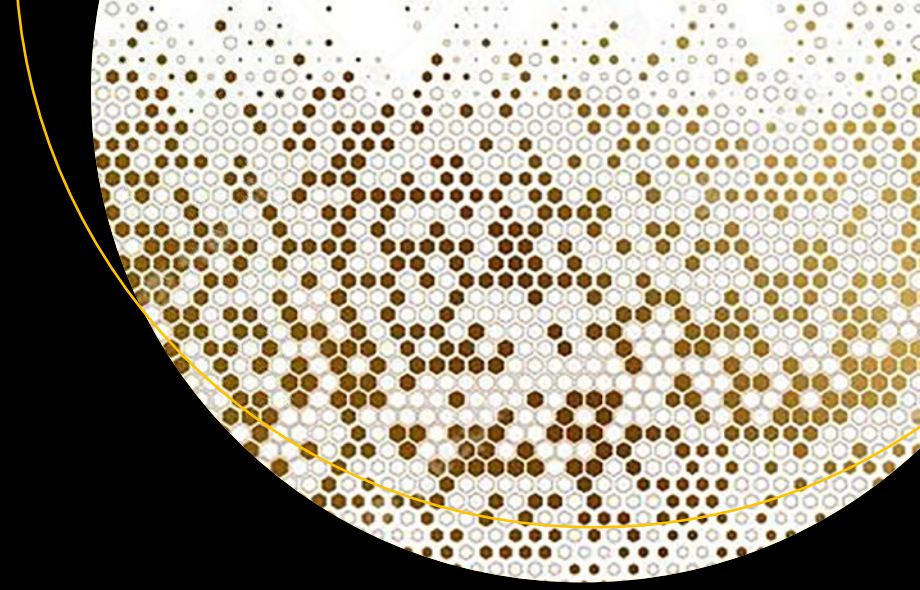

Pro Serverless Data Handling with Microsoft Azure

Architecting ETL and Data-Driven Applications in the Cloud

Benjamin Kettner
Frank Geisler

Apress®

Pro Serverless Data Handling with Microsoft Azure

Architecting ETL and Data-Driven Applications in the Cloud

Benjamin Kettner
Frank Geisler

Apress®

Pro Serverless Data Handling with Microsoft Azure: Architecting ETL and Data-Driven Applications in the Cloud

Benjamin Kettner
Berlin, Germany

Frank Geisler
Lüdinghausen, Germany

ISBN-13 (pbk): 978-1-4842-8066-9
ISBN-13 (electronic): 978-1-4842-8067-6
https://doi.org/10.1007/978-1-4842-8067-6

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Jonathan Gennick
Development Editor: Laura Berendson
Coordinating Editor: Jill Balzano
Copy Editor: April Rondeau

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science + Business Media LLC, 1 New York Plaza, Suite 4600, New York, NY 10004. Phone 1-800-SPRINGER, fax (201) 348-4505, email orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please email booktranslations@springernature.com; for reprint, paperback, or audio rights, please email bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub at https://github.com/Apress/pro-serverless-data-handling-w-microsoft-azure.

Printed on acid-free paper

This book is dedicated to all who—like us—enjoy learning new things. Stay curious. To our families, friends, and colleagues.

Table of Contents

About the Authors

Dr. Benjamin Kettner is co-founder and CTO of ML!PA Consulting GmbH, a company that focuses on the end-to-end delivery of Internet of Things (IoT) projects. Since 2020, he has been a Microsoft Data Platform MVP and a Friend of Red Gate. Together with Frank Geisler, he wrote the German books *Azure für Dummies* and *Docker für Dummies*, before they started the adventure of writing *Pro Serverless Data Handling with Microsoft Azure* together. He received his doctorate in applied mathematics at the Free University of Berlin in 2012. At the time of earning his doctorate, he was a member of the DFG Research Center Matheon-Mathematics for Key Technologies, as well as a member of the Computational Nano Optics group at the Zuse Institute Berlin. In his free time, Ben likes to listen to (and play) metal music, make ham and bacon, and play Xbox games with Frank, or spend time with his daughter, Pia.

Frank Geisler is owner and CEO of GDS Business Intelligence GmbH, a Microsoft Gold Partner in six categories that is located in Lüdinghausen, in the lovely Münsterland (Germany). He is a Long-Term Data Platform MVP, MCT, MCSE-Business Intelligence, MCSE-Data Platform, MCSE-Azure Solutions Architect, and DevOps Engineer Expert. In his job he is building business intelligence systems based on Microsoft technologies, mainly on SQL Server and Microsoft Azure. He also has a strong focus on database DevOps. Frank is a frequent speaker at international events and usergroups. In his free time, Frank likes to play with his daughter, Elisa, cook or grill some exotic dishes (and build up his skills very much though the pandemic), listen to heavy metal, and watch movies or play Xbox games with Ben and Dirk.

About the Technical Reviewer

Sascha Lorenz has worked with Microsoft SQL Server and efficiently automated database-related tasks for several decades. Still with a developer soul, he is now responsible for developing the PSG tool stack, which his colleagues and customers use to analyze and optimize database servers worldwide. Furthermore, he shares his experience and knowledge in numerous workshops and trainings and at community events.

Acknowledgments

There are many people we have met along our way, many who have helped us grow, and many who have taken part. Any listing of names will always have gaps, but nevertheless…

Frank: The first and biggest thank you has to go to Ben, who not only brought me aboard this project but who has also been a best friend and a very strong supporter whatever came my way professionally or privately though all the years. I would like to thank all my colleagues at GDS Business Intelligence GmbH, especially Melanie Vogel and Timur Dudhasch, who supported me through the phase of writing this book. I would also like to thank Aurelia, the mother of my wonderful little girl, Elisa, and of course Elisa (4), who brightens up my days and helps me to discover all the known things in a new light. Someday you will be able to read this. Your dad is very proud of you.

Ben: I would first like to express gratitude for the help and support of my colleagues at ML!PA. When my private life was turned upside down, they stepped up and enabled me to continue doing what I love, and supported me. Furthermore, I would like to thank Freya, who finds the strength to keep fighting. And, most important, our daughter, Pia, who is a constant source of strength and joy in my life and the best thing that has ever happened to me. I would also like to thank my parents, who were always there for me when I was struggling. Last but not least, I want to thank my special friend Frank, who picked me up so many times when I was feeling down, who was always there for me without hesitation, and who has helped me improve in so many areas.

Introduction

Congratulations on picking up this book. That you started flipping the pages means that you are interested in utilizing the power of cloud computing to meet your needs. We should warn you that, even though we are aware of the marketing hype surrounding serverless computing, this book is not a marketing book. It is more of a guide to utilizing serverless services in Microsoft Azure to meet your needs regarding the scalability, reliability, and simplicity of the design of your solution.

We are aware that serverless technology is not the means by which to solve all problems in any given cloud architecture, and of course some of the examples in this book will be overdoing the "serverless" aspect of the architectures, but the purpose here is to give you an idea of where you could use serverless Azure services to solve certain scenarios.

This book is aimed at you as an **IT manager** with technical interest who would like to understand where serverless services might be an option for implementing a part of your solution. And it is aimed at you as an **IT developer** who would like to write less code for applications that scale well and are well behaved with respect to problems in the infrastructure. It also is aimed at you as a **solution architect** if you would like to get some ideas for how to incorporate new services and concepts into your solution. Finally, it is aimed at you as an **IT professional** who would like to broaden your knowledge of cloud computing.

This book will first give you a basic introduction to Microsoft Azure and serverless computing and explain the basics that will be used throughout all other chapters of the book (**Chapters 1–3**).

The second part of the book (**Chapters 4–9**) will introduce the most important services in Microsoft Azure that offer a serverless tier. These chapters will focus on the serverless tiers and introduce some of the services' functionalities. You will also be provided with examples of how to create each service.

The third part of this book (**Chapters 10–13**) will introduce several architectural concepts and discuss how using serverless services can contribute to each of these concepts. You will see some architectural drafts here—both as abstract concepts and with actual Azure services—that you can use as a basis when building or enhancing

your own architecture. These chapters are primarily aimed at explaining the interaction between services and how to bring the pieces together into concise and working examples.

In the last part of the book (**Chapters 14–17**), we will give you a head start for implementing the concepts learned so far. We will introduce some tools and helpers that you can use to interact with the services when implementing a solution, some more-detailed patterns to highlight questions you should answer when designing your solution, and finally an architectural blueprint for a theoretical solution that brings together all the topics learned thoughout the book.

When reading this book, you will find many examples to explain what is happening. The code for all examples will be available on our GitHub page at `https://github.com/Serverless-Data-Handling/`. You can download it to save on all the editing. As the Azure CLI is available on Linux, Mac, and Windows, you should be able to run all examples on the hardware of your choice. If any examples do not work, we would like to encourage you to add a comment to the GitHub repository so that we can help you find the issue.

We would now like to encourage you to dive into the world of serverless computing. We hope that you will find this book a good companion when designing your next solution, enhancing your current solutions, or coming up with new ideas that even we did not think of yet.

PART I

The Basics

CHAPTER 1

Azure Basics

As of today, cloud technology—and especially *Azure*, Microsoft's implementation of a stack of cloud services—has become one of the main factors that drive the wheel of innovation, which just spins faster and faster. Cloud technology is so much more than someone else's computers. A mature cloud platform like Azure provides automation that keeps you from the daunting tasks that every IT professional knows and dreads lets you focus on what really matters: building IT systems that add value to your company.

At the time of writing, Azure consists of over 700 different services, spanning from simple things like storage or virtual machines to complex ones that implement things like image recognition or full-fledged data warehouses. This book concentrates on the services you can use to process your data in Azure. One of the great benefits that cloud computing provides is that you can build your systems on services that you do not have to maintain yourself. The cloud provider maintains the underlying infrastructure of these services. You can focus on your development instead of applying updates or tuning the performance of the hardware your system is based on. And that is what this book is all about: showing you how to build data-driven systems that do not involve infrastructure components.

The Different Cloud Service Models

A discussion about cloud technology is not complete without taking a look at the three service models for cloud offerings: *Infrastructure as a Service (IaaS)*, *Platform as a Service (PaaS)*, and *Software as a Service (SaaS)*. The following sections explain what these cloud service models are all about.

B. Kettner and F. Geisler, *Pro Serverless Data Handling with Microsoft Azure*,
https://doi.org/10.1007/978-1-4842-8067-6_1

Infrastructure as a Service (IaaS)

Infrastructure as a Service (IaaS) involves good old virtual machines, this time run in an Azure data center. IaaS is the easiest way to move to the cloud. You just shift your on-premises machines to the cloud and execute them there. While being the most versatile approach—you can run literally any software on a virtual machine in Azure, no matter how old or crappy it is—it is also the cloud service model that differs the least from your on-premises environment. This model does not help you to reduce the effort required to run and maintain your systems. You are responsible for installing updates, backing up the machine, and maintaining performance. There are features in Azure that support these tasks, but you are responsible for them.

Often IaaS is chosen as the first step toward implementing a cloud architecture because it is a quick win, and the learning curve is not so steep. If you run a virtual machine in the cloud, it is nevertheless a virtual machine, and most of the stuff you do is the same whether in Azure or on-premises, so you are on common ground. Besides virtual machines, Azure provides more services that can be seen as IaaS—all the components you need to connect and secure your virtual machines to the outside world, like virtual networks, VPN gateways, and alike.

The advantage of using IaaS over running virtual machines on-premises is that you are not responsible for the physical infrastructure your virtual machines are run on. This is the responsibility of the cloud provider. Another advantage is that you are always able to scale your systems because of the vast number of physical servers that Microsoft provides in its data centers.

You should refrain from using IaaS whenever possible because a modern cloud architecture should be based upon a different cloud service model.

Platform as a Service (PaaS)

Platform as a Service (PaaS) is where real cloud architecture starts. The advantage of PaaS is that everything that has to do with infrastructure, like providing virtual machines, running them, patching them, and securing them, is handled by your cloud provider. By using PaaS, you can focus on developing your application without the distraction of keeping all the nuts and bolts running.

A good example of a PaaS service is a database like Azure SQL Database or Cosmos DB. If you initiate such a database, all infrastructural parts will be handled behind the scenes by Microsoft. You can focus on creating the schema of the database—which consists of tables, stored procedures, functions, and everything else—without fussing over the underlying servers.

Compared to IaaS, PaaS lacks some flexibility. There is no way in PaaS to access the infrastructure components, since you are not responsible for them. Not all the functions that are available if you run a SQL server on a virtual machine are also available with Azure SQL Database. A good example of this is the absence of `xp_cmdshell` in Azure SQL Database, a stored procedure that can execute commands on the operating system level. Sometimes databases are called Database as a Service (DaaS), and DaaS is considered a subtype of PaaS.

Software as a Service (SaaS)

The cloud service model that has the highest abstraction level with the least flexibility is Software as a Service (SaaS). This service model provides a full-fledged application that can be used as is.

A good example of SaaS is your private email account. Your email account might be at Google Mail or Outlook.com. All you had to do to get it was to visit the page of the provider, insert some information like how your email address should read and your password, and off you went. Since then, you have received and sent emails without even knowing the technical parts in the background, like which email server system is used, how it was built, how redundancy was built, and so on.

Cloud Model Responsibilities

The different cloud service models have different levels of responsibility for maintaining and running the components of your application. You can find a comparison of the responsibilities of the different cloud service models in Figure 1-1.

When you host your application on-premises, you are responsible for everything. This is because it is your hardware, your data center, your systems, and your application. When you move to IaaS, the cloud provider is responsible for all the technical components, like compute, network, and storage. You are responsible for the rest.

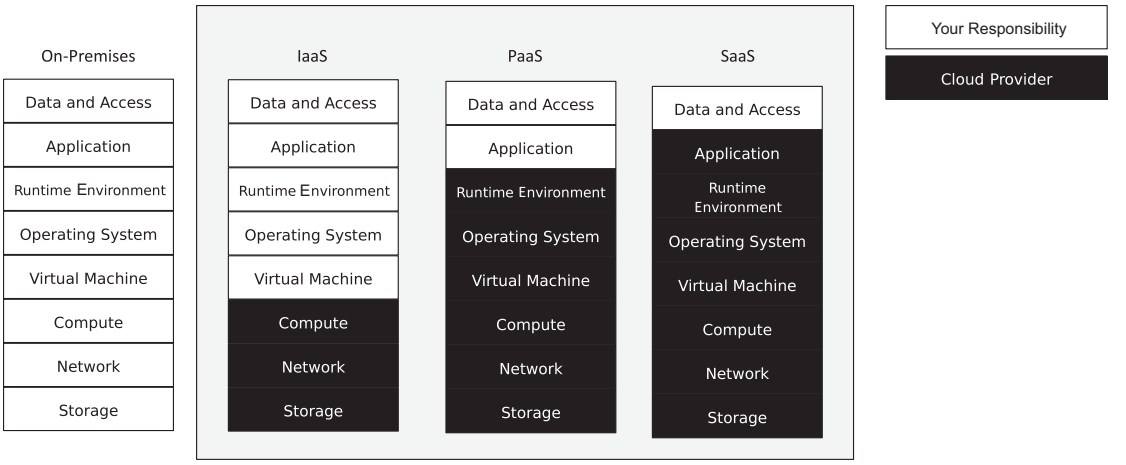

Figure 1-1. *Shared responsibilities for the different cloud service models*

Moving ahead to PaaS, the cloud provider will oversee even more things, like the virtual machine, the operating system, and the runtime environment. You are only responsible for the application you write on top of the PaaS service. When you use an SaaS application, everything will be taken care of, and the only thing you are responsible for is the data and the access to this data.

The Structure of Microsoft Azure

Microsoft Azure is a global cloud computing infrastructure that offers a great deal of redundancy, resilience, scalability, and fault tolerance. These provide the basis for all serverless services and offerings. To learn how these essential properties of Azure are achieved, let's dig in and understand the concepts on which Azure is built.

Azure Geographies

The largest structure in the Azure architecture are *Azure geographies*. There are five Azure geographies: Americas, Europe, Asia Pacific, Middle East, and Africa. An Azure geography is defined by geopolitical and legislative boundaries and can be seen as a

discrete market. Within an Azure geography there are at least two Azure regions. We will get to what exactly an Azure region is in a minute, but let's first look at the following advantages that Azure geographies offer:

- Data Residency and Compliance: Because they follow a legislative border, geographies offer customers the ability to keep their data within this area to fulfill compliance needs. Because of that, the requirements on data residency, sovereignty, compliance, and resiliency are fulfilled within an Azure geography.
- Fault Tolerance: Azure geographies are structured such that even in the unlikely event that a complete Azure region fails, fault tolerance is guaranteed through the high-capacity network infrastructure that connects the Azure regions within an Azure geography.

Azure Regions

Within an Azure geography there are several *Azure regions*. An Azure region comprises at least one but typically several *Azure data centers* that are nearby but nevertheless separated by up to 40 kilometers. Each of these data centers has its own independent electricity, climate, and network supply, so if one of the data centers within a region fails the other data centers can take over and manage the load of the failed one.

When you deploy most Azure resources you must choose the Azure region to which you want to deploy your resource. Be aware of the fact that not all Azure regions are the same. The regions differ by the services that they offer and by the prices that are charged for these services. There are some special regions that are government exclusive. Unless you work for a government organization you are not allowed and not able to deploy resources to these regions.

Each Azure region has another associated region within the same geography. These two regions build a *region pair*. If one of the regions of a region pair goes down, its associated region takes over. Regions in a region pair are divided by at least 450 km to prevent service interruptions by natural disasters, power or network outages, or civil unrest. If a large Azure outage occurs that affects both regions of a region pair, the restoration of one of the regions of the region pair is prioritized.

Azure Availability Zones

Within an Azure region, Azure data centers are structured in *Azure availability zones*. These availability zones comprise one or more separate data centers and provide an *isolation boundary*. This isolation boundary is achieved through the fact that each data center has its own electricity, cooling, and network access that are completely independent from the other data centers in the other availability zones. The availability zones are connected through private, high-throughput fiber network connections.

Azure Account

Your entry into the world of Azure is your *Azure account*. An Azure account is bound to a credit card and an Azure Active Directory user and is used for billing. An Azure account can have one or more Azure subscriptions.

Azure Subscription

An *Azure subscription* is a boundary for billing and access control. Although an Azure account can have more than one Azure subscription, each subscription is billed individually, and a separate invoice is generated for each.

Different subscriptions in an Azure account can have different billing models. Besides this, Users that are granted access on one Azure subscription do not have access to another Azure subscription within the same Azure account. Sometimes it is useful to create more than one Azure subscription within an Azure account because of organizational demands or because of restrictions within one Azure subscription. There are some limits within an Azure subscription that cannot be circumvented, so you have to create more than one subscription if you hit these limits.

Azure Resource Groups

Every resource you create in Azure has to be in one, and only one, *resource group*. Resource groups are logical containers and help to keep resources that have the same life span together.

Imagine you are building a SharePoint Farm from virtual machines in Azure. Each virtual machine has at least four resources (virtual machine, hard-disk, network adapter, IP address. A mid-size SharePoint Farm might exist of five servers, so you have 20 components at least. To prevent having to delete all of these components one after the other (and having to know the exact order in which you can delete them), you just delete the resource group to which all these components belong.

How you organize resources in resource groups is completely up to you and the needs of your organization. Besides life-cycle management, resource groups are also a boundary for authorization. Although you can define the permissions a user has for each Azure resource, it is more manageable to set user permissions on the resource-group level.

Azure Resource Manager

One key component of Azure is the *Azure Resource Manager*. The Azure Resource Manager controls how resources are deployed to Azure and how they are created, altered, and deleted. All tools you use to work with Azure communicate with the Azure Resource Manager.

The main tools that are used with Azure are as follows:

- **REST**: Azure has a very rich REST interface that can be utilized by your own applications.
- **Azure PowerShell**: Azure PowerShell is the main tool for automating Azure when you are working with Windows.
- **Azure CLI**: Azure CLI is another implementation of an Azure command-line interface that can be used with Windows and other operating systems like Linux or Mac OS.
- **Azure Portal**: The Azure Portal is a web-based interface that can be used to manage Azure in a more graphical way.

In Figure 1-2 you can see how the different Azure tools communicate with the Azure Resource Manager.

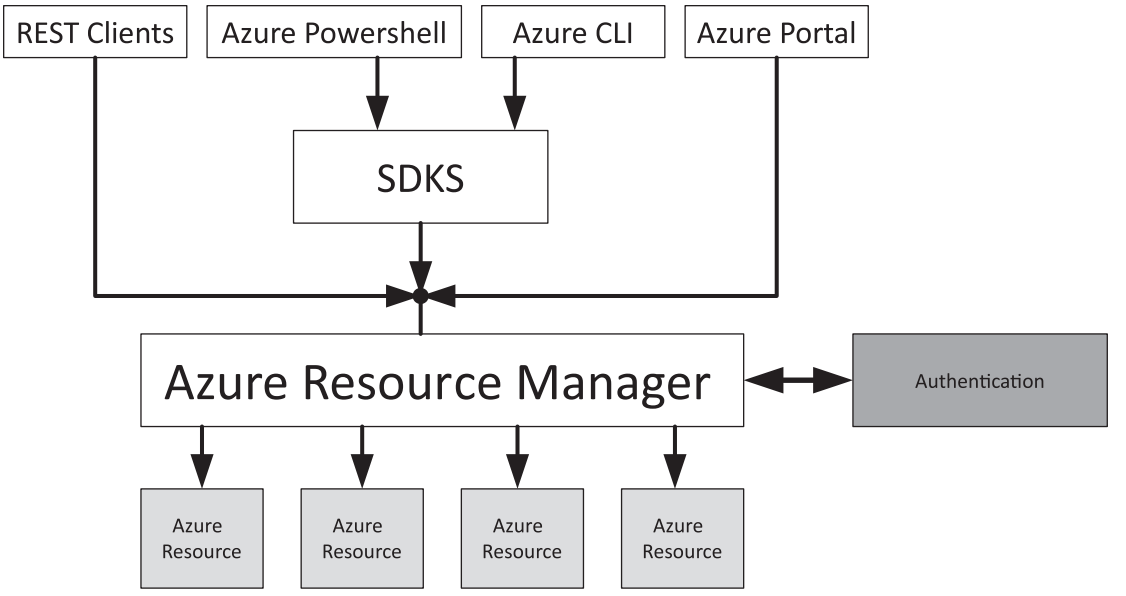

Figure 1-2. *The Azure Resource Manager*

All these tools communicate with the Azure Resource Manager, which then creates and manages the resources in Azure. One of the main advantages of Azure Resource Manager is that through it, Azure can be managed in a declarative way. In comparison to an imperative way of programming where a computer is instructed to execute commands one after another, a declarative way of programming only describes what the result should look like. How this result will be reached is up to the logic that builds this result, in this case Azure Resource Manager.

If you want to learn more about the foundations of Microsoft Azure, you can find lots of information at `https://azure.microsoft.com/en-us/global-infrastructure/`.

Creating and Naming the Resources

As you learned in the previous section, there are different ways to create resources in Azure. Although the Azure Portal might seem to be the easiest way to create resources in the beginning, it has many shortcomings in regard to repeatability. If, for example, you would like to create 100 virtual machines, it is very cumbersome to do so with the portal because you are going through the same process repeatedly.

Creating Resources

A better way to create resources is through code. The advantage of this approach is that you can repeat the same process easily, and if there is demand for many resources then you can build a loop. Besides this, you have documentation of what you have done—your code. So, our advice is that you should try to do as many tasks in Azure as you can through code. In this book we stick to Azure CLI because it is broadly available on all major operating systems.

There are two main ways to implement Azure resources in code. One is the imperative way, and the other is the declarative way. In the *imperative way* you will instruct Azure what to do step by step. First create a network, then create a network adapter, then create a virtual machine. In the *declarative way* you will just describe what the result should look like, and Azure Resource Manager will build all the necessary resources for you. The technique of describing your Azure environment in a declarative way is called ARM Templates.

Essentially *ARM Templates* are JSON files of a certain format that describe Azure resources. If you are interested in learning about ARM Templates, we recommend the Microsoft Learn Course on ARM Templates: `https://docs.microsoft.com/en-us/learn/paths/deploy-manage-resource-manager-templates/`.

Naming Resources

Another important thing we would like to draw your attention to is the naming of Azure resources. Although you are kind of free in how to name your Azure resources (with some limitations here and there), you should create a concise policy for naming conventions. This helps you to identify resources just by their name, which is especially important if you have lots of resources in your Azure subscription. If naming Azure resources based on the type of resource, there are different naming scopes, as follows:

- **Global**: If the scope of the name is global, which is the case, for example, for Azure resources like Azure SQL Database, the name must be unique in Azure. This is because in the case of global scope the name will be part of a DNS name.
- **Resource Group**: If the scope of the name is the resource group, as is the case for names of virtual machines, the name must be unique in the resource group that contains the resource. It is OK to have virtual machines that are named `dc` in different resource groups.

- **Resource Attribute**: If the scope of the name is resource attribute, the name of the resource must be unique inside the parent resource. This is the case for a subnet within an Azure virtual network.

If you create a naming convention for your resources, you can include much information in your resource names, like resource type, business unit, region, subscription type, deployment type, and so on. How you do it is completely up to you. The only important thing to remember is to be concise. If you build a naming convention, stick to it; otherwise, the intention of the naming convention is not achieved. A very good article on naming conventions for Azure resources can be found here: https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-naming#example-names-for-common-azure-resource-types.

Overview of Data Services

The focus of this book is serverless data handling in Azure. Although you will get plentiful information on different services that can store your data in Azure throughout the entire book, we will give you a short overview of different services that do so.

Data Categories

There are different factors that drive the decision of which Azure data service is the right one for your scenario. It largely depends on which type of data you would like to manage, how much data you want to manage, and of course what your existing skillset is regarding data storage.

We divide data into the following three categories:

- **Structured Data**: Structured data is the type of data that first comes to mind and is stored within a relational database. This data underlies a strict schema that is implemented in the database, and each dataset must conform to the restrictions that are implemented in the database. If there is, for example, a column that stores numeric values, you cannot store text in this column. The restrictions and constraints that are implemented in a relational database help to guard the quality of the data.

- **Semi-Structured Data**: While there is a global schema for structured data that each dataset must fulfill, this is not so with semi-structured data. Semi-structured data brings its own schema. Imagine a blob storage where you can store JSON files. Each of these JSON files can have its own schema describing the data stored in the file. Semi-structured data is often compared to books. A book has a table of contents, chapters, sections, and an index that define the structure of the book. Although each book has these structural components, each book has its own.
- **Unstructured Data**: Unstructured data does not have any kind of schema and can be seen as a stream of binary data. The term *unstructured* data is not completely adequate because the data has internal structure. This internal structure divides a PNG image file from an MP3 audio file.

Azure Data Services

To satisfy the different demands to store and manage data, Azure offers many services that can store data. In the rest of this chapter, we will give a short introduction to all of these services and discuss if they can be seen as "serverless" or not.

- **Azure Table Storage**: Azure Table storage is part of an Azure storage account. Azure Table storage is a simple key–attribute store where you can store large amounts of structured data. In comparison to a full-fledged relational database, there are no complex joins possible with Azure Table storage. For some applications this is enough, and you do not introduce the complexity of an Relational Data Base Management System (RDBMS) to your project. Azure Table storage can be seen as a serverless datastore because you do not define any infrastructure-specific attributes.
- **Azure Blob Storage**: As with Azure Table storage, Azure Blob storage is part of an Azure storage account. You can use Azure Blob storage to store large amounts of unstructured data like videos, pictures, or any other kind of file. Because Azure Blob storage is part of an Azure storage account, it is a serverless datastore too.

- **Azure File Storage**: Azure File storage is part of an Azure storage account too. The distinctive thing about Azure Blob Storage is that it can be used as Server Message Block (SMB) share. As part of an Azure storage account, Azure File storage is a serverless datastore.
- **Azure Queue Storage**: The fourth part of an Azure storage account is the Azure Queue storage. Azure Queue storage implements a highly scalable queue that can be used to buffer messages in event-driven applications. Because Azure Queue storage is part of an Azure storage account, Azure Queue storage is also a serverless datastore.
- **Azure SQL Database**: An Azure SQL database is a single Microsoft SQL server database that has some restrictions not seen in a regular SQL server. When you create an Azure SQL database you have to specify how many CPU cores you need and how much RAM your database will likely use. These options qualify Azure SQL database as a platform service and not as a serverless datastore, with one exception: there is a serverless option for Azure SQL database too, which we will discuss later in this book.
- **Azure SQL Managed Instance**: While an Azure SQL database is a single managed database, Azure SQL Managed Instance is a whole managed SQL server. For Azure Managed Instance, the same is true as for an Azure SQL database. When you create an Azure SQL managed instance you have to specify the number of CPU cores and the amount of RAM your managed instance will use, so Azure SQL Managed Instance is a platform service and not a serverless datastore.
- **SQL Server on Virtual Machines**: This is the fallback option if an Azure SQL database or Azure SQL Managed Instance do not work for you. You can still create a virtual machine and install a SQL server on it, as you would do on an on-premises virtual machine. The existence of a virtual machine in this context disqualifies this solution from being a serverless datastore. This solution is not even a platform service, but rather infrastructure as a service.

- **Azure Database for Postgres**: An Azure database for Postgres is the same as an Azure SQL database with the exception that this offering is not backed by a Microsoft SQL server, but rather a PostgreSQL server. Besides this, an Azure database for Postgres is the same as an Azure SQL database; that is, you have to define CPUs and RAM while creating an Azure SQL database for Postgres, so this is not a serverless datastore.
- **Azure Database for MySQL**: An Azure database for MySQL is a managed database offering for MySQL. When creating an Azure database for MySQL you also have to specify CPUs and RAM, and therefore an Azure database for MySQL is a platform offering and cannot be considered a serverless datastore.
- **Azure Database for MariaDB**: An Azure database for MariaDB is a managed database offering for MariaDB. For an Azure database for MariaDB, the same is true as for any other Azure database offering: during creation of the service, you have to specify how many CPUs and how much RAM should be assigned. Thus, an Azure database for MariaDB is a platform service and not a serverless datastore.
- **Azure Cosmos DB**: Azure Cosmos DB is a global NoSQL database that can be accessed through different APIs, like Core SQL, MongoDB, or Gremlin. When you create an Azure Cosmos DB you can choose one of two deployment models: provisioned throughput and serverless. While provisioned throughput is more like a platform service, serverless provides a serverless datastore. In this book we will examine the serverless deployment model of Cosmos DB.
- **Azure Cache for Redis**: An Azure cache for Redis is a high-throughput in-memory datastore that can be used to quickly cache data. When you create an Azure cache for Redis you have to specify a cache type, which includes how much RAM will be assigned to your cache. This disqualifies Azure cache for Redis as a serverless datastore; it is a platform service.

- **Azure Managed Instance for Apache Cassandra**: Azure Managed Instance for Apache Cassandra implements a datastore for Cassandra data as a managed service. When you create an Azure managed instance for Apache Cassandra you have to specify the SKU and the number of nodes. This qualifies Azure Managed Instance for Apache Cassandra as a platform service but not as a serverless datastore.
- **Azure Data Lake Storage Gen 2**: Azure Data Lake Storage Gen2 is an addition to Azure Blob storage and provides hierarchical storage for a large amount of data. As it is based on Azure Blob storage, Azure Data Lake Storage Gen 2 is a serverless datastore too.
- **Azure Synapse Analytics**: Azure Synapse Analytics is the Microsoft service to build modern cloud-based data warehouses. It offers many services to aggregate and analyze data. Within Azure Synapse Analytics you can initiate serverless databases. Azure Synapse Analytics itself is not only a datastore but much more. It can include serverless datastores, so whether Azure Synapse Analytics is serverless or not depends on how you build your environment within Azure Synapse Analytics.

Summary

In this chapter, we have introduced the basic concepts of Azure. You have learned about regions and geographies as well as subscriptions and resource groups.

You learned about the importance of naming conventions and where to find good suggestions for your first projects. Furthermore, you got an overview of the different data services that Azure offers you.

You now have covered the basics and are ready to dive into the more technical chapters.