Microsoft

Developing Microsoft Azure Solutions

Exam Ref 70-532

Zoiner Tejada
Michele Leroux Bustamante
Ike Ellis

Microsoft

Exam Ref 70-532 Developing Microsoft Azure Solutions

Zoiner Tejada
Michele Leroux Bustamante
Ike Ellis

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Control Number: 2014951860
ISBN: 978-0-7356-9704-1

Printed and bound in the United States of America.

First Printing

Acquisitions Editor: Karen Szall
Developmental Editor: Karen Szall
Editorial Production: Box Twelve Communications
Technical Reviewer: Magnus Märtensson; Technical Review services provided by Content Master, a member of CM Group, Ltd.
Cover: Twist Creative • Seattle

Contents

Introduction

This book covers Microsoft Azure from a high-level perspective, consistent with the Microsoft Certification Exam 70-532: Developing Microsoft Azure Solutions. The target audience for this book includes solution architects, DevOps engineers, and QA engineers already familiar with building, deploying, and monitoring scalable solutions with existing development tools, to some extent including Microsoft Azure. The material covered in this book builds on your existing knowledge and experience designing, developing, implementing, automating, and monitoring Microsoft Azure, extending that knowledge to the current state of platform features, development techniques, and management tools. In this book, you'll find coverage of design and implementation concepts, guidance on applying features, step-by-step instructions, and references to appropriate code listings for specific examples.

The 70-532 and 70-533 exams collectively validate that you have the skills and knowledge necessary to design, deploy, and manage Microsoft Azure solutions. This book focuses on exam 70-532 and prepares you from a development and DevOps perspective. Beyond supporting your exam preparation, where possible, we endeavored to include insights from our own experiences helping customers migrate and manage their solutions on the Microsoft Azure platform.

This book covers every exam objective, but it does not cover every exam question. Only the Microsoft exam team has access to the exam questions themselves and Microsoft regularly adds new questions to the exam, making it impossible to cover specific questions. You should consider this book a supplement to your relevant real-world experience and other study materials. If you encounter a topic in this book that you do not feel completely comfortable with, use the links you'll find in text to find more information and take the time to research and study the topic. Great information is available on MSDN, TechNet, and in blogs and forums.

Microsoft certifications

Microsoft certifications distinguish you by proving your command of a broad set of skills and experience with current Microsoft products and technologies. The exams and corresponding certifications are developed to validate your mastery of critical competencies as you design and develop, or implement and support, solutions with Microsoft products and technologies both on-premises and in the cloud. Certification brings a variety of benefits to the individual and to employers and organizations.

MORE INFO **ALL MICROSOFT CERTIFICATIONS**

For information about Microsoft certifications, including a full list of available certifications, go to *http://www.microsoft.com/learning/en/us/certification/cert-default.aspx*.

Acknowledgments

We'd like to thank the following people:

- To Colin Lyth at Microsoft, thank you for recommending us to author this book; we appreciate the opportunity.
- A well-deserved cheers to Devon Musgrave at Microsoft for helping us kick-start the editorial process, and a big thank you to our editor, Karen Szall at Microsoft, for seeing the whole editing process through and dealing with the insanity of an ever-changing platform under tight deadlines. Thank you also to the entire Microsoft Press team working on this book. It's incredible to see all of the effort you put in and how fast you move things forward!
- To the amazing people behind the features of Microsoft Azure: Many of you have provided first class support and guidance by our side to several of our marquee customers whom we have migrated to Azure. To name a few, we thank you Yochay Kieriati, Brady Gaster, Charles Sterling, Anna Timasheva, Suren Machiraju, and others who have enhanced our understanding of the underlying Microsoft Azure platform through our experiences together. Where appropriate, we share these insights with you, dear reader.
- To Brian Noyes, a founding member of Solliance, and several members of our Solliance Partner Network whom we work with regularly to implement Azure solutions: Our collective knowledge base is continually enhanced working together, and certainly that influences the quality of this book.
- To our technical reviewer, Magnus Martensson, thank you for your very thoughtful and detailed review of each chapter and for helping us by turning those reviews around quickly!
- To our families, thank you for your support and patience through the inevitable pressure that comes with publishing. We love you!

Free ebooks from Microsoft Press

From technical overviews to in-depth information on special topics, the free ebooks from Microsoft Press cover a wide range of topics. These ebooks are available in PDF, EPUB, and Mobi for Kindle formats, ready for you to download at:

http://aka.ms/mspressfree

Check back often to see what is new!

Errata, updates, & book support

We've made every effort to ensure the accuracy of this book and its companion content. You can access updates to this book—in the form of a list of submitted errata and their related corrections—at:

http://aka.ms/ER532/errata

If you discover an error that is not already listed, please submit it to us at the same page.

If you need additional support, email Microsoft Press Book Support at mspinput@microsoft.com.

Please note that product support for Microsoft software and hardware is not offered through the previous addresses. For help with Microsoft software or hardware, go to *http://support.microsoft.com*.

We want to hear from you

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

http://aka.ms/tellpress

The survey is short, and we read every one of your comments and ideas. Thanks in advance for your input!

Stay in touch

Let's keep the conversation going! We're on Twitter: *http://twitter.com/MicrosoftPress*.

Preparing for the exam

Microsoft certification exams are a great way to build your resume and let the world know about your level of expertise. Certification exams validate your on-the-job experience and product knowledge. While there is no substitution for on-the-job experience, preparation through study and hands-on practice can help you prepare for the exam. We recommend that you round out your exam preparation plan by using a combination of available study materials and courses. For example, you might use this Exam Ref and another study guide for your "at home" preparation and take a Microsoft Official Curriculum course for the classroom experience. Choose the combination that you think works best for you.

Note that this Exam Ref is based on publicly available information about the exam and the author's experience. To safeguard the integrity of the exam, authors do not have access to the live exam.

CHAPTER 4

Design and implement a storage strategy

Azure Storage and Azure SQL Database both play an important role in the Microsoft Azure Platform-as-a-Service (PaaS) strategy for storage. Azure Storage enables storage and retrieval of large amounts of unstructured data. You can store content files such as documents and media in the Blob service, use the Table service for NoSQL data, use the Queue service for reliable messages, and use the File service for Server Message Block (SMB) file share scenarios. Azure SQL Database provides classic relational database features as part of an elastic scale service.

In this chapter, you will learn how to implement each of the Azure Storage services, how to monitor them, and how to manage access. You'll also learn how to work with Azure SQL Database.

MORE INFO **INTRODUCTION TO STORAGE**

This chapter assumes you have a basic understanding of Azure Storage features. For an introduction to the topic, see *http://azure.microsoft.com/en-us/documentation/articles/storage-introduction/*.

EXAM TIP

There are many ways to interact with and develop against Azure Storage including the management portal, using Windows PowerShell, using client libraries such as those for the .NET Framework, and using the Storage Services REST API. In fact, the REST API is what supports all other options.

Objectives in this chapter:

- Objective 4.1: Implement Azure Storage blobs and Azure files
- Objective 4.2: Implement Azure Storage tables
- Objective 4.3: Implement Azure Storage queues
- Objective 4.4: Manage access
- Objective 4.5: Monitor storage
- Objective 4.6: Implement SQL databases

Objective 4.1: Implement Azure Storage blobs and Azure files

Azure blob storage is the place to store unstructured data of many varieties. You can store images, video files, word documents, lab results, and any other binary file you can think of. In addition, Azure uses blob storage extensively. For instance, when you mount extra logical drives in an Azure virtual machine (VM), the drive image is actually stored in by the Blob service associated with an Azure blob storage account. In a blob storage account, you can have many containers. Containers are similar to folders in that you can use them to logically group your files. You can also set security on the entire container. Each blob storage account can store up to 500 terabytes of data.

All blobs can be accessed through a URL format. It looks like this:

http://<storage account name>.blob.core.windows.net/<container name>/<blob name>

The Azure File service provides an alternative to blob storage for shared storage, accessible via SMB 2.1 protocol.

This objective covers how to:

- Read data
- Change data
- Set metadata on a container
- Store data using block and page blobs
- Stream data using blobs
- Access blobs securely
- Implement async blob copy
- Configure Content Delivery Network (CDN)
- Design blob hierarchies
- Configure custom domains
- Scale blob storage
- Work with file storage

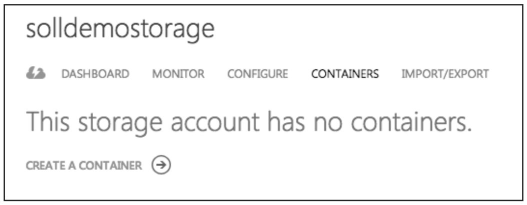
Creating a container

This section explains how to create a container and upload a file to blob storage for later reading.

Creating a container (existing portal)

To create a container in the management portal, complete the following steps:

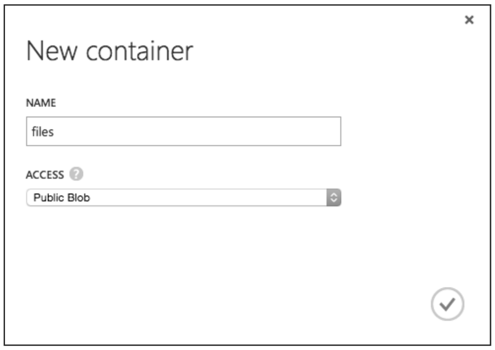
1. Navigate to the Containers tab for your storage account in the management portal accessed via *https://manage.windowsazure.com*.
2. Click Add on the command bar. If you do not yet have a container, you can click Create A Container, as shown in Figure 4-1.



This storage account has no containers.

FIGURE 4-1 The option to create a container for a storage account that has no containers

3. Give the container a name, and select Public Blob for the access rule, as shown in Figure 4-2.



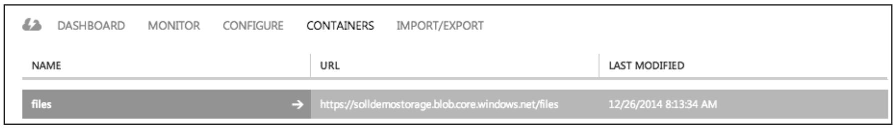
New container

NAME: files

ACCESS: Public Blob

FIGURE 4-2 New container dialog box

4. The URL for the container can be found in the container list, shown in Figure 4-3. You can add additional containers by clicking Add at the bottom of the page on the Containers tab.



NAME	URL	LAST MODIFIED
files	https://solldemostorage.blob.core.windows.net/files	12/26/2014 8:13:34 AM

FIGURE 4-3 Containers tab with a list of containers and their URLs

NOTE **CONTAINER ACCESS PERMISSIONS**

You can choose between the following access permissions on the container:

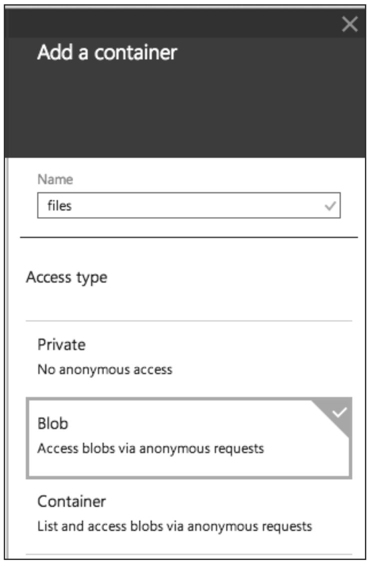
- **Private** All access to the container and its blobs require authentication.
- **Public Container** All access to the container and its blobs are anonymous.
- **Public Blob** You cannot list blobs in the container without authentication, but you can navigate to the blob URL, if you have it, and read it anonymously.

This setting can be changed at any time through the management portal, by using Windows PowerShell, or by configuring it programmatically.

Creating a container (Preview portal)

To create a container in the Preview portal, complete the following steps:

1. Navigate to the management portal accessed via *https://portal.azure.com*.
2. Click Browse on the command bar.
3. Select Storage from the Filter By drop-down list.
4. Select your storage account from the list on the Storage blade.
5. Click the Containers box.
6. On the Containers blade, click Add on the command bar.
7. Enter a name for the container, and select Blob for the access type, as shown in Figure 4-4.

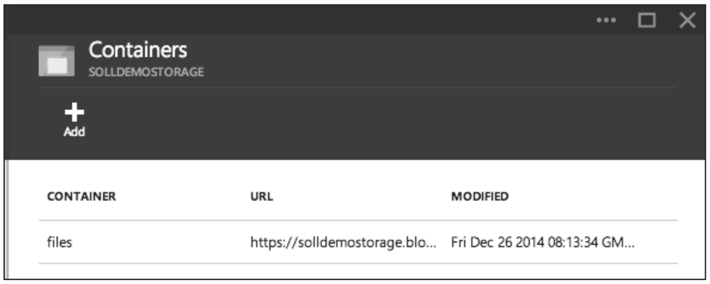

Add a container
Access type
Private
No anonymous access
Blob
Access blobs via anonymous requests
Container
List and access blobs via anonymous requests

Name: files

Access type: [✓ Blob]

FIGURE 4-4 The Add A Container blade

8. The URL for the container can be found in the container list, as shown in Figure 4-5.



CONTAINER	URL	MODIFIED
files	https://solldemostorage.blo...	Fri Dec 26 2014 08:13:34 GM...

FIGURE 4-5 Containers blade with a list of containers and URLs

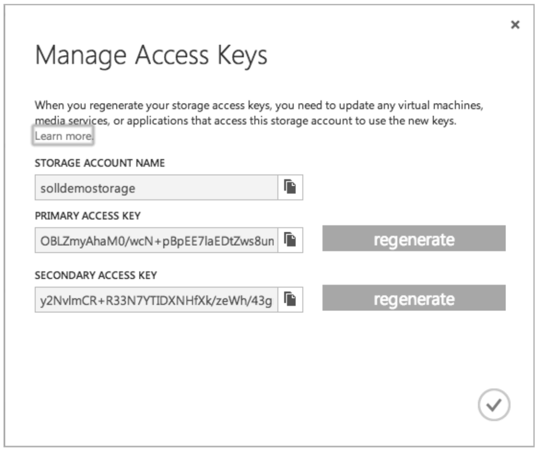
Finding your account access key

To access your storage account, you need the account name that was used to build the URL to the account and the primary access key. This section covers how to find the access keys for storage accounts.

Finding your account access key (existing portal)

To find your account access key using the management portal, complete the following steps:

1. Click the Dashboard tab for your storage account.
2. Click Manage Keys to find the primary and secondary key for managing your account, as shown in Figure 4-6. Always use the primary key for management activities (to be discussed later in this chapter).


Manage Access Keys

When you regenerate your storage access keys, you need to update any virtual machines, media services, or applications that access this storage account to use the new keys.
Learn more

STORAGE ACCOUNT NAME
solldemostorage

PRIMARY ACCESS KEY
OBLZmyAhaM0/wcN+pBpEE7laEDtZws8un

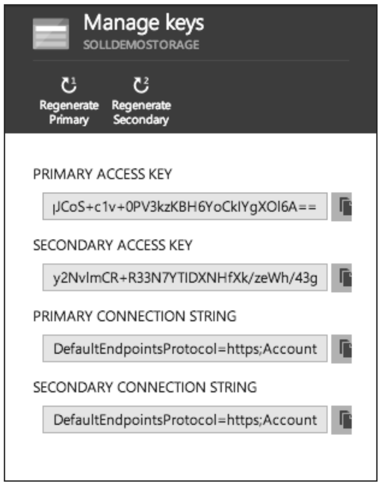
SECONDARY ACCESS KEY
y2NvlmCR+R33N7YTIDXNHfXk/zeWh/43g

FIGURE 4-6 Manage Access Keys dialog box for a storage account

Finding your account access key (Preview portal)

To find your account access key using the Preview portal, complete the following steps:

1. Navigate to your storage account blade.
2. Click the Keys box on the storage account blade (see Figure 4-7).



PRIMARY ACCESS KEY
jJCoS+c1v+0PV3kzKBH6YoCklYgXOl6A==
SECONDARY ACCESS KEY
y2NvlmCR+R33N7YTIDXNHfXk/zeWh/43g
PRIMARY CONNECTION STRING
DefaultEndpointsProtocol=https;Account
SECONDARY CONNECTION STRING
DefaultEndpointsProtocol=https;Account

FIGURE 4-7 Manage Keys blade

Uploading a blob

You can upload files to blob storage using many approaches, including the following:

- Using the AzCopy tool provided by Microsoft (*http://aka.ms/downloadazcopy*)
- Directly using the Storage API and writing HTTP requests
- Using the Storage Client Library, which wraps the Storage API into a language and platform-specific library (*http://msdn.microsoft.com/en-us/library/azure/dn806401.aspx*)
- Using Windows PowerShell cmdlets (*http://msdn.microsoft.com/en-us/library/azure/dn806401.aspx*)

To upload a blob using AzCopy, complete the following steps:

1. Download AZCopy from *http://aka.ms/downloadazcopy*. Run the .msi file downloaded from this link.
2. Open a command prompt and navigate to C:\Program Files (x86)\Microsoft SDKs\Azure\AzCopy.
3. Create a text file in a folder that is easy to get to. Insert some random text in it.

4. In the command window, type a command that looks like this: *AzCopy /Source:c:\test /Dest:https://myaccount.blob.core.windows.net/mycontainer2 /DestKey:key /Pattern:*.txt*.
5. Press Enter to issue the command to transfer the file.

Reading data

You can anonymously read blob storage content directly using a browser if public access to blobs is enabled. The URL to your blob content takes this format:

https://<your account name>.blob.core.windows.net/<your container name>/<your path and filename>

Reading blobs via a browser

Many storage browsing tools provide a way to view the contents of your blob containers. You can also navigate to the container using the existing management portal or the Preview portal to view the list of blobs. When you browse to the blob URL, the file is downloaded and displayed in the browser according to its content type.

Reading blobs using Visual Studio

You can also use Server Manager in Visual Studio 2013 to view the contents of your blob containers and upload or download files.

1. Navigate to the blob storage account that you want to use.
2. Double-click the blob storage account to open a window showing a list of blobs and providing functionality to upload or download blobs.

Changing data

You can modify the contents of a blob or delete a blob using the Storage API directly, but it is more common to do this programmatically as part of an application, for example using the Storage Client Library.

EXAM TIP

Any updates made to a blob are atomic. While an update is in progress, requests to the blob URL will always return the previously committed version of the blob until the update is complete.

The following steps illustrate how to update a blob programmatically. Note that this example uses a block blob. The distinction between block and page blobs is discussed in "Storing data using block and page blobs" later in this chapter.

1. Create a C# console application.

2. In your app.config file, create a storage configuration string and entry, replacing AccountName and AccountKey with your storage account values:

```
<configuration>
  <appSettings>
    <add key=”StorageConnectionString” value=”DefaultEndpointsProtocol=https;Accou
ntName=<your account name>;AccountKey=<your account key>” />
  </appSettings>
</configuration>
```

3. Use NuGet to obtain the Microsoft.WindowsAzure.Storage.dll. An easy way to do this is by using this command in the NuGet console:

```
Install-package windowsazure.storage –version 3.0.3
```

4. Create a new console application, and add the following using statements to the top of your Program.cs file:

```
using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Auth;
using Microsoft.WindowsAzure;
using Microsoft.WindowsAzure.Storage.Blob;
using System.Configuration
```

5. Add a reference to System.Configuration. Add the following code in the main entry point:

```
var storageAccount = CloudStorageAccount.Parse( ConfigurationManager.AppSettings[“
StorageConnectionString”]);
```

6. Use CloudBlobClient to gain access to the containers and blobs in your Azure storage account. After it is created, you can set permissions to make it publicly available:

```
CloudBlobClient blobClient = storageAccount.CreateCloudBlobClient();
```

7. Use a CreateIfNotExists method to ensure a container is there before you interact with it:

```
CloudBlobContainer container = blobClient.GetContainerReference(“files”);
container.CreateIfNotExists();

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

8. To upload a file, use the FileStream object to access the stream, and then use the UploadFromFileStream method on the CloudBlockBlob class to upload the file to Azure blob storage:

```
CloudBlockBlob blockBlob = container.GetBlockBlobReference(“myblob”);
using (var fileStream = System.IO.File.OpenRead(@”path\myfile”))
{
  blockBlob.UploadFromStream(fileStream);
}
```

9. To list all of the blobs, use the following code:

```
foreach (IListBlobItem item in container.ListBlobs(null, false))
{
  if (item.GetType() == typeof(CloudBlockBlob))
  {
    CloudBlockBlob blob = (CloudBlockBlob)item;
    Console.WriteLine("Block blob of length {0}: {1}", blob.Properties.Length,
blob.Uri);

  }
  else if (item.GetType() == typeof(CloudPageBlob))
  {
    CloudPageBlob pageBlob = (CloudPageBlob)item;
    Console.WriteLine("Page blob of length {0}: {1}", pageBlob.Properties.Length,
pageBlob.Uri);
  }
  else if (item.GetType() == typeof(CloudBlobDirectory))
  {
    CloudBlobDirectory directory = (CloudBlobDirectory)item;
    Console.WriteLine("Directory: {0}", directory.Uri);
  }
}
```

10. To download blobs, use the CloudBlobContainer class:

```
CloudBlockBlob blockBlob = container.GetBlockBlobReference("photo1.jpg");
using (var fileStream = System.IO.File.OpenWrite(@"path\myfile"))
{
  blockBlob.DownloadToStream(fileStream);
}
```

11. To delete a blob, get a reference to the blob and call Delete():

```
CloudBlockBlob blockBlob = container.GetBlockBlobReference("myblob.txt");
blockBlob.Delete();
```

Setting metadata on a container

Blobs and containers have metadata attached to them. There are two forms of metadata:

- System properties metadata
- User-defined metadata

System properties can influence how the blob behaves, while user-defined metadata is your own set of name/value pairs that your applications can use. A container has only read-only system properties, while blobs have both read-only and read-write properties.

Setting user-defined metadata

To set user-defined metadata for a container, get the container reference using GetContainerReference(), and then use the Metadata member to set values. After setting all the desired values, call SetMetadata() to persist the values, as in the following example:

```
CloudBlobContainer container = blobClient.GetContainerReference("files");
files.Metadata["counter"] = "100";
files.SetMetadata();
```

MORE INFO **BLOB METADATA**

Blob metadata includes both read-only and read-write properties that are valid HTTP headers and follow restrictions governing HTTP headers. The total size of the metadata is limited to 8 KB for the combination of name and value pairs. For more information on interacting with individual blob metadata, see *http://msdn.microsoft.com/en-us/library/azure/hh225342.aspx*.

Reading user-defined metadata

To read user-defined metadata for a container, get the container reference using GetContainerReference(), and then use the Metadata member to retrieve a dictionary of values and access them by key, as in the following example:

```
CloudBlobContainer container = blobClient.GetContainerReference("files");
Console.WriteLine("counter value: " + files.Metadata["counter"];
```

EXAM TIP

If the metadata key doesn't exist, an exception is thrown.

Reading system properties

To read a container's system properties, first get a reference to the container using GetContainerReference(), and then use the Properties member to retrieve values. The following code illustrates accessing container system properties:

```
CloudBlobContainer container = blobClient.GetContainerReference("files");
Console.WriteLine("LastModifiedUTC: " + container.Properties.LastModified);
Console.WriteLine("ETag: " + container.Properties.ETag);
```

MORE INFO **CONTAINER METADATA AND THE STORAGE API**

You can request container metadata using the Storage API. For more information on this and the list of system properties returned, see *http://msdn.microsoft.com/en-us/library/azure/dd179370.aspx*.

Storing data using block and page blobs

The Azure Blob service has two different ways of storing your data: block blobs and page blobs. Block blobs are great for streaming data sequentially, like video and other files. Page blobs are great for non-sequential reads and writes, like the VHD on a hard disk mentioned in earlier chapters.

Block blobs are blobs that are divided into blocks. Each block can be up to 4 MB. When uploading large files into a block blob, you can upload one block at a time in any order you want. You can set the final order of the block blob at the end of the upload process. For large files, you can also upload blocks in parallel. Each block will have an MD5 hash used to verify transfer. You can retransmit a particular block if there's an issue. You can also associate blocks with a blob after upload, meaning that you can upload blocks and then assemble the block blob after the fact. Any blocks you upload that aren't committed to a blob will be deleted after a week. Block blobs can be up to 200 GB.

Page bobs are blobs comprised of 512-byte pages. Unlike block blobs, page blob writes are done in place and are immediately committed to the file. The maximum size of a page blob is 1 terabyte. Page blobs closely mimic how hard drives behave, and in fact, Azure VMs use them for that purpose. Most of the time, you will use block blobs.

Streaming data using blobs

You can stream blobs by downloading to a stream using the DownloadToStream() API method. The advantage of this is that it avoids loading the entire blob into memory, for example before saving it to a file or returning it to a web request.

Accessing blobs securely

Secure access to blob storage implies a secure connection for data transfer and controlled access through authentication and authorization.

Azure Storage supports both HTTP and secure HTTPS requests. For data transfer security, you should always use HTTPS connections. To authorize access to content, you can authenticate in three different ways to your storage account and content:

- **Shared Key** Constructed from a set of fields related to the request. Computed with a SHA-256 algorithm and encoded in Base64.
- **Shared Key Lite** Similar to Shared Key, but compatible with previous versions of Azure Storage. This provides backwards compatibility with code that was written against versions prior to 19 September 2009. This allows for migration to newer versions with minimal changes.
- **Shared Access Signature** Grants restricted access rights to containers and blobs. You can provide a shared access signature to users you don't trust with your storage account key. You can give them a shared access signature that will grant them specific permissions to the resource for a specified amount of time. This is discussed in a later section.

To interact with blob storage content authenticated with the account key, you can use the Storage Client Library as illustrated in earlier sections. When you create an instance of the CloudStorageAccount using the account name and key, each call to interact with blob storage will be secured, as shown in the following code:

```
string accountName = "ACCOUNTNAME";
string accountKey = "ACCOUNTKEY";
CloudStorageAccount storageAccount = new CloudStorageAccount(new
StorageCredentials(accountName, accountKey), true);
```

Implementing an async blob copy

The Blob service provides a feature for asynchronously copying blobs from a source blob to a destination blob. You can run many of these requests in parallel since the operation is asynchronous. The following scenarios are supported:

- Copying a source blob to a destination with a different name or URI
- Overwriting a blob with the same blob, which means copying from the same source URI and writing to the same destination URI (this overwrites the blob, replaces meta-data, and removes uncommitted blocks)
- Copy a snapshot to a base blob, for example to promote the snapshot to restore an earlier version
- Copy a snapshot to a new location creating a new, writable blob (not a snapshot)

The copy operation is always the entire length of the blob; you can't copy a range.

MORE INFO **COPY BLOB**

For additional details on the underlying process for copying blobs, see *http://msdn.microsoft.com/en-us/library/azure/dd894037.aspx*.

The following code illustrates a simple example for creating a blob and then copying it asynchronously to another destination blob:

```
CloudBlobContainer files = blobClient.GetContainerReference("files");
files.CreateIfNotExists(BlobContainerPublicAccessType.Off);
ICloudBlob sourceBlob = files.GetBlockBlobReference("filetocopy.txt");
sourceBlob.Properties.ContentType = "text/plain";
string sourceFileContents = "my text blob to copy";
byte[] sourceBytes = new byte[sourceFileContents.Length * sizeof(char)];
System.Buffer.BlockCopy(sourceFileContents.ToCharArray(), 0, sourceBytes, 0,
sourceBytes.Length);
sourceBlob.UploadFromByteArray(sourceBytes, 0, sourceBytes.Length);

ICloudBlob blobCopy = files.GetBlockBlobReference("destinationcopy.txt");
AsyncCallback cb = new AsyncCallback(x => Console.WriteLine("copy completed with {0}",
x.IsCompleted));
blobCopy.BeginStartCopyFromBlob(sourceBlob.Uri, cb, null);
```

Ideally, you pass state to the BeginStartCopyFromBlob() method so that you can track multiple parallel operations.

EXAM TIP

A storage account can have multiple Copy Blob operations processing in parallel; however, an individual blob can have only one pending copy operation.

Configuring the Content Delivery Network

The Azure Content Delivery Network (CDN) distributes content across geographic regions to edge nodes across the globe. The CDN caches publicly available objects so they are available over high-bandwidth connections, close to the users, thus allowing the users to download them at much lower latency. You may be familiar with using CDNs to download popular Javascript frameworks like JQuery, Angular, and others.

By default, blobs have a seven-day time-to-live (TTL) at the CDN edge node. After that time elapses, the blob is refreshed from the storage account to the edge node. Blobs that are shared via CDN must support anonymous access.

Configuring the CDN (existing portal)

To enable the CDN for a storage account in the management portal, complete the following steps:

1. In the management portal, click New on the navigation bar.
2. Select App Services, CDN, Quick Create.
3. Select the storage account that you want to add CDN support for, and click Create.
4. Navigate to the CDN properties by selecting it from your list of CDN endpoints.
5. To enable HTTPS support, click Enable HTTPS at the bottom of the page.
6. To enable query string support, click Enable Query String Support at the bottom of the page.
7. To map a custom domain to the CDN endpoint, click Manage Domains at the bottom of the page, and follow the instructions.

EXAM TIP

It can take 60 minutes before the CDN is ready for use on the storage account.

To access blobs via CDN, use the CDN address as follows:

```
http://<your CDN subdomain>.vo.msecnd.net/<your container name>/<your blob path>
```

If you are using HTTPS and a custom domain, address your blobs as follows:

```
https://<your domain>/<your container name>/<your blob path>
```

Configuring the CDN (Preview portal)

You currently cannot configure the CDN using the Preview portal.

Designing blob hierarchies

Blob storage has a hierarchy that involves the following aspects:

- The storage account name, which is part of the base URI
- The container within which you store blobs, which is also used for partitioning
- The blob name, which can include path elements separated by a backslash (/) to create a sense of folder structure

Using a blob naming convention that resembles a directory structure provides you with additional ways to filter your blob data directly from the name. For example, to group images by their locale to support a localization effort, complete the following steps:

1. Create a container called **images**.
2. Add English bitmaps using the convention en/bmp/*, where * is the file name.
3. Add English JPEG files using the convention en/jpg/*, where * is the file name.
4. Add Spanish bitmaps using the convention sp/bmp/*, where * is the file name.
5. Add Spanish JPEG files using the convention sp/jpg/*, where * is the file name.

To retrieve all images in the container, use ListBlob() in this way:

```
var list = images.ListBlobs(null, true, BlobListingDetails.All);
```

The output is the entire list of uploaded images in the container:

```
https://solexpstorage.blob.core.windows.net/images/en/bmp/logo.bmp
https://solexpstorage.blob.core.windows.net/images/en/jpg/logo.jpg
https://solexpstorage.blob.core.windows.net/images/sp/bmp/logo.bmp
https://solexpstorage.blob.core.windows.net/images/sp/jpg/logo.jpg
```

To filter only those with the prefix *en*, use this:

```
var list = images.ListBlobs("en", true, BlobListingDetails.All);
```

The output will be this:

```
https://solexpstorage.blob.core.windows.net/images/en/bmp/logo.bmp
https://solexpstorage.blob.core.windows.net/images/en/jpg/logo.jpg
```

Configuring custom domains

By default, the URL for accessing the Blob service in a storage account is *https://<your account name>.blob.core.windows.net*. You can map your own domain or subdomain to the Blob service for your storage account so that users can reach it using the custom domain or subdomain.

MORE INFO **STORAGE AND CUSTOM DOMAINS**

For details related to mapping custom domains to your storage accounts, see *http://azure.microsoft.com/en-us/documentation/articles/storage-custom-domain-name/*.

Scaling Blob storage

Blobs are partitioned by container name and blob name, which means each blob can have its own partition. Blobs, therefore, can be distributed across many servers to scale access even though they are logically grouped within a container.

MORE INFO **STORAGE SCALABILITY TARGETS**

For additional information on storage scalability, see *http://msdn.microsoft.com/en-us/library/dn249410.aspx*.

Working with Azure File storage

Azure File storage provides a way for applications to share storage accessible via SMB 2.1 protocol. It is particularly useful for VMs and cloud services as a mounted share, and applications can use the File Storage API to access File storage.

MORE INFO **FILE STORAGE DOCUMENTATION**

For additional information on File storage, see the guide at *http://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-how-to-use-files/*.

Thought experiment

Partitioning strategy for localization

In this thought experiment, apply what you've learned about this objective. You can find answers to these questions in the "Answers" section at the end of this chapter.

You are localizing a mobile application for multiple languages. Some of your efforts revolve around having separate images for the regions you are targeting.

1. How will you structure the files in Blob storage so that you can retrieve them easily?
2. What can you do to make access to these images quick for users around the world?

Objective summary

- A blob container has several options for access permissions. When set to Private, all access requires credentials. When set to Public Container, no credentials are required to access the container and its blobs. When set to Public Blob, only blobs can be accessed without credentials if the full URL is known.
- To access secure containers and blobs, you can use the storage account key or a shared access signatures.
- AzCopy is a useful utility for activities such as uploading blobs, transferring blobs from one container or storage account to another, and performing these and other activities related to blob management in scripted batch operations.
- Block blobs allow you to upload, store, and download large blobs in blocks up to 4 MB each. The size of the blob can be up to 200 GB.
- You can use a blob naming convention akin to folder paths to create a logical hierarchy for blobs, which is useful for query operations.

Objective review

Answer the following questions to test your knowledge of the information in this objective. You can find the answers to these questions and explanations of why each answer choice is correct or incorrect in the "Answers" section at the end of this chapter.

1. Which of the following is *not* true about metadata? (Choose all that apply.)
   - **A.** Both containers and blobs have writable system properties.
   - **B.** Blob user-defined metadata is accessed as a key value pair.
   - **C.** System metadata can influence how the blog is stored and accessed in Azure Storage.
   - **D.** Only blobs have metadata; containers do not.
2. Which of the following are valid differences between page blobs and block blobs? (Choose all that apply.)
   - **A.** Page blobs are much faster for all operations.
   - **B.** Block blobs allow files to be uploaded and assembled later. Blocks can be resubmitted individually.
   - **C.** Page blobs are good for all sorts of files, like video and images.
   - **D.** Block blobs have a max size of 200 GB. Page blobs can be 1 terabyte.
3. What are good recommendations for securing files in Blob storage? (Choose all that apply.)
   - **A.** Always use SSL.
   - **B.** Keep your primary and secondary keys hidden and don't give them out.

C. In your application, store them someplace that isn't embedded in client-side code that users can see.

D. Make the container publicly available.

Objective 4.2: Implement Azure Storage tables

Azure Storage is a non-relational (NoSQL) entity storage service on Microsoft Azure. When you create a storage account, it includes the Table service alongside the Blob and Queue services. Table services can be accessed through a URL format. It looks like this:

http://<storage account name>.table.core.windows.net/<table name>.

There are many forms of NoSQL databases:

- Key-value stores that organize data with a unique key per record and often allow for jagged entries where each row might not have a complete set of values.
- Document databases that are similar to key-value stores with semi-structured, easy-to-query documents. Usually, information is stored in JavaScript Object Notation (JSON) format.
- Columnar stores that are used to organize large amounts of distributed information.
- Graph databases that do not use columns and rows; instead, they use a graph model for storage and query, usually for large amounts of highly distributed data.

Table storage is a key-value store that uses a partition key to help with scale out distribution of data and a row key for unique access to a particular entry. Together, these keys are used to uniquely identify a record in the account.

This objective covers how to:

- Implement CRUD with and without transactions
- Query using OData
- Design, manage, and scale table partitions

Using basic CRUD operations

In this section, you learn how to access table storage programmatically.

Creating a table

1. Create a C# console application.

2. In your app.config file, add an entry under the Configuration element, replacing the account name and key with your own storage account details:

```
<configuration>
  <appSettings>
    <add key=”StorageConnectionString” value=”DefaultEndpointsProtocol=https;Accou
ntName=<your account name>;AccountKey=<your account key>” />
  </appSettings>
</configuration>
```

3. Use NuGet to obtain the Microsoft.WindowsAzure.Storage.dll. An easy way to do this is by using the following command in the NuGet console:

```
Install-package windowsazure.storage –version 3.0.3
```

4. Add the following using statements to the top of your Program.cs file:

```
using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Auth;
using Microsoft.WindowsAzure.Storage.Table;
using Microsoft.WindowsAzure;
using System.Configuration
```

5. Add a reference to System.Configuration.

6. Type the following command to retrieve your connection string in the Main function of Program.cs:

```
var storageAccount = CloudStorageAccount.Parse(
var storageAccount =CloudStorageAccount.Parse( ConfigurationManager.AppSettings[“S
torageConnectionString”]);
```

7. Use the following command to create a table if one doesn’t already exist:

```
CloudTableClient tableClient = storageAccount.CreateCloudTableClient();
CloudTable table = tableClient.GetTableReference(“customers”);
table.CreateIfNotExists();
```

Inserting records

To add entries to a table, you create objects based on the TableEntity base class and serialize them into the table using the Storage Client Library. The following properties are provided for you in this base class:

- **Partition Key** Used to partition data across storage infrastructure
- **Row Key** Unique identifier in a partition
- **Timestamp** Time of last update maintained by Azure Storage
- **ETag** Used internally to provide optimistic concurrency

The combination of partition key and row key must be unique within the table. This combination is used for load balancing and scaling, as well as for querying and sorting entities.

Follow these steps to add code that inserts records:

1. Add a class to your project, and then add the following code to it:

```
using Microsoft.WindowsAzure.Storage.Table;
public class OrderEntity : TableEntity
{
 public OrderEntity(string customerName, String orderDate)
 {
  this.PartitionKey = customerName;
  this.RowKey = orderDate;
 }
 public OrderEntity() { }
 public string OrderNumber { get; set; }
 public DateTime RequiredDate { get; set; }
 public DateTime ShippedDate { get; set; }
 public string Status { get; set; }
}
```

2. Add the following code to the console program to insert a record:

```
CloudStorageAccount storageAccount = CloudStorageAccount.Parse(
CloudConfigurationManager.GetSetting("StorageConnectionString"));
CloudTableClient tableClient = storageAccount.CreateCloudTableClient();
CloudTable table = tableClient.GetTableReference("orders");
OrderEntity newOrder = new OrderEntity("Archer", "20141216");
newOrder.OrderNumber = "101";
newOrder.ShippedDate = Convert.ToDateTime("20141218");
newOrder.RequiredDate = Convert.ToDateTime("20141222");
newOrder.Status = "shipped";
TableOperation insertOperation = TableOperation.Insert(newOrder);
table.Execute(insertOperation);
```

Inserting multiple records in a transaction

You can group inserts and other operations into a single batch transaction. All operations in the batch must take place on the same partition. You can have up to 100 entities in a batch. The total batch payload size cannot be greater than 4 MB.

The following code illustrates how to insert several records as part of a single transaction:

```
CloudStorageAccount storageAccount = CloudStorageAccount.Parse(
CloudConfigurationManager.GetSetting("StorageConnectionString"));
CloudTableClient tableClient = storageAccount.CreateCloudTableClient();
CloudTable table = tableClient.GetTableReference("orders");
TableBatchOperation batchOperation = new TableBatchOperation();

OrderEntity newOrder1 = new OrderEntity("Lana", "20141217");
newOrder1.OrderNumber = "102";
newOrder1.ShippedDate = Convert.ToDateTime("1/1/1900");
newOrder1.RequiredDate = Convert.ToDateTime("1/1/1900");
newOrder1.Status = "pending";
OrderEntity newOrder2 = new OrderEntity("Lana", "20141218");
newOrder2.OrderNumber = "103";
newOrder2.ShippedDate = Convert.ToDateTime("1/1/1900");
```

```
newOrder2.RequiredDate = Convert.ToDateTime("12/25/2014");
newOrder2.Status = "open";
OrderEntity newOrder3 = new OrderEntity("Lana", "20141219");
newOrder3.OrderNumber = "103";
newOrder3.ShippedDate = Convert.ToDateTime("12/17/2014");
newOrder3.RequiredDate = Convert.ToDateTime("12/17/2014");
newOrder3.Status = "shipped";

batchOperation.Insert(newOrder1);
batchOperation.Insert(newOrder2);
batchOperation.Insert(newOrder3);
table.ExecuteBatch(batchOperation);
```

MORE INFO **ENTITY GROUP TRANSACTIONS**

You can batch transactions that belong to the same table and partition group for insert, update, merge, delete, and related actions programmatically or by using the Storage API. For more information, see the reference at *http://msdn.microsoft.com/en-us/library/dd894038.aspx*.

Getting records in a partition

You can select all of the entities in a partition or a range of entities by partition and row key. Wherever possible, you should try to query with the partition key and row key. Querying entities by other properties does not work well because it launches a scan of the entire table.

Within a table, entities are ordered within the partition key. Within a partition, entities are ordered by the row key. RowKey is a string property, so sorting is handled as a string sort. If you are using a date value for your RowKey property use the following order: year, month, day. For instance, use 20140108 for January 8, 2014.

The following code requests all records within a partition using the PartitionKey property to query:

```
CloudStorageAccount storageAccount = CloudStorageAccount.Parse(
CloudConfigurationManager.GetSetting("StorageConnectionString"));
CloudTableClient tableClient = storageAccount.CreateCloudTableClient();
CloudTable table = tableClient.GetTableReference("orders");
TableQuery<OrderEntity> query = new TableQuery<OrderEntity>().Where(
TableQuery.GenerateFilterCondition("PartitionKey", QueryComparisons.Equal, "Lana"));

foreach (OrderEntity entity in table.ExecuteQuery(query))
{
 Console.WriteLine("{0}, {1}\t{2}\t{3}", entity.PartitionKey, entity.RowKey,
 entity.Status, entity.RequiredDate);
}
```

Updating records

One technique you can use to update a record is to use InsertOrReplace(). This creates the record if one does not already exist or updates an existing record. Here's an example:

```
CloudStorageAccount storageAccount = CloudStorageAccount.
Parse(CloudConfigurationManager.GetSetting("StorageConnectionString"));
CloudTableClient tableClient = storageAccount.CreateCloudTableClient();
CloudTable table = tableClient.GetTableReference("orders1");
TableOperation retrieveOperation = TableOperation.Retrieve<OrderEntity>("Lana",
"20141217");
TableResult retrievedResult = table.Execute(retrieveOperation);
OrderEntity updateEntity = (OrderEntity)retrievedResult.Result;
if (updateEntity != null)
{
  updateEntity.Status = "shipped";
  updateEntity.ShippedDate = Convert.ToDateTime("12/20/2014");
  TableOperation insertOrReplaceOperation = TableOperation.
InsertOrReplace(updateEntity);
  table.Execute(insertOrReplaceOperation);
}
```

Deleting a record

To delete a record, first retrieve the record as shown in earlier examples, and then delete it with code, such as this:

```
TableOperation deleteOperation = TableOperation.Delete(deleteEntity);
table.Execute(deleteOperation);
Console.WriteLine("Entity deleted.");
```

Querying using ODATA

The Storage API for tables supports OData, which exposes a simple query interface for interacting with table data. Table storage does not support anonymous access, so you must supply credentials using the account key or a Shared Access Signature (SAS) (discussed in "Manage Access") before you can perform requests using OData.

To query what tables you have created, provide credentials, and issue a GET request as follows:

```
https://myaccount.table.core.windows.net/Tables
```

To query the entities in a specific table, provide credentials, and issue a GET request formatted as follows:

```
https://<your account name>.table.core.windows.net/<your table
name>(PartitionKey='<partition-key>',RowKey='<row-key>')?$select=<comma separated
property names>
```

NOTE **QUERY LIMITATIONS**

The result is limited to 1,000 entities per request, and the query will run for a maximum of five seconds.

MORE INFO **ODATA**

For more information on OData, see the reference at *http://msdn.microsoft.com/en-us/library/azure/dn535600.aspx*.

Designing, managing, and scaling table partitions

The Azure Table service can scale to handle massive amounts of structured data and billions of records. To handle that amount, tables are partitioned. The partition key is the unit of scale for storage tables. The table service will spread your table to multiple servers and key all rows with the same partition key co-located. Thus, the partition key is an important grouping, not only for querying but also for scalability.

There are three types of partition keys to choose from:

- **Single value** There is one partition key for the entire table. This favors a small number of entities. It also makes batch transactions easier since batch transactions need to share a partition key to run without error. It does not scale well for large tables since all rows will be on the same partition server.
- **Multiple values** This might place each partition on its own partition server. If the partition size is smaller, it's easier for Azure to load balance the partitions. Partitions might get slower as the number of entities increases. This might make further partitioning necessary at some point.
- **Unique values** This is many small partitions. This is highly scalable, but batch transactions are not possible.

For query performance, you should use the partition key and row key together when possible. This leads to an exact row match. The next best thing is to have an exact partition match with a row range. It is best to avoid scanning the entire table.

Thought experiment

Partitioning data

In this thought experiment, apply what you've learned about this objective. You can find answers to these questions in the "Answers" section at the end of this chapter.

Your organization produces lasers that etch patterns for processors and memory. On a regular interval, the laser sends the shot count of how many times the laser fired to your application. The shot count is cumulative by day. Your company built more than 500 of these lasers and distributed them around the world. Each laser has its own machine identifier. Each time the shot count is sent, it includes a time stamp. The analysts are mostly concerned with the most recent shot count sent.

1. **What should you use for the partition key? How many partitions should you create?**
2. **How should you create the row key?**
3. **How many tables should you build? What's in each table?**

Objective summary

- Table storage is a non-relational database implementation (NoSQL) following the key-value database pattern.
- Table entries each have a partition key and row key. The partition key is used to logically group rows that are related; the row key is a unique entry for the row.
- The Table service uses the partition key for distributing collections of rows across physical partitions in Azure to automatically scale out the database as needed.
- A Table storage query returns up to 1,000 records per request, and will time out after five seconds.
- Querying Table storage with both the partition and row key results in fast queries. A table scan is required for queries that do not use these keys.

Objective review

Answer the following questions to test your knowledge of the information in this objective. You can find the answers to these questions and explanations of why each answer choice is correct or incorrect in the "Answers" section at the end of this chapter.

1. Which of the following is not a method for replicating a Table storage account?
   - **A.** Transactional replication
   - **B.** Zone redundant storage
   - **C.** Read access geo-redundant storage
   - **D.** Geo-redundant storage

2. How should you choose a good partition key for a Table storage implementation? (Choose all that apply.)
   - **A.** They should always be unique, like a primary key in a SQL table.
   - **B.** You should always use the same partition key for all records.
   - **C.** Think about how you're likely to update the data using batch transactions.
   - **D.** Find an even way to split them so that you have relatively even partition sizes.
3. Which of the following statements are correct for submitting operations in a batch? (Choose all that apply.)
   - **A.** All operations have to be in the same partition.
   - **B.** Total batch size can't be greater than 4 MB.
   - **C.** Max operation count is 100.
   - **D.** Minimum operation count is three.

Objective 4.3: Implement Azure storage queues

The Azure Storage Queue service provides a mechanism for reliable inter-application messaging to support asynchronous distributed application workflows. This section covers a few fundamental features of the Queue service for adding messages to a queue, processing those messages individually or in a batch, and scaling the service.

MORE INFO **QUEUE SERVICE**

For a general overview of working with the Queue service, see the reference at *http://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-how-to-use-queues/*.

This objective covers how to:

- Add and process messages
- Retrieve a batch of messages
- Scale queues

Adding messages to a queue

You can access your storage queues and add messages to a queue using many storage browsing tools; however, it is more likely you will add messages programmatically as part of your application workflow.

The following code demonstrates how to add messages to a queue:

```
string connection = "DefaultEndpointsProtocol=https;AccountName=<ACCOUNTNAME>;AccountKey
=<ACCOUNTKEY>";
CloudStorageAccount account;
if (!CloudStorageAccount.TryParse(connection, out account))
{
 throw new Exception("Unable to parse storage account connection string.");
}
CloudQueueClient queueClient = account.CreateCloudQueueClient();
CloudQueue queue = queueClient.GetQueueReference("workerqueue");
queue.AddMessage(new CloudQueueMessage("Queued message 1"));
queue.AddMessage(new CloudQueueMessage("Queued message 2"));
queue.AddMessage(new CloudQueueMessage("Queued message 3"));
```

NOTE **MESSAGE IDENTIFIERS**

The Queue service assigns a message identifier to each message when it is added to the queue. This is opaque to the client, but it is used by the Storage Client Library to identify a message uniquely when retrieving, processing, and deleting messages.

MORE INFO **LARGE MESSAGES**

There is a limit of 64 KB per message stored in a queue. It is considered best practice to keep the message small and to store any required data for processing in a durable store, such as SQL Azure, storage tables, or storage blobs. This also increases system reliability since each queued message can expire after seven days if not processed. For more information, see the reference at *http://msdn.microsoft.com/en-us/library/azure/hh690942.aspx*.

Processing messages

Messages are typically published by a separate application in the system from the application that listens to the queue and processes messages. As shown in the previous section, you can create a CloudQueue reference and then proceed to call GetMessage() to de-queue the next available message from the queue as follows:

```
CloudQueueMessage message = queue.GetMessage(new TimeSpan(0, 5, 0));
if (message != null)
{
 string theMessage = message.AsString;
 // your processing code goes here
}
```

NOTE **INVISIBILITY SETTING**

By default, when you de-queue a message, it is invisible to the queue for 30 seconds. In the event message processing exceeds this timeframe, supply an alternate setting for this value when creating or updating the message. You can set the timeout to a value between one second and seven days. Visibility can also exceed the message expiry time.

Retrieving a batch of messages

A queue listener can be implemented as single-threaded (processing one message at a time) or multi-threaded (processing messages in a batch on separate threads). You can retrieve up to 32 messages from a queue using the GetMessages() method to process multiple messages in parallel. As discussed in the previous sections, create a CloudQueue reference, and then proceed to call GetMessages(). Specify the number of items to de-queue up to 32 (this number can exceed the number of items in the queue) as follows:

```
IEnumerable<CloudQueueMessage> batch = queue.GetMessages(10, new TimeSpan(0, 5, 0));
foreach (CloudQueueMessage batchMessage in batch)
{
 Console.WriteLine(batchMessage.AsString);
}
```

NOTE **PARALLEL PROCESSING OVERHEAD**

Consider the overhead of message processing before deciding the appropriate number of messages to process in parallel. If significant memory, disk space, or other network resources are used during processing, throttling parallel processing to an acceptable number will be necessary to avoid performance degradation on the compute instance.

Scaling queues

When working with Azure Storage queues, you need to consider a few scalability issues, including the messaging throughput of the queue itself and the design topology for processing messages and scaling out as needed.

Each individual queue has a target of approximately 2,000 messages per second (assuming a message is within 1 KB). You can partition your application to use multiple queues to increase this throughput value.

As for processing messages, it is more cost effective and efficient to pull multiple messages from the queue for processing in parallel on a single compute node; however, this depends on the type of processing and resources required. Scaling out compute nodes to increase processing throughput is usually also required.

As discussed in Chapter 2, “Create and manage virtual machines,” and Chapter 3, “Design and implement cloud services,” you can configure VMs or cloud services to auto-scale by queue. You can specify the average number of messages to be processed per instance, and the auto-scale algorithm will queue to run scale actions to increase or decrease available instances accordingly.

MORE INFO **BACK OFF POLLING**

To control storage costs, you should implement a back off polling algorithm for queue message processing. This and other scale considerations are discussed in the reference at *http://msdn.microsoft.com/en-us/library/azure/hh697709.aspx*.

Thought experiment

Asynchronous design patterns

In this thought experiment, apply what you've learned about this objective. You can find answers to these questions in the "Answers" section at the end of this chapter.

Your application must, on user request, generate PDF reports that include both data stored in SQL Azure and images stored in storage blobs. Producing these reports requires significant memory per report and local disk storage prior to saving reports in Blob storage. There are 50,000 users that could potentially request these reports daily; however, the number of requests per day varies widely.

1. **How would you design the system to handle asynchronous processing of these PDF reports?**
2. **Which type of compute instance would you choose?**
3. **How many reports would you process on a single compute instance?**
4. **How would you approach scaling the number of compute instances according to the number of requests?**

Objective summary

- Applications can add messages to a queue programmatically using the .NET Storage Client Library or equivalent for other languages, or you can directly call the Storage API.
- Messages are stored in a storage queue for up to seven days based on the expiry setting for the message. Message expiry can be modified while the message is in the queue.
- An application can retrieve messages from a queue in batch to increase throughput and process messages in parallel.
- Each queue has a target of approximately 2,000 messages per second. You can increase this throughput by partitioning messages across multiple queues.

Objective review

Answer the following questions to test your knowledge of the information in this objective. You can find the answers to these questions and explanations of why each answer choice is correct or incorrect in the "Answers" section at the end of this chapter.

1. Which of the following statements are true about queuing messages? (Choose all that apply.)
   - **A.** Storage queue messages have no size restrictions. The reason for using smaller messages sizes is to increase throughput to the queue.
   - **B.** Storage queue messages are limited to 64 KB.
   - **C.** Storage queue messages are durable.
   - **D.** The client application should save the message identifier returned after adding a message to a queue for later use.
2. Which of the following are valid options for processing queue messages? (Choose all that apply.)
   - **A.** A single compute instance can process only one message at a time.
   - **B.** A single compute instance can process up to 32 messages at a time.
   - **C.** A single compute instance can retrieve up to 32 messages at a time.
   - **D.** Messages can be read one at a time or in batches of up to 32 messages at a time.
   - **E.** Messages are deleted as soon as they are read.
3. Which of the following are valid options for scaling queues? (Choose all that apply.)
   - **A.** Distributing messages across multiple queues
   - **B.** Automatically scaling websites based on queue metrics
   - **C.** Automatically scaling VMs based on queue metrics
   - **D.** Automatically scaling cloud services based on queue metrics

Objective 4.4: Manage access

All storage accounts can be protected by a secure HTTPS connection and by using storage account keys to access all resources. In this section, you'll learn how to manage storage account keys, how to generate shared access keys with more granular control over which resources are accessible and for how long, how to manage policies for issued keys, and how to allow browser access to storage resources.

MORE INFO **MANAGING ACCESS TO STORAGE SERVICES**

For an overview of some of the topics discussed in this section, see *http://msdn.microsoft.com/en-us/library/azure/ee393343.aspx*.

This objective covers how to:

- Generate shared access signatures
- Create stored access policies
- Regenerate storage account keys
- Configure and use Cross-Origin Resource Sharing (CORS)

Generating shared access signatures

By default, storage resources are protected at the service level. Only authenticated callers can access tables and queues. Blob containers and blobs can optionally be exposed for anonymous access, but you would typically allow anonymous access only to individual blobs. To authenticate to any storage service, a primary or secondary key is used, but this grants the caller access to all actions on the storage account.

An SAS is used to delegate access to specific storage account resources without enabling access to the entire account. An SAS token lets you control the lifetime by setting the start and expiration time of the signature, the resources you are granting access to, and the permissions being granted.

The following is a list of operations supported by SAS:

- Reading or writing blobs, blob properties, and blob metadata
- Leasing or creating a snapshot of a blob
- Listing blobs in a container
- Deleting a blob
- Adding, updating, or deleting table entities
- Querying tables
- Processing queue messages (read and delete)
- Adding and updating queue messages
- Retrieving queue metadata

This section covers creating an SAS token to access storage services using the Storage Client Library.

MORE INFO **CONTROLLING ANONYMOUS ACCESS**

To control anonymous access to containers and blobs, follow the instructions provided at *http://msdn.microsoft.com/en-us/library/azure/dd179354.aspx*.

MORE INFO **CONSTRUCTING AN SAS URI**

SAS tokens are typically used to authorize access to the Storage Client Library when interacting with storage resources, but you can also use it directly with the storage resource URI and use HTTP requests directly. For details regarding the format of an SAS URI, see *http://msdn.microsoft.com/en-us/library/azure/dn140255.aspx*.

Creating an SAS token (Blobs)

The following code shows how to create an SAS token for a blob container:

```
string connection = "DefaultEndpointsProtocol=https;AccountName=<ACCOUNTNAME>;AccountKey
=<ACCOUNTKEY>";
CloudStorageAccount account;
if (!CloudStorageAccount.TryParse(connection, out account))
{
 throw new Exception("Unable to parse storage account connection string.");
}
CloudBlobClient blobClient = account.CreateCloudBlobClient();
SharedAccessBlobPolicy sasPolicy = new SharedAccessBlobPolicy();
sasPolicy.SharedAccessExpiryTime = DateTime.UtcNow.AddHours(1);
sasPolicy.SharedAccessStartTime = DateTime.UtcNow.Subtract(new TimeSpan(0, 5, 0));
sasPolicy.Permissions = SharedAccessBlobPermissions.Read | SharedAccessBlobPermissions.
Write | SharedAccessBlobPermissions.Delete | SharedAccessBlobPermissions.List;
CloudBlobContainer files = blobClient.GetContainerReference("files");
string sasContainerToken = files.GetSharedAccessSignature(sasPolicy);
```

The SAS token grants read, write, delete, and list permissions to the container (rwdl). It looks like this:

```
?sv=2014-02-14&sr=c&sig=B6bi4xKkdgOXhWg3RWIDO5peekq%2FRjvnuo5o41hj1pA%3D&st=2014
-12-24T14%3A16%3A07Z&se=2014-12-24T15%3A21%3A07Z&sp=rwdl
```

You can use this token as follows to gain access to the blob container without a storage account key:

```
StorageCredentials creds = new StorageCredentials(sasContainerToken);
CloudBlobClient sasClient = new CloudBlobClient("https://<ACCOUNTNAME>.blob.core.
windows.net/", creds);
CloudBlobContainer sasFiles = sasClient.GetContainerReference("files");
```

With this container reference, if you have write permissions, you can create a blob, for example as follows:

```
ICloudBlob blob = sasFiles.GetBlockBlobReference("note.txt");
blob.Properties.ContentType = "text/plain";
string fileContents = "my text blob contents";
byte[] bytes = new byte[fileContents.Length * sizeof(char)];
System.Buffer.BlockCopy(fileContents.ToCharArray(), 0, bytes, 0, bytes.Length);
blob.UploadFromByteArray(bytes,0, bytes.Length);
```

Creating an SAS token (Queues)

Assuming the same account reference as created in the previous section, the following code shows how to create an SAS token for a queue:

```
CloudQueueClient queueClient = account.CreateCloudQueueClient();
CloudQueue queue = queueClient.GetQueueReference("workerqueue");
SharedAccessQueuePolicy sasPolicy = new SharedAccessQueuePolicy();
sasPolicy.SharedAccessExpiryTime = DateTime.UtcNow.AddHours(1);
sasPolicy.Permissions = SharedAccessQueuePermissions.Read |
SharedAccessQueuePermissions.Add | SharedAccessQueuePermissions.Update |
SharedAccessQueuePermissions.ProcessMessages;
sasPolicy.SharedAccessStartTime = DateTime.UtcNow.Subtract(new TimeSpan(0, 5, 0));
string sasToken = queue.GetSharedAccessSignature(sasPolicy);
```

The SAS token grants read, add, update, and process messages permissions to the container (raup). It looks like this:

```
?sv=2014-02-14&sig=wE5oAUYHcGJ8chwyZZd3Byp5jK1Po8uKu2t%2FYzQsIhY%3D&st=2014-12-2
4T14%3A23%3A22Z&se=2014-12-24T15%3A28%3A22Z&sp=raup
```

You can use this token as follows to gain access to the queue and add messages:

```
StorageCredentials creds = new StorageCredentials(sasToken);
CloudQueueClient sasClient = new CloudQueueClient("https://<ACCOUNTNAME>.queue.core.
windows.net/", creds);
CloudQueue sasQueue = sasClient.GetQueueReference("workerqueue");
sasQueue.AddMessage(new CloudQueueMessage("new message"));
```

IMPORTANT **SECURE USE OF SAS**

Always use a secure HTTPS connection to generate an SAS token to protect the exchange of the URI, which grants access to protected storage resources.

Creating an SAS token (Tables)

The following code shows how to create an SAS token for a table:

```
CloudTableClient tableClient = account.CreateCloudTableClient();
CloudTable table = tableClient.GetTableReference("$logs");
SharedAccessTablePolicy sasPolicy = new SharedAccessTablePolicy();
sasPolicy.SharedAccessExpiryTime = DateTime.UtcNow.AddHours(1);
sasPolicy.Permissions = SharedAccessTablePermissions.Query |
SharedAccessTablePermissions.Add | SharedAccessTablePermissions.Update |
SharedAccessTablePermissions.Delete;
sasPolicy.SharedAccessStartTime = DateTime.UtcNow.Subtract(new TimeSpan(0, 5, 0));
string sasToken = table.GetSharedAccessSignature(sasPolicy);
```

The SAS token grants query, add, update, and delete permissions to the container (raud). It looks like this:

```
?sv=2014-02-14&tn=%24logs&sig=dsnI7RBA1xYQVr%2FTlpDEZMO2H8YtSGwtyUUntVmxstA%3D&s
t=2014-12-24T14%3A48%3A09Z&se=2014-12-24T15%3A53%3A09Z&sp=raud
```

Renewing an SAS token

SAS tokens have a limited period of validity based on the start and expiration times requested. You should limit the duration of an SAS token to limit access to controlled periods of time. You can extend access to the same application or user by issuing new SAS tokens on request. This should be done with appropriate authentication and authorization in place.

Validating data

When you extend write access to storage resources with SAS, the contents of those resources can potentially be made corrupt or even be tampered with by a malicious party, particularly if the SAS was leaked. Be sure to validate system use of all resources exposed with SAS keys.

Creating stored access policies

Stored access policies provide greater control over how you grant access to storage resources using SAS tokens. With a stored access policy, you can do the following after releasing an SAS token for resource access:

- Change the start and end time for a signature's validity
- Control permissions for the signature
- Revoke access

The stored access policy can be used to control all issued SAS tokens that are based on the policy. For a step-by-step tutorial for creating and testing stored access policies for blobs, queues, and tables, see *http://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-shared-access-signature-part-2*.

IMPORTANT **RECOMMENDATION FOR SAS TOKENS**

Use stored access policies wherever possible, or limit the lifetime of SAS tokens to avoid malicious use.

MORE INFO **STORED ACCESS POLICY FORMAT**

For more information on the HTTP request format for creating stored access policies, see *http://msdn.microsoft.com/en-us/library/azure/ee393341.aspx*.

Regenerating storage account keys

When you create a storage account, two 512-bit storage access keys are generated for authentication to the storage account. This makes it possible to regenerate keys without impacting application access to storage.

The process for managing keys typically follows this pattern:

1. When you create your storage account, the primary and secondary keys are generated for you. You typically use the primary key when you first deploy applications that access the storage account.
2. When it is time to regenerate keys, you first switch all application configurations to use the secondary key.
3. Next, you regenerate the primary key, and switch all application configurations to use this primary key.
4. Next, you regenerate the secondary key.

Regenerating storage account keys (existing portal)

To regenerate storage account keys using the management portal, complete the following steps:

1. Navigate to the Dashboard tab for your storage account in the management portal accessed via *https://manage.windowsazure.com*.
2. Select Manage Access Keys from the bottom of the page.
3. Click the regenerate button for the primary access key or for the secondary access key, depending on which key you intend to regenerate, according to the workflow above.
4. Click the check mark on the confirmation dialog box to complete the regeneration task.

IMPORTANT **MANAGING KEY REGENERATION**

It is imperative that you have a sound key management strategy. In particular, you must be certain that all applications are using the primary key at a given point in time to facilitate the regeneration process.

Regenerating storage account keys (Preview portal)

To regenerate storage account keys using the Preview portal, complete the following steps:

1. Navigate to the management portal accessed via *https://portal.azure.com*.
2. Click Browse on the command bar.
3. Select Storage from the Filter By list.
4. Select your storage account from the list on the Storage blade.
5. Click the Keys box.

6. On the Manage Keys blade, click Regenerate Primary or Regenerate Secondary on the command bar, depending on which key you want to regenerate.
7. In the confirmation dialog box, click Yes to confirm the key regeneration.

Configuring and using Cross-Origin Resource Sharing

Cross-Origin Resource Sharing (CORS) enables web applications running in the browser to call web APIs that are hosted by a different domain. Azure Storage blobs, tables, and queues all support CORS to allow for access to the Storage API from the browser. By default, CORS is disabled, but you can explicitly enable it for a specific storage service within your storage account.

MORE INFO **ENABLING CORS**

For additional information about enabling CORS for your storage accounts, see *http://msdn.microsoft.com/en-us/library/azure/dn535601.aspx*.

Thought experiment

Access control strategy

In this thought experiment, apply what you've learned about this objective. You can find answers to these questions in the "Answers" section at the end of this chapter.

Your web application generates large reports for your customers, and you are designing a strategy for granting access to those reports, which are stored in blobs. You want users to authenticate to download reports, but you want them to be able to share a link to the report with others in the company in a secure way that prevents unauthorized users from accessing content.

1. **How would you approach granting access to these reports within the web application and sharing that with authenticated users?**
2. **How would you ensure that if the report is shared with others via link, the reports are not available long term without authentication?**

Objective summary

- You can use SAS tokens to delegate access to storage account resources without sharing the account key.
- With SAS tokens, you can generate a link to a container, blob, table, table entity, or queue. You can control the permissions granted to the resource.
- Using Shared Access Policies, you can remotely control the lifetime of a SAS token grant to one or more resources. You can extend the lifetime of the policy or cause it to expire.

Objective review

Answer the following questions to test your knowledge of the information in this objective. You can find the answers to these questions and explanations of why each answer choice is correct or incorrect in the "Answers" section at the end of this chapter.

1. Which of the following are true regarding supported operations granted with an SAS token? (Choose all that apply.)
   - **A.** You can grant read access to existing blobs.
   - **B.** You can create new blob containers.
   - **C.** You can add, update, and delete queue messages.
   - **D.** You can add, update, and delete table entities.
   - **E.** You can query table entities.
2. Which of the following statements are true of stored access policies? (Choose all that apply.)
   - **A.** You can modify the start or expiration date for access.
   - **B.** You can revoke access at any point in time.
   - **C.** You can modify permissions to remove or add supported operations.
   - **D.** You can add to the list of resources accessible by an SAS token.
3. Which of the following statements are true of CORS support for storage? (Choose all that apply.)
   - **A.** It is recommended you enable CORS so that browsers can access blobs.
   - **B.** To protect CORS access to blobs from the browser, you should generate SAS tokens to secure blob requests.
   - **C.** CORS is supported only for Blob storage.
   - **D.** CORS is disabled by default.

Objective 4.5: Monitor storage

Azure Storage has a built-in analytics feature called Azure Storage Analytics used for collecting metrics and logging storage request activity. You enable Storage Analytics Metrics to collect aggregate transaction and capacity data, and you enable Storage Analytics Logging to capture successful and failed request attempts to your storage account. This section covers how to enable monitoring and logging, control logging levels, set retention policies, and analyze the logs.

NOTE **STORAGE ANALYTICS AVAILABILITY**

At the time of this writing, Storage Analytics is not available for Azure Files.

This objective covers how to:

- Enable monitoring and logging
- Set retention policies and logging levels
- Analyze logs

Configuring storage metrics

Storage Analytics metrics provide insight into transactions and capacity for your storage accounts. You can think of them as the equivalent of Windows Performance Monitor counters. By default, storage metrics are not enabled, but you can enable them through the management portal, using Windows PowerShell, or by calling the management API directly.

When you configure storage metrics for a storage account, tables are generated to store the output of metrics collection. You determine the level of metrics collection for transactions and the retention level for each service—Blob, Table, and Queue.

Transaction metrics record request access to each service for the storage account. You specify the interval for metric collection (hourly or by minute). In addition, there are two levels of metrics collection:

- **Service level** These metrics include aggregate statistics for all requests, aggregated at the specified interval. Even if no requests are made to the service, an aggregate entry is created for the interval, indicating no requests for that period.
- **API level** These metrics record every request to each service only if a request is made within the hour interval.

NOTE **METRICS COLLECTED**

All requests are included in the metrics collected, including any requests made by Storage Analytics.

Capacity metrics are only recorded for the Blob service for the account. Metrics include total storage in bytes, the container count, and the object count (committed and uncommitted).

Table 4-1 summarizes the tables automatically created for the storage account when Storage Analytics metrics are enabled.

TABLE 4-1 Storage metrics tables

METRICS	TABLE NAMES
Hourly metrics	$MetricsHourPrimaryTransactionsBlob $MetricsHourPrimaryTransactionsTable $MetricsHourPrimaryTransactionsQueue
Minute metrics (cannot set through the management portal)	$MetricsMinutePrimaryTransactionsBlob $MetricsMinutePrimaryTransactionsTable $MetricsMinutePrimaryTransactionsQueue
Capacity (only for the Blob ser-vice)	$MetricsCapacityBlob

MORE INFO **STORAGE ANALYTICS METRICS TABLE SCHEMA**

For additional details on the transaction and capacity metrics collected, see *http://msdn.microsoft.com/en-us/library/azure/hh343264.aspx*.

Retention can be configured for each service in the storage account. By default, Storage Analytics will not delete any metrics data. When the shared 20-terabyte limit is reached, new data cannot be written until space is freed. This limit is independent of the storage limit of the account. You can specify a retention period from 0 to 365 days. Metrics data is automatically deleted when the retention period is reached for the entry.

When metrics are disabled, existing metrics that have been collected are persisted up to their retention policy.

MORE INFO **STORAGE METRICS**

For more information about enabling and working with storage metrics, see *http://msdn.microsoft.com/en-us/library/azure/dn782843.aspx*.

Configuring storage metrics and retention (existing portal)

To enable storage metrics and associated retention levels for Blob, Table, and Queue services in the existing management portal, follow these steps:

1. Navigate to the Configure tab for your storage account in the management portal accessed via *https://manage.windowsazure.com*.
2. If this storage account uses blobs, set the metrics level for blobs to Minimal. Set retention according to your retention policy.

3. If this storage account uses tables, set the metrics level for tables to Minimal. Set retention according to your retention policy.
4. If this storage account uses queues, set the metrics level for queues to Minimal. Set retention according to your retention policy.
5. Click Save to commit the settings.

NOTE **CHOOSING A METRICS LEVEL**

Minimal metrics yield enough information to provide a picture of the overall usage and health of the storage account services. Verbose metrics provide more insight at the API level, allowing for deeper analysis of activities and issues, which is helpful for troubleshooting.

Configuring storage metrics and retention (Preview portal)

To enable storage metrics and associated retention levels for Blob, Table, and Queue services in the Preview portal, follow these steps:

1. Navigate to the management portal accessed via *https://portal.azure.com*.
2. Click Browse on the command bar.
3. Select Storage from the Filter By drop-down list.
4. Select your storage account from the list on the Storage blade.
5. Scroll down to the Usage section, and click the Capacity Past Week check box.
6. On the Metric blade, click Diagnostics on the command bar.
7. Click the On button under Status. This shows the options for metrics and logging.
8. If this storage account uses blobs, select Blob Aggregate Metrics to enable service level metrics. Select Blob Per API Metrics for API level metrics.
9. If this storage account uses tables, select Table Aggregate Metrics to enable service level metrics. Select Table Per API Metrics for API level metrics.
10. If this storage account uses queues, select Queue Aggregate Metrics to enable service level metrics. Select Queue Per API Metrics for API level metrics.
11. Provide a value for retention according to your retention policy. Through the Preview portal, this will apply to all services. It will also apply to Storage Analytics Logging if that is enabled. Select one of the available retention settings from the drop-down list, or enter a number from 0 to 365.

Configuring storage metrics and retention using Windows PowerShell

To enable storage metrics and associated retention levels using Windows PowerShell, use the Set-AzureStorageMetricsProperty cmdlet.

To enable service level metrics collected by minute for blobs, tables, and queues with unlimited retention, run the following:

```
Set-AzureStorageServiceMetricsProperty –MetricsType Minute –ServiceType Blob –
MetricsLevel Service –RetentionDays 0
Set-AzureStorageServiceMetricsProperty –MetricsType Minute –ServiceType Table –
MetricsLevel Service –RetentionDays 0
Set-AzureStorageServiceMetricsProperty –MetricsType Minute –ServiceType Queue –
MetricsLevel Service –RetentionDays 0
```

To enable service and API level metrics collected hourly for blobs, tables, and queues with 90 days of retention, run the following:

```
Set-AzureStorageServiceMetricsProperty –MetricsType Hour –ServiceType Blob –MetricsLevel
ServiceAndApi –RetentionDays 90
Set-AzureStorageServiceMetricsProperty –MetricsType Hour –ServiceType Table –
MetricsLevel ServiceAndApi –RetentionDays 90
Set-AzureStorageServiceMetricsProperty –MetricsType Hour –ServiceType Queue –
MetricsLevel ServiceAndApi –RetentionDays 90
```

To disable the collection of metrics, run the following:

```
Set-AzureStorageServiceMetricsProperty–ServiceType Blob –MetricsLevel None
Set-AzureStorageServiceMetricsProperty–ServiceType Table –MetricsLevel None
Set-AzureStorageServiceMetricsProperty–ServiceType Queue –MetricsLevel None
```

Analyzing storage metrics

Storage Analytics metrics are collected in tables as discussed in the previous section. You can access the tables directly to analyze metrics, but you can also review metrics in both Azure management portals. This section discusses various ways to access metrics and review or analyze them.

MORE INFO **STORAGE MONITORING, DIAGNOSING, AND TROUBLESHOOTING**

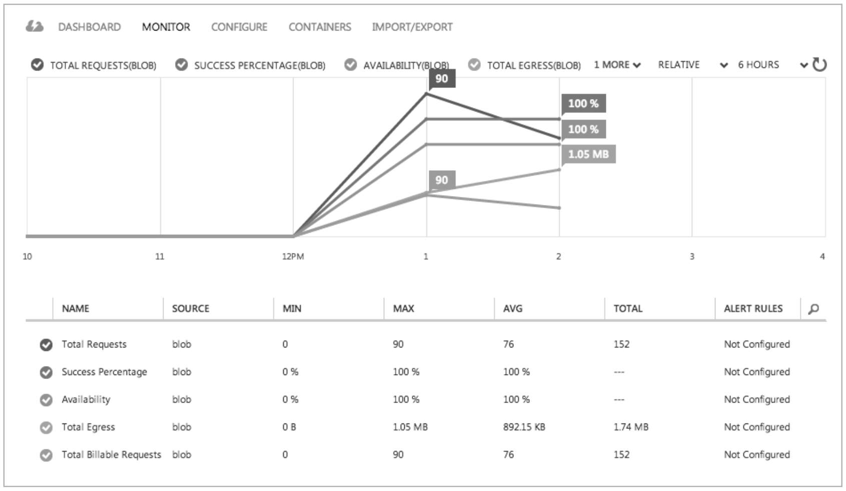
For more details on how to work with storage metrics and logs, see *http://azure.microsoft.com/en-us/documentation/articles/storage-monitoring-diagnosing-troubleshooting*.

Monitoring metrics (existing portal)

To monitor metrics in the existing portal, complete the following steps:

1. Navigate to the Monitor tab for your storage account in the management portal accessed via *https://manage.windowsazure.com*.
2. Click Add Metric to choose metrics to monitor in the management portal.
3. In the Choose Metrics dialog box, select from the list of metrics for blobs, tables, or queues.
4. Click the check mark to commit the settings.

5. You may choose up to six metrics to show in the monitoring graph, as shown in Figure 4-8.



NAME	SOURCE	MIN	MAX	AVG	TOTAL	ALERT RULES
✓ Total Requests	blob	0	90	76	152	Not Configured
✓ Success Percentage	blob	0 %	100 %	100 %	---	Not Configured
✓ Availability	blob	0 %	100 %	100 %	---	Not Configured
✓ Total Egress	blob	0 B	1.05 MB	892.15 KB	1.74 MB	Not Configured
✓ Total Billable Requests	blob	0	90	76	152	Not Configured

FIGURE 4-8 Monitoring metrics from the existing management portal

6. Select absolute or relative display of metrics from the appropriate drop-down list (RELATIVE is indicated in Figure 4-8).

7. Select the time range to display from the appropriate drop-down list (6 HOURS is indicated in Figure 4-8). Select 6 hours, 24 hours, or 7 days.

Monitoring metrics (Preview portal)

At the time of this writing, the Preview portal features for monitoring metrics is limited to some predefined metrics, including total requests, total egress, average latency, and availability (see Figure 4-9). Click each box to see a Metric blade that provides additional detail.

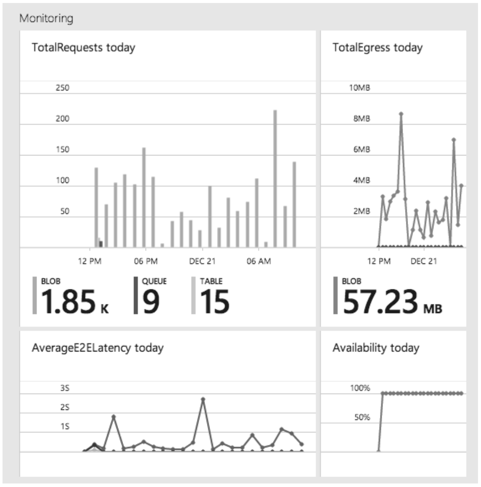


FIGURE 4-9 Monitoring overview from the Preview portal

To monitor the metrics available in the Preview portal, complete the following steps:

1. Navigate to the management portal accessed via *https://portal.azure.com*.
2. Click Browse on the command bar.
3. Select Storage from the Filter By drop-down list.
4. Select your storage account from the list on the Storage blade.
5. Scroll down to the Monitor section, and view the monitoring boxes summarizing statistics. You'll see TotalRequests, TotalEgress, AverageE2ELatency, and AvailabilityToday by default.
6. Click each metric box to view additional details for each metric. You'll see metrics for blobs, tables, and queues if all three metrics are being collected.

NOTE **CUSTOMIZING THE MONITORING BLADE**

You can customize which boxes appear in the Monitoring area of the Preview portal, and you can adjust the size of each box to control how much detail is shown at a glance without drilling into the metrics blade.

Configuring Storage Analytics Logging

Storage Analytics Logging provides details about successful and failed requests to each storage service that has activity across the account's blobs, tables, and queues. By default, storage logging is not enabled, but you can enable it through the management portal, by using Windows PowerShell, or by calling the management API directly.

When you configure Storage Analytics Logging for a storage account, a blob container named $logs is automatically created to store the output of the logs. You choose which services you want to log for the storage account. You can log any or all of the Blob, Table, or Queue services. You can also choose which type of requests to log: read, write, or delete. Logs are created only for those services that have activity, so you will not be charged if you enable logging for a service that has no requests. The logs are stored as block blobs as requests are logged and are periodically committed so that they are available as blobs.

NOTE **DELETING THE LOG CONTAINER**

After Storage Analytics has been enabled, the log container cannot be deleted; however, the contents of the log container can be deleted.

Retention can be configured for each service in the storage account. By default, Storage Analytics will not delete any logging data. When the shared 20-terabyte limit is reached, new data cannot be written until space is freed. This limit is independent of the storage limit of the account. You can specify a retention period from 0 to 365 days. Logging data is automatically deleted when the retention period is reached for the entry.

NOTE **DUPLICATE LOGS**

Duplicate log entries may be present within the same hour. You can use the RequestId and operation number to uniquely identify an entry to filter duplicates.

MORE INFO **STORAGE LOGGING**

For more information about enabling and working with Azure storage logging, see *http://msdn.microsoft.com/en-us/library/azure/dn782843.aspx* and *http://msdn.microsoft.com/en-us/library/azure/hh343262.aspx*.

Configuring storage logging and retention (existing portal)

To enable storage logging and associated retention levels for Blob, Table, and Queue services in the existing portal, follow these steps:

1. Navigate to the Configure tab for your storage account in the management portal accessed via *https://manage.windowsazure.com*. Scroll to the logging section.
2. If this storage account uses blobs, select Read, Write, and Delete requests to log all activity. Set retention according to your retention policy.
3. If this storage account uses tables, select Read, Write, and Delete requests to log all activity. Set retention according to your retention policy.
4. If this storage account uses queues, check Read, Write, and Delete requests to log all activity. Set retention according to your retention policy.
5. Click Save to commit the settings.

Configuring storage logging and retention (Preview portal)

To enable storage logging and associated retention levels for Blob, Table, and Queue services in the Preview portal, follow these steps:

1. Navigate to the management portal accessed via *https://portal.azure.com*.
2. Click Browse on the command bar.
3. Select Storage from the Filter By drop-down list.
4. Select your storage account from the list on the Storage blade.
5. Scroll down to the Usage section, and select the Capacity Past Week check box.
6. On the Metric blade, click Diagnostics on the command bar.
7. Click the On button under Status. This shows the options for enabling monitoring features.

8. If this storage account uses blobs, select Blob Logs to log all activity.
9. If this storage account uses tables, select Table Logs to log all activity.
10. If this storage account uses queues, select Queue Logs to log all activity.
11. Provide a value for retention according to your retention policy. Through the Preview portal, this will apply to all services. It will also apply to Storage Analytics Metrics if that is enabled. Select one of the available retention settings from the drop-down list, or enter a number from 0 to 365.

NOTE **CONTROLLING LOGGED ACTIVITIES**

From the Preview portal, when you enable or disable logging for each service, you enable read, write, and delete logging. To log only specific activities, use Windows PowerShell cmdlets.

Configuring storage logging and retention using Windows PowerShell

To enable storage metrics and associated retention levels using Windows PowerShell, use the Set-AzureStorageServiceLoggingProperty cmdlet.

To enable logging for read, write, and delete actions with retention of 30 days, run the following:

```
Set-AzureStorageServiceLoggingProperty –ServiceType Blob –LoggingOperations
read,write,delete –RetentionDays 30
Set-AzureStorageServiceLoggingProperty –ServiceType Table –LoggingOperations
read,write,delete –RetentionDays 30
Set-AzureStorageServiceLoggingProperty –ServiceType Queue –LoggingOperations
read,write,delete –RetentionDays 30
```

To disable the collection of metrics, run the following:

```
Set-AzureStorageServiceLoggingProperty –ServiceType Blob –LoggingOperations none
Set-AzureStorageServiceLoggingProperty –ServiceType Table –LoggingOperations none
Set-AzureStorageServiceLoggingProperty –ServiceType Queue –LoggingOperations none
```

Enabling client-side logging

You can enable client-side logging using Microsoft Azure storage libraries to log activity from client applications to your storage accounts. For information on the .NET Storage Client Library, see *http://msdn.microsoft.com/en-us/library/azure/dn782839.aspx*. For information on the Storage SDK for Java, see *http://msdn.microsoft.com/en-us/library/azure/dn782844.asp*x.

Analyzing storage logs

Logs are stored as block blobs in delimited text format. When you access the container, you can download logs for review and analysis using any tool compatible with that format. Within the logs, you'll find entries for authenticated and anonymous requests, as listed in Table 4-3.

TABLE 4-3 Authenticated and anonymous logs

Request Type	Logged Requests
Authenticated requests	■ Successful requests ■ Failed requests such as timeouts, authorization, throttling issues, and other errors ■ Requests that use an SAS ■ Requests for analytics data
Anonymous requests	■ Successful requests ■ Server errors ■ Timeouts for client or server ■ Failed GET requests with error code 304 (Not Modified)

Logs include status messages and operation logs. Status message columns include those shown in Table 4-4. Some status messages are also reported with storage metrics data. There are many operation logs for the Blob, Table, and Queue services.

MORE INFO **STATUS MESSAGES AND OPERATION LOGS**

For a detailed list of specific logs and log format specifics, see *http://msdn.microsoft.com/en-us/library/azure/hh343260.aspx* and *http://msdn.microsoft.com/en-us/library/hh343259.aspx*.

TABLE 4-4 Information included in logged status messages

Column	Description
Status Message	Indicates a value for the type of status message, indicating type of success or failure
Description	Describes the status, including any HTTP verbs or status codes
Billable	Indicates whether the request was billable
Availability	Indicates whether the request is included in the availability calculation for storage metrics

Finding your logs

When storage logging is configured, log data is saved to blobs in the $logs container created for your storage account. You can't see this container by listing containers, but you can navigate directly to the container to access, view, or download the logs.

To view analytics logs produced for a storage account, do the following:

1. Using a storage browsing tool, navigate to the $logs container within the storage account you have enabled Storage Analytics Logging for using this convention: *https://<accountname>.blob.core.windows.net/$logs*.
2. View the list of log files with the convention *<servicetype>/YYYY/MM/DD/HHMM/<counter>.log*.
3. Select the log file you want to review, and download it using the storage browsing tool.

MORE INFO **LOG METADATA**

The blob name for each log file does not provide an indication of the time range for the logs. You can search this information in the blob metadata using storage browsing tools or Windows PowerShell.

Accessing logs with Windows PowerShell

Using Windows PowerShell, you can access logs with the Get-AzureStorageBlob cmdlet and then filter logs by filename and metadata. The following example illustrates how to filter a list of write logs for Blob storage entries on a particular date during the ninth hour:

```
Get-AzureStorageBlob -Container '$logs' |
where {
  $_.Name -match 'blob/2014/12/01/09' -and
  $_.ICloudBlob.Metadata.LogType -match 'write'
} |
foreach {
  "{0} {1} {2} {3}" –f $_.Name,
  $_.ICloudBlob.Metadata.StartTime,
  $_.ICloudBlob.Metadata.EndTime,
  $_.ICloudBlob.Metadata.LogType
}
```

Downloading logs

To review and analyze your logs, first download them to your local machine. You can do this with storage browsing tools, programmatically, or with AzCopy.

MORE INFO **AZCOPY**

AzCopy is part of the Azure SDK. You can download the latest version directly from *http://aka.ms/AzCopy*.

Viewing logs with Microsoft Excel

Storage logs are recorded in a delimited format so that you can use any compatible tool to view logs. To view logs data in Excel, follow these steps:

1. Open Excel, and on the Data menu, click From Text.
2. Find the log file and click Import.
3. During import, select Delimited format. Select Semicolon as the only delimiter, and Double-Quote (") as the text qualifier.

Analyzing logs

After you load your logs into a viewer like Excel, you can analyze and gather information such as the following:

- Number of requests from a specific IP range
- Which tables or containers are being accessed and the frequency of those requests
- Which user issued a request, in particular, any requests of concern
- Slow requests
- How many times a particular blob is being accessed with an SAS URL
- Details to assist in investigating network errors

MORE INFO **LOG ANALYSIS**

You can run the Azure HDInsight Log Analysis Toolkit (LAT) for a deeper analysis of your storage logs. For more information, see *https://hadoopsdk.codeplex.com/releases/view/117906*.

Thought experiment

Proactive and reactive diagnostics

In this thought experiment, apply what you've learned about this objective. You can find answers to these questions in the "Answers" section at the end of this chapter.

Your application is now live, and you are planning how you will monitor the overall health of your storage account resources.

1. **What features would you use to give you a proactive, early warning of problems with storage services?**
2. **What features could you use to diagnose an issue retroactively so that you can perform root cause analysis and fix the problem?**

Objective summary

- Storage Analytics metrics provide the equivalent of Windows Performance Monitor counters for storage services.
- You can determine which services to collect metrics for (Blob, Table, or Queue), whether to collect metrics for the service or API level, and whether to collect metrics by the minute or hour.
- Capacity metrics are only applicable to the Blob service.
- Storage Analytics Logging provides details about the success or failure of requests to storage services.
- Storage logs are stored in blob services for the account, in the $logs container for the service.
- You can specify up to 365 days for retention of storage metrics or logs, or you can set retention to 0 to retain metrics indefinitely. Metrics and logs are removed automatically from storage when the retention period expires.
- Storage metrics can be viewed in the management portal. Storage logs can be downloaded and viewed in a reporting tool such as Excel.

Objective review

Answer the following questions to test your knowledge of the information in this objective. You can find the answers to these questions and explanations of why each answer choice is correct or incorrect in the "Answers" section at the end of this chapter.

1. Which statements are true of Storage Analytics Metrics? (Choose all that apply.)
   - **A.** Capacity metrics are recorded only for blobs.
   - **B.** You can set hourly or by minute metrics through the management portal.
   - **C.** By default, metrics are retained for one year.
   - **D.** If you disable metrics, existing metrics are deleted from storage.
2. Which statements are true of Storage Analytics Logging? (Choose all that apply.)
   - **A.** Logs are stored in the same storage account where they are enabled and are measured as part of your storage quota.
   - **B.** Logs can have duplicate entries.
   - **C.** Logs cannot be deleted.
   - **D.** You can log all read, write, and delete requests to blobs, queues, and tables in a storage account.

3. Which of the following are captured by Storage Analytics Logging? (Choose all that apply.)
   - **A.** Successful requests for authenticated calls only
   - **B.** Failed requests for authenticated calls only
   - **C.** Server errors
   - **D.** Requests using SAS URIs

Objective 4.6: Implement SQL databases

In this section, you learn about Microsoft Azure SQL Database, a PaaS offering for relational data.

This objective covers how to:

- Choose the appropriate database tier and performance level
- Configure and perform point in time recovery
- Enable geo-replication
- Import and export data and schema
- Scale SQL databases

Choosing the appropriate database tier and performance level

Choosing a SQL Database tier used to be simply a matter of storage space. Recently, Microsoft added new tiers that also affect the performance of SQL Database. This tiered pricing is called Service Tiers. There are three service tiers to choose from, and while they still each have restrictions on storage space, they also have some differences that might affect your choice. The major difference is in a measurement called database throughput units (DTUs). A DTU is a blended measure of CPU, memory, disk reads, and disk writes. Because SQL Database is a shared resource with other Azure customers, sometimes performance is not stable or predictable. As you go up in performance tiers, you also get better predictability in performance.

- **Basic** Basic tier is meant for light workloads. There is only one performance level of the basic service tier. This level is good for small use, new projects, testing, development, or learning.
- **Standard** Standard tier is used for most production online transaction processing (OLTP) databases. The performance is more predictable than the basic tier. In addition, there are four performance levels under this tier, levels S0 to S3.
- **Premium** Premium tier continues to scale at the same level as the standard tier. In addition, performance is typically measured in seconds. For instance, the basic tier can handle 16,600 transactions per hour. The standard/S2 level can handle 2,570

transactions per minute. The top tier of premium can handle 735 transactions per second. That translates to 2,645,000 per hour in basic tier terminology.

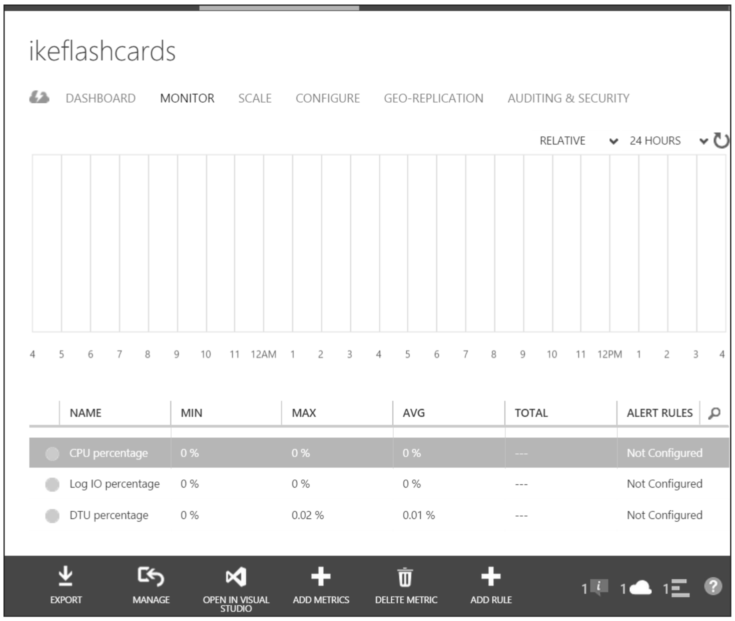
MORE INFO **SQL DATABASE TIERS AND THROUGHPUT**

For more information on SQL Database tiers, see *http://msdn.microsoft.com/en-us/library/azure/dn741336.aspx*.

There are many similarities between the various tiers. Each tier has a 99.9 percent uptime SLA, backup and restore capabilities, access to the same tooling, and the same database engine features. Fortunately, the levels are adjustable, and you can change your tier as your scaling requirements change.

The management portal can help you select the appropriate level. You can review the metrics on the Monitor tab to see the current load of your database and decide whether to scale up or down.

1. Click the SQL database you want to monitor.
2. Click the Monitor tab, as shown in Figure 4-10.
3. Add the following metrics:
   - CPU Percentage
   - Physical Data Reads Percentage
   - Log Writes Percentage



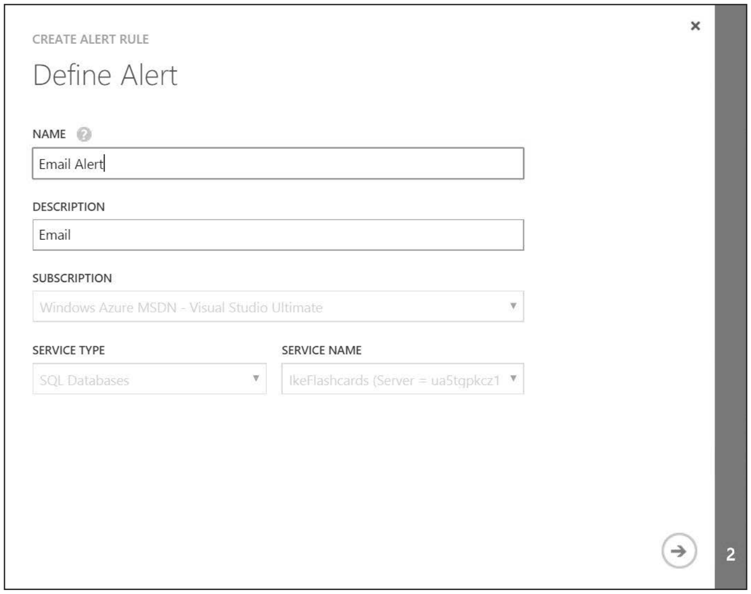
NAME	MIN	MAX	AVG	TOTAL	ALERT RULES
CPU percentage	0 %	0 %	0 %	---	Not Configured
Log IO percentage	0 %	0 %	0 %	---	Not Configured
DTU percentage	0 %	0.02 %	0.01 %	---	Not Configured

FIGURE 4-10 The Monitor tab

All three of these metrics are shown relative to the DTU of your database. If you reach 80 percent of your performance metrics, it's time to consider increasing your service tier or performance level. If you're consistently below 10 percent of the DTU, you might consider decreasing your service tier or performance level. Be aware of momentary spikes in usage when making your choice.

In addition, you can configure an email alert for when your metrics are 80 percent of your selected DTU by completing the following steps:

1. Click the metric.
2. Click Add Rule.
3. The first page of the Create Alert Rule dialog box is shown in Figure 4-11. Add a name and description, and then click the right arrow.



CREATE ALERT RULE

Define Alert

NAME: Email Alert

DESCRIPTION: Email

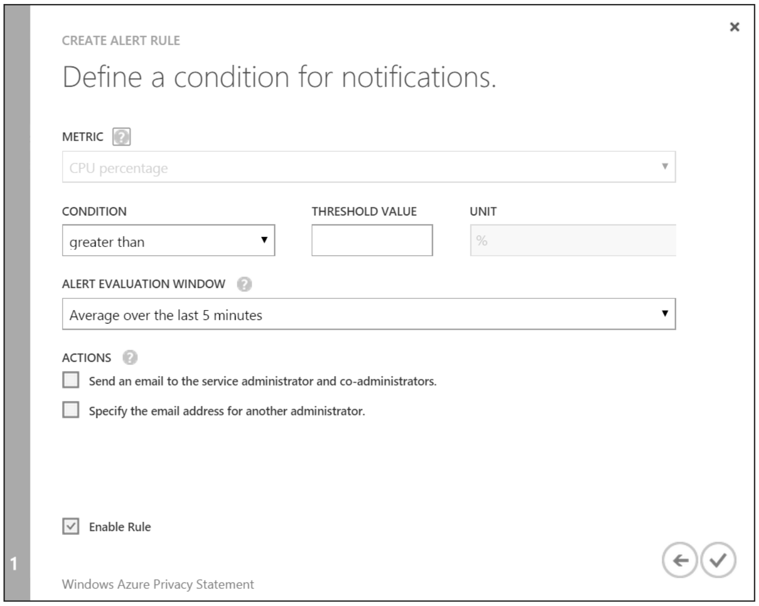
SUBSCRIPTION: Windows Azure MSDN - Visual Studio Ultimate

SERVICE TYPE: SQL Databases

SERVICE NAME: IkeFlashcards (Server = ua5tqpkcz1

FIGURE 4-11 The first page of the Create Alert Rule dialog box

4. On the next page of the Create Alert Rule dialog box, shown in Figure 4-12, select the condition and the threshold value.


CREATE ALERT RULE

Define a condition for notifications.

METRIC: CPU percentage

CONDITION: greater than

THRESHOLD VALUE:

UNIT: %

ALERT EVALUATION WINDOW: Average over the last 5 minutes

ACTIONS

- [ ] Send an email to the service administrator and co-administrators.
- [ ] Specify the email address for another administrator.

- [x] Enable Rule

FIGURE 4-12 The second page of the Create Alert Rule dialog box

5. Select your alert evaluation window. An email will be generated if the event happens over a specific duration. You should indicate at least 10 minutes.

6. Select the action. You can choose to send an email either to the service administrator(s) or to a specific email address.

Configuring and performing point in time recovery

Azure SQL Database does a full backup every week, a differential backup each day, and an incremental log backup every five minutes. The incremental log backup allows for a point in time restore, which means the database can be restored to any specific time of day. This means that if you accidentally delete a customer's table from your database, you will be able to recover it with minimal data loss if you know the timeframe to restore from that has the most recent copy.

The length of time it takes to do a restore varies. The further away you get from the last differential backup determines the longer the restore operation takes because there are more log backups to restore. When you restore a new database, the service tier stays the same, but the performance level changes to the minimum level of that tier.

Depending on your service tier, you will have different backup retention periods. Basic retains backups for 7 days, standard for 14 days, and premium for 35 days. In most cases, 14 days is enough time to determine that you have a problem and how to correct it.

You can restore a database that was deleted as long as you are within the retention period. Follow these steps to restore a database:

1. Select the database you want to restore, and then click Restore, as shown in Figure 4-13.

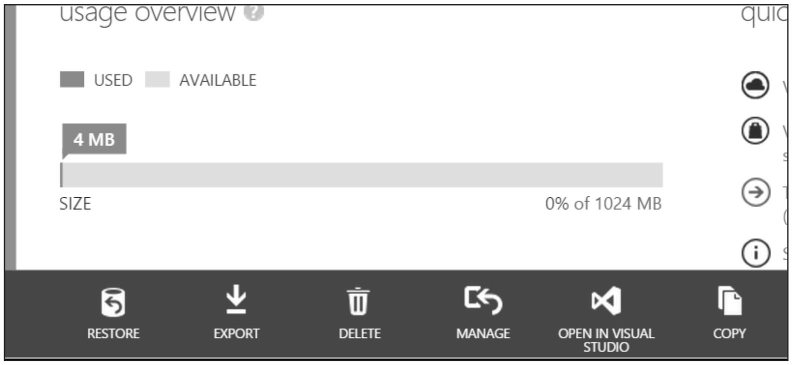
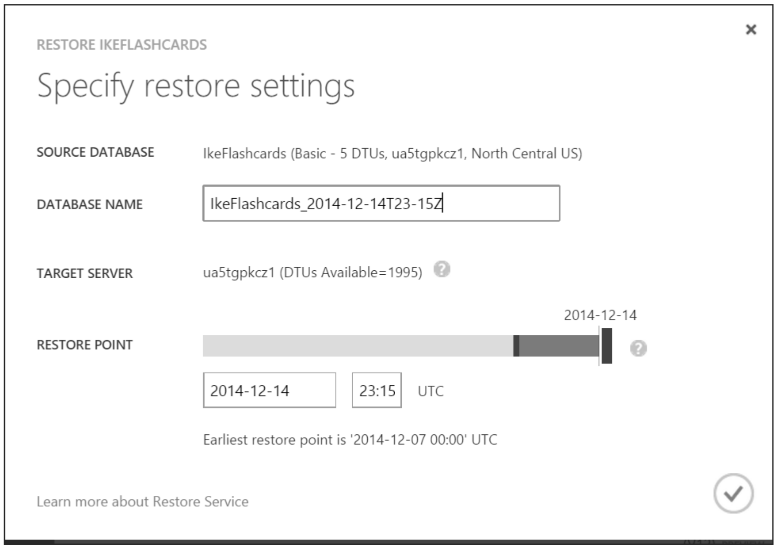


FIGURE 4-13 The Restore button

2. The Restore dialog box opens, as shown in Figure 4-14.


RESTORE IKEFLASHCARDS

Specify restore settings

SOURCE DATABASE IkeFlashcards (Basic - 5 DTUs, ua5tgpkcz1, North Central US)

DATABASE NAME IkeFlashcards_2014-12-14T23-15Z

TARGET SERVER ua5tgpkcz1 (DTUs Available=1995)

RESTORE POINT 2014-12-14

2014-12-14 23:15 UTC

Earliest restore point is '2014-12-07 00:00' UTC

Learn more about Restore Service

FIGURE 4-14 The Restore dialog box

3. Select a database name.
4. Select a restore point. You can use the slider bar or manually enter a date and time.
5. You can also restore a deleted database. Select the Deleted Databases tab, as shown in Figure 4-15.

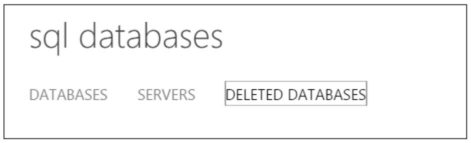


FIGURE 4-15 The Deleted Databases tab for SQL databases in the management portal

6. Select the database you want to restore.
7. Click Restore as you did in step 1.
8. Specify a database name for the new database.
9. Click Submit.

Enabling geo-replication

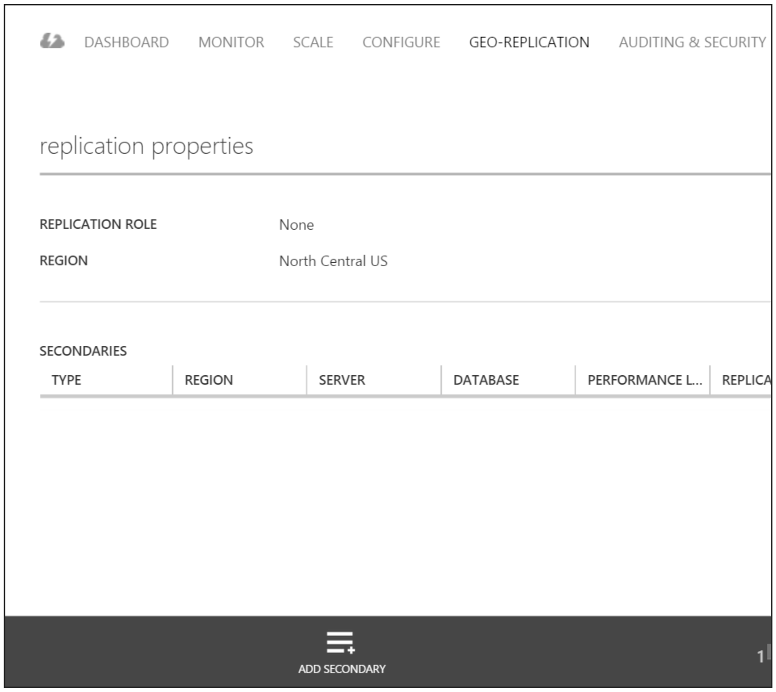
Every Azure SQL Database subscription has built-in redundancy. Three copies of your data are stored across fault domains in the datacenter to protect against server and hardware failure. This is built in to the subscription price and is not configurable. You can configure two more fault-tolerant options: standard geo-replication and active geo-replication.

Standard geo-replication allows the user to fail over the database to a different region when a database is not available. It is available on the standard and premium service tiers. The main difference between active and standard geo-replication is that standard geo-replication does not allow clients to connect to the secondary server. It is offline until it's needed to take over for the primary. The target region for the offline secondary server is pre-determined. For instance, if your primary server is in North Central US, then your secondary server will be in South Central US. The source and target servers must belong to the same subscription.

Creating an offline secondary database (existing portal)

Follow these steps to configure an offline secondary database:

1. Click the Geo-Replication tab for the database, as shown in Figure 4-16, and click Add Secondary.



replication properties

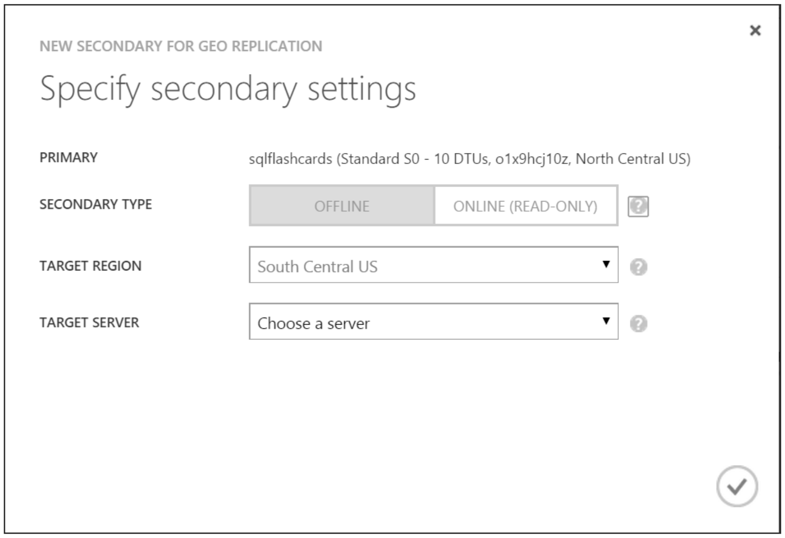
REPLICATION ROLE None
REGION North Central US

SECONDARIES

TYPE	REGION	SERVER	DATABASE	PERFORMANCE L...	REPLICA

FIGURE 4-16 Replication properties

2. On the Specify Secondary Settings page, shown in Figure 4-17, select a server from the server list or click New SQL Database Server, and then click the right arrow.

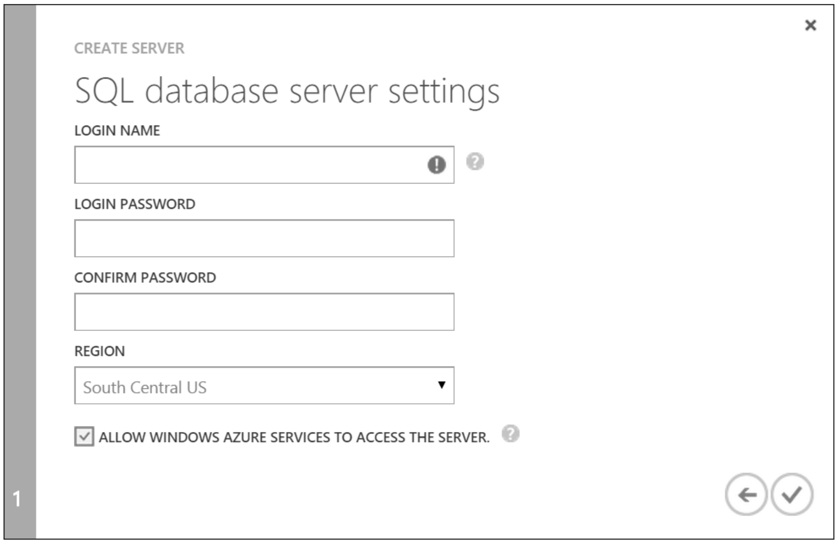

NEW SECONDARY FOR GEO REPLICATION

Specify secondary settings

PRIMARY	sqlflashcards (Standard S0 - 10 DTUs, o1x9hcj10z, North Central US)
SECONDARY TYPE	OFFLINE / ONLINE (READ-ONLY)
TARGET REGION	South Central US
TARGET SERVER	Choose a server

FIGURE 4-17 Creating a new secondary for geo replication

3. If you select a new server, the SQL Database Server Settings page opens (see Figure 4-18). Enter a login name and password, and select the Allow Windows Azure Services To Access The Server check box.


CREATE SERVER

SQL database server settings

LOGIN NAME

LOGIN PASSWORD

CONFIRM PASSWORD

REGION

South Central US

ALLOW WINDOWS AZURE SERVICES TO ACCESS THE SERVER.

FIGURE 4-18 The SQL Database Server Settings page

4. Monitor the Geo-Replication page for the progress of building the new secondary. You can watch the Replication Status of the database switch from Pending to Active.

   If there is a datacenter failure, the same page shows the Replication Status of your database as Unavailable. You will also see the Failover Enabled property set to true for the database and be able to initiate a failover by clicking Failover on the command bar.

OTE **USES FOR CREATING AN OFFLINE SECONDARY**

Another use for this feature has to do with the ability to terminate the continuous copy relationship between a primary and secondary database. You can terminate the relationship and then upgrade the primary database to a different schema to support a software upgrade. The secondary database gives you a rollback option.

Creating an offline secondary database (Preview portal)

To create an offline secondary database in the Preview portal, follow these steps:

1. Navigate to your SQL database in the management portal accessed via *https://portal.azure.com*.
2. Scroll to the Geo Replication section, and click the Configure Geo Replication box.
3. On the Geo Replication blade, select your target region.

4. On the Create Secondary blade, click Create.

Creating an online secondary database (existing portal)

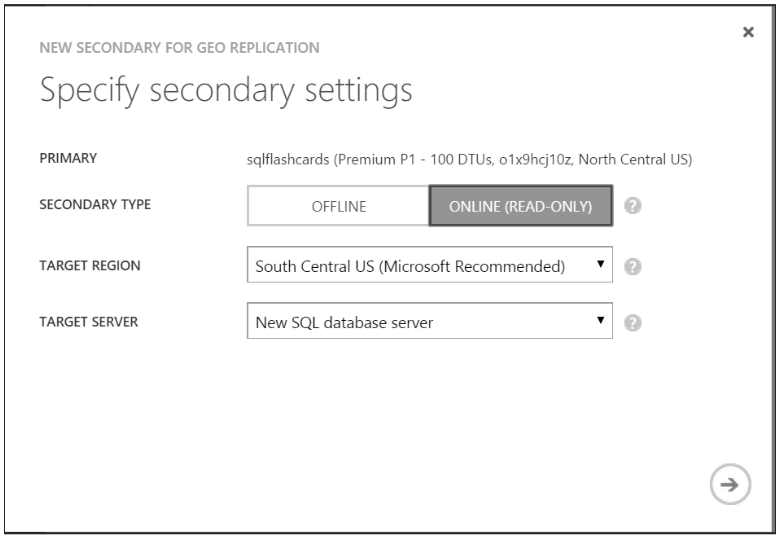
There are some differences between standard geo-replication and active geo-replication. Active geo-replication is different in these ways:

- You can have four secondary copies of your database.
- It is available only at the premium service tier.
- The online secondary will be consistent with the primary eventually.
- Of the four secondary copies of the database, four can be active, or three can be active and one can be an offline secondary.
- The online secondary server is readable. This allows you to put reports or ETL processes on the secondary, freeing up the locking overhead on the primary. Since the secondary copies are located in different regions, you can put readable databases close to remote users.

Before you create an online secondary, the following requirements must be met:

- The secondary database must have the same name as the primary.
- They must be on separate servers.
- They both must be on the same subscription.
- The secondary server cannot be a lower performance tier than the primary.

The steps for configuring an active secondary is the same as creating an offline secondary, except you can select the target region, as shown in Figure 4-19.


NEW SECONDARY FOR GEO REPLICATION

Specify secondary settings

PRIMARY	sqlflashcards (Premium P1 - 100 DTUs, o1x9hcj10z, North Central US)
SECONDARY TYPE	OFFLINE [✓ ONLINE (READ-ONLY)]
TARGET REGION	South Central US (Microsoft Recommended)
TARGET SERVER	New SQL database server

FIGURE 4-19 The New Secondary For Geo Replication dialog box for creating an active secondary

Creating an online secondary database (Preview portal)

If your SQL database is a Premium database, you will be able to create an online secondary. To create an online secondary in the Preview portal, follow these steps:

1. Navigate to your SQL database in the management portal accessed via *https://portal.azure.com*.
2. On the Create Secondary blade, change the Secondary Type to Readable.
3. Click Create to create the secondary.

Creating an online or offline secondary with Windows PowerShell

Creating an online or offline secondary can be done with Windows PowerShell using the Start-AzureSqlDatabaseCopy cmdlet.

To create an online secondary, use the following command:

```
Start-AzureSqlDatabaseCopy -ServerName "SecondarySrv" -DatabaseName "Flashcards"
-PartnerServer "NewServer" –ContinuousCopy
```

To create an offline secondary, use the following command:

```
Start-AzureSqlDatabaseCopy -ServerName "SecondarySrv" -DatabaseName "Flashcards"
-PartnerServer "NewServer" –ContinuousCopy -OfflineSecondary
```

Importing and exporting data and schema (existing portal)

Importing and exporting data and schema from a SQL database is essential for a number of situations, including creating a local development or test copy, moving a database to a local instance of SQL Server, or archiving data before you clean up a production database.

To export a database, follow these steps:

1. In the management portal, click the database you want to export.
2. On the task bar, click Export.
3. Enter values for the following:
   - FileName
   - Subscription
   - Blob Storage Account
   - Container
   - Login Name
   - Password

This will create a BACPAC file that can be used to create a database with either an on-premises SQL server, a SQL server in an Azure VM or in Azure SQL Database.

To import the BACPAC into Azure SQL Database, perform the following steps:

4. Click New, Data Services, SQL Database, Import.
5. Click the folder under the BACPAC URL to navigate to the BACPAC file stored in the storage account.
6. Click Open.
7. Enter the following information:
   - Subscription
   - Service Tier
   - Performance Level
   - Server
8. Click the Next arrow.
9. Enter the login details for the new server.
10. Click the check mark. Your new database appears online shortly.

The import process is faster if you use the standard service tier and at least the S2 performance level.

Importing and exporting data and schema (Preview portal)

The Preview portal does not currently support importing and exporting data and schema.

Thought experiment

Managing schema changes

In this thought experiment, apply what you've learned about this objective. You can find answers to these questions in the "Answers" section at the end of this chapter.

Your have 20 developers. They work on a central development copy of SQL Database hosted on Azure. They are constantly changing schema. Sometimes they overwrite each other's changes, which leads to frustration. These developers are not in the same city. Some are in Europe and some are in the United States.

1. What are some things you should consider to make schema changes easier for the developers?
2. What should you consider when creating a backup and restore strategy for this database?
3. What level of SQL Database is probably adequate for the developers? How would you determine that?

Objective summary

- The different editions of Azure SQL Database affect performance, SLAs, backup/restore policies, pricing, geo-replication options, and database size.
- The edition of Azure SQL Database determines the retention period for point in time restores. This should factor into your backup and restore policies.
- It is possible to create an online secondary when you configure Azure SQL Database geo-replication. It requires the Premium Edition.
- If you are migrating an existing database to the cloud, you can use the Azure management portal to move schema and data into your Azure SQL database.

Objective review

Answer the following questions to test your knowledge of the information in this objective. You can find the answers to these questions and explanations of why each answer choice is correct or incorrect in the "Answers" section at the end of this chapter.

1. Which of the following is *not* a requirement for creating an online secondary for SQL Database? (Choose all that apply.)
   - **A.** The secondary database must have the same name as the primary.
   - **B.** They must be on separate servers.
   - **C.** They both must be on the different subscription.
   - **D.** The secondary server cannot be a lower performance tier than the primary.
2. Which metrics should you add to monitoring that will help you select the appropriate level of SQL Database?
   - **A.** CPU Processor Count
   - **B.** CPU Percentage
   - **C.** Physical Data Reads Percentage
   - **D.** Log Writes Percentage
3. From what you know about SQL Database architecture, what should you include in your client application code?
   - **A.** Connection resiliency, because you could failover to a replica.
   - **B.** Transaction resiliency so you can resubmit a transaction in the event of a failover.
   - **C.** Query auditing so you can baseline your current query times and know when to scale up the instance.
   - **D.** A backup and restore operation for the database.

Answers

This section contains the solutions to the thought experiments and answers to the objective review questions in this chapter.

Objective 4.1: Thought experiment

1. You would consider structuring the blob hierarchy so that one of the portions of the path represented the language or region.
2. You would consider creating a CDN on a publicly available container to cache those files locally around the world.

Objective 4.1: Objective review

1. **Correct answers:** A and D
    - **A. Correct:** Only blobs have writable system properties.
    - **B. Incorrect:** Blob user-defined metadata is accessed as a key value pair.
    - **C. Incorrect:** System metadata can influence how the blob is stored and accessed in Azure Storage.
    - **D. Correct:** Containers also have system properties and user-defined metadata.
2. **Correct answers:** B and D
    - **A. Incorrect:** Page files are not faster for streaming files, but are very good for random I/O files like VHDs.
    - **B. Correct:** Block blobs allow files to be uploaded and assembled later. Blocks can be resubmitted individually.
    - **C. Incorrect:** Page blobs are for hard disks, not files for streaming.
    - **D. Correct:** Block blobs have a maximum size of 200 GB. Page blobs can be 1 terabyte.
3. **Correct answers:** A, B, and C
    - **A. Correct:** SSL encrypts all data between client and server and prevents network sniffing.
    - **B. Correct:** If the keys are hidden, they can't be compromised and used to gain access to Table storage.
    - **C. Correct:** Client-side code can easily be seen in the browser. Keep sensitive information stored where few people can access it.
    - **D. Incorrect:** Public containers are not secured.

Objective 4.2: Thought experiment

1. Machine ID seems like a logical candidate for PartitionKey.
2. Shot count time stamp, ordered descending.
3. There might be two tables, one for the machine metadata and one for the shots. You could also make an argument for consolidating both pieces of data into one table for speed in querying.

Objective 4.2: Objective review

1. **Correct answer:** A
   - **A. Correct:** Transactional replication is used in Microsoft SQL Server. Table storage doesn't have anything like that.
   - **B. Incorrect:** Zone redundant storage is valid.
   - **C. Incorrect:** Read access geo-redundant storage is valid.
   - **D. Incorrect:** Geo-redundant storage is valid.
2. **Correct answers:** C and D
   - **A. Incorrect:** They should not necessarily be unique, although they can be for rare-use cases.
   - **B. Incorrect:** You should only use the same partition key if you have a very small entity set.
   - **C. Correct:** Batches can only have operations that exist in the same partition, with the same partition key.
   - **D. Correct:** Even partition sizes will give your application predictable performance because one partition server won't be unevenly loaded with more entities than the others.
3. **Correct answers:** A, B, and C
   - **A. Correct:** All operations have to be in the same partition.
   - **B. Correct:** Total batch size can't be greater than 4 MB.
   - **C. Correct:** Maximum operation count is 100.
   - **D. Incorrect:** There is no minimum operation count for a batch.

Objective 4.3: Thought experiment

1. Typically, the application will store any relevant data and content to be used to produce reports in durable storage. When a report is requested, the request is written to a queue to trigger processing. The queue message must include enough information so that the compute instance listening to the queue can gather the information required for the specific user's report. In some cases, the message may point to a report request stored in a database record. In other cases, the queue message holds enough data to look up all the information required for the report. The work to generate a PDF should be performed on a compute instance that does not compete with mainline application resources, such as the core application web applications and services. This will allow the system to scale PDF generation separately from the main application as needed.
2. The most likely candidate for the compute instance for PDF generation is a cloud service worker role because it provides a built-in mechanism to deploy a Windows Service equivalent in a PaaS environment, but also provides some level of VM customization with startup tasks—possibly necessary for whatever PDF generation tool you may select. If no special software requirements are necessary for producing PDFs, you could also use a WebJob trigger to process queued messages. A VM can also be used, likely with a Windows Service deployed that processes the queue.
3. It will be important to take note of the average memory and disk space used while processing a single message to generate the PDF report. If you monitor the compute instance statistics and slowly begin to scale the number of concurrent messages processed on a single instance, you'll be able to see how much a single instance can handle for configuring auto-scale properties.
4. When you have an idea of the number of concurrent messages that can be processed on a single compute instance, you can identify the number of queued items that should trigger scaling the number of instances. For VMs and cloud services, you can automate this with auto-scale by metrics. For websites and WebJobs, you do not have the option to auto-scale by metric.

Objective 4.3: Objective review

1. **Correct answer:** B

   A. Incorrect: Storage queue messages have a size limit of 64 KB. It is true, however, that a smaller message size can increase throughput since the storage service can support more requests per second when those requests hold a smaller amount of data.

   B. Correct: Storage queues can only store up to 64 KB per message.

   C. Incorrect: Storage queue messages expire after seven days, unlike Service Bus Queue messages, which are persisted until explicitly read and removed.

   D. Incorrect: The message identifier should be considered opaque to the client, although it is returned from the AddMessage() method. When retrieving messages from the queue for processing, the message identifier is provided so that you can use it to subsequently delete the message.

2. **Correct answers:** C and D

   A. Incorrect: A single compute instance can process as many messages as its resources allow for. For example, if processing a message is memory intensive, the number of parallel messages that can be processed will depend on the amount of memory to be consumed for each message that is processed.

   B. Incorrect: A single compute instance can process as many messages as its resources allow for. For example, if processing a message is memory intensive, the number of parallel messages that can be processed will depend on the amount of memory to be consumed for each message that is processed.

   C. Correct: The queue client can request up to 32 messages in a single batch and then process them sequentially or in parallel. Each request from the queue client can request another 32 messages.

   D. Correct: The queue client can request a single message or request up to 32 messages in a batch for processing.

   E. Incorrect: Messages are not deleted when the message is read. Messages must be explicitly deleted.

3. **Correct answers:** A, C, and D

   A. Correct: By creating multiple queues for the application, broken down by logical heuristics that make sense to the application for distributing messages, you can increase scalability by reducing the pressure on a single queue for message processing and throughput.

   B. Incorrect: Websites do not support auto-scale by metric at this time.

   C. Correct: VMs can be scaled based on the number of items in a specified queue.

   D. Correct: Cloud services can be scaled based on the number of items in a specified queue.

Objective 4.4: Thought experiment

1. Users who authenticate to the application should be able to request the report, but since the reports are stored in blobs, it is convenient to be able to share the link directly to the blob. You could have the web application present a page that generates an SAS URI for a report on demand. The user could then copy that link and share in email with others even if they don't have access to the web application.
2. The duration that the link should be valid depends on the typical workflow for your customers who access these reports. For example, if it is acceptable to expect the user who authenticated to download the report right away, or to send the link to someone who will do so right away, limit the SAS token to 30 minutes so that if the email with the link is found at a later time by an unauthorized user, it will be expired. If the link should be shared with someone who may need more time to access the report, but you want to enforce that links can be revoked when some other action has taken place in the application, use a stored access policy with an initial duration that will be acceptable for this workflow. You can then allow users to extend the validity of the SAS token through the web application, or you can programmatically revoke access if you note suspicious activity on the report links through storage logs.

Objective 4.4: Objective review

1. **Correct answers:** A, C, D, and E
   - **A. Correct:** You can generate an SAS token that grants read access to blobs. You can also grant access to modify a blob's contents.
   - **B. Incorrect:** You cannot grant access to create new containers using SAS. This operation requires the storage access key.
   - **C. Correct:** You can grant access to an existing queue and allow add, update, and delete operations using SAS tokens.
   - **D. Correct:** You can grant access to an existing table and allow add, update, and delete operations on entities within that table.
   - **E. Correct:** You can grant access to query the entities of an existing table using SAS tokens.
2. **Correct answers:** A, B, and C
   - **A. Correct:** You can change both the start and expiration dates of an SAS token that is attached to a stored access policy.
   - **B. Correct:** You can revoke all access by an SAS token that is attached to a stored access policy.
   - **C. Correct:** You can revoke specific operations by an SAS token that is attached to a stored access policy. For example, you can remove support for delete operations that were originally granted.

   - **D.** **Incorrect:** You can use the same stored access policy for multiple resources (such as multiple blobs, for example) but this is done at the time of producing the SAS token and associating the stored access policy to the token. You cannot add resources at the policy level.

3. **Correct answers:** B and D
   - **A.** **Incorrect:** CORS is not generally recommended but is a necessary evil for certain types of browser applications to allow for efficient access to storage resources. Try to avoid the use of CORS by using an alternate design if possible.
   - **B.** **Correct:** If blobs are protected resources that require authentication, you should avoid using the storage account key to access them, in particular if this means sharing it with a browser. Instead, generate an SAS token that will be included in any links for requesting the resource and limit the duration the token is valid either with a short duration or a stored access policy you can revoke when the user session ends.
   - **C.** **Incorrect:** CORS is now supported for all storage services, including blobs, queues, and tables.
   - **D.** **Correct:** CORS is not enabled for a new storage account. You have to explicitly enable this feature.

Objective 4.5: Thought experiment

1. You should be looking at using monitoring (through the management portal) and configuring alerts based on latency and availability.
2. You should enable and review the Storage Analytics logs. You can look for usage patterns based on the type of activity seen or errors logged. You can also look for specific types of logs related to a specific event that occurred.

Objective 4.5: Objective review

1. **Correct answer:** A
   - **A.** **Correct:** Capacity metrics include total storage in bytes, the container count, and the object count for blob storage only.
   - **B.** **Incorrect:** You can only set minute metrics programmatically or by using Windows PowerShell cmdlets.
   - **C.** **Incorrect:** By default, retention is not specified, therefore metrics are retained indefinitely. You should set the retention policy to match your compliance requirements and seek to archive if beyond one year.
   - **D.** **Incorrect:** If you disable metrics, all metrics previously collected will be retained until the retention period expires.

2. **Correct answers:** B and D
   - **A. Incorrect:** Logs are stored in a $logs container in Blob storage for your storage account, but the log capacity is not included in your storage account quota. A separate 20-terabyte allocation is made for storage logs.
   - **B. Correct:** Logs can have duplicate entries within a one-hour period; however, you can identify a log entry uniquely with a RequestId and operation number.
   - **C. Incorrect:** The log container cannot be deleted once in use, but the logs within that container can be deleted by authorized callers.
   - **D. Correct:** You can log all or individual operations to all storage services.
3. **Correct answers:** C and D
   - **A. Incorrect:** A log entry is created for all successful authenticated and anonymous requests.
   - **B. Incorrect:** For authenticated calls, all known failed requests are logged, and for anonymous calls, only failed Get requests for error code 304 are logged.
   - **C. Correct:** Server errors generate a log entry.
   - **D. Correct:** All requests to storage resources using SAS tokens are logged.

Objective 4.6: Thought experiment

1. You might consider giving each developer his or her own copy of the database with SQL Database. Then create a central one for merging changes.
2. If developers write database objects and then don't access them again, you might need more than a 14-day backup retention policy. This might lead to a higher edition of SQL Database being used for reasons different than raw performance. You might also consider manually exporting the database if a developer says he or she will be doing something particularly risky.
3. If the developers don't need access to a database the same size as production, they might get away with the basic level of SQL Database. If they do need development databases that are just like production, then choose the level of SQL Database that corresponds with the right size. Developers don't usually put a high load on their servers, so you can ignore the hardware metrics when selecting the appropriate level.

Objective 4.6: Objective review

1. **Correct answer:** D
   - **A. Incorrect:** The secondary database must have the same name as the primary.
   - **B. Incorrect:** They must be on separate servers.
   - **C. Incorrect:** They need to be on the same subscription
   - **D. Correct:** The secondary server cannot be a lower performance tier than the primary.
2. **Correct answers:** B, C, and D
   - **A. Incorrect:** CPU Processor Count is not a valid metric
   - **B. Correct:** CPU Percentage is a valid metric.
   - **C. Correct:** Physical Data Reads Percentage is a valid metric
   - **D. Correct:** Log Writes Percentage is a valid metric.
3. **Correct answers:** A, B, and C
   - **A. Correct:** Connection resiliency, because you could failover to a replica.
   - **B. Correct:** Transaction resiliency so you can resubmit a transaction in the event of a failover.
   - **C. Correct:** Query auditing so you can baseline your current query times and know when to scale up the instance.
   - **D. Incorrect:** You can handle backup and restore operations from the Azure management portal. There's no reason to write custom code for this.

This page intentionally left blank

This page intentionally left blank

Index

About the authors

ZOINER TEJADA is a founder and CEO of Solliance, a Microsoft Azure MVP, and a Google Developer Expert (GDE) for Analytics. Additionally, he has been awarded the Azure Elite and Azure Insider status by Microsoft. Zoiner is passionate about the business of software and tackling innovative areas in software development that range from cloud computing, modern websites, graphics programming, networking, NoSQL/NewSQL distributed databases, scientific computing, digital privacy, and that side of security that involves thinking like hacker.

Zoiner has over 15 years of consulting experience, providing strategic, architectural, and implementation guidance to an array of enterprises and start-ups, all leveraging cutting-edge technologies. He enjoys engaging the greater community by speaking at conferences and user group meetings and by extending his reach through his online courses and published books. Zoiner has earned MCSD certification and has a degree in computer science from Stanford University. You can reach Zoiner at *zoinertejada@solialiance.net*.

MICHELE LEROUX BUSTAMANTE is a founder and CIO of Solliance (solliance.net), the founder of Snapboard.com, a Microsoft Regional Director, and a Microsoft Azure MVP. Additionally, she has been awarded Azure Elite and Azure Insider status and the ASP.NET Insider designation. Michele is a thought leader recognized in many fields, including software architecture and design, identity and access management, cloud computing technologies, security and compliance, and DevOps. During the past 20 years, Michele has held senior executive positions at several corporations, has assembled software development teams and implemented processes for all aspects of the software development lifecycle, and has facilitated numerous successful large-scale enterprise application deployments.

Michele has also been active in the start-up community, bringing a keen understanding of the technical and business needs of a startup. At Solliance, she provides "Start-up Architect" services for activities such as guiding Minimum Viable Product design and delivery, providing necessary preparations to secure funding events, and offering overall advice and guidance to select start-ups.

Michele shares her experiences through presentations and keynote addresses all over the world and has been publishing regularly in technology journals over her entire career. Michele has written several books, including the best-selling book *Learning WCF* (O'Reilly Media, 2007). Find out more about Michele at *linkedin.com/in/michelebusta*.

IKE ELLIS is a data and cloud architect for Solliance. He loves data in all its forms and shapes, whether relational, NoSQL, MPP, JSON, or just sitting in a CSV. Ike consults on SQL Server performance tuning, SQL Server architecture, data warehouse design, and business intelligence projects. Ike is well-known in the industry and speaks at SQL PASS, TechEd, SQL in the City, and other conferences around the world. Ike has been a Microsoft SQL Server MVP for four consecutive years and is a member of Microsoft Azure Insiders. He has MCDBA, MCSE, MCSD, and MCT certifications. Find out more about Ike at *linkedin.com/in/ikeellis* and at *ikeellis.com*.