.
Azure Windows Virtual Machines	Install the latest security rollup.

OFFERING	RECOMMENDED ACTION
Other Azure PaaS Services	There is no action needed for customers using these services. Azure automatically keeps your OS versions up-to-date.

Additional guidance if you are running untrusted code

Customers who allow untrusted users to execute arbitrary code may wish to implement some additional security features inside their Azure Virtual Machines or Cloud Services. These features protect against the intra-process disclosure vectors that several speculative execution vulnerabilities describe.

Example scenarios where additional security features are recommended:

- You allow code that you do not trust to run inside your VM.
 - *For example, you allow one of your customers to upload a binary or script that you then execute within your application.*
- You allow users that you do not trust to log into your VM using low privileged accounts.
 - *For example, you allow a low-privileged user to log into one of your VMs using remote desktop or SSH.*
- You allow untrusted users access to virtual machines implemented via nested virtualization.
 - *For example, you control the Hyper-V host, but allocate the VMs to untrusted users.*

Customers who do not implement a scenario involving untrusted code do not need to enable these additional security features.

Enabling additional security

You can enable additional security features inside your VM or Cloud Service.

Windows

Your target operating system must be up-to-date to enable these additional security features. While numerous speculative execution side channel mitigations are enabled by default, the additional features described here must be enabled manually and may cause a performance impact.

Step 1: Contact Azure Support to expose updated firmware (microcode) into your Virtual Machines.

Step 2: Enable Kernel Virtual Address Shadowing (KVAS) and Branch Target Injection (BTI) OS support. Follow the instructions in [KB4072698](#) to enable protections via the `Session Manager` registry keys. A reboot is required.

Step 3: For deployments that are using [nested virtualization](#) (D3 and E3 only): These instructions apply inside the VM you are using as a Hyper-V host.

1. Follow the instructions in [KB4072698](#) to enable protections via the `MinVmVersionForCpuBasedMitigations` registry keys.
2. Set the hypervisor scheduler type to **Core** by following the instructions [here](#).

Step 4: Follow the instructions in [KB4072698](#) to verify protections are enabled using the [SpeculationControl](#) PowerShell module.

NOTE

If you previously downloaded this module, you will need to install the newest version.

All VMs should show:

```
branch target injection mitigation is enabled: True  
kernel VA shadow is enabled: True  
L1TFWindowsSupportEnabled: True
```

Linux

Enabling the set of additional security features inside requires that the target operating system be fully up-to-date. Some mitigations will be enabled by default. The following section describes the features which are off by default and/or reliant on hardware support (microcode). Enabling these features may cause a performance impact. Reference your operating system provider's documentation for further instructions

Step 1: Contact [Azure Support](#) to expose updated firmware (microcode) into your Virtual Machines.

Step 2: Enable Branch Target Injection (BTI) OS support to mitigate CVE-2017-5715 (Spectre Variant 2) by following your operating system provider's documentation.

Step 3: Enable Kernel Page Table Isolation (KPTI) to mitigate CVE-2017-5754 (Meltdown Variant 3) by following your operating system provider's documentation.

More information is available from your operating system's provider:

- [Redhat and CentOS](#)
- [Suse](#)
- [Ubuntu](#)

Next steps

To learn more, see [Securing Azure customers from CPU vulnerability](#).

Azure Guest OS

1/8/2019 • 42 minutes to read • [Edit Online](#)

The following tables show the Microsoft Security Response Center (MSRC) updates applied to the Azure Guest OS. Search this article to determine if a particular update applies to the Guest OS you are using. Updates always carry forward for the particular [family](#) they were introduced in.

December 2018 Guest OS

Product Category	Parent KB Article	Vulnerability Description	Guest OS	Date First Introduced
Rel 18-12	4471321	Windows 10 Security	5.26	December 11, 2018
Rel 18-12	4471328	Windows Security	2.81	December 11, 2018
Rel 18-12	4471326	Windows Security	3.68	December 11, 2018
Rel 18-12	4471322	Windows Security	4.61	December 11, 2018
Rel 18-12	4470600	.Net 3.x Security	2.81	December 11, 2018
Rel 18-12	4470601	.Net 3.x Security	3.68	December 11, 2018
Rel 18-12	4470602	.Net 3.x Security	4.61	December 11, 2018
Rel 18-12	4470493	.Net 4.x Security	2.81	December 11, 2018
Rel 18-12	4470492	.Net 4.x Security	3.68	December 11, 2018
Rel 18-12	4470491	.Net 4.x Security	4.61	December 11, 2018
Rel 18-12	4471331	Flash	3.68, 4.61, 5.26	December 5, 2018
Rel 18-12	4470199	Internet Explorer	2.81, 3.68, 4.61	December 11, 2018
N/A	4468323	Timezone update	2.81, 3.68, 4.61	December 13, 2018
N/A	4467107	November Non-Security rollup	2.81	November 13, 2018
N/A	4467701	November Non-Security rollup	3.68	November 13, 2018
N/A	4467697	November Non-Security rollup	4.61	November 13, 2018

November 2018 Guest OS

Product Category	Parent KB Article	Vulnerability Description	Guest OS	Date First Introduced
Rel 18-11	4466536	Internet Explorer	2.80, 3.67, 4.60	November 13, 2018
Rel 18-11	4467694	Flash	3.67, 4.60, 5.25	November 13, 2018
Rel 18-11	4467106	Windows Security	2.80	November 13, 2018
Rel 18-11	4467678	Windows Security	3.67	November 13, 2018
Rel 18-11	4467703	Windows Security	4.60	November 13, 2018
Rel 18-11	4467691	Windows 10 Security	5.25	November 13, 2018
N/A	3173426	Servicing Stack Update	3.67	July 12, 2016
N/A	4465659	Servicing Stack Update	5.25	November 13, 2018
N/A	4462923	October Non-Security rollup	2.80	October 9, 2018
N/A	4462929	October Non-Security rollup	3.67	October 9, 2018
N/A	4462926	October Non-Security rollup	4.60	October 9, 2018
N/A	3109976	App compat shims	4.60	April 12, 2016
N/A	4457037	App compat shims	4.60	June 12, 2018

October 2018 Guest OS

Product Category	Parent KB Article	Vulnerability Description	Guest OS	Date First Introduced
Rel 18-10	4462917	Windows 10 Security	5.24	October 9, 2018
Rel 18-10	4462915	Windows Security	2.79	October 9, 2018
Rel 18-10	4462931	Windows Security	3.66	October 9, 2018
Rel 18-10	4462941	Windows Security	4.59	October 9, 2018
Rel 18-10	4462930	Flash	3.66, 4.59, 5.24	October 9, 2018
Rel 18-10	4462949	Internet Explorer	2.79, 3.66, 4.59	October 9, 2018
N/A	4339284	Time zone update	2.79, 3.66, 4.59	July 24, 2018

Product Category	Parent KB Article	Vulnerability Description	Guest OS	Date First Introduced
N/A	4457144	September Non-Security rollup	2.79	September 11, 2018
N/A	4457044	September Non-Security .NET 3.5 rollup	2.79	September 11, 2018
N/A	4457038	September Non-Security .NET 4.5.2 rollup	2.79	September 11, 2018
N/A	4457135	September Non-Security rollup	3.66	September 11, 2018
N/A	4457042	September Non-Security .NET 3.5 rollup	3.66	September 11, 2018
N/A	4457037	September Non-Security .NET 4.5.2 rollup	3.66	September 11, 2018
N/A	4457129	September Non-Security rollup	4.59	September 11, 2018
N/A	4457045	September Non-Security .NET 3.5 rollup	4.59	September 11, 2018
N/A	4457036	September Non-Security .NET 4.5.2 rollup	4.59	September 11, 2018

September 2018 Guest OS

Product Category	Parent KB Article	Vulnerability Description	Guest OS	Date First Introduced
Rel 18-09	4457131	Windows 10 Security	5.23	September 11, 2018
Rel 18-09	4457145	Windows Security only	2.78	September 11, 2018
Rel 18-09	4457140	Windows Security only	3.65	September 11, 2018
Rel 18-09	4457143	Windows Security only	4.58	September 11, 2018
Rel 18-09	4457055, 4457030	.NET 3.5, 4.5 Security	2.78	September 11, 2018
Rel 18-09	4457053, 4457029	.NET 3.5, 4.x Security	3.65	September 11, 2018

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 18-09	4457056 , 4457028	.NET 3.5, 4.x Security	4.58	September 11, 2018
Rel 18-09	4457146	Flash	3.65, 4.58, 5.23	September 11, 2018
Rel 18-09	4457426	Internet Explorer	2.78, 3.65, 4.58	September 11, 2018
N/A	4343900	August Non-Security rollup	2.78	August 14, 2018
N/A	4344152	August Non-Security .NET 3.5 rollup	2.78	August 14, 2018
N/A	4344149	August Non-Security .NET 4x rollup	2.78	August 14, 2018
N/A	4343901	August Non-Security rollup	3.65	August 14, 2018
N/A	4344150	August Non-Security .NET 3.5 rollup	3.65	August 14, 2018
N/A	4344148	August Non-Security .NET 4x rollup	3.65	August 14, 2018
N/A	4343898	August Non-Security rollup	4.58	August 14, 2018
N/A	4344153	August Non-Security .NET 3.5 rollup	4.58	August 14, 2018
N/A	4344147	August Non-Security .NET 4x rollup	4.58	August 14, 2018

August 2018 Guest OS

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 18-08	4343887	Windows 10 Security	5.22	August 14, 2018
Rel 18-08	4343899	Windows Security only	2.77	August 14, 2018
Rel 18-08	4343896	Windows Security only	3.64	August 14, 2018
Rel 18-08	4343888	Windows Security only	4.57	August 14, 2018
Rel 18-08	4344177 , 4344173	.NET 3.5, 4.5 Security	2.77	August 14, 2018

Product Category	Parent KB Article	Vulnerability Description	Guest OS	Date First Introduced
Rel 18-08	4344175 , 4344172	.NET 3.5, 4.x Security	3.64	August 14, 2018
Rel 18-08	4344178 , 4344171	.NET 3.5, 4.x Security	4.57	August 14, 2018
Rel 18-08	4346742 , 4346739	.NET 3.5, 4.x OOB release	3.64	July 30, 2018
Rel 18-08	4346745 , 4346408	.NET 3.5, 4.x OOB release	4.57	July 30, 2018
Rel 18-08	4343902	Flash	3.64, 4.57, 5.22	August 14, 2018
Rel 18-08	4343205	Internet Explorer	2.77, 3.64, 4.57	August 14, 2018
N/A	4338818	July Non-Security rollup	2.77	July 10, 2018
N/A	4019990	D3D Compiler update for .NET	2.77	July 10, 2018
N/A	4338830	July Non-Security rollup	3.64	July 10, 2018
N/A	4338421	July Non-Security .NET 3.5 rollup	3.64	July 10, 2018
N/A	4338416	July Non-Security .NET 4x rollup	3.64	July 10, 2018
N/A	4338815	July Non-Security rollup	4.57	July 10, 2018
N/A	4338424	July Non-Security .NET 3.5 rollup	3.64	July 10, 2018
N/A	4338415	July Non-Security .NET 4x rollup	3.64	July 10, 2018

July 2018 Guest OS

Product Category	Parent KB Article	Vulnerability Description	Guest OS	Date First Introduced
Rel 18-07	4338814	Windows 10 Security	5.21	July 10, 2018
Rel 18-07	4338823	Windows Security only	2.76	July 10, 2018
Rel 18-07	4338820	Windows Security only	3.63	July 10, 2018

Product Category	Parent KB Article	Vulnerability Description	Guest OS	Date First Introduced
Rel 18-07	4338824	Windows Security only	4.56	July 10, 2018
Rel 18-07	4345459	Security rollup refresh	2.76	July 10, 2018
Rel 18-07	4345425	Security rollup refresh	3.63	July 10, 2018
Rel 18-07	4345424	Security rollup refresh	4.56	July 10, 2018
Rel 18-07	4345418	Security rollup refresh	5.21	July 10, 2018
Rel 18-07	4338612, 4338602	.NET 3.5, 4.x Security	2.76	July 10, 2018
Rel 18-07	4338601, 4338604	.NET 3.5, 4.x, 4.5x Security	3.63	July 10, 2018
Rel 18-07	4338613, 4338600, 4338605	.NET 3.5, 4.x, 4.5x Security	4.56	July 10, 2018
Rel 18-07	4338832	Flash	3.63, 4.76, 5.21	July 10, 2018
Rel 18-07	4339093	Internet Explorer	2.76, 3.63, 4.76	July 10, 2018
N/A	4284826	June non-security rollup	2.76	June 12, 2018
N/A	4284855	June non-security rollup	3.63	June 12, 2018
N/A	4284815	June non-security rollup	4.56	June 12, 2018

June 2018 Guest OS

Product Category	Parent KB Article	Vulnerability Description	Guest OS	Date First Introduced
Rel 18-06	4284880	Windows 10 Security	5.20	June 12, 2018
Rel 18-06	4284867	Windows Security only	2.75	June 12, 2018
Rel 18-06	4284846	Windows Security only	3.62	June 12, 2018
Rel 18-06	4284878	Windows Security only	4.55	June 12, 2018
Rel 18-06	4230450	Internet Explorer	2.75, 3.62, 4.75	June 12, 2018
Rel 18-06	4287903	Flash	3.62, 4.75, 5.20	June 12, 2018

Product Category	Parent KB Article	Vulnerability Description	Guest OS	Date First Introduced
N/A	4103718	May non-security rollup	2.75	May 8, 2018
N/A	4103730	May non-security rollup	3.62	May 8, 2018
N/A	4103725	May non-security rollup	4.55	May 8, 2018
N/A	4040980, 4040977	Sept '17 .NET non-security rollup	2.75	November 14, 2017
N/A	4095874	May .NET 3.5 non-security release	2.75	May 8, 2018
N/A	4096495	May .NET 4.x non-security release	2.75	May 8, 2018
N/A	4040975	Sept '17 .NET non-security rollup	3.62	November 14, 2017
N/A	4095872	May .NET 3.5 non-security release	3.62	May 8, 2018
N/A	4096494	May .NET 4.x non-security release	3.62	May 8, 2018
N/A	4096416	May .NET 4.5x non-security release	3.62	May 8, 2018
N/A	4040974, 4040972	Sept '17 .NET non-security rollup	4.55	November 14, 2017
N/A	4043763	Oct '17 .NET non-security rollup	4.55	September 12, 2017
N/A	4095876	May .NET 4.x non-security release	4.55	May 8, 2018
N/A	4096417	May .NET 4.5x non-security release	4.55	May 8, 2018
N/A	4132216	May SSU	5.20	May 8, 2018

May 2018 Guest OS

Product Category	Parent KB Article	Vulnerability Description	Guest OS	Date First Introduced
Rel 18-05	4103721, 4103727, 4103723	Windows 10 Security	5.19	May 8, 2018

Product Category	Parent KB Article	Vulnerability Description	Guest OS	Date First Introduced
Rel 18-05	4103712	Windows Security only	2.74	May 8, 2018
Rel 18-05	4103726	Windows Security only	3.61	May 8, 2018
Rel 18-05	4103715	Windows Security only	4.54	May 8, 2018
Rel 18-05	4095514, 4095519	.NET 3.5, 4.x Security	2.74	May 8, 2018
Rel 18-05	4095512, 4095518, 4096235	.NET 3.5, 4.x, 4.5x Security	3.61	May 8, 2018
Rel 18-05	4095515, 4095517, 4096236	.NET 3.5, 4.x, 4.5x Security	4.74	May 8, 2018
Rel 18-05	4054856	.NET 4.7x Security	5.19	May 8, 2018
Rel 18-05	4103768	Internet Explorer	2.74, 3.61, 4.74	May 8, 2018
Rel 18-05	4103729	Flash	3.61, 4.74, 5.19	May 8, 2018
N/A	4093118	April non-security rollup	2.73	April 10, 2018
N/A	4093123	April non-security rollup	3.61	April 10, 2018
N/A	4093114	April non-security rollup	4.74	April 10, 2018
N/A	4093137	April SSU	5.19	April 10, 2018
N/A	4093753	Timezone update	2.74, 3.61, 4.74	April 10, 2018

April 2018 Guest OS

Product Category	Parent KB Article	Vulnerability Description	Guest OS	Date First Introduced
Rel 18-04	4093119	Windows 10 Security	5.18	April 10, 2018
Rel 18-04	4093108	Windows Security only	2.73	April 10, 2018
Rel 18-04	4093122	Windows Security only	3.60	April 10, 2018
Rel 18-04	4093115	Windows Security only	4.53	April 10, 2018

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 18-04	4092946	Internet Explorer	2.73, 3.60, 4.53	April 10, 2018
Rel 18-04	4093110	Flash	3.60, 4.53, 5.18	April 10, 2018
N/A	4088875	March non-security rollup	2.73	March 13, 2018
N/A	4099950	March non-security rollup pre-requisite	2.73	March 13, 2018
N/A	4088877	March non-security rollup	3.60	March 13, 2018
N/A	4088876	March non-security rollup	4.53	March 13, 2018

March 2018 Guest OS

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 18-03	4088787, 4088776	Windows 10 Security	5.17	March 13, 2018
Rel 18-03	4088878, 4088880, 4088879	Windows Security only	2.72, 3.59, 4.52	March 13, 2018
Rel 18-03	4089187	Internet Explorer	2.72, 3.59, 4.52	March 13, 2018
Rel 18-03	4074595	Flash	3.59, 4.52, 5.17	March 13, 2018
N/A	4074598	February non-security rollup	2.72	February 13, 2018
N/A	4074593	February non-security rollup	3.59	February 13, 2018
N/A	4074594	February non-security rollup	4.52	February 13, 2018
N/A	4074837	Timezone update	2.72, 3.59, 4.52	February 13, 2018

February 2018 Guest OS

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 18-02	4074590, 4074588	Windows 10 Security	5.16	February 13, 2018
Rel 18-02	4074587, 4074589, 4074597	Windows Security only	2.71, 3.58, 4.51	February 13, 2018

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 18-02	4074736	Internet Explorer	2.71, 3.58, 4.51	February 13, 2018
Rel 18-02	4074595	Flash	3.58, 4.51, 5.16	February 13, 2018
N/A	4056894	January non-security rollup	2.71	January 4, 2018
N/A	4056896	January non-security rollup	3.58	January 4, 2018
N/A	4056895	January non-security rollup	4.51	January 4, 2018
N/A	4054176, 4054172	January .NET rollup	2.71	January 4, 2018
N/A	4054175, 4054171	January .NET rollup	3.58	January 4, 2018
N/A	4054177, 4054170	January .NET rollup	4.51	January 4, 2018

January 2018 Guest OS

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 18-01	4056898, 4056897, 4056899	Windows Security only	2.70, 3.57, 4.50	January 3, 2018
Rel 18-01	4056890, 4056892	Windows Security only	5.15	January 3, 2018
N/A	4054518	December non-security rollup	2.70	December 12, 2017
N/A	4054520	December non-security rollup	3.57	December 12, 2017
N/A	4054519	December non-security rollup	4.50	December 12, 2017
N/A	4051956	January timezone update	2.70, 3.57, 4.50	December 12, 2017

December 2017 Guest OS

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 17-12	4053579, 4054517	Windows 10 Security updates	4.49, 5.14	December 12, 2017

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 17-12	4054521 , 4054522 , 4054523	Windows Security only	2.69, 3.56, 4.49	December 12, 2017
Rel 17-12	4052978	Internet Explorer	2.69, 3.56, 4.49	December 12, 2017
Rel 17-12	4052978	Flash	3.56, 4.49, 5.14	December 12, 2017
N/A	4048957	November non-security rollup	2.69	November 14, 2017
N/A	4048959	November non-security rollup	3.56	November 14, 2017
N/A	4048958	November non-security rollup	4.49	November 14, 2017
N/A	4049068	December Timezone update	2.69, 3.56, 4.49	December 12, 2017

November 2017 Guest OS

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 17-11	4048953	Windows 10 updates	5.13	November 14, 2017
Rel 17-11	4048960 , 4048962 , 4048961	Windows Security only	2.68, 3.55, 4.48	November 14, 2017
Rel 17-11	4047206	Internet Explorer	2.68, 3.55, 4.48	November 14, 2017
Rel 17-11	4048951	Flash	3.55, 4.48, 5.13	November 14, 2017
N/A	4041681	October non-security rollup	2.68	October 10, 2017
N/A	4041690	October non-security rollup	3.55	October 10, 2017
N/A	4041693	October non-security rollup	4.48	October 10, 2017
N/A	3191566	Update for Windows Management Framework 5.1	2.68	November 14, 2017
N/A	3191565	Update for Windows Management Framework 5.1	3.55	November 14, 2017

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
N/A	3191564	Update for Windows Management Framework 5.1	4.48	November 14, 2017

October 2017 Guest OS

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 17-10	4041691	Windows 10 updates	5.12	October 10, 2017
Rel 17-10	4041678, 4041679, 4041687	Windows Security only	2.67, 3.54, 4.47	October 10, 2017
Rel 17-10	4040685,	Internet Explorer	2.67, 3.54, 4.47	October 10, 2017
Rel 17-10	4041681, 4041690, 4041693	Windows Monthly Rollups	2.67, 3.54, 4.47	October 10, 2017
N/A	4038777	September non-security rollup	2.67	September 12, 2017
N/A	4038799	September non-security rollup	3.54	September 12, 2017
N/A	4038792	September non-security rollup	4.47	September 12, 2017
N/A	4040980	September .NET non-security rollup	2.67	September 12, 2017
N/A	4040979	September .NET non-security rollup	3.54	September 12, 2017
N/A	4040981	September .NET non-security rollup	4.47	September 12, 2017

September 2017 Guest OS

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 17-09	4038782	Windows 10 updates	5.11	September 12, 2017
Rel 17-09	4038779, 4038786, 4038793	Windows Security only	2.66, 3.53, 4.46	September 12, 2017
Rel 17-09	4040966, 4040960, 4040965, 4040959, 4033988, 4040955, 4040967, 4040958	September .NET update	2.66, 3.53, 4.46	September 12, 2017

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 17-09	4036586	Internet explorer	2.66, 3.53, 4.46	September 12, 2017
CVE-2017-8704	4038782	Denial of Service	5.11	September 12, 2017
N/A	4034664	August non-security rollup	2.66	August 8, 2017
N/A	4034665	August non-security rollup	5.11	August 8, 2017
N/A	4034681	August non-security rollup	4.46	August 8, 2017

August 2017 Guest OS

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 17-07	4034658	Windows 10 updates	5.10	August 8, 2017
Rel 17-07	4034679	Windows Security only	2.65	August 8, 2017
Rel 17-07	4034672	Windows Security only	4.45	August 8, 2017
Rel 17-07	4034666	Windows Security only	3.52	August 8, 2017
Rel 17-07	4034733	Internet Explorer	2.65, 3.52, 4.45, 5.10	August 8, 2017
Rel 17-07	4034664, 4034665, 4034681	Windows Monthly Rollups	2.65, 3.52, 4.45	August 8, 2017
Rel 17-07	4034668, 4034660, 4034658, 4034674	Re-release of CVE-2017-0071, Re-release of CVE-2017-0228	5.10	August 8, 2017
Rel 17-07	4025341	July non-security rollup	2.65	July 11, 2017
Rel 17-07	4025331	July non-security rollup	3.52	July 11, 2017
Rel 17-07	4025336	July non-security rollup	4.45	July 11, 2017

July 2017 Guest OS

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 17-07	4025339	Windows 10 updates	5.9	July 11, 2017
Rel 17-07	4025337	Windows Security only	2.64	July 11, 2017
Rel 17-07	4025333	Windows Security only	4.44	July 11, 2017
Rel 17-07	4025343	Windows Security only	3.51	July 11, 2017
Rel 17-07	4025376	Flash	3.51, 4.44, 5.9	July 11, 2017
Rel 17-07	4025252	Internet Explorer	2.64, 3.51, 4.44	July 11, 2017
N/A	4020322	Timezone Update	2.64, 3.51, 4.44	July 11, 2017
N/A	4022719	June non-security rollup	2.64	June 13, 2017
N/A	4022724	June non-security rollup	3.51	June 13, 2017
N/A	4022726	June non-security rollup	4.44	June 13, 2017

June 2017 Guest OS

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 17-06	4019472	WS 2016 OS Quality Improvements	5.8	May 9, 2017
Rel 17-06	4022722	Windows Security only	2.63	June 13, 2017
Rel 17-06	4022717	Windows Security only	4.43	June 13, 2017
Rel 17-06	4022718	Windows Security only	3.50	June 13, 2017
Rel 17-06	4021558	Internet Explorer	2.63, 3.50, 4.43	June 13, 2017
Rel 17-06	4022719, 4022724, 4022726	Windows Monthly Rollups	2.63, 3.50, 4.43	June 13, 2017
Rel 17-06	4022730	Security update for Adobe Flash Player	3.50, 4.43, 5.8	June 13, 2017

Product Category	Parent KB Article	Vulnerability Description	Guest OS	Date First Introduced
Rel 17-06	4015217 , 4015221 , 4015583 , 4015550 , 4015219	Re-release of CVE-2017-0167	4.43, 5.8	April 11, 2017
N/A	4023136	Timezone update	2.63, 3.50, 4.43	June 13, 2017
N/A	4019264	May non-security rollup	2.63	June 13, 2017
N/A	4014545	May .NET non-security rollup	2.63	April 11, 2017
N/A	4014508	May .NET non-security rollup	2.63	May 9, 2017
N/A	4014511	May .NET non-security rollup	2.63	May 9, 2017
N/A	4014514	May .NET non-security rollup	2.63	May 9, 2017
N/A	4019216	May non-security rollup	3.50	May 9, 2017
N/A	4014503	May .NET non-security rollup	3.50	May 9, 2017
N/A	4014506	May .NET non-security rollup	3.50	May 9, 2017
N/A	4014509	May .NET non-security rollup	3.50	May 9, 2017
N/A	4014513	May .NET non-security rollup	3.50	May 9, 2017
N/A	4019215	May non-security rollup	4.43	May 9, 2017
N/A	4014505	May .NET non-security rollup	4.43	May 9, 2017
N/A	4014507	May .NET non-security rollup	4.43	May 9, 2017
N/A	4014510	May .NET non-security rollup	4.43	May 9, 2017
N/A	4014512	May .NET non-security rollup	4.43	May 9, 2017

May 2017 Guest OS

Product Category	Parent KB Article	Vulnerability Description	Guest OS	Date First Introduced
Rel 17-05	4019472	WS 2016 OS Quality Improvements	5.7	May 9, 2017
Rel 17-05	4019263	Windows Security only	2.62	May 9, 2017
Rel 17-05	4019213	Windows Security only	4.42	May 9, 2017
Rel 17-05	4019214	Windows Security only	3.49	May 9, 2017
Rel 17-05	4018271	Cumulative security update for Internet Explorer	3.49	May 9, 2017
Rel 17-05	4010323	SHA1 Advisory	2.62, 4.42, 5.7	May 9, 2017
Rel 17-05	4022344	Microsoft Security Advisory	5.7	May 9, 2017
Rel 17-05	4022345	Microsoft Security Advisory	5.7	May 9, 2017
Rel 17-05	4021279	.Net /ASP.Net Core Advisory	2.62, 3.49, 4.42, 5.7	May 9, 2017
N/A	4012864	Timezone Update	2.62, 3.49, 4.42	May 9, 2017
N/A	4014565	April .NET non-security rollup	2.62	April 11, 2017
N/A	4014559	April .NET non-security rollup	2.62	April 11, 2017
N/A	4015549	April non-Security Rollup	2.62	April 11, 2017
N/A	4019990	D3DCompiler update - requirement for .NET 4.7	3.49	May 9, 2017
N/A	4014563	April .NET non-security rollup	3.49	April 11, 2017
N/A	4014557	April .NET non-security rollup	3.49	April 11, 2017
N/A	4014545	April .NET non-security rollup	3.49	April 11, 2017
N/A	4014548	April .NET non-security rollup	3.49	April 11, 2017

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
N/A	4015551	April non-security rollup	3.49	April 11, 2017
N/A	3173424	Servicing Stack Update	4.42	July 12, 2016
N/A	4014555	April .NET non-security rollup	4.42	April 11, 2017
N/A	4014567	April .NET non-security rollup	4.42	April 11, 2017
N/A	4015550	April non-security rollup	4.42	April 11, 2017
N/A	4013418	Servicing Stack Update	5.7	March 14, 2017

April 2017 Guest OS

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 17-04	4015217	WS 2016 OS Quality Improvements	5.6	April 11, 2017
Rel 17-04	4015546	Windows Security only	2.61	April 11, 2017
Rel 17-04	4015547	Windows Security only	4.41	April 11, 2017
Rel 17-04	4015548	Windows Security only	3.48	April 11, 2017
Rel 17-04	4014661	Internet explorer	2.61, 3.48, 4.41	April 11, 2017
Rel 17-04	4014550; 4014560; 4014562; 4014556; 4014574	.NET Security	4.41	April 11, 2017
Rel 17-04	4014564; 4014572; 4014549	.NET Security	3.48	April 11, 2017
Rel 17-04	4014566; 4014552; 4014573; 4014558	.NET Security	2.61	April 11, 2017
Rel 17-04	4015546; 4015547; 4015548; 4015217	CVE-2017-0181	5.6	April 11, 2017

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 17-04	4015546 ; 4015547 ; 4015548 ; 4015217	CVE-2017-0163; CVE-2017-0183; CVE-2017-0184; CVE-2017-0184; CVE-2017-0185; CVE-2017-0168	2.61, 3.48, 4.41	April 11, 2017
Rel 17-04	4015546 ; 4015547 ; 4015548 ; 4015217	CVE-2017-0178; CVE-2017-0179; CVE-2017-0162; CVE-2017-0169	4.41	April 11, 2017
Rel 17-04	4015546 ; 4015547 ; 4015548 ; 4015217	CVE-2017-0182; CVE-2017-0186; CVE-2017-0191	2.61, 3.48, 4.41, 5.6	April 11, 2017
Rel 17-04	4015193	DST changes in Windows for Magallanes (Chile)	2.61, 3.48, 4.41	April 11, 2017
Rel 17-04	4012215	March 2017 Security Monthly Quality Rollup for Windows 7 SP1 and Windows Server 2008 R2 SP1	2.61	April 11, 2017
Rel 17-04	4012217	March 2017 Security Monthly Quality Rollup for Windows Server 2012	3.48	April 11, 2017
Rel 17-04	4012216	March 2017 Security Monthly Quality Rollup for Windows 8.1 and Windows Server 2012 R2	4.41	April 11, 2017

March 2017 Guest OS

PRODUCT CATEGORY	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
Rel 17-03	4013429	Improvements and fixes	5.5	March 14, 2017
Rel 17-03	4012212	March 2017 Security Only Quality Update for Windows 7 SP1 and Windows Server 2008 R2 SP1	2.60	March 14, 2017

Product Category	Parent KB Article	Vulnerability Description	Guest OS	Date First Introduced
Rel 17-03	4012213	March 2017 Security Only Quality Update for Windows 8.1 and Windows Server 2012 R2	4.40	March 14, 2017
Rel 17-03	4012214	March 2017 Security Only Quality Update for Windows Server 2012	3.47	March 14, 2017
Rel 17-03	4012204	Security update for Internet Explorer	2.60, 3.47, 4.40, 5.5	March 14, 2017
N/A	4012864	DST changes in Windows for Northern Cypress, Mongolia, and Russian Saratov region	2.60, 3.47, 4.40	March 14, 2017
N/A	3212646	January 2017 Security Monthly Quality Rollup for Windows 7 SP1 and Windows Server 2008 R2 SP1	2.60	March 14, 2017
N/A	3205409	December 2016 Security Monthly Quality Rollup for Windows Server 2012	3.47	March 14, 2017
N/A	3205401	December 2016 Security Monthly Quality Rollup for Windows 8.1 and Windows Server 2012 R2	4.40	March 14, 2017
N/A	3211320	Servicing stack update for Windows 10 Version 1607 and Windows Server 2016: January 24, 2017	5.5	March 14, 2017

January 2017 Guest OS

Bulletin ID	Parent KB Article	Vulnerability Description	Guest OS	Date First Introduced
MS17-004	3216771	Security update for Local Security Authority Subsystem Service	2.59	Jan 10, 2017

December 2016 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS16-144	3204059	Cumulative Security Update for Internet Explorer	2.58, 3.46, 4.39	Dec 13, 2016
MS16-145	3204062	Cumulative Security Update for Microsoft Edge	5.4	Dec 13, 2016
MS16-146	3204066	Security Update for Microsoft Graphics Component	2.58, 3.46, 4.39, 5.4	Dec 13, 2016
MS16-147	3204063	Security Update for Microsoft Uniscribe	2.58, 3.46, 4.39, 5.4	Dec 13, 2016
MS16-149	3205655	Security Update for Microsoft Windows	2.58, 3.46, 4.39, 5.4	Dec 13, 2016
MS16-150	3205642	Security Update for Secure Kernel Mode	5.4	Dec 13, 2016
MS16-151	3205651	Security Update for Kernel-Mode Drivers	2.58, 3.46, 4.39, 5.4	Dec 13, 2016
MS16-152	3199709	Security Update for Windows Kernel	5.4	Dec 13, 2016
MS16-153	3207328	Security Update for Common Log File System Driver	2.58, 3.46, 4.39, 5.4	Dec 13, 2016
MS16-155	3205640	Security Update for .NET Framework	5.4	Dec 13, 2016
N/A	3197868	November 2016 Security Monthly Quality Rollup for Windows 7 SP1 and Windows Server 2008 R2 SP1	2.58	Dec 13, 2016
N/A	3197877	November 2016 Security Monthly Quality Rollup for Windows Server 2012	3.46	Dec 13, 2016
N/A	3197874	November 2016 Security Monthly Quality Rollup for Windows 8.1 and Windows Server 2012 R2	4.39	Dec 13, 2016

November 2016 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS16-129	3199057	Cumulative Security Update for Microsoft Edge	5.3	Nov 8, 2016
MS16-130	3199172	Security Update for Microsoft Windows	5.3, 4.38, 3.45, 2.57	Nov 8, 2016
MS16-131	3199151	Security Update for Microsoft Video Control	5.3, 4.38, 3.45, 2.57	Nov 8, 2016
MS16-132	3199120	Security Update for Microsoft Component	5.3, 4.38, 3.45, 2.57	Nov 8, 2016
MS16-134	3193706	Security Update for Common Log File System Driver	5.3, 4.38, 3.45, 2.57	Nov 8, 2016
MS16-135	3199135	Security Update for Kernel-Mode Drivers	5.3, 4.38, 3.45, 2.57	Nov 8, 2016
MS16-137	3199173	Security Update for Windows Authentication Methods	5.3, 4.38, 3.45, 2.57	Nov 8, 2016
MS16-138	3199647	Security Update to Microsoft Virtual Hard Drive	5.3, 4.38, 3.45	Nov 8, 2016
MS16-139	3199720	Security Update for Windows Kernel	2.57	Nov 8, 2016
MS16-140	3193479	Security Update For Boot Manager	5.3, 4.38, 3.45	Nov 8, 2016
MS16-142	3198467	Cumulative Security Update for Internet Explorer	2.57, 4.38, 5.3	Nov 8, 2016
N/A	3192321	Turkey ends DST observance	5.3, 4.38, 3.45, 2.57	Nov 8, 2016
N/A	3185330	October 2016 security monthly quality rollup for Windows 7 SP1 and Windows Server 2008 R2 SP1	2.57	Nov 8, 2016

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
N/A	3192403	October 2016 Preview of Monthly Quality Rollup for Windows 7 SP1 and Windows Server 2008 R2 SP1	2.57	Nov 8, 2016
N/A	3177467	Servicing stack update for Windows 7 SP1 and Windows Server 2008 R2 SP1: September 20, 2016	2.57	Nov 8, 2016
N/A	3185332	October 2016 security monthly quality rollup for Windows Server 2012	3.45	Nov 8, 2016
N/A	3192406	October 2016 Preview of Monthly Quality Rollup for Windows Server 2012	3.45	Nov 8, 2016
N/A	3185331	October 2016 security monthly quality rollup for Windows 8.1 and Windows Server 2012 R2	4.38	Nov 8, 2016
N/A	3192404	October 2016 Preview of Monthly Quality Rollup for Windows 8.1 and Windows Server 2012 R2	4.38	Nov 8, 2016
N/A	3199986	Servicing stack update for Windows 10 Version 1607: October 27, 2016	5.3	Nov 8, 2016
N/A	3197954	Cumulative Update for Windows 10 Version 1607 and Windows Server 2016: October 27, 2016	5.3	Nov 8, 2016

October 2016 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
-------------	-------------------	---------------------------	----------	-----------------------

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS16-118	3192887	Cumulative Security Update for Internet Explorer	4.37, 3.44, 2.56	Oct 11, 2016
MS16-120	3192884	Security Update for Microsoft Graphics Component	4.37, 3.44, 2.56	Oct 11, 2016
MS16-123	3192892	Security Update for Kernel-Mode Drivers	4.37, 3.44, 2.56	Oct 11, 2016
MS16-124	3193227	Security Update for Windows Registry	4.37, 3.44, 2.56	Oct 11, 2016
MS16-126	3196067	Security Update for Microsoft Internet Messaging API	2.56	Oct 11, 2016
MS16-101	3178465	Security Update for Windows Authentication Methods	4.37, 3.44, 2.56	Oct 11, 2016
N/A	3182203	September 2016 time zone change for Novosibirsk	4.37, 3.44, 2.56	Oct 11, 2016
N/A	3185278	September 2016 update rollup for Windows 7 SP1 and Windows Server 2008 R2 SP1	2.56	Oct 11, 2016
N/A	3185280	September 2016 update rollup for Windows Server 2012	3.44	Oct 11, 2016
N/A	3185279	September 2016 update rollup for Windows 8.1 and Windows Server 2012 R2	4.37	Oct 11, 2016
N/A	3194798	Cumulative update for Windows 10 Version 1607 and Windows Server 2016	5.2	Oct 11, 2016

September 2016 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
-------------	-------------------	---------------------------	----------	-----------------------

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS16-104	3183038	Cumulative Security Update for Internet Explorer	4.36, 3.43, 2.55	Sept 13, 2016
MS16-106	3185848	Security Update for Microsoft Graphics Component	4.36, 3.43, 2.55	Sept 13, 2016
MS16-110	3178467	Security Update for Windows	4.36, 3.43, 2.55	Sept 13, 2016
MS16-111	3186973	Security Update for Windows Kernel	4.36, 3.43, 2.55	Sept 13, 2016
MS16-112	3178469	Security Update for Windows Lock Screen	4.36	Sept 13, 2016
MS16-114	3185879	Security Update for Windows SMB Server	4.36, 3.43, 2.55	Sept 13, 2016
MS16-115	3188733	Security Update for PDF	4.35, 3.43	Sept 13, 2016
MS16-116	3188724	Security Update in OLE Automation for VBScript Scripting Engine	4.36, 3.43, 2.55	Sept 13, 2016
N/A	3174644	Updated Support for Diffie-Hellman Key Exchange	4.36, 3.43, 2.55	Sept 13, 2016
N/A	3177723	Timezone Update - Egypt cancels DST	4.36, 3.43, 2.55	Sept 13, 2016
N/A	3179573	August 2016 update rollup for Windows 7 SP1 and Windows Server 2008 R2 SP1	2.55	Sept 13, 2016
N/A	3179575	August 2016 update rollup for Windows Server 2012	3.43	Sept 13, 2016
N/A	3179574	August 2016 update rollup for Windows RT 8.1, Windows 8.1, and Windows Server 2012 R2	4.36	Sept 13, 2016

August 2016 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS16-095	3177356	Cumulative Security Update for Internet Explorer	4.35, 3.42, 2.54	Aug 9, 2016
MS16-097	3177393	Security Update for Microsoft Graphics Component	4.35, 3.42, 2.54	Aug 9, 2016
MS16-098	3178466	Security Update for Windows Kernel-Mode Drivers	4.35, 3.42, 2.54	Aug 9, 2016
MS16-100	3179577	Security Update for Secure Boot	4.35, 3.42	Aug 9, 2016
MS16-101	3178465	Security Update for Windows Authentication Methods	4.35, 3.42, 2.54	Aug 9, 2016
MS16-102	3182248	Security Update for Microsoft Windows PDF Library	4.35, 3.42	Aug 9, 2016
MS16-077 Re-Release	3165191	Security Update for Web Proxy Autodiscovery (WPAD)	4.35, 3.42, 2.54	Aug 9, 2016
N/A	3172605	July 2016 update rollup for Windows 7 SP1 and Windows Server 2008 R2 SP1	2.54	Aug 9, 2016
N/A	3172615	July 2016 update rollup for Windows Server 2012	3.42	Aug 9, 2016
N/A	3172614	July 2016 update rollup for Windows RT 8.1, Windows 8.1, and Windows Server 2012 R2	4.35	Aug 9, 2016

July 2016 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS16-084	3169991	Cumulative Security Update for Internet Explorer	4.34, 3.41, 2.53	July 12, 2016

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS16-087	3170005	Security Update for Microsoft Print Spooler	4.34, 3.41, 2.53	July 12, 2016
MS16-090	3171481	Security Update for Kernel Mode Drivers	4.34, 3.41, 2.53	July 12, 2016
MS16-091	3170048	Security Update for .NET Framework	4.34, 3.41, 2.53	July 12, 2016
MS16-092	3171910	Security Update for Windows Kernel	4.34, 3.41	July 12, 2016
MS16-094	3177404	Security Update for Secure Boot	4.34, 3.41	July 12, 2016
N/A	3162835	June 2016 DST and time zone update for Windows	4.34, 3.41, 2.53	July 12, 2016
N/A	3156417	May 2016 update rollup for Windows 7 SP1 and Windows Server 2008 R2 SP1	2.53	July 12, 2016
N/A	3161608	June 2016 update rollup for Windows 7 SP1 and Windows Server 2008 R2 SP1	2.53	July 12, 2016
N/A	3161609	June 2016 update rollup for Windows Server 2012	3.41	July 12, 2016
N/A	3161606	June 2016 update rollup for Windows RT 8.1, Windows 8.1, and Windows Server 2012 R2	4.34	July 12, 2016
N/A	3139923	Windows installer (MSI) repair doesn't work when MSI package is installed on an HTTP share in Windows	4.34	July 12, 2016

June 2016 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
-------------	-------------------	---------------------------	----------	-----------------------

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS16-035	3141780	Security Update for .NET Framework to Address Security Feature Bypass	4.33, 3.40, 2.52	June 14, 2016
Advisory	3155527	Update to Cipher Suites for FalseStart	4.33, 3.40	June 14, 2016
MS16-063	3163649	Cumulative Security Update for Internet Explorer	4.33, 3.40, 2.52	June 14, 2016
MS16-069	3163640	Cumulative Security Update for JScript and VBScript	2.52	June 14, 2016
MS16-071	3164065	Security Update for Microsoft Windows DNS Server	4.33, 3.40	June 14, 2016
MS16-072	3163622	Security Update for Group Policy	4.33, 3.40, 2.52	June 14, 2016
MS16-073	3164028	Security Update for Kernel Mode Drivers	4.33, 3.40, 2.52	June 14, 2016
MS16-074	3164036	Security Update for Microsoft Graphics Component	4.33, 3.40, 2.52	June 14, 2016
MS16-075	3164038	Security Update for Windows SMB Server	4.33, 3.40, 2.52	June 14, 2016
MS16-076	3167691	Security Update for Netlogon	4.33	June 14, 2016
MS16-077	3165191	Security Update for WPAD	4.33, 3.40, 2.52	June 14, 2016
MS16-080	3164302	Security Update for Microsoft Windows PDF	4.33, 3.40	June 14, 2016
MS16-081	3160352	Security Update for Active Directory	4.33, 3.40, 2.52	June 14, 2016
N/A	2922223	You cannot change system time if RealTimelUniversal registry entry is enabled in Windows	2.52	June 14, 2016

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
N/A	3121255	"0x00000024" Stop error in FsRtlNotifyFilterReportChange and copy file may fail in Windows	2.52	June 14, 2016
N/A	3125424	LSASS deadlocks cause Windows Server 2012 R2 or Windows Server 2012 not to respond	4.33, 3.40	June 14, 2016
N/A	3125574	Convenience rollup update for Windows 7 SP1 and Windows Server 2008 R2 SP1	2.52	June 14, 2016
N/A	3140245	Update to enable TLS 1.1 and TLS 1.2 as a default secure protocols in WinHTTP in Windows	3.40, 2.52	June 14, 2016
N/A	3146604	WMI service crashes randomly in Windows Server 2012 R2 or Windows Server 2012	4.33, 3.40	June 14, 2016
N/A	3149157	Reliability and scalability improvements in TCP/IP for Windows 8.1 and Windows Server 2012 R2	4.33	June 14, 2016
N/A	3156416	May 2016 update rollup for Windows Server 2012	3.40	June 14, 2016
N/A	3156418	May 2016 update rollup for Windows RT 8.1, Windows 8.1, and Windows Server 2012 R2	4.33	June 14, 2016
N/A	3153731	May 2016 DST update for Azerbaijan, Chile, Haiti, and Morocco in Windows	4.33, 3.40, 2.52	June 14, 2016

May 2016 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS16-051	3155533	Cumulative Security Update for Internet Explorer	4.32, 3.39, 2.51	May 10, 2016
MS16-053	3156764	Cumulative Security Update for JScript and VBScript	2.51	May 10, 2016
MS16-055	3156754	Security Update for Microsoft Graphics Component	4.32, 3.39, 2.51	May 10, 2016
MS16-057	3156987	Security Update for Windows Shell	4.32	May 10, 2016
MS16-060	3154846	Security Update for Windows Kernel	4.32, 3.39, 2.51	May 10, 2016
MS16-061	3155520	Security Update to RPC	4.32, 3.39, 2.51	May 10, 2016
MS16-062	3158222	Security Update for Kernel Mode Drivers	4.32, 3.39, 2.51	May 10, 2016
MS16-065	3156757	Security Update for .Net Framework	4.32, 3.39, 2.51	May 10, 2016
MS16-067	3155784	Security Update for Volume Manager Driver	4.32, 3.39	May 10, 2016
N/A	3148851	Time zone changes for Russia in Windows	4.32, 3.39, 2.51	May 10, 2016
N/A	3133977	BitLocker can't encrypt drives because of service crashes in svchost.exe process in Windows 7 or Windows Server 2008 R2	2.51	May 10, 2016
N/A	3133681	Virtual machines don't respond to your operation in SCVMM in Windows Server 2012 R2	4.32	May 10, 2016
N/A	3123245	Update improves port exhaustion identification in Windows Server 2012 R2	4.32	May 10, 2016

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
N/A	Disable RC4	Microsoft security advisory: Update for disabling RC4	4.32, 3.39, 2.51	May 10, 2016

April 2016 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS16-037	3148531	Cumulative Security Update for Internet Explorer	4.31, 3.38, 2.50	April 12, 2016
MS16-039	3148522	Security Update for Microsoft Graphics Component	4.31, 3.38, 2.50	April 12, 2016
MS16-040	3148541	Security Update for Microsoft XML Core Service	4.31, 3.38, 2.50	April 12, 2016
MS16-044	3146706	Security Update for Windows OLE	4.31, 3.38, 2.50	April 12, 2016
MS16-045	3143118	Security Update for Windows Hyper-V	4.31, 3.38	April 12, 2016
MS16-047	3148527	Security Update for Security Account Manager Remote Protocol	4.31, 3.38, 2.50	April 12, 2016
MS16-048	3148528	Security Update for CSRSS	4.31, 3.38	April 12, 2016

March 2016 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS16-023	3142015	Cumulative Security Update for Internet Explorer	4.30, 3.37, 2.49	March 8, 2016
MS16-026	3143148	Security Update to Graphic Fonts to Address Remote Code Execution	4.30, 3.37, 2.49	March 8, 2016

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS16-027	3143146	Security Updates for Windows Media Player to address Remote Code Execution	4.30, 3.37, 2.49	March 8, 2016
MS16-028	3143081	Security Update for Microsoft Windows PDF Library to Address Remote Code Execution	4.30, 3.37	March 8, 2016
MS16-030	3143136	Security Update for Windows OLE to Address Remote Code Execution	4.30, 3.37, 2.49	March 8, 2016
MS16-031	3140410	Security Update for Microsoft Windows to Address Elevation of Privilege	2.49	March 8, 2016
MS16-032	3143141	Security Update to Secondary Logon to Address Elevation of Privilege	4.30, 3.37, 2.49	March 8, 2016
MS16-033	3143142	Security Update to USB Mass Storage Class Driver to Address Elevation of Privilege	4.30, 3.37, 2.49	March 8, 2016
MS16-034	3143145	Security Updates for Kernel-Mode Driver to address Elevation of Privilege	4.30, 3.37, 2.49	March 8, 2016

February 2016 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS16-009	3134220	Cumulative Security Update for Internet Explorer	4.29, 3.36, 2.48	February 9, 2016
MS16-013	3134811	Security Update to Windows Journal to Address Remote Code Execution	4.29, 3.36, 2.48	February 9, 2016

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS16-014	3134228	Security update to Microsoft Windows to Address Remote Code Execution	4.29, 3.36, 2.48	February 9, 2016
MS16-016	3136041	Security Update to WebDAV to Address Elevation of Privilege	4.29, 3.36, 2.48	February 9, 2016
MS16-018	3136082	Security Update for Windows Kernel-Mode Driver to Address Elevation of Privilege	4.29, 3.36, 2.48	February 9, 2016
MS16-019	3137893	Security Update for .NET Framework to Address Remote Code Execution	4.29, 3.36, 2.48	February 9, 2016
MS16-021	3133043	Security Update for NPS RADIUS Server to Address Denial of Service	4.29, 3.36, 2.48	February 9, 2016
Microsoft Security Advisory	3109853	Update to Improve TLS Interoperability	4.29, 3.36	February 9, 2016
Re-Release - MS15-101	3089662	Vulnerabilities in .NET Framework Could Allow Elevation of Privilege	4.29, 3.36, 2.48	February 9, 2016
Re-Release - MS15-118	3104507	Security Updates for .NET Framework to Address Elevation of Privilege	4.29, 3.36, 2.48	February 9, 2016
Re-Release - MS15-128	3104503	Security Updates for Microsoft Graphics Component to Address Remote Code Execution	4.29, 3.36, 2.48	February 9, 2016

January 2016 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS16-001	3124903	Cumulative Security Update for Internet Explorer	4.28, 3.35, 2.47	January 12, 2016

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS16-003	3125540	Cumulative Security Update for JScript and VBScript	2.47	January 12, 2016
MS16-005	3124584	Security Update for Windows Kernel-Mode Drivers to Address Remote Code Execution	4.28, 3.35, 2.47	January 12, 2016
MS16-007	3124901	Security Update for Microsoft Windows to Address Remote Code Execution	4.28, 3.35, 2.47	January 12, 2016
MS16-008	3124605	Security Update for Kernel to Address Elevation of Privilege	4.28, 3.35, 2.47	January 12, 2016
Microsoft Security Advisory Revision	2755801	Update for Vulnerabilities in Adobe Flash Player in Internet Explorer (Package KB: TBD) - Advisory Placeholder	4.28, 3.35	January 12, 2016
Microsoft Security Advisory	3109853	Update to Improve TLS Interoperability	4.28, 3.35	January 12, 2016
Microsoft Security Advisory	3123479	Deprecation of SHA-1 Hashing Algorithm for Microsoft Root Certificate Program	4.28, 3.35, 2.47	January 12, 2016
Microsoft Security Advisory	2736233	Updates for ActiveX Kill Bits	4.28, 3.35, 2.47	January 12, 2016

December 2015 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-124	3116180	Cumulative Security Update for Internet Explorer	4.27, 3.34, 2.46	December 8, 2015
MS15-126	3116178	Security Update for Microsoft VBScript and JScript to Address Remote Code Execution	2.46	December 8, 2015

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-127	3100465	Security Update for Microsoft Windows DNS to Address Remote Code Execution	4.27, 3.34, 2.46	December 8, 2015
MS15-128	3104503	Security Updates for Microsoft Graphics Component to Address Remote Code Execution	4.27, 3.34, 2.46	December 8, 2015
MS15-132	3116162	Security Update for Windows to Address Remote Code Execution	4.27, 3.34, 2.46	December 8, 2015
MS15-133	3116130	Security Update for Windows PGM to Address Elevation of Privilege	4.27, 3.34, 2.46	December 8, 2015
MS15-134	3108669	Security Update for Windows Media Center to Address Remote Code Execution	4.27, 3.34, 2.46	December 8, 2015
MS15-135	3119075	Security Update for Windows Kernel Mode Drivers to Address Elevation of Privilege	4.27, 3.34, 2.46	December 8, 2015

November 2015 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-112	3104517	Cumulative Security Update for Internet Explorer	4.26, 3.33, 2.45	November 10, 2015
MS15-114	3100213	Security Update for Windows Journal to Address Remote Code Execution	2.45	November 10, 2015
MS15-115	3105864	Security Update for Microsoft Windows to Address Remote Code Execution	4.26, 3.33, 2.45	November 10, 2015

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-117	3101722	Security Update for NDIS to Address Elevation of Privilege	2.45	November 10, 2015
MS15-118	3104507	Security Updates for .NET Framework to Address Elevation of Privilege	4.26, 3.33, 2.45	November 10, 2015
MS15-119	3104521	Security Update for Winsock to Address Elevation of Privilege	4.26, 3.33, 2.45	November 10, 2015
MS15-120	3102939	Security Update for IPSec to Address Denial of Service	4.26, 3.33	November 10, 2015
MS15-121	3081320	Security Update to Schannel to Address Spoofing	4.26, 3.33, 2.45	November 10, 2015
MS15-122	3105256	Security Update for Kerberos to Address Security Feature Bypass	4.26, 3.33, 2.45	November 10, 2015
Microsoft Security Advisory	3097966	Inadvertently disclosed Digital Certificates Could Allow spoofing	4.26, 3.33, 2.45	November 10, 2015

October 2015 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-106	3096441	Cumulative Security Update for Internet Explorer	4.25, 3.32, 2.44	October 13, 2015
MS15-108	3089659	Security Update for JScript and VBScript to Address Potential Remote Code Execution	2.44	October 13, 2015
MS15-109	3096443	Security Update for Windows Shell to Address Remote Code Execution	4.25, 3.32, 2.44	October 13, 2015

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-111	3096447	Security Update for Windows Kernel to Address Elevation of Privilege	4.25, 3.32, 2.44	October 13, 2015
Microsoft Security Advisory	3092627	September 2015 update to fix Windows or application freezes after you install security update 3076895	4.25, 3.32, 2.44	October 13, 2015

September 2015 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-093	3088903	Security Update for Internet Explorer	4.24, 3.31, 2.43	September 8, 2015
MS15-094	3089548	Cumulative Security Update for Internet Explorer	4.24, 3.31, 2.43	September 8, 2015
MS15-096	3072595	Vulnerability in Active Directory Service Could Allow Denial of Service	4.24, 3.31, 2.43	September 8, 2015
MS15-097	3089656	Vulnerabilities in Microsoft Graphics Component Could Allow Elevation of Privilege	4.24, 3.31, 2.43	September 8, 2015
MS15-098	3089669	Vulnerabilities in Windows Journal Could Allow Remote Code Execution	4.24, 3.31, 2.43	September 8, 2015
MS15-101	3089662	Vulnerabilities in .NET Framework Could Allow Elevation of Privilege	4.24, 3.31, 2.43	September 8, 2015
MS15-102	3089657	Vulnerabilities in Windows Task Management Could Allow Elevation of Privilege	4.24, 3.31, 2.43	September 8, 2015

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-105	3091287	Vulnerability in Windows Hyper-V Could Allow Security Feature Bypass	4.24	September 8, 2015

August 2015 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-079	3082442	Cumulative Security Update for Internet Explorer	4.23, 3.30, 2.42	August 11, 2015
MS15-080	3078662	Vulnerabilities in Microsoft Graphics Component Could Allow Remote Code Execution	4.23, 3.30, 2.42	August 11, 2015
MS15-082	3080348	Vulnerabilities in RDP could allow Remote Code Execution	4.23, 3.30, 2.42	August 11, 2015
MS15-084	3080129	Vulnerabilities in XML Core Services Could Allow Information Disclosure	4.23, 3.30, 2.42	August 11, 2015
MS15-085	3082487	Vulnerability in Mount Manager Could Allow Elevation of Privilege	4.23, 3.30, 2.42	August 11, 2015
MS15-088	3082458	Unsafe Command Line Parameter Passing Could Allow Information Disclosure	4.23, 3.30, 2.42	August 11, 2015
MS15-089	3060716	Vulnerabilities in Microsoft Windows Could Allow Elevation of Privilege	4.23, 3.30, 2.42	August 11, 2015
MS15-090	3076949	Vulnerability in WebDAV Could Allow Information Disclosure	4.23, 3.30, 2.42	August 11, 2015

July 2015 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-065	3076321	Cumulative Security Update for Internet Explorer	4.22, 3.29, 2.41	July 14, 2015
MS15-066	3072604	Vulnerability in VBScript could allow Remote Code Execution	2.41	July 14, 2015
MS15-067	3073094	Vulnerability in RDP could allow Remote Code Execution	4.22, 3.29, 2.41	July 14, 2015
MS15-068	3072000	Vulnerability in Windows Hyper-V Could Allow Remote Code Execution	4.22, 3.29, 2.41	July 14, 2015
MS15-069	3072631	Vulnerabilities in Windows Could Allow Remote Code Execution	4.22, 3.29, 2.41	July 14, 2015
MS15-071	3068457	Vulnerability in NETLOGON Could Allow Spoofing	4.22, 3.29, 2.41	July 14, 2015
MS15-072	3069392	Vulnerability in Graphics Driver Could Allow Elevation of Privilege	4.22, 3.29, 2.41	July 14, 2015
MS15-073	3070102	Vulnerabilities in Kernel-Mode Driver Could allow Elevation of Privilege	4.22, 3.29, 2.41	July 14, 2015
MS15-074	3072630	Vulnerability in Windows Installer Service Could Allow Elevation of Privilege	4.22, 3.29, 2.41	July 14, 2015
MS15-075	3072633	Vulnerabilities in OLE could allow Elevation of Privilege	4.22, 3.29, 2.41	July 14, 2015
MS15-076	3067505	Vulnerability in Windows Remote Procedure Call Could Allow Elevation of Privilege	4.22, 3.29, 2.41	July 14, 2015
MS15-077	3077657	Vulnerability in ATM Font Driver Could Allow Elevation of Privilege	4.22, 3.29, 2.41	July 14, 2015

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
-------------	-------------------	---------------------------	----------	-----------------------

NA	3057154	Update to Restrict Use of DES Encryption	4.22, 3.29, 2.41	July 14, 2015
----	-------------------------	--	------------------	---------------

June 2015 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-056	3058515	Cumulative Security Update for Internet Explorer	4.21, 3.28, 2.40	June 9, 2015
MS15-057	3033890	Vulnerability in Windows Media Player Could Allow Remote Code Execution	4.21, 3.28, 2.40	June 9, 2015
MS15-060	3059317	Vulnerability in Microsoft Common Controls Could Allow Remote Code Execution	4.21, 3.28, 2.40	June 9, 2015
MS15-061	3057839	Vulnerabilities in Windows Kernel-Mode Drivers Could Allow Elevation of Privilege	4.21, 3.28, 2.40	June 9, 2015
MS15-062	3062577	Vulnerability in Active Directory Federation Services Could Allow Elevation of Privilege	4.21, 3.28, 2.40	June 9, 2015
MS15-063	3063858	Vulnerability in Windows Kernel Could Allow Elevation of Privilege	4.21, 3.28, 2.40	June 9, 2015

May 2015 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-043	3049563	Cumulative Security Update for Internet Explorer	4.20, 3.27, 2.39	May 12, 2015

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-044	3057110	Vulnerabilities in Microsoft Font Drivers Could Allow Remote Code Execution	4.20, 3.27, 2.39	May 12, 2015
MS15-045	3046002	Vulnerability in Windows Journal Could Allow Remote Code Execution	4.20, 3.27, 2.39	May 12, 2015
MS15-048	3057134	Vulnerabilities in .NET Framework Could Allow Elevation of Privilege	4.20, 3.27, 2.39	May 12, 2015
MS15-050	3055642	Vulnerability in Service Control Manager Could Allow Elevation of Privilege	4.20, 3.27, 2.39	May 12, 2015
MS15-051	3057191	Vulnerabilities in Windows Kernel-Mode Drivers Could Allow Elevation of Privilege	4.20, 3.27, 2.39	May 12, 2015
MS15-052	3050514	Vulnerability in Windows Kernel Could Allow Security Feature Bypass	4.20, 3.27, 2.39	May 12, 2015
MS15-053	3057263	Vulnerabilities in JScript and VBScript Scripting Engine Could Allow Security Feature Bypass	4.20, 3.27, 2.39	May 12, 2015
MS15-054	3051768	Vulnerability in Microsoft Management Console File Format Could Allow Denial of Service	4.20, 3.27, 2.39	May 12, 2015
MS15-055	3061518	Vulnerability in Schannel Could Allow Security Feature Bypass Important Information Disclosure	4.20, 3.27, 2.39	May 12, 2015
MS15-032	3038314	Cumulative Security Update for Internet Explorer	4.20, 3.27, 2.39	Apr 17, 2015

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-035	3046306	Vulnerability in Microsoft Graphics Component Could Allow Remote Code Execution	4.20, 3.27, 2.39	Apr 17, 2015
MS15-037	3046269	Vulnerability in Windows Task Scheduler Could Allow Elevation of Privilege	4.20, 3.27, 2.39	Apr 17, 2015
MS15-038	3049576	Vulnerabilities in Microsoft Windows Could Allow Elevation of Privilege Important	4.20, 3.27, 2.39	Apr 17, 2015
MS15-039	3046482	Vulnerability in XML Core Services Could Allow Security Feature Bypass	4.20, 3.27, 2.39	Apr 17, 2015
MS15-040	3045711	Vulnerability in ADFS Could Allow Information Disclosure	4.20, 3.27, 2.39	Apr 17, 2015
MS15-041	3048010	Vulnerability in .NET Framework Could Allow Information Disclosure	4.20, 3.27, 2.39	Apr 17, 2015
MS15-042	3047234	Vulnerability in Windows Hyper-V Could Allow Denial of Service	4.20, 3.27, 2.39	Apr 17, 2015
NA	3045755	Update to Improve PKU2U Authentication	4.20, 3.27, 2.39	Apr 17, 2015

April 2015 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-034	3042553	Vulnerability in HTTP.sys could allow remote code execution	4.19, 3.26, 2.38	Apr 17, 2015

March 2015 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-018	3032359	Cumulative Security Update for Internet Explorer	4.18, 3.25, 2.37	N/A
MS15-019	3040297	Vulnerability in VBScript Scripting Engine Could Allow Remote Code Execution	4.18, 3.25, 2.37	N/A
MS15-020	3041836	Vulnerabilities in Windows could allow Remote Code Execution	4.18, 3.25, 2.37	N/A
MS15-021	3032323	Vulnerabilities Adobe Font Driver Could Allow Remote Code Execution	4.18, 3.25, 2.37	N/A
MS15-023	3034344	Vulnerabilities in Kernel Mode Driver Could Allow Elevation of Privilege	4.18, 3.25, 2.37	N/A
MS15-024	3035132	Vulnerability in PNG Processing Could Allow Information Disclosure	4.18, 3.25, 2.37	N/A
MS15-025	3038680	Vulnerabilities in Windows Kernel could allow Elevation of Privilege	4.18, 3.25, 2.37	N/A
MS15-027	3002657	Vulnerability in NETLOGON Could Allow Spoofing	4.18, 3.25, 2.37	N/A
MS15-028	3030377	Vulnerability in Windows Task Scheduler Could Allow Security Feature Bypass	4.18, 3.25, 2.37	N/A
MS15-029	3035126	Vulnerability in Windows Photo Decoder Component Could Allow Information Disclosure	4.18, 3.25, 2.37	N/A
MS15-030	3039976	Vulnerability in Remote Desktop Protocol Could Allow Denial of Service	4.18, 3.25, 2.37	N/A

NOTE

Bulletin MS15-031 may appear as uninstalled. However, it does not apply to this Guest OS release.

February 2015 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-009	3034682	Security Update for Internet Explorer	4.17, 3.24, 2.36	N/A
MS15-010	3036220	Vulnerabilities in Windows Kernel Mode Drivers Could Allow Remote Code Execution	4.17, 3.24, 2.36	N/A
MS15-011	3000483	Vulnerability in Group Policy Could Allow Remote Code Execution	4.17, 3.24, 2.36	N/A
MS15-014	3004361	Vulnerability in SMB Could Allow Security Feature Bypass	4.17, 3.24, 2.36	N/A
MS15-015	3031432	Vulnerability in Microsoft Windows Could Allow Elevation of Privilege	4.17, 3.24, 2.36	N/A
MS15-016	3029944	Vulnerability in Microsoft Graphics Component Could Allow Information Disclosure	4.17, 3.24, 2.36	N/A
N/A	3004375	Update to Improve Windows Command Line Auditing Note: This is installed but the registry key to enable it is turned off	4.17, 3.24, 2.36	N/A

January 2015 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-001	3023266	Vulnerability in Windows AppCompatCache could allow Elevation of Privilege	4.16, 3.23, 2.35	Jan 19 2015
MS15-002	3020393	Vulnerability in Windows Telnet Service Could Cause Remote Code Execution	4.16, 3.23, 2.35	Jan 19 2015
MS15-003	3021674	Vulnerability in Windows User Profile Service could allow Elevation of Privilege	4.16, 3.23, 2.35	Jan 19 2015
MS15-004	3019978	Vulnerability in Windows Components Could Allow Elevation of Privilege	4.16, 3.23, 2.35	Jan 19 2015
MS15-005	3022777	Vulnerability in NLA Could Allow Security Feature Bypass	4.16, 3.23, 2.35	Jan 19 2015
MS15-006	3004365	Vulnerability in Windows Error Reporting could Allow Security Feature Bypass	4.16, 3.23, 2.35	Jan 19 2015
MS15-007	3014029	Vulnerability in Network Policy Server RADIUS Could Cause Denial of Service	4.16, 3.23, 2.35	Jan 19 2015
MS15-008	3019215	Vulnerability in Windows Kernel Mode Driver Could Allow Elevation of Privilege	4.16, 3.23, 2.35	Jan 19 2015
MS14-080	3008923	Cumulative Security Update for Internet Explorer	4.16, 3.23, 2.35	Jan 19 2015
MS15-002	3020393	Vulnerability in Windows Telnet Service Could Cause Remote Code Execution	4.16, 3.23, 2.35	Jan 19 2015

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
MS15-002	3020393	Vulnerability in Windows Telnet Service Could Cause Remote Code Execution	4.16, 3.23, 2.35	Jan 19 2015

December 2014 Guest OS

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
N/A	3013776	System freezes when you use a domain account to start an application	4.15, 3.22, 2.34	Jan 13 2015
N/A	3013043	File system data is corrupted on a Windows-based computer that has more than one NUMA node	4.15, 3.22, 2.34	Jan 13 2015
N/A	3012712	New data blocks initialize incorrectly when a differencing VHD is expanded	4.15, 3.22, 2.34	Jan 13 201
N/A	3004905	Windows Hyper-V improvement for Linux virtual machines that have file systems that are larger than 2 TB	4.15, 3.22, 2.34	Jan 13 2015
N/A	3004394	December 2014 update for Windows Root Certificate Program in Windows	4.15, 3.22, 2.34	Jan 13 2015
N/A	2999323	The text for event ID 17 is changed	4.15, 3.22, 2.34	Jan 13 2015
N/A	3013488	Long wait to reset WSUS server when you import CSA files in Windows Server 2012 R2 or Windows Server 2012	4.15, 3.22, 2.34	Jan 13 2015

BULLETIN ID	PARENT KB ARTICLE	VULNERABILITY DESCRIPTION	GUEST OS	DATE FIRST INTRODUCED
N/A	3012325	Windows APN database entries update for DIGI, Vodafone, and Telekom mobile operators in Windows 8.1 and Windows 8	4.15, 3.22, 2.34	Jan 13 2015
N/A	3007054	PIN-protected printing option always shows when you print a document within a Windows Store application in Windows	4.15, 3.22, 2.34	Jan 13 2015
N/A	2999802	Solid lines instead of dotted lines are printed in Windows	4.15, 3.22, 2.34	Jan 13 2015
N/A	2896881	Long logon time when you use the AddPrinterConnection VBScript command to map printers for users during logon process in Windows	4.15, 3.22, 2.34	Jan 13 2015

Azure Guest OS supportability and retirement policy

11/7/2018 • 4 minutes to read • [Edit Online](#)

The information on this page relates to the Azure Guest operating system ([Guest OS](#)) for Cloud Services worker and web roles (PaaS). It does not apply to Virtual Machines (IaaS).

Microsoft has a published [support policy for the Guest OS](#). The page you are reading now describes how the policy is implemented.

The policy is

1. Microsoft will support **at least the latest two families of the Guest OS**. When a family is retired, customers have 12 months from the official retirement date to update to a newer supported Guest OS family.
2. Microsoft will support **at least the latest two versions of the supported Guest OS families**.
3. Microsoft will support **at least the latest two versions of the Azure SDK**. When a version of the SDK is retired, customers will have 12 months from the official retirement date to update to a newer version.

At times, more than two families or releases may be supported. Official Guest OS support information will appear on the [Azure Guest OS Releases and SDK Compatibility Matrix](#).

When a Guest OS version is retired

New Guest OS **versions** are introduced about every month to incorporate the latest MSRC updates. Because of the regular monthly updates, a Guest OS version is normally disabled around 60 days after its release. This activity keeps at least two Guest OS versions for each family available for use.

Process during a Guest OS family retirement

Once the retirement is announced, customers have a 12 month "transition" period before the older family is officially removed from service. This transition time may be extended at the discretion of Microsoft. Updates will be posted on the [Azure Guest OS Releases and SDK Compatibility Matrix](#).

A gradual retirement process will begin six (6) months into the transition period. During this time:

1. Microsoft will notify customers of the retirement.
2. The newer version of the Azure SDK won't support the retired Guest OS family.
3. New deployments and redeployments of Cloud Services will not be allowed on the retired family

Microsoft will continue to introduce new Guest OS version incorporating the latest MSRC updates until the last day of the transition period, known as the "expiration date". On the expiration date, Cloud Services still running will be unsupported under the Azure SLA. Microsoft has the discretion to force upgrade, delete or stop those services after that date.

Process during a Guest OS Version retirement

If customers set their Guest OS to automatically update, they never have to worry about dealing with Guest OS versions. They will always be using the latest Guest OS version.

Guest OS Versions are released every month. Because of the rate of regular releases, each version has a fixed lifespan.

At 60 days into the lifespan, a version is "*disabled*". "Disabled" means that the version is removed from the portal. The version can no longer be set from the CSCFG configuration file. Existing deployments are left running. But new deployments and code and configuration updates to existing deployments will not be allowed.

Some time after becoming "disabled", the Guest OS version "expires" and any installations still running that version are force upgraded and set to automatically update the Guest OS in the future. Expiration is done in batches so the period of time from disablement to expiration can vary.

These periods may be made longer at Microsoft's discretion to ease customer transitions. Any changes will be communicated on the [Azure Guest OS Releases and SDK Compatibility Matrix](#).

Notifications during retirement

- **Family retirement**

Microsoft will use blog posts and portal notification. Customers who are still using a retired Guest OS family will be notified through direct communication (email, portal messages, phone call) to assigned service administrators. All changes will be posted to the [Azure Guest OS Releases and SDK Compatibility Matrix](#).

- **Version Retirement**

All changes and the dates they occur will be posted to the [Azure Guest OS Releases and SDK Compatibility Matrix](#), including release, disabled, and expiration. Services admins will receive emails if they have deployments running on a disabled Guest OS version or family. The timing of these emails can vary. Generally they are at least a month before disablement, though this timing is not an official SLA.

Frequently asked questions

How can I mitigate the impacts of migration?

We recommend that you use latest Guest OS family for designing your Cloud Services.

1. Start planning your migration to a newer family early.
2. Set up temporary test deployments to test your Cloud Service running on the new family.
3. Set your Guest OS version to **Automatic** (osVersion=* in the `.cscfg` file) so the migration to new Guest OS versions occurs automatically.

What if my web application requires deeper integration with the OS?

If your web application architecture depends on underlying features of the operating system, use platform supported capabilities such as [startup tasks](#) or other extensibility mechanisms. Alternatively, you can also use [Azure Virtual Machines](#) (IaaS – Infrastructure as a Service), where you are responsible for maintaining the underlying operating system.

Next steps

Review the latest [Guest OS releases](#).

Guest OS Family 1 retirement notice

11/7/2018 • 2 minutes to read • [Edit Online](#)

The retirement of OS Family 1 was first announced on June 1, 2013.

Sept 2, 2014 The Azure Guest operating system (Guest OS) Family 1.x, which is based on the Windows Server 2008 operating system, was officially retired. All attempts to deploy new services or upgrade existing services using Family 1 will fail with an error message informing you that the Guest OS Family 1 has been retired.

November 3, 2014 Extended support for Guest OS Family 1 ended and it is fully retired. All services still on Family 1 will be impacted. We may stop those services at any time. There is no guarantee your services will continue to run unless you manually upgrade them yourself.

If you have additional questions, visit the [Cloud Services Forums](#) or [contact Azure support](#).

Are you affected?

Your Cloud Services are affected if any one of the following applies:

1. You have a value of "osFamily = "1" explicitly specified in the ServiceConfiguration.cscfg file for your Cloud Service.
2. You do not have a value for osFamily explicitly specified in the ServiceConfiguration.cscfg file for your Cloud Service. Currently, the system uses the default value of "1" in this case.
3. The Azure portal lists your Guest Operating System family value as "Windows Server 2008".

To find which of your cloud services are running which OS Family, you can run the following script in Azure PowerShell, though you must [set up Azure PowerShell](#) first. For more information on the script, see [Azure Guest OS Family 1 End of Life: June 2014](#).

```
foreach($subscription in Get-AzureSubscription) {  
    Select-AzureSubscription -SubscriptionName $subscription.SubscriptionName  
  
    $deployments=get-azureService | get-azureDeployment -ErrorAction Ignore | where {$_.SdkVersion -NE ""}  
  
    $deployments | ft @{Name="SubscriptionName";Expression={$subscription.SubscriptionName}}, ServiceName,  
    SdkVersion, Slot, @{Name="osFamily";Expression={(select-xml -content $_.configuration -xpath  
    "/ns:ServiceConfiguration/@osFamily" -namespace $namespace).node.value }}, osVersion, Status, URL  
}
```

Your cloud services will be impacted by OS Family 1 retirement if the osFamily column in the script output is empty or contains a "1".

Recommendations if you are affected

We recommend you migrate your Cloud Service roles to one of the supported Guest OS Families:

Guest OS family 4.x - Windows Server 2012 R2 (*recommended*)

1. Ensure that your application is using SDK 2.1 or later with .NET framework 4.0, 4.5 or 4.5.1.
2. Set the osFamily attribute to "4" in the ServiceConfiguration.cscfg file, and redeploy your cloud service.

Guest OS family 3.x - Windows Server 2012

1. Ensure that your application is using SDK 1.8 or later with .NET framework 4.0 or 4.5.

2. Set the osFamily attribute to "3" in the ServiceConfiguration.cscfg file, and redeploy your cloud service.

Guest OS family 2.x - Windows Server 2008 R2

1. Ensure that your application is using SDK 1.3 and above with .NET framework 3.5 or 4.0.
2. Set the osFamily attribute to "2" in the ServiceConfiguration.cscfg file, and redeploy your cloud service.

Extended Support for Guest OS Family 1 ended Nov 3, 2014

Cloud services on Guest OS family 1 are no longer supported. Migrate off family 1 as soon as possible to avoid service disruption.

Next steps

Review the latest [Guest OS releases](#).

Azure Guest OS releases and SDK compatibility matrix

1/8/2019 • 6 minutes to read • [Edit Online](#)

Provides you with up-to-date information about the latest Azure Guest OS releases for Cloud Services. This information helps you plan your upgrade path before a Guest OS is disabled. If you configure your roles to use *automatic* Guest OS updates as described in [Azure Guest OS Update Settings](#), it is not vital that you read this page.

IMPORTANT

This page applies to Cloud Services web and worker roles, which run on top of a Guest OS. It does **not apply** to IaaS Virtual Machines.

TIP

Subscribe to the [Guest OS Update RSS Feed](#) to receive the most timely notification on all Guest OS changes.

IMPORTANT

Only the latest 2 versions of the Guest OS will be supported and available in the Azure portal.

Unsure about how to update your Guest OS? Check [this](#) out.

News updates

January 7, 2019

The December Guest OS has released.

December 14, 2018

The November Guest OS has released.

November 8, 2018

The October Guest OS has released.

October 12, 2018

The September Guest OS has released.

September 12, 2018

The August Guest OS has released.

August 3, 2018

The July Guest OS has released.

July 3, 2018

The June Guest OS has released.

Releases

Family 5 releases

Windows Server 2016

.NET Framework installed: 3.5, 4.6.2

NOTE

The RDP password for OS family 5 must be a minimum of 10 characters.

CONFIGURATION STRING	RELEASE DATE	DISABLE DATE
WA-GUEST-OS-5.26_201812-01	January 7, 2019	Post 5.28
WA-GUEST-OS-5.25_201811-01	December 14, 2018	Post 5.27
WA-GUEST-OS-5.24_201810-01	November 8, 2018	January 7, 2019
WA-GUEST-OS-5.23_201809-01	October 12, 2018	December 14, 2018
WA-GUEST-OS-5.22_201808-01	September 12, 2018	November 8, 2018
WA-GUEST-OS-5.21_201807-02	August 3, 2018	October 12, 2018
WA-GUEST-OS-5.20_201806-01	July 3, 2018	September 12, 2018

Family 4 releases

Windows Server 2012 R2

.NET Framework installed: 3.5, 4.5.1

CONFIGURATION STRING	RELEASE DATE	DISABLE DATE
WA-GUEST-OS-4.61_201812-01	January 7, 2019	Post 4.63
WA-GUEST-OS-4.60_201811-01	December 14, 2018	Post 4.62
WA-GUEST-OS-4.59_201810-01	November 8, 2018	January 7, 2019
WA-GUEST-OS-4.58_201809-01	October 12, 2018	December 14, 2018
WA-GUEST-OS-4.57_201808-01	September 12, 2018	November 8, 2018
WA-GUEST-OS-4.56_201807-02	August 3, 2018	October 12, 2018
WA-GUEST-OS-4.55_201806-01	July 3, 2018	September 12, 2018

Family 3 releases

Windows Server 2012

.NET Framework installed: 3.5, 4.5

CONFIGURATION STRING	RELEASE DATE	DISABLE DATE
WA-GUEST-OS-3.68_201812-01	January 7, 2019	Post 3.70
WA-GUEST-OS-3.67_201811-01	December 14, 2018	Post 3.69
WA-GUEST-OS-3.66_201810-01	November 8, 2018	January 7, 2019
WA-GUEST-OS-3.65_201809-01	October 12, 2018	December 14, 2018
WA-GUEST-OS-3.64_201808-01	September 12, 2018	November 8, 2018
WA-GUEST-OS-3.63_201807-02	August 3, 2018	October 12, 2018
WA-GUEST-OS-3.62_201806-01	July 3, 2018	September 12, 2018

Family 2 releases

Windows Server 2008 R2 SP1

.NET Framework installed: 3.5 (includes 2.0 and 3.0)

CONFIGURATION STRING	RELEASE DATE	DISABLE DATE
WA-GUEST-OS-2.81_201812-01	January 7, 2019	Post 2.83
WA-GUEST-OS-2.80_201811-01	December 14, 2018	Post 2.82
WA-GUEST-OS-2.79_201810-01	November 8, 2018	January 7, 2019
WA-GUEST-OS-2.78_201809-01	October 12, 2018	December 14, 2018
WA-GUEST-OS-2.77_201808-01	September 12, 2018	November 8, 2018
WA-GUEST-OS-2.76_201807-02	August 3, 2018	October 12, 2018
WA-GUEST-OS-2.75_201806-01	July 3, 2018	September 12, 2018

MSRC patch updates

The list of patches that are included with each monthly Guest OS release is available [here](#).

SDK support

Even though the [retirement policy for the Azure SDK](#) indicates that only versions above 2.2 are supported, specific Guest OS families allow you to use earlier versions. You should always use the latest supported SDK.

GUEST OS FAMILY	COMPATIBLE SDK VERSIONS
5	Version 2.9.5.1+
4	Version 2.1+

GUEST OS FAMILY	COMPATIBLE SDK VERSIONS
3	Version 1.8+
2	Version 1.3+
1	Version 1.0+

Guest OS release information

There are three dates that are important to Guest OS releases: **release** date, **disabled** date, and **expiration** date. A Guest OS is considered available when it is in the Portal and can be selected as the target Guest OS. When a Guest OS reaches the **disabled** date, it is removed from Azure. However, any Cloud Service targeting that Guest OS will still operate as normal.

The window between the **disabled** date and the **expiration** date provides you with a buffer to easily transition from one Guest OS to one newer. If you're using *automatic* as your Guest OS, you'll always be on the latest version and you don't have to worry about it expiring.

When the **expiration** date passes, any Cloud Service still using that Guest OS will be stopped, deleted, or forced to upgrade. You can read more about the retirement policy [here](#).

Guest OS family-version explanation

The Guest OS families are based on released versions of Microsoft Windows Server. The Guest OS is the underlying operating system that Azure Cloud Services runs on. Each Guest OS has a family, version, and release number.

- **Guest OS family**

A Windows Server operating system release that a Guest OS is based on. For example, *family 3* is based on Windows Server 2012.

- **Guest OS version**

Specific to a Guest OS family image plus relevant [Microsoft Security Response Center \(MSRC\)](#) patches that are available at the date the new Guest OS version is produced. Not all patches may be included.

Numbers start at 0 and increment by 1 each time a new set of updates is added. Trailing zeros are only shown if important. That is, version 2.10 is a different, much later version than version 2.1.

- **Guest OS release**

A rerelease of a Guest OS version. A rerelease occurs if Microsoft finds issues during testing; requiring changes. The latest release always supersedes any previous releases, public or not. The Azure portal will only allow users to pick the latest release for a given version. Deployments running on a previous release are usually not force upgraded depending on the severity of the bug.

In the example below, 2 is the family, 12 is the version and "rel2" is the release.

Guest OS release - 2.12 rel2

Configuration string for this release - WA-GUEST-OS-2.12_201208-02

The configuration string for a Guest OS has this same information embedded in it, along with a date showing which MSRC patches were considered for that release. In this example, MSRC patches produced for Windows Server 2008 R2 up to and including August 2012 were considered for inclusion. Only patches specifically applying to that version of Windows Server are included. For example, if an MSRC patch applies to Microsoft Office, it will not be included because that product is not part of the Windows Server base image.

Guest OS system update process

This page includes information on upcoming Guest OS Releases. Customers have indicated that they want to know when a release occurs because their cloud service roles will reboot if they are set to "Automatic" update. Guest OS releases typically occur 2-3 weeks after the MSRC update release that occurs on the second Tuesday of every month. New releases include all the relevant MSRC patches for each Guest OS family.

Microsoft Azure is constantly releasing updates. The Guest OS is only one such update in the pipeline. A release can be affected by many factors too numerous to list here. In addition, Azure runs on literally hundreds of thousands of machines. This means that it's impossible to give an exact date and time when your role(s) will reboot. We are working on a plan to limit or time reboots.

When a new release of the Guest OS is published, it can take time to fully propagate across Azure. As services are updated to the new Guest OS, they are rebooted honoring update domains. Services set to use "Automatic" updates will get a release first. After the update, you'll see the new Guest OS version listed for your service in the Azure portal. Rereleases may occur during this period. Some versions may be deployed over longer periods of time and automatic upgrade reboots may not occur for many weeks after the official release date. Once a Guest OS is available, you can then explicitly choose that version from the portal or in your configuration file.

For a great deal of valuable information on restarts and pointers to more information technical details of Guest and Host OS updates, see the MSDN blog post titled [Role Instance Restarts Due to OS Upgrades](#).

If you manually update your Guest OS, see the [Guest OS retirement policy](#) for additional information.

Guest OS supportability and retirement policy

The Guest OS supportability and retirement policy is explained [here](#).

Expose role configuration settings as an environment variable with XPath

7/12/2018 • 2 minutes to read • [Edit Online](#)

In the cloud service worker or web role service definition file, you can expose runtime configuration values as environment variables. The following XPath values are supported (which correspond to API values).

These XPath values are also available through the [Microsoft.WindowsAzure.ServiceRuntime](#) library.

App running in emulator

Indicates that the app is running in the emulator.

Type	Example
XPath	<code>xpath="/RoleEnvironment/Deployment/@emulated"</code>
Code	<code>var x = RoleEnvironment.IsEmulated;</code>

Deployment ID

Retrieves the deployment ID for the instance.

Type	Example
XPath	<code>xpath="/RoleEnvironment/Deployment/@id"</code>
Code	<code>var deploymentId = RoleEnvironment.DeploymentId;</code>

Role ID

Retrieves the current role ID for the instance.

Type	Example
XPath	<code>xpath="/RoleEnvironment/CurrentInstance/@id"</code>
Code	<code>var id = RoleEnvironment.CurrentRoleInstance.Id;</code>

Update domain

Retrieves the update domain of the instance.

Type	Example
XPath	<code>xpath="/RoleEnvironment/CurrentInstance/@updateDomain"</code>

TYPE	EXAMPLE
Code	<pre>var ud = RoleEnvironment.CurrentRoleInstance.UpdateDomain;</pre>

Fault domain

Retrieves the fault domain of the instance.

TYPE	EXAMPLE
XPath	<code>xpath="/RoleEnvironment/CurrentInstance/@faultDomain"</code>
Code	<pre>var fd = RoleEnvironment.CurrentRoleInstance.FaultDomain;</pre>

Role name

Retrieves the role name of the instances.

TYPE	EXAMPLE
XPath	<code>xpath="/RoleEnvironment/CurrentInstance/@roleName"</code>
Code	<pre>var rname = RoleEnvironment.CurrentRoleInstance.Role.Name;</pre>

Config setting

Retrieves the value of the specified configuration setting.

TYPE	EXAMPLE
XPath	<code>xpath="/RoleEnvironment/CurrentInstance/ConfigurationSettings/ConfigurationSetting[@name='Setting1']/@value"</code>
Code	<pre>var setting = RoleEnvironment.GetConfigurationSettingValue("Setting1");</pre>

Local storage path

Retrieves the local storage path for the instance.

TYPE	EXAMPLE
XPath	<code>xpath="/RoleEnvironment/CurrentInstance/LocalResources/LocalResource[@name='LocalStore1']/@path"</code>
Code	<pre>var localResourcePath = RoleEnvironment.GetLocalResource("LocalStore1").RootPath;</pre>

Local storage size

Retrieves the size of the local storage for the instance.

TYPE	EXAMPLE
XPath	xpath="/RoleEnvironment/CurrentInstance/LocalResources/LocalResource[@name='LocalStore1']/@sizeInMB"
Code	var localResourceSizeInMB = RoleEnvironment.GetLocalResource("LocalStore1").MaximumSizeInMegabytes;

Endpoint protocol

Retrieves the endpoint protocol for the instance.

TYPE	EXAMPLE
XPath	xpath="/RoleEnvironment/CurrentInstance/Endpoints/Endpoint[@name='Endpoint1']/@protocol"
Code	var prot = RoleEnvironment.CurrentRoleInstance.InstanceEndpoints["Endpoint1"].Protocol;

Endpoint IP

Gets the specified endpoint's IP address.

TYPE	EXAMPLE
XPath	xpath="/RoleEnvironment/CurrentInstance/Endpoints/Endpoint[@name='Endpoint1']/@address"
Code	var address = RoleEnvironment.CurrentRoleInstance.InstanceEndpoints["Endpoint1"].IPEndpoint.Address

Endpoint port

Retrieves the endpoint port for the instance.

TYPE	EXAMPLE
XPath	xpath="/RoleEnvironment/CurrentInstance/Endpoints/Endpoint[@name='Endpoint1']/@port"
Code	var port = RoleEnvironment.CurrentRoleInstance.InstanceEndpoints["Endpoint1"].IPEndpoint.Port;

Example

Here is an example of a worker role that creates a startup task with an environment variable named `TestIsEmulated` set to the [@emulated xpath value](#).

```
<WorkerRole name="Role1">
  <ConfigurationSettings>
    <Setting name="Setting1" />
  </ConfigurationSettings>
  <LocalResources>
    <LocalStorage name="LocalStore1" sizeInMB="1024"/>
  </LocalResources>
  <Endpoints>
    <InternalEndpoint name="Endpoint1" protocol="tcp" />
  </Endpoints>
  <Startup>
    <Task commandLine="example.cmd inputParm">
      <Environment>
        <Variable name="TestConstant" value="Constant"/>
        <Variable name="TestEmptyValue" value="" />
        <Variable name="TestIsEmulated">
          <RoleInstanceValue xpath="/RoleEnvironment/Deployment/@emulated"/>
        </Variable>
        ...
      </Environment>
    </Task>
  </Startup>
  <Runtime>
    <Environment>
      <Variable name="TestConstant" value="Constant"/>
      <Variable name="TestEmptyValue" value="" />
      <Variable name="TestIsEmulated">
        <RoleInstanceValue xpath="/RoleEnvironment/Deployment/@emulated"/>
      </Variable>
      ...
    </Environment>
  </Runtime>
  ...
</WorkerRole>
```

Next steps

Learn more about the [ServiceConfiguration.cscfg](#) file.

Create a [ServicePackage.cspkg](#) package.

Enable [remote desktop](#) for a role.

Certificates overview for Azure Cloud Services

11/7/2018 • 3 minutes to read • [Edit Online](#)

Certificates are used in Azure for cloud services ([service certificates](#)) and for authenticating with the management API ([management certificates](#)). This topic gives a general overview of both certificate types, how to [create](#) and [deploy](#) them to Azure.

Certificates used in Azure are x.509 v3 certificates and can be signed by another trusted certificate or they can be self-signed. A self-signed certificate is signed by its own creator, therefore it is not trusted by default. Most browsers can ignore this problem. You should only use self-signed certificates when developing and testing your cloud services.

Certificates used by Azure can contain a private or a public key. Certificates have a thumbprint that provides a means to identify them in an unambiguous way. This thumbprint is used in the Azure [configuration file](#) to identify which certificate a cloud service should use.

NOTE

Azure Cloud Services does not accept AES256-SHA256 encrypted certificate.

What are service certificates?

Service certificates are attached to cloud services and enable secure communication to and from the service. For example, if you deployed a web role, you would want to supply a certificate that can authenticate an exposed HTTPS endpoint. Service certificates, defined in your service definition, are automatically deployed to the virtual machine that is running an instance of your role.

You can upload service certificates to Azure either using the Azure portal or by using the classic deployment model. Service certificates are associated with a specific cloud service. They are assigned to a deployment in the service definition file.

Service certificates can be managed separately from your services, and may be managed by different individuals. For example, a developer may upload a service package that refers to a certificate that an IT manager has previously uploaded to Azure. An IT manager can manage and renew that certificate (changing the configuration of the service) without needing to upload a new service package. Updating without a new service package is possible because the logical name, store name, and location of the certificate is in the service definition file and while the certificate thumbprint is specified in the service configuration file. To update the certificate, it's only necessary to upload a new certificate and change the thumbprint value in the service configuration file.

NOTE

The [Cloud Services FAQ](#) article has some helpful information about certificates.

What are management certificates?

Management certificates allow you to authenticate with the classic deployment model. Many programs and tools (such as Visual Studio or the Azure SDK) use these certificates to automate configuration and deployment of various Azure services. These are not really related to cloud services.

WARNING

Be careful! These types of certificates allow anyone who authenticates with them to manage the subscription they are associated with.

Limitations

There is a limit of 100 management certificates per subscription. There is also a limit of 100 management certificates for all subscriptions under a specific service administrator's user ID. If the user ID for the account administrator has already been used to add 100 management certificates and there is a need for more certificates, you can add a co-administrator to add the additional certificates.

Create a new self-signed certificate

You can use any tool available to create a self-signed certificate as long as they adhere to these settings:

- An X.509 certificate.
- Contains a private key.
- Created for key exchange (.pfx file).
- Subject name must match the domain used to access the cloud service.

You cannot acquire an SSL certificate for the clouddapp.net (or for any Azure-related) domain; the certificate's subject name must match the custom domain name used to access your application. For example, **contoso.net**, not **contoso.clouddapp.net**.

- Minimum of 2048-bit encryption.
- **Service Certificate Only:** Client-side certificate must reside in the *Personal* certificate store.

There are two easy ways to create a certificate on Windows, with the `makecert.exe` utility, or IIS.

Makecert.exe

This utility has been deprecated and is no longer documented here. For more information, see [this MSDN article](#).

PowerShell

```
$cert = New-SelfSignedCertificate -DnsName yourdomain.clouddapp.net -CertStoreLocation "cert:\LocalMachine\My"  
-KeyLength 2048 -KeySpec "KeyExchange"  
$password = ConvertTo-SecureString -String "your-password" -Force -AsPlainText  
Export-PfxCertificate -Cert $cert -FilePath ".\my-cert-file.pfx" -Password $password
```

NOTE

If you want to use the certificate with an IP address instead of a domain, use the IP address in the -DnsName parameter.

If you want to use this [certificate with the management portal](#), export it to a **.cer** file:

```
Export-Certificate -Type CERT -Cert $cert -FilePath .\my-cert-file.cer
```

Internet Information Services (IIS)

There are many pages on the internet that cover how to do this with IIS. [Here](#) is a great one I found that I think explains it well.

Linux

[This](#) article describes how to create certificates with SSH.

Next steps

[Upload your service certificate to the Azure portal](#).

Upload a [management API certificate](#) to the Azure portal.

Configuring SSL for an application in Azure

11/8/2018 • 5 minutes to read • [Edit Online](#)

Secure Socket Layer (SSL) encryption is the most commonly used method of securing data sent across the internet. This common task discusses how to specify an HTTPS endpoint for a web role and how to upload an SSL certificate to secure your application.

NOTE

The procedures in this task apply to Azure Cloud Services; for App Services, see [this](#).

This task uses a production deployment. Information on using a staging deployment is provided at the end of this topic.

Read [this](#) first if you have not yet created a cloud service.

Step 1: Get an SSL certificate

To configure SSL for an application, you first need to get an SSL certificate that has been signed by a Certificate Authority (CA), a trusted third party who issues certificates for this purpose. If you do not already have one, you need to obtain one from a company that sells SSL certificates.

The certificate must meet the following requirements for SSL certificates in Azure:

- The certificate must contain a private key.
- The certificate must be created for key exchange, exportable to a Personal Information Exchange (.pfx) file.
- The certificate's subject name must match the domain used to access the cloud service. You cannot obtain an SSL certificate from a certificate authority (CA) for the clouddapp.net domain. You must acquire a custom domain name to use when access your service. When you request a certificate from a CA, the certificate's subject name must match the custom domain name used to access your application. For example, if your custom domain name is **contoso.com** you would request a certificate from your CA for **.contoso.com*** or ****www.contoso.com**.
- The certificate must use a minimum of 2048-bit encryption.

For test purposes, you can [create](#) and use a self-signed certificate. A self-signed certificate is not authenticated through a CA and can use the clouddapp.net domain as the website URL. For example, the following task uses a self-signed certificate in which the common name (CN) used in the certificate is **sslexample.clouddapp.net**.

Next, you must include information about the certificate in your service definition and service configuration files.

Step 2: Modify the service definition and configuration files

Your application must be configured to use the certificate, and an HTTPS endpoint must be added. As a result, the service definition and service configuration files need to be updated.

1. In your development environment, open the service definition file (CSDEF), add a **Certificates** section within the **WebRole** section, and include the following information about the certificate (and intermediate certificates):

```

<WebRole name="CertificateTesting" vmsize="Small">
...
    <Certificates>
        <Certificate name="SampleCertificate"
            storeLocation="LocalMachine"
            storeName="My"
            permissionLevel="limitedOrElevated" />
        <!-- IMPORTANT! Unless your certificate is either
            self-signed or signed directly by the CA root, you
            must include all the intermediate certificates
            here. You must list them here, even if they are
            not bound to any endpoints. Failing to list any of
            the intermediate certificates may cause hard-to-reproduce
            interoperability problems on some clients.-->
        <Certificate name="CAForSampleCertificate"
            storeLocation="LocalMachine"
            storeName="CA"
            permissionLevel="limitedOrElevated" />
    </Certificates>
...
</WebRole>

```

The **Certificates** section defines the name of our certificate, its location, and the name of the store where it is located.

Permissions (`permissionLevel` attribute) can be set to one of the following values:

PERMISSION VALUE	DESCRIPTION
limitedOrElevated	(Default) All role processes can access the private key.
elevated	Only elevated processes can access the private key.

2. In your service definition file, add an **InputEndpoint** element within the **Endpoints** section to enable HTTPS:

```

<WebRole name="CertificateTesting" vmsize="Small">
...
    <Endpoints>
        <InputEndpoint name="HttpsIn" protocol="https" port="443"
            certificate="SampleCertificate" />
    </Endpoints>
...
</WebRole>

```

3. In your service definition file, add a **Binding** element within the **Sites** section. This element adds an HTTPS binding to map the endpoint to your site:

```

<WebRole name="CertificateTesting" vmsize="Small">
...
    <Sites>
        <Site name="Web">
            <Bindings>
                <Binding name="HttpsIn" endpointName="HttpsIn" />
            </Bindings>
        </Site>
    </Sites>
...
</WebRole>

```

All the required changes to the service definition file have been completed; but, you still need to add the certificate information to the service configuration file.

4. In your service configuration file (CSCFG), ServiceConfiguration.Cloud.cscfg, add a **Certificates** value with that of your certificate. The following code sample provides details of the **Certificates** section, except for the thumbprint value.

```
<Role name="Deployment">
...
<Certificates>
    <Certificate name="SampleCertificate"
        thumbprint="9427befa18ec6865a9ebdc79d4c38de50e6316ff"
        thumbprintAlgorithm="sha1" />
    <Certificate name="CAForSampleCertificate"
        thumbprint="79d4c38de50e6316ff9427befa18ec6865a9ebdc"
        thumbprintAlgorithm="sha1" />
</Certificates>
...
</Role>
```

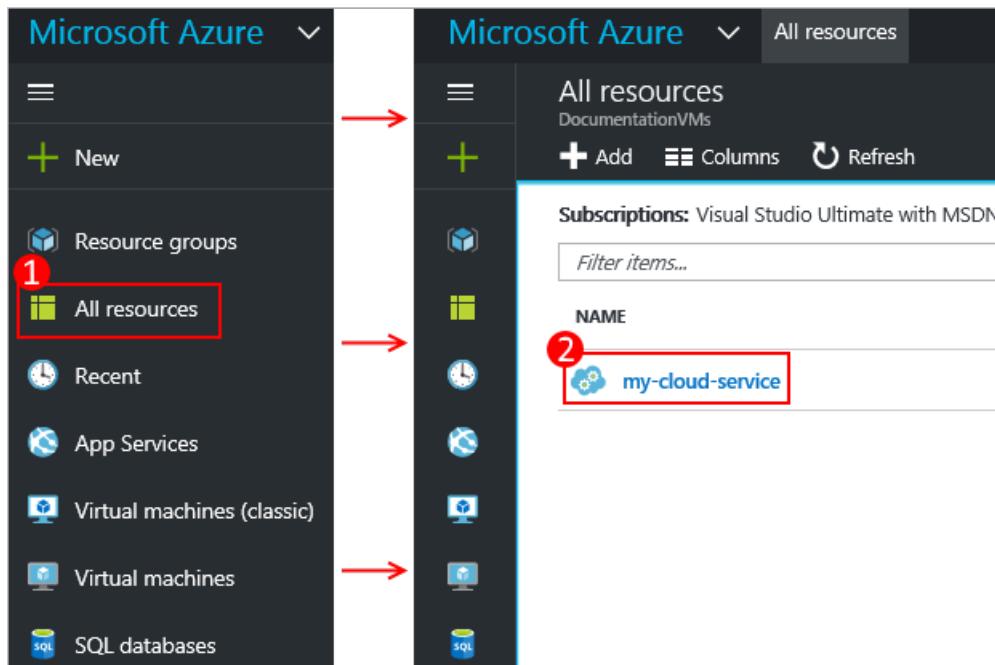
(This example uses **sha1** for the thumbprint algorithm. Specify the appropriate value for your certificate's thumbprint algorithm.)

Now that the service definition and service configuration files have been updated, package your deployment for uploading to Azure. If you are using **cspack**, don't use the **/generateConfigurationFile** flag, as that will overwrite the certificate information you just inserted.

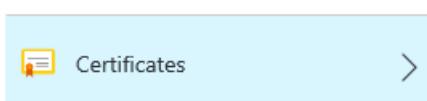
Step 3: Upload a certificate

Connect to the Azure portal and...

1. In the **All resources** section of the Portal, select your cloud service.



2. Click **Certificates**.



3. Click **Upload** at the top of the certificates area.



4. Provide the **File, Password**, then click **Upload** at the bottom of the data entry area.

Step 4: Connect to the role instance by using HTTPS

Now that your deployment is up and running in Azure, you can connect to it using HTTPS.

1. Click the **Site URL** to open up the web browser.

Essentials ^	
Resource group cspythonstorage	Site URL http://e45b502f---url---7f90bf5df.cloudapp.net
Status Running	Public IP addresses 13.91.252.255
Location West US	Deployment name 492240f1---guid---1303a79
Subscription name Visual Studio Ultimate with MSDN	Deployment label AzureCloudService3 - 10/4/2016 3:26:51 PM
Subscription ID c9810---guid---31743d5d	Deployment ID e45b502f6---guid---f90bf5df

2. In your web browser, modify the link to use **https** instead of **http**, and then visit the page.

NOTE

If you are using a self-signed certificate, when you browse to an HTTPS endpoint that's associated with the self-signed certificate you may see a certificate error in the browser. Using a certificate signed by a trusted certification authority eliminates this problem; in the meantime, you can ignore the error. (Another option is to add the self-signed certificate to the user's trusted certificate authority certificate store.)

The screenshot shows a web browser window with the URL <http://sslexample1.cloudapp.net/> in the address bar. The page content includes the title "ASP.NET", a brief description of the framework, and a blue "Learn more »" button. Below this section is a "Getting started" heading with a similar description and a "Learn more »" button.

TIP

If you want to use SSL for a staging deployment instead of a production deployment, you'll first need to determine the URL used for the staging deployment. Once your cloud service has been deployed, the URL to the staging environment is determined by the **Deployment ID** GUID in this format: <https://deployment-id.cloudapp.net/>

Create a certificate with the common name (CN) equal to the GUID-based URL (for example, **328187776e774ceda8fc57609d404462.cloudapp.net**). Use the portal to add the certificate to your staged cloud service. Then, add the certificate information to your CSDEF and CSCFG files, repackage your application, and update your staged deployment to use the new package.

Next steps

- General configuration of your cloud service.
- Learn how to deploy a cloud service.
- Configure a custom domain name.
- Manage your cloud service.

Introduction to Cloud Service Monitoring

12/27/2018 • 3 minutes to read • [Edit Online](#)

You can monitor key performance metrics for any cloud service. Every cloud service role collects minimal data: CPU usage, network usage, and disk utilization. If the cloud service has the `Microsoft.Azure.Diagnostics` extension applied to a role, that role can collect additional points of data. This article provides an introduction to Azure Diagnostics for Cloud Services.

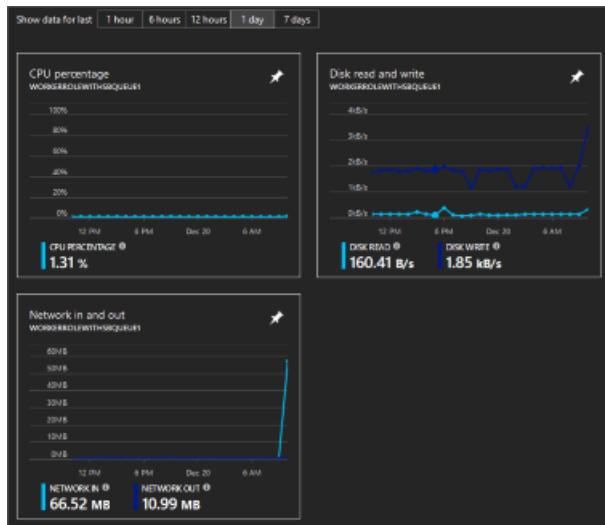
With basic monitoring, performance counter data from role instances is sampled and collected at 3-minute intervals. This basic monitoring data is not stored in your storage account and has no additional cost associated with it.

With advanced monitoring, additional metrics are sampled and collected at intervals of 5 minutes, 1 hour, and 12 hours. The aggregated data is stored in a storage account, in tables, and is purged after 10 days. The storage account used is configured by role; you can use different storage accounts for different roles. This is configured with a connection string in the `.csdef` and `.cscfg` files.

Basic monitoring

As stated in the introduction, a cloud service automatically collects basic monitoring data from the host virtual machine. This data includes CPU percentage, network in/out, and disk read/write. The collected monitoring data is automatically displayed on the overview and metrics pages of the cloud service, in the Azure portal.

Basic monitoring does not require a storage account.



Advanced monitoring

Advanced monitoring involves using the **Azure Diagnostics** extension (and optionally the Application Insights SDK) on the role you want to monitor. The diagnostics extension uses a config file (per role) named `diagnostics.wadcfgx` to configure the diagnostics metrics monitored. The Azure Diagnostic extension collects and stores data in an Azure Storage account. These settings are configured in the `.wadcfgx`, `.csdef`, and `.cscfg` files. This means that there is an extra cost associated with advanced monitoring.

As each role is created, Visual Studio adds the Azure Diagnostics extension to it. This diagnostics extension can collect the following types of information:

- Custom performance counters

- Application logs
- Windows event logs
- .NET event source
- IIS logs
- Manifest based ETW
- Crash dumps
- Customer error logs

IMPORTANT

While all this data is aggregated into the storage account, the portal does **not** provide a native way to chart the data. It is highly recommended that you integrate another service, like Application Insights, into your application.

Setup diagnostics extension

First, if you don't have a **classic** storage account, [create one](#). Make sure the storage account is created with the **Classic deployment model** specified.

Next, navigate to the **Storage account (classic)** resource. Select **Settings > Access keys** and copy the **Primary connection string** value. You need this value for the cloud service.

There are two config files you must change for advanced diagnostics to be enabled, **ServiceDefinition.csdef** and **ServiceConfiguration.cscfg**.

ServiceDefinition.csdef

In the **ServiceDefinition.csdef** file, add a new setting named

`Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString` for each role that uses advanced diagnostics. Visual Studio adds this value to the file when you create a new project. In case it is missing, you can add it now.

```
<ServiceDefinition name="AnsurCloudService"
  xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition" schemaVersion="2015-04.2.6">
  <WorkerRole name="WorkerRoleWithSBQueue1" vmsize="Small">
    <ConfigurationSettings>
      <Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString" />
```

This defines a new setting that must be added to every **ServiceConfiguration.cscfg** file.

Most likely you have two **.cscfg** files, one named **ServiceConfiguration.cloud.cscfg** for deploying to Azure, and one named **ServiceConfiguration.local.cscfg** that is used for local deployments in the emulated environment.

Open and change each **.cscfg** file. Add a setting named

`Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString`. Set the value to the **Primary connection string** of the classic storage account. If you want to use the local storage on your development machine, use `UseDevelopmentStorage=true`.

```
<ServiceConfiguration serviceName="AnsurCloudService"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfiguration" osFamily="4" osVersion="*"
schemaVersion="2015-04.2.6">
  <Role name="WorkerRoleWithSBQueue1">
    <Instances count="1" />
    <ConfigurationSettings>
      <Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"
value="DefaultEndpointsProtocol=https;AccountName=mystorage;AccountKey=KwWkdFmskOIS240jnB0eeXVGHT9QgKS4kIQ3wWVKzOYkfjdsjfkjdsaf+sddfwwfw+sdffsdafda/w==" />

      <!-- or use the local development machine for storage
      <Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"
value="UseDevelopmentStorage=true" />
    -->
  </Role>
</ServiceConfiguration>
```

Use Application Insights

When you publish the Cloud Service from Visual Studio, you are given the option to send the diagnostic data to Application Insights. You can create the Application Insights Azure resource at that time or send the data to an existing Azure resource. Your cloud service can be monitored by Application Insights for availability, performance, failures, and usage. Custom charts can be added to Application Insights so that you can see the data that matters the most. Role instance data can be collected by using the Application Insights SDK in your cloud service project. For more information on how to integrate Application Insights, see [Application Insights with Cloud Services](#).

Note that while you can use Application Insights to display the performance counters (and the other settings) you have specified through the Windows Azure Diagnostics extension, you only get a richer experience by integrating the Application Insights SDK into your worker and web roles.

Next steps

- [Learn about Application Insights with Cloud Services](#)
- [Set up performance counters](#)

Collect performance counters for your Azure Cloud Service

1/3/2019 • 8 minutes to read • [Edit Online](#)

Performance counters provide a way for you to track how well your application and the host are performing. Windows Server provides many different performance counters related to hardware, applications, the operating system, and more. By collecting and sending performance counters to Azure, you can analyze this information to help make better decisions.

Discover available counters

A performance counter is made up of two parts, a set name (also known as a category) and one or more counters. You can use PowerShell to get a list of available performance counters:

```
Get-Counter -ListSet * | Select-Object CounterSetName, Paths | Sort-Object CounterSetName

CounterSetName          Paths
-----
.NET CLR Data           {\.\.NET CLR Data(*)\SqlClient...
.NET CLR Exceptions     {\.\.NET CLR Exceptions(*)\# o...
.NET CLR Interop         {\.\.NET CLR Interop(*)\# of C...
.NET CLR Jit             {\.\.NET CLR Jit(*)\# of Metho...
.NET Data Provider for Oracle {\.\.NET Data Provider for Ora...
.NET Data Provider for SqlServer {\.\.NET Data Provider for Sql...
.NET Memory Cache 4.0    {\.\.NET Memory Cache 4.0(*)\C...
AppV Client Streamed Data Percentage {\AppV Client Streamed Data ...
ASP.NET                 {\ASP.NET\Application Restart...
ASP.NET Apps v4.0.30319   {\ASP.NET Apps v4.0.30319(*)... 
ASP.NET State Service    {\ASP.NET State Service\Stat...
ASP.NET v2.0.50727        {\ASP.NET v2.0.50727\Applica...
ASP.NET v4.0.30319        {\ASP.NET v4.0.30319\Applica...
Authorization Manager Applications {\Authorization Manager Appl... 

#... results cut to save space ...
```

The `CounterSetName` property represents a set (or category), and is a good indicator of what the performance counters are related to. The `Paths` property represents a collection of counters for a set. You could also get the `Description` property for more information about the counter set.

To get all of the counters for a set, use the `CounterSetName` value and expand the `Paths` collection. Each path item is a counter you can query. For example, to get the available counters related to the `Processor` set, expand the `Paths` collection:

```

Get-Counter -ListSet * | Where-Object CounterSetName -eq "Processor" | Select -ExpandProperty Paths

\Processor(*)\% Processor Time
\Processor(*)\% User Time
\Processor(*)\% Privileged Time
\Processor(*)\Interrupts/sec
\Processor(*)\% DPC Time
\Processor(*)\% Interrupt Time
\Processor(*)\DPCs Queued/sec
\Processor(*)\DPC Rate
\Processor(*)\% Idle Time
\Processor(*)\% C1 Time
\Processor(*)\% C2 Time
\Processor(*)\% C3 Time
\Processor(*)\C1 Transitions/sec
\Processor(*)\C2 Transitions/sec
\Processor(*)\C3 Transitions/sec

```

These individual counter paths can be added to the diagnostics framework your cloud service uses. For more information about how a performance counter path is constructed, see [Specifying a Counter Path](#).

Collect a performance counter

A performance counter can be added to your cloud service for either Azure Diagnostics or Application Insights.

Application Insights

Azure Application Insights for Cloud Services allows you specify what performance counters you want to collect. After you [add Application Insights to your project](#), a config file named **ApplicationInsights.config** is added to your Visual Studio project. This config file defines what type of information Application Insights collects and sends to Azure.

Open the **ApplicationInsights.config** file and find the **ApplicationInsights > TelemetryModules** element.

Each `<Add>` child-element defines a type of telemetry to collect, along with its configuration. The performance counter telemetry module type is

```
Microsoft.ApplicationInsights.Extensibility.PerfCounterCollector.PerformanceCollectorModule,
Microsoft.AI.PerfCounterCollector
```

If this element is already defined, do not add it a second time. Each performance counter to collect is defined under a node named `<Counters>`. Here is an example that collects drive performance counters:

```

<ApplicationInsights xmlns="http://schemas.microsoft.com/ApplicationInsights/2013/Settings">

    <TelemetryModules>

        <Add Type="Microsoft.ApplicationInsights.Extensibility.PerfCounterCollector.PerformanceCollectorModule,
Microsoft.AI.PerfCounterCollector">
            <Counters>
                <Add PerformanceCounter="\LogicalDisk(C:)\Disk Write Bytes/sec" ReportAs="Disk write (C:)" />
                <Add PerformanceCounter="\LogicalDisk(C:)\Disk Read Bytes/sec" ReportAs="Disk read (C:)" />
            </Counters>
        </Add>

    </TelemetryModules>

<!-- ... cut to save space ... -->

```

Each performance counter is represented as an `<Add>` element under `<Counters>`. The `PerformanceCounter` attribute defines which performance counter to collect. The `ReportAs` attribute is the title to display in the Azure portal for the performance counter. Any performance counter you collect is put into a category named **Custom** in the portal. Unlike Azure Diagnostics, you cannot set the interval these performance counters are collected and sent

to Azure. With Application Insights, performance counters are collected and sent every minute.

Application Insights automatically collects the following performance counters:

- \Process(??APP_WIN32_PROC??)% Processor Time
- \Memory\Available Bytes
- .NET CLR Exceptions(??APP_CLR_PROC??)# of Exceps Thrown / sec
- \Process(??APP_WIN32_PROC??)\Private Bytes
- \Process(??APP_WIN32_PROC??)\IO Data Bytes/sec
- \Processor(_Total)% Processor Time

For more information, see [System performance counters in Application Insights](#) and [Application Insights for Azure Cloud Services](#).

Azure Diagnostics

IMPORTANT

While all this data is aggregated into the storage account, the portal does **not** provide a native way to chart the data. It is highly recommended that you integrate another diagnostics service, like Application Insights, into your application.

The Azure Diagnostics extension for Cloud Services allows you specify what performance counters you want to collect. To set up Azure Diagnostics, see [Cloud Service Monitoring Overview](#).

The performance counters you want to collect are defined in the **diagnostics.wadcfgx** file. Open this file (it is defined per role) in Visual Studio and find the **DiagnosticsConfiguration > PublicConfig > WadCfg >**

DiagnosticMonitorConfiguration > PerformanceCounters element. Add a new

PerformanceCounterConfiguration element as a child. This element has two attributes: `counterSpecifier` and `sampleRate`. The `counterSpecifier` attribute defines which system performance counter set (outlined in the previous section) to collect. The `sampleRate` value indicates how often that value is polled. As a whole, all performance counters are transferred to Azure according to the parent `PerformanceCounters` element's `scheduledTransferPeriod` attribute value.

For more information about the `PerformanceCounters` schema element, see the [Azure Diagnostics Schema](#).

The period defined by the `sampleRate` attribute uses the XML duration data type to indicate how often the performance counter is polled. In the example below, the rate is set to `PT3M`, which means

`[P]eriod[T]ime[3][M]inutes` : every three minutes.

For more information about how the `sampleRate` and `scheduledTransferPeriod` are defined, see the **Duration Data Type** section in the [W3 XML Date and Time Date Types](#) tutorial.

```

<?xml version="1.0" encoding="utf-8"?>
<DiagnosticsConfiguration
  xmlns="http://schemas.microsoft.com/ServiceHosting/2010/10/DiagnosticsConfiguration">
  <PublicConfig>
    <WadCfg>
      <DiagnosticMonitorConfiguration overallQuotaInMB="4096">

        <!-- ... cut to save space ... -->

        <PerformanceCounters scheduledTransferPeriod="PT1M">
          <PerformanceCounterConfiguration counterSpecifier="\Memory\Available MBytes" sampleRate="PT3M" />
          <PerformanceCounterConfiguration counterSpecifier="\Web Service(_Total)\ISAPI Extension Requests/sec" sampleRate="PT3M" />
          <PerformanceCounterConfiguration counterSpecifier="\Web Service(_Total)\Bytes Total/Sec" sampleRate="PT3M" />
          <PerformanceCounterConfiguration counterSpecifier="\ASP.NET Applications(__Total__)\Requests/Sec" sampleRate="PT3M" />
          <PerformanceCounterConfiguration counterSpecifier="\ASP.NET Applications(__Total__)\Errors Total/Sec" sampleRate="PT3M" />
          <PerformanceCounterConfiguration counterSpecifier="\ASP.NET\Requests Queued" sampleRate="PT3M" />
          <PerformanceCounterConfiguration counterSpecifier="\ASP.NET\Requests Rejected" sampleRate="PT3M" />
          <PerformanceCounterConfiguration counterSpecifier="\Processor(_Total)\% Processor Time" sampleRate="PT3M" />
        <!-- This is a new perf counter which will track the C: disk read activity in bytes per second, every minute -->
        <PerformanceCounterConfiguration counterSpecifier="\LogicalDisk(C:)\Disk Read Bytes/sec" sampleRate="PT1M" />

      </PerformanceCounters>
      </DiagnosticMonitorConfiguration>
    </WadCfg>

    <!-- ... cut to save space ... -->

  </PublicConfig>
</DiagnosticsConfiguration>

```

Create a new perf counter

A new performance counter can be created and used by your code. Your code that creates a new performance counter must be running elevated, otherwise it will fail. Your cloud service `OnStart` startup code can create the performance counter, requiring you to run the role in an elevated context. Or you can create a startup task that runs elevated and creates the performance counter. For more information about startup tasks, see [How to configure and run startup tasks for a cloud service](#).

To configure your role to run elevated, add a `<Runtime>` element to the `.csdef` file.

```

<ServiceDefinition name="CloudServiceLoadTesting"
  xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition" schemaVersion="2015-04.2.6">
  <WorkerRole name="WorkerRoleWithSBQueue1" vmsize="Large">

    <!-- ... cut to save space ... -->

    <Runtime executionContext="elevated">

    </Runtime>

    <!-- ... cut to save space ... -->

  </WorkerRole>
</ServiceDefinition>

```

You can create and register a new performance counter with a few lines of code. Use the `System.Diagnostics.PerformanceCounterCategory.Create` method overload that creates both the category and the counter. The following code first checks if the category exists, and if missing, creates both the category and the counter.

```
using System.Diagnostics;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace WorkerRoleWithSBQueue1
{
    public class WorkerRole : RoleEntryPoint
    {
        // Perf counter variable representing times service was used.
        private PerformanceCounter counterServiceUsed;

        public override bool OnStart()
        {
            // ... Other startup code here ...

            // Define the category and counter names.
            string perfCounterCatName = "MyService";
            string perfCounterName = "Times Used";

            // Create the counter if needed. Our counter category only has a single counter.
            // Both the category and counter are created in the same method call.
            if (!PerformanceCounterCategory.Exists(perfCounterCatName))
            {
                PerformanceCounterCategory.Create(perfCounterCatName, "Collects information about the cloud
service.",

                                                PerformanceCounterCategoryType.SingleInstance,
                                                perfCounterName, "How many times the cloud service was
used.");
            }

            // Get reference to our counter
            counterServiceUsed = new PerformanceCounter(perfCounterCatName, perfCounterName);
            counterServiceUsed.ReadOnly = false;

            return base.OnStart();
        }

        // ... cut class code to save space
    }
}
```

When you want to use the counter, call the `Increment` or `IncrementBy` method.

```
// Increase the counter by 1
counterServiceUsed.Increment();
```

Now that your application uses your custom counter, you need to configure Azure Diagnostics or Application Insights to track the counter.

Application Insights

As previously stated, the performance counters for Application Insights are defined in the `ApplicationInsights.config` file. Open `ApplicationInsights.config` and find the `ApplicationInsights > TelemetryModules > Add > Counters` element. Create an `<Add>` child element and set the `PerformanceCounter` attribute to the category and name of the performance counter you created in your code. Set the `ReportAs` attribute to a friendly name you want to see in the portal.

```

<ApplicationInsights xmlns="http://schemas.microsoft.com/ApplicationInsights/2013/Settings">

    <TelemetryModules>

        <Add Type="Microsoft.ApplicationInsights.Extensibility.PerfCounterCollector.PerformanceCollectorModule,
Microsoft.AI.PerfCounterCollector">
            <Counters>
                <!-- ... cut other perf counters to save space ... -->

                <!-- This new perf counter matches the [category name]\[counter name] defined in your code -->
                <Add PerformanceCounter="\MyService\Times Used" ReportAs="Service used counter" />
            </Counters>
        </Add>

    </TelemetryModules>

    <!-- ... cut to save space ... -->

```

Azure Diagnostics

As previously stated, the performance counters you want to collect are defined in the **diagnostics.wadcfgx** file. Open this file (it is defined per role) in Visual Studio and find the **DiagnosticsConfiguration > PublicConfig > WadCfg > DiagnosticMonitorConfiguration > PerformanceCounters** element. Add a new **PerformanceCounterConfiguration** element as a child. Set the `counterSpecifier` attribute to the category and name of the performance counter you created in your code.

```

<?xml version="1.0" encoding="utf-8"?>
<DiagnosticsConfiguration
xmlns="http://schemas.microsoft.com/ServiceHosting/2010/10/DiagnosticsConfiguration">
    <PublicConfig>
        <WadCfg>
            <DiagnosticMonitorConfiguration overallQuotaInMB="4096">

                <!-- ... cut to save space ... -->

                <PerformanceCounters scheduledTransferPeriod="PT1M">
                    <!-- ... cut other perf counters to save space ... -->

                    <!-- This new perf counter matches the [category name]\[counter name] defined in your code -->
                    <PerformanceCounterConfiguration counterSpecifier="\MyService\Times Used" sampleRate="PT1M" />

                </PerformanceCounters>
            </DiagnosticMonitorConfiguration>
        </WadCfg>

        <!-- ... cut to save space ... -->

    </PublicConfig>
</DiagnosticsConfiguration>

```

More information

- [Application Insights for Azure Cloud Services](#)
- [System performance counters in Application Insights](#)
- [Specifying a Counter Path](#)
- [Azure Diagnostics Schema - Performance Counters](#)

Testing the Performance of a Cloud Service Locally in the Azure Compute Emulator Using the Visual Studio Profiler

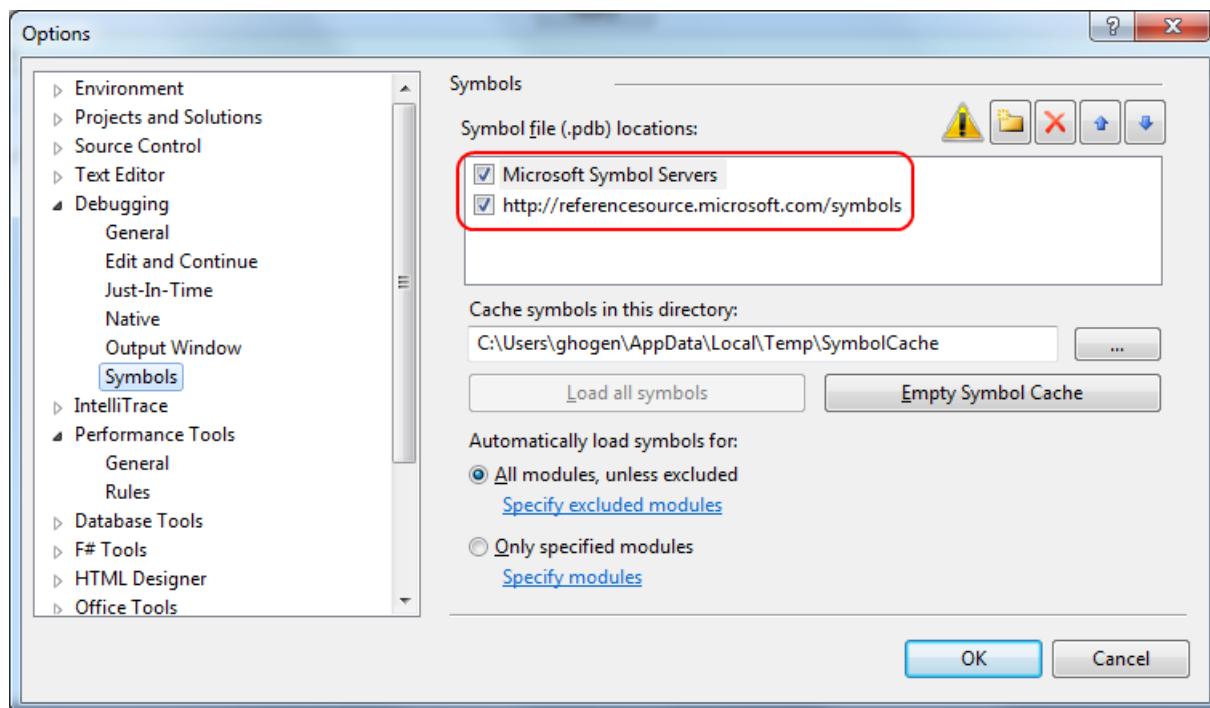
7/2/2018 • 6 minutes to read • [Edit Online](#)

A variety of tools and techniques are available for testing the performance of cloud services. When you publish a cloud service to Azure, you can have Visual Studio collect profiling data and then analyze it locally, as described in [Profiling an Azure Application](#). You can also use diagnostics to track a variety of performance counters, as described in [Using performance counters in Azure](#). You might also want to profile your application locally in the compute emulator before deploying it to the cloud.

This article covers the CPU Sampling method of profiling, which can be done locally in the emulator. CPU sampling is a method of profiling that is not very intrusive. At a designated sampling interval, the profiler takes a snapshot of the call stack. The data is collected over a period of time, and shown in a report. This method of profiling tends to indicate where in a computationally intensive application most of the CPU work is being done. This gives you the opportunity to focus on the "hot path" where your application is spending the most time.

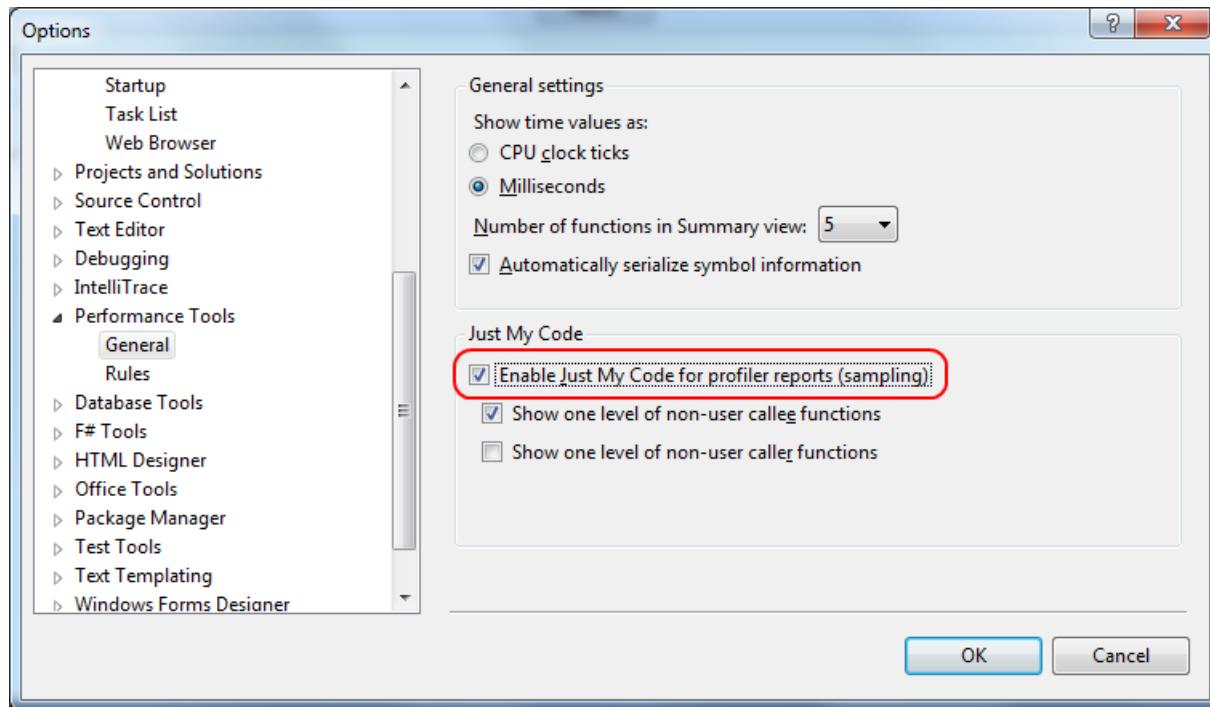
1: Configure Visual Studio for profiling

First, there are a few Visual Studio configuration options that might be helpful when profiling. To make sense of the profiling reports, you'll need symbols (.pdb files) for your application and also symbols for system libraries. You'll want to make sure that you reference the available symbol servers. To do this, on the **Tools** menu in Visual Studio, choose **Options**, then choose **Debugging**, then **Symbols**. Make sure that Microsoft Symbol Servers is listed under **Symbol file (.pdb) locations**. You can also reference <http://referencesource.microsoft.com/symbols>, which might have additional symbol files.

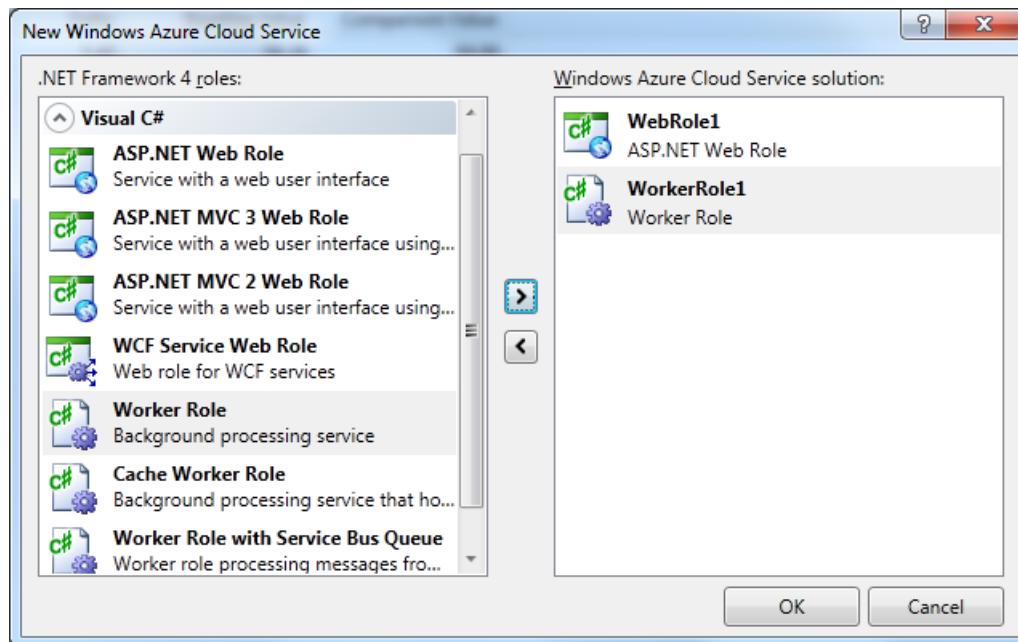


If desired, you can simplify the reports that the profiler generates by setting Just My Code. With Just My Code enabled, function call stacks are simplified so that calls entirely internal to libraries and the .NET Framework are hidden from the reports. On the **Tools** menu, choose **Options**. Then expand the **Performance Tools** node, and

choose **General**. Select the checkbox for **Enable Just My Code for profiler reports**.



You can use these instructions with an existing project or with a new project. If you create a new project to try the techniques described below, choose a C# **Azure Cloud Service** project, and select a **Web Role** and a **Worker Role**.



For example purposes, add some code to your project that takes a lot of time and demonstrates some obvious performance problem. For example, add the following code to a worker role project:

```

public class Concatenator
{
    public static string Concatenate(int number)
    {
        int count;
        string s = "";
        for (count = 0; count < number; count++)
        {
            s += "\n" + count.ToString();
        }
        return s;
    }
}

```

Call this code from the RunAsync method in the worker role's RoleEntryPoint-derived class. (Ignore the warning about the method running synchronously.)

```

private async Task RunAsync(CancellationToken cancellationToken)
{
    // TODO: Replace the following with your own logic.
    while (!cancellationToken.IsCancellationRequested)
    {
        Trace.TraceInformation("Working");
        Concatenator.Concatenate(10000);
    }
}

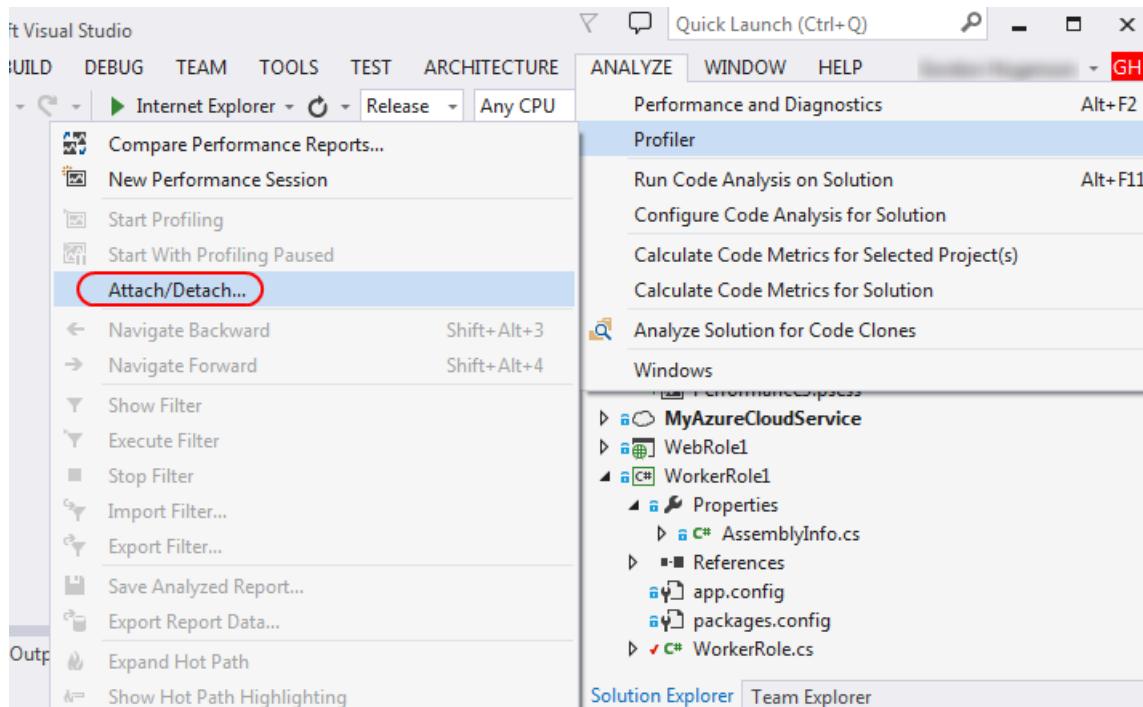
```

Build and run your cloud service locally without debugging (Ctrl+F5), with the solution configuration set to **Release**. This ensures that all files and folders are created for running the application locally, and ensures that all the emulators are started. Start the Compute Emulator UI from the taskbar to verify that your worker role is running.

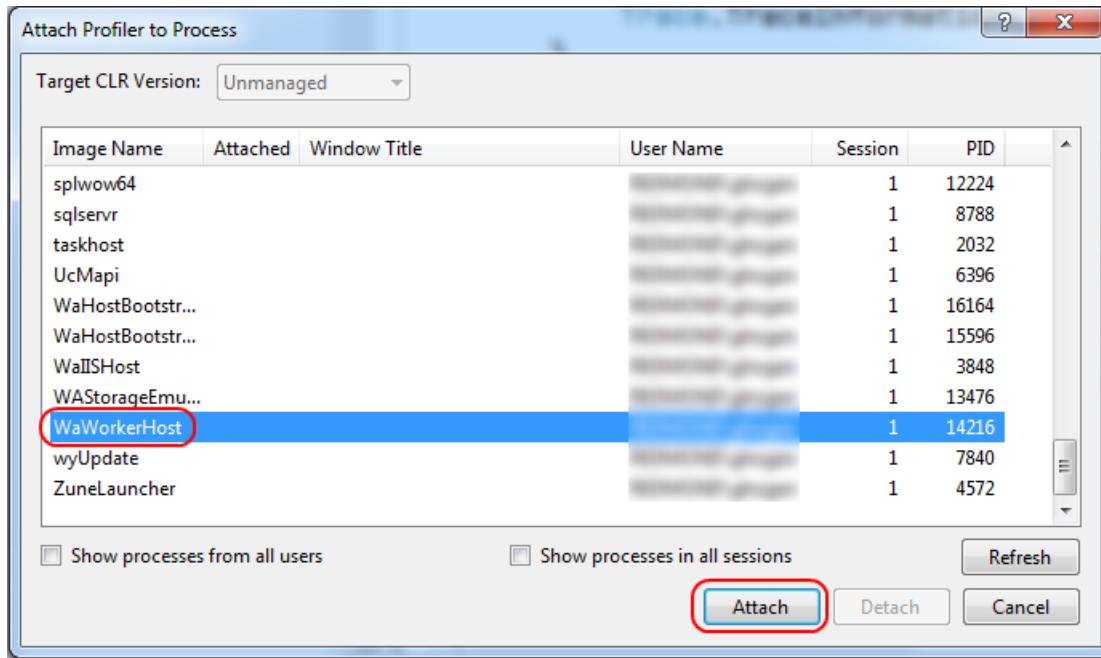
2: Attach to a process

Instead of profiling the application by starting it from the Visual Studio 2010 IDE, you must attach the profiler to a running process.

To attach the profiler to a process, on the **Analyze** menu, choose **Profiler** and **Attach/Detach**.



For a worker role, find the WaWorkerHost.exe process.

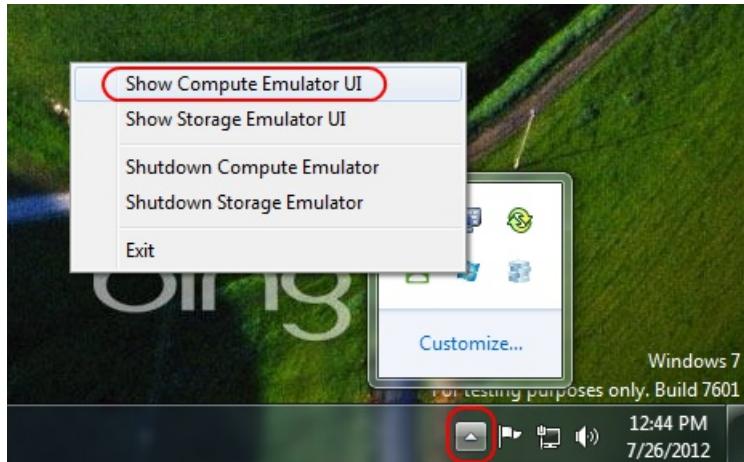


If your project folder is on a network drive, the profiler will ask you to provide another location to save the profiling reports.

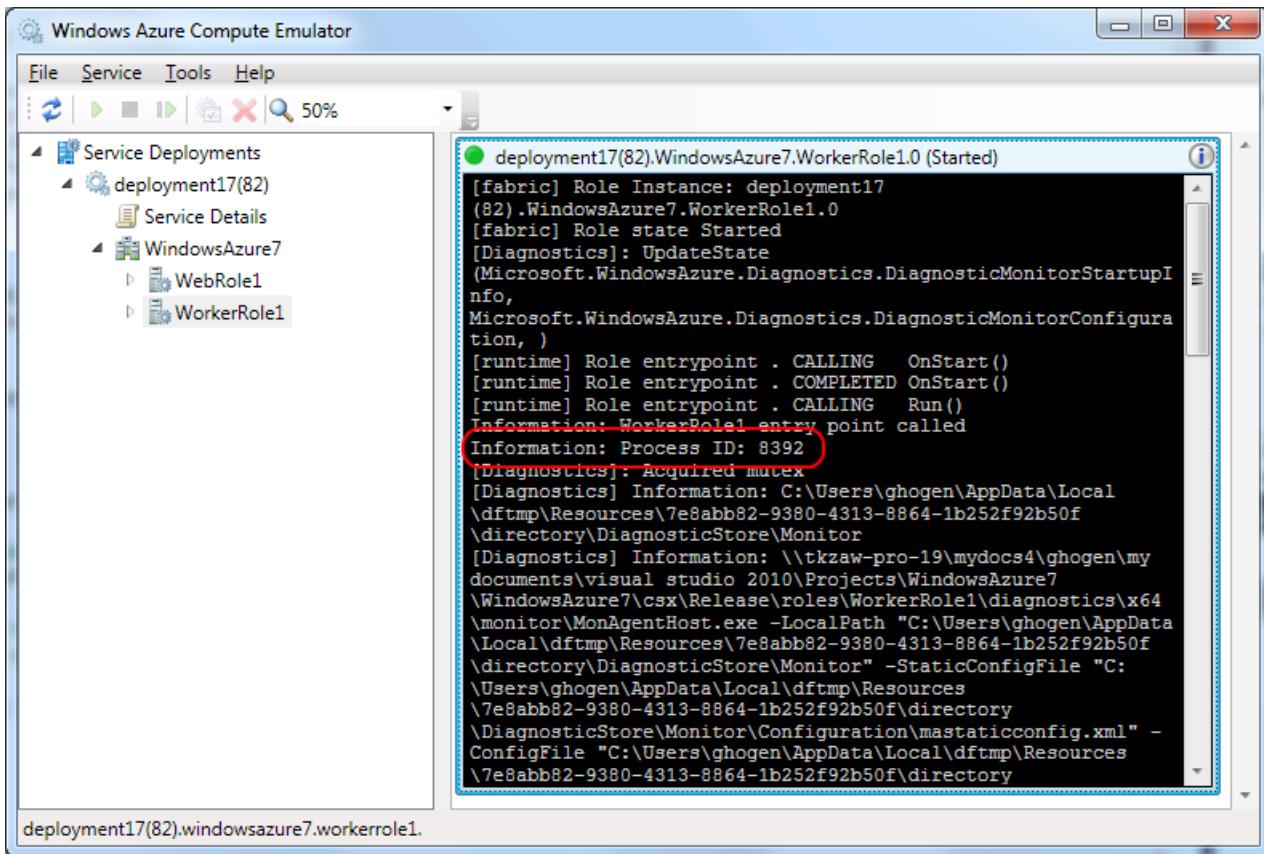
You can also attach to a web role by attaching to WaIISHost.exe. If there are multiple worker role processes in your application, you need to use the processID to distinguish them. You can query the processID programmatically by accessing the Process object. For example, if you add this code to the Run method of the RoleEntryPoint-derived class in a role, you can look at the log in the Compute Emulator UI to know what process to connect to.

```
var process = System.Diagnostics.Process.GetCurrentProcess();
var message = String.Format("Process ID: {0}", process.Id);
Trace.WriteLine(message, "Information");
```

To view the log, start the Compute Emulator UI.

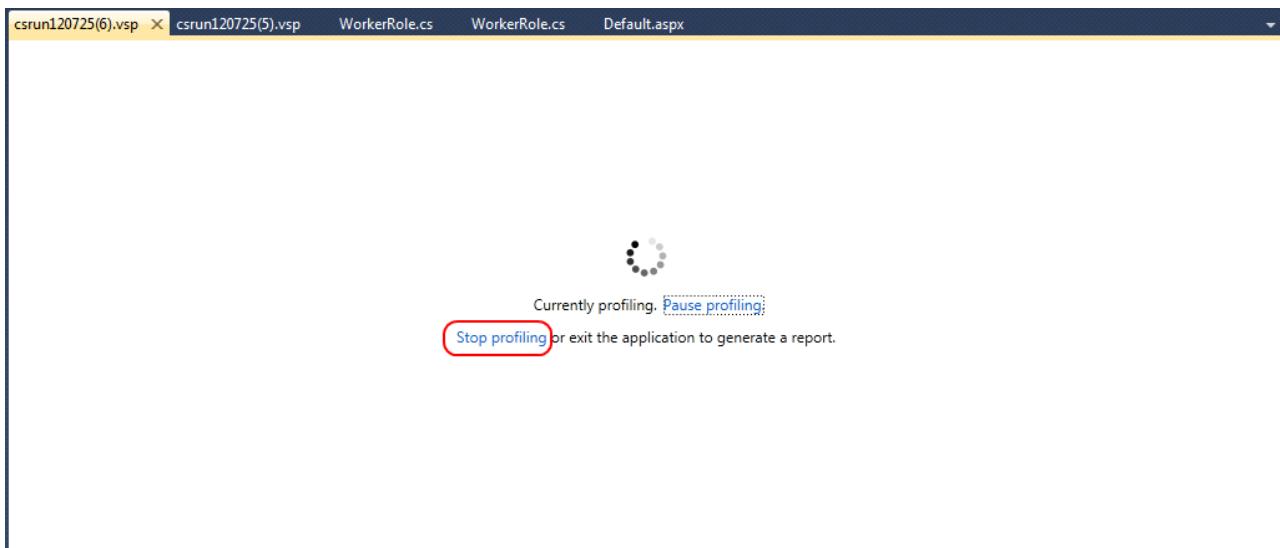


Open the worker role log console window in the Compute Emulator UI by clicking on the console window's title bar. You can see the process ID in the log.



Once you've attached, perform the steps in your application's UI (if needed) to reproduce the scenario.

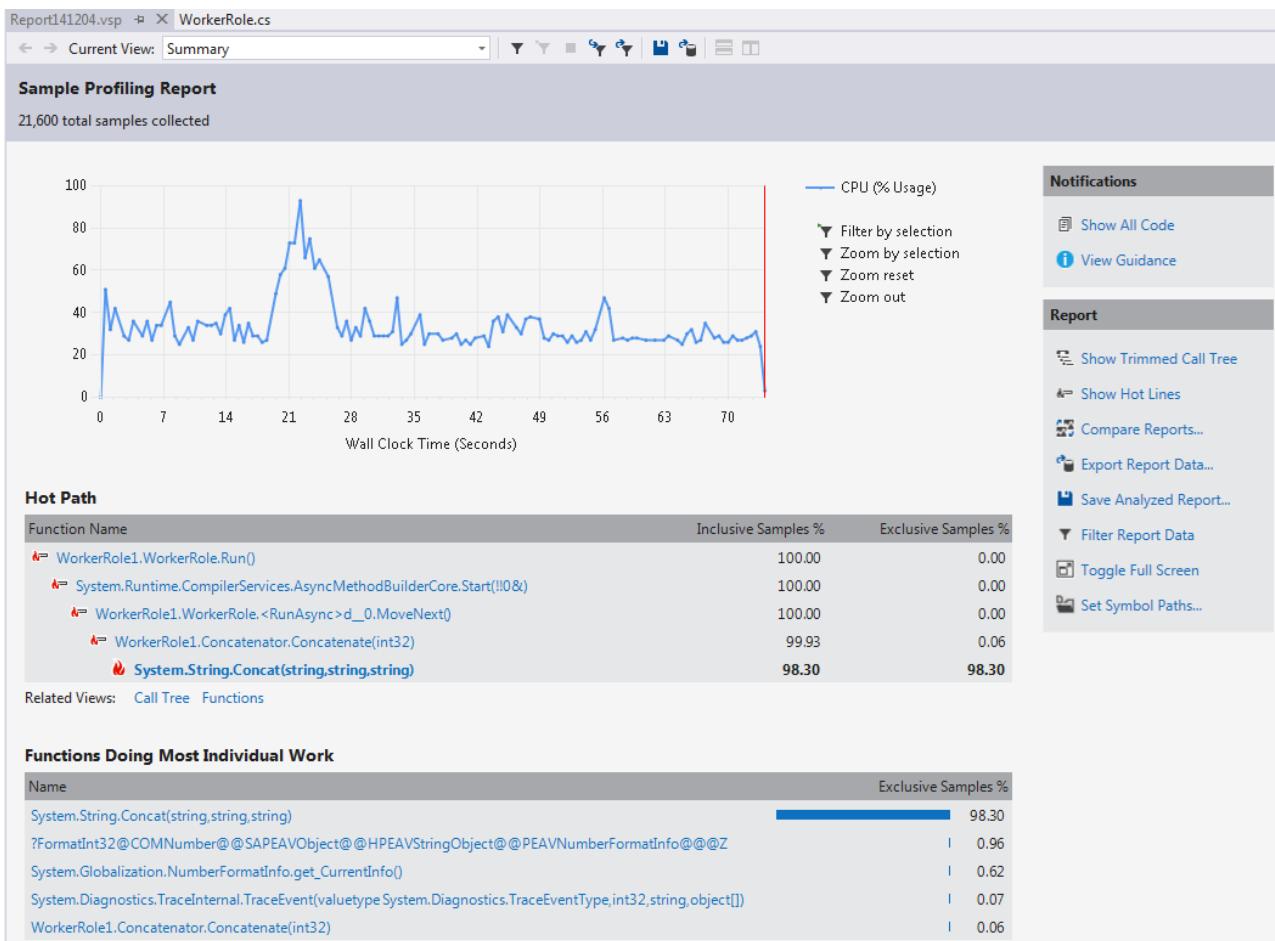
When you want to stop profiling, choose the **Stop Profiling** link.



3: View performance reports

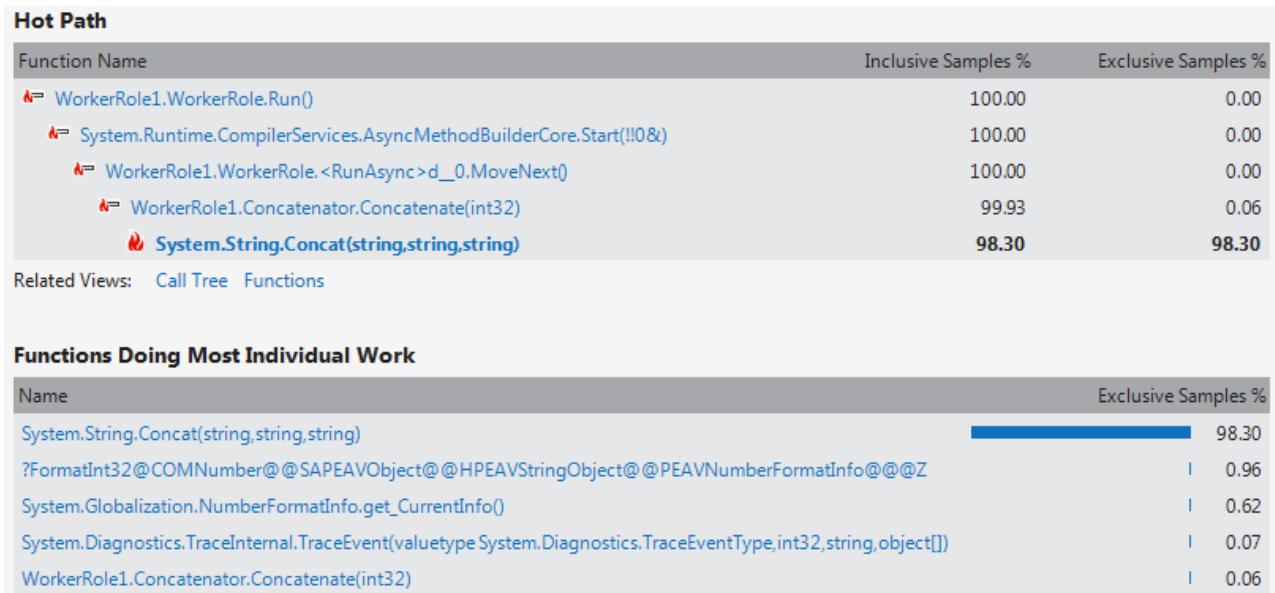
The performance report for your application is displayed.

At this point, the profiler stops executing, saves data in a .vsp file, and displays a report that shows an analysis of this data.



If you see String.wstrcpy in the Hot Path, click on Just My Code to change the view to show user code only. If you see String.Concat, try pressing the Show All Code button.

You should see the Concatenate method and String.Concat taking up a large portion of the execution time.



If you added the string concatenation code in this article, you should see a warning in the Task List for this. You may also see a warning that there is an excessive amount of garbage collection, which is due to the number of strings that are created and disposed.

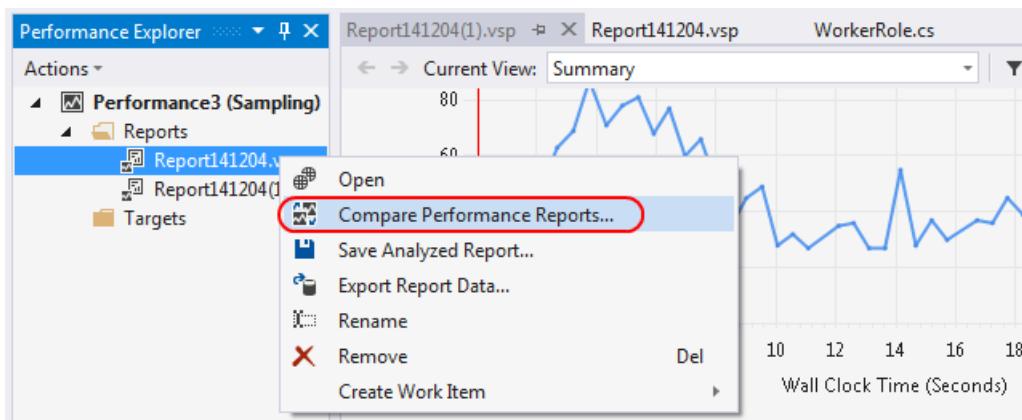
Error List			
2 Warnings	7 Messages	Search Error List	
Description	File	Line	Column
DA0001: System.String.Concat("*) = 98.30; Consider using StringBuilder for string concatenations.	Report141204.vsp	0	0
DA0022: # Gen 1 Collections / # Gen 2 Collections = 1.03; There is a relatively high rate of Gen 2 garbage collections occurring. If, by design, most of your program's data structures are allocated and persisted for a long time, this is not ordinarily a problem. However, if this behavior is unintended, your app may be pinning objects. If you are not certain, you can gather .NET memory allocation data and object lifetime information to understand the pattern of memory allocation your application uses.	Report141204.vsp	0	0
DA0023: (Average)% Time in GC = 23.60; % Time in GC is relatively high. This indication of excessive amount of garbage collection overhead could be impacting the responsiveness of your application. You can gather .NET memory allocation data and object lifetime information to understand the pattern of memory allocation your application uses better.	Report141204.vsp	0	0
DA0501: (Average)% Processor Time = 96.57; Average CPU consumption by the Process being profiled.	Report141204.vsp	0	0
DA0502: (Maximum)% Processor Time = 111.00; This rule is for information only. The Process()% Processor Time counter measures CPU consumption of the process you are profiling. The value reported is the maximum observed over all measurement intervals.	Report141204.vsp	0	0

4: Make changes and compare performance

You can also compare the performance before and after a code change. Stop the running process, and edit the code to replace the string concatenation operation with the use of `StringBuilder`:

```
public static string Concatenate(int number)
{
    int count;
    System.Text.StringBuilder builder = new System.Text.StringBuilder("");
    for (count = 0; count < number; count++)
    {
        builder.Append("\n" + count.ToString());
    }
    return builder.ToString();
}
```

Do another performance run, and then compare the performance. In the Performance Explorer, if the runs are in the same session, you can just select both reports, open the shortcut menu, and choose **Compare Performance Reports**. If you want to compare with a run in another performance session, open the **Analyze** menu, and choose **Compare Performance Reports**. Specify both files in the dialog box that appears.



The reports highlight differences between the two runs.

The screenshot shows the 'Comparison Report' window in Visual Studio. At the top, tabs include 'Comparison Report', 'Report141204(1).vsp', 'Report141204.vsp', and 'WorkerRole.cs'. The main area has two sections: 'Comparison Files' (Baseline File: Report141204.vsp, Comparison File: Report141204(1).vsp) and 'Comparison Options' (Table: Functions, Column: Exclusive Samples %, Threshold: 1). Below these is a yellow header bar with the message 'Comparison complete.' followed by a table of results.

Comparison Column	Delta	Baseline Value	Comparison Value
?FormatInt32@COMNumber	44.42	0.96	45.38
System.String.Concat(string	19.01	0.00	19.01
System.Globalization.Numb	14.93	0.62	15.55
System.Text.StringBuilder.A	12.86	0.00	12.86
WorkerRole1.Concatenator.	3.20	0.06	3.25
System.Text.StringBuilder.To	2.58	0.00	2.58
System.Diagnostics.TraceInt	1.29	0.07	1.36
System.String.Concat(string	-98.30	98.30	0.00

Congratulations! You've gotten started with the profiler.

Troubleshooting

- Make sure you are profiling a Release build and start without debugging.
- If the Attach/Detach option is not enabled on the Profiler menu, run the Performance Wizard.
- Use the Compute Emulator UI to view the status of your application.
- If you have problems starting applications in the emulator, or attaching the profiler, shut down the compute emulator and restart it. If that doesn't solve the problem, try rebooting. This problem can occur if you use the Compute Emulator to suspend and remove running deployments.
- If you have used any of the profiling commands from the command line, especially the global settings, make sure that VSPerfClrEnv /globaloff has been called and that VsPerfMon.exe has been shut down.
- If when sampling, you see the message "PRF0025: No data was collected," check that the process you attached to has CPU activity. Applications that are not doing any computational work might not produce any sampling data. It's also possible that the process exited before any sampling was done. Check to see that the Run method for a role that you are profiling does not terminate.

Next Steps

Instrumenting Azure binaries in the emulator is not supported in the Visual Studio profiler, but if you want to test memory allocation, you can choose that option when profiling. You can also choose concurrency profiling, which helps you determine whether threads are wasting time competing for locks, or tier interaction profiling, which helps you track down performance problems when interacting between tiers of an application, most frequently between the data tier and a worker role. You can view the database queries that your app generates and use the profiling data to improve your use of the database. For information about tier interaction profiling, see the blog post [Walkthrough: Using the Tier Interaction Profiler in Visual Studio Team System 2010](#).

Enable diagnostics in Azure Cloud Services using PowerShell

8/10/2018 • 4 minutes to read • [Edit Online](#)

You can collect diagnostic data like application logs, performance counters etc. from a Cloud Service using the Azure Diagnostics extension. This article describes how to enable the Azure Diagnostics extension for a Cloud Service using PowerShell. See [How to install and configure Azure PowerShell](#) for the prerequisites needed for this article.

Enable diagnostics extension as part of deploying a Cloud Service

This approach is applicable to continuous integration type of scenarios, where the diagnostics extension can be enabled as part of deploying the cloud service. When creating a new Cloud Service deployment you can enable the diagnostics extension by passing in the *ExtensionConfiguration* parameter to the [New-AzureDeployment](#) cmdlet. The *ExtensionConfiguration* parameter takes an array of diagnostics configurations that can be created using the [New-AzureServiceDiagnosticsExtensionConfig](#) cmdlet.

The following example shows how you can enable diagnostics for a cloud service with a WebRole and WorkerRole, each having a different diagnostics configuration.

```
$service_name = "MyService"
$service_package = "CloudService.cspkg"
$service_config = "ServiceConfiguration.Cloud.cscfg"
$webrole_diagconfigpath = "MyService.WebRole.PubConfig.xml"
$workerrole_diagconfigpath = "MyService.WorkerRole.PubConfig.xml"

$webrole_diagconfig = New-AzureServiceDiagnosticsExtensionConfig -Role "WebRole" -DiagnosticsConfigurationPath
$webrole_diagconfigpath
$workerrole_diagconfig = New-AzureServiceDiagnosticsExtensionConfig -Role "WorkerRole" -
DiagnosticsConfigurationPath $workerrole_diagconfigpath

New-AzureDeployment -ServiceName $service_name -Slot Production -Package $service_package -Configuration
$service_config -ExtensionConfiguration @($webrole_diagconfig,$workerrole_diagconfig)
```

If the diagnostics configuration file specifies a `StorageAccount` element with a storage account name, then the `New-AzureServiceDiagnosticsExtensionConfig` cmdlet will automatically use that storage account. For this to work, the storage account needs to be in the same subscription as the Cloud Service being deployed.

From Azure SDK 2.6 onward the extension configuration files generated by the MSBuild publish target output will include the storage account name based on the diagnostics configuration string specified in the service configuration file (.cscfg). The script below shows you how to parse the Extension configuration files from the publish target output and configure diagnostics extension for each role when deploying the cloud service.

```

$service_name = "MyService"
$service_package = "C:\build\output\CloudService.cspkg"
$service_config = "C:\build\output\ServiceConfiguration.Cloud.cscfg"

#Find the Extensions path based on service configuration file
$extensionsSearchPath = Join-Path -Path (Split-Path -Parent $service_config) -ChildPath "Extensions"

$diagnosticsExtensions = Get-ChildItem -Path $extensionsSearchPath -Filter "PaaS.Diagnostics.*.PubConfig.xml"
$diagnosticsConfigurations = @()
foreach ($extPath in $diagnosticsExtensions)
{
    #Find the RoleName based on file naming convention PaaS.Diagnostics.<RoleName>.PubConfig.xml
    $roleName = ""
    $roles = $extPath -split ".",0,"simplematch"
    if ($roles -is [system.array] -and $roles.Length -gt 1)
    {
        $roleName = $roles[1]
        $x = 2
        while ($x -le $roles.Length)
        {
            if ($roles[$x] -ne "PubConfig")
            {
                $roleName = $roleName + "." + $roles[$x]
            }
            else
            {
                break
            }
            $x++
        }
        $fullExtPath = Join-Path -path $extensionsSearchPath -ChildPath $extPath
        $diagnosticsconfig = New-AzureServiceDiagnosticsExtensionConfig -Role $roleName -
        DiagnosticsConfigurationPath $fullExtPath
        $diagnosticsConfigurations += $diagnosticsconfig
    }
}
New-AzureDeployment -ServiceName $service_name -Slot Production -Package $service_package -Configuration
$service_config -ExtensionConfiguration $diagnosticsConfigurations

```

Visual Studio Online uses a similar approach for automated deployments of Cloud Services with the diagnostics extension. See [Publish-AzureCloudDeployment.ps1](#) for a complete example.

If no `StorageAccount` was specified in the diagnostics configuration, then you need to pass in the `StorageAccountName` parameter to the cmdlet. If the `StorageAccountName` parameter is specified, then the cmdlet will always use the storage account that is specified in the parameter and not the one that is specified in the diagnostics configuration file.

If the diagnostics storage account is in a different subscription from the Cloud Service, then you need to explicitly pass in the `StorageAccountName` and `StorageAccountKey` parameters to the cmdlet. The `StorageAccountKey` parameter is not needed when the diagnostics storage account is in the same subscription, as the cmdlet can automatically query and set the key value when enabling the diagnostics extension. However, if the diagnostics storage account is in a different subscription, then the cmdlet might not be able to get the key automatically and you need to explicitly specify the key through the `StorageAccountKey` parameter.

```

$webrole_diagconfig = New-AzureServiceDiagnosticsExtensionConfig -Role "WebRole" -DiagnosticsConfigurationPath
$webrole_diagconfigpath -StorageAccountName $diagnosticsstorage_name -StorageAccountKey $diagnosticsstorage_key
$workerrole_diagconfig = New-AzureServiceDiagnosticsExtensionConfig -Role "WorkerRole" -
DiagnosticsConfigurationPath $workerrole_diagconfigpath -StorageAccountName $diagnosticsstorage_name -
StorageAccountKey $diagnosticsstorage_key

```

Enable diagnostics extension on an existing Cloud Service

You can use the [Set-AzureServiceDiagnosticsExtension](#) cmdlet to enable or update diagnostics configuration on a Cloud Service that is already running.

WARNING

When you enable diagnostics for an existing role, any extensions that you have already set are disabled when the package is deployed. These include:

- Microsoft Monitoring Agent Diagnostics
- Microsoft Azure Security Monitoring
- Microsoft Antimalware
- Microsoft Monitoring Agent
- Microsoft Service Profiler Agent
- Windows Azure Domain Extension
- Windows Azure Diagnostics Extension
- Windows Azure Remote Desktop Extension
- Windows Azure Log Collector

You can reset your extensions via the Azure portal or PowerShell after you deploy the updated role.

```
$service_name = "MyService"
$webrole_diagconfigpath = "MyService.WebRole.PubConfig.xml"
$workerrole_diagconfigpath = "MyService.WorkerRole.PubConfig.xml"

$webrole_diagconfig = New-AzureServiceDiagnosticsExtensionConfig -Role "WebRole" -DiagnosticsConfigurationPath
$webrole_diagconfigpath
$workerrole_diagconfig = New-AzureServiceDiagnosticsExtensionConfig -Role "WorkerRole" -
DiagnosticsConfigurationPath $workerrole_diagconfigpath

Set-AzureServiceDiagnosticsExtension -DiagnosticsConfiguration @($webrole_diagconfig,$workerrole_diagconfig) -
ServiceName $service_name
```

Get current diagnostics extension configuration

Use the [Get-AzureServiceDiagnosticsExtension](#) cmdlet to get the current diagnostics configuration for a cloud service.

```
Get-AzureServiceDiagnosticsExtension -ServiceName "MyService"
```

Remove diagnostics extension

To turn off diagnostics on a cloud service you can use the [Remove-AzureServiceDiagnosticsExtension](#) cmdlet.

```
Remove-AzureServiceDiagnosticsExtension -ServiceName "MyService"
```

If you enabled the diagnostics extension using either *Set-AzureServiceDiagnosticsExtension* or the *New-AzureServiceDiagnosticsExtensionConfig* without the *Role* parameter then you can remove the extension using *Remove-AzureServiceDiagnosticsExtension* without the *Role* parameter. If the *Role* parameter was used when enabling the extension then it must also be used when removing the extension.

To remove the diagnostics extension from each individual role:

```
Remove-AzureServiceDiagnosticsExtension -ServiceName "MyService" -Role "WebRole"
```

Next Steps

- For additional guidance on using Azure diagnostics and other techniques to troubleshoot problems, see [Enabling Diagnostics in Azure Cloud Services and Virtual Machines](#).
- The [Diagnostics Configuration Schema](#) explains the various xml configurations options for the diagnostics extension.
- To learn how to enable the diagnostics extension for Virtual Machines, see [Create a Windows Virtual machine with monitoring and diagnostics using Azure Resource Manager Template](#)

Enabling Azure Diagnostics in Azure Cloud Services

12/12/2018 • 5 minutes to read • [Edit Online](#)

See [Azure Diagnostics Overview](#) for a background on Azure Diagnostics.

How to Enable Diagnostics in a Worker Role

This walkthrough describes how to implement an Azure worker role that emits telemetry data using the .NET EventSource class. Azure Diagnostics is used to collect the telemetry data and store it in an Azure storage account. When creating a worker role, Visual Studio automatically enables Diagnostics 1.0 as part of the solution in Azure SDKs for .NET 2.4 and earlier. The following instructions describe the process for creating the worker role, disabling Diagnostics 1.0 from the solution, and deploying Diagnostics 1.2 or 1.3 to your worker role.

Prerequisites

This article assumes you have an Azure subscription and are using Visual Studio with the Azure SDK. If you do not have an Azure subscription, you can sign up for the [Free Trial](#). Make sure to [Install and configure Azure PowerShell version 0.8.7 or later](#).

Step 1: Create a Worker Role

1. Launch **Visual Studio**.
2. Create an **Azure Cloud Service** project from the **Cloud** template that targets .NET Framework 4.5. Name the project "WadExample" and click Ok.
3. Select **Worker Role** and click Ok. The project will be created.
4. In **Solution Explorer**, double-click the **WorkerRole1** properties file.
5. In the **Configuration** tab, un-check **Enable Diagnostics** to disable Diagnostics 1.0 (Azure SDK 2.4 and earlier).
6. Build your solution to verify that you have no errors.

Step 2: Instrument your code

Replace the contents of WorkerRole.cs with the following code. The class SampleEventSourceWriter, inherited from the [EventSource Class](#), implements four logging methods: **SendEnums**, **MessageMethod**, **SetOther** and **HighFreq**. The first parameter to the **WriteEvent** method defines the ID for the respective event. The Run method implements an infinite loop that calls each of the logging methods implemented in the **SampleEventSourceWriter** class every 10 seconds.

```
using Microsoft.WindowsAzure.ServiceRuntime;
using System;
using System.Diagnostics;
using System.Diagnostics.Tracing;
using System.Net;
using System.Threading;

namespace WorkerRole1
{
    sealed class SampleEventSourceWriter : EventSource
    {
        public static SampleEventSourceWriter Log = new SampleEventSourceWriter();
        public void SendEnums(MyColor color, MyFlags flags) { if (IsEnabled()) WriteEvent(1, (int)color, (int)flags); } // Cast enums to int for efficient logging.
        public void MessageMethod(string Message) { if (IsEnabled()) WriteEvent(2, Message); }
        public void SetOther(bool flag, int myInt) { if (IsEnabled()) WriteEvent(3, flag, myInt); }
        public void HighFreq(int value) { if (IsEnabled()) WriteEvent(4, value); }
    }
}
```

```

}

enum MyColor
{
    Red,
    Blue,
    Green
}

[Flags]
enum MyFlags
{
    Flag1 = 1,
    Flag2 = 2,
    Flag3 = 4
}

public class WorkerRole : RoleEntryPoint
{
    public override void Run()
    {
        // This is a sample worker implementation. Replace with your logic.
        Trace.TraceInformation("WorkerRole1 entry point called");

        int value = 0;

        while (true)
        {
            Thread.Sleep(10000);
            Trace.TraceInformation("Working");

            // Emit several events every time we go through the loop
            for (int i = 0; i < 6; i++)
            {
                SampleEventSourceWriter.Log.SendEnums(MyColor.Blue, MyFlags.Flag2 | MyFlags.Flag3);
            }

            for (int i = 0; i < 3; i++)
            {
                SampleEventSourceWriter.Log.MessageMethod("This is a message.");
                SampleEventSourceWriter.Log.SetOther(true, 123456789);
            }

            if (value == int.MaxValue) value = 0;
            SampleEventSourceWriter.Log.HighFreq(value++);
        }
    }

    public override bool OnStart()
    {
        // Set the maximum number of concurrent connections
        ServicePointManager.DefaultConnectionLimit = 12;

        // For information on handling configuration changes
        // see the MSDN topic at http://go.microsoft.com/fwlink/?LinkId=166357.

        return base.OnStart();
    }
}
}

```

Step 3: Deploy your Worker Role

WARNING

When you enable diagnostics for an existing role, any extensions that you have already set are disabled when the package is deployed. These include:

- Microsoft Monitoring Agent Diagnostics
- Microsoft Azure Security Monitoring
- Microsoft Antimalware
- Microsoft Monitoring Agent
- Microsoft Service Profiler Agent
- Windows Azure Domain Extension
- Windows Azure Diagnostics Extension
- Windows Azure Remote Desktop Extension
- Windows Azure Log Collector

You can reset your extensions via the Azure portal or PowerShell after you deploy the updated role.

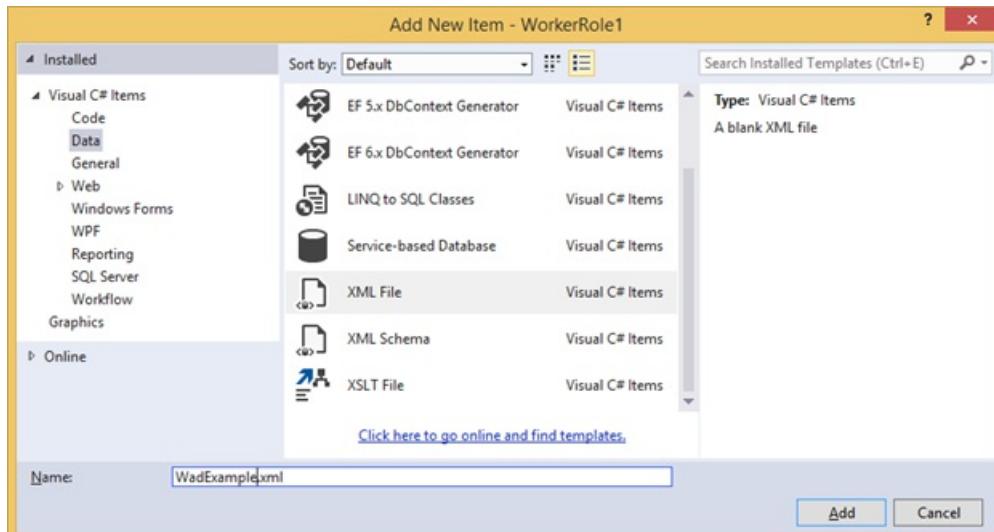
1. Deploy your worker role to Azure from within Visual Studio by selecting the **WadExample** project in the Solution Explorer then **Publish** from the **Build** menu.
2. Choose your subscription.
3. In the **Microsoft Azure Publish Settings** dialog, select **Create New....**
4. In the **Create Cloud Service and Storage Account** dialog, enter a **Name** (for example, "WadExample") and select a region or affinity group.
5. Set the **Environment** to **Staging**.
6. Modify any other **Settings** as appropriate and click **Publish**.
7. After deployment has completed, verify in the Azure portal that your cloud service is in a **Running** state.

Step 4: Create your Diagnostics configuration file and install the extension

1. Download the public configuration file schema definition by executing the following PowerShell command:

```
(Get-AzureServiceAvailableExtension -ExtensionName 'PaaSDiagnostics' -ProviderNamespace  
'Microsoft.Azure.Diagnostics').PublicConfigurationSchema | Out-File -Encoding utf8 -FilePath  
'WadConfig.xsd'
```

2. Add an XML file to your **WorkerRole1** project by right-clicking on the **WorkerRole1** project and select **Add -> New Item... -> Visual C# items -> Data -> XML File**. Name the file "WadExample.xml".



3. Associate the WadConfig.xsd with the configuration file. Make sure the WadExample.xml editor window is

the active window. Press **F4** to open the **Properties** window. Click the **Schemas** property in the **Properties** window. Click the ... in the **Schemas** property. Click the **Add...** button and navigate to the location where you saved the XSD file and select the file WadConfig.xsd. Click **OK**.

4. Replace the contents of the WadExample.xml configuration file with the following XML and save the file. This configuration file defines a couple performance counters to collect: one for CPU utilization and one for memory utilization. Then the configuration defines the four events corresponding to the methods in the SampleEventSourceWriter class.

```
<?xml version="1.0" encoding="utf-8"?>
<PublicConfig xmlns="http://schemas.microsoft.com/ServiceHosting/2010/10/DiagnosticsConfiguration">
  <WadCfg>
    <DiagnosticMonitorConfiguration overallQuotaInMB="25000">
      <PerformanceCounters scheduledTransferPeriod="PT1M">
        <PerformanceCounterConfiguration counterSpecifier="\Processor(_Total)\% Processor Time"
sampleRate="PT1M" unit="percent" />
        <PerformanceCounterConfiguration counterSpecifier="\Memory\Committed Bytes" sampleRate="PT1M"
unit="bytes"/>
      </PerformanceCounters>
      <EtwProviders>
        <EtwEventSourceProviderConfiguration provider="SampleEventSourceWriter"
scheduledTransferPeriod="PT5M">
          <Event id="1" eventDestination="EnumsTable"/>
          <Event id="2" eventDestination="MessageTable"/>
          <Event id="3" eventDestination="SetOtherTable"/>
          <Event id="4" eventDestination="HighFreqTable"/>
          <DefaultEvents eventDestination="DefaultTable" />
        </EtwEventSourceProviderConfiguration>
      </EtwProviders>
    </DiagnosticMonitorConfiguration>
  </WadCfg>
</PublicConfig>
```

Step 5: Install Diagnostics on your Worker Role

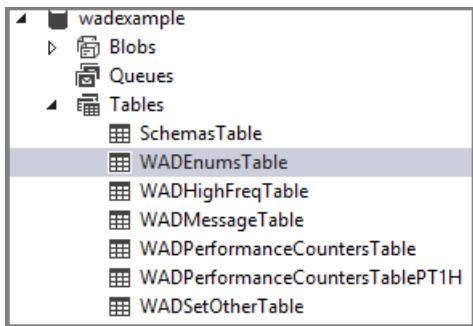
The PowerShell cmdlets for managing Diagnostics on a web or worker role are: Set-AzureServiceDiagnosticsExtension, Get-AzureServiceDiagnosticsExtension, and Remove-AzureServiceDiagnosticsExtension.

1. Open Azure PowerShell.
2. Execute the script to install Diagnostics on your worker role (replace *StorageAccountKey* with the storage account key for your wadexample storage account and *config_path* with the path to the *WadExample.xml* file):

```
$storage_name = "wadexample"
$key = "<StorageAccountKey>"
$config_path="c:\users\\documents\visual studio 2013\Projects\WadExample\WorkerRole1\WadExample.xml"
$service_name="wadexample"
$storageContext = New-AzureStorageContext -StorageAccountName $storage_name -StorageAccountKey $key
Set-AzureServiceDiagnosticsExtension -StorageContext $storageContext -DiagnosticsConfigurationPath
$config_path -ServiceName $service_name -Slot Staging -Role WorkerRole1
```

Step 6: Look at your telemetry data

In the Visual Studio **Server Explorer**, navigate to the wadexample storage account. After the cloud service has been running about five (5) minutes, you should see the tables **WADEnumsTable**, **WADHighFreqTable**, **WADMessageTable**, **WADPerformanceCountersTable** and **WADSetOtherTable**. Double-click one of the tables to view the telemetry that has been collected.



Configuration File Schema

The Diagnostics configuration file defines values that are used to initialize diagnostic configuration settings when the diagnostics agent starts. See the [latest schema reference](#) for valid values and examples.

Troubleshooting

If you have trouble, see [Troubleshooting Azure Diagnostics](#) for help with common problems.

Next Steps

See a [list of related Azure virtual-machine diagnostic articles](#) to change the data you are collecting, troubleshoot problems or learn more about diagnostics in general.

Store and view diagnostic data in Azure Storage

1/8/2019 • 2 minutes to read • [Edit Online](#)

Diagnostic data is not permanently stored unless you transfer it to the Microsoft Azure storage emulator or to Azure storage. Once in storage, it can be viewed with one of several available tools.

Specify a storage account

You specify the storage account that you want to use in the ServiceConfiguration.cscfg file. The account information is defined as a connection string in a configuration setting. The following example shows the default connection string created for a new Cloud Service project in Visual Studio:

```
<ConfigurationSettings>
    <Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"
value="UseDevelopmentStorage=true" />
</ConfigurationSettings>
```

You can change this connection string to provide account information for an Azure storage account.

Depending on the type of diagnostic data that is being collected, Azure Diagnostics uses either the Blob service or the Table service. The following table shows the data sources that are persisted and their format.

DATA SOURCE	STORAGE FORMAT
Azure logs	Table
IIS 7.0 logs	Blob
Azure Diagnostics infrastructure logs	Table
Failed Request Trace logs	Blob
Windows Event logs	Table
Performance counters	Table
Crash dumps	Blob
Custom error logs	Blob

Transfer diagnostic data

For SDK 2.5 and later, the request to transfer diagnostic data can occur through the configuration file. You can transfer diagnostic data at scheduled intervals as specified in the configuration.

For SDK 2.4 and previous you can request to transfer the diagnostic data through the configuration file as well as programmatically. The programmatic approach also allows you to do on-demand transfers.

IMPORTANT

When you transfer diagnostic data to an Azure storage account, you incur costs for the storage resources that your diagnostic data uses.

Store diagnostic data

Log data is stored in either Blob or Table storage with the following names:

Tables

- **WadLogsTable** - Logs written in code using the trace listener.
- **WADDiagnosticInfrastructureLogsTable** - Diagnostic monitor and configuration changes.
- **WADDirectoriesTable** – Directories that the diagnostic monitor is monitoring. This includes IIS logs, IIS failed request logs, and custom directories. The location of the blob log file is specified in the Container field and the name of the blob is in the RelativePath field. The AbsolutePath field indicates the location and name of the file as it existed on the Azure virtual machine.
- **WADPerformanceCountersTable** – Performance counters.
- **WADWindowsEventLogsTable** – Windows Event logs.

Blobs

- **wad-control-container** – (Only for SDK 2.4 and previous) Contains the XML configuration files that controls the Azure diagnostics .
- **wad-iis-failedreqlogfiles** – Contains information from IIS Failed Request logs.
- **wad-iis-logfiles** – Contains information about IIS logs.
- **"custom"** – A custom container based on configuring directories that are monitored by the diagnostic monitor. The name of this blob container will be specified in WADDirectoriesTable.

Tools to view diagnostic data

Several tools are available to view the data after it is transferred to storage. For example:

- Server Explorer in Visual Studio - If you have installed the Azure Tools for Microsoft Visual Studio, you can use the Azure Storage node in Server Explorer to view read-only blob and table data from your Azure storage accounts. You can display data from your local storage emulator account and also from storage accounts you have created for Azure. For more information, see [Browsing and Managing Storage Resources with Server Explorer](#).
- [Microsoft Azure Storage Explorer](#) is a standalone app that enables you to easily work with Azure Storage data on Windows, OSX, and Linux.
- [Azure Management Studio](#) includes Azure Diagnostics Manager which allows you to view, download and manage the diagnostics data collected by the applications running on Azure.

Next Steps

[Trace the flow in a Cloud Services application with Azure Diagnostics](#)

Trace the flow of a Cloud Services application with Azure Diagnostics

7/12/2018 • 2 minutes to read • [Edit Online](#)

Tracing is a way for you to monitor the execution of your application while it is running. You can use the [System.Diagnostics.Trace](#), [System.Diagnostics.Debug](#), and [System.Diagnostics.TraceSource](#) classes to record information about errors and application execution in logs, text files, or other devices for later analysis. For more information about tracing, see [Tracing and Instrumenting Applications](#).

Use trace statements and trace switches

Implement tracing in your Cloud Services application by adding the [DiagnosticMonitorTraceListener](#) to the application configuration and making calls to `System.Diagnostics.Trace` or `System.Diagnostics.Debug` in your application code. Use the configuration file `app.config` for worker roles and the `web.config` for web roles. When you create a new hosted service using a Visual Studio template, Azure Diagnostics is automatically added to the project and the [DiagnosticMonitorTraceListener](#) is added to the appropriate configuration file for the roles that you add.

For information on placing trace statements, see [How to: Add Trace Statements to Application Code](#).

By placing [Trace Switches](#) in your code, you can control whether tracing occurs and how extensive it is. This lets you monitor the status of your application in a production environment. This is especially important in a business application that uses multiple components running on multiple computers. For more information, see [How to: Configure Trace Switches](#).

Configure the trace listener in an Azure application

Trace, Debug and TraceSource, require you set up "listeners" to collect and record the messages that are sent. Listeners collect, store, and route tracing messages. They direct the tracing output to an appropriate target, such as a log, window, or text file. Azure Diagnostics uses the [DiagnosticMonitorTraceListener](#) class.

Before you complete the following procedure, you must initialize the Azure diagnostic monitor. To do this, see [Enabling Diagnostics in Microsoft Azure](#).

Note that if you use the templates that are provided by Visual Studio, the configuration of the listener is added automatically for you.

Add a trace listener

1. Open the `web.config` or `app.config` file for your role.
2. Add the following code to the file. Change the `Version` attribute to use the version number of the assembly you are referencing. The assembly version does not necessarily change with each Azure SDK release unless there are updates to it.

```
<system.diagnostics>
  <trace>
    <listeners>
      <add type="Microsoft.WindowsAzure.Diagnostics.DiagnosticMonitorTraceListener,
        Microsoft.WindowsAzure.Diagnostics,
        Version=2.8.0.0,
        Culture=neutral,
        PublicKeyToken=31bf3856ad364e35"
        name="AzureDiagnostics">
        <filter type="" />
      </add>
    </listeners>
  </trace>
</system.diagnostics>
```

IMPORTANT

Make sure you have a project reference to the Microsoft.WindowsAzure.Diagnostics assembly. Update the version number in the xml above to match the version of the referenced Microsoft.WindowsAzure.Diagnostics assembly.

3. Save the config file.

For more information about listeners, see [Trace Listeners](#).

After you complete the steps to add the listener, you can add trace statements to your code.

To add trace statement to your code

1. Open a source file for your application. For example, the .cs file for the worker role or web role.
2. Add the following using statement if it has not already been added: `using System.Diagnostics;`
3. Add Trace statements where you want to capture information about the state of your application. You can use a variety of methods to format the output of the Trace statement. For more information, see [How to: Add Trace Statements to Application Code](#).
4. Save the source file.

Troubleshooting allocation failure when you deploy Cloud Services in Azure

6/15/2018 • 4 minutes to read • [Edit Online](#)

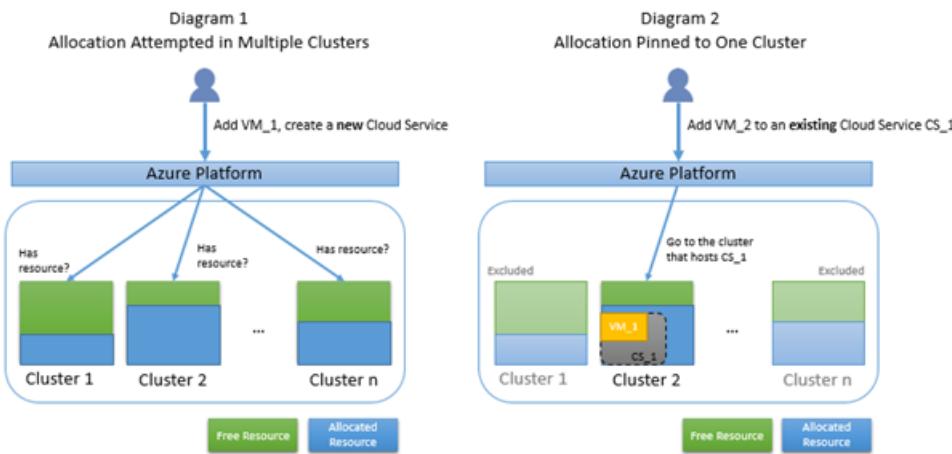
Summary

When you deploy instances to a Cloud Service or add new web or worker role instances, Microsoft Azure allocates compute resources. You may occasionally receive errors when performing these operations even before you reach the Azure subscription limits. This article explains the causes of some of the common allocation failures and suggests possible remediation. The information may also be useful when you plan the deployment of your services.

If your Azure issue is not addressed in this article, visit the Azure forums on [MSDN and Stack Overflow](#). You can post your issue in these forums, or post to [@AzureSupport on Twitter](#). You also can submit an Azure support request. To submit a support request, on the [Azure support](#) page, select **Get support**.

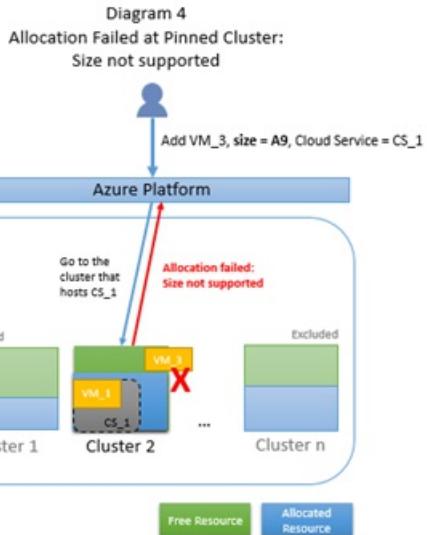
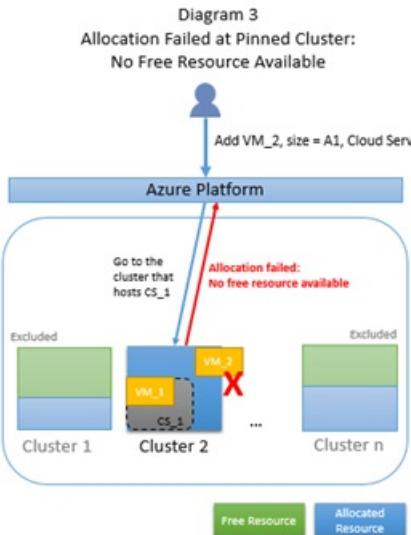
Background – How allocation works

The servers in Azure datacenters are partitioned into clusters. A new cloud service allocation request is attempted in multiple clusters. When the first instance is deployed to a cloud service(in either staging or production), that cloud service gets pinned to a cluster. Any further deployments for the cloud service will happen in the same cluster. In this article, we'll refer to this as "pinned to a cluster". Diagram 1 below illustrates the case of a normal allocation which is attempted in multiple clusters; Diagram 2 illustrates the case of an allocation that's pinned to Cluster 2 because that's where the existing Cloud Service CS_1 is hosted.



Why allocation failure happens

When an allocation request is pinned to a cluster, there's a higher chance of failing to find free resources since the available resource pool is limited to a cluster. Furthermore, if your allocation request is pinned to a cluster but the type of resource you requested is not supported by that cluster, your request will fail even if the cluster has free resource. Diagram 3 below illustrates the case where a pinned allocation fails because the only candidate cluster does not have free resources. Diagram 4 illustrates the case where a pinned allocation fails because the only candidate cluster does not support the requested VM size, even though the cluster has free resources.



Troubleshooting allocation failure for cloud services

Error Message

You may see the following error message:

```
"Azure operation '{operation id}' failed with code Compute.ConstrainedAllocationFailed. Details: Allocation failed; unable to satisfy constraints in request. The requested new service deployment is bound to an Affinity Group, or it targets a Virtual Network, or there is an existing deployment under this hosted service. Any of these conditions constrains the new deployment to specific Azure resources. Please retry later or try reducing the VM size or number of role instances. Alternatively, if possible, remove the aforementioned constraints or try deploying to a different region."
```

Common Issues

Here are the common allocation scenarios that cause an allocation request to be pinned to a single cluster.

- Deploying to Staging Slot - If a cloud service has a deployment in either slot, then the entire cloud service is pinned to a specific cluster. This means that if a deployment already exists in the production slot, then a new staging deployment can only be allocated in the same cluster as the production slot. If the cluster is nearing capacity, the request may fail.
- Scaling - Adding new instances to an existing cloud service must allocate in the same cluster. Small scaling requests can usually be allocated, but not always. If the cluster is nearing capacity, the request may fail.
- Affinity Group - A new deployment to an empty cloud service can be allocated by the fabric in any cluster in that region, unless the cloud service is pinned to an affinity group. Deployments to the same affinity group will be attempted on the same cluster. If the cluster is nearing capacity, the request may fail.
- Affinity Group vNet - Older Virtual Networks were tied to affinity groups instead of regions, and cloud services in these Virtual Networks would be pinned to the affinity group cluster. Deployments to this type of virtual network will be attempted on the pinned cluster. If the cluster is nearing capacity, the request may fail.

Solutions

1. Redeploy to a new cloud service - This solution is likely to be most successful as it allows the platform to choose from all clusters in that region.
 - Deploy the workload to a new cloud service
 - Update the CNAME or A record to point traffic to the new cloud service
 - Once zero traffic is going to the old site, you can delete the old cloud service. This solution should incur zero downtime.
2. Delete both production and staging slots - This solution will preserve your existing DNS name, but will

cause downtime to your application.

- Delete the production and staging slots of an existing cloud service so that the cloud service is empty, and then
 - Create a new deployment in the existing cloud service. This will re-attempt to allocation on all clusters in the region. Ensure the cloud service is not tied to an affinity group.
3. Reserved IP - This solution will preserve your existing IP address, but will cause downtime to your application.

- Create a ReservedIP for your existing deployment using Powershell

```
New-AzureReservedIP -ReservedIPName {new reserved IP name} -Location {location} -ServiceName {existing service name}
```

- Follow #2 from above, making sure to specify the new ReservedIP in the service's CSCFG.
4. Remove affinity group for new deployments - Affinity Groups are no longer recommended. Follow steps for #1 above to deploy a new cloud service. Ensure cloud service is not in an affinity group.
5. Convert to a Regional Virtual Network - See [How to migrate from Affinity Groups to a Regional Virtual Network \(VNet\)](#).

Common issues that cause roles to recycle

11/7/2018 • 3 minutes to read • [Edit Online](#)

This article discusses some of the common causes of deployment problems and provides troubleshooting tips to help you resolve these problems. An indication that a problem exists with an application is when the role instance fails to start, or it cycles between the initializing, busy, and stopping states.

If your Azure issue is not addressed in this article, visit the Azure forums on [MSDN and Stack Overflow](#). You can post your issue in these forums, or post to [@AzureSupport on Twitter](#). You also can submit an Azure support request. To submit a support request, on the [Azure support](#) page, select **Get support**.

Missing runtime dependencies

If a role in your application relies on any assembly that is not part of the .NET Framework or the Azure managed library, you must explicitly include that assembly in the application package. Keep in mind that other Microsoft frameworks are not available on Azure by default. If your role relies on such a framework, you must add those assemblies to the application package.

Before you build and package your application, verify the following:

- If using Visual studio, make sure the **Copy Local** property is set to **True** for each referenced assembly in your project that is not part of the Azure SDK or the .NET Framework.
- Make sure the web.config file does not reference any unused assemblies in the compilation element.
- The **Build Action** of every .cshtml file is set to **Content**. This ensures that the files will appear correctly in the package and enables other referenced files to appear in the package.

Assembly targets wrong platform

Azure is a 64-bit environment. Therefore, .NET assemblies compiled for a 32-bit target won't work on Azure.

Role throws unhandled exceptions while initializing or stopping

Any exceptions that are thrown by the methods of the [RoleEntryPoint](#) class, which includes the [OnStart](#), [OnStop](#), and [Run](#) methods, are unhandled exceptions. If an unhandled exception occurs in one of these methods, the role will recycle. If the role is recycling repeatedly, it may be throwing an unhandled exception each time it tries to start.

Role returns from Run method

The [Run](#) method is intended to run indefinitely. If your code overrides the [Run](#) method, it should sleep indefinitely. If the [Run](#) method returns, the role recycles.

Incorrect DiagnosticsConnectionString setting

If application uses Azure Diagnostics, your service configuration file must specify the `DiagnosticsConnectionString` configuration setting. This setting should specify an HTTPS connection to your storage account in Azure.

To ensure that your `DiagnosticsConnectionString` setting is correct before you deploy your application package to Azure, verify the following:

- The `DiagnosticsConnectionString` setting points to a valid storage account in Azure.
By default, this setting points to the emulated storage account, so you must explicitly change this setting before

you deploy your application package. If you do not change this setting, an exception is thrown when the role instance attempts to start the diagnostic monitor. This may cause the role instance to recycle indefinitely.

- The connection string is specified in the following [format](#). (The protocol must be specified as HTTPS.) Replace *MyAccountName* with the name of your storage account, and *MyAccountKey* with your access key:

```
DefaultEndpointsProtocol=https;AccountName=MyAccountName;AccountKey=MyAccountKey
```

If you are developing your application by using Azure Tools for Microsoft Visual Studio, you can use the property pages to set this value.

Exported certificate does not include private key

To run a web role under SSL, you must ensure that your exported management certificate includes the private key. If you use the *Windows Certificate Manager* to export the certificate, be sure to select **Yes** for the **Export the private key** option. The certificate must be exported in the PFX format, which is the only format currently supported.

Next steps

View more [troubleshooting articles](#) for cloud services.

View more role recycling scenarios at [Kevin Williamson's blog series](#).

Default TEMP folder size is too small on a cloud service web/worker role

11/7/2018 • 2 minutes to read • [Edit Online](#)

The default temporary directory of a cloud service worker or web role has a maximum size of 100 MB, which may become full at some point. This article describes how to avoid running out of space for the temporary directory.

If your Azure issue is not addressed in this article, visit the Azure forums on [MSDN and Stack Overflow](#). You can post your issue in these forums, or post to [@AzureSupport on Twitter](#). You also can submit an Azure support request. To submit a support request, on the [Azure support](#) page, select **Get support**.

Why do I run out of space?

The standard Windows environment variables TEMP and TMP are available to code that is running in your application. Both TEMP and TMP point to a single directory that has a maximum size of 100 MB. Any data that is stored in this directory is not persisted across the lifecycle of the cloud service; if the role instances in a cloud service are recycled, the directory is cleaned.

Suggestion to fix the problem

Implement one of the following alternatives:

- Configure a local storage resource, and access it directly instead of using TEMP or TMP. To access a local storage resource from code that is running within your application, call the [RoleEnvironment.GetLocalResource](#) method.
- Configure a local storage resource, and point the TEMP and TMP directories to point to the path of the local storage resource. This modification should be performed within the [RoleEntryPoint.OnStart](#) method.

The following code example shows how to modify the target directories for TEMP and TMP from within the OnStart method:

```
using System;
using Microsoft.WindowsAzure.ServiceRuntime;

namespace WorkerRole1
{
    public class WorkerRole : RoleEntryPoint
    {
        public override bool OnStart()
        {
            // The local resource declaration must have been added to the
            // service definition file for the role named WorkerRole1:
            //
            // <LocalResources>
            //   <LocalStorage name="CustomTempLocalStore"
            //     cleanOnRoleRecycle="false"
            //     sizeInMB="1024" />
            // </LocalResources>

            string customTempLocalResourcePath =
                RoleEnvironment.GetLocalResource("CustomTempLocalStore").RootPath;
            Environment.SetEnvironmentVariable("TMP", customTempLocalResourcePath);
            Environment.SetEnvironmentVariable("TEMP", customTempLocalResourcePath);

            // The rest of your startup code goes here...

            return base.OnStart();
        }
    }
}
```

Next steps

Read a blog that describes [How to increase the size of the Azure Web Role ASP.NET Temporary Folder](#).

View more [troubleshooting articles](#) for cloud services.

To learn how to troubleshoot cloud service role issues by using Azure PaaS computer diagnostics data, view [Kevin Williamson's blog series](#).

Troubleshoot cloud service deployment problems

11/7/2018 • 3 minutes to read • [Edit Online](#)

When you deploy a cloud service application package to Azure, you can obtain information about the deployment from the **Properties** pane in the Azure portal. You can use the details in this pane to help you troubleshoot problems with the cloud service, and you can provide this information to Azure Support when opening a new support request.

You can find the **Properties** pane as follows:

- In the Azure portal, click the deployment of your cloud service, click **All settings**, and then click **Properties**.

NOTE

You can copy the contents of the **Properties** pane to the clipboard by clicking the icon in the upper-right corner of the pane.

If your Azure issue is not addressed in this article, visit the Azure forums on [MSDN and Stack Overflow](#). You can post your issue in these forums, or post to [@AzureSupport on Twitter](#). You also can submit an Azure support request. To submit a support request, on the [Azure support](#) page, select **Get support**.

Problem: I cannot access my website, but my deployment is started and all role instances are ready

The website URL link shown in the portal does not include the port. The default port for websites is 80. If your application is configured to run in a different port, you must add the correct port number to the URL when accessing the website.

1. In the Azure portal, click the deployment of your cloud service.
2. In the **Properties** pane of the Azure portal, check the ports for the role instances (under **Input Endpoints**).
3. If the port is not 80, add the correct port value to the URL when you access the application. To specify a non-default port, type the URL, followed by a colon (:), followed by the port number, with no spaces.

Problem: My role instances recycled without me doing anything

Service healing occurs automatically when Azure detects problem nodes and therefore moves role instances to new nodes. When this occurs, you might see your role instances recycling automatically. To find out if service healing occurred:

1. In the Azure portal, click the deployment of your cloud service.
2. In the **Properties** pane of the Azure portal, review the information and determine whether service healing occurred during the time that you observed the roles recycling.

Roles will also recycle roughly once per month during host-OS and guest-OS updates.

For more information, see the blog post [Role Instance Restarts Due to OS Upgrades](#)

Problem: I cannot do a VIP swap and receive an error

A VIP swap is not allowed if a deployment update is in progress. Deployment updates can occur automatically when:

- A new guest operating system is available and you are configured for automatic updates.

- Service healing occurs.

To find out if an automatic update is preventing you from doing a VIP swap:

1. In the Azure portal, click the deployment of your cloud service.
2. In the **Properties** pane of the Azure portal, look at the value of **Status**. If it is **Ready**, then check **Last operation** to see if one recently happened that might prevent the VIP swap.
3. Repeat steps 1 and 2 for the production deployment.
4. If an automatic update is in process, wait for it to finish before trying to do the VIP swap.

Problem: A role instance is looping between Started, Initializing, Busy, and Stopped

This condition could indicate a problem with your application code, package, or configuration file. In that case, you should be able to see the status changing every few minutes and the Azure portal may say something like **Recycling**, **Busy**, or **Initializing**. This indicates that there is something wrong with the application that is keeping the role instance from running.

For more information on how to troubleshoot for this problem, see the blog post [Azure PaaS Compute Diagnostics Data](#) and [Common issues that cause roles to recycle](#).

Problem: My application stopped working

1. In the Azure portal, click the role instance.
2. In the **Properties** pane of the Azure portal, consider the following conditions to resolve your problem:
 - If the role instance has recently stopped (you can check the value of **Abort count**), the deployment could be updating. Wait to see if the role instance resumes functioning on its own.
 - If the role instance is **Busy**, check your application code to see if the **StatusCheck** event is handled. You might need to add or fix some code that handles this event.
 - Go through the diagnostic data and troubleshooting scenarios in the blog post [Azure PaaS Compute Diagnostics Data](#).

WARNING

If you recycle your cloud service, you reset the properties for the deployment, effectively erasing the information for the original problem.

Next steps

View more [troubleshooting articles](#) for cloud services.

To learn how to troubleshoot cloud service role issues by using Azure PaaS computer diagnostics data, see [Kevin Williamson's blog series](#).

Troubleshoot Cloud Service roles that fail to start

11/7/2018 • 4 minutes to read • [Edit Online](#)

Here are some common problems and solutions related to Azure Cloud Services roles that fail to start.

If your Azure issue is not addressed in this article, visit the Azure forums on [MSDN and Stack Overflow](#). You can post your issue in these forums, or post to [@AzureSupport on Twitter](#). You also can submit an Azure support request. To submit a support request, on the [Azure support](#) page, select **Get support**.

Missing DLLs or dependencies

Unresponsive roles and roles that are cycling between **Initializing**, **Busy**, and **Stopping** states can be caused by missing DLLs or assemblies.

Symptoms of missing DLLs or assemblies can be:

- Your role instance is cycling through **Initializing**, **Busy**, and **Stopping** states.
- Your role instance has moved to **Ready** but if you navigate to your web application, the page does not appear.

There are several recommended methods for investigating these issues.

Diagnose missing DLL issues in a web role

When you navigate to a website that is deployed in a web role, and the browser displays a server error similar to the following, it may indicate that a DLL is missing.

Server Error in '/' Application.

Runtime Error

Description: An application error occurred on the server. The current custom error settings for this application prevent the details of the application error from being viewed remotely (for security reasons). It could, however, be viewed by browsers running on the local server machine.

Details: To enable the details of this specific error message to be viewable on remote machines, please create a `<customErrors>` tag within a "web.config" configuration file located in the root directory of the current web application. This `<customErrors>` tag should then have its "mode" attribute set to "Off".

Diagnose issues by turning off custom errors

More complete error information can be viewed by configuring the web.config for the web role to set the custom error mode to Off and redeploying the service.

To view more complete errors without using Remote Desktop:

1. Open the solution in Microsoft Visual Studio.
2. In the **Solution Explorer**, locate the web.config file and open it.
3. In the web.config file, locate the system.web section and add the following line:

```
<customErrors mode="Off" />
```

4. Save the file.
5. Repackage and redeploy the service.

Once the service is redeployed, you will see an error message with the name of the missing assembly or DLL.

Diagnose issues by viewing the error remotely

You can use Remote Desktop to access the role and view more complete error information remotely. Use the following steps to view the errors by using Remote Desktop:

1. Ensure that Azure SDK 1.3 or later is installed.
2. During the deployment of the solution by using Visual Studio, enable Remote Desktop. For more information, see [Enable Remote Desktop Connection for a Role in Azure Cloud Services using Visual Studio](#).
3. In the Microsoft Azure portal, once the instance shows a status of **Ready**, remote into the instance. For more information on using the remote desktop with Cloud Services, see [Remote into role instances](#).
4. Sign in to the virtual machine by using the credentials that were specified during the Remote Desktop configuration.
5. Open a command window.
6. Type `IPconfig`.
7. Note the IPV4 Address value.
8. Open Internet Explorer.
9. Type the address and the name of the web application. For example, `http://<IPV4 Address>/default.aspx`.

Navigating to the website will now return more explicit error messages:

- Server Error in '/' Application.
- Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.
- Exception Details: System.IO.FileNotFoundException: Could not load file or assembly 'Microsoft.WindowsAzure.StorageClient, Version=1.1.0.0, Culture=neutral, PublicKeyToken=31bf856ad364e35' or one of its dependencies. The system cannot find the file specified.

For example:

Server Error in '/' Application.

Could not load file or assembly

'Microsoft.WindowsAzure.StorageClient, Version=1.1.0.0,

Culture=neutral, PublicKeyToken=31bf3856ad364e35' or one of its dependencies. The system cannot find the file specified.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.IO.FileNotFoundException: Could not load file or assembly

'Microsoft.WindowsAzure.StorageClient, Version=1.1.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35' or one of its dependencies. The system cannot find the file specified.

Diagnose issues by using the compute emulator

You can use the Microsoft Azure compute emulator to diagnose and troubleshoot issues of missing dependencies and web.config errors.

For best results in using this method of diagnosis, you should use a computer or virtual machine that has a clean installation of Windows. To best simulate the Azure environment, use Windows Server 2008 R2 x64.

1. Install the standalone version of the [Azure SDK](#).
2. On the development machine, build the cloud service project.
3. In Windows Explorer, navigate to the bin\debug folder of the cloud service project.
4. Copy the .csx folder and .cscfg file to the computer that you are using to debug the issues.
5. On the clean machine, open an Azure SDK Command Prompt window and type `csrun.exe /devstore:start`.
6. At the command prompt, type `run csrun <path to .csx folder> <path to .cscfg file> /launchBrowser`.
7. When the role starts, you will see detailed error information in Internet Explorer. You can also use standard Windows troubleshooting tools to further diagnose the problem.

Diagnose issues by using IntelliTrace

For worker and web roles that use .NET Framework 4, you can use [IntelliTrace](#), which is available in Microsoft Visual Studio Enterprise.

Follow these steps to deploy the service with IntelliTrace enabled:

1. Confirm that Azure SDK 1.3 or later is installed.
2. Deploy the solution by using Visual Studio. During deployment, check the **Enable IntelliTrace for .NET 4 roles** check box.
3. Once the instance starts, open the **Server Explorer**.
4. Expand the **Azure\Cloud Services** node and locate the deployment.
5. Expand the deployment until you see the role instances. Right-click on one of the instances.
6. Choose **View IntelliTrace logs**. The **IntelliTrace Summary** will open.
7. Locate the exceptions section of the summary. If there are exceptions, the section will be labeled **Exception**

Data.

8. Expand the **Exception Data** and look for **System.IO.FileNotFoundException** errors similar to the following:

Exception Data		
Type	Message	Thread
System.IO.FileNotFoundException	Could not load file or assembly 'Microsoft.WindowsAzure.StorageClient, Version=1.1.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35' or one of its dependencies. The system cannot find the file specified.	2388

Address missing DLLs and assemblies

To address missing DLL and assembly errors, follow these steps:

1. Open the solution in Visual Studio.
2. In **Solution Explorer**, open the **References** folder.
3. Click the assembly identified in the error.
4. In the **Properties** pane, locate **Copy Local property** and set the value to **True**.
5. Redeploy the cloud service.

Once you have verified that all errors have been corrected, you can deploy the service without checking the **Enable IntelliTrace for .NET 4 roles** check box.

Next steps

View more [troubleshooting articles](#) for cloud services.

To learn how to troubleshoot cloud service role issues by using Azure PaaS computer diagnostics data, see [Kevin Williamson's blog series](#).

What to do in the event of an Azure service disruption that impacts Azure Cloud Services

4/9/2018 • 3 minutes to read • [Edit Online](#)

At Microsoft, we work hard to make sure that our services are always available to you when you need them. Forces beyond our control sometimes impact us in ways that cause unplanned service disruptions.

Microsoft provides a Service Level Agreement (SLA) for its services as a commitment for uptime and connectivity. The SLA for individual Azure services can be found at [Azure Service Level Agreements](#).

Azure already has many built-in platform features that support highly available applications. For more about these services, read [Disaster recovery and high availability for Azure applications](#).

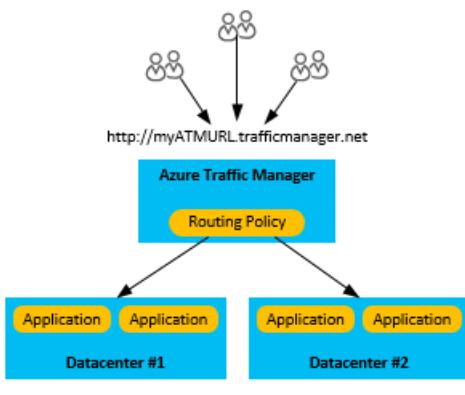
This article covers a true disaster recovery scenario, when a whole region experiences an outage due to major natural disaster or widespread service interruption. These are rare occurrences, but you must prepare for the possibility that there is an outage of an entire region. If an entire region experiences a service disruption, the locally redundant copies of your data would temporarily be unavailable. If you have enabled geo-replication, three additional copies of your Azure Storage blobs and tables are stored in a different region. In the event of a complete regional outage or a disaster in which the primary region is not recoverable, Azure remaps all of the DNS entries to the geo-replicated region.

NOTE

Be aware that you do not have any control over this process, and it will only occur for datacenter-wide service disruptions. Because of this, you must also rely on other application-specific backup strategies to achieve the highest level of availability. For more information, see [Disaster recovery and high availability for applications built on Microsoft Azure](#). If you would like to be able to affect your own failover, you might want to consider the use of [read-access geo-redundant storage \(RA-GRS\)](#), which creates a read-only copy of your data in another region.

Option 1: Use a backup deployment through Azure Traffic Manager

The most robust disaster recovery solution involves maintaining multiple deployments of your application in different regions, then using [Azure Traffic Manager](#) to direct traffic between them. Azure Traffic Manager provides multiple [routing methods](#), so you can choose whether to manage your deployments using a primary/backup model or to split traffic between them.



For the fastest response to the loss of a region, it is important that you configure Traffic Manager's [endpoint](#)

monitoring.

Option 2: Deploy your application to a new region

Maintaining multiple active deployments as described in the previous option incurs additional ongoing costs. If your recovery time objective (RTO) is flexible enough and you have the original code or compiled Cloud Services package, you can create a new instance of your application in another region and update your DNS records to point to the new deployment.

For more detail about how to create and deploy a cloud service application, see [How to create and deploy a cloud service](#).

Depending on your application data sources, you may need to check the recovery procedures for your application data source.

- For Azure Storage data sources, see [Azure Storage replication](#) to check on the options that are available based on the chosen replication model for your application.
- For SQL Database sources, read [Overview: Cloud business continuity and database disaster recovery with SQL Database](#) to check on the options that are available based on the chosen replication model for your application.

Option 3: Wait for recovery

In this case, no action on your part is required, but your service will be unavailable until the region is restored. You can see the current service status on the [Azure Service Health Dashboard](#).

Next steps

To learn more about how to implement a disaster recovery and high availability strategy, see [Disaster recovery and high availability for Azure applications](#).

To develop a detailed technical understanding of a cloud platform's capabilities, see [Azure resiliency technical guidance](#).

Application and service availability issues for Azure Cloud Services: Frequently asked questions (FAQs)

11/7/2018 • 2 minutes to read • [Edit Online](#)

This article includes frequently asked questions about application and service availability issues for [Microsoft Azure Cloud Services](#). You can also consult the [Cloud Services VM Size page](#) for size information.

If your Azure issue is not addressed in this article, visit the Azure forums on [MSDN and Stack Overflow](#). You can post your issue in these forums, or post to [@AzureSupport on Twitter](#). You also can submit an Azure support request. To submit a support request, on the [Azure support](#) page, select **Get support**.

My role got recycled. Was there any update rolled out for my cloud service?

Roughly once a month, Microsoft releases a new Guest OS version for Windows Azure PaaS VMs. The Guest OS is only one such update in the pipeline. A release can be affected by many other factors. In addition, Azure runs on hundreds of thousands of machines. Therefore, it's impossible to predict the exact date and time when your roles will reboot. We update the Guest OS Update RSS Feed with the latest information that we have, but you should consider that reported time to be an approximate value. We are aware that this is problematic for customers and are working on a plan to limit or precisely time reboots.

For complete details about recent Guest OS updates, see [Azure Guest OS releases and SDK compatibility matrix](#).

For helpful information on restarts and pointers to technical details of Guest and Host OS updates, see the MSDN blog post [Role Instance Restarts Due to OS Upgrades](#).

Why does the first request to my cloud service after the service has been idle for some time take longer than usual?

When the Web Server receives the first request, it first recompiles the code and then processes the request. That's why the first request takes longer than the others. By default, the app pool gets shut down in cases of user inactivity. The app pool will also recycle by default every 1,740 minutes (29 hours).

Internet Information Services (IIS) application pools can be periodically recycled to avoid unstable states that can lead to application crashes, hangs, or memory leaks.

The following documents will help you understand and mitigate this issue:

- [Fixing slow initial load for IIS](#)
- [IIS 7.5 web application first request after app-pool recycle very slow](#)

If you want to change the default behavior of IIS, you will need to use startup tasks, because if you manually apply changes to the Web Role instances, the changes will eventually be lost.

For more information, see [How to configure and run startup tasks for a cloud service](#).

Configuration and management issues for Azure Cloud Services: Frequently asked questions (FAQs)

1/9/2019 • 14 minutes to read • [Edit Online](#)

This article includes frequently asked questions about configuration and management issues for [Microsoft Azure Cloud Services](#). You can also consult the [Cloud Services VM Size page](#) for size information.

If your Azure issue is not addressed in this article, visit the Azure forums on [MSDN and Stack Overflow](#). You can post your issue in these forums, or post to [@AzureSupport on Twitter](#). You also can submit an Azure support request. To submit a support request, on the [Azure support](#) page, select **Get support**.

Certificates

- [Why is the certificate chain of my Cloud Service SSL certificate incomplete?](#)
- [What is the purpose of the "Windows Azure Tools Encryption Certificate for Extensions"?](#)
- [How can I generate a Certificate Signing Request \(CSR\) without "RDP-ing" in to the instance?](#)
- [My Cloud Service Management Certificate is expiring. How to renew it?](#)
- [How to automate the installation of main SSL certificate\(.pfx\) and intermediate certificate\(.p7b\)?](#)
- [What is the purpose of the "Microsoft Azure Service Management for MachineKey" certificate?](#)

Monitoring and logging

- [What are the upcoming Cloud Service capabilities in the Azure portal which can help manage and monitor applications?](#)
- [Why does IIS stop writing to the log directory?](#)
- [How do I enable WAD logging for Cloud Services?](#)

Network configuration

- [How do I set the idle timeout for Azure load balancer?](#)
- [How do I associate a static IP address to my Cloud Service?](#)
- [What are the features and capabilities that Azure basic IPS/IDS and DDOS provides?](#)
- [How to enable HTTP/2 on Cloud Services VM?](#)

Permissions

- [Can Microsoft internal engineers remote desktop to Cloud Service instances without permission?](#)
- [I cannot remote desktop to Cloud Service VM by using the RDP file. I get following error: An authentication error has occurred \(Code: 0x80004005\)](#)

Scaling

- [I cannot scale beyond X instances](#)
- [How can I configure Auto-Scale based on Memory metrics?](#)

Generic

- [How do I add "nosniff" to my website?](#)
- [How do I customize IIS for a web role?](#)
- [What is the quota limit for my Cloud Service?](#)
- [Why does the drive on my Cloud Service VM show very little free disk space?](#)

- How can I add an Antimalware extension for my Cloud Services in an automated way?
- How to enable Server Name Indication (SNI) for Cloud Services?
- How can I add tags to my Azure Cloud Service?
- The Azure portal doesn't display the SDK version of my Cloud Service. How can I get that?
- I want to shut down the Cloud Service for several months. How to reduce the billing cost of Cloud Service without losing the IP address?

Certificates

Why is the certificate chain of my Cloud Service SSL certificate incomplete?

We recommend that customers install the full certificate chain (leaf cert, intermediate certs, and root cert) instead of just the leaf certificate. When you install just the leaf certificate, you rely on Windows to build the certificate chain by walking the CTL. If intermittent network or DNS issues occur in Azure or Windows Update when Windows is trying to validate the certificate, the certificate may be considered invalid. By installing the full certificate chain, this problem can be avoided. The blog at [How to install a chained SSL certificate](#) shows how to do this.

What is the purpose of the "Windows Azure Tools Encryption Certificate for Extensions"?

These certificates are automatically created whenever an extension is added to the Cloud Service. Most commonly, this is the WAD extension or the RDP extension, but it could be others, such as the Antimalware or Log Collector extension. These certificates are only used for encrypting and decrypting the private configuration for the extension. The expiration date is never checked, so it doesn't matter if the certificate is expired.

You can ignore these certificates. If you want to clean up the certificates, you can try deleting them all. Azure will throw an error if you try to delete a certificate that is in use.

How can I generate a Certificate Signing Request (CSR) without "RDP-ing" in to the instance?

See the following guidance document:

[Obtaining a certificate for use with Windows Azure Web Sites \(WAWS\)](#)

The CSR is just a text file. It does not have to be created from the machine where the certificate will ultimately be used. Although this document is written for an App Service, the CSR creation is generic and applies also for Cloud Services.

My Cloud Service Management Certificate is expiring. How to renew it?

You can use following PowerShell commands to renew your Management Certificates:

```
Add-AzureAccount
Select-AzureSubscription -Current -SubscriptionName <your subscription name>
Get-AzurePublishSettingsFile
```

The **Get-AzurePublishSettingsFile** will create a new management certificate in **Subscription > Management Certificates** in the Azure portal. The name of the new certificate looks like "YourSubscriptionName-[CurrentDate]-credentials".

How to automate the installation of main SSL certificate(.pfx) and intermediate certificate(.p7b)?

You can automate this task by using a startup script (batch/cmd/PowerShell) and register that startup script in the service definition file. Add both the startup script and certificate (.p7b file) in the project folder of the same directory of the startup script.

What is the purpose of the "Microsoft Azure Service Management for MachineKey" certificate?

This certificate is used to encrypt machine keys on Azure Web Roles. To learn more, check out this advisory [<https://docs.microsoft.com/security-updates/securityadvisories/2018/4092731>].

For more information, see the following articles:

- [How to configure and run startup tasks for a Cloud Service](#)
- [Common Cloud Service startup tasks](#)

Monitoring and logging

What are the upcoming Cloud Service capabilities in the Azure portal which can help manage and monitor applications?

Ability to generate a new certificate for Remote Desktop Protocol (RDP) is coming soon. Alternatively, you can run this script:

```
$cert = New-SelfSignedCertificate -DnsName yourdomain.cloudapp.net -CertStoreLocation "cert:\LocalMachine\My" -  
KeyLength 20 48 -KeySpec "KeyExchange"  
$password = ConvertTo-SecureString -String "your-password" -Force -AsPlainText  
Export-PfxCertificate -Cert $cert -FilePath ".\my-cert-file.pfx" -Password $password
```

Ability to choose blob or local for your csdef and cscfg upload location is coming soon. Using [New-AzureDeployment](#), you can set each location value.

Ability to monitor metrics at the instance level. Additional monitoring capabilities are available in [How to Monitor Cloud Services](#).

Why does IIS stop writing to the log directory?

You have exhausted the local storage quota for writing to the log directory. To correct this, you can do one of three things:

- Enable diagnostics for IIS and have the diagnostics periodically moved to blob storage.
- Manually remove log files from the logging directory.
- Increase quota limit for local resources.

For more information, see the following documents:

- [Store and view diagnostic data in Azure Storage](#)
- [IIS Logs stop writing in Cloud Service](#)

How do I enable WAD logging for Cloud Services?

You can enable Windows Azure Diagnostics (WAD) logging through following options:

1. [Enable from Visual Studio](#)
2. [Enable through .Net code](#)
3. [Enable through Powershell](#)

In order to get the current WAD settings of your Cloud Service, you can use [Get-AzureServiceDiagnosticsExtensions](#) ps cmd or you can view it through portal from "Cloud Services --> Extensions" blade.

Network configuration

How do I set the idle timeout for Azure load balancer?

You can specify the timeout in your service definition (csdef) file like this:

```

<?xml version="1.0" encoding="utf-8"?>
<ServiceDefinition name="mgVS2015Worker"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition" schemaVersion="2015-04.2.6">
<WorkerRole name="WorkerRole1" vmsize="Small">
<ConfigurationSettings>
<Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString" />
</ConfigurationSettings>
<Imports>
<Import moduleName="RemoteAccess" />
<Import moduleName="RemoteForwarder" />
</Imports>
<Endpoints>
<InputEndpoint name="Endpoint1" protocol="tcp" port="10100" idleTimeoutInMinutes="30" />
</Endpoints>
</WorkerRole>

```

See [New: Configurable Idle Timeout for Azure Load Balancer](#) for more information.

How do I associate a static IP address to my Cloud Service?

To set up a static IP address, you need to create a reserved IP. This reserved IP can be associated to a new Cloud Service or to an existing deployment. See the following documents for details:

- [How to create a reserved IP address](#)
- [Reserve the IP address of an existing Cloud Service](#)
- [Associate a reserved IP to a new Cloud Service](#)
- [Associate a reserved IP to a running deployment](#)
- [Associate a reserved IP to a Cloud Service by using a service configuration file](#)

What are the features and capabilities that Azure basic IPS/IDS and DDOS provides?

Azure has IPS/IDS in datacenter physical servers to defend against threats. In addition, customers can deploy third-party security solutions, such as web application firewalls, network firewalls, antimalware, intrusion detection, prevention systems (IDS/IPS), and more. For more information, see [Protect your data and assets and comply with global security standards](#).

Microsoft continuously monitors servers, networks, and applications to detect threats. Azure's multipronged threat-management approach uses intrusion detection, distributed denial-of-service (DDoS) attack prevention, penetration testing, behavioral analytics, anomaly detection, and machine learning to constantly strengthen its defense and reduce risks. Microsoft Antimalware for Azure protects Azure Cloud Services and virtual machines. You have the option to deploy third-party security solutions in addition, such as web application fire walls, network firewalls, antimalware, intrusion detection and prevention systems (IDS/IPS), and more.

How to enable HTTP/2 on Cloud Services VM?

Windows 10 and Windows Server 2016 come with support for HTTP/2 on both client and server side. If your client (browser) is connecting to the IIS server over TLS that negotiates HTTP/2 via TLS extensions, then you do not need to make any change on the server-side. This is because, over TLS, the h2-14 header specifying use of HTTP/2 is sent by default. If on the other hand your client is sending an Upgrade header to upgrade to HTTP/2, then you need to make the change below on the server side to ensure that the Upgrade works and you end up with an HTTP/2 connection.

1. Run regedit.exe.
2. Browse to registry key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\HTTP\Parameters.
3. Create a new DWORD value named **DuoEnabled**.
4. Set its value to 1.
5. Restart your server.
6. Go to your **Default Web Site** and under **Bindings**, create a new TLS binding with the self-signed certificate

just created.

For more information, see:

- [HTTP/2 on IIS](#)
- [Video: HTTP/2 in Windows 10: Browser, Apps and Web Server](#)

These steps could be automated via a startup task, so that whenever a new PaaS instance gets created, it can do the changes above in the system registry. For more information, see [How to configure and run startup tasks for a Cloud Service](#).

Once this has been done, you can verify whether the HTTP/2 has been enabled or not by using one of the following methods:

- Enable Protocol version in IIS logs and look into the IIS logs. It will show HTTP/2 in the logs.
- Enable F12 Developer Tool in Internet Explorer/Microsoft Edge and switch to the Network tab to verify the protocol.

For more information, see [HTTP/2 on IIS](#).

Permissions

How can I implement role-based access for Cloud Services?

Cloud Services doesn't support the role-based access control (RBAC) model, as it's not an Azure Resource Manager based service.

See [Understand the different roles in Azure](#).

Remote desktop

Can Microsoft internal engineers remote desktop to Cloud Service instances without permission?

Microsoft follows a strict process that will not allow internal engineers to remote desktop into your Cloud Service without written permission (email or other written communication) from the owner or their designee.

I cannot remote desktop to Cloud Service VM by using the RDP file. I get following error: An authentication error has occurred (Code: 0x80004005)

This error may occur if you use the RDP file from a machine that is joined to Azure Active Directory. To resolve this issue, follow these steps:

1. Right-click the RDP file you downloaded and then select **Edit**.
2. Add "\" as prefix before the username. For example, use **\username** instead of **username**.

Scaling

I cannot scale beyond X instances

Your Azure Subscription has a limit on the number of cores you can use. Scaling will not work if you have used all the cores available. For example, if you have a limit of 100 cores, this means you could have 100 A1 sized virtual machine instances for your Cloud Service, or 50 A2 sized virtual machine instances.

How can I configure Auto-Scale based on Memory metrics?

Auto-scale based on Memory metrics for a Cloud Services is not currently supported.

To work around this problem, you can use Application Insights. Auto-Scale supports Application Insights as a Metrics Source and can scale the role instance count based on guest metric like "Memory". You have to configure Application Insights in your Cloud Service project package file (*.cspkg) and enable Azure Diagnostics extension on the service to implement this feat.

For more details on how to utilize a custom metric via Application Insights to configure Auto-Scale on Cloud Services, see [Get started with auto scale by custom metric in Azure](#)

For more information on how to integrate Azure Diagnostics with Application Insights for Cloud Services, see [Send Cloud Service, Virtual Machine, or Service Fabric diagnostic data to Application Insights](#)

For more information about to enable Application Insights for Cloud Services, see [Application Insights for Azure Cloud Services](#)

For more information about how to enable Azure Diagnostics Logging for Cloud Services, see [Set up diagnostics for Azure Cloud Services and virtual machines](#)

Generic

How do I add "nosniff" to my website?

To prevent clients from sniffing the MIME types, add a setting in your *web.config* file.

```
<configuration>
  <system.webServer>
    <httpProtocol>
      <customHeaders>
        <add name="X-Content-Type-Options" value="nosniff" />
      </customHeaders>
    </httpProtocol>
  </system.webServer>
</configuration>
```

You can also add this as a setting in IIS. Use the following command with the [common startup tasks](#) article.

```
%windir%\system32\inetsrv\appcmd set config /section:httpProtocol /+customHeaders.[name='X-Content-Type-Options',value='nosniff']
```

How do I customize IIS for a web role?

Use the IIS startup script from the [common startup tasks](#) article.

What is the quota limit for my Cloud Service?

See [Service-specific limits](#).

Why does the drive on my Cloud Service VM show very little free disk space?

This is expected behavior, and it shouldn't cause any issue to your application. Journaling is turned on for the %approot% drive in Azure PaaS VMs, which essentially consumes double the amount of space that files normally take up. However there are several things to be aware of that essentially turn this into a non-issue.

The %approot% drive size is calculated as <size of .cspkg + max journal size + a margin of free space>, or 1.5 GB, whichever is larger. The size of your VM has no bearing on this calculation. (The VM size only affects the size of the temporary C: drive.)

It is unsupported to write to the %approot% drive. If you are writing to the Azure VM, you must do so in a temporary LocalStorage resource (or other option, such as Blob storage, Azure Files, etc.). So the amount of free space on the %approot% folder is not meaningful. If you are not sure if your application is writing to the %approot% drive, you can always let your service run for a few days and then compare the "before" and "after" sizes.

Azure will not write anything to the %approot% drive. Once the VHD is created from your .cspkg and mounted into the Azure VM, the only thing that might write to this drive is your application.

The journal settings are non-configurable, so you can't turn it off.

How can I add an Antimalware extension for my Cloud Services in an automated way?

You can enable Antimalware extension using PowerShell script in the Startup Task. Follow the steps in these articles to implement it:

- [Create a PowerShell startup task](#)
- [Set-AzureServiceAntimalwareExtension](#)

For more information about Antimalware deployment scenarios and how to enable it from the portal, see [Antimalware Deployment Scenarios](#).

How to enable Server Name Indication (SNI) for Cloud Services?

You can enable SNI in Cloud Services by using one of the following methods:

Method 1: Use PowerShell

The SNI binding can be configured using the PowerShell cmdlet **New-WebBinding** in a startup task for a Cloud Service role instance as below:

```
New-WebBinding -Name $WebsiteName -Protocol "https" -Port 443 -IPAddress $IPAddress -HostHeader $HostHeader -SslFlags $sslFlags
```

As described [here](#), the \$sslFlags could be one of the values as the following:

VALUE	MEANING
0	No SNI
1	SNI Enabled
2	Non SNI binding which uses Central Certificate Store
3	SNI binding which uses Central Certificate store

Method 2: Use code

The SNI binding could also be configured via code in the role startup as described on this [blog post](#):

```
//<code snip>
var serverManager = new ServerManager();
var site = serverManager.Sites[0];
var binding = site.Bindings.Add(":443:www.test1.com", newCert.GetCertHash(), "My");
binding.SetAttributeValue("sslFlags", 1); //enables the SNI
serverManager.CommitChanges();
//</code snip>
```

Using any of the approaches above, the respective certificates (*.pfx) for the specific hostnames have to be first installed on the role instances using a startup task or via code in order for the SNI binding to be effective.

How can I add tags to my Azure Cloud Service?

Cloud Service is a Classic resource. Only resources created through Azure Resource Manager support tags. You cannot apply tags to Classic resources such as Cloud Service.

The Azure portal doesn't display the SDK version of my Cloud Service. How can I get that?

We are working on bringing this feature on the Azure portal. Meanwhile, you can use following PowerShell

commands to get the SDK version:

```
Get-AzureService -ServiceName "<Cloud Service name>" | Get-AzureDeployment | Where-Object -Property SdkVersion -NE -Value "" | select ServiceName,SdkVersion,OSVersion,Slot
```

I want to shut down the Cloud Service for several months. How to reduce the billing cost of Cloud Service without losing the IP address?

An already deployed Cloud Service gets billed for the Compute and Storage it uses. So even if you shut down the Azure VM, you will still get billed for the Storage.

Here is what you can do to reduce your billing without losing the IP address for your service:

1. [Reserve the IP address](#) before you delete the deployments. You will only be billed for this IP address. For more information about IP address billing, see [IP addresses pricing](#).
2. Delete the deployments. Don't delete the xxx.cloudapp.net, so that you can use it for future.
3. If you want to redeploy the Cloud Service by using the same reserve IP that you reserved in your subscription, see [Reserved IP addresses for Cloud Services and Virtual Machines](#).

Connectivity and networking issues for Azure Cloud Services: Frequently asked questions (FAQs)

11/7/2018 • 6 minutes to read • [Edit Online](#)

This article includes frequently asked questions about connectivity and networking issues for [Azure Cloud Services](#). For size information, see the [Cloud Services VM size page](#).

If your Azure issue is not addressed in this article, visit the Azure forums on [MSDN and Stack Overflow](#). You can post your issue in these forums, or post to [@AzureSupport on Twitter](#). You also can submit an Azure support request. To submit a support request, on the [Azure support](#) page, select **Get support**.

I can't reserve an IP in a multi-VIP cloud service.

First, make sure that the virtual machine instance that you try to reserve the IP for is turned on. Second, make sure that you use reserved IPs for both the staging and production deployments. *Do not* change the settings while the deployment is upgrading.

How do I use Remote Desktop when I have an NSG?

Add rules to the NSG that allow traffic on ports **3389** and **20000**. Remote Desktop uses port **3389**. Cloud service instances are load balanced, so you can't directly control which instance to connect to. The *RemoteForwarder* and *RemoteAccess* agents manage Remote Desktop Protocol (RDP) traffic and allow the client to send an RDP cookie and specify an individual instance to connect to. The *RemoteForwarder* and *RemoteAccess* agents require port **20000** to be open, which might be blocked if you have an NSG.

Can I ping a cloud service?

No, not by using the normal "ping"/ICMP protocol. The ICMP protocol is not permitted through the Azure load balancer.

To test connectivity, we recommend that you do a port ping. While Ping.exe uses ICMP, you can use other tools, such as PSPing, Nmap, and telnet, to test connectivity to a specific TCP port.

For more information, see [Use port pings instead of ICMP to test Azure VM connectivity](#).

How do I prevent receiving thousands of hits from unknown IP addresses that might indicate a malicious attack to the cloud service?

Azure implements a multilayer network security to protect its platform services against distributed denial-of-service (DDoS) attacks. The Azure DDoS defense system is part of Azure's continuous monitoring process, which is continually improved through penetration testing. This DDoS defense system is designed to withstand not only attacks from the outside but also from other Azure tenants. For more information, see [Azure network security](#).

You also can create a startup task to selectively block some specific IP addresses. For more information, see [Block a specific IP address](#).

When I try to RDP to my cloud service instance, I get the message "The user account has expired."

You might get the error message "This user account has expired" when you bypass the expiration date that is

configured in your RDP settings. You can change the expiration date from the portal by following these steps:

1. Sign in to the [Azure portal](#), go to your cloud service, and select the **Remote Desktop** tab.
2. Select the **Production** or **Staging** deployment slot.
3. Change the **Expires On** date, and then save the configuration.

You now should be able to RDP to your machine.

Why is Azure Load Balancer not balancing traffic equally?

For information about how an internal load balancer works, see [Azure Load Balancer new distribution mode](#).

The distribution algorithm used is a 5-tuple (source IP, source port, destination IP, destination port, and protocol type) hash to map traffic to available servers. It provides stickiness only within a transport session. Packets in the same TCP or UDP session are directed to the same datacenter IP (DIP) instance behind the load-balanced endpoint. When the client closes and reopens the connection or starts a new session from the same source IP, the source port changes and causes the traffic to go to a different DIP endpoint.

How can I redirect incoming traffic to the default URL of my cloud service to a custom URL?

The URL Rewrite module of IIS can be used to redirect traffic that comes to the default URL for the cloud service (for example, *.cloudapp.net) to some custom name/URL. Because the URL Rewrite module is enabled on web roles by default and its rules are configured in the application's web.config, it's always available on the VM regardless of reboots/reimages. For more information, see:

- [Create rewrite rules for the URL Rewrite module](#)
- [Remove a default link](#)

How can I block/disable incoming traffic to the default URL of my cloud service?

You can prevent incoming traffic to the default URL/name of your cloud service (for example, *.cloudapp.net). Set the host header to a custom DNS name (for example, www.MyCloudService.com) under site binding configuration in the cloud service definition (*.csdef) file, as indicated:

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceDefinition name="AzureCloudServicesDemo"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition" schemaVersion="2015-04.2.6">
  <WebRole name="MyWebRole" vmsize="Small">
    <Sites>
      <Site name="Web">
        <Bindings>
          <Binding name="Endpoint1" endpointName="Endpoint1" hostHeader="www.MyCloudService.com" />
        </Bindings>
      </Site>
    </Sites>
    <Endpoints>
      <InputEndpoint name="Endpoint1" protocol="http" port="80" />
    </Endpoints>
    <ConfigurationSettings>
      <Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString" />
    </ConfigurationSettings>
  </WebRole>
</ServiceDefinition>
```

Because this host header binding is enforced through the csdef file, the service is accessible only via the custom name "www.MyCloudService.com." All incoming requests to the "*.cloudapp.net" domain always fail. If you use a custom SLB probe or an internal load balancer in the service, blocking the default URL/name of the service might interfere with the probing behavior.

How can I make sure the public-facing IP address of a cloud service never changes?

To make sure the public-facing IP address of your cloud service (also known as a VIP) never changes so that it can be customarily whitelisted by a few specific clients, we recommend that you have a reserved IP associated with it. Otherwise, the virtual IP provided by Azure is deallocated from your subscription if you delete the deployment. For successful VIP swap operation, you need individual reserved IPs for both production and staging slots. Without them, the swap operation fails. To reserve an IP address and associate it with your cloud service, see these articles:

- [Reserve the IP address of an existing cloud service](#)
- [Associate a reserved IP to a cloud service by using a service configuration file](#)

If you have more than one instance for your roles, associating RIP with your cloud service shouldn't cause any downtime. Alternatively, you can whitelist the IP range of your Azure datacenter. You can find all Azure IP ranges at the [Microsoft Download Center](#).

This file contains the IP address ranges (including compute, SQL, and storage ranges) used in Azure datacenters. An updated file is posted weekly that reflects the currently deployed ranges and any upcoming changes to the IP ranges. New ranges that appear in the file aren't used in the datacenters for at least one week. Download the new .xml file every week, and perform the necessary changes on your site to correctly identify services running in Azure. Azure ExpressRoute users might note that this file used to update the BGP advertisement of Azure space in the first week of each month.

How can I use Azure Resource Manager virtual networks with cloud services?

Cloud services can't be placed in Azure Resource Manager virtual networks. Resource Manager virtual networks and classic deployment virtual networks can be connected through peering. For more information, see [Virtual network peering](#).

How can I get the list of public IPs used by my Cloud Services?

You can use following PS script to get the list of public IPs for Cloud Services under your subscription

```
$services = Get-AzureService | Group-Object -Property ServiceName

foreach ($service in $services)
{
    "Cloud Service '$($service.Name)'"

    $deployment = Get-AzureDeployment -ServiceName $service.Name
    "VIP - " + $deployment.VirtualIPs[0].Address
    "===="
}
```

Deployment issues for Azure Cloud Services: Frequently asked questions (FAQs)

10/31/2018 • 4 minutes to read • [Edit Online](#)

This article includes frequently asked questions about deployment issues for [Microsoft Azure Cloud Services](#). You can also consult the [Cloud Services VM Size page](#) for size information.

If your Azure issue is not addressed in this article, visit the Azure forums on [MSDN and Stack Overflow](#). You can post your issue in these forums, or post to [@AzureSupport on Twitter](#). You also can submit an Azure support request. To submit a support request, on the [Azure support](#) page, select **Get support**.

Why does deploying a cloud service to the staging slot sometimes fail with a resource allocation error if there is already an existing deployment in the production slot?

If a cloud service has a deployment in either slot, the entire cloud service is pinned to a specific cluster. This means that if a deployment already exists in the production slot, a new staging deployment can only be allocated in the same cluster as the production slot.

Allocation failures occur when the cluster where your cloud service is located does not have enough physical compute resources to satisfy your deployment request.

For help mitigating such allocation failures, see [Cloud Service allocation failure: Solutions](#).

Why does scaling up or scaling out a cloud service deployment sometimes result in allocation failure?

When a cloud service is deployed, it usually gets pinned to a specific cluster. This means scaling up/out an existing cloud service must allocate new instances in the same cluster. If the cluster is nearing capacity or the desired VM size/type is not available, the request may fail.

For help mitigating such allocation failures, see [Cloud Service allocation failure: Solutions](#).

Why does deploying a cloud service into an affinity group sometimes result in allocation failure?

A new deployment to an empty cloud service can be allocated by the fabric in any cluster in that region, unless the cloud service is pinned to an affinity group. Deployments to the same affinity group will be attempted on the same cluster. If the cluster is nearing capacity, the request may fail.

For help mitigating such allocation failures, see [Cloud Service allocation failure: Solutions](#).

Why does changing VM size or adding a new VM to an existing cloud service sometimes result in allocation failure?

The clusters in a datacenter may have different configurations of machine types (e.g., A series, Av2 series, D series, Dv2 series, G series, H series, etc.). But not all the clusters would necessarily have all the kinds of VMs. For example, if you try to add a D series VM to a cloud service that is already deployed in an A series-only cluster, you will experience an allocation failure. This will also happen if you try to change VM SKU sizes (for example, switching

from an A series to a D series).

For help mitigating such allocation failures, see [Cloud Service allocation failure: Solutions](#).

To check the sizes available in your region, see [Microsoft Azure: Products available by region](#).

Why does deploying a cloud service sometime fail due to limits/quotas/constraints on my subscription or service?

Deployment of a cloud service may fail if the resources that are required to be allocated exceed the default or maximum quota allowed for your service at the region/datacenter level. For more information, see [Cloud Services limits](#).

You could also track the current usage/quota for your subscription at the portal: Azure portal=> Subscriptions=> <appropriate subscription>=> "Usage + quota".

Resource usage/consumption-related information can also be retrieved via the Azure Billing APIs. See [Azure Resource Usage API \(Preview\)](#).

How can I change the size of a deployed cloud service VM without redeploying it?

You cannot change the VM size of a deployed cloud service without redeploying it. The VM size is built into the CSDEF, which can only be updated with a redeploy.

For more information, see [How to update a cloud service](#).

Why am I not able to deploy Cloud Services through Service Management APIs or PowerShell when using Azure Resource Manager Storage account?

Since the Cloud Service is a Classic resource which is not directly compatible with the Azure Resource Manager model, you can't associate it with the Azure Resource Manager Storage accounts. Here are few options:

- Deploying through REST API.

When you deploy through Service Management REST API, you could get around the limitation by specifying a SAS URL to the blob storage, which will work with both Classic and Azure Resource Manager Storage account. Read more about the 'PackageUrl' property [here](#).

- Deploying through [Azure portal](#).

This will work from the [Azure portal](#) as the call goes through a proxy/shim which allows communication between Azure Resource Manager and Classic resources.

Why does Azure portal require me to provide a storage account for deployment?

In the classic portal, the package was uploaded to the management API layer directly, and then the API layer would temporarily put the package into an internal storage account. This process causes performance and scalability problems because the API layer was not designed to be a file upload service. In the Azure portal (Resource Manager deployment model), we have bypassed the interim step of first uploading to the API layer, resulting in faster and more reliable deployments.

As for the cost, it is very small and you can reuse the same storage account across all deployments. You can use the [storage cost calculator](#) to determine the cost to upload the service package (CSPKG), download the CSPKG, then

delete the CSPKG.

Azure Cloud Services Definition Schema (.csdef File)

11/7/2018 • 2 minutes to read • [Edit Online](#)

The service definition file defines the service model for an application. The file contains the definitions for the roles that are available to a cloud service, specifies the service endpoints, and establishes configuration settings for the service. Configuration setting values are set in the service configuration file, as described by the [Cloud Service \(classic\) Configuration Schema](#).

By default, the Azure Diagnostics configuration schema file is installed to the

`C:\Program Files\Microsoft SDKs\Windows Azure\.NET SDK\<version>\schemas` directory. Replace `<version>` with the installed version of the [Azure SDK](#).

The default extension for the service definition file is .csdef.

Basic service definition schema

The service definition file must contain one `ServiceDefinition` element. The service definition must contain at least one role (`WebRole` or `WorkerRole`) element. It can contain up to 25 roles defined in a single definition and you can mix role types. The service definition also contains the optional `NetworkTrafficRules` element which restricts which roles can communicate to specified internal endpoints. The service definition also contains the optional `LoadBalancerProbes` element which contains customer defined health probes of endpoints.

The basic format of the service definition file is as follows.

```
<ServiceDefinition name="<service-name>" topologyChangeDiscovery="<change-type>"  
      xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceDefinition" upgradeDomainCount="<number-of-upgrade-domains>" schemaVersion="<version>">  
  
  <LoadBalancerProbes>  
    ...  
  </LoadBalancerProbes>  
  
  <WebRole ...>  
    ...  
  </WebRole>  
  
  <WorkerRole ...>  
    ...  
  </WorkerRole>  
  
  <NetworkTrafficRules>  
    ...  
  </NetworkTrafficRules>  
  
</ServiceDefinition>
```

Schema definitions

The following topics describe the schema:

- [LoadBalancerProbe Schema](#)
- [WebRole Schema](#)
- [WorkerRole Schema](#)
- [NetworkTrafficRules Schema](#)

ServiceDefinition Element

The `ServiceDefinition` element is the top-level element of the service definition file.

The following table describes the attributes of the `ServiceDefinition` element.

ATTRIBUTE	DESCRIPTION
name	Required. The name of the service. The name must be unique within the service account.
topologyChangeDiscovery	Optional. Specifies the type of topology change notification. Possible values are: <ul style="list-style-type: none">- <code>Blast</code> - Sends the update as soon as possible to all role instances. If you choose option, the role should be able to handle the topology update without being restarted.- <code>UpgradeDomainWalk</code> – Sends the update to each role instance in a sequential manner after the previous instance has successfully accepted the update.
schemaVersion	Optional. Specifies the version of the service definition schema. The schema version allows Visual Studio to select the correct SDK tools to use for schema validation if more than one version of the SDK is installed side-by-side.
upgradeDomainCount	Optional. Specifies the number of upgrade domains across which roles in this service are allocated. Role instances are allocated to an upgrade domain when the service is deployed. For more information, see Update a cloud service role or deployment . You can specify up to 20 upgrade domains. If not specified, the default number of upgrade domains is 5.

Azure Cloud Services Definition LoadBalancerProbe Schema

7/12/2018 • 4 minutes to read • [Edit Online](#)

The load balancer probe is a customer defined health probe of UDP endpoints and endpoints in role instances. The `LoadBalancerProbe` is not a standalone element; it is combined with the web role or worker role in a service definition file. A `LoadBalancerProbe` can be used by more than one role.

The default extension for the service definition file is .csdef.

The function of a load balancer probe

The Azure Load Balancer is responsible for routing incoming traffic to your role instances. The load balancer determines which instances can receive traffic by regularly probing each instance in order to determine the health of that instance. The load balancer probes every instance multiple times per minute. There are two different options for providing instance health to the load balancer – the default load balancer probe, or a custom load balancer probe, which is implemented by defining the `LoadBalancerProbe` in the .csdef file.

The default load balancer probe utilizes the Guest Agent inside the virtual machine, which listens and responds with an HTTP 200 OK response only when the instance is in the Ready state (like when the instance is not in the Busy, Recycling, Stopping, etc. states). If the Guest Agent fails to respond with HTTP 200 OK, the Azure Load Balancer marks the instance as unresponsive and stops sending traffic to that instance. The Azure Load Balancer continues to ping the instance, and if the Guest Agent responds with an HTTP 200, the Azure Load Balancer sends traffic to that instance again. When using a web role your website code typically runs in w3wp.exe which is not monitored by the Azure fabric or guest agent, which means failures in w3wp.exe (eg. HTTP 500 responses) is not be reported to the guest agent and the load balancer does not know to take that instance out of rotation.

The custom load balancer probe overrides the default guest agent probe and allows you to create your own custom logic to determine the health of the role instance. The load balancer regularly probes your endpoint (every 15 seconds, by default) and the instance is be considered in rotation if it responds with a TCP ACK or HTTP 200 within the timeout period (default of 31 seconds). This can be useful to implement your own logic to remove instances from load balancer rotation, for example returning a non-200 status if the instance is above 90% CPU. For web roles using w3wp.exe, this also means you get automatic monitoring of your website, since failures in your website code return a non-200 status to the load balancer probe. If you do not define a `LoadBalancerProbe` in the .csdef file, then the default load balancer behavior (as previously described) is be used.

If you use a custom load balancer probe, you must ensure that your logic takes into consideration the `RoleEnvironment.OnStop` method. When using the default load balancer probe, the instance is taken out of rotation prior to `OnStop` being called, but a custom load balancer probe can continue to return a 200 OK during the `OnStop` event. If you are using the `OnStop` event to clean up cache, stop service, or otherwise making changes that can affect the runtime behavior of your service, then you need to ensure that your custom load balancer probe logic removes the instance from rotation.

Basic service definition schema for a load balancer probe

The basic format of a service definition file containing a load balancer probe is as follows.

```

<ServiceDefinition ...>
  <LoadBalancerProbes>
    <LoadBalancerProbe name=<load-balancer-probe-name> protocol=[http|tcp] path=<uri-for-checking-
health-status-of-vm> port=<port-number> intervalInSeconds=<interval-in-seconds> timeoutInSeconds=<
timeout-in-seconds>/>
  </LoadBalancerProbes>
</ServiceDefinition>

```

Schema elements

The `LoadBalancerProbes` element of the service definition file includes the following elements:

- [LoadBalancerProbes Element](#)
- [LoadBalancerProbe Element](#)

LoadBalancerProbes Element

The `LoadBalancerProbes` element describes the collection of load balancer probes. This element is the parent element of the [LoadBalancerProbe Element](#).

LoadBalancerProbe Element

The `LoadBalancerProbe` element defines the health probe for a model. You can define multiple load balancer probes.

The following table describes the attributes of the `LoadBalancerProbe` element:

ATTRIBUTE	TYPE	DESCRIPTION
<code>name</code>	<code>string</code>	Required. The name of the load balancer probe. The name must be unique.
<code>protocol</code>	<code>string</code>	Required. Specifies the protocol of the end point. Possible values are <code>http</code> or <code>tcp</code> . If <code>tcp</code> is specified, a received ACK is required for the probe to be successful. If <code>http</code> is specified, a 200 OK response from the specified URI is required for the probe to be successful.
<code>path</code>	<code>string</code>	The URI used for requesting health status from the VM. <code>path</code> is required if <code>protocol</code> is set to <code>http</code> . Otherwise, it is not allowed. There is no default value.
<code>port</code>	<code>integer</code>	Optional. The port for communicating the probe. This is optional for any endpoint, as the same port will then be used for the probe. You can configure a different port for their probing, as well. Possible values range from 1 to 65535, inclusive. The default value is set by the endpoint.

ATTRIBUTE	TYPE	DESCRIPTION
intervalInSeconds	integer	<p>Optional. The interval, in seconds, for how frequently to probe the endpoint for health status. Typically, the interval is slightly less than half the allocated timeout period (in seconds) which allows two full probes before taking the instance out of rotation.</p> <p>The default value is 15, the minimum value is 5.</p>
timeoutInSeconds	integer	<p>Optional. The timeout period, in seconds, applied to the probe where no response will result in stopping further traffic from being delivered to the endpoint. This value allows endpoints to be taken out of rotation faster or slower than the typical times used in Azure (which are the defaults).</p> <p>The default value is 31, the minimum value is 11.</p>

See Also

[Cloud Service \(classic\) Definition Schema](#)

Azure Cloud Services Definition WebRole Schema

7/12/2018 • 19 minutes to read • [Edit Online](#)

The Azure web role is a role that is customized for web application programming as supported by IIS 7, such as ASP.NET, PHP, Windows Communication Foundation, and FastCGI.

The default extension for the service definition file is .csdef.

Basic service definition schema for a web role

The basic format of a service definition file containing a web role is as follows.

```
<ServiceDefinition ...>
  <WebRole name=<web-role-name> vmsize=<web-role-size> enableNativeCodeExecution="[true|false]">
    <Certificates>
      <Certificate name=<certificate-name> storeLocation=<certificate-store> storeName=<store-name> />
    </Certificates>
    <ConfigurationSettings>
      <Setting name=<setting-name> />
    </ConfigurationSettings>
    <Imports>
      <Import moduleName=<import-module> />
    </Imports>
    <Endpoints>
      <InputEndpoint certificate=<certificate-name> ignoreRoleInstanceStatus="[true|false]" name=<input-endpoint-name> protocol="[http|https|tcp|udp]" localPort=<port-number> port=<port-number> loadBalancerProbe=<load-balancer-probe-name> />
        <InternalEndpoint name=<internal-endpoint-name> protocol="[http|tcp|udp|any]" port=<port-number>>
          <FixedPort port=<port-number> />
          <FixedPortRange min=<minium-port-number> max=<maximum-port-number> />
        </InternalEndpoint>
      <InstanceInputEndpoint name=<instance-input-endpoint-name> localPort=<port-number> protocol="udp|tcp">
        <AllocatePublicPortFrom>
          <FixedPortRange min=<minium-port-number> max=<maximum-port-number> />
        </AllocatePublicPortFrom>
      </InstanceInputEndpoint>
    </Endpoints>
    <LocalResources>
      <LocalStorage name=<local-store-name> cleanOnRoleRecycle="[true|false]" sizeInMB=<size-in-megabytes> />
    </LocalResources>
    <LocalStorage name=<local-store-name> cleanOnRoleRecycle="[true|false]" sizeInMB=<size-in-megabytes> />
    <Runtime executionContext="[limited|elevated]">
      <Environment>
        <Variable name=<variable-name> value=<variable-value>>
          <RoleInstanceValue xpath=<xpath-to-role-environment-settings> />
        </Variable>
      </Environment>
      <EntryPoint>
        <NetFxEntryPoint assemblyName=<name-of-assembly-containing-entrypoint> targetFrameworkVersion=".net-framework-version"/>
      </EntryPoint>
    </Runtime>
    <Sites>
      <Site name=<web-site-name>>
        <VirtualApplication name=<application-name> physicalDirectory=<directory-path> />
        <VirtualDirectory name=<directory-path> physicalDirectory=<directory-path> />
        <Bindings>
          <Binding name=<binding-name> endpointName=<endpoint-name-bound-to> hostHeader=<url-of-the-
```

```
site"/>
  </Bindings>
</Site>
</Sites>
<Startup priority=<for-internal-use-only>">
  <Task commandLine=<command-to-execute>" executionContext=[limited|elevated]" taskType="
[simple|foreground|background]">
    <Environment>
      <Variable name=<variable-name> value=<variable-value>">
        <RoleInstanceValue xpath=<xpath-to-role-environment-settings>">/
      </Variable>
    </Environment>
  </Task>
</Startup>
<Contents>
  <Content destination=<destination-folder-name>">
    <SourceDirectory path=<local-source-directory>" />
  </Content>
</Contents>
</WebRole>
</ServiceDefinition>
```

Schema elements

The service definition file includes these elements, described in detail in subsequent sections in this topic:

[WebRole](#)

[ConfigurationSettings](#)

[Setting](#)

[LocalResources](#)

[LocalStorage](#)

[Endpoints](#)

[InternalEndpoint](#)

[InstanceInputEndpoint](#)

[AllocatePublicPortFrom](#)

[FixedPort](#)

[FixedPortRange](#)

[Certificates](#)

[Certificate](#)

[Imports](#)

[Import](#)

[Runtime](#)

[Environment](#)

[Variable](#)

[RoleInstanceValue](#)

[NetFxEntryPoint](#)

[Sites](#)

[Site](#)

[VirtualApplication](#)

[VirtualApplication](#)

[Bindings](#)

[Binding](#)

[Startup](#)

[Task](#)

[Contents](#)

[Content](#)

[SourceDirectory](#)

WebRole

The `WebRole` element describes a role that is customized for web application programming, as supported by IIS 7 and ASP.NET. A service may contain zero or more web roles.

The following table describes the attributes of the `WebRole` element.

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. The name for the web role. The role's name must be unique.
enableNativeCodeExecution	boolean	Optional. The default value is <code>true</code> ; native code execution and full trust are enabled by default. Set this attribute to <code>false</code> to disable native code execution for the web role, and use Azure partial trust instead.
vmsize	string	Optional. Set this value to change the size of the virtual machine that is allotted to the role. The default value is <code>Small</code> . For more information, see Virtual Machine sizes for Cloud Services .

ConfigurationSettings

The `ConfigurationSettings` element describes the collection of configuration settings for a web role. This element is the parent of the `Setting` element.

Setting

The `Setting` element describes a name and value pair that specifies a configuration setting for an instance of a role.

The following table describes the attributes of the `Setting` element.

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. A unique name for the configuration setting.

The configuration settings for a role are name and value pairs that are declared in the service definition file and set in the service configuration file.

LocalResources

The `LocalResources` element describes the collection of local storage resources for a web role. This element is the parent of the `LocalStorage` element.

LocalStorage

The `LocalStorage` element identifies a local storage resource that provides file system space for the service at runtime. A role may define zero or more local storage resources.

NOTE

The `LocalStorage` element can appear as a child of the `WebRole` element to support compatibility with earlier versions of the Azure SDK.

The following table describes the attributes of the `LocalStorage` element.

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. A unique name for the local store.
cleanOnRoleRecycle	boolean	Optional. Indicates whether the local store should be cleaned when the role is restarted. Default value is <code>true</code> .
sizeInMb	int	Optional. The desired amount of storage space to allocate for the local store, in MB. If not specified, the default storage space allocated is 100 MB. The minimum amount of storage space that may be allocated is 1 MB. The maximum size of the local resources is dependent on the virtual machine size. For more information, see Virtual Machine sizes for Cloud Services .

The name of the directory allocated to the local storage resource corresponds to the value provided for the name attribute.

Endpoints

The `Endpoints` element describes the collection of input (external), internal, and instance input endpoints for a role. This element is the parent of the `InputEndpoint`, `InternalEndpoint`, and `InstanceInputEndpoint` elements.

Input and Internal endpoints are allocated separately. A service can have a total of 25 input, internal, and instance input endpoints which can be allocated across the 25 roles allowed in a service. For example, if you have 5 roles you can allocate 5 input endpoints per role or you can allocate 25 input endpoints to a single role or you can allocate 1 input endpoint each to 25 roles.

NOTE

Each role deployed requires one instance per role. The default provisioning for a subscription is limited to 20 cores and thus is limited to 20 instances of a role. If your application requires more instances than is provided by the default provisioning see [Billing, Subscription Management and Quota Support](#) for more information on increasing your quota.

InputEndpoint

The `InputEndpoint` element describes an external endpoint to a web role.

You can define multiple endpoints that are a combination of HTTP, HTTPS, UDP, and TCP endpoints. You can specify any port number you choose for an input endpoint, but the port numbers specified for each role in the service must be unique. For example, if you specify that a web role uses port 80 for HTTP and port 443 for HTTPS, you might then specify that a second web role uses port 8080 for HTTP and port 8043 for HTTPS.

The following table describes the attributes of the `InputEndpoint` element.

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. A unique name for the external endpoint.
protocol	string	Required. The transport protocol for the external endpoint. For a web role, possible values are <code>HTTP</code> , <code>HTTPS</code> , <code>UDP</code> , or <code>TCP</code> .
port	int	Required. The port for the external endpoint. You can specify any port number you choose, but the port numbers specified for each role in the service must be unique. Possible values range between 1 and 65535, inclusive (Azure SDK version 1.7 or higher).
certificate	string	Required for an HTTPS endpoint. The name of a certificate defined by a <code>Certificate</code> element.

ATTRIBUTE	TYPE	DESCRIPTION
localPort	int	<p>Optional. Specifies a port used for internal connections on the endpoint. The <code>localPort</code> attribute maps the external port on the endpoint to an internal port on a role. This is useful in scenarios where a role must communicate to an internal component on a port that different from the one that is exposed externally.</p> <p>If not specified, the value of <code>localPort</code> is the same as the <code>port</code> attribute. Set the value of <code>localPort</code> to "*" to automatically assign an unallocated port that is discoverable using the runtime API.</p> <p>Possible values range between 1 and 65535, inclusive (Azure SDK version 1.7 or higher).</p> <p>The <code>localPort</code> attribute is only available using the Azure SDK version 1.3 or higher.</p>
ignoreRoleInstanceStatus	boolean	<p>Optional. When the value of this attribute is set to <code>true</code>, the status of a service is ignored and the endpoint will not be removed by the load balancer. Setting this value to <code>true</code> is useful for debugging busy instances of a service. The default value is <code>false</code>. Note: An endpoint can still receive traffic even when the role is not in a Ready state.</p>
loadBalancerProbe	string	<p>Optional. The name of the load balancer probe associated with the input endpoint. For more information, see LoadBalancerProbe Schema.</p>

InternalEndpoint

The `InternalEndpoint` element describes an internal endpoint to a web role. An internal endpoint is available only to other role instances running within the service; it is not available to clients outside the service. Web roles that do not include the `sites` element can only have a single HTTP, UDP, or TCP internal endpoint.

The following table describes the attributes of the `InternalEndpoint` element.

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. A unique name for the internal endpoint.

ATTRIBUTE	TYPE	DESCRIPTION
protocol	string	<p>Required. The transport protocol for the internal endpoint. Possible values are <code>HTTP</code>, <code>TCP</code>, <code>UDP</code>, or <code>ANY</code>.</p> <p>A value of <code>ANY</code> specifies that any protocol, any port is allowed.</p>
port	int	<p>Optional. The port used for internal load balanced connections on the endpoint. A Load balanced endpoint uses two ports. The port used for the public IP address, and the port used on the private IP address. Typically these are set to the same, but you can choose to use different ports.</p> <p>Possible values range between 1 and 65535, inclusive (Azure SDK version 1.7 or higher).</p> <p>The <code>Port</code> attribute is only available using the Azure SDK version 1.3 or higher.</p>

InstanceInputEndpoint

The `InstanceInputEndpoint` element describes an instance input endpoint to a web role. An instance input endpoint is associated with a specific role instance by using port forwarding in the load balancer. Each instance input endpoint is mapped to a specific port from a range of possible ports. This element is the parent of the `AllocatePublicPortFrom` element.

The `InstanceInputEndpoint` element is only available using the Azure SDK version 1.7 or higher.

The following table describes the attributes of the `InstanceInputEndpoint` element.

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. A unique name for the endpoint.
localPort	int	Required. Specifies the internal port that all role instances will listen to in order to receive incoming traffic forwarded from the load balancer. Possible values range between 1 and 65535, inclusive.
protocol	string	Required. The transport protocol for the internal endpoint. Possible values are <code>udp</code> or <code>tcp</code> . Use <code>tcp</code> for http/https based traffic.

AllocatePublicPortFrom

The `AllocatePublicPortFrom` element describes the public port range that can be used by external customers to access each instance input endpoint. The public (VIP) port number is allocated from this range and assigned to

each individual role instance endpoint during tenant deployment and update. This element is the parent of the `FixedPortRange` element.

The `AllocatePublicPortFrom` element is only available using the Azure SDK version 1.7 or higher.

FixedPort

The `FixedPort` element specifies the port for the internal endpoint, which enables load balanced connections on the endpoint.

The `FixedPort` element is only available using the Azure SDK version 1.3 or higher.

The following table describes the attributes of the `FixedPort` element.

ATTRIBUTE	TYPE	DESCRIPTION
port	int	<p>Required. The port for the internal endpoint. This has the same effect as setting the <code>FixedPortRange</code> min and max to the same port.</p> <p>Possible values range between 1 and 65535, inclusive (Azure SDK version 1.7 or higher).</p>

FixedPortRange

The `FixedPortRange` element specifies the range of ports that are assigned to the internal endpoint or instance input endpoint, and sets the port used for load balanced connections on the endpoint.

NOTE

The `FixedPortRange` element works differently depending on the element in which it resides. When the `FixedPortRange` element is in the `InternalEndpoint` element, it opens all ports on the load balancer within the range of the min and max attributes for all virtual machines on which the role runs. When the `FixedPortRange` element is in the `InstanceInputEndpoint` element, it opens only one port within the range of the min and max attributes on each virtual machine running the role.

The `FixedPortRange` element is only available using the Azure SDK version 1.3 or higher.

The following table describes the attributes of the `FixedPortRange` element.

ATTRIBUTE	TYPE	DESCRIPTION
min	int	Required. The minimum port in the range. Possible values range between 1 and 65535, inclusive (Azure SDK version 1.7 or higher).
max	string	Required. The maximum port in the range. Possible values range between 1 and 65535, inclusive (Azure SDK version 1.7 or higher).

Certificates

The `Certificates` element describes the collection of certificates for a web role. This element is the parent of the `Certificate` element. A role may have any number of associated certificates. For more information on using the certificates element, see [Modify the Service Definition file with a certificate](#).

Certificate

The `Certificate` element describes a certificate that is associated with a web role.

The following table describes the attributes of the `Certificate` element.

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. A name for this certificate, which is used to refer to it when it is associated with an HTTPS <code>InputEndpoint</code> element.
storeLocation	string	Required. The location of the certificate store where this certificate may be found on the local machine. Possible values are <code>CurrentUser</code> and <code>LocalMachine</code> .
storeName	string	Required. The name of the certificate store where this certificate resides on the local machine. Possible values include the built-in store names <code>My</code> , <code>Root</code> , <code>CA</code> , <code>Trust</code> , <code>Disallowed</code> , <code>TrustedPeople</code> , <code>TrustedPublisher</code> , <code>AuthRoot</code> , <code>AddressBook</code> , or any custom store name. If a custom store name is specified, the store is automatically created.
permissionLevel	string	Optional. Specifies the access permissions given to the role processes. If you want only elevated processes to be able to access the private key, then specify <code>elevated</code> permission. <code>limitedOrElevated</code> permission allows all role processes to access the private key. Possible values are <code>limitedOrElevated</code> or <code>elevated</code> . The default value is <code>limitedOrElevated</code> .

Imports

The `Imports` element describes a collection of import modules for a web role that add components to the guest operating system. This element is the parent of the `Import` element. This element is optional and a role can have only one imports block.

The `Imports` element is only available using the Azure SDK version 1.3 or higher.

Import

The `Import` element specifies a module to add to the guest operating system.

The `Import` element is only available using the Azure SDK version 1.3 or higher.

The following table describes the attributes of the `Import` element.

ATTRIBUTE	TYPE	DESCRIPTION
moduleName	string	<p>Required. The name of the module to import. Valid import modules are:</p> <ul style="list-style-type: none">- <code>RemoteAccess</code>- <code>RemoteForwarder</code>- <code>Diagnostics</code> <p>The <code>RemoteAccess</code> and <code>RemoteForwarder</code> modules allow you to configure your role instance for remote desktop connections. For more information see Enable Remote Desktop Connection.</p> <p>The <code>Diagnostics</code> module allows you to collect diagnostic data for a role instance.</p>

Runtime

The `Runtime` element describes a collection of environment variable settings for a web role that control the runtime environment of the Azure host process. This element is the parent of the `Environment` element. This element is optional and a role can have only one runtime block.

The `Runtime` element is only available using the Azure SDK version 1.3 or higher.

The following table describes the attributes of the `Runtime` element:

ATTRIBUTE	TYPE	DESCRIPTION
executionContext	string	<p>Optional. Specifies the context in which the Role Process is launched. The default context is <code>limited</code>.</p> <ul style="list-style-type: none">- <code>limited</code> – The process is launched without Administrator privileges.- <code>elevated</code> – The process is launched with Administrator privileges.

Environment

The `Environment` element describes a collection of environment variable settings for a web role. This element is the parent of the `Variable` element. A role may have any number of environment variables set.

Variable

The `Variable` element specifies an environment variable to set in the guest operating.

The `Variable` element is only available using the Azure SDK version 1.3 or higher.

The following table describes the attributes of the `Variable` element:

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. The name of the environment variable to set.
value	string	Optional. The value to set for the environment variable. You must include either a value attribute or a <code>RoleInstanceValue</code> element.

RoleInstanceValue

The `RoleInstanceValue` element specifies the xPath from which to retrieve the value of the variable.

The following table describes the attributes of the `RoleInstanceValue` element.

ATTRIBUTE	TYPE	DESCRIPTION
xpath	string	<p>Optional. Location path of deployment settings for the instance. For more information, see Configuration variables with XPath.</p> <p>You must include either a value attribute or a <code>RoleInstanceValue</code> element.</p>

EntryPoint

The `EntryPoint` element specifies the entry point for a role. This element is the parent of the `NetFxEntryPoint` elements. These elements allow you to specify an application other than the default WaWorkerHost.exe to act as the role entry point.

The `EntryPoint` element is only available using the Azure SDK version 1.5 or higher.

NetFxEntryPoint

The `NetFxEntryPoint` element specifies the program to run for a role.

NOTE

The `NetFxEntryPoint` element is only available using the Azure SDK version 1.5 or higher.

The following table describes the attributes of the `NetFxEntryPoint` element.

ATTRIBUTE	TYPE	DESCRIPTION

ATTRIBUTE	TYPE	DESCRIPTION
assemblyName	string	<p>Required. The path and file name of the assembly containing the entry point. The path is relative to the folder \%ROLEROOT%\Approot (do not specify \%ROLEROOT%\Approot in <code>commandLine</code>, it is assumed).</p> <p>%ROLEROOT% is an environment variable maintained by Azure and it represents the root folder location for your role. The \%ROLEROOT%\Approot folder represents the application folder for your role.</p> <p>For HWC roles the path is always relative to the \%ROLEROOT%\Approot\bin folder.</p> <p>For full IIS and IIS Express web roles, if the assembly cannot be found relative to \%ROLEROOT%\Approot folder, the \%ROLEROOT%\Approot\bin is searched.</p> <p>This fall back behavior for full IIS is not a recommend best practice and maybe removed in future versions.</p>
targetFrameworkVersion	string	<p>Required. The version of the .NET framework on which the assembly was built. For example,</p> <pre><code>targetFrameworkVersion="v4.0".</code></pre>

Sites

The `sites` element describes a collection of websites and web applications that are hosted in a web role. This element is the parent of the `site` element. If you do not specify a `sites` element, your web role is hosted as legacy web role and you can only have one website hosted in your web role. This element is optional and a role can have only one sites block.

The `sites` element is only available using the Azure SDK version 1.3 or higher.

Site

The `site` element specifies a website or web application that is part of the web role.

The `site` element is only available using the Azure SDK version 1.3 or higher.

The following table describes the attributes of the `site` element.

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. Name of the website or application.

ATTRIBUTE	TYPE	DESCRIPTION
physicalDirectory	string	The location of the content directory for the site root. The location can be specified as an absolute path or relative to the .csdef location.

VirtualApplication

The `VirtualApplication` element defines an application in Internet Information Services (IIS) 7 is a grouping of files that delivers content or provides services over protocols, such as HTTP. When you create an application in IIS 7, the application's path becomes part of the site's URL.

The `VirtualApplication` element is only available using the Azure SDK version 1.3 or higher.

The following table describes the attributes of the `VirtualApplication` element.

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. Specifies a name to identify the virtual application.
physicalDirectory	string	Required. Specifies the path on the development machine that contains the virtual application. In the compute emulator, IIS is configured to retrieve content from this location. When deploying to the Azure, the contents of the physical directory are packaged along with the rest of the service. When the service package is deployed to Azure, IIS is configured with the location of the unpacked contents.

VirtualDirectory

The `VirtualDirectory` element specifies a directory name (also referred to as path) that you specify in IIS and map to a physical directory on a local or remote server.

The `VirtualDirectory` element is only available using the Azure SDK version 1.3 or higher.

The following table describes the attributes of the `VirtualDirectory` element.

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. Specifies a name to identify the virtual directory.

ATTRIBUTE	TYPE	DESCRIPTION
value	physicalDirectory	Required. Specifies the path on the development machine that contains the website or Virtual directory contents. In the compute emulator, IIS is configured to retrieve content from this location. When deploying to the Azure, the contents of the physical directory are packaged along with the rest of the service. When the service package is deployed to Azure, IIS is configured with the location of the unpacked contents.

Bindings

The `Bindings` element describes a collection of bindings for a website. It is the parent element of the `Binding` element. The element is required for every `Site` element. For more information about configuring endpoints, see [Enable Communication for Role Instances](#).

The `Bindings` element is only available using the Azure SDK version 1.3 or higher.

Binding

The `Binding` element specifies configuration information required for requests to communicate with a website or web application.

The `Binding` element is only available using the Azure SDK version 1.3 or higher.

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. Specifies a name to identify the binding.
endpointName	string	Required. Specifies the endpoint name to bind to.
hostHeader	string	Optional. Specifies a host name that allows you to host multiple sites, with different host names, on a single IP Address/Port number combination.

Startup

The `startup` element describes a collection of tasks that run when the role is started. This element can be the parent of the `Variable` element. For more information about using the role startup tasks, see [How to configure startup tasks](#). This element is optional and a role can have only one startup block.

The following table describes the attribute of the `Startup` element.

ATTRIBUTE	TYPE	DESCRIPTION
priority	int	For internal use only.

Task

The `Task` element specifies startup task that takes place when the role starts. Startup tasks can be used to perform tasks that prepare the role to run such install software components or run other applications. Tasks execute in the order in which they appear within the `Startup` element block.

The `Task` element is only available using the Azure SDK version 1.3 or higher.

The following table describes the attributes of the `Task` element.

ATTRIBUTE	TYPE	DESCRIPTION
commandLine	string	Required. A script, such as a CMD file, containing the commands to run. Startup command and batch files must be saved in ANSI format. File formats that set a byte-order marker at the start of the file will not process properly.
executionContext	string	Specifies the context in which the script is run. - <code>limited</code> [Default] – Run with the same privileges as the role hosting the process. - <code>elevated</code> – Run with administrator privileges.
taskType	string	Specifies the execution behavior of the command. - <code>simple</code> [Default] – System waits for the task to exit before any other tasks are launched. - <code>background</code> – System does not wait for the task to exit. - <code>foreground</code> – Similar to background, except role is not restarted until all foreground tasks exit.

Contents

The `Contents` element describes the collection of content for a web role. This element is the parent of the `Content` element.

The `Contents` element is only available using the Azure SDK version 1.5 or higher.

Content

The `Content` element defines the source location of content to be copied to the Azure virtual machine and the destination path to which it is copied.

The `Content` element is only available using the Azure SDK version 1.5 or higher.

The following table describes the attributes of the `Content` element.

ATTRIBUTE	TYPE	DESCRIPTION
-----------	------	-------------

ATTRIBUTE	TYPE	DESCRIPTION
destination	string	Required. Location on the Azure virtual machine to which the content is placed. This location is relative to the folder %ROLEROOT%\Approot.

This element is the parent element of the `SourceDirectory` element.

SourceDirectory

The `SourceDirectory` element defines the local directory from which content is copied. Use this element to specify the local contents to copy to the Azure virtual machine.

The `SourceDirectory` element is only available using the Azure SDK version 1.5 or higher.

The following table describes the attributes of the `SourceDirectory` element.

ATTRIBUTE	TYPE	DESCRIPTION
path	string	Required. Relative or absolute path of a local directory whose contents will be copied to the Azure virtual machine. Expansion of environment variables in the directory path is supported.

See Also

[Cloud Service \(classic\) Definition Schema](#)

Azure Cloud Services Definition WorkerRole Schema

7/12/2018 • 16 minutes to read • [Edit Online](#)

The Azure worker role is a role that is useful for generalized development, and may perform background processing for a web role.

The default extension for the service definition file is .csdef.

Basic service definition schema for a worker role.

The basic format of the service definition file containing a worker role is as follows.

```

<ServiceDefinition ...>
  <WorkerRole name="<worker-role-name>" vmsize="<worker-role-size>" enableNativeCodeExecution="[true|false]">
    <Certificates>
      <Certificate name="<certificate-name>" storeLocation="[CurrentUser|LocalMachine]" storeName="
[My|Root|CA|Trust|Disallow|TrustedPeople|TrustedPublisher|AuthRoot|AddressBook|<custom-store>]" />
    </Certificates>
    <ConfigurationSettings>
      <Setting name="<setting-name>" />
    </ConfigurationSettings>
    <Endpoints>
      <InputEndpoint name="<input-endpoint-name>" protocol="[http|https|tcp|udp]" localPort="<local-port-
number>" port="<port-number>" certificate="<certificate-name>" loadBalancerProbe="<load-balancer-probe-name>" />
      <InternalEndpoint name="<internal-endpoint-name" protocol="[http|tcp|udp|any]" port="<port-number>">
        <FixedPort port="<port-number>"/>
        <FixedPortRange min="<minium-port-number>" max="<maximum-port-number>"/>
      </InternalEndpoint>
      <InstanceInputEndpoint name="<instance-input-endpoint-name>" localPort="<port-number>" protocol="
[udp|tcp]">
        <AllocatePublicPortFrom>
          <FixedPortRange min="<minium-port-number>" max="<maximum-port-number>"/>
        </AllocatePublicPortFrom>
      </InstanceInputEndpoint>
    </Endpoints>
    <Imports>
      <Import moduleName="[RemoteAccess|RemoteForwarder|Diagnostics]" />
    </Imports>
    <LocalResources>
      <LocalStorage name="<local-store-name>" cleanOnRoleRecycle="[true|false]" sizeInMB="<size-in-megabytes>" />
    </LocalResources>
    <LocalStorage name="<local-store-name>" cleanOnRoleRecycle="[true|false]" sizeInMB="<size-in-megabytes>" />
  </Runtime>
  <Environment>
    <Variable name="<variable-name>" value="<variable-value>">
      <RoleInstanceValue xpath="<xpath-to-role-environment-settings>"/>
    </Variable>
  </Environment>
  <EntryPoint>
    <NetFxEntryPoint assemblyName="<name-of-assembly-containing-entrypoint>" targetFrameworkVersion="
.net-framework-version"/>
    <ProgramEntryPoint commandLine="<application>" setReadyOnProcessStart="[true|false]" "/>
  </EntryPoint>
  </Runtime>
  <Startup priority="<for-internal-use-only>">
    <Task commandLine="" executionContext="[limited|elevated]" taskType="[simple|foreground|background]">
      <Environment>
        <Variable name="<variable-name>" value="<variable-value>">
          <RoleInstanceValue xpath="<xpath-to-role-environment-settings>"/>
        </Variable>
      </Environment>
    </Task>
  </Startup>
  <Contents>
    <Content destination="<destination-folder-name>">
      <SourceDirectory path="<local-source-directory>" />
    </Content>
  </Contents>
</WorkerRole>
</ServiceDefinition>

```

Schema Elements

The service definition file includes these elements, described in detail in subsequent sections in this topic:

[WorkerRole](#)

[ConfigurationSettings](#)

[Setting](#)

[LocalResources](#)

[LocalStorage](#)

[Endpoints](#)

[InputEndpoint](#)

[InternalEndpoint](#)

[InstanceInputEndpoint](#)

[AllocatePublicPortFrom](#)

[FixedPort](#)

[FixedPortRange](#)

[Certificates](#)

[Certificate](#)

[Imports](#)

[Import](#)

[Runtime](#)

[Environment](#)

[EntryPoint](#)

[NetFxEntryPoint](#)

[ProgramEntryPoint](#)

[Variable](#)

[RoleInstanceValue](#)

[Startup](#)

[Task](#)

[Contents](#)

[Content](#)

[SourceDirectory](#)

WorkerRole

The `WorkerRole` element describes a role that is useful for generalized development, and may perform background processing for a web role. A service may contain zero or more worker roles.

The following table describes the attributes of the `WorkerRole` element.

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. The name for the worker role. The role's name must be unique.
enableNativeCodeExecution	boolean	Optional. The default value is <code>true</code> ; native code execution and full trust are enabled by default. Set this attribute to <code>false</code> to disable native code execution for the worker role, and use Azure partial trust instead.
vmsize	string	Optional. Set this value to change the size of the virtual machine that is allotted to this role. The default value is <code>Small</code> . For a list of possible virtual machine sizes and their attributes, see Virtual Machine sizes for Cloud Services .

ConfigurationSettings

The `ConfigurationSettings` element describes the collection of configuration settings for a worker role. This element is the parent of the `Setting` element.

Setting

The `Setting` element describes a name and value pair that specifies a configuration setting for an instance of a role.

The following table describes the attributes of the `Setting` element.

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. A unique name for the configuration setting.

The configuration settings for a role are name and value pairs that are declared in the service definition file and set in the service configuration file.

LocalResources

The `LocalResources` element describes the collection of local storage resources for a worker role. This element is the parent of the `LocalStorage` element.

LocalStorage

The `LocalStorage` element identifies a local storage resource that provides file system space for the service at runtime. A role may define zero or more local storage resources.

NOTE

The `LocalStorage` element can appear as a child of the `WorkerRole` element to support compatibility with earlier versions of the Azure SDK.

The following table describes the attributes of the `LocalStorage` element.

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. A unique name for the local store.
cleanOnRoleRecycle	boolean	Optional. Indicates whether the local store should be cleaned when the role is restarted. Default value is <code>true</code> .
sizeInMb	int	<p>Optional. The desired amount of storage space to allocate for the local store, in MB. If not specified, the default storage space allocated is 100 MB. The minimum amount of storage space that may be allocated is 1 MB.</p> <p>The maximum size of the local resources is dependent on the virtual machine size. For more information, see Virtual Machine sizes for Cloud Services.</p>

The name of the directory allocated to the local storage resource corresponds to the value provided for the name attribute.

Endpoints

The `Endpoints` element describes the collection of input (external), internal, and instance input endpoints for a role. This element is the parent of the `InputEndpoint`, `InternalEndpoint`, and `InstanceInputEndpoint` elements.

Input and Internal endpoints are allocated separately. A service can have a total of 25 input, internal, and instance input endpoints which can be allocated across the 25 roles allowed in a service. For example, if you have 5 roles you can allocate 5 input endpoints per role or you can allocate 25 input endpoints to a single role or you can allocate 1 input endpoint each to 25 roles.

NOTE

Each role deployed requires one instance per role. The default provisioning for a subscription is limited to 20 cores and thus is limited to 20 instances of a role. If your application requires more instances than is provided by the default provisioning see [Billing, Subscription Management and Quota Support](#) for more information on increasing your quota.

InputEndpoint

The `InputEndpoint` element describes an external endpoint to a worker role.

You can define multiple endpoints that are a combination of HTTP, HTTPS, UDP, and TCP endpoints. You can specify any port number you choose for an input endpoint, but the port numbers specified for each role in the service must be unique. For example, if you specify that a role uses port 80 for HTTP and port 443 for HTTPS, you might then specify that a second role uses port 8080 for HTTP and port 8043 for HTTPS.

The following table describes the attributes of the `InputEndpoint` element.

ATTRIBUTE	TYPE	DESCRIPTION
-----------	------	-------------

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. A unique name for the external endpoint.
protocol	string	Required. The transport protocol for the external endpoint. For a worker role, possible values are <code>HTTP</code> , <code>HTTPS</code> , <code>UDP</code> , or <code>TCP</code> .
port	int	<p>Required. The port for the external endpoint. You can specify any port number you choose, but the port numbers specified for each role in the service must be unique.</p> <p>Possible values range between 1 and 65535, inclusive (Azure SDK version 1.7 or higher).</p>
certificate	string	Required for an HTTPS endpoint. The name of a certificate defined by a <code>Certificate</code> element.
localPort	int	<p>Optional. Specifies a port used for internal connections on the endpoint. The <code>localPort</code> attribute maps the external port on the endpoint to an internal port on a role. This is useful in scenarios where a role must communicate to an internal component on a port that different from the one that is exposed externally.</p> <p>If not specified, the value of <code>localPort</code> is the same as the <code>port</code> attribute. Set the value of <code>localPort</code> to <code>"+"</code> to automatically assign an unallocated port that is discoverable using the runtime API.</p> <p>Possible values range between 1 and 65535, inclusive (Azure SDK version 1.7 or higher).</p> <p>The <code>localPort</code> attribute is only available using the Azure SDK version 1.3 or higher.</p>
ignoreRoleInstanceStatus	boolean	Optional. When the value of this attribute is set to <code>true</code> , the status of a service is ignored and the endpoint will not be removed by the load balancer. Setting this value to <code>true</code> is useful for debugging busy instances of a service. The default value is <code>false</code> . Note: An endpoint can still receive traffic even when the role is not in a Ready state.

ATTRIBUTE	TYPE	DESCRIPTION
loadBalancerProbe	string	Optional. The name of the load balancer probe associated with the input endpoint. For more information, see LoadBalancerProbe Schema .

InternalEndpoint

The `InternalEndpoint` element describes an internal endpoint to a worker role. An internal endpoint is available only to other role instances running within the service; it is not available to clients outside the service. A worker role may have up to five HTTP, UDP, or TCP internal endpoints.

The following table describes the attributes of the `InternalEndpoint` element.

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. A unique name for the internal endpoint.
protocol	string	Required. The transport protocol for the internal endpoint. Possible values are <code>HTTP</code> , <code>TCP</code> , <code>UDP</code> , or <code>ANY</code> . A value of <code>ANY</code> specifies that any protocol, any port is allowed.
port	int	Optional. The port used for internal load balanced connections on the endpoint. A Load balanced endpoint uses two ports. The port used for the public IP address, and the port used on the private IP address. Typically these are set to the same, but you can choose to use different ports. Possible values range between 1 and 65535, inclusive (Azure SDK version 1.7 or higher). The <code>Port</code> attribute is only available using the Azure SDK version 1.3 or higher.

InstanceInputEndpoint

The `InstanceInputEndpoint` element describes an instance input endpoint to a worker role. An instance input endpoint is associated with a specific role instance by using port forwarding in the load balancer. Each instance input endpoint is mapped to a specific port from a range of possible ports. This element is the parent of the `AllocatePublicPortFrom` element.

The `InstanceInputEndpoint` element is only available using the Azure SDK version 1.7 or higher.

The following table describes the attributes of the `InstanceInputEndpoint` element.

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. A unique name for the endpoint.
localPort	int	Required. Specifies the internal port that all role instances will listen to in order to receive incoming traffic forwarded from the load balancer. Possible values range between 1 and 65535, inclusive.
protocol	string	Required. The transport protocol for the internal endpoint. Possible values are <code>udp</code> or <code>tcp</code> . Use <code>tcp</code> for http/https based traffic.

AllocatePublicPortFrom

The `AllocatePublicPortFrom` element describes the public port range that can be used by external customers to access each instance input endpoint. The public (VIP) port number is allocated from this range and assigned to each individual role instance endpoint during tenant deployment and update. This element is the parent of the `FixedPortRange` element.

The `AllocatePublicPortFrom` element is only available using the Azure SDK version 1.7 or higher.

FixedPort

The `FixedPort` element specifies the port for the internal endpoint, which enables load balanced connections on the endpoint.

The `FixedPort` element is only available using the Azure SDK version 1.3 or higher.

The following table describes the attributes of the `FixedPort` element.

ATTRIBUTE	TYPE	DESCRIPTION
port	int	<p>Required. The port for the internal endpoint. This has the same effect as setting the <code>FixedPortRange</code> min and max to the same port.</p> <p>Possible values range between 1 and 65535, inclusive (Azure SDK version 1.7 or higher).</p>

FixedPortRange

The `FixedPortRange` element specifies the range of ports that are assigned to the internal endpoint or instance input endpoint, and sets the port used for load balanced connections on the endpoint.

NOTE

The `FixedPortRange` element works differently depending on the element in which it resides. When the `FixedPortRange` element is in the `InternalEndpoint` element, it opens all ports on the load balancer within the range of the min and max attributes for all virtual machines on which the role runs. When the `FixedPortRange` element is in the `InstanceInputEndpoint` element, it opens only one port within the range of the min and max attributes on each virtual machine running the role.

The `FixedPortRange` element is only available using the Azure SDK version 1.3 or higher.

The following table describes the attributes of the `FixedPortRange` element.

ATTRIBUTE	TYPE	DESCRIPTION
min	int	Required. The minimum port in the range. Possible values range between 1 and 65535, inclusive (Azure SDK version 1.7 or higher).
max	string	Required. The maximum port in the range. Possible values range between 1 and 65535, inclusive (Azure SDK version 1.7 or higher).

Certificates

The `Certificates` element describes the collection of certificates for a worker role. This element is the parent of the `Certificate` element. A role may have any number of associated certificates. For more information on using the certificates element, see [Modify the Service Definition file with a certificate](#).

Certificate

The `Certificate` element describes a certificate that is associated with a worker role.

The following table describes the attributes of the `Certificate` element.

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. A name for this certificate, which is used to refer to it when it is associated with an HTTPS <code>InputEndpoint</code> element.
storeLocation	string	Required. The location of the certificate store where this certificate may be found on the local machine. Possible values are <code>CurrentUser</code> and <code>LocalMachine</code> .

ATTRIBUTE	TYPE	DESCRIPTION
storeName	string	Required. The name of the certificate store where this certificate resides on the local machine. Possible values include the built-in store names <code>My</code> , <code>Root</code> , <code>CA</code> , <code>Trust</code> , <code>Disallowed</code> , <code>TrustedPeople</code> , <code>TrustedPublisher</code> , <code>AuthRoot</code> , <code>AddressBook</code> , or any custom store name. If a custom store name is specified, the store is automatically created.
permissionLevel	string	Optional. Specifies the access permissions given to the role processes. If you want only elevated processes to be able to access the private key, then specify <code>elevated</code> permission. <code>limitedOrElevated</code> permission allows all role processes to access the private key. Possible values are <code>limitedOrElevated</code> or <code>elevated</code> . The default value is <code>limitedOrElevated</code> .

Imports

The `Imports` element describes a collection of import modules for a worker role that add components to the guest operating system. This element is the parent of the `Import` element. This element is optional and a role can have only one runtime block.

The `Imports` element is only available using the Azure SDK version 1.3 or higher.

Import

The `Import` element specifies a module to add to the guest operating system.

The `Import` element is only available using the Azure SDK version 1.3 or higher.

The following table describes the attributes of the `Import` element.

ATTRIBUTE	TYPE	DESCRIPTION
-----------	------	-------------

ATTRIBUTE	TYPE	DESCRIPTION
moduleName	string	<p>Required. The name of the module to import. Valid import modules are:</p> <ul style="list-style-type: none"> - RemoteAccess - RemoteForwarder - Diagnostics <p>The RemoteAccess and RemoteForwarder modules allow you to configure your role instance for remote desktop connections. For more information see Enable Remote Desktop Connection.</p> <p>The Diagnostics module allows you to collect diagnostic data for a role instance</p>

Runtime

The `Runtime` element describes a collection of environment variable settings for a worker role that control the runtime environment of the Azure host process. This element is the parent of the `Environment` element. This element is optional and a role can have only one runtime block.

The `Runtime` element is only available using the Azure SDK version 1.3 or higher.

The following table describes the attributes of the `Runtime` element:

ATTRIBUTE	TYPE	DESCRIPTION
executionContext	string	<p>Optional. Specifies the context in which the Role Process is launched. The default context is <code>limited</code>.</p> <ul style="list-style-type: none"> - <code>limited</code> – The process is launched without Administrator privileges. - <code>elevated</code> – The process is launched with Administrator privileges.

Environment

The `Environment` element describes a collection of environment variable settings for a worker role. This element is the parent of the `Variable` element. A role may have any number of environment variables set.

Variable

The `Variable` element specifies an environment variable to set in the guest operating.

The `Variable` element is only available using the Azure SDK version 1.3 or higher.

The following table describes the attributes of the `Variable` element:

ATTRIBUTE	TYPE	DESCRIPTION
-----------	------	-------------

ATTRIBUTE	TYPE	DESCRIPTION
name	string	Required. The name of the environment variable to set.
value	string	Optional. The value to set for the environment variable. You must include either a value attribute or a <code>RoleInstanceValue</code> element.

RoleInstanceValue

The `RoleInstanceValue` element specifies the xPath from which to retrieve the value of the variable.

The following table describes the attributes of the `RoleInstanceValue` element.

ATTRIBUTE	TYPE	DESCRIPTION
xpath	string	<p>Optional. Location path of deployment settings for the instance. For more information, see Configuration variables with XPath.</p> <p>You must include either a value attribute or a <code>RoleInstanceValue</code> element.</p>

EntryPoint

The `EntryPoint` element specifies the entry point for a role. This element is the parent of the `NetFxEntryPoint` elements. These elements allow you to specify an application other than the default WaWorkerHost.exe to act as the role entry point.

The `EntryPoint` element is only available using the Azure SDK version 1.5 or higher.

NetFxEntryPoint

The `NetFxEntryPoint` element specifies the program to run for a role.

NOTE

The `NetFxEntryPoint` element is only available using the Azure SDK version 1.5 or higher.

The following table describes the attributes of the `NetFxEntryPoint` element.

ATTRIBUTE	TYPE	DESCRIPTION

ATTRIBUTE	TYPE	DESCRIPTION
assemblyName	string	<p>Required. The path and file name of the assembly containing the entry point. The path is relative to the folder \%ROLEROOT%\Approot (do not specify \%ROLEROOT%\Approot in <code>commandLine</code>, it is assumed). %ROLEROOT% is an environment variable maintained by Azure and it represents the root folder location for your role. The \%ROLEROOT%\Approot folder represents the application folder for your role.</p>
targetFrameworkVersion	string	<p>Required. The version of the .NET framework on which the assembly was built. For example, <code>targetFrameworkVersion="v4.0"</code>.</p>

ProgramEntryPoint

The `ProgramEntryPoint` element specifies the program to run for a role. The `ProgramEntryPoint` element allows you to specify a program entry point that is not based on a .NET assembly.

NOTE

The `ProgramEntryPoint` element is only available using the Azure SDK version 1.5 or higher.

The following table describes the attributes of the `ProgramEntryPoint` element.

ATTRIBUTE	TYPE	DESCRIPTION
commandLine	string	<p>Required. The path, file name, and any command line arguments of the program to execute. The path is relative to the folder \%ROLEROOT%\Approot (do not specify \%ROLEROOT%\Approot in <code>commandLine</code>, it is assumed). %ROLEROOT% is an environment variable maintained by Azure and it represents the root folder location for your role. The \%ROLEROOT%\Approot folder represents the application folder for your role.</p> <p>If the program ends, the role is recycled, so generally set the program to continue to run, instead of being a program that just starts up and runs a finite task.</p>

ATTRIBUTE	TYPE	DESCRIPTION
setReadyOnProcessStart	boolean	Required. Specifies whether the role instance waits for the command line program to signal it is started. This value must be set to <code>true</code> at this time. Setting the value to <code>false</code> is reserved for future use.

Startup

The `Startup` element describes a collection of tasks that run when the role is started. This element can be the parent of the `Variable` element. For more information about using the role startup tasks, see [How to configure startup tasks](#). This element is optional and a role can have only one startup block.

The following table describes the attribute of the `Startup` element.

ATTRIBUTE	TYPE	DESCRIPTION
priority	int	For internal use only.

Task

The `Task` element specifies startup task that takes place when the role starts. Startup tasks can be used to perform tasks that prepare the role to run such install software components or run other applications. Tasks execute in the order in which they appear within the `Startup` element block.

The `Task` element is only available using the Azure SDK version 1.3 or higher.

The following table describes the attributes of the `Task` element.

ATTRIBUTE	TYPE	DESCRIPTION
commandLine	string	Required. A script, such as a CMD file, containing the commands to run. Startup command and batch files must be saved in ANSI format. File formats that set a byte-order marker at the start of the file will not process properly.
executionContext	string	Specifies the context in which the script is run. <ul style="list-style-type: none"> - <code>limited</code> [Default] – Run with the same privileges as the role hosting the process. - <code>elevated</code> – Run with administrator privileges.

ATTRIBUTE	TYPE	DESCRIPTION
taskType	string	<p>Specifies the execution behavior of the command.</p> <ul style="list-style-type: none"> - <code>simple</code> [Default] – System waits for the task to exit before any other tasks are launched. - <code>background</code> – System does not wait for the task to exit. - <code>foreground</code> – Similar to background, except role is not restarted until all foreground tasks exit.

Contents

The `Contents` element describes the collection of content for a worker role. This element is the parent of the `Content` element.

The `Contents` element is only available using the Azure SDK version 1.5 or higher.

Content

The `Content` element defines the source location of content to be copied to the Azure virtual machine and the destination path to which it is copied.

The `Content` element is only available using the Azure SDK version 1.5 or higher.

The following table describes the attributes of the `Content` element.

ATTRIBUTE	TYPE	DESCRIPTION
destination	string	Required. Location on the Azure virtual machine to which the content is placed. This location is relative to the folder <code>%ROLEROOT%\Approot</code> .

This element is the parent element of the `SourceDirectory` element.

SourceDirectory

The `SourceDirectory` element defines the local directory from which content is copied. Use this element to specify the local contents to copy to the Azure virtual machine.

The `SourceDirectory` element is only available using the Azure SDK version 1.5 or higher.

The following table describes the attributes of the `SourceDirectory` element.

ATTRIBUTE	TYPE	DESCRIPTION
path	string	Required. Relative or absolute path of a local directory whose contents will be copied to the Azure virtual machine. Expansion of environment variables in the directory path is supported.

See Also

[Cloud Service \(classic\) Definition Schema](#)

Azure Cloud Services Definition NetworkTrafficRules Schema

7/12/2018 • 2 minutes to read • [Edit Online](#)

The `NetworkTrafficRules` node is an optional element in the service definition file that specifies how roles communicate with each other. It limits which roles can access the internal endpoints of the specific role. The `NetworkTrafficRules` is not a standalone element; it is combined with two or more roles in a service definition file.

The default extension for the service definition file is .csdef.

NOTE

The `NetworkTrafficRules` node is only available using the Azure SDK version 1.3 or higher.

Basic service definition schema for the network traffic rules

The basic format of a service definition file containing network traffic definitions is as follows.

```
<ServiceDefinition ...>
  <NetworkTrafficRules>
    <OnlyAllowTrafficTo>
      <Destinations>
        <RoleEndpoint endpointName="<name-of-the-endpoint>" roleName="<name-of-the-role-containing-the-endpoint>"/>
      </Destinations>
      <AllowAllTraffic/>
      <WhenSource matches="[AnyRule]">
        <FromRole roleName="<name-of-the-role-to-allow-traffic-from>"/>
      </WhenSource>
    </OnlyAllowTrafficTo>
  </NetworkTrafficRules>
</ServiceDefinition>
```

Schema Elements

The `NetworkTrafficRules` node of the service definition file includes these elements, described in detail in subsequent sections in this topic:

[NetworkTrafficRules Element](#)

[OnlyAllowTrafficTo Element](#)

[Destinations Element](#)

[RoleEndpoint Element](#)

[AllowAllTraffic Element](#)

[WhenSource Element](#)

[FromRole Element](#)

NetworkTrafficRules Element

The `NetworkTrafficRules` element specifies which roles can communicate with which endpoint on another role. A service can contain one `NetworkTrafficRules` definition.

OnlyAllowTrafficTo Element

The `OnlyAllowTrafficTo` element describes a collection of destination endpoints and the roles that can communicate with them. You can specify multiple `OnlyAllowTrafficTo` nodes.

Destinations Element

The `Destinations` element describes a collection of RoleEndpoints than can be communicated with.

RoleEndpoint Element

The `RoleEndpoint` element describes an endpoint on a role to allow communications with. You can specify multiple `RoleEndpoint` elements if there are more than one endpoint on the role.

ATTRIBUTE	TYPE	DESCRIPTION
<code>endpointName</code>	<code>string</code>	Required. The name of the endpoint to allow traffic to.
<code>roleName</code>	<code>string</code>	Required. The name of the web role to allow communication to.

AllowAllTraffic Element

The `AllowAllTraffic` element is a rule that allows all roles to communicate with the endpoints defined in the `Destinations` node.

WhenSource Element

The `WhenSource` element describes a collection of roles than can communicate with the endpoints defined in the `Destinations` node.

ATTRIBUTE	TYPE	DESCRIPTION
<code>matches</code>	<code>string</code>	Required. Specifies the rule to apply when allowing communications. The only valid value is currently <code>AnyRule</code> .

FromRole Element

The `FromRole` element specifies the roles that can communicate with the endpoints defined in the `Destinations` node. You can specify multiple `FromRole` elements if there are more than one role that can communicate with the endpoints.

ATTRIBUTE	TYPE	DESCRIPTION
<code>roleName</code>	<code>string</code>	Required. The name for role from which to allow communication.

See Also

[Cloud Service \(classic\) Definition Schema](#)

Azure Cloud Services Config Schema (.cscfg File)

7/12/2018 • 2 minutes to read • [Edit Online](#)

The service configuration file specifies the number of role instances to deploy for each role in the service, the values of any configuration settings, and the thumbprints for any certificates associated with a role. If the service is part of a Virtual Network, configuration information for the network must be provided in the service configuration file, as well as in the virtual networking configuration file. The default extension for the service configuration file is .cscfg.

The service model is described by the [Cloud Service \(classic\) Definition Schema](#).

By default, the Azure Diagnostics configuration schema file is installed to the

`C:\Program Files\Microsoft SDKs\Windows Azure\.NET SDK\<version>\schemas` directory. Replace `<version>` with the installed version of the [Azure SDK](#).

For more information about configuring roles in a service, see [What is the Cloud Service model](#).

Basic Service Configuration Schema

The basic format of the service configuration file is as follows.

```
<ServiceConfiguration serviceName="<service-name>" osFamily="<osfamily-number>" osVersion="<os-version>" schemaVersion="<schema-version>">

  <Role ...>
    ...
  </Role>

  <NetworkConfiguration>
    ...
  </NetworkConfiguration>

</ServiceConfiguration>
```

Schema definitions

The following topics describe the schema for the `ServiceConfiguration` element:

- [Role Schema](#)
- [NetworkConfiguration Schema](#)

Service Configuration Namespace

The XML namespace for the service configuration file is:

`http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfiguration`.

ServiceConfiguration Element

The `ServiceConfiguration` element is the top-level element of the service configuration file.

The following table describes the attributes of the `ServiceConfiguration` element. All attributes values are string types.

ATTRIBUTE	DESCRIPTION
serviceName	Required. The name of the cloud service. The name given here must match the name specified in the service definition file.
osFamily	<p>Optional. Specifies the Guest OS that will run on role instances in the cloud service. For information about supported Guest OS releases, see Azure Guest OS Releases and SDK Compatibility Matrix.</p> <p>If you do not include an <code>osFamily</code> value and you have not set the <code>osVersion</code> attribute to a specific Guest OS version, a default value of 1 is used.</p>
osVersion	<p>Optional. Specifies the version of the Guest OS that will run on role instances in the cloud service. For more information about Guest OS versions, see Azure Guest OS Releases and SDK Compatibility Matrix.</p> <p>You can specify that the Guest OS should be automatically upgraded to the latest version. To do this, set the value of the <code>osVersion</code> attribute to <code>*</code>. When set to <code>*</code>, the role instances are deployed using the latest version of the Guest OS for the specified OS family and will be automatically upgraded when new versions of the Guest OS are released.</p> <p>To specify a specific version manually, use the <code>Configuration String</code> from the table in the Future, Current and Transitional Guest OS Versions section of Azure Guest OS Releases and SDK Compatibility Matrix.</p> <p>The default value for the <code>osversion</code> attribute is <code>*</code>.</p>
schemaVersion	Optional. Specifies the version of the Service Configuration schema. The schema version allows Visual Studio to select the correct SDK tools to use for schema validation if more than one version of the SDK is installed side-by-side. For more information about schema and version compatibility, see Azure Guest OS Releases and SDK Compatibility Matrix

The service configuration file must contain one `ServiceConfiguration` element. The `ServiceConfiguration` element may include any number of `Role` elements and zero or 1 `NetworkConfiguration` elements.

Azure Cloud Services Config Role Schema

7/12/2018 • 2 minutes to read • [Edit Online](#)

The `Role` element of the configuration file specifies the number of role instances to deploy for each role in the service, the values of any configuration settings, and the thumbprints for any certificates associated with a role.

For more information about the Azure Service Configuration Schema, see [Cloud Service \(classic\) Configuration Schema](#). For more information about the Azure Service Definition Schema, see [Cloud Service \(classic\) Definition Schema](#).

Role Element

The following example shows the `Role` element and its child elements.

```
<ServiceConfiguration>
  <Role name=<role-name> vmName=<vm-name>>
    <Instances count=<number-of-instances>/>
    <ConfigurationSettings>
      <Setting name=<setting-name> value=<setting-value> />
    </ConfigurationSettings>
    <Certificates>
      <Certificate name=<certificate-name> thumbprint=<certificate-thumbprint> thumbprintAlgorithm=<algorithm>/>
    </Certificates>
  </Role>
</ServiceConfiguration>
```

The following table describes the attributes for the `Role` element.

ATTRIBUTE	DESCRIPTION
name	Required. Specifies the name of the role. The name must match the name provided for the role in the service definition file.
vmName	Optional. Specifies the DNS name for a Virtual Machine. The name must be 10 characters or less.

The following table describes the child elements of the `Role` element.

ELEMENT	DESCRIPTION
Instances	Required. Specifies the number of instances to deploy for the role. The number of instances is defined by an integer for the <code>count</code> attribute.
Setting	Optional. Specifies a setting name and value in a collection of settings for a role. The setting name is defined by a string for the <code>name</code> attribute and the setting value is defined by a string for the <code>value</code> attribute.

ELEMENT	DESCRIPTION
Certificate	<p>Optional. Specifies the name, thumbprint, and algorithm of a service certificate that is to be associated with the role. The certificate name is defined by a string for the <code>name</code> attribute. The certificate thumbprint is defined by a string of hexadecimal numbers containing no spaces for the <code>thumbprint</code> attribute. The hexadecimal numbers must be represented using digits and uppercase alpha characters. The certificate algorithm is defined by a string for the <code>thumbprintAlgorithm</code> attribute.</p>

See Also

[Cloud Service \(classic\) Configuration Schema](#)

Azure Cloud Services Config NetworkConfiguration Schema

7/12/2018 • 3 minutes to read • [Edit Online](#)

The `NetworkConfiguration` element of the service configuration file specifies Virtual Network and DNS values. These settings are optional for cloud services.

You can use the following resource to learn more about Virtual Networks and the associated schemas:

- [Cloud Service \(classic\) Configuration Schema](#)
- [Cloud Service \(classic\) Definition Schema](#)
- [Create a Virtual Network \(classic\)](#)

NetworkConfiguration Element

The following example shows the `NetworkConfiguration` element and its child elements.

```
<ServiceConfiguration>
  <NetworkConfiguration>
    <AccessControls>
      <AccessControl name="aclName1">
        <Rule order="<rule-order>" action="<rule-action>" remoteSubnet="<subnet-address>" description="rule-
description"/>
      </AccessControl>
    </AccessControls>
    <EndpointAcls>
      <EndpointAcl role="<role-name>" endpoint="<endpoint-name>" accessControl="<acl-name>"/>
    </EndpointAcls>
    <Dns>
      <DnsServers>
        <DnsServer name="<server-name>" IPAddress="<server-address>" />
      </DnsServers>
    </Dns>
    <VirtualNetworkSite name="<site-name>"/>
    <AddressAssignments>
      <InstanceAddress roleName="<role-name>">
        <Subnets>
          <Subnet name="<subnet-name>"/>
        </Subnets>
      </InstanceAddress>
      <ReservedIPs>
        <ReservedIP name="<reserved-ip-name>"/>
      </ReservedIPs>
    </AddressAssignments>
  </NetworkConfiguration>
</ServiceConfiguration>
```

The following table describes the child elements of the `NetworkConfiguration` element.

ELEMENT	DESCRIPTION
---------	-------------

ELEMENT	DESCRIPTION
AccessControl	<p>Optional. Specifies the rules for access to endpoints in a cloud service. The access control name is defined by a string for <code>name</code> attribute. The <code>AccessControl</code> element contains one or more <code>Rule</code> elements. More than one <code>AccessControl</code> element can be defined.</p>
Rule	<p>Optional. Specifies the action that should be taken for a specified subnet range of IP addresses. The order of the rule is defined by a string value for the <code>order</code> attribute. The lower the rule number the higher the priority. For example, rules could be specified with order numbers of 100, 200, and 300. The rule with the order number of 100 takes precedence over the rule that has an order of 200.</p> <p>The action for the rule is defined by a string for the <code>action</code> attribute. Possible values are:</p> <ul style="list-style-type: none"> - <code>permit</code> – Specifies that only packets from the specified subnet range can communicate with the endpoint. - <code>deny</code> – Specifies that access is denied to the endpoints in the specified subnet range. <p>The subnet range of IP addresses that are affected by the rule are defined by a string for the <code>remoteSubnet</code> attribute. The description for the rule is defined by a string for the <code>description</code> attribute.</p>
EndpointAcl	<p>Optional. Specifies the assignment of access control rules to an endpoint. The name of the role that contains the endpoint is defined by a string for the <code>role</code> attribute. The name of the endpoint is defined by a string for the <code>endpoint</code> attribute. The name of the set of <code>AccessControl</code> rules that should be applied to the endpoint are defined in a string for the <code>accessControl</code> attribute. More than one <code>EndpointAcl</code> elements can be defined.</p>
DnsServer	<p>Optional. Specifies the settings for a DNS server. You can specify settings for DNS servers without a Virtual Network. The name of the DNS server is defined by a string for the <code>name</code> attribute. The IP address of the DNS server is defined by a string for the <code>IPAddress</code> attribute. The IP address must be a valid IPv4 address.</p>
VirtualNetworkSite	<p>Optional. Specifies the name of the Virtual Network site in which you want deploy your cloud service. This setting does not create a Virtual Network Site. It references a site that has been previously defined in the network file for your Virtual Network. A cloud service can only be a member of one Virtual Network. If you do not specify this setting, the cloud service will not be deployed to a Virtual Network. The name of the Virtual Network site is defined by a string for the <code>name</code> attribute.</p>

ELEMENT	DESCRIPTION
InstanceAddress	<p>Optional. Specifies the association of a role to a subnet or set of subnets in the Virtual Network. When you associate a role name to an instance address, you can specify the subnets to which you want this role to be associated. The <code>InstanceAddress</code> contains a Subnets element. The name of the role that is associated with the subnet or subnets is defined by a string for the <code>roleName</code> attribute.</p>
Subnet	<p>Optional. Specifies the subnet that corresponds to the subnet name in the network configuration file. The name of the subnet is defined by a string for the <code>name</code> attribute.</p>
ReservedIP	<p>Optional. Specifies the reserved IP address that should be associated with the deployment. You must use Create Reserved IP Address to create the reserved IP address. Each deployment in a cloud service can be associated with one reserved IP address. The name of the reserved IP address is defined by a string for the <code>name</code> attribute.</p>

See Also

[Cloud Service \(classic\) Configuration Schema](#)