Microsoft

Azure

The Developer's Guide to Microsoft Azure

Second Edition

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

Microsoft Press books are available through booksellers and distributors worldwide. If you need support related to this book, email Microsoft Press Support at mspinput@microsoft.com. Please tell us what you think of this book at http://aka.ms/tellpress.

This book is provided "as-is" and expresses the author's views and opinions. The views, opinions and information expressed in this book, including URL and other Internet website references, may change without notice.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

Acquisitions Editor: Phil Evans
Developmental Editor: Bob Russell, Octal Publishing, Inc.
Editorial Production and Project Management: Dianne Russell, Octal Publishing, Inc.
Copyeditor: Bob Russell
Cover Design: Justin Brown, Octal Publishing, Inc.

Contents

The Developer's Guide to Microsoft Azure

The demand for software and related services today is greater than ever before, and this trend will only continue—exponentially so—as applications of all sorts more and more become a part of every aspect of our daily lives. You, as a developer, are the "magician" who is bringing this digital revolution to your users, and that's not easy. Besides creating the unique tangible features that your software offers, there are so many behind-the-scenes elements that you need to take care of: your applications need to be highly available, performant all over the world, function seamlessly on a wide range of devices, be secure, and be able to run at massive scale for a reasonable price. These challenges are common across most software and most industries. The solutions, however, are ephemeral things that we keep reinventing, and they can be difficult and costly to build and maintain.

We created *The Developer's Guide to Microsoft Azure* to help you on your journey to the cloud, whether you're just considering making the move, or you've already decided and are underway. This e-book was written by developers for developers. And it is specifically meant to give you, as a developer, a fundamental knowledge of what Azure is all about, what it offers you and your organization, and how to take advantage of it all.

The second half of the guide walks you through scenarios such as a tour of the Azure Portal and creating a Virtual Machine. We also discuss developing and deploying a web application that uses Node.js and MongoDB. We cover typical tasks such as CI/CD (Continuous Integrations and Continuous Deployment), staging environments, scaling, logging, and monitoring. We wrap up by creating a backend for your mobile application that includes authentication and offline synchronization.

Introduction

There are countless books, articles, papers, blog posts, and so on that espouse the virtues of cloud computing, and you can find zetabytes worth of information that explain what cloud computing is in a technical sense. But before getting into any of that, you need to ask a couple of simple questions: why should I make the move to the cloud at all? How will it benefit me; my team; my company?

Information technology as a utility, not a burden

Cloud computing is computing as a utility. What do we mean by this? To help you visualize it, consider something else that's ubiquitous in your life: electricity.

Who today would think that having a mini-generating plant outside each home and business would be an efficient model for the distribution of electric power. We are accustomed to simply flipping a switch to turn on a light, and then we go about doing what we need to do; you certainly would never dream of having to buy, install, and maintain all of the equipment needed to illuminate your home or office. Instead, you pay your local electric company for the power you consume every month, and you never need to concern yourself with the physics of generating electricity, dealing with the capital expense, maintaining the equipment, repairing it after storms—nothing. Just flip the switch, and be productive at something else.

Well, just as you neither need nor want to know exactly how electricity is delivered to your house, neither do you need to know exactly how the computing power and services that you consume in the cloud are made available to you. Someone else takes care of that for you. Thus, first and foremost, moving your compute, storage, and network to the cloud provides ease of use, just like turning on a lamp. And you pay only for what you use, in the same manner as you pay only for the electricity you use to power your home.

Of course, moving beyond our analogy, the cloud offers many more specific benefits for your IT operations, like scalability and the ability to use intelligent services like search and facial recognition in your applications, to name but just two.

Computing as a utility

So, ultimately, you can think of cloud services such as Azure as *computing as a utility*. Of course, Azure offers far more in the way of products and services than the local electric utility does, so cloud computing can sometimes be difficult to clearly define. In fact, some say that the cloud is basically just using *somebody else's computers*, which is technically true, but the cloud is so much more:

*The cloud is a system that provides **robust**, **resilient**, **intelligent services** and **compute** resources at **infinite**, **elastic**, **global scale**.*

Let's break that down:

The cloud and Azure provide services that help you to accomplish so many things: from the mundane, such as adding Search to your application, to the more exotic, such as implementing Continuous Integration (CI) and Continuous Deployment (CD) workflows. You can automatically tune your

database, or set up push notifications to mobile devices, easily and quickly. These are just a few examples of some common things that developers have created for themselves over and over again but are now available as a service. This makes it possible for you to use these services with very little effort—almost like flipping a light switch! You can then focus on the pieces of your application that make it unique; the features that provide real added value for your users.

Besides services, the cloud offers compute resources in the form of Virtual Machines (VMs), containers, databases, and so on. You can use these to host applications or to provide a complete infrastructure for your users.

As we stated a moment ago, the power of the cloud is that services and resources are incredibly robust and resilient. It is very unlikely that they will fail to run. This is because the cloud is smart. It is self-healing and, as is the case with Azure, there are datacenters all over the world, filled with tens of thousands of servers. If one server fails, another takes over. If an entire datacenter were to fail (a highly unlikely scenario in and of itself), another would take over. All of this is possible because of the massive scale of the cloud.

One of the most compelling arguments in favor of the cloud is that you can scale-up your services and resources almost infinitely, and certainly to a degree that you simply can't do with on-premises resources, unless you're prepared to spend enormous sums of money on capital equipment and staffing to administer it all. And, you can scale globally. You can put your services anywhere in the world so that you can provide a performant experience to your users, regardless of where they are. It also means that you can keep your data where you need it to be.

Perhaps most important, though, when you use cloud resources, you can scale-back your services and resources when there is no longer high demand. Returning again to our electric utility analogy, if you're having many friends and relatives to your house for a party, you're going to turn on a lot of lights inside and outside the house, use your microwave oven frequently, turn on some music, and, generally, consume more electricity. But when the party is over and your guests have left, your power usage returns to normal, and so does your billing. It's the same thing when you take advantage of the cloud and Azure: you pay only for what you use, not for what you might need when usage is high.

More info To learn about the Azure portal and create your first VM, go to https://docs.microsoft.com/azure/virtual-machines/windows/quick-create-portal.

In addition to massive scalability, off-the-shelve intelligent services, and pay-per-use efficiency, the cloud offers increased security.

The cloud is used by millions of people, 24x7, worldwide; of course, it is attacked by many people, as well. Reputable and experienced cloud providers like Microsoft know the usage patterns of normal users versus those of malicious actors. This means that it knows how to protect against both the most common and most unique attacks out there. Intelligent monitoring tools, machine learning algorithms, and artificial intelligence give cloud providers the ability to detect attacks in real time and stop them in their tracks.

Decades of experience in security and massive-scale traffic, combined with top industry security expertise, make the cloud a much more secure environment than any on-premises datacenter.

More info To read more about how Azure secures your applications and data, go to How Azure Security Center detects DDoS attack using cyber threat intelligence, Azure Security Overview, and How Microsoft deals with cybercrime.

We've briefly explored the question of why you want to begin the migration to the cloud and Microsoft Azure. Now, let's examine the cloud's major components, what they do, and how they fit into the grand scheme.

Cloud computing deployment types

The cloud offers several levels of products and services, such as infrastructure as a service (IaaS) and platform as a service (PaaS), but before we get to those, let's step back and look at the three basic types of cloud platforms: *public*, *private*, and *hybrid*. These have rapidly become modern technology buzzwords. Let's demystify them.

The public cloud

In the public cloud model, you don't own any hardware; your cloud vendor is responsible for purchasing and maintaining all hardware. The public cloud consists of all of the services and compute resources that you do not own yourself but that you use from your cloud provider. The provider is responsible for keeping everything running and adhering to Service-Level Agreements (SLAs). *You pay for what you use, not for what you own*.

The private cloud

With a private cloud, you own all of the hardware or, at least, have complete control over it. The hardware that runs your services and houses your data is somewhere in your on-premises datacenter. Of course, this complete control comes at a price: you must purchase and maintain everything. *You pay for what you own, not for what you use*.

But, how do you set up a private cloud platform in a traditional on-premises datacenter. Fortunately, Azure provides a unique solution designed specifically to work in this environment: Azure Stack. In a nutshell, Azure Stack is your own private instance of Azure in a box that you can run in your on-premises datacenter.

Running services in a private cloud is not the same as running services in your on-premises datacenter. Azure Stack offers all of the advantages of Azure, including intelligent services that you can use in your applications, all within the confines of your on-premises datacenter. Thus, if for special security or governance reasons you need to keep certain operations "close to home" in your on-premises datacenter, you can still enjoy all of the benefits offered by Azure and the cloud paradigm.

The hybrid cloud

The hybrid cloud model is, as its name implies, a mixture between the public and the private cloud. You can use public cloud services that use resources in your private cloud, and vice versa. In Azure, you could run an application in an Azure Web App that connects to an on-premises database using Azure Hybrid Connections. This opens a lot of possibilities. You control where your applications and data are while still gaining the benefits of using intelligent cloud services.

Azure provides solutions for all cloud computing deployment types. This gives businesses more freedom to use Azure where and how they want to. And they can determine the amount of control that they want to have over their applications and data.

Demystifying the types of cloud computing

Within cloud computing, we can categorize services and resources into types. There are many different cloud computing type definitions. The differences between the types are the amount of control and responsibilities that you have versus the amount of time and effort you can spend on adding business value to your company or customers. Figure 1-1 and Figure 1-2 illustrate how we define cloud computing types for Azure.

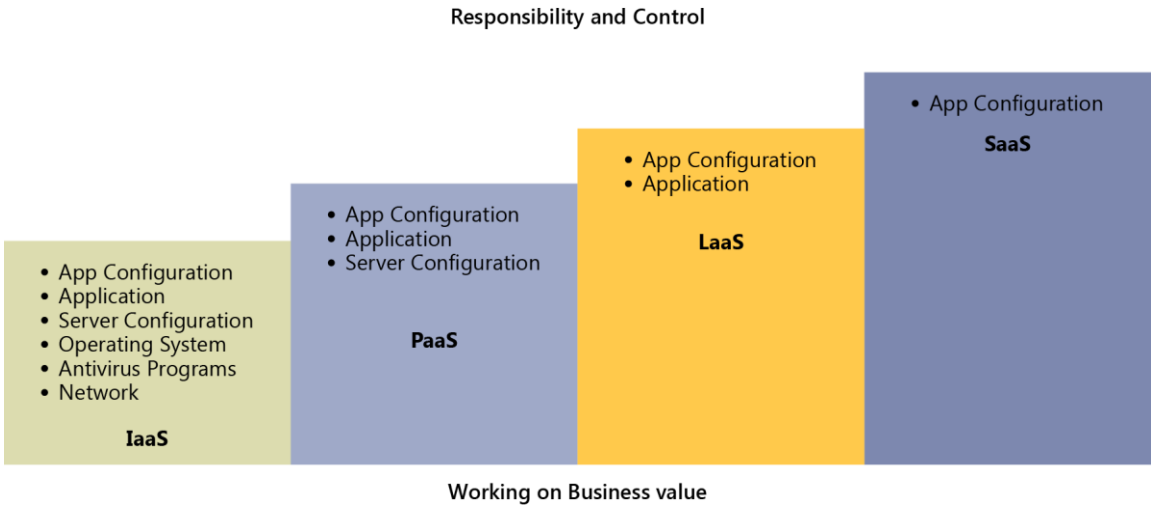

Figure 1-1: Types of cloud computing explained

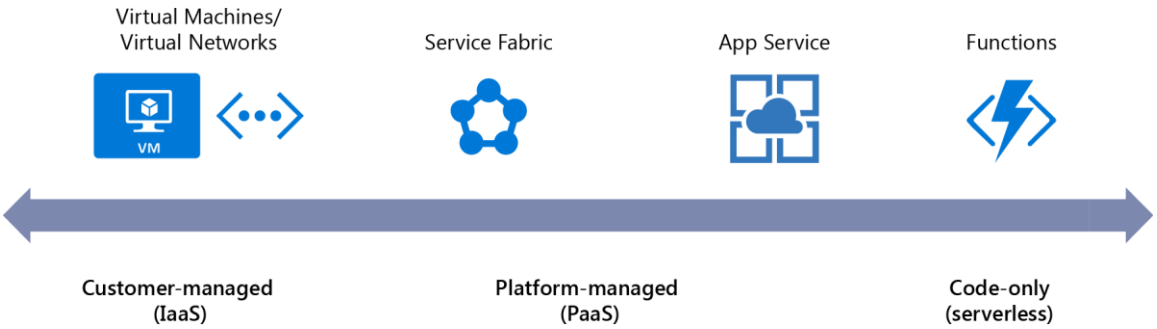

Figure 1-2: Examples of cloud-based offerings

Infrastructure as a service

With infrastructure as a service (IaaS [pronounced "eye-as"]), you are responsible for everything, *except* the hardware. Azure takes care of all the hardware, such as the solid-state drives (SSDs), network cabling, routers, power supplies, backup generators, cooling systems, and so on. Services that fall into this category include Azure Virtual Machines, Azure Virtual Networks, and Azure Containers.

You take care of everything else. Your application, patching of the operating system (OS), logical network configuration, and even keeping antivirus programs on the machines up to date. This means that you maintain total control over these elements. But, it also means that you spend a lot of time and effort keeping the services and resources running and less time working on adding business value to your core application products.

Platform as a service

Using the platform as a service (PaaS [pronounced "pahz"]) model, you are responsible for your application and its configuration; Azure provides the OS and hardware. As Figure 1-1 demonstrates, PaaS is an abstraction level higher than IaaS. You have far fewer responsibilities but less direct control over your hardware resources than with IaaS. Services that fall into the PaaS category are Azure App Service, Azure SQL Databases, and Azure Redis Cache.

With PaaS, you can host your application, but this level also offers additional capabilities that you can use out of the box, such as push notifications for mobile devices via Azure Mobile App, and automatically tuning your database in Azure SQL Database.

You deploy your application and configure it. You also do basic server configuration, like scaling the pricing tier or the number of instances on which your app runs. Your application still runs on a server somewhere, but you aren't responsible for that server (nor do you have any control over it), which means that you can spend much more time concentrating on adding business value to your applications, and less time keeping a server running.

Try it out Create a Node.js application with a MongoDB on Azure in this walk-through at https://docs.microsoft.com/azure/app-service/app-service-web-tutorial-nodejs-mongodb-app.

Function as a service

In function as a service (FaaS [pronounced "fahz"]), you are responsible only for your application and your business logic. You don't need to worry about scaling, the OS, or any of the hardware. FaaS is sometimes called *serverless computing*. Services in the FaaS category are Azure Functions, Azure Logic Apps, and Azure Event Grid.

In FaaS, all you need to do is create your application or weave together business logic and run it. Scaling happens automatically. One of the more noteworthy benefits of FaaS is that you pay for the resources you use only when the logic runs instead of paying for a service that is always on, waiting for somebody to use it.

But FaaS does more than just host your application; it connects your logic to triggers and sources. Outside sources such as Azure Storage Queues or WebHooks can initiate Azure Functions or Logic Apps and provide them with data to process. You do not need to write the activating logic or set up the plumbing to connect to external data stores or services, meaning you can focus on your logic and customer-facing features.

Software as a service

With software as a service (SaaS [pronounced "sass"]), you need only concern yourself with configuring the software—nothing else. Indeed, SaaS is the highest abstraction level of the cloud computing types. Services in the SaaS category include Azure Cognitive Services and Azure IoT Suite. Office 365 is another example of an Azure SaaS application suite, offering reliable business applications such as Microsoft Word, Excel, PowerPoint, and more. And all of this is available without you having to install or maintain anything.

With SaaS, the software is ready to use. All you need to do is configure it to your preferences and then you are ready to go. You don't write the software, and you don't need to think about deployment, scaling, operating systems, and hardware.

Why Microsoft Azure?

So, you're thinking the cloud might be what you need for your company to create incredible software products and deliver the value and user experience your customers expect. Certainly, the Azure cloud platform can help you accomplish that. With its vast resources and compute power, plus a wide array of intelligent services, you can build highly scalable, dynamic apps, and pay only for what you use. But there are many cloud vendors out there today, each offering unique capabilities and products. So, again, why Azure?

With Azure, you get the deep knowledge and expertise of Microsoft, a company that has been at the forefront of both personal and enterprise-level computing and IT solutions for more than 40 years. Microsoft has been designing, building, and running massive-scale datacenter operations and networks for its own purposes for decades. That experience and know-how is woven right into the fabric of Azure. It's at its very core.

Who uses Azure?

Adobe, Jet.com, Geico, GeekWire, and DocuSign are just some examples of customers all around the world that look to the Azure platform to help them achieve their business goals and deliver more value to their customers. Some of those goals include the request for App Services to Serverless, Artificial Intelligence to Azure IoT and even Media streaming. Let's look at some interesting ways that customers have used the Azure platform to meet their business needs.

South Korean industrial giant Samsung creates everything from phones to televisions, to washing machines, major kitchen appliances, and air conditioners. The company decided to use Azure to increase the efficiency and robustness of its air conditioners by performing remote monitoring and maintenance. The company uses Azure IoT Hubs to collect massive amounts of data sent from sensors embedded in its air conditioners that monitor the status of the machines as well as the environment in which they run. This data is captured in Azure SQL for real-time analysis. Engineers then can use the information they extract to adjust the behavior of the air conditioners, resulting in energy savings of 15 to 18% on average for its customers.

The Coca-Cola Company, one of the worlds most recognized brands, first introduced its eponymous soft drink in 1886 and has been a global leader ever since. In 2012, the company launched Coco-Cola Journey, a highly ambitious project and website whose purpose is to help Coca Cola better connect with a new generation of customers who more and more take their purchasing cues from nontraditional marketing channels. The company wanted to take advantage of social media and a Microsoft service called How-Old.net to create a fun and engaging interactive experience for users, while simultaneously gaining insights into its customers. Based on the Cognitive Services technology, How-Old.net analyzes data extracted from facial imagery to establish the age and gender of individuals. But Coca-Cola asked the Microsoft team to take the project a step beyond facial recognition by using machine learning models to detect individual objects in pictures. For one month, the site invited users to submit a photo of themselves and a Coca-Cola bottle with a visible logo to see what happens. If the web app detected a bottle, the bottle's age appears and the page turns red. Then, Coke fans were encouraged to share their experience on social media and to further explore the Coca-Cola Journey site to learn more about the venerable contour bottle.

CarMax, the largest used-car retailer in the United States, redesigned its website using the Azure platform services to deliver faster response to the 16 million people who visit its site every month. Rather than "lift and shift" the site into an Azure IaaS environment, CarMax instead chose to rebuild the site using the Azure PaaS offering. This made it possible for CarMax to modernize the site with new cloud functionality, make it 100 percent mobile-friendly, and, most important, move to a dynamic DevOps development paradigm. CarMax is quickly moving toward a microservices architecture, which will give the company the ability to scale and update parts of applications independently of the other parts. Although this will take years to accomplish, choosing Azure as its development foundation will help the auto retailer get to microservices faster.

NBC Sports works with Microsoft to deliver mega broadcast events. In 2016, the two companies collaborated to broadcast the Summer Olympics from Rio de Janiero. NBC used Azure Media Services to stream content from more than 50 Olympic events simultaneously, providing a 1080p profile and covering a whopping 100 million unique users that consumed 2.71 billion minutes of content from a broad range of devices and form factors. Media Services provided the scalability and robustness to deliver this content flawlessly to the viewers.

Azure gives businesses, large and small, as well as government organizations, educational institutions, and any user of IT services the ability to more easily and cost-effectively manage their digital worlds, both for themselves and their customers. In a nutshell, here's how:

- **Deliver cross-platform software experiences, like mobile, desktop, web, and hybrid experiences:** Azure provides services with which you can implement these scenarios "out of the box," like Azure App Service, including Web App and Mobile App, and Mobile Center and Visual Studio Team Services for delivering and managing all application types.
- **Deliver services wherever they need to be:** Azure is unique in its breath of deployment options: the public cloud, hybrid solutions, and Azure on-premises, in your own datacenter (via Azure Stack). All of these options can provide massive scale at an affordable price, while keeping your data where it needs to be.
- **Deliver an intelligent, massive scale, data platform:** Azure has many services that you can use to capture, store, analyze, and present your data. These are services like Azure SQL Databases, Data Lake Store and Data Lake Analytics, HDInsight, Event Hubs, Cognitive Services, and many more. All of these services can create an affordable, intelligent, and massively scalable data platform that gives businesses the tools to extract valuable insights from the vast amount of data being churned out today.
- **Deliver high-quality software, fast:** You need to test always and fail fast to deliver quality software. With Azure, you can do so through integrated CI and CD mechanisms directly in services like Web App, or through Visual Studio Team Services. Features like App Services Deployment Slots make it possible for you to deliver fast, with no downtime. And, to know that your app is working as expected in production, Azure provides monitoring services, like Application Insights that let you know exactly how your app is doing and where you can improve.

Azure has global reach

With datacenters in more than 42 countries and regions around the globe, Azure's incredible coverage offers many benefits. First, it can reduce any downtime your applications might experience should an entire datacenter fail (an extremely low probability, in and of itself), because your application can failover to another datacenter and continue to run almost as if nothing happened at all. This can also reduce your disaster recovery costs.

If you use Content Delivery Networks (CDNs) to reduce latency, the chances are that there is an Azure datacenter in close proximity to where your customers are and where you need to deliver their content. Another benefit that Azure can offer with so many facilities in so many locations is that you have a wider range of choices for data storage. This can help you to ensure that your company remains in compliance with local or regional regulations and laws with respect to data sovereignty.

Microsoft continues to invest heavily in datacenter infrastructure and innovation through open-source hardware development and novel new datacenter solutions such as the *underwater datacenters* from project Natick.

Azure is extremely resilient

Azure is an intelligent, self-monitoring, and self-healing platform that you can rely on to keep your applications up and available and running well. To back that up, all Azure services operate under comprehensive SLAs that define their performance in specific terms. SLAs range from 99.9% (three nines) uptime to 99.99% (four nines). This means that Azure services are extremely resilient and reliable. Should your services fail to perform to the governing SLA's specifications, you might be eligible for a discount of up to 100% of the service costs on your Azure invoice.

To make sure that your services remain at peak performance, Microsoft monitors the service health for each individual Azure service in every datacenter around the world. These metrics are available for you to monitor, as well, on the public Azure status page.

As we mentioned earlier, Microsoft has been operating datacenters and providing IT services at massive scale since 1989. It has accumulated a lot of knowledge in that time. Today, Microsoft has some of the best minds in the IT industry working continuously to keep Azure up and running so that you can rely on it for hosting your services and storing your data. That's why all of the Azure services have SLAs to back that up.

Azure is compliant with almost every industry

Azure offers the most comprehensive set of compliance offerings of any cloud provider. The list of compliances is huge and growing continually. Some of the standards to which Azure complies include ISO 27001, HIPAA, and SOC 3. Region-specific compliances include the EU-US Privacy Shield and China DJCP.

For instance, by complying with the ISO 27001 certification, Microsoft guarantees that it implements, monitors, maintains, and continually improves its security standards for its global network of datacenters as well as for the individual services in Azure.

More info For a complete listing of compliance statements, go to https://www.microsoft.com/trustcenter/compliance/complianceofferings.

Azure even has a specific offering for US government customers called Azure Government, which is basically a "special edition" of the Azure cloud that addresses the unique needs of government entities. Azure Government is run separately from the Azure public cloud, in its own *instance*, meaning that it runs on separate, dedicated servers on a completely separate and isolated infrastructure.

Azure focuses on capabilities for developers

Azure helps you, as a developer, to be more productive by providing much of the "plumbing" that you would otherwise need to build yourself. With capabilities such as autoscaling and authentication/authorization tools, you can add advanced features and functionality to your applications with little to no effort. You can integrate services like Azure Search or Cognitive Services that will enhance your applications and please your users.

Azure is open and supports the frameworks of your choice

Write your applications in JavaScript and deploy them to Web App. Or, write them in Ruby, Django, Java, PHP, or .NET; whichever is your preferred environment. Azure has wide support for lots of languages. You can bring your existing application or write one from scratch, and it will run in Azure.

Additionally, you can choose the ecosystem you prefer. Azure supports most operating systems, like Linux and Windows, and you can script everything using Bash if you want. You can also run your application how you want—using containers, Azure Service Fabric, App Services, Azure Stack, and so on.

After you have learned how to use Azure with one toolset, you can use it with any other toolset. The services and the Azure portal work the same for everything.

You can monitor your Azure services on the go

Are you the sort of person who likes to keep a constant eye on operations? With Azure, you can monitor how your services are performing and fix issues that might arise by using the Azure mobile app. This helpful app is designed to give you visibility into your resources even when you are not behind your desk at your main computer. You can handle tasks like restarting a web app and stopping a virtual machine. It also provides a full command-line interface (CLI) experience via Cloud Shell, making it possible for you to do basically everything that you can do with the Azure portal.

Additionally, there are services in Azure that perform smart detections and recommendations for your services and can send you alerts about them. These are things like detecting that a web app is running slower than usual or informing you that an App Service scaled up automatically. Azure Security Center and Azure Monitor are some of the services that can alert you about these things so that you never have to miss anything important.

We are here if you need help

There are lots of ways to get help with Azure if you need it.

You can buy a support plan, that gives you access to the Azure technical support teams and provides other services like guidance based on best practices to design for the cloud, or assistance in planning your migration. Depending on your needs, you can buy a support plan that guarantees responses from the technical support teams within 15 minutes.

Additionally, you can get help for free, 24x7, through many channels. You can tweet to @AzureSupport or reach out to the support teams through MSDN forums, StackOverflow, Reddit, or the Microsoft Tech Community.

2

Getting started with Microsoft Azure

After some thoughtful deliberations, you've joined the Azure community, and now you want to get started building applications. What do you need? Not much really. The most important thing to have is just a connection to Azure to deploy your app. You can use the tools, applications, and frameworks of your choice.

Your IDEs and editors are welcome

Whatever you use to create or edit your applications in your on-premises environment, you can use with Azure. Choose anything: from Notepad to Visual Studio Code, to Sublime Text, to Visual Studio, to the code editors in the Azure portal. It's your call!

Plug-ins are available for all the popular IDEs, giving you the ability to do things like publish directly to Azure. And even if you prefer using a bare-bones tool like Notepad, you can still automate your deployments to Azure using Continuous Integration (CI)/Continuous Delivery (CD) practices.

Use the Azure command-line interface for scripting

Azure provides a powerful command-line interface (CLI) with which you can do basically everything in Azure. You can use the Azure CLI to start and stop virtual machines (VMs), to deploy your application to an app in Azure Web App, to create new resources, and everything else. It's also very useful for automating tasks and running them in your CI/CD pipelines.

You can use the Azure CLI available on the Azure portal in the Azure Cloud Shell, or locally, on your machine, through the Azure CLI tooling. Another advantage is that you can apply your existing Bash or Windows PowerShell expertise in the Azure CLI.

Run anything on Azure

Azure is great for web applications and APIs. It's also great as a host for desktop and mobile applications. You could, for instance, use it to authenticate users in your desktop application or send push notifications to your mobile app. No matter what scenario you have in mind, Azure can add value to it.

What about costs?

Clearly, it's important to know what things cost in Azure. To assist you in that, Microsoft offers the Azure Pricing Calculator. With this handy tool, you select the services that you are going to use and specify how much you anticipate using them, and the calculator will let you know what your monthly and yearly bill will be. This, together with how much data you'll use, determines your costs.

Note The Azure Pricing Calculator estimates costs for pay-as-you-go subscriptions. There are other payment options for enterprises, partners, and MSDN subscribers, which can affect the costs of services.

You can also keep an eye on your costs using the Azure Billing APIs and Pricing alerts. You can take advantage of these to gain insights into costs and spending trends so that you can control your costs.

Selecting the right Azure services

Azure provides a lot of services to help you build and run your applications. There are so many services, in fact, that it can sometimes be a bit confusing to choose just the right services for your scenario. Which ones should you pick as a developer? Let's explore some of your options.

Where to host your application

The first choice you'll need to make right out of the gate is where to host your application. Azure offers several hosting options.

VMs

One of the ways to host your application is in a VM in Azure Virtual Machine. This provides you with a lot of control over how you host your application, but you are responsible for maintaining the environment, such as patching the operating system (OS) and keeping antivirus programs up to date.

You can, for instance, use an VM to test the latest preview version of Visual Studio, without getting your machine "dirty."

Containers

Containers are much more lightweight than VMs and you can start and stop them in a few seconds. Containers also offer tremendous portability, which makes them ideal for developing an app locally, on your machine, and then hosting it in the cloud, in test, and later in production. You can even run containers on-premises or in other clouds—the environment that you use on your development machine travels with your container, so your app always runs in the same ecosystem.

Just like VMs, containers provide you with a lot of control over your environment. You can install what you need to run your applications. But, again, you are responsible for patching and maintaining the OS that runs in the container and for ancillaries like antivirus programs.

Host containers with Azure Container Instances

You can host your container using Azure Container Instances. Using this service, you can quickly spin up a container, without the need for a container orchestrator, like Kubernetes, and without having to manage the resources that host the container.

The Container Instances service is billed per second, per virtual CPU, per gigabyte, or memory.

More info To learn more about Container Instances, go to Get started with Azure Container Instances.

Host containers with Azure Container Service

Another way to host containers is by using Azure Container Service. With this service, you can scale and manage your containers using orchestrators like Mesosphere DC/OS, Docker Swarm, or Kubernetes. This service is a great way to begin moving your containers to the cloud.

More info To learn more about Container Service, go to Get started with Azure Container Service.

Azure Functions

With Azure Functions, you can write just the code you need for a solution without worrying about building a full application or the infrastructure to run it. A function is a unit of code logic that is triggered by an HTTP request, an event in another Azure service, or based on a schedule. Input and output bindings connect your function code to other services, like Azure Blob Storage, Azure Cosmos DB, and Azure Service Bus, with minimal code. Using Functions, you can build small pieces of functionality quickly and host them in an elastic environment that automatically manages scaling.

Another thing that makes Azure Functions special is that you can choose to pay only for functions that run, without having to keep compute instances running all month. This is also called *serverless* because it requires only that you to create your application and not deal with any servers or even scaling of servers.

You can write Azure Functions in C#, F#, Node.js, Java, PHP, and a growing list of languages.

An example of an application that uses Functions is one that activates a function every time a new image file is uploaded to Azure Blob Storage. The function would then resize the image and write it to another Blob Storage account. The function signature for this example would look like this (in C# script):

```
public static void Run(Stream myBlob, string name, TraceWriter log, BlockBlob outputBlob) { }
```

Data from the Blob that triggered the function is passed into the function as the `myBlob` parameter, which includes the Blob URL. You can use `outputBlob` output binding parameter to specify the Blob to which to write the result. There's no need to write the plumbing for connecting to Blob Storage, you just configure it.

Try it out Create your first Azure Function using the Azure Portal.

Azure Logic Apps

You can orchestrate business logic with Logic Apps, automating a business process or integrating with software as a service (SaaS) applications. Just like in Azure Functions, Logic Apps can be activated by an outside source; for instance, a new message on an Azure Storage Queue. You weave together API calls to connectors to create a (possibly complex) workflow that can involve resources in the cloud and on-premises. Logic Apps has many available connectors to APIs, such as one for connecting to Azure SQL Databases, to SalesForce, to SAP, and so on. You can also expose your own APIs or Azure Functions as connectors to use in a Logic App, making it possible for you to easily perform actions against external systems in your workflow or have your Logic App be activated by one of them.

Just like Azure Functions, Logic Apps are serverless, are scaled automatically, and you pay for them only when it runs.

Here's an example of a workflow in Logic Apps:

1. The Logic App is activated by an email containing a shipping order that arrives in Office 365.
2. Using the data in the email, the Logic App checks on the availability of the ordered item in SQL Server.
3. The Logic App sends a text message to the customer's phone using Twilio (the phone number was also in the email), indicating that the order was received and the item has been shipped.

Try it out Get started with Azure Logic Apps.

Azure App Service

Alternatively, you can host your applications in one of the core service offerings in Azure: Azure App Service. Azure App Service is a collection of hosting and orchestrating services that share features and capabilities. For instance, all App Services have the capability to secure an application using Azure Active Directory and can use custom domains.

Azure App Service comprises the following:

- **Web App:** Web App is one of the most widely used Azure services. You can use this to host your web applications or APIs. A Web App is basically an abstraction of a web server, like Internet Information Services (IIS) or Tomcat, that you use to host HTTP-driven applications.

  Web App can host applications that are written in .NET, Node.js, PHP, Java, or Python, and there are extensions that you can use to run even more languages.

Try it out We walk you through a sample Node.js and MongoDB app in our example section.

- **Web App for Containers:** Web App for Containers helps you easily deploy and run containerized web apps at scale. Just pull container images from Docker Hub or a private Azure Container Registry, and Web App for Containers will deploy the containerized app with your preferred dependencies to production in seconds. The platform automatically takes care of OS patching, capacity provisioning, and load balancing.

Try it out We walk you through deploying a web app with a Docker formatted container in our example section.

- **Mobile App:** Mobile App provides a backend for your mobile applications. You host an API in Mobile App that your mobile applications connect with through the cross-platform client SDK. This is available for iOS, Android, Windows, and Xamarin for iOS and Android and Xamarin Forms. Mobile App provides unique features like Offline Sync and Push Notifications that help you to create a modern, performant, and secure mobile experience.

  You can write your Mobile App backend in .NET or Node.js.

Try it out We walk you through creating a mobile backend in our example section.

Azure App Service Features

Azure App Service is one of the key services in Azure that you can use to host your applications. Each of these services bring unique capabilities to the table, but they all share some common features:

Scaling

Azure App Service runs on App Service Plans, which are abstractions from VMs. One or more VMs run your Azure App Service, but you don't need to know which ones because Azure takes care of them. You can, however, scale the resources that run your Azure App Service. You can either choose a higher pricing tier (ranging from free to premium) or increase the number of application instances that are running. You can even have Azure App Service automatically scale the number of instances for you, based on a schedule or metrics like CPU, memory, or HTTP queue length.

Deployment slots

This is a very useful feature of Azure App Service. You can deploy a new version of your application to a deployment slot, where you can then test whether it works as expected, and then move it into your production slot. You can even use Azure's *Testing in Production* feature to route a percentage of traffic from your production app to a deployment slot. For instance, you could shunt 10 percent of your users to the new version of your app in the deployment slot to see whether the new features are functioning as expected and whether users are actually using them.

After you are satisfied with how the new version of your app is performing in the deployment slot, you can carry out a *swap*, which exchanges the app in the deployment slot with that in your production slot. You can also swap from a development slot to a staging slot, and then to the production slot, as illustrated in Figure 2-1. Before doing this, the swap operation verifies that the new version of your website is warmed up and ready to go. When this has been confirmed, the swap operation switches the slots, and your users now see the new version of the app, with no downtime. If you want, you can also swap back, reverting the deployment of the new version.

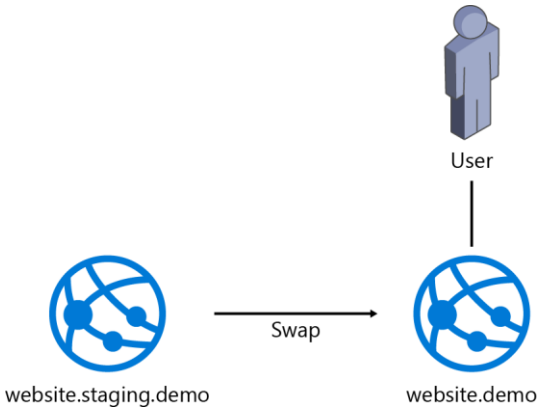

Figure 2-1: Swapping a deployment slot

You use deployment slots *within* an environment like Development, Test, or Production. You don't use deployment slots *as* environments, because they all reside in the same App Service Plan, and you want those to be separated for security, scaling, billing, and performance.

You can swap deployment slots manually, through the Azure command-line interface (CLI) and through the Azure Management API. This allows tools like Visual Studio Team Services to perform swap operations during a release.

A deployment slot is another element of Azure App Service (like a Web App) that runs in the same Azure App Service Plan, next to your original Azure App Service. Because deployments slots run in the same Azure App Service Plan as your original Azure App Service, they don't cost anything extra to use.

Continuous Delivery

To publish your application to App Services, you can use external services such as Visual Studio Team Services, Jenkins, or Octopus Deploy. You also can use the Continuous Delivery (CD) feature in App Services. This makes it possible for you to create a build-test-release pipeline right in the App Service. The process does the following:

1. Retrieves the latest source code from the repository that you indicate
2. Builds the code according a template that you pick (ASP.NET, Node.js, and so on)
3. Deploys the app in a staging environment and load-tests it
4. Deploys the app to production after approval (you can indicate whether you want to use a deployment slot)

Connect to on-premises resources

You can connect external resources such as data stores to your App Services. These resources do not need to be located in Azure; they can be anywhere, such as on-premises, in your own datacenter. You can connect to services on-premises through many mechanisms, depending on your requirements. You can use Azure Hybrid Connections, Azure Virtual Networks, and Azure ExpressRoute to connect to on-premises resources.

Custom domains and Azure App Service certificates

When you spin up an app in Azure App Service, it exposes a URL—for example, https://myazurewebsite.azurewebsites.net. Most likely, you will want to use your own custom domain, which you can do by mapping that domain name to App Services. Here's how to do that.

Additionally, you can ensure that your application is served over HTTPS by using a Secure Sockets Layer (SSL) certificate. You can bring your own certificate or buy one directly from the Azure portal. When you buy an SSL certificate from the Azure portal, you buy an Azure App Service certificate, which you can configure to be used by your custom domain bindings.

Try it out See how to purchase and configure a certificate in this walk-through.

App Service Environment

In a multitier web application, you often have a database or services that are used by your app in Web App. Ideally, you want these services to be exposed only to the app and not to the internet. Of course, the app itself is often internet-facing given that it provides the entry point for your users.

To isolate these support services from the internet, you can use an Azure Virtual Network. This service wraps your support services and connects them to your app in Web App in such as a way that the support services are exposed only to the app, not to the internet. This article describes this service in more detail and shows you how to use it.

Sometimes, you might want even more control. Maybe you want your app to be wrapped in a Virtual Network so that you can control access to it. Perhaps you want it to be called by another app in Web App and have it be a part of your backend. For this scenario, you can use an Azure App Service Environment. This affords you a very high scale and gives you control over isolation and network access. Note, though, that App Service Environment is available for App Services in the premium pricing tiers only.

Note App Service Environment currently doesn't work for Web App for Containers.

What to use when?

Even though App Service, Functions, and Logic Apps often works hand in hand, each is designed with specific application needs in mind.

Table 2-1 briefly outlines the purpose for the category of needs that you might have.

Table 2-1: Which App Service should you use when

Category	Purpose	Azure App Service type
Host applications	Host web applications and APIs	Web App and/or Web App for Containers from App Service
	Host backend for mobile apps	Mobile App from App Service
Orchestrate processes	Orchestrate a step in a process	Functions
	Orchestrate an entire process	Logic Apps

Making your application faster

After your application is up and running in Azure, you want it to be as performant as it can be. Azure provides a range of services that can help you with that.

Azure Content Delivery Network

One of the Azure services that can help you to make your application faster is Azure Content Delivery Network. You upload your static files—videos, images, JavaScript, CSS, and even static HTML files—to a data store such as Azure Blob Storage and then couple Azure Content Delivery Network to that. Content Delivery Network will then take those static files and replicate them to hundreds of Points-of-Presence (PoP) all over the world. All you need to do in your app is to change the reference to the static files to a different URL. For example, where that reference previously might have been ~/images/image.png, it would now be https://example.azureedge.com/image.png. This is very easy to do and improves the performance of your application in the following ways:

- It offloads serving content from your application. It is now served by Content Delivery Network, thereby freeing up processing cycles for your application
- It brings static content physically closer to your users by distributing it to PoPs all over the world

You can benefit from Content Delivery Network in web applications but also in mobile and desktop applications.

An example of using Content Delivery Network is to serve videos for a mobile app. The videos can be large, and you don't want to store them on the mobile device (nor do your users!). Using Content Delivery Network, they are served from the PoP, which also improves performance, because it is close to the user.

Try it out Get started with Azure Content Delivery Network.

Azure Redis Cache

Every modern application works with data. When you retrieve data from a data store like a database, this typically involves scanning multiple tables or documents in some distant server, weaving the results together, and then sending the result to the requesting device. This, of course, takes time and can frustrate and annoy your users.

To eliminate some of these "roundtrips," you can cache data that doesn't change often. This way, instead of querying the database every time, you could retrieve some of the data from a cache, like Azure Redis Cache. The benefit of the cache is that it stores data in a simple format such as key–value. You don't need to run a complex query to get this data, you just need to know the key to retrieve the value. This can improve the performance of your application dramatically. Here's how this workflow operates:

1. The app needs some data and attempts to retrieve it from cache.
2. If the data is not there, get it from the database and also store the data in the cache.
3. The next time the app is looking for that particular piece of data, it will find it in the cache, saving a trip to the database.

Azure provides Cache-as-a-Service with Redis Cache. This is based on the open-source Redis project, and is now backed by Microsoft engineers and SLAs. It is highly performant and has advanced options like clustering and geo-replication.

Try it out Get started with Azure Redis Cache.

Azure Traffic Manager

Many modern applications have users all over the world. Providing a performant experience for everyone is challenging to say the least. The most obvious problem you need to deal with is latency. Latency is the time it takes for a signal or a request to travel to a user. The further away users are from your application, the more latency they experience.

Azure Traffic Manager scales across regions which helps to reduce latency and to provide users a performant experience, regardless of where they are. Traffic Manager is an intelligent routing mechanism that you put in front of, for instance, your Web App applications, all over the world. Web App acts as endpoints, which Azure Traffic Manager monitors for health and performance. As Figure 2-2 demonstrates, when a user accesses your application, Traffic Manager routes her to the Web App application that is most performant in her proximity.

Including Traffic Manager in your architecture is a great way to improve the performance of your application.

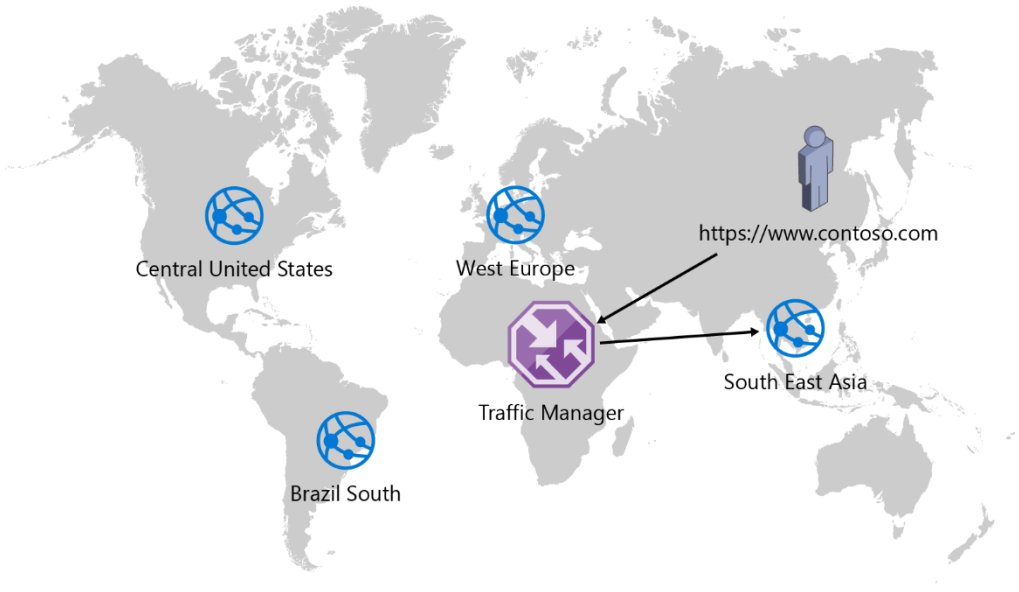

Figure 2-2: Azure Traffic Manager directs traffic to the most geographic performant endpoint

Where to store your data

Data is a very important aspect of any modern application, and it comes in all shapes and sizes. Azure provides many types of data stores that can help you to maintain and retrieve data in any scenario. Table 2-2 presents the storage options available in Azure.

Table 2-2: Storage options in Azure

	SQL Databases	MySQL	PostgreSQL	Cosmos DB	Blob	Table	Queue	File	Disk	Data Lake Store	SQL Data Warehouse
Relational data	X	X									
Object-relational data			X								
Unstructured data				X	X						
Semi-structured data						X					
Queue messages							X				
Files on disk								X			
High-performance files on disk									X		
Store large data					X			X	X	X	X
Store small data	X	X	X	X		X	X	X	X		
Geographic data replication	X			X							

Note You can use almost all of the storage options mentioned in this section as activators and bindings for Azure Functions.

Let's take a closer look at each storage option.

Azure SQL Databases

If you want to use tables with columns and rows to store data, Azure SQL Database is a great choice. It is a relational database system that is similar to on-premises Microsoft SQL Server. Because SQL Database runs in the cloud, it is fully managed, performant, scalable, automatically backed up, and has many advanced featured.

You can use SQL Database with your favorite tools, including SQL Server Management Studio and the Entity Framework.

Here are some of its more advanced features:

- Geo-replication, which replicates data to other geographical regions in real time (Get started with geo-replication)
- Dynamic data masking, which dynamically masks sensitive data for certain users at runtime (Get started with dynamic data masking)
- Auditing, which provides a complete audit trail of all the actions that happen to the data (Get started with Azure SQL Database auditing)

Databases in SQL Database are extremely reliable and robust and offer an Service-Level Agreement (SLA) that guarantees 99.99 percent uptime.

Azure Database for MySQL

MySQL is an open-source relational database system that is used by millions of applications, like WordPress, Joomla, and Drupal. Now, MySQL is available as a managed database service in the Azure Database for MySQL service. This means that you can get up and running quickly, without the need to set up or maintain servers. Azure does it all for you. You get security and scalability right out of the box. Also, the Database for MySQL service enjoys the same reliability as that of SQL Database, providing an industry leading 99.99 percent uptime SLA.

Just as with SQL Database and Azure Database for PostgreSQL, incremental backups are made every five minutes, and full backups are performed every hour, which you can use to recover your data to an earlier state as far back as 35 days.

Try it out Create and connect to a MySQL database in Azure.

Azure Database for PostgreSQL

Based on the popular PostgreSQL database technology, Azure Database for PostgreSQL provides an object-relational database service. This is slightly different to relational database services in that object-relational databases can store more complex data types, such as nested data types, and are therefore more closely aligned to the object-oriented programming languages that we use in our applications.

Database for PostgreSQL is a managed service that has the same characteristics as Database for MySQL. You can scale it up and down, it is highly available (99.99 percent SLA), and it is automatically backed up.

Try it out Create an Azure Database for PostgreSQL using the Azure CLI.

Azure Cosmos DB

Azure Cosmos DB is the new version and rebranding of DocumentDB—and more. Cosmos DB is a new kind of database that is truly made for the cloud. Here are some of its key features:

- A 99.99 percent SLA that includes low latencies (less than 10 ms on reads; less than 15 ms on writes).
- Geo-replication, which replicates data to other geographical regions in real time (How to distribute data globally with Azure Cosmos DB).
- Traffic Management, which sends users to the data replica to which they are closest.
- Limitless global scale; you pay only for the throughput and storage that you need.
- Automatic indexing of data. No need to maintain or tune the database anymore.

In addition to all of these features, Cosmos DB offers different APIs with which you can store and retrieve data, including SQL, JavaScript, Gremlin, MongoDB, and Azure Table Storage. Different APIs handle data in different ways. You can use documents as data as well as unstructured tables, graphs, blobs, and so on. You use the API that fits your needs, and Cosmos DB takes care of the rest. You benefit from cloud-grade performance, scalability, and reliability, and still use the programming model to which you're accustomed.

Try it out Get started with Azure Cosmos DB DocumentDB API.

Azure Storage

Another option you have for storing data is Azure Storage. This is one of the oldest and most reliable, performant services in Azure. Azure Storage offers five types of storage that all benefit from the following shared features:

- Geo-redundancy that replicates data to different datacenters so that you can recover it in the event that a disaster causes an individual datacenter to fail
- Encryption of data at runtime
- Custom domains

The four Azure Storage types are *Blob*, *Queue*, *File*, and *Disk* (Figure 2-3).

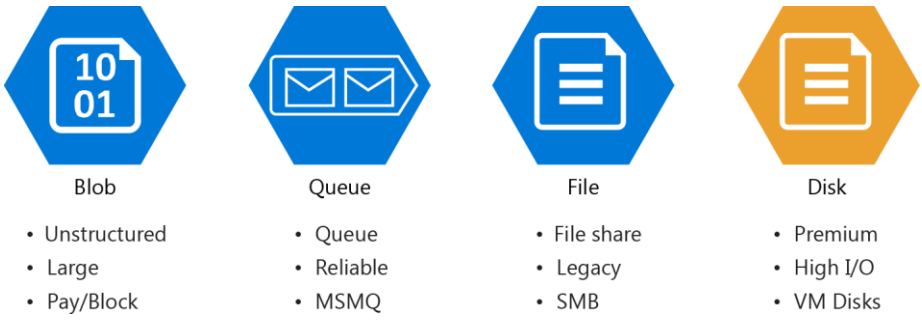

Figure 2-3: Azure Storage services overview

Blob Storage

You can use Azure Blob Storage to store large unstructured data—literally, blobs of data. This can be video, image, audio, or text, or even virtual hard drive (VHD) files for VMs.

There are two types of Blobs: Page and Block Blobs. Page Blobs are optimized for random read and write operations. These are perfect for storing a VHD. Block Blobs are optimized for efficiently uploading large amounts of data. These are perfect for storing large video files that don't change often.

Try it out Get started with Azure Blob Storage.

Queue Storage

Azure Queue Storage is an unusual type of storage in that it is used to store small messages of data, but its main purpose is to serve as a queue. You put messages on the queue and other processes pick it up. This pattern decouples the message sender from the message processor and results in performance and reliability benefits. Azure Queue Storage is based on the Microsoft Message Queueing that you can find in previous versions of Windows.

Try it out Get started with Azure Queue Storage.

File Storage

You can use Azure File Storage as a drive from which to share files. Its uses the Server Message Block (SMB) protocol, which means that you can use it with Windows and Linux, and you can access it from the cloud or from on-premises systems. Like the other Azure Storage types, File Storage is scalable and inexpensive.

Try it out Get started with Azure File Storage.

Disk Storage

Azure Disk Storage is similar to File Storage, but is specifically meant for high I/O performance. It is perfect for use as a drive in a VM that needs high performance, like a VM that runs SQL Server. Note, though, that Disk Storage is available only in the premium pricing tier of Azure Storage.

Azure Data Lake Store

The previous data stores were meant for regular application use or for use with VMs. Azure Data Lake Store is as storage for big data applications. You use it to store large amounts of data in its native format—structured, unstructured, or anything in between. The point of the Data Lake Store is to hold your raw data so that you can analyze it or transform and move it. Following are the main characteristics of Azure Data Lake:

- Unlimited storage capacity. A single file can be larger than one petabyte in size—200 times larger than other cloud providers offer.
- Scalable performance to accommodate massively parallel analytics.
- You can store data in any format, without a schema.

This is a very different approach from the traditional data warehouse, in which you define data schemas upfront.

For instance, you could use a Data Lake Store to store all of the data that you get from your Internet of Things (IoT) devices that are collecting temperature data. You can leave the data in the store and then filter through it and create a view of the data per hour, or per week. Storing the data in Data Lake Store is quite inexpensive, so you can keep years of data there at a very low cost.

Try it out Get started with Azure Data Lake Store using the Azure portal.

Azure SQL Data Warehouse

You use Azure SQL Data Warehouse when you need a traditional data warehousing solution that is completely managed, scalable in size, and performant and secure. You store data in predefined schemas and query it by using the familiar SQL Server SQL dialect.

Because SQL Data Warehouse runs in Azure, you can use advanced features like automatic threat detection that uses machine learning to understand the patterns of your workload and serve as an alarm system to alert you of a potential breach.

An example of using SQL Data Warehouse is when you know which reports you want to show to users and know what the data schema for these reports is. You can then create schemas in SQL Data Warehouse and populate it with data so that users can navigate through the data.

Try it out Create an Azure SQL Data Warehouse.

Adding intelligence to your application

Hosting your application and data and having it scalable, secure, and highly performant is nice, but wouldn't it be great if you could add intelligent features to it?

Azure Search

Search is a common feature in most applications, and yet it has traditionally been a difficult function to implement. Azure Search provides a lot of the "plumbing" to do search. You spin up an Azure Search instance, create an index that helps you search, and fill it with data—that's it. There are lots of options that you can use to tweak Azure Search, and there are lots of great features that will make searching easier for your users:

- Geo-search that gives users the ability to explore data based on the proximity of a search result to a physical location.
- Language analyzers from Lucene as well as Microsoft's natural language processors (NLPs), available in 56 languages to intelligently handle language-specific linguistics, including verb tenses, gender, irregular plural nouns (for example, "mouse" versus "mice"), word de-compounding, word-breaking (for languages with no spaces), and more.
- Monitoring and reporting that inform you as to what was searched for and how fast and successful the search was.
- User experience features like sorting and paging search results and intelligent filtering, and providing search suggestions.

You could implement Azure Search to help users search your product catalogue in an e-commerce application.

Try it out Create your first Azure Search index in the portal.

Azure Cognitive Services

With Azure Cognitive Services, you can add incredible intelligence to your applications. Cognitive Services is a collection of APIs that you can use to accomplish what at times appears to be magic. There are more than 25 APIs in the categories of Vision, Speech, Language, Knowledge, and Search. Here's a glimpse of the APIs that you can use:

- Using the Face API, you can perform facial recognition and identification and can even determine the color of a person's eyes, even if that person is wearing glasses.
- The Emotion API determine which emotions an individual is showing in a picture.
- The Language Understanding Intelligent Services API (or LUIS) can actually understand text, based on the model you use to train it.
- The Translator Speech API turns speech into text and translates it for you in real time

The Cognitive Services get their smarts through machine learning and artificial intelligence, which you can train with your own datasets. These services are simply amazing and open up possibilities that we couldn't even dream of a mere few years ago.

Azure Bot Service

The Azure Bot Service makes it easy for you to create a *bot*—a piece of software that can automatically and autonomously interact with users.

Creating a bot is no trivial task. You need to keep track of the context of your interaction with the user, and you must be ready to respond to a multitude of possible interaction parameters. Bot Service helps you with this in the following ways:

- It helps you to keep track of the interaction context and provides you with templates to get started from the Bot Framework.
- It has tight integrations with Cognitive Services, making it easier to make your bot "smart."
- It helps you to integrate with services like Facebook, Slack, Microsoft Teams, Telegram, and so on.
- It offers all the benefits of a managed service in Azure: massive scale, built-in Continuous Delivery (CD), and you pay only for what you use.

An example of a bot that you can build with Bot Service is one that provides your users with answers to their most frequently asked questions. The interface of the bot can be a chat box that is on your website.

Try it out Getting Started Chatbot using Azure Bot Service.

Using events and messages in your application

Modern, globally distributed applications often must deal with large amounts of messages coming in and need to be designed with decoupling and scaling in mind. Azure provides several services to help with event ingestion and analysis and messaging patterns.

Azure Service Bus

The core of messaging in Azure is the Azure Service Bus. The Service Bus service encompasses a collection for services that you can use for messaging patterns, most important among them Azure Services Bus Queues and Topics.

Try it out Gets started with Azure Service Bus Queues and Topics.

Azure Service Bus Queues

You use Azure Service Bus Queues to decouple systems from one another. Here's an example: A web application receives orders from users and needs to invoke a web service to process the order. The web service will take a long time, maybe up to five minutes, to process the order completely. Of course, it's unacceptable for the web application to wait five minutes between the user placing an order and showing feedback. A good way to solve this is to use a queue to decouple the web application from the web service.

The web application receives the order and writes it in a message on an Azure Service Bus Queue. After it has done that, it can inform the user that the order is being processed. The web service takes messages from the queue, one by one, and processes them. When the web service is done processing an order, it sends an email notification to the user that the item has been ordered.

By decoupling the systems, the web application can work at a different speed from the web service, and both can be scaled individually to the applications' needs.

Services Bus Queue is a simple mechanism. Multiple applications can put messages on the queue, but a queue message can be processed by only one application at a time. It has some clever features to work with messages on the queue, like duplicate detection and a dead-letter subqueue to which messages are moved when they fail to be processed correctly.

Azure Service Bus Topics

Just like Service Bus Queues, Azure Service Bus Topics are a form of application decoupling. Here's the differences between the queue and topics:

- With a queue, multiple applications write messages to the queue, but *only one application* at a time can process a message.
- With a topic, multiple applications write messages to the topic, and *multiple applications* can process a message at the same time.

Service Bus Topics works just like a queue, and multiple applications can process the same message. Applications can create a subscription on the topic, that indicates what type of messages they are interested in. Just like queues, topics have features like duplicate detection and a dead-letter subqueue to which messages are moved when they fail to be processed correctly.

Comparing Service Bus Queues and Azure Storage Queues

There are Service Bus Queues, but there is also Azure Storage Queues. They basically do the same thing, but there are subtle differences between them. Table 3-1 looks at each of them.

Table 3-1: A comparison of features between Service Bus Queues and Storage Queues

Azure Service Bus Queues	Azure Storage Queues
Message lifetime >7 days	Message lifetime <7days
Guaranteed (first in–first out) ordered	Queue size >80 GB
Duplicate detection	Transaction logs
Message size ≤1 MB	Message size ≤64 KB

Azure Event Hubs

In our world of ubiquitous software, data is being generated from many applications, devices, and locations, all the time. Today, it's becoming more common for enterprises to capture massive amounts of that data to analyze it or transform and move it for later use. Azure Event Hubs can help.

Event Hubs is designed for massive data ingestion. You can throw millions of messages per second at it, and it will handle that data for you effortlessly. It can retain messages for up to seven days, or retain them indefinitely by writing them to a data store using the Event Hubs Capture feature.

You can use Event Hubs to filter the data with queries, as it comes in, and output it to a data store like Azure Cosmos DB. You can even replay messages if you need to.

Try it out Get started sending messages to Azure Event Hubs in .NET Standard.

Azure IoT Hub

Just like Event Hubs, Azure IoT Hub is built for massive data ingestion. It is specifically geared toward handling the enormous volume of data messages from devices on the Internet of Things (IoT; refer to the sidebar in Chapter 1, "Who uses Azure?" for an example), like smart thermostats or sensors in cars. It has many of the same properties as Event Hubs, like the ability to retain messages for up to seven days and replay them.

What makes IoT Hub unique is that in addition to receiving messages from devices, it can send messages to them, as well. It has the ability to manage your complete IoT infrastructure. You can use IoT Hub to register devices and report their state, manage them by securing them and restarting them and sending data to devices.

Try it out Connect your device to your IoT hub using .NET.

Azure Event Grid

Azure Event Grid, offers a different type of messaging. Event Grid offers a fully managed, publish/subscribe service that hooks into almost every service in Azure and can hook into custom publishers and subscribers.

This is different from working with the Service Bus Queues and Topics, for which you'd need to poll the Queue or Topics for new messages. Event Grids automatically pushes messages to subscribers, making it a real-time, reactive event service.

Services in and outside of Azure publish events (for example, when a new Blob is added or when a new user is added to an Azure subscription). Azure Event Grid can detect these events and make them available to event handlers and services that subscribed to the events (Figure 3-1). Event handlers can be things like Azure Functions or Azure Logics Apps, which can then act on the data in the event.

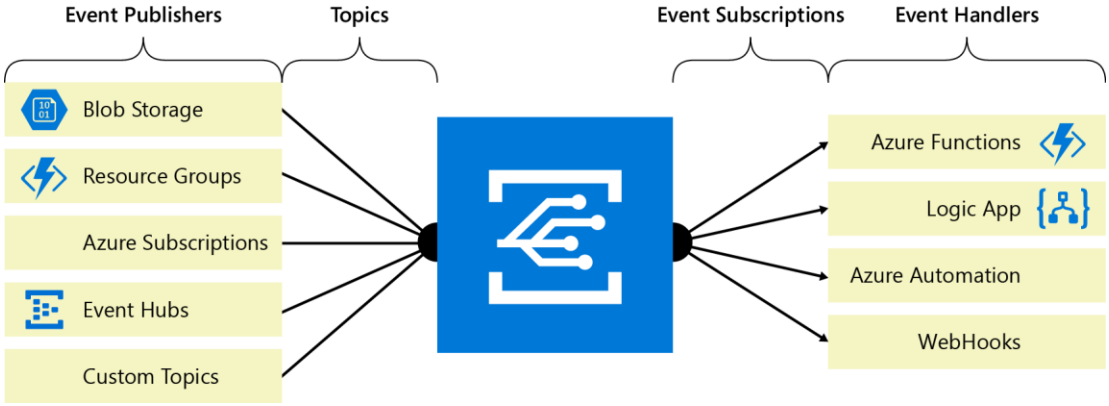

Figure 3-1: Azure Event Grid overview

Another important aspect of Event Grid is that it is serverless. This means that like Azure Logic Apps and Azure Functions, Event Grid scales automatically and doesn't need an instance of it being deployed. You just configure and use it. And, you pay only when it is used, not for it just being there.

Here's a usage example for the Azure Event Grid:

You want to be notified by email every time a user is added or deleted to your mailing list in MailChimp.

Azure Event Grid is used to activate an app in Azure Logic Apps and is configured to listen to changes to the MailChimp mailing list. The next step is to send an email containing the name of the user that has been added or deleted and the action that was performed (add or delete). Now, when a new user is added to the mailing list, Event Grid signals the Logic Apps, which sends an email.

Try it out Monitor virtual machine changes with Azure Event Grid and Logic App.

What to use when?

Azure provides myriad options to do messaging and decouple applications. Which one should you use, and when? Table 3-2 sums up the differences to help you sort it out.

Table 3-2: When to use which Azure service for messages or events

	Event Grid	Event Hubs	IoT Hub	Topics	Service Bus Queues	Storage Queues
Event ingestion	X	X	X			
Device management			X			
Messaging	X	X	X	X	X	X
Multiple consumers	X	X	X	X		
Multiple senders	X	X	X	X	X	X
Use for decoupling		X	X	X	X	X
Use for publish/subscribe	X					
Max message size	64 KB	256 KB	256 KB	1 MB	1 MB	64 KB

Securing your application

Security today is one of the most important aspects of any application, and it is no simple thing to get right. Fortunately, Azure provides many services to help you secure your application. We take a look at some of them in this chapter.

Azure Active Directory

An important part of your application's security is authenticating users before they can use it. Authentication is not an easy thing to implement. You need to store user identities and credentials somewhere, implement password management, create a secure authentication handshake, and so on.

Azure Active Directory provides all of these things and more out of the box. You store your user identities in Azure Active Directory and have users authenticate against it, redirecting them to your application only after they are authenticated. Azure Active Directory takes care of password management, including common scenarios like "I forgot my password."

Azure Active Directory is used by millions of applications every day, including the Azure portal, Outlook.com and Office 365. Because of this, it is able to more readily detect malicious behavior and act on it. For instance, if a user were to sign in to an application from a location in Europe and then one minute later sign in from Australia, Azure Active Directory would flag this as malicious behavior and ask the user for additional credentials through multifactor authentication.

Azure API Management

APIs should be secure. This is true for APIs that you create yourself as well as those that you consume from third-party vendors. To assist in making your APIs secure, you can use Azure API Management. This is basically a proxy that you put in front of your APIs or those from third-party vendors that adds features like caching, throttling, and authentication/authorization.

You can secure your API with API Management by requiring consumers to create a subscription to your API. This way, calling applications need to authenticate before they can use your API. You can use various authentication methods like access tokens, basic authentication, and certificates. Additionally, you can track who is calling your API and block unwanted callers.

Try it out Get started with Azure API Management.

Azure Key Vault

As part of your security architecture, you need a secure place to store certificates, keys, and other secrets. Azure Key Vault provides that. With Key Vault, you have one central location where you store the secrets that your applications use.

These can be the credentials in a `connectionstring` that your application uses. Your application would get the `connectionstring` from Key Vault instead of from the configuration system. This way, administrators can control the secrets, and developers never need to deal with them.

You also use Key Vault to store certificates like Secure Sockets Layer (SSL) certificates that you use to secure the traffic to and from your applications over HTTPS.

Try it out Get started with Azure Key Vault.

Azure DDoS protection

You've heard it too many times on the news, and you certainly don't want it to happen to your enterprise: your application is attacked by a Distributed Denial of Service (DDoS) attack. These types of attacks are becoming more common and can overwhelm your application to the point that nobody can use it anymore.

Azure protects you from DDoS attacks with its Azure DDoS protection layer. This is a layer in the Azure physical network that is not accessible to customers. This is part of the Azure platform, and you get it for free with all your services. You don't need to do anything or configure anything to use it. It is always there to protect you.

More info Read more about Azure DDoS protection.

Azure Web Application Firewall

You need to secure your application against many possible threats, including those defined in the Open Web Application Security Project (OWASP) specification, such as SQL injection and Cross-Site Scripting (XSS). Azure Web Application Firewall can lend a hand with that. Web Application Firewall, a feature of the Azure Application Gateway service, provides real-time protection of your application.

It can detect a malicious attack, as defined in the OWASP core rule set, and block that attack from reaching your application. It also reports on the attacks that were attempted or are ongoing so that you can see which threats are trying to get to your application. This provides an extra layer of security that protects your applications.

Default encryption of data

You protect your data stores as well as you can by having secure `connectionstrings` and firewall rules in place. But if an attacker somehow is able to gain access to your data, you want to render it useless via encryption.

By default, your data is encrypted when you store it in Azure SQL Database, Azure SQL Data Warehouse, Azure Database for MySQL, Azure Database for PostgreSQL, Azure Storage, Azure Cosmos DB or Azure Data Lake Store. And this all works automatically; you don't need to do a thing to configure it or to encrypt or decrypt data when you use it.

5

Where and how to deploy your Microsoft Azure services

The Azure services that you choose to work with determine your deployment options, and vice versa. Therefore, it is important to understand the deployment options that you have in Azure and what their ramifications are.

The DevOps mindset

Delivering fast and reiterating fast is crucial to creating great software. Therefore, your new code should be merged with your team's code, and that should be deployed and tested as fast and as often as possible to see if everything works and that what you've build is what the user needs.

Many of the Azure services that we've looked at so far in this book are able to have code automatically delivered to them, often through a Continuous Delivery (CD) pipeline that you set up within the service.

Besides the native capabilities of Azure services, you can use Visual Studio Team Services to build, test, and deploy your application. You can easily create new build and deployment pipelines in Visual Studio Team Services as well as do things like automate load-testing and swap deployment slots into production.

Try it out Get started with Visual Studio Team Services.

Azure Resource Manager Templates

All of the Azure services that introduced in this guide are based on Azure Resource Manager. We can use this to script our environment into Infrastructure-as-Code (IaC), using Resource Manager templates. These templates are JSON files that describe what you want to deploy and what its parameters are.

You can create Resource Manager templates in Visual Studio and Visual Studio Code using the Azure Resource Group project template. You can also generate Resource Manager templates from the Azure portal by clicking the Automation Script button, which is available on the menu bar of every resource in the Azure portal. This generates the Resource Manager template for the given resource and even generates code for creating the resource using the Azure command-line interface (CLI), Windows PowerShell, .NET, and Ruby.

After you have a Resource Manager Template, you can deploy it to Azure by using Windows PowerShell, the Azure CLI, or Visual Studio. Or, you can automate its deployment in a CD pipeline using Visual Studio Team Services.

You use Resource Manager templates to deploy applications to run on the Azure platform, either in the public cloud or on-premises, on Azure Stack.

A great example of deploying resources to the cloud using Resource Manager is the Deploy To Azure button that you can find in many GitHub repositories, as illustrated in Figure 5-1.

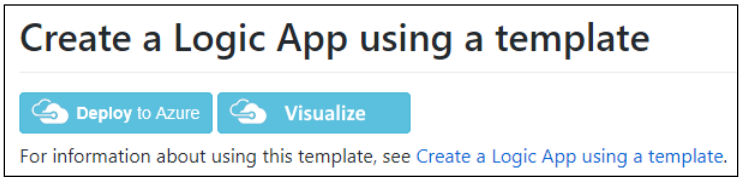

Figure 5-1: One-button deployment of an Azure Logic App using an Resource Manager template in GitHub

Azure Service Fabric

Another way to run your applications is to run them on Azure Service Fabric. Service Fabric is the foundation that Microsoft uses to run many Azure resources and to make them highly available, performant, and self-healing. You can now use Service Fabric yourself, to host your own services.

You can use the Azure Service Fabric SDK to create applications for Service Fabric. You can also run any executable in Service Fabric and you can even use it to host containers.

Service Fabric is amazing at making your applications just as performant, reliable, and secure as many Azure services, and you can use it anywhere: you can deploy Service Fabric in Azure, on-premises, on your own computer, and even on Virtual Machines (VMs) in other clouds.

After you have deployed your application in Azure Service Fabric, it provides you with benefits such as these:

- Load balancing
- Automatic scaling
- High availability
- Self-healing
- Replication and failover
- Rolling upgrades
- And automatic rollback

Containers in Azure

Containers is one of those technology buzzwords that flies around the news. But, they are more than just buzz—they are actually very useful for running your applications. A container is basically a light-weight VM that starts and stops much faster than a VM and is therefore much more useful for development, testing, and running applications in production.

The major benefit that you derive from containers is that an individual container is always the same. You run a container locally when you develop your app, and you use the same container configuration in the cloud or anywhere else. Your entire team uses the exact same container configuration, so you know that the infrastructure is the same for everybody *and* in production. With containers, the age-old developer's fallback statement, "works on my machine," now means that it will also work in production.

There are many technologies for running containers, and Docker is one of them. Azure can run and manage containers for you with Azure Container Instances and Azure Container Service, and even in Services Fabric and Azure App Service running on Linux.

Table 5-1 shows which service you can use for a given scenario when using containers.

Table 5-1: Choosing which Azure service to use for containers

	Azure Container Services	Azure Container Instances	Azure Service Fabric
For production deployments of complex systems (with a container orchestrator)	X		
For running simple configurations (possibly without orchestrator)		X	
For long-running workloads on containers	X		
For short-running workloads on containers		X	
For orchestrating a system based on containers	X		X
Orchestrating with open-source orchestrators (DC/OS, Docker Swarm, Kubernetes)	X		
Orchestrating with built-in orchestrator			X

Note Keep in mind that when you use containers, you are using an infrastructure as a service (IaaS) product and that you are responsible for the operating system (OS), patching, load balancing, and so on.

Azure Stack

If you need your applications and data to remain on-premises, but you still want to benefit from the power that Azure has to offer, Azure Stack is the product for you. Unique in the industry, Azure Stack is an extension of Azure that you host in your own environment. Essentially, it is Azure-in-a-box.

You use Azure Stack in the same way as you do Azure, with the same Azure portal experience and the same APIs with which you can use the Azure CLI, Windows PowerShell, or your favorite IDE.

You can run things like Azure App Service and Azure Virtual Machines on Azure Stack. Everything is exactly the same as in the public cloud, only now you are running it on-premises. And if you decide to move to the public cloud, you simply push services from Azure Stack to Azure.

Where to deploy, when?

Table 5-2 summarizes the deployment options for Azure. Note that using all of the platform as a service (PaaS) services that we have discussed in this guide is possible only when using the Public Cloud with Resource Manager templates or on-premises with Azure Stack.

Table 5-2: Comparing Azure deployment options

	IaaS	PaaS
On-premises	Azure Stack Containers Service Fabric	Azure Stack Service Fabric
Public cloud	Containers Service Fabric	Service Fabric Resource Manager templates

A walk-through of Microsoft Azure

Now that you've know what Azure is and have learned about the services that is has to offer, let's begin using it.

Walk-through #1: the Azure portal

One of the most important Azure tools is the Azure portal, which you can find at https://portal.azure.com/. This is your central Azure hub where you do everything with Azure that you want. Most of the things that you can do in the Azure portal you can do through the Azure API, the Azure command-line interface (CLI), and Azure PowerShell, as well.

You can use a preview of the Azure portal by visiting https://preview.portal.azure.com. The preview already contains features that aren't generally available yet that you can try out and provide feedback on. The rest of this guide will focus on the generally available version of the portal, not the preview.

The first thing you'll see in the Azure portal is a dashboard with tiles. You can create and customize dashboards and share them with team members or keep them just for yourself.

Tiles in the Azure portal

Dashboards contain tiles that display information for a service or act as a shortcut to a service. You can find these tiles, shown in Figure 6-1, all over the portal, in the pages of all the services. They can be very useful to get a quick overview of how a service is doing.

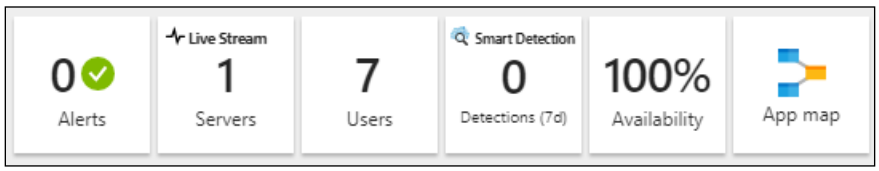

Figure 6-1: Informational tiles in the Azure portal

For instance, The Application Insights service, by default, shows tiles that inform you if there are any alerts active, if there is any live data coming in, how many users have been active in the past 24 hours, what the availability has been, and so on. This is very useful information in a very consumable format. You can customize the size and information of tiles. You can also customize the appearance of charts by adjusting their timelines, and having them show data in different formats such as lines or bars.

You can also pin tiles (Figure 6-2) directly to your dashboards so that they are the first thing that you see when you enter the portal. You can, for instance, pin tiles from all the service metrics that you care about, to create a monitoring dashboard that you share with your team, or display on a monitor in the team room.

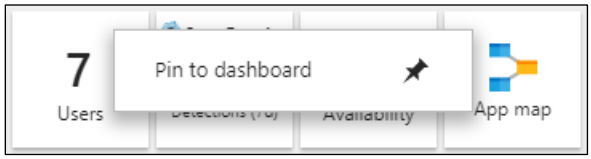

Figure 6-2: You can pin tiles to a dashboard

Finding services

Azure services are the central subject of the Azure portal. You can add and find them in several ways.

To create new services, in the upper-left corner of the portal window, click the plus sign ("+"). This opens the search box for the marketplace, where you'll find everything from Web App to Linux Servers, as depicted in Figure 6-3.

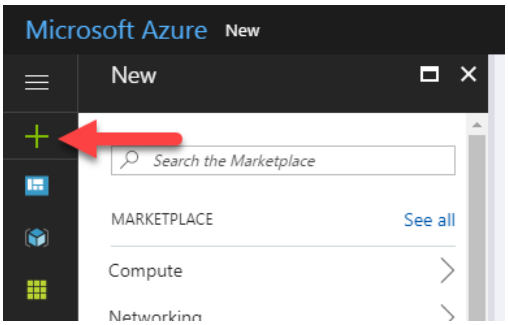

Figure 6-3: The New Resource button in the Azure Portal

When you've found the service that you want (Figure 6-4), a wizard takes you through configuring and deploying it.

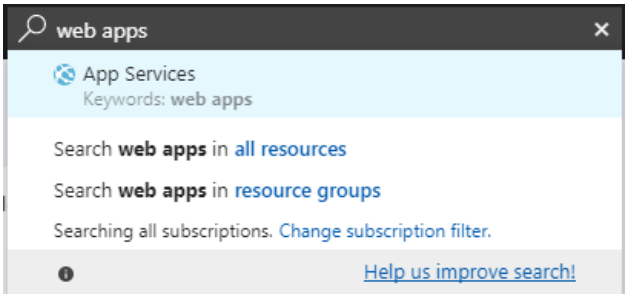

Figure 6-4: Resource search results

When you have some resources, you can find them through search. You can use the search box at the top of the portal (Figure 6-5) to search through all of your resources and take you directly to them.

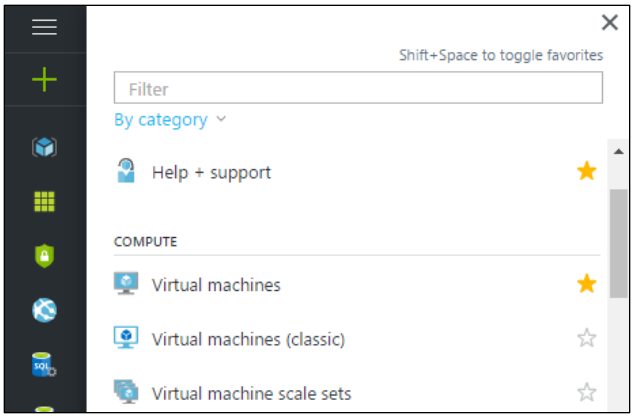

Figure 6-5: The favorites menu in the Azure portal

In the pane on the left side of the portal is the favorites menu. This menu displays the resource categories (represented by their icons) such as Azure App Service. You can rearrange the icons by dragging them up and down. You also can select which ones you want to see by expanding the favorites menu and clicking the stars of those categories.

Understanding Blades

Pages in Azure are also called *blades*. Blades are everywhere, and you can even pin them to your dashboards. When you open a web app, you first see the overview blade. This particular blade provides you with tools to stop, start, and restart the web app and displays tiles showing its metrics, such as number of requests and errors. When you choose another menu item, a new blade opens. Blades always open in context. So, if you open the Deployment Slots blade and click the Create New Deployment Slot button, a new blade shows up to the right of the Deployment Slots blade, preserving the context you are in.

Creating a new virtual machine

Let's use the Azure portal to create a new virtual machine (VM) that runs Red Hat Linux. After creating it, we will show you how to shut it down and remove it so that you are not paying for it anymore.

A word about resource groups

The VM will be deployed in a *resource group*. All Azure resources reside inside resource groups. A resource group is a logical container that holds your resources. You can manage security of a resource group and you can see what the resources in the group cost. It is a common practice to bundle related services together in a resource group so that they are easy to secure and you can easily find out what they cost.

1. In the Azure portal, in the upper-left corner, click the Create A New Service button.
2. In the Search box, type **Linux virtual machine**.

   You'll get many search results, including Red Hat Enterprise Linux. Let's try out its latest version.
3. Click the latest version of Red Hat Enterprise Linux.

4. Click Create.

   The Create Virtual Machine Wizard opens.

   a. Choose a name for the VM.

   b. Choose the drive type. SSD provides a faster VM, but is also more expensive. For this walk-through, choose SSD.

   c. Type a username.

   d. Select Password for the Authentication type.

   e. Type a password and confirm.

   f. In the Resource Group box, type a new name.

   g. Pick the location of the VM, and then click OK.

5. Choose the size of the VM.

   The performance of the VM determines how much you pay for it. There are many sizing options for VMs, some small, some incredibly large. You can use the wizard to select how many cores and how much memory you want and choose options based on that. Besides that, there are other features that come with size options:

   a. Type of hard drive (SSD or normal HDD).

   b. The amount of max IOPS (Input/Output Operations Per Second). This determines the performance of the VM in a significant way, especially if the applications that you run read and write a lot from and to the hard drive.

   c. The amount of data drives that can be installed in the VM.

   d. The ability to do load balancing.

   e. The graphics card that is installed in the VM. This is useful if you need to do a lot of graphics rendering or heavy computational workloads.

6. After you've selected the size, you can configure additional settings such as the Virtual Network and IP address and installing extensions on the machine. For now, leave everything as is and click OK.

7. Review the summary, agree to the terms, and then click Purchase.

   The VM will now be deployed. This usually takes just a few minutes.

Connect to the VM by using Secure Shell

Now that you've created a VM with Red Hat Linux on it, you can connect to it and begin using it. You can use various tools like Secure Shell (SSH) or even remote desktop into the VM. We are going to use SSH to connect to the VM using the Cloud Shell in the Azure portal.

Cloud Shell is the Azure CLI in the browser—there is no longer a need for you to install anything on your PC. Cloud Shell uses the Azure CLI version 2.0. You can use Bash or PowerShell in the Cloud Shell to interact with your Azure resources.

To start the Cloud Shell, in the upper-right of the Azure portal, click the button shown here:

To connect to the VM, you need its IP address. Find the VM in the Azure portal (either by searching for it, or navigating to it). In the Overview blade of the VM, in the upper-left corner click the connect button (see Figure 6-6). When you click it, the blade displays the IP address to use to connect with the VM, as shown in Figure 6-6. Copy the IP address.

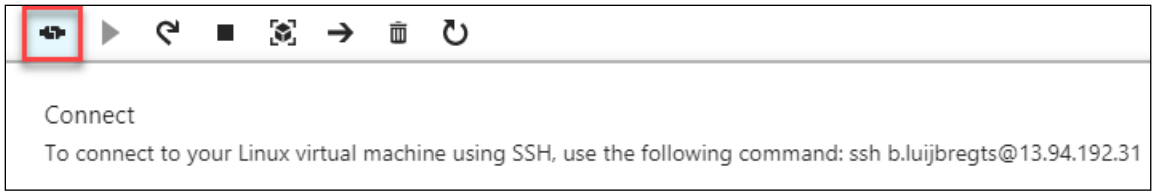

Figure 6-6: Connect to a VM in the Azure portal

Now, you can use this command directly in the Cloud Shell: just paste it and press Enter. You will see a warning that there is no authentication key. This is because we chose to work with a password, instead. Choose Yes, type your password, and you are in! Now you are connected to Red Hat Linux enterprise and you can begin using it. Figure 6-7 shows that we are signed in to the Virtual Machine through Cloud Shell and traversing the file system:

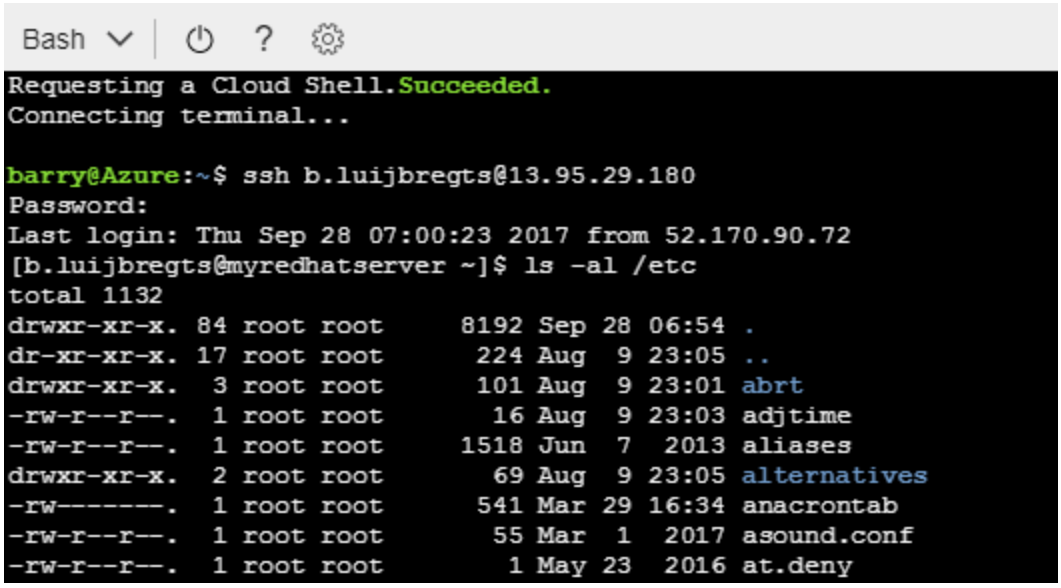

Figure 6-7: Traversing the Linux file system using the Cloud Shell in the Azure portal

Add additional storage to the VM

If you are using your VM as a database server, you often want to have extra storage space to handle database data. You can create extra storage space by adding an additional drive through the Azure portal.

1. Go to the VM, select the Disks tab, and then click Add Data Disk, as shown in Figure 6-8.

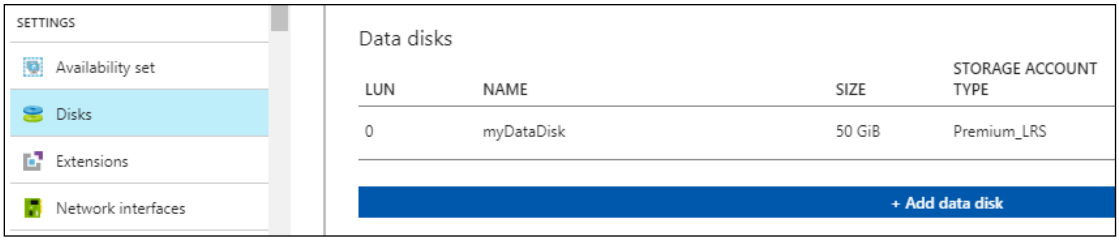

Figure 6-8: The Disk tab of a VM in the Azure portal

Because we have no existing disks to attach, select Create Disk.

2. The Create Managed Disk Wizard opens. Use this wizard to create a managed drive (instead of an unmanaged drive).

   Managed disks are completely managed by Azure; you don't need to create and maintain an underlying storage account for them. Optionally, you can choose the size of the drive and what should be on it, but we are going to leave everything as is. Type a name, and then click Create. You now have an additional drive attached to your VM.

   You can also create and attach a new drive via the Cloud Shell or Azure CLI by using this command:

```
az vm disk attach -g myResourceGroup --vm-name myVM --disk myDataDisk --new --size-gb 50
```

If you need help with the Azure CLI, you can type **az -h**. All Azure CLI commands follow the same structure: `az noun(s)` (e.g., `vm`) `verb` (e.g., `attach`) `options`.

After you've created a new drive for your VM, you need to initialize it in Linux. This document guides you through the process.

Creating additional drives for extra VM storage is simple. The steps after that can be more complicated, depending on the OS that you use.

Clean up the walk-through resources

If you are done with the VM, you can shut it down and remove by deleting the resource group that we created at the outset of creating the VM. This contains the VM and all the other resources that are automatically created. After the resource group is deleted, you no longer pay for any of the resources that you've used in this walk-through.

Walk-through #2: developing a Node.js web app with MongoDB on Azure

In this walk-through, we deploy a simple Node.js application that connects with MongoDB and host that in Azure. We will host the application in Azure Web App and use Azure Cosmos DB as our database, because that has a MongoDB interface.

To follow along you'll need to have the Azure CLI (v2.0 or higher), Git (v2 or higher), Node.js and NPM (v6 or higher), and MongoDB (3.4 or higher) installed on your computer.

We'll use a sample application that uses the MEAN.js web framework.

Creating the web app and database using the Azure portal

To host the Node.js application, we'll create a new web app in Azure Web App.

1. In the Azure portal, click the Create A New Service button.
2. Search for Web App. The Web App blade opens. Click Create to get started.
3. The Web App Create blade opens.
    a. Type a name for the Web App.
    b. Create a new resource group by giving it a name.
    c. Select the OS to run on (Windows or Linux).
    d. Select or create an App Service Plan, and then click Create.

Web App runs on App Service Plans

App Services, like Web App, run on Azure App Service Plans. An App Service Plan is an abstraction of resources and features, like CPU and memory and are represented in pricing tiers. App Service Plans are also bound to a specific geographical region that you choose. You can, for instance, run your Web App application in an App Service Plan of pricing tier S1 (see Figure 6-9), which has 1 core and 1.75 GB RAM.

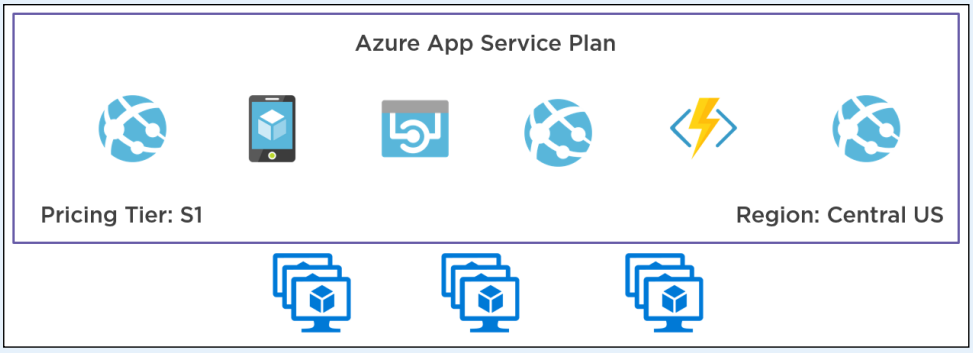

Figure 6-9: Azure App Service Plan explained

You can run as many App Services on an App Service Plan as you want, as long as you realize that you need to share the resources amongst all of the App Services.

To host the database, we'll create a Cosmos DB database that uses the MongoDB API. This will work exactly the same as MongoDB would and it has all of the benefits that Cosmos DB offers.

1. In the Azure portal, click the Create A New Service button.
2. Search for Cosmos DB. The Cosmos DB blade opens. Click Create to begin.
    a. Type a name (ID) for the database account.
    b. Choose the MongoDB API.
    c. Select the resource group that you created for the web app.

d. Choose the location. Preferably, a location that is the same or close to where the web app is.

e. Click Create.

Run the Node.js application and MongoDB database locally

Let's run the app locally, before we run it in Azure.

If you haven't done so already, install MongoDB Community edition:

1. Open a command prompt on your computer and navigate to the bin directory of your MongoDB installation. This could be something like C:\Program Files\MongoDB\Server\3.4\bin.
2. Type **mongo** to connect to the local MongoDB server.
3. If it fails, make sure that your MongoDB database is started. Often, you need to start it by running `mongod`.

Now to clone the Node.js starter application and run it:

4. In the command window, navigate to a directory to which you want the application source to be downloaded.
5. Run `git clone https://github.com/Azure-Samples/meanjs.git` to clone the application.
6. Start the application by running the following commands:

```
cd meanjs
npm install
npm start
```

7. When it is fully loaded, you'll see a message that contains the application URL. Open a browser and navigate to the server address (something like http://localhost:3000). This should look similar to Figure 6-10.

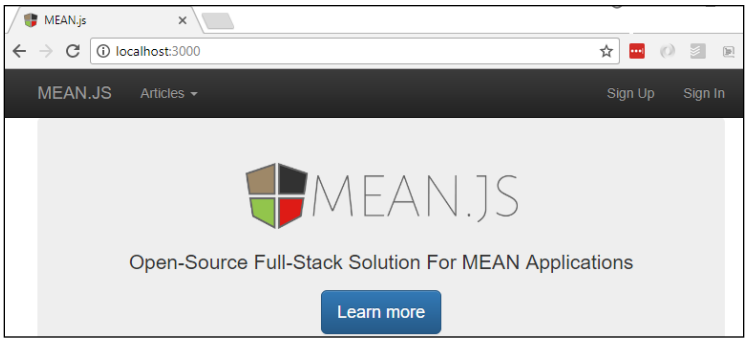

Figure 6-10: Mean.JS boilerplate application

8. On the menu bar, click Sign Up, and then try to create a new user. If it works, the app has access to the local MongoDB database, and it uses in which to write users.

Deploy the Node.js application to Azure

You now have a working application running locally. Let's deploy it to Azure. We'll begin by deploying the Node.js application to the web app in Web App that we created earlier.

We are going to use the Azure CLI to push the application from Git to Azure. Make sure that you have the Azure CLI 2.0 or newer installed on your computer. When that is done, you need to run the

following `az login` command to log in to your azure subscription. This opens a website where you can authenticate and authorize the usage of your Azure subscription by the CLI.

For FTP and local Git, you must have a deployment user configured on the server to authenticate your deployment. This deployment user is account-level and is different from your Azure subscription account. You need to configure this deployment user only once. In the Azure CLI, run the following command, where *username* and *password* are values that you make up and fill in (do not include the < or > character):

```
az webapp deployment user set --user-name <username> --password <password>
```

Next, we associate the local Git repository with the web app by running the following:

```
az webapp deployment source config-local-git --name <app_name> --resource-group
<resource_group_name>
```

This will return a URL in the form of:

```
https://<username>@<app_name>.scm.azurewebsites.net:443/<app_name>.git
```

We will now use this URL to push the application to Azure. First, we add an Azure remote to the local Git repository. cd to the meanjs directory of the application source code. Now run:

```
git remote add azure <url_for_deployment>
```

To deploy the app, run `git push azure master`. This prompts you for the credentials that you created earlier. The deployment can take a while. It will notify you when it's done.

After it's done, you can browse to the application using http://<app_name>.azurewebsites.net.

Now, the application is deployed to the web app and we have a Cosmos DB database running with the MongoDB API, but the two aren't connected yet. Let's do that next.

Connect the application and the database

To connect the application to the database, it needs to know where the database is. We can do that by setting the environment variable `MONGODB_URI` in the web app. We can do that using the Azure CLI by running the following command:

```
az webapp config appsettings update \
--name <app_name> \
--resource-group myResourceGroup \
--settings
MONGODB_URI="mongodb://<cosmosdb_name>:<primary_master_key>@<cosmosdb_name>.documents.azure.com:102
50/mean?ssl=true"
```

You can find the Cosmo DB connection string that you need to use for the `MONGODB_URI` value in the Azure portal. There, you can find it in the Connection String menu of the Cosmos DB account.

After running the command with the Azure CLI, the Application Settings of the web app will look like Figure 6-11.

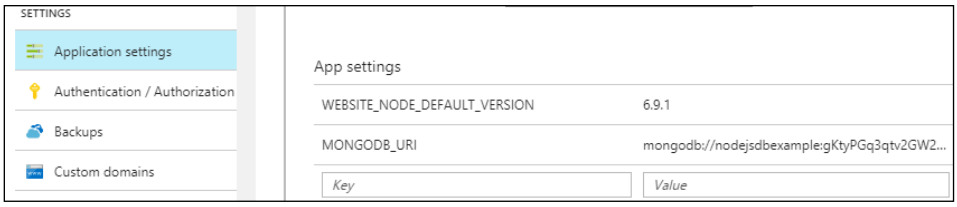

Figure 6-11: Application Settings of the Azure Web App

Instead of running the Azure CLI command, you could've also added the setting using the Azure portal.

Now, navigate to the web app again using http://<app_name>.azurewebsites.net. Try signing up a new user. When that succeeds, the application is communicating with the database. Now you have a Node.js application with an Azure Cosmos DB using the MongoDB API.

Setting up Continuous Delivery with GitHub

So far, we've been pushing code from our local Git repository to Azure. This is fine if you work by yourself, but if you work in a team, you will need another type of source control, like Visual Studio Team Services or GitHub.

We are going to use GitHub to push our code to and link that to our web app so that changes are deployed automatically in a Continuous Delivery (CD) pipeline.

1. Go to https://github.com/ and create a new repository.

   You need to sign in or create a new account to do this.

2. Create a new repository by going to https://github.com/new.

   a. Type a name for the repository.

   b. Leave the other settings as they are (public repository, don't create a README)

   c. Create the repository. This results in a screen that should look similar to Figure 6-12

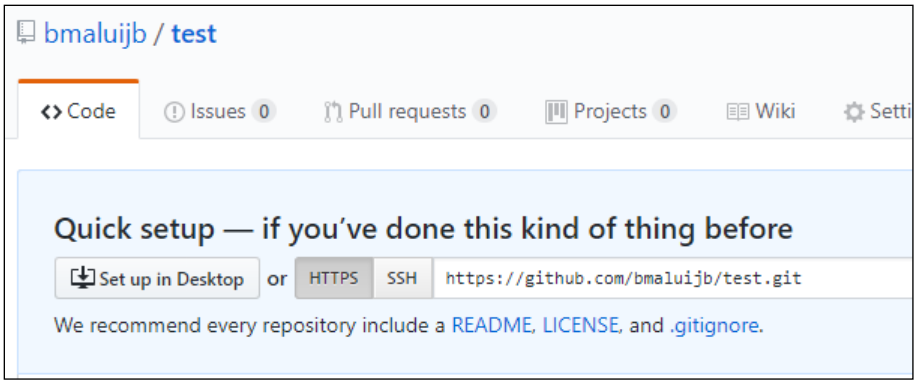

Figure 6-12: Repository link in GitHub

3. Use the URL to set a remote destination for the local Git repository. You can do that in the command window.

   a. `cd` to the meanjs directory of the application source code.

   b. Run the following command:

   ```
   git remote add github https://github.com/bmaluijb/test.git
   ```

   c. Run the following command to push the code to GitHub:

   ```
   git push github
   ```

Now, our code is in GitHub, and you can share it with your team. Next, let's set up CD. We'll use the Deployment Options feature of Web App through the Azure portal. We could've also used the Continuous Delivery feature in Web App, but that requires a Visual Studio Team Services account, and we want to keep it simple.

1. In the Azure portal, go to the web app that hosts the Node.js application.

2. On the menu bar, click Deployment Options.

3. It's possible that this is already configured for the local Git Repository. If this is the case, you need to click Disconnect.
4. In Choose Source, select GitHub.
5. In the Authorization section, authorize Azure to use GitHub by clicking Authorize and granting permission.
6. In the Choose Project section, choose the GitHub repository that you just created.
7. Leave the Branch set to master
8. Click OK and wait for a few seconds.

Go back to the Deployment Options menu. You now can see that it is connected. From this point, whenever you push a new version of source code to GitHub, it will be built and deployed to the web app automatically, as demonstrated in Figure 6-13. You can also force this process by clicking the Sync button.

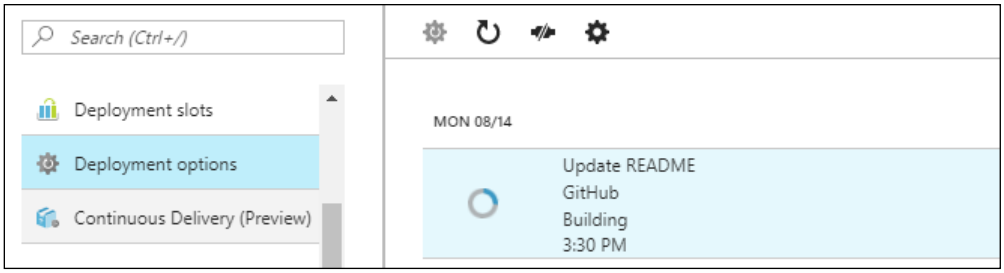

Figure 6-13: The Deployment Options blade of the Web App

Setting up staging environments

Using Web App, you can set up a staging slot in which to test the new version of your application. You can use deployment slots for this. Deployment slots are full App Services on their own which you can use to test your code before you promote it to the next slot.

You can have deployment slots for staging, load-testing, and production (which is always the original App Service, so the Node.js web app in our case). In fact, you can have as many deployment slots as you want, without incurring any additional costs. The deployment slots all run in the same App Service Plan, and that's what you pay for. You should keep in mind that having additional deployment slots in an App Service Plan will consume resources like CPU and memory.

You can create new deployment slots from the Deployment Slots menu item in the web app. Make sure that you are running the web app in the Standard or Premium pricing tier because the free plan doesn't come with any Deployment Slots.

In each deployment slot that you create, you can configure the deployment options as we did earlier to deploy code automatically. You can even work on different *source code branches* for different environments and automatically deploy specific branches to specific deployment slots.

Additionally, you can test your final version in a deployment slot and when your happy, you *swap* it with the production slot. This warms-up the application before it swaps, which results in a deployment with no downtime.

Let's go through creating a deployment slot and swapping to it.

1. In the Azure portal, go to the web app that hosts the Node.js application.
2. On the menu bar, click Deployment Slots. The deployment slots blade opens.

3. Click the plus sign (+) to create a new deployment slot.
    a. Type a name for the slot (e.g., staging).
    b. Choose the Node.js web app as the Configuration Source (this copies the application settings to the new slot).
    c. Click OK to create the slot.
4. After the slot is created, it is similar to the original web app.
    a. Set up CD for the slot just as you did in the previous procedure for the web app.
    b. Disconnect the CD connection in the original Node.js web app. This way, when you push new code, it's delivered only into the staging slot
5. Make a change to the Node.js application.
    a. Change some text in the home.client.view.html file (you can find it in the meanjs\modules\core\client\views folder).
    b. Commit it to Git and Push it to GitHub, the same as when you deployed the Node.js app.

The new version is now in the staging slot and not in the original web app (which we call the production slot). You can verify this by navigating to the URL of the Node.js web app and to the URL of the staging slot (which you can find in the overview blade of the slot, just like in the web app overview blade).

Let's put the new version into production.

1. In the Azure portal, go to the Node.js web app.
2. On the menu bar, click Deployment Slots to open deployment slots blade.
3. Click the swap button to open the Swap blade.
    a. Leave all the settings as they are.
    b. Click OK to initiate the swap.

After the swap is complete, the new version of the Node.js web app is in production (test it by navigating to the URL of the Node.js web app). Using deployment slots in this way is very beneficial because you can test the new version before it goes into production and then deploy it to production with no downtime.

Additional diagnostics with diagnostics logs

When the application is running, it is vital to know how it is performing. A great way of monitoring the app is by using diagnostics logs to see live diagnostic logging from the web app. You can even pipe them into the console window. To do that run the following command in the Azure CLI:

```
az webapp log tail --name <app_name> --resource-group myResourceGroup
```

You'll see some logging when you use the application in the web app to generate some traffic.

How to set up sophisticated monitoring and alerts

Another great way to keep track of how your application is doing is by using Application Insights. This is a monitoring tool in Azure that can keep you informed as to how everything in in your application is doing, from how many visitors used your app to how many exceptions occurred and where in the code they occurred. Unlike Diagnostic Logs, Application Insights come at a nominal fee.

You can set up Application Insights in the Azure portal from the web app.

1. Go to the Azure portal and then to the web app that hosts the Node.js application.
2. On the menu bar, click Application Insights.
3. Select Create New Resource.
    a. Type a name and select a location for the Application Insights instance.
    b. Click OK. Now, Application Insights will be deployed and starts to collect data for the application.

You do need to configure your application to begin sending data to Application Insights. For our sample Node.js application, you need to do the following:

1. At the command prompt, `cd` to the meanjs folder of the Node.js application.
2. Run `npm install applicationinsights –save`.
3. Go to the Azure portal and retrieve the Application Insights instrumentation key.

   You'll find this in the Application Insights instance, under Properties.
4. Open the file server.js of the Node.js web app.
5. Add the following three lines of JavaScript code *before* the line:

```
var app = require('./config/lib/app');
const appInsights = require("applicationinsights");
appInsights.setup("<instrumentation_key>");
appInsights.start();
```

6. Push the changes to GitHub and start the automatic deployment. From the command prompt, run the following:

```
git commit -a -m "added Application Insights support"
```

7. Run `git push github`, and the changes will be pushed to GitHub and automatically deployed to the web app.

When the deployment is finished, the application will send data to Application Insights.

By default, Application Insights performs *smart detection*. This is a very clever feature that detects when things are going wrong and alerts you when they do. It can detect things like a sudden increase in failed requests, and when the application is behaving unusually slow. You can also create your own, custom events for all sorts of metrics and conditions in the Alerts menu of Application Insights, as demonstrated in Figure 6-14.

Figure 6-14: Application Insights Alerts

Now, go to the Azure portal and find the Application Insights resource and click it. When you're there, you will see the overview, which shows you basic metrics such as the server response time, the page view load time and the amount of server requests and failed requests. If you see some data there, Application Insights is working.

Note You can collect additional information to send to Application Insights by following this guide.

Scaling the web app

When you have many users, you need the web app to scale to be able to accommodate the traffic. When it's not busy, we need it to scale back to save costs. You can do that by using the automatic scaling feature of App Service. Be aware that you need to run the Web App in the Standard or Premium pricing tier to use this feature.

The web app has a menu item called Scale Out, as shown in Figure 6-15. You can use this to scale out manually or automatically.

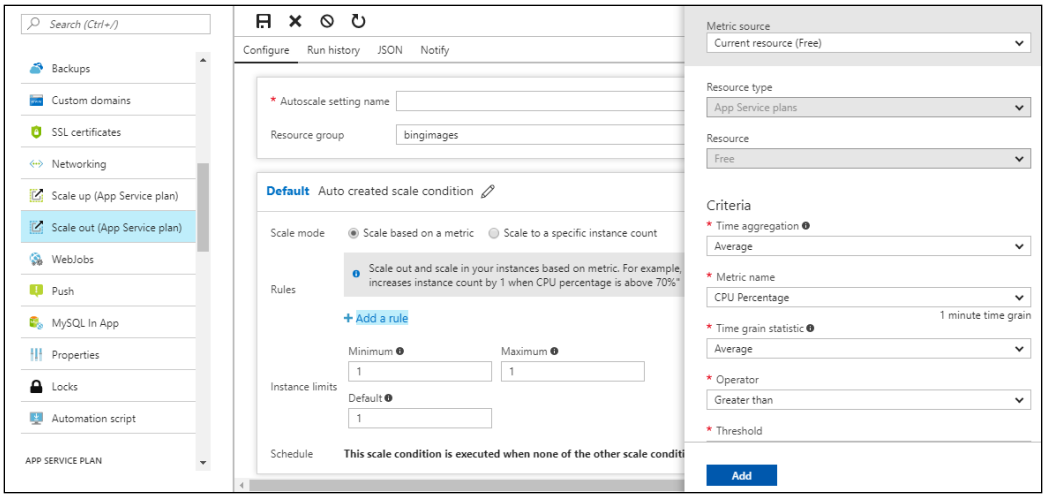

Figure 6-15: Scaling a Web App through the Azure portal

Scaling out means that you add more instances of your application to handle the load. When you scale out or scale in automatically, you can do that based on metrics, such as percentage of use of CPU or memory, on a schedule (every day at 5 PM), or a mix of both. This is very easy to set up and monitor.

Ready for production! Add Secure Sockets Layer

Now that the app is ready for production, you should confirm that it is secure. Besides authentication and authorization, serving the web application over HTTPS is one of the most important things you can do. This is because without HTTPS, intruders could see the traffic between your resources and use it to do bad things with (like signing in to your application). Additionally, HTTPS is a requirement for cutting-edge features like services workers.

Serving traffic to your web app over Secure Sockets Layer (SSL) is possible by importing an SSL certificate into the web app and binding it to one of your (custom) domain names. You can either import your own SSL certificate that you or your company bought, or buy a new one through Azure App Service Certificates, which makes it easy to buy and validate the certificate. After importing the certificate, you couple it to one of the domain name bindings of your web app. You can do all of this from the SSL Certificates menu in the web app.

Walk-through #3: create a backend for your mobile app with Azure

Almost everybody has a mobile device these days. Traffic for most applications is mostly from mobile devices and that trend continues to grow. Building a mobile app for your company is one thing, but providing a robust and reliable experience is something else. Azure can help with that by providing an advanced backend with Azure App Service Mobile App.

Azure Mobile App applications consist of two parts:

- A backend that provides APIs to communicate with a database and do things like authentication, offline synchronization, and push notifications
- SDKs for all sorts of platforms to easily connect to the backend

You can build the backend for the Mobile App application in Node.js and C#. The client SDK is available for basically every mobile platform that is in use. For cross-platform, native, and hybrid mobile apps. It is available for Android, Cordova, iOS, Windows, Xamarin.Android, Xamarin.Forms, and Xamarin.IOS.

In this walk-through, we create a new Mobile App backend with C# that we use with an Android app, which we build with Xamarin.Android.

To follow along, you need to install the Mobile Development with .NET workload using the Visual Studio 2017 installer. This installs the Xamarin for Android tools and the necessary SDKs.

You can do the same when you are working on a Mac in Visual Studio for Mac.

Create a mobile app through the Azure portal

It's easy to get started. Let's create the mobile backend and a starter app for the client through the Azure portal.

1. In the Azure portal, click Create A New Service, and then, in the Search box, type **Mobile App Quickstart**.
2. Select the Mobile App Quickstart result, and then click Create.

   The Mobile App Create Wizard opens.

   a. Type a name for your mobile app.

   b. Create a new resource group.

   c. Create a new App Service Plan for the Mobile App to run on. You can use the Free Pricing Tier for this walk-through.

   d. Leave the Application Insights setting off.

   e. Click Create. That's it! The mobile app will now be deployed.

After the Mobile App is deployed, you can begin using it. In the Azure portal, in the mobile app, click the Quickstart menu item, and then select Xamarin.Android. The Quickstart Wizard opens.

Step 1: connect to a database

You need to connect a database to store data. Click the information icon to get started. On the Data Connections tab that just opened, click Create A New Service. In the Add Data Connection Wizard that opens, do the following:

1. For Type, select SQL Database.
2. Create a new Database with a new Target Server.
3. Leave the Connection String as is.
4. Click OK.

The data connection is created.

Step 2: create a table API

To communicate with the database, you need an API. You can create an API backend hosted in your mobile app by following these steps:

1. In the Step 2 area, select C#, and then click Download. This downloads the backend project that you are going to deploy to the mobile app.
2. Unzip the file that you just downloaded, and then open it with Visual Studio 2017 or Visual Studio for Mac.
3. In Visual Studio, right-click the project file, and then select Publish.
4. Select App Service, and then select Existing, and click Publish.
5. Find your mobile app, and then click OK to begin publishing. That's it! The mobile backend is up and running.

Leave Step 3 to **Create a new App** and click **download.**

Unzip the file that you just downloaded and open it in Visual Studio 2017 or Visual Studio for Mac.

Run the application by pressing F5. This starts the Android Emulator and deploys the client app. The client app is a todo application. You can add an item in the textbox and it will be written to the backend your mobile app.

Adding authentication to the app

Now that you have a working mobile app with a backend, you should make sure that it is secure. Let's add authentication via Azure Active Directory. To complete this process, you first need to register the mobile app for authentication with Azure Active Directory. (This tutorial shows you how to do that.)

Secure the application

In the backend APIs, you need to define which ones that you want to secure.

1. Go to your mobile app backend project in Visual Studio 2017 or Visual Studio for Mac.
2. Navigate to Controllers > TodoItemController.cs.
3. Add the `[Authorize]` attribute to the `TodoItemController` class.

   This restrict any unauthenticated use of this controller.
4. Right-click the project file, and then publish it to your mobile app.

Add authentication to the mobile app

The backend is secured. Now, you need to make the mobile app aware of this so that is able to sign in. We need to require users to authenticate before they can use the app.

1. In Visual Studio, go to the Xamarin Mobile App project.
2. Add the following code to the `TodoActivity` class:

```
// Define a authenticated user.
 private MobileServiceUser user;
 private async Task<bool> Authenticate()
 {
         var success = false;
         try
         {
             // Sign in with AAD login using a server-managed flow.
             user = await client.LoginAsync(this,
                 MobileServiceAuthenticationProvider. WindowsAzureActiveDirectory);
             CreateAndShowDialog(string.Format("you are now logged in - {0}",
                 user.UserId), "Logged in!");

             success = true;
         }
         catch (Exception ex)
         {
             CreateAndShowDialog(ex, "Authentication failed");
         }
         return success;
 }

 [Java.Interop.Export()]
 public async void LoginUser(View view)
 {
     // Load data only after authentication succeeds.
     if (await Authenticate())
     {
         //Hide the button after authentication succeeds.
         FindViewById<Button>(Resource.Id.buttonLoginUser).Visibility = ViewStates.Gone;

         // Load the data.
         OnRefreshItemsSelected();
     }
 }
```

   This creates a new method to authenticate a user and a method handler for a new Sign In button.

3. In the `OnCreate` method, delete or comment-out the following line of code:

```
OnRefreshItemsSelected();
```

4. In the Activity_To_Do.axml file, add the following `LoginUser` button definition before the existing `AddItem` button:

```
<Button
android:id="@+id/buttonLoginUser"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:onClick="LoginUser"
android:text="@string/login_button_text" />
```

5. Add the following element to the Strings.xml resources file:

```
<string name="login_button_text">Sign in</string>
```

6. Run the application.

A Sign In button now appears next to the Add button. You won't be able to add any items without signing in because all the actions on the `TodoItemController` are now secure. When you click Sign In, you are directed to the Azure Active Directory authentication screen. Here, use the account that you use to sign in to the Azure portal. This one is added to the Azure Active Directory by default. Sign in and you'll see that you are successfully logged in, as depicted in Figure 6-16, and can begin adding todo items.

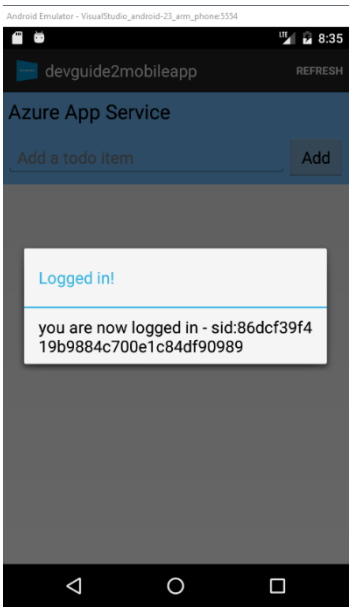

Figure 6-16: Just authenticated in the mobile app in the emulator

Adding Offline Sync to the app

In many mobile scenarios, users won't always have internet access. Even without it, though, users should be able to continue working. To accommodate this, they can use the Offline Sync feature in Azure Mobile App. Lets' turn it on for our app:

1. In Visual Studio 2017 or Visual Studio for Mac, go to the Xamarin Android client app.
2. In the `ToDoActivity` class, uncomment the following line:

```
#define OFFLINE_SYNC_ENABLED
```

3. That's it! Run the app to deploy it to the Android emulator.

Now, the app is working exactly the same as before, but with a few changes. It uses a `IMobileServiceSyncTable` instead of a `IMobileServiceTable` object. It also uses the SQLite NuGet package. This initiates a SQLite database on the android device that the app uses to save data when it is not online. When access to the internet is restored, changes from the local SQLite database are pushed to your mobile app, and potential changes that happened on the backend are synchronized to the app on the Android device. Let's try that out:

1. Have the app running on the Android emulator.
2. Sign in as you did before.
3. Add an item.
4. Disconnect from the internet. In the Android emulator, you can do that in the Extended Controls, by setting the Signal Strength of the Network to None.
5. Add another item and complete an item by checking it. The app continues to work on the local SQLite database.

   You can verify that the items aren't written to the backend database by connecting to the Azure SQL Database that we've created earlier using a tool like SQL Server Management Studio. You can find the `connectionstring` for the Azure SQL Database when you navigate to it in the Azure portal.

6. Connect the Android device to the internet again. In the Android emulator, go to the extended controls, and then set the Signal Strength to Great.
7. In your mobile app, click the Refresh button. The local changes will now be synced to the Azure Mobile App.

   Check the Azure SQL Database to see the changes that are now synchronized to the backend.

Additional features and moving to production

It's easy to set up an app using Mobile App and include advanced features like Authentication and Offline Sync.

Because the mobile app is an Azure App service, it has all the same features that the other App Services have. This means that mobile app can also scale automatically, that you can have the same monitoring and alerting and that you can do additional diagnostics with streaming logs, and even incorporate application debugging.

When you move the app to production, you can provide additional security using App Service Certificates. You can also set up CD, just like we did for the Node.js application. And you can use deployment slots to deploy with no downtime.

Setting up all of this used to be difficult; now they are solved problems. Let Azure to do the heavy lifting for you, so that you can focus on building things that matter!

7

Using the Microsoft Azure Marketplace

The Azure ecosystem not only helps you to more quickly and effectively create dynamic applications, but with the Azure Marketplace, you can find solutions of all sorts to assist your development efforts. And then it can help you to connect your product to users all over the globe.

How can the Azure Marketplace help me as a developer?

The Azure Marketplace provides you with ready-to-run setups of all sorts of products, from Windows Server to Octopus Deploy, to Oracle Databases, to WordPress instances, to Minecraft servers. You just click a solution, configure it, deploy it and you can begin using it. The Marketplace is full of products and solutions from Microsoft and Microsoft Partners. There are also a lot of open-source solutions there.

The advantage of the Marketplace is the ease of use. Solutions are deployed ready-to-run, along with all of the required bits and pieces. Sometimes, this is a Virtual Machine with software installed, a Virtual Network configured and an integrated license installed, or a license that you bring yourself. Sometimes, this is a service that you acquire, like a subscription to the Bing Maps API for Enterprise. You can use the Marketplace to easily test-drive new software, and for using enterprise-grade solutions. No matter what you need, you can probably find a solution for it in the Marketplace.

Azure Solutions

You can learn from companies using Azure successfully, through Azure Solutions. These show you product documentation, case studies and architecture best practices for common Azure use cases like Digital Marketing, SharePoint on Azure, Line-of-Business applications, and Mobile applications.

The most valuable thing of Azure Solutions for developers are the solution architectures. These provide you with reference architectures for common problems and explain why you should use certain products in certain scenarios.

Summary and where to go next

In this guide, we've introduced the power that Azure can bring to your applications. Using Azure, you can do incredible things with your applications—facial and speech recognition, manage your devices on the Internet of Things in the cloud, scale as much as you want and pay for what you use.

You've seen that no matter what programming language you use or what platform you write applications for, Azure can help you, with services for almost every scenario.

We hope that you continue to use this e-book to become better acquainted with the vast range of Azure services and determine which ones best fit your scenario.

The days of having to write complicated "plumbing" yourself are over; you can now take advantage of a wealth of prebuilt solutions. Free yourself up to work on the things that matter, and let Azure take care of the solved problems.

Keep learning with an Azure free account

Sign up for an Azure free account and receive:

- A $200 credit to use on any Azure product for 30 days
- Free access to our most popular products for 12 months, including compute, storage, networking, and database
- 25+ products that are always-free

About the authors

Michael and Barry are passionate about Microsoft Azure and would encourage you to reach out to them on Twitter for questions regarding this book.

Michael Crump works at Microsoft on the Azure platform and is a coder, blogger, and international speaker on various cloud development topics. He's passionate about helping developers understand the benefits of the cloud in a no-nonsense way.

You can reach him on Twitter @mbcrump or by following his blog at https://www.michaelcrump.net.

Barry Luijbregts is an independent software architect and developer with a passion for the cloud. He is also a Pluralsight author. He has worked for many companies over the past 10 years and is keen to share his knowledge with the community. He has a broad and deep knowledge of the Microsoft stack with a special interest in web technology and the cloud. Barry is co-operator of a user group focused on technology and soft-skills called .NET Zuid (South) and currently teaches people about the benefits of the cloud. He lives in the Netherlands with his beautiful wife and baby girl and loves to play with their two Siberian huskies.

You can reach Barry on Twitter @AzureBarry and through his website at https://www.azurebarry.com/.