Deploy Windows Azure Pack V1: Web Sites V2

Microsoft Corporation

Published date: October 20, 2013

Copyright

Contents

Deploy Windows Azure Pack V1: Web Sites V2

Introduction

Windows Azure Pack: Web Sites enables an on-premises, high-density, multi-tenant web hosting service for service providers and enterprise IT. Windows Azure Pack: Web Sites provides an experience similar to Windows Azure Web Sites. It is a scalable, shared, and secure web hosting platform that supports both template web applications and a broad range of programming languages like ASP.NET, PHP and Node.js. In addition to a web sites service, it includes a self-service management portal, uses both SQL and MySQL database servers, integrates with popular source control systems, and offers a customizable web application gallery of popular open source web applications. For more in-depth information on Windows Azure Pack and Windows Azure Pack: Web Sites, including a downloadable white paper, see Windows Azure Pack.

The Windows Azure Pack: Web Sites deployment guide assumes that you have already installed and configured Windows Azure Pack for Windows Server and its corresponding management portals for administrators and tenants. For more information, see **Deploy Windows Azure Pack for Windows Server**.

Upgrading from Preview Versions

To upgrade Web Sites from a preview version (v1 or v2) of Windows Azure Pack for Windows Server, see Upgrading Windows Azure Pack: Web Sites from Preview Versions.

Test Installations

This guide offers a depth of information for a variety of user scenarios. For a test or "proof of concept" installation, you should read at minimum the following chapters, which cover overview, prerequisite, and installation steps.

- Windows Azure Pack: Web Sites Overview
- Windows Azure Pack: Web Sites Pre-installation Steps
- Start the installation of Windows Azure Pack: Web Sites
- Register the Web Sites Cloud and Add Front End, Web Worker, and Publisher Roles

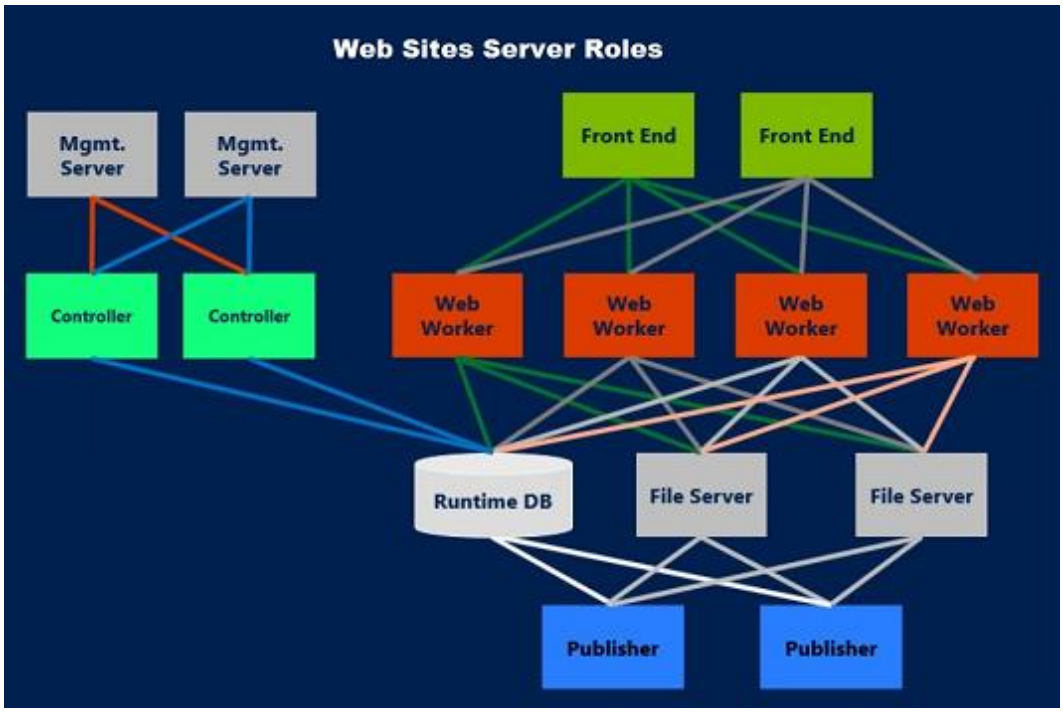
A test installation may also require steps from other chapters depending on the usage scenario that you are trying to test.

Contents

Windows Azure Pack: Web Sites Overview

Overview of Web Sites Roles

The Windows Azure Pack: Web Sites service uses a minimum of 6 server roles: Controller, Management Server, Front End, Web Worker, File Server, and Publisher. Also required is a SQL Server for the Web Sites runtime database. These roles are separate from, and in addition to, the servers that form an Express or Distributed installation of the Service Management API. The roles can be installed on physical servers or virtual machines.



The Windows Azure Pack Web Sites service includes the following server roles:

Web Sites Controller - The controller provisions and manages the other Web Sites Roles. This role is installed first.

Management Server - This server exposes a REST endpoint that handles management traffic to the Windows Azure Pack Web Sites Management API.

Web Workers - These are web servers that process client web requests. Web workers are either **Shared** or **Reserved** (at minimum, one of each is required) to provide differentiated levels of service to customers. Reserved workers are categorized into small, medium, and large sizes.

Important
Because Web Workers run customer code, they represent a potential risk to the Web Sites infrastructure. After installation, you should configure IP Filtering from the Management Portal for Administrators to reduce the risk. For more information, see Configure IP filtering.

Front End - Accepts web requests from clients, routes requests to Web Workers, and returns web worker responses to clients. Front End servers are responsible for load balancing and SSL termination.

File Server - Provides file services for hosting web site content. The File Server houses all of the application files for every web site that runs on the Web Sites Cloud. For more detailed information, see Capacity Planning for Windows Azure Pack: Web Sites.

Publisher - Provides content publishing to the Web Sites farm for FTP clients, Visual Studio, and WebMatrix through the Web Deploy and FTP protocols.

Overview of SQL Server roles

A Windows Azure Pack environment that includes Windows Azure Pack: Web Sites requires the following three database categories:

- **Service Management API database** - The core installation of the Windows Azure Pack Service Management API uses a SQL Server to store its configuration data. This database should have already been installed before performing the steps in this deployment guide. For more information, see **Install Microsoft SQL Server** in the **Deploy Windows Azure Pack for Windows Server** guide.
- **Web Sites Runtime Database** - Prior to installing Windows Azure Pack: Web Sites, you will need to prepare a SQL Server to contain the runtime database that Web Sites uses for its operations. For more information, see Prepare a SQL Server to hold the Windows Azure Pack Web Sites Runtime Database.
- **Application Databases** - If your usage scenario includes providing database functionality for the tenant web sites, you will need to install separate SQL server and/or MySQL databases to provide this service. For more information, see Provision SQL Server and MySQL Application Databases for Tenant Use.

For information on scaling up SQL Server, see Configuring SQL Server for High Availability.

See Also

Deploy Windows Azure Pack: Web Sites

Capacity Planning for Windows Azure Pack: Web Sites

Servers: Physical or Virtual?

Windows Azure Pack: Web Sites roles can be installed on Windows Server 2012 R2 on physical computers or on Hyper-V virtual machines. As the performance gap between virtual machines on Hyper-V and physical hardware shrinks, the cost/performance advantage of virtual machines makes them more attractive.

Capacity Planning by Web Sites Server Role

Controller

The Web Sites Controller typically experiences low consumption of CPU, memory, and network resources. However, for High Availability, you should have two controllers. Two controllers is also the maximum number of controllers permitted. You can create the second Web Sites Controller

by using PowerShell and command line scripts. For more information, see Provision a Second Web Sites Controller.

Front End

The Front End routes requests to Web Workers depending on Web Worker availability. For High Availability, you should have more than one Front End, and you can have more than two. For capacity planning purposes, consider that each core can handle approximately 100 requests per second. For information on adding additional Front End servers, see Scaling Windows Azure Pack: Web Sites for High Availability.

Management Server

The Web Sites Management Server role handles Web Sites Management traffic by using the Windows Azure Pack Web Sites Service REST API. The Management Server role typically requires only about 4 GB RAM in a production environment. However, it may experience high CPU levels when many management tasks (such as web site creation) are performed. For High Availability, you should have more than one server assigned to this role, and at least two cores per server.

For information on adding additional Management Servers, see Provision Additional Management Servers.

Publisher

The Publisher role may experience heavy CPU utilization if many tenants are publishing simultaneously. For High Availability, make more than one Publisher role available. For information on adding additional Publisher servers, see Scaling Windows Azure Pack: Web Sites for High Availability.

File Server

For the File Server role, you can use the Standalone file server for development and testing. For production purposes, you should use a pre-configured Windows File Server, or a pre-configured non-Windows file server.

The Standalone file server is included as part of the default Windows Azure Pack: Web Sites installation. The Standalone installation provisions the File Server role on a single machine, places ACLs for the appropriate accounts, and creates the necessary network shares.

In production environments, the File Server role experiences intensive disk I/O. Because it houses all of the content and application files for tenant web sites, you should pre-configure a Windows File Server, File Server Cluster, or a non-Windows file server, file server cluster, or NAS (Network Attached Storage) device for this role. For more information, see Pre-configure a Windows File Server Cluster or NAS device for Windows Azure Pack: Web Sites.

Warning

Windows Azure Pack: Web Sites relies on File Server Resource Manager (FSRM), which does not support scale-out file servers.

Web Worker

For High Availability, you should have at least four Web Worker Roles, two for Shared web site mode and two for Reserved web site mode. The Shared and Reserved web site modes provide different levels of service to tenants. Of course, if you have many customers using Reserved mode (which is resource intensive), or many customers running in shared mode, more Web Workers will be required.

Important

When considering the number of Web Worker roles to provision, remember that after a subscriber has placed a Web Worker in Reserved mode, that Web Worker will no longer be available to subscribers in Shared mode. For this reason, installing Windows Azure Pack: Web Sites without a Shared Web Worker instance is an unsupported configuration.

To help you determine the number of Web Worker roles required, consider the following:

- **Memory** - Memory is the most critical resource for a Web Worker role. Insufficient memory impacts web site performance when virtual memory is swapped from disk. Each server requires approximately 1.2 GB of RAM for the operating system; the RAM available above this threshold can be used to run web sites.
- **Percentage of active web sites** - Based on observed production workloads, approximately 5 percent of web sites in a Web Site Cloud are typically active. However, the percentage of web sites that are active at any given moment can be significantly higher or lower. Assuming an "active web site" rate of 5 percent, the maximum number of web sites to place in a Web Site Cloud should be no more than 20 times the number of active web sites *(5 x 20 = 100)*.
- **Average memory footprint** - The average memory footprint for websites observed in production environments is about 70 MB. Based on this number, the amount of memory that should be allocated across all Web Worker role computers or VMs installed on a Web Site Cloud may be calculated as follows:

  *Number of Provisioned web sites * 70MB * 5% - (Number of Web Worker Roles * 1.2 GB)*

  For example, if 5,000 web sites are provisioned on a Web Site Cloud that is running 10 Web Worker roles, then each Web Worker role computer or VM should be allocated 7060 MB of RAM determined as follows:

  *5,000 * 70 * .05 – (10 * 1044) = 6.89* (=about 7 GB)

For information on how to add Web Worker instances, see Scaling Windows Azure Pack: Web Sites for High Availability.

Windows Azure Pack Web Sites Runtime SQL Server Database

Windows Azure Pack Web Site Cloud makes extensive use of SQL Server. For High Availability, follow these guidelines for allocating RAM, Disk, and CPU resources:

- **Memory** - Because SQL Server performance is most dependent on available memory, allocate at least 4 GB of RAM to your SQL Server for every 30,000 sites that are provisioned. For most scenarios, SQL performance will benefit from additional memory, and SQL Server will use as much memory as you allocate to it.
- **Disk space** - For every 10,000 sites that are provisioned, allocate at least 4 GB of disk space.
- **CPU Count** - To determine the number of cores to allocate to your SQL Server computer, you can use the following criteria:

  When Task Manager or Performance Monitor shows that the CPU usage of SQL Server service approaches 70%, allocate one additional core.

For additional measures that you take to increase the availability of your SQL Servers, see Configuring SQL Server for High Availability.

See Also

Deploy Windows Azure Pack: Web Sites

Windows Azure Pack: Web Sites Pre-installation Steps

The steps in this chapter are important for a successful installation of Windows Azure Pack: Web Sites. Be sure to read each section carefully and perform the actions as required. The sections are as follows:

Domain vs. non-Domain considerations

Create Servers for the Web Sites Roles

Advice for Preparing your VHDs or Servers

Prepare a SQL Server to hold the Windows Azure Pack Web Sites Runtime Database

Provision SQL Server and MySQL Application Databases for Tenant Use

Web Sites Roles Firewall Configuration

Configure the Front End and Publisher roles for inbound access from the Internet

Configure Windows Azure Pack: Web Sites to use Proxy Servers

Modify User Account Control for Remote Access

Configure DNS mappings for the Web Sites Cloud

Domain vs. non-Domain considerations

Windows Azure Pack: Web Sites can be installed in either a domain-joined or non-domain joined (workgroup) environment. While a workgroup environment might be appropriate for development or test scenarios, we recommend that you use a domain-joined environment for the following reasons:

- Users can leverage Kerberos for authentication handshakes, which is much more secure than either NTLM or NTLM v2.
- Simplified user and credential management.
- Improved management through group policy

Create Servers for the Web Sites Roles

Prior to installing the Web Sites cloud, prepare one Windows Server 2012 R2 server per Web Site role (the servers can be either physical or virtual machines). Later, for High Availability, you should add more than one server per role. For ease of maintenance, use descriptive computer names such as those below. The naming convention suggested here is that you replace the "x" in each name with "1" (padding with zeros as needed), and then increment that number as you add server instances to a role.

- *SitesCNx* – Web Sites Cloud Controller
- *SitesMNx* – Web Sites Cloud Management Server
- *SitesFEx* – Web Sites Cloud Front End
- *SitesPBx* – Web Sites Cloud Publisher
- *SitesWWSx* – Web Sites Cloud Shared Web Worker
- *SitesWWRx* – Web Sites Cloud Reserved Web Worker
- *SitesFSx* – Web Sites Cloud File Server

Advice for Preparing your VHDs or Servers

- Use clean installations of Windows Server 2012 (Windows Server 2012 R2 is recommended).
- Do not co-locate Web Sites roles; each Web Site role should have its own VM or Server.
- Do not co-locate Web Sites roles with the Admin or Tenant Portal or API servers. For example, the Web Sites Cloud Controller and the Admin API roles must be on separate machines.
- It is recommended that you use domain-joined servers, as noted previously.
- If you are using VMs, it is a good idea to take snapshots for each role at the following junctures:
  - Pre installation
  - Post installation
  - Post configuration

 **Note**

To avoid NetBIOS errors and automatic name truncation, make sure the computer names you choose are no longer than 15 characters.

Prepare a SQL Server to hold the Windows Azure Pack Web Sites Runtime Database

For the Web Sites runtime database, prepare a SQL Server database. As a best practice, all SQL Server roles, including the Web Sites runtime database, the Service Management API database that is part of should be on separate machines.

- For testing, development, or "proof of concept" purposes, you can use SQL Express 2012 SP1 or later. For download information, see Download SQL Server 2012 Express with SP1.
- For production purposes, you should use a full version of SQL 2012 SP1 or later. For information on installing SQL Server, see Installation for SQL Server 2012.
- Mixed Mode authentication should be enabled.
- The Web Sites runtime SQL Server should be accessible from all Web Sites roles.
- For any of the SQL Server roles, you can use a default instance or a named instance. However, if you use a named instance, be sure that you manually start the SQL Browser Service and open port 1434.

For more information, see **Using SQL Server or MySQL with Windows Azure Pack**.

Provision SQL Server and MySQL Application Databases for Tenant Use

Your environment may require that you support tenant SQL Server and/or MySQL database applications. If so, you should install and configure separate SQL Server and MySQL instances dedicated for tenant application database use.

SQL Server tenant application database server requirements:

- Mixed Mode authentication should be enabled.
- The SQL server should be accessible from the Web Worker and Publisher roles
- The SQL server should be accessible from the Admin and Tenant API Roles

MySQL tenant application database server requirements:

- Using the MySQL command line interface, the root user should be enabled for remote access with a password. Example syntax:

```
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY
'password' WITH GRANT OPTION;

FLUSH PRIVILEGES;
```

  For more information on the MySQL command line interface, see the MySQL Documentation.
- The MySQL server should be accessible from the Web Worker and Publisher roles
- The MySQL server should be accessible from the Admin and Tenant API Roles
- For IPv6 support, use MySQL version 5.5.30 or later

For more information, see **Configure SQL Server and MySQL Application databases for tenant use**. For information on scaling up SQL Server for production use, see Configuring SQL Server for High Availability.

Web Sites Roles Firewall Configuration

If you implement Windows Azure Pack: Web Sites entirely in an intranet environment, public internet access is not necessary. (Even if internet access is necessary, a well-planned network topology can prevent the exposure of the Web Sites roles directly to the internet.)

A server's firewall settings should be configured before you add a Web Sites role to the server. The following list summarizes internet access requirements for each Web Site role:

- **Management Server** - does not require public internet access.
- **File Server** - does not require internet access and should never be internet accessible.
- **Web Sites Controller** - requires outbound HTTP access but not inbound. The Controller requires outbound access so it can download software dependencies for the installation of Web Sites roles.
- **Web Workers** - never need inbound internet access, but may require outbound access if (for example) a web application consumes an external web service.
- **Front End** servers may or may not require direct inbound network access to accept client requests for websites, depending on network topology.
- **Publishers** may or may not require direct inbound access to accept FTP and WebDeploy publishing requests, depending on network topology.

For the Front End and Publisher roles, you can avoid direct internet access if you have upstream load balancers or upstream NAT devices that handle network address translation between public and private addresses.

Configure the Front End and Publisher roles for inbound access from the Internet

On the servers that will become the Front End and Publisher, allow inbound access to the following services:

- File and printer sharing (SMB-In)
- Windows Management Instrumentation (WMI-In)

To configure inbound access, you can use the following netsh commands:

```
netsh advfirewall firewall set rule group="File and Printer Sharing" new enable=Yes

netsh advfirewall firewall set rule group="Windows Management Instrumentation (WMI)" new
enable=yes
```

Note

The network adapters for the Front End and Publisher roles should be configured in Windows Firewall to use the Public Profile.

Configure Windows Azure Pack: Web Sites to use Proxy Servers

If you are using a proxy address for outbound access to the internet, you must enable Web Sites to use the proxy address before you can install and run Windows Azure Pack: Web Sites.

Note

Web Farm authentication through a proxy server is not currently supported. This section describes steps that must be taken to install and run Windows Azure Pack: Web Sites behind a proxy server. It does not describe how to configure proxy servers.

To enable the farm to communicate through a proxy without authentication, run the following PowerShell commands with administrative privileges on the Controller. Replace <*ProxyAddress*> with the address of the proxy

```
Import-Module WebSites


Set-WebSitesConfig Global -ProxyEnabled True -ProxyAddress <ProxyAddress>
```

Note

When installing the Web Sites service for Windows Azure Pack, run the Web Platform Installer using an account that can authenticate against the proxy.

Allow Microsoft Updates access to Windows Azure Pack: Web Sites behind the proxy

As an optional step when Windows Azure Pack: Web Sites is behind a proxy, it is suggested that you enable Microsoft Updates so that they can work correctly with Windows Azure Pack: Web Sites. To do so, run the following PowerShell commands with administrative privileges on the Controller, where user <*ProxyUser*> with password <*ProxyUserPassword*> can authenticate against the proxy:

```
Import-Module WebSites


New-WebSitesConfig Credential -CredentialName ProxyUserCredential -UserName <ProxyUser> -
Password <ProxyUserPassword> -CredentialType SystemCredential
```

Modify User Account Control for Remote Access

On each server to be used for a Windows Azure Pack: Web Sites role, disable User Account Control (UAC) for remote connections as described below. This will allow remote administration to be carried out.

Note

Local UAC configuration will remain unchanged.

1. Run the following command from an elevated command prompt:

```
reg add
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\system
/v LocalAccountTokenFilterPolicy /t REG_DWORD /d 1 /f
```

2. Restart the server.

Configure DNS mappings for the Web Sites Cloud

During Web Sites Cloud setup, when configuring the Web Sites Controller, you will be prompted for the default domain name to be used for end user web sites.

By default, web sites are created under a default domain. Once a web site is created, users can add custom domain names to each web site. While tenant web sites can be configured to support custom domains, Windows Azure Pack: Web Sites does not update custom DNS records. To ensure that all internet-facing web sites are configured with the appropriate IP address assignments, follow the guidance below.

For a given domain such as Contoso.com, create the following DNS A records:

Host name	**IP Address for**
*	Front End Server(s)
*.scm	Front End Server(s)
ftp	Publishing Server(s)
publish	Publishing Server(s)

This mapping scheme allows users to log on to both http://www.contoso.com and http://contoso.com to manage their sites. The Management Portal for Administrators and the Management Portal for Tenants are described in Deploy Windows Azure Pack for Windows Server.

In the sample configuration above, user-created web sites are created using child domains such as **site1.contoso.com**, **site2.contoso.com**.

When users publish content to their web sites via Web Deploy or FTP, they will use **publish.contoso.com** or **ftp.contoso.com** respectively. Content publishing via Git uses ***.scm.contoso.com**.

Note

There is no requirement for a special domain for this deployment. You can use a subdomain like my.yourdomain.com under an existing domain.

See Also

Deploy Windows Azure Pack: Web Sites

Pre-configure a Windows File Server Cluster or NAS device for Windows Azure Pack: Web Sites

This chapter shows you how to configure your own File Server or File Server Cluster for use with Windows Azure Pack: Web Sites.

Background

If you choose the Standalone Windows File Server option during installation, file server preparation is not required and is automated for you. However, although the Standalone option is useful for "proof of concept" installations, a production environment usually requires a more robust solution such as a Windows File Server Cluster or third-party Network Attached Storage device (NAS). Windows Azure Pack: Web Sites uses does not depend on per-web site file share permissions, which enables it to work with heterogeneous file storage implementations such as NAS devices.

Warning
Windows Azure Pack: Web Sites relies on File Server Resource Manager (FSRM), which does not support scale-out file servers.

Five Main Steps

Pre-configuring your own Windows File Server, Windows File Server Cluster, or third party NAS device involves the following five main steps. The implementation of these steps varies depending on whether you are working in an Active Directory domain or in a workgroup environment. Steps for both environments are presented.

Note
Although it is beyond the scope of this document to provide configuration instructions for third-party NAS devices, you should generally follow the procedures presented here, making adjustments as required by your non-Windows file cluster or NAS device.

1. Provision Groups and Accounts

Provision Groups and Accounts in Active Directory

1. Create the following Active Directory global security groups:
   a. **FileShareOwners**
   b. **FileShareUsers**
   c. **CertStoreFSUsers**
2. Create the following Active Directory accounts as service accounts. The accounts to create are
   a. **FileShareOwner**
   b. **FileShareUser**
   c. **CentralCertStoreUser** (When the standalone file server option is used, this account is called CertificateShareUser).

   Note
   As a security best practice, the users for these accounts (and for all Web Roles) should be distinct from each other and have strong user names and passwords. For more information, see Windows Azure Pack: Web Sites Security Enhancements.

   The FileShareOwner, FileShareUser, and CentralCertStoreUser passwords must be set with the following conditions:
   - Enable **Password never expires**
   - Enable **User cannot change password**
   - Disable **User must change password at next logon**
3. Add the accounts to the group memberships as follows:
   a. Add **FileShareOwner** to the **FileShareOwners** group
   b. Add **FileShareUser** to the **FileShareUsers** group
   c. Add **CentralCertStoreUser** to the **CertStoreFSUsers** group

Provision Groups and Accounts in a Workgroup

On a workgroup, run net and WMIC commands to provision groups and accounts.

1. Run the following commands to create the FileShareOwner, FileShareUser, and CentralCertStoreUser accounts. Replace *<password>* with your own values.

```
net user FileShareOwner <password> /add /expires:never
/passwordchg:no

net user FileShareUser <password> /add /expires:never
/passwordchg:no

net user CentralCertStoreUser <password> /add /expires:never
/passwordchg:no
```

2. Set the passwords for the accounts just created to never expire by running the following WMIC commands:

```
WMIC USERACCOUNT WHERE "Name='FileShareOwner'" SET
PasswordExpires=FALSE
WMIC USERACCOUNT WHERE "Name='FileShareUser'" SET
PasswordExpires=FALSE
WMIC USERACCOUNT WHERE "Name='CentralCertStoreUser'" SET
PasswordExpires=FALSE
```

3. Create the local groups **CertStoreFSUsers**, **FileShareUsers** and **FileShareOwners**, and add the accounts in the first step to them.

```
net localgroup CertStoreFSUsers /add
net localgroup CertStoreFSUsers CentralCertStoreUser /add
net localgroup FileShareUsers /add
net localgroup FileShareUsers FileShareUser /add
net localgroup FileShareOwners /add
net localgroup FileShareOwners FileShareOwner /add
```

2. Enable Windows Remote Management (WinRM) and File Server Resource Manager (FSRM)

On the File Server role, or on each node of the Windows File Server Cluster if you are using a cluster, run the following commands at an elevated command prompt to configure WinRM and FSRM:

```
powershell.exe Enable-PSRemoting –Force
winrm.cmd set winrm/config/winrs
@{MaxConcurrentUsers="10";MaxShellsPerUser="50";MaxProcessesPerShell="5000";IdleTimeout="
10000"}


netsh advfirewall firewall set rule name="Windows Remote Management (HTTP-In)" new
remoteip=any


%windir%\system32\dism.exe /online /enable-feature /featurename:FSRM-Management
/featurename:FSRM-Infrastructure /all
```

3. Provision the Content Share and the Certificate Share

The Content Share contains tenant web site content, whereas the Certificate Share contains custom tenant certificates.

The procedure to provision the content share and the certificate share on a single file server is the same for both Active Directory and Workgroup environments, but different for a Failover cluster in Active Directory.

Provision the content and certificate shares on a single file server (AD or Workgroup)

On a single file server, run the following commands at an elevated command prompt. Replace the values for <*C:\WebSites*> and <*C:\Certificates*> with the corresponding paths in your environment.

```
set WEBSITES_SHARE=WebSites

set CERTIFICATES_SHARE=Certificates

set WEBSITES_FOLDER=<C:\WebSites>

set CERTIFICATES_FOLDER=<C:\Certificates>


md %WEBSITES_FOLDER%

md %CERTIFICATES_FOLDER%


net share %WEBSITES_SHARE% /delete

net share %WEBSITES_SHARE%=%WEBSITES_FOLDER% /grant:Everyone,full


net share %CERTIFICATES_SHARE% /delete

net share %CERTIFICATES_SHARE%=%CERTIFICATES_FOLDER% /grant:Everyone,full
```

Provision the content and certificate shares on a Failover cluster (Active Directory)

On the Failover cluster, create the following UNC clustered resources:

1. WebSites
2. Certificates

4. Add the FileShareOwners group to the local Administrators group to enable WinRM

In order for Windows Remote Management to work properly, you must add the FileShareOwners group to the local Administrators group.

Active Directory

Execute the following commands at an elevated command prompt on the File Server, or on every File Server Failover Cluster node. Replace the value for *<DOMAIN>* with the domain name you will use.

```
set DOMAIN=<DOMAIN>

net localgroup Administrators %DOMAIN%\FileShareOwners /add
```

Workgroup

Execute the following command at an elevated command prompt on the File Server.

```
net localgroup Administrators FileShareOwners /add
```

5. Configure access control to the shares

Execute the following commands at an elevated command prompt on the File Server or on the File Server Failover Cluster node which is the current cluster resource owner. Replace values in italics with values specific to your environment.

Active Directory

```
set DOMAIN=<DOMAIN>

set WEBSITES_FOLDER=<C:\WebSites>

set CERTIFICATES_FOLDER=<C:\Certificates>


icacls %WEBSITES_FOLDER% /reset

icacls %WEBSITES_FOLDER% /grant Administrators:(OI)(CI)(F)

icacls %WEBSITES_FOLDER% /grant %DOMAIN%\FileShareOwners:(OI)(CI)(M)

icacls %WEBSITES_FOLDER% /inheritance:r

icacls %WEBSITES_FOLDER% /grant %DOMAIN%\FileShareUsers:(CI)(S,X,RA)

icacls %WEBSITES_FOLDER% /grant *S-1-1-0:(OI)(CI)(IO)(RA,REA,RD)


icacls %CERTIFICATES_FOLDER% /reset

icacls %CERTIFICATES_FOLDER% /grant %DOMAIN%\FileShareOwners:(OI)(CI)(F)

icacls %CERTIFICATES_FOLDER% /inheritance:r

icacls %CERTIFICATES_FOLDER% /grant %DOMAIN%\CertStoreFSUsers:(OI)(CI)(RX)
```

Workgroup

```
set WEBSITES_FOLDER=<C:\WebSites>
```

```
set CERTIFICATES_FOLDER=<C:\Certificates>


icacls %WEBSITES_FOLDER% /reset
icacls %WEBSITES_FOLDER% /grant Administrators:(OI)(CI)(F)
icacls %WEBSITES_FOLDER% /grant FileShareOwners:(OI)(CI)(M)
icacls %WEBSITES_FOLDER% /inheritance:r
icacls %WEBSITES_FOLDER% /grant FileShareUsers:(CI)(S,X,RA)
icacls %WEBSITES_FOLDER% /grant *S-1-1-0:(OI)(CI)(IO)(RA,REA,RD)


icacls %CERTIFICATES_FOLDER% /reset
icacls %CERTIFICATES_FOLDER% /grant FileShareOwners:(OI)(CI)(F)
icacls %CERTIFICATES_FOLDER% /inheritance:r
icacls %CERTIFICATES_FOLDER% /grant CertStoreFSUsers:(OI)(CI)(RX)
```

See Also

Deploy Windows Azure Pack: Web Sites

Windows Azure Pack: Web Sites Dependencies

Windows Azure Pack: Web Sites has a number of dependencies on third-party products that the Web Platform Installer (Web PI) installs for you automatically (you do not need to install these dependencies manually). These dependencies are listed in the Web PI user interface during installation, and are also provided here, in alphabetical order, for convenience.

Note
For compatibility reasons, different versions of the same product may in some cases be dependencies.

Third-party dependencies for Windows Azure Pack: Web Sites

The following list is subject to change. See the Web Platform Installer for a current list of third-party dependencies.

Third-party Dependency	License
Git 1.7.11	http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
Mercurial 2.4.1	http://www.gnu.org/licenses/old-licenses/gpl-2.0.txt
MySQL Connector 5.5.30	http://www.gnu.org/licenses/old-licenses/gpl-2.0.txt
MySQL Connector 6.5.4	http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
Node.js 0.10.5 Executable	https://raw.github.com/joyent/node/v0.10.5/LICENSE
Node.js 0.6.17 Executable	https://raw.github.com/joyent/node/v0.6.17/LICENSE
Node.js 0.6.20 Executable	https://raw.github.com/joyent/node/v0.6.20/LICENSE
Node.js 0.6.20 Installer	https://raw.github.com/joyent/node/v0.6.20/LICENSE
Node.js 0.8.2 Executable	https://raw.github.com/joyent/node/v0.8.2/LICENSE
Note.js 0.8.19 Executable	https://raw.github.com/joyent/node/v0.8.19/LICENSE
PHP 5.2.17	http://www.php.net/license/3_01.txt
PHP 5.3.19	http://www.php.net/license/3_01.txt
PHP 5.4.9	http://www.php.net/license/3_01.txt
PHP Manager for IIS	http://phpmanager.codeplex.com/license
Python 2.7.3	http://docs.python.org/2/license.html
WFastCGI 1.5	http://pytools.codeplex.com/license

See Also

Deploy Windows Azure Pack: Web Sites

Windows Azure Pack: Web Sites Pre-installation Checklist

For convenience, here is a summary of pre-installation tasks for Windows Azure Pack: Web Sites. For detailed information, consult the indicated section of this guide.

	Task	References

1	Understand Web Sites Roles and SQL Roles	Windows Azure Pack: Web Sites Overview
2	Assess capacity requirements for the Web Sites and Web Sites Runtime SQL Server roles in your environment	Capacity Planning for Windows Azure Pack: Web Sites
3	Decide on domain vs. non-domain	Domain vs. non-Domain considerations
4	Provision Web Sites roles servers	Create Servers for the Web Sites Roles
5	Prepare a SQL Server for the Web Sites Runtime	Prepare a SQL Server to hold the Windows Azure Pack Web Sites Runtime Database
6	Provision SQL Server and MySQL Tenant Databases	Provision SQL Server and MySQL Application Databases for Tenant Use
7	Configure firewalls on Web Sites Roles	Web Sites Roles Firewall Configuration
8	(If applicable) configure Web Sites to use Proxy Servers	Configure Windows Azure Pack: Web Sites to use Proxy Servers
9	(If applicable) configure Microsoft Updates to use Proxy	Allow Microsoft Updates access to Windows Azure Pack: Web Sites behind the proxy
10	Modify UAC for Remote Access	Modify User Account Control for Remote Access
11	Configure DNS mappings	Configure DNS mappings for the Web Sites Cloud
12	Decide which File Server type to use (Standalone or pre-configured)	Pre-configure a Windows File Server Cluster or NAS device for Windows Azure Pack: Web Sites
13	(If applicable) pre-configure a Windows File Server Cluster or NAS (includes 5 major sub-steps)	

See Also

Deploy Windows Azure Pack: Web Sites

Start the installation of Windows Azure Pack: Web Sites

At this point you should already have installed and configured Windows Azure Pack for Windows Server and the corresponding management portals for administrators and tenants. If you have not yet done so, install Windows Azure Pack by following the steps in the **Deploy Windows Azure Pack for Windows Server** documentation set.

After the core installation of Windows Azure Pack is in place and you have determined the architecture that is appropriate for your Web Sites scenario (Capacity Planning for Windows Azure Pack: Web Sites) and performed the Windows Azure Pack: Web Sites Pre-installation Steps, you are ready for the formal installation of Windows Azure Pack: Web Sites.

To begin, you will use the Web Platform Installer on the server or VM that will become the Web Sites Controller.

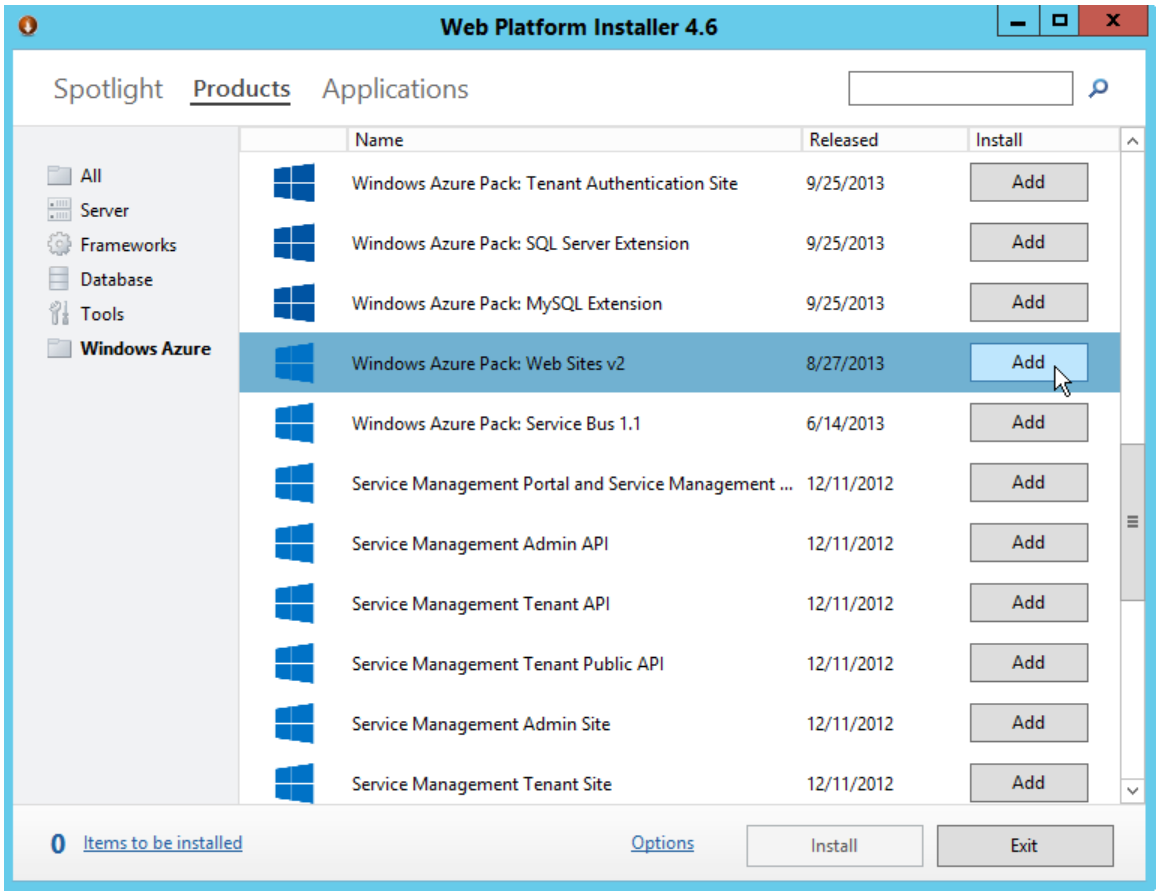
You will then be taken to a Web Sites Setup page where you will specify a SQL Server for the Web Sites Runtime Database, and provide management, worker role, and service endpoint credentials.

After choosing a file server type and specifying a file server name, you will specify a site store (content share) on the file server, provide file share owner and file share user credentials, and specify a path and credentials to the certificate store on the file server.

Finally, you can decide whether to participate in the Customer Experience Improvement Program and/or opt to receive Microsoft Updates for Windows Azure Pack: Web Sites, which is highly recommended.

Install the Web Sites Controller

1. On the server that will become the Web Sites controller (for example, *SitesCN1*), install and launch the Web Platform Installer.
2. On the **Products** tab, click **Windows Azure**. Click **Add** next to **Windows Azure Pack: Web Sites v2**, and then click **Install**.



3. Review the items to be installed. Some of these items are third-party dependencies. For a separate reference list, see Windows Azure Pack: Web Sites Dependencies.
4. Click **I Accept** to accept the license terms if you agree with them.
5. On the **Prerequisites** page, choose **Use Microsoft Update when I check for updates (recommended)** to keep Windows Azure Pack components updated.

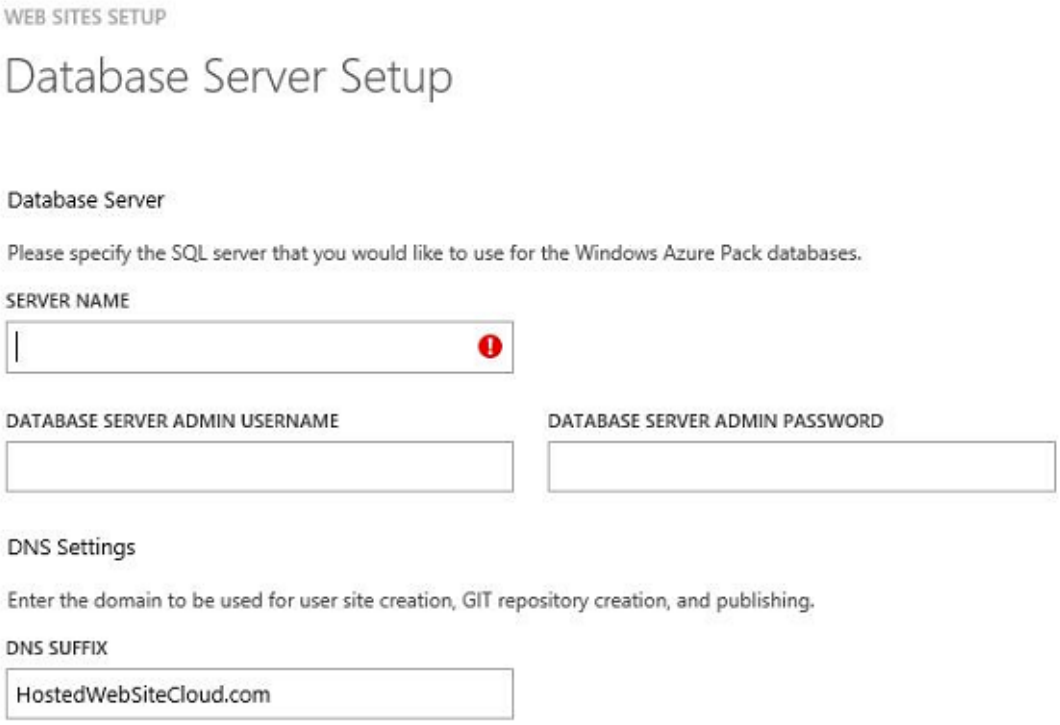
   The installation begins. You can view the progress in the setup wizard.

   If, for any reason, the installation is not successful, the installer tells you which components did not install and provides a link to the installation log file.
6. After installation is complete, click **Continue** to open the Configuration site.

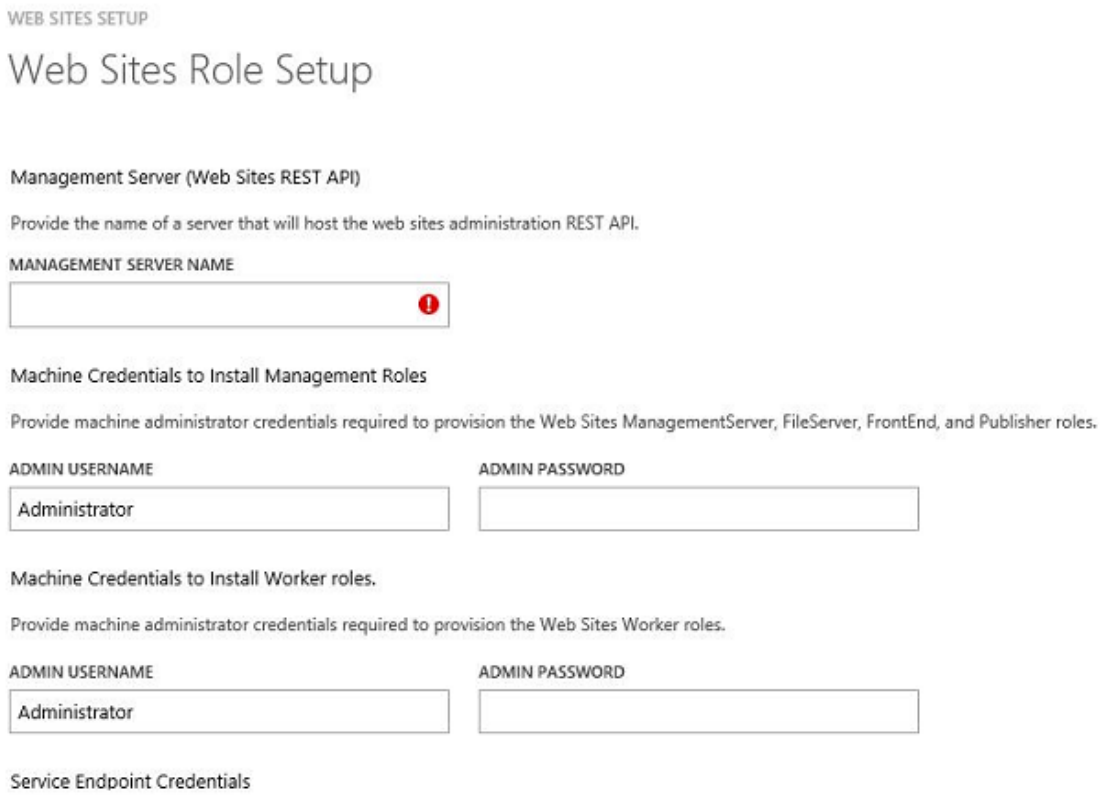
   If your browser displays a certificate security warning (a self-signed certificate is being used at this point), click **Continue to this website (not recommended)**. The **Web Sites Setup** configuration page will appear.

Specify database and file servers and shares, and provide credentials

1. On the Database Server Setup page, in the **Server Name** field, enter the name of the SQL Server instance that the Controller will use to store the Web Sites Runtime database. This database will store web site hosting and resource usage information.



2. Enter the server name on which the SQL Server database is located, the database server administrator user name (**sa**), and password.
3. Enter the DNS suffix (the default domain for user web sites), and then click the next arrow on the bottom right of the page.
4. On the Web Sites Role Setup page, in the **Management Server Name** field, enter the name of the server that will run the management server role, such as **SitesMN1**.



5. Under **Machine Credentials to install Management roles**, enter the administrator user name and password.

   The admin user should be either:

   - A *domain account* that is a member of the local Administrators group on all web site cloud role servers, excluding the web workers; or,
   - A *local account* that is a member of the local Administrators group on all web site cloud role servers, excluding the web workers. If you use a local account, the account name and password must be identical on all servers, excluding the web workers.

     Note
     When specifying a domain account administrator user name, remember to include the domain name prefix (<*DomainName*>\<*UserName*>).

6. Under **Machine Credentials to install Worker roles**, enter the administrator user name and password.

   The administrator user should be either:

   - A *domain account* that is member of the local Administrators group on all web workers; or,
   - A *local account* that is a member of the local Administrators group on all web workers. If you use a local account, the account name and password must be identical on all servers.

7. ***Scroll down*** to provide Service endpoint credentials. Enter the user name and password to be used to connect to the REST endpoint. Confirm the password by entering it again.

Important
Make a note of these credentials as they are required to register your Web Sites REST endpoint in the management portal for administrators.

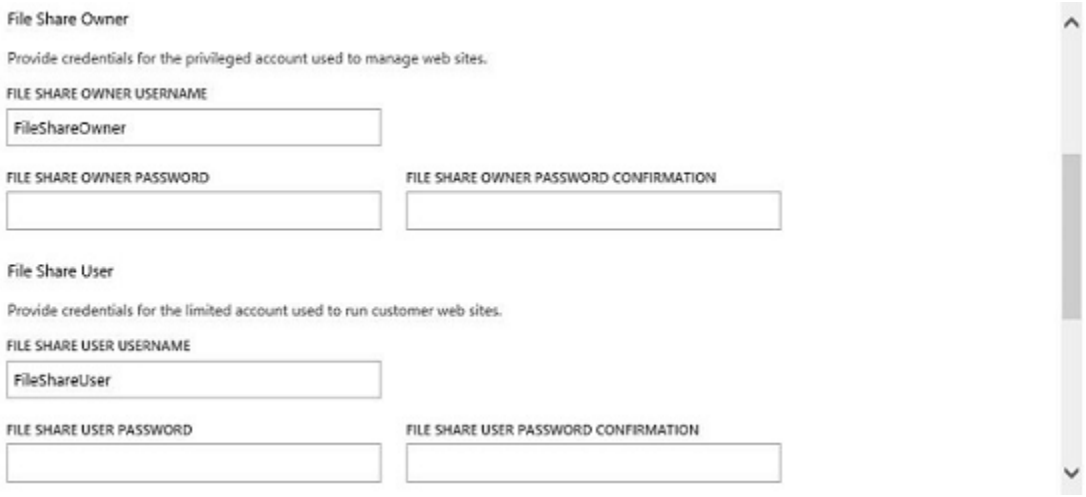
8. Click the next arrow on the bottom right corner of the page.
9. Indicate the type of file server you are using: **Standalone, Pre-configured Windows File Server**, or a **Pre-configured Non-Windows File Server**.

10. If you are creating a standalone file server, provide the **File Server Name**. If you are using a pre-configured file server after following the steps in Pre-configure a Windows File Server Cluster or NAS device for Windows Azure Pack: Web Sites, use the appropriate pre-configured file server option and specify the network path including the file server name.
11. Enter the **Site Store** information. If you are using a pre-configured file server, enter the content share network path including the file server name. Use the format \\<*fileserver*>\WebSites. If you are creating a standalone file server, the path is automatically

populated with \\<*new file server*>\WebSites. For a standalone file server, you also need to enter the physical path to the content share (by default, C:\WebSites).

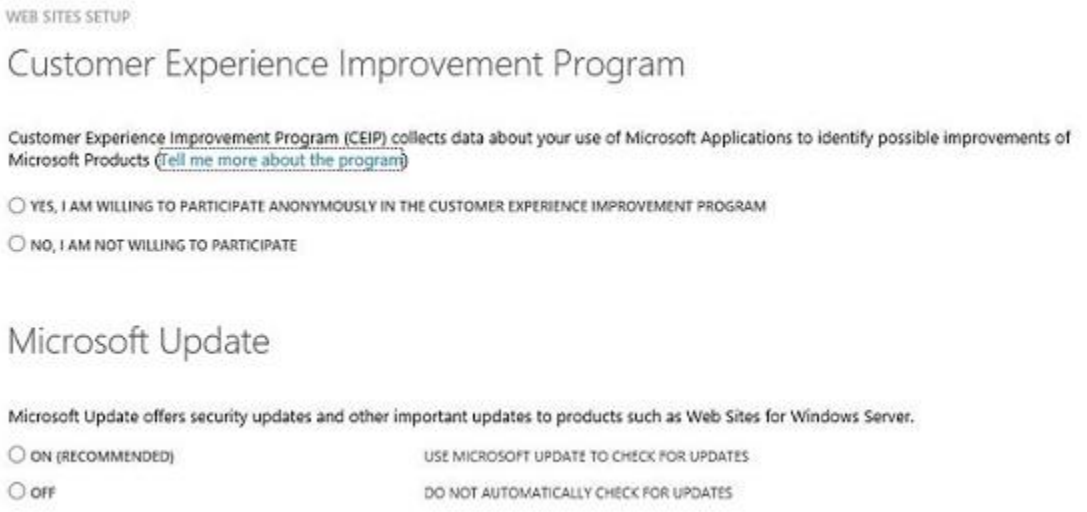
12. Enter the **File Share Owner** user name and password. Confirm the password by entering it again.

    Note
    You may need to scroll down to see these fields.



13. Enter the **File Share User** user name and password. Confirm the password by entering it again.
14. Enter the **Certificate Store** information. Enter the network path to the certificate share. If you are creating a standalone file server, you also need to enter the physical path to the content share.

15. Enter the user name and password for the **Certificate Store Account**. Confirm the password by entering it again.
16. On the bottom right corner of the page, click the **Next** arrow.



17. On the **Customer Experience Improvement Program** page, indicate whether to participate in the Microsoft Customer Experience Improvement Program (CEIP). Microsoft uses the data collected by CEIP to understand how customers like you use the controller. The data contains no personal information. The information collected helps identify problems and results in software improvements to better meet the needs of our customers.

    Under Microsoft Update, select **On (Recommended)** if you want to use Microsoft Update to keep the Web Sites Cloud controller up to date. It is strongly recommended that you opt in to Microsoft Update so that you will know when there are security patches or vulnerability fixes available for Windows Azure Pack: Web Sites.

    Important
    Updates for Windows Azure Pack: Web Sites are provided through Microsoft Update, not Windows Update. It is highly recommended that you enable both Microsoft Update and Windows Update for all of your Web Sites server roles. For more information on configuring Windows Update for Windows Server, see How to configure automatic updates by using Group Policy or registry settings.

    Note
    Opting in to Microsoft Update does not affect the settings for when you will receive Windows updates.

    On the bottom right corner of the page, click the **Next** arrow.

18. Review the features that are going to be configured, and then click the checkmark.

See Also

Deploy Windows Azure Pack: Web Sites

Register the Web Sites Cloud and Add Front End, Web Worker, and Publisher Roles

At this point you have installed the Web Sites Controller and specified a Management Server and a Web Sites runtime database Server. You have also specified a File Server and Certificate Share, and set some important credentials. Now you are ready to register the Web Site Cloud REST endpoint, after which you will install the Front End, Web Worker, and Publisher roles.

Note
By this point you should have already provisioned servers or VMs for the remaining Web Sites roles. If you have not yet done so, see Create Servers for the Web Sites Roles.

Register the Web Site Cloud REST Endpoint

1. Log into the server where you deployed the Windows Azure Pack Management Portal for Administrators.
2. Launch the management portal (https://localhost:30091) if it is not already launched.
3. Click **Web Site Cloud**, and then click **Register your Web Site Cloud REST Endpoint**.
4. Enter the following information for the resource provider:

Display name	Enter a name to display in the management portal.
Web Site Cloud REST Endpoint	Enter the URL for the REST endpoint (for example, https:// <*SitesRESTAPI*>).
User name	Enter the user name that you specified when creating the Service Endpoint Credentials during Web Sites cloud controller installation.
Password	Enter the password that you specified when creating the Service Endpoint Credentials.

5. Click the checkmark in the bottom right of the page to continue. When you receive a message indicating that registration was successful, close the message.

Set up the Front End

1. Choose the cloud you created to open the Web Site cloud dashboard.
2. Click **Setup Frontend** to setup the Front End role for the web site cloud.
3. Enter the hostname or IP for the Front End (for example, *SitesFE*). Click the checkmark to continue.
4. To create additional Front End instances, repeat these steps.

Add the Web Workers

Note
You can add the Web Worker roles while the Front End is installing.

1. Click **Add Role** at the bottom of the Web Site Cloud Quickstart page.
2. Click **Add New Web Worker**. The **Setup a new Web Worker** dialog is displayed.
3. Enter the name of the server that you created for your Shared (multitenant) Web Worker role (for example, *SitesWWS1*), select the **Shared** option, and then click the checkmark to continue.
4. Repeat the process to create a Reserved (single tenant) Web Worker role.

   Important
   Ensure that at least two servers are configured to run the shared Web Worker role (one for Shared, and one for Reserved). Creation of a Web Site cloud without a server configured to run the Shared Web Worker role is not a supported configuration.
5. To create additional Shared or Reserved workers, repeat the previous steps.

Add the Publisher

1. Click **Add Role** and add the **Publisher** role. Enter the server name for the Publisher (for example, *SitesPB1*) and click the checkmark to continue.
2. To add additional publishers, repeat the previous steps.

See Also

Deploy Windows Azure Pack: Web Sites

Validate Your Installation with the Web Sites MBCA2 Model

To help validate your installation of the Windows Azure Pack: Web Sites, you can use the Web Sites MBCA2 Model. The Web Sites MBCA Model works with Microsoft Baseline Configuration Analyzer v2.0 (MBCA2) to check specific aspects of your installation. To perform the checks, you run the Microsoft Baseline Configuration Analyzer on the Web Sites Controller.

List of Installation Checks

The following table shows the checks that are made for an installation of Windows Azure Pack: Web Sites and the severity levels that appear when the checks fail.

Installation Check	Severity
Check that there is more than one of each role	Warning
Check that there is at least one of each role	Error
Check that the servers have unique names	Error
Check that there are available spare workers for Reserved Instances	Warning
Check FSRM (File Server Resource Manager) configuration	Error
Check that the File Server role is accessible via WinRM	Error
Check that the metering and hosting SQL database connection strings are correct	Error
Check that the PowerShell module is installed	Error

To use the Web Sites MBCA2 Model

1. Install and configure Windows Azure Pack: Web Sites.
2. On the Web Sites Controller, download and install the Microsoft Baseline Configuration Analyzer v2.0 at http://www.microsoft.com/en-us/download/details.aspx?id=16475.
3. On the Web Sites Controller, download and install WebSitesMBCA2.msi from the Microsoft Download Center. This installs the Web Sites MBCA2 model, making the checks available in the Microsoft Baseline Configuration Analyzer.
4. Run MBCA2, which is normally located at "C:\Program Files\Microsoft Baseline Configuration Analyzer 2\MBCA.exe".
   a. For the product, select **Windows Azure Pack Web Sites**.
   b. Choose **Start Scan**.

After a delay, the scan results will be shown. Any condition reported as an Error is an issue that can cause Web Sites to malfunction.

Warning
Any condition reported as a Warning should be corrected immediately.

Important Notes

- The MBCA2 model is meant to run only on the Controller role. The results are reported only on the Controller.

- The Web Sites MBCA2 Model can help you debug issues with your environment and will therefore be most helpful and useful after installation and configuration of Windows Azure Pack: Web Sites are complete.
- To use the Web Sites MBCA2 Model, the Web Sites Roles do not have to be ready for load balancing. In fact, the Web Sites MBCA2 Model can help you find out why roles might be failing to get into the ready state for load balancing.

See Also

Deploy Windows Azure Pack: Web Sites

Configure Windows Azure Pack: Web Sites

This chapter provides information about additional post-provisioning configuration, including configuring the SSL Certificate Store, configuring IP SSL, and configuring shared certificates. For information on configuring source control, see Configure source control for Windows Azure Pack: Web Sites. For information on security best practices for Web Sites, see Windows Azure Pack: Web Sites Security Enhancements.

Configure the SSL Certificate Store

Configure the SSL Certificate Polling Interval by running the following PowerShell cmdlets on the controller.

```
Add-pssnapin WebHostingSnapin

Set-HostingConfiguration -CentralCertPollingInterval 300 -
CentralCertificateSChannelCleanupInterval 300
```

Configure IP SSL

If you want to enable tenant web sites to use IP-based SSL certificates, you must configure the Front Ends, the Controller, and optionally, a hardware load balancer to do so.

Note
SNI (Server Name Indication) SSL is enabled by default. To make it available to tenants, include it in the plans that you author in the Management Portal for Administrators.

To configure IP SSL

1. Bind the IP addresses that you want to use:
   a. On each Front End server, open the network management interface.
   b. Click **Internet Protocol Version 6 (TCP/IPv6)**, and then click **Properties**.

c. Click **Advanced** to open the Advanced properties.
d. Click **Add** to add the IP addresses.
e. Repeat these steps for Internet Protocol Version 4 (TCP/IPv4).

> **Tip**
> Each customer or web site that uses IP SSL needs to have an IP address on each front end server. Because this can become labor intensive, you may want to use a script to automate the binding of IP addresses.

2. Next, configure the Web Site cloud to use the IP addresses for IP SSL traffic.
   a. In the management portal for administrators, click **Web Site Clouds**, and then double-click the cloud that you want to configure.
   b. Click **Roles**, and then choose the front end server.
   c. Click **IP SSL**.
   d. Click **Add** to add the IP address range.
   e. Enter the start address and end address, and the click the checkmark.

   > **Note**
   > The IP address range must be unique for each front end server.

   f. Repeat these steps for both IPv4 and IPv6 addresses.

   Repeat these steps for each front end server in the web farm.
3. If you are using an upstream hardware load balancer to balance traffic to the front end servers, the final step is to edit the register and deregister callback scripts so the Web Site cloud can communicate with the load balancer to create the load balancer pools for a given IP address.

   The callback scripts are located on the Web Site cloud controller in the web farm, in the path C:\Program Files\IIS\Microsoft Web Hosting Framework\Scripts\Provision\Win.

   a. Edit the DNS-RegisterSSLBindings.ps1 script. This script is used any time a user creates or edits a web site that uses IP SSL.
      i. Use the `$bindings` to create a load balancer pool. You can use the `$hostname` as a key for tracking it.
      ii. Return the Virtual IP address assigned to the load balancer pool (using `$retval`).
   b. Edit the DNS-DeRegisterSSLBindings.ps1 script. This script is used any time a user removes IP SSL from their web site or deletes or de-provisions the web site.

      Pass back an empty value (using `$retval`).

Configure shared certificates

The Web Site service uses certificates to encrypt data between the Front End servers, the Publishers, and the Controller.

By default, Windows Azure Pack: Web Sites provides self-signed certificates so that your initial operations do not occur in clear text. Of course, self-signed certificates cause certificate warning messages and must not be used in a production environment.

In a production environment, three certificates are required for securing endpoints in the web sites farm:

- **Front End** - The Front End certificate is used for shared SSL and for source control operations and has a binding on 'all unassigned'. The Front End certificate must be a two-subject certificate.
- **Publisher** - The Publishing certificate secures FTPS and Web Deploy traffic.

You obtain these certificates from a Certification Authority (CA) and upload them through the Management Portal for Administrators. You provide the password for each certificate so that it can be deployed to the farm.

The default domain certificate

The default domain certificate is placed on the Front End role and is used by tenant web sites for wildcard or default domain requests to the web site farm. The default certificate is also used for source control operations.

This certificate needs to be in .pfx format and should be a two-subject wildcard certificate. This allows both the default domain and the scm endpoint for source control operations to be covered by one certificate:

- *.<*DomainName*>.com
- *.scm. <*DomainName*>.com

Tip
A two-subject certificate is sometimes called a Subject Alternative Name (SAN) certificate. One advantage of a two-subject certificate is that the purchaser only has to buy one certificate instead of two.

Specify the certificate for the default domain

1. In the Management Portal for Administrators, click **Web Site Clouds**, and then choose the cloud that you want to configure.
2. Click **Configure** to open the Web Site cloud configuration page.
3. In the **Websites Default Certificate** field, click the folder icon. The **Upload Default Website Certificate** dialog appears.
4. Browse to and upload the certificate that you want to use.
5. Enter the password for the certificate, and then click the checkmark. The certificate will be propagated to all Front End servers in the web farm.

The certificate for publishing

The certificate for the Publisher role secures the Web Deploy and FTPS traffic for web site owners when they upload content to their web sites.

In the Management Portal for Administrators, the **Configure** page for the web site cloud contains a **Publishing Settings** section where you view or configure the Web Deploy and FTP Deploy DNS entries.

The certificate for publishing needs to contain a subject that matches the Web Deploy DNS entry and a subject that matches the FTPS Deploy DNS entry.

Note
If you used wildcards in the default certificate, you can also use the default certificate for the publisher. However, providing a separate certificate is more secure.

Specify the certificate for publishing

1. In the Management Portal for Administrators, click **Web Site Clouds**, and then choose the cloud that you want to configure.
2. Click **Configure** to open the Web Site cloud configuration page.
3. In the **Publisher Certificate** field, click the folder icon. The **Upload Publisher Certificate** dialog appears.
4. Browse to and upload the certificate that you want to use.
5. Enter the password for the certificate, and then click the checkmark. The certificate will be propagated to all Publishing servers in the web farm.

Best practices for certificates

- Be sure that certificate subject matching is correct. Windows Azure Pack: Web Sites does not allow certificates to be uploaded if there are mismatches.
- The most secure setup is to have separate certificates and separate domains. This helps defend against phishing scenarios and social engineering attacks.
- Watch for certificate expiration. Refresh certificates on a somewhat regular basis.
- For information about replacing untrusted Self-Signed Certificates with trusted certificates in Windows Azure Pack itself, see **Post-installation best practices** in the **Deploy Windows Azure Pack for Windows Server** guide.

See Also

Deploy Windows Azure Pack: Web Sites

Configure source control for Windows Azure Pack: Web Sites

The Web Site service supports using the following applications as source control repositories for web sites:

- Bitbucket

- GitHub
- CodePlex
- Dropbox

Before you can use one of these applications, you need to configure Windows Azure Pack: Web Sites with access information.

To configure source control

1. In the management portal for administrators, click **Web Site Clouds** to open the Web Site Clouds dashboard.
2. Click **Source Control**.
3. Enter the information for the source control application you are using.

While these applications are not part of the Windows Azure Pack: Web Sites itself, the following information helps you gather the data that you need from the applications in order to configure the Web Sites cloud controller.

Bitbucket

Use the following steps for Bitbucket.

1. Create an account at https://bitbucket.org.
2. Create a project.
3. Go to https://bitbucket.org/account/user/<user name>/api and add an Oauth consumer.

   For the URL, enter the URL for the management portal for tenants.
4. After you add the consumer, you will see the client ID and client secret. This is the information you need for the Web Sites cloud controller.

GitHub

Use the following steps for GitHub.

1. Create an account at https://github.com.
2. Go to https://github.com/applications/settings and click **Register new application**.
3. For the URL, enter the URL for the management portal for tenants.
4. For the Callback URL, enter **<tenantURL>/websites/GitHubTokenAuthorize**.
5. Click **Register application**.
6. After you register the application, you will see the client ID and client secret. This is the information you need for the Web Sites cloud controller.

Codeplex

Use the following steps for Codeplex.

1. Create an account at https://www.codeplex.com.
2. Create a new project.

3. Go to http://www.codeplex.com/site/developers/apps and create an application.

   For the URL and callback URL, enter the URL for the management portal for tenants.
4. After you create the application, you will see the client ID and client secret. This is the information you need for the Web Sites cloud controller.

Dropbox

Use the following steps for Dropbox.

1. Register at http://www.dropbox.com.
2. Go to https://www.dropbox.com/developers.
3. Click **Apps Console**, and then click **Create an app**.
4. Enter an app name and domains, and then click **Create app**.
5. After you create the application, you will see the app key, app secret, and app folder. This is the information you need for the Web Sites cloud controller.

   The app key corresponds to the client ID, and the app secret corresponds to the client secret.

See Also

Deploy Windows Azure Pack: Web Sites

Plan Authoring for Windows Azure Pack: Web Sites

This page describes procedures and settings are specific to plan authoring for Windows Azure Pack: Web Sites. For general information about authoring plans and add-ons in Windows Azure Pack, see **Administer plans and add-ons**.

Web Sites Plans: Essential Points

- After setting up Windows Azure Pack: Web Sites, you can create sets of offerings (or “plans”) for the tenant customers who will use your web site hosting service.
- You can create as many plans as you want and name them as you wish.
- Each plan that you create will have three levels of service: **Intro (Shared)**, **Basic (Shared)**, and **Reserved**.
- For each of the three levels, you can configure 17 different quotas related to web site features and capacities.
- The combination of quotas that you set for each level of service determines the restrictions and benefits that each level of service in the plan will have.
- You can add other services to the plan besides Web Sites clouds, but those choices are not covered here. For more information, see **Administer plans and add-ons**.

The sections that follow present the steps for authoring and configuring a plan for Windows Azure Pack: Web Sites.

To create a plan for Windows Azure Pack: Web Sites

1. In the Windows Azure Pack management portal for administrators, choose **Plans** in the left pane, and then click **New**.
2. In the **Author a Hosting Plan** dialog, provide a name for the plan, and then click the right arrow.
3. In the **Plan Services** dialog, select **Web Site Cloud**, and then choose a web site cloud from the list on the right. (You can also choose other services at this point, but for Web Site hosting, you should choose at least one web site cloud.)
4. In the **Plan Add-Ons** dialog, you have the option to include add-ons in your plan if you have made add-ons available. For more information, see **Administer plans and add-ons**.
5. Click the checkmark to finish.

To configure a Web Sites plan

1. In the Windows Azure Pack management portal for administrators, choose **Plans** in the left pane, and then choose the name of the plan that you created. The plan's dashboard appears.
2. On the plan dashboard, there are options in the command bar to **Change Access** to make the plan public (it is private by default), **Clone** or **Delete** your plan, **Link** an add-on, and **Add** or **Remove** a service. For more information on these features, see **Administer plans and add-ons**.
3. Click the name of the web site cloud that your plan is using. The next page presents the list of quotas for the web site cloud that you can configure for the **Intro (Shared)**, **Basic (Shared)**, and **Reserved** service levels.
4. Click the name of a quota to select it, and then choose **Edit** in the command bar at the bottom of the page to edit it.
5. In the **Quota** dialog, specify the values that you want to enforce for each service level. Note that:
   - Some quotas have an **Enforcement Duration (Minutes)** setting.
   - Some quotas have an **Exceed Action** setting, for which the options are **Do nothing**, **Stop web site**, or **Redirect to parking page**.

All of the quotas are described in the section that follows.

Configurable Quotas for Windows Azure Pack Web Site Plans

Quota Name	Description
Subscription CPU Time	Represents, in minutes, the CPU time a subscription's websites can consume across all instances over a specified enforcement period. You can specify a value or choose the **Unlimited** option. The default for each service level is **Unlimited**. There is an **Enforcement Duration (Minutes)** setting, for which the default is 1440 (=24 hours). The **Exceed Action** setting is available.
Subscription CPU Burst Time	Represents, in minutes, the CPU time a subscription's websites can consume across all instances over a specified enforcement period. A shorter enforcement duration can help reduce CPU spikes. You can specify a value or choose the **Unlimited** option. The default for each service level is **Unlimited**. The default **Enforcement Duration (Minutes)** setting is 1440 (=24 hours). The **Exceed Action** setting is available.
Process CPU Burst %	Represents the CPU percentage a worker process can consume over a specified enforcement period. The default for each service level is 100 percent. The default **Enforcement Duration (Minutes)** setting is 10 minutes.
Subscription Memory – Maximum Working Set	Represents, in megabytes, the physical memory (RAM) a subscription's websites can consume over a specified enforcement duration. You can specify a value in megabytes or choose the **Unlimited** option. The default for each service level is **Unlimited**. The default **Enforcement Duration (Minutes)** setting is 1440 (=24 hours). The **Exceed Action** setting is available.
Process Memory Limit	Represents the total memory a worker process can consume. The default for each service level is 1024 (=1 gigabyte).
Process Memory – Maximum Working Set	Represents, in megabytes, the physical memory (RAM) a worker process can consume.

Quota Name	Description
	The default for each service level is 1024 (=1 gigabyte).
Subscription Bytes In	Represents, in megabytes, the incoming bandwidth a subscription's websites can consume over a specified enforcement period. You can specify a value in megabytes or choose the **Unlimited** option. The default for each service level is **Unlimited**. The default **Enforcement Duration (Minutes)** setting is 1440 (=24 hours). The **Exceed Action** setting is available.
Subscription Bytes Out	Represents, in megabytes, the outgoing bandwidth a subscription's websites can consume over a specified enforcement period. You can specify a value in megabytes or choose the **Unlimited** option. The default for each service level is **Unlimited**. The default **Enforcement Duration (Minutes)** setting is 1440 (=24 hours). The **Exceed Action** setting is available.
Subscription Storage Space	Represents, in megabytes, the storage space a subscription's websites can consume. You can specify a value in megabytes or choose the **Unlimited** option. The **Shared** setting configures both the Intro (Shared) and Basic (Shared) service levels. There is a separate setting for **Reserved**. The default for each service level is 1024 (=1 gigabyte).
Subscription Site Count	Represents the number of websites in a subscription. You can specify a number or choose the **Unlimited** option. **Unlimited** is the default for each service level.
Subscription Web Worker Count	Represents the number of web workers a subscription's websites can consume simultaneously. You can specify a number or choose the **Unlimited** option. **Unlimited** is the default for each service level.
Subscription Custom Domain Support	Enables or disables custom domain names. **Off** is the default setting for Intro (Shared). **On** is

Quota Name	Description
	the default setting for Basic (Shared) and Reserved.
Subscription SSL Support	Enables or disables custom SSL certificates. Possible values are **Off**, **SNI**, **SNI and IPv4**, **SNI and IPv6**, and **SNI, IPv4 and IPv6**. The default for Intro (Shared) is **Off**. The default for Basic (Shared) is **SNI**. The default for Reserved is **SNI and IPv4**.
Subscription 64 bit Worker Process Support	Enables or disables 64 bit worker processes. **Off** is the default for Intro (Shared) and Basic (Shared). **On** is the default for Reserved.
Subscription WebSocket Support	Enables or disables the WebSocket protocol. **Off** is the default for Intro (Shared) and Basic (Shared). **On** is the default for Reserved.
Process Concurrent Request Limit	Represents the number of concurrent requests permitted per worker process. You can specify the number of connections. The default is 5000 for each service level.
Process Idle Timeout	Represents, in minutes, the amount of time that a worker process is allowed to remain idle before it is stopped. The default for Intro (Shared) is 20 minutes. The default for Basic (Shared) is 60 minutes. The default for Reserved is 10080 (=7 days).

See Also

Deploy Windows Azure Pack: Web Sites

Administer plans and add-ons

Windows Azure Pack: Web Sites Security Enhancements

After installation, you can enhance your security by implementing additional best practices. These include configuring IP filtering (also known as "blacklisting"), setting quotas to counter DoS attacks, and other steps.

Configure IP filtering

Setting an IP filter is very important because one of the easiest ways to launch a denial of service (DoS) attack is to launch the attack from inside the service itself. Therefore, at minimum, the hosting service provider should blacklist the farm from itself.

For example, if the web farm is deployed to a subnet, then the subnet IP addresses should be filtered to prevent web sites from calling back into the farm and launching (for example) a DoS attack.

To restrict tenant worker processes from accessing the IP address ranges corresponding to servers inside the Web Site cloud, you can configure IP filtering either in the Windows Azure Pack management portal or by using PowerShell.

To configure IP filtering in the Management Portal

To configure IP filtering in the Management Portal for Administrators, perform the following steps:

1. In the left pane of the portal, choose **Web Site Clouds**.
2. Select the web site cloud that you want to configure.
3. Choose **Block List**.
4. In the command bar at the bottom of the portal, choose **Add**.
5. In the **Enter an IP Address Range** dialog, enter an IP address in the **Start Address** and **End Address** boxes to create the range.
6. Click the check mark to complete the operation.

To configure IP filtering by using PowerShell

To configure IP filtering by using PowerShell, run the following PowerShell cmdlets on the controller. Replace *<start-of-ip-blacklist-range>* and *<end-of-ip-blacklist-range>* with valid IP addresses.

```
Add-pssnapin WebHostingSnapin

Set-Hostingconfiguration -WorkerRegKeyRejectPrivateAddresses 1

Set-Hostingconfiguration –WorkerRegKeyPrivateAddressRange <start-of-ip-blacklist-range>,
<end-of-ip-blacklist-range>
```

Restart the Dynamic WAS Service

Finally, restart the Dynamic WAS Service (DWASSVC) on servers configured to run the web worker role. Run the following commands from an elevated command prompt:

```
net stop dwassvc

net start dwassvc
```

Set Quotas

To prevent Denial of Service (DoS) attacks, you should set quotas on CPU, memory, bandwidth and disk usage. These quotas can be configured in the Management Portal for Administrators as part of plan authoring.

When a plan has these quotas set and a web site belonging to the plan suffers a DoS attack or an inadvertent CPU spike, the web site will be stopped when the quotas are reached, thus stopping the attack.

The quotas mentioned are also helpful against attacks originating from within the farm. For example, a brute password attack from within the farm would consume a lot of CPU time and, assuming strong passwords are used, the CPU quota would be reached before the password could be broken.

Assign a separate set of credentials for each Web Sites role

As a security best practice after installation, you should edit the credential sets for the Web Server roles so that they are all unique. After creating new, unique accounts, you can update the credentials in the Management Portal for Administrators to use the new accounts.

To edit Web Sites server role credentials

1. In the Management Portal for Administrators, click **Web Site Clouds**, and then choose the cloud that you want to configure.
2. Click **Credentials**. Under **User Name**, you can verify whether the user names are unique among the Web Site roles (for example, they might all be 'Administrator', in which case they should be changed).
3. Select one of the credential names (for example, **Management Server Credential**), and then click **Edit** in the command bar at the bottom of the portal.
4. In the dialog box that appears (for example, **Update Management Server Credential**), provide a new user name and password.
5. Click the checkmark to complete the operation.
6. Repeat steps 3 through 5 until all credential sets are unique.

Change ("roll") credentials on a regular basis

As a security best practice, it is a good idea to change (or "roll") credentials on a regular basis. For the role services, it’s better to change both user name and password at the same time, not just the password. Changing both the user name and password avoids the "out of sync" problem that can occur when only the password is changed, but the change hasn’t been propagated completely throughout the environment.

When you change both username and password, both sets of credentials are temporarily available during the rollover process. For example, two disconnected systems that need to

authenticate can still connect after the change. When the new credentials are in place and fully functioning on all systems, you can disable the old set.

Define a restrictive trust profile for .NET applications

For .NET applications, you should define a restrictive trust profile. By default, Windows Azure Pack: Web Sites runs in Full Trust mode to provide the broadest application compatibility possible. Choosing the optimal trust level involves a security versus compatibility tradeoff. Because each usage scenario is different, you should determine and follow your own best practices in securing the multi-tenant web servers in your environment.

Other Best Practices

Other Best Practices include using the principle of least privilege when creating user accounts, minimizing your network surface area, and modifying system ACLs to protect your file system and registry.

When creating accounts, use the principle of least privilege

When you create user accounts, apply the principle of least privilege to them. For more information, see Applying the Principle of Least Privilege to User Accounts on Windows.

Minimize your network surface area

Configure your firewall to minimize network surface area on internet facing servers. For information on Windows Firewall with Advanced Security, see the following resources (the references for Windows Server 2008 R2 are still useful for Windows Server 2012 and Windows Server 2012 R2).

- Windows Server 2008 R2 Step-by-Step Guide: Deploying Windows Firewall and IPsec Policies (Microsoft document download link)
- Windows Server 2008 R2 Firewall Security (WindowsITPro)
- Troubleshooting Windows Firewall with Advanced Security in Windows Server 2012 (TechNet)

Modify system ACLs to secure the file system and registry

The following downloadable utilities can help evaluate a server's file system and registry security settings.

- AccessChk
- AccessEnum

See Also

Deploy Windows Azure Pack: Web Sites

Scaling Windows Azure Pack: Web Sites for High Availability

To achieve High Availability in your Windows Azure Pack: Web Sites cloud, you can provision additional instances of each Web Sites role. Additional instances of Web Workers, Front Ends, and Publishers can be easily provisioned in the Management Portal for Administrators. Additional Management Servers and a second Web Sites controller can be provisioned through command line and/or PowerShell scripting. In addition, you can take steps to increase the robustness of the SQL Servers in your environment.

For information on adding a second Web Sites controller, see Provision a Second Web Sites Controller. The other topics are discussed here.

Create additional Web Worker, Front End, or Publisher instances

To create another Web Worker, Front End, or Publisher instance, perform the following steps in the Management Portal for Administrators.

1. In the left pane of the portal, choose **Web Site Clouds**.
2. Select the web site cloud that you want to configure.
3. Choose **Roles**.
4. In the command bar at the bottom of the portal, choose **Add Role**.
5. In the **Add Server** dialog, choose **Add New Web Worker**, **Add New Front End**, or **Add New Publisher**.
6. In the Setup dialog that follows, provide the hostname or IP address of the server. For a new Web Worker, specify the **Worker Type** (Shared or Reserved Small/Medium/ Large).
7. Click the check mark to complete the operation.

Provision Additional Management Servers

To provision an additional Management Server, run the following PowerShell commands on the Controller role. Replace <*NewManagementServer*> with the name of a newly prepared Windows Server 2012 R2 server or virtual machine.

```
Import-Module Websites


New-WebSitesServer -Name <NewManagementServer> -ServerType ManagementServer
```

Configuring SQL Server for High Availability

Microsoft SQL Server 2012 Enterprise edition allows you to use AlwaysOn Availability Groups, which reduce the dependency of the database on a single SQL instance. You can use Availability Groups together with Failover Clustering for even greater reliability. For more information on AlwaysOn Availability Groups, see: **Configure SQL AlwaysOn Availability Groups in Windows Azure Pack**. For background information, see Overview of AlwaysOn Availability Groups (SQL Server). For information on Failover Clustering, see Failover Clustering Overview.

See Also

Deploy Windows Azure Pack: Web Sites

Provision a Second Web Sites Controller

Windows Azure Pack: Web Sites can have a maximum of two Web Sites Controllers. For High Availability, it is strongly recommended that you exercise this maximum and provision a second Web Sites Controller. This can be done by using the following scripts, which are provided here:

OnStartSecondaryController.cmd

HostingBootstrapperBootstrapper.ps1

OnStartSecondaryController.ps1

Common.ps1

Descriptions

- **OnStartSecondaryController.cmd** - Installs the WebPlatform Installer (Web PI) on the Windows 2012 server or Virtual Machine that you have prepared. It then calls the HostingBootstrapperBootstrapper and OnStartSecondaryController scripts in succession to complete the provisioning of the secondary controller. Required parameters include the SQL Server and Controller administrative credentials that will be used in your setup.
- **HostingBootstrapperBootstrapper.ps1** - Calls Program Files\Microsoft\Web Platform Installer\WebpiCmd.exe to install the second Web Sites Controller. It uses the same procedure as when installing the primary controller, but discards the configuration portal by specifying the command line switch /SuppressPostFinish.
- **OnStartSecondaryController.ps1** - Copies configuration data from the primary controller to the secondary controller, including the **SystemCore** and **SiteRuntime** keys and the hosting and resource metering connection strings. When finished, it starts the WebFarmService.
- **Common.ps1** - Provides supporting functions for the OnStartSecondaryController.ps1 script.

Important
These scripts require that Windows Remote Management (WinRM) be enabled on the primary controller.

Steps to Run the Scripts

1. Copy the script files to a folder on the server that will be the Secondary Controller.
2. Copy the WebPlatformInstaller.msi file to the same folder as the scripts. This is necessary because the OnStartSecondaryController.cmd script automates the installation of Web PI.
3. Run **OnStartSecondaryController.cmd** with administrator rights, supplying the parameters described in the table below. Administrative privileges are required so that the appropriate products can be installed through Web PI automation and the primary controller can be accessed through WinRM.

   
   Note
   In a WORKGROUP scenario, you might need to enable WinRM on the primary controller, or run the commands in the OnStartSecondaryController.cmd script manually.

OnStartSecondaryController.cmd

Syntax

```
OnStartSecondaryController.cmd -feed %Feed% -webSitesInstanceName %WebSitesInstanceName%
-sqlservername %DatabaseServerName% -sqlsysadmin %DatabaseSysAdminAccount% -
sqlsysadminpwd %DatabaseSysAdminPassword% -controllerAdminUserName
%ControllerAdminUserName% -controllerAdminPassword %ControllerAdminPassword%
```

Parameters

Parameter Name	Description	Notes
feed	Optionally specifies the Web PI feed that will be used to install the controller.	If this parameter is not specified, the default Web PI primary feed is used.
webSitesInstanceName	Used as prefix for database objects	Must match the prefix name of the database in the installation.
sqlservername	Name of the instance of SQL server	
sqlsysadmin	SQL Server sys admin user account name	Needs to be member of sysadmin server roles.
sqlsysadminpwd	SQL Server sys admin user account password	

controllerAdminUserName	Web Farm Framework (WFF) Administrator account name to provision	If this is a domain account, it will be added to the local administrators group. If the server is in WORKGROUP, it will try to create the user if the user doesn't exist and then add it to the local administrators group.
controllerAdminPassword	WFF Administrator account password to provision	

OnStartSecondaryController.cmd Script

```
@echo off

echo Starting OnStartSecondaryController.cmd


rem ---------------------------------------------

rem Initialize Variables

rem ---------------------------------------------

    set POWERSHELL=%windir%\System32\WindowsPowerShell\v1.0\powershell.exe


    set FEED=

    set INSTANCE_NAME=

    set SQL_SERVERNAME=

    set SQL_SYSADMIN=

    set SQL_SYSADMINPWD=

    set CONTROLLER_ADMIN_USERNAME=

    set CONTROLLER_ADMIN_PASSWORD=


rem ---------------------------------------------

rem Parse command line parameters

rem ---------------------------------------------

:parse_param

    set PARAM_MATCHED=0

    if "%1"=="" (
```

```
    goto :parse_param_completed
)

if /I "%1"=="-feed" (
    set FEED=%2
    shift
    set PARAM_MATCHED=1
)

if /I "%1"=="-webSitesInstanceName" (
    set INSTANCE_NAME=%2
    shift
    set PARAM_MATCHED=1
)

if /I "%1"=="-sqlservername" (
    set SQL_SERVERNAME=%2
    shift
    set PARAM_MATCHED=1
)

if /I "%1"=="-sqlsysadmin" (
    set SQL_SYSADMIN=%2
    shift
    set PARAM_MATCHED=1
)

if /I "%1"=="-sqlsysadminpwd" (
    set SQL_SYSADMINPWD=%2
    shift
    set PARAM_MATCHED=1
)

if /I "%1"=="-controllerAdminUserName" (
```

```
        set CONTROLLER_ADMIN_USERNAME=%2
        shift
        set PARAM_MATCHED=1
    )

    if /I "%1"=="-controllerAdminPassword" (
        set CONTROLLER_ADMIN_PASSWORD=%2
        shift
        set PARAM_MATCHED=1
    )

    if "%PARAM_MATCHED%"=="0" (
       echo Parameter %1 was not matched
       exit 1
    )

    shift
    goto :parse_param
:parse_param_completed

rem ----------------------------------------------------------------------
rem Provision Controller
rem ----------------------------------------------------------------------

    echo Installing WebPlatformInstaller.
    start /wait %windir%\system32\msiexec.exe /qn /i WebPlatformInstaller.msi /l
%SystemDrive%\WebPlatformInstaller.log
    if errorlevel 1 (
        echo WebPlatform Installer installation failed. See log at
%SystemDrive%\WebPlatformInstaller.log for more details.
        exit 1
    )

    echo WebPlatformInstaller setup completed successfully.
```

```
    echo Enabling remote desktop access.
    start /wait cscript %windir%\system32\scregedit.wsf /ar 0
    if errorlevel 1 (
        echo Enabling remote desktop failed.
        exit 1
    )

    echo Remote desktop access has been enabled successfully.

    if exist StrongNameHijack.msi (

        echo Installing StrongNameHijack...
        start /wait %windir%\system32\msiexec.exe /qn /i StrongNameHijack.msi /l
%SystemDrive%\StrongNameHijack.log
        if errorlevel 1 (
            echo "StrongNameHijack installation failed. See log at
%SystemDrive%\StrongNameHijack.log for more details."
            exit 1
        )

        echo StrongNameHijack setup completed successfully.

        net stop msiserver & net start msiserver
    )

    echo Starting HostingBootstrapperBootstrapper.ps1

    if "%FEED%"=="" (
        %POWERSHELL% -ExecutionPolicy Unrestricted -File
HostingBootstrapperBootstrapper.ps1
    ) else (
        %POWERSHELL% -ExecutionPolicy Unrestricted -File
HostingBootstrapperBootstrapper.ps1 -mainFeed "%FEED%"
```

```
    )

    if errorlevel 1 (
        echo HostingBootstrapperBootstrapper.ps1 failed.
        exit 1
    )

    echo HostingBootstrapperBootstrapper.ps1 completed successfully.

    echo Starting OnStartSecondaryController.ps1

    %POWERSHELL% -ExecutionPolicy Unrestricted -File OnStartSecondaryController.ps1 -
webSitesInstanceName "%INSTANCE_NAME%" -sqlservername "%SQL_SERVERNAME%" -sqlsysadmin
"%SQL_SYSADMIN%" -sqlsysadminpwd "%SQL_SYSADMINPWD%" -controllerAdminUserName
"%CONTROLLER_ADMIN_USERNAME%" -controllerAdminPassword "%CONTROLLER_ADMIN_PASSWORD%"
    if errorlevel 1 (
        echo OnStartSecondaryController.ps1 failed.
        exit 1
    )

    echo OnStartSecondaryController.ps1 completed successfully.

echo OnStartSecondaryController.cmd completed successfully.

exit 0
```

HostingBootstrapperBootstrapper.ps1

```
# PowerShell script to setup Web Sites Controller using WebPI.
# Copyright (c) Microsoft Corporation. All rights reserved.

Param
(
    [string] $boostrapperProductId = "HostingPrimaryControllerBootstrapper_v2",
    [string] $mainFeed = "",
```

```
    [string] $customFeed = ""
)

# Change Error Action to Stop
$ErrorActionPreference="Stop"

Function BootstrapBootstrapper ()
{
    $WebPiCmd =
[System.Environment]::ExpandEnvironmentVariables("%ProgramW6432%\Microsoft\Web Platform
Installer\WebpiCmd.exe")
    $WebPiLog =
[System.Environment]::ExpandEnvironmentVariables("%SystemDrive%\HostingPrimaryControllerB
ootstrapper.log")

    If ($mainFeed -eq "")
    {
        Invoke-Command -ScriptBlock { & $WebPiCmd /Install
/Products:$boostrapperProductId /AcceptEula /SuppressReboot /SuppressPostFinish
/Log:$WebPiLog }
    }
    Else
    {
        If ($customFeed -eq "")
        {
            Invoke-Command -ScriptBlock { & $WebPiCmd /Install
/Products:$boostrapperProductId /AcceptEula /SuppressReboot /SuppressPostFinish
/XML:$mainFeed /Log:$WebPiLog }
        }
        Else
        {
            Invoke-Command -ScriptBlock { & $WebPiCmd /Install
/Products:$boostrapperProductId /AcceptEula /SuppressReboot /SuppressPostFinish
/XML:$mainFeed /Feeds:$customFeed /Log:$WebPiLog }
        }
```

```
    }

    If ($lastexitcode -ne $Null -And $lastexitcode -ne 0)
    {
        Exit $lastexitcode
    }
}

# Entry Point
BootstrapBootstrapper
```

OnStartSecondaryController.ps1

```
# PowerShell script to setup a Web Sites secondary controller.
# Copyright (c) Microsoft Corporation. All rights reserved.

Param
(
    [string] $webSitesInstanceName,
    [string] $sqlservername,
    [string] $sqlsysadmin,
    [string] $sqlsysadminpwd,
    [string] $controllerAdminUserName,
    [string] $controllerAdminPassword
)

# Init Global Variables
$cnstr =
"server=$($sqlservername);database=$($webSitesInstanceName)_hosting;uid=$($sqlsysadmin);p
wd=$($sqlsysadminpwd);"

# Load common script file
$startDir = "."
$common = [System.IO.Path]::Combine($startDir, "Common.ps1")
. $common
```

```
Function Get-PrimaryController()
{
    Try
    {
        $siteManager = New-Object Microsoft.Web.Hosting.SiteManager $cnstr
        $primaryController =
$siteManager.Controllers.GetPrimaryController([Microsoft.Web.Hosting.PlatformOptions]::Vi
rtualMachineManager)

        Return $primaryController.MachineName;
    }
    Finally
    {
        If($siteManager -ne $Null)
        {
            $siteManager.Dispose()
        }
    }
}

Function Test-PrimaryController()
{
    Try
    {
        $PrimaryController = Get-PrimaryController

        Return $PrimaryController -ne $Null
    }
    Catch [Microsoft.Web.Hosting.WebHostingObjectNotFoundException]
    {
        Write-Host "$(Get-Date): Primary Controller NOT Ready"

        Return $False;
```

```
    }
}

Function WaitForPrimaryControllerToBeReady()
{
    $WaitIndex=0;
    $MaxWait=15
    $WaitInterval=60000

    Write-Host "$(Get-Date): Waiting for Primary Controller to be ready"

    $valid = Test-PrimaryController
    If ($valid -eq $False)
    {
        While ($WaitIndex -lt $MaxWait -and $valid -eq $False)
        {
            [System.Threading.Thread]::Sleep($WaitInterval);
            $WaitIndex = $WaitIndex + 1
            $valid = Test-PrimaryController
        }
    }

    If ($valid -eq $False)
    {
        Throw New-Object [System.Exception] "Primary Controller NOT ready. Timed out
waiting."
    }

    Write-Host "$(Get-Date): Primary Controller is Ready"
}

Function Copy-SystemCoreKeyFromPrimaryController([string] $PrimaryController)
{
    Write-Host "$(Get-Date): Copy SystemCore key"
```

```
    $remoteCommand = {
        Add-PSSnapIn WebHostingSnapIn
        Get-WebSitesConfig -Type SecurityKey -SymmetricKeyName SystemCore
    }

    $SystemCoreKey = Invoke-Command -ComputerName $PrimaryController -ScriptBlock
$remoteCommand
    Set-WebSitesConfig -Type SecurityKey -SymmetricKeyName SystemCore -SymmetricKey
$SystemCoreKey -Force
}

Function Copy-SiteRuntimeKeyFromPrimaryController([string] $PrimaryController)
{
    Write-Host "$(Get-Date): Copy SiteRuntime key"

    $remoteCommand = {
        Add-PSSnapIn WebHostingSnapIn
        Get-WebSitesConfig -Type SecurityKey -SymmetricKeyName SiteRuntime
    }

    $SiteRuntimeKey  = Invoke-Command -ComputerName $PrimaryController -ScriptBlock
$remoteCommand
    Set-WebSitesConfig -Type SecurityKey -SymmetricKeyName SiteRuntime -SymmetricKey
$SiteRuntimeKey -Force
}

Function Copy-HostingConnectionStringFromPrimaryController([string] $PrimaryController)
{
    Write-Host "$(Get-Date): Copy hosting connection string"

    $remoteCommand = {
        Add-PSSnapIn WebHostingSnapIn
        [Microsoft.Web.Hosting.SiteManager]::GetDefaultConnectionString()
```

```
    }

    $HostingCnStr  = Invoke-Command -ComputerName $PrimaryController -ScriptBlock
$remoteCommand
    Set-WebSitesConnectionString -Type Hosting -ConnectionString $HostingCnStr
}

Function Copy-MeteringConnectionStringFromPrimaryController([string] $PrimaryController)
{
    Write-Host "$(Get-Date): Copy metering connection string"

    $remoteCommand = {
        Add-PSSnapIn WebHostingSnapIn
        [Microsoft.Web.Hosting.SiteManager]::GetMeteringConnectionString()
    }

    $computerName =
[Microsoft.Web.Hosting.Common.NetworkHelper]::GetComputerName([Microsoft.Web.Hosting.Comm
on.ComputerNameFormat]::DnsFullyQualified)

    $MeteringCnStr  = Invoke-Command -ComputerName $PrimaryController -ScriptBlock
$remoteCommand
    Set-WebSitesConnectionString -Type Metering -ConnectionString $MeteringCnStr -
ServerName $computerName
}

Function OnStartSecondaryController()
{
    Try
    {
        Write-Host "$(Get-Date): Starting OnStartSecondaryController()" -ForegroundColor
Green

        Add-PSSnapin WebHostingSnapin
```

```
        ConfigureRoleAdministrator $controllerAdminUserName $controllerAdminPassword

        WaitForHostingDatabaseToBeReady $cnstr
        WaitForPrimaryControllerToBeReady

        $PrimaryController = Get-PrimaryController
        Copy-SystemCoreKeyFromPrimaryController $PrimaryController
        Copy-SiteRuntimeKeyFromPrimaryController $PrimaryController
        Copy-HostingConnectionStringFromPrimaryController $PrimaryController
        Copy-MeteringConnectionStringFromPrimaryController $PrimaryController

        Start-Service WebFarmService

        Write-Host "$(Get-Date): OnStartSecondaryController completed successfully" -
ForegroundColor Green
    }
    Catch [System.Exception]
    {
        Write-Host "$(Get-Date): Exception encountered while executing
OnStartSecondaryController:" -ForegroundColor Red
        Write-Host $_ -ForegroundColor Red

        Write-Host "$(Get-Date): Exiting OnStartSecondaryController.ps1" -ForegroundColor
Red
        Exit -1
    }
}

# Entry Point
OnStartSecondaryController
```

Common.ps1

```
# PowerShell Web Sites common script.
# Copyright (c) Microsoft Corporation. All rights reserved.
```

```
Function LoadHostingFramework()
{
    $mwhc = Get-Item ".\Microsoft.Web.Hosting.Common.dll"
    [void] [System.Reflection.Assembly]::LoadFrom($mwhc.FullName)
    Write-Host "$(Get-Date): Microsoft.Web.Hosting.Common assembly was successfully
loaded from: $mwhc"

    $mwh  = Get-Item ".\Microsoft.Web.Hosting.dll"
    [void] [System.Reflection.Assembly]::LoadFrom($mwh.FullName)
    Write-Host "$(Get-Date): Microsoft.Web.Hosting assembly was successfully loaded from:
$mwh"
}

Function IsHostingDatabaseReady([string] $cnstr)
{
    Try
    {
        $siteManager = New-Object Microsoft.Web.Hosting.SiteManager $cnstr

$siteManager.TestConnection([Microsoft.Web.Hosting.PlatformOptions]::VirtualMachineManage
r)

        Return $true;
    }
    Catch [Exception]
    {
        Write-Host "$(Get-Date): Hosting Database NOT Ready"

        Return $false;
    }
    Finally
    {
        If($siteManager -ne $Null)
```

```
        {
            $siteManager.Dispose()
        }
    }
}

Function WaitForHostingDatabaseToBeReady ([string] $cnstr)
{
    $WaitIndex=0;
    $MaxWait=120
    $WaitInterval=60000

    Write-Host "$(Get-Date): Waiting for Hosting Database to be ready"

    $ready = IsHostingDatabaseReady $cnstr
    If($ready -ne $true)
    {
    While ($WaitIndex -lt $MaxWait -and $ready -ne $true)
        {
            [System.Threading.Thread]::Sleep($WaitInterval);
            $WaitIndex = $WaitIndex + 1
            $ready = IsHostingDatabaseReady $cnstr
        }
    }

    If ($ready -ne $true)
    {
        Throw New-Object [System.Exception] "Hosting Database NOT ready. Timed out
waiting."
    }

    Write-Host "$(Get-Date): Hosting Database is Ready"
}
```

```
Function ConfigureRoleAdministrator([string] $roleadminusr, [string] $roleadminpwd)
{
    $isDomain = $false

    # Identify if user is a domain user account
    $values = $roleadminusr.Split('\');
    If ($values.Length -eq 1)
    {
        $domain = $env:COMPUTERNAME
        $username = $values[0]
    }
    ElseIf ($values.Length -eq 2)
    {
        If ([String]::Equals($values[0], ".") -Or
            [String]::Equals($values[0], [Environment]::MachineName,
[StringComparison]::OrdinalIgnoreCase))
        {
            $isDomain = $false
            $domain = $env:COMPUTERNAME
            $username = $values[1]
        }
        Else
        {
            $isDomain = $true
            $domain = $values[0]
            $username = $values[1]
        }
    }
    Else
    {
        Throw New-Object ArgumentException "Invalid user name" "roleadminusr"
    }

    # Create user if specified user is not a domain account
```

```
    if ($isDomain -eq $false)
    {
        Try
        {
            $computer = [ADSI]"WinNT://$env:COMPUTERNAME"
            $user = $computer.Create("User", $username)
            $user.setpassword($roleadminpwd)
            $user.SetInfo()
        }
        Catch [System.Runtime.InteropServices.COMException]
        {
            # User already exists
            If ($_.Exception.ErrorCode -eq -2147022672)
            {
                Write-Host "$(Get-Date): User $domain\$username already exits."
                Write-Host "$(Get-Date): Updating password for User $domain\$username."
                $user = [ADSI]("WinNT://$env:COMPUTERNAME/$username,user")
                $user.setpassword($roleadminpwd)
                $user.SetInfo()
            }
            Else
            {
                Write-Host "$(Get-Date): Error creating user $domain\$username." -
ForegroundColor Red

                Throw
            }
        }
    }

    # Add user to local administrators group

    #first translate the "Administrators" name to the name based on locale (make well
known SID -> Name translation)
```

```
    $administratorsSID = New-Object System.Security.Principal.SecurityIdentifier("S-1-5-
32-544")
    $administratorsGroupName =
$administratorsSID.Translate([System.Security.Principal.NTAccount]).Value

    # retrieve the short name eg in german translate VORDEFINIERT\Administratoren -
>Administratoren
    # the translation should always return fully qualified name equivalent to
BUILTIN\Administrators

    $localizedAdministratorsGroupName=$administratorsGroupName.Split("\\")[1]

    $adminGroup =
[ADSI]("WinNT://$env:COMPUTERNAME/$localizedAdministratorsGroupName,group")

    Try
    {
        If ($isDomain -eq $true)
        {
            $adminGroup.add("WinNT://$domain/$username,user")
        }
        Else
        {
            $adminGroup.add("WinNT://$env:COMPUTERNAME/$username,user")
        }
    }
    Catch [System.Runtime.InteropServices.COMException]
    {
        If ($_.Exception.ErrorCode -eq -2147023518) # ERROR_MEMBER_IN_ALIAS 1378 (0x562)
        {
            Write-Host "$(Get-Date): User $domain\$username is already member of
Administrators group."
        }
        Else
```

```
        {
            Write-Host "$(Get-Date): Error adding user $domain\$username to
Administrators group." -ForegroundColor Red

            Throw
        }
    }
}
```

See Also

Deploy Windows Azure Pack: Web Sites

Backing up Windows Azure Pack: Web Sites

Backing up Windows Azure Pack: Web Sites involves three major components: the Web Sites Controller, SQL Server, and the File Server. Links to the respective sections are provided below.

A. Web Sites Controller Backup

B. SQL Server Backup

C. File Server Backup

A. Web Sites Controller Backup

In order to back up the Web Sites Controller, you can use the **Backup.ps1** PowerShell script presented in this section. This script invokes the Windows Volume Shadow Service (VSS) writer to perform the backup.

Copy the Backup.ps1 script onto the Web Sites Controller, and then run the following command with Administrator privileges:

```
net use /Y $backupLocation /user:$backupMachineAdmin $backupMachinePassword

.\Backup.ps1 $backupLocation $encryptionKey
```

Note

The $encryptionKey flag is optional, but it is highly recommended as an added security precaution.

Warning

Do not forget the encryption key, because it is not stored for you in any way.

The Backup.ps1 script follows.

```
##
```

```
##  Script to backup the controller using the Hosting VSS writer
##

param (
[parameter(Position=2)]
$backupPath,
[parameter(Position=3)]
$passphrase
)

function ShowHelp
{
    Write-Host '==================== BACKUP.PS1 HELP ====================='
    Write-Host 'This is a script that uses the Hosting VSS writer and creates a backup of
the keys and offline feed'
    Write-Host 'Invoke it using .\Backup.ps1 and follow the prompts'
    Write-Host 'It can also be invoked as follows:'
    Write-Host '.\Backup.ps1 <Backup path> <passphrase to encrypt keys with>'
Write-Host "Note: before running this script you may need to run:`r`n   'net use /Y
<Backup path> /user:<username> <password>'"
    Write-Host '============================================================='
}

function CopyFiles
{
    # copy from the exposed location to where we're backing up to
    $commands = @()
    # $exposedDrive is the VSS shadow copy drive
    $commands += "'D' | xcopy /Y /q /E '${exposedDrive}:\$feedLocationNQ'
'$backupPath\$feedLocationNQ'"
    $commands += "'F' | xcopy /Y /q '${systemDrive}encryptedkeys.txt' '$backupPath'"
    # wrap each command in retry logic
    foreach ($command in $commands)
    {
```

```
        $final += ('$c = 0' +"`r`n")
        $final += ('do {'+"`r`n")
        $final += (' $c++' + "`r`n Start-Sleep -s 2`r`n ")
        $final += ($command + "`r`n")
        $final += '} while (!($?) -and $c -lt 10)'+"`r`n"
        $command = $command -replace "'", '"'
        $final += ('if($?)'+"{'Successfully executed: $command'}`r`n")
        $final += ("else{ 'There was a problem executing: $command'}`r`n")
    }
    $final | Set-Content "copyfiles.ps1"
}

function EncryptKeys($keysFile, $passphrase, $salt, $init, $systemDrive)
{
    $encryptscript = @"
function EncryptString(`$keysFile, `$passphrase, `$salt, `$init)
{
    `$ret = @()
    `$stringsToEncrypt = (Get-Content `$keysFile)
    foreach (`$stringToEncrypt in `$stringsToEncrypt)
    {
        `$r = new-Object System.Security.Cryptography.RijndaelManaged
        `$pass = [Text.Encoding]::UTF8.GetBytes(`$passphrase)
        `$salt = [Text.Encoding]::UTF8.GetBytes(`$salt)
        `$r.Key = (new-Object Security.Cryptography.PasswordDeriveBytes `$pass, `$salt,
'SHA1', 5).GetBytes(32) #256/8
        `$r.IV = (new-Object Security.Cryptography.SHA1Managed).ComputeHash(
[Text.Encoding]::UTF8.GetBytes(`$init) )[0..15]
        `$c = `$r.CreateEncryptor()
        `$ms = new-Object IO.MemoryStream
        `$cs = new-Object Security.Cryptography.CryptoStream `$ms,`$c,'Write'
        `$sw = new-Object IO.StreamWriter `$cs
        `$sw.Write(`$stringToEncrypt)
        `$sw.Close()
```

```
        `$cs.Close()
        `$ms.Close()
        `$r.Clear()
        [byte[]]`$result = `$ms.ToArray()
        `$ret += [Convert]::ToBase64String(`$result)
    }
    return `$ret
}

"@
    $encryptscript += "EncryptString '$keysFile' '$passphrase' '$salt' '$init' >
'${systemDrive}encryptedkeys.txt'"
    # $encryptscript += "`r`ndel ${systemDrive}keys.txt"
    $encryptscript | set-content "encryptkeys.ps1"
}

if ($backupPath -and $backupPath.Contains('/?'))
{
    ShowHelp
    return
}
Write-Host 'Starting the backup process. Run with /? to see help.'
Write-Host "Note: before running this script you may need to run:`r`n   'net use /Y
<backupPath> /user:<username> <password>'"
# argument parsing
if (!$backupPath)
{
    $backupPath = Read-Host "Please enter the fully qualified backup path (e.g.
\\backupmachine\C$\backuplocation)"
}
if (!$passphrase)
{
    $passphrase = Read-Host "Please enter a passphrase to encrypt keys (leave blank for
no encryption)"  -AsSecureString
```

```
if (!$passphrase)
{
$passphrase = ""
}
else
{
$passphrase =
[Runtime.InteropServices.Marshal]::PtrToStringAuto([Runtime.InteropServices.Marshal]::Sec
ureStringToBSTR($passphrase))
}
}

$usedDisks = ((Get-WmiObject -Class Win32_LogicalDisk).DeviceID|%{$_ -replace ':',''})
foreach ($l in ([char[]]([char]'a'..[char]'z')))
{
    if ($usedDisks -notcontains $l)
    {
        $exposedDrive = $l
        break
    }
}

$logfile = "backup.log"
$metadataLocation = 'metadata.cab'
# expand environment variables
$backupPath = ([System.Environment]::ExpandEnvironmentVariables($backupPath))
$systemDrive = [System.Environment]::ExpandEnvironmentVariables('%systemdrive%\')
$feedLocation = "${systemDrive}HostingOfflineFeed"
$feedLocation = ([System.Environment]::ExpandEnvironmentVariables($feedLocation))
$feedLocationNQ = Split-Path $feedLocation -NoQualifier
$feedLocationNQ = $feedLocationNQ.TrimStart('\')
$letterLocation = Split-Path $feedLocation -Qualifier
$letterLocation = $letterLocation -replace ':',''
# create powershell scripts
```

```
EncryptKeys "${systemDrive}keys.txt" $passphrase "salt12345" "init12345" $systemDrive
CopyFiles
# backup using diskshadow
$diskshadowScript += "set context persistent`r`n"
$diskshadowScript += "set metadata ${letterLocation}:\${metadataLocation}`r`n"
$diskshadowScript += "begin backup`r`n"
$diskshadowScript += "add volume ${feedLocation} alias ${feedLocationNQ}`r`n"
$diskshadowScript += "writer verify {079462f1-1079-48dd-b3fb-ccb2f2934ecf}`r`n"
$diskshadowScript += "create`r`n"
# copy files
$diskshadowScript += "expose %${feedLocationNQ}% ${exposedDrive}: `r`n"
$diskshadowScript += "exec ${env:windir}\System32\WindowsPowerShell\v1.0\powershell.exe
.\encryptkeys.ps1`r`n"
$diskshadowScript += "exec ${env:windir}\System32\WindowsPowerShell\v1.0\powershell.exe
.\copyfiles.ps1`r`n"
$diskshadowScript += "unexpose %${feedLocationNQ}%`r`n"
$diskshadowScript += "end backup`r`n"
$diskshadowScript += "delete shadows all`r`n"
$diskshadowScript += "exit`r`n"
$diskshadowScript | Set-Content "diskshadow1.txt"
write-host "====================== BEGINNING BACKUP ====================="
diskshadow /s "diskshadow1.txt" > $logfile
write-host "====================== BACKUP COMPLETE ======================"
write-host "======================== CLEANING UP ========================"
#  CLEAN UP
del ${letterLocation}:\${metadataLocation} # metadata.cab
del "diskshadow1.txt"
write-host "====================== DONE CLEANING UP ====================="
write-host "================ SEE BACKUP.LOG FOR DETAILS ================"
del "copyfiles.ps1"
del "encryptkeys.ps1"
del "${systemDrive}encryptedkeys.txt"
del "${systemDrive}keys.txt"
```

B. SQL Server Backup

In backing up SQL Server, you must back up the Hosting database, the Resource Metering database, and the master database. Because each user's SQL environment is different, no one script can fit every user's requirements. The following example script is provided for illustrative purposes only and is unsupported. The script that you create must be run with administrative privileges.

Sample SQL Server Backup Script

Note

This script is not supported by Microsoft.

```
param ([string] $backupUser = "Administrator", $backupPassword, $sqlServer, $sqlUser =
"sa", $sqlPassword, $backupLocation = "\\backupMachine\c$\Backup")

sqlcmd -S $sqlServer -U $sqlUser -P $sqlPassword -Q "BACKUP DATABASE [Hosting] TO
DISK='C:\HostingOfflineFeed\Hosting.bak'"

sqlcmd -S $sqlServer -U $sqlUser -P $sqlPassword -Q "BACKUP DATABASE [ResourceMetering]
TO DISK='C:\HostingOfflineFeed\ResourceMetering.bak'"

sqlcmd -S $sqlServer -U $sqlUser -P $sqlPassword -Q "BACKUP DATABASE [master] TO
DISK='C:\HostingOfflineFeed\master.bak'"

net use $backupLocation /user:$backupUser $backupPassword

xcopy /Y /q C:\HostingOfflineFeed\Hosting.bak $backupLocation\

xcopy /Y /q C:\HostingOfflineFeed\ResourceMetering.bak $backupLocation\

xcopy /Y /q C:\HostingOfflineFeed\master.bak $backupLocation\

del C:\HostingOfflineFeed\Hosting.bak

del C:\HostingOfflineFeed\ResourceMetering.bak

del C:\HostingOfflineFeed\master.bak
```

C. File Server Backup

When backing up the File Server, you must back up the Certificate share, the WebSites share, the ACLs for the previously stated folders, and the File Server Resource Manager (FSRM) quotas for the WebSites share.

Because each user's File Server environment is different, no one script can fit every user's requirements. The following example scripts are provided for illustrative purposes only and are unsupported. The scripts that you create must be run with administrative privileges.

Sample File Server Backup Script

Note

This script is not supported by Microsoft.

```
param ([string] $backupUser = "Administrator", $backupPassword, $certificateFolder =
"C:\Certificates", $websiteFolder = "C:\websites", $backupLocation =
"\\backupmachine\c$\backup" )

net use $backupLocation /user:$backupUser $backupPassword

xcopy /Y /q /E $certificateFolder $backupLocation\

xcopy /Y /q /E $websiteFolder $backupLocation\
```

Sample FSRM Quota Data Backup Script

Note

This script is not supported by Microsoft.

```
param ([string] $backupUser = "Administrator", $backupPassword, $backupLocation =
"\\machine\c$\backup")

net use \\$backupLocation /user:$backupUser $backupPassword

dirquota template export /File:C:\templates.xml

xcopy /Y /q C:\templates.xml $backupLocation\

net stop srmReports

net stop srmSvc

net stop quota

net stop Datascrn

robocopy "C:\System Volume Information\SRM" $backupLocation\SRM /E /ZB /R:3 /W:5

net start Datascrn

net start quota

net start srmSvc

net start srmReports
```

See Also

Restoring Windows Azure Pack: Web Sites

Deploy Windows Azure Pack: Web Sites

Restoring Windows Azure Pack: Web Sites

It is highly recommended that you restore to servers that have the same names and administrative accounts as they did during the backup. For the restore to be successful, the File

Server and SQL Server must be the exact same in terms of configuration, users, and permissions as they were during the backup. When restoring your Web Sites service, use the following order:

1. Restore SQL Server databases
2. Restore the File Server
3. Restore the Web Sites Controller
4. Run a repair on all Roles

You can use scripts to perform the restore operations. The steps are described below in detail.

1. Restore SQL Server databases

Restore the SQL server Hosting, Resource Metering, and master databases. The SQL server must have the same name as when it was backed up.

Sample SQL Restore script

The following example script is provided for illustrative purposes only and is unsupported. The script that you create must be run with administrative privileges.

Note

This script is not supported by Microsoft.

```
param ([string] $backupUser = "Administrator", $backupPassword, $sqlServer, $sqlUser =
"sa", $sqlPassword, $backupLocation = "\\backupMachine\c$\backup" )

net use $backupLocation /user:$backupUser $backupPassword

xcopy /Y /q \\$backupMachine\c$\$backupLocation\Hosting.bak C:\HostingOfflineFeed\

xcopy /Y /q \\$backupMachine\c$\$backupLocation\ResourceMetering.bak
C:\HostingOfflineFeed\

xcopy /Y /q $backupLocation\master.bak C:\HostingOfflineFeed\

net start "SQL Server (MSSQLSERVER)" /f

sqlcmd -S $sqlServer -U $sqlUser -P $sqlPassword -Q "RESTORE DATABASE [master] FROM
DISK='C:\HostingOfflineFeed\master.bak' WITH REPLACE"

net stop "SQL Server (MSSQLSERVER)"

net start "SQL Server (MSSQLSERVER)"

sqlcmd -S $sqlServer -U $sqlUser -P $sqlPassword -Q "RESTORE DATABASE [Hosting] FROM
DISK='C:\HostingOfflineFeed\Hosting.bak' WITH REPLACE"

sqlcmd -S $sqlServer -U $sqlUser -P $sqlPassword -Q "RESTORE DATABASE [ResourceMetering]
FROM DISK='C:\HostingOfflineFeed\ResourceMetering.bak' WITH REPLACE"

del C:\HostingOfflineFeed\Hosting.bak

del C:\HostingOfflineFeed\ResourceMetering.bak

del C:\HostingOfflineFeed\master.bak
```

2. Restore the File Server

The File Server must have the same name that it had when it was backed up. You should restore the File Server components in the following order:

1. Restore the Certificate Share
2. Restore the WebSites Share
3. Reapply ACLs if necessary
4. Reapply FSRM quotas on the WebSites share

Two sample scripts are provided: one for steps a through c above, and one to reapply the FSRM quotas on the WebSites share.

Sample File Server Restore script

The following example script includes steps a-c above (it does not restore the FSRM quotas). The script is provided for illustrative purposes only and is unsupported. The script that you create must be run with administrative privileges.

Note

This script is not supported by Microsoft.

```
param ([string] $backupUser = "Administrator", $backupPassword, $certificateFolder =
"C:\Certificates", $websiteFolder = "C:\websites", $backupLocation =
"\\backupMachine\c$\backup" )

net use $backupLocation /user:$backupUser $backupPassword

mkdir $certificateFolder

mkdir $websiteFolder

xcopy /Y /q /E $backupLocation\ $certificateFolder

xcopy /Y /q /E $backupLocation\ $websiteFolder
```

Sample script to restore FSRM quotas

The following example script restores the FSRM quotas. The script is provided for illustrative purposes only and is unsupported. The script that you create must be run with administrative privileges.

Note

This script is not supported by Microsoft.

```
param ([string] $backupUser = "Administrator", $backupPassword, $backupLocation =
"\\backupMachine\c$\backup" )

net use $backupLocation /user:$backupUser $backupPassword

xcopy /Y /q $backupLocation\templates.xml C:\templates.xml

dirquota template import /File:C:\templates.xml
```

```
net stop srmReports

net stop srmSvc

net stop quota

net stop Datascrn

robocopy $backupLocation\SRM "C:\System Volume Information\SRM" /E /ZB /R:3 /W:5

net start Datascrn

net start quota

net start srmSvc

net start srmReports
```

3. Restore the Web Sites Controller

In order to restore the Web Sites Controller, you can use the **Restore.ps1** PowerShell script presented in this section.

Copy the Restore.ps1 script onto the Web Sites Controller, and then run the following command with Administrator privileges:

```
net use /Y $backupLocation /user:$backupMachineAdmin $backupMachinePassword

.\Restore.ps1 $backupLocation $encryptionKey
```

Note

The $encryptionKey flag is only needed if you used it during backup.

The Restore.ps1 script follows.

```
##

##  Re-install and restore the controller from a backup

##


param ($backupPath,$password)


function ShowHelp

{

    Write-Host '====================== RESTORE.PS1 HELP ======================'

    Write-Host 'This is a script that restores based on a backup from the Hosting VSS
writer'

    Write-Host 'Invoke it using .\Restore.ps1'

    Write-Host 'It can also be invoked as follows:'
```

```
    Write-Host '.\Restore.ps1 <backup path (e.g. \\backupmachine\C$\backuplocation)>
<password for keys file>'
    Write-Host ("Note: before running this script you may need to run:`r`n" + ' "net use
/Y <backup path> /user:<username> <password>"')
    Write-Host '============================================================='
}

function CreateFeedWebAppIfNeeded ([string] $localFeedLocation)
{
    import-module WebAdministration
    $app = Get-WebApplication -Name HostingOfflineFeed
    if ($app -eq $null)
    {
        New-WebApplication -Name HostingOfflineFeed -Site 'Default Web Site' -
PhysicalPath $localFeedLocation -ApplicationPool DefaultAppPool -Force
    }
    # Add mime types needed for downloading .msp files for offline installations
    $msp = Get-WebConfiguration //staticContent/* | where {$_.fileExtension -eq '.msp'}
    if ($msp -eq $null)
    {
        Add-WebConfiguration //staticContent -Value
@{fileExtension=".msp";mimeType="application/octet-stream"}
    }

    # Add mime types needed for downloading .msu files for offline installations
    $msu = Get-WebConfiguration //staticContent/* | where {$_.fileExtension -eq '.msu'}
    if ($msu -eq $null)
    {
        Add-WebConfiguration //staticContent -Value
@{fileExtension=".msu";mimeType="application/octet-stream"}
    }
}

function InstallController ([string]$offlineFeedUrl, [string]$customFeed)
{
```

```
    $WebPiCmd =
([System.Environment]::ExpandEnvironmentVariables("%ProgramW6432%\Microsoft\Web Platform
Installer\WebpiCmd.exe"))
    if (!(Test-Path $WebPiCmd))
    {
        $WebPiCmd = Join-Path -Path $localFeedLocation -ChildPath "bin\WebpiCmd.exe"
    }
    Invoke-Command -ScriptBlock { & $WebPiCmd /Install /Products:HostingController
/AcceptEula /XML:$offlineFeedUrl /SuppressReboot /Log:HostingController.log }

    if ($lastexitcode -ne $null -And $lastexitcode -ne 0)
    {
        Write-Host "ERROR: There was a problem installing using WebPI!"
        exit $lastexitcode
    }
}

function DecodeBase64($string)
{
    return
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($string))
}

function DecryptString($EncryptedFile, $Passphrase, $salt, $init)
{
    $encryptedStrings = (Get-Content $EncryptedFile)
    $ret = @()
    foreach ($Encrypted in $encryptedStrings)
    {
        # If the value in the Encrypted is a string, convert it to Base64
        if($Encrypted -is [string])
        {
            $Encrypted = [Convert]::FromBase64String($Encrypted)
        }
```

```
        # Create a COM Object for RijndaelManaged Cryptography
        $r = new-Object System.Security.Cryptography.RijndaelManaged
        # Convert the Passphrase to UTF8 Bytes
        $pass = [Text.Encoding]::UTF8.GetBytes($Passphrase)
        # Convert the Salt to UTF Bytes
        $salt = [Text.Encoding]::UTF8.GetBytes($salt)
        # Create the Encryption Key using the passphrase, salt and SHA1 algorithm at 256
bits
        $r.Key = (new-Object Security.Cryptography.PasswordDeriveBytes $pass, $salt,
"SHA1", 5).GetBytes(32) #256/8
        # Create the Intersecting Vector Cryptology Hash with the init
        $r.IV = (new-Object Security.Cryptography.SHA1Managed).ComputeHash(
[Text.Encoding]::UTF8.GetBytes($init) )[0..15]
        # Create a new Decryptor
        $d = $r.CreateDecryptor()
        # Create a New memory stream with the encrypted value.
        $ms = new-Object IO.MemoryStream @(,$Encrypted)
        # Read the new memory stream and read it in the cryptology stream
        $cs = new-Object Security.Cryptography.CryptoStream $ms,$d,"Read"
        # Read the new decrypted stream
        $sr = new-Object IO.StreamReader $cs
        # Return from the function the stream
        $ret += $sr.ReadToEnd()
        # Stops the stream
        $sr.Close()
        # Stops the crypology stream
        $cs.Close()
        # Stops the memory stream
        $ms.Close()
        # Clears the RijndaelManaged Cryptology IV and Key
        $r.Clear()
    }
    return $ret
}
```

```
if ($backupPath -and $backupPath.Contains('/?'))
{
    ShowHelp
    return
}

Write-Host 'Starting the hosting restore process. Run with /? to see help.'
Write-Host ("Note: before running this script you may need to run:`r`n" + ' "net use /Y
<backupPath> /user:<username> <password>"')
# argument parsing
if (!$backupPath)
{
    $backupPath = Read-Host "Please enter the name of the backup path (e.g.
\\backupmachine\C$\backuplocation)"
}
if (!$password)
{
    $password = Read-Host "Please enter the password of the keys file" -AsSecureString
    $password =
[Runtime.InteropServices.Marshal]::PtrToStringAuto([Runtime.InteropServices.Marshal]::Sec
ureStringToBSTR($password))
}

$systemDrive = [System.Environment]::ExpandEnvironmentVariables('%systemdrive%\')
# Fetch restore data from remote machine
$localFeedLocation = ($systemDrive +'HostingOfflineFeed\')
$c = 0
do
{
    $c++
    "D" | xcopy /q /Y (Join-Path -Path $backupPath -ChildPath "HostingOfflineFeed")
"$localFeedLocation" /E
} while ($c -lt 10 -and !$?)
# Install the IIS cmdlets
```

```
$dismLocation = Join-Path -Path $systemDrive -ChildPath 'Windows\System32\dism.exe'
& $dismLocation /online /enable-feature /featurename:IIS-ManagementScriptingTools /all
CreateFeedWebAppIfNeeded $localFeedLocation

Stop-Service ResourceMetering -ErrorAction SilentlyContinue
# install webpi
$wpi = (dir ($systemDrive + 'hostingofflinefeed\installers\HostingWebPlatformInstaller')
-r -i 'wpi.msi').DirectoryName
if ($wpi.Count -gt 1)
{
    $wpi = $wpi[0]
}
$wpi = Join-Path -Path $wpi -ChildPath "wpi.msi"
msiexec /quiet /i $wpi
$offlineFeedUrl = 'http://localhost/HostingOfflineFeed/feeds/latest/WebSites0.9.0.xml'
InstallController $offlineFeedUrl
$keys = DecryptString (Join-Path -Path $backupPath -ChildPath 'encryptedkeys.txt')
$password 'salt12345' 'init12345'
Stop-Service WebFarmService -ErrorAction SilentlyContinue
Add-PSSnapIn WebHostingSnapIn
# Restore the keys
# Keys are Base64 encoded
Set-ControllerConnectionString -ConnectionString (DecodeBase64($keys[0])) 3>$null
# Set-MeteringConnectionString -MeteringConnectionString (DecodeBase64($keys[1])) -
ServerName (HostName)
Set-SymmetricKey -SymmetricKeyName SystemCore -SymmetricKey (DecodeBase64($keys[1]))
3>$null
Set-SymmetricKey -SymmetricKeyName SiteRuntime -SymmetricKey (DecodeBase64($keys[2]))
3>$null
Set-MeteringConnectionString -MeteringConnectionString
([Microsoft.Web.Hosting.SiteManager]::GetMeteringConnectionString()) -ServerName
(HostName) 3>$null

Start-Service WebFarmService -ErrorAction SilentlyContinue
```

Restoring to non-file servers with different names or administrative accounts

If you must restore a server (that is not a Filer Server or SQL Server) to a server or servers with server names or administrative accounts that are different from those of the original, you must:

1. Import the WebSites module
2. Update the credentials
3. Remove the old server names from the farm
4. Add the new servers to the proper farms

1. First, before running any of the other commands, import the WebSites module:

   ```
   Import-Module WebSites
   ```

   Now run the commands 2-4 on the Controller as Administrator.

2. If the credentials for the role <*RoleType*> have changed, run the following for each changed credential:

   ```
   Set-WebSitesConfig Credential -CredentialName <RoleType>
   Credential -UserName <RoleAdminUser> -Password
   <RoleAdminPassword>
   ```

   Note
   Possible values for <*RoleType*> in the Set-WebSitesConfig command are: ManagementServer, FileServer, FrontEnd, Publisher, and Worker.

3. For each old server name <*OldName*> that is no longer used, run:

   ```
   Remove-WebSitesServer -Name <OldName>
   ```

4. For each new server name <*NewName*> of role <*RoleType*> run this:

   ```
   New-WebSitesServer -Name <NewName> -ServerType <RoleType>
   ```

   Note
   Possible values for <*RoleType*> in the New-WebSitesServer command are: ManagementServer, FileServer, LoadBalancer, Publisher, and WebWorker.

Example

If you had an old Web Worker role named "OldWorker" and a new Web Worker role called "NewWorker", and you had updated the WebWorker credentials to "WebWorkerAdmin", you would run:

```
Import-Module WebSites


Set-WebSitesConfig Credential -CredentialName WorkerCredential -UserName WebWorkerAdmin -
Password $WebWorkerPassword


Remove-WebSitesServer -Name OldWorker
```

```
New-WebSitesServer -Name NewWorker -ServerType WebWorker
```

4. Run a repair on all Roles

After completing the restore, run a repair on all the roles and monitor them to verify that they come back up healthy.

See Also

Backing up Windows Azure Pack: Web Sites

Deploy Windows Azure Pack: Web Sites

Upgrading Windows Azure Pack: Web Sites from Preview Versions

This topic provides scripts and commands that you can use to upgrade a supported preview version of Windows Azure Pack: Web Sites to the R2 release version. For information on upgrading Windows Azure Pack itself, see **Upgrade from the Preview version of Windows Azure Pack**.

When upgrading Windows Azure Pack: Web Sites from a preview version, keep in mind the following:

- You must start the upgrade from the Web Sites controller.
- The Web Sites roles can be upgraded at the default rate of 5% of servers per server farm, or you can specify a different percentage.
- If you are upgrading from the V2 preview to the R2 release, you must also execute a set of SQL commands against the hosting database.

> **Warning**
> The Web Sites Controller will be fully functional only after the new Windows Azure Pack: Web Sites version is installed and the databases are upgraded. Until the database upgrade, the controller will be in an unusable state because the web farm service cannot be started.

Start the Upgrade

On the Web Sites controller, perform the following steps.

1. At an elevated command prompt, optionally stop the web farm service. This step eliminates the need to restart the controller after installation:

   ```
   net stop webfarmservice
   ```

2. Although the installing the new version will automatically uninstall the old version, you can manually uninstall the old version first if you want. To uninstall the previous version manually, go to **Control Panel > Programs and Features**, and uninstall **Web Hosting Framework**.
3. Using the Web Platform Installer, install the new version of Windows Azure Pack: Web Sites. For more information, see Start the installation of Windows Azure Pack: Web Sites.

   You can upgrade the Windows Azure Pack: Web Sites roles at the default rate of 5% of servers per server farm, or in a controlled fashion.

To upgrade 5% of the servers per server farm at a time

On the controller, run the following PowerShell script to upgrade the database, set the feedUrl in the hostingConfiguration and start the upgrade on all roles. Change the database server name and password information as required by your environment.

```
"%ProgramFiles%\IIS\Microsoft Web Sites\Feed\WebSitesSetupHelper.ps1" -actions
CompleteUpgrade -upgradeConnectionString "Server=DB-Websites;User
Id=sa;Password=password"
```

Note

- The -upgradeConnectionString parameter is required only to upgrade V1 to the R2 release (it is not require to upgrade V2 Preview to the R2 release).
- The connection string must use sa credentials.

To upgrade Windows Azure Pack: Web Sites servers at a specified rate:

1. Run the following command to upgrade the database and set the new feed URL in the hosting configuration:

   ```
   "%ProgramFiles%\IIS\Microsoft Web
   Sites\Feed\WebSitesSetupHelper.ps1" -actions CompleteUpgrade -
   skipRoleUpgrade
   ```

2. Use the following PowerShell commands to control the rate at which the roles are upgraded. Set the value for **WFFMaximumStoppedServersPercentage** to the percentage of servers per server farm that you want to upgrade at a time. The default is 5%.

   ```
   Add-pssnapin WebHostingSnapin
   Set-WebSitesConfig -Type Global -WffMaxStoppedServersPercent
   <percentage>
   ```

   Example

   The following example upgrades 20% of the servers in each server farm at a time.

   ```
   Set-WebSitesConfig -Type Global -WffMaxStoppedServersPercent 20
   ```

Initiate the role upgrade for all Windows Azure Pack: Web Sites roles, or on a per-role basis

1. To initiate the upgrade for all Web Sites roles, run the following PowerShell command:

   ```
   "%ProgramFiles%\IIS\Microsoft Web
   Sites\Feed\WebSitesSetupHelper.ps1" -actions 'UpgradeAllServers'
   ```

2. To initiate the upgrade of Web Sites roles on a per-role basis, you can run each of these PowerShell commands separately:

   ```
   Start-Operation -OperatorName WFF -OperationName Upgrade
   @{"WebFarmName"="FrontEndServers"}


       Start-Operation -OperatorName WFF -OperationName Upgrade
   @{"WebFarmName"="FileServers"}


       Start-Operation -OperatorName WFF -OperationName Upgrade
   @{"WebFarmName"="ManagementServers"}


       Start-Operation -OperatorName WFF -OperationName Upgrade
   @{"WebFarmName"="PublisherServers"}


       Start-Operation -OperatorName WFF -OperationName Upgrade
   @{"WebFarmName"="WorkerServers"}
   ```

When upgrading from V2 Preview to the R2 release

When upgrading from V2 Preview to the R2 release (but not from V1 to the R2 release), you must also run the following SQL commands against the "Hosting" database. These commands enable support to update existing subscriptions and plans after the upgrade.

```
-- Get the ResourceId for the 'CpuTime' resource.

DECLARE @CpuTimeResourceId INT;

SELECT @CpuTimeResourceId = (SELECT TOP 1 ResourceId FROM runtime.QuotaResources WHERE
ResourceName = N'CpuTime')


INSERT  INTO [runtime].[OwnerQuotas]

               ([QuotaName],

                [PolicyId],

                [ResourceId],

                [Limit],
```

```
                [ExceededAction],
                [TimeUnits],
                [Period],
                [ActionId])
SELECT 'CpuTimeBurst', POL.PolicyID, @CpuTimeResourceId, -1, 0, 1, 5, NULL
FROM [admin].[subscriptions] SUB
INNER JOIN [runtime].[SitePolicies] POL
ON SUB.Name = POL.PlanName
WHERE NOT EXISTS (SELECT 1 FROM runtime.OwnerQuotas O WHERE O.PolicyId = POL.PolicyID AND
QuotaName = N'CpuTimeBurst')

INSERT  INTO [runtime].[SiteQuotas]
               ([QuotaName],
                [PolicyId],
                [ResourceId],
                [Limit],
                [ExceededAction],
                [TimeUnits],
                [Period],
                [ActionId])
SELECT 'CpuTimeBurst', POL.PolicyID, @CpuTimeResourceId, -1, 0, 1, 5, NULL
FROM [admin].[subscriptions] SUB
INNER JOIN [runtime].[SitePolicies] POL
ON SUB.Name = POL. PlanName
WHERE NOT EXISTS (SELECT 1 FROM runtime.SiteQuotas O WHERE O.PolicyId = POL.PolicyID AND
QuotaName = N'CpuTimeBurst')
```

See Also

Upgrade from the Preview version of Windows Azure Pack

Deploy Windows Azure Pack: Web Sites