Azure Server - Setup Guide

2017-06-20

Orckestra, Europe
Adelgade 12, First Floor
DK-1304 Copenhagen
Phone +45 3915 7600

c1.orckestra.com

Contents

1 Introduction

C1 CMS Azure Server enables you to synchronize a website in an Azure blob storage to one or more target servers.

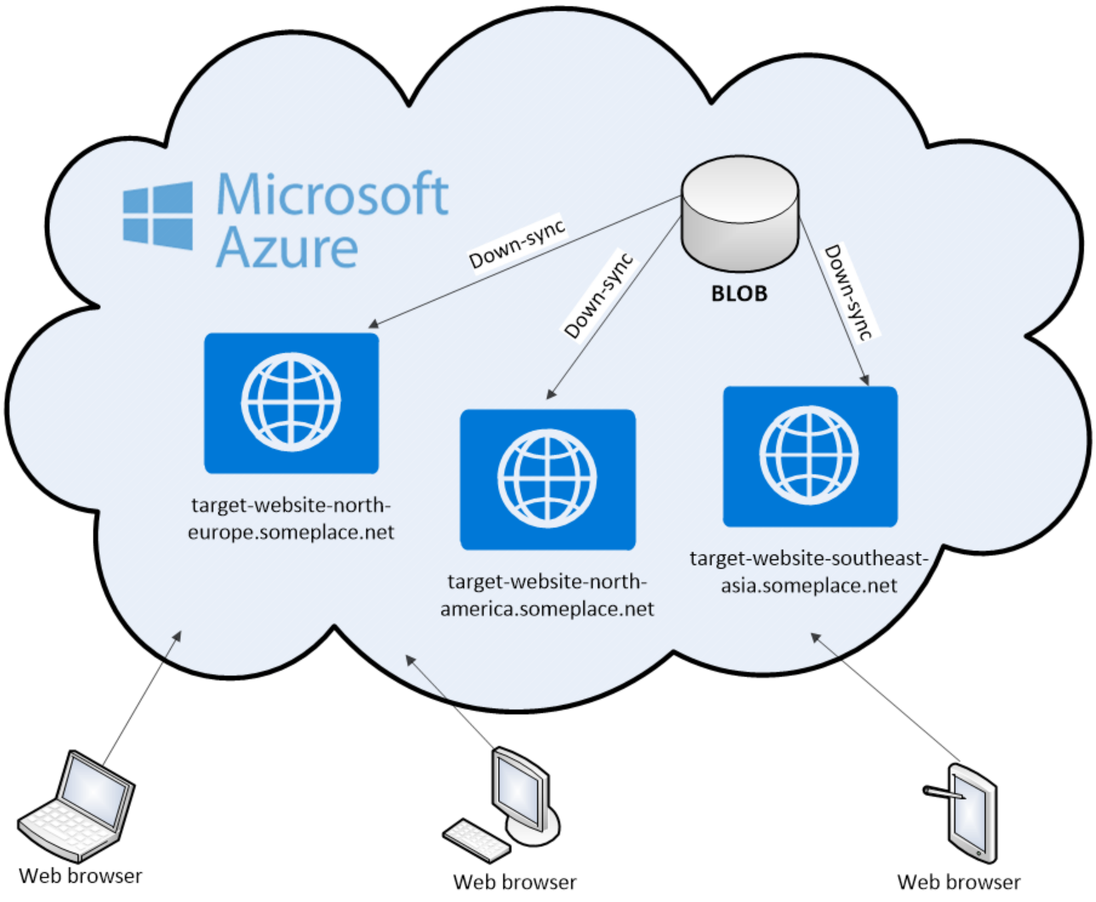
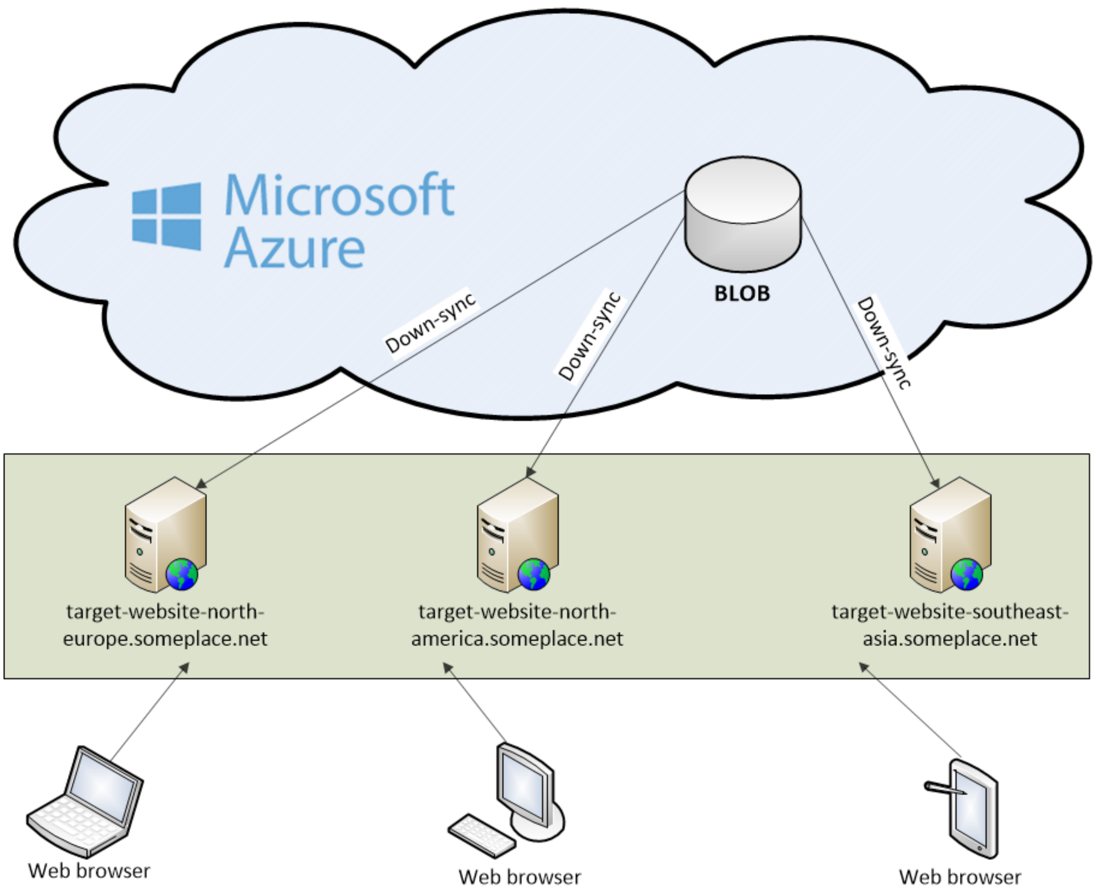

Microsoft Azure
BLOB
Down-sync
Down-sync
Down-sync
target-website-north-europe.someplace.net
target-website-north-america.someplace.net
target-website-southeast-asia.someplace.net
Web browser
Web browser
Web browser

Figure 1: Synchronizing to multiple web roles on Microsoft Azure



Microsoft Azure
BLOB
Down-sync
Down-sync
Down-sync
target-website-north-europe.someplace.net
target-website-north-america.someplace.net
target-website-southeast-asia.someplace.net
Web browser
Web browser
Web browser

Figure 2: Synchronizing to multiple Windows Servers

When combined with Azure Publisher, you have an end-to-end deployment tool that can synchronize from one environment – like a staging server – to another environment – like a scaled-out live environment.

In this guide, we assume that you already have your website in the blob storage. Please see the Azure Publisher Setup Guide for more information.

This guide primarily covers setting up one or more servers that will read from this blob storage and update hosted websites.

1.1 Who should read this guide

The guide is intended for administrators with access to the Microsoft Azure portal and / or Windows Servers, who are capable of working with blob storages as well as deploying web roles with service packages on Microsoft Azure and / or installing and running services on Windows Servers.

Optionally - if you choose to build a custom service package, you are required programming skills, particularly with Azure SDK in Visual Studio.

1.2 Getting started

You can down-sync a website from a blob storage to various types of target servers. Currently, two options are available for the target servers:

- In web roles on Microsoft Azure
- On Windows Servers

There are two deployment scenarios:

- Simple-site deployment
- Scaled-out deployment

In the **simple-site deployment**, you can run a single host with a single IIS site on Microsoft Azure much like standard web hosting.

In the **scaled-out deployment**, you can have a staging website and push changes on it via a dedicated blob storage to multiple copies of your website on Microsoft Azure/Windows Servers.

Before you begin, you need to make sure that the prerequisites are in place.

Then, you need to take 3 major steps to get your website from a blob storage to a target server:

1. Configure your website in the blob storage before deployment.
2. Deploy Azure server functionality on one or more target servers.
3. Down-sync your website from the blob storage to the target server.

Steps 1 and 3 above are identical for both target server options and for both deployment scenarios.

The Azure server deployment step is different for Azure web roles and Windows Servers as target servers and include a few steps on their own.

Deploying on **Microsoft Azure**:

1. Preparing the service package
2. Configuring the service package
3. Deploying with the service package

Deploying on a **Windows Server**:

1. Preparing a deployment container
2. Configuring Azure Blob Sync
3. Running Azure Blob Sync as a service

Besides, you need to take additional steps when scaling your web site out across multiple servers.

- Scaling out on Microsoft Azure, which should be followed by setting up Azure Traffic Manager.
- Scaling out on Windows Servers

1.3 Limitations

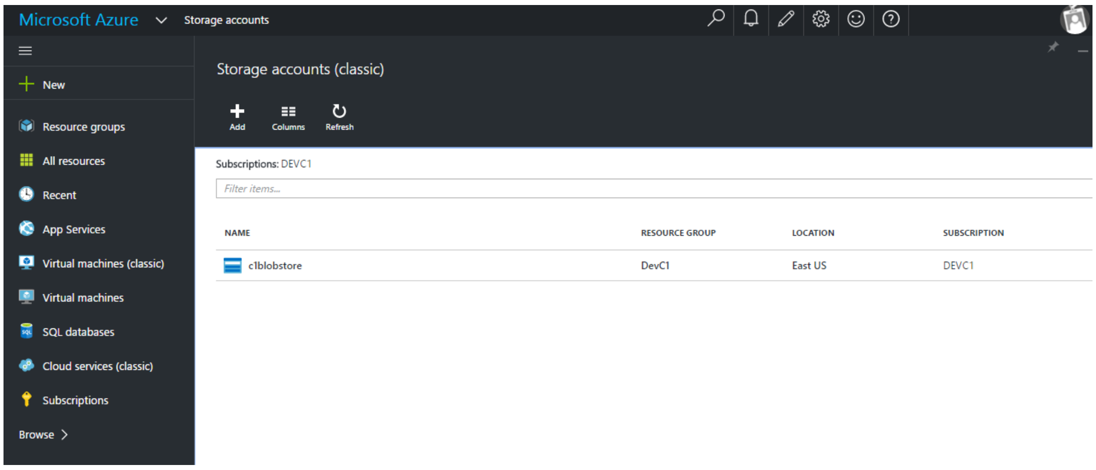
1. This is the setup where there are one or more "source" websites uploaded to a blob storage and there are one or more “target” servers. Thus, this is one-way data handling - a “source” website to a “target” server.
2. If websites on target servers are supposed to collect any data from visitors and/or store other data, you should address it on your own. C1 CMS Azure functionality does not handle these cases.

2 Prerequisites

Before you set up the target server, you need to make sure that you have a blob storage on Microsoft Azure and have a C1 CMS website uploaded to this storage.

2.1 Azure blob storage

You should have an Azure blob storage (classic) where you will upload your "source" website or changes on it.



NAME	RESOURCE GROUP	LOCATION	SUBSCRIPTION
c1blobstore	DevC1	East US	DEVC1

Figure 3: A blob storage on Microsoft Azure

If you do not have a storage on Microsoft Azure, create one:

1. Log in to your Microsoft Azure Portal.
2. Select **New** / **Storage** / **Storage account**.

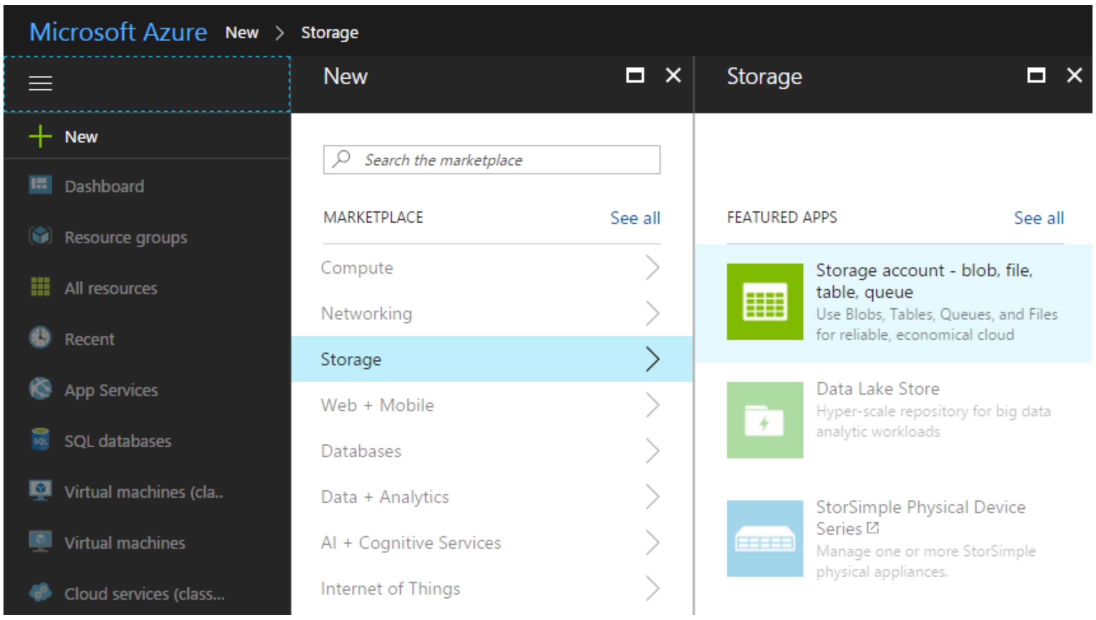
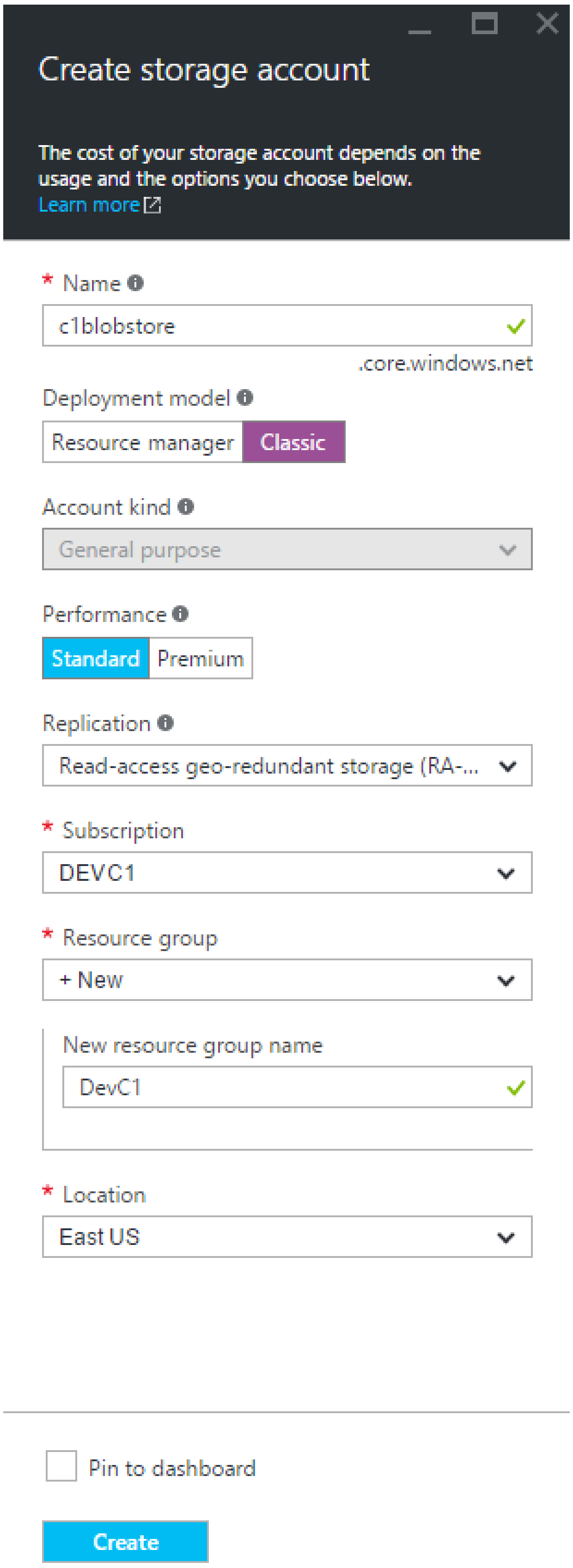


Figure 4: Creating a new storage account on Microsoft Azure

3. Select **Classic** for the Deployment model.
4. Enter a name for your storage account and select other options where necessary: **Subscription**, **Resource Group**, **Location** etc.
5. Click **Create**.



Create storage account

The cost of your storage account depends on the usage and the options you choose below.
Learn more

* Name: c1blobstore .core.windows.net

Deployment model: Resource manager [✓ Classic]

Account kind: General purpose

Performance: [✓ Standard] Premium

Replication: Read-access geo-redundant storage (RA-...

* Subscription: DEVC1

* Resource group: + New

New resource group name: DevC1

* Location: East US

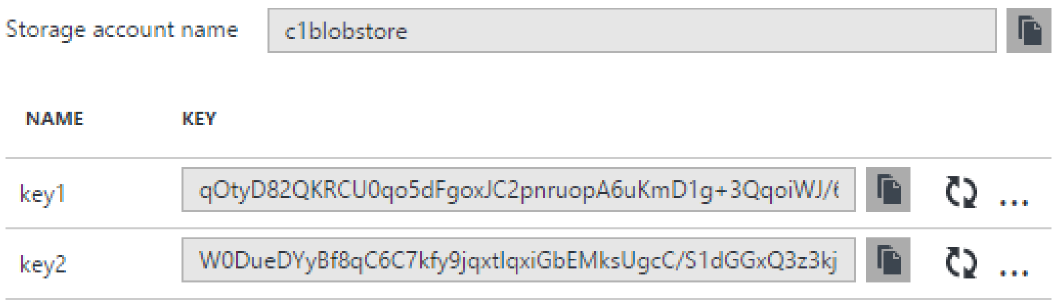
Pin to dashboard

Figure 5: Filling out configuration parameters of a blob storage

2.1.1 Blob connection information

Next, copy or make a note of, the storage account's *name* and *access key (key1)*:

- **All Resources** / **[your storage account]** / **Access Keys**



Storage account name	c1blobstore

NAME	KEY
key1	qOtyD82QKRCU0qo5dFgoxJC2pnruopA6uKmD1g+3QqoiWJ/
key2	W0DueDYyBf8qC6C7kfy9jqxtlqxiGbEMksUgcC/S1dGGxQ3z3kj

Figure 6: Getting the blob storage name and access key

When configuring Azure Server you will need to specify the storage account name and primary access key for the blob connection string.

2.2 The website uploaded to an Azure blob storage

The website synchronization from a source environment to a target environment consists of two consecutive steps:

1. The website (or changed files only) gets uploaded to a website container in a blob storage on Microsoft Azure.
2. The website (or changed files only) gets down-synced from the website container in the storage to one or more target servers within one deployment.

Before you proceed to configuring and deploying the website on target servers, you should have the website uploaded to the blob storage (Step 1).

With Azure Publisher installed and configured on your website, you can complete Step 1 by creating a website container in your blob storage and uploading the C1 CMS website there.

(Please see the “Azure Publisher Setup Guide” and “Azure Publisher User Guide” for more information.)

Alternatively, you can use CloudBerry Explorer for Azure Blob Storage or other 3rd party tools to create containers and upload files to these containers.

3 Configuring the Website

Before deploying your website on one or more target servers you need to set up its host name / port bindings and configure some other parameters.

You have to access the blob storage and edit the website configuration file there.

We recommend using CloudBerry Explorer for Azure Blob Storage to access files in the storage. (We assume that you've downloaded and installed CloudBerry Explorer and configured it to access your blob storage.)

As you have no website configuration file at this point, you should create it locally and upload it to the blob storage with the CloudBerry Explorer.

3.1 Creating the website configuration file

To create the website configuration:

1. Create an XML file naming it "WebsiteConfiguration.xml".
2. Add the following to its content:

```xml
<?xml version="1.0" encoding="utf-8"?>
<Configuration>
  <Runtime writeback="false" websiteType="composite-c1" />
  <Bindings>
    <Binding port="80" />
  </Bindings>
</Configuration>
```

Listing 1: Website configuration sample

3. Save the file.

Use one or more <Binding/> elements to bind one or more host names and / or port numbers to the website. Set up SSL support if needed.

In the sample configuration above, the host name assigned by default to a target server will be used for website access at port 80 (<Binding port="80" />), and the host name set up on the web server for the default website will be used. You can however add your own host name.

For information on parameters in WebsiteConfiguration.xml, please see "WebsiteConfiguration.xml".

For information about configuring SSL support, please see "SSL".

3.2 Uploading website configuration to the blob storage

To upload the website configuration to the blob storage:

1. Open your blob storage in CloudBerry Explorer.
2. Open the website container.
3. Create a folder named "Configuration".
4. Upload “WebsiteConfiguration.xml” to this folder.

Whenever you need to change parameters in this configuration file:

1. Download it to your local computer.
2. Edit it locally.
3. Upload it back to the blob storage overwriting the existing “WebsiteConfiguration.xml”.

Please note that if you have multiple websites on the same server, you need to configure each website creating or updating WebsiteConfiguration.xml in the corresponding website containers in the storage. (Please see "Multiple Websites" for more information.)

3.3 Disabling write-backs from live websites

In the scaled-out scenario, since you push changes to your multiple live website copies on target servers, consider preventing the live websites from writing back (up-syncing) to the website's blob storage container.

1. Edit the "WebsiteConfiguration.xml" file.
2. Set the “writeback” attribute to “false”.
3. Save the changes.

Example:

```
<Runtime writeback="false" websiteType="composite-c1" />
```

Listing 2: Disabling write-backs in the website configuration file

In the scaled-out scenario, the latest version will thus be on the staging website and you can always push the latest version to all the live website copies. Write-backs from the live websites to the website copy in the blob storage are highly undesirable because this can lead to unpredictable results, and must be disabled as suggested above.

In the simple-site scenario, changes on a single live website can be synchronized back to the website's blob storage container. This allows the latest version of the website to be copied from the container back to the target server with the live site, if the target server has been redeployed etc.

4 Deploying on Target Servers

You can deploy C1 CMS websites from the blob storage on:

- Microsoft Azure in a web role with a pre-built or custom CMS Azure service package
- a regular Windows Server with the Azure Blob Sync service

To deploy on Microsoft Azure:

1. Prepare a CMS Azure service package
2. Configure the service package
3. Deploy the web role with the service package

To deploy on a Windows Server:

1. Prepare a deployment container in the blob storage
2. Configure the Azure Blob Sync tool
3. Run Azure Blob Sync as a service

After you have deployed the server part on a target server, you should down-sync the website from the blob storage to the server.

Please note that you can have multiple websites within one deployment as well as scale out the deployment across multiple target servers (please see below).

4.1 Scaling out

You can scale out your deployment across:

- Multiple web roles on Microsoft Azure
- Multiple Windows Servers

Please note that you should set up and use Traffic Manager when scaling out on Microsoft Azure.

5 Deploying on Microsoft Azure

To prepare a target server on Microsoft Azure, you need to use a service package configured with the blob connection information.

You can use a prebuilt service package provided by Orckestra.

The service package will deploy a web role in an Azure cloud service with the required Azure server part, which will monitor the website uploaded to the blob storage for changes and down-sync these changes (or the entire website on the first run) to the web role.

The service package will automatically set up the website on the web server (IIS) and, if chosen to, install the SMTP service.

5.1 Preparing the service package

You can use the pre-built service package provided by Orckestra and available at the download page:

- Download a pre-built CMS Azure service package

The package is distributed as a pre-built solution for virtual machines of the Small (Standard A1) size with SSL support.

The service package itself (.cspkg) comes with a configuration file (.cscfg), which you will use to configure the package before deploying with it and a “dummy” certificate (.pfx) to be used with the web role by default.

5.2 Configuring the service package

Before you deploy the web role with the service package, you should configure it specifying the blob connection string and names for the deployment and setting up other parameters.

The following is the boilerplate configuration where you should replace the placeholders ({...}) with your own values.

```
<ConfigurationSettings>
  <Setting
name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"
value="DefaultEndpointsProtocol=https;AccountName={ACCOUNT_NAME};AccountKey
={KEY}" />
  <Setting name="WebRole.DisplayName" value="{WEBROLE_DISPLAY_NAME}" />
  <Setting name="WebRole.Blob.ConnectionString"
value="DefaultEndpointsProtocol=https;AccountName={ACCOUNT_NAME};AccountKey
={KEY}" />
  <Setting name="WebRole.Blob.ContainerName" value="{DEPLOYMENT_NAME}" />
  <Setting name="WebRole.Blob.ConnectionStringEncrypted" value="false" />
  <Setting name="WebRole.Blob.UpdateCheckSleepInMilliSeconds" value="5000"
/>
  <Setting name="WebRole.Blob.MaximumDownloadConcurrency" value="50" />
  <Setting name="Webrole.Blob.StatusFilesTouchRateInMinutes" value="10" />
  <Setting name="WebRole.Iis.AppPoolIdleTimeoutMinutes" value="60" />
  <Setting name="WebRole.Iis.ManageIisConfiguration" value="true" />
  <Setting name="WebRole.Iis.DeleteDefaultSites" value="true" />
  <Setting name="WebRole.SmtpServer.Install" value="false" />
</ConfigurationSettings>
```

Listing 3: Example of ServiceConfiguration.cscfg

Edit the service configuration file that comes with the package and set the following parameters replacing the corresponding placeholders:

- **{ACCOUNT_NAME}**: The name of the blob storage account used for the deployment.
- **{KEY}**: The access key for the blob storage account used for the deployment.
- **{DEPLOYMENT_NAME}**: The name of your deployment.
- **{WEBROLE_DISPLAY_NAME}**: The name to be used when referring to the web role being deployed.

The **ConnectionString** (*Account Name* + *Key*) is used in the blob connection strings for the deployment and diagnostics. (Please see “Azure blob storage”.)

It is a good practice to specify the same {ACCOUNT_NAME} and {KEY} for both WebRole.Blob.ConnectionString and Diagnostics.ConnectionString.

If you, however, choose to use different sets of account names/keys, make sure they refer to the storages in the same data center to avoid unwanted traffic expenses and / or connection issues.

The **ContainerName** uniquely identifies the deployment and also used as the name for the deployment container when the latter being created in the blob storage by the service package. The configuration file comes with this value set to “deployment”. You can keep it or change it to the one you like.

The web role’s descriptive **DisplayName** helps identify a web role where your website is deployed, and makes sense when you scale out your website across multiple web roles. The configuration file comes with this value set to “CMS Cloud Service”. You can keep it or change it to the one you like.

Please keep the default certificate settings and use the default certificate that comes with the package when deploying the role.

For these and other configuration settings in ServiceConfiguration.cscfg, please see "ServiceConfiguration.cscfg".

5.3 Deploying with the service package

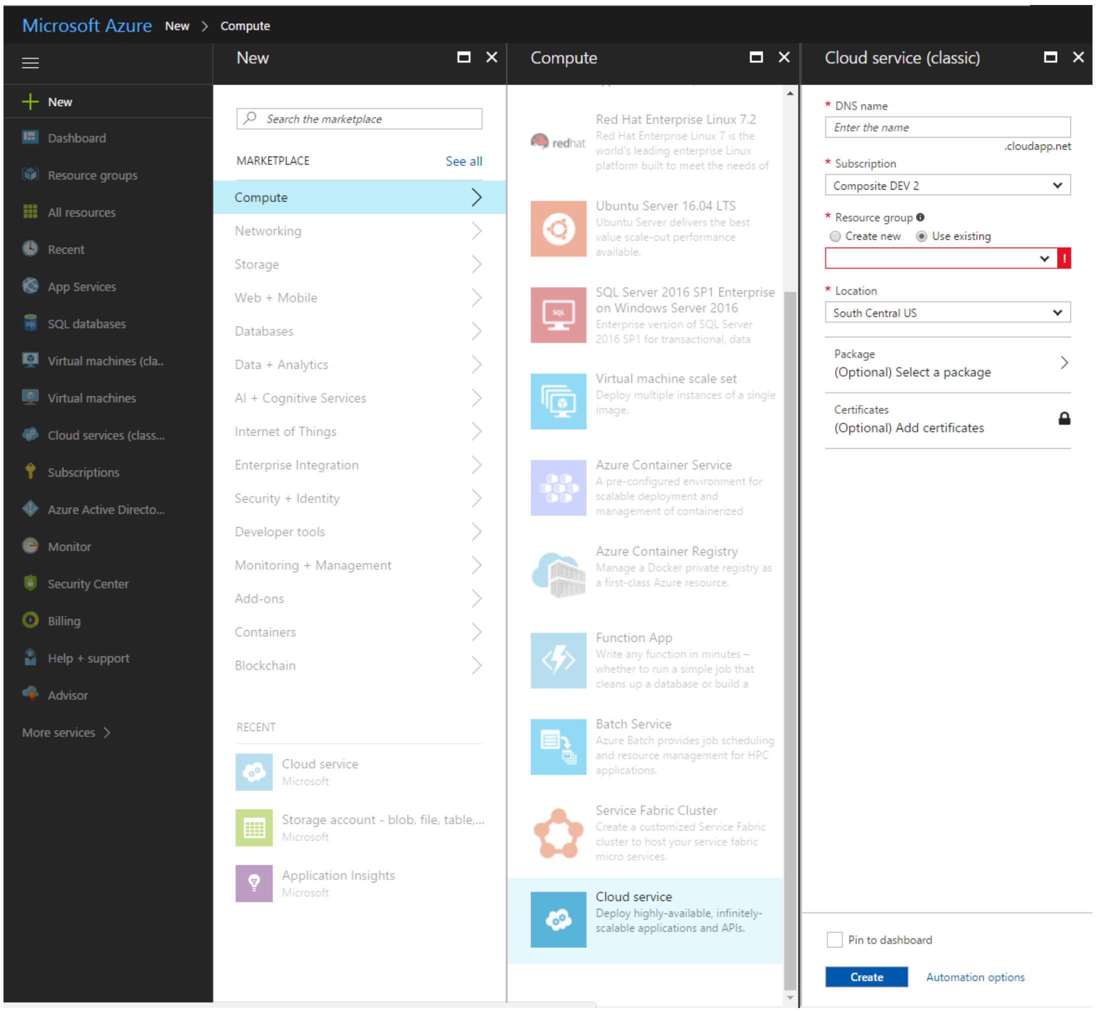
To deploy the web role with the Azure server part using the service package, you need to have a cloud service created on Microsoft Azure and the certificate (specified in the configuration file) uploaded to this service.

Please note that you can always do the following 3 steps (create a service, upload a certificate and deploy the web role) immediately as a combined step while creating a service. Here, we will consider each step separately.

To create a cloud service:

1. Log in to your Microsoft Azure Portal.
2. Create a cloud service (**New** / **Compute** / **Cloud service**).
3. Specify:
    a. the **DNS name**
    b. the **Subscription**
    c. the **Resource group**
    d. the **Location**
4. Click **Create**.

It will take a few moments to create the service.



* DNS name: Enter the name .cloudapp.net
* Subscription: Composite DEV 2
* Resource group: Create new [✓ Use existing]
* Location: South Central US

Package: (Optional) Select a package

Certificates: (Optional) Add certificates

Figure 7: Creating a cloud service on Microsoft Azure

Next, upload the certificate that comes with the package.

1. Open the cloud service you’ve just created (**All Resources** / [your cloud service]).
2. Click “Certificates” below “Settings”.

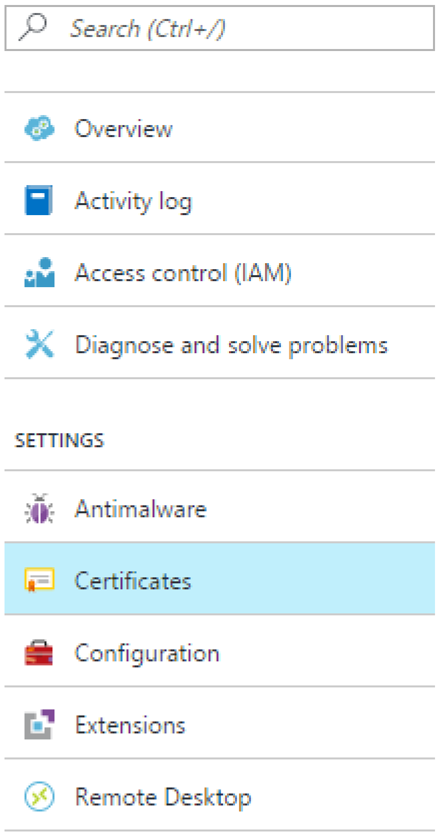
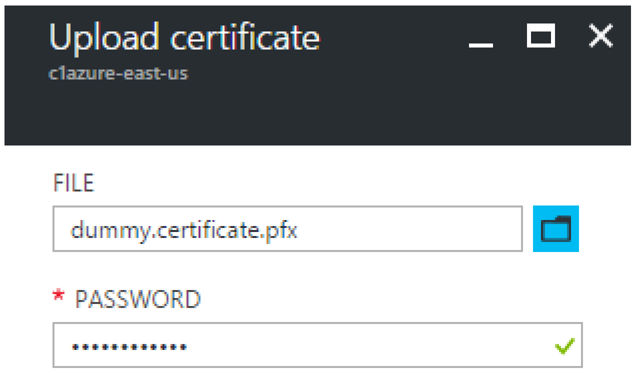


Figure 8: Selecting a cloud service's "Certificates".

3. Click "Upload" to upload the default certificate.
4. Browse to, and select, the default certificate that comes with the package ("dummy.certificate.pfx").
5. Specify the password written in the file "dummy.certificate.password.txt".
6. Click "Upload" down below.


Upload certificate
c1azure-east-us

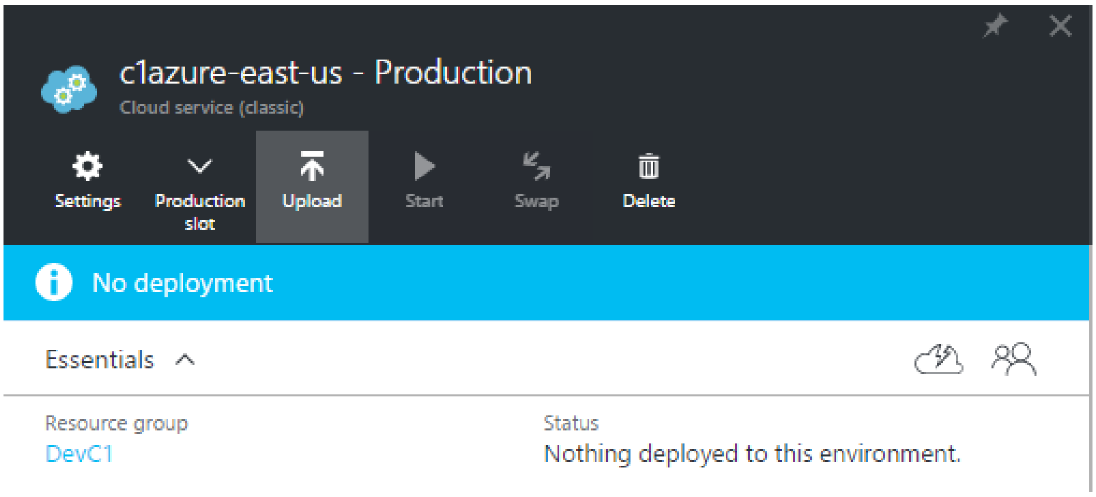
FILE
dummy.certificate.pfx

* PASSWORD
••••••••••••

Figure 9: Uploading a certificate to a cloud service

Once the certificate gets uploaded, go on and upload the service package and its configuration to the cloud service to deploy the web role:

1. Open the cloud service you've just created (**All Resources** / [your cloud service]).


c1azure-east-us - Production

Cloud service (classic)

No deployment

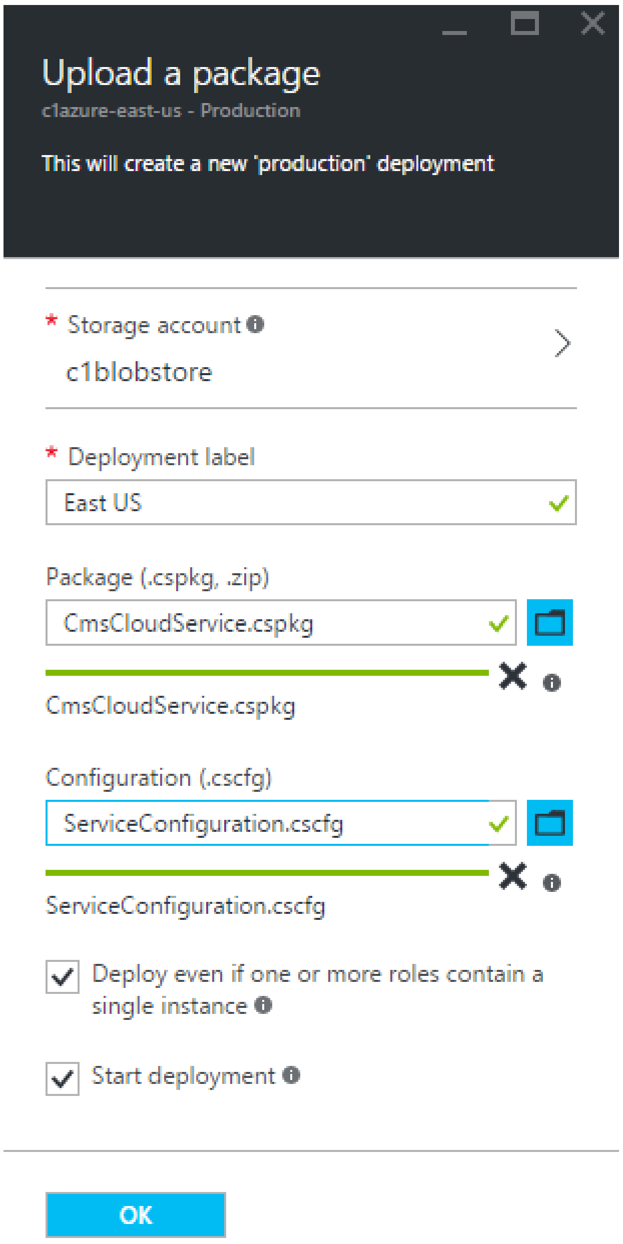
Essentials

Resource group
DevC1

Status
Nothing deployed to this environment.

Figure 10: Opening a cloud service

2. If necessary, select between a production and staging deployment (**Production slot** or **Staging slot**).
3. Click **Upload** to upload your service package and configuration to the cloud service.


Upload a package

c1azure-east-us - Production

This will create a new 'production' deployment

* Storage account

c1blobstore

* Deployment label

East US

Package (.cspkg, .zip)

CmsCloudService.cspkg

CmsCloudService.cspkg

Configuration (.cscfg)

ServiceConfiguration.cscfg

ServiceConfiguration.cscfg

✓ Deploy even if one or more roles contain a single instance

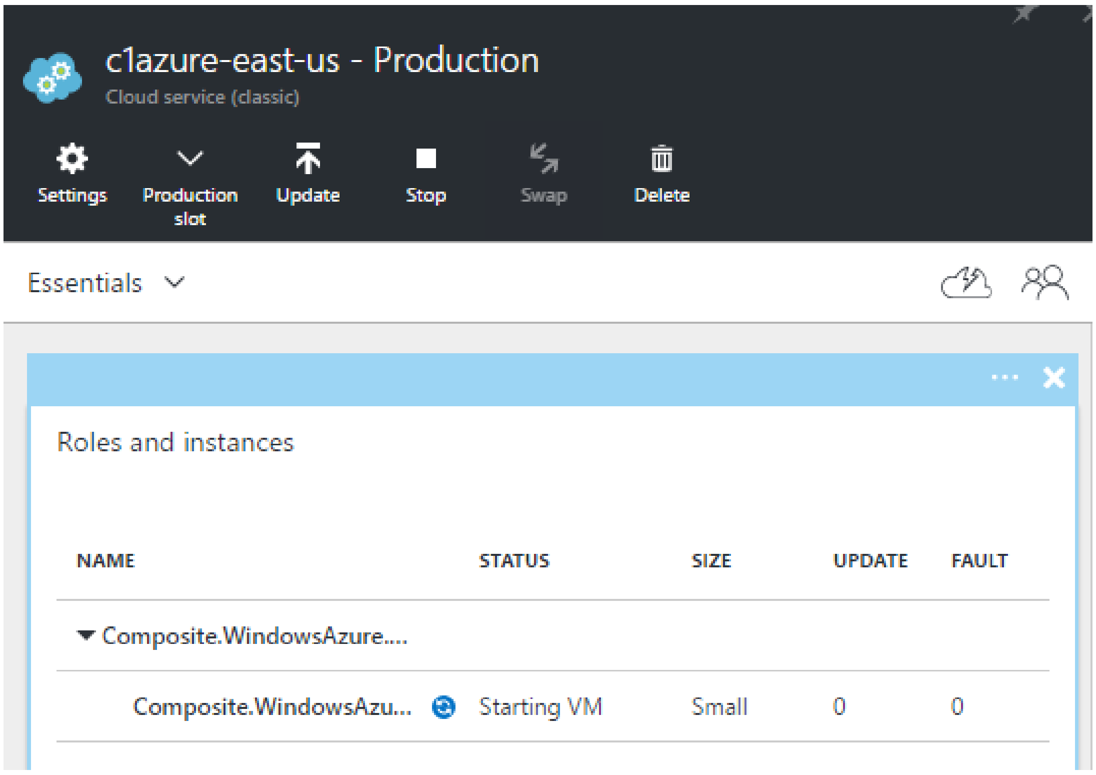
✓ Start deployment

Figure 11: Uploading the service package

4. In the “Upload a package” form, set these parameters:
    a. Select a storage account (it should be of the “Classic” deployment model)
    b. Specify the “Deployment label”
    c. Browse for, and select, the service package and configuration you've prepared earlier for the “Package” and “Configuration” parameters. This will upload your package and configuration to the blob storage immediately.
5. Also, make sure these options are enabled:
    a. “Deploy even if one or more roles contain a single instance.”
    b. “Start deployment.”
6. Click the “OK” button.

This will start deploying the web role. Please note that the deployment might take some time.

You can monitor the status and progress of the deployment in the "Role and instances" box.



NAME	STATUS	SIZE	UPDATE	FAULT
Composite.WindowsAzure....				
Composite.WindowsAzu...	Starting VM	Small	0	0

Figure 12:

You can also check a trace log for a specific message that signifies that the web role has been deployed properly and now is running. You can use Visual Studio or other 3rd -party tools such as Azure Management Studio for monitoring diagnostics data. The message will read something like: “CmsCloudService: Ready... ; TraceSource 'WaIISHost.exe' event”.

For a scaled-out deployment, you should repeat the steps above for as many deployments as you need. Please see below “Scaling out on Microsoft Azure”.

Now that you've deployed the web role(s) in the cloud, you are ready for the final step - having your website down-synced to the web role(s).

5.4 Scaling out on Microsoft Azure

You can scale out your website for load balancing and geo-targeting purposes across a number of data centers on Microsoft Azure.

For the scaled-out deployment, you need to repeat the deployment of the web role with the service package but use ServiceConfiguration.cscfg with a different value for the “DisplayName” setting.

1. Configure the service package specifying a unique web role display name {WEBROLE_DISPLAY_NAME} in the service configuration file.
2. Create a new cloud service, upload the default certificate and deploy the web role in it with the configuration from Step 1.
3. Repeat steps 1-2 for as many deployments as you need

Please note that keeping the same display name for the role will not affect the deployment itself. It helps distinguishing one copy from another within the scaled-out deployment, for example in the Azure Publisher.

Please also note that when scaling out, you must keep the same deployment name.

C1 CMS

It makes sense to prepare in advance as many service configuration files as you need for your scaled-out deployment on Microsoft Azure. The service package should be reused.

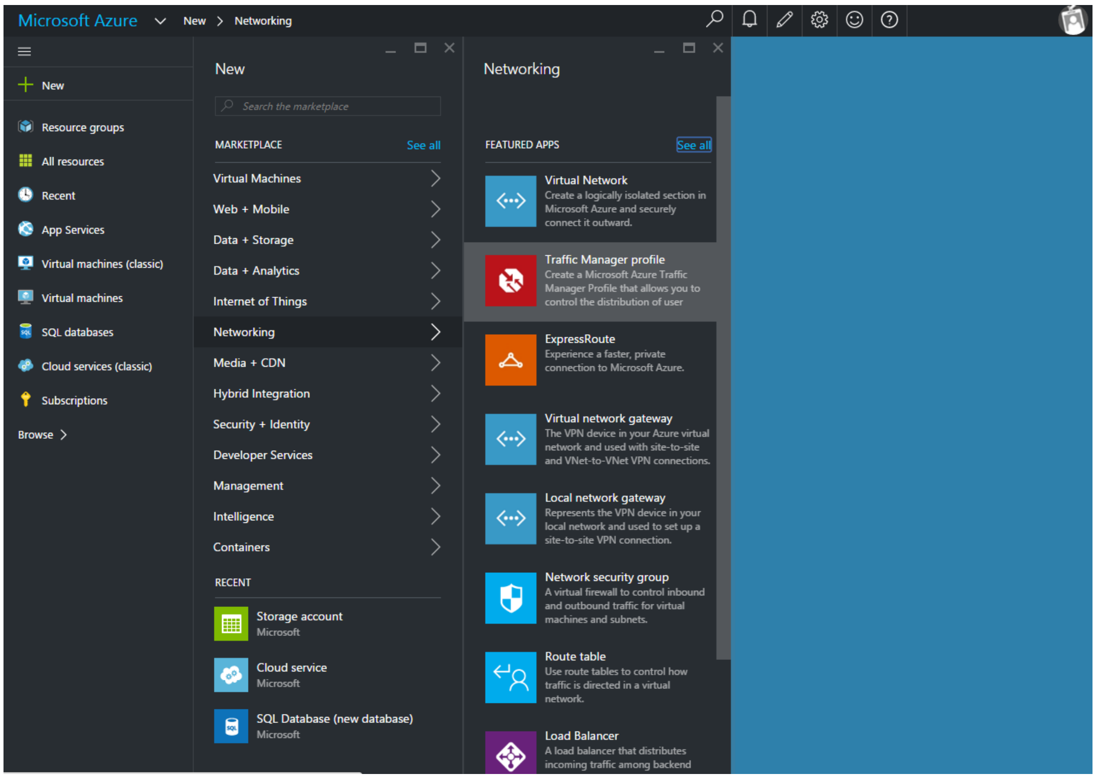
Important: Make sure to disable write-backs on the websites in the cloud when scaling out by setting the writeback parameter to 'false' in the website configuration file (configuration/Runtime/@writeback). For more information, please refer to ”Disabling write-backs from live websites”.

5.5 Setting up Azure Traffic Manager

NOTE: Take this step only for the scaled-out deployment.

You should set up a Traffic Manager on Microsoft Azure to load balance incoming traffic to your website geographically.

1. In the Microsoft Azure portal, click **New** / **Networkig** / **Traffic Manager profiles**.



Virtual Network
Create a logically isolated section in Microsoft Azure and securely connect it outward.

Traffic Manager profile
Create a Microsoft Azure Traffic Manager Profile that allows you to control the distribution of user

ExpressRoute
Experience a faster, private connection to Microsoft Azure.

Virtual network gateway
The VPN device in your Azure virtual network and used with site-to-site and VNet-to-VNet VPN connections.

Local network gateway
Represents the VPN device in your local network and used to set up a site-to-site VPN connection.

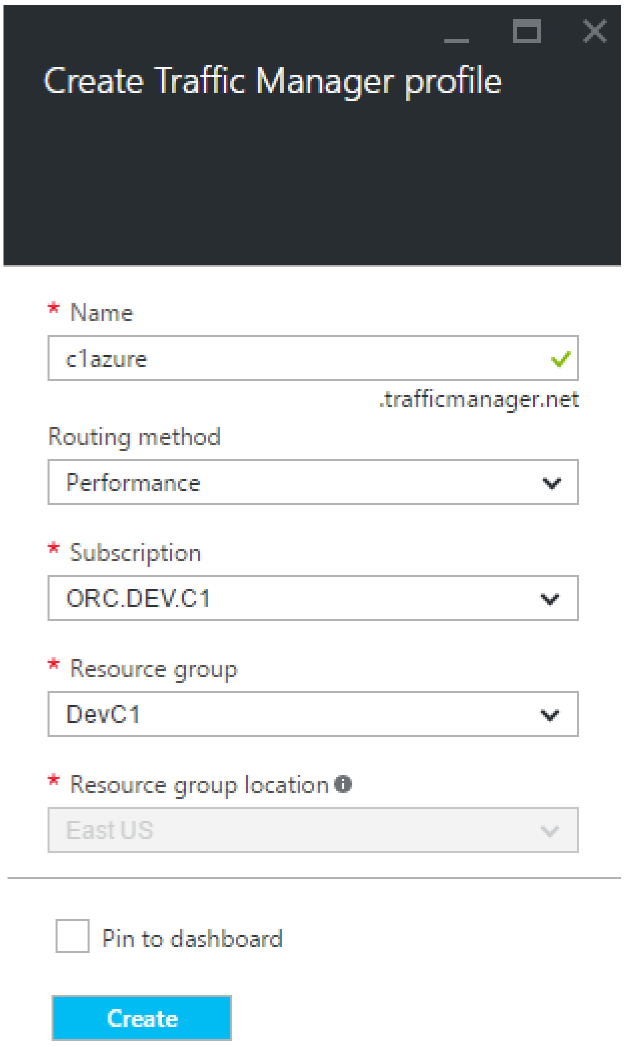
Network security group
A virtual firewall to control inbound and outbound traffic for virtual machines and subnets.

Route table
Use route tables to control how traffic is directed in a virtual network.

Load Balancer
A load balancer that distributes incoming traffic among backend

Figure 13: Creating a Traffic Manager

2. Specify the **Name** and select "Performance" for the **Routing method**.
3. Also, select the **Subscription** and **Resource group** if needed.
4. Click **Create**.


Create Traffic Manager profile

* Name
c1azure
.trafficmanager.net

Routing method
Performance

* Subscription
ORC.DEV.C1

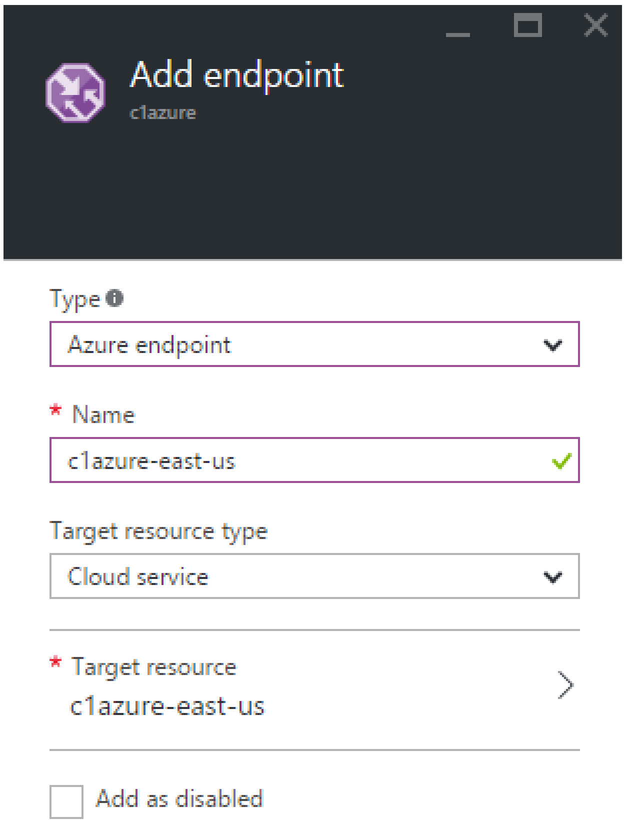
* Resource group
DevC1

* Resource group location
East US

Figure 14: Setting parameters for the Traffic Manager

Once it has been created, add endpoints (individual deployments in your scaled-out deployment):

1. Open it (**All Resources** / [your traffic manager]).
2. Click on the **Endpoints** box to open the list of endpoints (empty for now).
3. Click **Add** to add an endpoint.
4. Set the parameters:
    a. Set the **Type** to “Azure endpoint”
    b. come up with the **Name**
    c. Set the **Target resource type** to “Cloud service”
    d. Set the **Target resource** to the cloud service with the web role deployed as part of your scaled-out deployment.
5. Click **OK**.
6. Repeat Steps 3-5 for as many individual deployments as you have.



Type
Azure endpoint

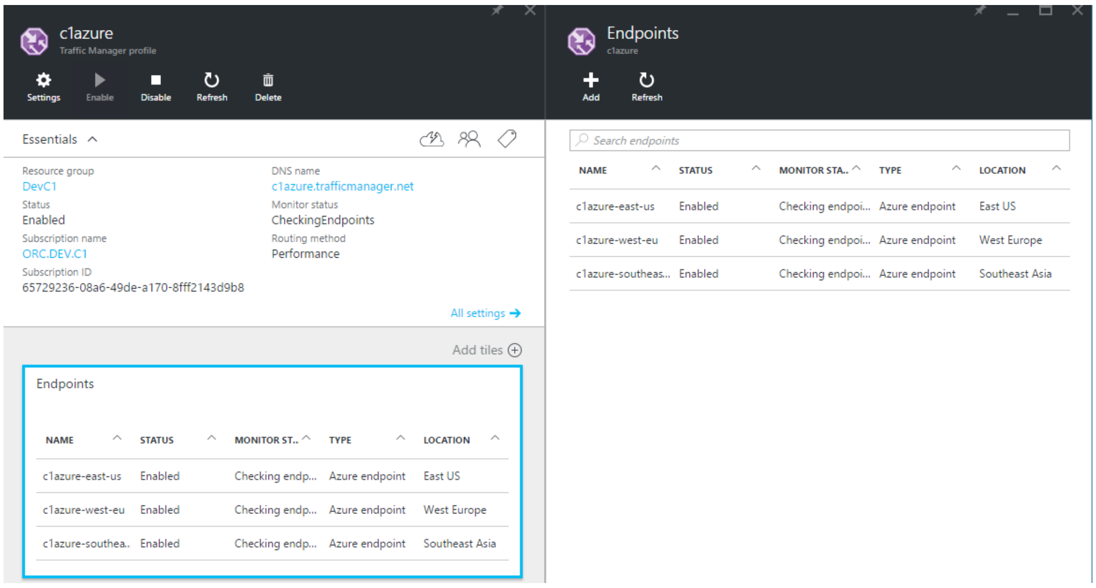
* Name
c1azure-east-us

Target resource type
Cloud service

* Target resource
c1azure-east-us

Add as disabled

Figure 15: Adding endpoints



Resource group: DevC1
DNS name: c1azure.trafficmanager.net
Status: Enabled
Monitor status: CheckingEndpoints
Subscription name: ORC.DEV.C1
Routing method: Performance
Subscription ID: 65729236-08a6-49de-a170-8fff2143d9b8

Endpoints

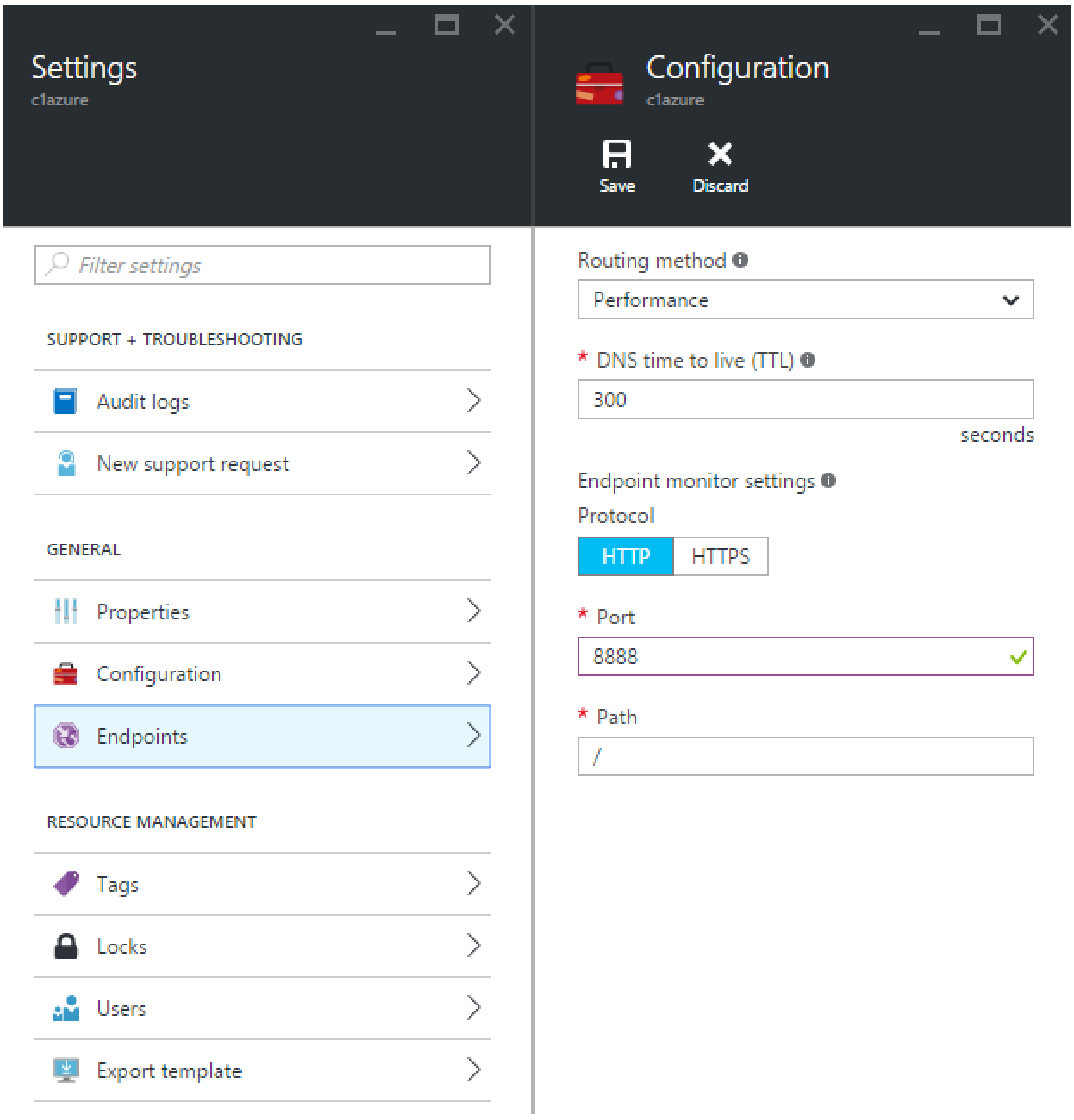
NAME	STATUS	MONITOR ST..	TYPE	LOCATION
c1azure-east-us	Enabled	Checking endp...	Azure endpoint	East US
c1azure-west-eu	Enabled	Checking endp...	Azure endpoint	West Europe
c1azure-southea..	Enabled	Checking endp...	Azure endpoint	Southeast Asia

NAME	STATUS	MONITOR STA..	TYPE	LOCATION
c1azure-east-us	Enabled	Checking endpoi...	Azure endpoint	East US
c1azure-west-eu	Enabled	Checking endpoi...	Azure endpoint	West Europe
c1azure-southeas...	Enabled	Checking endpoi...	Azure endpoint	Southeast Asia

Figure 16: Endpoints added to the traffic manager

Finally:

1. Open the traffic manager’s configuration (**Settings** / **Configuration**)
2. Change the port number from 80 to 8888.
3. Save the changes.



Routing method: Performance

* DNS time to live (TTL): 300 seconds

Endpoint monitor settings

Protocol: [✓ HTTP] HTTPS

* Port: 8888

* Path: /

Figure 17: Changing the port for the traffic manager

6 Deploying on a Windows Server

When deploying a website on a Windows Server, you should normally install and start the Azure Blob Sync service, which serves here as a target server part, equivalent to the Azure server part deployed on Microsoft Azure.

Before you run the service, you need to make sure that there is a dedicated container for the deployment in the blob storage, and configure the service specifying the blob connection string and other parameters.

6.1 Preparing a deployment container

You should prepare a deployment container in the blob storage for your Windows Server deployment with the C1AzureBlobSync tool.

1. Access your blob storage on Microsoft Azure:
2. Create a "deployment" container within the blob storage where you have already uploaded your website.

To access the storage, you can use a tool of your choice (for example, CloudBerry Explorer for Azure Blob Storage) or access it directly on Microsoft Azure Portal.

Please note that since website has been uploaded to a dedicated "website" container, you need to use a different container for the deployment.

6.2 Configuring Azure Blob Sync

Before you install and start the C1AzureBlobSync service, you need to configure it.

1. Locate and edit Composite.WindowsAzure.C1AzureBlobSync.Service.exe.config.
2. Set the **roleConfig** parameter to the path to the role’s configuration file (*role.config*).
3. Set the **environmentConfig** parameter respectively to the path to the environment’s configuration (*env.config*).
4. Save the changes.

Please note that if these configuration files are in the same folder as the service configuration file, you can only specify their relative paths.

```
<configuration>
  <!-- skipped -->
  <appSettings>
    <add key="environmentConfig" value=" env.config" />
    <add key="roleConfig" value=" role.config" />
  <!-- skipped -->
  </appSettings>
<!-- skipped -->
</configuration>
```

Listing 4: Composite.WindowsAzure.C1AzureBlobSync.Service.exe.config

Now you need to configure the “role” and “environment” in corresponding files.

6.2.1 Configuring the role.config

1. Locate and edit role.config.
2. Set the parameters with the following placeholders in the sample below:
   - **{DISPLAY_NAME}**: The name to be used when referring to the website on a specific Windows Server.
   - **{DEPLOYMENT_NAME}**: The name of your deployment. It must be the same as the name of the deployment container you've created earlier (see above).

C1 CMS

   - **{ACCOUNT_NAME}**: The name of the blob storage account used for the deployment.
   - **{KEY}**: The primary access key for the blob storage account used for the deployment.
3. Save the changes.

```
<ServiceConfiguration
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfigura
tion">
  <Role name="Composite.WindowsAzure.WebRole.SimpleBoot">
    <ConfigurationSettings>
      <Setting name="Composite.WindowsAzure.WebRole.DisplayName"
value="{DISPLAY_NAME}" />
      <Setting name="Composite.WindowsAzure.WebRole.DeploymentName"
value="{DEPLOYMENT_NAME}" />
      <Setting
name="Composite.WindowsAzure.WebRole.Storage.ConnectionString"

value="DefaultEndpointsProtocol=https;AccountName={ACCOUNT_NAME};AccountKey
={ACCOUNT_KEY}" />
      <Setting
name="Composite.WindowsAzure.WebRole.Iis.AppPoolIdleTimeoutMinutes"
value="60" />
      <Setting
name="Composite.WindowsAzure.WebRole.Manager.SleepTimerInMilliSeconds"
value="5000" />
      <Setting name="Composite.WindowsAzure.WebRole.Manager.RollingUpdate"
value="true" />
      <Setting
name="Composite.WindowsAzure.WebRole.Manager.RollingUpdateIntervalInSeconds
" value="30" />
      <!-- Maximum supported value is 60 -->
    </ConfigurationSettings>
    <Certificates></Certificates>
  </Role>
</ServiceConfiguration>
```

Listing 5: role.config

For more information on the role configuration parameters, please see "Role Configuration" for more information on this configuration.

6.2.2 Configuring the env.config

1. Locate and edit env.config.
2. Set the parameters with the following placeholders in the sample below:
   - **{DEPLOYMENT_ID}**
   - **{INSTANCE_ID}**
   - **{INSTANCE_NAME}**
3. Save the changes.

```
<Settings>
  <add name="Environment.IsThisTheLowestInstance" value="true" />
  <add name="Environment.NumberOfInstances" value="1" />
  <add name="Environment.DeploymentId" value="{DEPLOYMENT_ID}" />
  <add name="Environment.RoleInstanceId" value="{INSTANCE_ID}" />
  <add name="Environment.RoleInstanceName" value="{INSTANCE_NAME}" />
  <add name="Environment.WebsitesPath" value="c:\www" />
  <add name="Environment.RootPath" value="c:\AzureBlobSync" />
  <add name="Composite.WindowsAzure.WebRole.Iis.DeleteDefaultSites"
value="false" />
  <add name="Composite.WindowsAzure.WebRole.Iis.ManageIisConfiguration"
value="true" />
  <add name="Composite.WindowsAzure.WebRole.SmtpServer.Install"
value="true" />
</Settings>
```

Listing 6: env.config

The parameters you are supposed to set above are used internally but should be unique:

The deployment ID should be unique for the deployment and the role instance's ID and name – unique for a single role within this deployment, which makes more sense in the scaled-out deployment.

By default, the tool would clear all the websites from the IIS before setting up a new one. When deploying locally, it is important to turn off this setting ('false'):

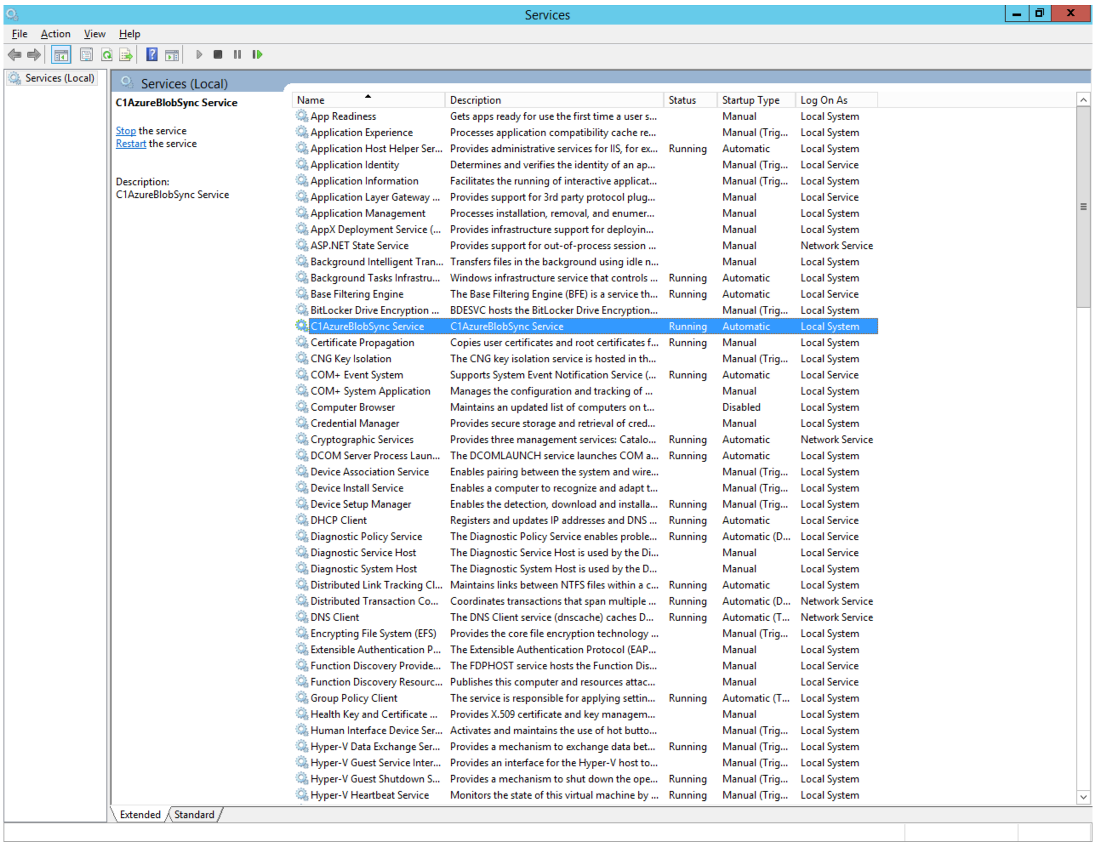
```
<add name="Composite.WindowsAzure.WebRole.Iis.DeleteDefaultSites"
value="false" />
```

You can change other parameters, too. For more information on the environment configuration parameters, please see "Environment Configuration."

6.3 Running Azure Blob Sync as a service

The C1AzureBlobSync tool comes as a Windows executable, which you need to install as a Windows service on the Windows Server and then start it as a service.

1. Locate and run install.bat, which installs the C1AzureBlobSync tool as a Windows service locally.
2. Invoke the "Run" box (Win + R), type in "services.msc" and press "Enter" to run "services.msc".
3. In the "Services" window, locate and run the "C1AzureBlobSync" service.


Services

C1AzureBlobSync Service

Stop the service
Restart the service

Description:
C1AzureBlobSync Service

Name	Description	Status	Startup Type	Log On As
App Readiness	Gets apps ready for use the first time a user s...		Manual	Local System
Application Experience	Processes application compatibility cache re...		Manual (Trig...	Local System
Application Host Helper Ser...	Provides administrative services for IIS, for ex...	Running	Automatic	Local System
Application Identity	Determines and verifies the identity of an ap...		Manual (Trig...	Local Service
Application Information	Facilitates the running of interactive applicat...		Manual (Trig...	Local System
Application Layer Gateway ...	Provides support for 3rd party protocol plug...		Manual	Local Service
Application Management	Processes installation, removal, and enumer...		Manual	Local System
AppX Deployment Service (...	Provides infrastructure support for deployin...		Manual	Local System
ASP.NET State Service	Provides support for out-of-process session ...		Manual	Network Service
Background Intelligent Tran...	Transfers files in the background using idle n...		Manual	Local System
Background Tasks Infrastru...	Windows infrastructure service that controls ...	Running	Automatic	Local System
Base Filtering Engine	The Base Filtering Engine (BFE) is a service th...	Running	Automatic	Local Service
BitLocker Drive Encryption ...	BDESVC hosts the BitLocker Drive Encryption...		Manual (Trig...	Local System
C1AzureBlobSync Service	C1AzureBlobSync Service	Running	Automatic	Local System
Certificate Propagation	Copies user certificates and root certificates f...	Running	Manual	Local System
CNG Key Isolation	The CNG key isolation service is hosted in th...		Manual (Trig...	Local System
COM+ Event System	Supports System Event Notification Service (...	Running	Automatic	Local Service
COM+ System Application	Manages the configuration and tracking of ...		Manual	Local System
Computer Browser	Maintains an updated list of computers on t...		Disabled	Local System
Credential Manager	Provides secure storage and retrieval of cred...		Manual	Local System
Cryptographic Services	Provides three management services: Catalo...	Running	Automatic	Network Service
DCOM Server Process Laun...	The DCOMLAUNCH service launches COM a...	Running	Automatic	Local System
Device Association Service	Enables pairing between the system and wire...		Manual (Trig...	Local System
Device Install Service	Enables a computer to recognize and adapt t...		Manual (Trig...	Local System
Device Setup Manager	Enables the detection, download and installa...	Running	Manual (Trig...	Local System
DHCP Client	Registers and updates IP addresses and DNS ...	Running	Automatic	Local Service
Diagnostic Policy Service	The Diagnostic Policy Service enables proble...	Running	Automatic (D...	Local Service
Diagnostic Service Host	The Diagnostic Service Host is used by the Di...		Manual	Local Service
Diagnostic System Host	The Diagnostic System Host is used by the D...		Manual	Local System
Distributed Link Tracking Cl...	Maintains links between NTFS files within a c...	Running	Automatic	Local System
Distributed Transaction Co...	Coordinates transactions that span multiple ...	Running	Automatic (D...	Network Service
DNS Client	The DNS Client service (dnscache) caches D...	Running	Automatic (T...	Network Service
Encrypting File System (EFS)	Provides the core file encryption technology ...		Manual (Trig...	Local System
Extensible Authentication P...	The Extensible Authentication Protocol (EAP...		Manual	Local System
Function Discovery Provide...	The FDPHOST service hosts the Function Dis...		Manual	Local Service
Function Discovery Resourc...	Publishes this computer and resources attac...		Manual	Local Service
Group Policy Client	The service is responsible for applying settin...	Running	Automatic (T...	Local System
Health Key and Certificate ...	Provides X.509 certificate and key managem...		Manual	Local System
Human Interface Device Ser...	Activates and maintains the use of hot butto...		Manual (Trig...	Local System
Hyper-V Data Exchange Ser...	Provides a mechanism to exchange data bet...	Running	Manual (Trig...	Local System
Hyper-V Guest Service Inter...	Provides an interface for the Hyper-V host to...		Manual (Trig...	Local System
Hyper-V Guest Shutdown S...	Provides a mechanism to shut down the ope...	Running	Manual (Trig...	Local System
Hyper-V Heartbeat Service	Monitors the state of this virtual machine by ...	Running	Manual (Trig...	Local System

Figure 18: C1AzureBlobSync service started

The C1AzureBlobSync service will be monitoring the “websites” and “deployment” container in the Azure blob storage for changes and down-sync the changes to the server or re-configure the deployment on the server accordingly.

6.4 Scaling out on Windows Servers

To scale out the deployment, you need to configure and run the service on each Windows Server (node):

1. Configure the service making additional changes to the configuration as described below.
2. Install and run the service on each new node.

On the first (”master”) node:

1. Edit the env.config and set the Environment.NumberOfInstances parameter set to the number of instances (nodes) in your scaled-out deployment.
2. Run “services.msc” and restart the “C1AzureBlobSync” service.

On the second (”slave”) and any other nodes withing the deployment:

1. Edit env.
2. In addition to the required settings, make sure to set these parameters:
   - Environment.IsThisTheLowestInstance: Set to “false”
   - Environment.NumberOfInstances: Set to the number of instances (nodes)
   - Environment.RoleInstanceId: Set to the unique ID (for example, ”2”)
   - Environment.RoleInstanceName: Set to the unique name (for example, ”Role2”)
3. Install and run the ”C1AzureBlobSync” service.

```
<Settings>
  <add name="Environment.IsThisTheLowestInstance" value="false" />
  <add name="Environment.NumberOfInstances" value="3" />
  <add name="Environment.DeploymentId" value="deployment1" />
  <add name="Environment.RoleInstanceId" value="2" />
  <add name="Environment.RoleInstanceName" value="Role2" />
  <add name="Environment.WebsitesPath" value="c:\www" />
  <add name="Environment.RootPath" value="c:\AzureBlobSync" />
  <add name="Composite.WindowsAzure.WebRole.Iis.DeleteDefaultSites"
value="false" />
  <add name="Composite.WindowsAzure.WebRole.Iis.ManageIisConfiguration"
value="true" />
  <add name="Composite.WindowsAzure.WebRole.SmtpServer.Install"
value="true" />
</Settings>
```

Listing 7: env.config for a “slave” node in the scaled-out deployment on Windows Servers

7 Down-Syncing Websites to Target Servers

After you have uploaded a website to a blob storage and deployed the server functionality on a target server, you need to down-sync the website from the storage to the server.

The synchronization goes unattended but the server needs to “know” where the website is in the blob storage. That’s why you need to manually map the website on the server to the “website” container in the blob container.

You can do the mapping in the “Websites.xml” configuration file in the “deployment” container.

We recommend using CloudBerry Explorer for Azure Blob Storage to access files in the storage. (We assume that you've downloaded and installed CloudBerry Explorer and configured it to access your blob storage.)

1. Open your blob storage in CloudBerry Explorer.
2. Open the “deployment” container, then the folder named "Configuration" in it.
3. Download Websites.xml to your local computer.
4. Edit the Websites.xml file locally and add your website information, specifying:
    - **name**: the name for your website to be created on a target server
    - **storename**: the name of the website container where your website is uloaded
5. Save the changes and upload the modified Websites.xml file back to the blob storage overwriting the existing file.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<Websites>
  <Website name="mywebsite" storename="mywebsite"/>
</Websites>
```

Listing 8: Websites configuration sample

Soon after the Websites.xml gets updated in the storage, the target server will start down-syncing the website based on the information you've specified in Websites.xml (where the website files should be taken from) as well as in WebsiteConfiguration.xml earlier (how the website should be configured and bindings set up).

If the down-syncing process hasn't started automatically:

1. Create an empty file named LastSynchronized.txt.
2. Upload it to [website container]/Configuration.

Normally, new and modified files are down-synced, while unchanged ones are left out and no longer existing files are deleted on the target server.

In the scaled-out deployment, it will down-sync the same website to as many target servers as you have.

Please note that in this manner you can down-sync more than one website within the same deployment. (Please see "Multiple Websites" for more information.)