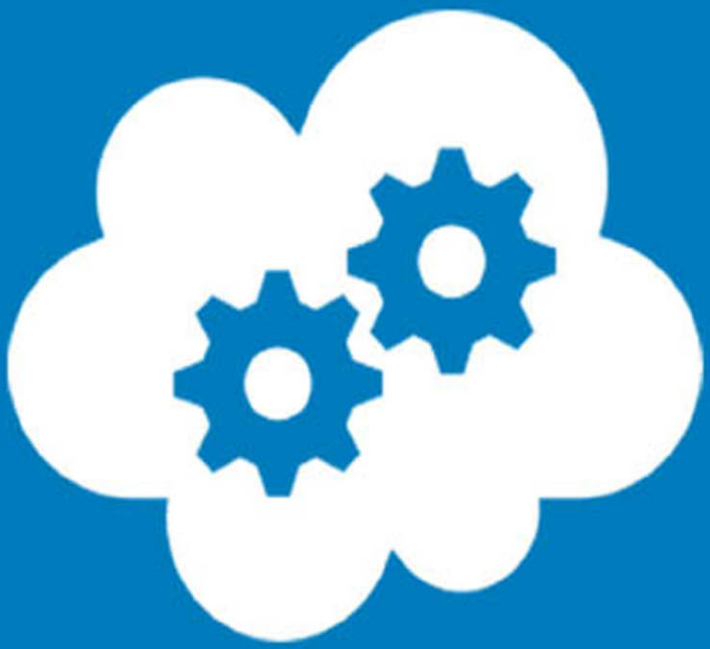
Drupal on Windows Azure

Rama Ramani Jason Roth Brian Swan

Quick Guide



Microsoft

Drupal on Windows Azure

Rama Ramani, Jason Roth, Brian Swan

Summary: This e-book explains the migration, architecture patterns and management of your Drupal based website on Windows Azure.

Category: Quick Guide
Applies to: Windows Azure
Source: MSDN blogs (link to source content | link to source content)
E-book publication date: June 2012

Microsoft

Contents

Introduction

Drupal is an open source content management system that runs on PHP. Windows Azure offers a flexible platform for hosting, managing, and scaling Drupal deployments. This paper focuses on an approach to hosting Drupal sites on Windows Azure, based on learning from a BPD Customer Programs Design Win engagement with the Screen Actors Guild Awards Drupal website. The Screen Actors Guild (SAG) is the United States' largest union representing working actors. In January of every year since 1995, SAG has hosted the Screen Actors Guild Awards (SAG Awards) to honor performers in motion pictures and TV series. In 2011, the SAG Awards Drupal website, deployed on a LAMP stack, was impacted by site outages and slow performance during peak-usage days, with SAG having to consistently upgrade their hardware to meet demand for those days. That upgraded hardware was then not optimally used during the rest of the year. In late 2011, SAG Awards engineers began working with Microsoft engineers to migrate the website to Windows Azure in anticipation of its 2012 show. In January of 2012, the SAG Website had over 350K unique visitors and 1.1M page views, with traffic spiking to over 160K visitors during the show.

With the recent release of Windows Azure Websites, installing and creating a Drupal based website is straight-forward and seamless. However, as stated earlier, this paper is based on a customer design win project (SAG Awards) that used the Windows Azure platform-as-a-service (PaaS) offering, as Windows Azure Websites was not available at the time SAG Awards was migrating to Windows Azure.

> **Note**: Windows Azure Websites is in beta. Some of the challenges faced by SAG Awards might only have been addressed by using the Windows Azure PaaS offering.

Customer Scenario: Overview and Challenges

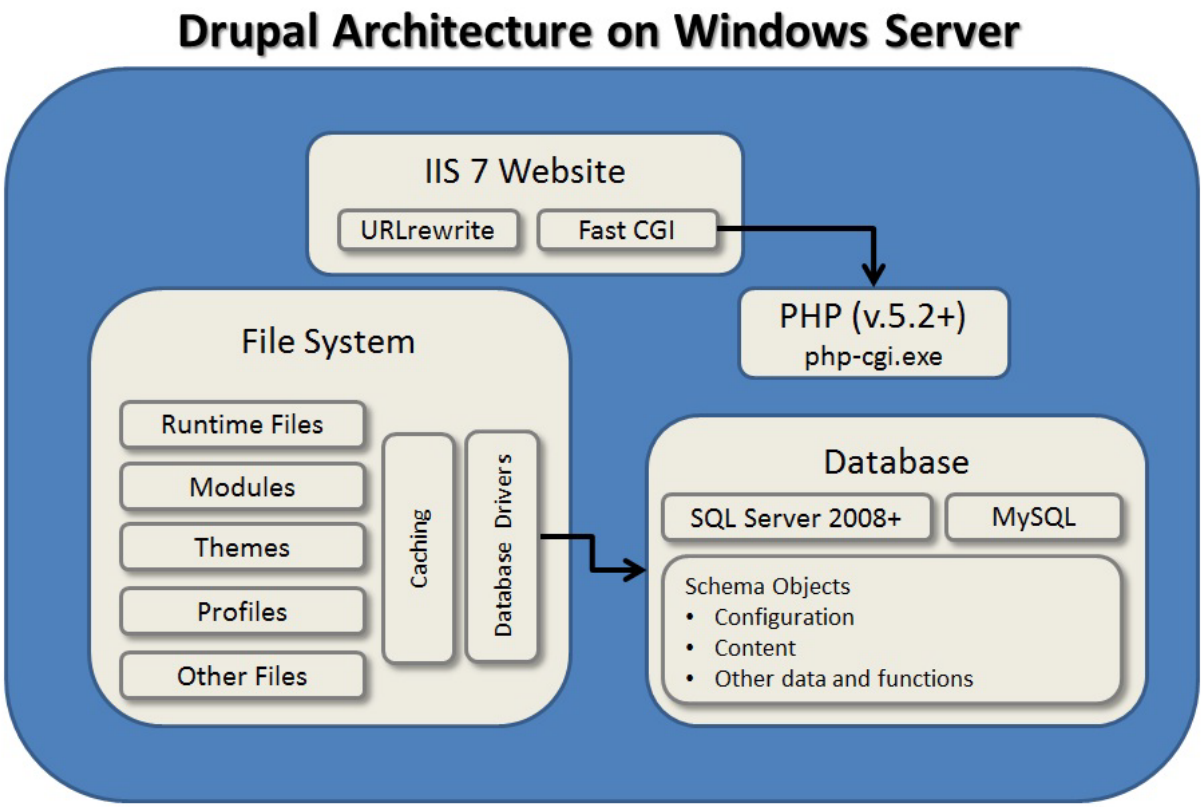
In many ways, the SAG Awards website was a perfect candidate for Windows Azure. The website has moderate traffic throughout most of the year, but has a sustained traffic spike shortly before, during, and after the awards show in January. The elastic scalability and fast storage services offered by the Azure platform were designed to handle this type of usage.

The main challenge that SAG Awards and Microsoft engineers faced in moving the SAG Awards website to Windows Azure was in architecting for a very high, sustained traffic spike while accommodating the need of SAG Awards administrators to frequently update media files during the awards show. Both the intelligent use of Windows Azure Blob Services and a custom module for invalidating cached pages when content was updated were keys to delivering a positive user experience.

The next sections cover the architecture and design, migration steps, and management and monitoring of the production system.

Architecture

Before looking at the SAG Awards architecture for Drupal on Windows Azure, it is important to first understand the architecture of a typical Drupal deployment on Windows Azure. The following diagram displays the basic architecture of Drupal running on Windows and IIS7.



At its core, Drupal 7 is a PHP application. Drupal on Windows can be hosted within an IIS 7 website that leverages the FastCGI module to invoke the PHP runtime. The file system supports both application logic and file storage for runtime created files. Actual application logic for Drupal exists on the file system in the form of PHP scripts, PHP include files, and `.info` manifest metadata files that are processed server-side by the PHP runtime. In addition, the traditional CSS, JavaScript, and image files you would expect of any modern website are part of a typical Drupal installation.

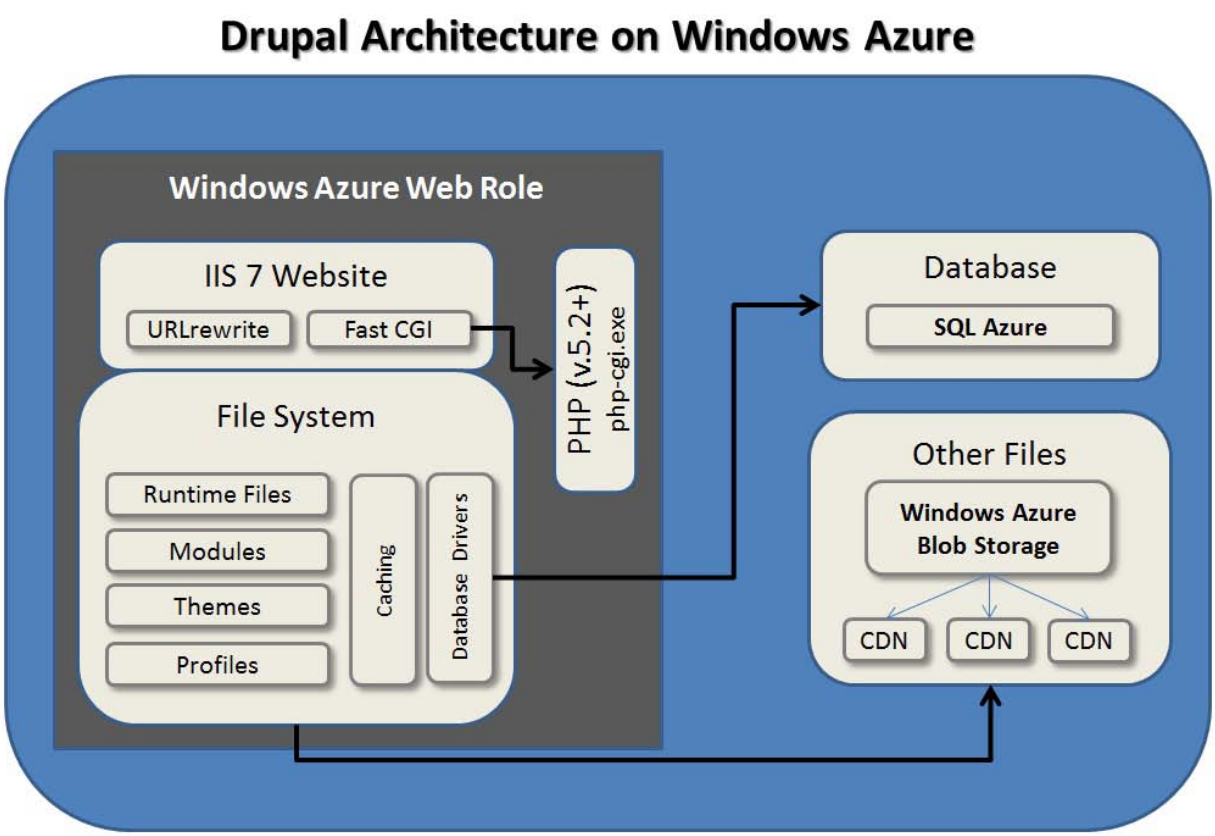
Modules in Drupal provide application logic and consist primarily of PHP scripts. They occasionally include JavaScript, CSS and image files as required by output of the module. All modules are described by a `.info` file that provides the name of the module, a description, and the required version of Drupal, as well as an optional manifest of all the files required by the module. In fact, even the database drivers used by Drupal to communicate with SQL Server, PostgreSQL, SqlLite and MySQL are just modules consisting of PHP scripts that extend the PHP Data Objects (PDO) API.

For the SAG Awards website, all configuration and site content used by Drupal is stored within a SQL Database (with a few exceptions). The database is used as both as a cache for storing transient data in tables, and also used to create proprietary tables created by using database API. It is worth noting thatthe database schema does not rely on any stored procedures or triggers. By having the PHP Driver for SQL Server 2.0 installed (within the file system), Drupal can use either SQL Server 2008 (or later) or even Windows Azure SQL Database.

The concept of “cache” is critical in such an architecture, especially given the large portion of the site which is static and read-only. This can be maintained in several layers – as mentioned in the earlier paragraph, there are a set of tables within the database to keep transient data in addition to the in-memory cache layer within the database server. Drupal also has modules for popular application caches such as memcached. This is a very popular approach to reduce load on the database server and increase overall application performance.

Mapping architecture for Windows Azure

For Windows Azure, the basic architecture is the same, but there are some differences. In Windows Azure, the site is hosted on a web role. A web role instance is hosted on a Windows Server 2008 virtual machine within a Windows Azure datacenter. Like a web farm, you can have multiple instances running. However, there is no persistence guarantee for the data on the file system. Because of this, much of the shared site content should be stored using the Windows Azure Blob Service. This allows data to be highly available and durable. Finally, the database can be located in SQL Database. The following diagram shows these differences.

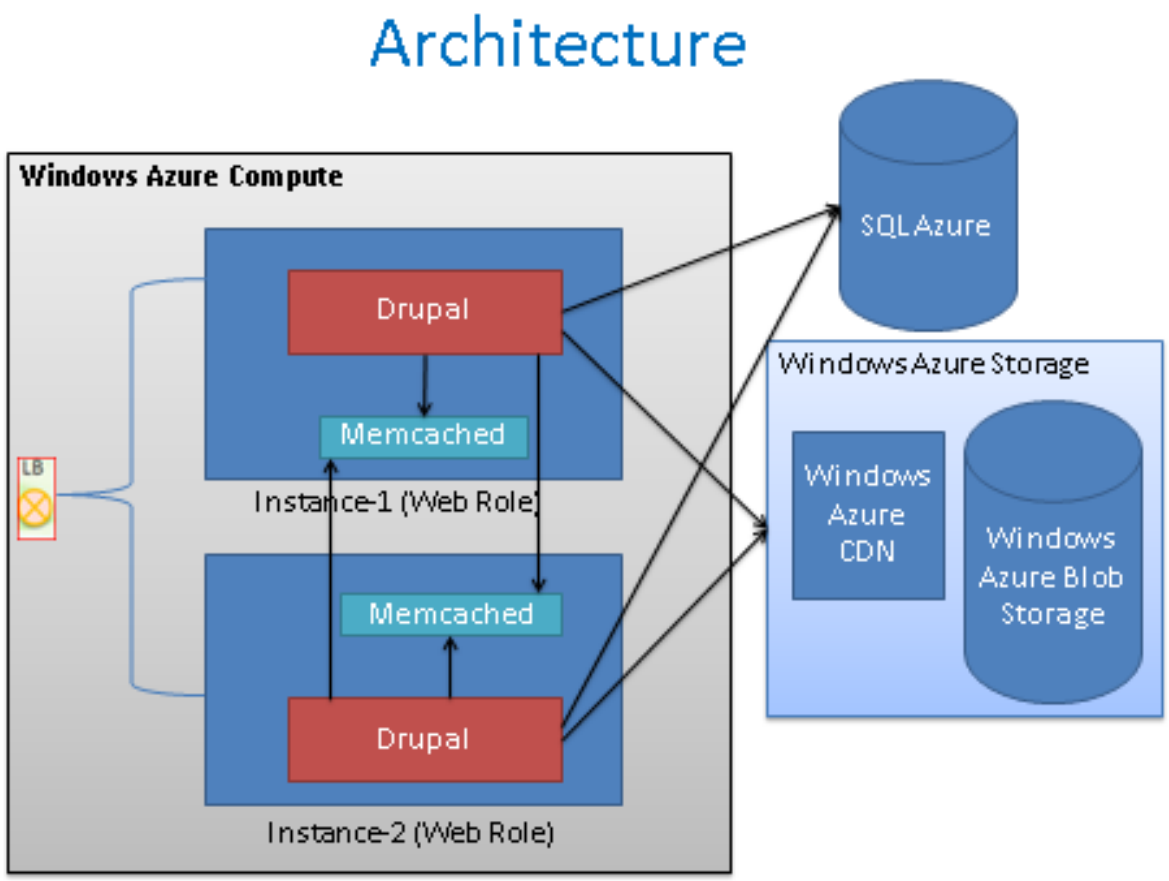


As mentioned earlier, a large portion of the site caters to static content which lends well to caching. Caching can be applied in a set of places – browser level caching, CDN to cache content in edge nodes

that are closer to the browser clients, Azure caching to reduce the load on backend, etc. Implementing a distributed cache helps to scale the backend database server which is critical when there is significantly more user load. So, it is important to ensure each tier of the solution can be scaled out. In case of a Drupal website, there are 2 key usersthat must be considered when designing for scale out– Drupal admins who make content updates and end users who predominantly do reads. Depending on the actual application, the end users might have an impact on the "write workload" (e.g.: uploading pictures, sending comments, etc.).

Deployment Topologies

Based on these details, here is the simple deployment topology for the SAG Awards requirements.



In this topology, all web role instances also run memcached and together they form a distributed cache. A user request emanating from any web role will access the appropriate memcached for looking up the key and on a cache miss will access SQL Database and populate the database. In this deployment topology, key-value pairs are not replicating and only 1 copy is maintained. Co-location of the cache as part of the compute instance primarily allows keeping costs down (because data transfer within a data center is free) and also builds a model where by the distributed cache gets more nodes when more web role instances are introduced.

A variant of the above architecture involvesa dedicated memcached deployment where the memcached instances are running in their own deployment and the web roles are accessing them. This separates the cache workload from the web workload. Even though this increases the cost, it helps make for predictable capacity planning.

Both of the models mentioned above work well if cache keys are well distributed across the entire cluster and a particular key does not get significantly "hot". In certain applications, like an awards show, it is typical that a particular page is accessed by all users. In such cases, a replicated cache deployed along with the compute as a scale unit deployed as a "POD" would work well. In this case, as the user load increases, more instances can be deployed and since the cache is also replicated, the user request can be satisfied right from that instance. The drawback to this approach is in handling cache invalidations. When the data source updates an item, now the invalidation has to be performed in several instances instead of just one. In summary, one should weigh the frequency of updates and actual workload when choosing this deployment topology over the rest.

Finally, the deployment topology should consider the Drupal content author (admin) persona. When content authors are logged in as Drupal administrators, caching is usually disabled since the edits need to make it to the backend. In all the deployment topologies above, since end users and content authors access the identical system, it is possible that the content updates take longer to complete when the system is under load. To mitigate this problem, separating the URL for admins (content creators) and end-users while sharing the database and Azure blobs would help. Thus, the content authors would access dedicated web roles that do not get any end user requests, and admin intensive tasks, such as image edits, and uploads, can complete much faster.

Migrating Drupal website to Windows Azure

The process for moving the SAG-Awards website from a LAMP environment to the Windows Azure platform can be broken down into five high-level steps:

1. **Export data.** A custom Drush command (portabledb-export) was used to create a database dump of MySQL data. A .zip archive of media files was created for later use.
2. **Install Drupal on Windows.** The Drupal files that comprised the installation in the LAMP environment were copied to Windows Server/IIS as an initial step in discovering compatibility issues.
3. **Import data to Windows Azure SQL Database.** A custom Drush command (portabledb-import) was used together with the database dump created in step 1 to import data to SQL Database.
4. **Copy media files to Azure Blob Storage.** After unpacking the .zip archive in step 1, CloudXplorer was used to copy these files to Windows Azure Blob Storage.
5. **Package and deploy Drupal.** The Azure packaging tool cspack was used to package Drupal for deployment. Deployment was done through the Windows Azure Portal.

Note: The portabledb commands mentioned above are authored and maintained by Damien Tournoud.

Details for each of these high-level steps are in the sections below.

Export Data

Microsoft and SAG engineers began investigating the best way to export MySQL data by looking at Damien Tournoud's portabledb Drush commands. They found that this tool worked perfectly when moving Drupal to Windows and SQL Server, but they needed to make some modifications to the tool for exporting data to SQL Database. (These modifications have since been incorporated into the **portabledb** commands, which are now available as part of the Windows Azure Integration Module.)

The names of media files stored in the `file_managed`table were of the form `public://field/image/file_name.avi`. In order for these files to be streamed from Windows Azure Blob Storage (as they would be by the Windows Azure Integration module when deployed in Azure), the file names needed to be modified to this form: `azurepublic://field/image/file_name.avi`. This was an easy change to make.

Because the SAG Awards website would be retrieving all data from the cloud, Windows Azure Storage connection information needed to be stored in the database. The **portabledb** tool was modified to create a new table, `azure_storage`, for containing this information.

Finally, to allow all media files to be retrieved from Blob Storage, the `file_default_scheme`table needed to be updated with the stream wrapper name: `azurepublic`.

Using the modified **portabledb** tool, the following command produced the database dump:

```
drush portabledb-export --use-windows-azure-storage=true --windows-
azure-stream-wrapper-name=azurepublic --windows-azure-storage-account-
name=azure_storage_account_name --windows-azure-storage-account-
key=azure_storage_account_key --windows-azure-blob-container-
name=azure_blob_container_name --windows-azure-module-
path=sites/all/modules --ctools-module-path=sites/all/modules >
drupal.dump
```

Note that the **portabledb-export** command does not copy media files themselves. Instead, the local media files were compressed in a .zip archive for use in a later step.

Install Drupal on Windows

In order to use the **portabledb-import** command (the counter part to the **portabledb-export** command above), a Drupal installation needed to be set up on Windows (with Drush for Windows installed). This was necessary, in part, because connectivity to SQL Databasewas to be managed by the Commerce Guys' SQL Server/SQL Databasemodule for Drupal, which relies on the SQL Server Drivers for PHP, a Windows-only PHP extension. Having a Windows installation of Drupal would also make it possible to package the application for deployment to Windows Azure. For this reason, Microsoft and SAG Awards engineers copied the Drupal files from the LAMP environment to a Windows Server machine. The team incrementally moved the rest of the application to an IIS/SQL Server Express stack before moving the backend to SQL Database.

Note: The Windows Server machine was actually a virtual machine running in a Windows Azure Web Role in the same data center as SQL Database. The Web Role was configured to allow RDP connections, which the team used to install and configure the SAG website installation. This was done to avoid timeouts that occurred when attempting to upload data from an on-premises machine to SQL Database.

There were, however, some customizations made to the Drupal installation before running the **portabledb-import** command. Specifically,

- The SQL Server/SQL Databasemodule for Drupal was installed and enabled.
- The memcache module for Drupal was installed and enabled.
- The Windows Azure Integration module for Drupal was installed and enabled. Note that this module has a dependency on the CTools module. It also requires Damien Tournoud's branch of the Windows Azure SDK for PHP, which must be unpacked and put into a folder called **phpazure** in the module's main directory. (As of this writing, Damien Tournoud's changes have not been merged with the Windows Azure SDK for PHP. However, they may be merged in the future.)
- Database connection information in the settings.php file was modified to connect to SQL Database.
- A custom caching module was installed and enabled.

Some customizations to PHP were also necessary since this PHP installation would be packaged with the application itself:

- The php_pdo_sqlsrv.dll extension was installed and enabled. This extension provided connectivity to SQL Database.
- The php_memcache.dll extension was installed an enabled. This would be used for caching purposes.
- The php_azure.dll extension was installed and enabled. This extension allowed configuration information to be retrieved from the Windows Azure service configuration file after the application was deployed. This allowed changes to be made without having to re-package and re-deploy the entire application. For example, database connection information could be retrieved in the settings.php file like this:

```
$databases['default']['default']['driver'] = 'sqlsrv';

$databases['default']['default']['username'] =
azure_getconfig('sql_azure_username');

$databases['default']['default']['password'] =
azure_getconfig('sql_azure_password');

$databases['default']['default']['host'] =
azure_getconfig('sql_azure_host');

$databases['default']['default']['database'] =
azure_getconfig('sql_azure_database');
```

With Drupal running on Windows, and with the customizations to Drupal and PHP outlined above, the importing of data could begin.

Import Data to Windows Azure SQL Database

There were two phases to importing the SAG Awards website data: importing database data to SQL Databaseand copying media files to Windows Azure Blob Storage. As alluded to above, importing data to SQL Databasewas done with the **portabledb-import** Drush command. With SQL Databaseconnection information specified in Drupal's `settings.php`file, the following command copied data from the `drupal.dump` file (which was copied to Drupal's root directory on the Windows installation) to SQL Database:

```
drush portabledb-import --delete-local-files=false --copy-files-blob-
storage=false --use-production-storage=true mysite.dump
```

> **Note**: The copy-files-blob-storage flag was set to **false** in the command above. While the portabledb-import command *can* copy media files to Blob Storage, Microsoft and SAG engineers had some work to do in modifying media file names (discussed in the next section). For this reason, they chose not to use this tool for uploading files to Blob Storage.

The next step was to create stored procedures on SQL Databasethat are designed to handle some SQL that is specific to MySQL. The SQL Server/SQL Databasemodule for Drupal normally creates these stored procedures when the module is enabled, but since Drupal would be deployed with the module already enabled, these stored procedures needed to be created manually. Engineers executed the stored procedure creation DDL that is defined in the module by accessing SQL Databasethrough the management portal.

After the import was complete, the Windows installation of the SAG Awards website was now retrieving all database data from SQL Database. However, recall that the **portabledb-export** command modified the names of media files in the `file_managed`table so that the Drupal Azure module would retrieve media files from Blob Storage. The final phase in importing data was to copy media files to Blob Storage.

> **Note**: After this phase was complete, engineers cleared the cache through the Drupal admin panel.

Copy Media Files to Blob Storage

The main challenge in copying media files to Windows Azure Blob Storage was in handling Linux file name conventions that are not supported on Windows. While Linux supports a colon (:) as part of a file name, Windows does not. Consequently, when the .zip archive of media files was unpacked on Windows, file names were automatically changed: all colons were converted to underscores (_). However, colons *are* supported in Blob Storage as part of blob names. This meant that files could be uploaded to Blob Storage from Windows with underscores in the blob names, but the blob names would have to be modified manually to match the names stored in SQL Database.

Engineers used WinRAR to unpack the .zip archive of media files. WinRAR provided a record of all file names that were changed in the unpacking process. Engineers then used CloudXplorer to upload the media files to Blob Storage and to change the modified files names, replacing underscores with colons.

At this point in the migration process, the SAG Awards website was fully functional on Windows and was retrieving all data (database data and media files) from the cloud.

Package and Deploy Drupal

There were two main challenges in packaging the SAG Awards website for deployment to Drupal: packaging a custom installation of PHP and creating the necessary startup tasks.

Because customizations were made to the PHP installation powering Drupal, engineers needed to package their custom PHP installation for deployment to Windows Azure. The other option was to rely on Microsoft's Web Platform Installer to install a "vanilla" installation of PHP and then write scripts to modify it on start up. Since it is relatively easy to package a custom PHP installation for deployment to Azure, engineers chose to go that route. (For more information, see Packaging a Custom PHP Installation for Windows Azure.)

The startup tasks that needed to be performed were the following:

- Configure IIS to use the custom PHP installation.
- Register the Service Runtime COM Wrapper (which played a role in the caching strategy).
- Put Drush (which was packaged with the deployment) in the PATH environment variable.

The final project structure, prior to packaging, was the following:

```
\SAGAwards
      \WebRole
      \bin
            \php
            install-php.cmd
            register-service-runtime-COM-Wrapper.cmd
            WinRMConfig.cmd
            startup-tasks-errorlog.txt
            startup-tasks-log.txt
      \resources
            \drush
            \ServiceRuntimeCOMWrapper
            \WebPICmdLine
            (Drupal files and folders)
```

Finally, the Windows Azure Memcached Plugin was added to the Windows Azure SDK prior to packaging so that **memcache** would run on startup and restart if killed.

The SAG Awards website was then packaged using cspack and deployed to Windows Azure through the developer portal.

Managing & Monitoring your website

For monitoring and management, we will look at Drupal on Windows Azure from three perspectives:

- **Availability**: Ensure the web site does not go down and that all tiers are setup correctly. Apply best practices to ensure that the site is deployed across data centers and perform backup operations regularly.
- **Scalability**: Correctly handle changes in user load. Understand the performance characteristics of the site.
- **Manageability**: Correctly handle updates. Make code and site changes with no downtime when possible.

Although some management tasks span one or more of these categories, it is still helpful to discuss Drupal management on Windows Azure within these focus areas.

Availability

One main goal is that the Drupal site remains running and accessible to all end-users. This involves monitoring both the site and the SQL Database instance that the site depends on. In this section, we will briefly look at monitoring and backup tasks. Other crossover areas that affect availability will be discussed in the next section on scalability.

Monitoring

With any application, monitoring plays an important role with managing availability. Monitoring data can reveal whether users are successfully using the site or whether computing resources are meeting the demand. Other data reveals error counts and possibly points to issues in a specific tier of the deployment.

There are several monitoring tools that can be used.

- The Windows Azure Management Portal.
- Windows Azure diagnostic data.
- Custom monitoring scripts.
- System Center Operations Manager.
- Third party tools such as Azure Diagnostics Manager and Azure Storage Explorer.

The Windows Azure Management Portal can be used to ensure that your deployments are successful and running. You can also use the portal to manage features such as Remote Desktop so that you can directly connect to machines that are running the Drupal site.

Windows Azure diagnostics allows you to collect performance counters and logs off of the web role instances that are running the Drupal site. Although there are many options for configuring diagnostics in Azure, the best solution with Drupal is to use a diagnostics configuration file. The following configuration file demonstrates some basic performance counters that can monitor resources such as memory, processor utilization, and network bandwidth.

```
<DiagnosticMonitorConfigurationxmlns="http://schemas.microsoft.com/ServiceHos
ting/2010/10/DiagnosticsConfiguration"
configurationChangePollInterval="PT1M"
overallQuotaInMB="4096">
```

```
<DiagnosticInfrastructureLogsbufferQuotaInMB="10"
scheduledTransferLogLevelFilter="Error"
scheduledTransferPeriod="PT1M"/>

<LogsbufferQuotaInMB="0"
scheduledTransferLogLevelFilter="Verbose"
scheduledTransferPeriod="PT1M"/>

<DirectoriesbufferQuotaInMB="0"scheduledTransferPeriod="PT5M">
<!-- These three elements specify the special directories that are set up for
the log types -->
<CrashDumpscontainer="wad-crash-dumps"directoryQuotaInMB="256"/>
<FailedRequestLogscontainer="wad-frq"directoryQuotaInMB="256"/>
<IISLogscontainer="wad-iis"directoryQuotaInMB="256"/>
</Directories>

<PerformanceCountersbufferQuotaInMB="0"scheduledTransferPeriod="PT1M">
<!-- The counter specifier is in the same format as the imperative
diagnostics configuration API -->
<PerformanceCounterConfigurationcounterSpecifier="\Memory\Available
MBytes"sampleRate="PT10M"/>
<PerformanceCounterConfigurationcounterSpecifier="\Network Interface(*)\Bytes
Total/sec"sampleRate="PT10M"/>
<PerformanceCounterConfigurationcounterSpecifier="\Processor(*)\% Processor
Time"sampleRate="PT10M"/>
<PerformanceCounterConfigurationcounterSpecifier="\System\Processor Queue
Length"sampleRate="PT10M"/>
<PerformanceCounterConfigurationcounterSpecifier="\Web
Service(_Total)\Current Connections"sampleRate="PT10M"/>
<PerformanceCounterConfigurationcounterSpecifier="\Web Service(_Total)\Bytes
Total/sec"sampleRate="PT10M"/>
<PerformanceCounterConfigurationcounterSpecifier="\Web
Service(_Total)\Connection Attempts/sec"sampleRate="PT10M"/>
<PerformanceCounterConfigurationcounterSpecifier="\Web Service(_Total)\Get
Requests/sec"sampleRate="PT10M"/>
<PerformanceCounterConfigurationcounterSpecifier="\Web Service(_Total)\Files
Sent/sec"sampleRate="PT10M"/>
</PerformanceCounters>

<WindowsEventLogbufferQuotaInMB="0"
scheduledTransferLogLevelFilter="Verbose"
scheduledTransferPeriod="PT5M">

<!-- The event log name is in the same format as the imperative diagnostics
configuration API -->

<DataSourcename="System!*"/>

</WindowsEventLog>

</DiagnosticMonitorConfiguration>
```

For more information about setting up diagnostic configuration files, see How to Use the Windows Azure Diagnostics Configuration File. This information is stored locally on each role instance and then transferred to Windows Azure storage per a defined schedule or on-demand. See Getting Started with Storing and Viewing Diagnostic Data in Windows Azure Storage. Various monitoring tools, such as Azure Diagnostics Manager, help you to more easily analyze diagnostic data.

Monitoring the performance of the machines hosting the Drupal site is only part of the story. In order to plan properly for both availability and scalability, you should also monitor site traffic, including user load patterns and trends. Standard and custom diagnostic data could contribute to this, but there are also third-party tools that monitor web traffic. For example, if you know that spikes occur in your application during certain days of the week, you could make changes to the application to handle the additional load and increase the availability of the Drupal solution.

Backup Tasks

To remain highly available, it is important to backup your data as a defense-in-depth strategy for disaster recovery. This is true even though SQL Databaseand Windows Azure Storage both implement redundancy to prevent data loss. One obvious reason is that these services cannot prevent administrator error if data is accidentally deleted or incorrectly changed.

SQL Databasedoes not currently have a formal backup technology, although there are many third-party tools and solutions that provide this capability. Usually the database size for a Drupal site is relatively small. In the case of SAG Awards, it was only ~100-150 MB. So performing an entire backup using any strategy was relatively fast. If your database is much larger, you might have to test various backup strategies to find the one that works best.

Apart from third-party SQL Databasebackup solutions, there are several strategies for obtaining a backup of your data:

· Use the Drush tool and the portabledb-export command.

· Periodically copy the database using the CREATE DATABASE Transact-SQL command.

· Use Data-tier applications (DAC) to assist with backup and restore of the database.

SQL Databasebackup and data security techniques are described in more detail in the topic, Business Continuity in Windows Azure SQL Database.

Note that bandwidth costs accrue with any backup operation that transfers information outside of the Windows Azure datacenter. To reduce costs, you can copy the database to a database within the same datacenter. Or you can export the data-tier applications to blob storage in the same datacenter.

Another potential backup task involves the files in Blob storage. If you keep a master copy of all media files uploaded to Blob storage, then you already have an on-premises backup of those files. However, if multiple administrators are loading files into Blob storage for use on the Drupal site, it is a good idea to enumerate the storage account and to download any new files to a central location. The following PHP

script demonstrates how this can be done by backing up all files in Blob storage after a specified modification date.

```
<?php
/**
* Backup the blob storage to a local folder
*
* PHP version 5.3
*
* @category Blob Backup
* @package Windows Azure
* @author Bibin Kurian
* @copyright 2011 Microsoft Corp. (http://www.microsoft.com)
* @license New BSD license, (http://www.opensource.org/licenses/bsd-
license.php)
* @version SVN: 1.0
* @link
*
*/

/** Microsoft_WindowsAzure_Storage_Blob */
require_once'Microsoft/WindowsAzure/Storage/Blob.php';

//Windows Azure BLOB storage account name
define("AZURE_STORAGE_ACCOUNT", "YOUR_ACCOUNTNAME");

//Windows Azure BLOB storage container name
define("AZURE_STORAGE_CONTAINER", "YOUR_STORAGECONTAINERNAME");

//Windows Azure BLOB storage secret key
define(
"AZURE_STORAGE_KEY",
"YOUR_SECRETKEY"
);

//backup folder
define("STORAGE_BACKUP_DIR", "YOUR_LOCALDRIVE");

//backup from date
define("DEFAULT_BACKUP_FROM_DATE", strtotime("Mon, 19 Dec 2011 22:00:00
GMT"));

//backup to date
define("DEFAULT_BACKUP_TO_DATE", time());

//directory seperator
define("DS", "\\");

//start backup logic

//current datetime to create the backup folder
$now = date("F j, Y, g.i A");
```

```
$fullBackupDirPath = STORAGE_BACKUP_DIR . DS . $now . DS.
AZURE_STORAGE_CONTAINER;
mkdir($fullBackupDirPath, 0755, true);

//For the directory creations, pointing the current directory to user
specified directory
chdir($fullBackupDirPath);

//BLOB object
$blobObj = new Microsoft_WindowsAzure_Storage_Blob(
'blob.core.windows.net',
    AZURE_STORAGE_ACCOUNT,
    AZURE_STORAGE_KEY
);

//$blobObj->setProxy(true, 'YOUR_PROXY_IF_NEEDED', 80);

$blobs = (array)$blobObj->listBlobs(AZURE_STORAGE_CONTAINER, '', '', 35000);
backupBlobs($blobs, $blobObj);
function backupBlobs($blobs, $blobObj) {
foreach ($blobs as $blob) {
if (strtotime($blob->lastmodified) >= DEFAULT_BACKUP_FROM_DATE &&
strtotime($blob->lastmodified) <= DEFAULT_BACKUP_TO_DATE) {
            $path = pathinfo($blob->name);
if ($path['basename'] != '$$$.$$$') {
                $dir = $path['dirname'];
                $oldDir = getcwd();
if (handleDirectory($dir)) {
                    chdir($dir);
                    $blobObj->getBlob(
                        AZURE_STORAGE_CONTAINER,
                        $blob->name,
                        $path['basename']
                    );
                    chdir($oldDir);
                }
            }
        }
    }
}

function handleDirectory($dir) {
if (!checkDirExists($dir)) {
return mkdir($dir, 0755, true);
    }
returntrue;
}

function checkDirExists($dir) {
if(file_exists($dir) && is_dir($dir)) {
returntrue;
    }
returnfalse;
}
```

```
?>
```

This script has a dependency on the Windows Azure SDK for PHP. Also note there are several parameters that you must modify such as the storage account, secret, and backup location. As with SQL Database, bandwidth and transaction charges apply to a backup script like this.

Scalability

Drupal sites on Windows Azure can scale as load increased through typical strategies of scale-up, scale-out, and caching. The following sections describe the specifics of how these strategies are implemented in Windows Azure.

Typically you make scalability decisions based on monitoring and capacity planning. Monitoring can be done in staging during testing or in production with real-time load. Capacity planning factors in projections for changes in user demand.

Scale Up

When you configure your web role prior to deployment, you have the option of specifying the Virtual Machine (VM) size, such as **Small** or **ExtraLarge**. Each size tier adds additional memory, processing power, and network bandwidth to each instance of your web role. For cost efficiency and smaller units of scale, you can test your application under expected load to find the smallest virtual machine size that meets your requirements.

The workload usually in most popular Drupal websites can be separated out into a limited set of Drupal admins making content changes and a large user base who perform mostly read-only workload. End users can be allowed to make 'writes', such as uploading blogs or posting in forums, but those changes are not 'content changes'. Drupal admins are setup to operate without caching so that the writes are made directly to SQL Databaseor the corresponding backend database. This workload performs well with **Large** or **ExtraLarge** VM sizes. Also, note that the VM size is closely tied to all hardware resources, so if there are many content-rich pages that are streaming content, then the VM size requirements are higher.

To make changes to the Virtual Machine size setting, you must change the vmsize attribute of the WebRole element in the service definition file, ServiceDefinition.csdef. A virtual machine size change requires existing applications to be redeployed.

Scale Out

In addition to the size of each web role instance, you can increase or decrease the number of instances that are running the Drupal site. This spreads the web requests across more servers, enabling the site to handle more users. To change the number of running instances of your web role, see How to Scale Applications by Increasing or Decreasing the Number of Role Instances.

Note that some configuration changes can cause your existing web role instances to recycle. You can choose to handle this situation by applying the configuration change and continue running. This is done

by handling the **RoleEnvironment.Changing** event. For more information see, How to Use the RoleEnvironment.Changing Event.

A common question for any Windows Azure solution is whether there is some type of built-in automatic scaling. Windows Azure does not provide a service that provides auto-scaling. However, it is possible to create a custom solution that scales Azure services using the Service Management API. For an example of this approach, see An Auto-Scaling Module for PHP Applications in Windows Azure.

Caching

As mentioned earlier the caching strategy is important in scaling Drupal applications on Windows Azure. One reason for this is that SQL Databaseimplements throttling mechanisms to regulate the load on any one database in the cloud. Code that uses SQL Databaseshould have robust error handling and retry logic to account for this. For more information, see Error Messages (Windows Azure SQL Database). Because of the potential for load-related throttling as well as for general performance improvement, it is strongly recommended to use caching.

Although Windows Azure provides a Caching service, this service does not currently have interoperability with PHP. Because of this, the best solution for caching in Drupal is to use a module that uses an open-source caching technology, such as Memcached.

Outside of a specific Drupal module, you can also configure Memcached to work in PHP for Windows Azure. For more information, see Running Memcached on Windows Azure for PHP. Here is also an example of how to get Memcached working in Windows Azure using a plugin: Windows Azure Memcached plugin.

Here are several design and management considerations related to caching.

Area	Consideration
Design and Implementation	For a technology like Memcached, will the cache be collocated (spread across all web role instances)? Or will you attempt to setup a dedicated cache ring with worker roles that only run Memcached?
Configuration	What memory is required and how will items in the cache be invalidated?
Performance and Monitoring	What mechanisms will be used to detect the performance and overall health of the cache?

For ease of use and cost savings, collocation of the cache across the web role instances of the Drupal site works best. However, this assumes that there is available reserve memory on each instance to apply toward caching. It is possible to increase the virtual machine size setting to increase the amount of available memory on each machine. It is also possible to add additional web role instances to add to the overall memory of the cache while at the same time improving the ability of the web site to respond to load. It is possible to create a dedicated cache cluster in the cloud, but the steps for this are beyond the scope of this paper.

For Windows Azure Blob storage, there is also a caching feature built into the service called the Content Delivery Network (CDN). CDN provides high-bandwidth access to files in Blob storage by caching copies

of the files in edge nodes around the world. Even within a single geographic region, you could see performance improvements as there are many more edge nodes than Windows Azure datacenters. For more information, see Delivering High-Bandwidth Content with the Windows Azure CDN.

Manageability

It is important to note that each hosted service has a **Staging** environment and a **Production** environment. This can be used to manage deployments, because you can load and test and application in staging before performing a VIP swap with production.

From a manageability standpoint, Drupal has an advantage on Windows Azure in the way that site content is stored. Because the data necessary to serve pages is stored in the database and blob storage, there is no need to redeploy the application to change the content of the site.

Another best practice is to use a separate storage account for diagnostic data than the one that is used for the application itself. This can improve performance and also helps to separate the cost of diagnostic monitoring from the cost of the running application.

As mentioned previously, there are several tools that can assist with managing Windows Azure applications. The following table summarizes a few of these choices.

Tool	**Description**
Windows Azure Management Portal	The web interface of the Windows Azure management portal shows deployments, instance counts and properties, and supports many different common management and monitoring tasks.
Azure Diagnostics Manager	A Red Gate Software product that provides advanced monitoring and management of diagnostic data. This tool can be very useful for easily analyzing the performance of the Drupal site to determine appropriate scaling decisions.
Azure Storage Explorer	A tool created by Neudesic for viewing Windows Azure storage account. This can be useful for viewing both diagnostic data and the files in Blob storage.