

# Microsoft Azure IN ACTION

Lars Klint



MANNING

# Microsoft Azure IN ACTION

Lars Klint



# **Microsoft Azure in Action MEAP V06**

1. [Copyright 2023 Manning Publications](#)
2. [welcome](#)
3. [1 What is Microsoft Azure?](#)
4. [2 Using Azure: Azure Functions and Image Processing](#)
5. [3 Using Virtual Machines](#)
6. [4 Networking in Azure](#)
7. [5 Storage](#)
8. [6 Security](#)
9. [7 Serverless](#)
10. [8 Optimizing Storage](#)

MEAP Edition

Manning Early Access Program

Microsoft Azure in Action

Version 6

# Copyright 2023 Manning Publications

©Manning Publications Co. We welcome reader comments about anything in the manuscript - other than typos and other simple mistakes.

These will be cleaned up during production of the book by copyeditors and proofreaders.

<https://livebook.manning.com/book/microsoft-azure-in-action/discussion>

For more information on this and other Manning titles go to

[manning.com](https://manning.com)

# welcome

You are pretty special, you know that? You are one of the very first amazing people to have a peek at my upcoming book, *Microsoft Azure in Action*. This book isn't like any other tech book. At least not entirely. It is a journey into cloud computing where *you* are at the center. I am here to convey over 10 years of Microsoft Azure experience to you in a way that is free of jargon and fluff.

I first started using cloud computing to explore a way to offload inconsistent workloads to a service that could scale up and down at a moment's notice. Since then, the services on Azure have grown exponentially to include machine learning, data analytics, big data, large security application and so much more. However, at the core is still compute, networking and storage. It is with explaining that foundation I start the book to make sure you also get off on your cloud journey with the right tools from the start.

The book has four parts. Part 1 of the book introduces Azure to your world, what are the benefits and advantages of cloud computing, as well as going straight into building a photo resizing app with serverless. Yes, really. Part 2 is all about those fundamentals, and getting them nailed down, as the rest of the book builds on them. Part 3 is managing data in all the various ways that Azure offers (there are many flavors). Finally, part 4 is about DevOps, security, and performance.

It will help if you have some understanding of scripting and coding principles, but you don't need any cloud computing knowledge. I wanted to write a book that is both engaging and educational, caters for those that are new to cloud, but also offers little details and tidbits to sweeten the taste for those that might know a thing or two already.

Because your feedback is essential to creating the best book possible, I hope you'll be leaving comments in the [liveBook Discussion forum](#). After all, I may already know how to do all this stuff, but I need to know if my explanations are working for you! In particular, I want to know if you are

enjoying yourself as you read through the chapters.

With cookies and hats.

- Lars Klint

**In this book**

[Copyright 2023 Manning Publications](#) [welcome](#) [brief contents](#) [1 What is Microsoft Azure?](#) [2 Using Azure: Azure Functions and Image Processing](#) [3 Using Virtual Machines](#) [4 Networking in Azure](#) [5 Storage](#) [6 Security](#) [7 Serverless](#) [8 Optimizing Storage](#)

# 1 What is Microsoft Azure?

## This chapter covers

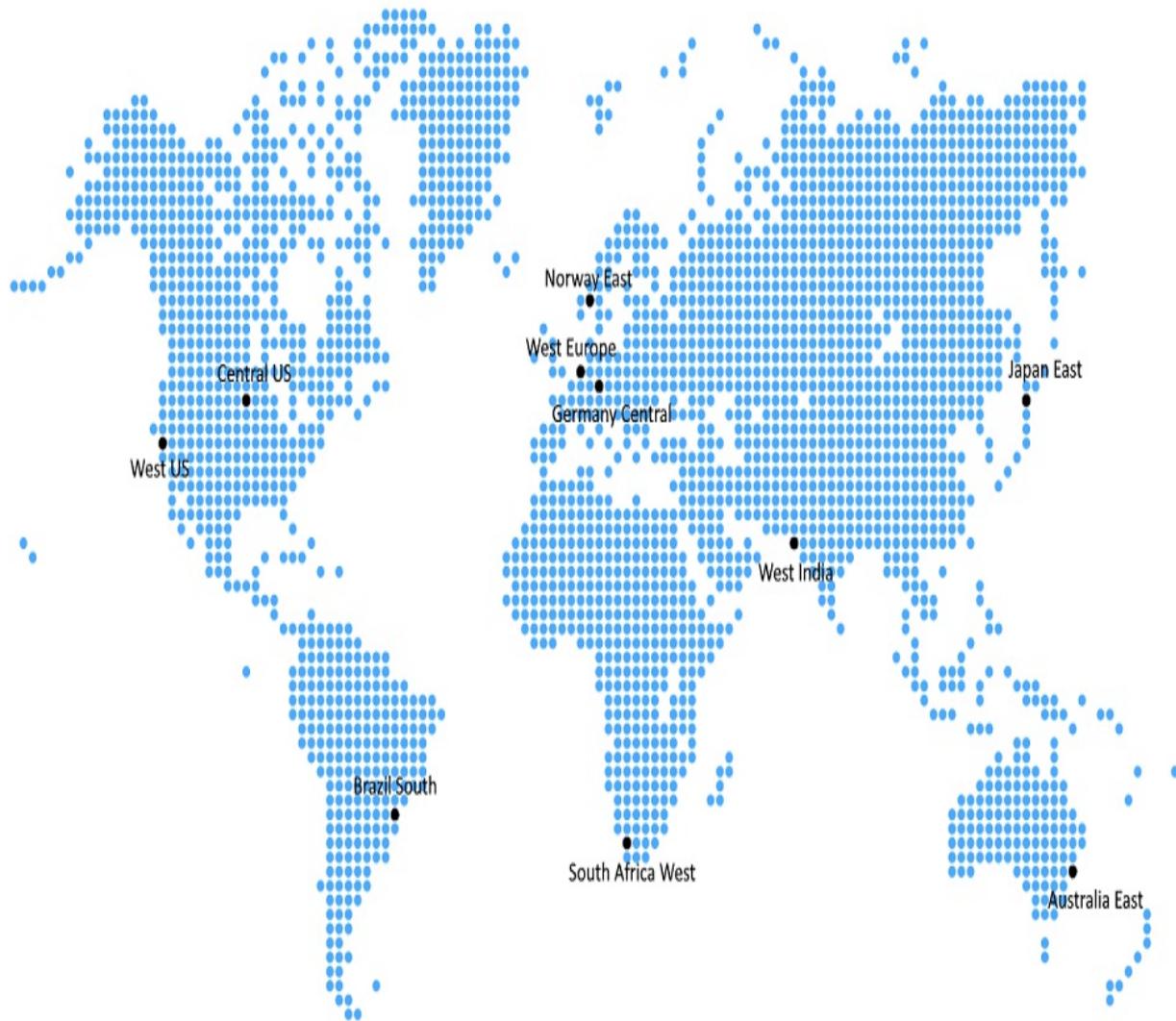
- Overview of Microsoft Azure
- The benefits of using cloud computing and Azure
- What you can and can't do with Azure
- How to interact with Azure services
- Creating a free Azure account

Azure is a web-based platform consisting of hundreds of different services that allows anyone, with any budget, to jump straight into creating and publishing internet services and applications. Cloud services for computing, networking and storage are readily available, as well as hundreds of services that are built on top. These services can be used for simple tasks such as hosting a web site or storing files, as well as incredibly complex tasks analyzing vast amounts of data to send rockets into space and solve how photosynthesis works. Everything on Azure is accessed via the Internet using standard tools and protocols. Azure Virtual Machines running Windows Server or Linux, and Azure Storage are two of the most used services on Azure, and like most services on Azure they integrate with other Azure services with relative ease.

Azure is available globally, which means Microsoft has built data centers in many regions around the world. When you create an application, it doesn't matter if you create it in Australia or Norway<sup>[1]</sup>; It is the same approach and commands you use. This makes it very simple to create products that are close to your customers, but that scale globally.

At the time of writing this book, Azure has more than 65 regions each containing 1-4 data centers for total of over 160 data centers. And it is growing all the time. Some regions are restricted to government bodies and their contractors, while operating in China has very special conditions. Figure 1.1 shows some of the current Azure regions available.

**Figure 1.1: A selection of some of the many Azure Regions spanning the globe.**

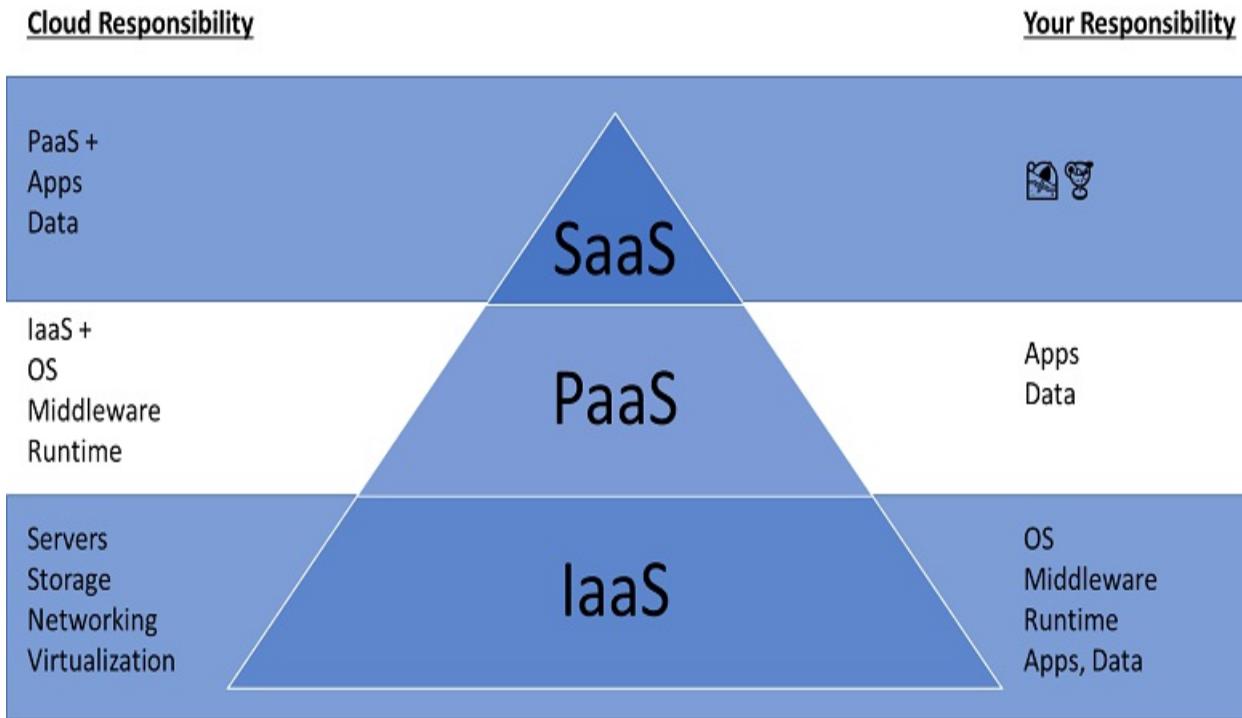


## 1.1 What is Cloud Computing

As you might have guessed, cloud computing does not actually include any clouds. For the most parts *cloud computing* as a term is used to describe using computing services on someone else's server. More specifically, using computing services on one of the major cloud computing platforms, such as Azure or Amazon Web Services (AWS), or even on a *private cloud*.

The general term *cloud computing* is describing an abstraction of computing resources. When you want to use a service on a cloud computing platform, various layers of technology and implementation are abstracted away from you. In general terms, services are divided in Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). Each of the three types of as-a-Services abstracts some portion of the underlying technology and infrastructure. IaaS abstracts the hardware layer and uses *virtualization* to provide computing, networking and storage access to the cloud computing user, you. PaaS further abstracts the operating system, several maintenance tasks, software updates, patches, and more to provide a development platform service. Finally, SaaS provides a “pay and play” approach to using software that is hosted in the cloud. Everything, except using the software, is abstracted away from the end user. These many layers of abstraction are critical to cloud computing, and provides the real value to developers, system administrators and other professionals creating sophisticated and complex solutions. They only need to focus on the parts that make their business case stronger, and not be tied up with what is often known as “yak shaving”<sup>[2]</sup>. Figure 1.2 shows the relationship between IaaS, PaaS and SaaS.

**Figure 1.2: Cloud computing abstraction layers.**



Once you know what services you need, how much data to store and how your users are going to access your service, it is all available to you in what seems like infinite amounts. You request a service or product, and it is served up for you in minutes or even seconds. Need another virtual machine? No problem. Need another 100GB of SQL Server storage? Certainly, ma'am. You only pay for what you use, and everything is available at your fingertips.

To place Azure in the context of other cloud platforms, there are three main types of clouds:

- **Public** – Anyone can set up an account and use the services offered on a public cloud. The common offerings are Azure, AWS and Google Cloud Platform (GCP), among others.
- **Private** – A company can create a private cloud, which is accessible only to that company. They host their own hardware and abstraction layers.
- **Hybrid** – Many companies, and especially government departments, are not ready to move to the cloud 100% and they will have some services on-premises and some in the cloud. This is called a hybrid cloud approach and mixes public and private clouds.

Azure is a public cloud offering, providing IaaS, PaaS and some SaaS

products, although you can also use Azure as part of a hybrid cloud setup.

## 1.2 Azure in the Real World

When you combine the right services, configure the applications just so and deploy code and templates with success, Azure can do almost anything. You can deploy applications securely, automate releases of new versions, get notified of potential security breaches, store incredibly large amounts of data, connect networks in different parts of the globe and so much more. Azure won't do your laundry though. Yet.

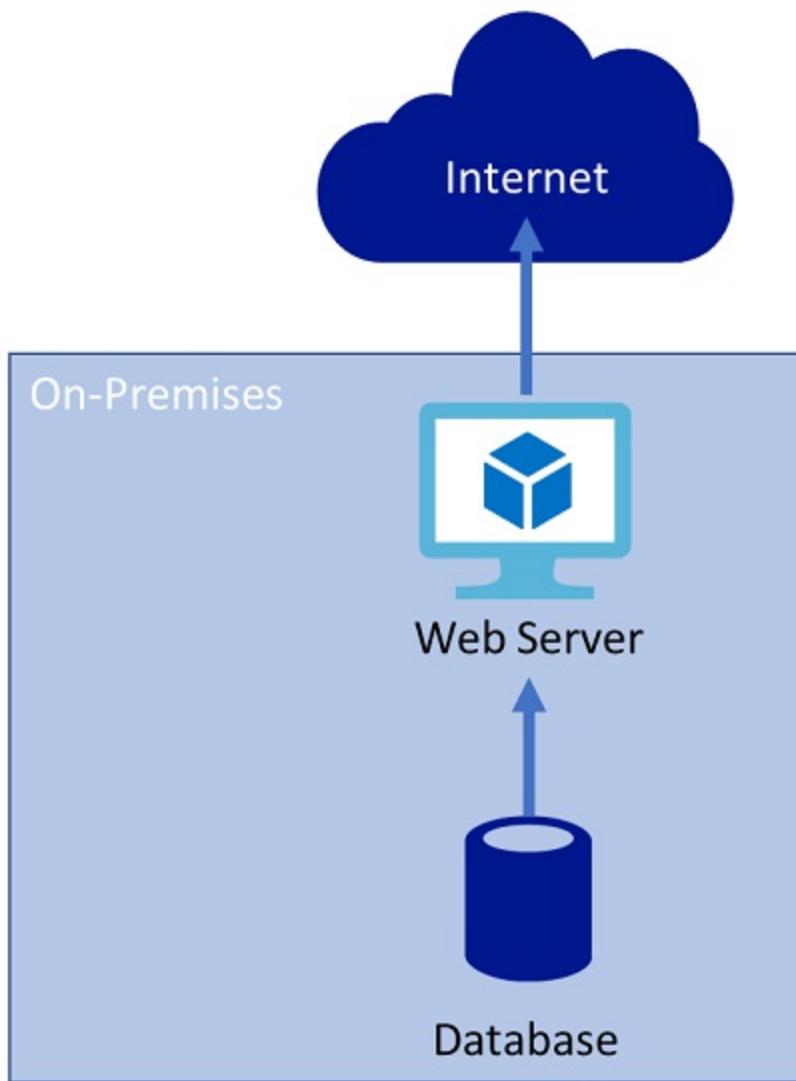
Let me show you some examples of using Azure that makes a lot of sense in the real world.

### 1.2.1 Hosting a Web Application

One of the most common tasks is to host a web application. This could be a niche site selling scented candles, or a popular high-traffic technology news site.

Mr. Wayne has a small website that includes an online shop selling various items, mostly black in color. The current setup for the website is hosted on premises by Mr. Wayne, as shown in Figure 1.3.

**Figure 1.3:** Simple on-premises web hosting.



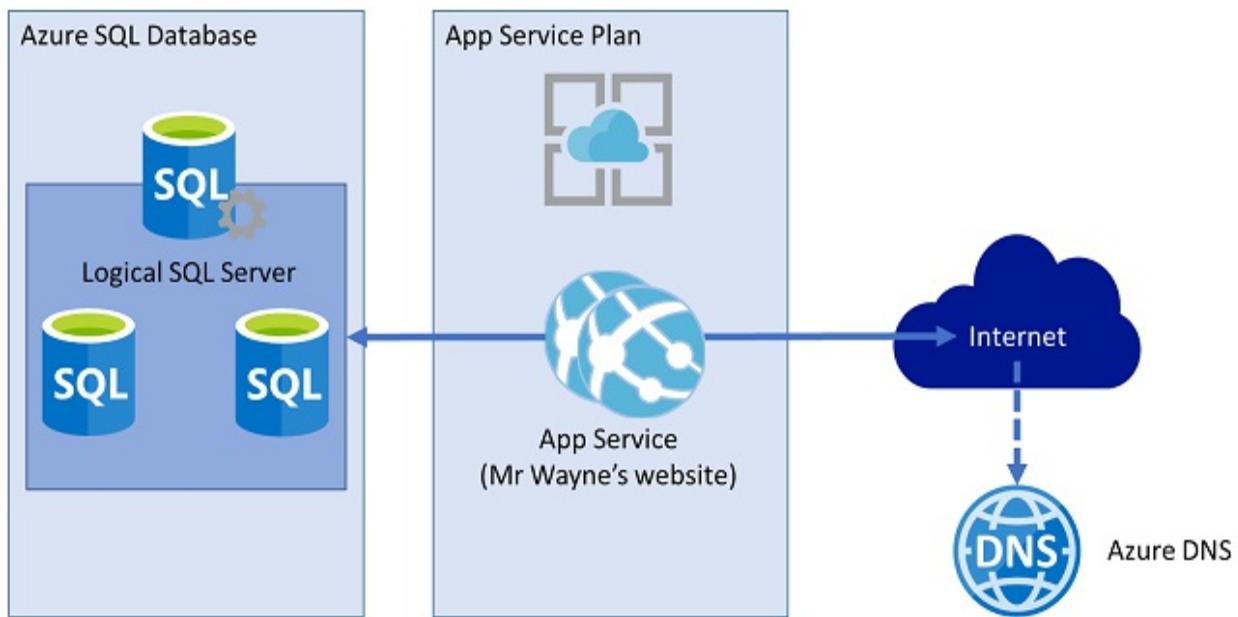
Business and traffic are booming, and Mr. Wayne needs to move the website to the cloud, which can solve multiple current issues with the site.

- Users visiting from far away destinations experience significant delays in loading the site, and it is also slow to respond.
- When there is a sale on the site, the traffic can become too much and the site crashes. That makes Mr Wayne sad.
- Any maintenance needed to fix the hardware, connection, site or database currently has to be done by Mr. Wayne or a hired consultant. This takes up a lot of time.

- The update of the website code is fragile, and backups are made infrequently.

An example of how Mr Wayne can improve his business and web shop using Azure services is shown in Figure 1.4.

**Figure 1.4: Mr. Wayne's cloud architecture.**



This will partly solve the latency issue using Azure DNS (by resolving the location of your site faster), peaks in demand can be managed by scaling the App Service Plan horizontally (creating more computing instances), there is no maintenance of hardware, and regular backups are made of the SQL databases and App Service.

Using Azure, all of the above issues have been addressed and B. Wayne is saving costs of owning hardware, managing internet connections and more, on top. As his business grows, scaling the cloud architecture with it is vastly easier. You will learn more about optimizing performance in Azure later in the book as well.

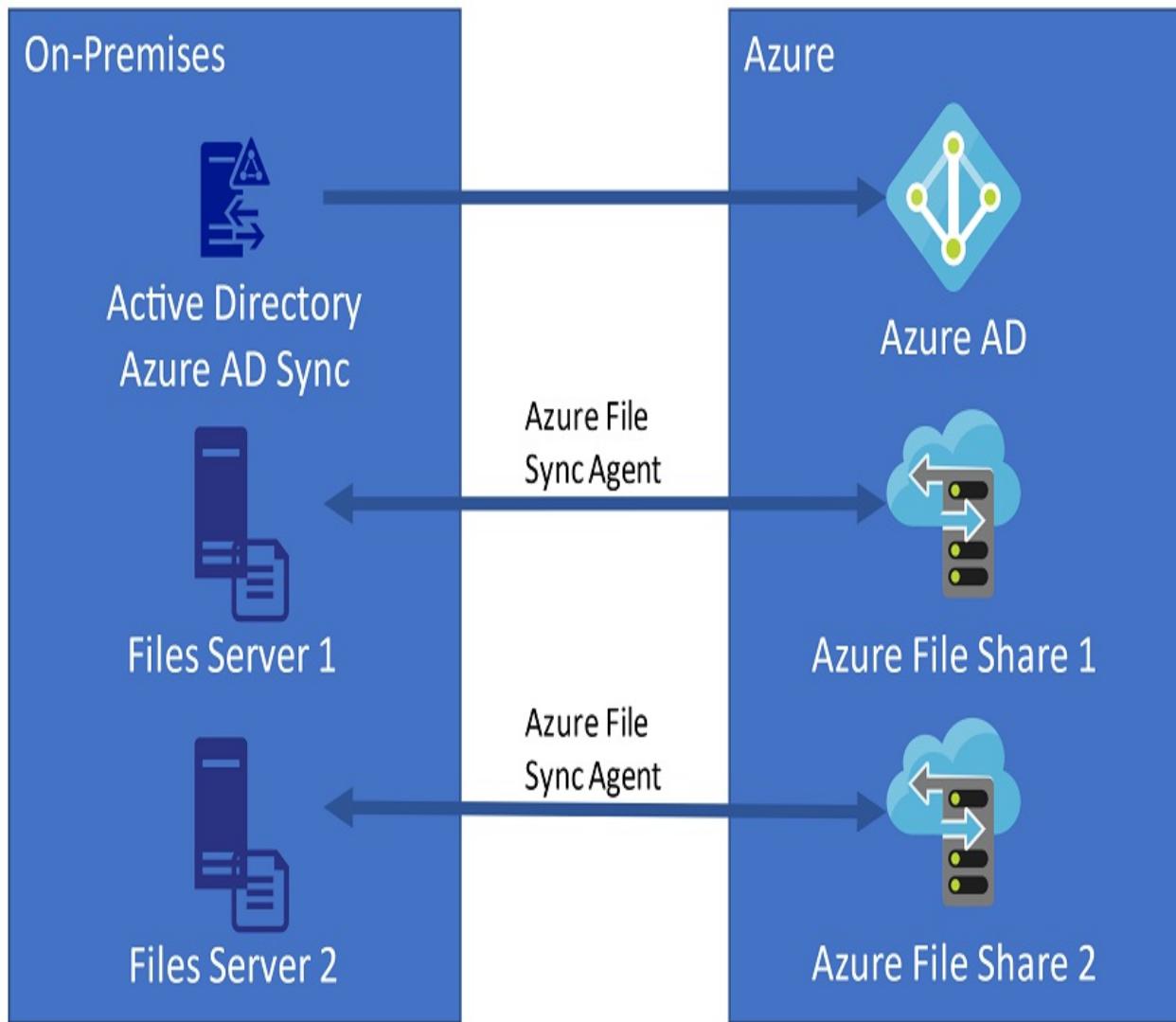
### 1.2.2 Running a Hybrid Network

Natasha is managing the IT infrastructure for a medium sized business. The

file storage requirements for the business are starting to outpace the hardware and network capabilities on premises. She wants to take advantage of Azure and have both file storage and file backup in the cloud.

She configures an Azure Active Directory Connect sync service on premises, which allows users and processes to authenticate with both the on-premises domain controller and the Azure Active Directory tenant. An Azure tenant is a dedicated and trusted Azure Active Directory's instance, and the first AAD instance created for a new Azure account. She then creates the necessary Azure File Shares in an Azure Storage account and configures the required backup routines which are then handled by Azure. Natasha then installs Azure File Sync agents on the on-premises servers that hosts the files needed to be synchronized to Azure. The Azure File Sync agents will keep both on-premises and Azure File Shares in sync at all times. Figure 1.5 illustrates Natasha's architecture.

**Figure 1.5: Hybrid file synchronization setup.**



This removes a lot of stress from Natasha's world, and she is now ensured regular backups of the company data. The security is handled implicitly by Azure, through the Azure AD Sync process, only allowing authenticated and authorized users to access the company files. As an added bonus, files can be accessed both locally and in the cloud, as they are always in sync. This is a common hybrid setup that takes advantage of the best of on-premises and Azure.

As a result, Natasha has more time to focus on other projects. She is saving part of her company budget, as she has avoided having to buy extra hardware

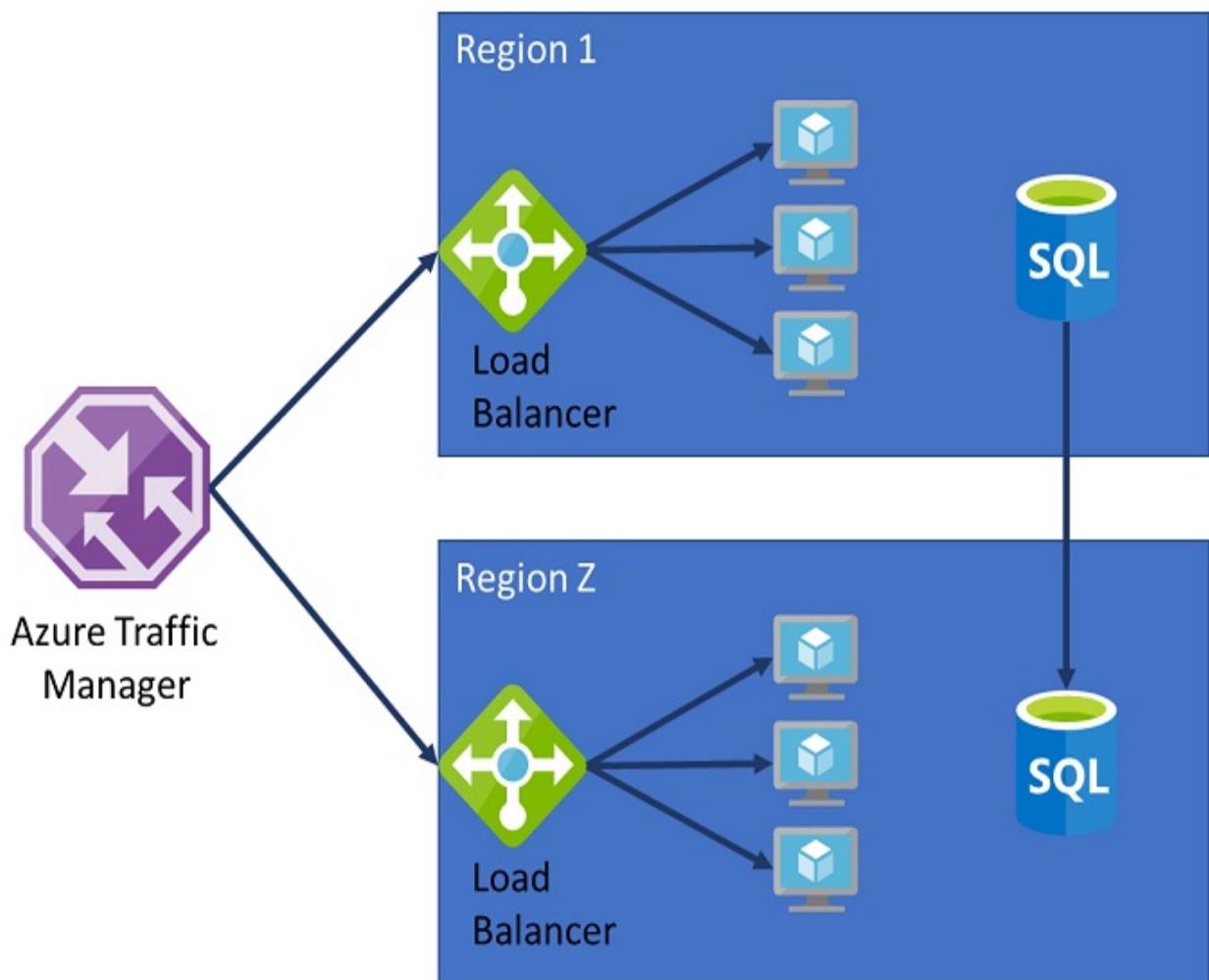
and instead only pay for the storage as it grows in size. She can now say with confidence that company intellectual property is backed up and secured.

### 1.2.3 Mitigating Risks and Outages

Clark is a principal engineer and works for a rapidly expanding start-up. He has been tasked with mitigating any risks for the company infrastructure hosted on Azure. For a first phase, he will have to consider the load on the system, any single points of failure, security and appropriate alerts.

Clark decides to design an architecture that has redundancy as one of its primary building blocks. Using Azure services, this can look like Figure 1.6.

Figure 1.6: Redundant Azure architecture example.



While some services have built-in redundancy, such as Azure Storage, most Azure services are designed to fit into a redundant architecture. Clark's solution uses elements that you will learn much more about later in the book too.

- *Azure Traffic Manager*: If the web front end becomes unreachable in region 1, Traffic Manager will route new requests to region Z.
- *Load Balancer*: If any of the virtual machines hosting the front-end application stop responding, the load balancer distributes traffic to the remaining healthy VMs.
- Use multiple VMs to ensure there is always a machine serving the application to the end user.
- Use *Azure SQL* with replication between the instances to ensure data is up to date, no matter which region and VM is serving the request.

With the solution in hand, Clark can now fly off like a bird. Or a plane.

#### 1.2.4 Automating infrastructure with ARM templates

Carol is a system administrator for a company that is going through a transformation phase to streamline their deployment processes with automation. Part of the workflow requires virtual machines to be created and destroyed very frequently, in order to test performance and scalability of various applications. Currently, Carol and her team does this more or less manually, which is boring, easy to get wrong and slow. Using automation, Carol reckons they can be quick, accurate and have much more time for office Skee-Ball. Carol is right.

She starts using Azure Resource Manager (ARM) templates, which describes resources with JSON syntax.

```
{  
    "$schema": "https://schema.management.azure.com/schemas/2015-  
    "contentVersion": "1.0.0.0",  
    "parameters": {  
        "tagValues": {  
            "type": "object",  
            "defaultValue": {  
                "Environment": "<Prod>",  
                "Region": "North Europe",  
                "Subscription": "My Company"  
            }  
        }  
    }  
}
```

```

        "Projectid": "<1234>",
        "Projectname": "<some name>",
        "Subcontractor": "<some vendor>",
        "Routineid": "<some number>",
        "Routinename": "<some name>",
        "Applicationname": "<some application name>"
    }
},
"GalleryImageSKU": {
    "type": "string",
    "metadata": {
        "description": "Image SKU."
    },
    "defaultValue": "2016-Datacenter"
},
"GalleryImagePublisher": {
    "type": "string",
    "metadata": {
        "description": "."
    },
    "defaultValue": "MicrosoftWindowsServer"
},
...
... Much more

```

Using ARM templates as above, Carol can fine tune the deployment of the VMs, which not only achieves her goals of less errors, speed and more time free for other things, but also saves money on running unnecessary VMs, lets her use source control to keep track of version history and integrate the VM creation workflow with other functions in the business. Win.

## 1.3 Why move to cloud?

Usually, the first reason many companies starts to investigate a cloud infrastructure on Azure is cost saving. And while you can definitely cut a lot of expenses up front by using Azure, such as hardware and networking costs, there are benefits that have a much greater significance in the long run. This section covers some of the main benefits Azure provides, and at the same time sets up some of the topics you will learn about later in the book.

### 1.3.1 Scalability

Imagine you have a website that runs on your on-premises web server. The

hardware is designed to handle a specific maximum capacity of traffic to the website. This capacity can be limited by the CPU power, the amount of RAM, how fast a network connection you have and much else. When the server is maxed out, how do you ensure users of your Nicholas Cage meme generator service can still use it? How can you manage seasonal patterns in your traffic?

Scalability is the answer. There is a lot more on scaling your Azure infrastructure later in the book, but for now let's look at the concept of scalability. While it is possible to ensure scalability on-premises, it can be an increasingly costly affair as you buy more hardware and install more systems to manage the new hardware. Not to mention, most of the time all that grunt will sit idling, waiting for the next peak. Azure provides scalability as a core principle of the platform. *Auto-scaling using Virtual Machine Scale Sets* (Figure 1.7), as well as for *App Services* is a trivial process on Azure. You only have to estimate average and peak workloads, then define thresholds for when the scaling should occur and how many machine instances you want to have as a minimum and maximum.

**Figure 1.7: Azure Virtual Machine Scale Set – Autoscaling Settings**

#### Choose how to scale your resource



#### Custom autoscale

Autoscale setting name	ScaleSetAction-Autoscale-661
Resource group	AzureAction
Instance count	2

**Default\*** Auto created scale condition [Edit](#) [Delete](#)

Delete warning The very last or default recurrence rule cannot be deleted. Instead, you can disable autoscale to turn off autoscale.

Scale mode  Scale based on a metric  Scale to a specific instance count

Rules No metric rules defined; click [Add a rule](#) to scale out and scale in your instances based on rules. For example: 'Add a rule that increases instance count by 1 when CPU percentage is above 70%'.

[+ Add a rule](#)

Instance limits [Minimum](#) [Maximum](#) [Default](#)

1	✓
5	✓
1	✓

Schedule This scale condition is executed when none of the other scale condition(s) match

Where on-premises infrastructure makes scalability expensive and difficult to manage, Azure makes it cost effective and trivial. Later in the book you will learn about scale sets in particular, but also how scaling of Functions, databases and traffic works. Stay tuned.

### 1.3.2 Reliability

Nothing can erode trust in your online services as much as reliability issues. If your services are not responding, run out of storage space or lose network connection, your users' trust in your service will disappear faster than ice cream on a hot summer day. When you use Azure to host your services, reliability is part of the package you pay for. In this book you will learn how to take full advantage of the cloud reliability using VMs, virtual networks, caching data to speed up your applications, drilling into service logs and so much more.

Azure services are at their core built for reliability. Azure regions are paired to ensure business continuity and disaster recovery by being able to switch from one to the other at short notice. Services are fault tolerant by using availability zones for hardware failures, power outages and any other gremlins that might rear their ugly heads. Azure Storage is resilient by design and is at a minimum replicated three times within each region, even though you just have one Storage account, but much more about storage and availability zones later in the book. In a few chapters' time you will also learn about Cosmos DB and its geographic replication and reliability. That is one of my favorite services.

### **1.3.3 Expanding Infrastructure**

Rarely will an online service remain stagnant. Usually, they either grow or die. When they do grow, it can be at an incredible speed that requires rapid expansion of the IT infrastructure. Azure makes this much easier by solving many common problems for expanding your computing and traffic capacity.

Tasks such as load balancing of traffic, replicating files, connecting a SQL Server instance to Active Directory, creating Kubernetes clusters and much more are done with only a click of a button. Managing your expanding infrastructure is dealing with localities, configuration and customer expectation, rather than plugging in more cables, buying hard drives, installing software and patching servers.

Azure is expanding all the time and as the cloud platform grows, gets more users and streamlines products and services, you will benefit from economies of scale. Azure computing services, such as Virtual Machines, keep getting

cheaper, and it is the same with storage. This makes keeping up with your product growth much more manageable.

Other services that support expanding infrastructure natively on Azure include SQL Server, load balancing and Azure file storage. Those are explained much more in due time later in the book.

### **1.3.4 Support**

If a service on Azure isn't performing, there is a bug in a feature, or you need help with setting up a service, there are multiple ways you can get support.

- Lodge a support ticket with Azure Support. Depending on your subscription type, you can get more in-depth support, faster.
- Use the Azure documentation<sup>[3]</sup> to troubleshoot and learn about the services you need.
- Engage with the millions of users of Azure through community forums, such as the MSDN forum<sup>[4]</sup>.
- Use this book to guide you on your journey to efficient and sustainable Azure usage.

Timely and well articulated support can often make the difference between success and mediocrity. If you are stuck on a problem and waiting for help, you want that support to be both quick and accurate. This book can form part of the support network for your project by teaching you the best practices for both individual services and complex architecture.

### **1.3.5 Compliance**

Azure has 90<sup>[5]</sup> compliance certifications that cover a broad range of countries, regions, industries and government regulations. For many industries this makes a huge difference, as Azure has already done the hard work for you.

The US Federal Risk and Authorization Management Program (FedRAMP) is a standard approach for assessing security in cloud computing platforms and is mandated to use in the US. Azure has this certification, which makes it

much simpler for US government bodies to validate the security of Azure cloud services.

To help financial institutions in the EU follow the European Banking Authority (EBA) recommendations for cloud adoption, Azure complies with the EBA guidelines. This ensures that banks using Azure in the EU comply with audit, reporting, data residency and other specific requirements.

Because Azure complies with this wealth of guidelines and certifications, industries can focus on their core business cases instead of red tape. This makes it extremely attractive for many businesses and removes doubt for many others, whether they should invest in cloud computing with Azure.

## 1.4 Costs

Costs in Azure, and cloud computing in general has a rumor of “blowing costs out without you knowing”. You might have heard stories of Virtual Machines racking up thousands of dollars in cost overnight, or some service left running that ruins the company budget. Yes, it happens. No, it is not common.

Throughout this book costs and pricing will be treated as a “fourth pillar” of cloud computing, with compute, network and storage being the first three. While you won’t learn about every single price and pricing tier, because watching paint dry is more interesting, you will learn how to keep costs down as we go through the wonderful world of cloud computing with Azure.

Cost saving is a definite plus for using Azure, so let’s go over some of the ways Azure can be very cost effective and benefit you, your company and the evil overloads.

### 1.4.1 Free account

Ahhh, free stuff. We all love some freebies, and Azure also offers a free account to get started.

- You get USD200 to spend on any Azure service you want, such as

Azure Kubernetes Services (AKS) or Azure Cognitive Services.

- There are services that are free for twelve months, such as a 250GB instance of SQL Server or 750 hours of use on certain Virtual Machines instances.
- And then there are services that are always free, which include Cosmos DB free tier, 1 million requests every month for Azure Functions and 50 Virtual Networks.

With a free account you have access to enough services and resources that you can build a prototype, try out an unfamiliar service or try out some of the examples in this book. Later in this chapter you will learn how to create a free Azure account.

### **1.4.2 PAYG**

Pay as You Go (PAYG) is the most common, and most expensive, billing model. As the name implies you pay for what you use, whether that is 24 minutes on a Virtual Machine, 300,000 RU/s on Cosmos DB or another App Service Plan.

PAYG usage is billed monthly, and while it embodies the idea of cloud computing and only paying for what you use, it is also the most expensive way, when looking at cost per unit. For a lot of Azure services, PAYG is the way you are billed for use, but some services have ways to make them cheaper. Much cheaper. Read on.

### **1.4.3 Reserved instances**

If you know that you are going to be using a VM for a longer period of time, Azure offers *reserved instances*. As is often the case, you know a VM is going to be used consistently, such as for production purposes, and you can agree to use this VM for 1 or 3 years. You can reserve the instance, and by committing Azure will give you a massive discount of up to 80%.

It's like renting a car vs. leasing a car. If you rent a car, you pay a higher daily price, but you only have it for a few days. The rental company needs to set the price higher, as they might not know when the car will be rented

again, they have to clean it, and staff needs to manage it all. On the other hand, if you lease a car, you commit to a much longer period of using the car, often several years. Your daily cost is greatly reduced, and the car company doesn't need to clean it, it doesn't sit idle, and less staff can manage it. The same logic applies to Azure reserved VM instances.

#### **1.4.4 Spot pricing**

At the complete opposite end of the “how-long-do-I-need-the-VM-for”-ometer is Spot VM pricing. Azure infrastructure has a lot of free capacity for compute power at any given moment. This is to make sure your service can scale with traffic demand at a moment’s notice. While that compute power is sitting there ready to swing into action, Azure will let you use it at up to 90% off the regular VM price. “What is the catch?”, you might ask? At any point, Azure can evict you from using the VM if the cloud platform needs the resources. Evicted!

Spot VMs are excellent for interruptible processes, for testing and for development. Azure will even tell you how likely you are to be evicted at what times, to let you plan your use even better. Oh yeah, don’t use them for any production purposes whatsoever. It’ll end in tears.

#### **1.4.5 Billing**

Billing of Azure services come in three delicious flavors:

- Time: A VM is billed by the minute and a Container Instance is billed by the second.
- Traffic: Data sent from Azure to the Internet is billed in GB transmitted (this is often called Internet Egress); Azure Functions is billed in number of executions.
- Storage: Azure Files and Azure Data Lake Storage are billed per GB stored.

How does all this work together to form your invoice then? That is almost a topic for a whole other book, as cloud computing invoices can be notoriously difficult to decipher. Throughout the book we will continue to make billing

and pricing part of the discussion, as your manager will want to know what it costs.

## 1.5 What are the alternatives?

“You should only ever use Azure for all your computing projects” ....is not the way. Of course, there are alternatives to Azure, and it is important to be aware of them to make an informed decision. If you are aware of the choices, the solution you arrive at is often better.

### 1.5.1 On premises

While not a topic for this book, having on-premises infrastructure is a solution that is right for many and has worked for decades. On-premises can provide a number of benefits due to the greater control you have of especially the physical premises and hardware.

- Sovereignty of data can be very specific. You know exactly where your data is stored, which is critical for legislation in some countries.
- Security of hardware and software can conform easier to certain company-specific requirements.
- Existing investment in infrastructure could warrant use of this, as the costs is low for new applications.

The move to cloud computing for more and more businesses means growth for on-premises infrastructure is slowing down, especially brand-new freshly smelling setups. Depending on where you get your statistics from, more than 90% of companies use a cloud platform in some way.

### 1.5.2 Other cloud providers

It may come as a surprise, but there are other cloud platform offerings! Yeah, I know! Providers such as AWS and GCP provide very similar products and services to Azure, at least at first glance. For a lot of greenfield projects, choosing an alternate cloud provider can make sense, but for companies with existing infrastructure, Azure is often a better fit.

This book doesn't attempt to compare Azure to other cloud providers, but neither does it pretend there aren't any. I hope you've read this far and you will join me to the end of the book, because you've decided Azure is the cloud for you. We have cookies....

### **1.5.3 Multi-cloud approach**

One of your marketing department's favorite terms is "multi-cloud". It makes their brochures look informed and cutting edge. However, when you dig through the glossy paper, there is a real solution underneath that does indeed use multi-cloud.

Multi-cloud most often means "two clouds". Every time a company decides to use an additional cloud, they effectively double their investment.

- You need to train engineers to use the new features, services, APIs and technology.
- You have to train architects in what the best practices are.
- Managers must learn how the billing side of things work.
- Quality assurance must understand the intricacies of services to test for implementation flaws.
- Security engineers need to understand vulnerabilities and weak points in the cloud platform.

Once a company decides to invest in a second cloud (or third... or fourth?), there are various approaches to the multi-cloud strategies.

- Use a single cloud for a single product, or single technology branch. Various products, or family of products, will all be using only a single cloud each, but that choice may vary.
- Major company infrastructure all use the same cloud for maintenance and security reasons. Additional services and applications can choose a cloud of choice.
- Projects choose the combination of cloud products that they prefer. This could be a single or multiple clouds.
- Any combination of any of them.

One of the issues with multi-cloud can be that there isn't a single entity inside

the organisation that is controlling what the strategy is and how to implement it. Various departments can sometimes decide independently which cloud to use for a project. Often this leads to a pile of cloud spaghetti that becomes increasingly difficult to unravel and consume.

This isn't to say there isn't value in multi-cloud when managed and implemented with care and thoughtful process. Many organisations successfully implement a multi-cloud strategy, and benefit from a broader selection of cloud services, cost benefits and innovation opportunities.

## 1.6 Interacting with Azure

Azure is only worth as much as what its users can get out of it. And you get the most out of Azure by using the tools that are right for you, and that you are comfortable with. There are a large number of Integrated Development Environments (IDEs), console integrations and language libraries that all work with Azure. And they all pass through the Azure Resource Manager, which is a single layer that ensures all tools interact in the same way with Azure. You will learn about some of this in greater details later in this book, but for now let me introduce you to a few of the popular tools of the Azure cloud.

### 1.6.1 Azure Portal

The very first stop for any Azure administrator, developer or professional in general, is the Azure Portal (Figure 1.8).

Figure 1.8: Azure Portal

The screenshot shows the Azure Portal interface. On the left is a dark sidebar with a navigation menu. The main area has a light background. At the top, there's a search bar and a user profile icon. Below the search bar is a row of icons for creating a resource, virtual machines, front doors, app services, cost management, management groups, Azure sentinel, all resources, resource groups, and more services. A section titled "Recent resources" lists items like "lasklint" (App Service), "staging (lasklint/staging)" (App Service Slot), "lamacamwp" (App Service), "Default1" (App Service plan), "lasklint" (Application Insights), "Visual Studio Ultimate with MSDN" (Subscription), "NetworkWatcher\_eastus" (Network Watcher), "staging (lamacamwp/staging)" (App Service Slot), "cloud-shell-storage-southeastasia" (Resource group), "Default-Web-NorthEurope" (Resource group), "SendGrid" (SendGrid Account), and "c1af609fe3bare40d94e" (Storage account). Below this is a "Navigate" section with links for Subscriptions, Resource groups, All resources, and Dashboard. There's also a "Tools" section with links for Microsoft Learn, Azure Monitor, Security Center, and Cost Management.

The Azure Portal is the most common tool to use when you get started with Azure. In essence it is a website where you log in with your Azure credentials, and you can then start ruling the world... I mean, start using all your Azure resources. It is a visual way to create, manage and organize all your Azure things. You can see your invoices, your traffic, your scale set, your storage accounts, and everything else you have hosted on Azure.

The Portal lets you investigate logs, set up alerts, monitor your security posture, implement policies, manage users and so much more. Almost everything you can do in Azure you can do in the Portal. The few exceptions are services in preview, and specific features in non-native Azure services such as Azure Container Instances and Azure Kubernetes Services.

A lot of the examples and tutorials in this book will be using the Azure Portal too. It is an important tool, and Azure professionals use it. It isn't the only

tool though.

## 1.6.2 Azure CLI

The Azure command line interface is just that: an interface for your favorite command line tool, such as Bash. As the name implies, there are no visual interactions or buttons you can click on. It is all white text on black background. Or pink on white, or whatever color you choose, but no buttons!

## Figure 1.9 Azure CLI – Listing App Service Plans

```
C:\Users\me>az appservice plan list
[
{
    "freeOfferExpirationTime": null,
    "hostingEnvironmentProfile": null,
    "hyperV": false,
    "id": "/subscriptions//resourceGroups/Default-Web-NorthEurope/providers/Microsoft.Web/serverfarms/Default1",
    "isSpot": false,
    "isXenon": false,
    "kind": "app",
    "location": "North Europe",
    "maximumElasticWorkerCount": 0,
    "maximumNumberOfWorkers": 10,
    "name": "Default1",
    "numberOfSites": 2,
    "perSiteScaling": false,
    "provisioningState": null,
    "reserved": false,
    "resourceGroup": "Default-Web-NorthEurope",
    "sku": {
        "capabilities": null,
        "capacity": 1,
        "family": "S",
        "locations": null,
        "name": "S2",
        "size": "S2",
        "skuCapacity": null,
        "tier": "Standard"
    },
    "spotExpirationTime": null,
    "status": "Ready",
    "tags": null,
    "targetWorkerCount": 0,
    "targetWorkerSizeId": 0,
    "type": "Microsoft.Web/serverfarms",
    "workerTierName": null
}
]
```

In Figure 1.9 you can see a typical Azure CLI command for listing all Azure App Service Plans for a subscription.

```
az appservice plan list
```

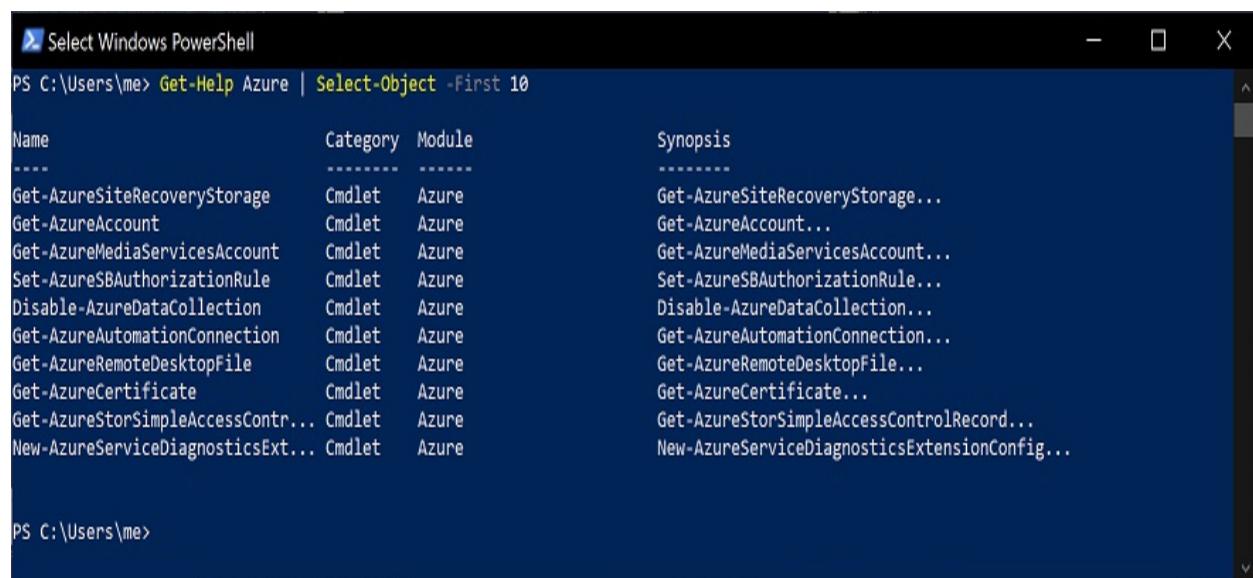
Where the Azure Portal is visual and you click through sections of screens, or panes as they are called, and forms, the CLI is direct and fast. Because of these benefits, many Azure professionals will use the CLI day-to-day.

### 1.6.3 Azure PowerShell

If you think the Azure CLI is appealing, but want a sprinkling of scripting and configuration, PowerShell is here. While you can use PowerShell as an ordinary command line tool, the real power is two-fold.

1. Cmdlets (pronounced command-lets), which are specialized .NET classes implementing a particular operation, such as creating a new storage account or updating a user account. These work by accessing data in different data stores, like the file system or registry.
2. The PowerShell scripting language that lets you create automated scripts for performing batches of commands. These scripts can include cmdlets as well as standard scripting commands.

Figure 1.10: PowerShell console.



The screenshot shows a Windows PowerShell window titled "Select Windows PowerShell". The command entered was "PS C:\Users\me> Get-Help Azure | Select-Object -First 10". The output displays a table with columns: Name, Category, Module, and Synopsis. The table lists ten Azure cmdlets:

Name	Category	Module	Synopsis
Get-AzureSiteRecoveryStorage	Cmdlet	Azure	Get-AzureSiteRecoveryStorage...
Get-AzureAccount	Cmdlet	Azure	Get-AzureAccount...
Get-AzureMediaServicesAccount	Cmdlet	Azure	Get-AzureMediaServicesAccount...
Set-AzureSBAuthorizationRule	Cmdlet	Azure	Set-AzureSBAuthorizationRule...
Disable-AzureDataCollection	Cmdlet	Azure	Disable-AzureDataCollection...
Get-AzureAutomationConnection	Cmdlet	Azure	Get-AzureAutomationConnection...
Get-AzureRemoteDesktopFile	Cmdlet	Azure	Get-AzureRemoteDesktopFile...
Get-AzureCertificate	Cmdlet	Azure	Get-AzureCertificate...
Get-AzureStorSimpleAccessContr...	Cmdlet	Azure	Get-AzureStorSimpleAccessControlRecord...
New-AzureServiceDiagnosticsExt...	Cmdlet	Azure	New-AzureServiceDiagnosticsExtensionConfig...

At the bottom of the window, the prompt "PS C:\Users\me>" is visible.

Figure 1.10 shows how PowerShell can be used to list out 10 Azure cmdlets with a single command. This approach can be used to automate, maintain, create and update any part of Azure in a customized way.

PowerShell isn't only for Azure though. It is a tool and platform that can be used to manage Windows in general, and it runs on macOS and Linux platforms too. You'll get to use PowerShell throughout this book as well.

#### 1.6.4 SDKs

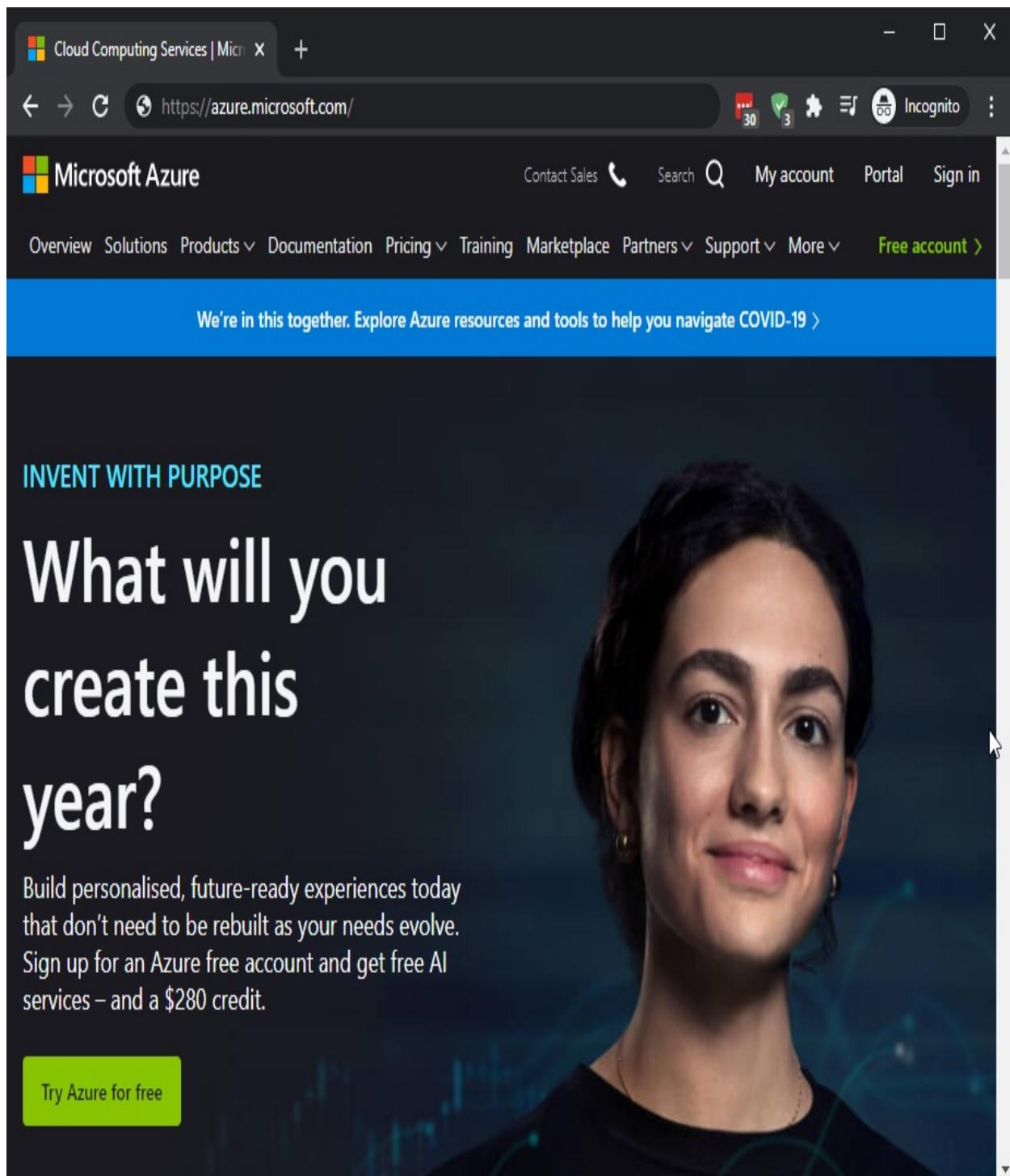
Using the Azure CLI or PowerShell can fall short if you are wanting to integrate Azure services and APIs with your own applications. While those two tools are the preference of systems administrators for managing infrastructure and spinning up new systems, developers prefer to use a software development kit, commonly known as an SDK.

Writing code for an application or service that uses Azure cloud computing means using the Azure SDK, which is available for .NET, Java, Python and JavaScript to name the most common programming languages. The SDK is a collection of libraries designed to make using Azure services in your programs easy and smooth sailing. Cloud sailing. We will dig into the Azure SDK and its use later in the book.

### 1.7 Creating an Azure account

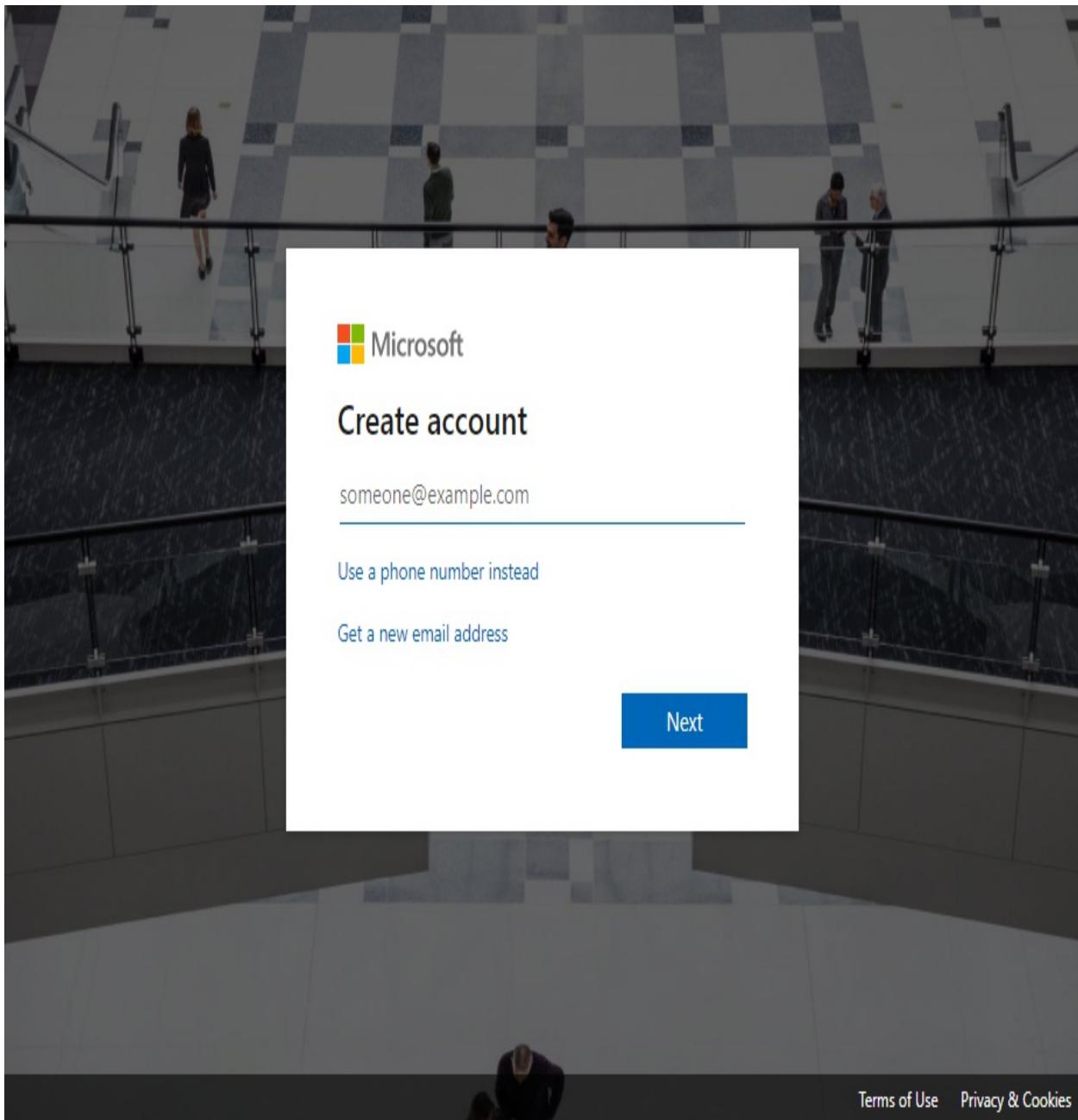
Considering the majority of this book will give you an opportunity to learn and try out Azure for yourself, let's end this chapter by creating a brand new, shiny, lemon scented Azure account. Go to [azure.com](https://azure.com), which will redirect you to the current Azure landing page. On that click **Try Azure for free, Free account** or the current similar button.

**Figure 1.11:** Azure landing page.



When you get to the actual signup page, you will be asked to create a new Microsoft account, which will be your login for all the vast family of Microsoft services on offer, not just Azure. If you need to, create a new free outlook.com email address in the process (Figure 1.12).

**Figure 1.12: Create a new Microsoft account.**



Create a password (remember to use a password manager), or use the passwordless approach, and your new account will be created. You will then be asked for your personal details, which you must fill in to proceed (Figure 1.13).

**Figure 1.13: Azure account creation.**

The screenshot shows the Azure free account creation interface. On the left, a 'Your profile' section contains fields for Country/Region (Australia), First name (Lars), Last name (Klint), Email address (redacted), Phone (redacted), and ABN (Optional). On the right, a dark panel titled 'Create your Azure free account' lists benefits: 'Popular services free for 12 months', '25+ services always free', and '\$260 credit to use in your first 30 days'. A 'No automatic charges' box states that after the credit, users can choose to continue with pay-as-you-go. A 'Chat with Sales' button is at the bottom.

Your profile

Country/Region i

Australia

Choose the location that matches your billing address. You cannot change this selection later. If your country is not listed, the offer is not available in your region. [Learn More](#)

First name

Lars

Last name

Klint

Email address i

Phone

ABN i

Optional

Create your Azure free account

Popular services free for 12 months

25+ services always free

\$260 credit to use in your first 30 days

No automatic charges

After your credit, we'll ask you if you want to continue with pay-as-you-go. If you do, you'll only pay if you use more than the free amounts of services.

Chat with Sales

In the process you will be required to authenticate your identity by phone and you will have to provide a valid credit card. The card will not be charged, unless you agree to pay for certain services, but it is needed to validate your account (and that you aren't Rob the Robot). Finally, you are good to go and can use your new credentials to log into the Azure Portal at <https://portal.azure.com>. Give it a go. It so pretty.

## 1.8 Summary

- Azure is applicable to real-world scenarios and can be effective and desirable for a variety of projects and companies.
- Scalability and reliability are two of the main factors that define Azure and cloud computing in general.
- Cost can be managed by using a combination of PAYG services, reserved instances of VMs and spot pricing VMs.
- Azure has services that are always free, such as Cosmos DB free tier, a certain number of Azure Function executions and some Azure App Service Plans.
- The three types of computing relevant to Azure is cloud, on-premises and multi-cloud.
- The main ways to interact with Azure is using the Azure Portal, the Azure CLI, PowerShell, or integrating with an Azure software development kit.
- You can create a free Azure account to get started trying out services and features.

[1] Not all services are available in all regions.

[2] Any apparently useless activity which, by allowing you to overcome intermediate difficulties, allows you to solve a larger problem.

[3] <https://docs.microsoft.com/en-us/azure/>

[4] <https://docs.microsoft.com/en-us/answers/products/azure?product=all>

[5] 16 February 2021

# 2 Using Azure: Azure Functions and Image Processing

## This chapter covers

- Creating cloud infrastructure for an application
- Exploring and creating integrated logic work flows
- Interconnecting Azure services
- Exploring a serverless architecture

Now that you have an idea why Azure is so popular and why it is a great choice for computing infrastructure, let us dive into a practical example of using Azure to solve a real problem of storing images from emails and compressing them. We will keep it simple, but that doesn't mean it will lack in power and application options.

### Note:

This example can be done entirely with the free subscription tier in Azure. It is recommended to have a new Azure account for the free credits, hours and services to still be available.

Relating knowledge to examples and implementations that are as close to real world as possible is, in my humble opinion, the best way to learn a technical topic, and cloud computing with Azure is no exception. In this chapter we will build an application that follows just that principle.

## 2.1 Understanding the Problem

Imagine you are working for a company that sells photography and imaging services. One of the products is a service that lets users send their images via email for an online photo album. The online albums are growing in size as images are getting bigger and bigger with modern camera technology. This in

turn means that the website showing the images in the photo albums is getting slower and slower, as images gets larger and more of them needs to load onto the website.

To solve this problem, you have been tasked with creating a process or application that gets the images from the email account and gets them ready for display on the website. This will include storing the images in both their original and web-optimized formats. There are several project outcomes that needs to be achieved:

- Scaling of service to accommodate for increased traffic.
- Cost effective.
- Reduced maintenance, as the company has limited resources.
- Robust and fault tolerant.

And while there are project goals to achieve, there are also some parts we explicitly leave out. It is as important to know what *not* to include, as it is what *to* include.

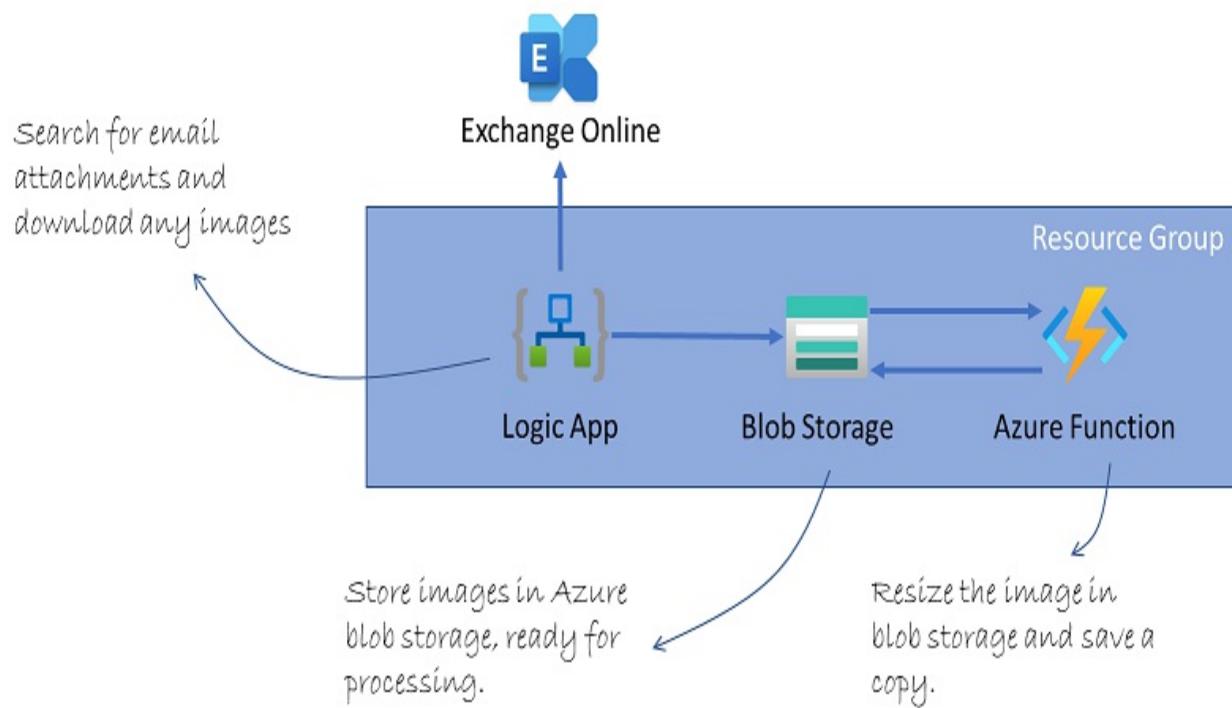
- The album website, which already exists.
- The email account setup and maintenance.
- The logic for associating an email sender with an album.

Knowing what we know so far from chapter 1, this image optimization project is a great fit for using Azure cloud computing.

## 2.2 Creating the Architecture

Before we start building, let's look at the architecture for this new application of fantastic wonderment. Or application of moderate image compression logic at least. It is always a good idea to lay out the intended approach before you start building an application. Often this step can identify any obstacles or issues early. Figure 2.1 shows the architecture we will use to build the image-and-album-compression-via-email application.

**Figure 2.1: Application architecture overview**



The first step is to hook into the arriving emails on the email service Exchange Online. We won't go through how to set up that service, but simply assume emails arrive. In fact, this could be *any* email account, not just Exchange. We will use a Logic App to "interrogate" the email account and catch any attachments that arrive. One of my favorite services on Azure is Logic Apps. It is a simple visual way to integrate a vast range of services using logic constructs, such as conditional operators (*and*, *or*, *else*), loops and comparisons. Logic Apps on Azure are incredibly powerful and you only need a limited technical understanding of Azure to use them.

Attachments, which should be images, are then stored in Azure Blob Storage. Blob storage is a versatile, ever expanding storage mechanism for storing pretty much anything. It is a native part of an Azure storage account, and this is where the images will live and the album website gets them from too.

Once there are new images in Blob storage, an Azure Function will be triggered and process them, compress them, and store the compressed copy back in Blob storage. That way both the original image is kept, as well as a version that is more efficient to show on the website.

All of these Azure services will live in a resource group, which is a logical container for any Azure service. Make sense? Alright, let's start building with cloud.

### 2.2.1 What is a resource group?

The first thing that any Azure project needs is one or more resource groups (Figure 2.2). All resources on Azure must live inside a resource group, and resource groups must live inside a single Azure subscription. They are a logical container for your resources. They can be created in any geographical Azure region, and they can hold up to 800 resources in general. Please don't have 800 though. That would most likely be a mess. Instead, resource groups can be used to group resources by project, department, service, feature, or any other way that make sense to you. When used appropriately, such as grouping by project, resource groups can make your cloud life a whole lot easier.

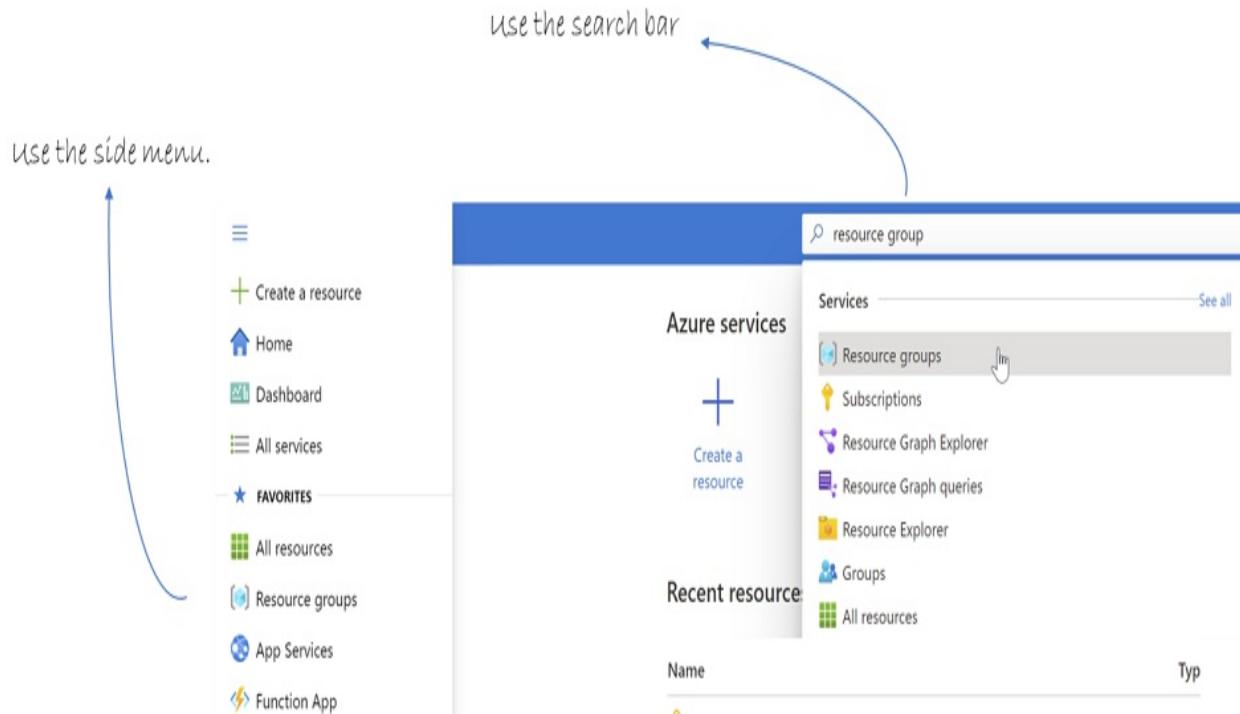
**Figure 2.2: The first step is the resource group.**



### 2.2.2 Creating a resource group

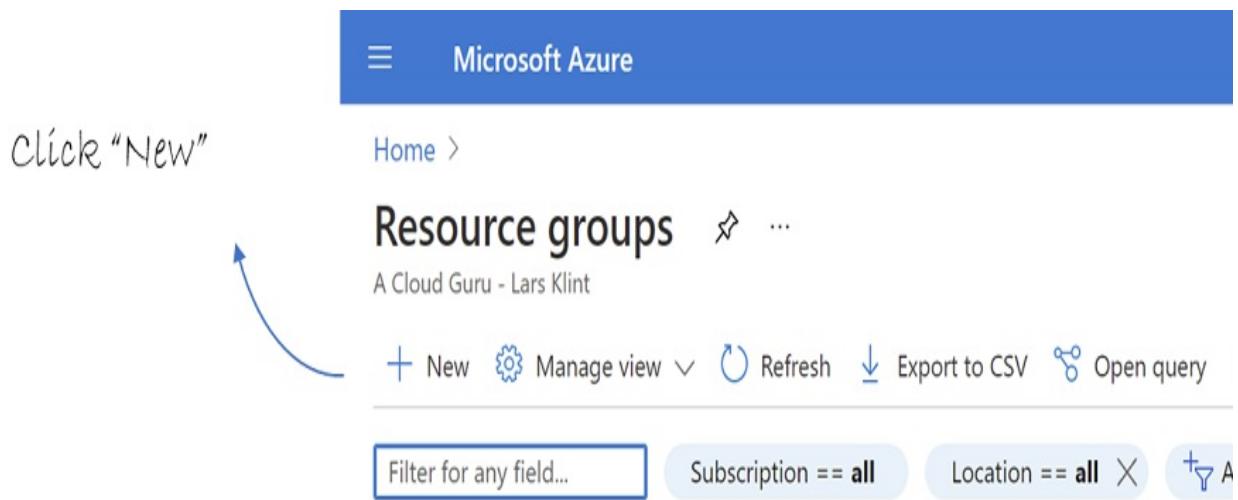
To start creating the application, first open up the Azure Portal and create a new resource group as shown in Figure 2.3.

**Figure 2.3: Go to the resource group section.**



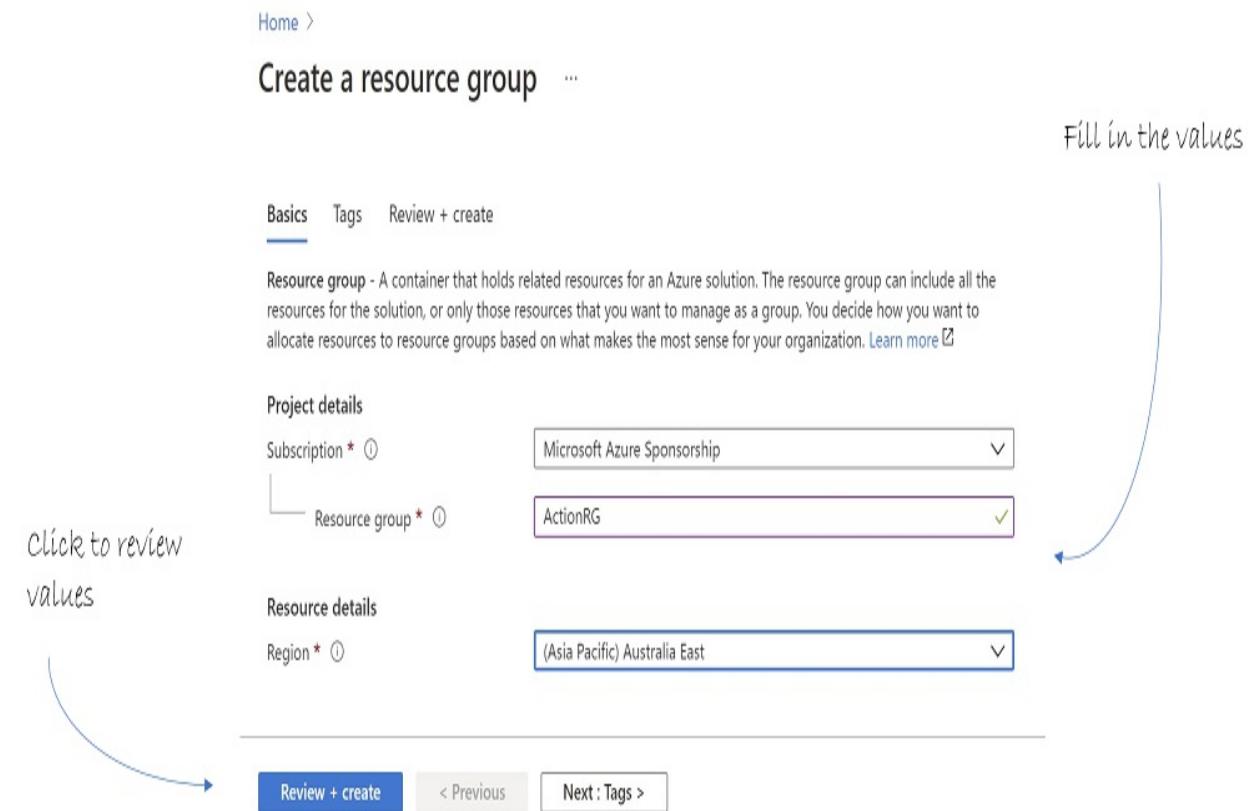
Click New to start creating a new resource group. This will open the resource wizard, which is how most resources can be created in Azure.

**Figure 2.4: Create a new resource group**



Go through the wizard as shown in Figure 2.5 and fill in the values as you see best. Follow a naming convention that works for you as well to get into the habit of good naming of resources.<sup>[1]</sup>

**Figure 2.5: Enter values for the resource group.**



It doesn't matter what name you give a resource group. They are all valid, however it is recommended to decide on a naming strategy before you start. I like using the format <project name>RG. This makes it instantly clear which project I am dealing with and the kind of resource, which is a resource group in this case. Often companies and teams will have a naming strategy they follow for just these reasons.

The wizard experience is a taste for how resources are created through the Azure Portal. As you will learn later in the book, whichever way you create resources in Azure it is always done through the Azure Resource Manager. You get the same result regardless of the tool you choose. For now, we'll use the Azure Portal wizard experience and click *Review + Create* when you have filled in the details.

**Figure 2.6: When validation has passed, you can create the resource group.**

# Create a resource group

Validation passed.

Basics Tags Review + create

---

Basics

Subscription	Microsoft Azure Sponsorship
Resource group	ActionRG
Region	Australia East

Tags

None
------

---

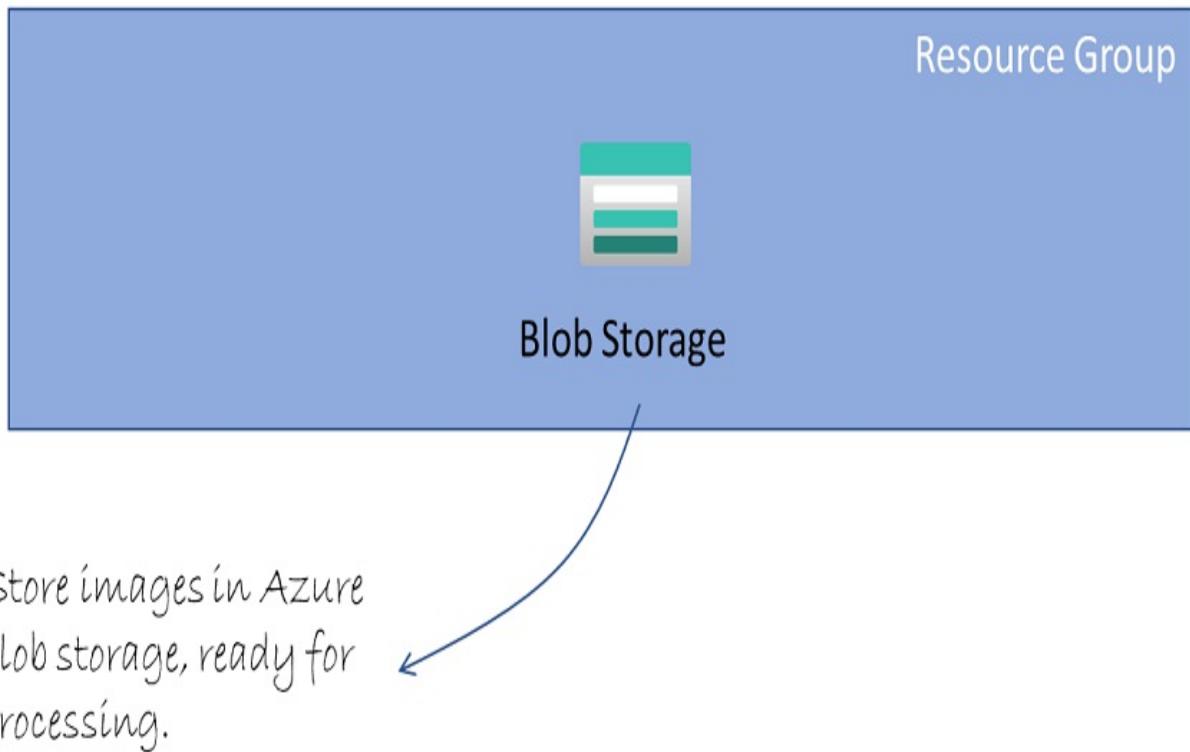
[Create](#) [< Previous](#) [Next >](#) [Download a template for automation](#)

Finally, click the *Create* button as shown in Figure 2.6. This is the first step done for pretty much any new project or logical group in Azure. The resource group is a fundamental part of your Azure life, and you will create a ton of them. As long as a ton is less than 800 resource groups, of course.

## 2.2.3 Blob storage – the basics

Alright, you got your toe slightly wet from creating a resource group in Azure, so let's jump all the way in and create our first service: a storage account with a blob storage container inside it (Figure 2.7).

**Figure 2.7: We are now creating the Storage Account and blob storage of the solution**



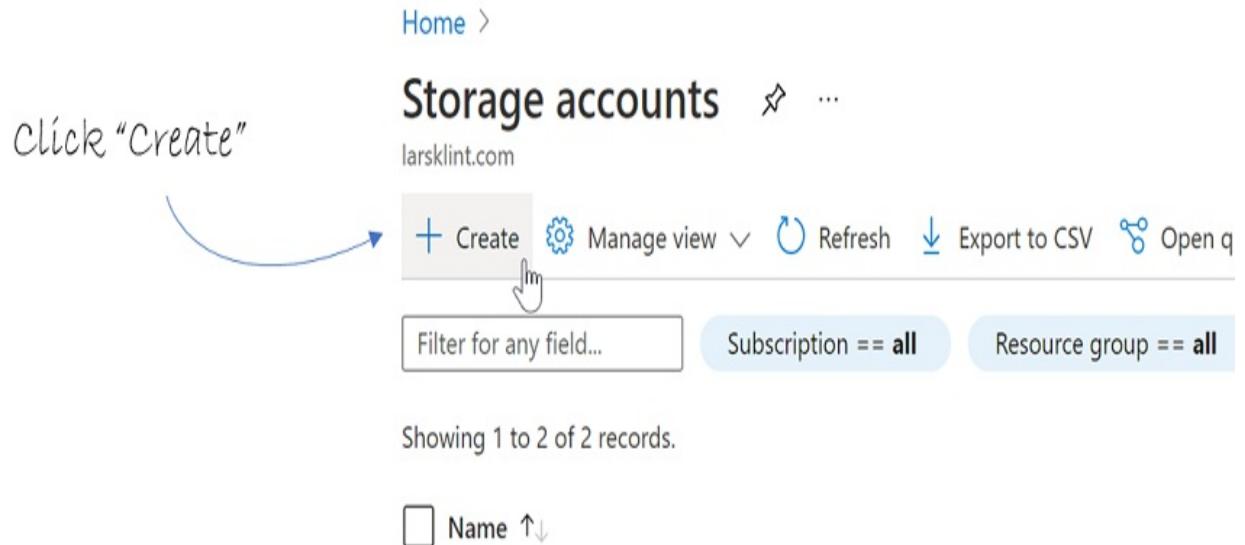
It isn't the most interesting or exciting service, but it is an integral part of a lot of Azure architectures. Storage accounts in Azure are a bit of a Swiss army knife. They have a bunch of tools packed up inside them, such as containers, tables, queues, file shares, data migrations and much more. Later in the book you'll learn about storage in Azure in practical details, but for now we just need a storage account to put the email attachments somewhere for our service. Go to the Storage Accounts section of the Azure Portal using the search bar.

## 2.2.4 Creating a storage account

You now have an overview of all the storage accounts available on this *tenant*. A tenant is the entity that controls all your users, resources, and applications when it comes to identity and access management on Azure. There are different ways to create a storage account, but for now click on the *Create* button as shown in Figure 2.8. You'll find that services in Azure can be created and found in multiple ways, and I will show you some of them

throughout this book.

**Figure 2.8: Click Create to create a new container within the storage account**



While clicking the “+ Create” button is a simple act, a lot goes on behind the scenes throughout the wizard experiences. Part of the appeal of cloud computing with Azure is the abstraction of “yak shaving”, which I mentioned in the previous chapter. Tasks that have to be done before you get to the tasks you *really* want to do.

Alright, time to go through another wizard and fill in the necessary values to create a storage account as shown in Figure 2.9. Go on, it’ll be a beauty.

**Figure 2.9: Fill in the basic values for the storage account.**

# Create storage account

...

Basics Networking Data protection Advanced Tags Review + create

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below.

[Learn more about Azure storage accounts](#)

## Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \*

Microsoft Azure Sponsorship



Resource group \*

ActionRG



[Create new](#)

## Instance details

The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model instead. [Choose classic deployment model](#)

Storage account name \* ⓘ

azureaction



Location \*

(Asia Pacific) Australia East



Performance ⓘ

Standard  Premium

Account kind ⓘ

StorageV2 (general purpose v2)



Replication ⓘ

Read-access geo-redundant storage (RA-GRS)



As you go through creating more and more resources on Azure, this wizard experience will become familiar. While they all follow the same format of “Basics” -> “Networking” -> other steps, the amount of data needed to create a resource varies a lot. A resource group, like you created a minute ago, requires very few bits of information to get created. Others require a significant amount. Often resources require other resources to be created first, such as a resource group for our storage account.

Choose the values you want, but make sure you choose the resource group you created before. You can leave the rest of the values as default for now. Later in the book we go through what most of them mean in detail.

## 2.2.5 Storage account – networking

Move on to the *Networking* tab by clicking Next in the Portal Wizard experience.

**Figure 2.10:** Networking settings for the storage account.

## Create storage account ...

X

Basics Networking Data protection Advanced Tags Review + create

This setting will allow anyone to access the storage account.

### Network connectivity

You can connect to your storage account either publicly, via public IP addresses or service endpoints, or privately, using a private endpoint.

#### Connectivity method \*

- Public endpoint (all networks)
  - Public endpoint (selected networks)
  - Private endpoint
- ⓘ All networks will be able to access this storage account.  
[Learn more about connectivity methods](#)

### Network routing

Determine how to route your traffic as it travels from the source to its Azure endpoint. Microsoft network routing is recommended for most customers.

#### Routing preference \*

- Microsoft network routing (default)
- Internet routing

[Review + create](#)

[< Previous](#)

[Next : Data protection >](#)

For some Azure resources there are a lot of settings. It can therefore be tempting to just glance over them, accept the default and be done with it. It will certainly get the resource created faster, but at what cost? Take the **Networking** section for the storage account, as shown in Figure 2.10. Leaving the *Connectivity method* as default will leave the access to the storage endpoint open to the public. This might be okay, if the data is for public consumption, but it might also be not okay. As in “if this is left public, we will expose all our customer data” not okay.

The lesson for today: Don’t accept default values as correct, as they may affect the entire project down the line. Having said that, we *are indeed* going

to accept the default values in this instance. Don't argue. I'm writing this book.

## 2.2.6 Storage account – data protection

Now, click “Next: Data Protection”, which will reveal Figure 2.11.

**Figure 2.11: Data Protection settings for the storage account.**

The screenshot shows the 'Create storage account' wizard at the 'Data protection' step. The top navigation bar includes 'Basics', 'Networking', 'Data protection' (which is underlined), 'Advanced', 'Tags', and 'Review + create'. The main content area is titled 'Recovery' and contains several configuration options with checkboxes:

- Turn on point-in-time restore for containers: A note explains that this requires versioning, change feed, and blob soft delete. A 'Learn more' link is provided.
- Turn on soft delete for blobs: A note explains that this enables recovering blobs previously marked for deletion. A 'Learn more' link is provided.
- Turn on soft delete for containers: A note explains that this enables recovering containers previously marked for deletion. A 'Learn more' link is provided.
- Turn on soft delete for file shares: A note explains that this enables recovering file shares previously marked for deletion. A 'Learn more' link is provided.

Below these, the 'Tracking' section contains two options:

- Turn on versioning for blobs: A note explains that this maintains previous versions for recovery and restoration. A 'Learn more' link is provided.
- Turn on blob change feed: A note explains that this tracks changes to blobs. A 'Learn more' link is provided.

At the bottom are three buttons: 'Review + create' (highlighted in blue), '< Previous', and 'Next : Advanced >'.

The **Data Protection** section in Figure 2.11 is a great example of service

specific settings as opposed to the **Basics** section that defines generic settings like name and resource group. In this case you can turn on *soft delete* for various part of a storage account. Soft delete lets you recover data marked for deletion, giving you a second chance to keep it. It is a feature specific to storage accounts. These service specific settings are found in most resource creation wizards in the Azure Portal.

And again, leave the default values (after reading what they do carefully of course), then click the big blue button “Review + create”. Yes, we will skip the **Advanced** and **Tags** steps in this example.

The *Create* step, also has a review built in, which checks that all the values and options you have entered are valid. Once confirmed, you can create the storage account by clicking the “Create” button. Surprise surprise.

You have now gone through an entire experience for an Azure Portal Wizard. This one was for creating a storage account, but all the Azure Portal wizards are very similar. The wizard will make sure you fill in all the needed values, but it is still your responsibility to make sure those values are accurate and make sense for your scenario.

## 2.2.7 Creating a storage container

Once the Azure Storage account is created, go the overview of the resource (Figure 2.12). Although you have now created a storage account, you can't store any data in it yet. Yes, I know, that is a bit strange, but bear with me.

**Figure 2.12:** Storage account overview.

The screenshot shows the Azure Storage account overview for the 'azureaction' storage account. The left sidebar includes links for Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, Storage Explorer (preview), and Settings (Access keys, Geo-replication, CORS, Configuration, Encryption, Shared access signature, Networking). The main content area displays the following details:

Resource group (change) : ActionRG		Performance/Access tier : Standard/Hot
Status	: Primary: Available, Secondary: Available	Replication : Read-access geo-redundant storage (RA-GRS)
Location	: Australia East, Australia Southeast	Account kind : StorageV2 (general purpose v2)
Subscription (change)	: Microsoft Azure Sponsorship	
Subscription ID	: 694ae2bd-ea92-4302-8069-db43c3533f9c	
Tags (change)	: Click here to add tags	

Below this, there are four cards:

- Containers**: Scalable, cost-effective storage for unstructured data. [Learn more](#)
- File shares**: Serverless SMB and NFS file shares. [Learn more](#)
- Tables**: Tabular data storage. [Learn more](#)
- Queues**: Effectively scale apps according to traffic. [Learn more](#)

At the bottom, there are links for Tools and SDKs: Storage Explorer (preview), PowerShell, Azure CLI, .NET, Java, Python, and Node.js.

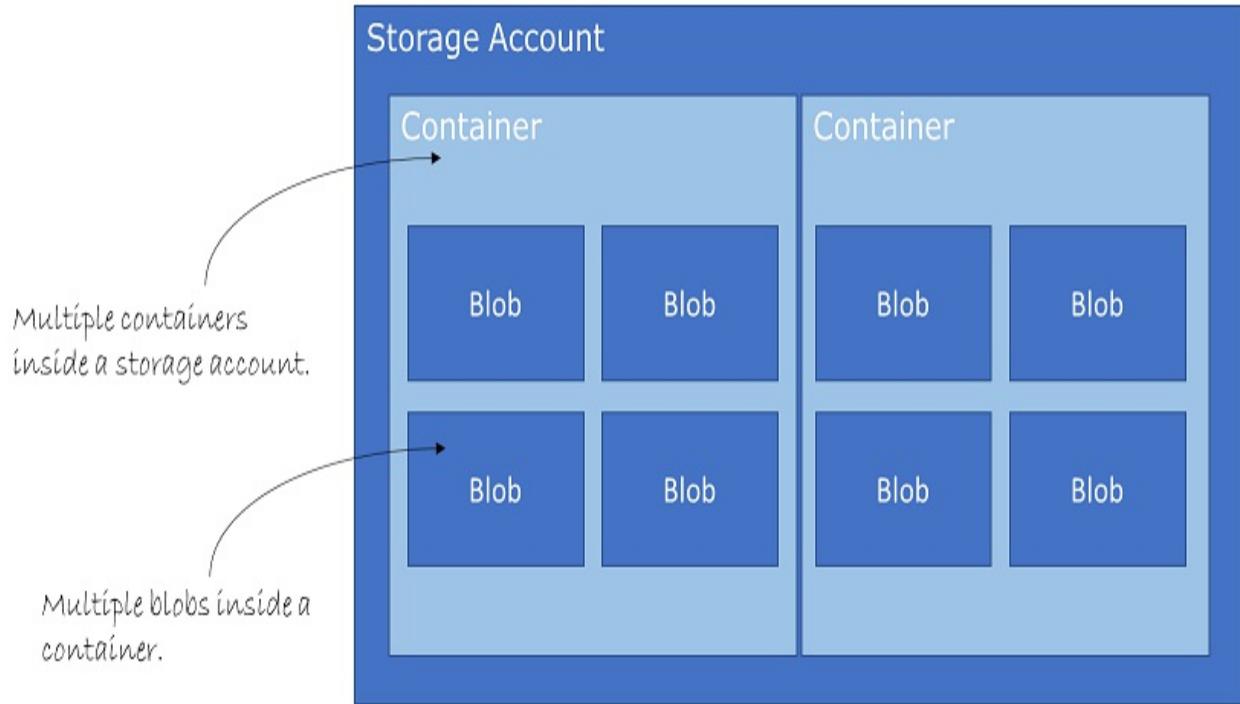
A storage account is like a framework for storing data in different ways.

- **Containers**: This type of data can be anything. It is unstructured data, which means you can mix photos with executable files with text documents. Anything goes.
- **File shares**: These are your old school structured files like on Windows 95. Okay, slightly more modern, but it is just a directory of files. In the cloud.
- **Tables**: A key-value store of semi-structured data. It is very fast, and very cheap to store enormous datasets.
- **Queues**: An asynchronous message queue for communication between application components. Add data that needs to be passed from one part of an application to another.

We are going to use containers, which can in turn store blobs, or binary large objects. A container organizes a set of blobs, similar to a directory in a file system. A storage account can include an unlimited number of containers, and a container can store an unlimited number of blobs. Storage account,

container, blob (Figure 2.13).

**Figure 2.13: Storage account hierarchy: blobs inside containers inside a storage account.**



To create a container, click on the “Containers” area of the overview in Figure 2.12. This will take you to a list of all the containers you currently have in your storage account, which will be none (Figure 2.14).

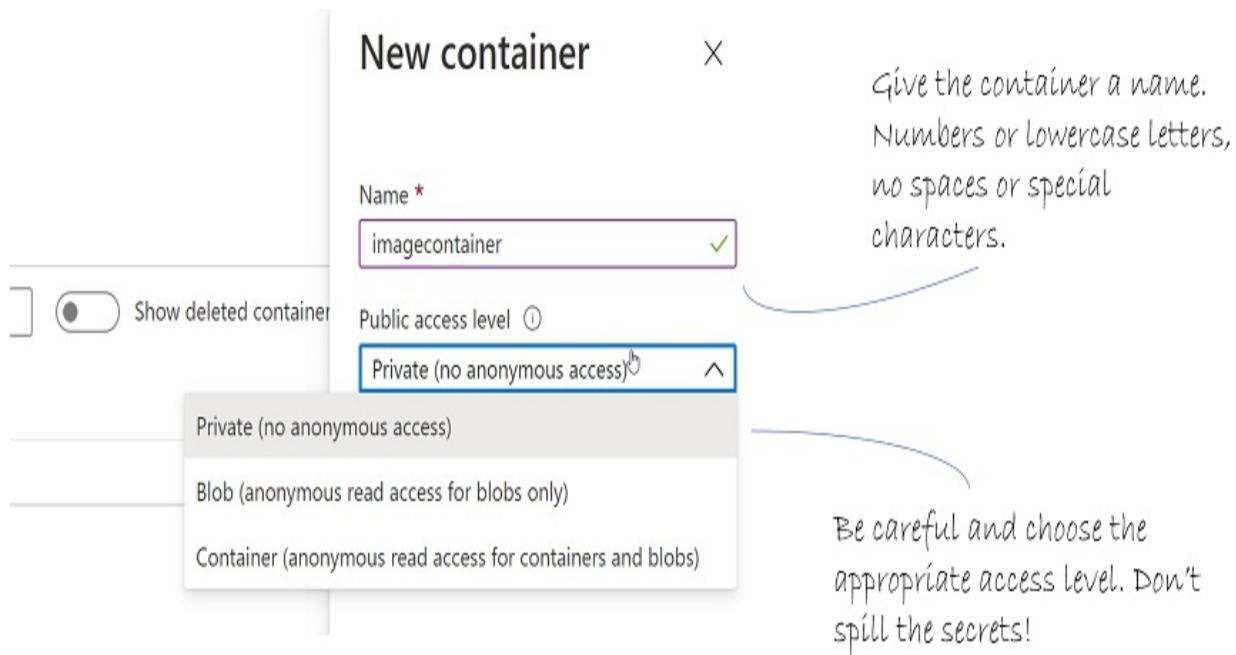
**Figure 2.14: Container overview**

Azure storage container list interface. The left sidebar shows navigation links: Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, Storage Explorer (preview), Settings, and Access keys. The main content area has a search bar, a '+ Container' button, and other controls like 'Change access level', 'Restore containers', 'Refresh', and 'Delete'. A callout bubble points to the '+ Container' button with the text: 'click the “+ Container” button create a new container'.

Click on the “+ Container” button to create a new container. No surprises there. You can have as many containers as you want. There is no limit. Although, some restraint in how many you create is recommended as it can get out of hand, and then you don’t know where any of your blobs are. Don’t lose track of your blobs.

When you create a new container it happens in this anti-climactic kind of way. A small side panel opens with two properties to fill in: **Name** and **Public access level** as shown in Figure 2.15.

**Figure 2.15: New container properties**



While there are only two mandatory fields, the **Public access level** is very important. This is where you determine how your blobs can be accessed. As Figure 2.15 shows there are three tiers:

- **Private:** Access only with an authenticated account, such as a service principal.
- **Blob:** Anonymous access to blobs inside the container, such as images and other multimedia data. There is no access to container data.
- **Container:** This tier will give public anonymous access to both the container and the blobs.

When you select a tier that allows *some* public access, a warning (Figure 2.16) will help you be *completely* sure that is the right level for you. For this application you can choose **Blob** to allow access to the blobs from the Azure Function we will build soon.

Figure 2.16: Warning for public access level of “Container”

## New container

X

Name \*

imagecontainer



) Public access level ⓘ

Container (anonymous read access for containers and blobs)

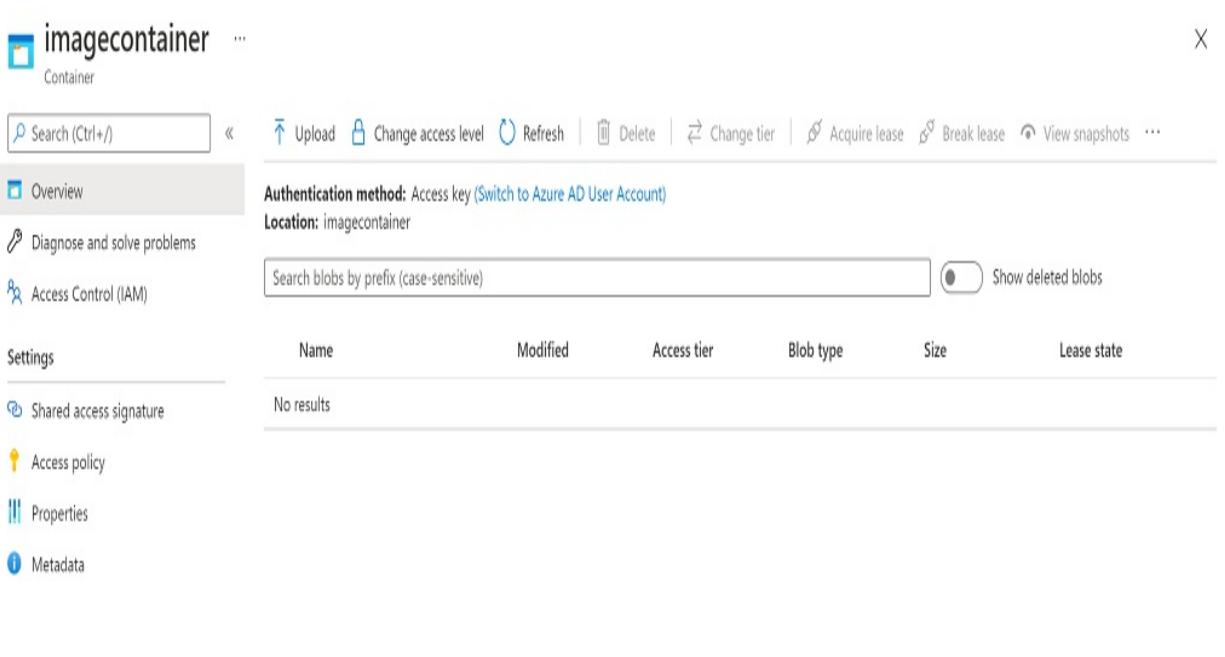


All container and blob data can be read by anonymous request. Clients can enumerate blobs within the container by anonymous request, but cannot enumerate containers within the storage account.

▼ Advanced

Now you have created a container to hold our blobs in, shown in Figure 2.17. However, a blob storage needs some blobs. Let's fill some blobs into it then using logic apps and Outlook integration.

**Figure 2.17:** A complete container shown in the Azure Portal.

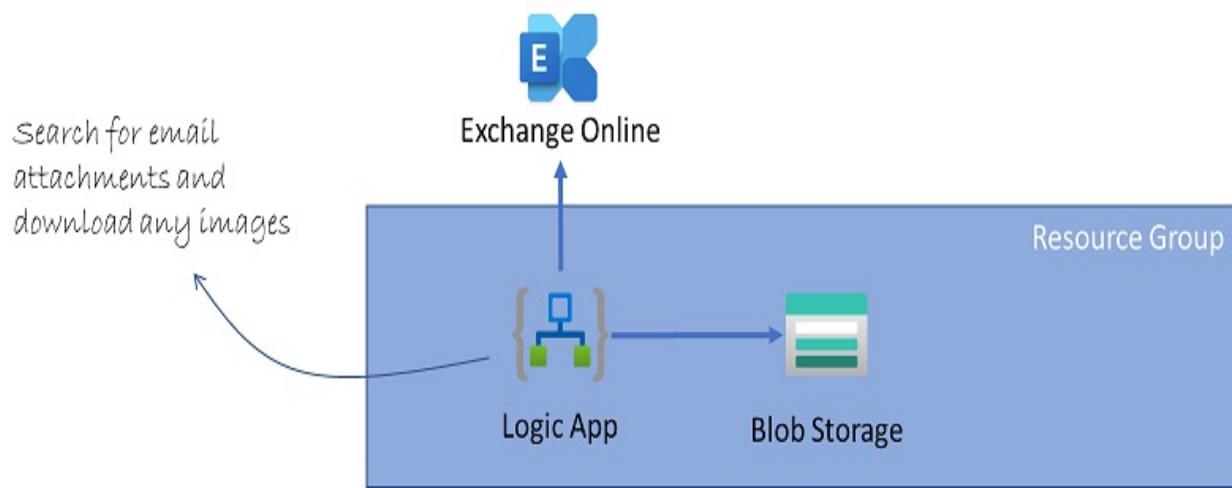


## 2.2.8 What is a Logic App?

A Logic App. This aptly named service lets you create a logic flow of events that can handle, manipulate, and direct data between services in and out of Azure. Logic Apps are immensely powerful and relatively simple to set up. Did I mention they are in the cloud category of serverless, just like Azure Functions, which adds at least 100 extra cloud points? Where Azure Functions are all code and only does a single function, Logic Apps don't require you to code, and often do multiple steps to achieve the workflow goal.

For this project, we need a Logic App that can interrogate an Outlook online account and get any attachment that is in an email (Figure 2.18). After all, we are asking customers to email us photos, so we need to collect them.

**Figure 2.18: Add the Logic App and email account connection to the project.**



## 2.2.9 Creating a Logic App

Let's start by finding the Logic Apps section, as shown in Figure 2.19. Again, there are multiple ways to create any resource in Azure, and this is one way. Throughout the book you will learn more ways to create resources apart from the Portal.

**Figure 2.19: Create a new Logic App using the Azure Portal.**

The screenshot shows the Microsoft Azure portal interface. At the top, there is a search bar with the text "Logic Apps". Below the search bar, the "Azure services" section is visible, featuring a "Create a resource" button and a "Recent resources" list containing "ActionRG" and "DefaultResourceGroup". A handwritten note on the left side of the screen says "Use the search bar to find 'Logic Apps'". On the right, the search results are displayed under the heading "Services". The "Logic apps" item is highlighted with a gray background and a cursor icon. Other listed services include "Logic Apps Custom Connector", "AppDynamics", "Snapshots", "App Configuration", "App proxy", "App registrations", "App Services", "Application gateways", and "Application groups". To the right of the service names, there are "See all" links, marketplace icons, and document links.

Using the search bar in the Azure Portal is an excellent way of finding resources, third party services, documentation, and other relevant information. In this case we use it to quickly navigate to Logic Apps. You'll see a list of all your Logic Apps like in Figure 2.20, but chances are you won't have any yet.

Figure 2.20: Create a new Consumption type Logic App.

The screenshot shows the Azure Logic Apps portal. At the top, there's a navigation bar with 'Logic apps' and a user icon. Below it is a sub-header 'larsklin.com'. The main area has a toolbar with 'Add', 'Manage view', 'Refresh', 'Export to CSV', 'Open query', 'Feedback', 'Assign tags', and 'Enable/Start'. A dropdown menu is open under 'Add' with 'Consumption' selected. Below the toolbar are filter buttons for 'Subscription == all', 'Resource group == all', and 'Location == all', each with a clear button ('X'). There's also a 'Preview' option and a 'Add filter' button. A callout bubble points from the 'Add' button to the text 'Click 'Add', then choose Consumption as your type.' The text 'Showing 0 to 0 of 0 records.' is displayed. At the bottom, there are sorting options for 'Name ↑', 'Status ↑', and 'Plan ↑'.

Add a new Logic App by clicking “Add”. In this case you get a dropdown, because at the time of writing there are two ways to create Logic Apps.

- **Consumption**, which means you pay for what you use, such as per execution and per connector. This is the one we'll use in this chapter.
- **Preview**, which is a newer way that allows you to create stateful and stateless applications. It is also single tenant, so workflows in the same logic app and a single tenant share the same processing (compute), storage, network, and more. This is outside the scope of this chapter. However, as Confucius says “Friends don’t let Friends put anything Preview into Production”.<sup>[2]</sup>

Creating a new consumption Logic App will open the familiar Azure Portal wizard experience (Figure 2.21), where you specify all the details of your new resource.

**Figure 2.21: Fill in the details for the Logic App.**

Create a logic app ...

Basics Tags Review + create

Create workflows leveraging hundreds of connectors and the visual designer. [Learn more](#)

**Project details**

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* Microsoft Azure Sponsorship ▾

Resource group \* ActionRG ▾ Create new

Choose the resource group you made before.

**Instance details**

Logic app name \* imagefetcher ▾

Region \* Australia East ▾ Give the Logic App a name.

Associate with integration service environment ⓘ

Integration service environment \* ▾

Enable log analytics ⓘ

Log Analytics workspace \* ▾ Then click "Review + Create"

Review + create < Previous : Basics Next : Tags > Download a Template for automation ⓘ

The screenshot shows the 'Create a logic app' wizard in the Azure portal. The 'Basics' tab is selected. The 'Subscription' dropdown is set to 'Microsoft Azure Sponsorship'. The 'Resource group' dropdown is set to 'ActionRG' and has a purple border, indicating it's selected or highlighted. The 'Logic app name' field contains 'imagefetcher'. The 'Region' dropdown is set to 'Australia East'. There are several optional settings: 'Associate with integration service environment' (unchecked), 'Integration service environment' (dropdown), 'Enable log analytics' (unchecked), and 'Log Analytics workspace' (dropdown). At the bottom, there are navigation buttons: 'Review + create' (highlighted in blue), '< Previous : Basics', 'Next : Tags >', and 'Download a Template for automation ⓘ'.

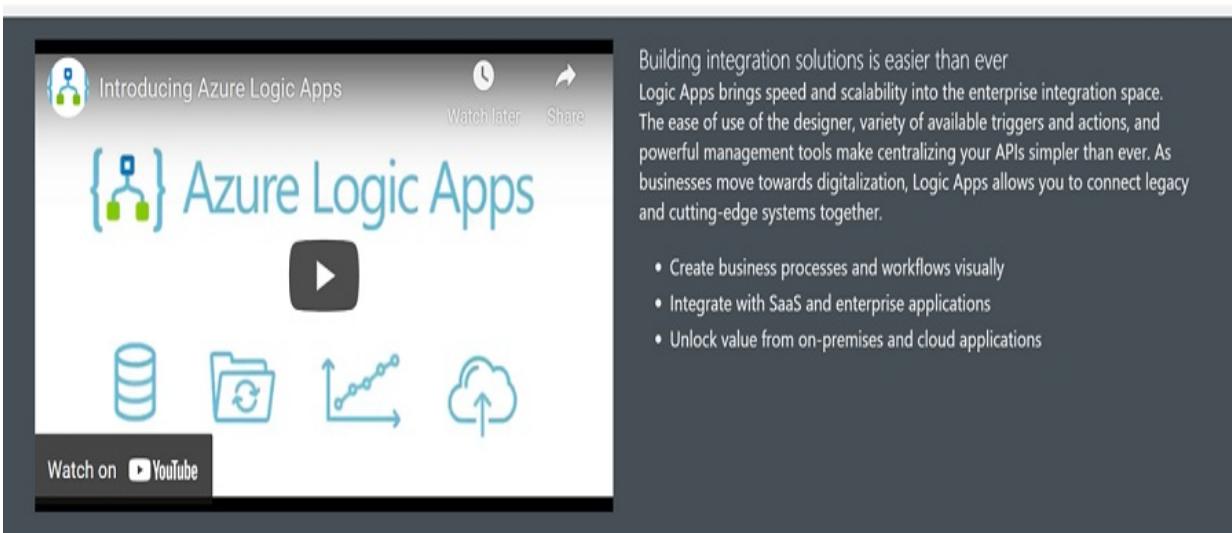
Again, you have to fill in the **Basics**, such as subscription, resource group, name of the resource and region. Most services will ask for this info, whether it is created through the Portal, the CLI or some other tool. Fill it in and then click “Create + review”. We’ll skip the **Tags** section.

Azure will perform a check of your input before the resource is created, just to make sure it all makes sense.

## 2.2.10 Logic App – triggers and connectors

However, clicking “Create” is only the start of getting Logic App brought to life. The resources needed to run the Logic App have now been allocated in Azure, but the App doesn’t *do* anything. Of course, that is about to change. When the Logic App is created, you get presented with a Logic App welcome screen (Figure 2.22).

**Figure 2.22:** Use the “new email received in Outlook.com” trigger.

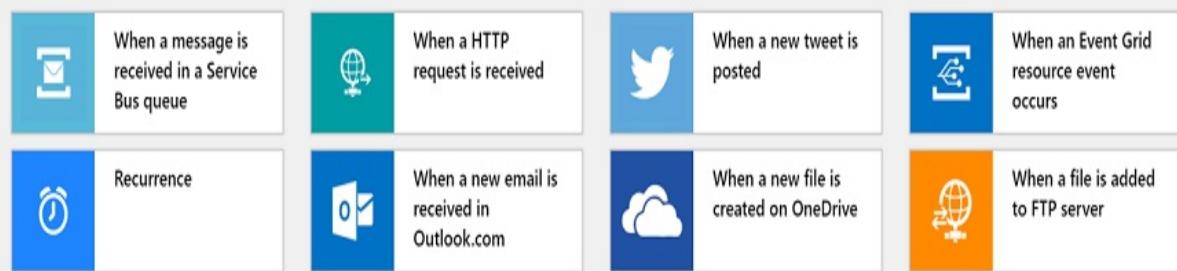


Building integration solutions is easier than ever. Logic Apps brings speed and scalability into the enterprise integration space. The ease of use of the designer, variety of available triggers and actions, and powerful management tools make centralizing your APIs simpler than ever. As businesses move towards digitalization, Logic Apps allows you to connect legacy and cutting-edge systems together.

- Create business processes and workflows visually
- Integrate with SaaS and enterprise applications
- Unlock value from on-premises and cloud applications

#### Start with a common trigger

Pick from one of the most commonly used triggers, then orchestrate any number of actions using the rich collection of connectors



#### Templates

Choose a template below to create your Logic App.

Category:

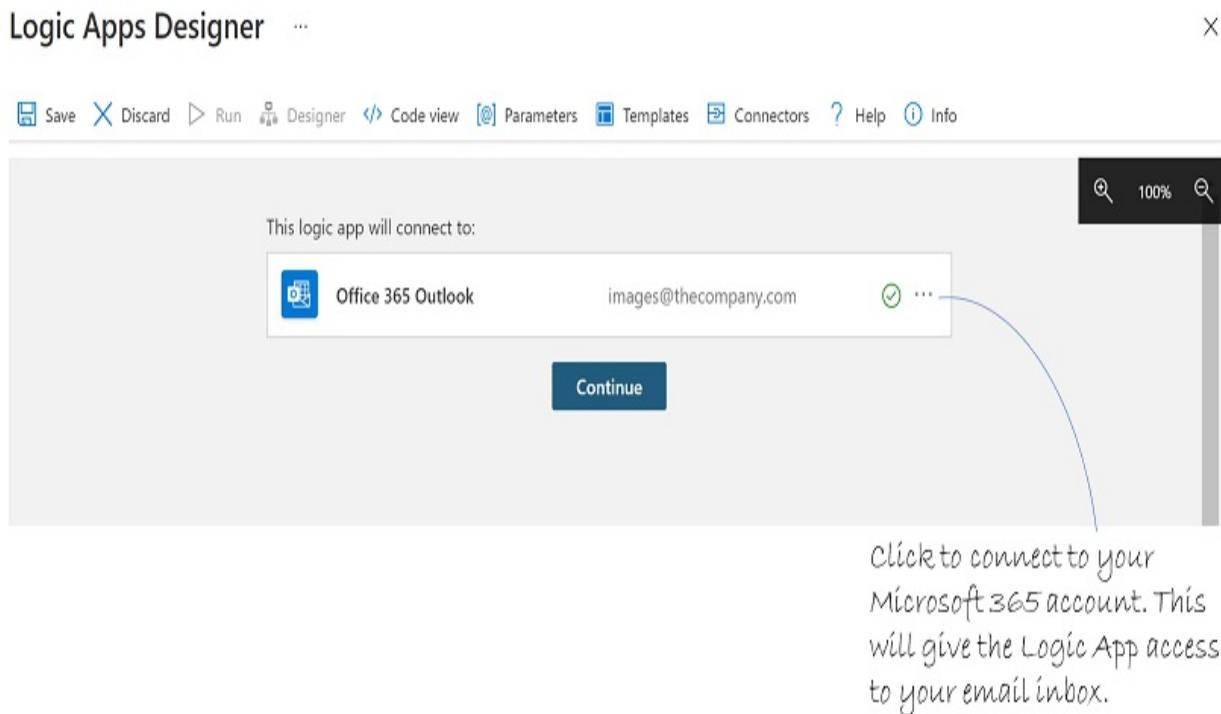
Sort by:

Choose this trigger to get started. It's delicious.

You can watch the introduction if you want, but the more important part is the list of triggers and templates. You don't have to use the templates, but you will have to use triggers. Triggers are what Logic Apps are all about. A trigger is just what the name says, an event triggered by a service. As you can see in Figure 2.22, it can be a file added to an FTP service, a new tweet on Twitter, a webhook, a timer and so much more. In this case find the "new

email received in Outlook.com” trigger and select it. This will start the Logic App visual design experience (Figure 2.23). This is a very approachable and relatively simple way to create incredibly powerful automations.

**Figure 2.23: Connect your email account using the designer.**

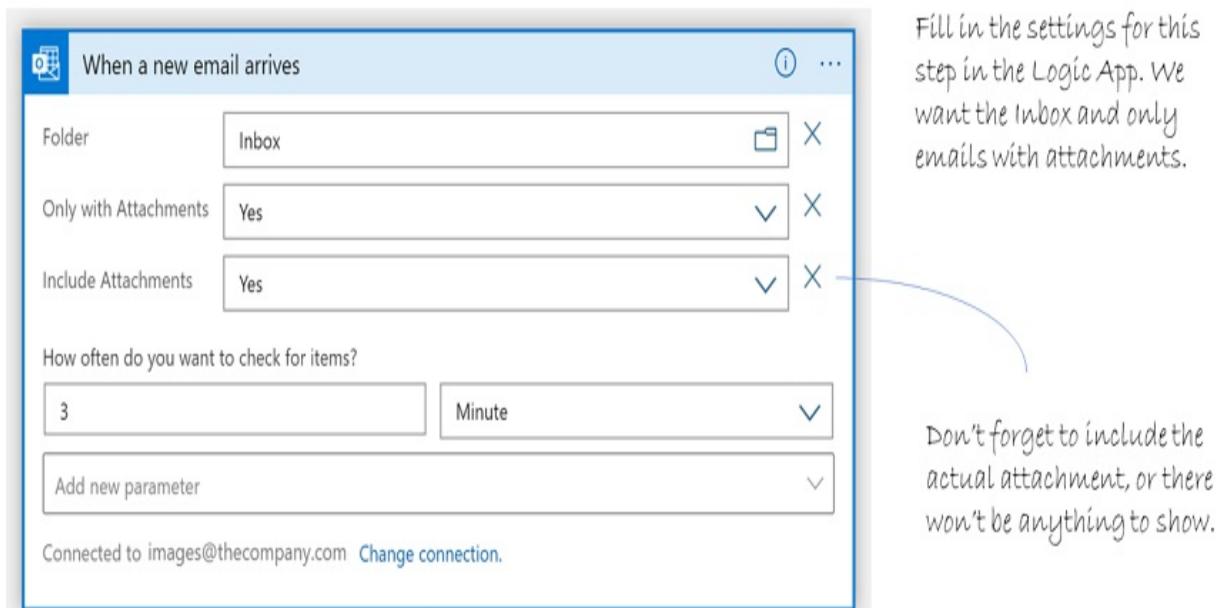


Each step of the Logic App will look like the Outlook step in Figure 2.23. You can clearly see what the sequence of steps is, and how each is linked. First you need to connect your Outlook.com account, in order to receive any emails we can work with. If you already have connections configured, you can choose an existing one, or you can create a new connection. Either way, click the three dots on the trigger to set it up. If you are creating a new connection, you will be asked to log into your account and approve the connection for use in the Logic App.

## 2.2.11 Configuring a Logic App connector

Once you have a valid connection, it’s time to select the parameters we want to use from it, which is Figure 2.24.

**Figure 2.24: Adjust the settings for the “email arrives” step. The “Add new parameter” dropdown lets you adjust the parameters for the step.**



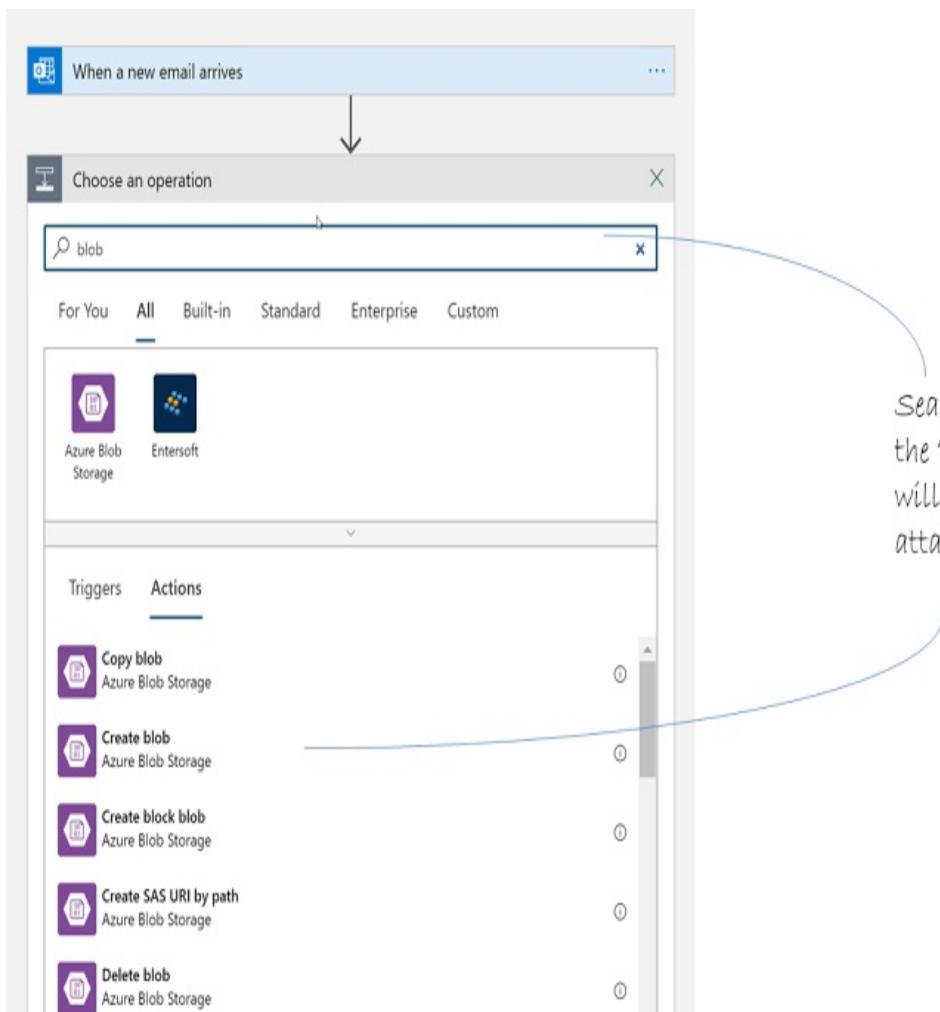
This is an important step to get right. We are now configuring the parameters we need to select the emails we want, which in this case are the ones that have attachments. There are many ways to select the required emails for any scenario, so getting this right is critical to creating our image service.

Make sure you choose the following settings for the parameters

- **Folder** set to *Inbox*
- **Only with Attachments** is set to *Yes*
- **Include Attachments** is set to *Yes*

If you see other parameters than those above, then use the *Add new parameter* dropdown at the bottom of the dialog to select the correct ones. The right values are now available for us to store the image received via email, to our newly created storage container. Which brings us to adding the storage step in the Logic App, shown in Figure 2.25.

**Figure 2.25: Choose the action “Create blob” which will create a blob for us and fill it with the email attachment.**



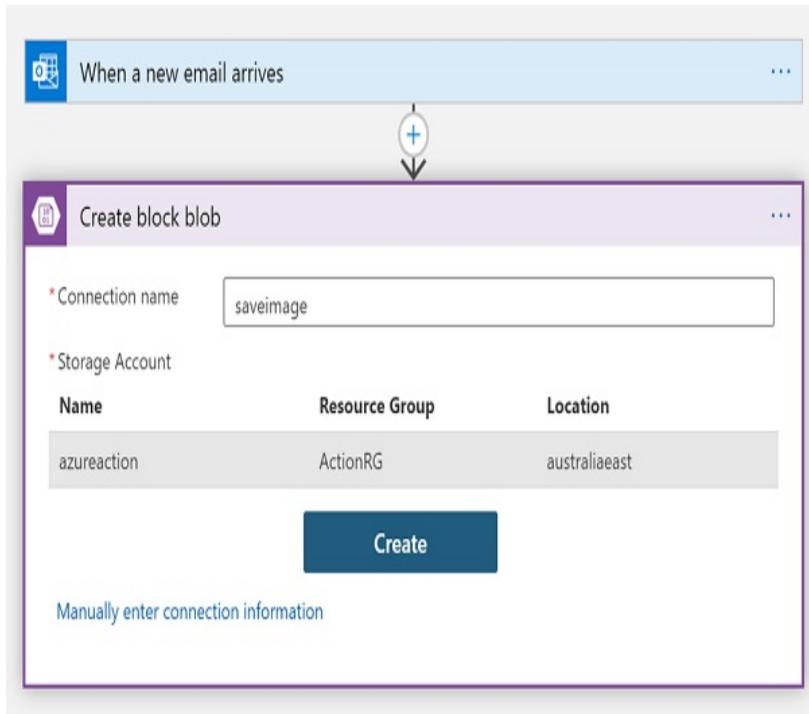
Search for 'blob' and choose the "Create blob" action. We will use this to copy the attachment to the blob.

## 2.2.12 Logic App operation – storing data from email to blob storage

We want to add a step to pass the attachment data from the email step into the container. Rather than a *trigger*, which is what we added to collect the email, we are now adding an *operation*. There are a LOT of operations available for Logic Apps. We are interested in creating a blob in the storage container, and the easiest way to find that is searching for “blob” as shown in Figure 2.25.

As you might have expected there are many options for working with blobs, and the operation we’re interested in is simply **Create block blob**. Click on it, and the step is added to our Logic App as shown in Figure 2.26.

Figure 2.26: Connect the action in the Logic App to the storage account.



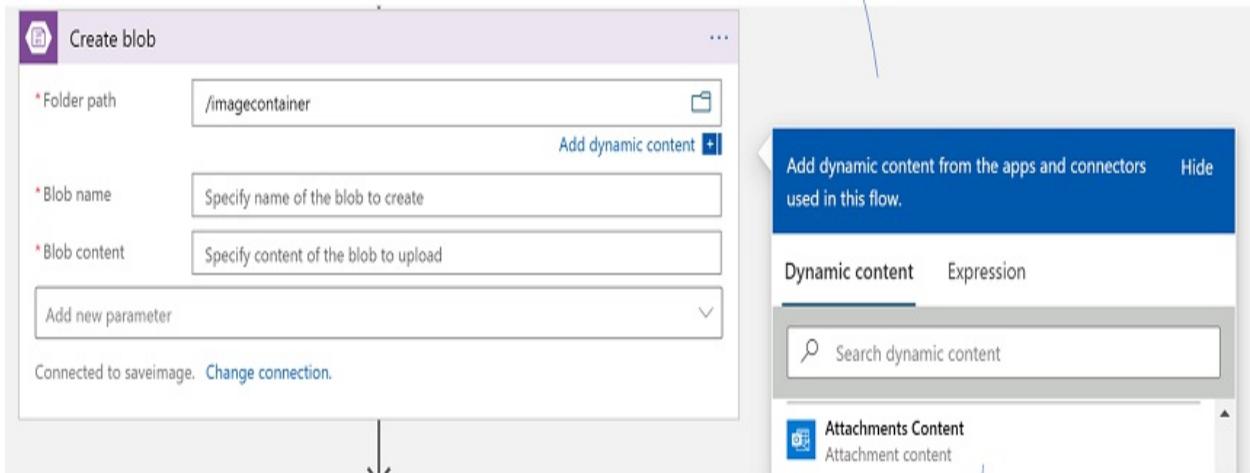
First, we need to connect to the storage account we created earlier.

It isn't enough to just add the step though. It needs to be configured both to collect the right data and to store it in the right place. Azure does most of the hard lifting for you and pre-populates a list of the available storage accounts in the current Azure subscription that the user creating the Logic App has access to. You then select the appropriate storage account, which is the one we created earlier in the chapter. Azure now creates an authenticated connection from the Logic App to the storage account, and you don't have to keep track of passwords, tokens, connection strings or any other configuration value. Neat.

Now that we know *where* to put the data, it's time to select *what* data to store there, which brings us to Figure 2.29, and an excellent example of the power and ease of Azure Logic Apps.

**Figure 2.27: Choose the Attachments Content dynamic field to iterate through the email attachments.**

The “Add dynamic content” dialog will open to choose the email content we’re working with.

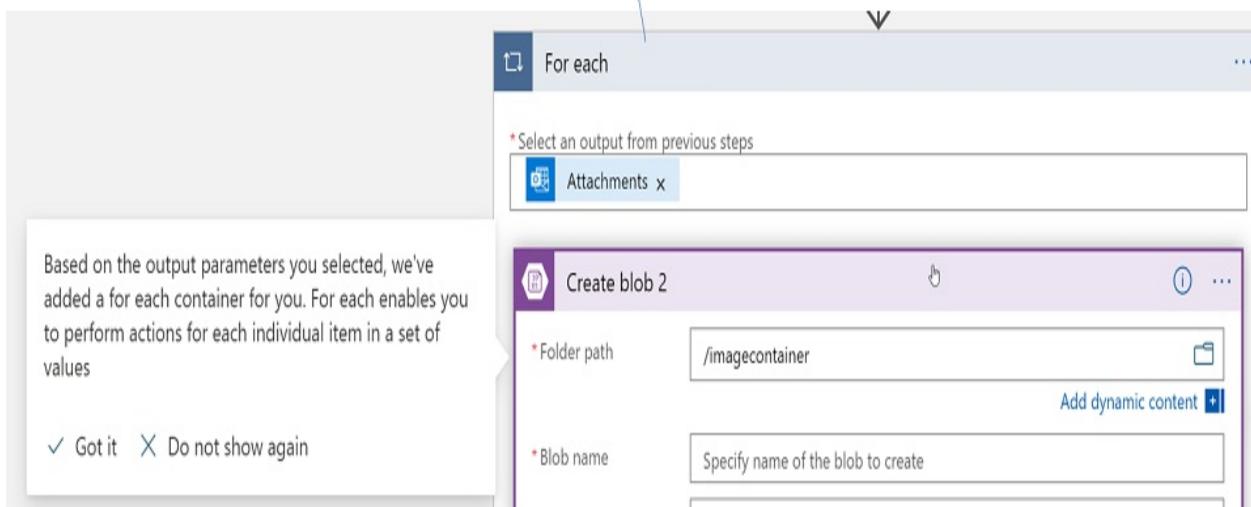


Choose the “Attachments Content” dynamic field as input.

First, choose the container within the storage account to use by clicking the folder icon in the **Folder path** field. The *Add dynamic content* dialog will show and when you choose the “Attachment Content” field as input, Logic apps will change the operation to a “For each” automatically (Figure 2.28), as there might be more than one attachment in the email. Clever.

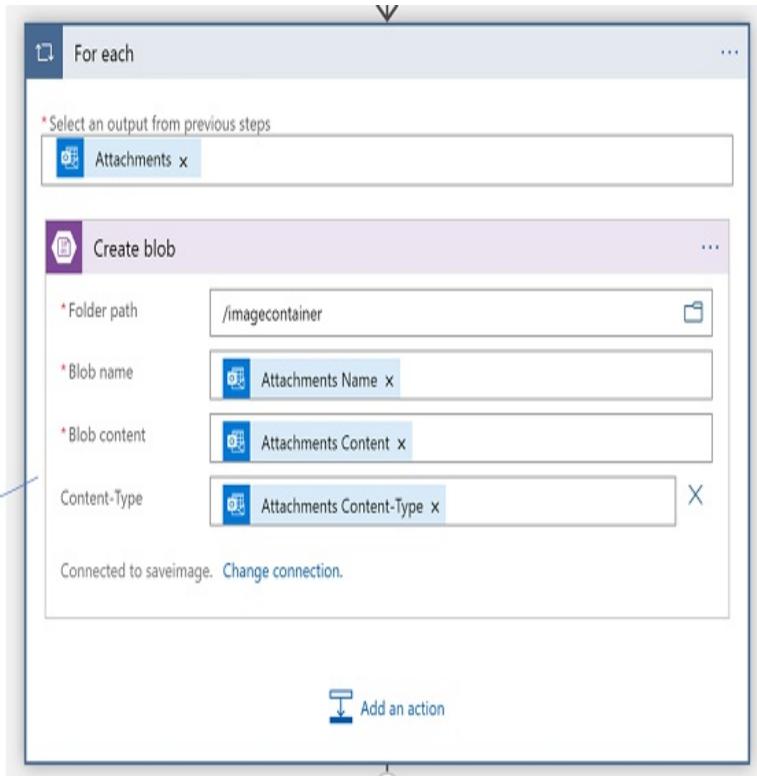
**Figure 2.28: The operation changes to a ‘For each’ when choosing a collection of input, such as Attachments.**

Choosing the email  
attachments as input  
changes the operation to 'For  
each' automatically



**Figure 2.29: Select the email properties to use.**

Select the email properties needed to create a new blob.



Second, select the email properties we want to map to the blob value in order to create a valid blob entry, shown in Figure 2.29. Click on each field and select the following dynamic content entries from the dialog shown:

- **Blob name:** Attachments Name
- **Blob content:** Attachments Content
- **Content type:** Attachments Content-Type

I mentioned before the power of Logic Apps, and Azure in general. With a few clicks, you have just created a workflow that connects to an email account, interrogates and filters those emails to only get the ones with attachments. The data is then mapped from the email directly into a container on a storage account, making it possible to store all the attachments for later use. You didn't have to write any code, manage any hardware or virtual machines, or set up any new users and passwords. That is the power of cloud. It lets you focus on the problem you are trying to solve, rather than all the “yak shaving”. I love it.

## 2.2.13 Testing the Logic App workflow

Make sure you save your work, and then it is time to send an email to test. Open your favorite email client, compose an email to the account you configured in the Logic App, attach an image file, and send it off. Within a couple of minutes the Logic App will run, find the email, copy the attachment into blob storage and let you know if it failed or succeeded (Figure 2.30).

**Figure 2.30: The complete Logic App has run for the first time.**

The screenshot shows the 'Runs history' tab selected in the Azure Logic Apps interface. At the top, there are tabs for 'Get started', 'Runs history' (which is underlined), 'Trigger history', and 'Metrics'. Below the tabs is a search bar with dropdowns for 'All', 'Start time earlier than', 'Pick a date', and 'Pick a time'. A note below the search bar says 'Specify the run identifier to open monitor view directly'. The main table lists one run entry:

Status	Start time	Identifier	Duration
Succeeded	4/21/2021, 9:45 PM	08585826013450921347863238625CU21	1.6 Seconds

A handwritten annotation in blue ink points from the bottom left towards the table, with the text 'The first successful run of the Logic App.'

From now on the Logic App will continue to run until you delete or disable it. Azure will make sure it is running and performing. If you want to be notified of any issues or failures, you can set up alerts, but we will get to alerts later in the book.

When you have a successful run of the Logic App, navigate to the storage account and confirm you see the email attachment there (Figure 2.31).

**Figure 2.31: The image file received on email and now stored in the storage account.**

The screenshot shows the Azure Storage Blob details page for a file named 'test\_image.jpg'. The left sidebar displays the blob's properties: 'Authentication method: Access key (Switch to Azure AD User Account)', 'Location: imagecontainer', and a search bar for blobs by prefix. The main content area shows the blob's properties in a table format:

Properties	Value
URL	<a href="https://azureaction.blob...">https://azureaction.blob...</a>
LAST MODIFIED	4/21/2021, 9:45:42 PM
CREATION TIME	4/21/2021, 9:45:42 PM
VERSION ID	-
TYPE	Block blob
SIZE	2.3 MiB
ACCESS TIER	Hot (Inferred)
ACCESS TIER LAST MODIFIED	N/A
SERVER ENCRYPTED	true
ETAG	0x8D904BAFE404C09
CONTENT-TYPE	image/jpeg
CONTENT-MD5	PvGAeadH3KLk3UMrXPqXXA==
LEASE STATUS	Unlocked
LEASE STATE	Available
LEASE DURATION	-
COPY STATUS	-
COPY COMPLETION TIME	-

A blue 'Undelete' button is located at the bottom of the properties table.

The text on the left side of the screenshot reads: "The test\_image.jpg image received via email, and now stored in the storage account."

You can see all of the properties of the file, such as its public URL for retrieval, type, size and much more. Almost there! Let's move on to the last part of our app infrastructure.

## 2.2.14 What is an Azure Function?

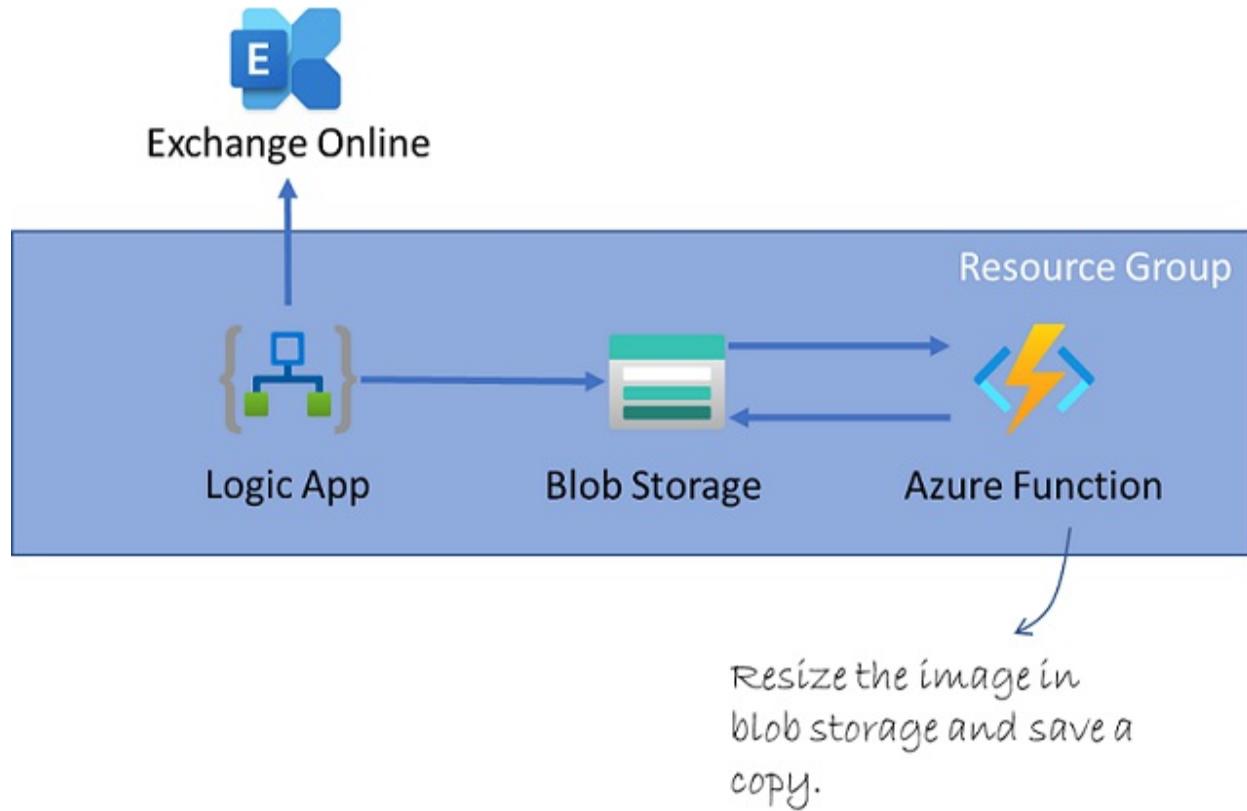
Serverless. It's what the cool kids do. Let's take a second and define serverless. It is when the cloud provider allocates resources on demand as the service runs. When the service stops running, the resource is deallocated. We are already creating a serverless application by using Logic Apps, but let's take it one step further and create a Function App, which is the container for

your Azure Functions. Think of a Function App as the VM that runs your Azure Function. You can have multiple Azure Functions in a single Function App.

Often thought of as the “original” serverless service, Azure Functions are single-purpose compute processes that run entirely within Azure. They’re called functions because that is all they do; a single function. It is not uncommon to build entire applications from hundreds or even thousands of Azure Functions, each responsible for a single purpose. This makes it simple to build, debug and maintain. At least in theory.

We are using an Azure Function for our online photo upload application, specifically for the image resizing part as shown in Figure 2.32. It is a single function of work needed to complete the application.

**Figure 2.32: We are now at the “resizing images with Azure Functions” part of the application.**

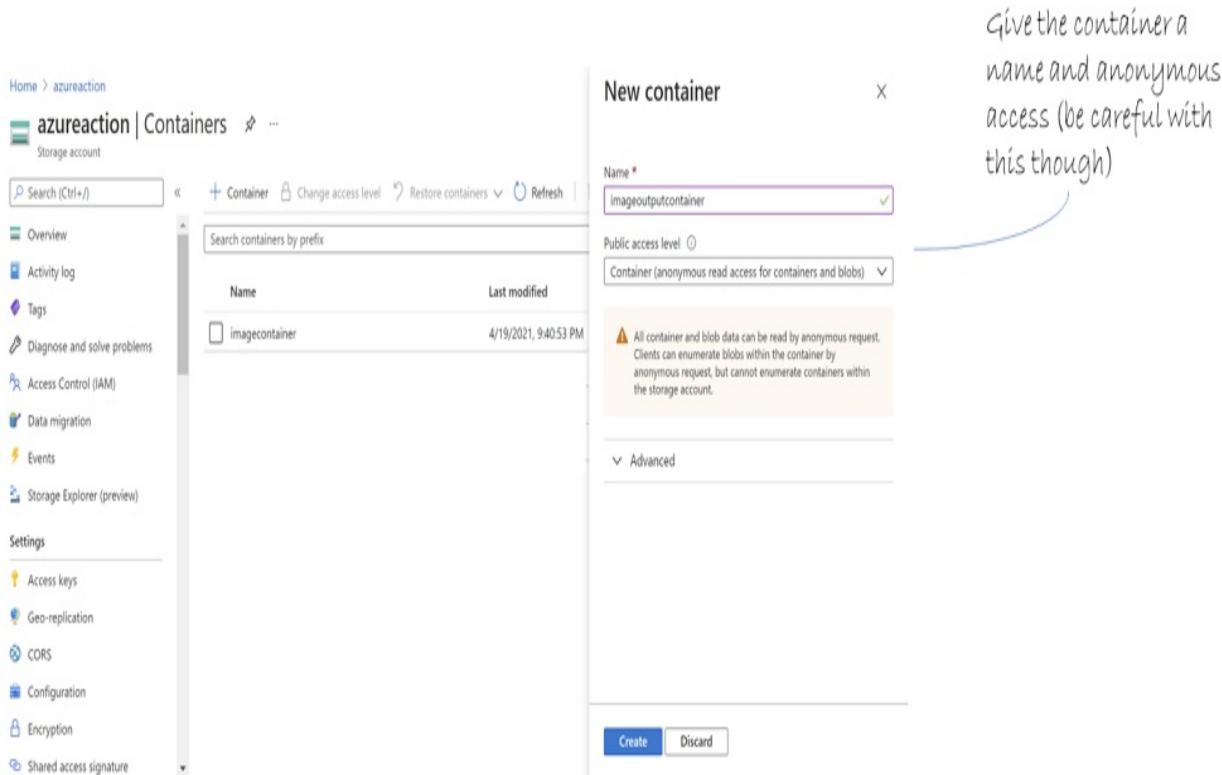


## 2.2.15 Creating a storage container for resized images

Before we can start resizing any images, we need somewhere to put the resized images. After all, we can't just put them into the drawer of forgotten resized images. The obvious option is another container in the blob storage account we created just a moment ago. Let's do that then! Go to the *azureaction* storage account created earlier and click the “+ Container” button.

This is the exact same approach as we used in Figure 2.14, so it should be familiar. You can create as many containers as you like in the storage account<sup>[3]</sup>. Name the new container something like *imageoutputcontainer* and set the access to public and anonymous (Figure 2.33). As mentioned earlier, take great care in choosing the right scope for access to the container. In this case we are choosing public, but that may not always be appropriate.

**Figure 2.33: Decide on the container name and the access level.**

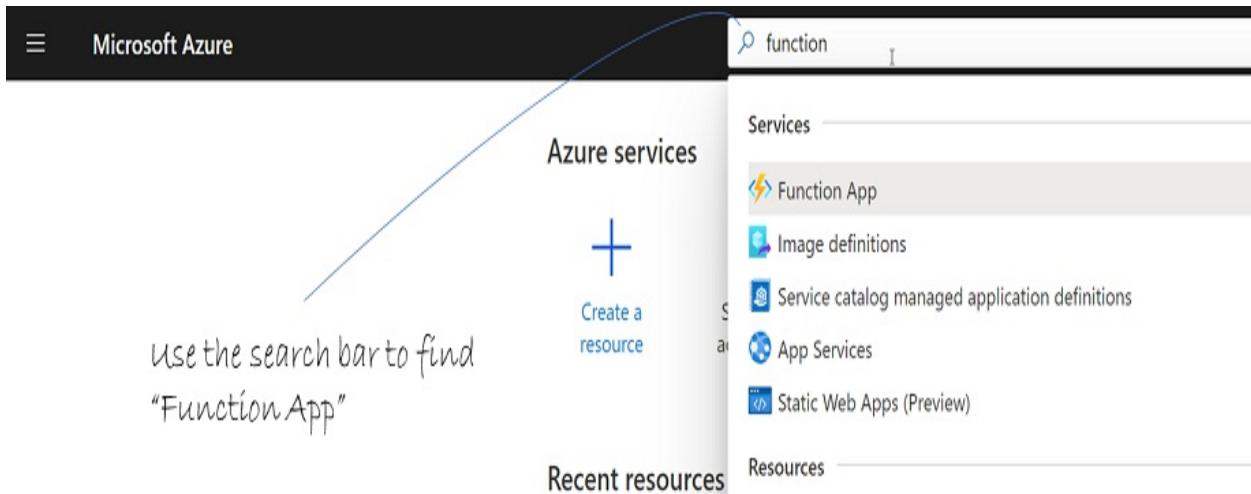


## 2.2.16 Creating a Function App

Before we can process any of the images uploaded and stored in the blob

storage, we'll have to create the Function App in Azure. A Function App is the service which the actual functions live within. Use the search in the Azure Portal to find the Function App section (Figure 2.34).

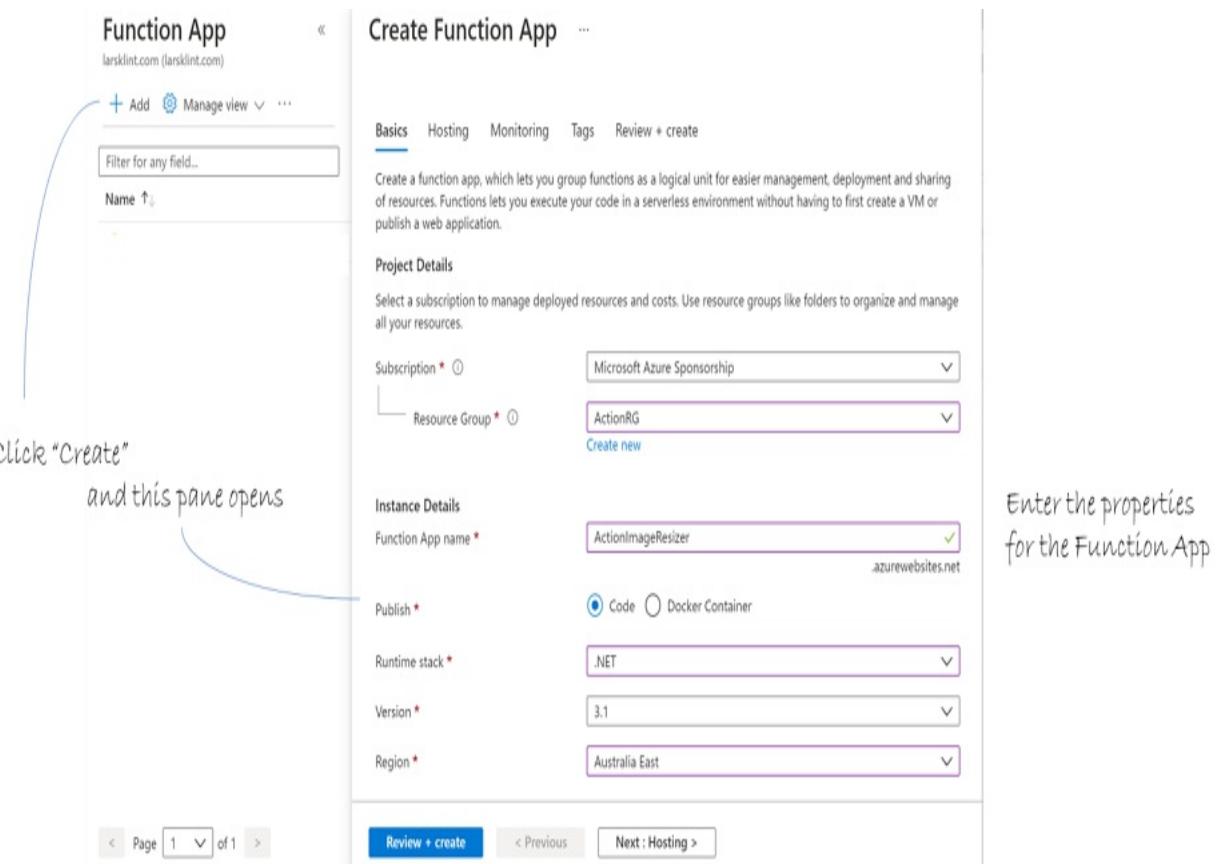
**Figure 2.34: Find the Function App section in the Azure Portal.**



Click through to the Azure Function App overview, which looks like most other Azure service overviews. There are differences between the services, but the experience is consistent and if you haven't already, getting used to the Azure Portal doesn't take long.

To create a new Azure Function, click on the “+ Add” button and enter the properties for the function as shown in Figure 2.35.

**Figure 2.35: Add the properties for the Function App**



Use these values for the basic settings.

- **Subscription:** Choose the same subscription used for the other services in this chapter.
- **Resource Group:** Choose the same resource group as for the other services in this chapter.
- **Function App Name:** ActionImageResizer
- **Publish:** Code
- **Runtime stack:** .NET
- **Version:** 3.1
- **Region:** Australia East. This needs to be the same region as you created the storage account in.

Once you have the properties filled out, click on “Next: Hosting” to configure the mechanics of the Function App (Figure 2.36).

**Figure 2.36: Set up the hosting for the Azure Function App.**

Basics   Hosting   Monitoring   Tags   Review + create

Storage

When creating a function app, you must create or link to a general-purpose Azure Storage account that supports Blobs, Queue, and Table storage.

Storage account \*

azurereaction

Create new

Operating system

The Operating System has been recommended for you based on your selection of runtime stack.

Operating System \*

Linux  Windows

Plan

The plan you choose dictates how your app scales, what features are enabled, and how it is priced. [Learn more](#)

Plan type \* ⓘ

Consumption (Serverless)

Review + create   < Previous   Next : Monitoring >

Every Function App needs to use a storage account for various tasks. These can include:

- Maintaining bindings for states and triggers, which is metadata for using the features.
- A file share for storing and running the function app code, as well as logs and other metadata. After all the code files themselves need to live somewhere.
- Queue and table storage for task hubs in Durable Functions.

If you want to use an existing storage account, it is highly recommended it is in the same region as the Function App. This ensures performance and is also enforced when creating the Function App through the Azure Portal. The operating system is for the virtual machine which runs the function, and which you will never see or interact with. Leave the plan as consumption, which is not only the serverless approach, but also gives you one million free

function executions per month. ONE MILLION! Amazing.

## 2.2.17 Monitoring Function Apps

Click “Next: Monitoring” to get to the next step in the wizard, which, surprisingly, has to do with monitoring and looks like Figure 2.37.

**Figure 2.37: Enable Application Insights for the Function App**

The screenshot shows the 'Create Function App' wizard in the 'Monitoring' step. The top navigation bar includes 'Basics', 'Hosting', 'Monitoring' (which is underlined), 'Tags', and 'Review + create'. A descriptive text explains that Azure Monitor application insights is an Application Performance Management (APM) service for developers and DevOps professionals, enabling automatic monitoring of the application to detect performance anomalies and provide diagnostic tools. Below this, there's a section titled 'Application Insights' with a question 'Enable Application Insights \*'. Two radio buttons are present: 'No' (unchecked) and 'Yes' (checked). To the right of this is a dropdown menu showing '(New) ActionImageResizer (Australia East)' with a 'Create new' link below it. Further down, a 'Region' field is set to 'Australia East'. At the bottom of the screen, there are three buttons: 'Review + create' (in blue), '< Previous', and 'Next : Tags >'.

Application Insights is a brilliant tool for optimizing performance and

monitoring the application. You'll learn more about Application Insights later in the book, but for now enable it. Then click the "Review + create" button. Almost there.

You will see the customary sanity check before you are allowed to create the resource. At times you can get an error here if you have chosen an incorrect option or service, but most of the time the wizard (shall we call him Gandalf?) keeps you out of trouble. All there is left to do is click "Create" then go to the Function App dashboard (Figure 2.38).

**Figure 2.38: Function App Dashboard**

The public URL for the Function App.

The screenshot shows the Azure Functions Overview screen for the app 'ActionImageResizer'. The top navigation bar includes 'Search (Ctrl+P)', 'Browse', 'Refresh', 'Stop', 'Restart', 'Swap', 'Get publish profile', 'Reset publish profile', 'Download app content', 'Delete', and 'Send us your feedback'. On the left, a sidebar lists 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Security', 'Events (preview)', 'Functions' (selected), 'App keys', 'App files', 'Proxies', 'Deployment' (selected), 'Deployment slots', 'Deployment Center', 'Settings' (selected), 'Configuration', 'Authentication', 'Authentication (classic)', 'Application Insights', 'Identity', and 'Backups'. The main content area has a heading 'Click here to access Application Insights for monitoring and profiling for your app.' Below this is the 'Essentials' section with details: Resource group (change) : ActionRG, Status : Running, Location : Australia East, Subscription (change) : Microsoft Azure Sponsorship, Subscription ID : 694ae2bd-ea92-4302-8069-db43c3533f9c, Tags (change) : Click here to add tags, URL : https://actionimageresizer.azurewebsites.net, Operating System : Windows, App Service Plan : ASP-ActionRG-a2b4 (Y1: 0), Properties : See More, and Runtime version : 3.0.15571.0. At the bottom, tabs for 'Metrics' (selected), 'Features (9)', 'Notifications (1)', and 'Quickstart' are shown, followed by two charts: 'Memory working set' and 'Function Execution Count'.

You now have an Azure Function App up and running. Essentially this is running on an app service plan, but it is “serverless” so you don’t have to worry about any infrastructure. It is running on “someone else’s server”. As Figure 2.38 shows there is also a public URL for the Function App on the *Overview* screen, which is part of what is needed to access Functions publicly. In this case, because the name of the Function app is **ActionImageResizer**, the URL is *actionimageresizer.azurewebsites.net*. From this you can deduce that all Function Apps need to have a unique name, as URLs are obviously unique. And because the Function App has a

URL, you also get a lovely web frontend (Figure 2.39).

**Figure 2.39: The web frontend for a brand new Azure Function. Pretty, right?**



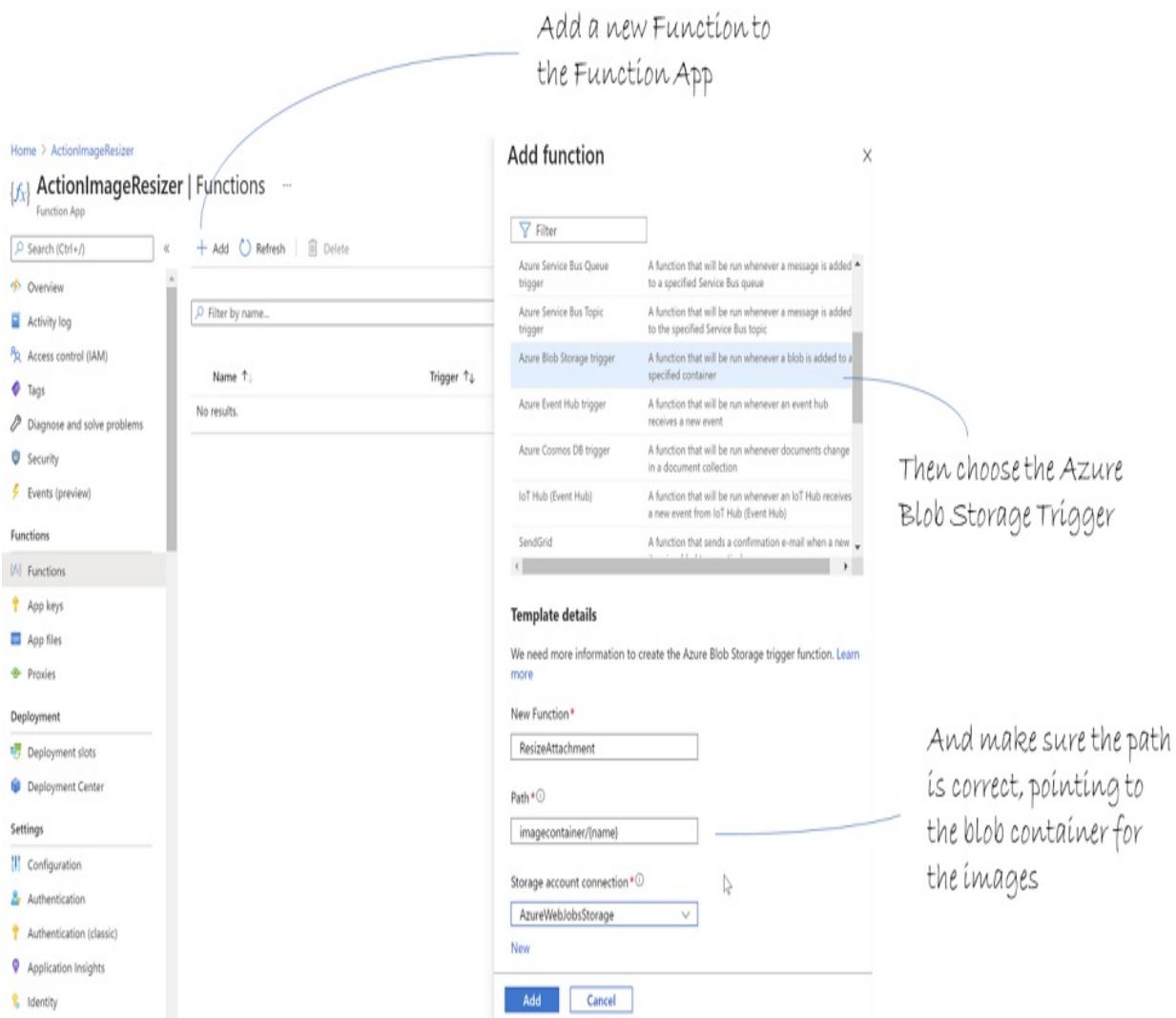
As I mentioned before, the Azure Function App is the container that the Azure Functions live inside. You get all the tools for managing the app,

which is a similar toolset to what you get to work with in a standard App Service.

## 2.2.18 Creating an Azure Function

By now, you are likely wondering “hey Lars, where is the actual Function to resize the images?” Fair enough. We are about to create the function itself. Just like Figure 2.40 shows, go to the **Functions** section and then click “+ Add”.

**Figure 2.40: Create a new function and use the blob storage trigger template.**



When you add a new function through the Azure Portal you can choose from

a range of templates that represent common ways of using Azure Functions. From this list choose the *Azure Blob Storage Trigger*, and then fill in the template details:

- **New Function** (name): ResizeAttachment
- **Path**: imagecontainer/{name}
- **Storage account connection**: this will be prefilled to the connection name for the storage account created earlier.

Of the three parameters, the **Path** is the crucial one: *imagecontainer/{name}*. This is the path relative to the storage account itself. We want to use the *imagecontainer* blob container, and the *{name}* is a variable, as indicated by the curly brackets. This will be the name of the file being processed, and the function can reference that image file by using the variable *name*. Make sense? The path is where the trigger will hook into and watch for new blobs.

Click “Add” and you get a brand spanking new shiny Azure Function, ready to go (Figure 2.41).

**Figure 2.41: Azure Function overview.**

Home > ActionImageResizer >

{fx} **ResizeAttachment** ✎ ... X

Function

Search (Ctrl+ /) « ✓ Enable ⚙ Disable 🗑 Delete 🌐 Get Function Url ⌛ Refresh

[f] Overview JSON View

Developer

Function app  
ActionImageResizer Application Insights  
Status  
Enabled ActionImageResizer

Code + Test

Integration

Monitor

Function Keys

Resource group (change)  
ActionRG

Subscription (change)  
Microsoft Azure Sponsorship

Subscription ID  
694ae2bd-ea92-4302-8069-db43c3533f9c

Total Execution Count

Successful Execution Count

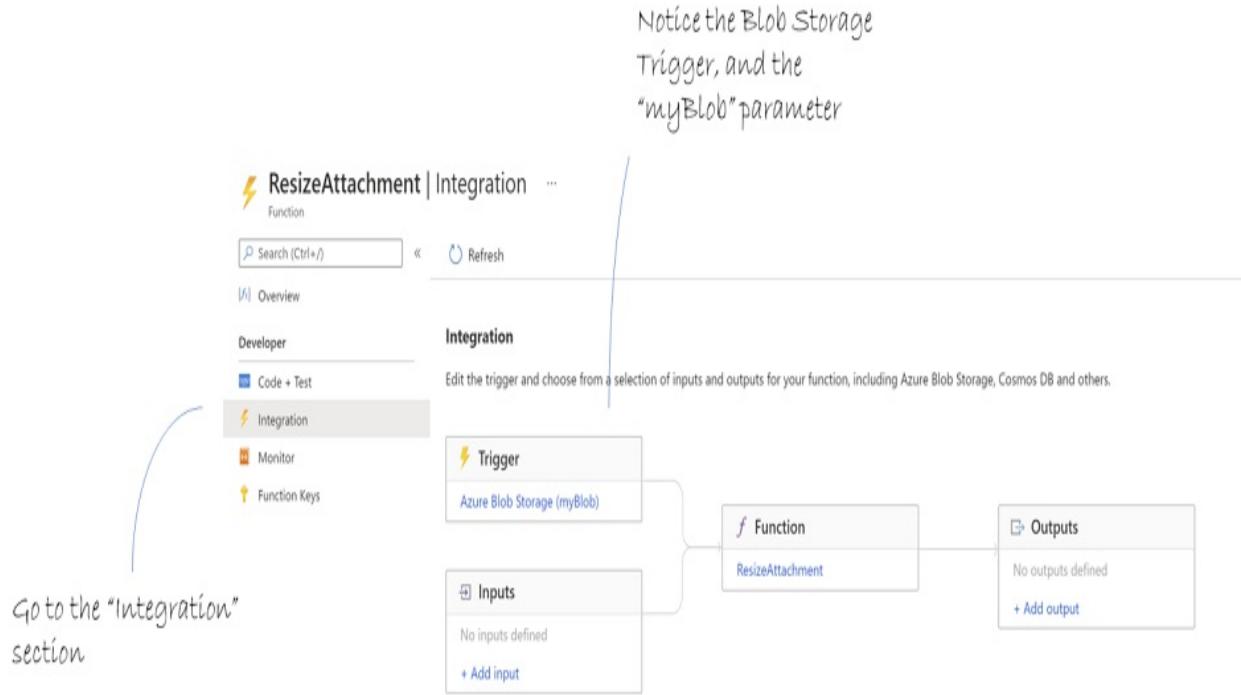
The screenshot shows the Azure Functions portal interface. At the top, it displays the function name 'ResizeAttachment'. Below the title, there's a toolbar with buttons for 'Search (Ctrl+ /)', 'Enable', 'Disable', 'Delete', 'Get Function Url', and 'Refresh'. On the left, a sidebar lists navigation options: Overview, Developer, Code + Test, Integration, Monitor, and Function Keys. Under 'Developer', it shows the function app name 'ActionImageResizer', status 'Enabled', resource group 'ActionRG', subscription 'Microsoft Azure Sponsorship', and a specific subscription ID. Two cards at the bottom show execution counts: 'Total Execution Count' (120) and 'Successful Execution Count' (120). The overall layout is clean and organized, typical of the Azure portal.

While we now have an actual function inside a Function App, it doesn't do much yet, apart from logging the files being inserted into the blob storage container.

## 2.2.19 Adding integrations to an Azure Function

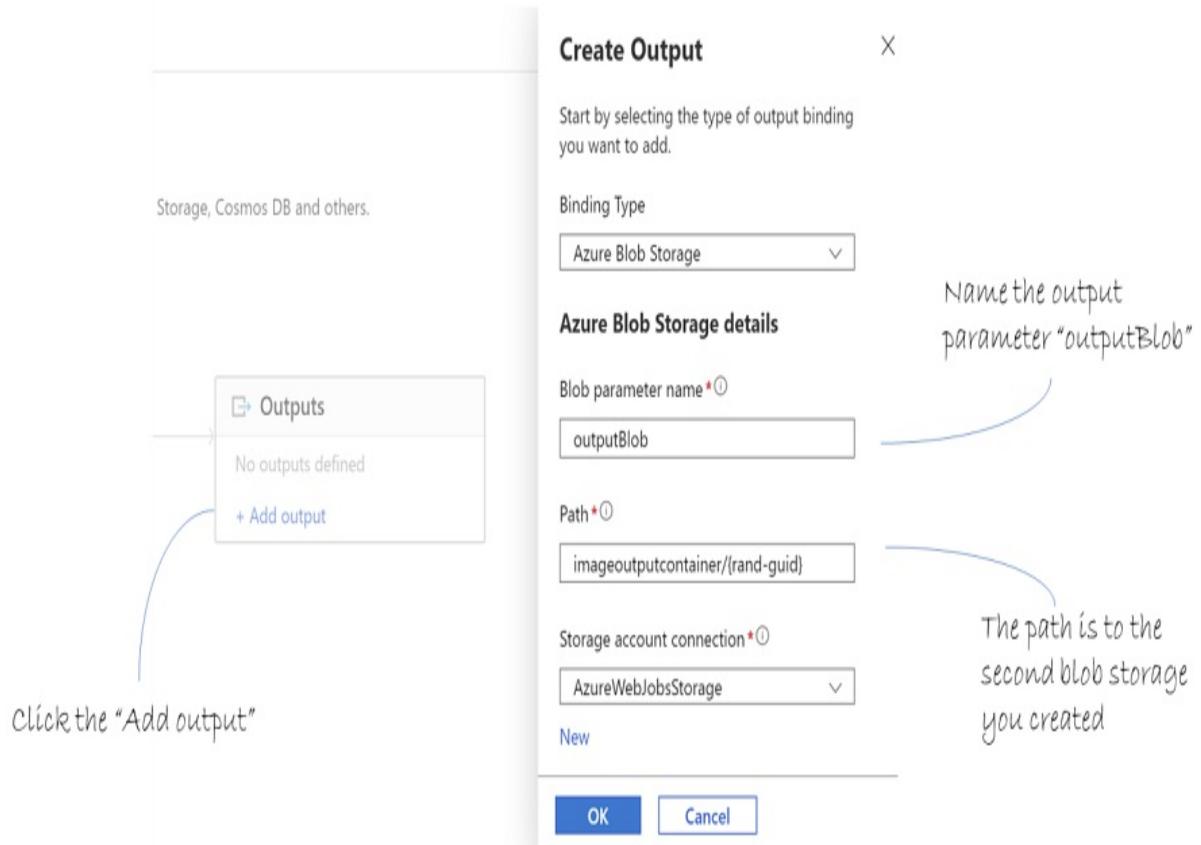
Before you get to bathe yourself in the glorious code of an Azure Function, we need to set up a couple of integrations first, by going to the “Integration” section of the function (Figure 2.42).

**Figure 2.42: The Integration section of the Function.**



The Integration overview is an excellent visualization of how the data flows around and through a particular Azure Function. Because we chose the Azure Blob Storage Trigger, a trigger has been created for us, as well as the function itself. When a new file attachment from an email is inserted into the blob storage, the trigger will....well, trigger and let the function know through the `{name}` parameter. Before we get to the function code, let's create an output for the resized image (Figure 2.43), so it can be put back into blob storage.

**Figure 2.43: Create an output for the function that targets the “imageoutputcontainer” blob storage.**



Click on the *Add output* link in the **Outputs** box, which will open the **Create Output** sidebar. Enter the following values for the Output properties.

- **Binding Type:** Azure Blob Storage
- **Blob Parameter Name:** outputBlob
- **Path:** imageoutputcontainer/{rand-guid}
- **Storage account connection:** AzureWebJobsStorage

When you choose the **Binding Type** as *Azure Blob Storage* the rest of the output parameters are updated. The **Blob parameter name** is the variable name that will be returned from the function. Just like before the **Path** is the relative path in the storage account for where the resized image is being placed. The variable *{rand-guid}* is the name of the resized image, and in this case it will just get a random GUID<sup>[4]</sup> as file name.

Click “OK” to create the output, and then we are ready to look at the function itself.

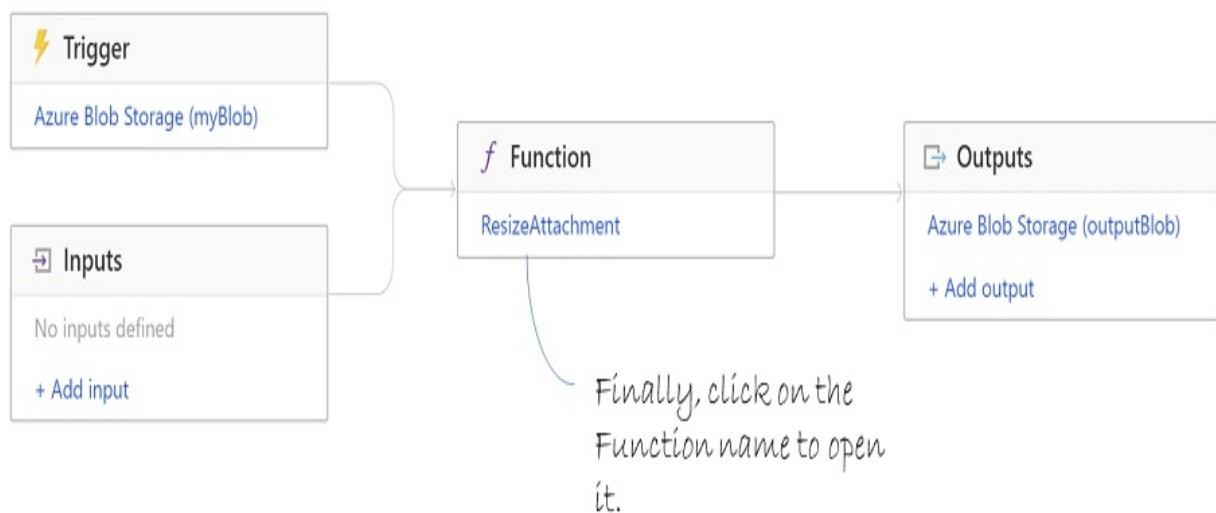
## 2.2.20 Writing code in a function

Start by clicking on the function name on the Integration visualization (Figure 2.44).

Figure 2.44: Now we have input and output open up the Function code.

### Integration

Edit the trigger and choose from a selection of inputs and outputs for your function, including Azure Blob Storage, Cosmos DB and others.



This is just one of the many ways you can get to the Azure Function. There is now a path for the data to go from trigger to function to output. We are going to massage the image a bit and reduce its size for the web album. This is the part where we first look at the code of an Azure Function. If you haven't been exposed to the C# programming language before, follow along for now. You can also use other languages like Java, Python and Node.js. It is okay not to understand all the parts, as we will get back to more code further on in the book. For now, let's have a first look at the code (Figure 2.45).

Figure 2.45: The template Azure Function code.

```
public static void Run(Stream myBlob, string name, ILogger log)
{
    log.LogInformation($"C# Blob trigger function Processed blob\n Name:{name}\n Size: {myBlob.Length} Bytes");
}
```

This is the actual code for the Function. Not much, right?

The entire code block for the function is:

```
public static void Run(Stream myBlob, string name, ILogger log)
{
    log.LogInformation($"C# Blob trigger function Processed b
```

All this does is output a log of the name and size of the blob being inserted and thus triggering the function to run. Not very exciting, not very productive. We need to update the code to resize the *myBlob* input parameter. Replace the template code in the file *run.csx* with the delicious code shown in Figure 2.46.

**Figure 2.46: C# code to resize an image.**

Make use of the SixLabors  
ImageSharp library for  
manipulating an image.

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.Formats.Jpeg;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;

public static void Run(Stream myBlob, string name, ILogger log, Stream outputBlob)
{
    log.LogInformation($"C# Blob trigger function Processed blob\n Name:{name} Size:
(myBlob.Length) Bytes");
    using (var image = Image.Load(myBlob))
    {
        image.Mutate(c => c.Resize(100, 100));
        image.SaveAsJpeg(outputBlob);
    }
}
```

Use two streams of data, *myBlob* and  
*outputBlob*, to work with the image.

Resize the image to 100x100 pixels. Yes, this  
might skew the image. We're not building the next  
Facebook here though.

This chapter isn't aimed at teaching you how to code but let me point out a few main points of what the code does.

- The *SixLabors ImageSharp* library is a packaged bit of code that will manipulate an image for you easily. In general, using tested and stable libraries is a good thing to ensure your code is as clean, simple and robust as possible
- Two streams of data, *myBlob* and *outputBlob*, is the input data of the original image and the output for the resized image, respectively.
- The image is resized to 100x100 pixels. This will likely skew the image, but we are after simple code, not perfection.

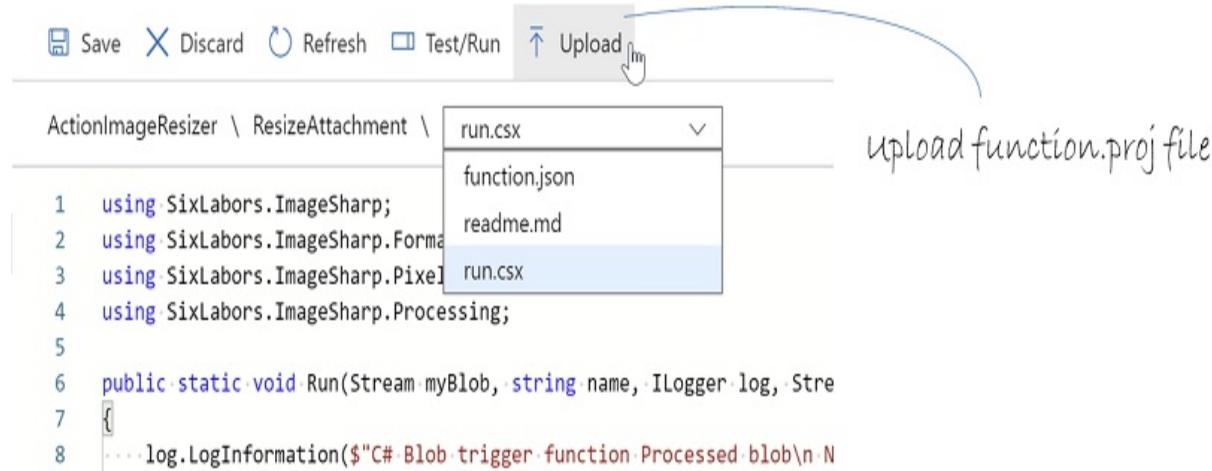
Once you have replaced the code with that in Figure 2.46, click “Save”.

## 2.2.21 Referencing code libraries in Azure Functions

There is one more thing to do, before the function can process the emailed image attachments though. The function doesn't currently know how to compile and use the SixLabors library, so we need to tell it that by uploading

a reference file (Figure 2.47).

**Figure 2.47: Upload the function.proj file.**



You can upload any file you want to your function, but often it isn't needed. The whole point of functions is to keep them light and simple. Adding lots of files defeats that purpose. However, we do need to tell the function where to find the *SixLabors.ImageSharp* library, which is done in a file called *function.proj* as shown below.

```
<Project Sdk="Microsoft.NET.Sdk">
    <PropertyGroup>
        <TargetFramework>netstandard2.0</TargetFramework>
    </PropertyGroup>
    <ItemGroup>
        <PackageReference Include="SixLabors.ImageSharp" Version="1.0.3" />
    </ItemGroup>
</Project>
```

This small XML file has three main parts to it

- A definition of the main Software Development Kit (SDK) for the project, or function, being .NET SDK.
- A target framework, which is .NET Standard 2.0.
- A package reference to SixLabors.ImageSharp version 1.0.3.

You can only have one target framework for a function, but you can have multiple package references, in case you have to include multiple libraries.

This is all that is needed for the function to compile successfully (Figure 2.48).

**Figure 2.48: Saving a file in the editor triggers compilation of the function.**

The screenshot shows the Azure Functions developer tools interface. At the top, there are buttons for Save, Discard, Refresh, Test/Run, and Upload. Below that is a navigation bar showing the path ActionImageResizer \ ResizeAttachment \ function.proj. The main area contains the following C# code:

```
1 <Project Sdk="Microsoft.NET.Sdk">
2     <PropertyGroup>
3         <TargetFramework>netstandard2.0</TargetFramework>
4     </PropertyGroup>
5     <ItemGroup>
6         <PackageReference Include="SixLabors.ImageSharp" Version="1.0.3" />
7     </ItemGroup>
8 </Project>
```

Handwritten notes on the left side of the code editor say "When you click 'Save', the Function is compiled." A blue curved arrow points from the word "Save" in the notes to the "Save" button at the top of the editor.

At the bottom of the screen is a log window titled "Logs". It shows the following log entries:

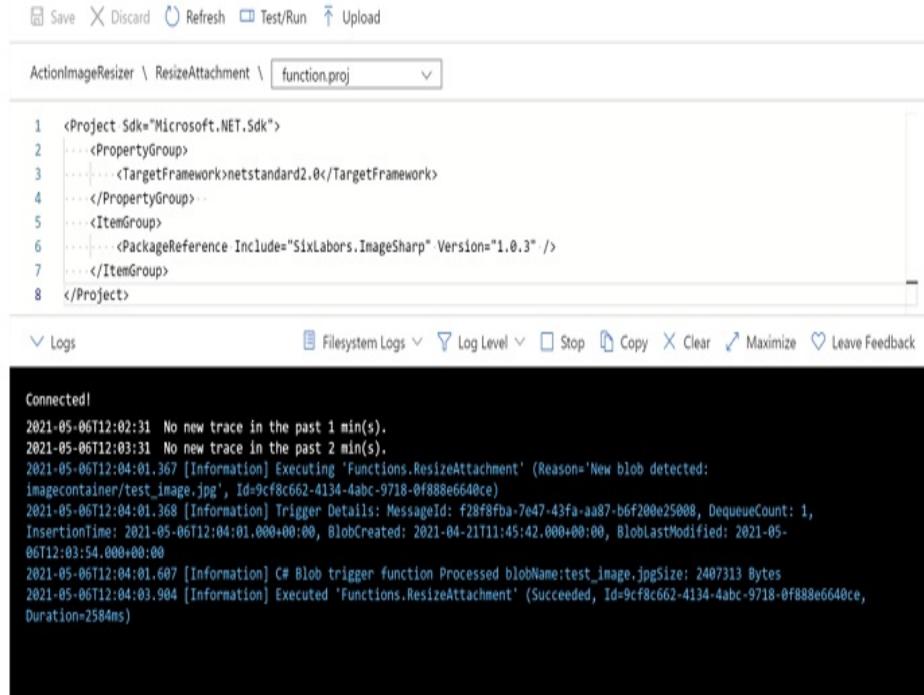
```
2021-05-06T11:59:25.927 [Information] Starting packages restore
2021-05-06T11:59:25.912 [Information] Starting packages restore
2021-05-06T11:59:27.850 [Information] Determining projects to restore...
2021-05-06T11:59:28.383 [Information] Determining projects to restore...
2021-05-06T11:59:30.624 [Information] Restored C:\local\Temp\9146bfef-f3cc-4ae0-ad97-d8a2e22d2273\function.proj (in 2.15 sec).
2021-05-06T11:59:32.228 [Information] Restored C:\local\Temp\aa394780-8a09-402b-adbf-7c42a56e78fd\function.proj (in 3.15 sec).
2021-05-06T11:59:32.572 [Information] Packages restored.
2021-05-06T11:59:32.881 [Information] Script for function 'ResizeAttachment' changed. Reloading.
2021-05-06T11:59:33.166 [Information] Compilation succeeded.
```

Every time you save a new or updated file, the whole function is compiled, and the output window at the bottom of the screen shows you any compilation errors there might be. The function will then run whenever triggered. For this application, the function is triggered by a file being inserted into blob storage.

Grab your favorite email client and send off an email with an image attached. Make sure you send it to the email address you configured in the Logic App back in Figure 2.23. When the Logic App inserts the email attachment into the blob storage, the blob storage trigger fires and the function runs (Figure 2.49).

**Figure 2.49: The function is triggered when a new image is inserted into the storage account.**

When an image is inserted  
into the storage account,  
the function runs.



The screenshot shows the Azure Functions developer tools interface. At the top, there are navigation buttons: Save, Discard, Refresh, Test/Run, and Upload. Below the buttons, the project path is displayed as ActionImageResizer \ ResizeAttachment \ function.proj. The main area contains the project's .csproj file code:

```
1 <Project Sdk="Microsoft.NET.Sdk">
2     <PropertyGroup>
3         <TargetFramework>netstandard2.0</TargetFramework>
4     </PropertyGroup>
5     <ItemGroup>
6         <PackageReference Include="SixLabors.ImageSharp" Version="1.0.3" />
7     </ItemGroup>
8 </Project>
```

Below the code editor is a log viewer titled 'Logs' with the following content:

```
Connected!
2021-05-06T12:02:31 No new trace in the past 1 min(s).
2021-05-06T12:03:31 No new trace in the past 2 min(s).
2021-05-06T12:04:01.367 [Information] Executing 'Functions.ResizeAttachment' (Reason='New blob detected: imagecontainer/test_image.jpg', Id=9cf8c662-4134-4abc-9718-0f888e6640ce)
2021-05-06T12:04:01.368 [Information] Trigger Details: MessageId: f28f8fba-7e47-43fa-aa87-b6f200e25008, DequeueCount: 1, InsertionTime: 2021-05-06T12:04:01.000+00:00, BlobCreated: 2021-04-21T11:45:42.000+00:00, BlobLastModified: 2021-05-06T12:03:54.000+00:00
2021-05-06T12:04:01.607 [Information] C# Blob trigger function Processed blobName:test_image.jpgSize: 2407313 bytes
2021-05-06T12:04:03.904 [Information] Executed 'Functions.ResizeAttachment' (Succeeded, Id=9cf8c662-4134-4abc-9718-0f888e6640ce, Duration=2584ms)
```

When the trigger fires off the function, a reference to the newly inserted blob is passed to the function as well. And that is why the function can then create a resized copy of the image and insert it into the second blob storage we created (Figure 2.50).

**Figure 2.50: The resized image in the blob container for output.**

The second blob container for resized images.

Properties	Value
URL	<a href="https://azureaction.blob...">https://azureaction.blob...</a>
LAST MODIFIED	5/6/2021, 10:04:03 PM
CREATION TIME	5/6/2021, 10:04:03 PM
VERSION ID	-
TYPE	Block blob
SIZE	7.47 KIB
ACCESS TIER	Hot (Inferred)
ACCESS TIER LAST MODIFIED	N/A
SERVER ENCRYPTED	true
ETAG	0x8D910870AE3BF3E
CONTENT-TYPE	application/octet-stream
CONTENT-MDS	8dxlPRuAcGbwD9hPRXQt1w==
LEASE STATUS	Unlocked
LEASE STATE	Available
LEASE DURATION	-
COPY STATUS	-
COPY COMPLETION TIME	-

We made it to the end! That was quite the journey through Azure and various services. You now have a fully automated application that will collect emails, copy attachments, resize them and add them to a storage for further use. Neat!

## 2.3 Exploring the Solution in Azure

Give yourself a pat on the back and do a little dance, because you now have a complete Azure application running in the cloud. Yep, pretty cool. While the solution is fresh in your mind, let's look at some of the features and advantages you get for using Azure to create the application.

### 2.3.1 Scaling

When the service takes off and you need more computing power to handle all

the incoming email attachments, Azure will ensure any Logic App can handle up to 300,000 requests every 5 minutes. There are other limits that come into effect as well, such as data throughput and concurrent inbound calls, but at no point do you have to buy more hardware or provision more virtual machines. The scaling of Logic Apps is Azure's responsibility, not yours. This is the same story with Azure Functions, which will continue to spin up more instances of the function to cope with the increased number of images needing processing.

Like you learned in chapter 1, being able to scale an application is one of the primary benefits of using cloud computing. Resources are there when you need them, and when you don't, they scale right back. Where traditional data centers and company infrastructure only scales to the capacity of the hardware you have bought, cloud computing has almost infinite resources. Of course, it's not "infinite", but it might as well be. And if you do create "The Devourer of Resources" application that would bring Azure to its knees, that is why there are limits on most resources. If you hit those limits, you most likely have other problems with your application.

### 2.3.2 Performance

Performance used to be directly linked to the power of the hardware, how much memory you had on the server and how fast the internet connection was. Not with cloud computing. Performance isn't a direct indication of your hardware budget. Performance is both a fundamental cloud component, as well as something that can be purchased when needed *and* for the time needed. This allows even small teams to have the computing power they need at any given time.

Translating that to the image resizing application we created, the Azure Function component will only be more expensive when we process a LOT more images. Not only do you get 1 million executions for free, but an additional million executions is \$0.20. Yep, 20 cents. Try and do *anything* else for 20 cents, let alone process large amounts of images. When we need the performance, it is just there. There will be a whole lot more emphasis on performance throughout the book.

### **2.3.3 Fault Tolerance**

Applications are only valuable when they run and perform the tasks they were made for. When hardware fails, new code deployed has critical bugs or John presses the wrong switch on the admin tool (it's always John), which results in the application stopping, the consequences can be expensive. Expensive in terms of money, but also reputation, trust and momentum can be greatly impacted.

What would happen if the image service we built in this chapter suddenly stopped working? The best case scenario would be for the emails to stay in the inbox and not be processed until the app is ready again. Users of the web album service would have to wait for their photos to appear. However, using Azure services we get built-in tolerance of outages and faults. If part of the underlying hardware fails for the Logic App or Azure Function, Azure will automatically and instantly use one of several other identical pieces of hardware in the Azure infrastructure. You wouldn't even notice any outage. Of course, if you mess up your own code or configuration, then Azure can only do so much.

### **2.3.4 This isn't production**

Perhaps by now you have noticed a few "lacking features" in the application you have built in this chapter. And you are right. While the application works exactly as intended by collecting email attachments and resizing photos, we have skipped a few details, such as:

- Only storing images to the storage account: Currently, all attachments are stored in the storage account. An extra step in the logic app to only select images would be a good idea.
- Configuring the optimal storage tiers for the data: Considering there might be large amounts of images to store, it could be both more efficient and more cost effective to choose a different storage tier for some images.
- Naming images sensibly: The current GUID naming strategy is not amazing for sorting through images and finding the right ones for each album. This could also be fixed with a database linking email addresses

to images.

I have left a bunch of deeper level details out on purpose, as building the application is meant to give you an insight into the power of Azure in the real world, rather than being bogged down with a million nuances and details. Don't worry if you are here for nuances and details though. There are plenty of both in the rest of the chapters.

### 2.3.5 Cost savings

Before we get into the Azure details and foundational knowledge, let's look at one of the most enticing aspects of cloud computing: cost. But first, a disclaimer. Cloud computing costs are notorious for being difficult to figure out. There are lots of tools to help you both calculate costs and monitor costs, but it is still difficult because of the nature of "compute by the minute" and "compute by feature" pricing. Hence, the following cost analysis is an example of how the cost of the image resizing application could turn out.

Table 2.1 outlines the cost for each of the three Azure services we have used: Logic App, Function and Storage Account.

**Table 2.1: Monthly cost of the image resizing application.**

Service type	Region	Description	Estimated monthly cost	Estimated upfront cost
Azure Logic Apps	Australia East	100,000 Action Executions x 1 day(s), 100,000 Standard Connector Executions x 1 day(s)	\$15.80	\$0.00
Azure Functions	Australia East	Consumption tier, 128 MB memory, 400 milliseconds		

<b>Functions</b>	East	execution time, 100,000 executions/month	\$0.00	\$0.00
<b>Storage Accounts</b>	Australia East	Block Blob Storage, General Purpose V2, LRS Redundancy, Hot Access Tier, 1,000 GB Capacity.	\$21.14	\$0.00
		<b>Total</b>	<b>\$36.94</b>	<b>\$0.00</b>

The assumptions made for the application are

- 100,000 executions of the Logic App, which equates to 100,000 email attachments.
- 100,000 executions of the Azure Function, each taking 400 milliseconds.
- Up to 1,000GB blob storage for the images.

All up, this will cost you \$36.94 per month. That's it. Imagine you'd have to buy hardware to run the code, then connect it to a network, write the application code, and then maintain all the hardware. You'd very quickly end up with a bill a lot larger than \$36.94.

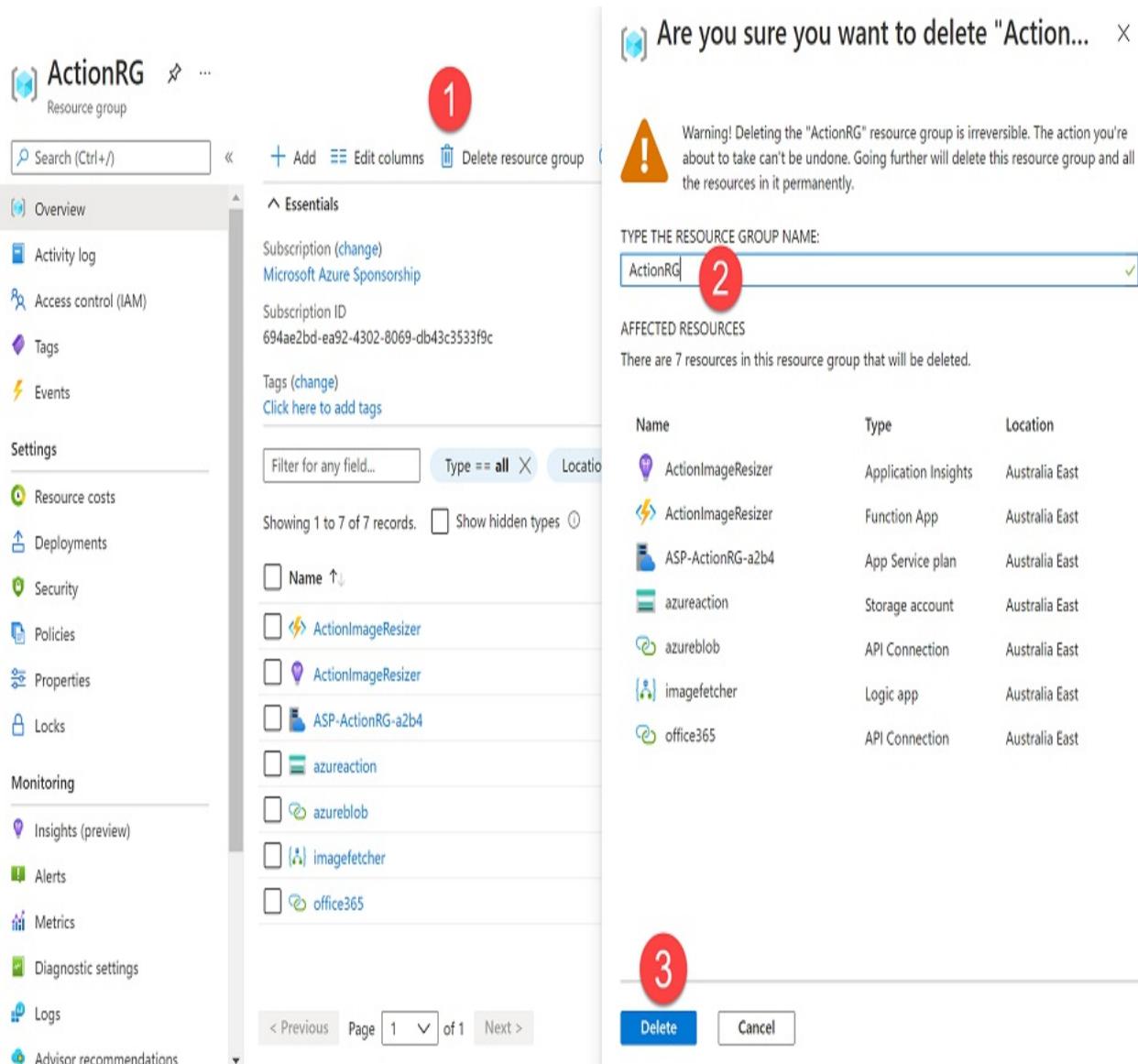
## 2.4 Deleting the Solution

You have built a solution that shows the power of Azure, and for not much cash we are resizing images to our hearts content. If you don't really need it though, now is the time to delete the cloud resources to make sure you don't end up paying for computing resources you didn't expect.

Remember that resource group we created right at the start of the chapter? Because we placed all resources inside the resource group, all we have to do is delete that. All the resources inside will be deleted too. Follow the steps in

Figure 2.51 to delete all the resources we just created.

**Figure 2.51: Steps to deleting the resources used in this chapter.**



Go to the resource group *ActionRG* in the Azure Portal, then do the following

1. Click on “Delete resource group”, which will open the side panel on the right.
2. Enter the name of the resource group to confirm you know what you are doing.
3. Click “Delete” and all your sins are forgiven. Well, the resources are

gone at least.

That is all there is to it. Because we added all the resources in the same resource group, managing them is much easier.

## 2.5 Summary

- Every Azure resource has to be in a resource group, which is one of the first parts to create for any project on Azure.
- Storage accounts can have blob containers that can hold blobs, which are pretty much any kind of binary object, such as images.
- Logic apps are workflows consisting of connectors and triggers. Connectors let you hook into third party services such as email clients to manipulate their data. Triggers makes the workflow start when a certain event happens.
- Azure services such as storage, Functions and Logic Apps can be connected to create more powerful applications that can add value to your business.
- Azure Functions are single-purpose compute processes that run entirely within Azure. They're called functions because that is all they do; a single function.
- It is not uncommon to build entire applications from hundreds or even thousands of Azure Functions, each responsible for a single purpose. This makes it simple to build, debug and maintain.
- Azure application lifecycles can be much easier to manage than traditional applications, as you can focus only on the services you interact with and not worry about hardware and other infrastructure.
- Azure Services can be extremely cost effective, such as using free Azure Function executions, only paying for the storage you are using (and not the whole allocation), and only paying for the actual executions of a Logic App.

[1] Naming convention guidelines: <https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/naming-and-tagging>

[2] Confucius didn't actually say this. Still true though.

[3] The total size of the storage account is limited to 500TB.

[4] A Globally Unique Identifier is a 128 bit unique reference number, e.g.  
d83c2e41-1e52-4b0c-9794-f9712b08fd3d

# 3 Using Virtual Machines

## This chapter covers

- Creating a virtual machine with Linux
- Accessing and managing a virtual machine remotely with RDP
- Optimizing virtual machines for cost and performance
- Understanding some best practices for virtual machines

One of the pillars of cloud computing, virtual machines are the backbone of almost any application, whether explicitly or implicitly. There are few company IT infrastructure ecosystems that don't require a virtual machine in some capacity to get the job done, and because they are a "raw" machine that you can use for almost any computing task, they are very popular and there are a vast array of types, sizes and flavours.

In this chapter we will cover one of the fundamental parts of not just cloud infrastructure, but any application, big or small. Compute. And the specific type of compute in Azure that is used by the most projects and organizations, is virtual machines (VM).

## 3.1 What is a Virtual Machine?

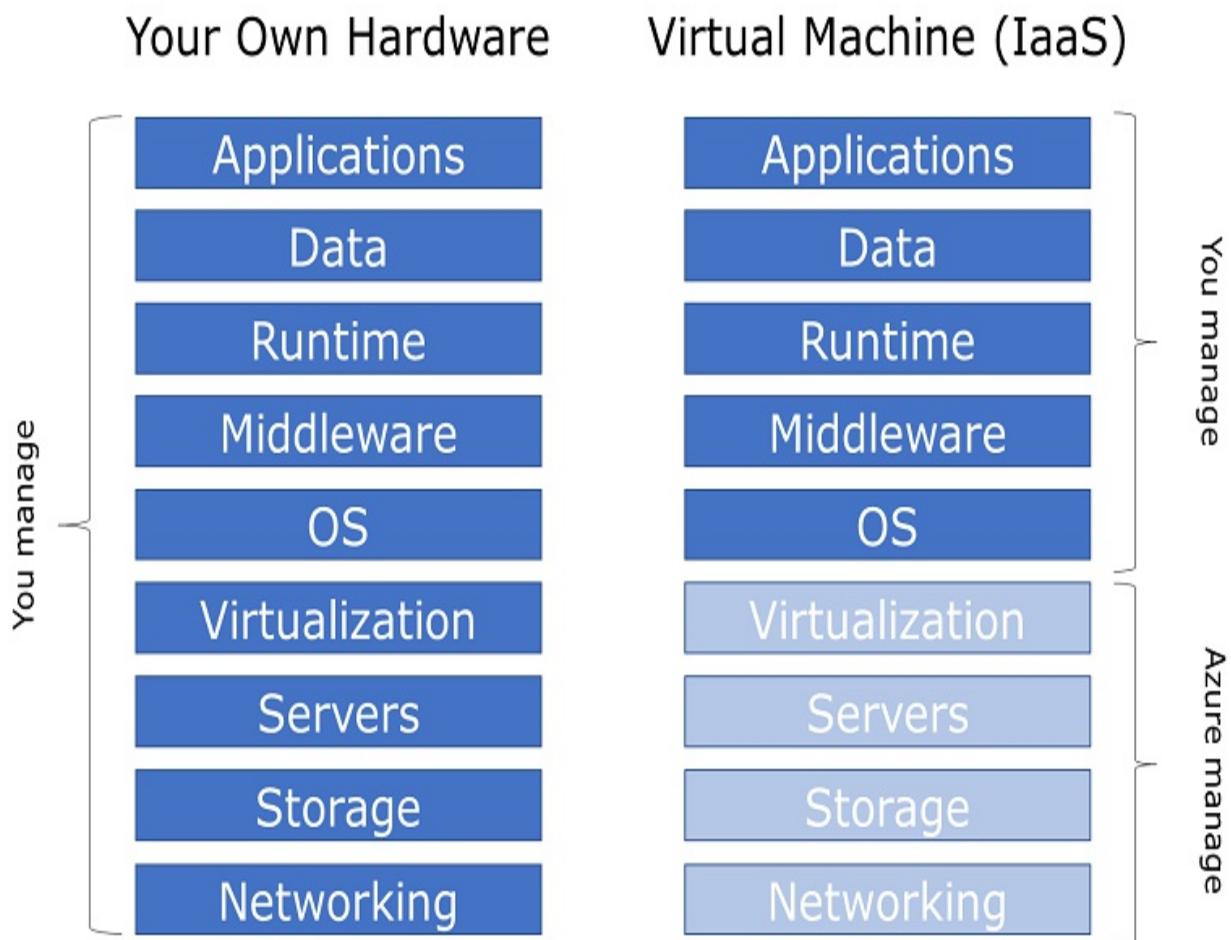
To understand the purpose and place of a virtual machine in Azure, consider this infrastructure scenario. You have been tasked with creating the infrastructure for a new project that has the following requirements and behaviors.

- Hosting of a custom software application, either Windows or Linux.
- Increase in use and resource demand over time.
- Control over the operating system and platform.
- Limited budget to purchase computing resources.

This might seem like any other IT project that you have ever come across, and for good reason. Most traditional IT projects exhibit these requirements

and behaviors. In fact, it would seem unusual if they didn't. The cool thing is that cloud computing loves exactly this scenario and one of the ways it can be solved is using virtual machines. While you might have heard of a virtual machine and possibly even used one or more of them, what does it actually mean? What is a VM, as they are commonly referred to, and why is it one of the most critical services on Azure? The short answer: VMs give you all the power of owning hardware without, well, owning hardware as shown in Figure 3.1. Azure will abstract all the hardware layers away, so you can focus on just managing and configuring the virtual machine.

Figure 3.1: Azure abstracts all the hardware layers when using a VM.



If we take the infrastructure scenario from the start of this chapter and place it into the world of cloud, a VM make a lot of sense.

- You can install any piece of software on a VM, with full control of any custom integration features and registration processes.
- When there is an increase in demand, there are multiple options for ensuring your application has enough resources to cope. Scale up the VM, use multiple VMs, increase disk sizes and more. And you do this without ever buying any hardware.
- You can choose from several difference operating systems, such as Windows Server, Red Hat Enterprise Linux, CentOS, CoreOS, Debian, Oracle Linux, SUSE Linux Enterprise, openSUSE, and Ubuntu.
- And of course, there are no hardware costs. It is Infrastructure-as-a-Service (IaaS), and you only pay for the time you use the VM. This is the essence of pay-as-you-go (PAYG) pricing that we covered in chapter 1 as well. If you don't use the VM all the time, you can switch it off, you can upgrade it, you can cuddle it to keep warm at night. Well, almost.

Remember from chapter 1 we discussed reliability? VMs are extremely reliable, which they should be considering they are one of the fundamental services on Azure. We will get to creating a virtual machine, and all the nuances of doing so, in just a moment. Let's start with an actual virtual machine and how it looks in the world of Azure. Figure 3.2 shows the Azure Portal overview of a virtual machine.

**Figure 3.2: Azure virtual machine overview.**

The screenshot shows the Azure Portal interface for managing a virtual machine named "azureactionvm".

**Essentials Summary:**

- Resource group: ActionRG
- Status: Running
- Location: Australia East (Zone 1)
- Subscription: Microsoft Azure Sponsorship
- Subscription ID: 694ae2bd-ea92-4302-8069-db43c3533f9c
- DNS name: Not configured
- Availability zone: 1
- Tags: Click here to add tags

**Properties Tab (Selected):**

Virtual machine		Networking	
Computer name	azureactionvm	Public IP address	20.70.232.34
Operating system	Linux (ubuntu 18.04)	Public IP address (IPv6)	-
Publisher	Canonical	Private IP address	10.0.0.4
Offer	UbuntuServer	Private IP address (IPv6)	-
Plan	18.04-LTS	Virtual network/subnet	ActionRG-vnet/default
VM generation	V1	DNS name	Configure
Agent status	Ready	Host	-
Agent version	2.3.0.2		
Host group	None		
Host	-		

The Azure Portal experience for a VM gives you access to all functions and features you need to manage it and provides one of several ways to interact with a VM on Azure. We will get to another later in this chapter.

### 3.1.1 Infrastructure-as-a-Service

A virtual machine is one of the three main services on the Infrastructure-as-a-Service (IaaS) platform, which we briefly touched on in chapter 1. IaaS is the lowest level of service you can get on Azure, meaning the closest you get to the physical hardware, such as disks and CPUs. You provision only the hardware resources you need, but through virtualisation.

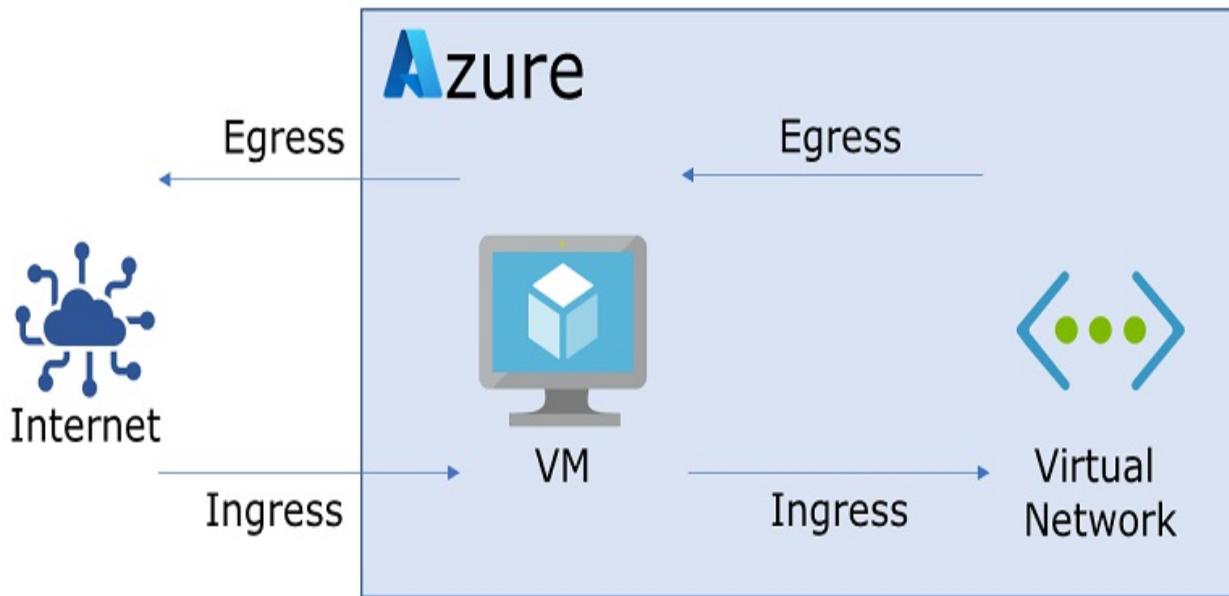
Virtualization uses software to create a layer of abstraction on top of the computer hardware that allows the individual elements of a single computer—processors, memory, storage and more—to be divided into multiple virtual computers, which is what a virtual machine (VM) is.

IaaS enables vast number of users to request resources on Azure to use, as VMs rarely use anywhere near full capacity. This means Azure can “oversell” their resources to a certain degree. Of course, there are a lot of clever algorithms being used on the Azure platform to make sure no server is over-provisioned, but as an end user of Azure, you’d never notice. The net result? You get access to almost infinite compute resources in an instant for very little cost.

### 3.1.2 Ingress and egress

A part of Azure that is often hidden in the background of the cloud focused consciousness is the use of bandwidth. Ingoing and outgoing data, known as ingress and egress, is measured separately on Azure(Figure 3.3), and ingress is free, egress is charged on Azure. In other words, it won’t cost you anything to access your VM, but sending data out from it will. If you send data out on the Internet, to another Azure availability zone (we will get to those in a moment) or another region, you are charged per GB.

**Figure 3.3: Ingress and egress traffic.**

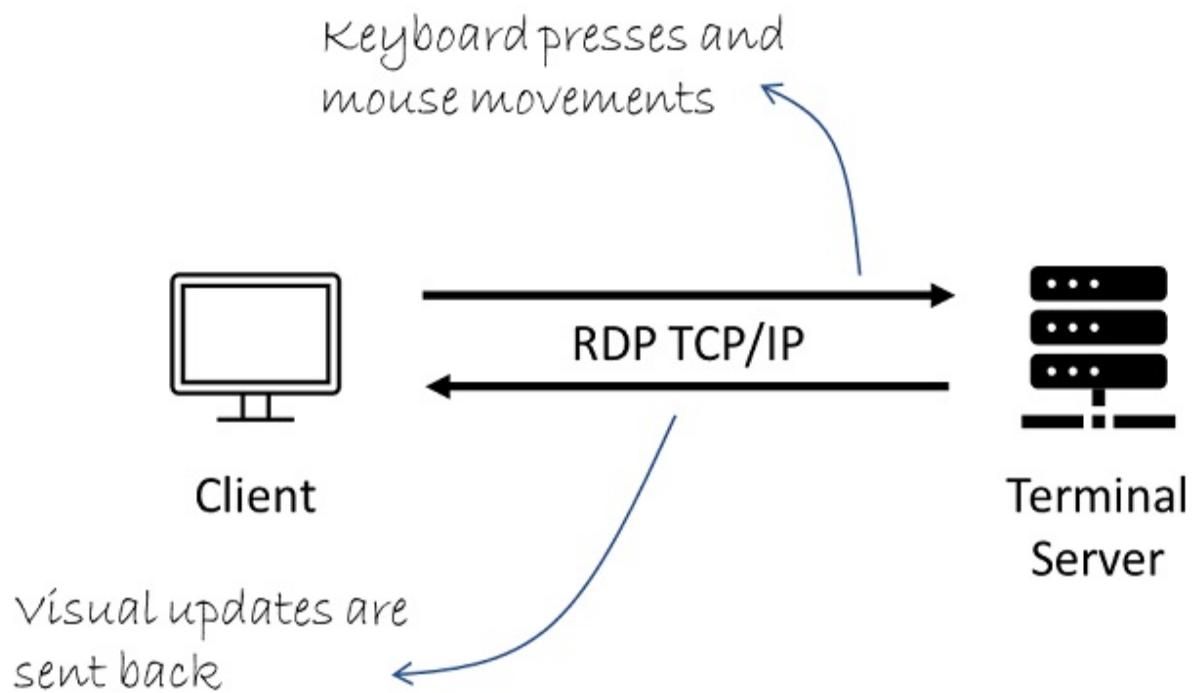


The egress cost can often be something that catches out a project and is one of the things that often gives cloud computing the dubious award of “most unexpected cost of the year”. As you learnt in chapter 1 there are several ways to buy compute time in Azure, as well as limit the costs of running a VM. While you can limit egress data only to a certain extent (you’ll have to send *some* data, right), you can take advantage of for example reserved instances and spot VMs to reduce and manage costs.

### 3.1.3 Remote Desktop Protocol (RDP)

Once you have a virtual machine up and running on Azure, you need to use it somehow. The most common way to connect to a virtual machine is by using the remote desktop protocol (RDP) to establish a connection. This is done over a network connection, often using the public internet to connect to any machine anywhere in the world. The RDP data flow is shown in Figure 3.4.

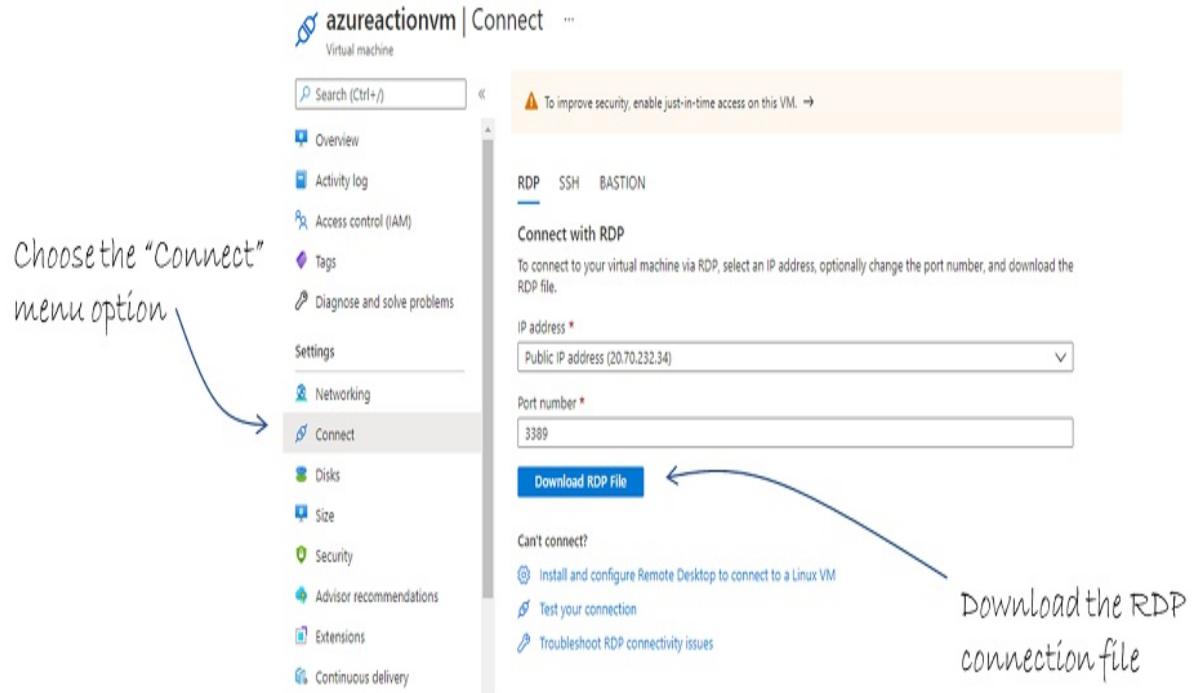
**Figure 3.4:** Remote Desktop data flow.



The RDP protocol has been widely used since 1998 when it was first part of Windows NT 4.0 Terminal Server Edition. Since then, it has been included with every version of Windows, and there are several implementations for using RDP with Linux as well.

Virtual machines on Azure all come ready to use RDP from the start. To connect, go to the *Connect* menu for the VM in the Azure Portal as show in Figure 3.5.

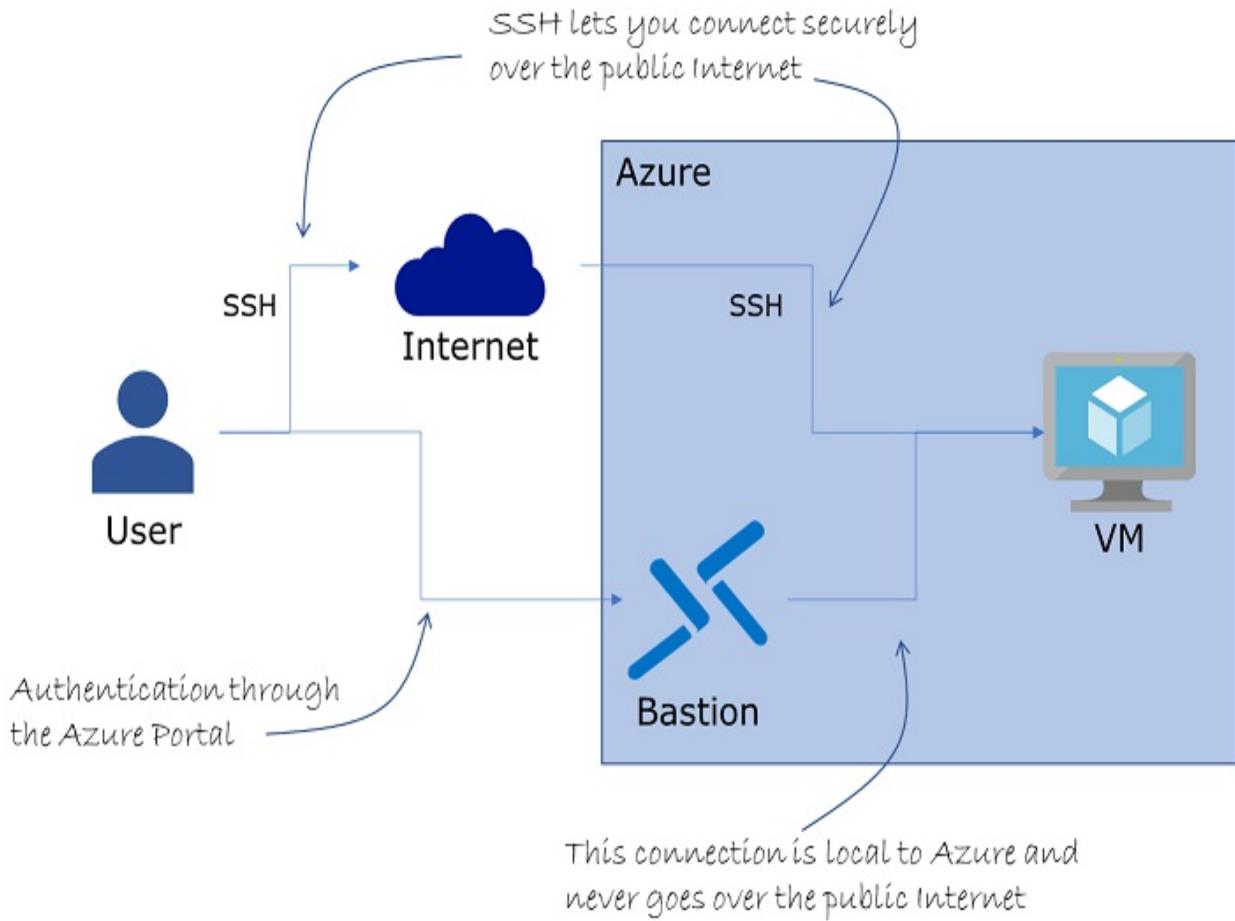
**Figure 3.5: Downloading an RDP connection file for an Azure VM.**



This will let you download an RDP file, which you can open either on Windows or with your favorite RDP program on Linux. The file will have all the necessary connection details configured, so you are connected straight to the VM desktop in no time. Okay, that isn't entirely true, as you most likely need to configure the incoming rules for the network security group attached to the VM, but we will go into more detail on that later in the chapter.

As you might have noticed there are two other ways to *Connect* to a virtual machine in Figure 3.5: Secure Shell, commonly known as SSH, and Bastion (Figure 3.6).

**Figure 3.6: Using either SSH or Bastion to connect to a VM.**



First, SSH is an encrypted connection protocol that lets you use unsecured connections, such as the public Internet, securely. It's like BYO security for connecting to a VM. Where RDP (using port 3389) is mostly used on Windows VMs, SSH (using port 22) is preferred to connect to VMs running Linux. It is a command line only protocol, so you don't get any pretty desktop experience, but it is fast and efficient. You can get a desktop experience on Linux though, by using other tools which we won't cover here. Also, Linux users often prefer a command line, and Windows users tend to go for a desktop experience, at least initially.

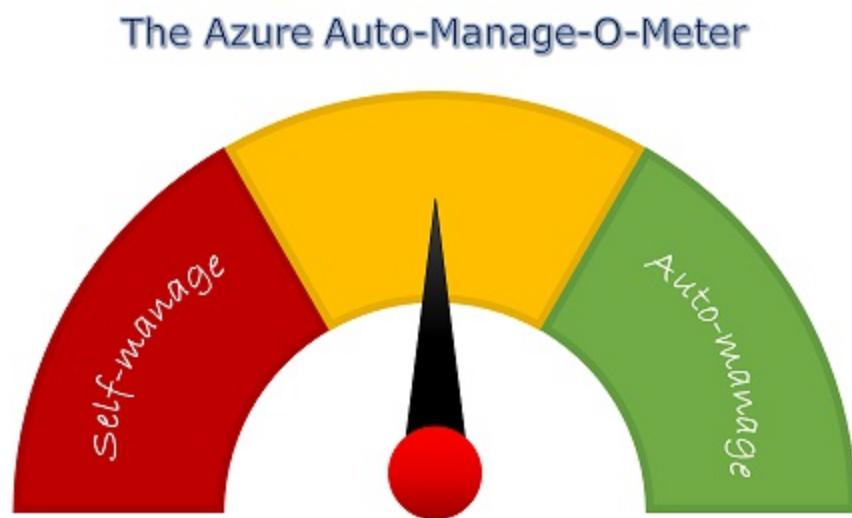
Finally, Azure Bastion is a service to connect to VMs entirely through Azure virtual networks without ever accessing the public internet. This increases access security and trust in the infrastructure that the VMs are part of. More on Bastion later in this book though. It is a whole section of delicious cross pollination of networking and security in itself.

### 3.1.4 Managing Software on a VM

Part of the choice of using a virtual machine, and IaaS in general, is the understanding that you are only provided the infrastructure to work with. Which services and software runs on it, and how this is maintained is up to you. This is part of the appeal of using a VM: You get a full machine you can control completely, but you don't pay for the actual hardware. If you want to run a piece of legacy software that three people in the company use, that is totally cool. If you want to run a backend process that relies on three different databases and is triggered twice every second, no worries. You are in control of the software and decide on the maintenance schedule and lifecycle of the product.

To help you manage the virtual machine itself and adhere to Azure best practices, there is **Azure Automanage**. This service can remove some of the headaches of onboarding Azure services and maintaining them over time. As illustrated in Figure 3.7 you can choose to fully self-manage the VM or you can dial it all the way to auto-manage and let Azure help you by applying patches and updating policies, security and configurations to industry best practices.

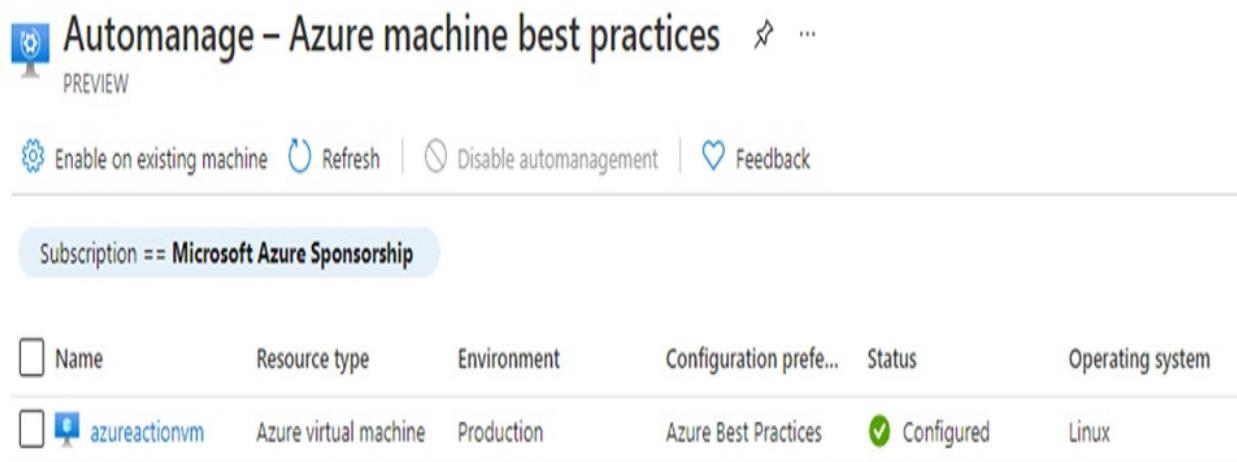
**Figure 3.7: Self-manage your VM and be in charge of policies and configurations, or use Azure Automanage to let Azure apply update patches and apply security best practices.**



The point is that even though a VM is as close as you get to the actual

hardware in Azure, and offers you the most control, you aren't left completely to your own devices, unless you choose to be. Managing your own business critical software and applications on a VM is often enough work without having to also factor in the maintenance of the VM itself and the Azure ecosystem it lives in. Azure Automanage can help with backups, logs, security, monitoring, updates and much more with just a single service.

**Figure 3.8: Azure Automanage configured for a VM.**



The screenshot shows the 'Automanage - Azure machine best practices' interface. At the top, there's a 'PREVIEW' button and a '... More options' button. Below that is a toolbar with 'Enable on existing machine' (with a gear icon), 'Refresh' (with a circular arrow icon), 'Disable automanagement' (with a circular off switch icon), and 'Feedback' (with a blue heart icon). A message box says 'Subscription == Microsoft Azure Sponsorship'. The main table has columns: Name, Resource type, Environment, Configuration pref..., Status, and Operating system. One row is shown: 'azureactionvm' (Resource type: Azure virtual machine, Environment: Production, Configuration pref...: Azure Best Practices, Status: Configured, Operating system: Linux).

Name	Resource type	Environment	Configuration pref...	Status	Operating system
azureactionvm	Azure virtual machine	Production	Azure Best Practices	Configured	Linux

As shown in Figure 3.8 it is a simple action to enable Automanage on an existing VM, and then you can almost leave it to its own devices, knowing that Azure services will be managed and kept in line with the best practices.

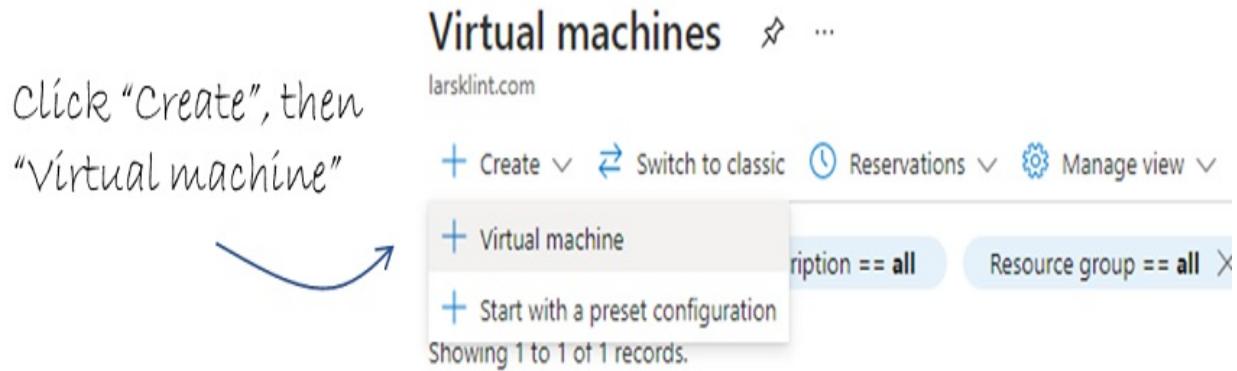
## 3.2 Creating a VM

Okay, enough talk. Let's create a VM now that you know the best scenarios for them, how to connect to one, and that you aren't left completely alone. There are several ways that you can create a VM on Azure, such as using PowerShell, the Azure CLI, ARM templates and more. In this chapter we will do it using the Azure Portal, as it provides the best way to get your head around the nuances and features of a virtual machine. I'll sprinkle the chapter with references to PowerShell<sup>[1]</sup>, as you start to get more and more into the fluffy world of cloud.

Go to the Virtual machines section of the Azure Portal and click “Create” as

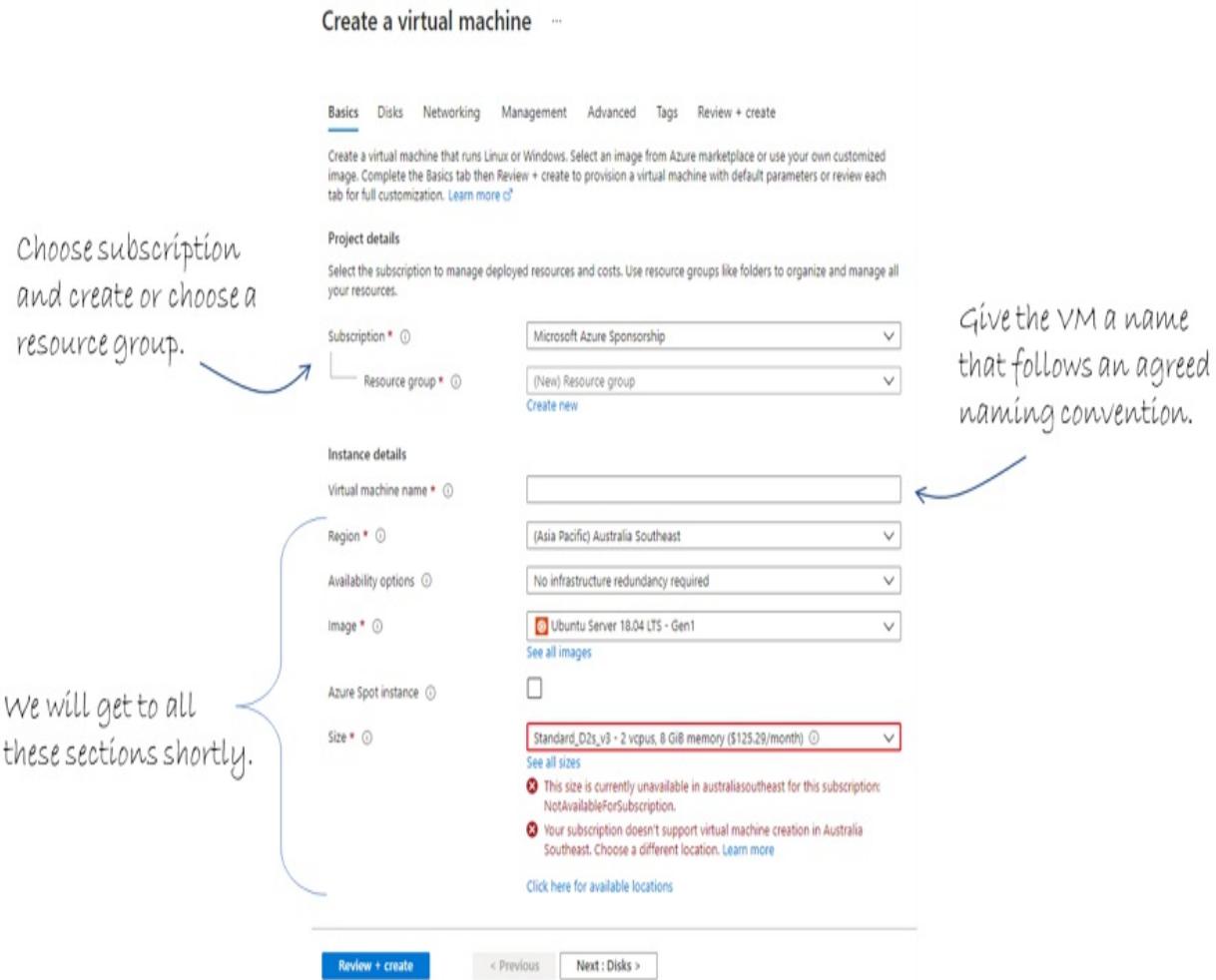
shown in Figure 3.9.

**Figure 3.9: Create a new Virtual Machine.**



There are quite a few things going on in the following screen (Figure 3.10), as well as the subsequent steps, so we will use the next several sections to go through it. No, you shouldn't skip it just because it is long. Yes, you should try and follow along on your own Azure instance. We can do it together, right? It'll be fun.

**Figure 3.10: The first part of the wizard for creating a new VM.**



The first part of the **Basics** should not cause any concern by now. Fill in the Subscription, Resource Group and VM name as shown in Figure 3.11. Easy peasy.

**Figure 3.11: Set resource group and name for the VM.**

The screenshot shows the 'Subscription' dropdown set to 'Microsoft Azure Sponsorship' and the 'Resource group' dropdown set to 'ActionRG'. A 'Create new' link is visible below the resource group dropdown.

**Subscription \*** ⓘ Microsoft Azure Sponsorship

**Resource group \*** ⓘ ActionRG

Create new

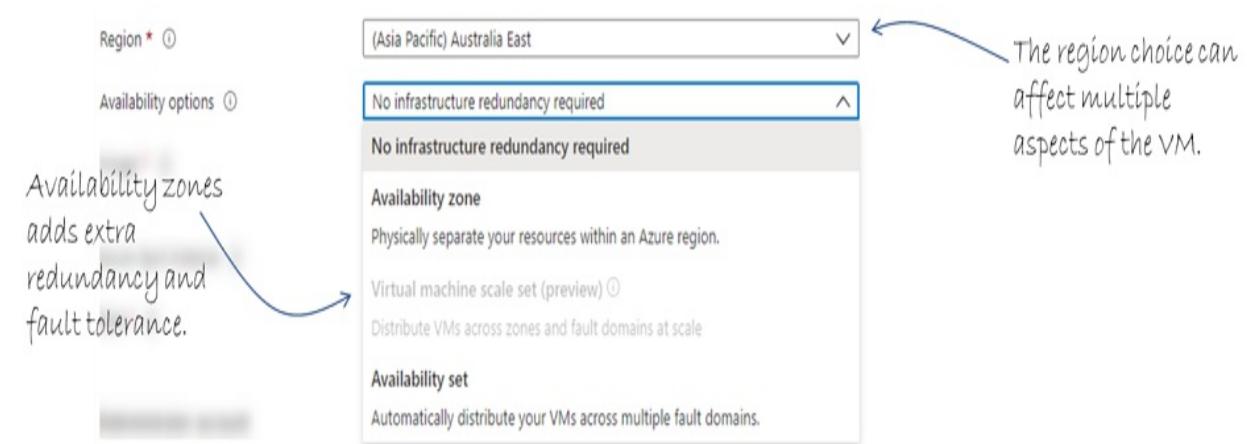
**Instance details**

**Virtual machine name \*** ⓘ ActionVM ✓

### 3.2.1 Location

Now we are starting to get into the more interesting part: region, or location, of where you choose to create your VM might seem trivial. However, it can have a big impact on performance, pricing, features, size and more. Choosing the right location for a VM is critical and once you have selected a region, it *can* be a chore to move it. Choose a region that you prefer but pay attention shortly to how the choice of region affects various factors. Set the availability options to *No infrastructure redundancy required* as shown in Figure 3.12.

**Figure 3.12: Choose region and availability options.**



To investigate the importance of choosing an appropriate regions, let's look at VM pricing as a dependent of region choice. Because there are different costs for providing cloud services in each region, such as regulatory and

energy pricing, the price you pay as a user will change as well. Table 3.1 shows the average price for running a VM for an hour in various regions.

**Table 3.1: Average VM price per region (USD).**

Region name	Avg price per VM per hour (USD)
West US 2	0.57
East US 2	0.58
North Central US	0.64
South Central US	0.65
North Europe	0.66
Central India	0.68
UAE North	0.7
West US 3	0.71
Australia East	0.76
UK West	0.77

South Africa North	0.77
Norway East	0.78
Japan East	0.83
Switzerland North	0.85
South Africa West	0.96
Norway West	1.02
Brazil Southeast	1.2

First, it is obvious that spinning up a VM in Brazil Southeast region will cost more than twice as much as West US 2. More interesting is that if you place your VM in West US 3 instead of West US 2, you are paying about 20% more. That is a significant difference for resources that are roughly in the same region. Other costs, such as ingress and egress traffic, may also affect a decision to use one region or another. More on bandwidth usage and pricing later in the chapter.

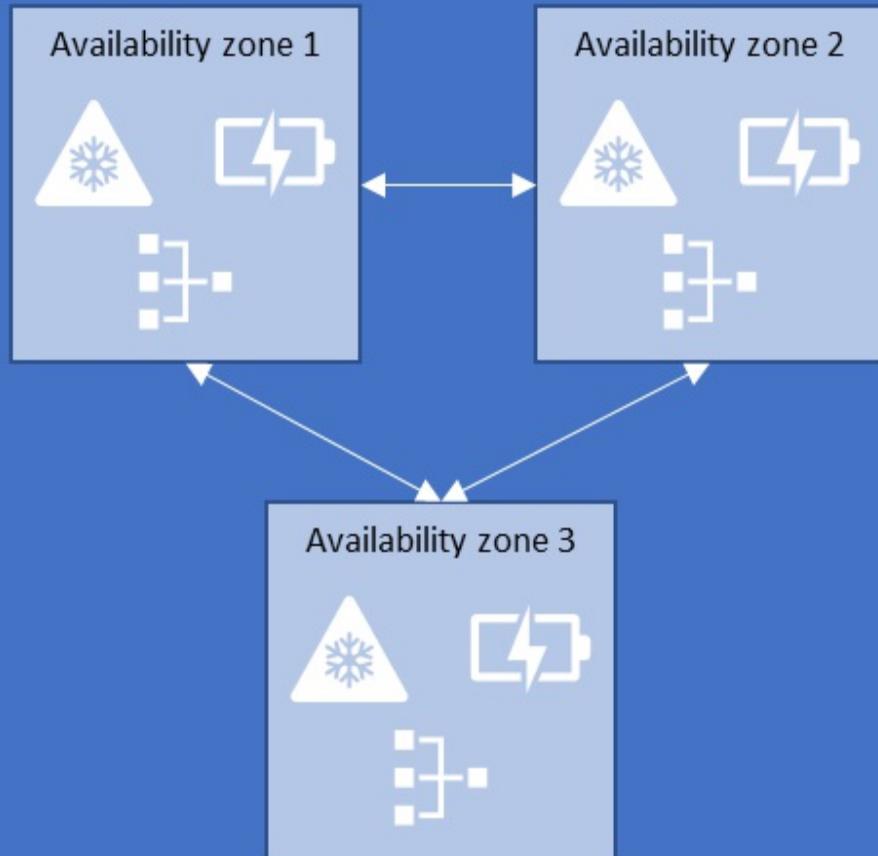
If you do create a virtual machine (or one of a number of other compute and network IaaS resources) in a location you regret, all hope is not lost. If you sacrifice your favorite stuffed toy to the cloud gods and chant the secret passphrase three times, you can move it. Or you can just use the Azure Resource Mover. This excellent service will let you move your VM to another region in a validated and managed way.

Another important consideration for choosing a location is the need for redundancy of the VMs, also called availability. As you can see in Figure 3.12, Australia East region has **Availability Zones** and **Availability Sets**.

- Availability zones (Figure 3.13) are three distinct physical and logical locations inside some Azure regions, such as Australia East. Each zone is made up of one or more datacenters and is equipped with independent power, cooling and networking infrastructure. All data and applications are synchronized between the availability zones.  
This means you will eliminate any single point of failure and the risk of a “crash and burn” scenario is much smaller.

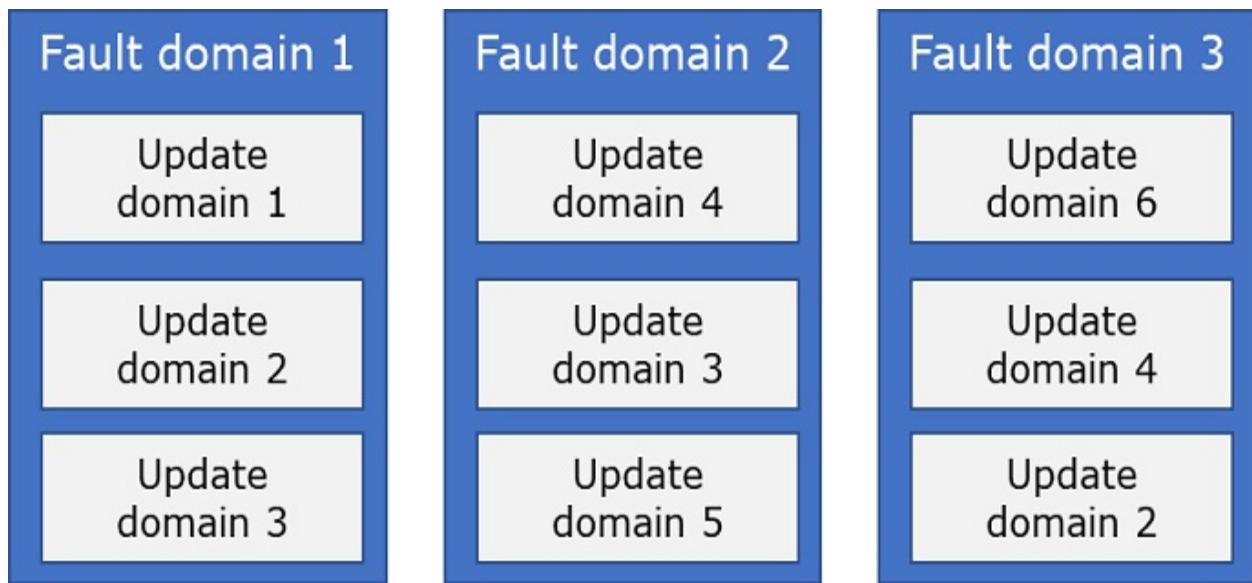
**Figure 3.13: Azure availability zones with separate cooling, power, and network.**

# Region



- Availability sets are groups of VMs that are each assigned to a fault domain and an update domain (Figure 3.14). A fault domain is a logical group of underlying hardware that share a common power source and network switch, similar to a rack within an on-premises datacenter. Update Domains are only ever updated one at a time, two would never be updated at the same time and this is how service outages are avoided. These domains inform Azure which VMs can be rebooted at the same time in case of an outage (fault domain) or platform updates (update domains).

Figure 3.14: An availability set with 9 VMs, 6 update domains and 3 fault domains.

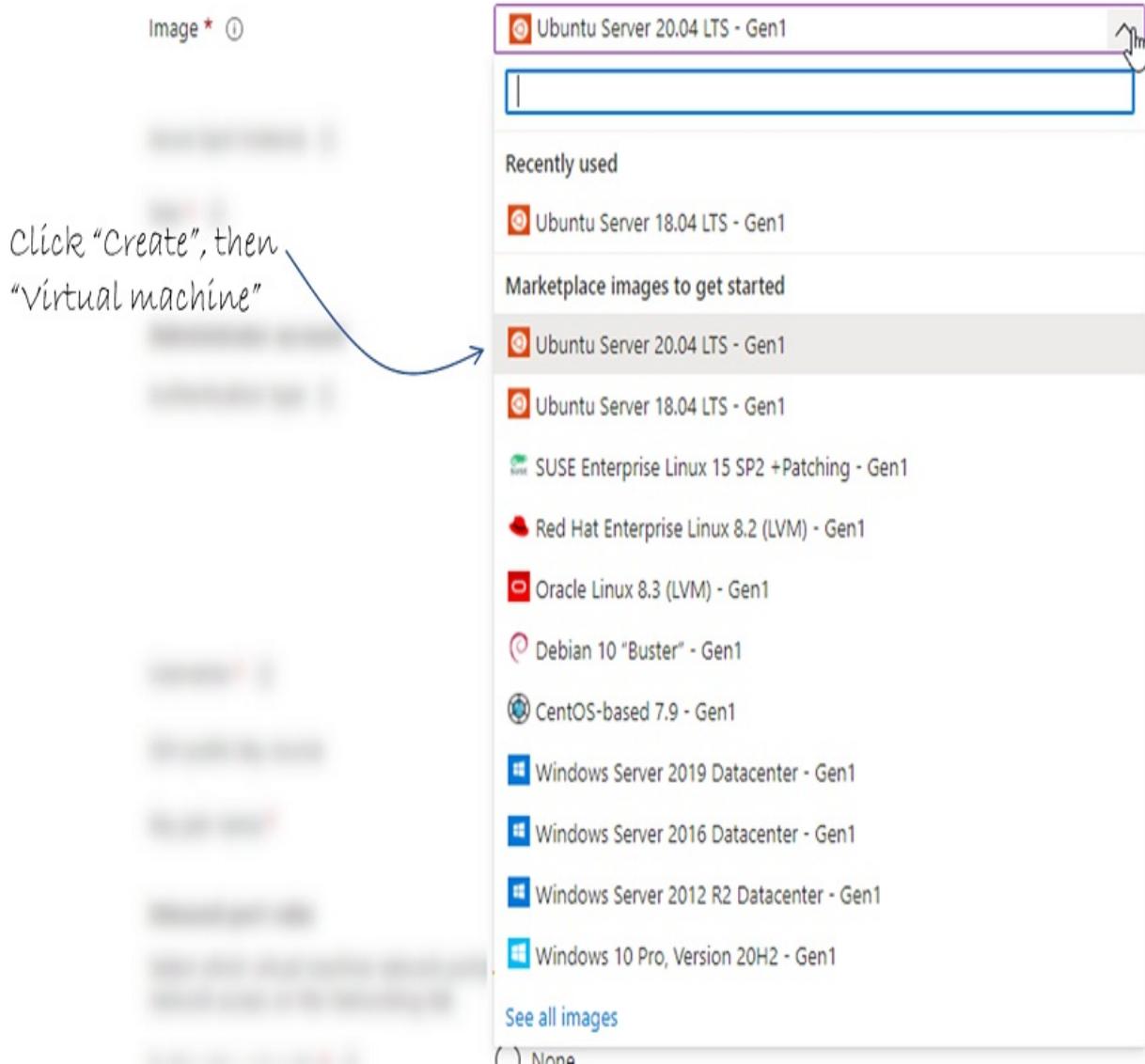


If you choose to use an availability zone, you will then also choose which zone to place your VM in. This is the physical placement of the VM within the region, although you don't actually know where availability zones 1, 2 and 3 are. If you want to use an availability set, you must set one up before you create the VM. For now, you can choose to not have any infrastructure redundancy, but it is important to know what both availability zone and availability set can do to improve the fault tolerance and high availability of your virtual machines.

### 3.2.2 Operating system

Back to the VM we are creating using the Azure Portal Wizard. For the VM image, choose the latest LTS (long term support) version of Ubuntu (Figure 3.15).

**Figure 3.15: Choose the latest Ubuntu LTS version as VM image.**



You might think there are two operating systems (OS) to choose from: Windows and Linux. And you'd be kind of right. While Windows and Linux are the main categories of operating systems there is currently a large range of specific OS options for your VM as shown in Figure 3.16.

**Figure 3.16: Azure Marketplace options for VM images.**

Microsoft | Azure Marketplace Apps ▾

Search Marketplace

More ▾ Sign in

Browse apps

Get Started

Analytics

AI + Machine Learning

Azure Active Directory

Blockchain

Compute

Containers

Databases

Developer Tools

DevOps

Identity

Integration

Internet of Things

IT & Management Tools

Monitoring & Diagnostics

Media

Migration

Mixed Reality

Networking

Security

Storage

Web

Trials All All All All Virtual Machine Im...

Operating System All All All All

Publisher All All All All

Pricing Model All All All All

Product Type All All All All

Reset filters

Results in All apps (4083)

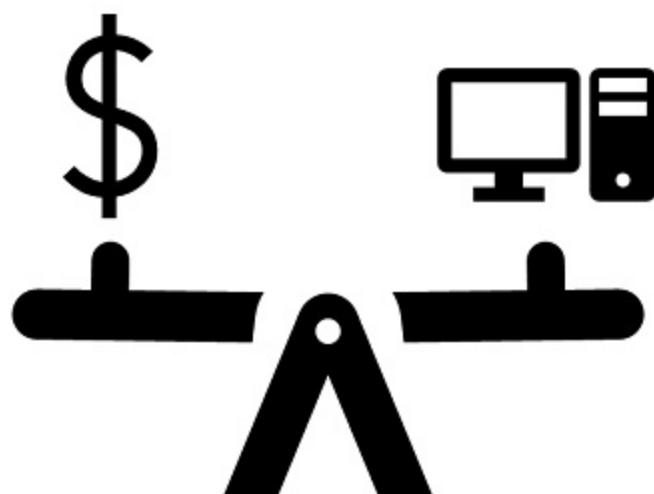
 OpenVPN Access Server By OpenVPN OpenVPN Inc.	 VM-Series Next-Generation Firewall... By Palo Alto Networks, Inc. Looking to secure your applications in Azure, protect against threats and prevent data	 Remote Desktop Services (RDS)... By Microsoft Create a basic Remote Desktop Services (RDS) deployment.	 CloudGuard Network Security - Firewall &... By Check Point CloudGuard Network Security delivers advanced threat prevention to protect mission-critical assets.
 Bring your own license Get it now	 Price varies Test Drive	 Price varies Get it now	 Price varies Get it now
 Elasticsearch (Self-Managed) By Elastic The Elastic Stack (ELK Stack) on Azure in a Few Clicks	 Hortonworks Data Platform (HDP) Sandbox By Hortonworks Powered by HDP 2.6.4 100% open source platform for Hadoop, Spark, Storm, HBase,	 Barracuda CloudGen WAF for Azure By Barracuda Networks, Inc. The most deployed WAF in public cloud.	 Oracle Database 12.1.0.2 Enterprise Edition By Oracle Oracle Database 12c Enterprise Edition (12.1.0.2.0) is a next-generation database designed
 Price varies Get it now	 Bring your own license Get it now	 Software plans start at US\$1.04/hour Free software trial	 Get it now

The available OS options (or SKUs as they are called) in general depends on which location you have chosen, but at the time of writing there are more than 4,000 images available. 4,000! How could you ever know which one to choose? Most of the images are very specific. If you need to set up an Oracle database, there is an image for that. Need a specific VM with tools for developing blockchain applications? There is an image for that. If you set up a VM for a general purpose, such as running some custom software, it is most likely you will choose one of the common Windows Server versions or plain Linux distributions.

### 3.2.3 Size

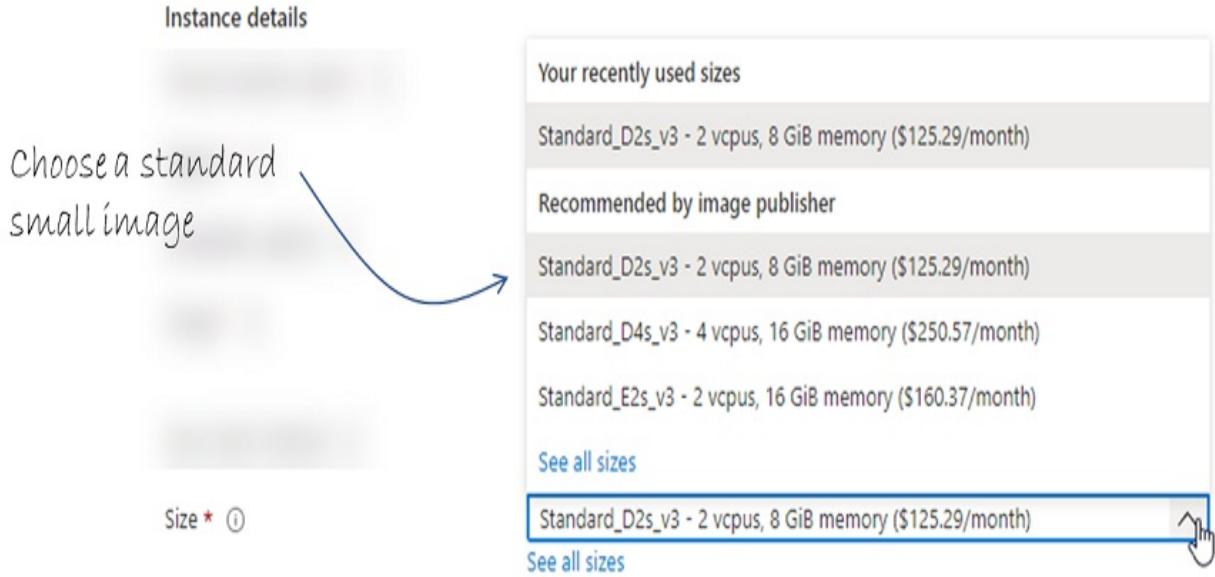
The size of a virtual machine is crucial in balancing costs vs. needed resources. Create a too large VM and you will be paying for resources you don't use. Create a too small VM, and your applications won't serve your users adequately. You need to find the right balance between the two sides of the equation (Figure 3.17).

**Figure 3.17:** Balance the need for compute resources with the cost.



I can't tell you what size you need, as each case for a VM should be evaluated individually. All I know is that the path to your manager's heart goes through the budget in his spreadsheet. For the exercise in this chapter, choose something small and inexpensive like a *Standard\_D2s\_v3* image (Figure 3.18).

**Figure 3.18:** Select a small relatively inexpensive VM size, such as *Standard\_D2s\_v3*.



See that little *See all sizes* label in Figure 3.18? If you click that you will get a list the length of a small country. There are a LOT of choice in virtual machines. They are split up into categories of general purpose, compute optimized, memory optimized, storage optimized, GPU intensive and high performance. Finding the right one for a project is outside the scope of this book though.

### 3.2.4 Other Resources

I know we have spent a lot of time and pages so far on creating a VM, and we haven't even got through the first page in the Portal wizard. However, it is critical we get off on the right foot (and don't put it in the mouth) when creating VMs. Only a couple more things left on this screen.

To log in securely to our new VM, you need to set up an administrator account. You have the choice of using an SSH key pair or a password to authenticate the administrator (Figure 3.19).

**Figure 3.19: Use the SSH key pair as authentication to improve security.**



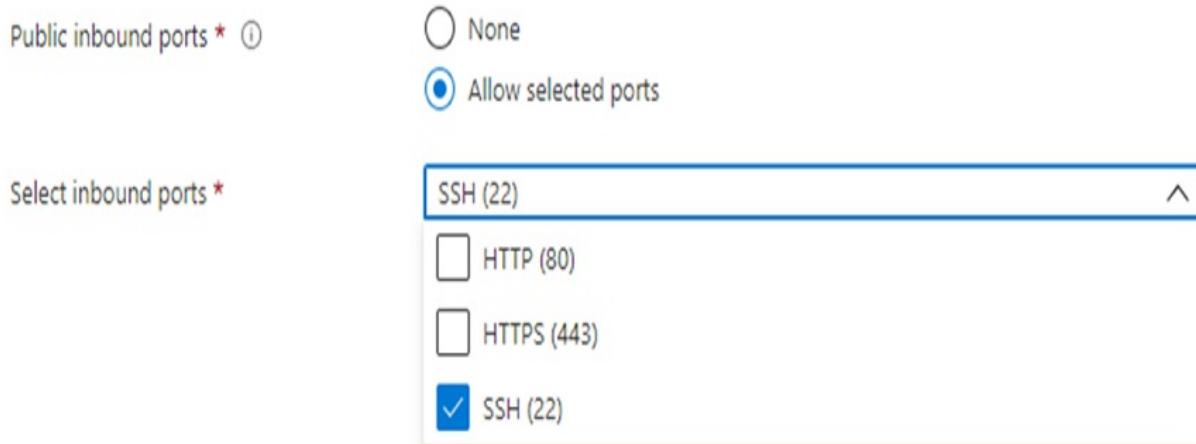
A general recommendation is to use an SSH key pair, as a standard password can be more easily acquired and abused to gain access. You can choose to generate a new key pair or use an existing one you have saved for just this occasion. Once you have given it a name, that is it. Follow the prompts on the screen to create and store the SSH key pair.

The last step on the **Basics** tab of the wizard is to choose which inbound ports are open from the start (Figure 3.20). Up to you, but in this case, you could go with SSH, which is port 22. Port 80 is for unencrypted web traffic, and port 443 is for encrypted web traffic.

**Figure 3.20:** Select the ports you want to open on the new VM.

### Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.



Alright, that is it for the basics. Now you may click the intriguing “Next” button. Go on. You earnt it.

### 3.2.5 Disks

A computer isn’t worth much without a hard disk, and the same goes for a VM. We need to create a disk, or at least attach an existing one, as part of the VM creation process. As shown in Figure 3.21, you need to choose a primary OS disk that your main VM image will be deployed to, and then any additional data disks.

**Figure 3.21: Create OS and data disks for the VM.**

Azure VMs have one operating system disk and a temporary disk for short-term storage. You can attach additional data disks. The size of the VM determines the type of storage you can use and the number of data disks allowed. [Learn more](#)

Disk options

OS disk type \* ⓘ

Premium SSD (locally-redundant storage)

Encryption type \*

(Default) Encryption at-rest with a platform-managed key

Enable Ultra Disk compatibility ⓘ



Ultra disk is available only for Availability Zones in australiaeast.

Choose disk type.

Data disks

You can add and configure additional data disks for your virtual machine or attach existing disks. This VM also comes with a temporary disk.

LUN	Name	Size (GiB)	Disk type	Host caching	
Create and attach a new disk		Attach an existing disk			
▼ Advanced					

Attach any data disks needed.

The main choice for the OS disk is whether you want to use a solid state drive (SSD) or a traditional hard disk drive (HDD). Choosing either is an actual physical choice, as the drives are real, rather than virtualized. You pay more for a premium SSD over a standard SSD, and the cheapest is an HDD. Again, you pay for performance, and if you have an application running on the VM with little traffic or resource needs, then an HDD could work just fine. All the disk options are locally redundant, meaning there are at least three copies of the data in the local datacenter.

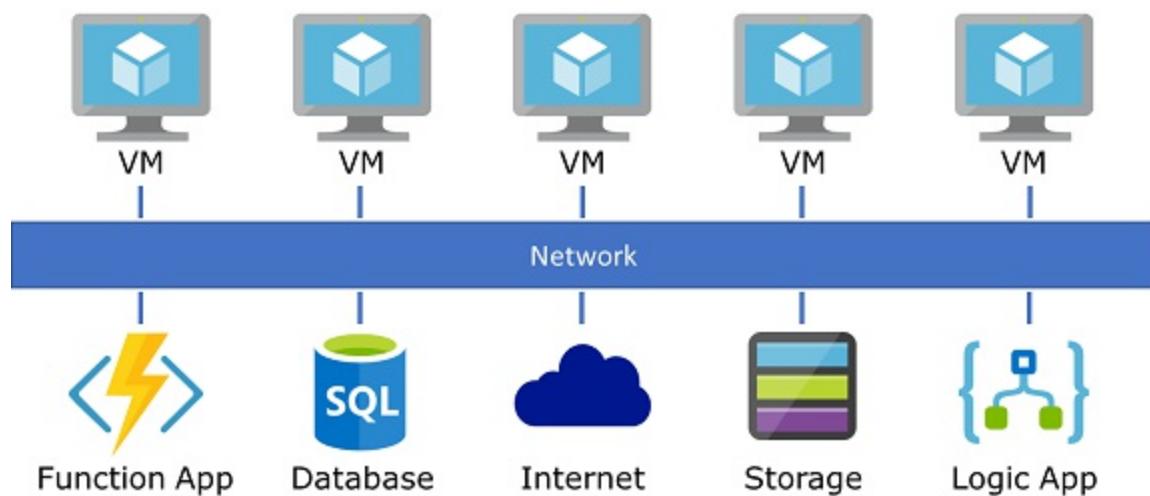
The second choice is the type of encryption. All managed disks on Azure are encrypted at rest, so the choice is whether you want to manage the key yourself, and if you want double encryption! Double must be better, right? It is encryption with both the platform managed key *and* your own key.

Finally attach any data disks you need. You can attach an existing disk you have or create a brand spanking new one. We will look at lot more at storage in the chapter on...well, storage. And now, click that “Next” button.

### 3.3 Networking

The VM can't do much without a connection to it. We need to get all that delicious data in and out of the machine, and that is what networking is for (Figure 3.22).

**Figure 3.22:** VMs need to connect to a network to access the Internet and other Azure services.



Unless you have a network that Azure resources, such as a VM, can connect to they are completely isolated. For that reason, you can't create a VM without a virtual network attached to it. The networking tab is where this happens (Figure 3.23).

**Figure 3.23:** The networking tab for the VM.

Define network connectivity for your virtual machine by configuring network interface card (NIC) settings. You can control ports, inbound and outbound connectivity with security group rules, or place behind an existing load balancing solution.  
[Learn more ↗](#)

#### Network interface

When creating a virtual machine, a network interface will be created for you.

Virtual network *	<input type="text" value="ActionRG-vnet"/> <span style="float: right;">▼</span>
	<a href="#">Create new</a>
Subnet *	<input type="text" value="default (10.0.0.0/24)"/> <span style="float: right;">▼</span>
	<a href="#">Manage subnet configuration</a>
Public IP	<input type="text" value="(new) ActionVM-ip"/> <span style="float: right;">▼</span>
	<a href="#">Create new</a>
NIC network security group	<input type="radio"/> None <input checked="" type="radio"/> Basic <input type="radio"/> Advanced
Public inbound ports *	<input type="radio"/> None <input checked="" type="radio"/> Allow selected ports
Select inbound ports *	<input type="text" value="SSH (22)"/> <span style="float: right;">▼</span>

⚠ This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.

### 3.3.1 Virtual Network

A virtual network, often just called VNet, is the lifeline for your VM. Without a network connection you can't even get to your own VM. It goes without saying that a VM must have a VNet connection. The Networking chapter goes into a ton more details on VNets, subnets and everything in between, so we'll glance over it here. Choose an existing VNet for the VM or create a new one. Leave the subnet as default. Dance.

### 3.3.2 Public IP Address

If you want to access your VM from the public internet, you will need a public IP address. This could be to let users access a web server, host an API, or just be able to connect remotely to the VM using RDP. Whatever the reason, the one thing to keep in mind is that a public facing IP address means the VM is open to the whole Internet. Of course, we have many ways to protect from attacks and unauthorized access, but a public facing network connection still offers up a target. More on connecting to a VM using RDP later in this chapter too.

Create a public IP address for the VM in this case though. We will use it to connect to the VM in a minute.

### 3.3.3 Network security group

To filter the traffic to and from a VM, you use a network security group (NSG). This is a critical part of keeping your VM, as well as your greater cloud infrastructure, safe from intruders and unauthorized players.

When setting up a VM, you can choose from three approaches.

- **None:** Don't set one up. Which is what you would normally do. No kidding, but I will get to that shortly.
- **Basic:** You can choose from several preset port configurations such as RDP, HTTP, HTTPS, SSH and more, to allow access to the VM.
- **Advanced:** This lets you pretty much set up the NSG as you want.

And that is about as much as we will go into NSGs at this time, for two reasons. Number one: we will go through NSGs in the networking chapter in much more detail. Number two: best practices suggest not to create an NSG that is attached to a network interface card (NIC) for a specific VM. Instead NSGs are associated with a virtual network, which the VMs are then attached to. This is both more maintainable and can save cost. For now, leave it as **Basic**.

### 3.3.4 Load Balancing

The last section on the *Networking* tab of the VM wizard is load balancing. This is balancing the load on the VMs, and you do that through a combination of a load balancer and using multiple VMs in what is called a VM scale set. More on scale sets later in the book too. The load can be anything from computations requiring CPU cycles to inbound traffic to a web server. If the load gets past a certain threshold, a load balancer can direct incoming traffic to other VMs.

There are two services you can use to balance the load for a VM, or group of VMs (Figure 3.24).

**Figure 3.24: Choose one of two load balancing options, if required.**

#### Load balancing

You can place this virtual machine in the backend pool of an existing Azure load balancing solution. [Learn more](#)

Place this virtual machine behind an  
existing load balancing solution?

#### Load balancing settings

- **Application Gateway** is an HTTP/HTTPS web traffic load balancer with URL-based routing, SSL termination, session persistence, and web application firewall. [Learn more about Application Gateway](#)
- **Azure Load Balancer** supports all TCP/UDP network traffic, port-forwarding, and outbound flows. [Learn more about Azure Load Balancer](#)

Load balancing options \* ⓘ

Azure load balancer

Application gateway

Azure load balancer

Select a load balancer \* ⓘ

Choose one of two load  
balancing services.

The Application Gateway is a service that lets you route traffic to your application based on the web address it is requesting. For example, all traffic going to <https://llamadrama.com/image> can be routed to a specific VM that manages images, and all traffic requesting <https://llamadrama.com/video> can be routed to a VM that processes video.

The Azure Load Balancer doesn't route based on the web address, but rather the type of protocol being used for the traffic, such as TCP and UDP. This has higher performance and lower latency than the Application Gateway, but you can only do load balancing (distributing the traffic to multiple VMs) and

none of the fancy URL magic routing.

For either load balancing service, you need to have created an existing service before creating the VM. We won't do that now, and we will return to load balancing later in the book too. It's a great feature and one that is used all the time to run and manage millions of applications. Finally, click the "Management" button to go to the next tab of the wizard.

## 3.4 Managing a VM

We are almost at the end of creating our VM. I am aware it has taken awhile to get through only 3 tabs on a single wizard, however, I hope you appreciate the many nuances and considerations, before creating a VM for your project. We are almost at the end, and at the point of creating and provisioning the VM, so let's go through the management part of the wizard (Figure 3.25).

**Figure 3.25:** The first part of the Management tab properties.

Configure monitoring and management options for your VM.

#### Azure Security Center

Azure Security Center provides unified security management and advanced threat protection across hybrid cloud workloads.  
[Learn more](#)

✓ Your subscription is protected by Azure Security Center basic plan.

#### Monitoring

- Boot diagnostics  Enable with managed storage account (recommended)  
 Enable with custom storage account  
 Disable

Enable OS guest diagnostics

#### Identity

System assigned managed identity

#### Azure AD

Login with Azure AD (Preview)

RBAC role assignment of Virtual Machine Administrator Login or Virtual Machine User Login is required when using Azure AD login. [Learn more](#)

Leave all these settings as default for now.



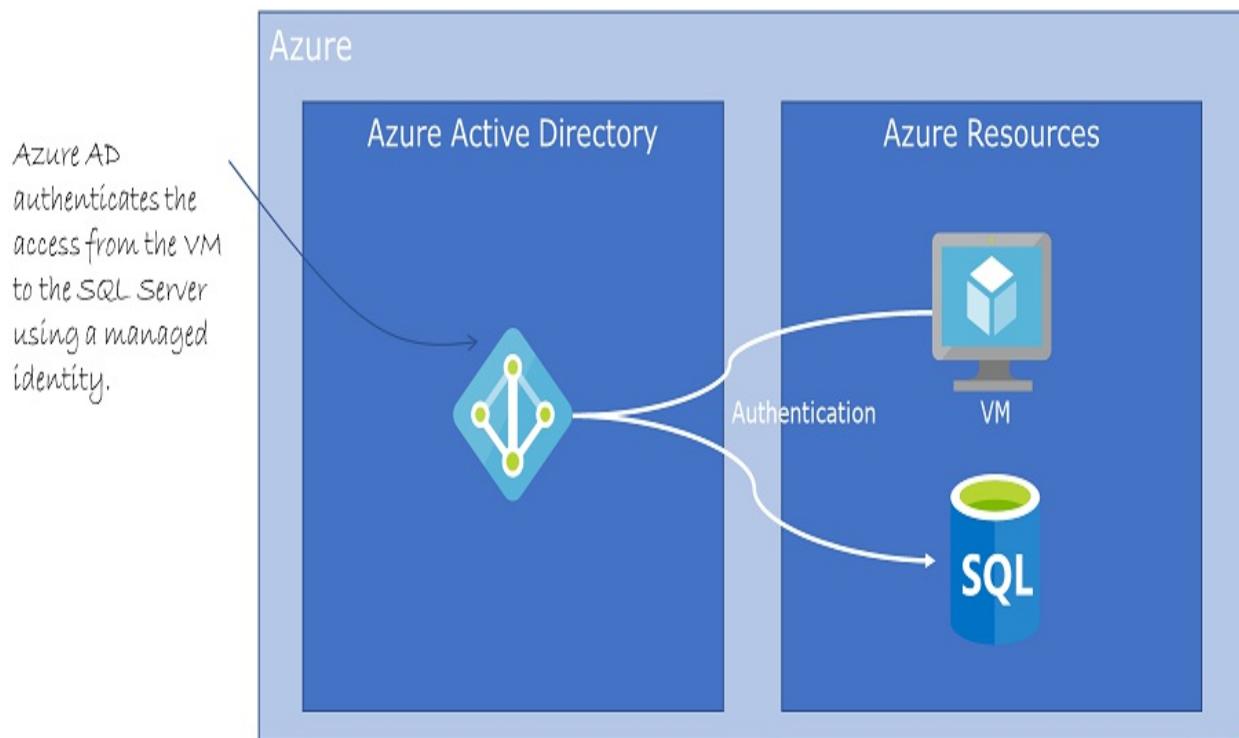
### 3.4.1 Identity

I'll skip over the *Monitoring* section for now, but we will return to the topic later in the book. It is a critical topic for maintenance and performance, but is a part of a much bigger picture. That single checkbox under *Identity* holds a lot more power than you might think at first glance. Imagine seeing Clark Kent on the street in this suit and glasses. Not exactly an intimidating superhero but give him a phone box and flying is done and lasers deployed. If you assign a managed identity to your VM, you open up the world of Azure cloud superpowers.

A managed identity avoids having to manage credentials to other Azure services being connected to from the VM. For example, if you have an application running on the VM, and you need to connect to a database in

Azure, you need to authenticate yourself. This would traditionally mean having a username and password that you had to pass across to the database for authentication. How you stored, managed, and used the password was paramount in keeping the connection secure. With managed identity you assign an Azure Active Directory identity to the VM. And you can then assign that identity access to the Azure database or any other Azure service that supports managed identity as shown in Figure 3.26. It is Azure Active Directory that controls the authentication between services. No more credential management!

**Figure 3.26: Using a managed identity to authenticate access between Azure resources.**

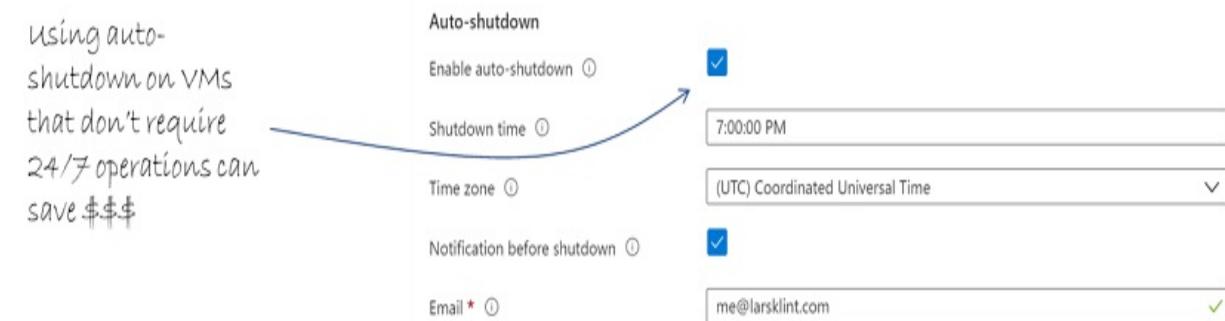


### 3.4.2 Optimizing Costs

As I alluded to earlier when discussing VM sizes, outgoing data and location, managing and optimizing costs for a VM (or any cloud service for that matter) is critical. There are a few relatively straight forward ways to save costs on using a VM on Azure, which are easy to consider and easy to implement. Azure Advisor is one such tool, and comes for free with Azure. However, let's for a minute go back to the Azure Portal and the VM wizard

we are going through in this chapter. The next part is auto-shutdown as shown in Figure 3.27.

**Figure 3.27: Use auto-shutdown whenever you can.**



If the VM doesn't have to be always on, enable auto-shutdown. This can enable the VM to shut down at a specific time every day, such as at the end of the work day. Remember, that you (or your manager) pay for the VM per minute. Every minute the VM is on and not being used is wasted cash that could have been spent on Lego. Or hats.

Next, there are spot instances. You might have noticed back when we chose the size of the VM, you could check “Azure Spot instance”. If you need a development VM or have a transient workload that can be interrupted at any time, Spot instances are a cheap way of getting compute time. Be aware though that Azure can evict the VM at any time they need the capacity or the price of the compute power goes above a threshold you have set. According to Microsoft you can save as much as 90% on costs. More money for Lego. Or hats.

Finally, if you are in for the long haul with your VM usage, use reserved instances. If you can commit to 1 or 3 years of use of the VM, you can get it much cheaper. Microsoft knows you are committed to it long term, and in turn you get a huge discount. And more Lego hats.

Now you can finally click “Review + Create” and set that VM to work. Once the validation has passed you will be shown the price of the VM (Figure 3.28).

**Figure 3.28: Always double-check the VM price is right.**

Create a virtual machine ...

The screenshot shows the 'Review + create' step of the Azure portal's VM creation wizard. At the top, a green bar indicates 'Validation passed'. Below it, tabs for Basics, Disks, Networking, Management, Advanced, Tags, and Review + create are visible, with 'Review + create' being the active tab. Under 'PRODUCT DETAILS', it shows 'Standard D2s v3' by Microsoft, with links to 'Terms of use' and 'Privacy policy'. It also notes 'Subscription credits apply' and lists the price as '0.0459 USD/hr' for 'Pricing for other VM sizes'. A blue callout arrow points from the text 'Double check the VM price, just to make sure.' to the price information. At the bottom, there's a 'TERMS' section with a detailed legal agreement and a 'Create' button.

Always check the price at this point. It is easy to accidentally choose a larger VM without noticing and then your costs will go up a lot faster. Other services like scale sets, backups, choice of disk types and more will also affect price, but that is covered elsewhere in the book. Once you are ready, click the “Create” button and let the VM come to life.

### 3.4.3 Introducing Azure CLI

You might not like me all that much after this section. We have just spent some 20-ish pages on creating a VM. Of course, this was through an educational process, where you learnt about the reason behind a lot of the choices for the decisions we made for our VM, however, it could be done like this as well using the Azure CLI in a command prompt on your favorite operating system:

```
az vm create -n AzureVM -g ActionRG --image UbuntuLTS -vnet-name
```

Before you throw your dummy out of the pram, I did this for you. I wanted you to have the best possible background for how VMs are created and appreciate some of the nuances before I just gave you the “cheat code”. You’re welcome.

From this point on in the book we will continue to use both Azure CLI and PowerShell commands more frequently, as they are a big part of using Azure, day to day. As new services, concepts and implementations come up, we will use a mix of the Portal, the CLI and PowerShell where it makes the most sense. If you want to catch up on the basics of the CLI, make sure to check out the appendices.

## 3.5 Getting the most out of a VM

You now have a VM running in Azure, and it is glorious. Having this IaaS service at your beck and call means you can quickly access powerful compute service through a website. But how do you get the most out of the VM over time? There are tasks and features you can use to make sure you get the best value for your cloud cash.

### 3.5.1 Changing VM type and size

The most immediate way to save, or spend more, money on a virtual machine is the instance type. You can change the size and type of the VM at any time. This means you can start out with the smallest possible size for what you *think* you need to run your application. If you need more resources, scale the VM up. If you need less, scale it down. Figure 3.29 shows a somewhat compressed view of all the many sizes of VMs there are to choose from for our ActionVM.

Figure 3.29: The Portal page for changing VM size and type.

When you change the size of your VM, that is known as vertical scaling. Changing to a larger size is *scaling up*, changing to a smaller is *scaling down*. You can also use horizontal scaling, which is adding or removing VMs to a pool of compute power. This is most often done using a *scale set*, which is a group of VMs that can be put into action when needed and switched off again just as easily.

### 3.5.2 Maintenance and updates

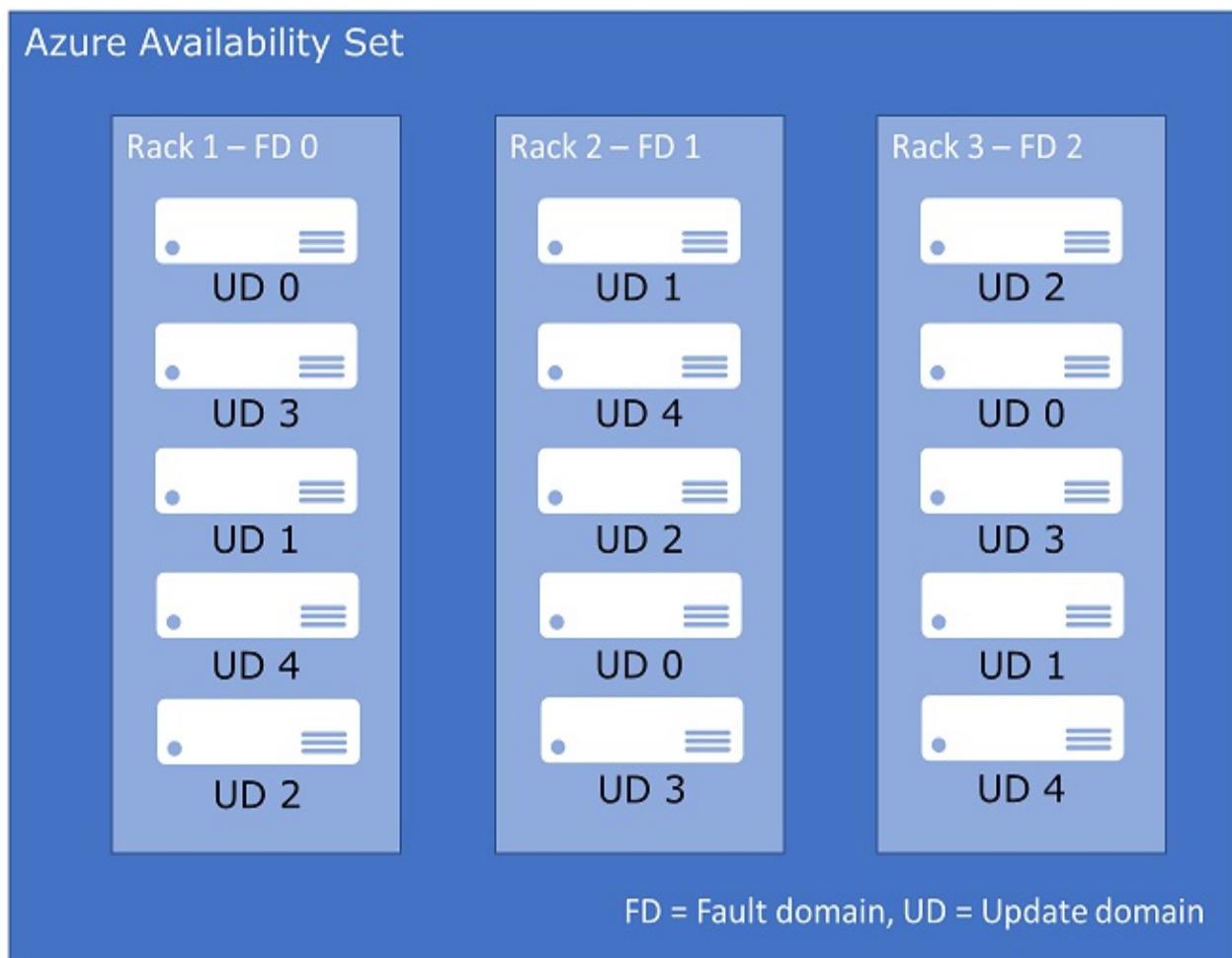
Like any resources on your network, VMs also need a bit of love. Not too much, so you get attached to them, but enough that they run smoothly. There is a saying “treat VMs like cattle, not pets”. While that analogy is a bit severe, it does make sense in the way that resources like VMs can be replaced easily.

When it comes to maintenance, you have to update everything from the

operating system and up. This includes drivers, applications, and anything else you have installed. While you don't control any hardware on Azure, updates to your IaaS solution do depend on how you have configured two things: availability sets and update domains. We covered these features back in Figure 3.13 and Figure 3.14, so let's expand on that knowledge.

When you place a VM into an availability set, it is assigned an update domain and a fault domain automatically as shown in Figure 3.30, where there are 3 fault domains and 5 update domains in the availability set.

**Figure 3.30: An Azure availability set with 3 fault domains and 5 update domains.**



An update domain is a group of VMs that can all be updated and/or rebooted at the same time without any downtime to the greater infrastructure and users won't notice it at all. Only one update domain is ever rebooted at the same

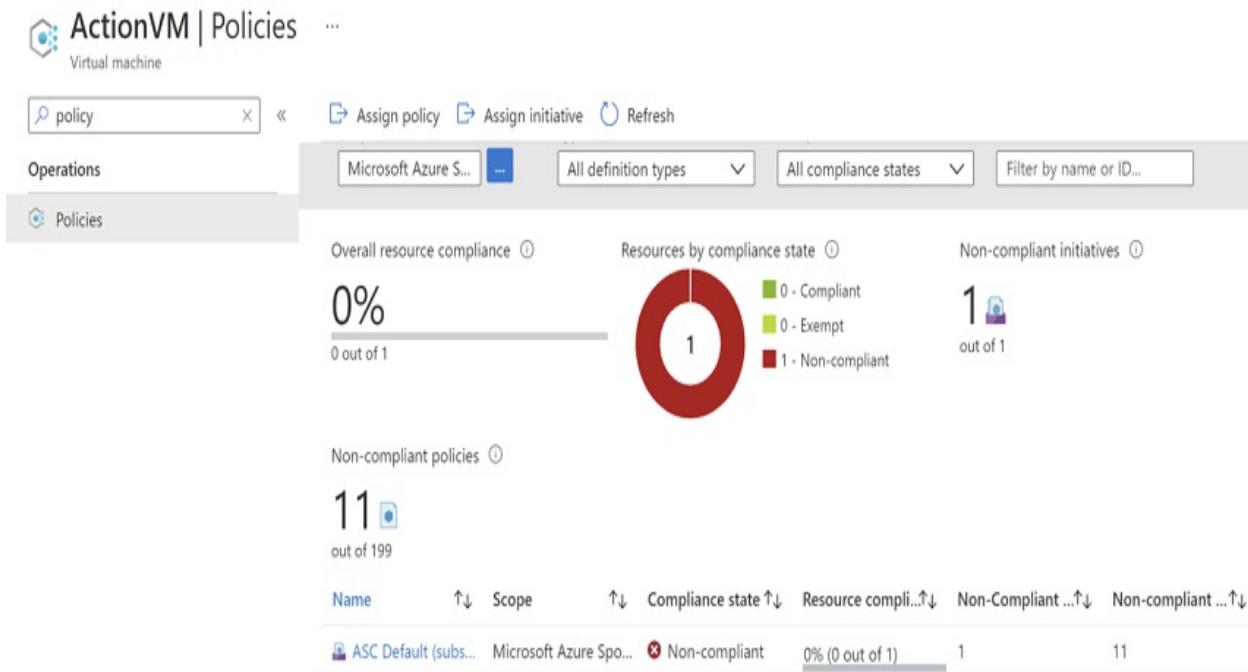
time, and you can have up to twenty update domains for an availability set, if you need it. A fault domain is similar, but the VMs in it share power, cooling and networking. In Figure 3.30 a fault domain is represented as a rack. In a single datacenter these racks would be located physically away from each other. Having more than one fault domain ensures that VMs on separate fault domains aren't affected by disruptions to the power, cooling and networking.

Using availability sets, update domains and fault domains ensure you don't have any downtime during infrastructure faults or planned maintenance, which can be critical. Of course, it does cost more, as you need to have multiple VM instances running, but that might still be worth it depending on how critical the infrastructure is.

### 3.5.3 Policies

To keep a whole team, department, or even company working as one seamless team, make sure the policies and guidelines for the company reflect chosen best practices wherever possible. A policy is a collection of rules and guidelines such as what type of VM you can create, the backup policy and auditing whether managed disks are used or not. When a new VM is created a number of implicit policies can be automatically added. The VM we created in this chapter had a single policy (Figure 3.31), which is already marked as non-compliant.

**Figure 3.31: The single policy assigned to the new ActionVM.**



In this case you can investigate why it is non-compliant and then implement procedures to make sure that doesn't happen again. Examples of policies for virtual machines include:

- Only certain images are used to create new VMs.
- Require the use of managed disks for all VMs. This means Azure manages the storage for the VMs, and no custom virtual hard drives (VHD) are used.
- Only certain regions are allowed for creating resources.

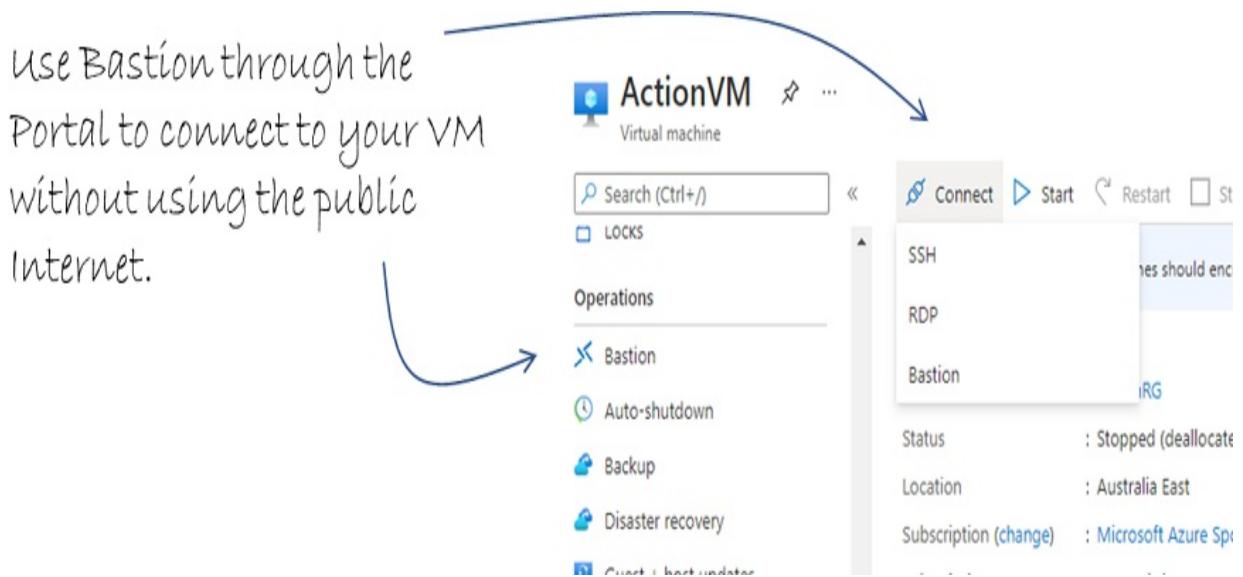
Azure has a lot of built-in policies you can use, and you can create your own custom ones too, by using specific syntax in JSON notation.

### 3.5.4 Azure Bastion

We have already briefly touched on Azure Bastion at the top of the chapter, but it is worth reiterating. To keep your VM as secure as possible, you want to remove access to the public Internet. Often this is not that easy, but if you are only accessing the VM via remote desktop protocol, then you can use Azure Bastion to avoid the public Internet and thus cut off most of the security vulnerabilities for a VM.

By logging in to the Azure Portal and then going to the VM you just created, there is a *Bastion* menu option to connect to the VM (Figure 3.32).

**Figure 3.32: Connect to the VM only using the Microsoft Azure network.**



This will open an RDP connection to the VM in the browser and you can then use remote desktop like you would normally. The only missing feature is copying files directly to the VM from your local machine, but using a cloud storage service such as OneDrive can easily mitigate that.

Had enough of hearing and learning about VMs? Good, because that is all I have for now. A quick summary and then on to the next chapter. Don't forget to delete your VM, or at least turn it off.

## 3.6 Summary

- A virtual machine is one of the three main pillars in infrastructure-as-a-service (IaaS) services on Azure, along with networking and storage.
- You use the remote desktop protocol (RDP) or SSH to connect to a virtual machine to configure and manage it.
- Creating a virtual machine requires planning of location/region, operating system, size, disk types, networking, and security. Any of these properties can significantly impact performance, cost, and overall application success.

- Without a network connection, there is no use of a VM. Creating or using the right virtual network is critical to both the performance and security of the VM.
- There are several ways to save costs on using a VM, such as auto-shutdown, reserved instances and managing the size and power of VMs.
- The Azure CLI or PowerShell can be a much more efficient way of creating and managing VMs, once you understand how they work and which features influence the performance and cost of them.
- If you need to change the VM type or location, you can do this at any time.
- You can use fault domains and update domains inside an availability set to mitigate the risk of outages for your application.

[\[1\]](#) See Appendix 1 for an introduction to the Azure CLI and Azure PowerShell

# 4 Networking in Azure

## This chapter covers

- Creating and using virtual network, subnets and network security groups
- Using best practices for creating and managing virtual networks
- Distributing traffic from an outside network to internal resources

Where compute is the brains of a cloud project, the networks that combine them are the arteries and blood lines. Just like computing, you need networking for anything you do with cloud computing on Azure. There is no choice. At times the network parts of your infrastructure will be abstracted away from you, but they are still there.

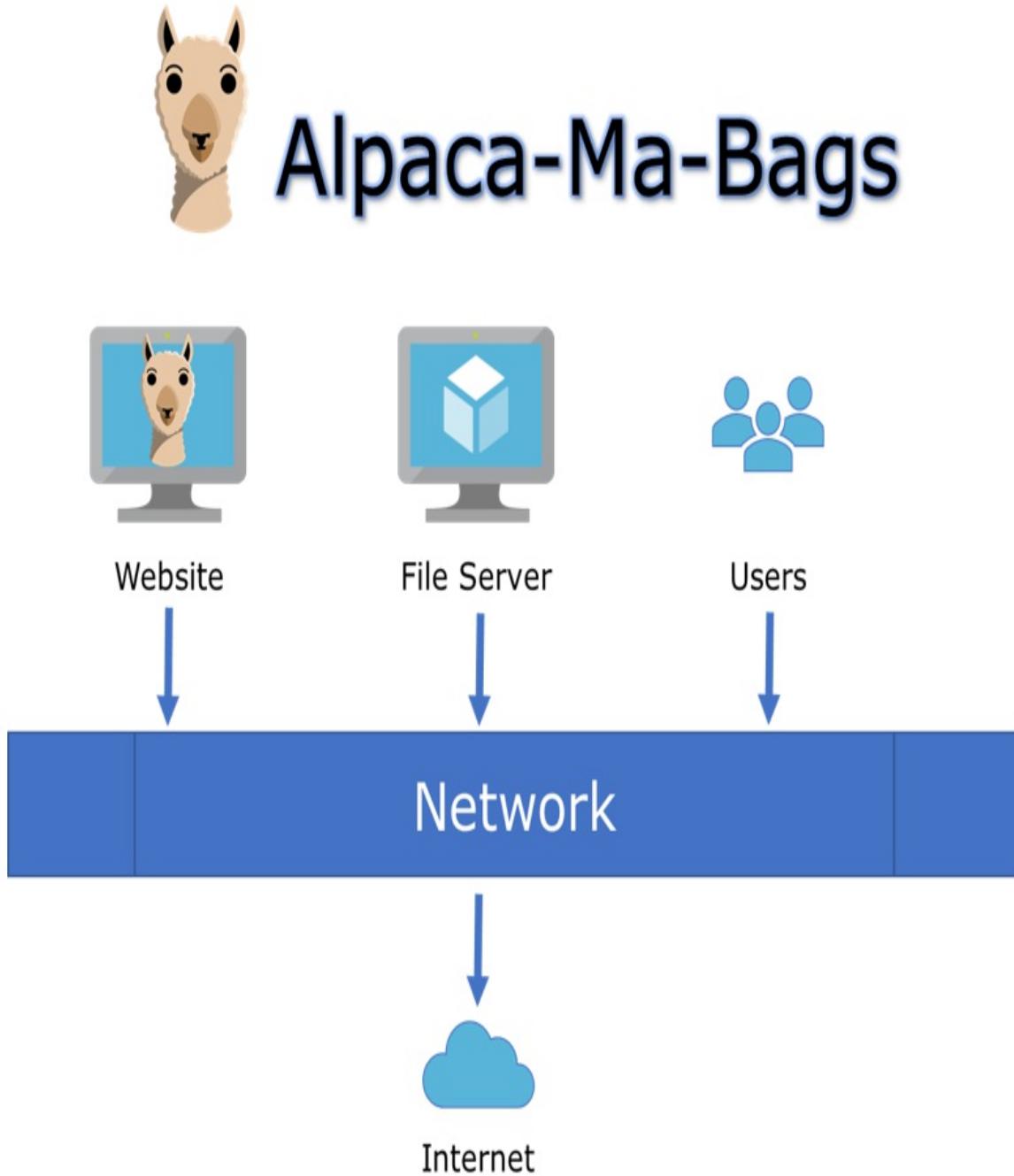
In this chapter we are going to focus on the virtual networks in Azure that you can control, manage, and make jump through hoops made of IP address ranges. When managed effectively, virtual networks in Azure can secure against intruders, optimize data efficiency, help with traffic authorization, and form part of your applications internet footprint.

## 4.1 What is a virtual network?

When two resources on Azure need to communicate securely with each other, a resource needs to send and receive data with the public Internet, or a service has to communicate with an on-premises network, you have to use a virtual network. Just like cars need roads to drive on, cloud data from services need virtual networks to travel through. Azure virtual networks also embody the very foundations of cloud computing we discussed in chapter 1: reliability and scalability.

The company Alpaca-Ma-Bags is selling clothing made from Alpaca wool. They have a traditional on-premises setup with a server that hosts their website, a server with company files and a simple network infrastructure (Figure 4.1).

Figure 4.1: Alpaca-Ma-Bags network overview.



The company is wanting to expand their product line to include a new range of Alpaca conditioners and hairstyling products. This will complement the current range, and the plan is to use Microsoft Azure for this. In this chapter

we will go through various steps and features to enable Alpaca-Ma-Bags to thrive in the cloud.

First, let's introduce a new tool, or at least new in this book, to create a virtual network: PowerShell. I won't go through the setup and connection to Azure with PowerShell though. If you are not sure how to install or get started with PowerShell as well as talking to Azure through it, check out the Appendices in the book. It will explain the basic syntax and composition of cmdlets, as well as introduce you to PowerShell as a tool. With that out of the way, the first step for creating any resource on Azure, as you learned in the previous chapters, is creating a resource group. I'll show you how to in PowerShell this time, but then I will assume you have a resource group ready for the examples in the rest of the book. Deal? Awesome. Let's power some shell. Once you are logged into Azure with PowerShell, run this command to set up a new virtual network:

```
New-AzResourceGroup -Name 'AlpacaRG' -Location 'Australia So
```

Once you have a resource group, let's add a virtual network to it, also using PowerShell.

```
New-AzVirtualNetwork -Name AlpacaVNet -ResourceGroupName 'AlpacaR
```

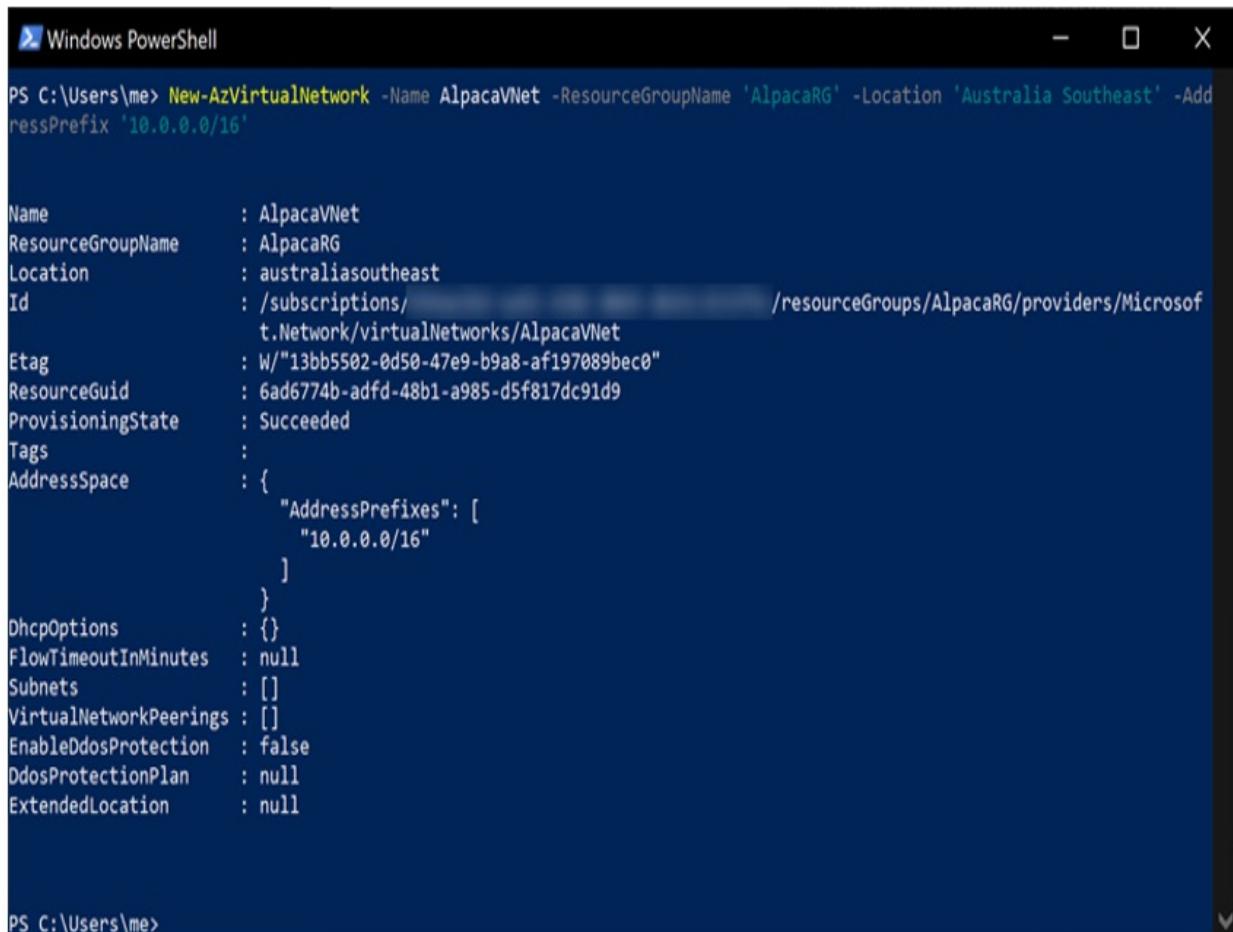
This command contains the minimum number of parameters you can explicitly set to create a valid PowerShell command for creating a VNet. They are:

- Name: This is the name of the VNet, which is up to you. It is unique within your Azure subscription.
- ResourceGroupName: As the name states, this is the resource group you want to place the VNet inside. In this case we use AlpacaRG, which follows the naming strategy of <project><resource type>. However, there can be many other naming strategies for resources in Azure. The important thing is that you decide on a single one and then follow and mandate it throughout. See the book appendices for more information on this.
- Location: The physical region for the VNet. Remember this does not have to be the same location as your resource group.

- AddressPrefix: This is the range of IP addresses available in the VNet, in this case 65536 addresses, but more on that shortly.

This will spit out the output shown in Figure 4.2, which shows a few properties for the VNet, including Id and name.

**Figure 4.2: Creating a virtual network with PowerShell.**



```
PS C:\Users\me> New-AzVirtualNetwork -Name AlpacaVNet -ResourceGroupName 'AlpacaRG' -Location 'Australia Southeast' -AddressPrefix '10.0.0.0/16'

Name          : AlpacaVNet
ResourceGroupName : AlpacaRG
Location       : australiasoutheast
Id             : /subscriptions/.../resourceGroups/AlpacaRG/providers/Microsoft.Network/virtualNetworks/AlpacaVNet
Etag           : W/"13bb5502-0d50-47e9-b9a8-af197089bec0"
ResourceGuid   : 6ad6774b-adfd-48b1-a985-d5f817dc91d9
ProvisioningState : Succeeded
Tags           :
AddressSpace   : {
    "AddressPrefixes": [
        "10.0.0.0/16"
    ]
}
DhcpOptions    : {}
FlowTimeoutInMinutes : null
Subnets        : []
VirtualNetworkPeerings : []
EnableDdosProtection : false
DdosProtectionPlan : null
ExtendedLocation : null

PS C:\Users\me>
```

Aren't you happy now that we don't have to click click click through the Azure Portal wizard? With just a couple of commands in PowerShell, we have a virtual network at our disposal. The Azure Portal and the PowerShell cmdlets talk to the same API endpoint, meaning the exact same information is passed on to Azure whichever tool you use. If you aren't sure, go check the Azure Portal and inspect the VNet there.

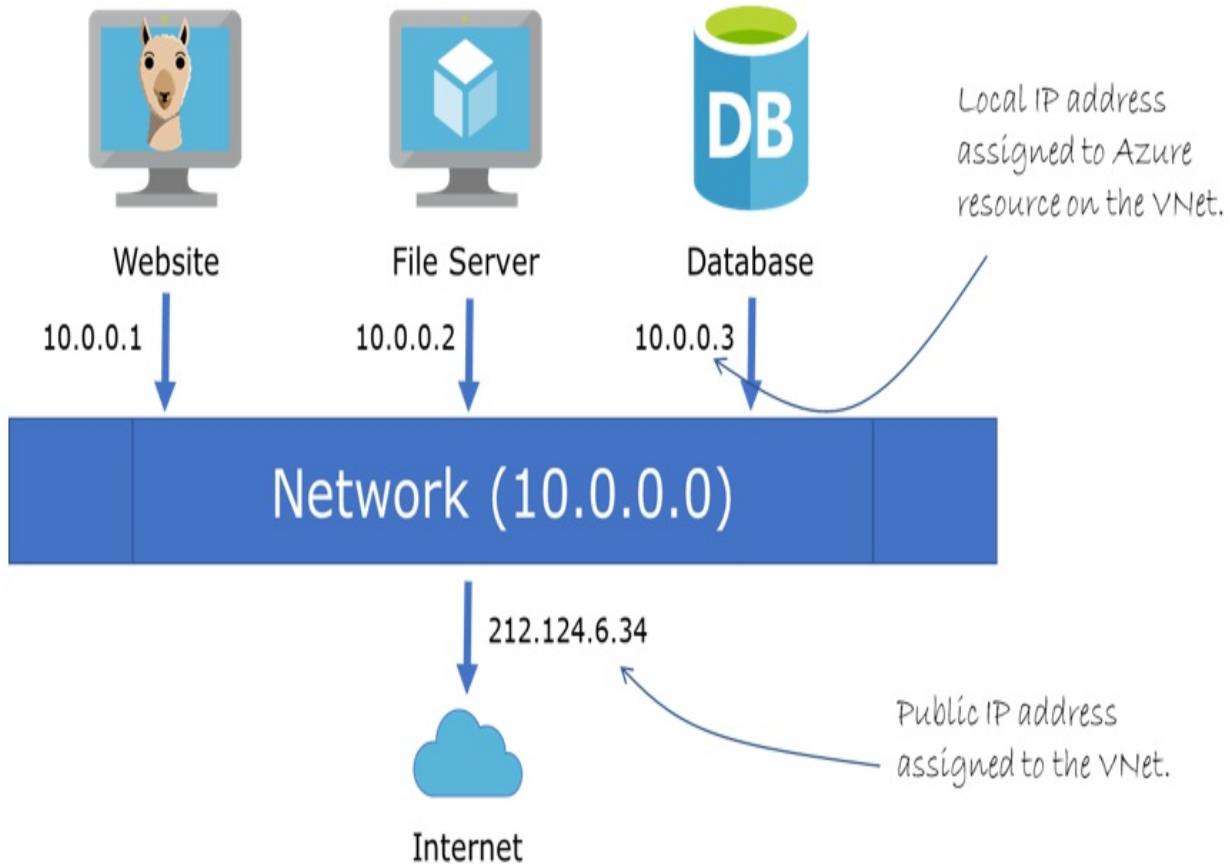
As you might have heard before, choosing the right tool for the job is often a

critical step, and PowerShell can often be that tool. Anything you can do in Azure, you can do with PowerShell, including creating and managing virtual networks. Next, we do just that, and delve into subnets.

## 4.2 Subnets

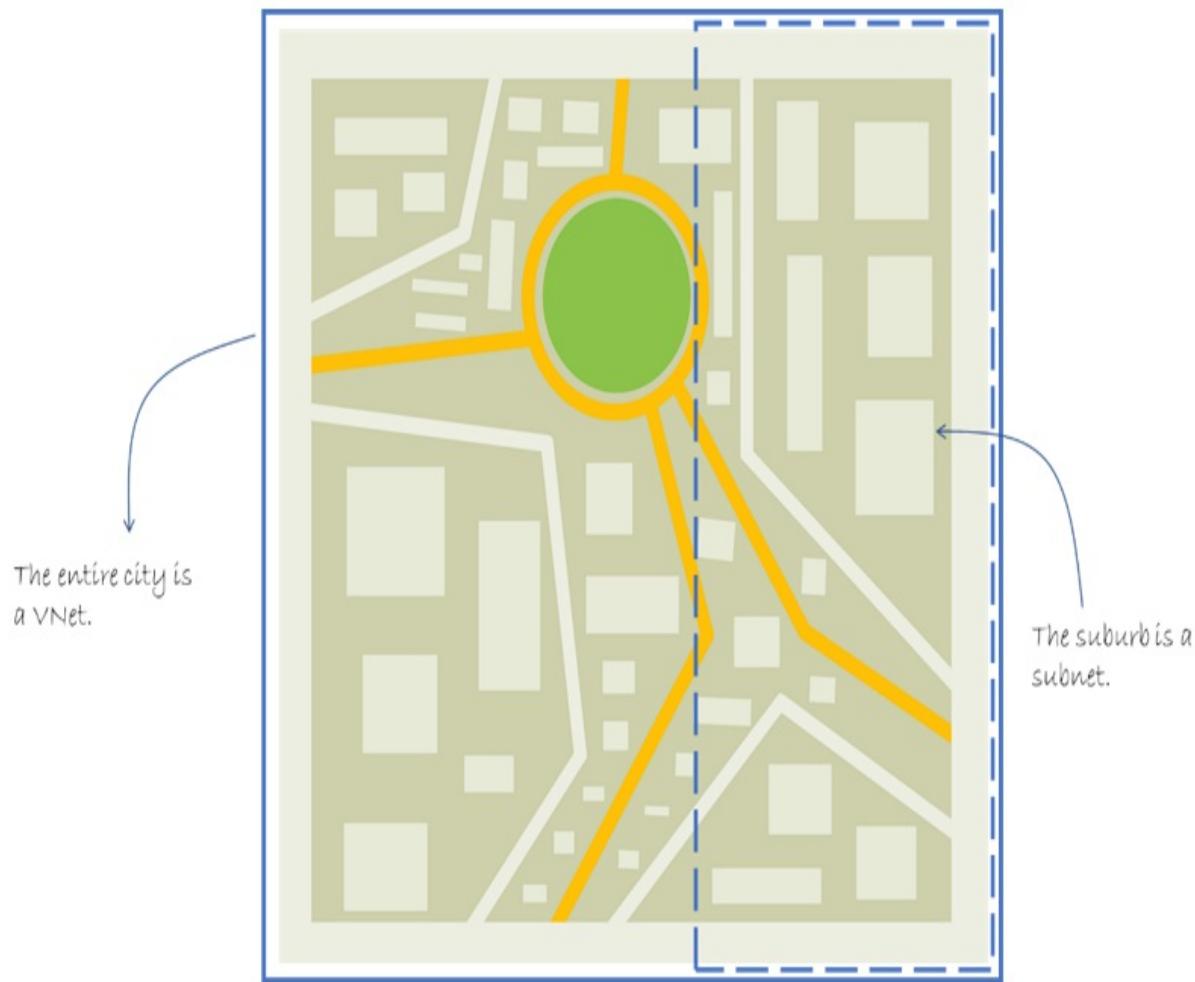
Any device that wishes to connect to the internet or any other network, must have an IP address. When you connect your smartphone to the public Internet, that device is assigned an IP address, which is how data can be sent to the phone. However, there is only a finite number of IP addresses (roughly 4.3 billion)<sup>[1]</sup> and there are way more devices than there are public IP addresses. This problem is in part solved by using local networks that have their own IP address range, and if a device needs an Internet connection only a single, or very few, IP address is needed for the entire local network. In Figure 4.3 the Alpaca-Ma-Bags devices are assigned local IP addresses in the IP range 10.0.0.0, which is the local address space on the local network. When, and if, the devices need to send traffic through the public Internet, they will use the public IP address assigned to the VNet.

**Figure 4.3: Devices on a VNet get local addresses, and traffic to the public internet get a public IP address.**



A subnet is a logical division of the virtual network, where a subnet gets a range of IP addresses allocated, and those IP addresses fall within the total address range of the VNet, such as 10.0.0.0 in Figure 4.3. If you think of the entire VNet as the road network of a city (Figure 4.4), then a subnet is the road network in a suburb. It is a logically defined part of the road network (roads within a suburb), and it is still part of the entire road network (the VNet).

**Figure 4.4: Comparing a VNet and a subnet to a city and suburb.**



In short, a subnet is a range of IP addresses within a VNet, and no two subnets' IP address ranges can overlap. With that clear, let's talk IP network classes. You might think that IP addresses are just ranges of numbers that fall within defined binary ranges, however, that is only half the story. There are rules and standards for how IP addresses are used. One of those says that there are three defined classes of networks that can be used to create private subnets, as you can see in Table 4.1. These three network classes, A, B and C, are the only ranges you can use for creating private networks in Azure.

**Table 4.1: The three classes of networks for private subnets.**

Network Class	IP Address Range	No. Networks Possible	No. Devices
---------------	------------------	-----------------------	-------------

A	10.0.0.0 - 10.255.255.255	1	16,777,216
B	172.16.0.0 - 172.31.255.255	16	1,048,576
C	192.168.0.0 - 192.168.255.255	256	65,536

From this table you can deduct that it is important to plan how many subnets and which network class you are going to use. In our example in Figure 4.2 we are using a class A network of 10.0.0.0, which only offers us a single network choice, but you can associate more than 16 million devices with it. If you then split up the network in logic subnets, you can have a huge network, which can be a nightmare to manage but offers plenty of options. Again, planning how you intend to use the network is crucial before choosing a network class.

It is a common pitfall to use the wrong network class without considering how many networks or devices are needed for the future of the system.

To create a subnet in Azure using PowerShell, use the following two commands

```
PS C:\> $vnet = Get-AzVirtualNetwork -Name "AlpacaVNet" -Resource
PS C:\> Add-AzVirtualNetworkSubnetConfig -Name hairstyleSubnet -V
```

At this point the subnet is only defined in memory on the instance running PowerShell. The subnet is not yet created on Azure. For that change to be provisioned, use this cmdlet

```
PS C:\> $vnet | Set-AzVirtualNetwork
```

This creates a subnet *hairstyleSubnet* within the *AlpacaVNet* virtual network, which is expressed as the variable *\$vnet*. The parameter *AddressPrefix* defines the address range to be 256 addresses in the subnet, using CIDR

notation. More on that shortly, but it is first important to understand that subnets need a range of IP addresses that can be assigned to the devices attached to it. Each device on a network must have a unique IP address on that network to create a logical separation. More on that too, in just a minute.

### 4.2.1 Classless Inter-Domain Routing

To efficiently divide a network into subnets, it requires a common way to describe these subnet ranges. The Classless Inter-Domain Routing, or just CIDR, notation is a method for expressing IP address ranges within network. This notation is widely used and looks like this

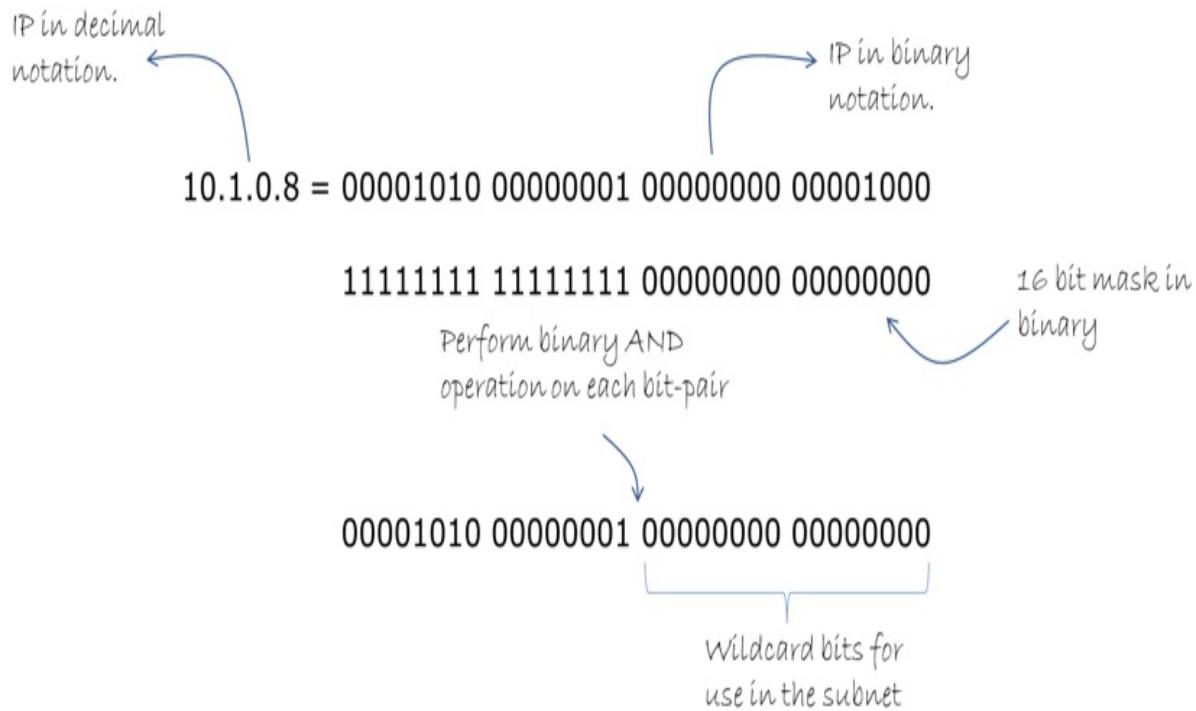
10.1.0.8/16

To make sense of how that works, it requires a little bit of binary mathematics. The CIDR notation consists of two parts: the IP address, which is the *10.1.0.0* portion above, and a mask, which is *16* here. To understand how that mask is applied we are using the AND binary operation, which is shown in Table 4.2.

**Table 4.2: AND binary operation.**

Operation	Result
0 AND 0	0
0 AND 1	0
1 AND 0	0
1 AND 1	1

**Figure 4.5: Applying the mask defined in the CIDR notation.**



Applying the AND binary operation to each bit pair in the IP address and mask (Figure 4.5), you end up with a possible IP address range from 10.1.0.0 to 10.1.255.255, or 65536 different hosts. To put it into context, for a virtual network with a subnet of 10.1.0.8/16, you can have 65536 different devices on that subnet. Well, almost. Azure reserves 5 IP addresses within each subnet of size /25 and larger. For smaller subnets Azure reserves only 4. These reserved IP addresses are used for internal Azure purposes to manage and run the VNet.

CIDR is a great way to improve the efficiency of IP address distribution, and it is a critical part of defining and describing your Azure networks. Understanding the meaning and impact will help you design and manage better networks in Azure.

## 4.2.2 Logical separation

As you might have guessed there are a couple of good reasons for subnetting a virtual network. Yeah, that is what it is called. Subnetting. The first is to use

subnets as logical separation of your Azure resources. This can be for creating a separation between resources used for different projects, for departmental separation, to wall off certain services from others, to have different security requirements for resources in various subnets and much more.

Certain PaaS services (which was introduced in chapter 1) on Azure require a subnet to be delegated to them to operate fully. For example, you can assign, or delegate, a subnet to the resource provider for Azure SQL database, *Microsoft.Sql\managedInstance*. This logic subnet will then be controlled by the assigned resource provider for any instances of the corresponding resources. Alpaca-Ma-Bags need to use an Azure SQL instance to hold the data for the conditioner product. There are a lot of variables to making Alpacas look fabulous. By delegating the subnet we created before, *hairstyleSubnet*, Azure SQL instances can create resources within that subnet that they need, such as policies. Run the PowerShell shown in **Error! Reference source not found.** to delegate the subnet.

```
PS C:\> $vnet = Get-AzVirtualNetwork -Name "AlpacaVNet" -Resource
PS C:\> $subnet = Get-AzVirtualNetworkSubnetConfig -Name "hairsty
PS C:\> Add-AzDelegation -Name "hairstyleDelegation" -ServiceName
PS C:\> Set-AzVirtualNetwork -VirtualNetwork $vnet      #D
```

Another takeaway from the PowerShell commands above is the concept of chaining of PowerShell commands, where one statement is chained to the next by using variables such as *\$vnet*. It can make the approach more manageable, as you split each action into its own line of PowerShell action. In **Error! Reference source not found.**, we delegate both the VNet and subnet to variables, which we can use later.

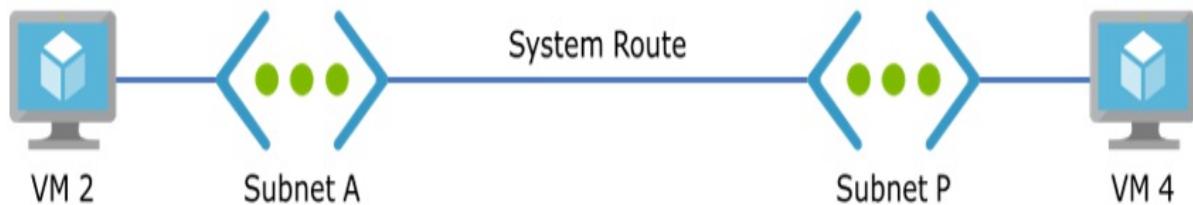
Segmenting the VNet into smaller subnets can help organise devices and services logically, but also improve data transfers. Don't make the subnets too small though, as most organisations always end up adding more resources than they initially planned on. Re-allocating addresses in a VNet is hard work and takes forever.

### 4.2.3 Routing optimization

The other equally important reason for using subnet is the Azure routing optimization that happens as well. Although you don't have to explicitly set anything up for this to happen, it is a fundamental part of how Azure networking works.

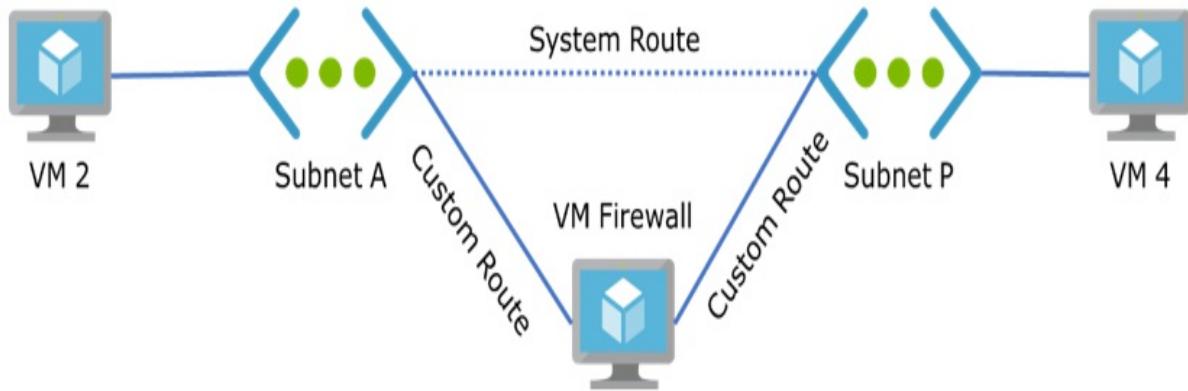
Azure automatically adds system routes for each subnet in a VNet to a route table, as well as adds IP addresses to each device. You can't create system routes, nor can you delete them, but you can override them using custom routes and subnet specific route tables. What are routes, you might ask? They are the optimal path from one endpoint to another across a network. In Figure 4.6 subnet A has a route defined to subnet P, so when VM 2 needs to talk to VM 4, the route ensures that happens using the most efficient path.

**Figure 4.6: System route connecting Subnet A and Subnet P.**



Imagine going to a specific address in your city with no map, GPS, or road signs. How would you ever find the optimal route across the city? Network routing is similar and a route in a route table (which is where routes live) describes the optimal path to an endpoint, which can be a device or network. The implicit system routes for any subnet you create ensures traffic gets to that subnet on the optimal route, thus optimizing your network efficiency. A custom route is when you need to stop off at a certain shop for every trip across the city, or, for Azure, when you need to go through a specific VM that runs a firewall when using the Internet. In Figure 4.7, the custom route overrides the system route to make sure traffic goes through the firewall VM.

**Figure 4.7: Custom route make sure the traffic goes through the Firewall VM.**



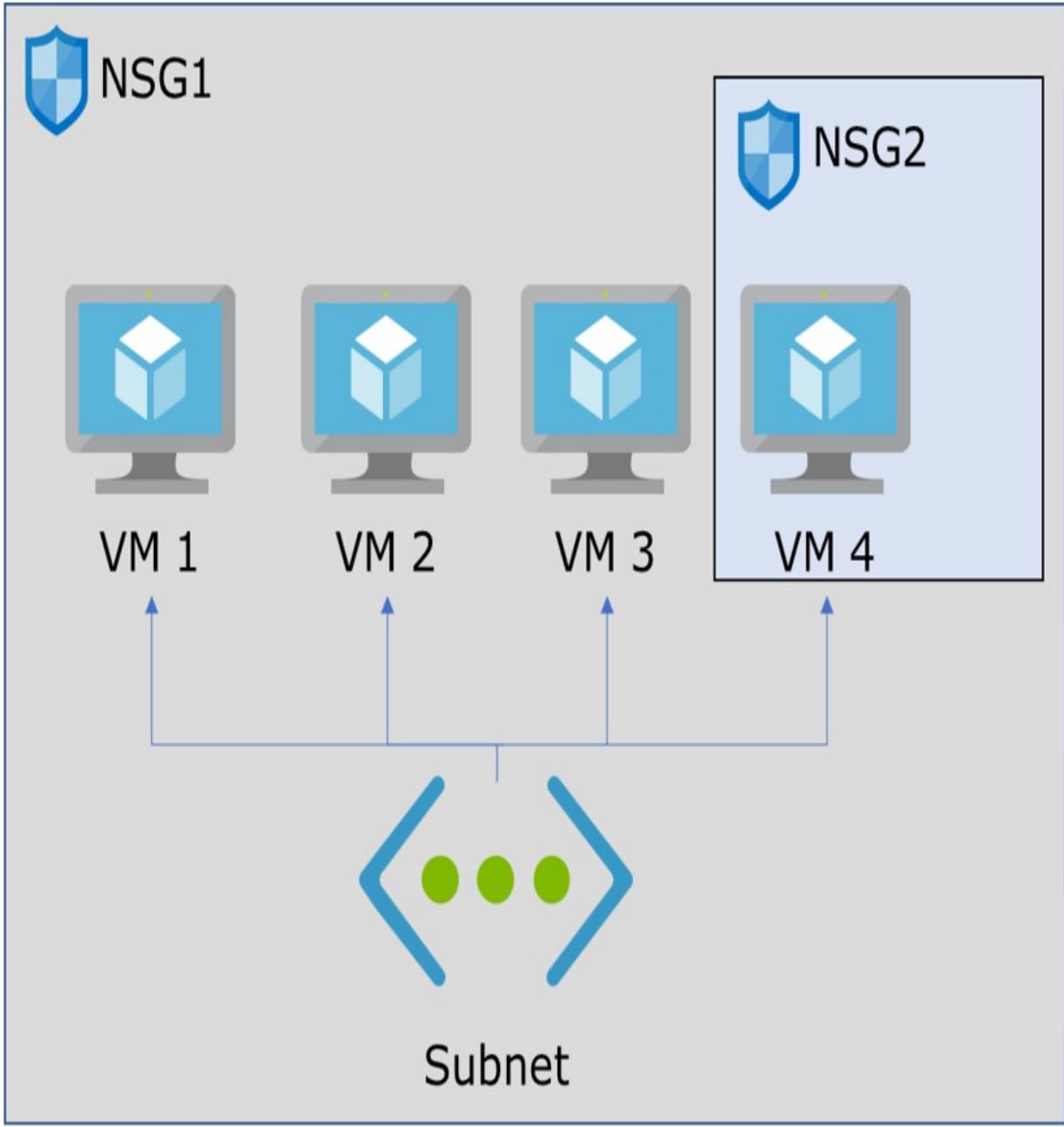
It isn't enough to have optimal network routes though. You also have to make sure they are adequately secure for your purposes.

## 4.3 Network Security Groups

There once was virtual network living in the cloud. Every day it would route packets of data to and from its little subnets and all was well. Then one day, a VM on one of the subnets was exposed to the evil dark clouds called the public internet, and with no protection for the virtual network it was infected with evil packets of data, and all was lost. The end.

A network security groups, often just referred to as NSG, is part of the security blanket that you use on Azure to save your virtual networks and subnets from packets of evil data. You can assign an NSG to either a network interface card (NIC), which is for a single VM, or a subnet, which is a range of IP addresses (Figure 4.8).

**Figure 4.8: NSG1 applies to all VMs in the subnet, but NSG2 only applies to VM4.**



At the core of an NSG is a list of rules that filters what kind of traffic comes in and out of the network or device the NSG is attached to (Table 4.3). Where the Azure routing tables defines *how* traffic goes from one end to the other, NSGs define *what* traffic is allowed to arrive or leave the endpoints of that route.

**Table 4.3: Inbound security rules for a default NSG.**

Priority	Name	Port	Protocol	Source	Destination
300	SSH	22	TCP	Any	Any
65000	AllowVNetInbound	Any	Any	Virtual Network	Virtual Network
65001	AllowAzureLoadBalancer InBound	Any	Any	Azure LoadBalancer	Any
65002	DenyAllInbound	Any	Any	Any	Any

The tables of inbound and outbound security rules are what NSGs are all about. Before we create our own NSG, let's inspect what a rule consists of, and the relationship between rules.

- Priority: a value between 100 and 4096. Rules are processed one at the time from lowest number (highest priority) to highest number (lowest priority). When a rule matches the traffic, and an action is taken, no other rules are processed for that data packet. The three rules with priority 65000-65002 are default rules that can't be removed.
- Name: The name of the rule. While you can name a rule whatever you like, it is highly recommended the name reflects the rule such as AllowVnetInbound.
- Port: The port of the traffic. This is between 0 and 65535, or Any.
- Protocol: Use TCP, UDP, or Any.
- Source & Destination: Any, an individual IP address, a CIDR range of addresses or a service tag. An Azure service tag defines a group of Azure services, such as AppService, AzureLoadBalancer or Sql, and they are managed by Microsoft. This avoids hard coding IP addresses, which may change later.

- Action: Allow or Deny.

You can have zero rules in an NSG or as many as you'd like. **Caution:** Be very mindful of the number of rules, how they overlap, and how they are prioritized. It is exceptionally easy to make a complete mess of it, where you lose track of which rules are invoked when and if your network or device is indeed secured. Some tips on how to manage this coming up shortly, but first let's create a new NSG for the subnet we created before.

```
PS C:\> $nsg = New-AzNetworkSecurityGroup -Name "nsgAlpaca" -Reso
```

This will create our new NSG with a bunch of output in PowerShell. This is common for most PowerShell commands, where you are shown the properties for a new or updated resource. In this case, most of the output is relating to the security rules that are created by default. However, we want to attach the NSG to our subnet *hairstyleSubnet* to put it to work. That is done with this beautiful piece of PowerShell code, which may look slightly familiar.

```
PS C:\> Set-AzVirtualNetworkSubnetConfig -Name hairstyleSubnet -V  
PS C:\> Set-AzVirtualNetwork -VirtualNetwork $vnet
```

The main difference in this piece of PowerShell code from when we created the subnet above, is that we use the *Set* variant of the *AzVirtualNetworkSubnetConfig* cmdlet, rather than the *Add*. This is because it is an update to an existing subnet, rather than creating a new one. Apart from that we are associating the network security group, and then provisioning the changes.

Consider creating subnets that align with the network security groups you want to design. If you have a group of VMs that can only be connected to via RDP and another that can only be used with Bastion, then create two subnets. The NSG for each group can be clearly defined and associated with a subnet each.

### 4.3.1 Creating rules

We now have a VNet with a subnet, and an NSG that is associated with that

subnet to protect any services within it. Say you have some VMs attached to that subnet, and you want to adjust what traffic can access those VMs, and what traffic they can send. That is all done through the NSG by creating rules. Let's create a rule that allows a specific IP address *my.ip.address* to access the VNet via remote desktop protocol (RDP) on port 3389.

```
PS C:/> Add-AzNetworkSecurityRuleConfig -Name 'allowRDP' -Network  
PS C:/> Set-AzNetworkSecurityGroup
```

As I mentioned before, NSG rules are one of the most important parts of your network to get right, and at the same time they are easy to mess up. However, there are some best practices you can follow to make it much more manageable and stay on top of the rules.

- Apply an NSG to the broadest range of devices or services, meaning apply to a subnet whenever possible, then to a VM. This will limit the number of NSGs needed, hence the number of rules you need to keep track of.
- Be intentional about applying the rules. Have a plan, purpose and scenario for each rule that makes it count.
- Use service tags wherever possible instead of IP addresses or ranges.
- Simplicity is your friend. Keep rules simple and have a single set of rules for a specific subnet. If some services need a new set of rules or changes to existing ones, then put those services in a new subnet.
- Increment rules by 100. First rule to be evaluated will have a value of 100, the second 200, third 300 and so on. If you decide you need a rule between 100 and 200, then you can give it 150, and still have room to place more rules in between. This avoids having to suddenly shuffle all rules to get your hierarchy to work.
- Don't assign an NSG to a NIC directly, unless absolutely necessary. Having an NSG on both the subnet and a single NIC/VM can be very confusing. Rules are evaluated first on the NIC for outbound traffic, then the subnet, and vice versa for inbound.

Having firm guidelines for your NSG rules will pay off exponentially and make your life much easier, and secure. Now that the subnet, and VNet, is secured it is time to make it talk to other networks.

## 4.4 Connecting Networks

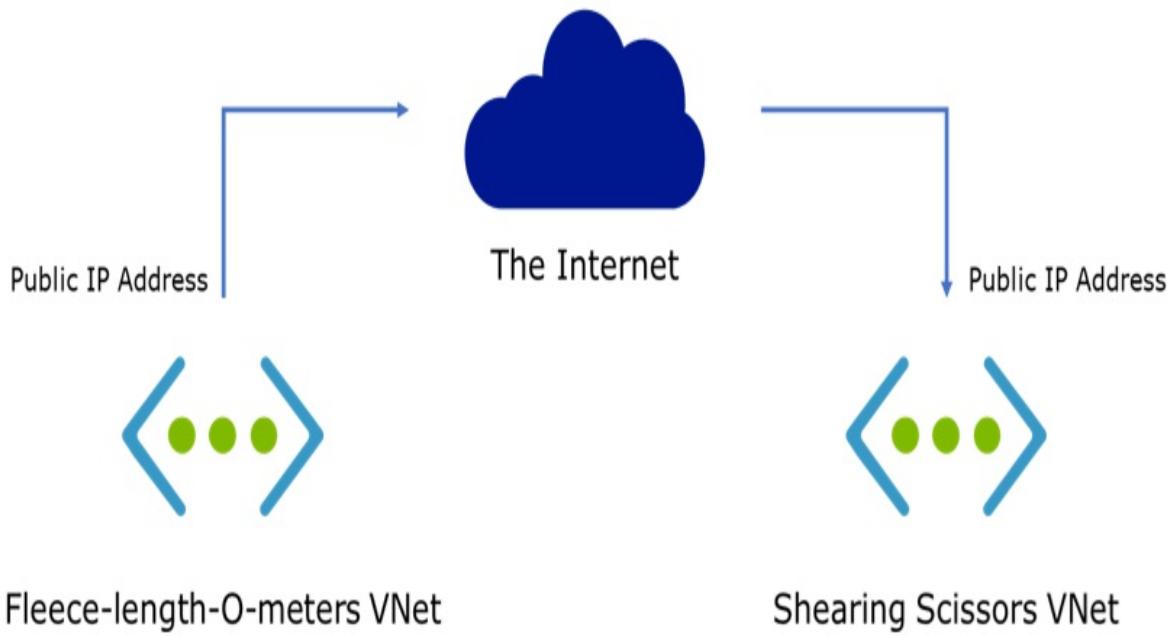
What is better than one VNet? Many of course! It is unlikely you will have just a single network in your Azure infrastructure, and as soon as you have more than one, they'll want to talk to each other. They will be like teenagers on the phone, discussing the latest pop music all night long, and there is no stopping them, but I digress.

Azure offers a range of ways to connect networks that need to communicate and share data. Depending on the kind of networks, and the kind of connection needed, there are options for your connecting enjoyment.

### 4.4.1 VNet peering

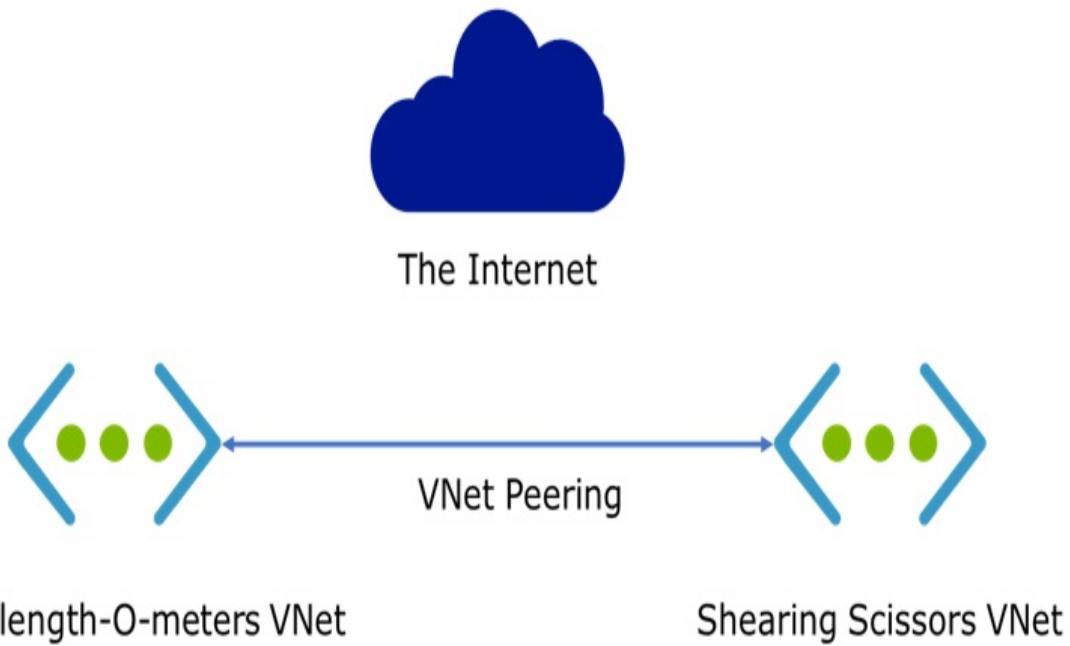
There are two main new products at Alpaca-Ma-Bags, which are hosted on Azure. One manages the fleece length that alpacas can be shorn to, and one sells alpaca shearing scissors. Each product has a virtual network with VMs, databases and other services to make the products function. Currently, if the shearing scissor products need data from the fleece data, a connection is made through public endpoints over the internet using a public IP address on either network as shown in Figure 4.9.

**Figure 4.9:** Two VNets connecting via the public Internet.



A much better way to connect the two networks is using VNet peering. This Azure feature creates a direct link between two VNets using the Azure backbone network. Not only is the traffic between the two VNets never entering the public internet, but you also get the reduced latency and higher throughput on top. The two VNets will function as if they are one and the same, with devices attached to either being able to communicate across both VNets (Figure 4.10). You pay for both the egress and ingress traffic for both end of the peering, which can make it more expensive than using the public Internet.

**Figure 4.10: Peering two VNets to get lower latency and higher throughput.**



To set up a VNet peering using PowerShell, use the following cmdlets and parameters. First, we need to create a second VNet to connect to though.

```
PS C:\> $vnet2 = New-AzVirtualNetwork -Name ShearingVNet -Resourc
```

And now we can then connect the two VNets using peering. We have to create the peering in both directions, so there are two parts to the process.

```
PS C:\> Add-AzVirtualNetworkPeering -Name AlpacaShearingPeering
PS C:\> Add-AzVirtualNetworkPeering -Name ShearingAlpacaPeering
```

Once both directions have been peered, the output in PowerShell will show *PeeringState* to be **Connected**, which means you have a successful peering (Figure 4.11).

**Figure 4.11:** PeeringState property is “Connected” when successfully peered.

```

Windows PowerShell

PS C:\Users\me> Add-AzVirtualNetworkPeering -Name ShearingAlpacaPeering -VirtualNetwork $vnet2 -RemoteVirtualNetworkId $vnet.Id

Name          : ShearingAlpacaPeering
Id           : /subscriptions/694ae2bd-ea92-4302-8069-db43c3533f9c/resourceGroups/AlpacaRG/providers/Microsoft.Network/virtualNetworks/ShearingVNet/virtualNetworkPeerings/ShearingAlpacaPeering
Etag         : W/"484e0210-7dec-49a4-b0bb-bf77fb92e7b0"
ResourceGroupName : AlpacaRG
VirtualNetworkName : ShearingVNet
PeeringSyncLevel : FullyInSync
PeeringState    : Connected
ProvisioningState : Succeeded
RemoteVirtualNetworkId : /subscriptions/694ae2bd-ea92-4302-8069-db43c3533f9c/resourceGroups/AlpacaRG/providers/Microsoft.Network/virtualNetworks/AlpacaVNet
AllowVirtualNetworkAccess : True
AllowForwardedTraffic : False
AllowGatewayTransit : False
UseRemoteGateways : False
RemoteGateways : null
PeeredRemoteAddressSpace : {
    "AddressPrefixes": [
        "10.0.0.0/16"
    ]
}
RemoteVirtualNetworkAddressSpace : {
    "AddressPrefixes": [
        "10.0.0.0/16"
    ]
}

PS C:\Users\me>

```

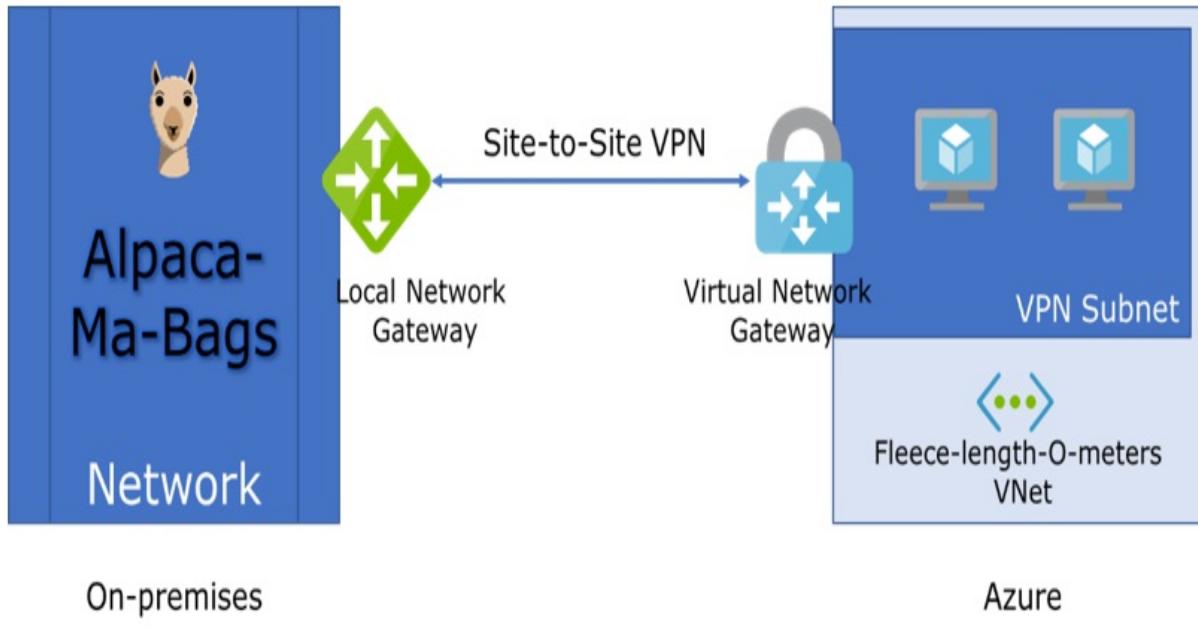
If successfully peered, VNet PeeringState is “Connected”

You can set up VNet peering within the same region, or across regions. You can even use peering across Azure subscriptions and Azure active directory tenants. If you have two separate VNets in Azure that you want to connect, use VNet Peering. You can also use the Azure Portal to set up the peering, of course, and that can sometimes be convenient too.

#### 4.4.2 VPN Gateway

What if you have two networks you want to connect, but only one of them is hosted on Azure, and the other is in your own data center? For that, there is the VPN Gateway, which is one of two types of network gateways in Azure, the other being ExpressRoute. More on the latter shortly. Figure 4.12 shows how you connect an on-premises network to Azure using a local network gateway on the on-premises side and a virtual network gateway attached to your Azure subnet.

**Figure 4.12:** A site-to-site VPN connection using a Virtual Network Gateway.



As you can see from Figure 4.12 the VPN gateway requires its own dedicated subnet. In addition, when you create the gateway, two or more VMs are deployed inside the subnet. You can't directly configure or manage these VMs. They are there to help facilitate the network gateway. Let's create a VPN gateway for our *AlpacaVNet* virtual network. First step is to set up a subnet we'll use for the VPN Gateway.

```
PS C:\> $vnet = Get-AzVirtualNetwork -ResourceGroupName AlpacaRG
PS C:\> $subnet = Add-AzVirtualNetworkSubnetConfig -Name 'Gateway'
PS C:\> Set-AzVirtualNetwork -VirtualNetwork $vnet
```

A VPN gateway must be allocated a public IP address, which is used to connect to from the other networks.

```
PS C:\> $ip= New-AzPublicIpAddress -Name AlpacaIP -ResourceGroupNameN
PS C:\> $ipconfig = New-AzVirtualNetworkGatewayIpConfig -Name ipc
```

And we can now finally create the VPN Gateway itself. There are several SKUs, or models and sizes of Gateways to choose from. They define the number of concurrent connections and throughput.

#### Warning

It can take up to 45min to provision a new VPN Gateway in Azure.

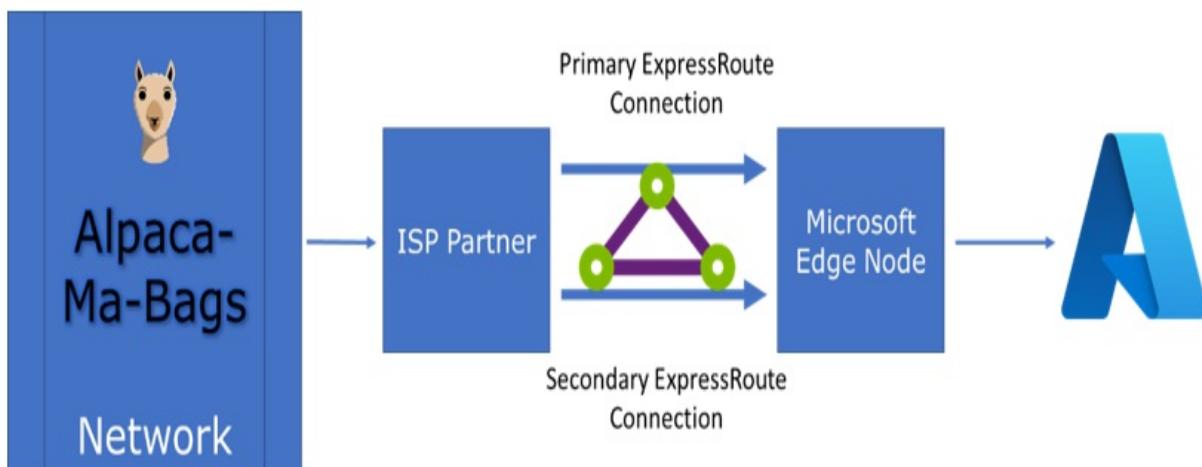
```
PS C:\> New-AzVirtualNetworkGateway -Name AlpacaGateway -Resource
```

You can only have a single VPN gateway per virtual network in Azure. However, you can create multiple connections to that single VPN gateway to allow as many other networks to connect as you need. If you connect the VPN gateway to an on-premises VPN device it is called *site-to-site* networking, and if you create a connection between two Azure VPN gateways it is called *VNet-to-VNet*.

#### 4.4.3 ExpressRoute

The other type of network gateway is an ExpressRoute. Where a VPN Gateway gives you a secure connection between Azure and your on-premises infrastructure, ExpressRoute provides a direct and private connection (Figure 4.13). The traffic going over an ExpressRoute never touches the public Internet.

**Figure 4.13: ExpressRoute network diagram. Traffic never enters the public internet and flows directly into Azure.**



ExpressRoute offers faster speeds, consistent latency, and higher reliability as the connection doesn't use the public Internet. Oh, the downside of using ExpressRoute. It can be seriously expensive depending on your use and requirements, and you are at the mercy of the Internet Service Provider to create the connection on their end too.

#### 4.4.4 Which connection to use when?

We have now been introduced to three different ways of connecting from one network to a virtual network, but when do you use each? If you take costs out of the picture, it boils down to whether you want to use the public internet or not, and if you have to connect to Azure from outside of Azure. Table 4.4 has all the details.

**Table 4.4: Comparing the different network-to-network connections**

Connection Type	Public Internet Use	Connect from non-Azure
Peering	No	No
VPN Gateway	Yes	Yes
ExpressRoute	No	Yes

It might sound odd to tell you to avoid Internet traffic in the cloud, which is, by definition, on the Internet somewhere. However, the Internet is also where the biggest risk lies when it comes to intruders, attackers, and super villains.

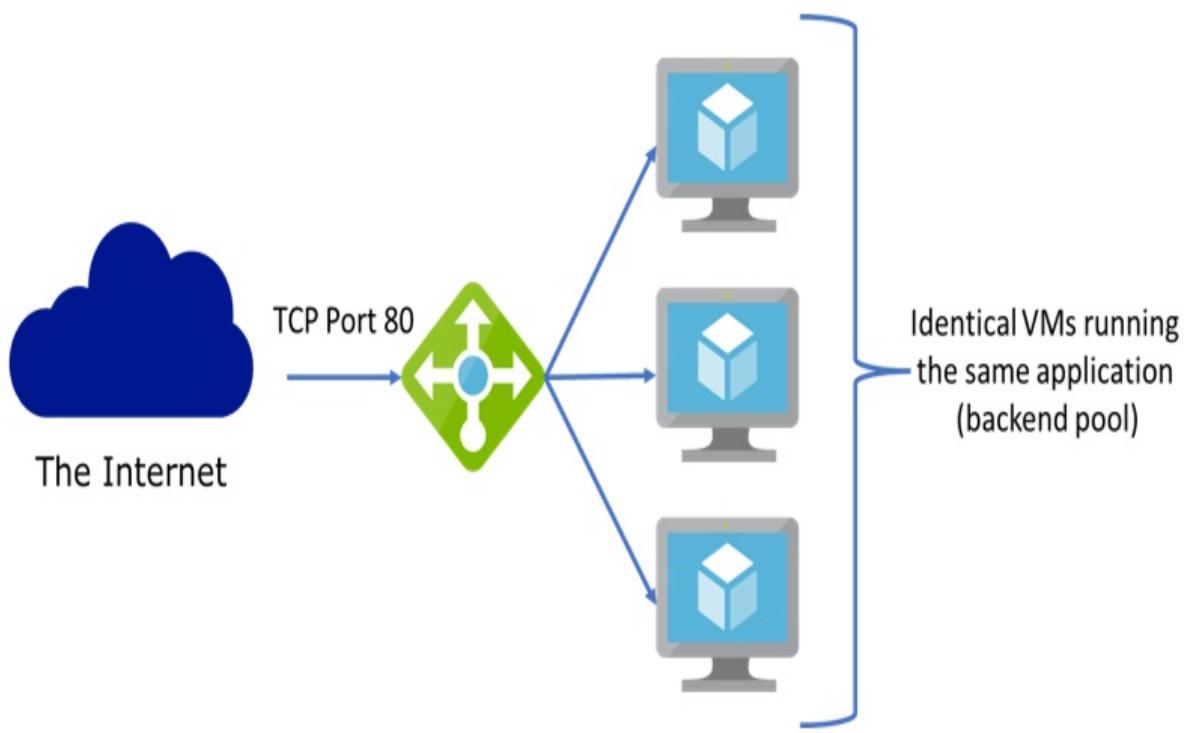
Whenever possible, avoid sending data over the public Internet. You can use ExpressRoute to avoid this by creating a connection that only goes through a dedicated private fiber network. You can also use site-to-site VPN connections to secure the traffic between Azure and your on-premises, and finally use Bastion to only connect to VMs through the Azure Portal.

### 4.5 Distributing network traffic with Load Balancer

When your business for alpaca shearing tools and services really takes off, you will get a lot more traffic coming into your application too. If you are

using VMs to host your application and manage incoming requests, at some point that traffic can become too much for a single VM. You will need to distribute the traffic across multiple VMs to handle the load, and that is what Azure Load Balancer does. The load balancer becomes the sole entry point for all traffic as illustrated in Figure 4.14.

**Figure 4.14: A load balancer distributing traffic from the Internet to three VMs.**



The Internet traffic requests that arrive for your application don't know they are hitting a load balancer instead of the application itself. The load balancer takes that traffic and uses rules to determine which VM in the backend pool that will receive the request. There are a few moving parts that make all this possible in a what seems like a choreographed dance of bits and bytes, so before we jump into creating a load balancer, let's look at the other parts of the service.

## 4.5.1 Public IP address

The entry point for the load balancer is a public IP address, which is the first resource we need to create to eventually provision a load balanced solution.

The IP address is critical for any web service to be identified on the Internet, and the load balancer is no different.

```
PS C:/> $ip= New-AzPublicIpAddress -Name AlpacaLBIP -ResourceGrou
```

The IP address is the public face of your app. It is the destination for all the traffic from the internet. For use with a load balancer in Azure, the IP address needs to be packaged up in a frontend configuration using this cmdlet

```
PS C:/> $frontendconfig = New-AzLoadBalancerFrontendIpConfig -Nam
```

Now the IP is ready for use. More on that in a moment.

## 4.5.2 Backend Pool and VM Scale Sets

To serve the requests to the load balancer, you need a group of VMs that traffic can be directed to. As shown in Figure 4.14 this is called a *backend pool* and is a critical component of a Load Balance setup. The first step is to create the address configuration for the backend pool.

```
PS C:/> $poolconfig = New-AzLoadBalancerBackendAddressPoolConfig
```

The backend pool config is where the VMs are placed and removed from. Often the backend pool is organized in a VM scale set. You use a scale set to allow the number of VM instances within it to automatically decrease or increase as demand dictates according to a schedule. Scale sets provide both high availability and redundancy without you having to manage it.

In a scale set all the VMs are created equal, meaning from the same VM image. All the configuration and applications are the same across all VM instances, which is key to having a consistent experience no matter which VM serves the request from the load balancer.

## 4.5.3 Health probes

If a VM in the backend pool stops responding, times out or fails in some other way, you want the load balancer to know instantly, so traffic isn't being sent to the "sick" VM. This is what the *health probe* does. Let's create one of

those now.

```
PS C:/> $probe = New-AzLoadBalancerProbeConfig -Name "woolProbe"
```

There are four parameters used in that PowerShell cmdlet:

- Protocol: This specifies the protocol to use for the probe and can be tcp, http or https.
- Port: The port to connect to the service on the VM that is being load balanced.
- IntervalInSeconds: The number of seconds between each check the health probe does. Minimum is 5.
- ProbeCount: The number of times in a row the probe has to fail before a VM is considered ‘sick’. Minimum is 2.

The two latter parameters are what you adjust depending on how critical and susceptible to errors the service is. The combined time of the interval between probes times the number of probes that have to fail, provides the time it will take for an endpoint to be determined as ‘sick’.

The health probe informs the load balancer which in turn determines which VM in the backend pool receives any given request. When a health probe fails, the load balancer will stop sending traffic to that sick VM.

#### **4.5.4 Load balancing rules**

Once you have an IP, a backend pool configuration and a health probe, we need to have at least one rule for distributing traffic from the frontend, the IP address, to the backend pool. Let’s create a rule.

```
PS C:/> $rule = New-AzLoadBalancerRuleConfig -Name 'alpacaTraffic'
```

You can have several rules to route your traffic based on protocol, backend pool, port and health probe that should handle the traffic.

#### **4.5.5 Putting it all together to create a Load Balancer**

We are finally at the point where we can combine all the parts to create one

magnificent load balancer. Use this cmdlet to create a new load balancer ready to be put to work.

```
PS C:/> New-AzLoadBalancer -ResourceGroupName 'AlpacaRG' -Name 'a
```

And that is it. By combining an IP address, a backend pool, a health probe, and a rule, you have defined a complete load balance service. This service will now receive the traffic for your site, evaluate which VM is best capable of serving the request, and pass it on, while making sure the VMs are healthy and ready. A load balancer makes both your application perform better, as well as takes some of the angst out of running the application.

## 4.6 Summary

- You create a virtual network (VNet) in Azure to connect other Azure service to, such as a VM. A VNet has a local range of IP addresses that it can assign to device that connect to it.
- Subnets are logical separations of a VNet that get a portion of the available IP addresses in the VNet.
- You use subnets to logically separate resources and apply specific network security rules to them.
- Classless Inter-Domain Routing (CIDR) notation is used to accurately and efficiently describe and IP address range for a network.
- A Network security group (NSG) defines a set of rules that are applied to the resource it is attached to, such as a VM or subnet.
- The rules in a NSG are evaluated according to priority and evaluation stops once a rule is hit. Rules determine if certain types of network traffic is allowed or denied.
- You can connect VNets on Azure with each other using peering, or with on-premises networks using VPN gateways and ExpressRoute.
- When a single VM can't manage the traffic to it, you can use a load balancer to distribute the traffic to more VMs that are identical and live in a backend pool.

[1] This is the number of IP v4 addresses, which is commonly used at the time of writing.

# 5 Storage

## This chapter covers

- Creating a storage account and using its features
- Creating blob, file, queue and table storage, uploading data, and storing it cost-effectively
- Understanding some best practices for using storage on Azure

Data is the life blood of any application, infrastructure or computing ecosystem. Data flows all the time, creating orders, passing on transactions, measuring user input and, oh, so much more. Where does all this data go though? When it finally comes to rest, the data is persisted in storage, which is the third pillar of cloud compute after compute and networking.

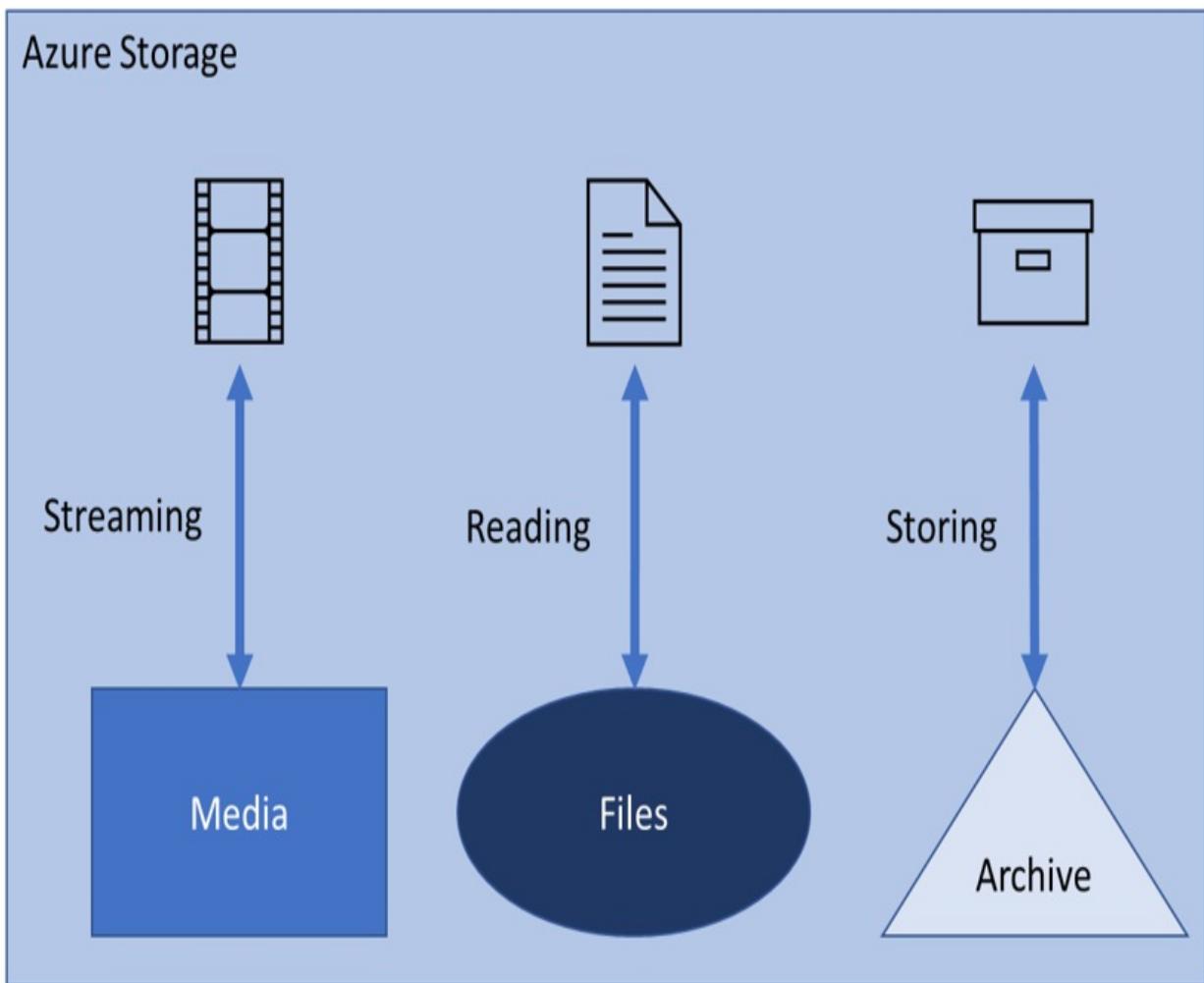
Storage is used in every part of Azure, whether explicitly or implicitly. Create a VM, and you will have a storage account to hold the files for it. Create a website using Azure App Services, and the files are in Azure storage. Write an Azure Function and the code and meta data is in storage. It is like the water of cloud computing. Whether you open the tap deliberately or use the leftover meat loaf from last night, it all has water in it. Yeah, that analogy works. Don't argue.

## 5.1 Storage paradigm

Storage is all over the cloud. You can't avoid it. However, all storage isn't created equal. In this chapter you will learn about the most common ways storage is both used and defined. We store binary objects that only make sense if you have the current program or algorithm to open or use it, as well as common text documents, spreadsheets, and images. We store structured data that is neatly ordered such as user profiles, as well as unstructured data like your emails. All of these various ways of using storage require different solutions, and Azure caters for them all.

While most of us think of storage as somewhere to insert data until we need it, that is only part of the story. Retrieving data from storage is as important as being able to do the storage in the first place. You need to consider the scenarios and circumstances in which you retrieve the data, as that has a direct influence on where you store it. Figure 5.1 shows three very different types of storage: streaming media that requires data with immediate access, files that are read from the storage one at the time, and archiving data that won't be used for months, years or even ever.

**Figure 5.1: Different data purposes requires different data storages.**



This is the key lesson of using storage on Azure: choosing the right service will have an impact on the performance, the cost and ultimately the viability and success of the application using that storage. Yes, it really is that

important. No, it isn't always straight forward to pick the best storage option. More on that conundrum later in this chapter.

Storage accounts are simple to create and can be incredibly cheap. You only pay for what you use and depending on the access frequency and redundancy model of the data, the cost can go way down.

### **5.1.1 Banning Books**

The underground bookstore Banning Books trading in banned technical literature has recently decided to start trading their goods online. As much as they love the printed page, the world is demanding digital copies of their forbidden wares. They are new to cloud computing, but they plan on progressively moving their administration, inventory, and several business processes online. Banning Books want to achieve the following for their business:

- Ensure administrative files and documents are safe from hardware failure and local disruptions.
- Store all books in a digital format in a cost-efficient manner.
- Provide a messaging system between different systems and applications that ensures loose coupling, meaning less critical dependencies on other systems.
- Create a list of all technical books for fast retrieval for websites and internal processes.

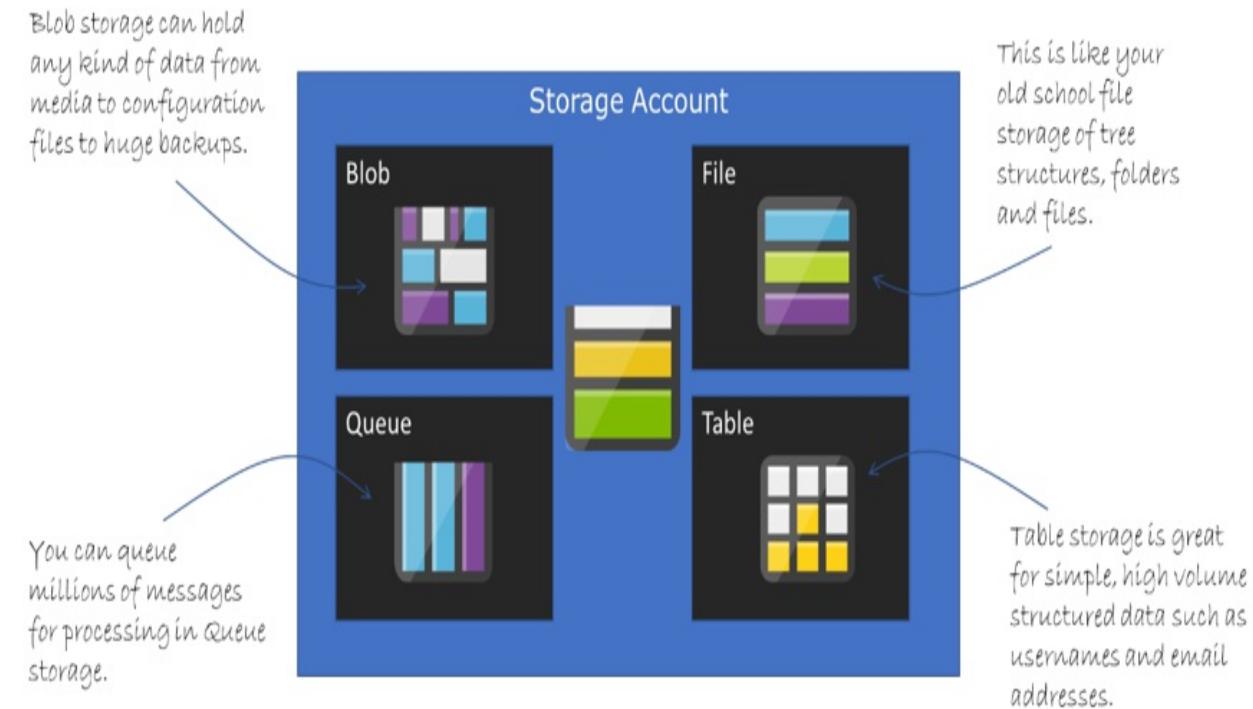
Banning Books want to achieve all these objectives using Azure storage, as they want to reap the benefit of this cheap, accessible, efficient and highly scalable service. Read on to see how Azure can help them out.

## **5.2 Storage Accounts**

We'll start with a storage account in Azure. Just like a VM and a VNet you learnt about in the previous chapters, a storage account is a fundamental part of using Azure. Almost any project involving Azure will have one, or often several, storage accounts that holds all kinds of data. It can seem strange at first, but a storage account isn't where you store the data. It is a logical

container for the four types of data you can store there, blob, file, queue and table, as shown in Figure 5.2. As you can see, each type of storage is a separate and distinct type of storage within the storage account.

**Figure 5.2: The four types of storage that can live inside a storage account.**



That means a single storage account can hold all four types of data storage at the same time, which makes it simpler to logically separate your data storages by project, client or department. Banning Books need a new storage account to get started moving all their assets to Azure. Let's start by creating a storage account for them using PowerShell. As you have learnt in the previous chapters, the first step is, as for any resource on Azure, to create a resource group. The PowerShell cmdlet *New-AzResourceGroup* is how you create a new resource group.

```
PS C:\> New-AzResourceGroup -Name 'BanningRG' -Location westus
```

That should be a command you recognize from the previous chapter. Next, we can then create the actual storage account in the resource group with a new cmdlet *New-AzStorageAccount*, which defines the properties for a new storage account.

```
PS C:\> New-AzStorageAccount -ResourceGroupName 'BanningRG' -Name
```

Once you have run the cmdlet in PowerShell, you should see an output like Figure 5.3, which shows the most important properties of the *banningstorage* storage account, such as storage account name, location, SKU Name and access tier.

**Figure 5.3: Output in the PowerShell command prompt when creating a storage account (edited for clarity).**

StorageAccountName	ResourceGroupName	PrimaryLocation	SkuName	Kind	AccessTier	EnableHttpsTrafficOnly
banningstorage	BanningRG	westus	Standard_RAGRS	StorageV2	Hot	True

In the PowerShell code snippet above there are a few parameters we need to be familiar with for the cmdlet *New-AzStorageAccount* we are using. Resource group should be familiar by now, and the location is where you want to place the storage account physically.

While the *Name* parameter seem straight forward, there are a couple of considerations. The name must be all lowercase. The name is used as part of the public domain name for the storage account, and therefore must be unique across all of Azure. While domain names aren't case sensitive, this is a decision Azure has made to ensure all storage accounts comprise valid domain names. In the case of the Banning books storage account above, the public URL would be

<https://banningstorage.<storage type>.core.windows.net>

This also means that the storage account name is tied to all elements in it, such as blobs, files, queues and tables. More on that later in this chapter.

### 5.2.1 Replicating your storage

The last two parameters are new though and specific to storage accounts. *SkuName* is short for Stock Keeping Unit Name, which is a throwback to

when you had wares in a warehouse, and they were identified by an SKU. For Azure storage accounts SKU refers to the type of storage account and redundancy. In the case above of *Standard\_RAGRS*, it refers to a storage account of type *Standard* (*Premium* is also available) and redundancy *Read Access – Geo Redundant Storage*, or RAGRS. The available SKUs for *Standard* and *Premium* storage accounts are

- Standard\_LRS: Locally redundant storage, which means three copies of your data is stored within the same data center. Adequate for most storage scenarios.
- Standard\_GRS: Geo redundant storage. Not only do you get the three local copies, but Azure will also place *another* three copies in a paired region, such as “East US” for “West US” as we have chosen. If the West US region we chose goes down, we are ensured an exact copy of the storage account lives in East US.
- Standard\_RAGRS: Read access geo redundant storage is the same as geo redundant storage, but you can read the data from both regions, which adds reliability in case the primary region can’t be read.
- Standard\_ZRS: Zone redundant storage utilizes the availability zones you learnt about in chapter 3. Data is replicated among three availability zones in the primary region, which adds another layer of reliability.
- Standard\_GZRS: Geo-zone-redundant storage combines geo and zone types of redundancy. It writes three copies of your data synchronously across multiple Azure Availability zones, like ZRS, as well as replicates your data to the secondary geo-pair region.
- Standard\_RAGZRS: Just like GZRS, but you also get read access to the paired regions data.
- Premium\_LRS: Locally redundant storage, but for a premium storage account.
- Premium\_ZRS: Zone redundant storage, but for a premium storage account.

You have surely noticed by now, and yes, I know you want to know what this *Standard/Premium* malarkey is all about. And for that we need Table 5.1 to outline some of the main differences.

**Table 5.1: Comparison of Standard vs. Premium storage account type.**

	Standard	Premium
Storage Type	Blob, file, queue, table	Blob and file only
Pricing	Charged by usage, such as 100GB used from 1TB reserved	Charged by reserve. You pay for the full 1TB disk, if you only use 100GB.
SLA	Best effort	Performance SLA
Sharing	You share the physical storage with other users	You get a physical SSD that isn't shared
Performance	Not guaranteed	Guaranteed by SLA targets
Redundancy	Geo redundant options available	Only locally redundant
Cost	Very cost efficient depending on tier	Higher cost

As you might have surmised, Premium gets you higher performance, a service level agreement and exclusive use of the hardware at a higher price. On the downside you have fewer redundancy options, and you can only use it with blob and file types of storage.

## 5.2.2 Storage type = Kind + SKUName

We aren't done with the PowerShell command from before though. There is still the *kind* parameter. The possible values are *StorageV2*, *BlockBlobStorage* and *FileStorage*. It isn't quite as simple as just choosing one, as they depend on your choice of *SKUName* and which type of storage you are after. And yes, you guessed it. We need another table, like Figure 5.2, to show what the valid combinations of storage type and kind are.

**Table 5.2: Valid combinations of Kind and SKUName parameters for a specific storage type.**

Storage Type	Kind	SKUName
Standard V2	StorageV2	Standard_LRS / Standard_GRS / Standard_RAGRS/ Standard_ZRS / Standard_GZRS / Standard_RAGZRS
Premium Block blobs	BlockBlobStorage	Premium_LRS / Premium_ZRS
Premium File Shares	FileStorage	Premium_LRS / Premium_ZRS
Premium Page blobs	StorageV2	Premium_LRS

The *storage type* is the more specific type of storage you want to create. The

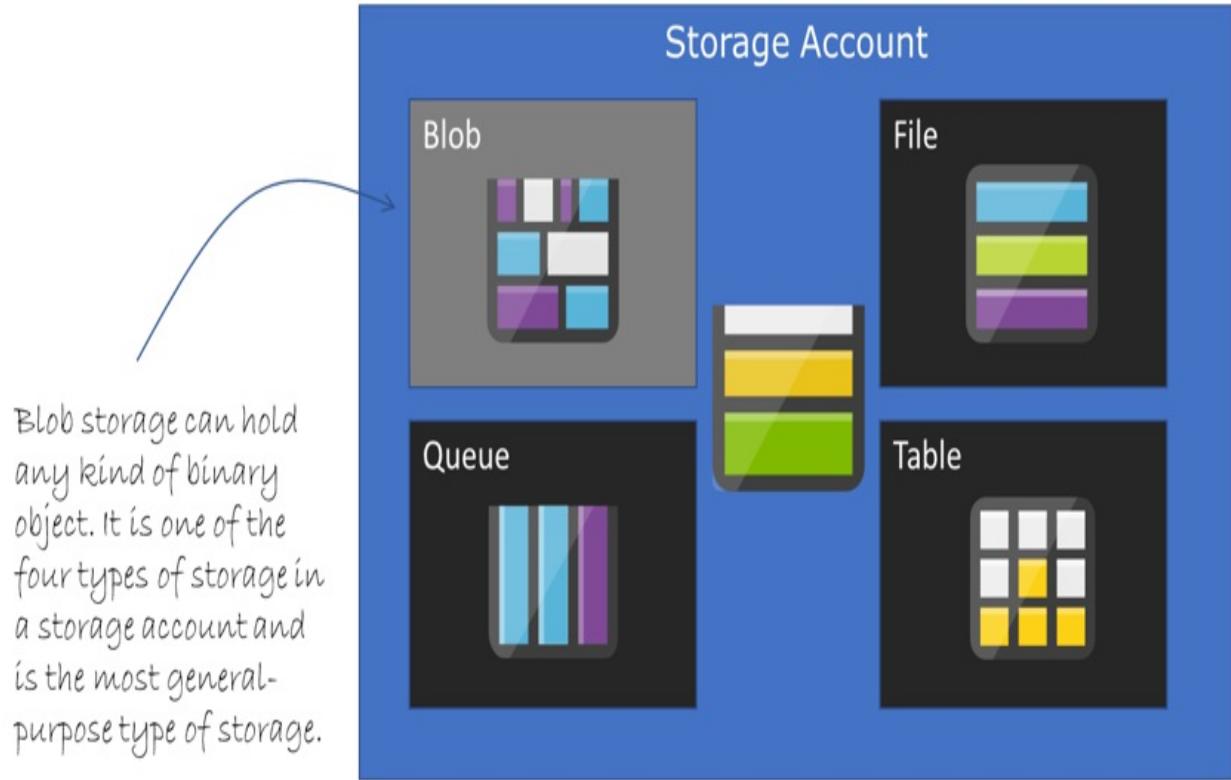
combination of *Kind* and *SKUName* defines which type storage you create. For example, if we had chosen *FileStorage* for *Kind* and *Premium\_LRS* as *SKUName*, then the storage account would be of type *Premium File Share*. Wow, that was a fair bit wrapped up in that one bit of PowerShell code, right? As I mentioned earlier in the book, it is often simple to create resources on Azure, but understanding when to use a feature, or how a parameter can influence a service is both harder and much more valuable. To create an efficient and it is critical to get these selections right. To ensure the right storage quality for your customers, you must consider appropriate redundancy and type. For example, if you have a simple low usage solution for a few customers, and it isn't critical, then there is no need to pay for an expensive multi-region redundant solution. You might want to choose premium storage though to ensure immediate access to the data using a solid state drive.

Now it is time to move on and learn about what lives inside storage account. Repeat after me: *blooooooob*.

## 5.3 Blob

While *blob* undeniably is a fun word to say a lot, it does have an actual meaning too. It is short for *binary large object*, which is another way of saying “store pretty much anything”. It was one of the first types of cloud storage on Azure and remains one of the most popular today. Blob storage is one of the four types of data you can have in a storage account, elegantly depicted in Figure 5.4 with excellent iconography. I hope you like it.

**Figure 5.4: The blob section of a storage account, which is the most versatile.**



### 5.3.1 Access Tiers

Before we get into the ins and outs of blob storage, there is one more thing you need to know about. If we revisit the PowerShell output from before (Figure 5.5), there is a column name *AccessTier*.

**Figure 5.5: The access tier is set by default to “hot” when creating a storage account.**

Name	-Name	AccessTier
SkuName	Kind	
-	-	-
Standard_RAGRS	StorageV2	Hot

The access tier is per file in storage and can be cold, hot or archive.



The access tier can take one of three values depending on how often you need the data: cold, hot or archive. The less urgent a piece of data is, the less it can cost to store on Azure, if you use the correct access tier.

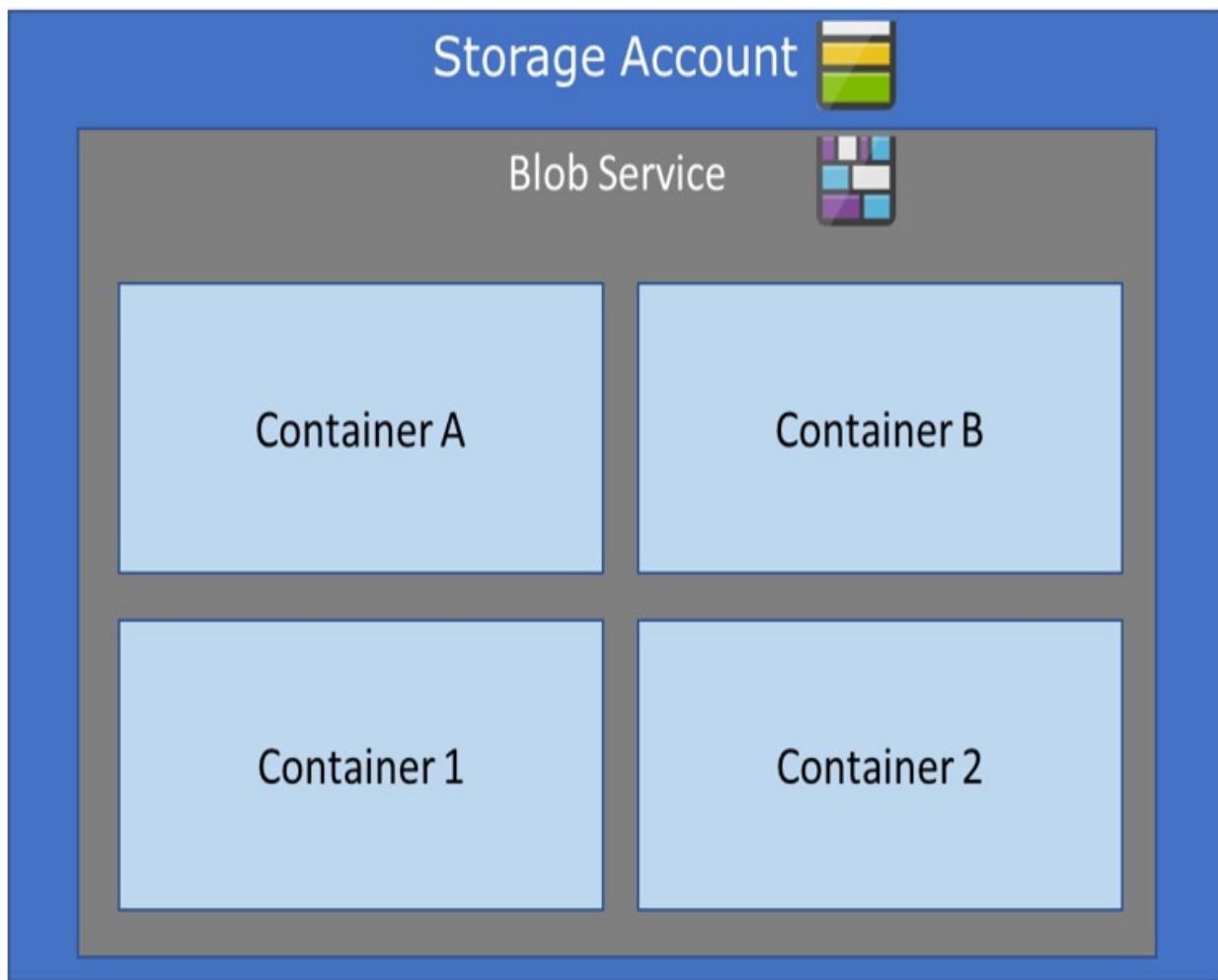
- Hot is for files that need to be readily available. This could be files that are either used frequently or needed straight away when requested. For example, this could be customer order files for Banning books. When they are requested, they need to show up straight away.
- Cold is used when files are accessed infrequently and aren't urgent when they are. It is also used for larger files that needs to be stored more cost effectively. For Banning Books this might be staff images used for the monthly newsletter. They aren't urgent and are rarely used at all.
- Finally, the Archive tier is an offline tier for rarely accessed files. It is dirt cheap. Really cheap. It is designed for large backups, original raw data, data to comply with legislation and other datasets that are unlikely to ever be used again, but are required to be kept. It can often take hours to get the files out of archive, and you can't do it too often.

The three tiers are important for both price and performance. Getting the combination wrong and you will either be paying way too much for storage, or your application depending on the blob storage items will not perform optimally. Or probably both. However, there is more to blob storage than just performance tiers.

### 5.3.2 Containers

All blobs are stored within a parent container within your storage account (Figure 5.6). Think of the container as a folder in a traditional file system. It organizes the blobs logically, and you can have an unlimited number of containers in your storage account.

**Figure 5.6: A storage account can have an unlimited number of blob service containers.**



Creating a container within the Banning storage account can be done with the following PowerShell. First, we need to get the context of the storage account, so we can add the container inside of it. Remember this bit, as we will continue to use the storage account context throughout this chapter.

```
PS C:\> $storageContext = Get-AzStorageAccount -ResourceGroupName
```

Then simply add the storage container inside that context using the *New-AzStorageContainer* cmdlet.

```
PS C:\> New-AzStorageContainer -Name bannedbooks -Context $storag
```

The *Get-AzStorageAccount* cmdlet gets a specified storage account, from which we can then use the context to create a container using the cmdlet *New-AzStorageContainer*. There is one parameter, which we haven't seen before and which is critical: *Permission*. This specifies the public access level to the container and can take one of three values:

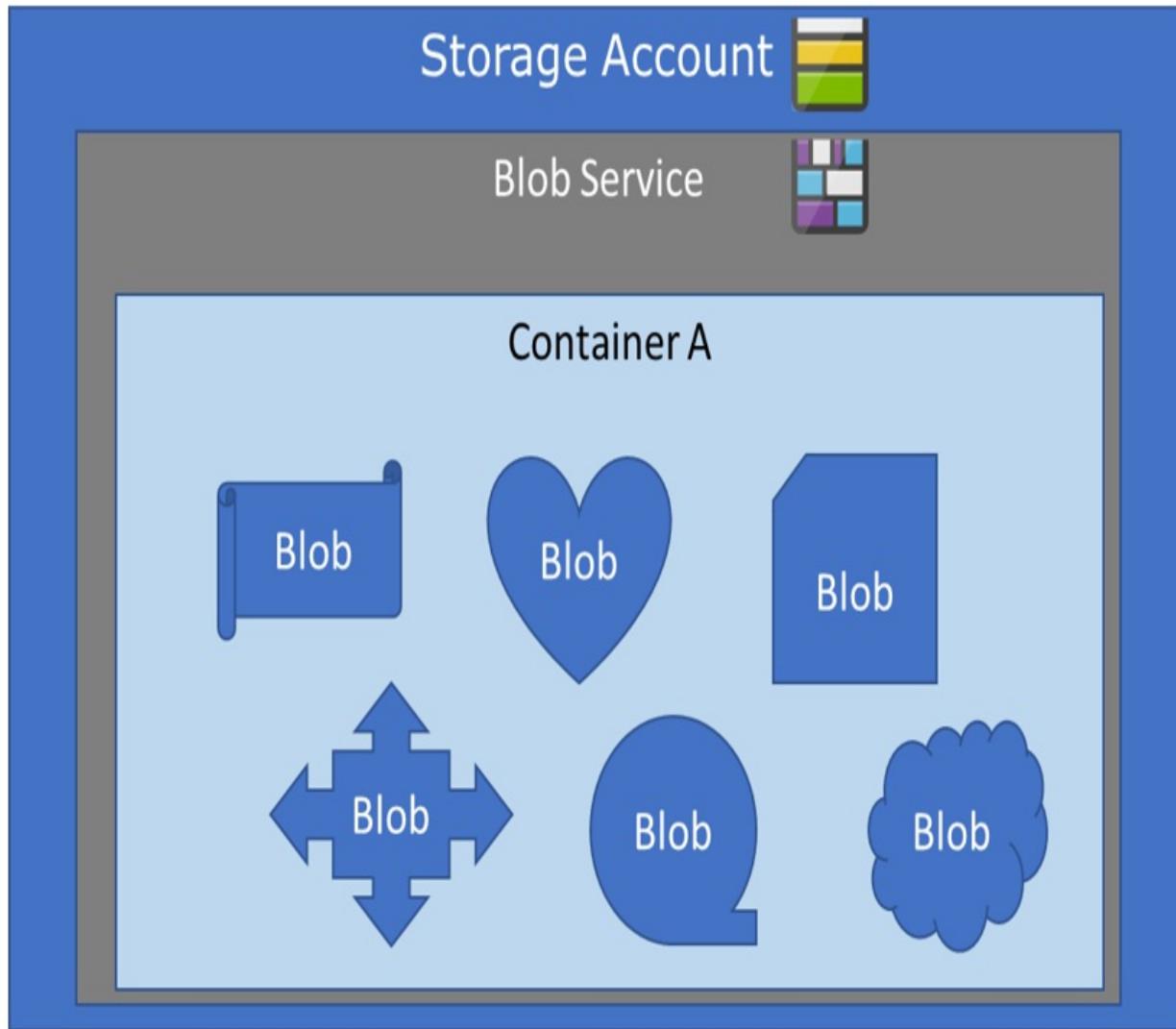
- Off: Only the owner of the storage account can access the container.
- Blob: Allows anonymous requests to access any blob within the container, but only if you have the direct address. We'll get to how those addresses work shortly.
- Container: Provides full read access to the container and all the blobs inside it.

In this case we choose Off as the books being stored here should not be accessible publicly. This is critical to get right, as choosing the wrong *Permission* parameter value could expose all the blobs to the public internet. This is also why *Permission* is set to Off by default. Let's get some blob action happening.

### 5.3.3 Blob Types

Inside a container is where all the blobs live. You can have an unlimited number of blobs in a container, but in a container they must be. As Figure 5.7 shows, blobs can be anything in a binary form, which makes this an extremely versatile Azure service. You can store video files, text files, images, whole executable programs, proprietary formats, well, anything.

**Figure 5.7: Blobs are in a container, which are in a storage account. Blobs can be anything in a binary format.**



There is a bit more complexity to storing blobs though, as there are three different types of blobs you can insert into a container.

- **Block:** The most common type of blob is the block blob. They are both text and binary data, and can hold up to 190.7 TiB<sup>[1]</sup>, which is a lot. Uploading a single file to a container will create a block blob.
- **Append:** Very similar to block blobs, but optimized for append operations, such as a continuous log. As the name indicates, when you modify an append blob, blocks are added to the end of the blob only. Each block can be 4 MiB<sup>[2]</sup> in size and you do a maximum of 50,000 append operations on a blob.
- **Page:** These are your random-access files, and are split up into pages of

512 bytes, hence the name. Most commonly page blobs store disks for VMs, thus they are optimized for frequent read/write operations. For example, Azure SQL DB, which is covered later in this chapter, uses page blobs as the underlying persistent storage for its databases.

Finally, Banning Books want to upload their books in PDF format to blob storage for use later. They use the below PowerShell command for each PDF, where the PDF filename *bannedbook.pdf* is the file being uploaded. We can use the same context *\$storageContext* that we used before to create the container, to designate where to upload the blob.

```
PS C:\> Set-AzStorageBlobContent -File "C:\books\bannedbook.pdf"
```

Adjust the file path to a valid one in your environment, and ideally make it a variable such as *\$blobfilePath* to enable automation of the upload of many files. The *-Blob* parameter defines the blob's name in the storage container, equivalent to the destination file name. This is the name you'll use to reference the blob in the container. The *StandardBlobTier* is the hot, cold or archive tier. If we inspect the container in the Azure Portal, you can see the blob inside it now (Figure 5.8).

**Figure 5.8: A blob in the blob container on the storage account.**

The pdf file we uploaded via PowerShell

Property	Value
URL	<a href="https://banningstorage.blob.core.windows.net/bannedbooks/bannedBook.pdf">https://banningstorage.blob.core.windows.net/bannedbooks/bannedBook.pdf</a>
LAST MODIFIED	10/25/2021, 9:13:08 PM
CREATION TIME	10/25/2021, 9:13:08 PM
VERSION ID	-
TYPE	Block blob
SIZE	39.28 KB
ACCESS TIER	Hot
ACCESS TIER LAST MODIFIED	10/25/2021, 9:13:08 PM
SERVER ENCRYPTED	true
ETAG	0x8D997A00AD24350
VERSION-LEVEL IMMUTABILITY POLICY	Disabled
CONTENT-TYPE	application/octet-stream
CONTENT-MD5	1e2d51lo38OnQwVj/k0qVA==
LEASE STATUS	Unlocked
LEASE STATE	Available
LEASE DURATION	-
COPY STATUS	-
COPY COMPLETION TIME	-

Undelete

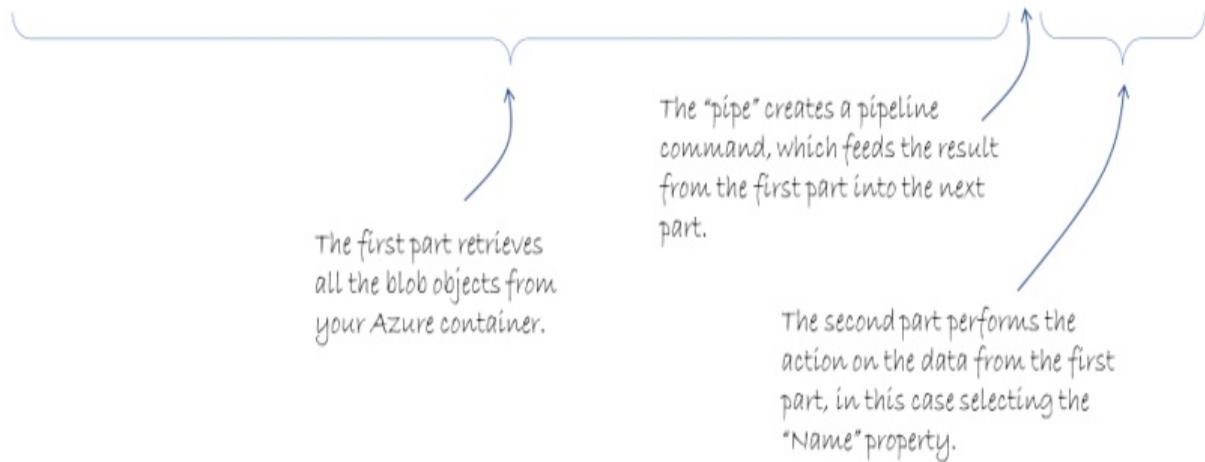
You can inspect all the files in the container through the Azure portal, such as the direct URL for accessing the blob, size, type, and access tier. You can of course also use PowerShell.

```
PS C:\> Get-AzStorageBlob -Container bannedbooks -Context $storag
```

The interesting part in the command above is not the cmdlet *Get-AzStorageBlob* but the “pipe” command with the *select Name* after it is a way to chain multiple commands together. In this case the cmdlet, container and context produce a result containing all the blobs in the container. That result is then fed into the next part of the pipeline, defined by the “pipe” character as shown in Figure 5.9.

**Figure 5.9: The anatomy of a PowerShell pipeline command.**

```
Get-AzStorageBlob -Container bannedbooks -Context $storageContext.Context | select Name
```



The pipe command is a very powerful way to manipulate the data to create a custom dataset for either output or further processing.

For a moment, let's go back in time, or at least a couple of paragraphs. In Figure 5.8 there was a textbox for the URL of the blob. The URL follows a very specific format, which you need to understand to use it effectively. As mentioned at the top of the chapter, all storage accounts have a public universal resource indicator, or URI, structured as following:

`https://<storage account name>.<storage type>.core.windows.net`

If we then put that into context of the blob container *bannedbooks*, you get the following URL:

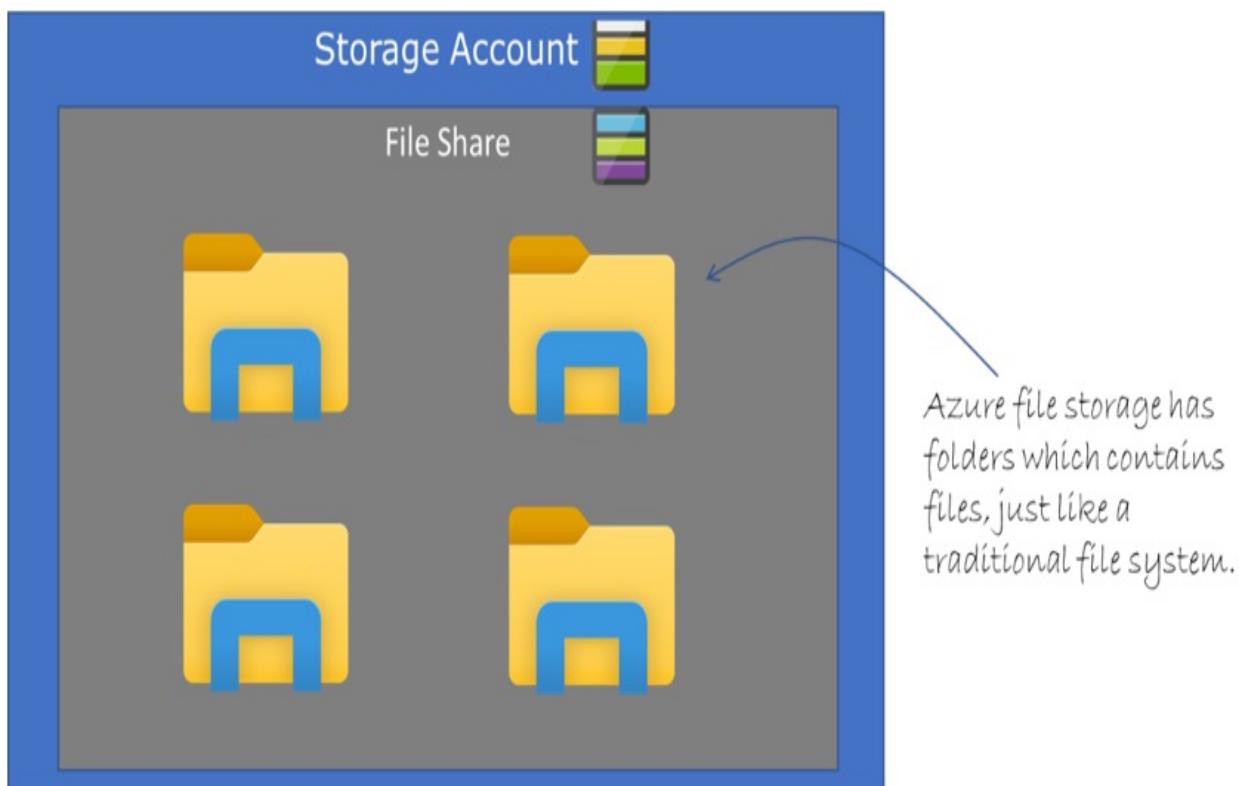
`https://banningstorage.blob.core.windows.net/bannedbooks/<blob na`

The container *bannedbooks* is added to the URI similarly to how a folder path is structured in a file system. Of course, access is managed through the permissions on the storage account or container. There are other ways to access blobs in Azure too, which we will cover in later chapters of the book as you build up your Azure knowledge. For now, it is time for another type of storage that might be more familiar.

## 5.4 File

File storage on Azure is a managed service, meaning it works like, well, a file system. If you have used Windows Explorer on almost any version of Windows, you will be familiar with the system of folders and files it presents. Figure 5.10 illustrates how Azure File Storage resembles the traditional file system.

**Figure 5.10: File share in a storage account with folders.**



One of the common use cases for file shares on Azure is to attach it to a computer, such as a VM, so the files are accessible directly from within the host's file system. This is not to run the main operating system disk, but rather the files that are stored on the file share. Just like plugging in that external hard drive with all your movies on it at your friend's house, you can attach a file share to a VM.

To get a file storage service to use, let's start by creating a File storage

section in our storage account for Banning Books and then discover some of the details. Banning wants to store their administration files, such as time sheets, lists of banned customers, and lists of top banned books on Azure too. A file share in an Azure storage account is perfect for this. Using the following PowerShell command we create a file share in the existing *banningstorage* storage account.

```
PS C:\> New-AzRmStorageShare -ResourceGroupName BanningRG -Storage
```

The cmdlet used is *New-AzRmStorageShare*, which creates a storage file share. The first three parameters *ResourceGroupName*, *StorageAccountName* and *Name* should all seem familiar. The *AccessTier* is very similar to the tiering for blob storage. You have 4 different types of access to choose from:

- **TransactionOptimized:** This is the default tier and is stored on standard hardware, meaning HDD devices. *TransactionOptimized* (yes, that is really the spelling for the parameter) is suitable for workloads with a high number of transactions, but without a need for low latency or high bandwidth like the Premium tier. You pay less for having files accessed a lot, but more for the amount of storage you use.
- **Hot:** This is also storage on an HDD, but the transaction costs are slightly higher, and the storage costs slightly lower compared to the transaction optimized tier. The hot tier is right in between transaction optimized and cold.
- **Cold:** I kind of gave the prize away already, but the cold file storage tier has the cheapest storage costs, but highest transaction costs. It is also using HDD hardware for all files.
- **Premium:** This is the best performing tier and is hosted on a solid state drive (SSD). It has the lowest latency for file access, the fastest read and write to storage, and the absolute high score in Donkey Kong. Okay, I made that last point up. Much more about premium storage in just a sec.

While you get an HDD backed service when you create a cold, hot or transaction optimized storage, you only pay for the storage space you *actually use*. And that brings me to the last parameter in the PowerShell command.

*QuotaGiB* is the provisioned size of the file share in Gibibyte, which in this

case is 1024. You need to define the size of the file share you want, so Azure can make sure it is there for you when you need it.

If you store infrequently accessed files in the transaction optimized tier, you will pay almost nothing for the few times in a month that you request those files, but you will pay a high amount for the data storage costs. If you were to move these same files to the cool tier, you would still pay almost nothing for the transaction costs, because you hardly use them, but the cool tier has a much cheaper data storage price. Selecting the right tier for your scenario allows you to considerably reduce your costs.

As with blob storage, you can also access the file share through a specific end point URI, which for file shares is structured like this:

`https://banningstorage.file.core.windows.net/bannedfiles/`

The key here being the *file* part of the URI, which identifies the file share within the *banningstorage* account, and the *bannedfiles* path that identifies the specific file share.

### 5.4.1 Premium Files

The Premium tier for file shares is, like premium tier for blobs, a bit different than the other tiers. Two main differences are key to understanding how, and when, to use the premium tier. First, the hardware backing the tier is all solid-state drives (SSD), so performance is much higher. You get input and output (IO) operations done in less than 10 milliseconds, and the time to retrieve files, called latency, is very low. This translates to very fast storage that Microsoft recommend you use for anything critical to your business. Use Figure 5.11 for a quick summary of using a HDD vs a SSD.

**Figure 5.11: HDD vs. SSD storage properties.**

 HDD	 SSD
Cheaper	Not so cheap
Less performance	More performance
High latency	Low latency
For non-critical stuff	For all stuffs

That figure also brings me to the second difference: pricing. Where the other three tiers are billed for the storage you use, premium is billed for the storage you provision. Banning Books creates a storage account with 100GiB file share and fills it with 50GiB of data. If the file share is transaction optimized, hot or cold, Banning pays for 50GiB of used storage. If the file share is on the premium tier, they pay for 100GiB regardless of how many GiB are used. And if you have created a premium tier file share you can't change it back to non-premium and vice versa. Choose wisely.

### 5.4.2 Large files

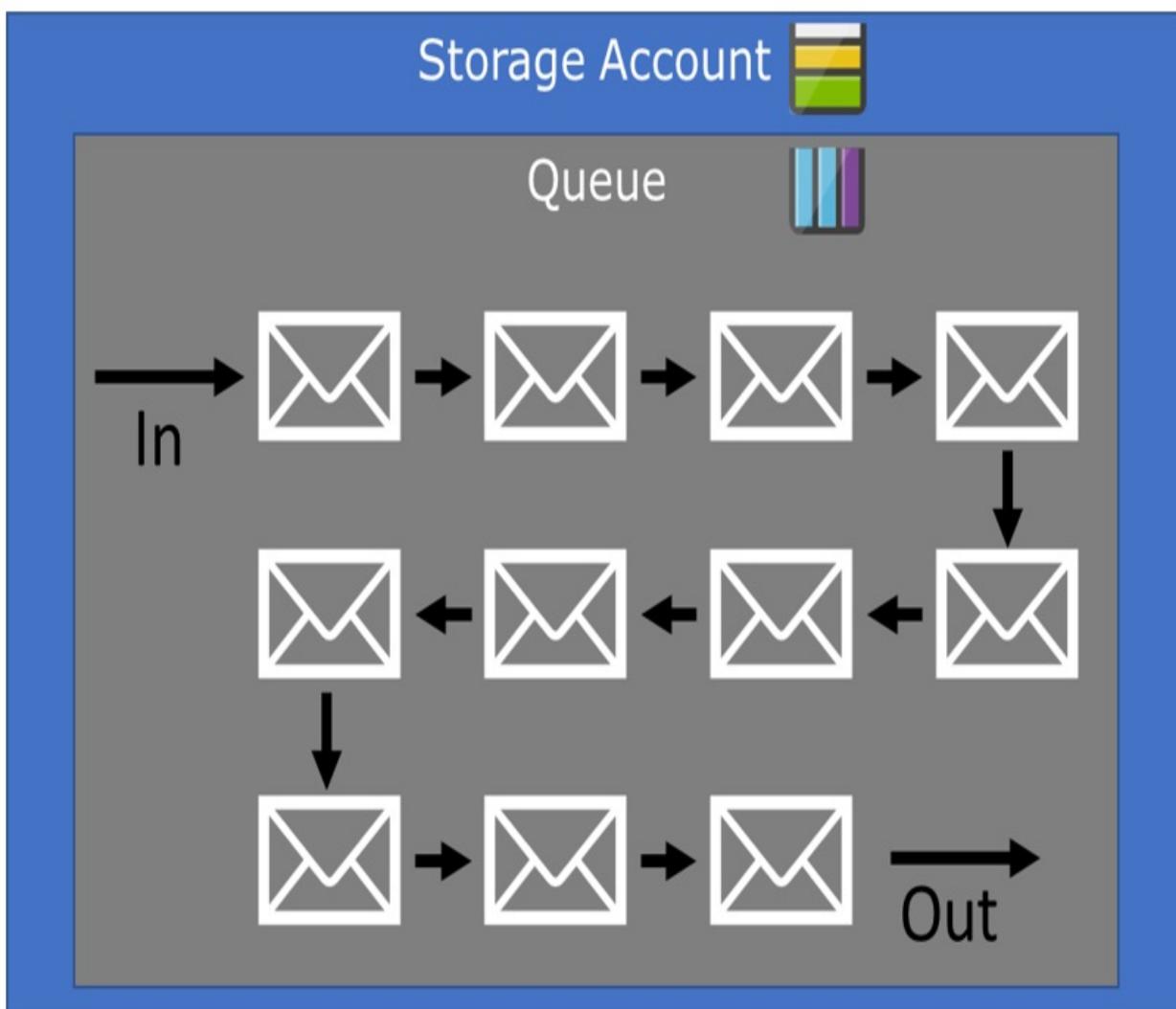
If you are after a non-premium file share that is from 5 TiB to 100 TiB in size, you need to enable the *large file share* option. While it creates a very large file share, it limits your redundancy option to only locally redundant and zone redundant. Presumably this is to limit vast amounts of data going over larger distances, such as for geo-redundancy. Once you have converted to a large file share, you can't go back. That file share is a big one forever, however you can enable it on existing accounts too. As long as they are set up as locally or zone redundant of course.

Premium file shares support 100 TiB from inception, so this option isn't relevant for the premium tier.

## 5.5 Queue

At first glance, a queue doesn't seem to be related to storage. A queue is a line of items or objects that progress from the back to the front of the queue, then disappear as shown in Figure 5.12.

Figure 5.12: Queue storage with messages coming in one end and out the other end, in order.



Storage on the other hand is a persistent service that keeps hold of your data for as long as you need it. Queues and storage thus seem to be somewhat opposites, but Azure queue storage serves several important purposes. Before we get to those, let's understand what a queue is in Azure computing terms. A queue holds messages, and those messages are just data objects up to 64Kb

in size. The unique property of a queue is that you push new messages into the end of the queue and retrieve messages already queued, from the front, just like you saw before in Figure 5.12.

The first of those important purposes I mentioned before is volume. You can store millions of messages in a queue storage up to the capacity limit of the storage account itself. And it is very cost effective as well, especially if you already pay for a storage account. Now to that pressing question I can sense you have: How does queue storage work and how do I create one? Okay, that was two questions, but let's answer those. First, we will create a queue storage inside our existing storage account. Banning Books will use the queue to store incoming book orders before they are processed. Use the PowerShell cmdlet *New-AzStorageQueue* with the storage account context we used to create the blob storage.

```
PS C:\>$queue = New-AzStorageQueue -Name bannedorderqueue -Contex
```

That is pretty simple, and there isn't much to explain, other than we now have a queue storage. Yay! As with blob and file storage, you also get a URI for direct HTTP/HTTPS access over the public Internet. It looks like this:

<https://banningstorage.queue.core.windows.net>

Remember, the storage account can only be accessed if you set the permissions to allow it, as we did at the start of the chapter. Time to insert some messages into the queue, which requires a new bit of PowerShell that defines the message to insert, then inserts it.

```
# Create a new message using a constructor of the CloudQueueMessage
PS C:\>$message = [Microsoft.Azure.Storage.Queue.CloudQueueMessage]::new("A new book has arrived!")

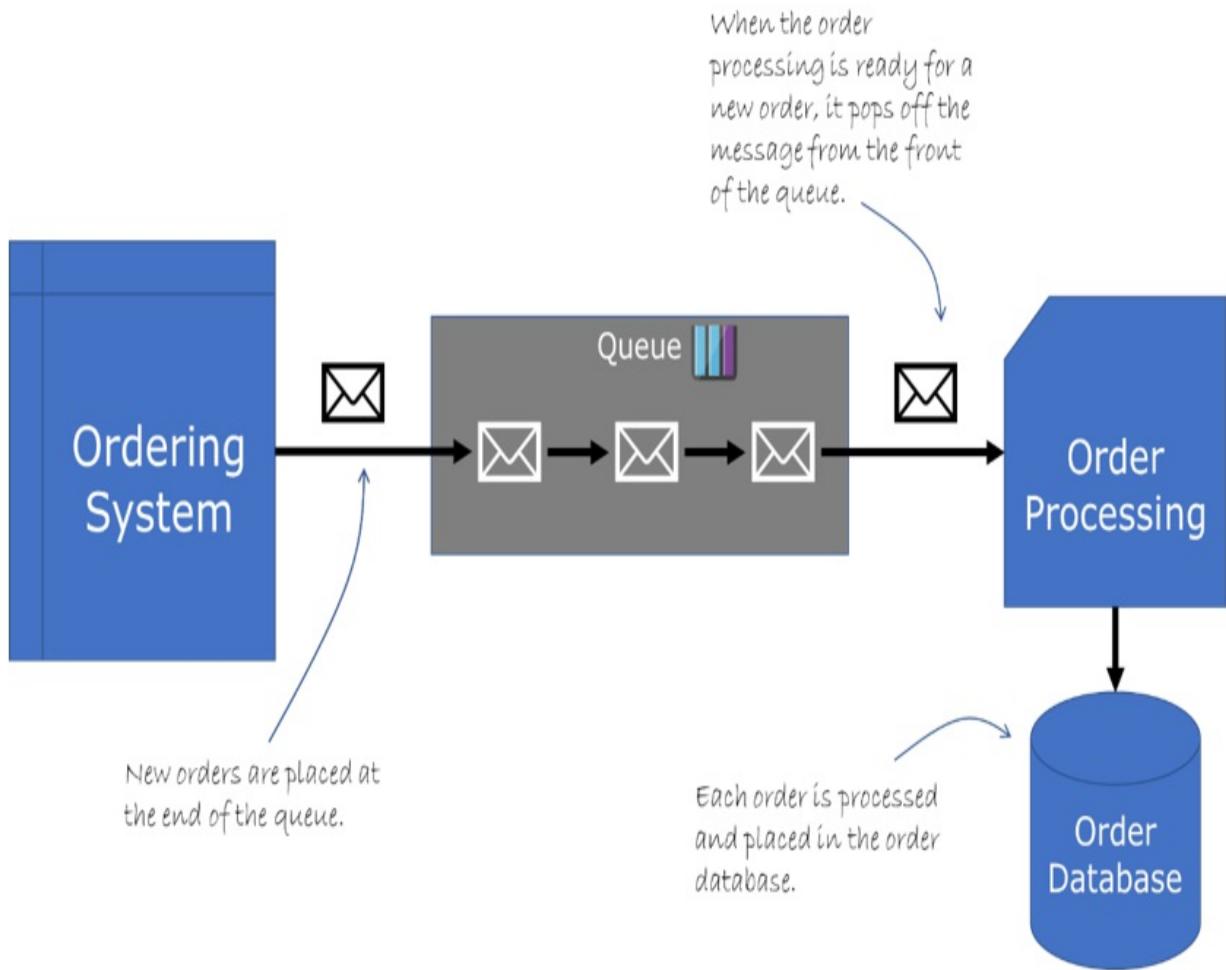
# Then add a new message to the queue we created
PS C:\>$queue.CloudQueue.AddMessageAsync($message)
```

The most obvious new part is the message itself, which has the syntax for creating an object of type *CloudQueueMessage* in the namespace of *Microsoft.Azure.Storage.Queue*, where the `::new` part tells PowerShell that you want to create a new object. We then add the message as the parameter for that object constructor. Finally, we queue the new message on the storage queue using the *AddMessageAsync* function. The keyword here is *Async*,

which is short for asynchronous and ensures PowerShell won't wait for the PowerShell call to complete but can finish or continue without "hanging". This would be critical if you were to upload millions of messages, for example.

At this point you might be thinking that you wouldn't use PowerShell to upload a single message like this, and you'd be right. We got to start somewhere though, and now that you understand what queue storage is and how a message is stored, we can move a bit further up the food chain. Banning Books has a website that generates orders, which are eventually stored in a database. Banning has concerns about the processing capacity when there are sales on and during other high activity events. For each order a message will be added to the storage queue for asynchronous processing by a separate process that can continue to work at a constant pace regardless of the input rate to the queue as shown in Figure 5.13.

**Figure 5.13: using an Azure storage queue for holding orders without overwhelming the orderprocessing.**



And that brings us to another important purpose for queue storage: decoupling. If services and applications that depend on each other are too tightly coupled, it makes scalability difficult. Instead, by using queue storage you can decouple services that add messages to the queue and services that consume the messages, and the services don't even have to know about each other.

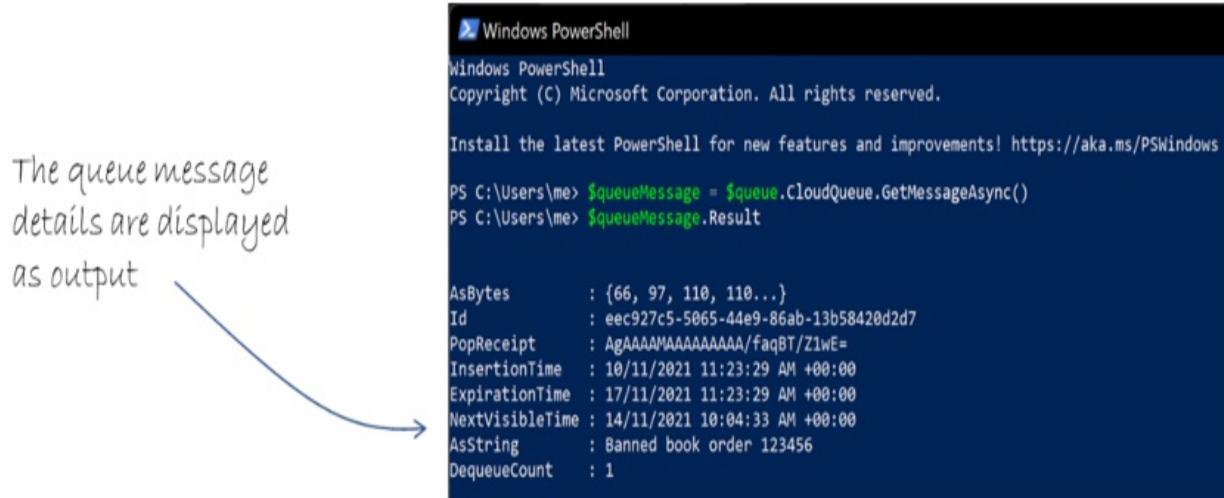
And speaking of consuming, let's end this section with reading the message from the queue that we just added to it.

```
PS C:\> $queueMessage = $queue.CloudQueue.GetMessageAsync()
PS C:\> $queueMessage.Result
```

If we inspect the PowerShell code, we can see that you don't define which message you want. *GetMessageAsync()* gets the next message in the queue, whatever that message might be. That is the whole point of a queue though.

Messages go in one end, and you process them in order from the other end. In Figure 5.14 the message details are displayed when the message is successfully retrieved from the queue.

**Figure 5.14: The queue details are displayed in PowerShell.**



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

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\me> $queueMessage = $queue.CloudQueue.GetMessageAsync()
PS C:\Users\me> $queueMessage.Result

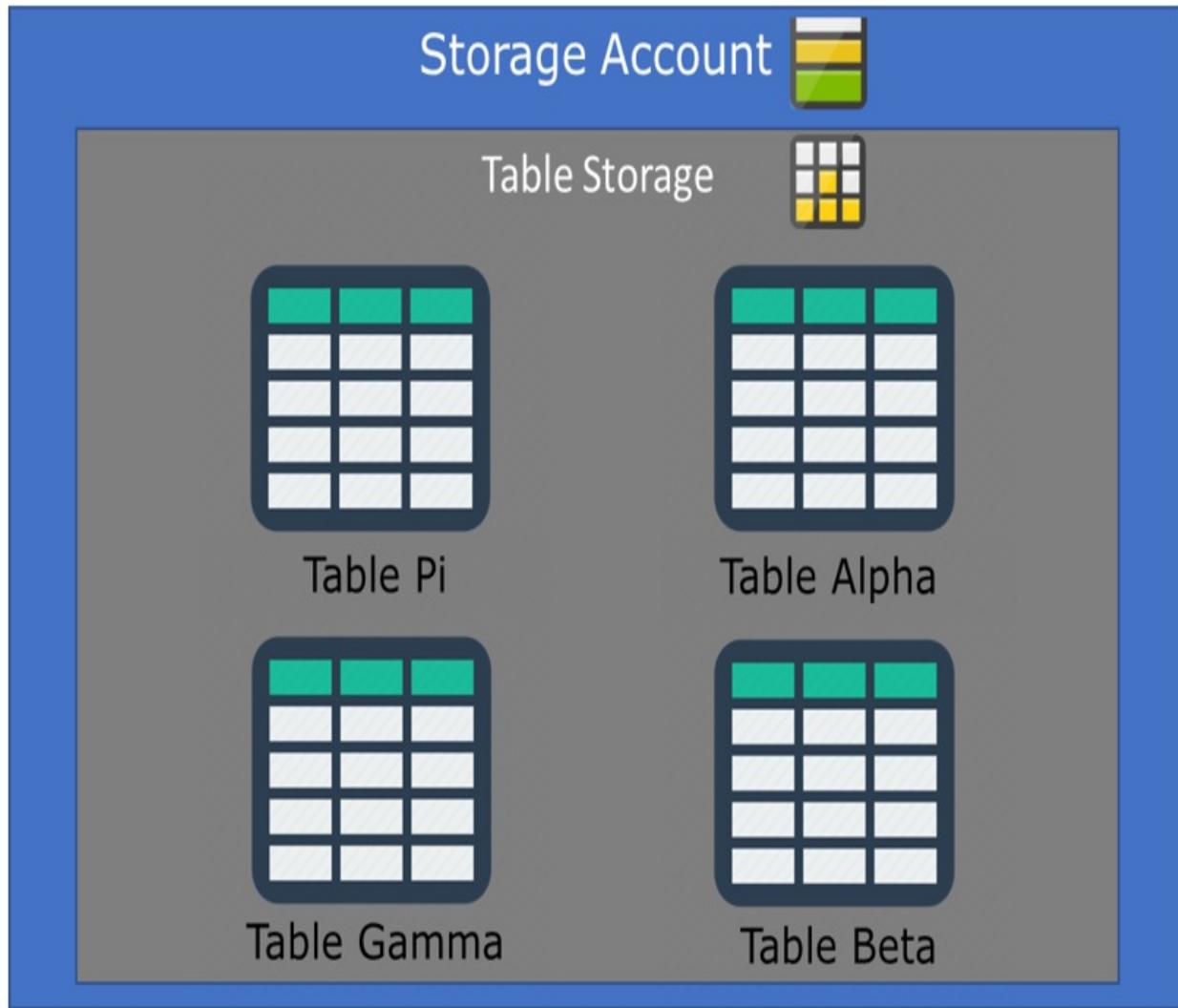
AsBytes      : {66, 97, 110, 110...}
Id          : eec927c5-5065-44e9-86ab-13b58420d2d7
PopReceipt   : AgAAAAAMAAAAAAA/faqBT/Z1wE=
InsertionTime : 10/11/2021 11:23:29 AM +00:00
ExpirationTime : 17/11/2021 11:23:29 AM +00:00
NextVisibleTime : 14/11/2021 10:04:33 AM +00:00
AsString     : Banned book order 123456
DequeueCount  : 1
```

If the *GetMessageAsync()* call failed, the message would remain on the queue and the next process requesting a message from the queue would then get that message again. You are guaranteed that queue messages are only de-queued when they have been processed, or at least until the process reading the message removes it. Alright, only one more core type of storage to go.

## 5.6 Table

The last of the four types of storage you can have within a storage account is table storage. This is not a database of tables like Microsoft SQL or Cosmos DB (covered later in the book), but instead structured data that is placed into buckets called tables as shown in Figure 5.15. In this example you can see four tables inside the table storage account, which again is inside the storage account itself.

**Figure 5.15: Multiple tables inside table storage.**



Think of table storage as the go-kart of storing data on Azure. It isn't particularly sophisticated, nor does it have a lot of features for you to play with compared to a full car. Instead, it is lightning fast, very easy to use and incredibly cheap. You get a no-frills type of database that has a bunch of benefits for the right scenarios.

Banning Books wants to create a searchable list of all the tech books ever made to keep on top of their inventory and which books have been banned. Banned books are their core business and they need to find simple book information *really* fast to deliver the data to applications and customers. Table storage is perfect for this, so they use the PowerShell cmdlet *New-AzStorageTable* to create their first table storage.

```
PS C:\>New-AzStorageTable -Name 'Books' -Context $storageContext.
```

The return value from this script is the table universal resource indicator, or URI, endpoint to access the data in the table storage. The URI will look something like this.

```
<pre class="codea"><a href="https://banningstorage.table.core.win"></pre>
```

Does that look familiar? The URI is very similar to the one used for blob, file and queue storage, which, of course, is by design, so it is easy to use and remember. Just like folders in a file share that holds files and data, tables in table storage are where the data lives. Each table has entities, which are like a row in a traditional database. These entities are a set of properties, which are key/value pairs of data, with the key being the specific property name as shown in Figure 5.16.

**Figure 5.16: The entity, property, and key/value pairs of a table in table storage.**



### 5.6.1 Schema-less

Table storage is schema-less, which means there is no pre-determined rules for how the data is structured. Many databases, which we will cover later in the book, rely on a schema to validate data against. If you insert or update data that doesn't conform to the schema, it is rejected. With table storage there is no such validation. If you think of the tables as buckets, you can put any kind of data in each bucket. It is up to you to keep track of what is in each, and in fact you can have many formats of data in the same bucket... I mean table. This makes it super-fast, but there is no safeguard for the formatting of your data. You can't have everything.

### 5.6.2 Inserting data

Let's insert some data into the table *Books*. After all, the tables are no good without data in them. Data inserted is done in *entities*, which are like rows in a table, as shown in Figure 5.16 before. Each entity defines the data schema for itself, using JSON notation. To insert data into a table in table storage using PowerShell, you use the *CloudTable* object. An object is a defined set of data, which we can reference, and then insert into the specified table. Using the *CloudTable* object is mandatory and means we need that as a start.

```
PS C:\> $cloudTable = (Get-AzStorageTable -Name Books -Context $s
```

We are using a new kind of syntax this time too, which is really a combination of two PowerShell statements into one. Everything inside the bracket *Get-AzStorageTable -Name Books -Context* *\$storageContext.Context* returns a reference to the *Books* table. We could have stored that in a variable such as *\$tableref* and used *\$tableref.CloudTable*. Instead, we enclose it in brackets and reference the *CloudTable* property directly from the object that *Get-AzStorageTable* returns. This is easier to read and more concise.

Now that we have the *CloudTable* property stored in the *\$cloudTable* variable, we can then use that to insert some data, but what data? You use the cmdlet *Add-AzTableRow* with three distinct pieces of information; partition key, row key and property. Run the below PowerShell script and we will go through what it all means.

```
PS C:\>Add-AzTableRow -table $cloudTable -partitionKey 'bookkey1'
```

**Note:**

If the *Add-AzTableRow* doesn't work, you might have to explicitly install the Azure Table Storage library into PowerShell using the command: *Install-Module AzTable*

The *partition key* is the first half of a unique identifier for the row, or entity. The partition key is also an indication to which physical partition you want the data to be stored in. If a partition is experiencing high load, it can be moved to another physical machine in Azure, or if it grows too big it can be moved to a physical machine with more space. With careful planning you can

optimize how the data is accessed if you use adequate partitions for the right data in the table. More on that later in the book too.

The *row key* is the other half of the primary key for the entity. This is a unique value, which is why we use a GUID value in the PowerShell above. Each row in the table must have a unique row key. Think of the row key as the unique identifier within a partition and the partition key + row key as the unique identifier within the entire table.

Finally, the *property* parameter is the data the row holds. The syntax is a long list of key/value pairs with a maximum of 252 keys, or columns (Azure reserves a few to make it 255). In this case we have two columns, *title* and *bookid*. The max size of the data in a single row is 1MB, so we aren't talking about storing your collection of images from "stuff on my cat", nor your collection of 1980s Japanese anime. Azure Table storage is cheap, flexible, and massively scalable. It is a foundational service of Azure, and one you will be using all the time.

And that brings us to the end of the four core types of storage inside an Azure Storage account. As you have learnt so far, each type of storage serves a basic purpose, and each comes with their own strengths as outlined in Figure 5.17.

**Figure 5.17: The benefits of each type of Azure storage.**

## Storage Account



### Blob



Huge files  
Any kind of binary format  
Data stored in containers  
Tiering of data for performance and cost

### File



Like a normal file system  
Attach to VM  
Tiering of data for performance and cost

### Queue



First in – First out queue (FIFO)  
Decouples services  
Holds millions of messages

### Table



Simple and fast  
Key/Value Storage  
Massively scalable  
Very cheap

There is one more basic type of storage that I want to show you though. Keep reading. Go on.

## 5.7 Managed Disk

When we talked about virtual machines, and we created one, I kind of glanced over the disks that VMs use. These are managed disks and is a special kind of storage within Azure. I briefly mentioned earlier in the chapter that one of the common use cases is to attach a file storage account to a VM. Managed disks are like actual physical hard drives that you have in your computer. You attach one or more to a VM and use them like you would a hard drive. Although they are storage just like all the other types we have gone through in this chapter, they aren't quite the same.

Remember the URLs for blob, file share, queue, and table storage? Looked something like this: `https://banningstorage.<storage type>.core.windows.net`. That URL exists because all of the data can be accessed over HTTP/HTTPS using the Internet, if permissions allow. Managed disks don't have this option for access. They are designed to be attached to a VM, but that also brings a lot of the benefits of VMs, and Azure, with it.

- **Durability:** Just like Azure storage, managed disks have at least three replicas of your data. This ensures persistence of the data and gives a 99.999% availability guarantee.
- You can create up to 50,000 managed disks per Azure subscription, which in conjunction with VM scale sets provides massive **scalability**.
- Using **availability sets**, managed disks are isolated from each other to eliminate any single point of failure.
- Your managed disk will be in the same **availability zones** as your VM.
- And finally, there is full backup support, just like for your VM.

Managed disks are important when creating a virtual machine, as there is a lot less overhead and maintenance involved in keeping the disk healthy. You can create managed disks at any time for a VM, and the data on these disks is persisted, even if you turn the VM off.

Managed disks are a foundational part of running Azure VMs and storing data, but we will go into more detail about them later in the book. They are a foundational part of the storage story in Azure.

## 5.8 Summary

- A storage account is one of the three main pillars in infrastructure-as-a-service (IaaS) services on Azure, along with virtual machines and networking.
- Each storage account will have a defined redundancy, which can be local, geo, zone or geo-zone.
- A single storage account can hold all four types of data storage at the same time, which makes it simpler to logically separate data storages by project, client or department.

- Each storage account has a unique URI that identifies access to it either from within Azure or the Internet if the permissions allow.
- Blob storage can hold any kind of data objects, up to 190.7 TiB in size for block blobs.
- File shares are like a traditional file system and is excellent as a detached centralised storage.
- Queue storage can hold millions of messages while each message is processed one at the time.
- Table storage is a schema-less super-fast way of storing data in logical tables. Each table consists of entities, properties and key/value pairs
- Each entity in a table in table storage has a unique identifier made up of the partition key and the row key.

[1] Pronounced tebibyte and is  $2^{40}$  bytes = 1,099,511,627,776 bytes

[2] Pronounced mebibyte and is  $2^{20}$  bytes = 1,048,576 bytes.

# 6 Security

## This chapter covers:

- Understanding Azure security fundamental concepts including defense in depth, shared responsibility and zero-trust approach
- Creating new identities in Azure Active Directory
- Improving your security posture using Microsoft Defender for Cloud
- Enabling multi-factor authentication for Azure Active Directory

Every aspect of Azure involves security. After all, we are trusting someone else with all our data, processing, transfers and top-secret business logic. Luckily, Azure is designed and built from the ground up with security as a top priority, which means we can trust it, right?

This chapter focuses on that built-in security, all the things that you get for free when you use the cloud services that Azure offers. However, we will also dive into the parts that you can control, configure, and, of course, mess up. As security is in a sense another foundational part of Azure, this chapter won't be delving into each product and how to make it the most secure for your project or application. Instead, we focus on the security concepts that run through all the products like a shining vein of titanium-plated armor and secures your architectural wonders.

## 6.1 A secure foundation

Let's start with the question I get most of the time when talking to people about Azure and security: Is the cloud secure? It is a common question, but it isn't a very good question. Why, might you ask? Because security isn't a toggle switch you switch on or off. It isn't a binary state where you can claim "I am now secure!" Instead, it is a sliding scale going from less secure to more secure (Figure 6-1), which then implies that you can never "be secure".

Figure 6-1: The security continuum.

## Security Continuum

Less Secure

More Secure

You can only aspire to become as secure as possible within certain constraints, such as budget, time, technology, and knowledge. I'll help you with the latter two, but you will have to sort out the former two yourself. With the *security continuum*, as I like to call it, defined we can start digging into how Azure can provide one of the most secure public computing platforms for your applications.

As I mentioned at the top of the chapter, Azure has security built in from the very foundation of every single service. In fact, the physical infrastructure of Azure data centres has state-of-art security measures as well, which make sure no one tampers with the servers, disks and network cables. Starting with this foundation, security is then at the forefront of every part of Azure. Let me give you an example.

As you learnt previously, it is very easy to create a virtual machine on Azure. Use your tool of choice, such as the Azure Portal or PowerShell, and in minutes you have a fully featured virtual machine that will do your bidding. The VM is created on a network that is secured, and the storage attached to the VM is encrypted by default. The hardware is bespoke and made for Azure data centres specifically with security in mind. When you connect to the VM, it is either via SSL encryption or SSH to keep it secure. Once you have connected, role-based access control (RBAC) might ensure you are both authenticated and authorized to access the VM, or you might even use a passwordless approach. Once the VM is in use, Azure Policies will ensure only approved features, applications and functions are used by a particular user.

Once the VM is in use, Azure Defender for Cloud will watch over the machine and surface any alerts about best practices that can make your VM even more secure, as well as detect potential vulnerabilities. If you want

intelligent security analytics on top, Azure Sentinel lets you do that too, which we cover later in the book. Using artificial intelligence, Azure can detect patterns in user behavior that is out of the normal routine, flag them, and let you investigate.

All of this happens without you having to write any code or configure any scripts. Some features need to be switched on, and some will require basic input, but it is all part of the secure foundation that Azure provides. As I mentioned before, you need to do some of the work too but compare the effort to “rolling your own” and the advantages are obvious. We won’t actually do that comparison now, but you get the idea. All of what I described just then is part of why this chapter is so important for getting a solid understanding of Azure and the foundations of cloud computing. Now, let’s dive into the depth of Azure defenses.

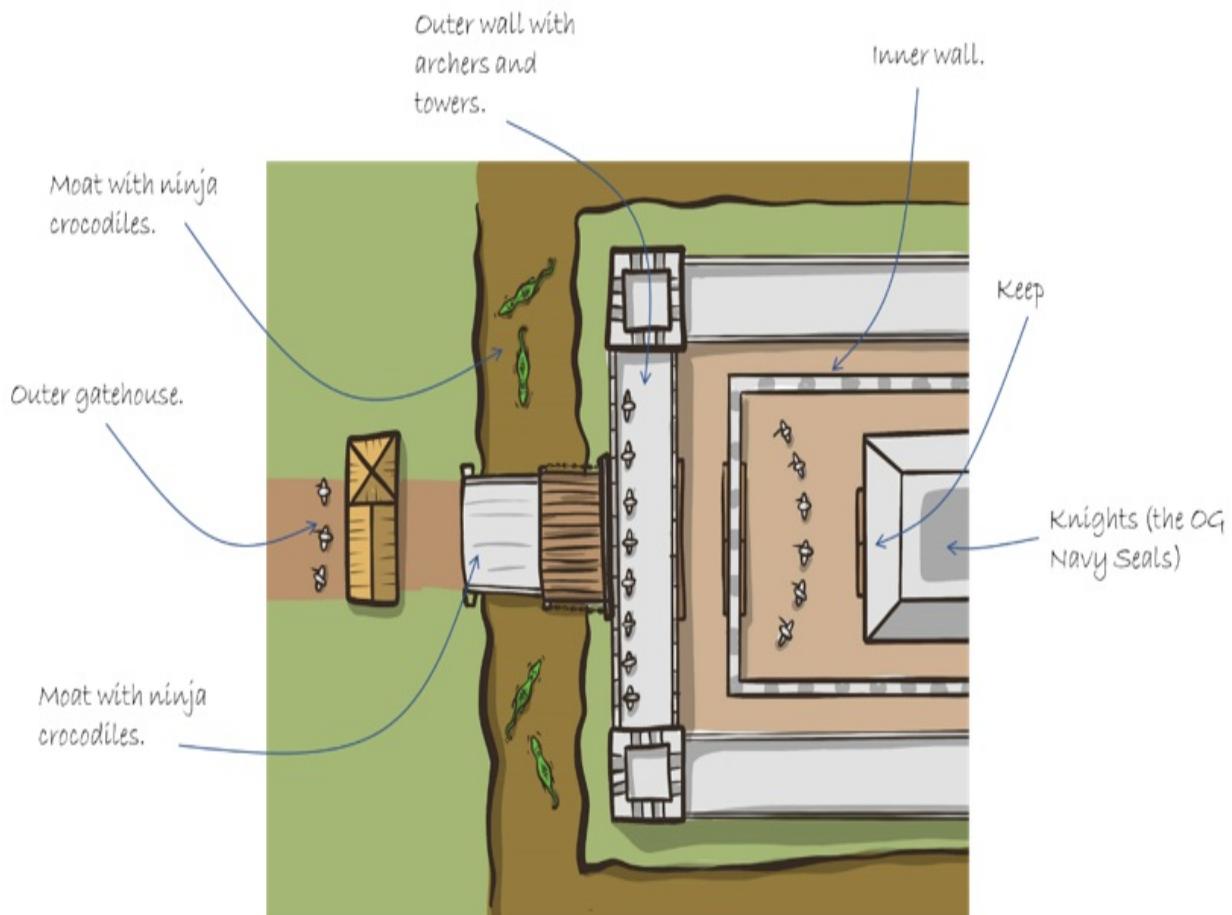
### 6.1.1 Defense in depth

You probably get it by now, and apologies for repeating myself, but it is an important point: Azure security is not a single switch or feature you subscribe to, turn on, or even a service you can subscribe to. Because of the *security continuum*, any computing infrastructure, cloud or not, requires a layered approach, which is often referred to as *defense in depth*. To explain the idea, we need to travel about 600 years back in time. Bear with me, it will be a journey of knights and fair princesses.

When a medieval king was sitting in his castle, he didn’t rely on just a few guards to defend himself, the princess and all the treasures in the vault. Did they have vaults in castles? Anyway, he needed several methods of protection both for backup and for redundancy. As shown in Figure 6-2 the castle has an outer gatehouse, which is a first check on anyone approaching the castle. Beyond that is a moat surrounding the entire castle. Perhaps the moat is even filled with ninja crocodiles. Who knows? To get over the moat, there is a drawbridge, which of course can be drawn up in case of an attack or emergency. The drawbridge leads to an outer wall, which is manned with knights and archers. The outer wall also has towers and turrets to provide cover. Past the outer wall is a piece of no-mans land before you get to the inner wall. That is again manned by knights, archers and other defenses. Past

the inner wall is the actual castle, which holds the keep. The keep is where the king is sitting, but he is surrounded by a last ring of navy seals. Oh wait... knights. Yes, knights!

**Figure 6-2: The seven layers of castle defense.**

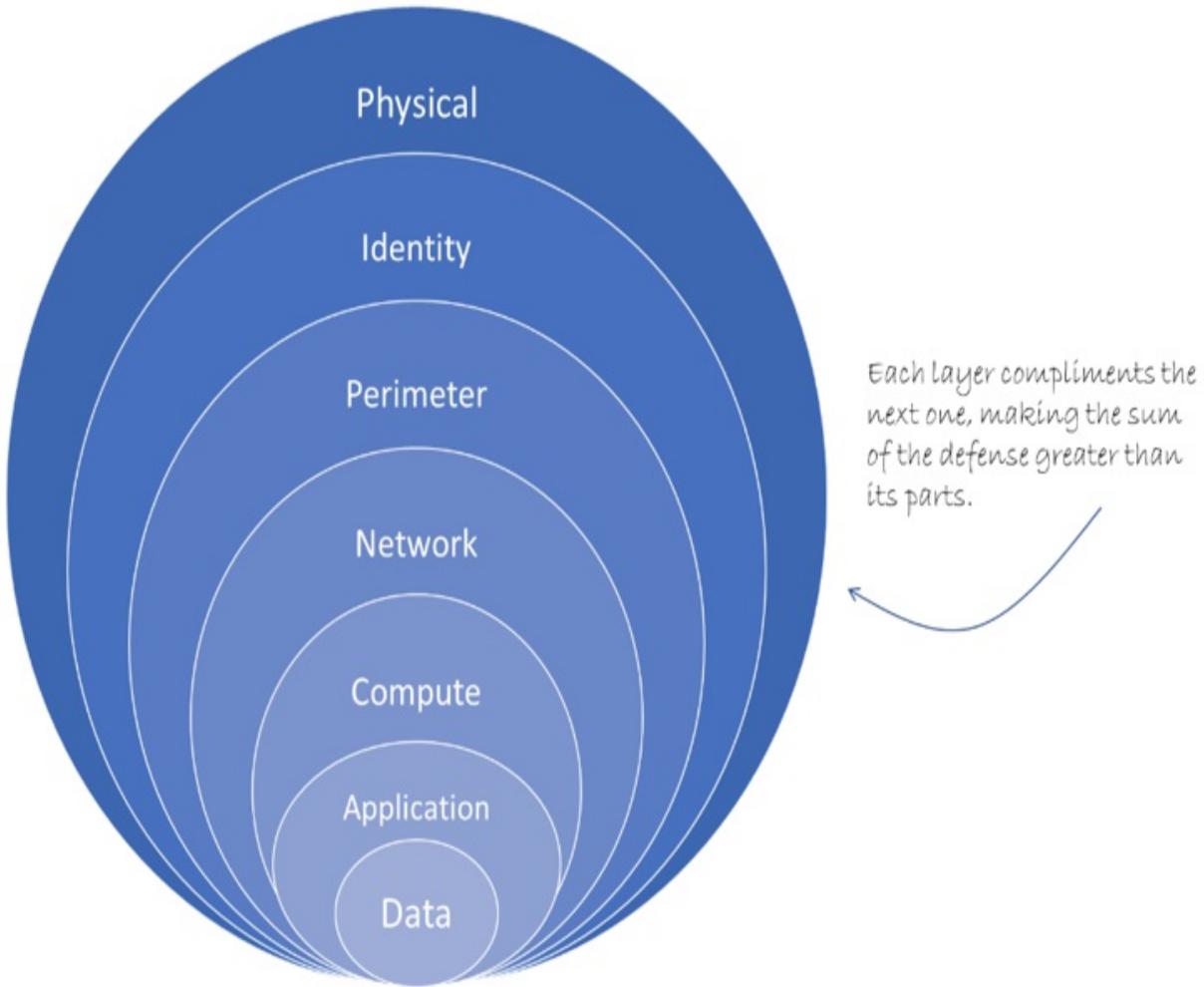


In other words, the king is behind gatehouse, moat, drawbridge, outer wall, inner wall, keep and special forces knights. That make seven layers of defense for his majesty. Fast forward to present day, and Azure adopts the same approach of defense in depth, but of course the king is now your data. Azure also has seven layers of defense that are all designed to protect your king... I mean data, as shown in Figure 6-3.

- Physical: The first line of defense is the physical datacenter, the building itself. Only authorized personnel is allowed and is authenticated upon entry.

- Identity: Being able to log in to Azure using the identity of the user. Each user on Azure has their own identity that includes credentials such as username and password, multi-factor authentication (MFA), and audits of user access. This is all managed through Azure Active Directory. We'll cover MFA later in the chapter in more detail.
- Perimeter: This is what Azure puts around their entire platform. A perimeter of services that protects the entire platform against large scale attacks, such as a distributed denial of service (DDoS) attack. It also includes services specifically for your network, such as a firewall.
- Network: By default, services on Azure can't connect over the network, or at least the communication is very limited. Traffic to and from the public Internet is restricted and monitored to ensure no bad actors are finding their way in. This defense layer also includes the facility to secure any connections from on-premises to Azure.
- Compute: Whenever a VM is fired up and you start using it, it becomes a security risk. Nothing against you and your security efforts, but VMs are prime targets for attacks and exploitations. Azure knows this and provides secure infrastructure to run the VMs on and services for you to connect securely to them, such as with Bastion. Azure also ensures that security patches and updates are always applied to the entire platform.
- Application: Your application is at the heart of why you use Azure. To solve a business problem and provide a service to your customers. Azure knows this and provides application specific services such as the Application Gateway to manage the traffic to and from the application.
- Data: Finally, the king of the layer cake. Yeah, that analogy totally works. Data is what every business collects, protects, sells and manages. Every product has data in it in some way, and often this is what attackers are after. They want your users' passwords and data, they want the competitive information for your products, they want the source code for your application. Data is stored in databases, on disks, in files and many other places. Azure employs a range of security measures to protect your data, including encryption, secure hardware, and access policies.

**Figure 6-3: Azure layers of defense in depth.**



Just like with the castle 800 years ago, the multiple layers of defense make Azure greater than the sum of its parts. When the enemy approached the castle, they would have to fight each layer of defense, losing some strength every time. As they progressed through the gatehouse, moat, drawbridge and walls, they would lose some forces at each step making it harder and harder to get to the treasury. This is the same principle with Azure. As attackers have to go through multiple layers to get your data, the effort needed increases exponentially at each step. Alright, enough about knights and kings. Let's talk about the parts that you are responsible for.

## 6.2 Shared responsibility model

While you can rely on Azure to cover your backside for a lot of areas, there are still parts you need to take responsibility for. The shared responsibility

model defines at what point responsibility transfers from you to Azure, in each of the three types of cloud computing hosting: infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). You might remember the definitions of those right back from the start of the book in chapter 1. As shown in Figure 6-4 each of the cloud hosting options carry with them a different split of the responsibilities, of which there are seven. These seven responsibilities are different to the seven layers of defense though, just to be clear.

**Figure 6-4: The shared responsibility for security and integrity, between Azure and the customer.**

Responsibility	On-Prem	IaaS	PaaS	SaaS
Physical Hardware				
Operating System				
Networks				
Applications				
Identity & Access				
Connected Devices & Endpoints				
Data & Compliance				

 Customer    Azure    It depends

### 6.2.1 On-premises

Starting with on-premises infrastructure, you are responsible for.... everything. Microsoft isn't going to come and patch your servers or watch

your back for DDoS attacks. You are on your own. Of course, that is part of the value proposition of cloud computing, that you don't have to do *a lot* of the heavy lifting.

On-premises isn't all bad though and depending on your business requirements and existing infrastructure it can make sense. In fact, lots of businesses use an on-premises setup. However, I won't cover it in this book of course, as we are talking Azure and cloud computing. I include it in this comparison for completeness, and as a starting point for where your infrastructure responsibilities start.

## 6.2.2 IaaS

Once you move into the world of cloud and start with IaaS, it is a different scenario though. First, Microsoft doesn't want you to have access to their hardware, and you don't want that either. The whole idea is to use the hardware only when you need it, and not worry about where it is or how it works. With IaaS the physical servers, network and storage are all managed by Microsoft on the platform. You are guaranteed by Azure that you have access to the VMs, the disks are maintained and replicated, network cables are plugged in properly, and the infrastructure is working optimally.

Creating the core services that we went through in the first chapters of this book, such as VMs, Azure Storage and Virtual Networks, means that the responsibility of the security and maintenance of those services in part lies with Azure. When you create a VM, a default security posture is applied to the VM. This includes that all hardware complies with the ISO/IEC 27001 standard for legal, physical, and technical controls involved in an organization's information risk management processes. This is a very stringent standard that ensures Azure complies with numerous regulatory and legal requirements that relate to information security.

With IaaS you still have a responsibility to look after the security and maintenance when it comes to the operating system on the VM, your applications, network access, user management, the data access and any devices connected to your infrastructure. Of course there is always an exception to the rule, and Azure offers automatic patching for VMs, which is

almost a PaaS product.

No matter which service and which hosting type you use on Azure, the responsibility of the physical hardware is always Azure's. No more staying up until 3am building that new server you need because Dave fried the motherboard. Again.

### 6.2.3 PaaS

When you use PaaS services, your responsibility become less, and Azure's go up. However, it gets a bit less well defined, as the security responsibility falls into the category of "it depends". PaaS products and services vary in nature and depending on how you implement your solution, the responsibility of security and maintenance can be either Azure's or yours. As I said, "it depends".

To start with, one area is always Azure's responsibility when it comes to PaaS; the operating system. Whatever platform you choose to use, Azure will always apply security updates, maintain platform dependencies, and perform maintenance on the operating system. Platform as a service is just that; a platform you can use without worrying about its security posture.

There are three areas though where it depends on your scenario, whether Azure or you are responsible for the security of the platform.

- **Network control and configuration:** As we have already established in chapter 4, there is no cloud computing without networks. Regardless of which hosting solution you choose, networks are involved. For PaaS, the networking part is mostly managed by Azure, but in a few cases, you still need to manage access to VNets such as for hybrid solutions using VMs.
- **Applications:** Platform managed applications such as web services, IoT solutions, analytics services and more are a great way to reduce your responsibilities when it comes to security, as a lot of the switches, toggles and dials are managed by Azure. For example, setting up a web service on Azure means the security of the service is configured by Azure, as opposed to creating a VM, where you are responsible for it.

You still have some power to configure parts of the web service such as public internet access, but most of the security work is on Azure.

- **Identity:** One of the most crucial connections of any computing application is knowing the identity of users and granting them access to the system. In PaaS this is a shared responsibility between you and Azure. You need to configure the identity provider, the user identities and monitoring and logging of identity usage. Azure will provide powerful services like Azure Active Directory to enable you to carry out these actions.

## 6.2.4 SaaS

The last of the three hosting options, SaaS, has the least responsibility, handing over most of the security reigns to Azure. Just like with PaaS, the identity is a shared responsibility, as you still need to manage users for your application.

The second area for SaaS applications where responsibility is shared, is when it comes to client devices and endpoints. Azure can provide tools for managing some of these parts of your application, but it isn't responsible for what users get up to and how that affects your SaaS application.

## 6.2.5 Compliance and data classification

There are of course also a few areas of responsibility that never gets transferred to Azure, regardless of which hosting solution you are working with. Compliance and data classification is always your responsibility, so let's break that up a bit more.

- **Compliance:** Your solution may need to meet certain compliance requirements, such as for your specific industry, your local legislation, or for other security requirements. Azure will provide compliance with a lot of standards for the infrastructure and services they offer, but they can't know what your solution's compliance obligations are. That is for you to identify, implement and ensure at all times.
- **Data classification:** Hand in hand with compliance is the classification of your data, which only you can do. It is often a complex process that also

identifies what data is public and what data is sensitive, when it comes to a security posture. For example, if you are using a VM to store data for an application, it is up to you to decide how to protect and configure access to that data.

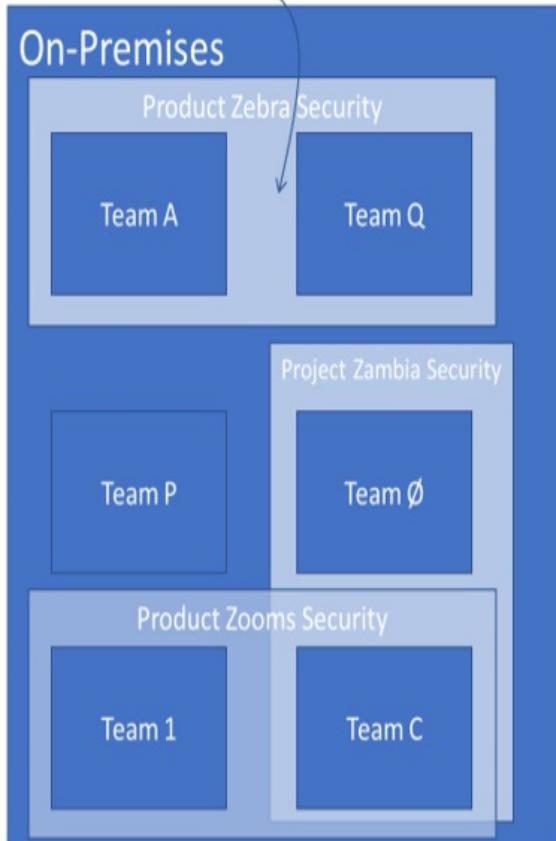
### 6.2.6 Leveraging cloud-enabled security

That was a lot of talk about how the shared responsibility applies to either you, Azure, or both. You may think by now “but if Azure only provides some of the security I need, and I am still responsible for a bunch of areas that could get attacked by cyber ninjas, why don’t I stick to my own environment that I can control better?”

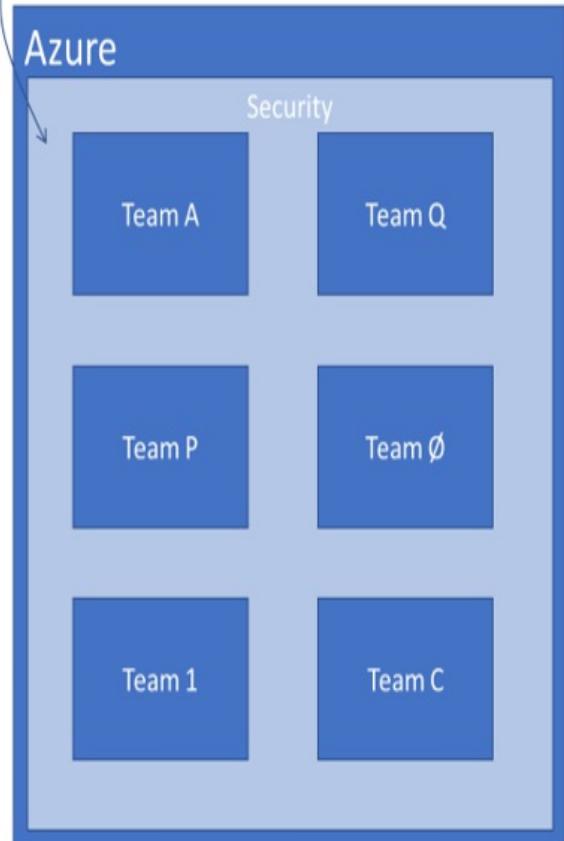
As shown in Figure 6-5 it is likely that an on-premises environment has limited resources when it comes to governance, security, and maintenance. After all, the IT infrastructure team only has one customer to look after, namely the business itself. There are many areas that may lack or have subpar security features implemented. When it comes to Azure, most of those areas are covered by dedicated staff that serve *all* of Azure’s customers, not just you.

**Figure 6-5: Security on-premises can be challenging and under-resourced compared to leveraging the Azure platform.**

Traditional on-premises approaches are challenged by multiple projects, teams and products each having their own security requirements supported by limited staff.



Leveraging Azure provides a security foundation that you don't have to maintain, update or even worry about.



Not only does that mean you have all of Microsoft's Azure engineers working for you, but they are specialized in certain areas and will apply best practices to areas you will never worry about. This means you have much more time, energy and budget to focus on the areas that really make a difference to your business or project. I am completely aware that this isn't going to give you any hands-on examples to take away, but it is very important that you have a good grasp of what security in Azure gives you, and what the platform doesn't.

The areas you *are* responsible for can now be focused on, knowing the rest of the infrastructure is secure. Furthermore, Azure will provide tools, such as Sentinel, which can help predict and prevent security incidents in real time. We will cover Sentinel in more detail later in the book too. Now, it is time to

talk about how you can't trust your users. Zero trust for them.

## 6.3 The Zero Trust approach and Azure AD

I am aware that there hasn't been a lot of practical hands-on fun in this chapter so far. Alpaca-Ma-Bags thinks the same thing, because they have a bunch of security improvements to do as well. We will get to all those upgrades in a moment, but we do have to cover the security fundamentals first though, so you know what to focus on and not, when it comes to your security posture. Yeah, that is the term information security experts love to use. "What is your security posture?" It just means how prepared you are to tackle any security issues coming your way, such as a cyber-attack.

Most of the services and features mentioned in this chapter are paid services. You won't be able to use them with a free Azure subscription. There are two reasons for this: 1) Azure knows you are keen on being as secure as possible by using the services already offered on Azure, so they will charge for them, and 2) security services require a large number of engineers, products owners and other staff to keep up to date and effective. Unlike a VM or website, security is a constantly moving target which require large investments from Microsoft. Hence, you need to pay a bit to tap into that large effort.

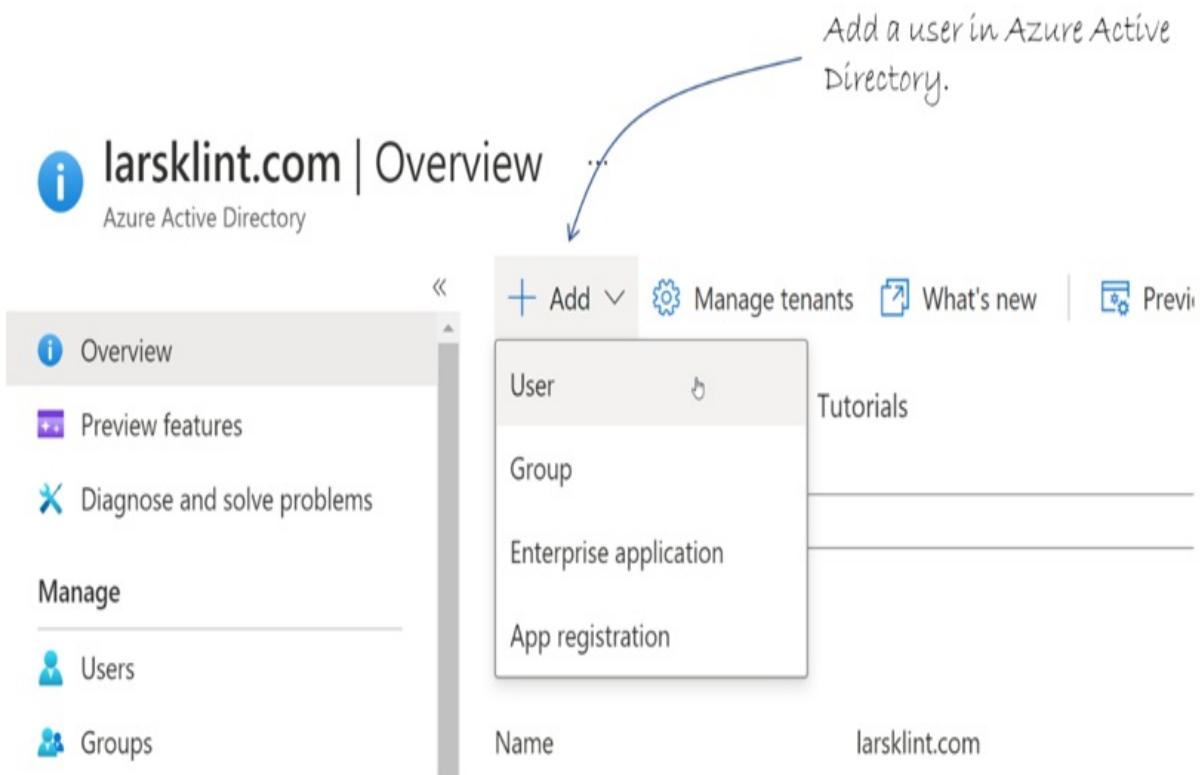
One of the best, and most recognized, ways to build an Azure environment that is more secure is to adopt a zero-trust approach, which is exactly what it sounds like. Don't trust anyone. Not your grandma, not your dog, not that new guy that needs access to your Azure infrastructure. Zero trust means you deny access to everything until the user is verified to access the resources in question. The idea is that it is much more secure to deny everyone access, and then slowly open up to each user when needed, instead of trying to guess what each user needs access to. It also stops anyone from either accidentally breaking stuff they shouldn't have touched or tempt users to access and manipulate resources they shouldn't have. If you don't have access, you aren't going to make a mess. In practical terms, implementing a zero-trust infrastructure has several parts and in the middle of it is Azure Active Directory.

We have discussed Azure Active Directory (AAD) in some of the previous

chapters, but to reiterate: AAD is where you manage all your users and permissions to Azure resources, among other things. If you have created an Azure account, you will have an AAD instance too. You can't create an Azure account without having a *tenant*, as the first instance of AAD is called. In fact, we will use that in a minute to create a user and then not trust that user at all. Zero trust.

AAD is key for a zero-trust approach, because we want to manage users. The users we don't trust. AAD is in the category of an identity management system (IMS), and that is where each identity lives. We use the term *identity* rather than *user* because an application or process can also have an identity. We aren't just talking about human users. To create a zero-trust infrastructure, we start with creating a user, or identity, in AAD as shown in Figure 6-6. Any identity that wants access to Azure resources must be in AAD to have credentials assigned to it.

**Figure 6-6: Create a new user in AAD.**



Creating a new user is trivial, and initially you can leave pretty much all the

options as default. As shown in Figure 6-7 a new user only needs a user name and a name. The rest of the options (not visible in the figure) such as password, group membership and permissions, can all be added later.

**Figure 6-7: Creating a new user only need a user name and name to start off with.**

The screenshot shows the 'New user' creation interface. At the top, there's a header with 'New user' and a '... larsklint.com' link. Below it is a 'Got feedback?' button. The main area has two options: 'Create user' (selected) and 'Invite user'. The 'Create user' section describes creating a new user in the organization with a user name like alice@larsklint.com, with a link to 'I want to create users in bulk'. The 'Invite user' section describes inviting a guest user with a link to 'I want to invite guest users in bulk'. A 'Help me decide' button is at the bottom. A callout arrow from the text 'Create a new user in the AAD tenant.' points to the 'Create user' option. Another callout arrow from the text 'Only user name and name are needed at first.' points to the 'Name' field in the 'Identity' section.

Create a new user in the AAD tenant.

Only user name and name are needed at first.

New user ...  
larsklint.com

Got feedback?

Create user

Create a new user in your organization. This user will have a user name like alice@larsklint.com.  
[I want to create users in bulk](#)

Invite user

Invite a new guest user to collaborate with your organization. The user will be emailed an invitation they can accept in order to begin collaborating.  
[I want to invite guest users in bulk](#)

Help me decide

Identity

User name \* ⓘ alpacamabags @ larsklint.com ✓ The domain name I need isn't shown here

Name \* ⓘ Alpaca Ma Bags ✓

First name

Last name

Click *Create* when you have had a look at the various options and put in a name and user name for the user. A bit later in the chapter we will go through how to set up multi-factor authentication to make sure users are safe (and who they say they are), and then also assign permissions to a user using role-based access control. Stay tuned for those goodies, but first, let's go through how Azure can help us stay safe and let us know how to improve our security posture.

## 6.4 Microsoft Defender for Cloud

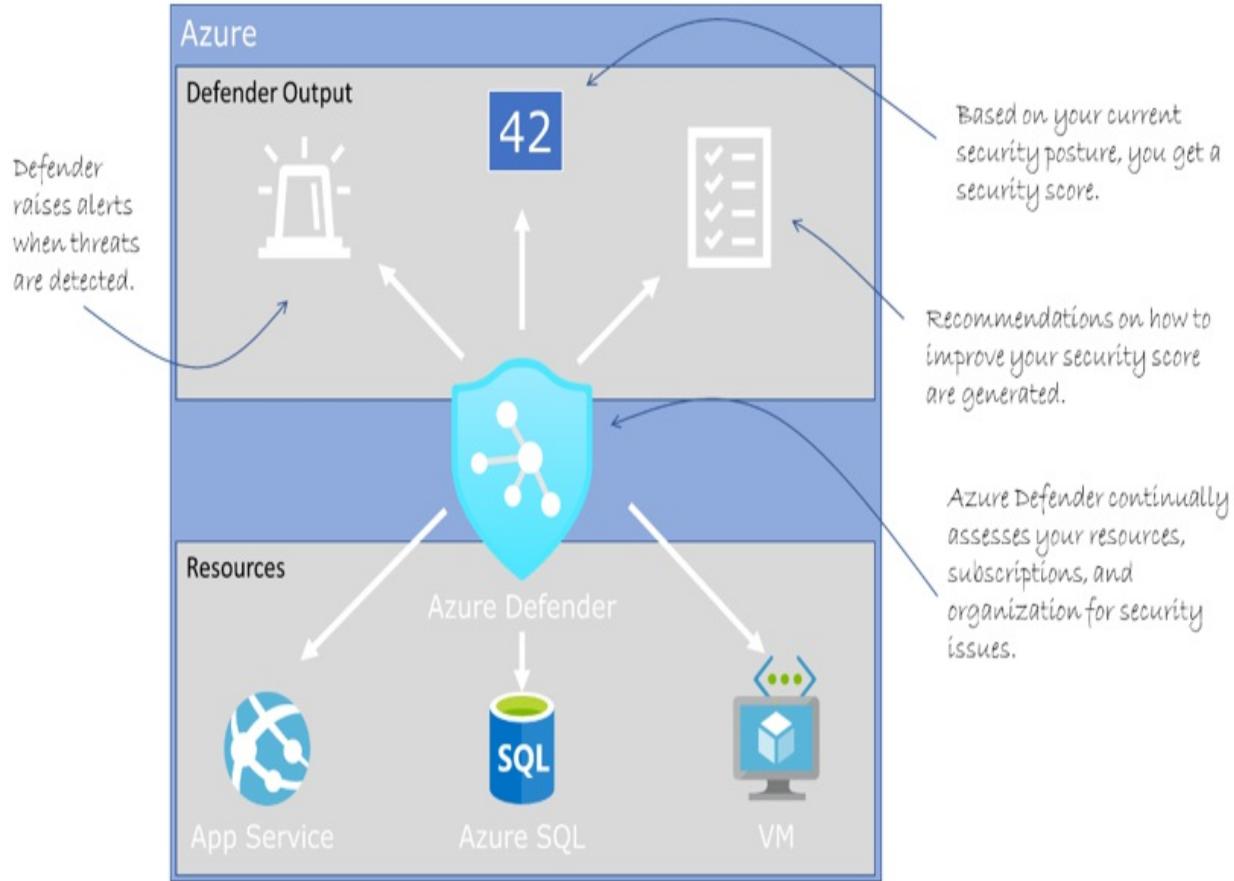
Even though we use a zero-trust approach, and we can monitor users and their access to services and resources, there will always be attacks on your infrastructure. To understand where Alpaca-Ma-Bags might have weak points in their cloud armor, they use Microsoft Defender for Cloud. Think of it as your own personal ninja that not only knows about incoming threats to your system and defends against them, but also learns over time. A machine learning ninja.

Defender works for Azure built-in services, such as databases, storage accounts, and app services. For Alpaca-Ma-Bags, this is ideal as they have a website, a file server, and a database, among other services. All of these services can benefit from using Defender. “But how?” you might ask, and that is the part we are getting to now.

Defender is what is called an *Azure-native service*. This means it lives entirely inside of Azure and doesn’t requi

re any deployment to be activated. As shown in Figure 6-8, it can help detect threats for PaaS services such as Azure App Service, data services like Azure SQL, and networks such as limiting access to VM ports. Defender will continually monitor your infrastructure and learn about new threats at the same time, by using machine learning. Putting all this together, a secure score is calculated to see just how secure your environment is and where on the security continuum you are. Defender will also raise alerts about threats and recommendations for improving your security posture and score. More on all that in just a moment.

**Figure 6-8: Azure Defender gets data from resources and provides security alerts and recommendations.**



First, let's orient ourselves on where Defender lives in Azure and what it looks like. Find the Defender for Cloud service using one of the Azure search tools, such as the Portal search bar shown in Figure 6-9. This will take you to the main Defender blade in the Azure Portal.

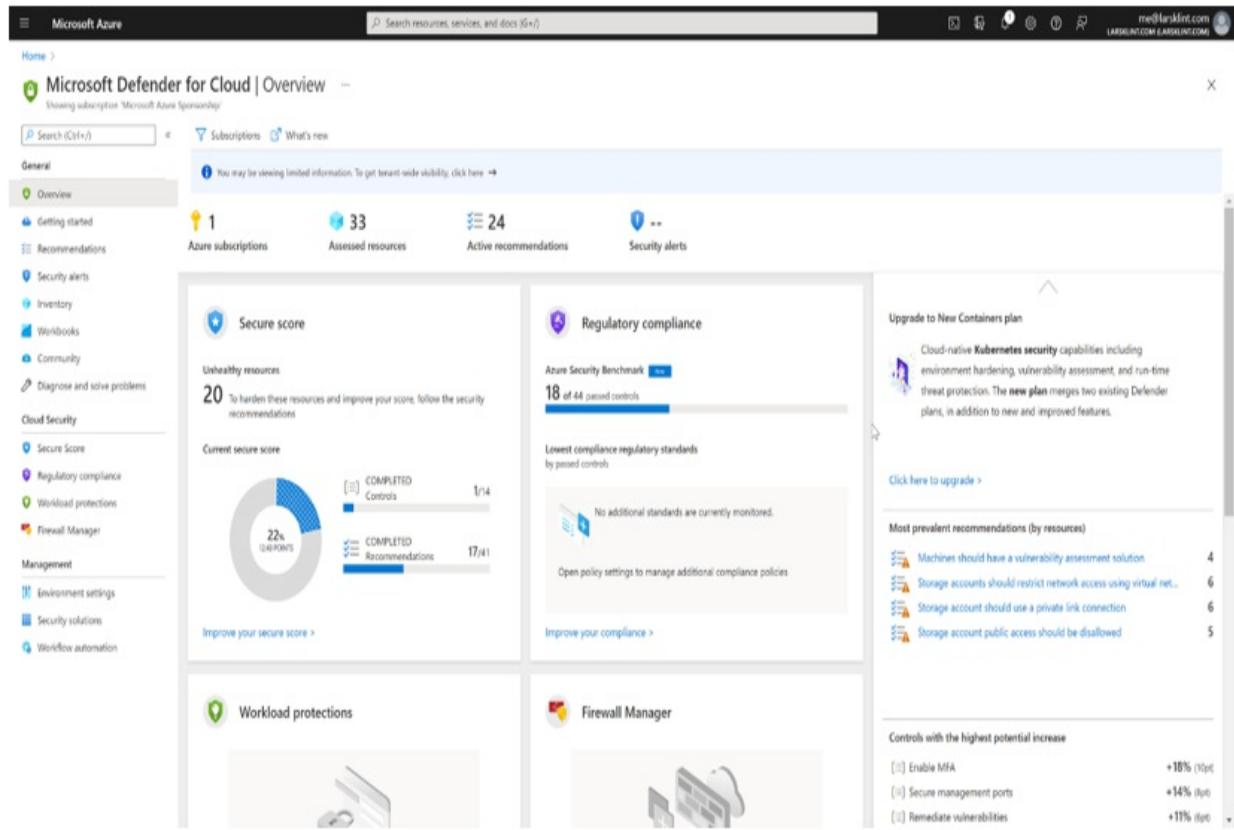
**Figure 6-9: Use the Azure Portal search bar to find Azure Defender.**

Find the Defender for Cloud section using the search bar.



The overview of all things Defender (Figure 6-10) is a compressed view of all the various parts Defender monitor for you, such as the secure score, compliance numbers and more. Notice the secure score here being 20. We will find ways to improve that in this chapter, while looking at parts of the Azure security continuum.

**Figure 6-10: Defender overview blade in the Azure Portal.**



There are three main outcomes from Defender, as I also alluded to in Figure 6-8: recommendations, the secure score and alerts.

#### 6.4.1 Security recommendations

The file server for Alpaca-Ma-Bags has been left as it was the day it was created. Nothing has been secured any further than what comes out of the Azure box from the start. We need to improve on that. On the file server, let's go to the security menu (Figure 6-11).

**Figure 6-11: The security menu is where Defender lives for a VM.**

Defender and other Azure security options are under the "Security" menu.

AlpacaMaBagsFileServer Virtual machine

Search (Ctrl+ /) Overview Connect Start Restart

Activity log Access control (IAM) Tags Diagnose and solve problems

Settings Networking Connect Windows Admin Center (preview)

Disks Size Security Advisor recommendations Extensions + applications

Advisor (1 of 8): Virtual machines sl

Essentials

Resource group (Move) : AlpacaRG

Status : Stopped (de)

Location : Australia East

Subscription (Move) : Microsoft Az

Subscription ID : 694ae2bd-e

Tags (Edit) : Click here to

Properties Monitoring Capa

Virtual machine

Computer name	A1
Health state	-
Operating system	W
Publisher	M
Offer	W

The security menu for the VM resource is not unique to the VM as a resource type. By far the majority of Azure services will have a similar security blade that will provide recommendations for how to improve the security for that resource, as well as the severity of each of those recommendations as shown in Figure 6-12.

Figure 6-12: Each of the recommended security fixes are described and rated for severity.

There are 8 recommended security improvements for this file server VM.

The screenshot shows the Microsoft Defender for Cloud dashboard. At the top, there are two counts: 'Recommendations' (8) and 'Security alerts' (0). Below these, a section titled 'Microsoft Defender for Servers' is shown as 'Off'. A callout bubble provides a link to 'Explore just-in-time access in Defender for Cloud'. On the right, there's a 'Learn more' section with links to 'About Microsoft Defender for Cloud' and 'Explore VM security capabilities'. A horizontal scrollbar is visible at the bottom of the dashboard area.

### Recommendations

Defender for Cloud continuously monitors the configuration of your virtual machines to identify potential security vulnerabilities and recommends actions to mitigate them.

Description	Severity
Machines should have a vulnerability assessment solution	Medium
Azure Backup should be enabled for virtual machines	Low
Virtual machines should encrypt temp disks, caches, and data flows between Compute and Storage resources	High
Log Analytics agent should be installed on virtual machines	High
Management ports should be closed on your virtual machines	Medium

Each recommendation is described, and the severity is rated.

Each of the recommendations will have details of why the issue is raised, such as for the encryption of temporary disks and data caches as the one shown in Figure 6-13. In this instance you can see the following

- The **severity** of the issue, in this case high.
- A **freshness interval**, which means how old the check for the issue is at a maximum. Here the check to see if it is still an issue is 24 hours.
- A **description** of the issue, which includes scenario where you can disregard the issue. This is important, as there is always more than one way to skin the cloud cat. There is more than one way to achieve a more secure infrastructure. In this case for encryption of disks and data flows, there are two scenarios where you can disregard this issue and close it.

- Some **remediation steps** which outline how you can fix the issue. For this VM we need to enable disk encryption. In certain cases, Azure can fix the security issue for you automatically without you having to change things manually. In this case, the solution is to manually enable disk encryption on the disks.

**Figure 6-13: The details of a specific security issue raised by Microsoft Defender.**

Virtual machines should encrypt temp disks, caches, and data flows between Compute and Storage resources

( Exempt View policy definition Open query

Severity	Freshness interval
High	24 Hours

By default, a virtual machine's OS and data disks are encrypted-at-rest using platform-managed keys; temp disks and data caches aren't encrypted, and data isn't encrypted when flowing between compute and storage resources. For a comparison of different disk encryption technologies in Azure, see <https://aka.ms/diskencryptioncomparison>. Use Azure Disk Encryption to encrypt all this data.

Disregard this recommendation if:

1. You're using the encryption-at-host feature, or
2. Server-side encryption on Managed Disks meets your security requirements.

Learn more in [Server-side encryption of Azure Disk Storage](#).

^ Remediation steps

Manual remediation:

To enable disk encryption on your virtual machines, follow [Encryption instructions](#).

Take action Trigger logic app Exempt

Finally, at the bottom of the security issue blade there is a *Take Action* button. Click it. Go on. It takes you to the disk blade for the VM, where we can now perform the necessary changes to fix the security issue. Isn't this cool? Azure tells you where there are problems with your system. The platform then goes on to give details on why it is an issue, so you can learn from it, rather than just accept it blindly. Finally, there is a single button that can either fix the issue for you, if you choose, or at least take you to the right place to fix it if Azure can't. Suddenly, keeping to the right end of the *Security Continuum* is made much easier.

## 6.4.2 Secure score

The secure score is a one stop shop for how well you are doing on Azure, security wise. It's a single number that tells you in an instant where on the security continuum your Azure environment is. Before I continue, I can tell what you are thinking right now though. "Lars, this score seems like a silly gimmick to please managers that just want a number saying how secure they are." You are kinda right, and kinda not right. Let me explain.

Yes, your manager will love that there is a number they can monitor, which gives an indication of your security posture on Azure. However, there is more to it than that. The score is calculated based on the individual security posture for each service in Azure you are using. This creates a number of recommendations from Azure to improve the secure score, shown in the secure score recommendations dashboard in Figure 6-14. Each of these recommendations are then broken down into their current score as a contribution to the overall secure score, and then, here is the really cool part, how much the score will improve for each recommendation when you fix the issue.

**Figure 6-14: The secure score recommendations, including possible improvements.**



By knowing how much the score will improve, you get an indication of which issues are more important in the overall scheme of your security journey, and you can then prioritize them accordingly. In the case of Figure 6-14 enabling multi-factor authentication (MFA) will give us 10 points, or an increase of 18%. In fact, we will enable MFA later in this chapter too.

### 6.4.3 Security alerts

Last, but not least, are alerts from Defender. These are raised when Defender detects a threat to one or more of your cloud resources. Security alerts within Microsoft Defender for Cloud are a paid service, however, you can create a bunch of sample alerts as shown in Figure 6-15.

**Figure 6-15: Create sample security alerts to learn before pay.**

The screenshot shows the Microsoft Defender for Cloud | Security alerts interface. A blue arrow points from the text "Use the Security alerts menu of Defender for Cloud." to the "Security alerts" link in the left sidebar. Another blue arrow points from the text "You can create a batch of sample alerts to learn from before you pay." to the "Create sample alerts" button in the center of the page.

**Microsoft Defender for Cloud | Security alerts**

Showing subscription 'Microsoft Azure Sponsorship'

Search (Ctrl+F)

General

- Overview
- Getting started
- Recommendations
- Security alerts**
- Inventory
- Workbooks
- Community
- Diagnose and solve problems

Cloud Security

- Secure Score
- Regulatory compliance
- Workload protections
- Firewall Manager

Management

- Environment settings
- Security solutions
- Workflow automation

What are Azure Defender security alerts?

Azure Defender protects your hybrid cloud environment by continuously monitoring it, running advanced security analytics, and generating a security alert for every suspicious activity that was detected. In the Security Alerts page in Azure Security Center you can triage your security alerts, investigate the findings, and quickly respond manually or with automated workflows.

How does it work?

Azure Defender analyzes a vast amount of logs from your Azure resources and the Azure supportive infrastructure, and using advanced algorithms and machine learning modules to detect threats targeting your hybrid cloud environment

Learn more >

Discover threats in early stage to quickly respond and prevent future attacks

Sample of the alerts you'll receive when Azure Defender is enabled:

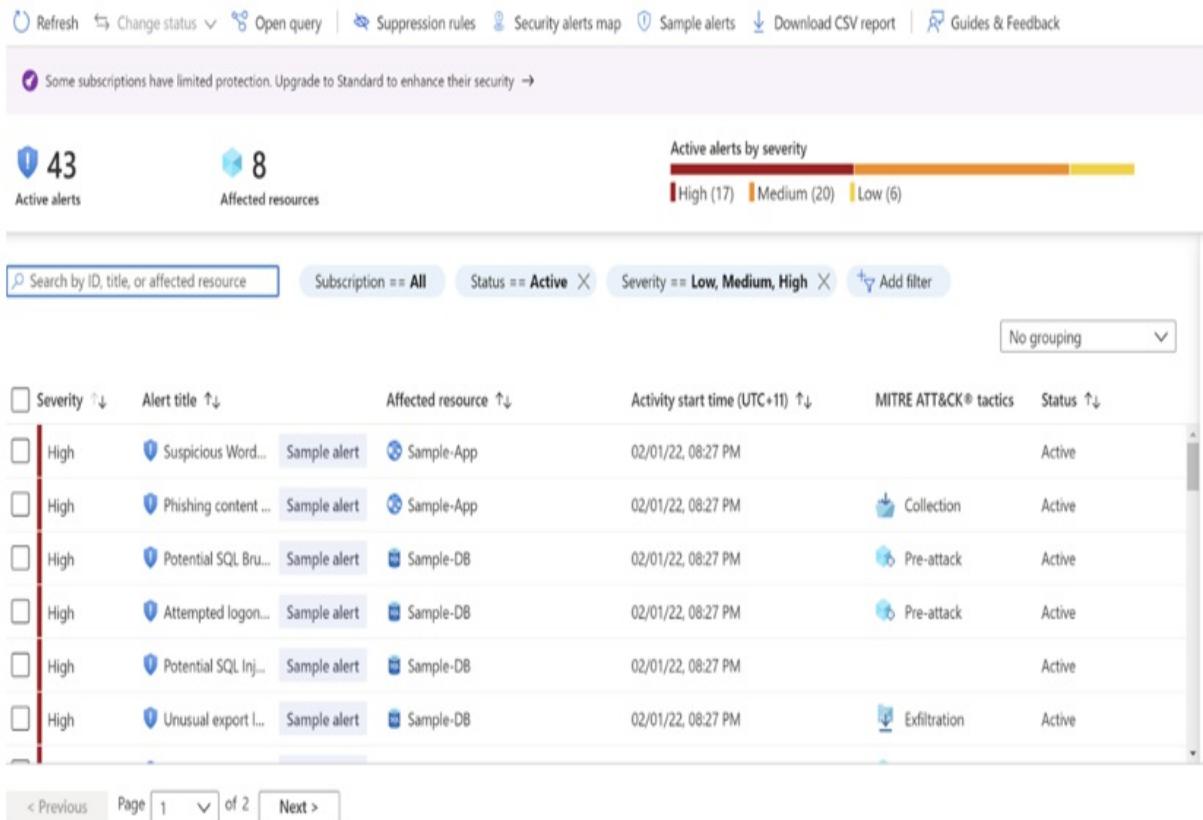
Severity	Description	Count	Detected by	Date	Status
High	Security incident detected	1	Microsoft	03/18/20	Active
High	Security incident detected	1	Microsoft	03/18/20	Active
High	Potential SQL injection	1	Microsoft	03/18/20	Active
High	Modified system binary discovered in dump file	1	Microsoft	03/18/20	Active
High	Successful RDP brute force attack	1	Microsoft	03/18/20	Active

Try Azure Defender free Create sample alerts

You can create a batch of sample alerts to learn from before you pay.

Choose the subscription and services you want sample alerts for on the dialog, and then click *Create* to get a batch of alerts (Figure 6-16).

**Figure 6-16: A list of sample security alerts generated by Azure.**



In this case Azure has generated 43 alerts for us, and as you can tell, there is a fair bit going on. The most important column is the *severity*, as you generally want to address high severity alerts first. That can change in certain circumstances such as if a particular resource is high value but has many medium or low severity alerts.

To get a better view of a single alert, click on it and a detail pane opens on the right (Figure 6-17), which provides more information about the chosen alert.

**Figure 6-17: Security alert detail pane.**

A screenshot of an Azure alert details page. The alert title is "Suspicious WordPress theme invocation detected". It has a "High Severity" and is currently "Active". The activity time is listed as "02/01/22, 08:27 P...". A callout arrow points from the text "A quick overview of the high-level alert info." to the alert title and severity. Another callout arrow points from the text "The description of the alert can reveal a lot of information to help resolve it." to the "Alert description" section. The "Alert description" contains sample text about a possible code injection attack on a WordPress theme. The "Affected resource" section lists "Sample-App" as a Web application IaaS and "Microsoft Azure Sponsorship" as a Subscription. At the bottom are "View full details" and "Take action" buttons.

Suspicious WordPress theme invocation detected

High Severity

Active

02/01/22, 08:27 P...

A quick overview of the high-level alert info.

The description of the alert can reveal a lot of information to help resolve it.

Alert description

THIS IS A SAMPLE ALERT: The Azure App Service activity log indicates a possible code injection activity on your App Service resource.  
The suspicious activity detected resembles that of a manipulation of WordPress theme to support server side execution of code, followed by a direct web request to invoke the manipulated theme file.  
This type of activity was seen in the past as part of an attack campaign over WordPress.

Copy alert JSON

Affected resource

Sample-App  
Web application IaaS

Microsoft Azure Sponsorship  
Subscription

View full details

Take action

Pay attention to the description of the alert, as there are often important details that can help resolve it. In the example in Figure 6-17 Azure has detected a vulnerability in a WordPress theme which seems to allow an attacker to execute code on the server. The obvious fix would be to replace the theme with a trusted one.

#### Note

Security alerts are not a free service within Microsoft Defender for Cloud. Consider creating alerts in circumstances where resources are both exposed to vulnerabilities and critical for your business processes.

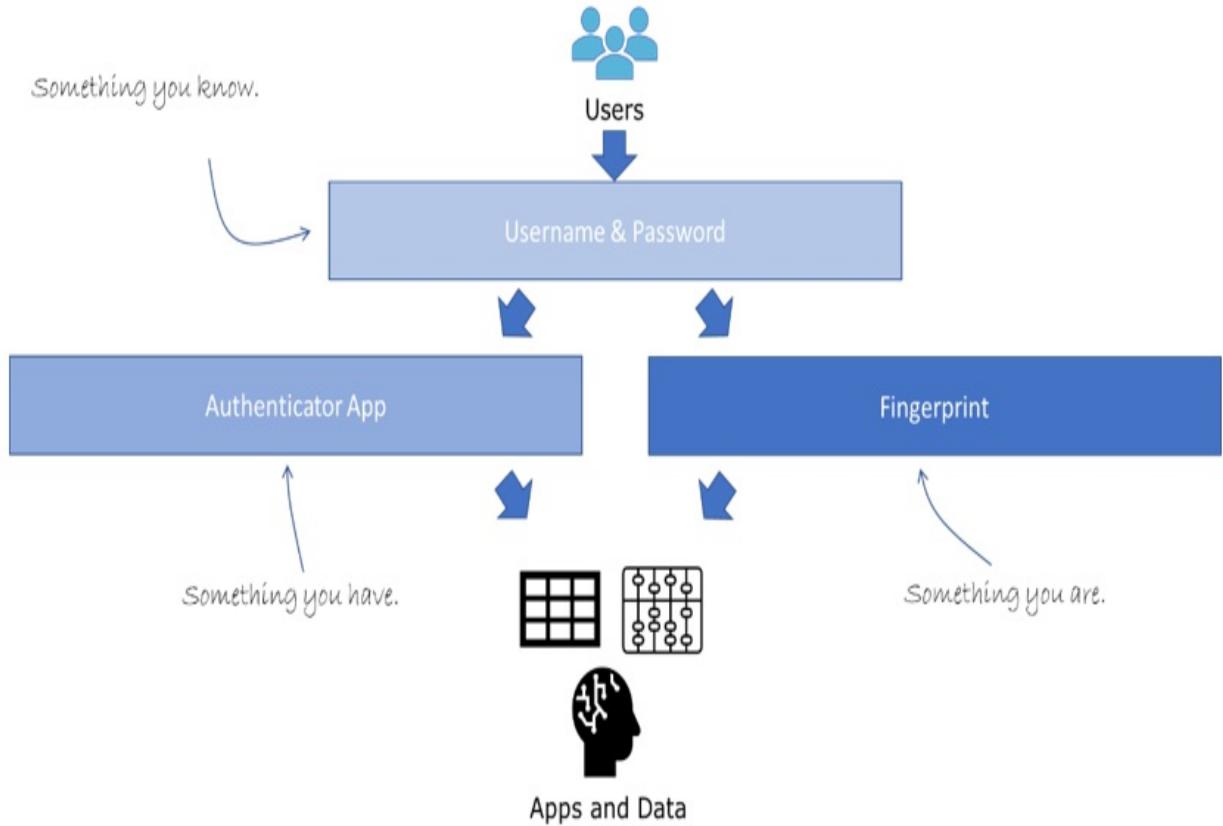
Alerts forms a great defense against threats along with the secure score and the security recommendations. Microsoft Defender has a lot more features than these three key components, and I will encourage you to explore them. However, the score, the recommendations, and the alerts are the foundation of Defender for Cloud. Use them. They will save your bacon. Let's change our focus a little bit and look at how to make the authentication process more robust.

## 6.5 Multi-Factor Authentication

Even if you aren't familiar with the term "multi-factor authentication", or MFA, you have most likely used it several times. You might even have used it today. When you authenticate with a service, the most common way is, still, the trusted username and password. But should it be trusted? If someone obtains your username and password, it is quick and easy to access the service with those credentials. The username and password is only one factor of authentication. You need to ensure your users have to provide more than a single factor of authentication for your Azure service if authentication is required.

Alpaca-Ma-Bags need to upgrade their security posture for their users, and part of that is using MFA through Azure. As shown in Figure 6-18 they want to make sure users need at least one more form of authentication before they are let into the wonderful world of plush luggage.

**Figure 6-18: You need at least two of the three types of authentication to satisfy the MFA flow.**



As the figure shows, you need at least two of the three types of authentication:

- Something you know, most often being a username and password.
- Something you have, such as a phone with an authenticator app or a physical hardware security key.
- Something you are, referring to your person. This is often a fingerprint or a face scan.

Having multiple factors of authentication is critical to keep user access secure, but also protect your infrastructure from weak passwords or compromised credentials. Users often use the same username and password combination across multiple sites and if it is exposed in some context, it is only a matter of time before attackers use the credentials on your infrastructure.

In the Azure Portal, let's head over to the Active Directory section and then click on the security menu, as shown in Figure 6-19.

**Figure 6-19: Access the MFA settings in the security part of AAD.**

larslint.com | Overview

(Preview) Add Manage tenants What's new

Licenses Azure AD Connect Custom domain names Mobility (MDM and MAM) Password reset Company branding User settings Properties Security Monitoring Sign-in logs

Overview Monitoring Tutorials

Search your tenant

Basic information

Name	larslint.com
Tenant ID	aded1c8b-a2e4-46dc-b...
Primary domain	larslint.com
License	Azure AD Premium P2
Alerts	

You will need a paid AAD subscription to use MFA.

Choose Security to get to the MFA section.

#### Note

You won't be able to configure MFA for a free version of Azure AD. You will need Premium P1 or equivalent to take advantage of MFA. Luckily, you can take advantage of one of the free trials of the tier, which Azure always seem to offer. Just don't forget to cancel it before first payment is due.

Once in the security section of AAD, use the *Conditional Access* menu item, which opens the blade shown in Figure 6-20. From there you want to create a new policy, which is a concept we briefly touched on in chapter 3. A policy is a defined set of rules that determines an outcome. In this case the rule is the MFA function for one or more users, or one or more groups. Whenever you see the word *policy* in Azure, think *rules that must be obeyed*.

**Figure 6-20: The security blade in AAD, where you can create a new MFA policy.**

Create a new policy to enable MFA for the AAD account.

Conditional Access | Policies

New policy | What If | Got feedback?

Create new policy

Create new policy from templates (Preview)

Conditions	Controls
When any user is outside the company network	They're required to sign in with multi-factor authentication
When users in the 'Managers' group sign-in	They are required be on an Intune compliant or domain-joined device

Want to learn more about Conditional Access?

Get started

- Create your first policy by clicking "+ New policy"
- Specify policy Conditions and Controls
- When you are done, don't forget to Enable policy and Create

Interested in common scenarios?

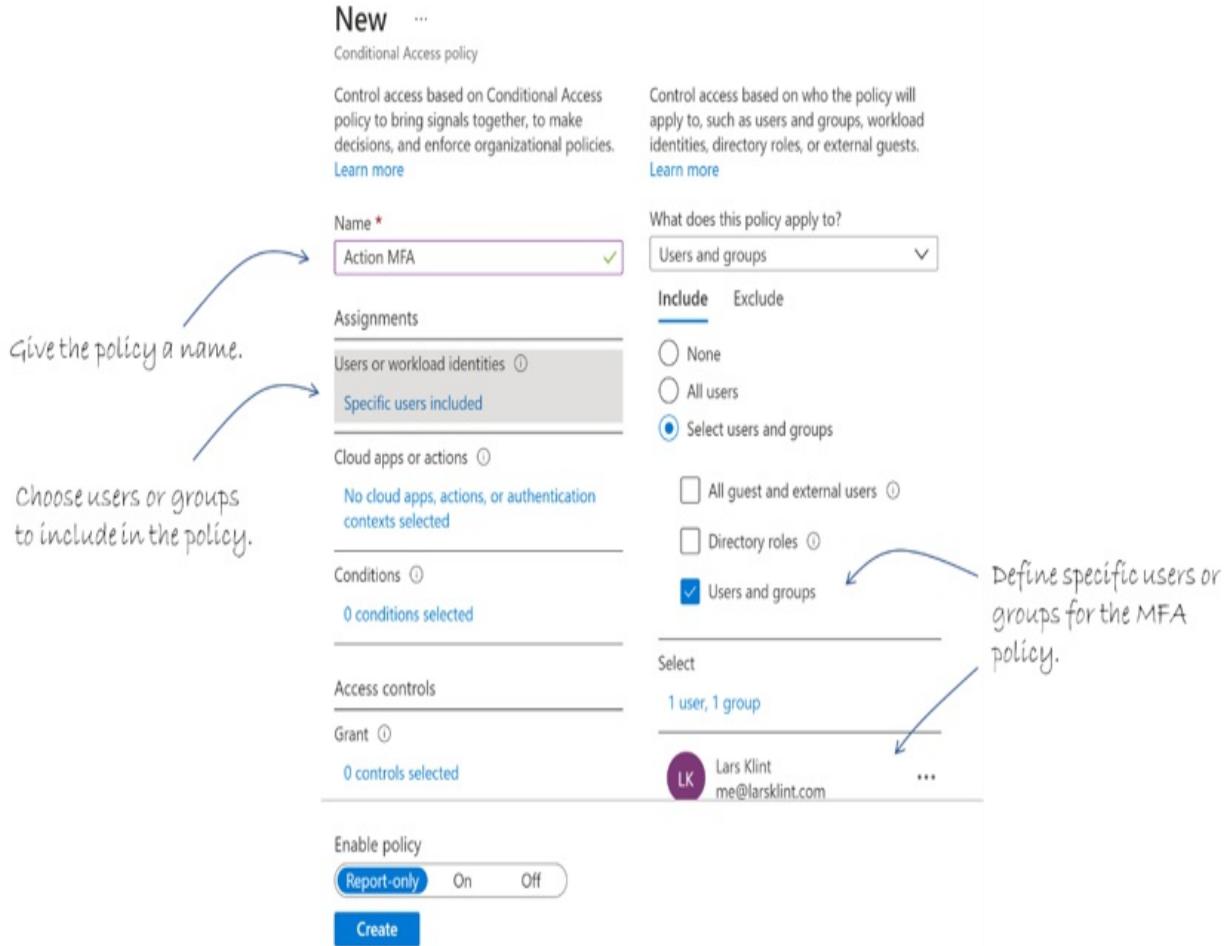
Named locations | Custom controls (Preview) | Terms of use | VPN connectivity | Authentication context (Preview) | Classic policies

Alright, click that *New Policy* button and let's create some MFA goodness for Alpaca-Ma-Bags. That will show you the wizard for creating a new policy as shown in Figure 6-21. You will have to give the policy a name and then choose some users or groups. These groups and users are the accounts in AAD that will be affected by the policy. Normally, an Azure AAD tenant will have a group that includes all users. This group will have other groups in it, such as administrators, standard users and other main users.

#### Note

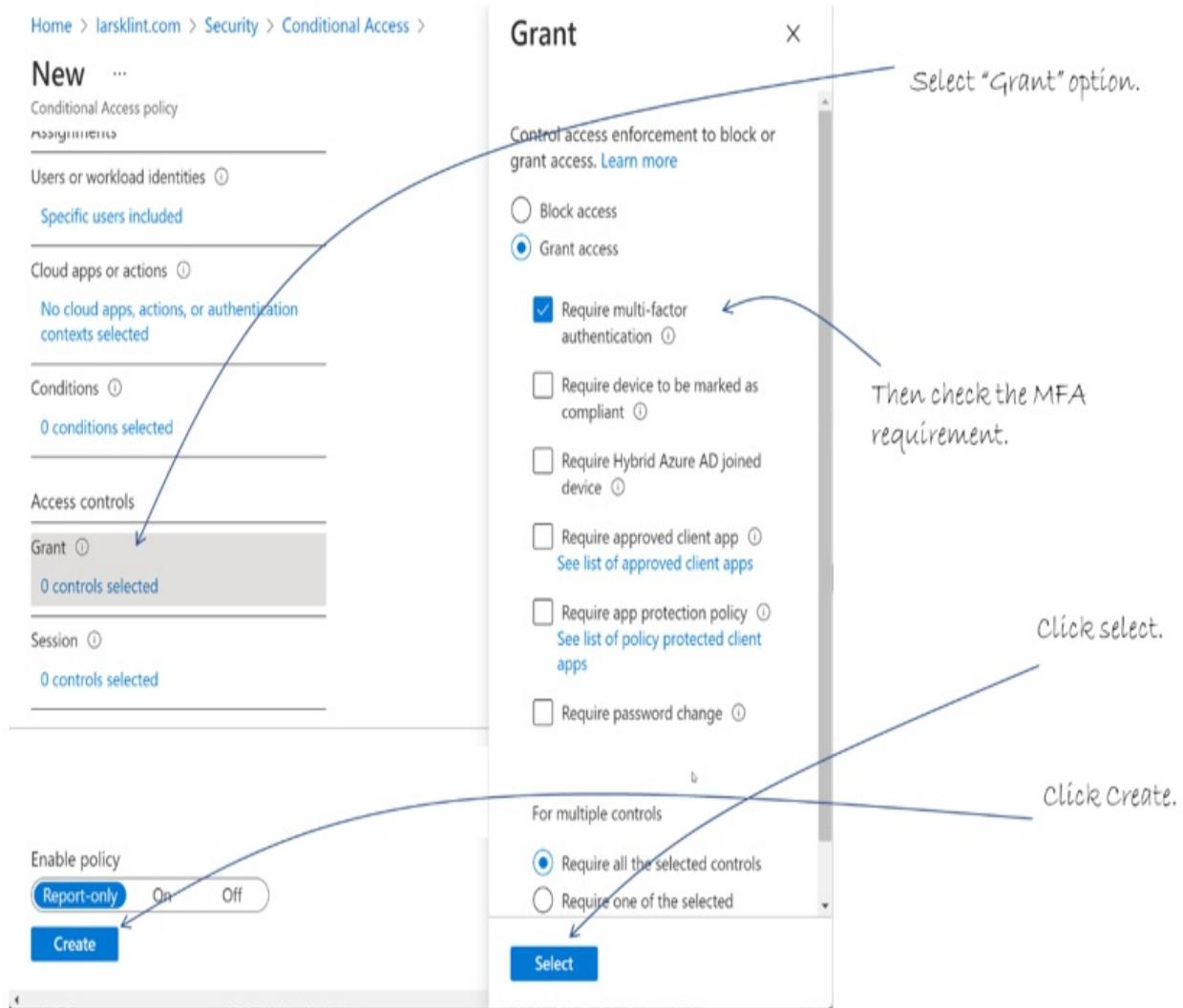
It is important to apply general policies like MFA to as generic a group of users as possible. Ensuring new users are added to this group ensures the MFA policy will be effective and as simple to manage as possible.

**Figure 6-21: Defining users and groups for a conditional access policy.**



There is one more step to do before our policy can be created. We have to add, or enforce, an *access grant*. This grant says only to grant access if a certain condition is met, which in this case is to *require multi-factor authentication* as shown in Figure 6-22. You can see there are other grants that can be selected for users to be authenticated, such as for specific devices or apps. We won't go into the details of those grants but just know there are others to use besides MFA.

**Figure 6-22: Add at least one grant or session, which in this case is to only grant access to MFA authenticated users.**



As set out in Figure 6-22 select the grant, click on *Select*, then *Create*. You are now creating an MFA policy for Azure AAD. Exciting! Although, before we can call it a day and go celebrate with chicken wings and lemonade, there is a bit more to do. So far, we have just created a policy requiring MFA. Nothing is using the policy yet.

A trigger for the policy could be your custom application, a management tool or something else. For Alpaca-Ma-Bags they want to make sure that users logging into the Azure Portal to manage any resources are using MFA, exactly like we saw in the security recommendations in Figure 6-12. To do this, open up the policy we created before, then select *Cloud apps or actions* and search for *Azure management* as shown in Figure 6-23. Check the Microsoft Azure Management app, then click *Select* again.

**Figure 6-23: Adding a target application to the MFA policy.**

The screenshot shows the 'Action MFA' policy configuration page in the Azure portal. The policy name is 'Action MFA'. Under 'Cloud apps or actions', the 'Select apps' option is selected, and a 'Select' dialog box is open. The search bar contains 'azure management', and the result 'Microsoft Azure Management' is selected. The 'Selected items' list contains 'Microsoft Azure Management'. The 'Enable policy' section shows 'Report-only' is selected. A callout points to the 'Select' button in the dialog box with the text 'Click select.' Another callout points to the 'Save' button with the text 'Choose how to enable the access policy.'

Home > larsklint.com > Security > Conditional Access >

## Action MFA

Conditional Access policy

Delete

Control access based on Conditional Access policy to bring signals together, to make decisions, and enforce organizational policies. [Learn more](#)

Name \* Action MFA

Assignments

Users or workload identities ⓘ

Specific users included

Cloud apps or actions ⓘ

Configured

✖ "Select apps" must be configured

Conditions ⓘ

0 conditions selected

Enable policy

Report-only On Off

Save

Select

Cloud apps

Search for "azure management".

MA Microsoft Azure Management  
797f4846-ba00-4fd7-ba43-dac1f8f63013

Selected items

MA Microsoft Azure Management  
797f4846-ba00-4fd7-ba43-dac1f8f63013 Remove

Click select.

Choose how to enable the access policy.

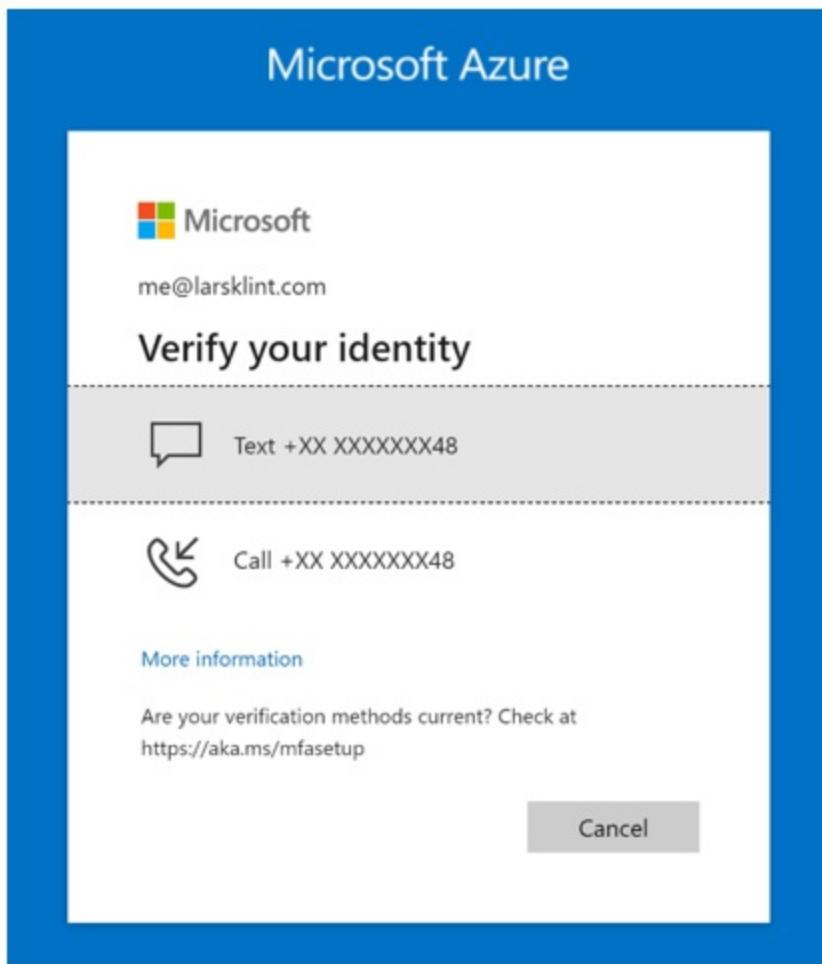
The final part of finishing the policy is to enable it. You can see in Figure 6-23 there are three options for enabling the policy:

- **Report only:** This lets you monitor and evaluate how a policy would impact the users it applies to. You get a report that outlines how your policy works for your users without them noticing any change.
- **On:** Turn the policy on. Camera. Lights. Action.

- **Off:** Turn the policy off. Yeah, you knew that already.

Choose the option that suits you best, but if you want to use it for real, choose “On”. Then log out of the Azure Portal and in again with an account that you set the new conditional policy for, and you will be presented with a MFA screen like Figure 6-24 or asked to set up MFA if you haven’t already.

**Figure 6-24: The MFA check for my account asking for a text message or phone call confirmation.**



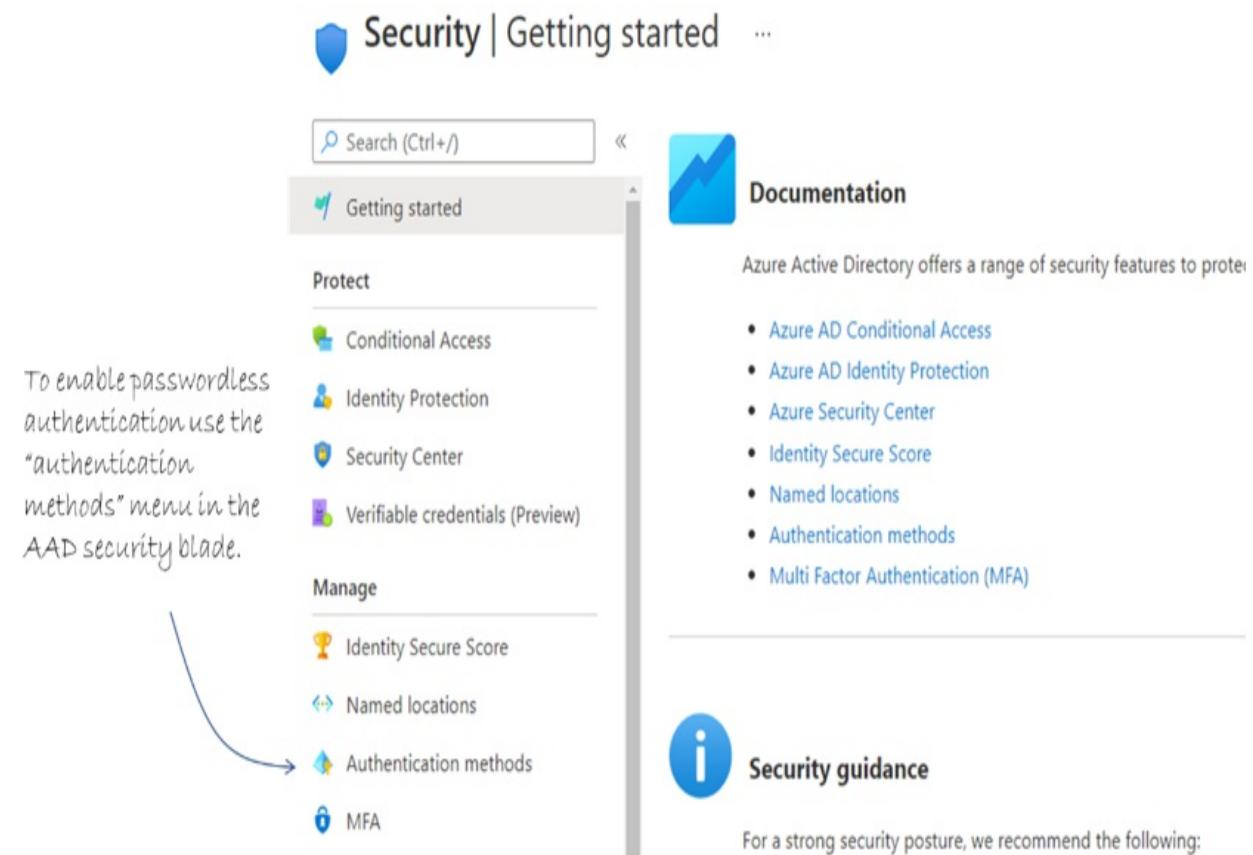
### 6.5.1 Passwordless

An emerging way to authenticate users is to use no password at all. Humans are in general quite rubbish at remembering random words and phrases, which is why we have a single password that we add a number or character to

at the end. You know what I am talking about. Don't deny it.

Azure has recognized this and come up with a variation of the MFA process, which eliminates the need for a password, called passwordless. The name kinda gives it away, but it uses Windows Hello (face recognition, fingerprint or PIN), an authenticator app, or a hardware security key. Once set up, these methods are much more secure than a password that can be leaked, forgotten or reused. To set this up, head back to the security blade of AAD, just like you did before when setting up MFA, and click on the *Authentication methods* menu (Figure 6-25).

**Figure 6-25:** To set up passwordless authentication, use the authentication methods menu option in AAD Security.



If it isn't already, you need to enable the combined security info experience by clicking on that nice blue ribbon as shown in Figure 6-26. Go on, click it.

**Figure 6-26:** To enable users to use passwordless, enable the combined security info.

Authentication methods | Policies ...

larsklin.com - Azure AD Security

Search (Ctrl+ /) « Got feedback?

Manage Policies

>Password protection Registration campaign

Configure your users in the authentication methods policy to enable passwordless authentication register these authentication methods and use them to sign in.

Method	Target
FIDO2 Security Key	

And then choose either *Selected* to only select certain users that can use passwordless authentication or *All* for every user in the AAD tenant as shown in Figure 6-27.

Figure 6-27: Choose “Selected” or “All” for combined security information registration.

User features ...

Save Discard

Users can use preview features for

None Selected All

This will enable users to register and manage their security info for Multi-Factor Authentication and self-service password reset in a single experience. [Learn more](#)

Users can use the combined security information registration experience ⓘ

None Selected All

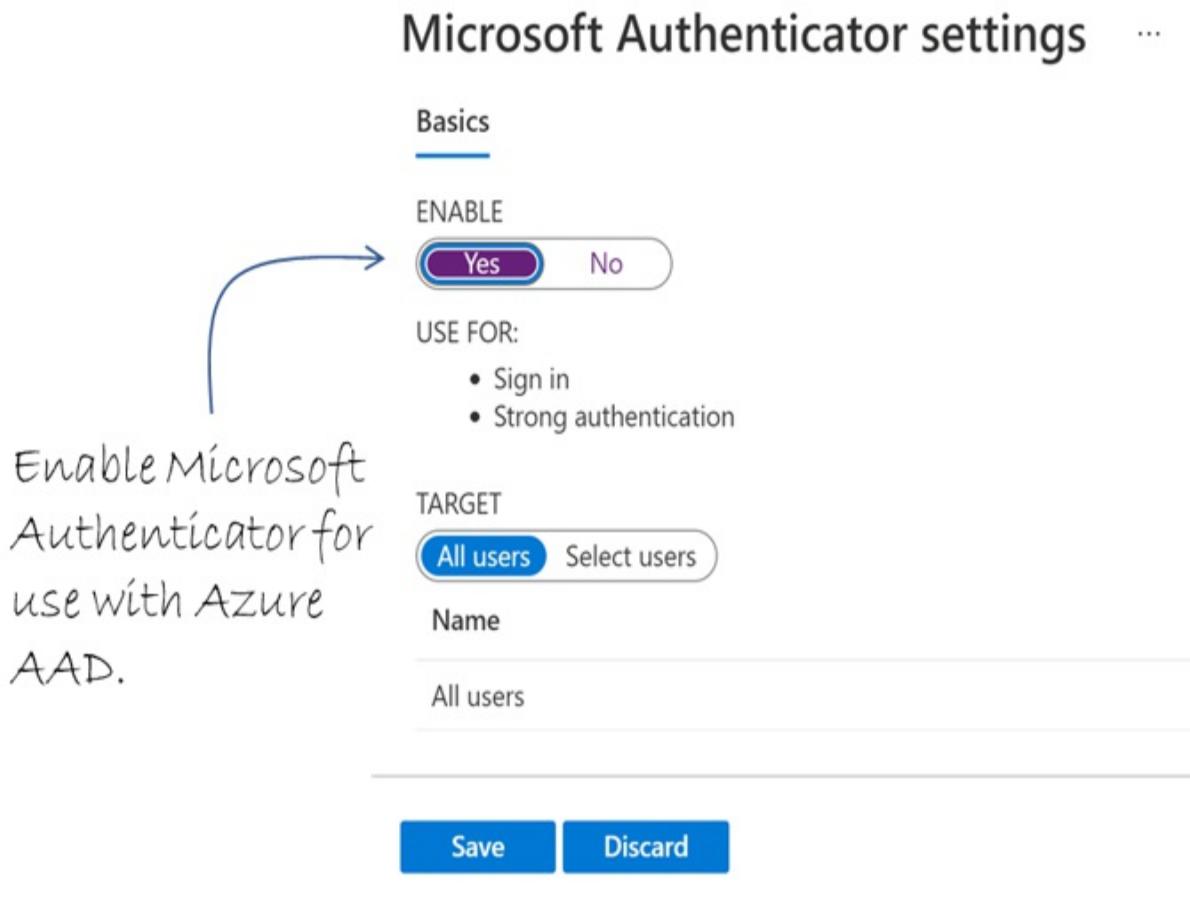
Administrators can access My Staff ⓘ

None Selected All

Then hit the *Save* button. The last step is to enable the passwordless methods

of authentication you want to allow. At a minimum I would recommend enabling Microsoft Authenticator and consider FIDO2 security key. Those are currently the two strongest authentication mechanisms in Azure. In the *Authentication methods* blade, click on Microsoft Authenticator and enable it as shown in Figure 6-28.

**Figure 6-28: Enable Microsoft Authenticator for use with Azure AAD authentication.**



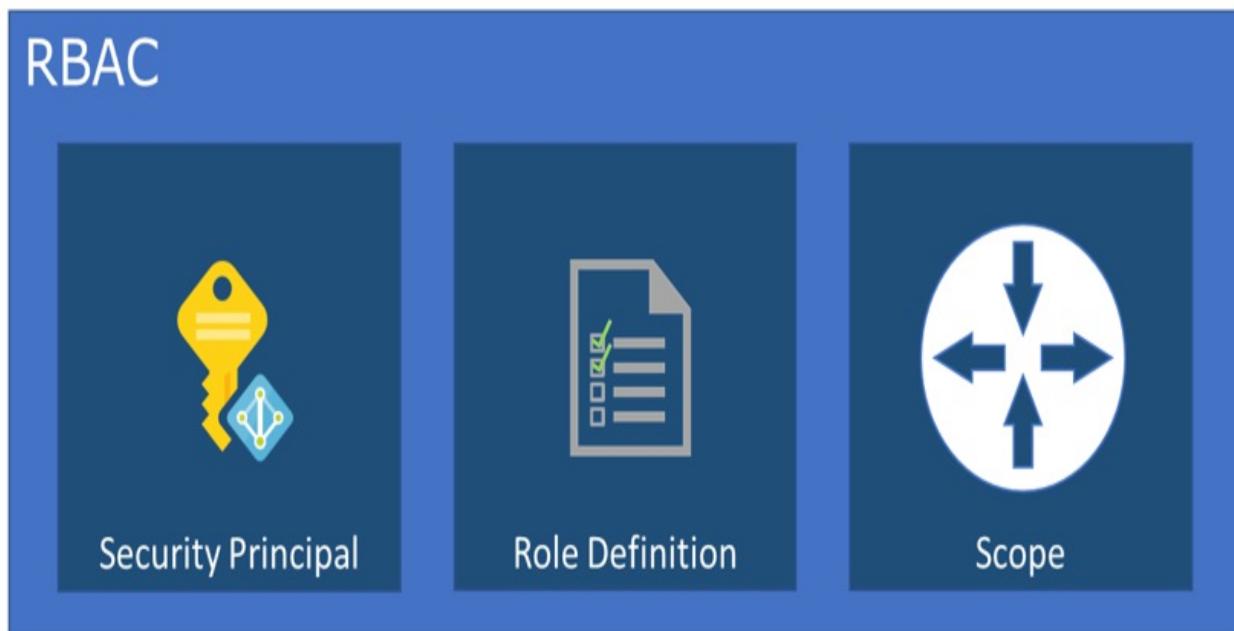
Do the same for any other methods you want to enable. Users will now have to download the Microsoft Authenticator app on their Android or iOS device to use passwordless authentication or use any other methods you have enabled.

Next, let's dive into how the newfound authentication for users lead to authorization to use resources with role-based access control.

## 6.6 Role Based Access Control

Arguably one of the most important ways to manage users and the resources they can access is role-based access control, generally known as RBAC. This is the key feature of Azure when it comes to avoiding losing track of what users can access and keeping your entire security posture to the right on the security continuum. RBAC determines who has access to a resource, what they can do with that resource, and what areas in general they have access to. As shown in Figure 6-29 RBAC works through the combination of three elements: a security principal, a role definition and a scope.

Figure 6-29: RBAC components.



### 6.6.1 Security Principal

A security principal is at its core an object. This object can be defined as a user, a group, an application, or indeed any identity that needs to access a resource on Azure. For example, this could be a specific user, such as Joe from accounting, a group called “sysadmin”, or the new order system Alpaca-Ma-Bags are working on.

You can think of the security principal as the *who* of getting access to Azure, and it must exist in Azure Active Directory.

## 6.6.2 Role definition

The role definition is a collection of permissions. This is the *what* of RBAC and it lists the actions allowed, such as read, write, delete and dance. Okay, maybe not dance, but it would be a nice role.

Anyway, Azure provides a whole bunch of built-in roles that are already defined for you, such as owner, contributor and reader. In fact, there are hundreds of specific built-in roles for all sorts of purposes, including DNS Zone Contributor, Azure Event Hubs Data Owner, and Cognitive Services Metrics Advisor Administrator. As you can tell, they can get incredibly specific, and chances are that you can use a built-in role for defining your RBAC implementation.

If you can't find a role from the Azure provided list, you can also create your custom role definition. Avoid this if you can though, as it quickly becomes difficult to keep track of what the roles do, and thus maintaining the permissions in them, and who are using them.

## 6.6.3 Scope

Finally, the scope is the *where* of the RBAC equation. You define the scope of where the who and the what apply. This is really useful, if you for example want a user to be a database contributor, but only for a certain resource group.

There are four levels of scope you can target, being management group, subscription, resource group, and resource. Using these effectively can dramatically improve how you manage authorization of users.

Together those three parts make up a role assignment, which is what you call the process of attaching permissions to an identity. For example, you could take the user we created in Figure 6-7 and give them access to the *AlpacaRG* resource group. Go to the resource group and then click on the *Access*

*Control (IAM)* menu, then add role assignment as shown in Figure 6-30.

**Figure 6-30: Add a role assignment to the AlpacaRG resource group.**

The screenshot shows the 'Access control (IAM)' blade for the 'AlpacaRG' resource group. At the top, there's a search bar and several navigation links: 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Resource visualizer', 'Events', 'Settings', 'Resource costs', 'Deployments', 'Security', 'Policies', 'Properties', 'Locks', 'Monitoring', 'Insights (preview)', 'Alerts', and 'Metrics'. The 'Access control (IAM)' link is highlighted with a blue arrow. Below the navigation, there are tabs: 'Check access' (which is selected), 'Role assignments', 'Roles', 'Deny assignments', and 'Classic administrators'. The 'Check access' section contains a 'My access' summary, a 'Check access' review, and a search interface. To the right, there are three main sections: 'Grant access to this resource' (with a 'User, group, or service principal' dropdown and a 'Search by name or email address' input), 'View access to this resource' (with a 'View' button), and 'View deny assignments' (with a 'View' button). A large blue arrow points from the 'Add' button in the top navigation bar down to the 'Add role assignment' button in the 'Grant access to this resource' section. Handwritten text above the 'Check access' tab says 'Go to the Access control (IAM) menu on the resource group.' and next to the 'Add role assignment' button says 'Then click add role assignment'.

You might think that a lot of roles would be standard, such as allowing read, but not write, or allowing read and write but not delete. You'd be right! There are a whole lot of standard roles that we use on Azure all the time, which is why Azure provides a bunch of built-in roles you can use. I always recommend using those roles wherever possible, as creating your own brings a lot of challenges. You have to maintain the roles, make sure they don't give too few or too many permissions, not to mention naming them! We all know that naming things is the hardest thing in technology. For this assignment to the Alpaca-Ma-Bags identity, we will use the *Contributor* role as shown in Figure 6-31. This particular role is commonly used where an identity needs

access to fully manage all applicable resources, but not pass that capability on to any other users.

**Figure 6-31: Choosing a built-in role, if possible, is the best choice for RBAC.**

Choose the contributor role, which is a built-in Azure role. This gives the user full access to the resource group, apart from assigning roles to other identities.

Add role assignment ...

Got feedback?

Role Members Review + assign

A role definition is a collection of permissions. You can use the built-in roles or you can create your own custom roles. [Learn more](#) [Use classic experience](#)

Search by role name or description Type : All Category : All

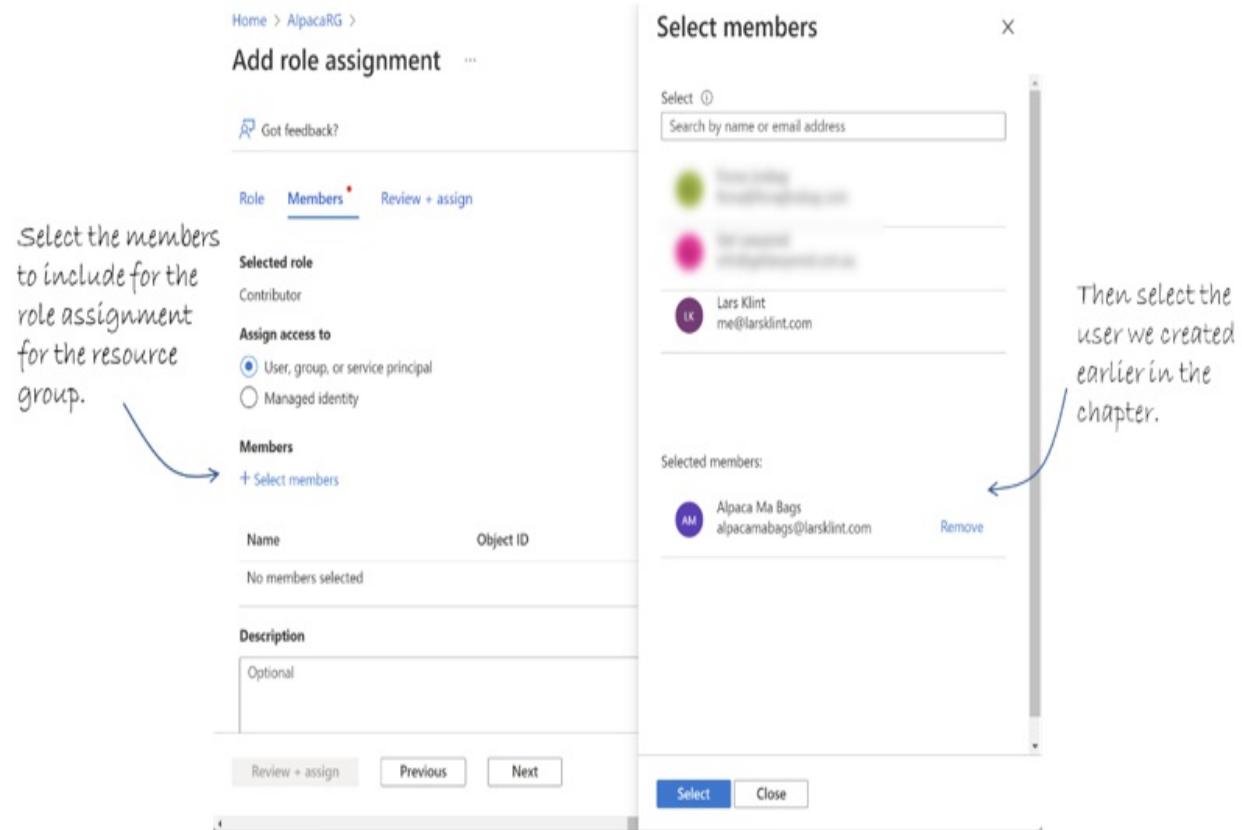
Name ↑↓	Description ↑↓
Owner	Grants full access to manage all resources, including th
Contributor	Grants full access to manage all resources, but does no
Reader	View all resources, but does not allow you to make any
AcrDelete	acr delete
AcrImageSigner	acr image signer
AcrPull	acr pull
AcrPush	acr push
AcrQuarantineReader	acr quarantine data reader
AcrQuarantineWriter	acr quarantine data writer

Review + assign Previous Next

The built-in roles have a pre-defined role definition. You have already chosen the scope when we went to the resource group and selected the *Access Control (IAM)* menu. Finally, we need to select the security principals for the role assignment. Click *Next* and then it is time to choose some members. Figure 6-32 shows the next part of the role assignment wizard where you

choose the identities that are included in the assignment.

**Figure 6-32: Select the members of the role you are assigning.**



Hit *Select* and then *Review + assign*, after which you can save that role assignment to the identity. The *Alpaca Ma Bags* identity now has access to all resources in the *AlpacaRG* resources group and can even create new resources as well. That is the brief introduction to RBAC, as there are a lot of nuances to it, which are out of the scope of the book.

While you now understand the fundamental concepts of security in Azure, there is a lot more about security and specific products and services you should consider, later in the book. Keep on reading.

## 6.7 Summary

- Azure uses 7 layers of defense to provide *defense in depth*. They are

physical, identity, perimeter, network, compute, application, and data.

- Azure Active Directory is where you manage all your users' authentication and authorization by using MFA and RBAC.
- Use the shared responsibility model to understand when and where you need to take responsibility for your security on Azure.
- A zero-trust approach is the best way to ensure only needed access is granted to Azure identities. Assume from the start, no one needs access to anything until proven otherwise.
- Leveraging the security foundation in Azure means you can focus on adding business value and do the fun stuff, as thousands of security engineers work on improving your security posture every day.
- All hardware used by Azure complies with the ISO/IEC 27001 standard for legal, physical, and technical controls involved in an organization's information risk management processes. This is a very stringent standard that ensures Azure complies with numerous regulatory and legal requirements that relate to information security.
- Microsoft Defender for Cloud security recommendations are a to-do list of the security improvements you can (and should) do for a particular service.
- The Microsoft Defender for Cloud secure score directs you to the security improvements that will add the most value to your Azure infrastructure.
- Security alerts are raised when Defender detects a threat to one or more of your cloud resources.
- Multi factor authentication is enabled through Azure Active Directory and then assigned to groups and identities.
- Passwordless authentication is a special type of MFA that increases the security posture by removing passwords the user has to remember, and instead relying on hardware keys or biometrics for authentication.
- RBAC defines identity authorization by using a combination of a security principal, a role definition and a scope.
- RBAC

If you have been wondering, since the start of the chapter, what a ninja crocodile looks like, I have good news. I managed to get a closeup of one. You're welcome.



# 7 Serverless

## This chapter covers:

- Creating a function app and function from Visual Studio Code.
- Creating Logic Apps to integrate with other services.
- Introducing API Management and creating a public facing API.
- Understanding what serverless cloud services are good at.

Serverless is one of those terms that your manager loves and which has come to signify using and implementing cloud computing like the cool kids. There is a certain expectation of efficiency and cost saving if you start describing your solution as serverless. While that can certainly be true when approached correctly, let's get one thing clear right from the start: there is no such thing as server-less. You can't do any kind of computing without a server of some kind, whether that is your laptop, an on-premises datacentre, or cloud computing. What is referred to is that you are using *a server that someone else manages*. You and your project are serverless, but the computation you perform is not.

The best way to look at serverless is like chapter 6 on security in the sense that serverless is a spectrum, rather than a yes/no decision. As shown in Figure 7-1 you are somewhere between a server-based architecture and fully serverless. It is perfectly okay to have a mix of services relying on servers and some relying on serverless services. As always, use the right tool for the job.

**Figure 7-1: The server to serverless spectrum.**

# Server Spectrum

Servers                      Less servers                      Serverless

With that clear, welcome to a chapter that can indeed provide real benefits for your Azure projects when it comes to development time, performance, and costs. This chapter delves into some of the most popular serverless products on Azure, which are quick to get started with, simple to use, but only really shine when used together with each other or with other services on Azure. Serverless is a way to indicate what types of services you use, and what approach to your application architecture you choose. Let's get into it.

## 7.1 Alpaca-Ma-Bags web shop and NFTs

Alpaca-Ma-Bags wants to improve their online shop and public image. They want to move their business into the modern online era and offer non-fungible tokens, or NFTs.

### Non-Fungible tokens

Non-fungible tokens are used to determine ownership of a specific digital asset. They are all the rage at time of writing, and if you are a cool internet citizen you want them. All of them. Read more here:

<https://ethereum.org/en/nft/>

Also, this book does not presume to know anything about NFT trading, valuation, or any other aspect of this blockchain enabled digital trading world. I am not suggesting you invest, acquire or do anything with these assets. It is merely in this book as part of a serverless scenario.

These unique digital assets can make their customers feel special and loved,

but Alpaca-Ma-Bags also want their partners to get some benefit from them, and to have access to their customers' NFTs. Up until now the web shop data flow has been a record inserted into a database every time an order is made of some of their products, such as the popular alpaca hairstyling gel. Now, when an NFT is created by a blockchain process, a reference to the NFT and the NFT image itself will be stored too.

**NOTE:**

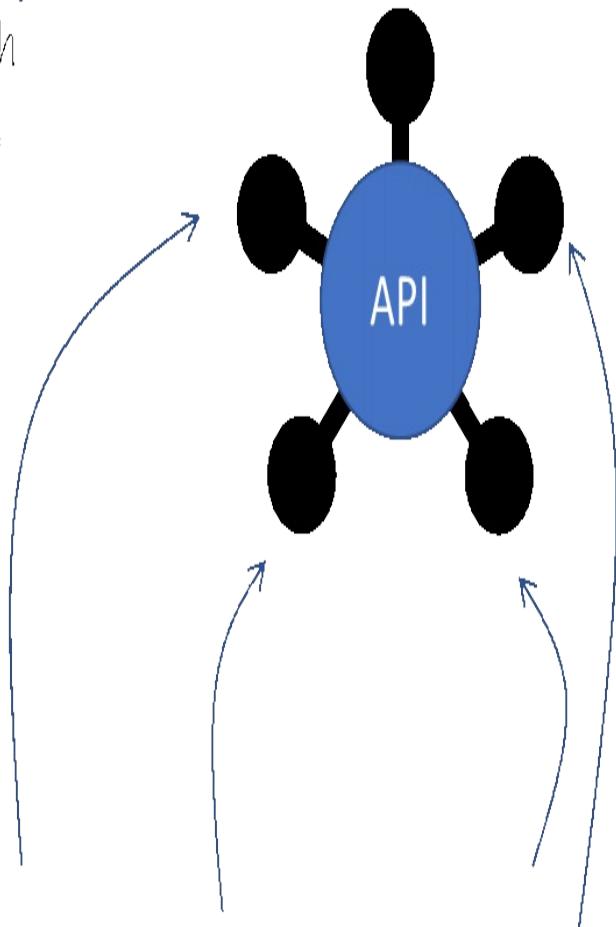
Blockchain is a cryptographic chain of records, where each record's integrity is calculated based on the records that came before it. We won't cover it in this book though.

The images will then be accessible through a public-facing API, which partners can be granted access to. The API architecture and endpoints will be created in a serverless setup which makes it scalable and low cost. There was that word again: Endpoints. I have mentioned it a few times in the book, but now is the time to understand exactly what that word encapsulates. An API is a collection of functions that a user of the API can call, or access. For example, an API might have a function that calculates the power of one number to another, such as  $2^4$ . This function requires two inputs, being the two numbers. For an API, this is exposed through an endpoint, which is most often called using HTTP via a unique URL as shown in Figure 7-2. In short, an endpoint is a web address that can be accessed over a network, usually the Internet, which you can pass a number of arguments to, and get a result.

**Figure 7-2: An API has endpoints, which are often HTTP URLs that can be accessed over the Internet.**

This is an endpoint that can be accessed over an HTTP URL such as [api.alpacamabags.com/xpowy](http://api.alpacamabags.com/xpowy)

xpowy(x,y)



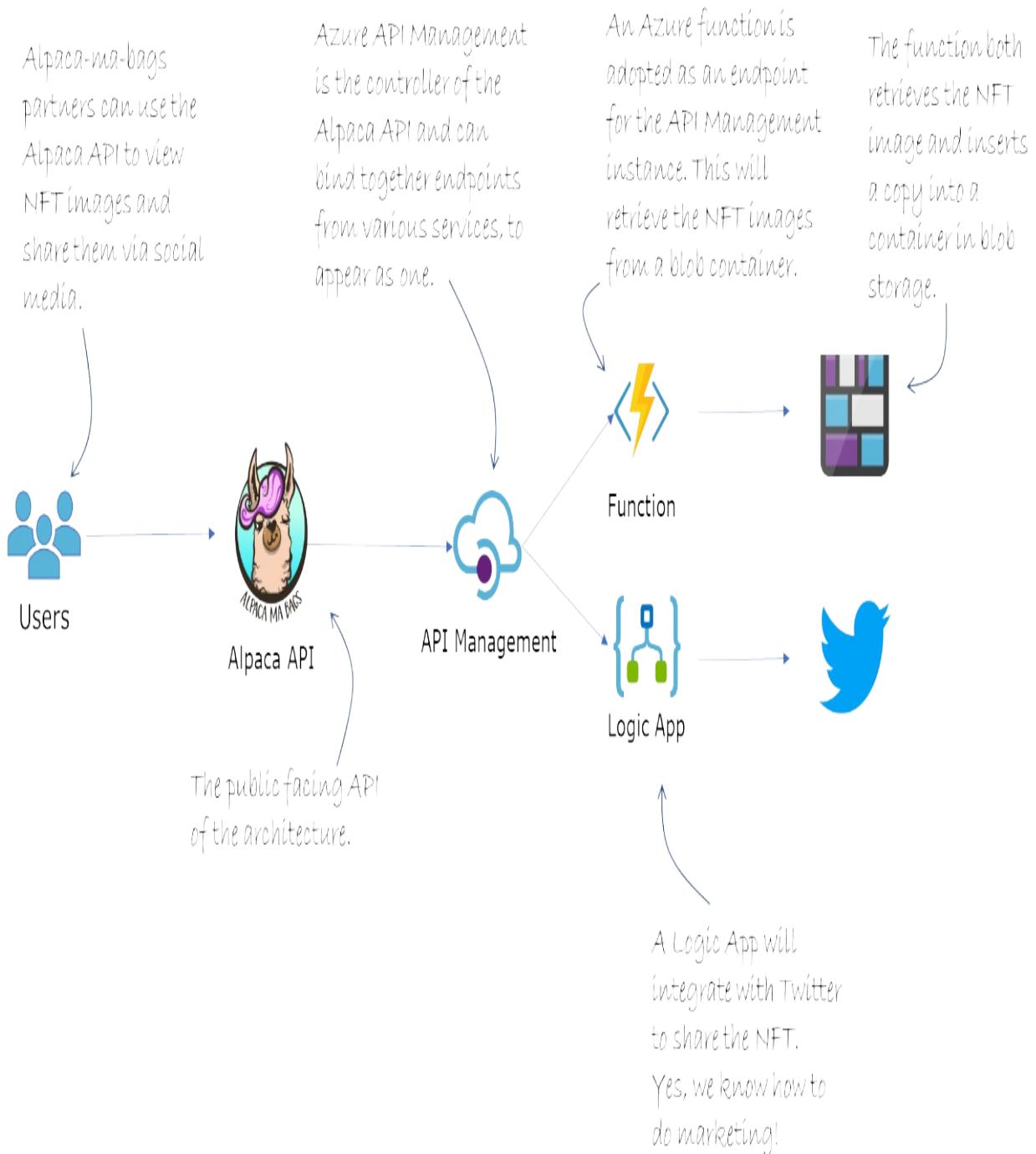
APIs can have many endpoints that all have their unique URLs. You can have different authorization levels for each as well to allow only certain types of users to use them.

While the current flow is adequate for the shop as it operates right now, Alpaca-Ma-Bags wants to make sure the shop can grow and integrate with other services of the future. For example, if a customer creates an order, it

could trigger a marketing email workflow to recommend other related products. You know you want more marketing emails about alpacas. For this reason, they are planning on expanding their cloud footprint with serverless additions. The parts we will need for the initial architecture are shown in Figure 7-3 and include:

- **API Management:** This is the service that binds all the API endpoints, of which we will have two initially.
- **Azure Function:** The function will retrieve the NFT from the blob container.
- **Logic app:** Post a copy of the image to Twitter. Because, why not?
- **Blob storage:** Blob storage is where the images are stored. We will also send output from the Azure Function to here. Blob containers are within a storage account, which we went over in chapter 5.

**Figure 7-3: The serverless architecture we will use. Look at that fancy alpaca.**



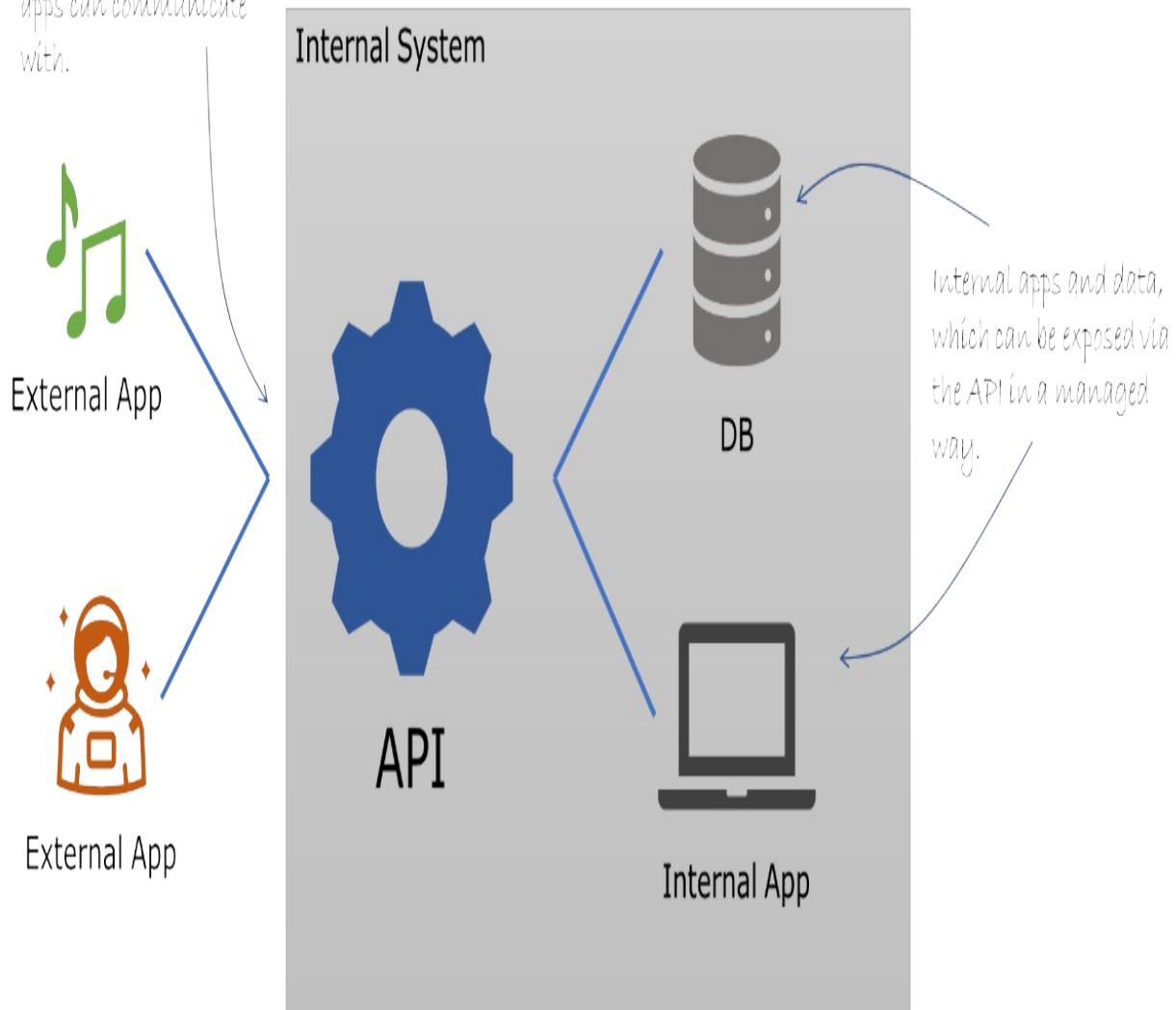
Of course, this scenario is a tiny bit contrived and made up, but it will give us a chance to learn about three major serverless services in Azure: API Management, Functions, and Logic Apps. And if you can accept alpaca hair products, I am sure we can agree on using this business venture into NFTs too. Right?

### 7.1.1 What is an API?

I have mentioned “API” quite a few times already, so it is probably a good time to talk about what that means, at least in the context of our project for Alpaca-Ma-Bags. API is short for application programming interface and is one of the most important concepts of modern software applications. As shown in Figure 7-4 an API is the connection that allows two or more applications to communicate with each other. An API has *endpoints* which are the entry points to communicate with the API. When an application connects to an endpoint, any measures of protection needed, such as authentication and authorization, can be implemented.

**Figure 7-4: Two external apps are communicating with the API to access internal data.**

The public facing API endpoint, which external apps can communicate with.



You might be thinking that you can do all this with a standard website, or indeed your tried and tested desktop application. The difference between those scenarios and an API, is that there is no user interface. An API is exclusively for pieces of software to connect and communicate with each other in a managed way.

As you might have noticed from Figure 7-4 one purpose of APIs is to hide the internal details of how a system works, exposing only those parts a programmer will find useful. This has the added benefit of future changes to the internal system doesn't necessarily mean the API endpoint and usage

have to change. APIs are what binds together a LOT of software solutions. And with that, let's start building our own serverless API.

## 7.2 Azure Functions

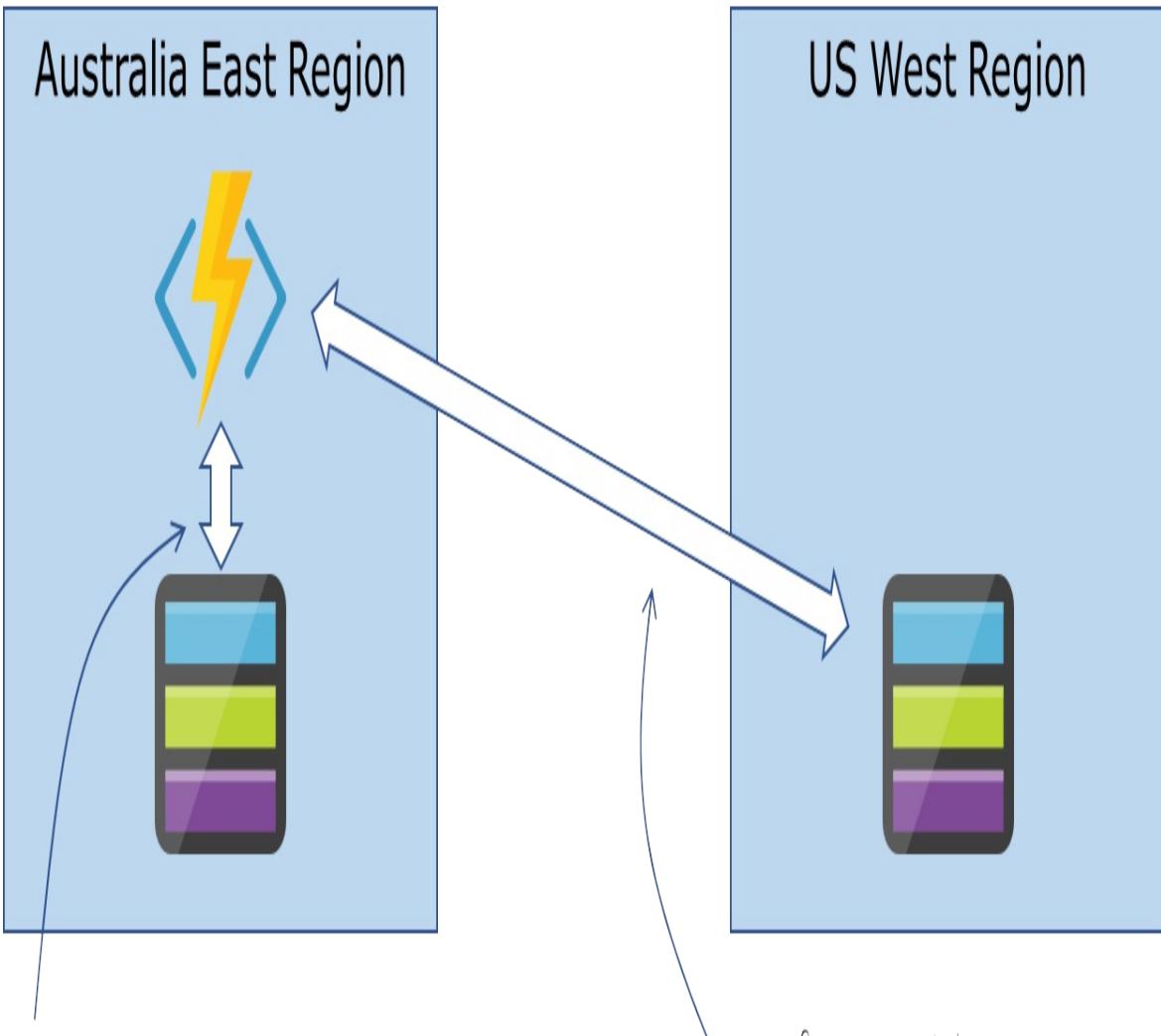
You have already had your first exposure to both serverless and Azure Function in chapter 2, when you built the image resizing application. It's time to dig deeper into these magical constructs of Azure and cloud computing. I say magical, because at times it genuinely feels that way. In fact, read through this part on functions and let's see if you agree. Deal?

As you learnt in chapter 2, when we built the image resizing app, functions live inside a function app. This function app can be loosely translated to the VM that will run your function, which is why *serverless* simply means *someone else's computer*. Let's create a function app, but this time we will use the Azure CLI instead of the Portal. As mentioned before in chapter 3 of this book when creating VMs, the CLI is a common tool for creating and managing Azure resources, once you start working with them frequently. It is just faster and more convenient.

```
C:/> az functionapp create --name ActionFunctionApp --storage-acc
```

Let's go through that command, as there are a few parameters to understand. The `functionapp` keyword indicates that this whole command has to do with a Function App. The `create` after it is obvious of course, and is for creating our function app. The `name` parameter we have seen multiple times by now and is the name of the function app. You can call it whatever you want or use the value given above. You might remember that a function app needs a storage account to have somewhere to put meta data about the functions, the code to run and other bits and pieces. The `storage-account` parameter is for this, and you should give the name of a storage account in the same resource group as the function app. It is also recommended you use a storage account in the same region as the function app, which makes sense. The data needs to travel from the function app to the storage account and back, so the shortest distance to do so is a big deal, as shown in figure 7-5.

**Figure 7-5: Use a storage account in the same region as the function app to minimize latency when running your function.**



The time to get to a storage account in the same region as the function app is minimized, because the physical distance is the shortest.

If you could use a storage account in a different region, the distance would be much greater, and the function would run much slower as it accesses data.

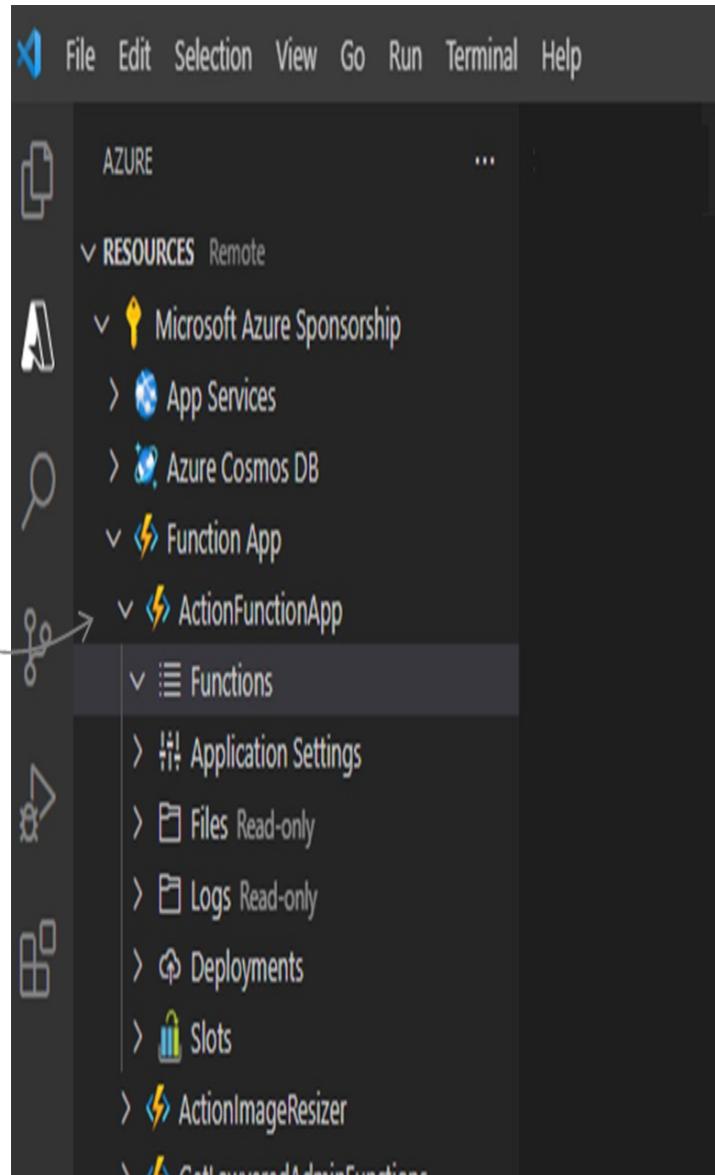
Because we are creating a function app of type consumption, the parameter consumption-plan-location is needed to determine which region to host the function app in. A whole lot more on hosting options and constraints is coming up in a little bit. Finally, the functions-version refers to the major version of the function app, which at the time of writing goes up to 4. Fire that command off in the CLI and you have a shiny new function app. You should see a ton of JSON output once the command succeeds. However, the function app doesn't do anything on its own because we need to have one or more functions inside it. How many functions you can have in a function app depends on the type of function app and the scenario you are using it in.

Before we get into the details of what makes up a function, let's create one first. In chapter 2 this happened using the Azure Portal, but that isn't the most common way of creating a function. We do that using a much more efficient tool: Visual Studio Code. If you aren't familiar with VS Code, go check out the appendices in this book to get started.

Once you have the Azure extension installed in VS Code and logged in, you should be able to find the function app we created a minute ago shown in Figure 7-6. In fact, you can also see the function app we created in chapter 2.

**Figure 7-6: The Azure function we created is shown in VS Code.**

The Azure function  
can be shown here  
under the Resources ->  
Function App section of  
the Azure extension in  
VS Code.



Open the terminal in VS Code (shortcut `Ctrl+Shift+'`), navigate to the folder you want to store the functions files in locally using the command line, then run this command to create the function project in C#.

```
func init AlpacaNFTProject --dotnet
```

This will create a folder called `AlpacaNFTProject` inside the folder you navigated to. This new folder will have various files for the dotnet project as shown in Figure 7-7.

**Figure 7-7: Files created by the `func init` command.**

Mode	LastWriteTime	Length	Name
----	-----	-----	
d----	19/07/2022 9:17 PM	4626	.vscode
-a----	19/07/2022 9:17 PM	629	.gitignore
-a----	19/07/2022 9:17 PM	227	AlpacaNFTProject.csproj
-a----	19/07/2022 9:17 PM	163	host.json
-a----	19/07/2022 9:17 PM		local.settings.json

Files created for the function app project.  
There is not yet an actual function file in here though.



These files are standard for any .NET project and are described in Table 7-1.

**Table 7-1: Overview of the files created for a new Azure function.**

File	Purpose
.vscode	Folder for workspace specific files for running the function and debugging it.
.gitignore	Definition of the rules for files and folders not to include for any source control actions using git.
AlpacaNFTProject.csproj	The project definition for the Azure function.
host.json	Contains configuration options that affect all functions in a function app instance, such as logging and extensions.
local.settings.json	This is where you define the settings for your local development environment. These aren't deployed to production or Azure.

There isn't an actual file for the function yet. What we created first was the initialization of an Azure function project. Next, we add our first function. Well, second function if you count the one in chapter 2. It will be our first function created from within VS Code though. Right, here we go.

```
func new --name ProcessNFT --template "HTTP trigger" --authlevel
```

While this command is short and relatively simple, there is a lot going on. First, the new parameter indicates we are creating a new function. Nothing surprising about that, but then we get to template, which is a whole story in itself. The template for the function describes which *trigger* the function has. We will dive into triggers in just a second. To see a list of available triggers for your current system use the command

```
func templates list
```

We are using the HTTP trigger template for the function, which triggers the function whenever its HTTP endpoint is called. Finally, the authlevel parameter can have one of three values, anonymous, function, or admin. This determines what sort of authentication is required to access the function, as outlined in Table 7-2.

**Table 7-2: Overview of Azure function authentication types.**

Authentication Level	Description
Anonymous	No authentication is required. Any valid HTTP request passes.
Function	Key based authorization. The key can be used for only this function.

Admin

Key based authorization. The key can be used for all functions in the function app.

We now have a function in VS Code, but it is only on our local machine for now. That is okay. We will get into Azure soon enough. However, as we now have a function to look at, let's go through more of the details and features of functions. And we start with triggers.

## 7.2.1 Triggers

Triggers are what cause a function to run. A trigger defines how a function is invoked and a function must have exactly one trigger. If there wasn't a trigger, a function would never run, as it wouldn't know when to do so. In the function we just created, we used the *HTTP trigger*, which lets you invoke the function with a call to a web address. A *blob storage trigger* would invoke the function when a new or updated blob in a container is detected. There are many other triggers as well, such as a queue trigger with the queue item as data. These additional types of triggers are outside the scope of this book though.

Triggers have associated data, which is often provided as the payload of the function. If you look at the signature for the *Run* method in our function:

```
public static async Task<IActionResult> Run(  
    [HttpTrigger(AuthorizationLevel.Anonymous, "get", "po  
    ILogger log)
```

The data coming from the trigger is the *req* parameter of type *HttpRequest*. This data object will have data that is common to an HTTP request, such as the host and referrer of the request.

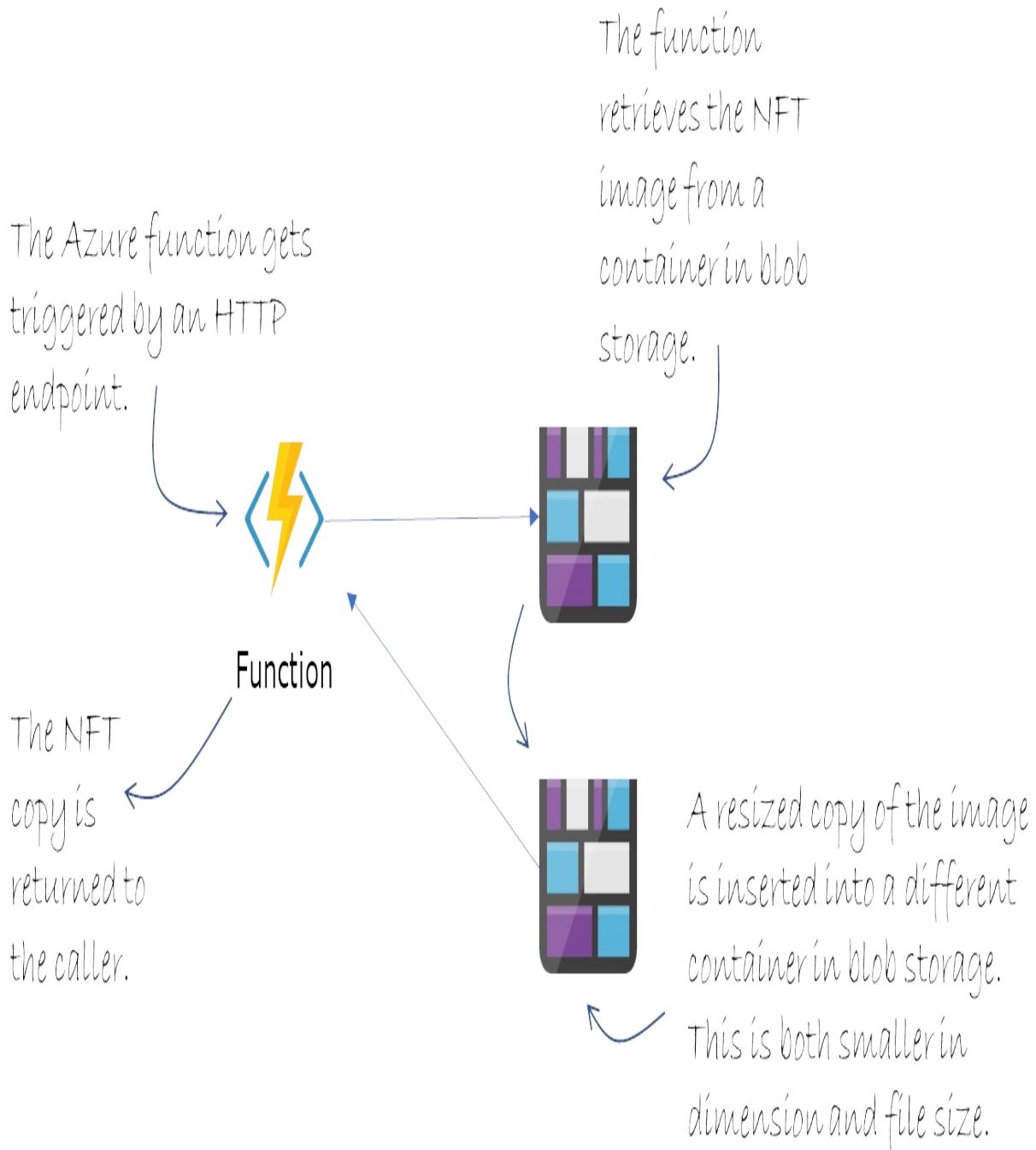
## 7.2.2 Bindings

You know how to start a function using a trigger, but how do you interact with other resources that the function needs to do its work? That is where bindings come in. Bindings let us connect other resources to the function using input and output parameters. The data from the parameters is then made

available to the function from the bound service.

For our function, let's add a binding to the input as well as the output. Eventually, this function is being connected to our Alpaca API (Apipaca? 😊), which will in turn trigger the function. The function will retrieve the NFT image using a binding to blob storage, and then also create a smaller version of the image, store it in blob storage and then return it to the API caller. This smaller image is both smaller in dimensions as well as file size, to make it faster to retrieve it for previews or thumbnails. I created Figure 7-8 to make that a bit clearer.

**Figure 7-8: The flow we are creating for the Azure function.**



As you can see in Figure 7-8 we need to extract the image from a blob container, then also create a copy of the NFT image, which is inserted into the second blob container. We are going to do all this in the function, but let's start with a look at the first part of the code of the function as it stands from when we created it.

```
[FunctionName("ProcessNFT")]
public static async Task<IActionResult> Run(
    [HttpTrigger(AuthorizationLevel.Anonymous, "get", "post")]
    ILogger log)
{
```

The function's name is defined in the *decorator* `FunctionName` at the top of the function, and the trigger is another decorator `HttpTrigger`. In the `HttpTrigger` definition, you can see the authorization level has been injected, along with the *get* and *post* allowed HTTP methods. For the bindings, we will add them inside the function definition in the same way. The bindings will connect directly to the blob storage and both retrieve and insert the images as they are resized. The body of the function then uses these bindings to get the image and add it to an object. The object is resized and saved in the blob container of the output binding.

Update the function code to the following:

```
[FunctionName("ProcessNFT")]
public static void Run(
    [HttpTrigger(AuthorizationLevel.Anonymous, "get", "post", Route = "nfts/{nftid}")]
    [Blob("originalnfts/{nftid}", FileAccess.Read)] Stream nftImage,
    [Blob("copynfts/{nftid}", FileAccess.Write)] Stream nftSmall)
{
    IImageFormat format;

    using (Image<Rgba32> input = Image.Load<Rgba32>(nftImage, out format))
    {
        input.Mutate(x => x.Resize(320, 200));      #E
        input.Save(nftSmall, format);      #F
    }
}
```

There are two more things we need to do, to make this function work. First, we need to add a reference to the `SixLabors` image manipulation library, which we also used in chapter 2. Using Visual Studio Code, open the project file, such as `AlpacaNFTPProject.csproj`, and add the following line with the other package references.

```
<PackageReference Include="SixLabors.ImageSharp" Version="2.1.3"
```

You might have to use a different version, but 2.1.3 is current at the time of

writing. You then need to add the using references to the top of your function file to let your function know how to manipulate our NFT image.

```
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.PixelFormats;
using SixLabors.ImageSharp.Processing;
using SixLabors.ImageSharp.Formats;
```

**Note:**

If you are getting errors for the Blob bindings, it might be because you have to install the Storage extension in VS Code. Run the command `dotnet add package Microsoft.Azure.WebJobs.Extensions.Storage` in the terminal windows to add the reference library.

The second thing is to add some images to the blob container defined in the input binding `originalnfts/{nftid}`. You can add whatever you like, but in this case, I have used the local Azurite open-source emulator, which will allow you to run everything locally. If you haven't used the Azurite open-source emulator before, check out the appendices. Once you have one or more images in your blob container, press F5 in VS Code to run the function. This will build the code for the function, start up a small local web server, and host your new shiny function ready for use. And use it we will.

In the Terminal window, you'll see a line like the following at the very end.

**Functions :**

```
ProcessNFT: [GET,POST] http://localhost:5004/api/nft/{nft
```

This shows the functions that have been built and are now running on your local host. In this case the *ProcessNFT* function is being hosted on localhost port 5004. Your port might be different. One way to call the function is to open your web browser and enter the URL specified above but replacing the `{nftid}` with the name of the blob you want to process.

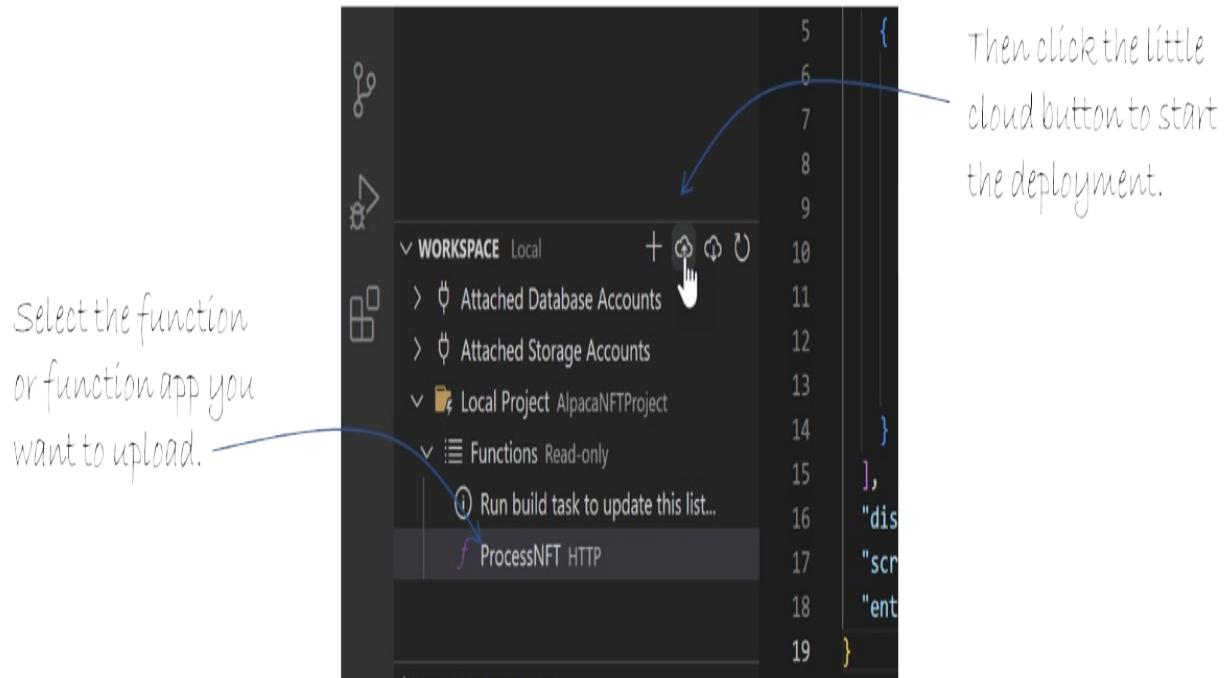
`http://localhost:5004/api/nft/nft-1-2022.png`

Once the function has run, a second image has been generated and inserted into the path in the output binding, such as `copynfts/{nftid}`. You may

have noticed that the input and output bindings are very similar. A binding is a binding in general, and it depends on how you use it, which determines if it is input or output. In this example, the input binding is set to *FileAccess.Read* and the output is *FileAccess.Write*, which makes sure I don't accidentally change the original NFT.

The final question is how do we get the local function deployed to Azure proper? In VS Code, it is quite simple. Make sure you are logged in to Azure in the IDE, then choose your function app and click the deploy icon as shown in Figure 7-9.

**Figure 7-9: Deploying a local function app to Azure in VS Code.**



Alright, now that we have a function app, it is a good time to dive a bit more into hosting your code.

### 7.2.3 Hosting

While I love to look at the code and learn how to create the function, let's for a minute pause to talk about the hosting of the function. You might think "wait, hosting? I thought this was serverless?" As we discussed at the

beginning of the chapter, serverless just means it runs on *someone else's* server. We need a host to run the function. While you only do this once when setting it up, it can make a big difference how you approach it. There are three different ways to host your function on Azure: consumption, premium, and on an app service plan. The pricing of each model is very different and are outlined in Table 7-3.

**Table 7-3: Azure Function pricing for various hosting options.**

Hosting option	Cost model	Description
Consumption	Per execution	The traditional serverless model. You get 1,000,000 free executions per month, then pay per million executions.
Premium	Duration of vCPU and memory	Pay per hour you use a virtual CPU combined with the GBs of memory used per hour.
App Service Plan	Cost of the app service plan	You pay a fixed amount for the App Service Plan, no matter what it runs. If you already have an active one, you could effectively run your functions for free.

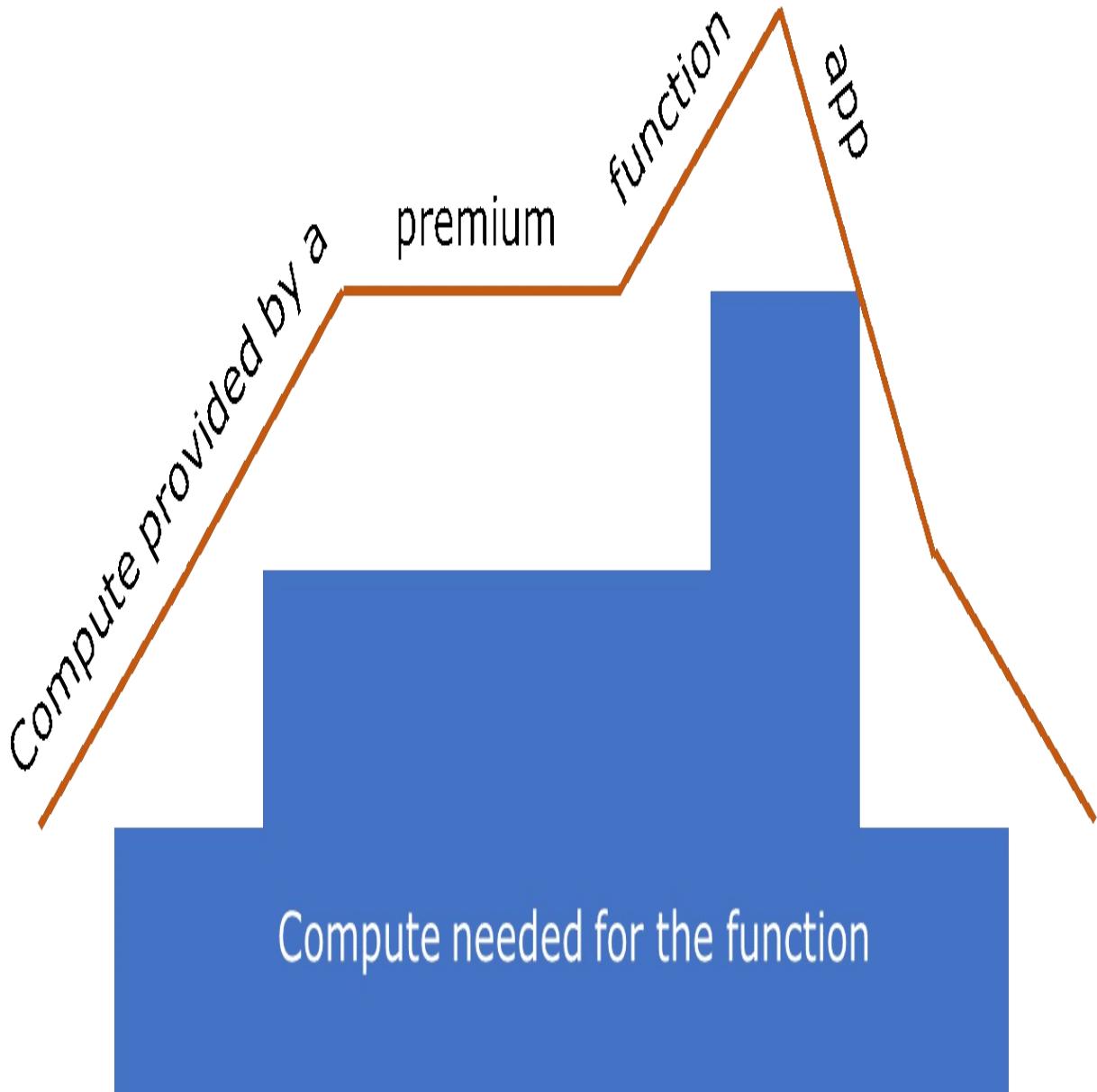
While pricing is certainly a significant factor in how you decide the right hosting for your Azure function, there are other factors that should be considered carefully too.

Let's start with the consumption option. This is the default option, and it will suit a large part of your function needs. Whenever your function is triggered, the consumption hosting will create a VM to run your function, load the code,

and execute it. If there is high demand for the function, the consumption plan will automatically scale up to meet that demand by creating VMs to meet it.

The premium plan has one main feature: pre-warmed VM instances. We're not talking about cooking here of course, but the reference to having at least one VM running (or spun up) and ready to trigger the function at any time. On the consumption plan, the VM running your function will be recycled at some point. This might be when the resource is needed or when your function is not being triggered for a period. When your function is triggered again, the VM has to be created, started and your code loaded onto it. This takes time and is referred to as "cold start". Sometimes, the few seconds this can take can be critical in your workflow, and this cold start has to be avoided. That is what pre-warmed instances are there to prevent. With the premium hosting you never have a cold start, as at least one more VM than currently needed to run your function workload is always ready as shown in Figure 7-10, which is probably the best designed figure in this whole book. Right?

**Figure 7-10: The compute resources provided in a premium function app is always more than needed to prevent cold starts.**



There are other benefits of the premium hosting as well. The timeout for the function is increased from 5 to 30 minutes by default, you get more powerful VMs for computing, and you can integrate the function app with your virtual networks. The last point deserves a bit more information though. Remember back in the chapter on networking that everything on Azure is connected to a network? If so, you might wonder why this is a *premium* feature for functions. Virtual network integration allows your function app to access resources inside a specific virtual network, which will provide the benefits of

gateways, subnet access, and peering.

Finally, you can choose a new or existing app service plan. I briefly mentioned app service plans in chapter 2 but let me recap. They are measured in compute units and are really VMs that you provision to run your web apps, being websites, APIs, docker containers and more. A very cool feature of app service plans is that you can host your Azure functions apps on them too. Why is this cool? First, you are already paying for the app service plan, so the functions are essentially “free”, as you don’t pay extra to host a function app on it as well. Second, you have some insight into the health and current environment of the VM your function runs on. This can sometimes be the difference between fixing a problem and sitting in a corner rocking back and forth.

To summarise the hosting options for Azure functions, it can make a big difference which you choose. As the comparison in Table 7-4 shows, you need to consider your requirements for the Azure function you are creating.

**Table 7-4: Comparing Azure function hosting options.**

	Consumption	Premium	App Service Plan
<b>Price</b>	Per execution	Per VCPU and memory usage	As per the app service plan cost
<b>Scaling</b>	Automatic	Automatic	As per service plan
<b>Cold starts</b>	It happens	No chance!	Can be avoided
<b>Docker</b>	Not available	Go for it	Yep, sure

<b>Networking</b>	No such luck	Most certainly	Yes, as per app service plan features
-------------------	--------------	----------------	---------------------------------------

Coming back to the question at the start of the section. Do you think functions are kind of magic? You have almost unlimited compute power with only a few lines of code, and you can integrate with other Azure services to create a complex workflow with very little code. I love functions. Let's move on to one of my other top 3 serverless services.

## 7.3 Logic Apps

One of my favourite services on Azure has to be Logic Apps. Think of Logic Apps as the glue that can bind Azure services to each other, as well as to third party services. Let's go through what they do by creating the one we need for Alpaca-Ma-Bags to send tweets of the NFT images.

While you can create a Logic App through the Azure CLI, as well as define the triggers, actions, integrations and flows through JSON notation, we will use the Azure Portal in this case. It is easier to visualise and follow that way.

Open up the Azure portal and go to the Logic Apps section, where you can create a new instance. Click on *Add*, then go through the standard Azure wizard to create the Logic App by providing your subscription, resource group, name, and region. Those properties are by now a known part of your Azure services, as they need to be defined for everything. Let's dive a bit deeper into the *Plan type* though. You have two options, being *Standard* and *Consumption* as shown in Figure 7-11.

**Figure 7-11: The logic app plan types.**

A Standard plan for logic apps is essentially running on a dedicated VM you pay for, similar to an app service plan.

## Plan

The plan type you choose dictates how your app scales, what features are enabled, and how it is priced. [Learn more](#)

### Plan type \*

**Standard:** Best for enterprise-level, serverless applications, with event-based scaling and networking isolation.

**Consumption:** Best for entry-level. Pay only as much as your workflow runs.

Consumption means serverless.  
You pay only for what you use.

The Standard plan runs on a dedicated resource, similar to a VM, which you then pay for. It is very similar to how you can run an Azure function on an app service plan, which we went through earlier in this chapter. You have control over the resource, and you pay a fixed amount for that resource. However, you pay that fixed amount no matter whether you use it or not. A consumption plan on the other hand is *serverless* in the sense that you only pay for the executions you use. You even get a certain number of executions free every month, which can get you going cheaply during development. For this example, we'll choose the consumption plan. After all, this chapter is about serverless. Finally, create the Logic App, and go to the resource, which should look similar to Figure 7-12.

**Figure 7-12: The Logic App landing page in the Azure Portal.**

Microsoft Azure

Search resources, services, and docs (G+)

LARSKLINT.COM (LARSKLINT.COM)

Home > alpacanftlogicapp >

## Logic Apps Designer

Introducing Azure Logic Apps

Azure Logic Apps

Watch later Share

Watch on YouTube

Building integration solutions is easier than ever

Logic Apps brings speed and scalability into the enterprise integration space. The ease of use of the designer, variety of available triggers and actions, and powerful management tools make centralizing your APIs simpler than ever. As businesses move towards digitalization, Logic Apps allows you to connect legacy and cutting-edge systems together.

- Create business processes and workflows visually
- Integrate with SaaS and enterprise applications
- Unlock value from on-premises and cloud applications

Start with a common trigger

Pick from one of the most commonly used triggers, then orchestrate any number of actions using the rich collection of connectors

	When a message is received in a Service Bus queue		When a HTTP request is received		When a new tweet is posted		When an Event Grid resource event occurs
	Recurrence		When a new email is received		When a new file is added		When a file is added

What you have created now, doesn't do anything yet. Just like an Azure function app is needed to host the functions you create, a logic app is needed to host the logic flow of the logic app.

### 7.3.1 Connectors

Connectors in logic apps are the abstraction for using hundreds of services to help you access data, events, and resources in other apps, services, systems, protocols, and platforms. It is a prebuilt part of the logic app ecosystem which Azure provides to you. Often a connector will ask you for credentials to connect your data to the logic app, but then you can use it as a native step in your workflow. Connectors comes in both triggers and actions, which are the next two sections.

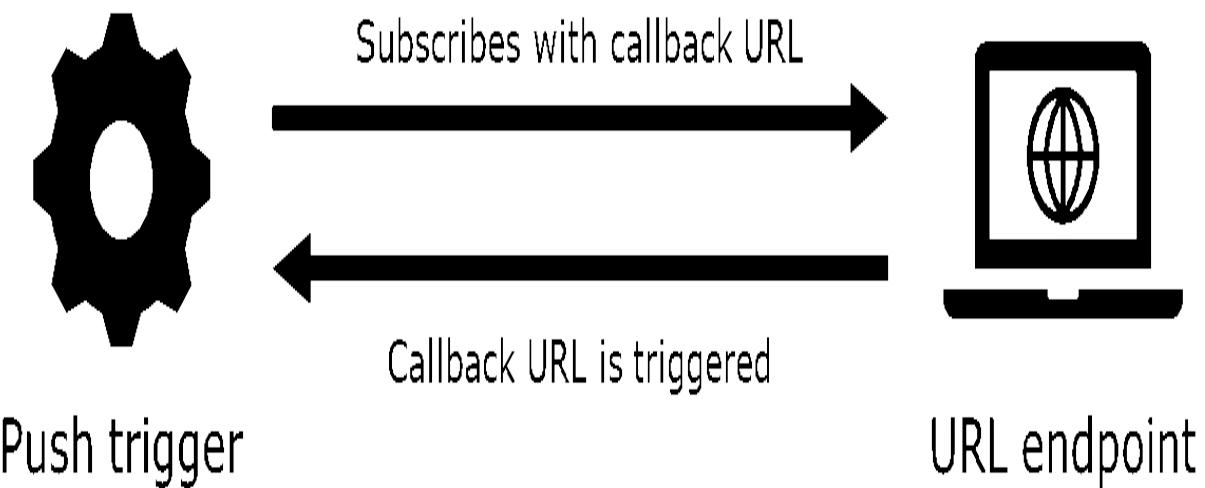
Next, we need to create that logic flow of the logic app, which will tweet the NFT images, like there is no tomorrow. The first step of the process for any logic app is a trigger, which is also a connector.

### 7.3.2 Triggers

A trigger is what starts the logic app, what makes it come alive. You can't have a logic app without a trigger, and there are many triggers to choose from. In general, there are two categories of triggers (Figure 7-13).

- Poll: This is when the trigger periodically check's the service endpoint to see if there is an event. This could be an HTTP trigger checking if there is new data on an API endpoint.
- Push: The trigger subscribes to an endpoint with a callback URL. The endpoint then pushes a notification to that callback URL when there is an event.

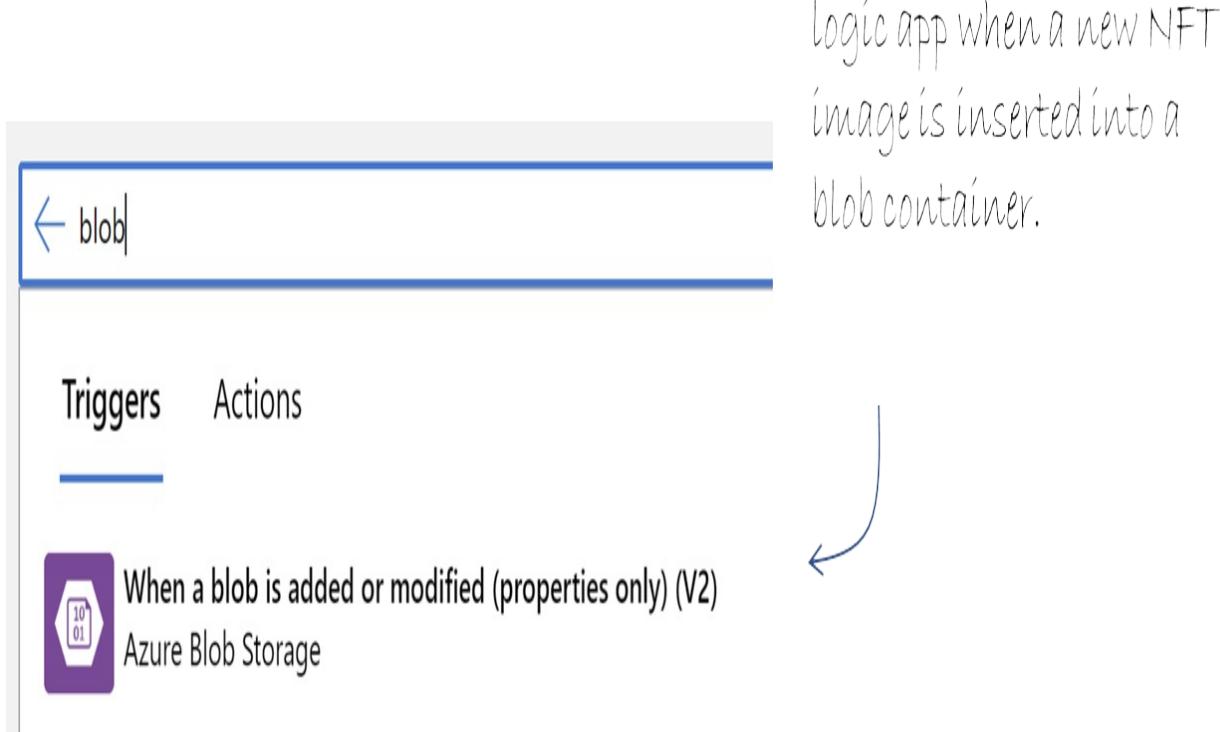
**Figure 7-13: Poll and push triggers.**



We are creating a *Blob Storage* trigger, which will tweet the latest NFT as it is inserted into the blob container in the Alpaca-Ma-Bags storage account. First step is to select a *blank template* for the logic app. If you want to, have a look at some of the templates available out of the box. While you are unlikely to ever use these for anything production ready (e.g., emailing yourself certain tweets...yeah, right), the templates will give you an insight into the kind of integrations and connections that are possible.

Once you have the blank template, we can add the trigger. Search for “blob” in the search field for the triggers as shown in Figure 7-14.

Figure 7-14: Find the blob storage trigger.



Once you select the trigger, you are asked to create a connection to the storage account you want to use. The following values are needed to create the connection.

- **Connection name:** This is your choice of connection name for the connection to the storage account.
- **Authentication type:** Leave it as access key. This is the specific key for the storage account we are connecting to. The options *Azure AD integrated* and *Managed Identity* will be covered later in the book.
- **Azure Storage Account name:** This is the name of the storage account that you have created, such as *alpacamabagsstorage*.
- **Azure Storage Account Access Key:** This is the super-secret access

key which is used to identify the connection for the storage account. Think of it as the password for the data in the storage account. Find it under **Access keys** on the storage account as shown in Figure 7-15.

**Figure 7-15: Copy the key from the storage account Access keys to use in the logic app connection.**

The screenshot shows the 'Access keys' section of the Azure Storage Account settings. On the left, a sidebar lists various management options like Networking, Azure CDN, and Access keys. The 'Access keys' option is selected and highlighted. The main pane displays two keys: 'key1' and 'key2'. Below each key is its 'Last rotated' date and a 'Rotate key' button. Underneath the keys is a 'Connection string' field, which contains a long, obscured string of characters. A blue arrow points from the handwritten note at the bottom right towards the 'Connection string' field. The page has a standard header with a search bar, a 'Set rotation reminder' button, and a 'Refresh' button.

alpacamabagsstorage | Access keys

Storage account

Search (Ctrl+ /) Set rotation reminder Refresh

Security + networking

Networking

Azure CDN

Access keys

Shared access signature

Encryption

Microsoft Defender for Cloud

Data management

Geo-replication

Data protection

Object replication

Access keys

key1 Rotate key

Last rotated: 7/17/2022 (42 days ago)

Key

Connection string

Copy the key from the Access Keys section of the Storage account.

A connection in a logic app is an authenticated interface to a service such as the blob storage. The connection is defined within the logic app and is available to any trigger or action in it.

Once you have created the connection to the storage account, we can start defining what data we are going to retrieve from it as part of the trigger. We now have to define a few more values (Figure 7-16), which will configure the trigger.

- **Storage account name:** Yes, I know we only just added this, but now it is a drop down, and the connection is defined for use.
- **Container:** This is the specific blob container we will get the NFT images from. You can either enter the path manually or use the very handy folder icon to select the container.
- **Number of blobs to return:** Every time the storage account is polled by the logic app, how many blobs do you want? We can leave this as 10 for now, but it could be more or less depending on how often you check.
- **Interval:** Finally, you can choose how often the logic app checks for new items. The default is every 3 minutes, but you can choose anywhere from every second, to once a month.

Figure 7-16: Configuring the blob trigger for the logic app.

use the dropdown to select the connection we just created before.

This folder icon makes it much easier to search through all the blob containers you have in the storage account.

When a blob is added or modified (properties only) (V2)

\* Storage account name: Use connection settings (alpacamabagsstorage)

\* Container: /alpacanftimages

Number of blobs to return: 10

How often do you want to check for items?

3 Minute

Add new parameter

Connected to alpacastorage. [Change connection.](#)

Azure Blob Storage

- alpacanftimages
- azure-webjobs-hosts
- azure-webjobs-secrets

Click the *Save* button if you haven't already, and now we have a trigger for the logic app. Of course, a trigger is only useful if there are one or more actions that follow, and now that we are retrieving all the blobs we insert, it is time for some action.

### 7.3.3 Actions

The action we want to perform on the blobs coming from the trigger is tweeting. I know you think that is extremely exciting too, so let's get right to it. Before we can get to the actual tweeting, we need to first get the blob data itself. The trigger only passes a reference to the blob, and not the data itself,

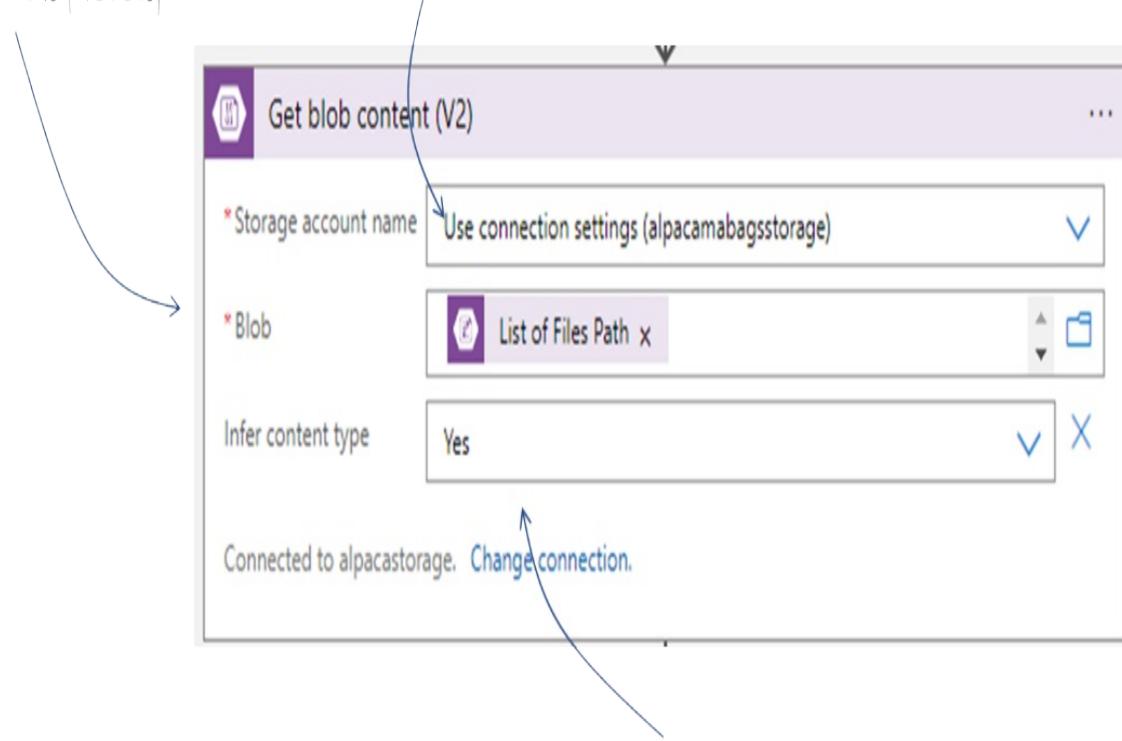
but we need that to add to our tweet, so it's time for some blob action.

Click the *+ New step* button underneath the trigger, and search for *blob* again. This time choose the *Get blob content (V2)* action and fill in the parameters as shown in Figure 7-17. The storage account drop down should contain the connection to the storage account we set up before, the blob is the dynamic field to the path of the blobs within the storage, and finally, leave the content type as default. This will make the blob data available to the next step in the flow, which will be the tweet action. Exciting!

**Figure 7-17: Retrieve the blob content for the inserted blob that triggered the logic app.**

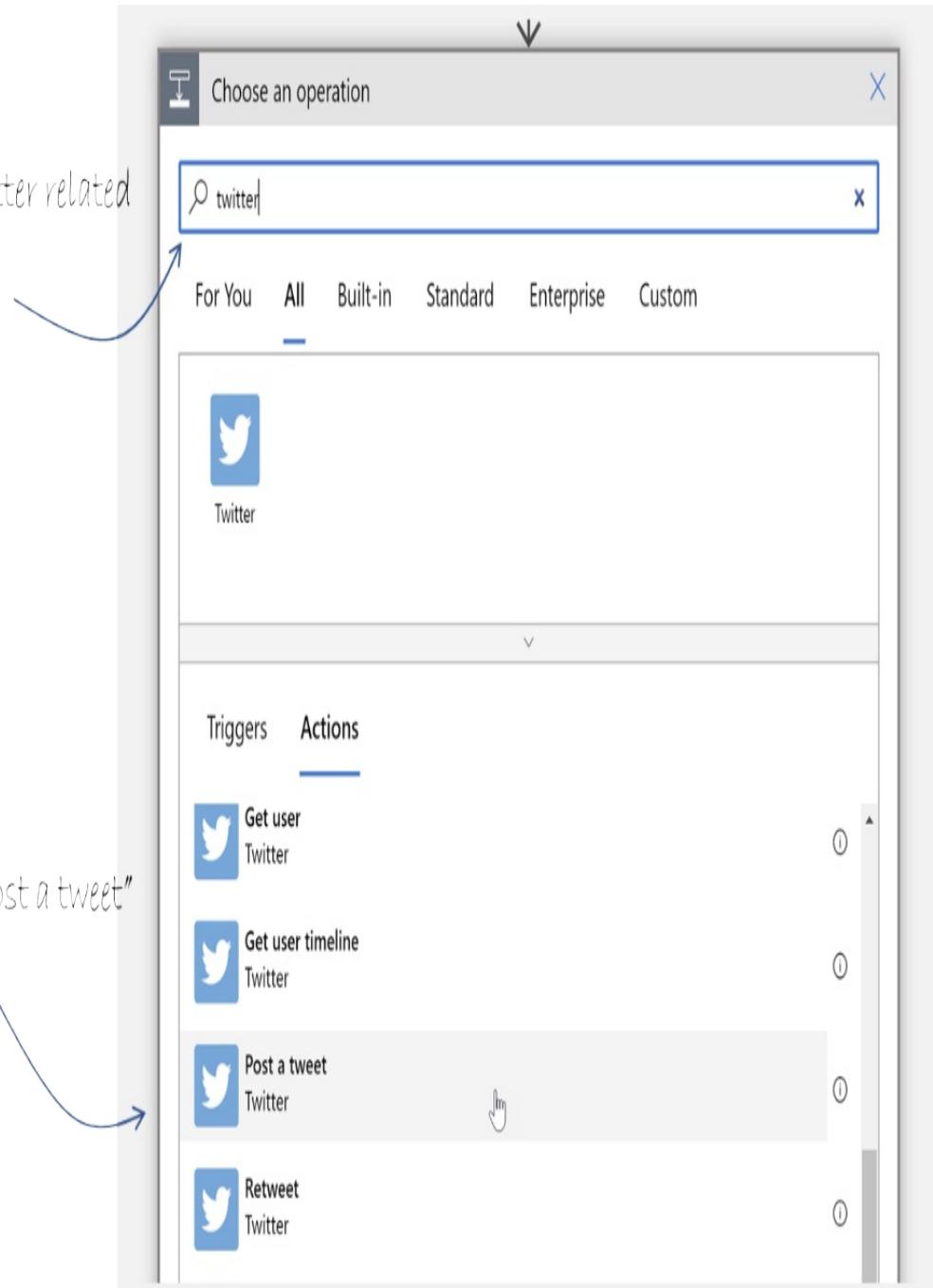
Select the file path for the blob returned by the trigger in the first step.

Use the dropdown to select the connection we created for the trigger.



Again, click on the *+ New step* button to open up the operation selection box. In the search field, type in *Twitter* and then find the “Post a tweet” operation as shown in Figure 7-18. Also, have a look through the list of available twitter actions to get a sense for the kinds of operations you can do in a logic app.

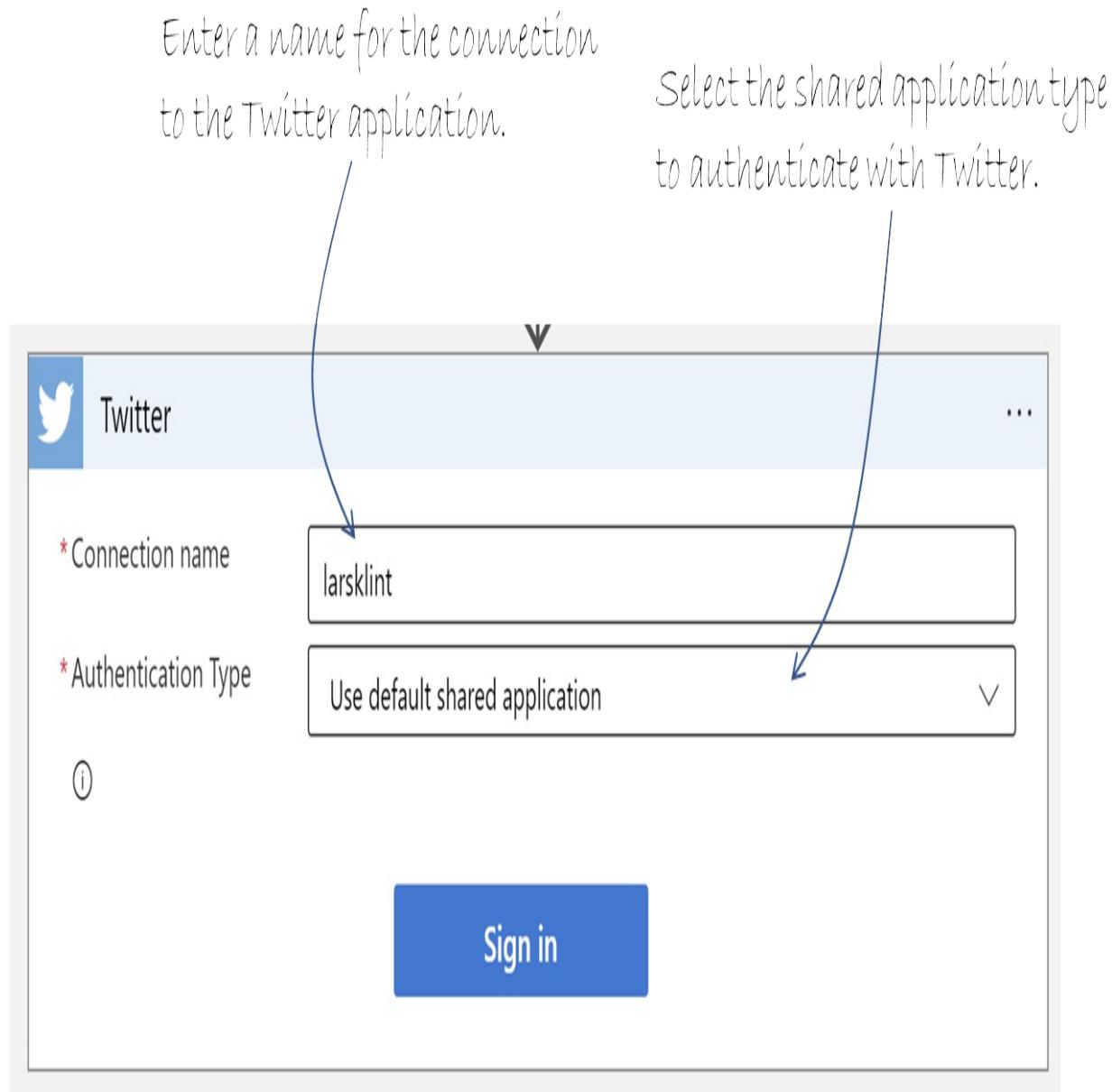
**Figure 7-18: Find the “Post a tweet” operation for the step.**



When you select the action, you will be asked to create a connection to a twitter account (Figure 7-19), which is where the tweet will be posted. To allow this interaction, Twitter requires an *application* which is a predefined entity within the Twitter ecosystem. You can create your own application using Twitter's developer portal, or you can use the application that Microsoft has already made for Azure. We are going to use the latter. The reason you would use your own Twitter application is to have more control

over the permissions and statistics for its use. We don't need that in this case.

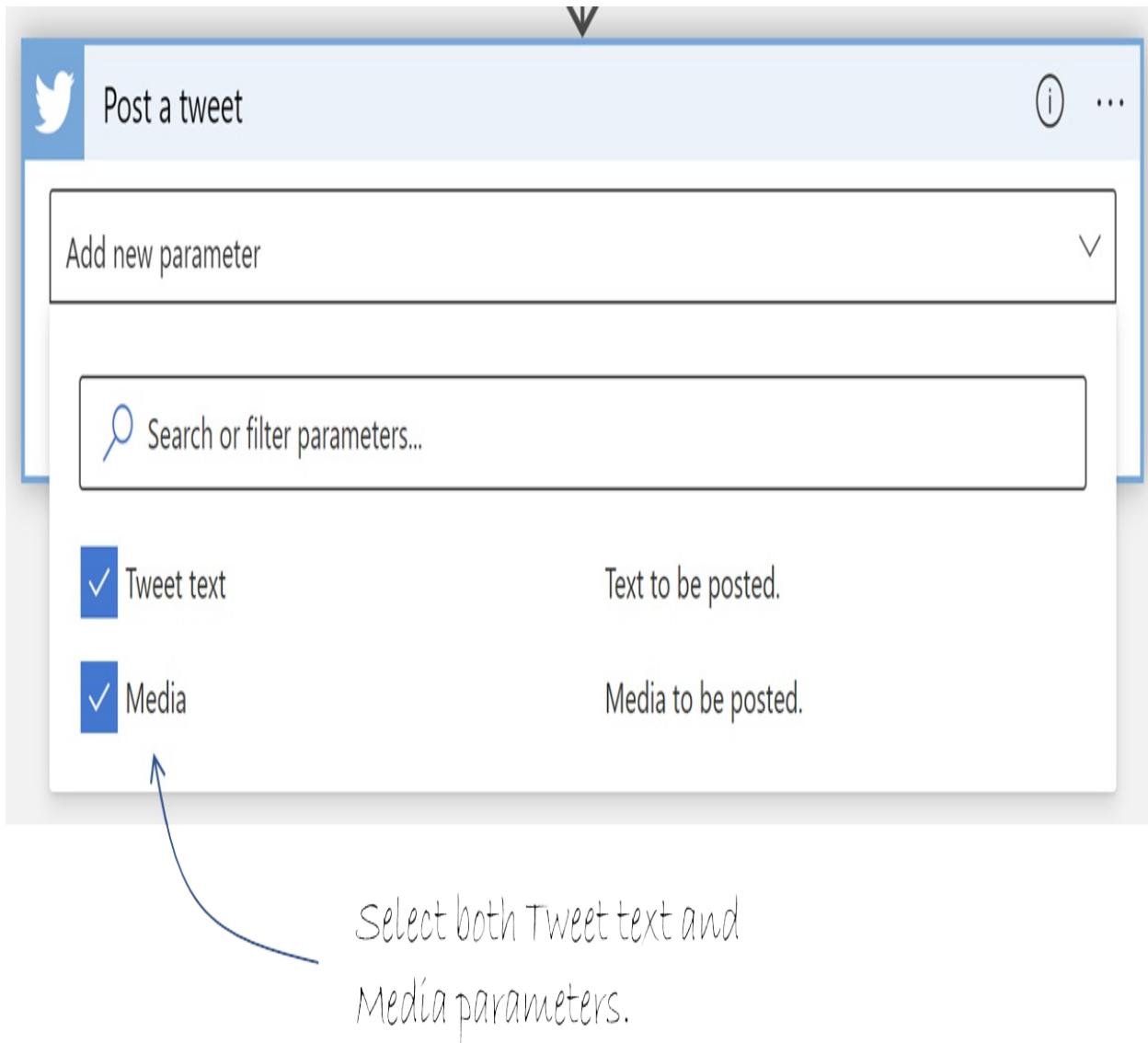
**Figure 7-19: Connect the logic app action to Twitter by authenticating with an application.**



When you click the big blue *Sign in* button, you will be asked to log into Twitter with a valid account, which will then authenticate the Azure Twitter application to tweet on that account's behalf. In other words, the logic app will have permission to send tweets out from the account you just logged in with.

Once authenticated with Twitter, you have to select the parameters you want to include in the tweeting stage. There are only two, *Tweet text* and *Media*, and we will need both (Figure 7-20). The tweet text is what you want the text in the tweet to say, and the media is the NFT image, which is the blob data we now have available from the previous step.

**Figure 7-20 Choose both the available parameters for the imminent tweet.**



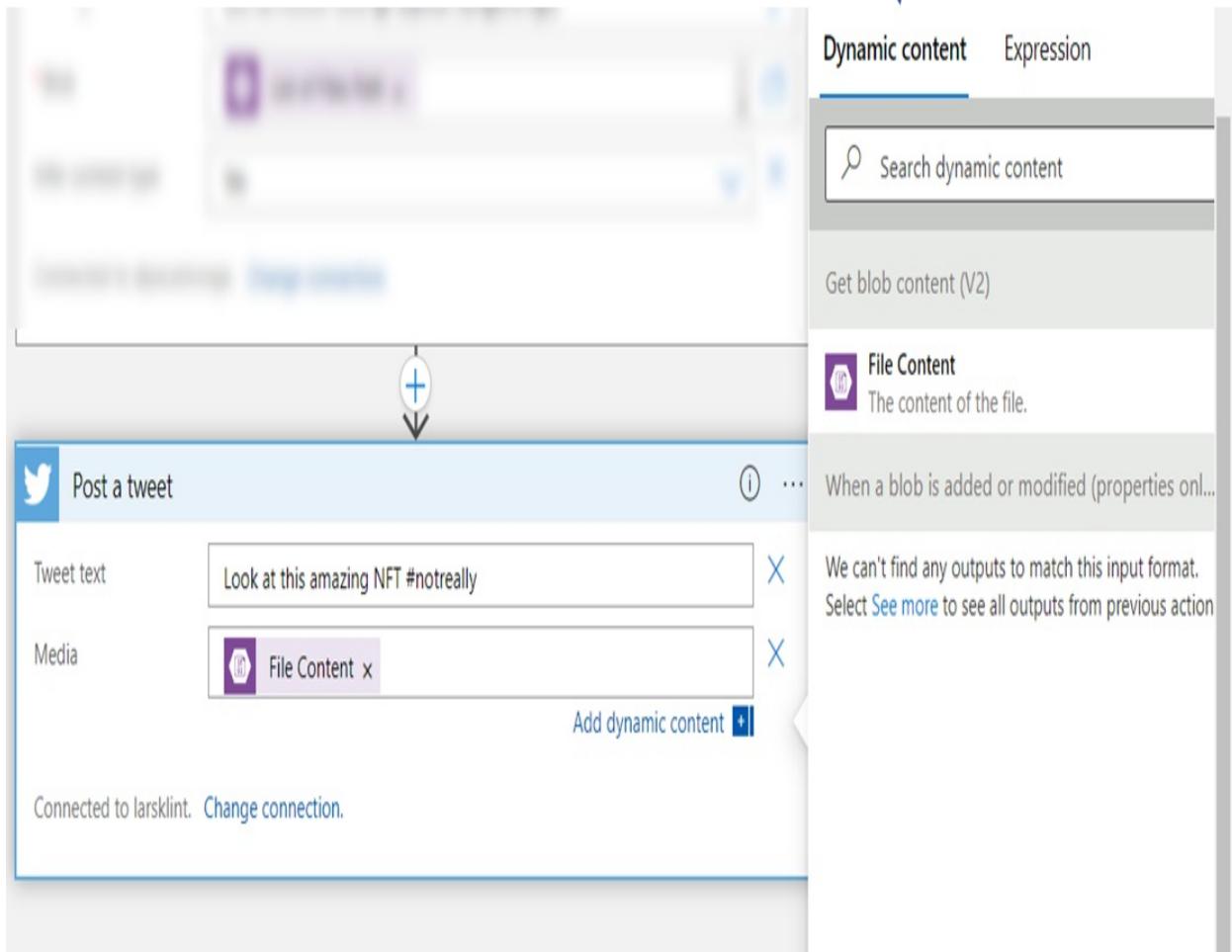
Fill in the tweet text and notice when you click on it and the *Media* field that the dynamic fields dialog opens again (Figure 7-21). This will happen on most interactions within logic apps, as that is really the key to their power.

The whole idea of logic apps is the flow of data between seemingly disparate entities, such as our blob storage and tweets. If you were to code all of this yourself, which of course is possible, you'd be at it for days, maybe even weeks. While I appreciate the excitement and satisfaction in getting applications to work together in code, I also appreciate my time to deliver a solution. A business problem isn't about solving it in the most interesting and clever way, but about doing it fast and with confidence in the stability and security of that solution. That is what logic apps give us. I hear you saying, "I like to write code", because I do too, but a service like logic apps not only gives you a faster and more tested solution, but also serves the business in most cases. You are a better team player, thus adding more value to a business, by using the best tool for the job. Logic apps can often be that best tool, even if it doesn't touch actual code.

What logic does touch on is, well, logic. And that is a core part of any application development and solution. The dynamic fields of the various steps in a logic app are where your tech brain needs to shine. The dynamic fields are your parameters, inputs and outputs you need to manage and massage to get the data flow you need. And that is why I find them so exciting. They are quick, simple, yet deliciously logical as well.

**Figure 7-21: The dynamic content dialog is the power of logic apps.**

The dynamic content pane is present for most actions.



Back to the task at hand. Fill in the tweet text and choose the *File Content* for the media (there is only that dynamic field available), and save the logic app, using the *Save* button. And then it's time to test. Upload an image file into your blob container by using the upload pane and selecting your file (Figure 7-22), then either wait for the logic app to trigger or go and trigger it manually yourself.

**Figure 7-22: Upload an image to blob storage to trigger the logic app.**



Phew! That was a lot of triggers and actions to get our tweet out to the world. If we take a step back, this approach can be used for more than just sharing vain alpaca pictures on social media. Every workflow you create for a logic app will follow this pattern of a trigger to start the process, then any number of actions and logic constructs to manipulate the data and achieve the desired outcome for your business or project. This could be sending an email when a specific type of record is inserted into a database, running an Azure function when a message is received on a queue, and much more. Try it out for yourself too. Go press all the buttons and see what you can make happen in your own workflow with a new logic app.

If you have set up everything correctly, the logic app will now have run and you have a tweet on the account you linked. Wondering which image I used? Check out Figure 7-23. You're welcome.

Figure 7-23: The actual tweet I created at the time of writing. It is still there.



Only one more step in creating our serverless architecture for Alpaca-Ma-Bags: API Management.

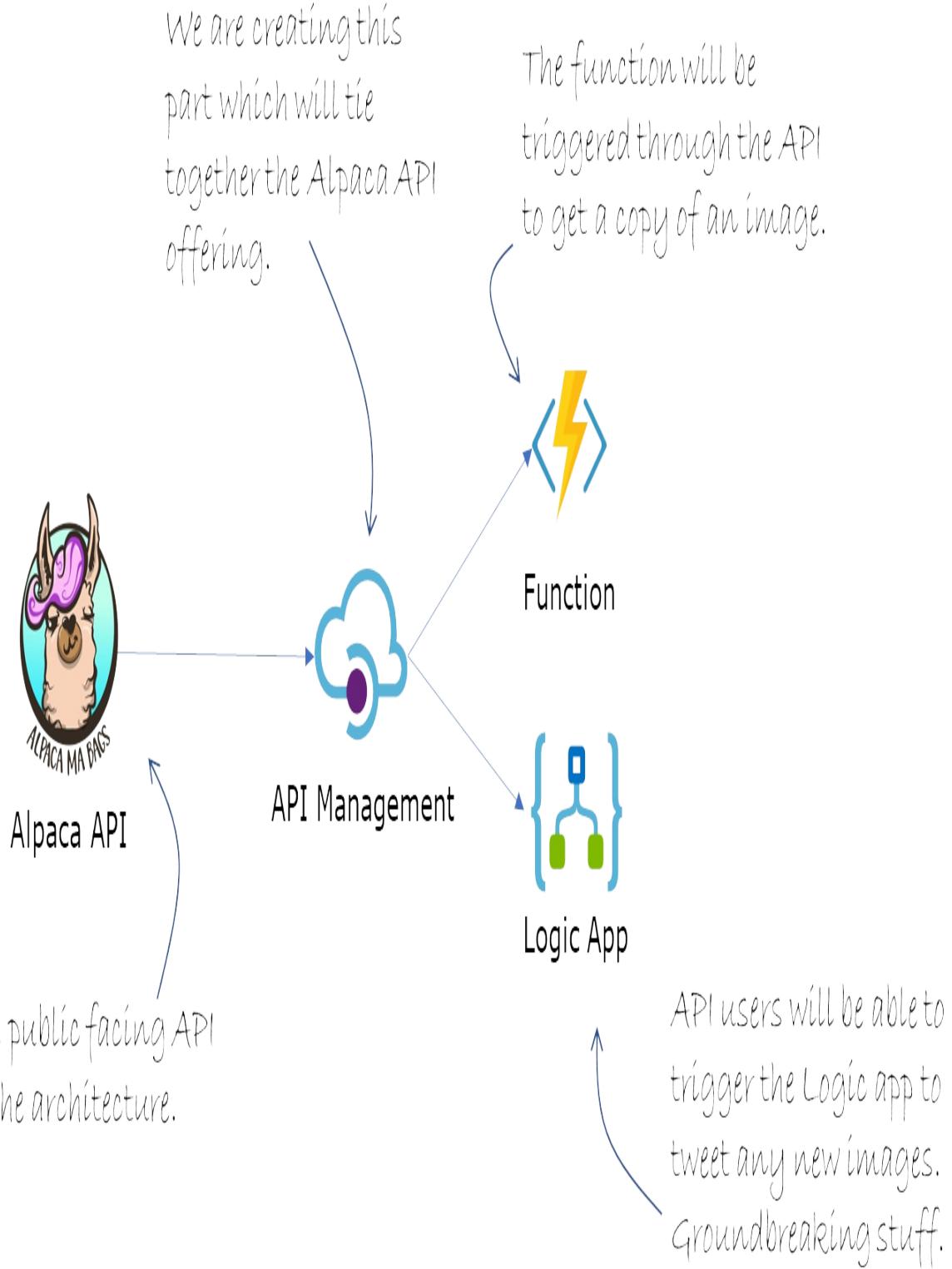
## 7.4 API Management

The final step in the master plan for Alpaca-Ma-Bags and their NFT marketing masterpiece is to create an API that allows their partners and other

third parties to view and share the images too. For this we will use API Management, a service that lets you bind together a range of disparate services into a single product, also called an API, that you can manage, monitor and distribute through Azure. Developers can then subscribe to the API and use its functionality in their own applications leveraging your work, which in turn is leveraging other developers' work. Beautiful, isn't it?

The part of the Alpaca-Ma-Bags serverless project we are creating now is the API Management service (Figure 7-24). This will glue together the function and logic app we created earlier and make them available to third parties in an authenticated and managed way using, of course, serverless.

**Figure 7-24: The API Management service glues the serverless project together.**



Let's create an API Management instance and then delve into the main parts

of the service. We are going to create the service using the Azure CLI, but further on will also use the Azure Portal as the visuals are often easier to understand, and some of the features only make sense when using the Azure Portal. Use the following command template to create a new instance, but don't execute it just yet. You'll have to change a couple of things.

```
C: /> az apim create --name alpacaapim --resource-group AlpacaRG
```

There are a few parameters here to understand. By now you should be familiar with the `az` command, which is the Azure CLI keyword for a new command. The next part `apim` indicates we are doing something with API management and `create` is that something. Choose the same resource group as for the other services in this chapter. The `publisher-name` is the name of your organization and will appear in the developer portal and on emails and other communications. The name must be unique across all instances on Azure, so choose your own. We will get to what the developer portal is shortly. The `publisher-email` is the email address which receives all the information from the service, such as system notifications. Finally, there is `no-wait` which doesn't affect the creation of the API Management service but creating one can take a very long time. It often takes 35-40 minutes to fully provision an instance, and the `no-wait` parameter tells the Azure CLI not to hang while it is being done. Instead, it fires off the command, and you can continue to do other things in the CLI. Okay, now you can execute the command.

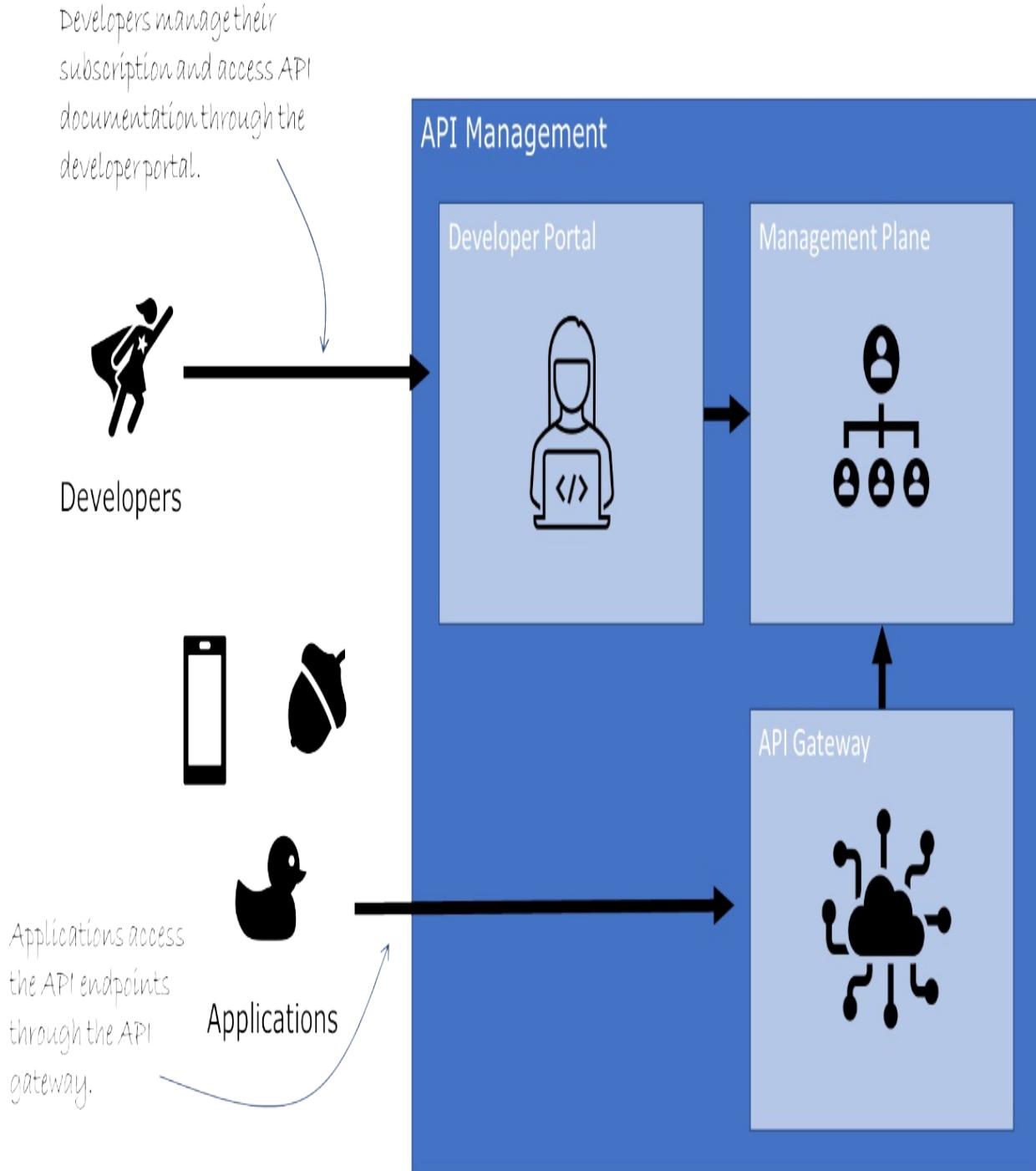
### **API Management Pricing Tier**

By default, the API Management pricing tier is Developer which is the most economical but not meant for production use. You can choose Consumption too, which is the typical serverless tier that comes with 1 million calls for free.

As you are waiting for the API Management to be provisioned, this makes it a good time to explain the main parts of the service, the API Gateway, the Management Plane, and the developer portal, as depicted in Figure 7-25. It gives developers access to their subscription level and API documentation through the developer portal, and it lets developers connect their own applications through the API gateway. The management plane is what binds

it all together.

**Figure 7-25: The three parts of API Management.**



The developer portal, management plane, and API gateway are all equally

important and together provide all the tools you need to create a robust API, as we are about to. They are all fully managed by Azure and hosted by Azure. Let's start with the API Gateway.

### 7.4.1 API Gateway

Every single request that a client application makes to the API host by the API Management services goes through the API gateway. Remember earlier when I mentioned that changes to the internal architecture doesn't necessarily mean that there are changes to the API the applications are using? That is what the gateway oversees. It is a layer of abstraction between what the outside world of third-party applications see and what backend services are doing. This is a critical feature for two reasons. First, it will allow you to introduce new features and fix bugs continuously without affecting the public API. Second, and most importantly, your users of the API won't have to change their use of the endpoints every time you change something on the backend. Ideally, the users of an API will never know when changes are made to the services behind the API.

Open the Azure Portal to the newly created API Management service with the name you gave it. Then go to the APIs menu as shown in Figure 7-26, and scroll down on the API templates on the right. These are all different ways that you can create APIs for your API Management service. Find the *Azure Function* API template, which we will use in a moment.

**Figure 7-26: The API section of API Management.**

The screenshot shows the Azure API Management service interface. On the left, there's a sidebar with various navigation options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events, Settings, Properties, Locks, and APIs. Below APIs, there's a list of sub-options: APIs, Products, Subscriptions, Named values, Backends, Policy fragments, API Tags, and Schemas. A large callout arrow points from the text "use the 'APIs' menu to get to some of the features of the API Gateway." to the APIs menu item.

**Define a new API**

- HTTP**: Manually define an HTTP API
- WebSocket**: Streaming, full-duplex communication with a WebSocket server
- GraphQL**: Access the full capabilities of your data from a single endpoint
- Synthetic GraphQL**: Build a GraphQL service using existing HTTP APIs

**Create from definition**

- OpenAPI**: Standard, language-agnostic interface to REST APIs
- WADL**: Standard XML representation of your RESTful API
- WSDL**: Standard XML representation of your SOAP API

Templates for API endpoints you can add to the API Management service.

We will get to creating the APIs in just a moment, when you hook up the Azure function and logic app to the API Management instance, but this is the main part of the service that has to do with the API Gateway.

## 7.4.2 Management Plane

When you need to manage users, package the API endpoints into various

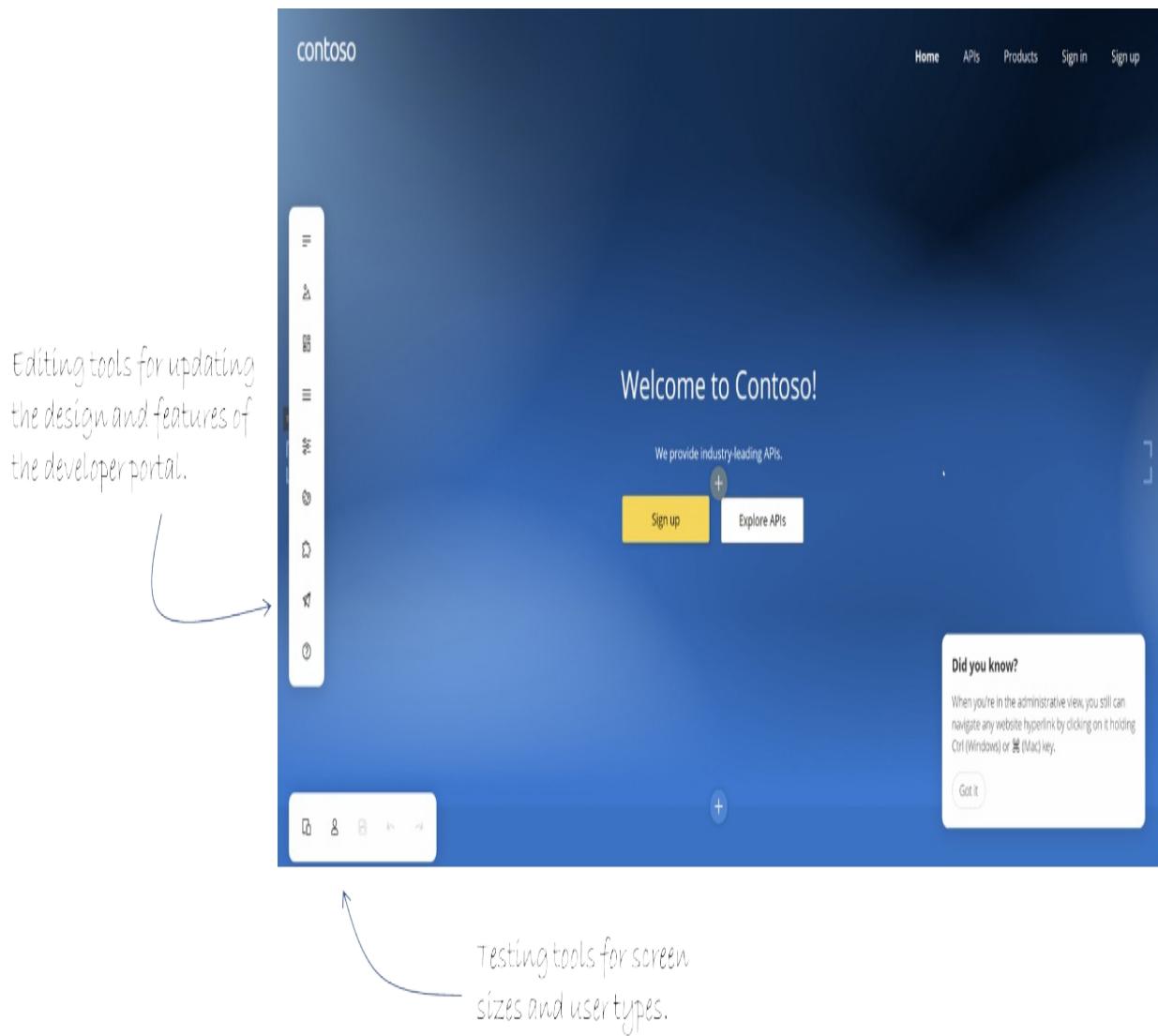
products, or look through the analytics for the API, you use the management plane. As the name, and Figure 7-25, suggest there is no access to the management plan for developers or applications. This is purely your domain as the Azure API provider and API admin.

The management plane is made up of a bunch of areas of the service, such as backend services, policy fragments, and products. For example, if you want to create an API product that is made up of certain endpoints within the API Management service, and it has certain quotas on its usage, that is all done through the management plane. Any changes to the settings for the service such as validating schemas for requests and responses in the API are also part of the management plane.

### 7.4.3 Developer Portal

If you want developers to use your API, you have to show them how through examples and documentation of the endpoints. The developer portal is an auto-generated, fully customizable website that documents your API. The website is generated for you, but you can change almost every part of it, including adding your own content, change the styling, add your own branding, and navigation elements, just to name a few. Figure 7-27 shows the developer portal when it is brand new and smelling like lemons on a warm spring evening.

**Figure 7-27: The vanilla freshly created developer portal. There are tools for updating the design and features, as well as testing it.**



It is a public website, which developers can use to sign up for and acquire API keys for use in their own applications. They can find the API documentation, test API endpoints and access usage statistics for their account.

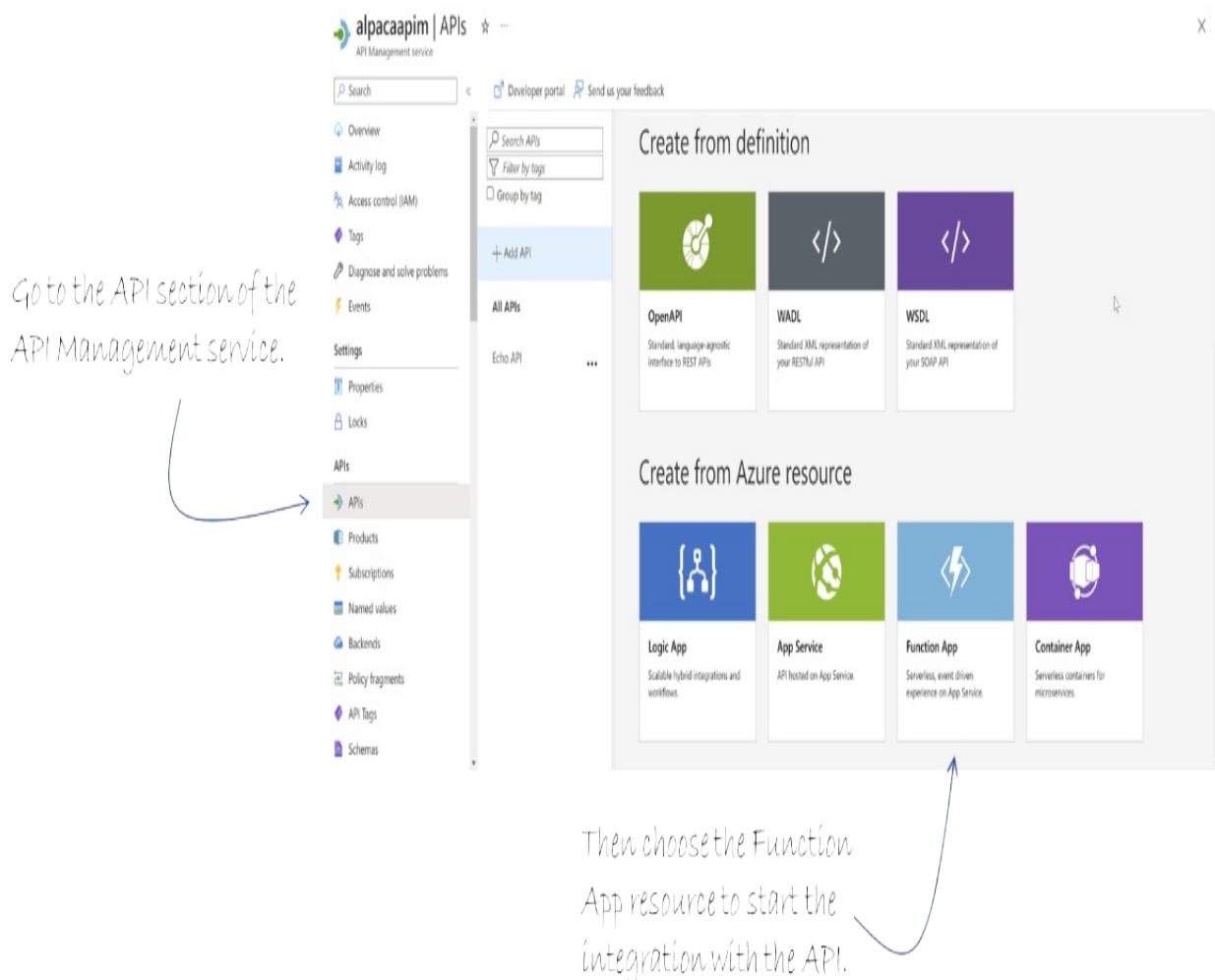
At this point you might ask “Lars, why are you showing me this developer portal? Are we going to configure it or press some buttons?” No, sorry. The point here is that you know it gets created and it is the main way developers interact with your API. We just don’t have enough pages to go into details about how you design and configure the developer portal. I do invite you to access it now that you have it and press all the buttons. Get a sense for it.

Okay, now let's connect some services to make some API endpoints.

#### 7.4.4 Connecting the Azure function

Enough talk about what API Management consists of. Let's get into hooking up some endpoints to let the users of Alpaca-Ma-Bags use the services we have created. We'll start with integrating the function that processes the NFT. In the Azure portal, go to your API management instance, then go to the API section, and then choose the *Function App* as shown in Figure 7-28.

**Figure 7-28: Choosing the Function app to start the integration with the API Management service.**



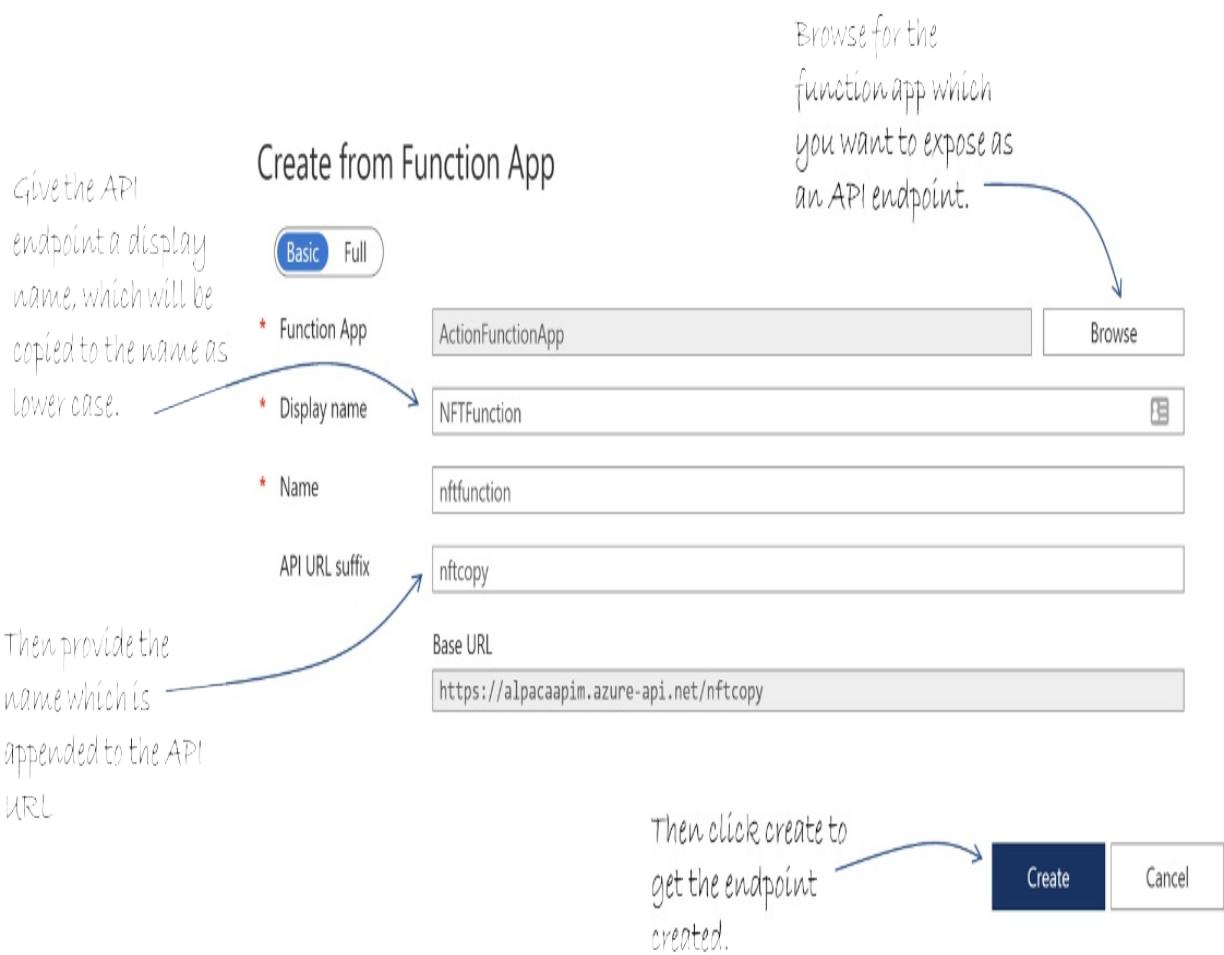
This will prompt you to fill in the function app name, a display name, name

and API URL suffix as shown in Figure 7-29. Enter values that make sense for you, then click *Create*.

**Note:**

All functions within the function app will be exposed in the API. If you have 6 functions hosted inside a single function app, then you will get 6 API endpoints in API Management when using this integration template.

**Figure 7-29: Find the function app to use for an endpoint, then give it a name and URL suffix.**



This will create the endpoints for the API based on the functions in the function app. We just have the one function we created before, which is then added as shown in Figure 7-30. If you select the GET method for example, you can see the URL for the endpoint, as well as which parameters are

needed.

**Figure 7-30: Once the function app has been linked to the API management service, it shows under APIs in the portal.**

The screenshot shows the Azure API Management portal interface. On the left, there's a sidebar with search and filter options, and a main area for managing APIs. A callout points to the 'NFTFunction' API, with the annotation: 'Select the GET method of the new API operation.' In the center, the 'Operations' tab is selected, showing a list of operations: 'Frontend', 'Inbound processing', 'Backend', 'Outbound processing', and 'Definitions'. The 'Frontend' section contains a 'GET /nft/{nftid}' operation with a 'Template parameters' section containing 'nftid \*'. The 'Backend' section lists 'Azure FunctionApp' and 'ActionFunctionApp'. Arrows from handwritten annotations point to the 'Frontend' section and the 'Backend' section, with the annotation: 'Notice the backend references the function app.'

How simple was that!? To make sure the endpoint works, let's test it by going to the *test* tab for the API, as shown in Figure 7-31. All you have to do is enter a value in the template parameter, which in this example is the file name, and then click *send*. If you scroll down a bit further in the portal window, you will see the request URL is shown as well. It would be something like <https://alpacaapim.azure-api.net/nftcopy/nft/nft-1-2022.png>.

**Figure 7-31: Testing an API directly through API management.**

The screenshot shows the 'Test' tab of the API management interface for a 'ProcessNFT' API. The left sidebar lists operations: 'Search operations' (selected), 'Filter by tags', and 'Group by tag'. Below are two endpoints: 'GET ProcessNFT' and 'POST ProcessNFT'. A handwritten note says 'Choose the test tab for the API.' pointing to the 'Test' tab in the top navigation bar. Another note says 'Enter a value for the nftid parameter, in this case it is the blob name.' pointing to the 'nftid' query parameter field which contains 'nft-1-2022.png'. A third note says 'Then press send to test the API.' pointing to the 'Send' button at the bottom. The right side shows 'Template parameters' and 'Query parameters' sections, both currently empty. The 'Headers' section also has an empty table. At the bottom are 'Send', 'Trace', and 'Bypass CORS proxy' buttons.

This will trigger the function we built earlier and create a copy of the image in your bound blob container.

**NOTE:**

If you are getting an error, it could be because you haven't created the blob containers on the linked Azure storage account. In the configuration settings for the function app, you can see which storage account is being used by inspecting the `AzureWebJobsStorage` setting.

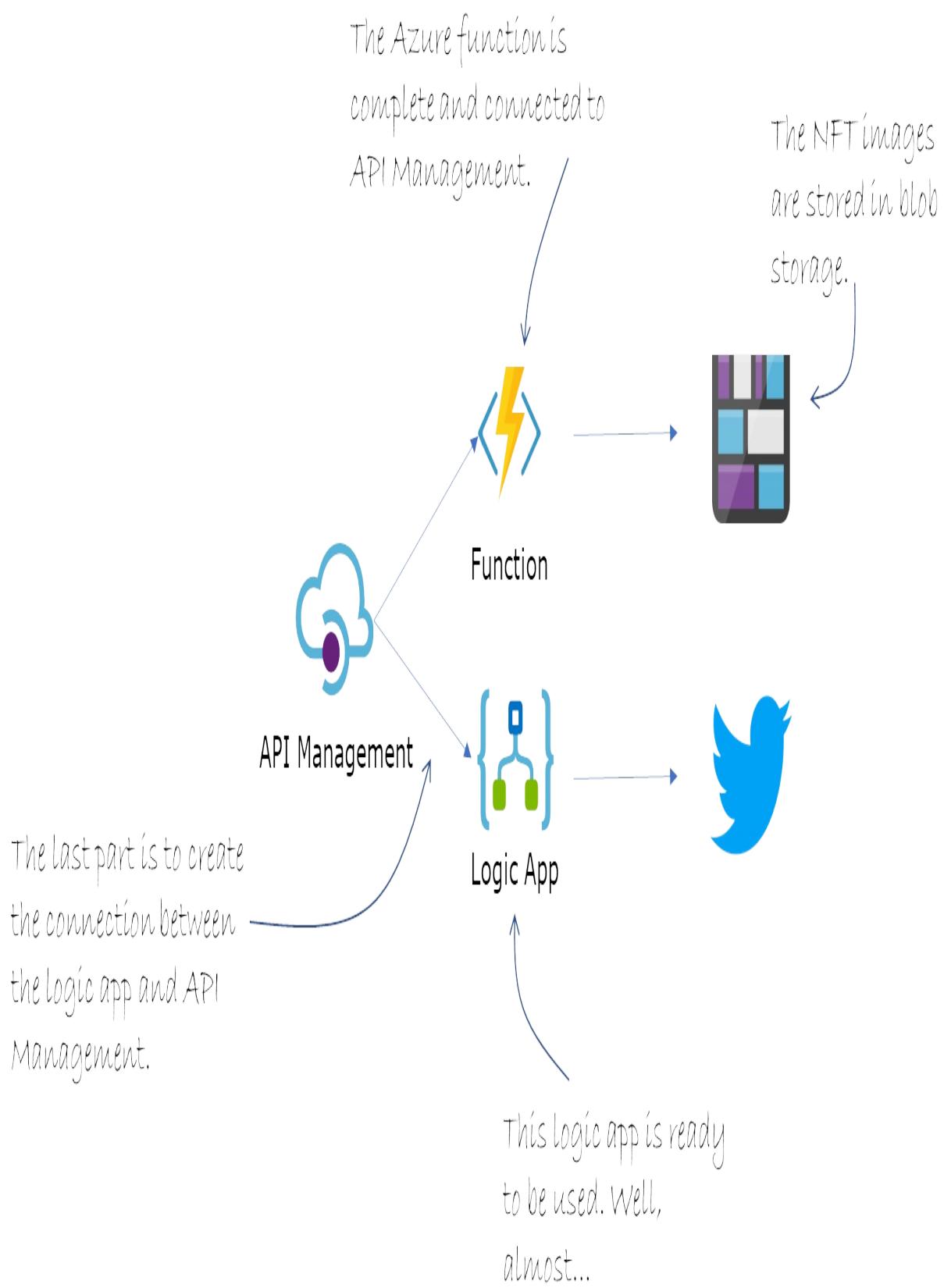
That is the first of our two API endpoints done for the serverless Alpaca-Ma-

Bags API. Now for the second one using logic apps.

#### 7.4.5 Connecting the logic app

The second part of the API is hooking up our logic app tweet machine. As shown in Figure 7-32 this is the last part of the Alpaca-Ma-Bags architecture that we defined at the start of the chapter.

**Figure 7-32: The last part of the architecture is connecting the logic app to API Management.**



You might have guessed it, and yes, this is equally simple to do. However, there is a catch. To connect a logic app to API Management, it must have an HTTP endpoint. If you think about it, that make sense. An API is a collection of HTTP endpoints exposed to users over a network, specifically with no user interface. It stands to reason that any endpoint must be accessible over HTTP, and that then trickles down to the kind of integrations you can do within the API Management API gateway. Our logic app from earlier doesn't have an HTTP trigger, but rather a blob trigger.

Considering we are roughly halfway through the book, it is time to step up the learning challenge a bit. For this last part of the serverless chapter, I propose a little challenge: create a second logic app that has an HTTP trigger, and then integrate it with the API management. It should have the following characteristics:

- Triggered by HTTP.
- Receives the NFT id via a query string parameter
- Retrieves the NFT image from blob storage using the id
- Posts a tweet with the image and a description of your choosing
- Connect the logic app to the API Management API gateway
- Bonus points: Use some form of authentication for the incoming request to the API

Go ahead and have a go. I will wait here until you are done. I promise.

That last bonus point brings us on to a small elephant in the room. Sort of Dumbo sized. I am talking about authentication of course. A lot of the chapter has elegantly avoided the topic, but that doesn't mean it isn't part of the story. However, for this part of your Azure journey, we focus on the three services, and authentication in various forms is coming up in the book. We aren't avoiding or neglecting it, but it isn't quite a part of the story just yet.

API Management is a very large service on Azure, and you have only got the smallest of introductions to it in this chapter. At this point it is worth your time to "press more buttons" in the service to get a better understanding of what it does in more detail. It is out of scope to dive into API Management more in this book, with the goal of giving you a functional introduction to the

service being achieved. Almost at the end of the chapter, but first a slightly contemplative piece to finish.

## 7.5 Serverless is just PaaS?

We have now gone through three distinct serverless services on Azure; functions, logic apps, and API management. At this point you may have a little voice in the back of your head trying to get your attention. Remember back at the beginning of the book in chapter 1, we went through the three main types of cloud platform abstractions: Infrastructure-as-a-Service, Platform-as-a-Service, and Software-as-a-Service? The little voice in your head is saying “isn’t serverless very similar to PaaS?”

You are right, that there are certain similarities between the two models, the main one being to save time for developers. Both serverless and PaaS are designed to remove yak shaving and just let developers get on with the job of writing code and solving business problems. However, there are at least three distinct differences between the two approaches as outlined in Table 7-5: scaling, time to compute, and control.

**Table 7-5: Comparing the main three differences between PaaS and serverless services.**

Concept	PaaS	Serverless
Scaling	You get more ways to decide how the service scales as compute requirements change, but you have to configure scaling yourself.	Scaling is done for you and “just works”. There are some options, such as having a premium tier function app.
Time to compute	Applications can take a while to spin up and get ready to provide the	Applications are only executed when invoked, but it happens very fast.

	compute capacity.	
Control	You have much more control over the development environment, but that also means more configuration and maintenance.	You have very little control over the development environment, but it is fast to get started.

Both serverless and PaaS have their strengths and weaknesses and, as always, the important thing is that you use the right tool for the right job.

## 7.6 Summary

- Serverless architecture is a spectrum ranging from using many servers to using less servers. There is no such thing as “serverless” where no servers are used.
- An API is the connection that allows two or more applications to communicate with each other using a collection of defined endpoints.
- An API has no graphical interface and is exclusively for pieces of software to connect and communicate with each other in a managed way.
- Azure functions are units of compute that perform a single task.
- Azure functions live within a function app, which defines the tier such as premium or standard, as well as the compute power for the functions.
- You can create an Azure function using various methods, including the CLI and Visual Studio Code.
- Every function must have exactly one trigger, which is the entry point for the function. This is how the function starts every single time.  
Triggers can be via HTTP, from a new blob file, a new queued message, and many other ways.
- A binding in a function is a connection to another Azure resource, such as blob storage. Bindings are a shortcut to manipulating the connected resource within the function at run-time.
- Using VS Code to develop Azure functions will provide you with a

complete local environment, so you don't accidentally publish anything to production.

- The three most common hosting methods for functions is consumption, premium and with an Azure app service plan. Each have their advantages and drawbacks.
- Logic apps can combine disparate third-party services into a single workflow to provide very quick and valuable additions to your infrastructure.
- All logic apps have exactly one trigger which starts it. These triggers can be anything from a new blob being inserted to a Tweet being posted with a specific keyword.
- Triggers in logic apps can be poll, using a simple http call, or push, which uses a callback service for the trigger service.
- Actions in logic apps manipulates the data from the trigger and provides the business value. Some workflows have one action, some have many.
- API Management provides a serverless way to create an API in Azure using existing services and resources. The services don't have to be hosted on Azure necessarily.
- Serverless and PaaS architecture approaches are similar but have differences, including scaling, time to compute and control over the development environment.

# 8 Optimizing Storage

**This chapter covers:**

- Creating a free static website hosted in your storage account.
- Managing data inside the storage using azcopy
- Sharing access to data securely and with limitations
- Using lifecycle management to save costs and optimize data use

Let me start this chapter by saying this: no, you aren't going mad, this really is another chapter on storage. While chapter 5 provided the fundamentals of storage on Azure by delving into the storage accounts and the types of data you can store there, we only scratched the surface of what you *can* do, and what you *should* do. To use storage on Azure efficiently we need to understand it better than only having the foundations, which are indeed important, but not enough. It is like learning to ride a bike and never taking the training wheels off. Yes, you can get along and eventually you will get there, but it is slow and much harder to go around the obstacles efficiently.

In this chapter we dive into ways that you can save on cost, securely share access to your data, keep track of data versions, manage data life cycles and much more. These are all important parts of Azure storage usage and maintenance. Join me as we visit the weird world of banned technical literature again.

## 8.1 Banning Books dial up their storage game

Banning Books now have some of their data online in a blob container on Azure storage, but they are ready to take advantage of more features of Azure storage to improve their shady business model even more. They have hired a consultant, you, and they have these business requirements which you need to implement.

- A simple website that is as low maintenance as possible, showing just

the company services.

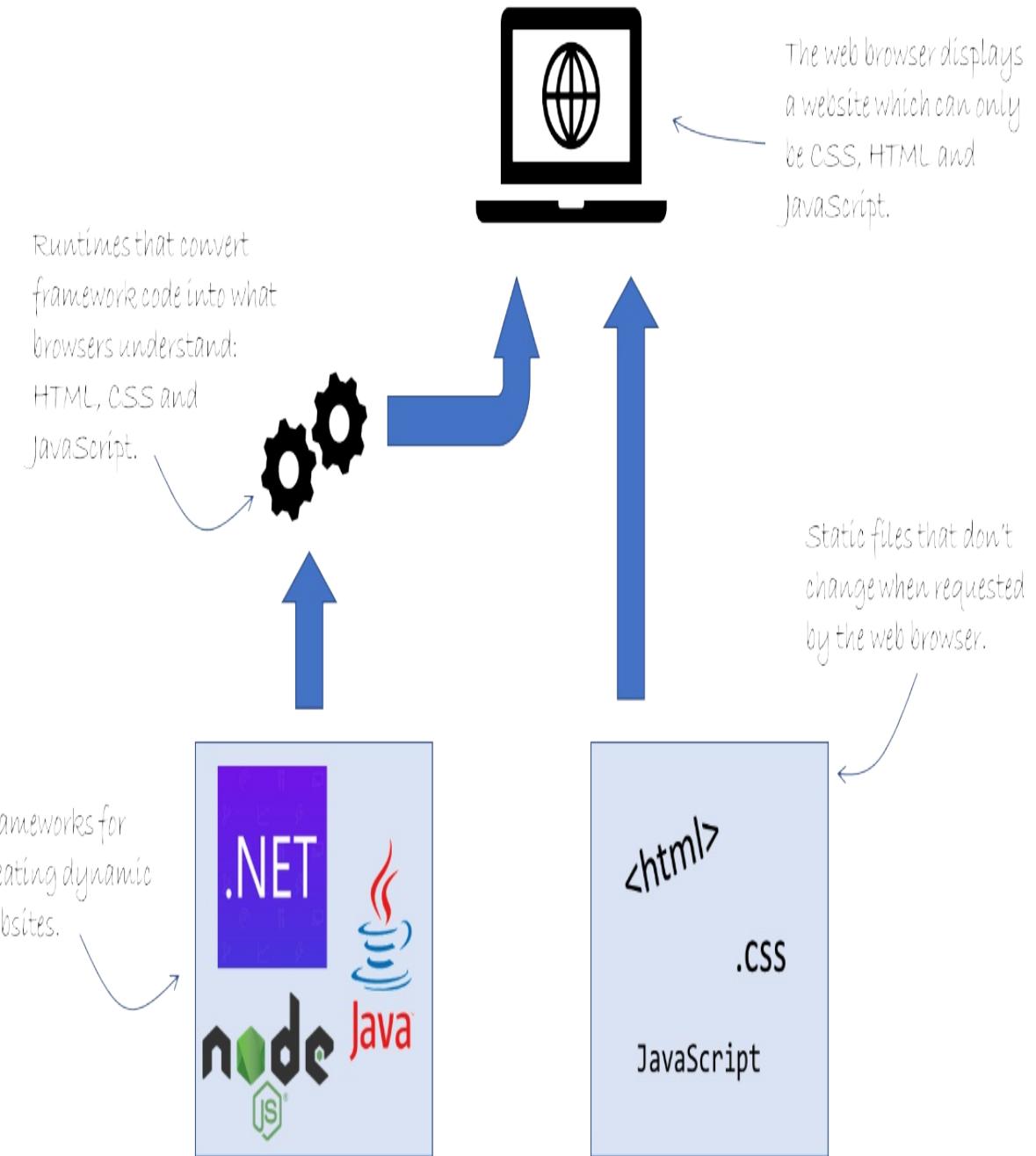
- A simple way to copy data to and from the storage account.
- Data access to third parties in a secure and authenticated way that is simple to manage.
- Versioning of the data that is uploaded so Banning can keep track of changes and dates.
- Replication of the books in their shop to prevent sudden loss of literary assets due to annoying legislative hurdles.
- Management of data to reduce costs and streamline the use of the data.

All these features need to be done within the Azure storage account to ensure easy maintenance and cost reduction. As you will see, Azure Storage has so much more to offer than first meets the eye, and first is setting up a website that requires almost no maintenance and is free. Yep, free.

## 8.2 Static website in Azure storage

Websites are a dime a dozen. They are what keeps the modern world turning for many businesses and lets them talk with their customers. Websites come in all colors, all sizes, and all designs. One thing they all have in common though, is that they are made from HTML, CSS and JavaScript. These three distinct parts are what make up any website you have ever visited. Websites can generate this web content dynamically by using runtimes such as Node.js, ASP.NET, and Java, or they can be made up from static files which are served directly to the web browser as shown in Figure 8-1.

**Figure 8-1: A dynamic website being served versus a static website.**



Banning wants a simple, low maintenance website, which won't change very often, and not service any dynamic content. For this, we are going to host a static website on Azure storage. It used to require a web server running several processes, as well as a domain service, to run the simplest of websites. It would require specific skills to get it working, and it could take longer than expected to make happen. This isn't the case anymore.

You can host a simple static website directly within Azure Storage, which will suit the requirements of Banning Books. And guess what!? We are going to do that right now. For this you'll need a website, and you can use your own website consisting of HTML, CSS and JavaScript files, or you can use the official Banning Books website available at <https://github.com/lkлин/banning-books>. The website is shown in Figure 8-2.

**Figure 8-2: The Banning Books website.**

 [Home](#)

Banning Books

## The largest range of underground tech literature

[See More](#)

**New release**

**Dark Cloud IN ACTION**

Lars Klint  BOOKS

Learn all about how to use the dark clouds in cloud computing to get that extra bit of computing into your projects. Dark clouds lurk in the shadows and slurp on your dormant VMs when you aren't looking.

**High-Quality Products**  
All our goods are professionally manufactured to a high specification in the dark forests of Germany.

**Free Shipping Worldwide**  
Wherever you are in the world, you'll receive your products within 10 working days of your order using unmarked postage.

**Full Warranty**  
Something wrong with one of our products? No problem. We will deny ever sending it.

**Personalization**  
Choose colour, size, engraving and other options to disguise your literature from pesky litigious eyes.

### Reviews

"I bought Undercover Kubernetes 3 years ago, and no one ever found out."

- D. Niall

"You can't beat Banning for the quality of their products at such reasonable prices."

- Charlie E. Apsgate

**Let us help you find what you need**

[Shop](#)

**Customer Service:**  
123-456-7890

[Shop](#)  
[Shipping](#)  
[Store Policy](#)

Powered by [Banningbooks.net](#)



Get your files ready, whether you are using your own or the ones provided in the Github repository, and then we'll get them uploaded to the Banning storage account using one of the simplest and most effective tools for copying files to and from your storage accounts.

#### **Paths and Punctuation for demos**

My repository of the website code will be at "C:\Repos\Banning Books\banning-books" but your path may be different. Also, I am using a Windows machine to run my CLI commands, so you may have to adjust some of the quotation marks and punctuation where appropriate, if you use a MAC or Linux machine.

### **8.2.1 AzCopy**

We want to move the files from our local computer to the Azure storage hosting target destination. We are going to do this using the tool AzCopy, authenticate the commands with a Shared Access Signature (SAS) token, then create a static website with a custom domain all hosted within Azure Storage. It'll be a work of wonder.

I have mentioned a few times before in this book that the Azure command line interface, or CLI, is the preferred way to do a lot of tasks in Azure, as it is more concise, rarely changes, and is much more efficient. Bear in mind that automation is the ideal way to deploy anything in tech, as you remove the "human error" element. For now though, we don't cover automation, so the CLI is the best "human" interface. Make sense? A commonly used CLI tool for moving files is AzCopy, which is what we'll use to copy the static website files to Azure storage. If you haven't set up AzCopy on your current machine, see the appendices to get started. Once you have AzCopy ready to go, the first step is to create a blob container in the storage account, which will be the home for the static website. Open up your command shell of choice, such as the Windows Command Shell (cmd.exe), and start typing in this command, replacing the storage account name (banningstorage) with the one you are using. Don't press enter yet though.

```
C:/> azcopy make "https://banningstorage.blob.core.windows.net/$w
```

### **Windows Command Shell Tip**

If you're using a Windows Command Shell (cmd.exe), use double quotes ("") for your path/URL instead of single quotes (''). Most other command shells use a single quote for strings.

We are not quite done yet with the command. The keen reader may have noticed that if this was a valid command, then *anyone* could create a container in the storage account. We need to have authentication of the command to not get into trouble. There are two ways of doing this in Azure, either through *Azure Active Directory (AAD)* or using a *shared access signature (SAS) token*. We aren't covering AAD in this book, so in this case we will use a SAS token, which will do the job of authenticating AzCopy very nicely, just like one of those small umbrellas you get in your cocktail to make it just right.

### **8.2.2 SAS Tokens**

SAS tokens are often used in scenarios where you want to grant temporary, limited access to Azure resources to external users or applications, such as copying a website's files to its new home on Banning Books' storage.

Before we create a SAS (Shared Access Signature) token, let's answer the, I assume, obvious question: "what exactly is a SAS token?" A SAS token is a type of secure token that is used to grant access to specific Azure resources. It can be used as an alternative to using a username and password to authenticate requests to Azure resources, and provides a way to grant granular, time-limited access to specific resources or resource groups. You can grant very detailed authentication indeed.

A SAS token consists of a string that is appended to a resource URI (Uniform Resource Identifier), as shown in Figure 8-3. The string includes information about the access permissions granted by the SAS token, as well as the start and end time for the token's validity. When a request is made to the resource using the SAS token, Azure verifies the token and grants or denies access based on the information contained in the token.

**Figure 8-3: The SAS URI consists of the resource URI and then the SAS token as a query**

**parameter.**

`https://alpacamabagsstorage.blob.core.windows.net/?sv=2021-06-08&ss=bfqt&srt=s&sp=rwdlacupiytfx`

Storage Resource URI

SAS Token

`&se=2023-01-09T19:08:22Z&st=2023-01-09T11:08:22Z&spr=https`

SAS Token

`&sig=H8uSIwjy6uS4SKx9ERxazxadD6FOQtjmeRcAhM8bYyg%3D`

SAS Token

Now you know what it looks like, but how do we generate the SAS token itself? Open the Azure Portal, go to the storage account, in this case `banningstorage`, and then to the *Shared Access Signature* menu item. That will show a screen similar to Figure 8-4, which lays out all the parameters you have for creating a SAS token.

**Figure 8-4: Creating a new SAS token for the storage account using the Azure Portal**

**Shared access signature**

A shared access signature (SAS) is a URI that grants restricted access rights to Azure Storage resources. You can provide a shared access signature to clients who resources. By distributing a shared access signature URI to these clients, you grant them access to a resource for a specified period of time.

An account-level SAS can delegate access to multiple storage services (i.e. blob, file, queue, table). Note that stored access policies are currently not supported f

Learn more about creating an account SAS

**Allowed services**

- Blob
- File
- Queue
- Table

**Allowed resource types**

- Service
- Container
- Object

**Allowed permissions**

- Read
- Write
- Delete
- List
- Add
- Create
- Update
- Process
- Immutable storage
- Permanent delete

**Blob versioning permissions**

- Enables deletion of versions

**Allowed blob index permissions**

- Read/Write
- Filter

**Start and expiry date/time**

Start: 09/02/2023

End: 10/02/2023

(UTC+10:00) Canberra, Melbourne, Sydney

**Allowed IP addresses**

For example, 168.1.5.65 or 168.1.5.65-168.1.5.70

**Allowed protocols**

- HTTPS only
- HTTPS and HTTP

Let's understand some of those parameters you can set for a SAS token.

- **Allowed services:** You can choose which of the storage services to allow, being blob, file, queue, and table.
- **Allowed resource types:** Do you want to give access and allow manipulation of the entire storage account, containers (or folders), and/or just the objects themselves? You can get into trouble if you give too much access to resources, or limit what is possible if you give too little.
- **Allowed permissions:** These relate to resource types, for example create and delete aren't valid for the service resource type, but only for

container and object. If you select any that aren't valid, they will be ignored.

- **Start and expiry dates:** These are critical to set, as you can control down to the second when you will allow access and when not. Furthermore, you don't want a SAS token with no expiration. You always want a SAS token to expire to keep the storage account secure.
- **Allowed IP addresses:** Consider limiting the access by IP address if feasible. The more you can lock down access the better.

Once you have chosen the parameters you want for your SAS token, press the button that says *Generate SAS and connection string*, which will give you three values as output just below the button.

**Figure 8-5: The three outputs for a SAS token.**



Wow, that took a while to get to the part where you press *Enter* from the command above, just to create a container for our website, but we are almost there. We can now use the SAS token part of the output from Figure 8-5 and append it to the command

```
C:/> azcopy make "https://banningstorage.blob.core.windows.net/$w
```

Which then gives us the full command including SAS token

```
C:/> azcopy make "https://banningstorage.blob.core.windows.net/$w
```

Run this in your command shell, and you should see an output saying Successfully created the resource. We now have a container in blob storage called website, which is where we will upload the Banning Books website to. And finally, to upload the entire website to the storage account using *azcopy*, we use this command.

```
C:/> azcopy copy "C:\Repos\Banning Books\banning-books" "https://
```

This command copies all the files from the first path C:\Repos\Banning Books\banning-books to the second path, which is the full service URL from Figure 8-5. You will need to have the files from either the Banning Books website GitHub repository, or another static website of your choosing, in the first path of the command. Finally, we do the copy action recursively with the parameter --recursive=true, which means that any subfolders and files are copied across too.

Okay, time to execute!! Once the command has run (it is usually very quick) you now have all the files for the website in your Azure storage account, ready to become a website just like the cool files are.

### 8.2.3 \$web folder and static website

When it comes to static websites there is a very important detail I have glossed over until now. The folder we created and subsequently uploaded the website into is named \$web and that isn't a coincidence. When you enable a static website on a storage account, as we are about to do, the folder named \$web is where it will look. If you don't have a \$web folder, then one will be created for you, but your website files must live inside it to be hosted as a website. And it is worth reiterating at this point, that static websites can only host static content. It might be obvious, but we often relate websites to relying on a web server to render the pages and provide the data.

Once you have your files in the \$web folder it is time to activate this static

masterpiece! You will need two pieces of information to get it happening:

- an error document, or page, which is what users will see when they request a page that doesn't exist, or something else goes wrong. If you are using the Banning Books website provided, then that file is `error.html`
- **an index file, which is the static website itself.** If you are using the Banning Books website provided, then that file is `index.html`

Bundling all that together we use this Azure CLI command to enable the static website.

```
C:/> az storage blob service-properties update --account-name ban
```

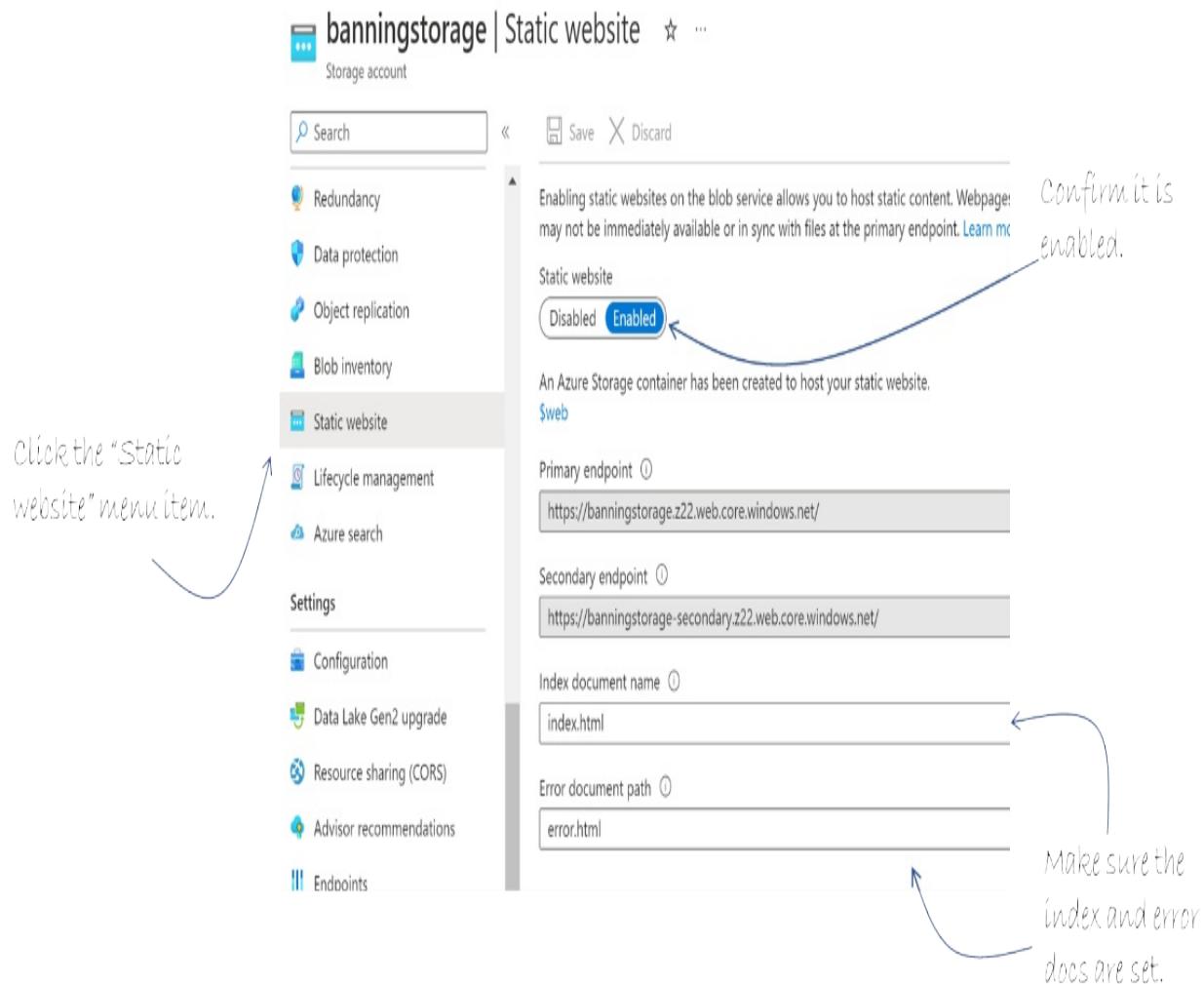
The storage and blob keywords denote that we are going to do something with the blob storage. The service-properties update keywords then specify that we want to update the service properties of the storage account, which is where the static website feature lives. We then explicitly choose the static-website service property to explicitly address this feature of the Azure storage account, and finally we choose the error page file name and the index (home page) file.

Once you successfully run the command, you will get a JSON output which outlines some of the current service properties for the storage account. One of the sections should read like this, confirming static website is enabled.

```
"staticWebsite": {  
    "defaultIndexDocumentPath": null,  
    "enabled": true,  
    "errorDocument_404Path": "error.html",  
    "indexDocument": "index.html"  
},
```

To confirm the static website is up and running, let's do two things. First, check the Azure Portal to make sure the static website is running(Figure 8-6). Go to the static website blade of your storage account, and make sure it is enabled as well as the index and error documents are configured.

**Figure 8-6: Confirm the static website and documents have been configured correctly.**



Second, let's check that the website works, and to do that we need the URL to enter into the browser. There are two ways to find the URL, one being looking it up in the Azure Portal, which is what Figure 8-6 shows as the *Primary endpoint*, in this case

<https://banningstorage.z22.web.core.windows.net/>. Another way to find that address is using the Azure CLI (of course). The following command will query the storage account for any primary endpoints for the web, which is another way of saying website address.

```
az storage account show -n banningstorage -g BanningRG --query
"primaryEndpoints.web" --output tsv
```

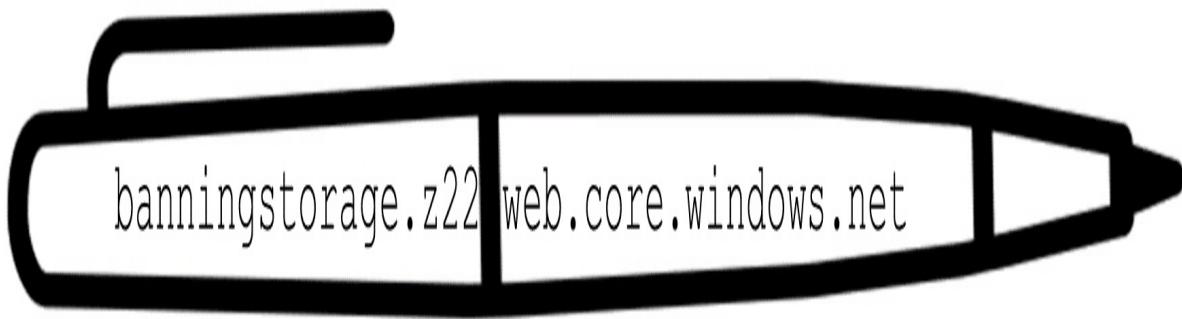
How cool is that! Without having to pay any extra, Banning Books now has a

simple static website live on the internet, and they don't have to worry about hosting, traffic, security, or any of the other maintenance tasks that often come with having your own website.

### 8.2.4 Custom domain

There is one more thing that we really need to address for the static website. That URL `banningstorage.z22.web.core.windows.net` is neither very easy to remember nor very *pretty*. Imagine putting that thing on a mug to promote your business or on a pen. That is going to be a looooong pen (Figure 8-7).

**Figure 8-7: This pen is too long. Shorten that URL!**



What we need is our own custom domain. Like `banningbooks.net` for example. While the approach to configuring a domain is very similar in Azure to other places, it is still a bit of work, which involves 4 steps.

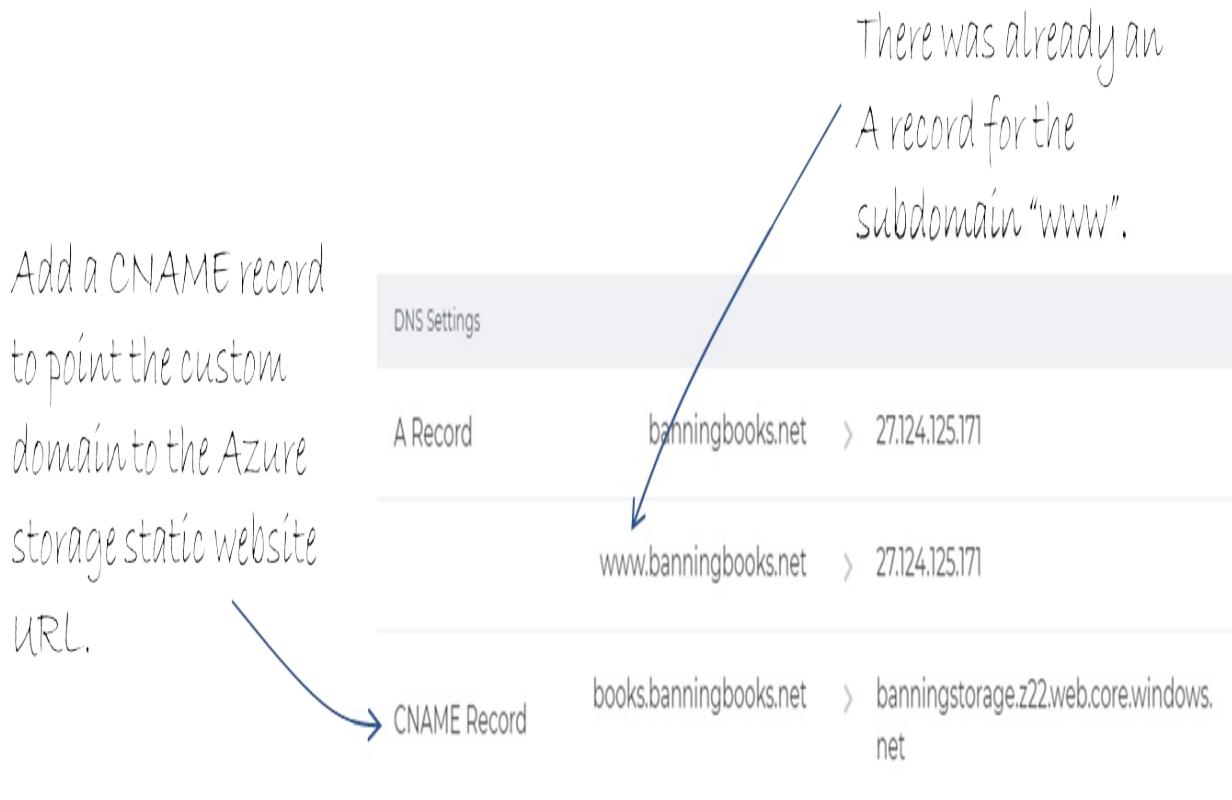
1. Acquire a custom domain.
2. Insert a CNAME record with your domain provider.
3. Connect the custom domain with your Azure storage account.
4. Test it. Then test it again.

Let's go through these steps together. I am not going to tell you a specific company where you can buy a domain. Either choose one you know or use your favorite search engine to search for a domain name provider.

Next, you need to create a CNAME, or Canonical Name, record for your

domain. This will point your new shiny domain to the static website domain Azure provided above. The CNAME record management section location will vary depending on which domain name provider you use. The important part is that you enter a subdomain such as *www* or *books*, making the full final domain *www.banningbooks.net* or *books.banningbooks.net* as shown in Figure 8-8.

**Figure 8-8: The CNAME record for the domain books.banningbooks.net**



The third step is to connect the new domain and CNAME record to the Azure storage and static website (Figure 8-9). On the networking menu for the storage account in the Azure portal, go to the custom domain tab.

**Figure 8-9: Setting up a custom domain for the storage account.**

On the custom domain tab fill in the custom domain name and

press "SAVE".

Go to the networking tab on the Azure storage account.

The screenshot shows the 'Networking' blade for a storage account named 'banningstorage'. On the left, there's a sidebar with 'Containers', 'File shares', 'Queues', 'Tables', and 'Security + networking' (which is selected). Under 'Security + networking', there are options for 'Networking', 'Azure CDN', 'Access keys', 'Shared access signature', 'Encryption', and 'Microsoft Defender for Cloud'. The main area is titled 'banningstorage | Networking' and contains sections for 'Firewalls and virtual networks' and 'Private endpoint connections'. A blue arrow points from the handwritten note 'Go to the networking tab on the Azure storage account.' to the 'Networking' button in the sidebar. Another blue arrow points from the handwritten note 'On the custom domain tab fill in the custom domain name and press "SAVE"' to the 'Custom domain' link in the top right. A third blue arrow points from the handwritten note 'Enter your new domain name, then click Save and you are done.' to the 'Domain name' input field, which contains 'books.banningbooks.net'. Below the input field is a checkbox for 'Use indirect CNAME validation'.

Enter your new domain name, then click *Save* and you are done. Well, you are done if step 4, testing, is successful. You can now access your static website using your new domain.

#### HTTPS vs HTTP for static websites

You may get an error accessing the website using http, if the storage account is configured for https/SSL only. You can add “https” in front of your custom domain, which will warn you of continuing. For testing purposes, you can use this method, but for production it is advised to obtain an SSL certificate or

utilize a service such as Azure CDN to secure the site traffic adequately.

With Banning Books' website up and running it is time to consider how to best protect the data on the storage account.

## 8.3 Data protection

Banning Books stores all of their publications digitally in Azure storage, so of course they are keen to understand the ways the data can be protected. Although, the obvious reason for protecting data is to prevent unauthorized access to the books in the inventory, there are other important reasons to protect data in Azure storage.

- Customer trust: When you purchase digital literature from Banning Books you expect personal information and purchase history, as well as the products themselves, to be kept confidential. By ensuring data protection, we can maintain customer trust and loyalty.
- Intellectual property protection: Digital literature represents the creative work and intellectual property of authors and publishers. Ensuring the protection of these assets is crucial to maintain the trust of content creators and safeguard their rights.
- Business continuity: Data protection helps Banning Books to minimize downtime and maintain business operations in case of data loss or corruption due to technical issues or cyber-attacks.

When it comes to the job of implementing the protecting of their data, Banning Books wants to know what is possible with Azure storage. Which means, diving into the tools again. First, however, let me just clarify that your data in an Azure Storage account is never unprotected or unencrypted. *All* data is encrypted and decrypted transparently using 256-bit AES encryption, one of the strongest block ciphers available, and is FIPS 140-2 compliant, a U.S. government computer security standard used to approve cryptographic modules. In other words, your data stored physically on the Azure hard drives is *very* secure, and at no extra cost. The main issue with data protection is usually when it is accessed over the internet.

### 8.3.1 Blob version tracking

Sometimes files get changed, deleted or overwritten. We've all been there, wondering how you are going to cancel your weekend plans so no one notices your whoopsie by Monday morning. Oh, is that just me? Anyway, version tracking in Azure blob Storage is a powerful feature that allows users to preserve, retrieve, and restore previous versions of a blob. This functionality is particularly useful in scenarios where accidental data overwriting, deletion, or corruption occurs, as it enables the recovery of the original data. With version tracking enabled, each time a blob is modified or deleted, Azure blob Storage automatically creates and maintains a new version of the blob. These versions are accessible and can be managed alongside the current version of the blob, ensuring that historical data is always available for recovery when needed. Nifty!

To enable versioning for Banning Books and their blob container of underground banned books, use this CLI command.

```
C:/> az storage account blob-service-properties update --account-
```

There are two parts of this command that are new and which you should take note of. First, we are updating the properties of the storage account using blob-service-properties update which is how you update specific properties of an existing Azure storage account. Second, we define which property we want to update, in this case --enable-versioning, then set it to true. Once this command is executed, versioning will be enabled for all blob containers within the storage account.

After version tracking is enabled, you can interact with and manage blob versions using various Azure CLI commands. For example, you can list all versions of a specific blob by running:

```
C:/> az storage blob version list --account-name banningstorage -
```

Replace <ContainerName> with the name of the blob container and <BlobName> with the name of the blob you wish to view versions for. This command will display a table containing details of all versions of the specified blob, including their version ID, creation time, and content length.

Tracking is simple to set up and can get you out of trouble more than once.

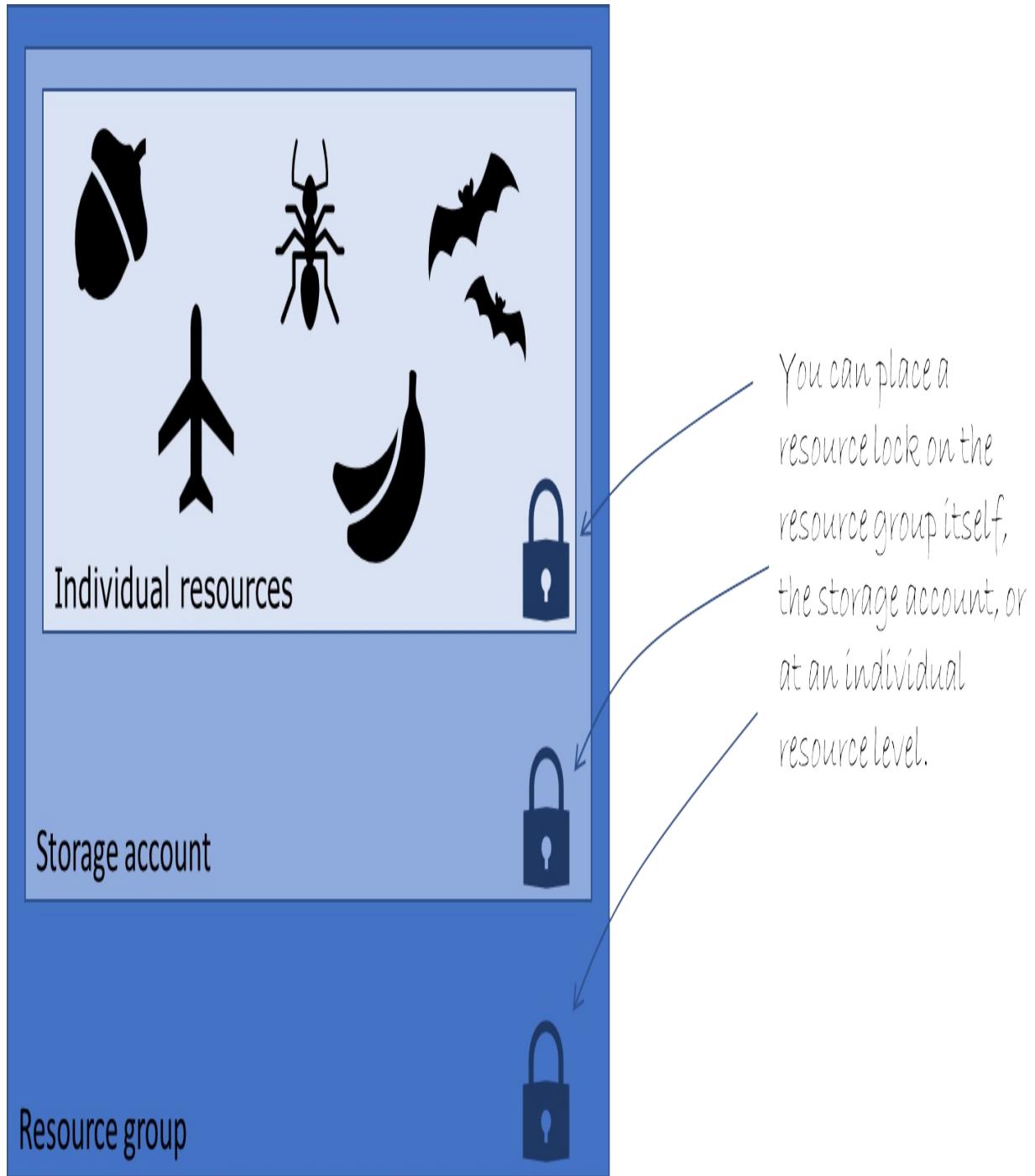
This could be when someone accidentally deletes or modifies a file, and you can quickly restore the file and its specific version. Perhaps you have an audit request for detailed logs and metrics of all the operations on your data, or maybe you have a corrupted file you want to quickly restore to a previous version.

Let's shift our focus to another valuable feature that can further protect your resources: implementing a Resource Management lock in Azure.

### **8.3.2 Azure storage account lock**

Azure Resource Management (ARM) locks provide an extra layer of protection for resources in an Azure storage account by preventing accidental modifications or deletions. These locks can be applied at different levels, such as the resource group, storage account, or individual resources within the storage account (Figure 8-10).

**Figure 8-10: Resource locks can be placed on the resource group, storage account or individual objects.**



There are two types of ARM locks: 'CanNotDelete' and 'ReadOnly.'

- The 'CanNotDelete' lock ensures that the locked resource cannot be deleted but allows for modifications.
- The 'ReadOnly' lock prevents both deletion and modification of the

locked resource.

To apply an ARM lock to the 'banningstorage' storage account, you can use the Azure Command-Line Interface (CLI) using the following command.

```
C:/> az lock create --name banningLock --resource-group BanningRG
```

This command will create a lock of type `CanNotDelete` for the *banningstorage* storage account, restricting any actions to not deleting the storage account. To guide us into the last data protection measure let me turn to poetry from years gone by.

From locks of your data, we now venture away,

To soft delete, our next stop on this data protection bay,

Like a digital safety net, it'll catch what you've tossed,

So even if you hit 'delete,' your data's not lost!

### 8.3.3 Soft delete

Azure Soft Delete is a valuable data protection feature that allows users to recover blobs, blob snapshots, and blob versioning data that has been accidentally or intentionally deleted. When Soft Delete is enabled, deleted data is marked as soft-deleted and retained in the system for a specified retention period. During this period, soft-deleted data is invisible to users but can be recovered if necessary. This feature provides a safety net against data loss due to accidental deletions, offering additional peace of mind and ensuring business continuity.

To enable Soft Delete for the 'banningstorage' storage account use this CLI command

```
C:/> az storage account blob-service-properties update --account-
```

Look familiar? We are using the same update command to the `blob-service-properties` to change the soft delete or delete retention as the parameter is called. We then also set the retention period to 7 days, which

means any data that is deleted will be kept for 7 days before it is permanently deleted.

If you do accidentally (or otherwise) delete a blob and need to restore it, you can use the Azure CLI command:

```
C:/> az storage blob undelete --account-name banningstorage --con
```

Replace <ContainerName> with the name of the container containing the soft-deleted blob and <BlobName> with the name of the soft-deleted blob to be restored. This command will undelete the specified blob, making it accessible again.

Azure Blob Storage offers a comprehensive suite of data protection features for Banning Books, including blob version tracking, Azure Resource Management (ARM) locks, and soft delete. Blob version tracking preserves historical versions of blobs, enabling easy recovery of previous data states in case of accidental modifications or deletions. ARM locks provide an extra layer of security by preventing unauthorized or inadvertent changes and deletions of critical resources at various levels within the storage account. Soft delete acts as a safety net, allowing users to recover deleted blobs, snapshots, and versioning data within a specified retention period. Together, these features create a robust data protection strategy, ensuring the integrity, security, and availability of data stored in Azure Blob Storage.

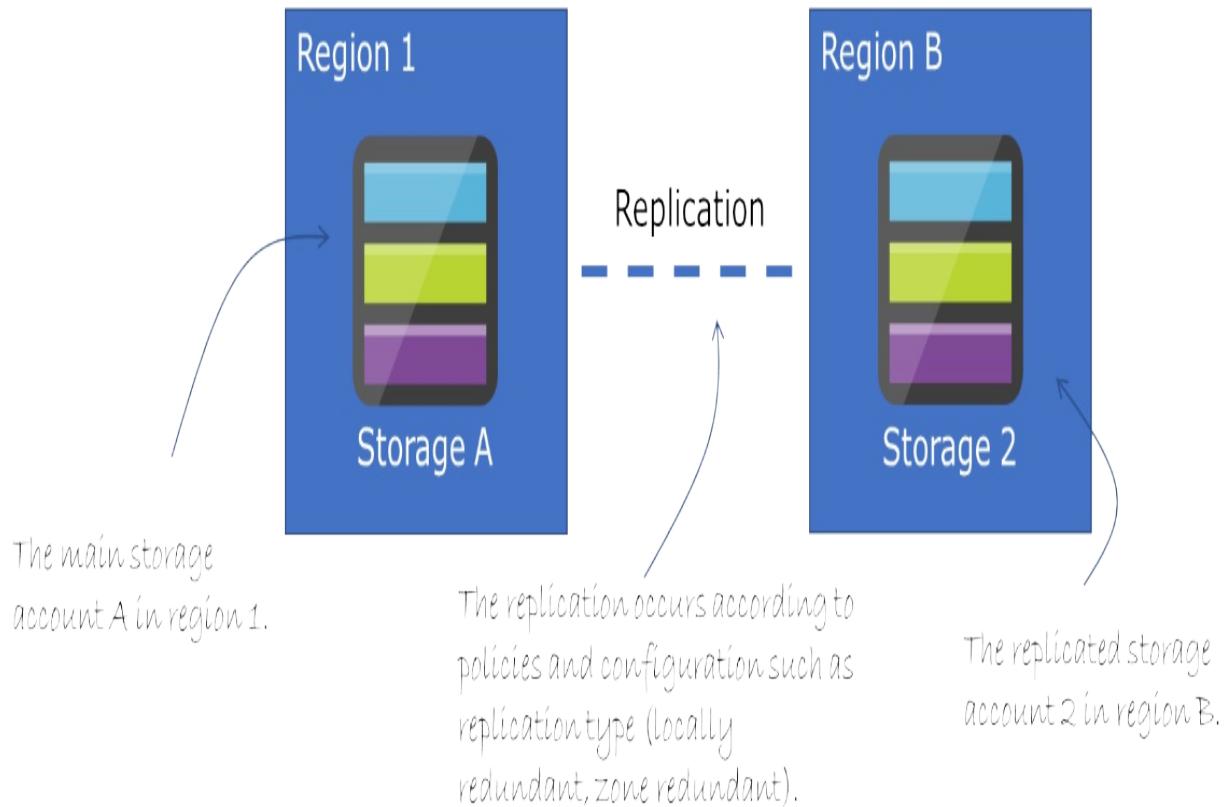
## 8.4 Object replication

Banning Books wants to ensure their data is both easy to access for customers, as well as protect against disaster and improve data resiliency. This will allow them to maintain a high-quality user experience and ensure the reliability and availability of their digital literature in Azure Storage.

Azure Storage Object Replication is a feature that enables you to replicate your data between different Azure storage accounts, regions, or even between Azure and an on-premises storage location as shown in Figure 8-11. This feature helps you to protect your data against regional outages or disasters by keeping a copy of your data in another location. With object replication, you

can configure replication policies to specify which containers or blobs should be replicated, and how often replication should occur. You can also choose replication types, such as locally redundant storage (LRS), zone-redundant storage (ZRS), geo-redundant storage (GRS), or read-access geo-redundant storage (RA-GRS), depending on your business needs.

**Figure 8-11: Azure storage object replication between two storage accounts in different regions.**



Before we dive into the potential benefits of object replication, let's configure it in practice using the Azure CLI. As you might have guessed, we need a second storage account as the destination for the replication, so let's create that first.

```
C:/> az storage account create --name banningstoragereplica --res
```

This creates the Azure storage account `banningstoragereplica`, but we also need a container in that account to target. I'll create one called `booksreplica`, so you can do the same or another of your choosing.

There are a couple of properties we need to make sure are enabled on the source and destination storage accounts, in order to enable object replication. First, for the source account, we need versioning of the data, as well as enabling the change feed that keeps track of the changes made to the files.

```
C:/> az storage account blob-service-properties update --resource
```

The destination storage account then needs to have versioning enabled so the replication knows if a version of a file is up to date or not.

```
C:/> az storage account blob-service-properties update --resource
```

Then it is time to create the object replication policy which will manage the replication. This involves specifying the source and destination storage accounts and containers. Use this last CLI command to set up the replication between Banning Book's two storage accounts.

```
C:/> az storage account or-policy create --account-name banningst
```

The `or-policy` command is short for object replication, and the rest of the parameters define the source and destination storage accounts and containers. One point worth noting is that the policy is created for the destination storage account, not the source.

**Note:**

It is possible to set up a policy for object replication across Azure tenants too, but you will need add the parameter and you will need to supply a full resource ID such as  
`/subscriptions/<subscriptionId>/resourceGroups/<resource-group>/providers/Microsoft.Storage/storageAccounts/<storage-account>` rather than just the storage account name.

Now that object replication is set up for Banning Books, let's explore for a moment a couple of the immediate benefits it brings.

### **8.4.1 Data Redundancy and Availability**

By replicating digital literature across multiple storage accounts, Banning

Books can maintain backup copies of their content. This strategy ensures that if one storage account experiences an issue, such as a data center outage or hardware failure, the content remains available in other locations, minimizing downtime and ensuring business continuity. In our example above the two storage accounts used for object replication are in the same region, but they don't have to be. Object Replication also plays a crucial role in disaster recovery planning. In the event of a natural disaster, cyberattack, or technical issue affecting one region, Banning Books can quickly recover their data from a replicated storage account located in another region, reducing data loss and recovery time.

Additionally, object replication is performed asynchronously, which means that replication occurs in near-real-time without impacting the performance of your primary storage account.

#### **8.4.2 Geo-Distribution and Latency Reduction**

Geo-distribution in Azure Storage is an essential feature for ensuring data redundancy, accessibility, and reduced latency across geographically separate regions. When using object replication, data stored in one container can be replicated to another container in a different storage account, even across different regions. This not only provides an additional layer of protection against data loss due to regional outages, hardware failures, or system errors but also offers performance benefits to users accessing the data from different locations. An added benefit is the latency reduction that comes with object replication. By replicating data to different regions, users can access the data from the nearest storage account, leading to faster response times and improved user experience. A better way of implementing this is often using a content delivery network, which we will cover later in the book.

### **8.5 Lifecycle management**

Banning Books, as a digital literature provider, wants to optimize storage costs, ensure efficient utilization of resources, and maintain data compliance. The lifecycle management tools for Azure storage can do just that. By implementing custom-defined rules, Banning Books can automatically transition data between different access tiers based on usage patterns, delete

obsolete or outdated data, and manage blob versions and snapshots efficiently. This not only helps in reducing storage costs but also ensures that the company adheres to data retention policies and regulations.

### 8.5.1 Tier transitioning

Azure Storage Lifecycle Management rules enable seamless transitions of data blobs between different access tiers—Hot, Cool, and Archive—based on the defined policies. This tier-based approach provides significant cost benefits while ensuring efficient use of storage resources.

As we learnt in chapter 5 the Hot access tier is designed for frequently accessed data and provides low latency and high throughput, albeit at a higher storage cost. As data usage patterns change over time, transitioning the data to the Cool access tier makes sense for infrequently accessed data, as it offers lower storage costs compared to the Hot tier. The trade-off, as shown in Figure 8-12, is slightly higher access costs and latency. Finally, for long-term storage and rarely accessed data, the Archive tier offers the lowest storage costs, with the understanding that data retrieval will take longer and incur higher access costs. By creating lifecycle management rules that automatically transition data blobs between these access tiers, Banning Books can optimize storage costs and maintain high performance for their customers.

**Figure 8-12: The trade-off -The access costs go up for cold and archive tiers, where the storage costs go up for cold and hot tiers.**



To create a lifecycle management rule using Azure CLI that transitions data blobs to cool if it hasn't been modified in 30 days, move all the way to archive if it's not modified for 90 days, and delete after a year without modification, you can use the following policy represented in JSON.

```
{  
  "rules": [  
    {  
      "name": "TransitionHotToColdToArchive",  
      "enabled": true,  
      "type": "Lifecycle",  
      "definition": {  
        "actions": {  
          "baseBlob": {  
            "tierToCool": {  
              "daysAfterModificationGreater Than": 30  
            }  
          }  
        }  
      }  
    }  
  ]  
}
```

```

        },
        "tierToArchive": {
            "daysAfterModificationGreaterThan": 90
        },
        "delete": {
            "daysAfterModificationGreaterThan": 365
        }
    }
},
"filters": {
    "blobTypes": [
        "blockBlob"
    ]
}
}
]
}
}

```

There is a lot going on here, but using JSON makes it very readable and defined. In fact, almost all templates, policies, definitions, and anything else on Azure can be defined in JSON. Save the JSON in a file called `policy.json`. For this specific policy let's go through each part of the definition.

- **"rules"**: This is the top-level array that contains all the rules in the policy. Each rule is represented as an object within the array. So, yes, there can be more than one rule in the policy.
- **"name": "TransitionHotToColdToArchive"**: This is the unique name assigned to the rule, which helps in identifying and managing the rule.
- **"enabled": true**: This is a boolean value that indicates whether the rule is active. If set to true, the rule will be executed by the Azure Storage Lifecycle Management system. If set to false, the rule will be ignored.
- **"type": "Lifecycle"**: This specifies the type of rule. In this case, it's a *"Lifecycle"* rule, which means it manages the lifecycle of the blobs based on the specified actions.
- **"definition"**: This object contains the actions and filters that define the rule's behavior.
- **"actions"**: This object contains actions to be performed on the baseBlob.
- **"baseBlob"**: This object contains the actions to be performed on the base blobs, such as tier transitions and deletion.

- **"tierToCool": {"daysAfterModificationGreaterThan": 30}**: This action specifies that the blob should transition from the Hot tier to the Cool tier if it hasn't been modified for more than 30 days.
- **"tierToArchive": {"daysAfterModificationGreaterThan": 90}**: This action specifies that the blob should transition from the Cool tier to the Archive tier if it hasn't been modified for more than 90 days.
- **"delete": {"daysAfterModificationGreaterThan": 365}**: This action specifies that the blob should be deleted if it hasn't been modified for more than 365 days.
- **"filters"**: This object contains filters that determine which blobs the rule should apply to.
- **"blobTypes": ["blockBlob"]**: This array specifies the types of blobs the rule should apply to. In this case, the rule applies to "blockBlob" type blobs.

You might ask, “don’t we need to define when the data is being moved up to hot again?” No, Azure Storage will do this automatically when you start using the files, whether that is read or modify. Also, I added the *delete* part to the policy to show a full flow of lifecycle management. However, as Banning Books are unlikely to want to delete their books, this part can be removed.

With the policy we need to now implement it using the Azure CLI using this command

```
C:/> az storage account management-policy create --account-name b
```

We are creating a management policy on the storage account, using the file we created before defining the policy. Depending on which implementation of the Azure CLI (stand alone, in Portal etc.) you can upload the file the CLI instance and execute it from there.

The implementation of a lifecycle policy in Banning Books' Azure Storage can have a positive impact on the business by optimizing storage costs, enhancing data management, and ensuring compliance with data retention policies. By automatically transitioning data between Hot, Cool, and Archive tiers based on access patterns and age, Banning Books can allocate resources more efficiently, resulting in reduced storage costs. Additionally, if the automatic deletion of outdated or obsolete data is used, it ensures that the

storage account remains organized and clutter-free, reducing the time and effort required for manual data management. Lastly, by adhering to data retention policies and regulations through a lifecycle policy, Banning Books can maintain compliance, build trust with its customers, and mitigate potential legal risks. Overall, adopting a lifecycle policy will streamline Banning Books' storage management while enhancing cost-effectiveness and compliance. That is pretty sweet from something you only have to define and implement once.

## 8.5.2 Cleaning up

Even though you just saw that files can be deleted using a lifecycle management policy, there are still more areas we can clean up and save storage space, hence cold hard cash. Earlier we set up version tracking for the blobs, but that means we are potentially keeping many more versions of a file than we need to. Well, color me purple, there is a service for that too. In fact, it is another lifecycle management policy defined in JSON.

```
{
  "rules": [
    {
      "name": "CleanUpOldBlobVersions",
      "enabled": true,
      "type": "Lifecycle",
      "definition": {
        "actions": {
          "version": {
            "delete": {
              "daysAfterCreationGreaterThanOrEqual": 90
            }
          }
        },
        "filters": {
          "blobTypes": [
            "blockBlob"
          ]
        }
      }
    }
  ]
}
```

You might recognize quite a few sections from the first lifecycle policy we created. So let's first go through the new parts.

- **"actions"**: This object contains actions to be performed on the versioned blobs.
- **"version"**: This object contains the actions to be performed on the blob versions, such as deletion.
- **"delete": {"daysAfterCreationGreater Than": 90}**: This action specifies that the older versions of the blobs should be deleted if they were created more than 90 days ago.

See how much easier and less frightening it was the second time? Well, I hope that was the case. While the word “policy” might make you fall asleep, on Azure it is a word that saves you time and money. Policies are your friends. Save the JSON into a file called `deletepolicy.json`, and then we can almost reuse the CLI command from before.

```
C:/> az storage account management-policy create --account-name b
```

However, when you do that, you overwrite the existing lifecycle policy we created, so instead we need to combine the two JSON files into a single file. This means copying everything inside the `rules` section and placing it into the first file's `rules` section.

```
{
  "rules": [
    {
      "name": "TransitionHotToColdToArchive",
      "enabled": true,
      "type": "Lifecycle",
      "definition": {
        "actions": {
          "baseBlob": {
            "tierToCool": {
              "daysAfterModificationGreater Than": 30
            },
            "tierToArchive": {
              "daysAfterModificationGreater Than": 90
            },
            "delete": {
              "daysAfterModificationGreater Than": 365
            }
          }
        }
      }
    }
  ]
}
```

```
        }
    },
    "filters": {
        "blobTypes": [
            "blockBlob"
        ]
    }
},
{
    "name": "CleanUpOldBlobVersions",
    "enabled": true,
    "type": "Lifecycle",
    "definition": {
        "actions": {
            "version": {
                "delete": {
                    "daysAfterCreationGreaterThanOrEqual": 90
                }
            }
        },
        "filters": {
            "blobTypes": [
                "blockBlob"
            ]
        }
    }
}
]
```

Save this JSON as policies.json, then use the CLI command from before again with the new file.

```
C:/> az storage account management-policy create --account-name b
```

Finally, to confirm we have the policies in place, let's get the CLI to show the current lifecycle management policies for the storage account.

```
C:/> az storage account management-policy show --account-name ban
```

That is it. Banning Books now has a much more efficient storage account on Azure, getting the most value for their money with the least effort.

## 8.6 Summary

- You can host a simple static website using Azure storage by uploading CSS, HTML and JavaScript files to the \$web folder.
- AzCopy is a commonly used tool for moving files from on-premises to Azure storage accounts, or between Azure storage accounts.
- You can authenticate AzCopy commands with either Azure Active Directory or using a Shared Access Signature (SAS) token.
- A SAS token is a type of secure token that is used to grant access to specific Azure resources. It can be used as an alternative to using a username and password to authenticate requests to Azure resources, and provides a way to grant granular, time-limited access to specific resources or resource groups.
- A SAS token can be configured and generated using the SAS token menu for the storage account in the Azure portal.
- Files for a static website on Azure storage must reside in the \$web folder.
- Protecting data isn't just to keep out unwanted people and processes, but also to increase customer trust, protect intellectual property and ensure business continuity.
- With Blob version tracking enabled, each time a blob is modified or deleted, Azure blob Storage automatically creates and maintains a new version of the blob.
- Azure locks can be applied to an entire resource group, a storage account, or an individual object, such as a blob.
- There are two types of ARM locks: 'CanNotDelete' and 'ReadOnly.' The 'CanNotDelete' lock ensures that the locked resource cannot be deleted but allows for modifications. The 'ReadOnly' lock prevents both deletion and modification of the locked resource.
- Versioning of the storage account has to be enabled to use object replication.
- Data transitions automatically between hot, cold and archive tiers according to your storage policies.
- Use lifecycle management policies to ensure data is moved to the appropriate tier or deleted to save costs and optimize the storage.
- Policies are defined in JSON format, which also allows the Azure CLI to implement them using plain text files.

- Lifecycle management policies are your best friend when it comes to managing versioning, data retention, object replication, and sanity.