

Understanding Azure Storage 101

This chapter introduces Microsoft Azure Storage, its types, and the differences between Azure's different models. It also introduces Azure Storage accounts and how to work with them. Moreover, you will have learned how to automate all of these tasks by the end of the chapter.

The following topics will be covered:

- An introduction to Microsoft Azure Storage
- Why Azure Storage?
- Azure terminologies
- **Azure Service Management (ASM)** versus the **Azure Resource Manager (ARM)** model
- Azure Storage types
- Azure Storage accounts
- Automating Azure tasks

An introduction to Microsoft Azure Storage

Storage has always been one of the most important cornerstones of every system. You cannot imagine a **virtual machine (VM)**, web application, or mobile application running without any sort of dependency on storage, and that is what we will cover throughout this book, but from the perspective of the cloud generally, and Azure specifically.

Microsoft Azure Storage is the bedrock of Microsoft's core storage solution offering in Azure. No matter what solution you are building for the cloud, you'll find a compelling use for Azure Storage.

Microsoft Azure Storage is not just a traditional storage system; it's scalable and can store up to hundreds of terabytes of data, meaning that it fits almost every scenario you can ever imagine in many fields, such as IT, science, medical fields, and so on.

At the time of writing, Microsoft Azure is generally available in 36 regions, with plans announced for six additional regions, as shown in the following figure:

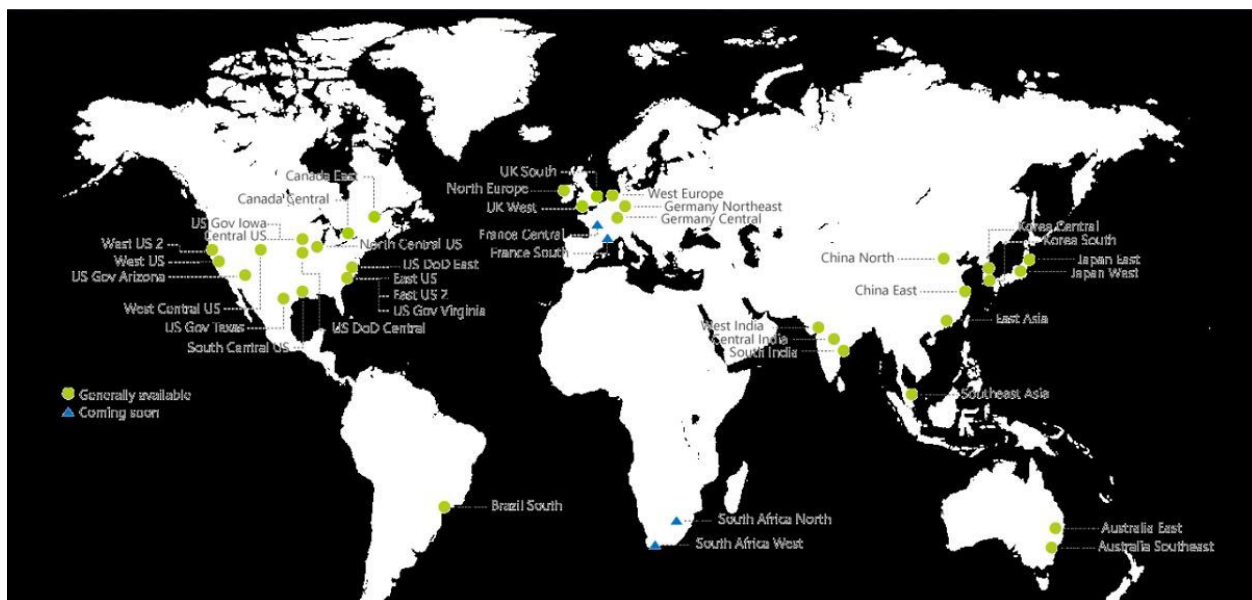


Figure 1.1: Azure regions

This global presence means you can host your storage in the nearest region and access it from anywhere in the world. Considering that Microsoft continues to build new data centers in new regions, the latency between you and your services in Azure will decrease.



You can find out the nearest region to you with the lowest latency via the following website <http://www.azureSpeed.com/>.

Azure services are available in 140 countries around the globe and supports 17 languages and 24 currencies.

Why Azure Storage?

There are many reasons for using Azure Storage which will be covered throughout this book. Below is a sneak peak of a couple of them:

- **Global presence:** You can host your storage wherever you want in the available Azure regions, allowing you to provide applications close to your user base.
- **Redundancy and recovery:** As mentioned earlier in this chapter, Azure has a global presence which can be leveraged to maintain storage availability using data replication even if a disaster occurs in a region, which will be covered later in this chapter.
- **Many flavors:** Azure Storage has many flavors, based on resiliency, durability, connectivity, performance, and so on, which can be used according to your needs in different scenarios. This will be covered later in this chapter.
- **Pay as you go:** Pay as you go has always been one of the distinguished reasons for using the cloud generally. It is no surprise that Azure Storage supports this model as well.

Terminologies

Due to an overlap of terms and some misperceptions about the ways that Azure Services are delivered, terminology is a sticking point even for people who have been working with the technology for some time. The following table provides accurate but short definitions for terms related to Azure services. These definitions will be expanded upon in detail throughout the book, so don't worry if they are confused at first:

Term	Definition
On-premises	Means that your data center is hosted and managed from within your company.
Off-premises	Means that your data center is hosted and managed in a remote place (for example, hosted and managed outside your company).
Azure VM	The feature of providing VMs to Azure subscribers.
Blade	The window that pops up when you click on one of the Azure services in the Azure portal, such as Virtual machines.
Journey	A set of blades or chain of selections. For instance, when you select Virtual Machines inside the Azure Portal, click on an existing virtual machine and then select its settings.
Resource group	Provides a logical container for Azure resources (to help manage resources that are often used together).

Images	The VMs you've created in Azure and then captured to be used as templates for later use, or the VMs you've imported to Azure.
Disks	Virtual Hard Disks (VHDs) that you attach to the VMs you create in Azure.
Virtual network	Allows VMs and services that are part of the same virtual network to access each other. However, services outside the virtual network have no way of connecting to services hosted within virtual networks unless you decide to do so.
Fault domain	A group of resources that could fail at the same time. For example, they are in the same rack.
Upgrade/update domain	A group of resources that can be updated simultaneously during system upgrades.
Storage container	The place where storage Blobs are stored, it is also used to assign security policies to the Blobs stored inside it.
Network Security Group (NSG)	Determines the protocols, ports, who and what can access Azure VMs remotely.
VM agent /extensions	Software components that extend the VM functionality and simplify various management operations.
Scale set	A set of identical VMs, that auto scale without pre-provisioning the VMs based on metrics such as CPU, memory, and so on.

Availability set	When VMs are placed in an availability set, the VMs are spread over different fault domains and update domains, which ensures that, in the event of a rack failure, not all instances are brought down at the same time. If any updates are applied to a host on which there is one of your VMs and a restart is required, it will not be applied to the other VM within the same availability set.
System Preparation (SysPrep)	A Windows preparation tool that's used when you have captured a VM and want to use it as a template, which ensures that there's no more than one VM with the same properties, which would lead to a conflict between the VMs.

ASM versus ARM model

At the time of writing, Azure services are being provided via two portals, which follow two different models. The Azure classic portal follows the ASM model and the Azure portal follows the ARM model.

Azure classic portal (ASM model)

Historically, Azure services were provided via one portal prior to 2014, the classic portal, which can be accessed via the following URL <https://manage.windowsazure.com/>.

The model that was used for that portal is called the **ASM model**, within which each resource existed independently. You could not manage your resources together; you had to build up and track each resource. For example, you would have to manage storage from the storage blade, and the same goes for the virtual networks, VMs, and so on. So, when your environment got bigger, there would be chaos in the management scheme. You would have to know which VMs were stored in which storage account, and that could lead to some critical situations, such as reaching the IOPs limits of the storage account. In turn, this could result in you accidentally creating a new VM and assigning it to that storage account. As a result, the VM would run with terrible performance. This would not be your only concern when working with the ASM model; you might want to delete a solution with multiple resources, which you would have to do manually for each resource.

When you open the Azure classic portal, it will look like the following screenshot:

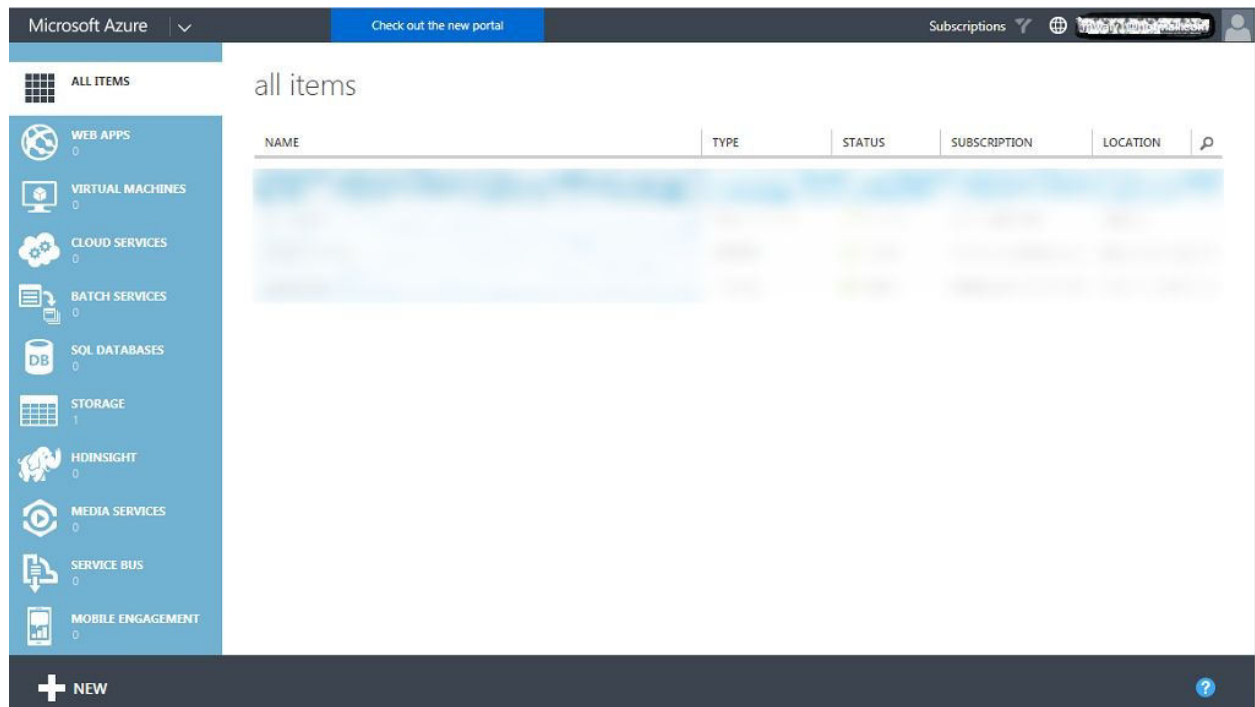


Figure 1.2: Azure classic portal

Azure portal (ARM model)

In 2014, Microsoft launched a new portal which follows a new model, called the **ARM model**. This portal can be accessed via the following URL <https://portal.azure.com/>.

This model depends on the concept of resource groups, which means you can group all your resources within a container, resulting in resources being deployed in parallel. As a result, you will not face the same problems as you did with the classic portal.

The following diagram describes the deployed resources through the ARM model:

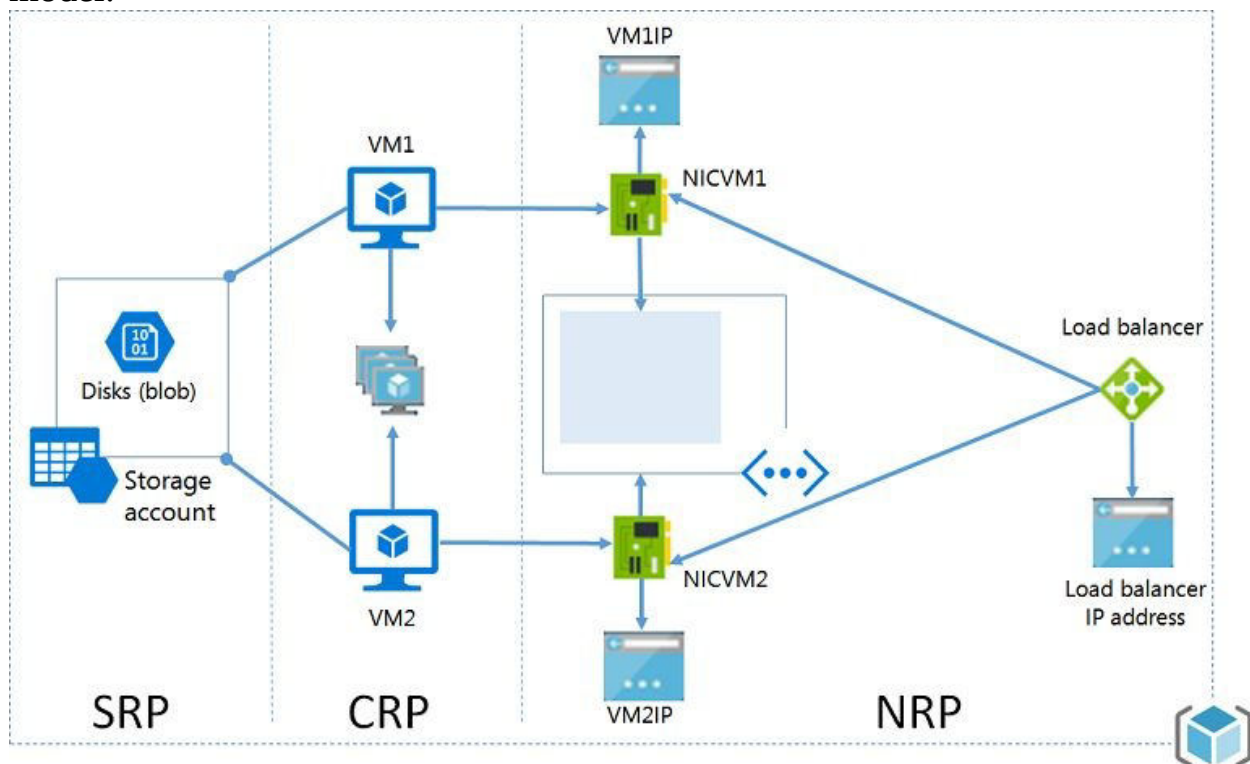


Figure 1.3: Resource Manager management model at a high level Here are the benefits you will gain using this portal:

- Ability to manage your resources as a group instead of managing them separately.
- Use **Role Based Access Control (RBAC)** to control access to resources, so that you can assign permissions to a user on a resource or some resources but not to other resources (as it was in the classic portal).
- Use tags to organize and classify your resources, which can help you with billing. For example, you might want to monitor the billing of some resources that make up a solution, for example, a web server. By assigning a tag to the resources that make up

- that solution, you would be able to monitor its billing, and so on.
- Support the usability of JSON to deploy resources instead of using the portal.
- Deploy resources in parallel instead of deploying them sequentially and waiting until every resource deployment finishes to deploy another one.
- Specify dependencies during the deployment of resources. For example, a VM will not be created until a storage account and a virtual network are deployed because the VM VHD would need a place to be stored in and an IP Address from a virtual network./li>
- Reuse the JSON template to deploy a solution with the same specifications.
- Resources with the same life cycle should be gathered in the same resource group.
- Resources in different regions can be in the same resource group.
- The resource cannot exist in multiple resource groups.
- A resource group supports RBAC, wherein a user can have access to some specific resources, but no access to others.
- Some resources can be shared across resource groups, such as storage accounts.
- ARM VMs can only be placed in ARM storage accounts.

When you open the Azure portal, it will look like the following screenshot:

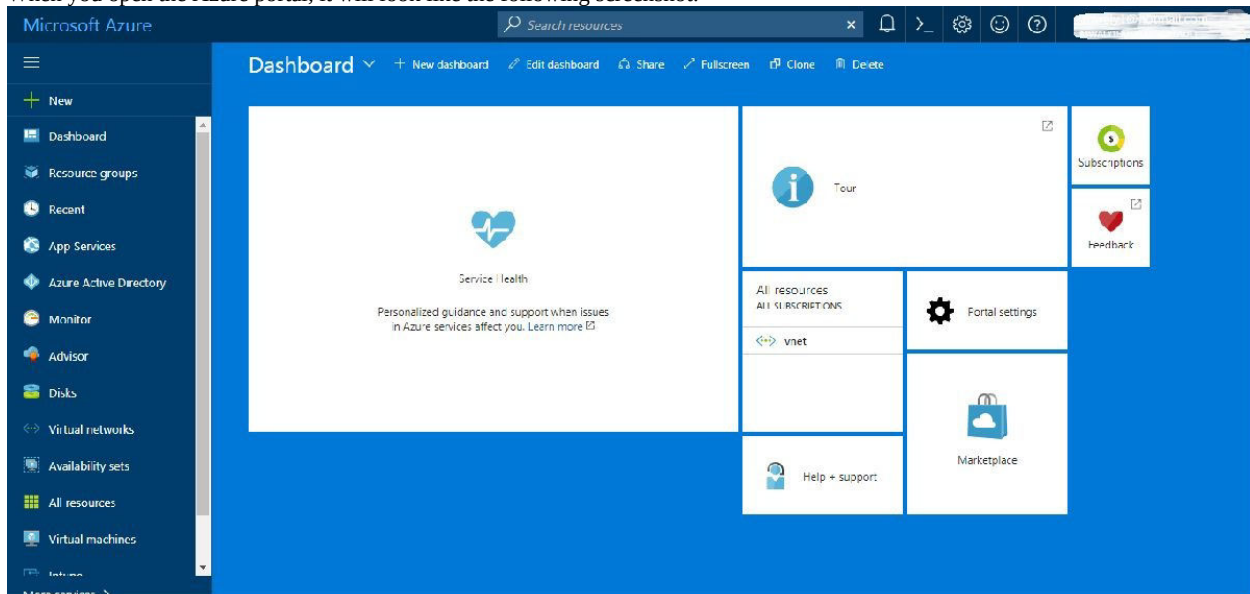


Figure 1.4: Azure portal



- You can change the background of the portal by double-clicking on any unused area of the dashboard. You can navigate between four colors (blue, dark blue, white, and black).
- For further information about the difference between the ARM and ASM models, check out the following article: <https://blogs.technet.microsoft.com/meamcs/2016/12/22/difference-between-azure-service-manager-and-azure-resource-manager/>.

Deployment model tricks

Here are some things you need to consider:

- You cannot create a VM using the ARM model and assign it to a virtual network built using the ASM model
- You cannot use a prebuilt image that was created using ASM APIs to build a VM using the ARM model, but as a workaround, you can copy the VHD files from the storage account in the classic portal to a storage account created in the ARM model
- You can migrate assets from the ASM model to the ARM model
- Every resource must be assigned to a resource group, so whenever you want to move a resource between resource groups you must remove it from its current resource group, then add it to the new resource group.

Azure Storage types

Azure Storage has many types and even subtypes of those types in order to satisfy Azure services' consumer needs and to fit most scenarios.

The most common types can be classified based on the following factors:

- Durability (replication)
- Performance (Standard versus Premium)
- Persistency (persistent versus non-persistent)

Durability

One of the most buzzing questions about the cloud generally is: *What if, for some reason, the SAN/servers that store my data are completely damaged? How can I restore my data?*

The answer is very simple because Microsoft Azure Storage is durable and supports data replication, therefore you can make sure your storage is highly available.

Replication ensures that your data is copied somewhere else, whether in the same data center, another data center, or even another region.



For more info about the SLA of Azure Storage, you can access it via the following link: https://azure.microsoft.com/en-us/support/legal/sla/storage/v1_2/.

Replication types

Microsoft Azure supports multiple options for data replication. You can use whatever you feel suits your business, especially as every type has its own price.

In order to calculate your solution's cost, you can use the **Azure Pricing Calculator**, which can be reached via the following URL: <https://azure.microsoft.com/en-us/pricing/calculator/>.

Locally redundant storage

Locally redundant storage (LRS) replicates three copies of your data within the same data center you have your data in. The write requests you do with your storage are not committed until they are replicated to all three copies, which means it replicates synchronously. Not only this, it also makes sure that these three copies exist in different update domains and fault domains. You can revise the terms guide at the beginning of the chapter to understand what the update domain and the fault domain are.

Drawbacks:

- The least durable option, as it replicates only within the same data center
- Your data will be lost if a catastrophic event, such as a volcanic eruption or flood, affects the data center

Advantages:

- It is the cheapest type compared to the other types
- It is the fastest type of data replication, offering the highest throughput since it replicates within the same data center, mitigating the risk of data loss that would occur during data replication caused by a failure having occurred on the original data host
- It is the only available replication type that can be used with Premium Storage at the time of writing

Zone Redundant Storage

Zone Redundant Storage (ZRS) replicates three copies of data across two or three data centers within one of two regions asynchronously, plus the three copies of data stored within the same data center of the original source of the data.

Drawbacks:

- This type can only be used for Block Blobs (one of the Azure services covered in the next chapter), and a Standard Storage account (general purpose Standard Storage accounts will be covered later in this chapter)
- Does not support metrics or logging
- Does not support conversion for other replication types, such as LRS, GRS, and vice versa
- If a disaster occurs, some data might be lost, because the data replicates to the other data center asynchronously
- If a disaster occurs, there will be some delay in accessing your data until Microsoft failover to the secondary zone

Advantage: It provides higher durability and availability for data than LRS, as it not only replicates in the same data center but also in other data centers.

Geo-redundant storage

Geo-redundant storage (GRS) replicates data not only within the same region but also to another region. Firstly, it replicates three copies of data within the same region synchronously, then it replicates another three copies of data to other regions asynchronously.

Drawbacks:

- If a disaster occurs, some data might be lost, because the data replicates to the other regions asynchronously
- If a disaster occurs, there will be some delay in accessing your data until Microsoft initiates failover to the secondary region

Advantages:

- It provides the highest durability and availability, even if a disaster occurs in an entire region
- Unlike ZRS, if the original source of data faces an outage, there will be no possibility of data loss if the other three copies that exist within the same region don't face an outage too, as it replicates synchronously within the same region.

Read-access geo-redundant storage

Read-access geo-redundant storage (RA-GRS) follows the same replication mechanism of GRS, in addition, to read access on your replicated data in the other regions.

Drawback: If a disaster occurs, some data might be lost, because the data replicates to the other region asynchronously.

Advantages:

- It provides the highest durability and availability, even if a disaster occurs in a whole region
- If a disaster occurs, you still only have read access to the storage, but no write access until Microsoft initiates failover to the secondary region
- The region with the read access can be used for data retrieval by the nearest offices to it without the need to go to another region to access the data; as a result, the data latency will decrease



Regarding replication between different regions, it will not work with just any two regions; the regions must be paired.

For example the West Europe region can replicate with North Europe, and not any other region.

For more information about paired regions, check the following article: <https://docs.microsoft.com/en-us/azure/best-practices-availability-paired-regions>.

Performance

As mentioned earlier, Azure provides services for all business types and needs. There are two types based on Azure Storage performance--Standard and Premium.

Standard Storage

Standard Storage is the most common type for all the VMs sizes available in Azure. The Standard Storage type stores its data on non-SSD disks. It is commonly used with workloads within which the latency is not critical. Plus, it is low cost and has support for all Azure Storage services (which will be covered in the next chapter). It is also available in all regions.

Premium Storage

Premium Storage is designed for low latency applications, such as SQL server, which needs intensive IOPs. Premium Storage is stored on SSD disks, that is why it costs more than Standard Storage. Microsoft recommends using Premium Storage for better performance and higher availability.



More details about Standard and Premium Storage will be covered throughout the book.

Persistency

Another type of Azure Storage depends on data persistence, which means whether data will be there or not after stopping and starting the VM within which your data exists.

Persistent storage

Persistent storage means that the data will be there after stopping and restarting the VM within which your data exists.

Non-persistent storage

Non-persistent storage means that the data will be gone after restarting the VM within which your data exists.



Further details about storage persistency will be covered in Chapter 3, Azure Storage for VMs.

Azure Storage accounts

An Azure Storage account is a secure account that provides access to Azure Storage services (which will be covered in the next chapter), and a unique namespace for storage resources.

During the creation of a new Azure Storage account, you will have the option to choose one of two kinds of storage accounts:

- General-purpose storage account
- Blob storage account

General-purpose storage accounts

A general-purpose storage account gives you access to all Azure Storage services, such as Blobs, Tables, Files, and Queues (these will be covered in the next chapter), and has two performance tiers:

- Standard Storage tier
- Premium Storage tier

Both were covered within the *Performance* type topic earlier in this chapter.

Blob storage accounts

Unlike a general-purpose storage account, not all Azure Storage services are meant to be stored in a Blob storage account because they are dedicated to storing unstructured data. Therefore, a Blob storage service is the only type allowed to be accessed by a Blob storage account. However, it only supports block and appends Blobs.

A Blob storage account has a usage pattern called access tiers, which determines how frequently you access your data and based on that you will get billed.

Currently, there are two types:

- Hot access tier
- Cool access tier

Hot access tier

With the hot access tier, objects will be accessed more frequently, so you will pay less for data access, but pay more for data size.

Cool access tier

With the cool access tier, objects will be accessed less frequently, so you will pay more for data access, but less for data size.

Azure Storage Account tips

The following tips will increase your knowledge about Azure Storage, and will definitely help you when you want to design a storage solution on Azure:

- You cannot switch between an Azure general-purpose storage account and an Azure Blob storage account
- You can switch between access tiers with a Blob storage account, but with the possibility of additional charges being incurred
- A Blob storage account does not support ZRS replication type at the time of writing
- Premium Storage only supports Locally Redundant Storage as a replication type at the time of writing
- Premium Storage is not supported for a Blob storage account at the time of writing
- Azure supports up to 200 storage accounts per subscription by default
- A storage account can store data up to 500 TB
- Azure Storage supports encryption for only two storage services at the time of writing (Blobs and Files), and you can enable it during the storage account creation
- If you are using REST APIs to connect to Azure Storage, you can secure the transfer by enabling that option during the creation of a storage account
- Only lowercase letters and numbers are supported for the name of the storage account

Creating an Azure Storage account

Let's get our hands dirty with creating a storage account with the following parameters:

- Name: packtpubsa
- Deployment model: Resource Manager
- Account kind: General purpose
- Performance: Standard
- Replication: Locally-redundant storage (LRS)
- Storage service encryption (blobs and files): Disabled
- Secure transfer required: Disabled
- Subscription: Select the right subscription for this task
- Resource group: Create a new or select an existing resource group, as per your needs
- Location: Select the nearest location to you

Without further ado, let's get started:

1. Open the Azure portal from here: <https://portal.azure.com/>.
2. Click on More services and a new blade will open. In the search bar, write storage account, as shown in the following screenshot:

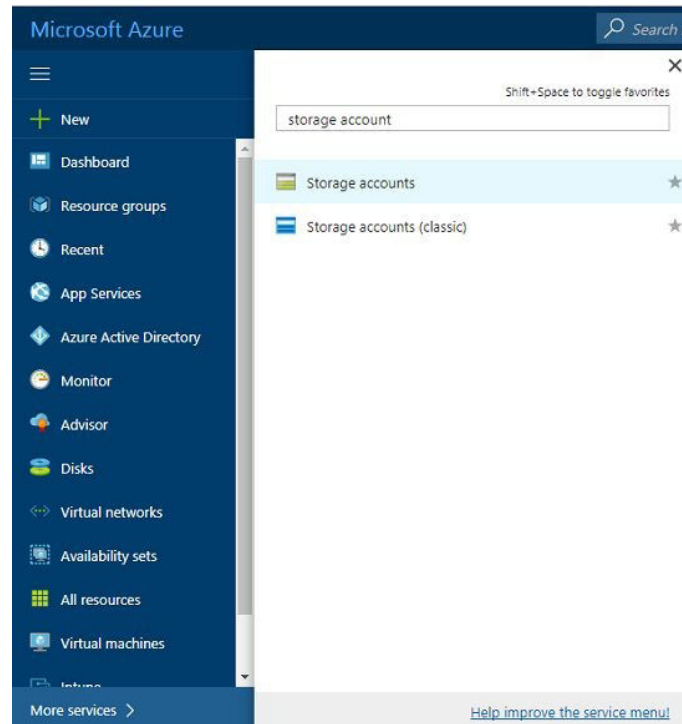


Figure 1.5: Searching for a storage accounts service

3. Click on Storage accounts and a new blade will open. Click on Add, as shown in the following screenshot:

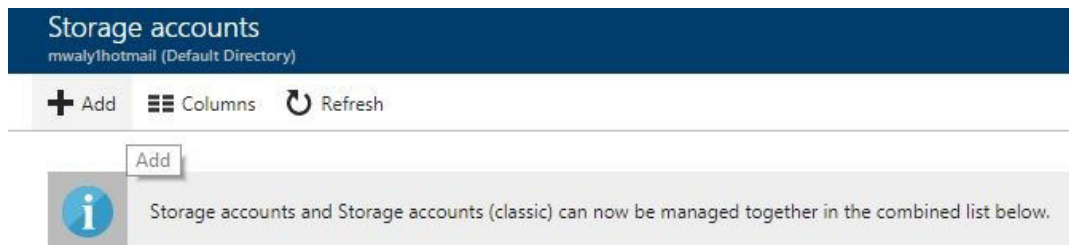


Figure 1.6: Adding a new storage account

4. A new blade will open, wherein you need to fill in the fields and determine the types as per your needs:

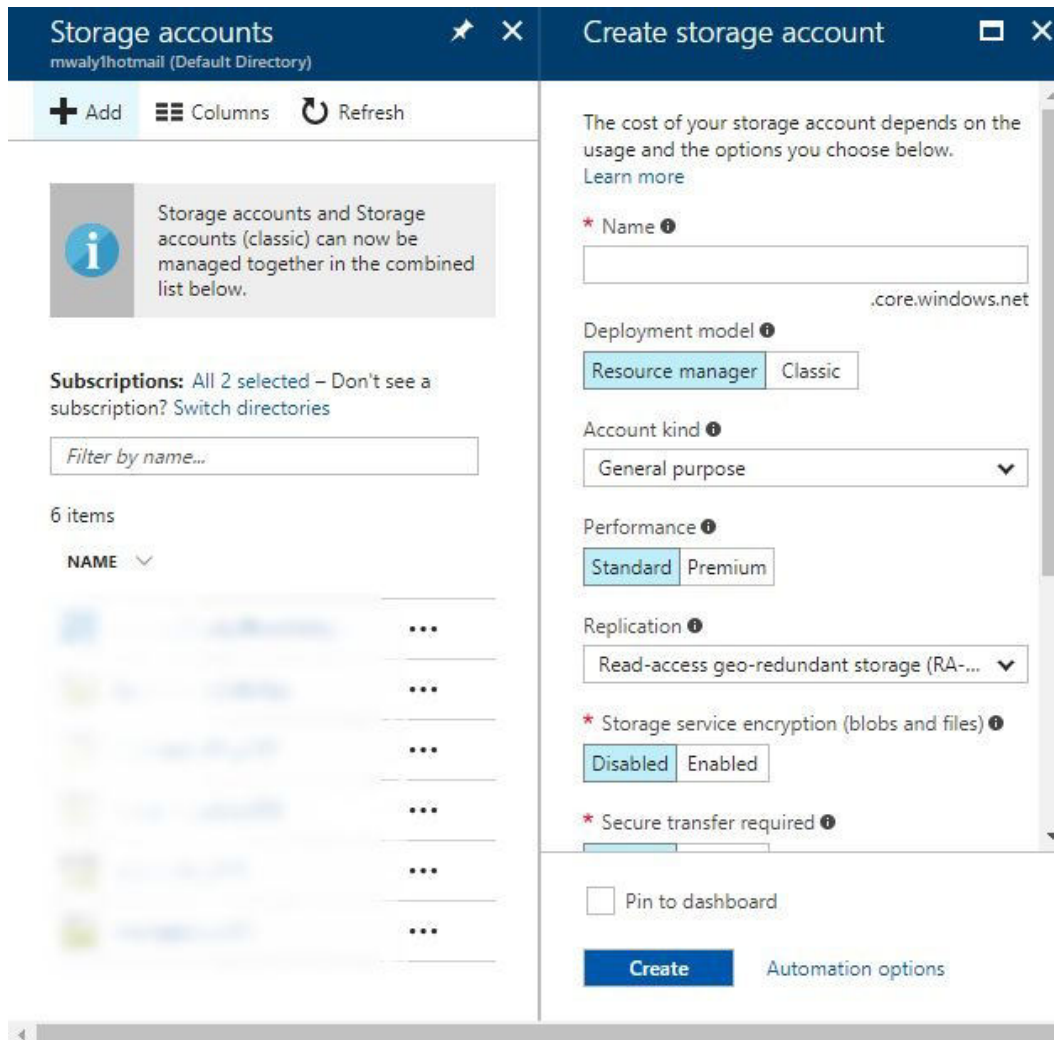


Figure 1.7: Creating a new storage account blade

5. Fill in the fields as before, and click on Create:

Create storage account

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

* Name

packtpubsa

.core.windows.net

Deployment: model

Resource manager

Classic

Account kind

General purpose

Performance

Standard

Premium

Replication

Locally-redundant storage (LRS)

* Storage service encryption (blobs and files)

Disabled

Enabled

* Secure transfer required

Disabled

Enabled

* Subscription

Subscription

* Resource group

Create new

Use existing

PacktPub

* Location

West Europe

☐ Pin to dashboard

Create

Automation options

Figure 1.8: Filling in the fields of the blade

6. Once done, you can find your storage account in the Storage accounts blade:

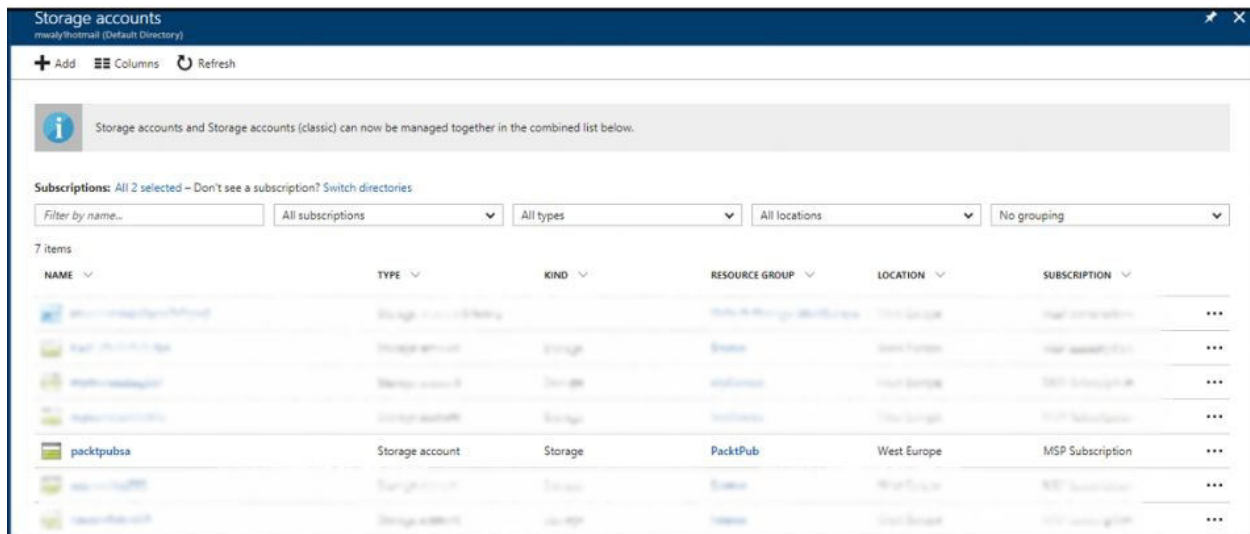


Figure 1.9: Storage accounts blade

Something to keep in mind:



- When using Storage service encryption (blobs and files), your data is encrypted once it is written in Azure and gets decrypted once you try to access it.
- When you enable Secure transfer required, the storage account will only be accessed using HTTPS if you are using REST APIs, and since Azure file service uses **Server Message Block (SMB)**, the connection will fail if you are using SMB 2.1 and SMB 3.0 without encryption, and the same goes for the Linux SMB client in some flavors.
- When you enable Secure transfer required, you will not be able to use a custom domain, because Azure Storage does not currently support that. As a result, you can only use the default `.core.windows.net` domain.

Automating your tasks

It is no surprise that we commonly face repetitive and time-consuming tasks. For example, you might want to create multiple storage accounts. You would have to follow the previous guide multiple times to get your job done. This is why Microsoft supports its Azure services with multiple ways of automating most of the tasks that can be implemented in Azure. Throughout this book, two of the automation methods that Azure supports will be used.

Azure PowerShell

PowerShell is commonly used with most Microsoft products, and Azure is no less important than these products.

Mainly, you can use Azure PowerShell cmdlets to manage your Azure Storage, however, you should be aware that Microsoft Azure has two types of cmdlets: one for the classic portal, and another for the portal we are using.

The main difference between the cmdlets of the classic portal and the current portal is there will be an `RM` added to the cmdlet of the current portal.

For example, if you want to create a storage account in the classic portal, you would use the following cmdlet: **New-AzureStorageAccount**

But for the current portal, you would use:

| **New-AzureRMStorageAccount**

By default, you can use Azure PowerShell cmdlets in Windows PowerShell; you will have to install its module first.

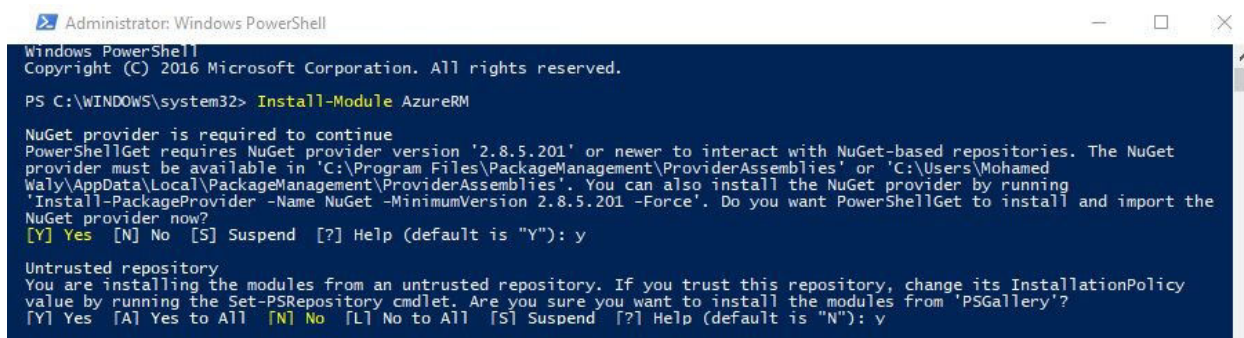
Installing the Azure PowerShell module

There are two ways to install the Azure PowerShell module:

- Download and install the module from the following link: <https://www.microsoft.com/web/downloads/platform.aspx>
- Install the module from the PowerShell Gallery

Installing the Azure PowerShell module from the PowerShell Gallery

1. Open PowerShell in elevated mode.
2. To install the Azure PowerShell module for the current portal, run the `Install-Module AzureRM cmdlet`. If your PowerShell requires the NuGet provider, you will be asked to agree to install it and you will have to agree to the installation policy modification, as the repository is not available on your environment, as shown in the following screenshot:



```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\WINDOWS\system32> Install-Module AzureRM

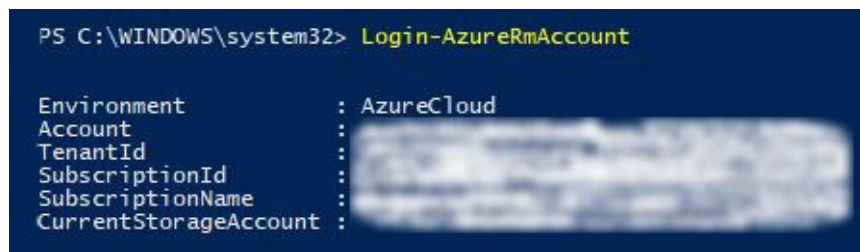
NuGet provider is required to continue
PowerShellGet requires NuGet provider version '2.8.5.201' or newer to interact with NuGet-based repositories. The NuGet
provider must be available in 'C:\Program Files\PackageManagement\ProviderAssemblies' or 'C:\Users\Mohamed
Waly\AppData\Local\PackageManagement\ProviderAssemblies'. You can also install the NuGet provider by running
'Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force'. Do you want PowerShellGet to install and import the
NuGet provider now?
[Y] Yes [N] No [S] Suspend [?] Help (default is "Y"): y

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this repository, change its InstallationPolicy
value by running the Set-PSRepository cmdlet. Are you sure you want to install the modules from 'PSGallery'?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): y
```

Figure 1.10: Installing the AzureRM PowerShell module

Creating a storage account in the Azure portal using PowerShell

1. Log in to your Azure account using the `Login-AzureRmAccount` cmdlet. You will be prompted to enter your account credentials:

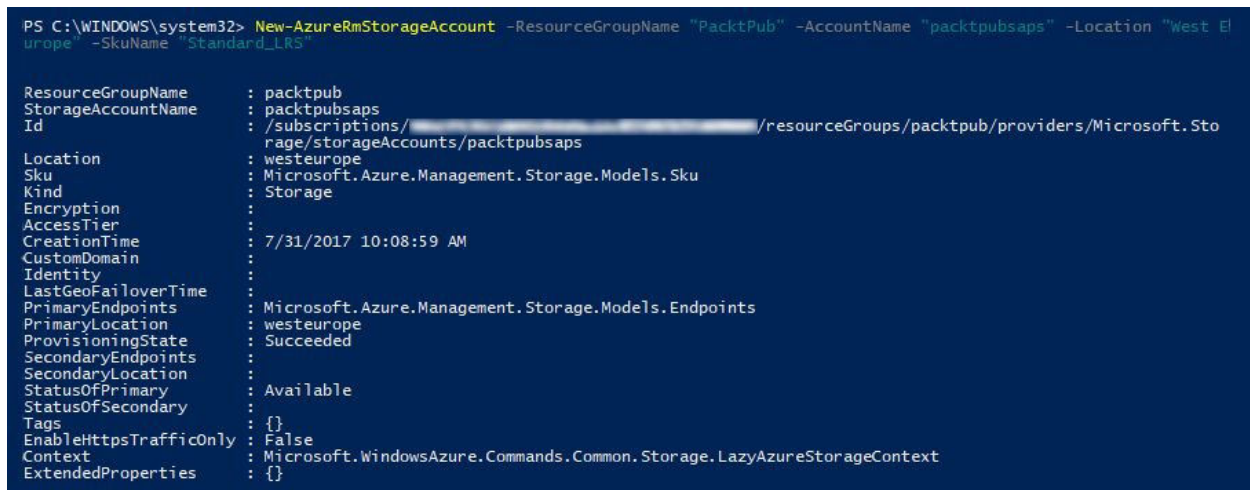


```
PS C:\WINDOWS\system32> Login-AzureRmAccount

Environment      : AzureCloud
Account          : 
TenantId         : 
SubscriptionId   : 
SubscriptionName : 
CurrentStorageAccount :
```

Figure 1.11: Log in to Azure via PowerShell

2. Create another storage account with the same properties as we used for the portal, but with a different name:



```
PS C:\WINDOWS\system32> New-AzureRmStorageAccount -ResourceGroupName "PacktPub" -AccountName "packtpubsaps" -Location "West Europe" -SkuName "Standard_LRS"

ResourceGroupName      : packtpub
StorageAccountName     : packtpubsaps
Id                     : /subscriptions/[redacted]/resourceGroups/packtpub/providers/Microsoft.StorageAccounts/packtpubsaps
Location               : westeurope
Sku                    : Microsoft.Azure.Management.Storage.Models.Sku
Kind                   : Storage
Encryption             : 
AccessTier             : 
CreationTime           : 7/31/2017 10:08:59 AM
CustomDomain           : 
Identity               : 
LastGeoFailoverTime    : 
PrimaryEndpoints       : Microsoft.Azure.Management.Storage.Models.Endpoints
PrimaryLocation        : westeurope
ProvisioningState       : Succeeded
SecondaryEndpoints     : 
SecondaryLocation      : 
StatusOfPrimary        : Available
StatusOfSecondary      : 
Tags                   : {}
EnableHttpsTrafficOnly : False
Context                : Microsoft.WindowsAzure.Commands.Common.Storage.LazyAzureStorageContext
ExtendedProperties      : {}
```

Figure 1.12: Creating a new storage account using PowerShell

3. Congratulations! You have created a storage account using PowerShell.

Azure command-line interface

The Azure **command-line interface (CLI)** is open source, cross-platform, and supports implementing all the tasks you can do in the Azure portal with commands.

Azure CLI comes in two flavors:

- Azure CLI 2.0, which only supports the current Azure portal
- Azure CLI 1.0, which supports both portals

Throughout the book, we will be using Azure CLI 2.0. So, let's get started with its installation.

Installing the Azure CLI 2.0

To understand what the Azure CLI 2.0 is capable of, we need to install it. Let's do so by following these steps:

1. Download Azure CLI 2.0 from the following link: <https://azurecliprod.blob.core.windows.net/msi/azure-cli-2.0.12.msi>.
2. Once downloaded, you can start the installation by following the screenshots shown here:
 1. Run the executable files as administrator, and once the wizard opens, click on Install:



Figure 1.13: Installing the Azure CLI 2.0

2. Once you click on Install, it will start to validate your environment to check whether it is compatible with it or not, then it starts the installation:

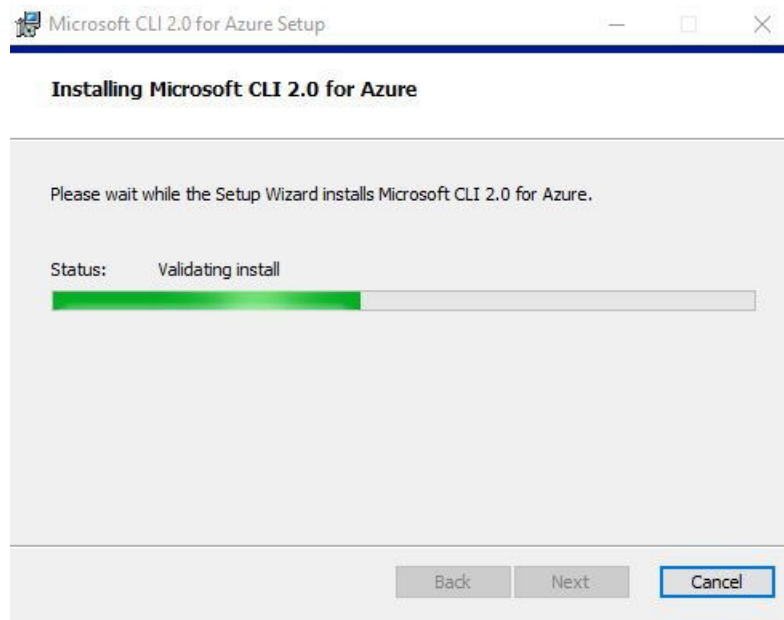


Figure 1.14: Installing Azure CLI 2.0

3. Once the installation completes, you can click on Finish, and you are good to go:

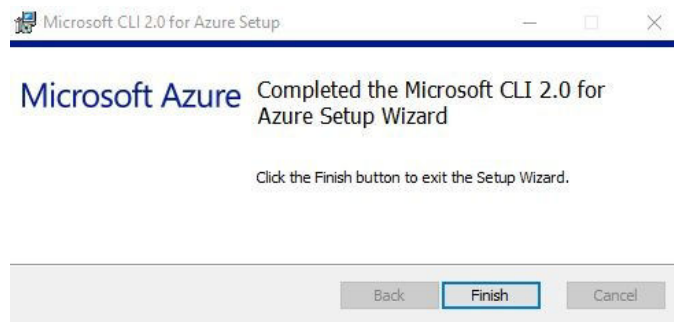


Figure 1.15: Installing Azure CLI 2.0

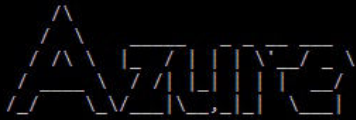
3. Once done, you can open the cmd and type `az` to access Azure CLI commands, as shown in the following diagram:

```
C:\WINDOWS\system32>az

Welcome to Azure CLI!
-----
Use `az -h` to see available commands or go to https://aka.ms/cli.

Telemetry
-----
The Azure CLI collects usage data in order to improve your experience.
The data is anonymous and does not include commandline argument values.
The data is collected by Microsoft.

You can change your telemetry settings with `az configure`.



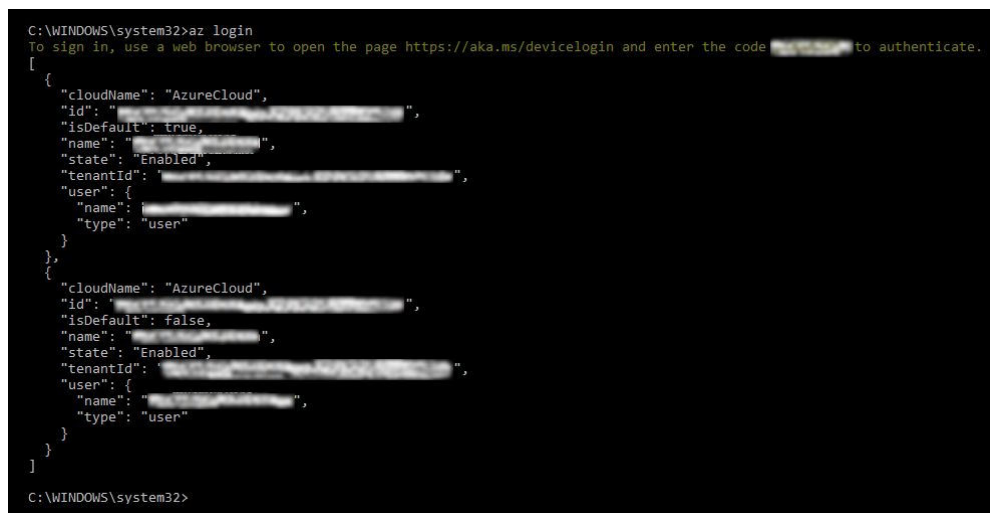
Welcome to the cool new Azure CLI!
```

Figure 1.16: Opening the Azure CLI using CMD

Creating a Storage account using the Azure CLI 2.0

Let's get our hands dirty with the Azure CLI 2.0 to create an Azure Storage account:

1. Log in to your Azure account using the `az login` command. You have to open the URL that pops up on the CLI and enter the code, as shown in the following screenshot:



```
C:\WINDOWS\system32>az login
To sign in, use a web browser to open the page https://aka.ms/devicelogin and enter the code [REDACTED] to authenticate.
[
  {
    "cloudName": "AzureCloud",
    "id": "[REDACTED]",
    "isDefault": true,
    "name": "[REDACTED]",
    "state": "Enabled",
    "tenantId": "[REDACTED]",
    "user": {
      "name": "[REDACTED]",
      "type": "user"
    }
  },
  {
    "cloudName": "AzureCloud",
    "id": "[REDACTED]",
    "isDefault": false,
    "name": "[REDACTED]",
    "state": "Enabled",
    "tenantId": "[REDACTED]",
    "user": {
      "name": "[REDACTED]",
      "type": "user"
    }
  }
]
C:\WINDOWS\system32>
```

Figure 1.17: Logging in to Azure via the Azure CLI 2.0

2. Create another storage account with the same properties as we used for the portal, but with a different name, as shown in the following screenshot:

```

C:\WINDOWS\system32>az storage account create --location "West Europe" --name packtpubsacli --resource-group "PacktPub"
--sku "Standard_LRS"
{/ Finished ..
  "accessTier": null,
  "creationTime": "2017-07-31T11:14:18.249342+00:00",
  "customDomain": null,
  "enableHttpsTrafficOnly": false,
  "encryption": null,
  "id": "/subscriptions/[REDACTED]/resourceGroups/packtpub/providers/Microsoft.Storage/storageAccounts/packtpubsacli",
  "kind": "Storage",
  "lastGeoFailoverTime": null,
  "location": "westeurope",
  "name": "packtpubsacli",
  "primaryEndpoints": {
    "blob": "https://packtpubsacli.blob.core.windows.net/",
    "file": "https://packtpubsacli.file.core.windows.net/",
    "queue": "https://packtpubsacli.queue.core.windows.net/",
    "table": "https://packtpubsacli.table.core.windows.net/"
  },
  "primaryLocation": "westeurope",
  "provisioningState": "Succeeded",
  "resourceGroup": "packtpub",
  "secondaryEndpoints": null,
  "secondaryLocation": null,
  "sku": {
    "name": "Standard_LRS",
    "tier": "Standard"
  },
  "statusOfPrimary": "available",
  "statusOfSecondary": null,
  "tags": {},
  "type": "Microsoft.Storage/storageAccounts"
}

```

Figure 1.18: Creating an Azure storage account using the Azure CLI 2.0

Summary

So far, we have covered some preliminary subject matters regarding Azure generally, and Azure Storage specifically. Some things were not covered in detail, but detailed discussions will be raised in the coming chapters.

Next, Azure Storage architecture and Azure services will be covered in detail. Therefore, the knowledge gained in this chapter is required for a better understanding of the coming chapter.

Delving into Azure Storage

This chapter covers Microsoft Azure Storage services and how to work with them. For a better understanding of what is going on behind the scenes, the Azure Storage architecture and how to secure your Azure Storage will be covered too. The best practices that need to be followed to have a highly available application are also covered. Then, we will go through client libraries, which can be used as a way of managing Azure Storage. Finally, all manually created tasks will be automated using PowerShell and the Azure CLI 2.0.

The following topics will be covered in this chapter:

- Azure Storage services
- Understanding the Azure Storage architecture
- Securing Azure Storage
- Storage design for highly available applications
- Understanding client libraries
- Automating tasks

Azure Storage services

Azure Storage has multiple services that would fit most scenarios. At the moment, there are four types of Azure Storage services, which are as follows:

- Blob storage
- Table storage
- Queue storage
- File storage

Each of these services can be used for different scenarios, which we will cover in detail shortly.

Blob storage

Blob stands for **binary large object**. This type of service can store almost everything since it stores unstructured data, such as documents, files, images, VHDs, and so on.

Using the Azure Blob storage service makes you capable of storing everything we have just mentioned, and able to access them from anywhere using different access methods, such as URLs, REST APIs, or even one of the Azure SDK Storage Client Libraries, which will be covered later in this chapter.

There are three types of Blob storage:

- **Block blobs:** They are an excellent choice to store media files, documents, backups, and so on. They are good for files that are read from A-Z (sequential reading).
- **Page blobs:** They support random access for files stored on them, that is why, they are commonly used for storing VHDs of Azure Virtual Machines.
- **Append blobs:** They are similar to block blobs, but are commonly used for append operations. For example, each time a new block is created, it will be added to the end of the blob. One of the most common use cases within which append blobs can be used is logging, where you have multiple threads that need to be written to the same blob. This is an excellent solution that would help you to dump the logs in a fast and safe way.

Creating Blob storage

Let's see how we can create Blob storage that everyone has read/write access to in the storage account we created in the last chapter.

1. Navigate to the storage we created in the last chapter using the portal, as shown in the following screenshot:

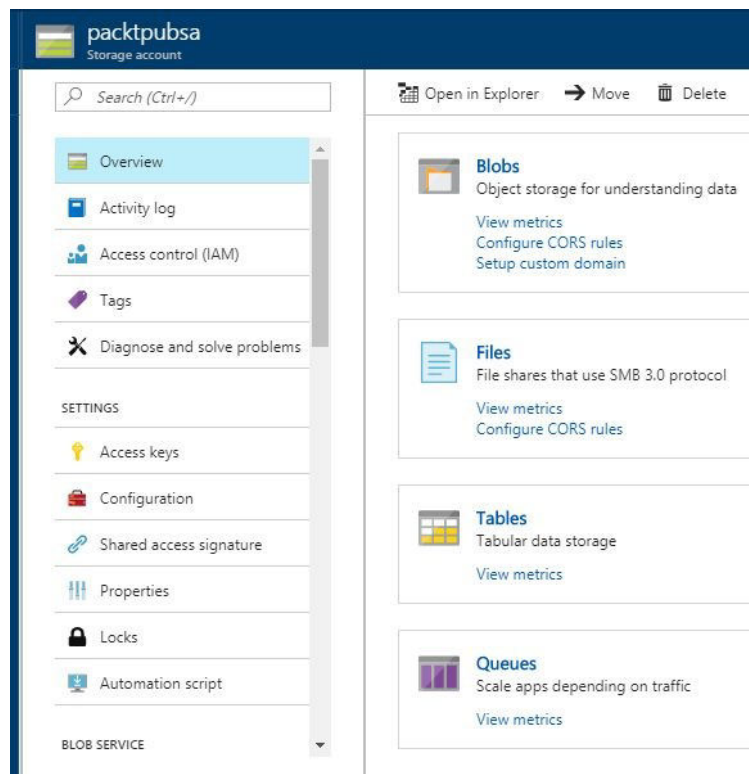


Figure 2.1: Azure Storage services overview

2. You can see all the storage services in the previous screenshot. To manage blobs, you have to click on Blobs, and a new blade will appear, as shown in the following screenshot:

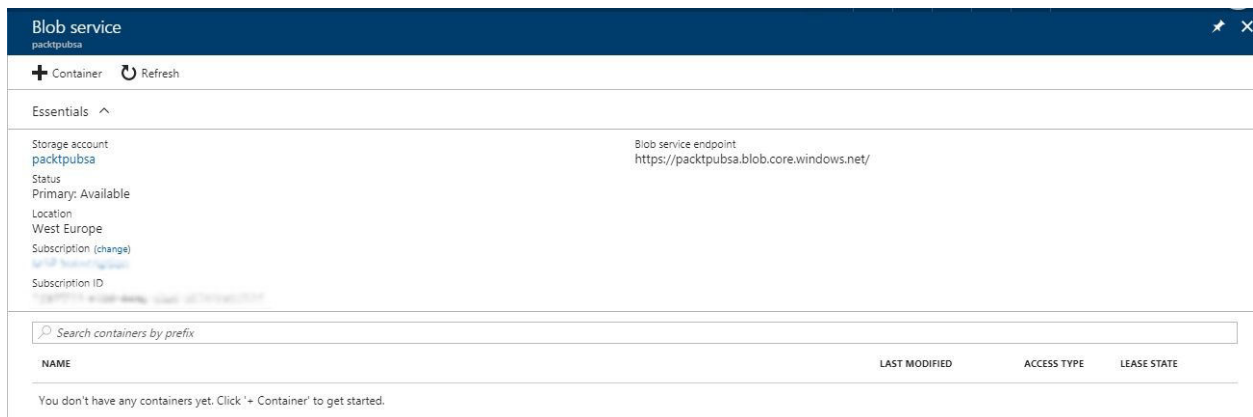


Figure 2.2: Azure Blob service overview

3. In order to create a blob service, you have to create a container in which the blob service will be stored. To do this, you click on Container to create a new one. However, it is not a straightforward creation process, as you will be asked to specify a name and an access type, as shown in the following screenshot:

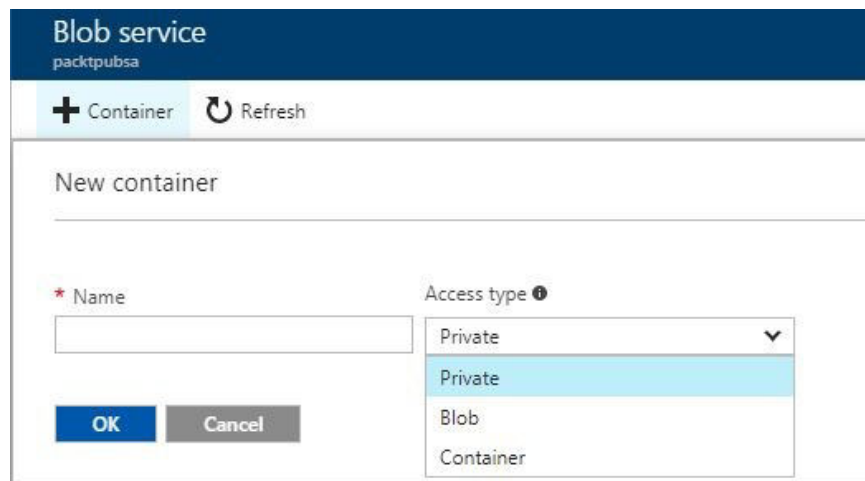
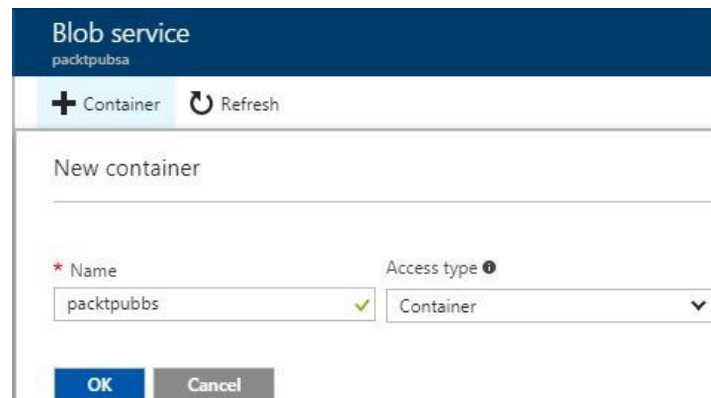


Figure 2.3: Creating a blob service

Here is a short description for the access types:

- Private: This option means that only the storage account owner has access to the blobs created within this container using the access key, therefore you can grant access privileges to any other users
- Blob: This option means that the blobs created within this container will be accessed from outside by read permissions only

- Container: This option means that the blobs created within the container will be publicly available with read and write access
4. Since we want to create a blob service that everyone has read/write access to, we will choose Access type as Container and name it packtpubbs, as shown in the following screenshot:



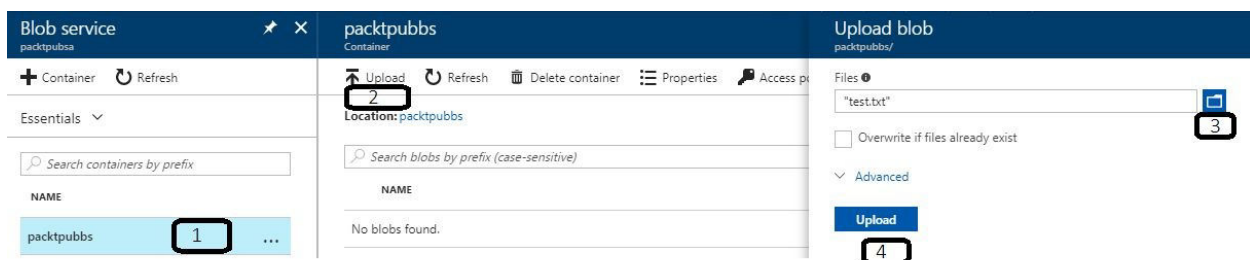
The screenshot shows a 'Blob service' creation window. At the top, it says 'Blob service' and 'packtpubsa'. Below that, there are buttons for '+ Container' and 'Refresh'. A 'New container' section contains a text input for 'Name' with the value 'packtpubbs' and a green checkmark, and a dropdown for 'Access type' set to 'Container'. At the bottom are 'OK' and 'Cancel' buttons.

Figure 2.4: Creating a blob service



The access type of storage containers can be changed even after creation.

5. Once created, you can open the blob and start uploading your data to it, as shown in the following screenshot:



The screenshot shows the Azure portal interface. On the left, the 'Blob service' 'packtpubsa' is shown with a list of containers, including 'packtpubbs' which is circled with a '1'. In the center, the 'packtpubbs' container is selected, showing an 'Upload' button circled with a '2'. On the right, the 'Upload blob' dialog is open, showing a file named 'test.txt' circled with a '3' and an 'Upload' button circled with a '4'.


Figure 2.5: Uploading a .txt file to the blob

6. For further customization of the uploaded blob, click on Advanced and you will see options such as specifying the Blob type, Block size, to which folder it will upload the blob, and so on:

Upload blob

packtpubbs/

Files ⓘ

"test.txt" 

☐ Overwrite if files already exist

Blob type ⓘ

Block blob ▼

☒ Upload .vhd files as page blobs (recommended)

Block size ⓘ

100 MB ▼

Upload to folder

Upload

Figure 2.6: Advanced customization to the uploaded blob

Blob storage key points

The following tips should be considered, as they will help you when designing your storage solution using blob services:

- Blob storage supports both standards, but only page blobs support Premium Storage.
- Block blobs are named as such because files larger than 64 MB are uploaded as smaller blocks, then get combined into one final blob.
- You cannot change the type of blob once it has been created.
- Block blobs are named as such because they provide random read/write access to 512-byte pages.
- Page blobs can store up to 8 TB.
- Storage containers built for Blob storage may only contain lowercase letters, hyphens, and numbers, and must begin with a letter or a number, however, the name cannot contain two consecutive hyphens. The name length can vary between 3 to 63 characters.
- The maximum number of blocks in a block blob or append blob is 50,000.
- The maximum size of the block in a block blob is 100 MB. As a result, a block blob can store data of up to 4.75 TB.
- The maximum size of the block in an append blob is 4 MB. As a result, an append blob can store data of up to 195 TB.

Table storage

High availability and scalability are key factors when you want to work with your storage, and that is exactly what is offered by Table storage. Table storage is a Microsoft NoSQL data store that can be used for the massive amount of semi-structured, non-relational data.

Data is stored in tables as a collection of entities, where entities are like rows, and each entity has a primary key and a set of properties, considering that a property is like a column.

The Table storage service is schema-less, therefore multiple entities in the same table may have different properties.

An entity has three properties:

- PartitionKey
- RowKey
- Timestamp

PartitionKey

The `PartitionKey` is a sequential range of entities that have the same key value. The way that tables are partitioned is to support load balancing across storage nodes, where tables entities are organized by partition. It is considered the first part of an entity's primary key.

It may be a string value with a size of up to 1 KB, and every insert, update, or delete operation must be included in the partition key property.

RowKey

`RowKey` is the second part of the entity's primary key. It is a unique identifier for the entity, and every entity within the table is uniquely identified by the combination of `PartitionKey` and `RowKey`.

Like `PartitionKey`, it is a string value that may be up to 1 KB, and every insert, update, or delete operation must be included in the `RowKey` property.

Timestamp

`Timestamp` is a datetime value, and it is kept on the server side to record when the last modification of the entity occurred.

Every time there is a modification for the entity, the `Timestamp` value is increased. Considering that this value should not be set on insert or update operations.

Creating Table storage

Let's see how we can create Table storage in the storage account we created in the last chapter:

1. Navigate to the storage we created in the last chapter using the portal, as shown in the following screenshot:

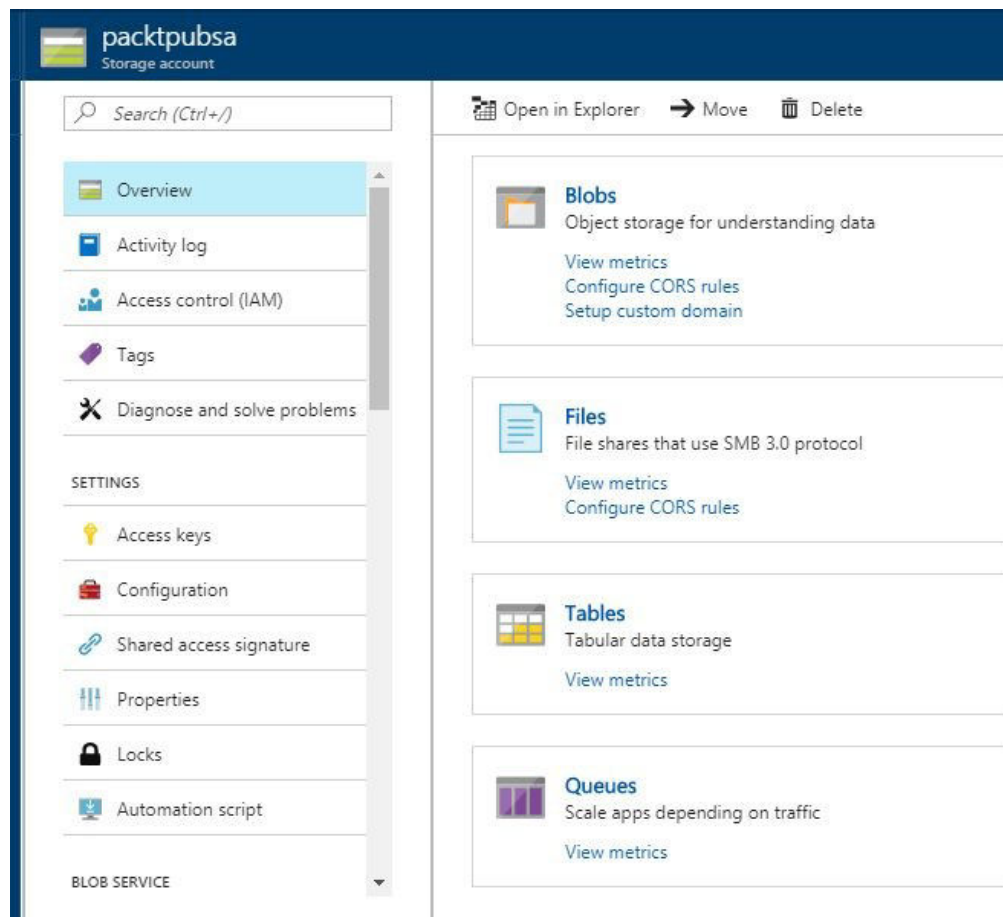


Figure 2.7: Azure Storage services overview

2. You can see all the storage services in the previous screenshot. To manage tables, you have to click on Tables, and a new blade will appear, as shown in the following screenshot:

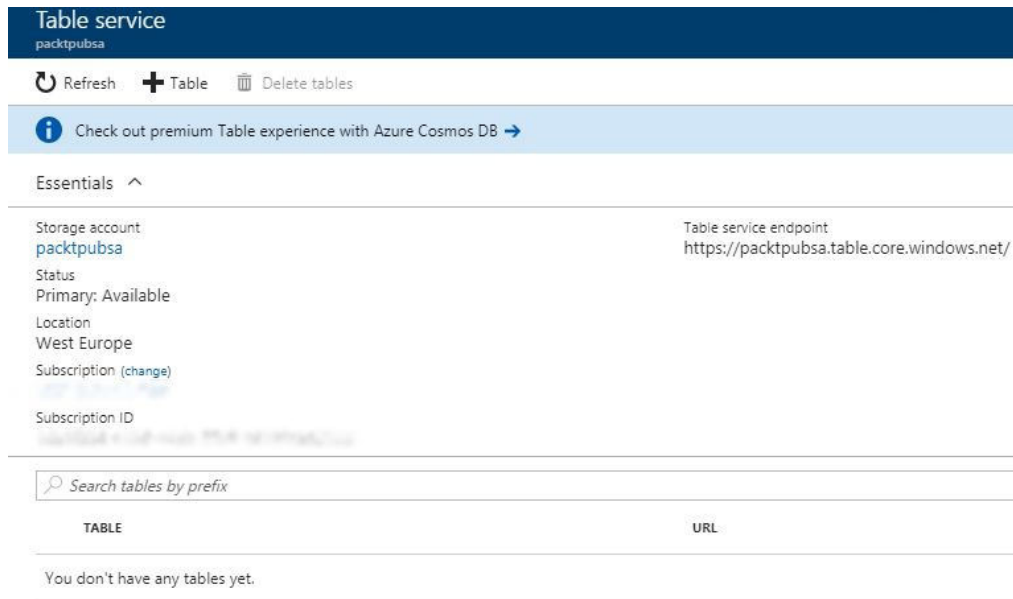


Figure 2.8: Azure Table service overview

3. In order to create a table service, just click on Table, and specify the Table name, as shown in the following screenshot:



Figure 2.9: Azure Table service creation

4. Once done, you will see that the table has been created, as shown in the following screenshot:

Table service

packtpubsa

Refresh

+ Table

Delete tables

Check out premium Table experience with Azure Cosmos DB

Essentials

Storage account

packtpubsa

Status

Primary: Available

Location

West Europe

Subscription (change)

Subscription

Subscription ID

15a7f524-e75b-44ab-b5ab-af23f6a4d52d

Table service endpoint

https://packtpubsa.table.core.windows.net/

Search tables by prefix

TABLE	URL
packtpubtable	https://packtpubsa.table.core.windows.net/packtpubtable

Figure 2.10: The created table



For database developers and administrators who are interested in learning how to access a created table and start working with it, you can check the following link: <https://docs.microsoft.com/en-us/azure/storage-dotnet-how-to-use-tables>.

Table storage key points

The following tips should be considered, as they will help you when designing your storage solution using the Table storage service:

- Table storage supports Standard Storage, but its support for Premium Storage is in preview at the time of writing
- Table storage is significantly lower in cost than traditional SQL
- The entity can have up to 252 custom properties, and 3 system properties (PartitionKey, RowKey, and Timestamp)
- The entity's properties data cannot exceed 1 MB
- Table names must follow the following rules:
 - They are case sensitive
 - They contain only alphanumeric characters, considering that they cannot begin with a numeric character
 - They cannot be redundant within the same storage account
 - You can name a table with another table name written in reverse
 - Their length varies between 3 and 63 characters

Queue storage

Queue storage is a storage service that is used to provide persistent and reliable messaging for application components.

Generally, it creates a list of messages that process asynchronously, following the **First-In, First-Out (FIFO)** model. Not only this, asynchronous tasks and building process workflows can be managed with Queue storage too.

One of the most common scenarios is passing messages from an Azure Web Role to an Azure Worker Role.

Queue storage is not the only messaging solution available at Azure; there are also Service Bus queues, which can be used for more advanced scenarios.



More information about the differences between Azure Queues storage and Azure Service Bus queues, can be found via the following URL: <https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-azure-and-service-bus-queues-compared-contrasted>.

Creating Queue storage

Let's see how we can create Queue storage in the storage account we created in the last chapter.

1. Navigate to the storage we created in the last chapter using the portal, as shown in the following screenshot:

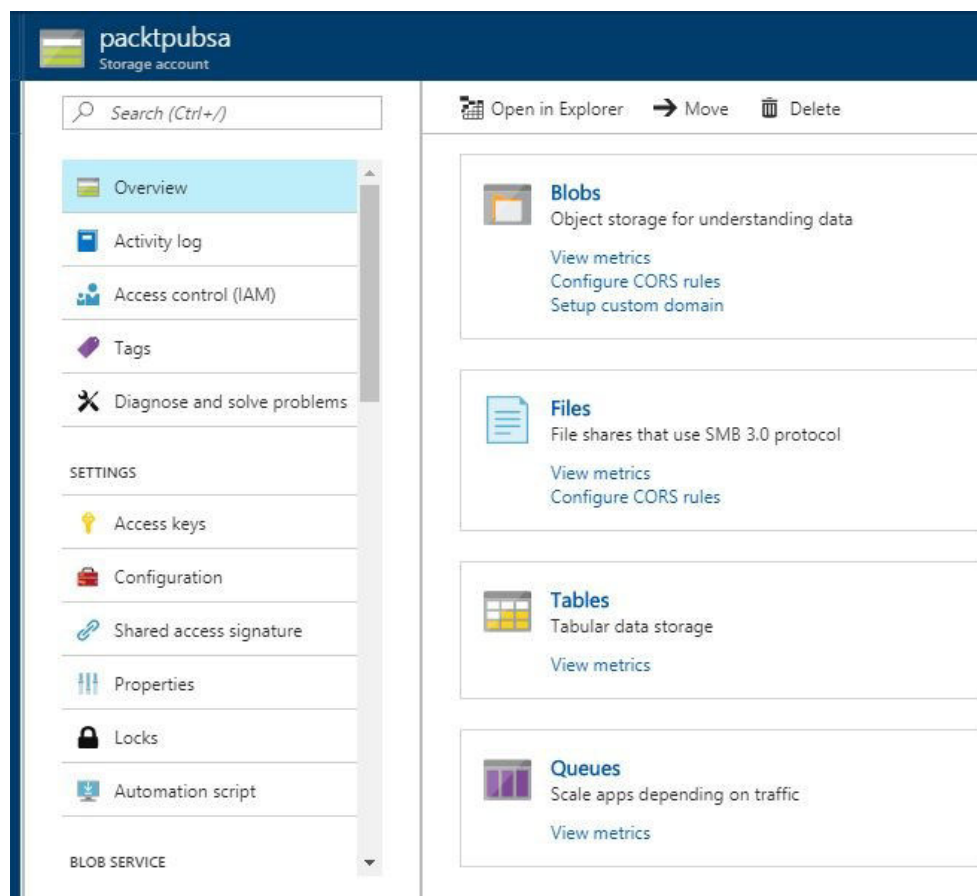


Figure 2.11: Azure Storage services

2. You can see all the storage services in the previous screenshot. To manage queues, you have to click on Queue, and a new blade will appear, as shown in the following screenshot:

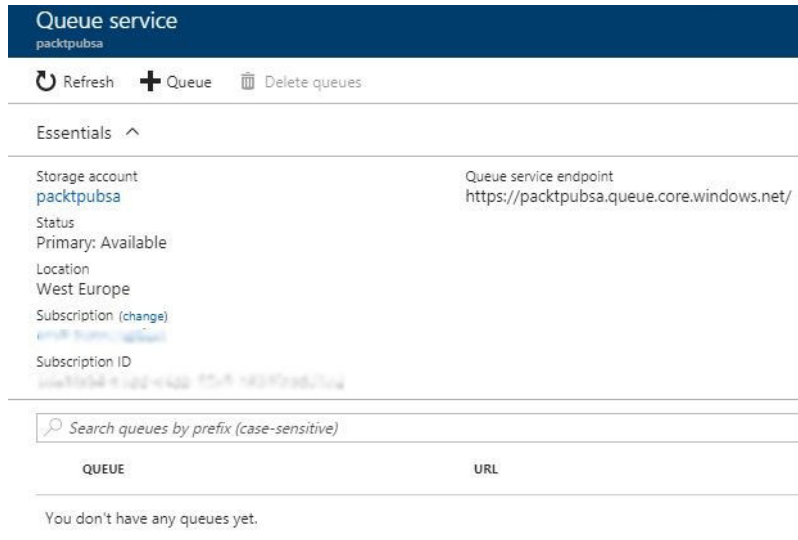


Figure 2.12: Azure Queue service overview

3. In order to create a Queue service, just click on Queue and specify the Queue name, as shown in the following screenshot:

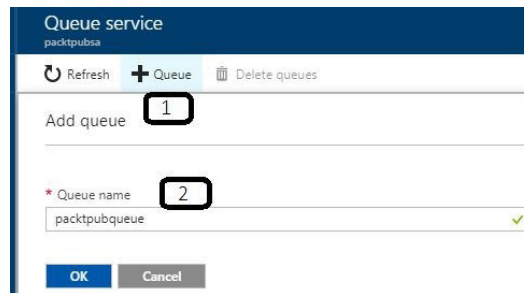


Figure 2.13: Azure Queue service creation

4. Once done, you will see that the Queue has been created, as shown in the following screenshot:

Queue service

packtpubsa

Refresh

+ Queue

Delete queues

Essentials

Storage account

packtpubsa

Queue service endpoint

https://packtpubsa.queue.core.windows.net/

Status

Primary: Available

Location

West Europe

Subscription (change)

small training queue

Subscription ID

12345678901234567890123456789012

Search queues by prefix (case-sensitive)

QUEUE	URL
packtpubqueue	https://packtpubsa.queue.core.windows.net/packtpubqueue

Figure 2.14: The created queue



For developers who are interested in learning how to access a created Queue and start working with it, you can check the following link: <https://docs.microsoft.com/en-us/azure/storage/storage-dotnet-how-to-use-queues>.

Queue storage key points

The following tips should be considered, as they will help you when designing your storage solution using the Queues service:

- Queue messages can be up to 64 KB in size, however, a Queue can contain messages up to the limiting size of the storage account.
- The maximum lifetime of a message in a queue is 7 days.
- As mentioned previously, messages follow the FIFO order, however, they can be out of order if an application crash occurs, which is why it would be better to use Azure Service Bus queues for a scenario where the FIFO order is highly important.
- Messages can be scheduled for delivery later.
- A Queue name may only contain lowercase letters, hyphens, and numbers, and must begin with a letter or number. It cannot contain two consecutive hyphens. Name length varies from between 3 and 63 characters.

File storage

The File storage service is the easiest and coolest service to work with. You can use it to create network file shares on Azure, and access them from anywhere in the world.

Server Message Block (SMB) and **Common Internet File System (CIFS)** are the only protocols that can be used to access these file shares.

As a result, multiple Azure VMs and on-premises machines can access the same file share and have read and write privileges on it. Azure File shares can be mounted to different operating systems, such as, Windows, Linux, and even macOS concurrently.

File storage advantages

The File storage service has some good reasons to use it, which are:

- **Software as a service (SaaS) service:** The Azure File storage service is considered a SaaS service because you do not have to manage the hardware, operating system, patches, and so on. It is simply fully managed.
- **Shareability:** It can be shared across multiple machines providing read and write privileges for all of those machines.
- **Automation:** It supports working with PowerShell and the Azure CLI, which can be used to create scripts to automate repetitive tasks with minimal administration.
- **Flexibility and high availability:** Using the Azure File storage service eliminates concerns regarding administration and outage issues that you face with traditional file servers.

Creating File storage

Let's see how we can create File storage in the storage account we created in the last chapter.

1. Navigate to the storage we created in the last chapter using the portal, as shown in the following screenshot:

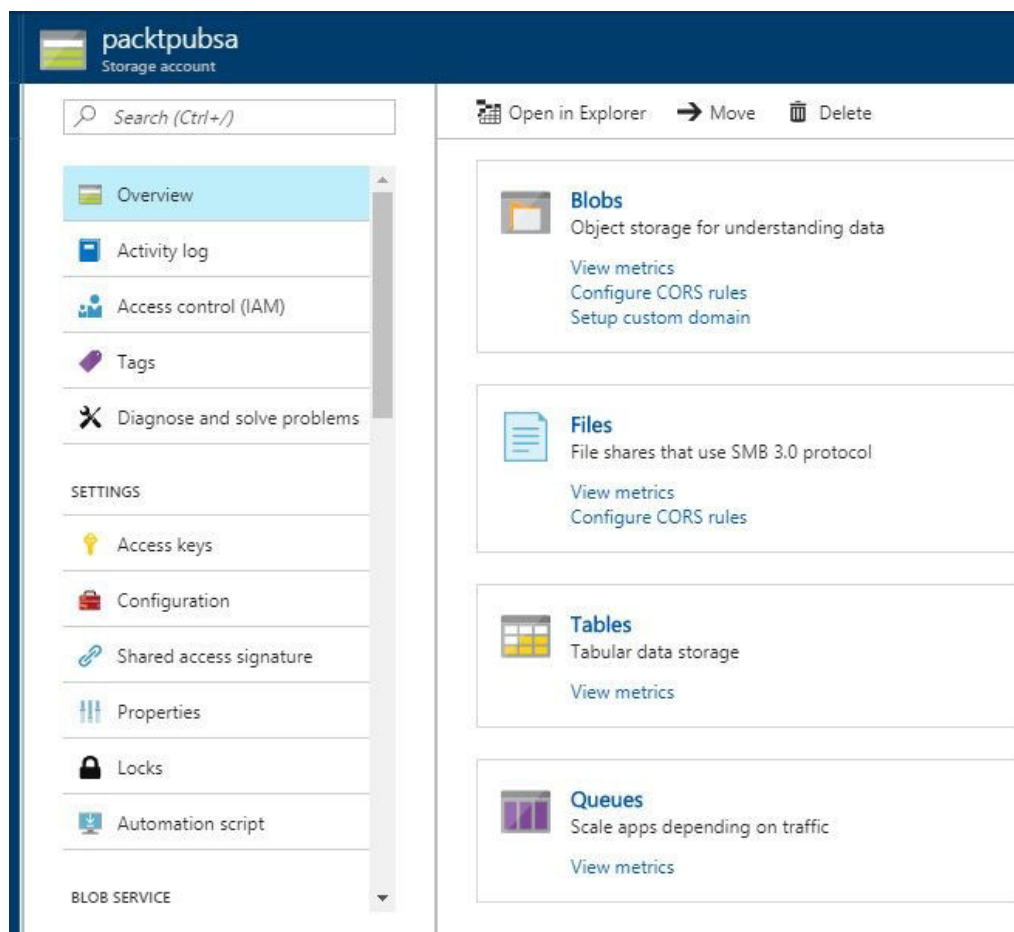


Figure 2.15: Azure Storage services

2. You can see all the storage services in the previous screenshot. To manage files, you have to click on Files and a new blade will appear, as shown in the following screenshot:



Figure 2.16: Azure File service overview

3. In order to create a file share, just click on File share, and specify the file share Name and its Quota, considering that the quota is optional, as shown in the following screenshot:

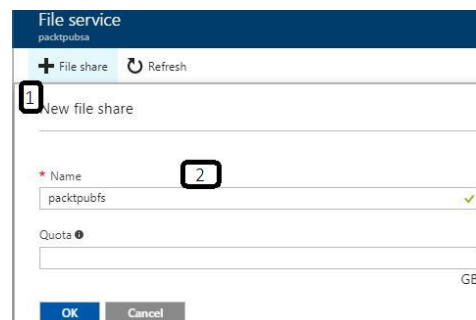


Figure 2.17: Azure File share creation

4. Once done, you will see that the file share has been created, as shown in the following screenshot. Considering that we never specified a quota, it used the maximum space the storage account can store; therefore, you have to design your file share properly according to your needs, and with the proper quota, to avoid any future issues caused by the space used. Also, you can change the quota even after file share creation:

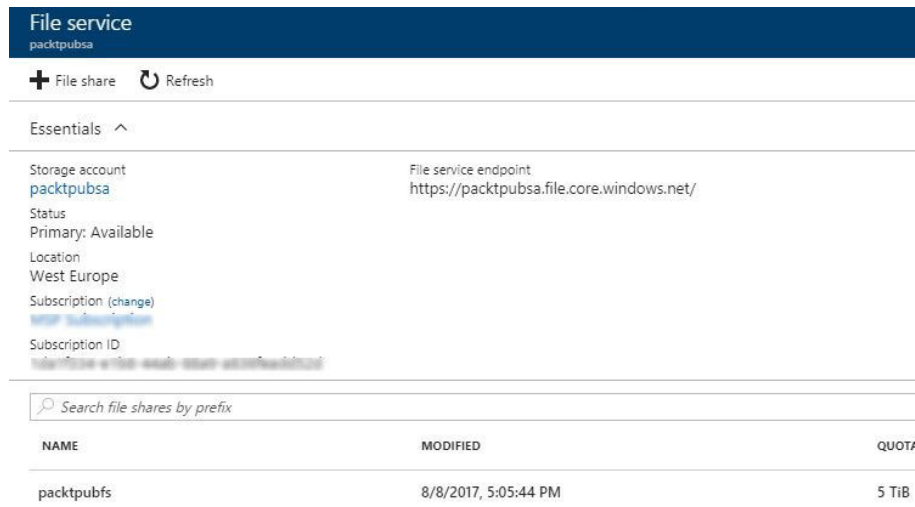


Figure 2.18: The created file share

5. You can map the file share to your Windows machine or Linux machine, adding directories within the file share, uploading data to it, and so on, if you opened it after creation, as shown in the following screenshot:

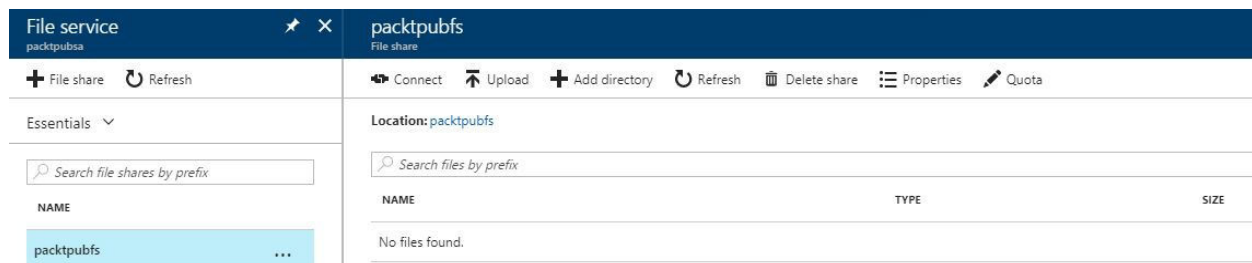


Figure 2.19: Overview of the created file share

6. To map the file share as a drive on your Windows machine or Linux machine, click on Connect, which will open a new blade, displaying the commands required to map it to your machine, as shown in the following screenshot:

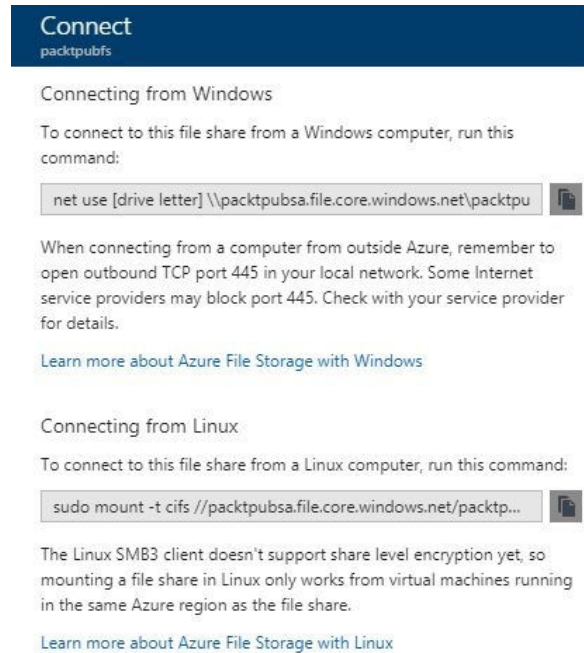


Figure 2.20: Connecting to your file share from your Windows or Linux Machine

7. To upload files to it, click on Upload and browse for the desired file, as shown in the following screenshot:

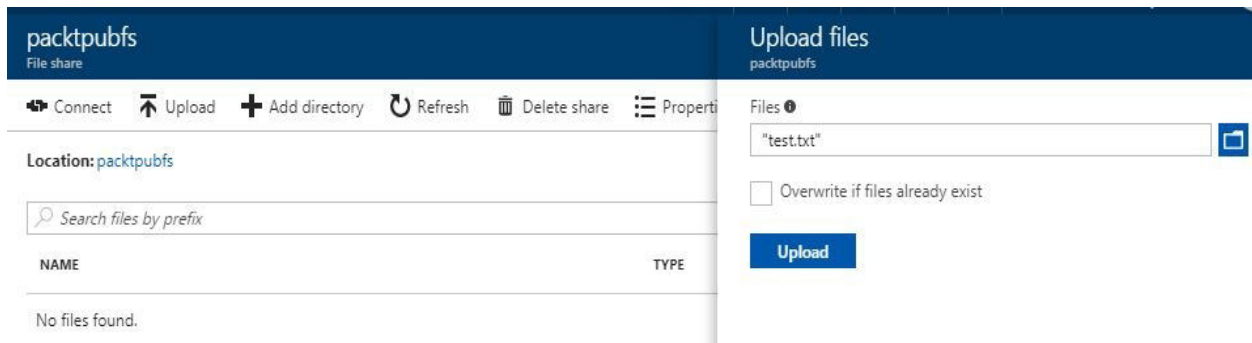


Figure 2.21: Uploading a file to the file share

File storage key points

The following tips should be considered, as they will help you when designing your storage solution using the File service:

- Since SMB 2.1 does not support encryption, then only the VMs within the same region as the storage account will be able to access it if you are using SMB 2.1. As a result, you have to consider that you cannot access the read-only data available in another region if you are using **Read-access geo-redundant storage (RA-GRS)** as a replication type.
- Since SMB 3.0 S supports encryption, you will be able to mount the file share to any VM around the globe, but port 445 must be opened.
- At the time of writing, only 2 versions of macOS were supported for Azure File shares (Sierra 10.12, and El Capitan 10.11).
- For better performance when working with Azure File shares on macOS, I recommend disabling SMB packet signing.
- The maximum size of a file share is 5 TB, considering that a file in the file share cannot exceed 1 TB.
- Every Azure File share supports up to 1000 IOPS, and 60 MB/s throughput.
- File share names can contain only lowercase letters, numbers, and hyphens, and must begin and end with a letter or number. The name cannot contain two consecutive hyphens.



*At the time of writing, Active Directory-based authentication and **Access Control Lists (ACLs)** are not supported. However, you can assign specific users to access specific file shares, but unfortunately, you cannot customize them anymore because every user has permission to a specific file share, which will be full access to the share.*

Understanding the Azure Storage architecture

Learning how to work with Azure Storage and how to design it to fit your solution is everyone's purpose, but learning what is going on behind the scenes and what every piece means is what makes you an expert.

Azure Storage is a distributed storage software stack built by Microsoft. The storage access architecture consists of the following three layers:

- Front-End layer
- Partition layer
- Stream layer

Front-End layer

The Front-End layer is responsible for receiving incoming requests, their authentication, and authorization, and then delivers them to a partition server in the Partition layer.

You may wonder, how does the frontend know which partition server to forward each request to? The answer is pretty easy, because the frontend caches a partition map.

And here, a new question will pop up, what is a partition map? It is responsible for keeping track of the partitions of the storage service being accessed, and which partition server controls access to each partition in the system.

Partition layer

The Partition layer is responsible for partitioning all the data objects in the system. Not only that, it is also responsible for assigning the partitions to partition servers, plus load balancing the partitions across partition servers to meet the traffic needs of the storage services, considering that a single partition server would handle multiple partitions.

Stream layer

The Stream layer or **Distributed and replicated File System (DFS)** layer is the layer responsible for storing bits on the disk and the data durability as it distributes and replicates data across many servers. All data stored in this layer is accessible from any partition server.

Sparse storage and TRIM in Azure

When you create a VHD on Azure to store your data on it, all the space you have chosen as a size for your VHD is completely allocated because Azure uses fixed-size VHDs. Therefore, you may wonder, will I really pay for the whole space even if I'm not using it, especially as it is not a dynamic disk but a fixed one.

Let's discuss this in more detail.

When you create a VHD, all the size is allocated, and that might trick you into using smaller VHDs to save costs, but actually that is not what really happens behind the scenes.

Azure uses sparse storage, which means no matter the size of the VHD you have created, you will only pay for what you have stored on the VHD. For example, you have a 1 TB VHD, but you have only 200 MB of storage stored on it. You will only pay for the 200 MB storage, therefore as a best practice, you should create the VHD with the maximum storage to avoid any downtime later during the resizing process.

Microsoft does its best to charge you only for what you use, that's why Azure Storage supports TRIM, which means whenever you delete data from it, you no longer pay for the deleted storage.



When you add a VHD, you should use quick format for the disk, not the full format. Doing so will write OS to the entire disk, which means you will have to pay for the entire disk. You should also consider not using defragmentation to avoid the movement of the disk blocks, which means greater costs will have to be paid. For further information about the Azure Storage architecture, you can download the following PDF file: <http://www.sigops.org/sosp/sosp11/current/2011-Cascais/11-calder-online.pdf>.

Securing Azure Storage

It's great to know how to manage Azure Storage, and even to follow best practices throughout the process. However, securing your storage should be your biggest concern, especially because storage is the base on which all your **Infrastructure as a service (IaaS)** services run.

Throughout this topic, we will cover the following methods to secure Azure Storage:

- **Role-Based Access Control (RBAC)**
- Access keys
- **Shared access signature (SAS)**

RBAC

Giving every user the exact permissions they need should be your first concern in order to avoid a complete disaster if a user's credentials were exposed.

RBAC would help you with segregating duties within your team; specifically, everyone would only be granted the required permissions to get their job done.

RBAC role assignments would be granted based on:

- Subscription
- Resource group
- Resource

For example, RBAC can be used to grant permissions for a user to manage the virtual machines within a subscription, or to grant permissions for a user to manage a complete resource group that contains virtual machines, **network interfaces (NICs)**, storage accounts, availability sets, and so on, or granting permissions for a user to manage a specific resource such as a specific virtual machine. This does not deny the fact that the same user can be granted permissions to another resource, resource group, or even a subscription.

Granting the reader role to a user using RBAC

Throughout this topic, we will cover how to grant a user read permissions on our previously created storage account, packtpubsa. So, without further ado, let's get started.

1. First, you must have a user in the Azure Active Directory. If not, you can learn how via the following link: <https://docs.microsoft.com/en-us/azure/active-directory/active-directory-users-create-azure-portal>.
2. Open the Azure portal and navigate to Storage account, then select the storage account you want the user to have read permissions on, as shown in the following screenshot:

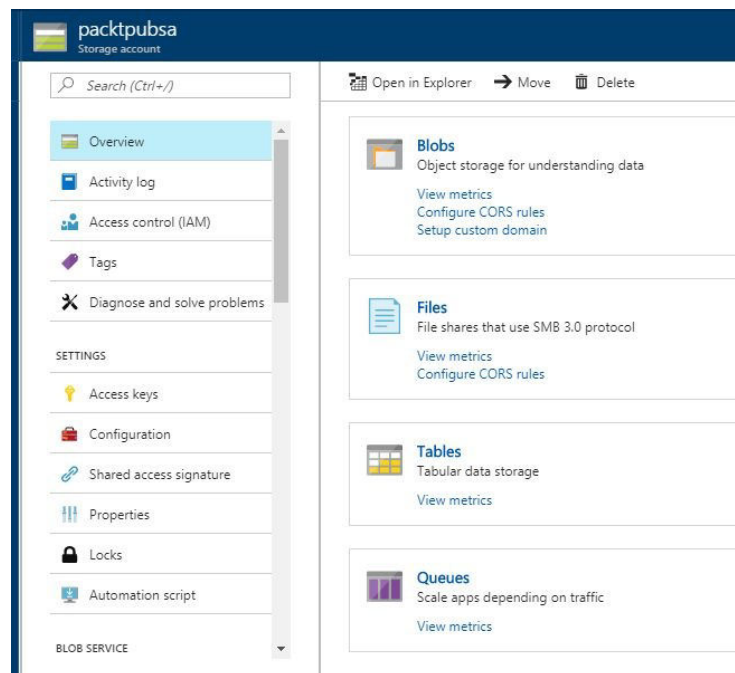


Figure 2.22: Azure Storage account on which we will grant a user read permissions

3. Navigate to Access control (IAM) and click on Add, as shown in the following screenshot:

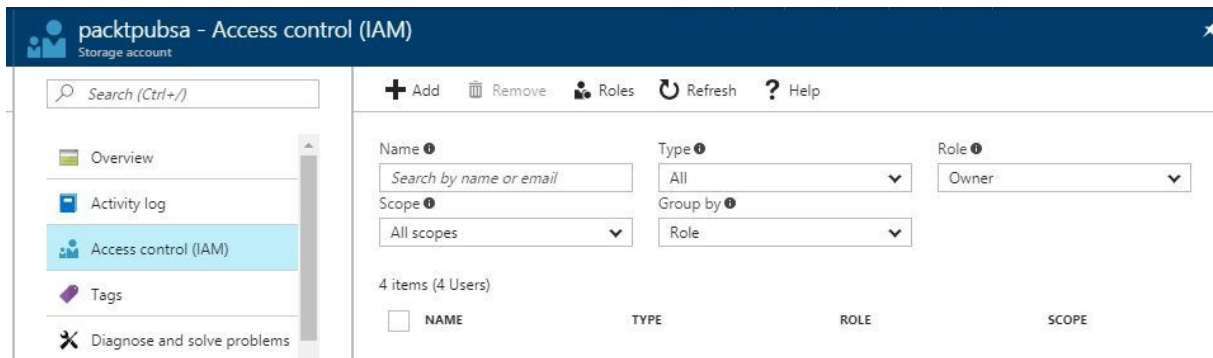


Figure 2.23: Access control blade

4. Select the Role, which in our case is Reader role, and select the user you are willing to grant this role to, as shown in the following screenshot:

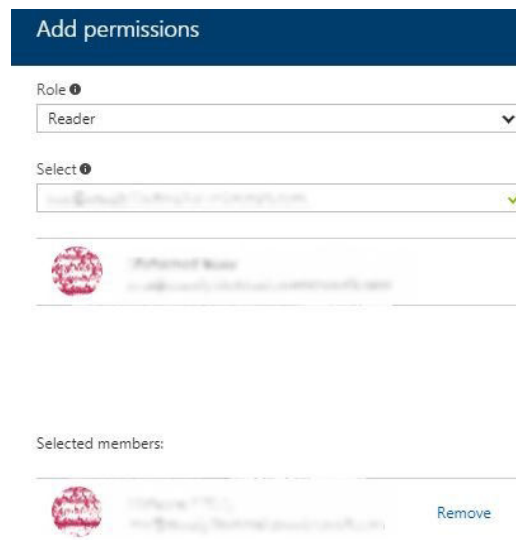


Figure 2.24: Selecting the role and the user to which the role is being granted

5. Once done, you will see it under **READER** role in Access control (IAM) blade, as shown in the following screenshot:

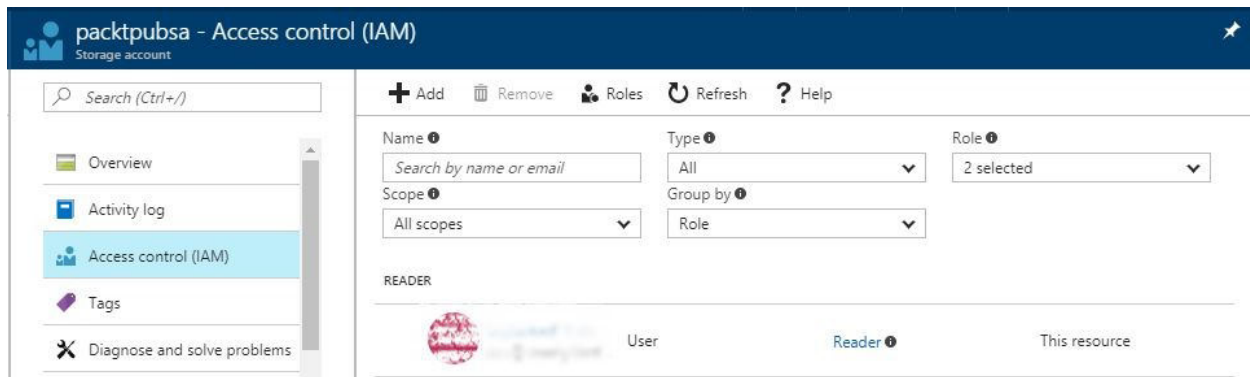


Figure 2.25: The users with reader role access to packtpubsa storage account

To learn more about RBAC, you can check out the following links:



- **RBAC built-in roles:** <https://docs.microsoft.com/en-us/azure/active-directory/role-based-access-built-in-roles>
- **Custom roles in Azure RBAC:** <https://docs.microsoft.com/en-us/azure/active-directory/role-based-access-control-custom-roles>.

Access keys

Storage account access keys are 512-bit strings, which are generated once you create a new storage account, and get paired with it. These keys are for authenticating storage services whenever you try to access them.

Fortunately, Azure provides two access keys. So, if the primary key is compromised, you can regenerate the key, and use the secondary key in the meantime.

To regenerate access keys, you have to navigate to the storage account, then navigate to Access keys under SETTINGS, as shown in the following screenshot:

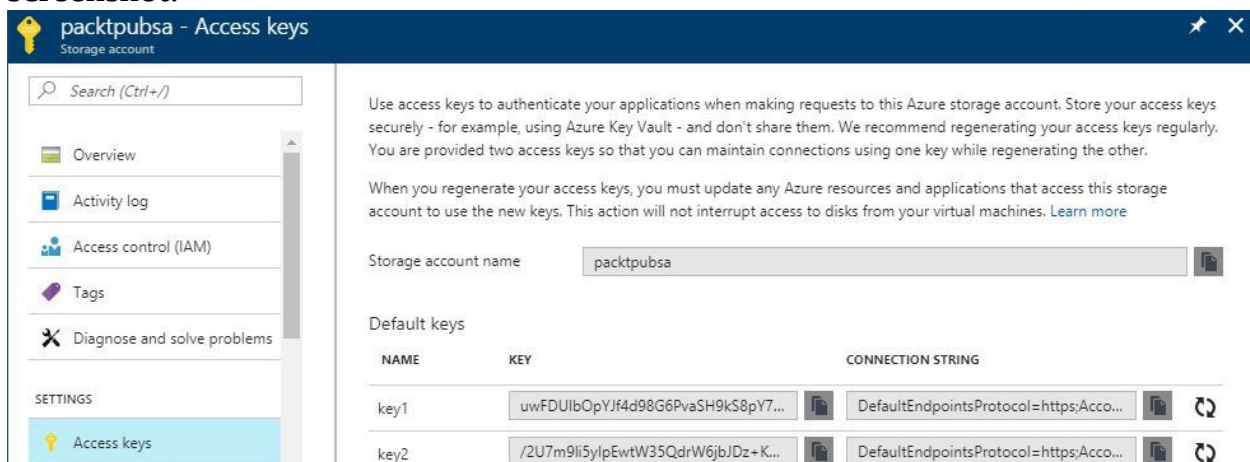


Figure 2.26: Storage account access keys

To regenerate the keys, you have to click on the regenerate icon, as shown in the following screenshot:

Regenerate access key

The current key will become immediately invalid and is not recoverable. Do you want to regenerate access key 'key1'?

2

Yes

No

Default keys

NAME	KEY	CONNECTION STRING
key1	uwFDUIbOpYJf4d98G6PvaSH9kS8pY7...	DefaultEndpointsProtocol=https;Acco...
key2	/2U7m9li5ylpEwtW35QdrW6jbJDz+K...	DefaultEndpointsProtocol=https;Acco...

Figure 2.27: Regenerating the primary key



Whenever you regenerate the access key, you have to update all the clients who were using the old access key to access the storage account to avoid any disruption with your storage services that are based on the storage account for which the access keys were changed.