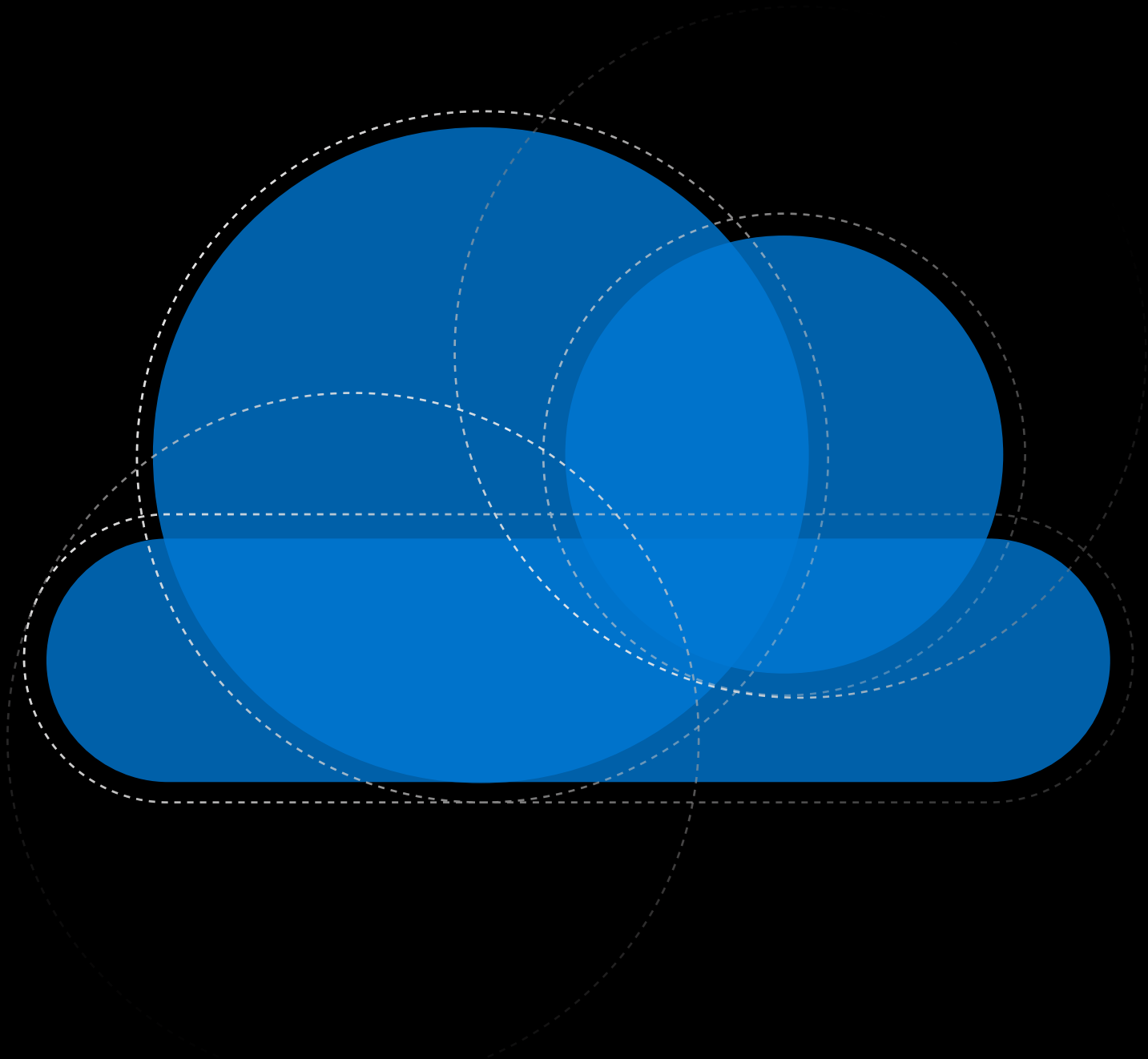
E-book Series

Microsoft Azure

The Developer's Guide to Azure

Published May 2019



03 /

Introduction

We're here to help

05 /

Chapter 1: Getting started with Azure

What can Azure do for you?
Where to host your application
Azure App Service Features
Azure Functions
Azure Logic Apps
Azure Batch
Containers
What to use, and when?
Making your application more performant
Azure Front Door
Azure Content Delivery Network
Azure Redis Cache

22 /

Chapter 2: Connecting your app with data

What can Azure do for your data?
Where to store your data
Azure Cosmos DB
Azure SQL Database
Azure databases for MySQL, PostgreSQL, and MariaDB
Azure Storage
Azure data analytics solutions
Walkthrough: Publish an on-premises website to Azure with a SQL database

40 /

Chapter 3: Securing your application

How can Azure help secure your app?
Encryption
Azure Security Center
Logging and monitoring

51 /

Chapter 4: Adding intelligence to your application

How can Azure integrate AI into your app?
Azure Search
Cognitive Services
Azure Bot Service
Azure Machine Learning Studio
Developer tooling for AI
AI and mixed reality
Using events and messages in your application

72 /

Chapter 5: Connect your business with IoT

How can Azure connect, secure, manage, monitor, and control your devices in the cloud?
Azure IoT Hub
Azure IoT Central
Azure IoT solution accelerators
Azure IoT Edge
Azure Digital Twins
Azure Sphere
Learn more about Azure IoT
What to use, and when?

82 /

Chapter 6: Where and how to deploy your Azure services

How can Azure deploy your services?
Infrastructure as Code
Azure Blueprints
Containers in Azure
Azure Stack
Where to deploy, and when?

89 /

Chapter 7: Share your code, track work, and ship software

How can Azure help you plan smarter, collaborate better, and ship your apps faster?
Azure Boards
Azure Repos
Azure Pipelines
Azure Test Plans
Azure Artifacts

98 /

Chapter 8: Azure in Action

Walk-through: Azure portal
Walk-through: Developing a web app and database
Walk-through: Extending apps
Walk-through: Ready for production

119 /

Chapter 9: Summary and resources

Keep Learning with Azure
About the authors

The Developer's Guide to Azure

This guide is designed for developers and architects who are starting their journey into Microsoft Azure. In this guide, we'll take you through the ins and outs of Microsoft Azure. You'll learn how to get started and which services you can use for the scenarios you might have.

From creating websites, databases, and desktop and mobile applications to integrating the latest technologies into your app, Azure does the heavy lifting for you. Azure services are designed to work together so you can build complete solutions that last the lifetime of your app.

We're here to help

We can assist you in a variety of ways to suit your needs.

With our support plans, you'll get access to Azure technical support teams, guidance for cloud design, and assistance with migration planning. You can even acquire a support plan that guarantees a response from the technical support teams within 15 minutes.

You can also get help through other channels, such as:

Documentation and guides that give you an overview of everything in Azure and provide deep insights through the documentation of each feature.

Service License Agreements (SLAs), which can inform you about our uptime guarantees and downtime credit policies.

@AzureSupport on Twitter, which is operated by skillful Azure engineers who respond quickly to issues that you tweet to them.

Stack Overflow, which provides answers to Azure questions and includes many active posts by members of the Azure engineering teams.

Azure Community Support, which provides a place for discussion with the Azure community and contains answers to community questions.

Azure Advisor, which automatically makes personalized recommendations for your Azure resources, including what you need to do to be more secure, have higher availability, increase performance, and reduce costs.

Azure Service Health, which gives you a personalized view of the health of your Azure services.

01 /

Getting started with Azure

You've made the decision to build applications on Azure, and now you want to get started. You don't need to do much—just sign up for an Azure free account. This includes credits to explore paid Azure services and over 25 services you can use for free forever.

Simply choose which tools, applications, and frameworks you want to use, and then start running your apps on Azure.

What can Azure do for you?

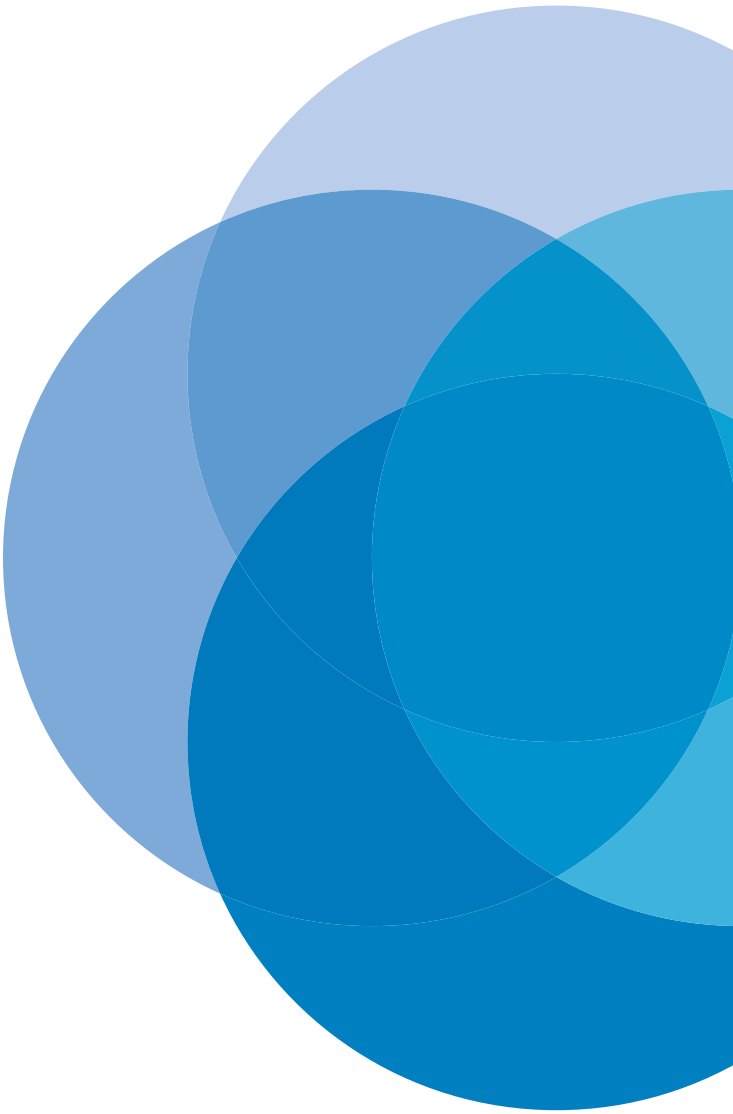
Whether you're a professional developer or write code for fun, developing with Azure puts the latest cloud technology and best-in-class developer tools at your fingertips and makes it easy to build for the cloud in your preferred language.
With Azure, you can get your work done faster, take your skills to the next level, and imagine and build tomorrow's applications.

Multiply your impact with:

- A cloud platform
- Developer tools
- Management services

Integrated tightly together, these form a true ecosystem that enables you to create amazing applications and seamless digital experiences that run on any device.

Take advantage of the incredible and always growing capabilities of Azure. Let's dive in and see what you can do.



Where to host your application

Azure offers services designed to provide what you need to deliver and scale every application. When you use Azure services to run your application, you get scalability, high availability, a fully managed platform, and database services. Azure also offers the following options for running your application.

Azure App Service

You can host your applications in a fully managed application platform loved by enterprise developers: Azure App Service. Azure App Service is a collection of hosting and orchestrating services that share features and capabilities. All services in App Service have the capability, for example, to secure an application using Azure Active Directory and can use custom domains.

Azure App Service comprises the following:

Web Apps: As one of the most widely used Azure services, Web Apps can host your web applications or APIs. A web app is basically an abstraction of a web server, like Internet Information Services (IIS) or Tomcat, used to host HTTP-driven applications. Web Apps can host applications written in .NET, Node.js, Python, Java, or GO, and you can use available extensions to run even more languages.

```
// Try it out: Create an ASP.NET
   Core web app in Azure
```

Mobile Apps: Provide a back end for your mobile applications with Mobile Apps. When you host an API in Mobile Apps, your mobile applications connect with it through the cross-platform client SDK. This is available for iOS, Android, and Windows. Mobile Apps provides features like offline sync and push notifications to help you create a modern, performant, and secure mobile experience.

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 brings 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 virtual machines (VMs). One or more VMs run your Azure App Service, but since Azure takes care of them, it's not necessary for you to know which ones. 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. It's even possible to have 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

After deploying a new version of your application to a deployment slot, you can 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 example, if you shunt 10 percent of your users to the new version of your app in the deployment slot, you can see whether the new features are functioning as expected and whether users are using them.

When you're 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. 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. You can also swap back and revert the deployment of the new version.

You use deployment slots within environments, such as development, test, or production. You don't use deployment slots as environments, because they all reside in the same App Service plan.

Those should 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 Azure DevOps to perform swap operations during a release.

Continuous Deployment

To publish your application to App Service, you can use services such as Jenkins, Octopus Deploy, and more. You also can use the Continuous Deployment (CD) feature from Azure DevOps in App Service. This makes it possible for you to create a build-test-release pipeline right in App Service.

The process does the following:

1. Retrieves the latest source code from the repository that you indicate
2. Builds the code according to 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 like data stores to your App Services. These resources don't need to be located in Azure; they can be anywhere, such as on-premises or in your own datacenter. Depending on your requirements, you can connect to services on-premises through many mechanisms, such as Azure Hybrid Connections, Azure Virtual Networks, and Azure ExpressRoute.

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 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. Just 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. You can configure this to be used by your custom domain bindings.

```
// Try it out: Purchase and
   configure an SSL certificate
   in this walk-through
```

App Service Environment

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

To isolate these support services from the internet, you can use Azure Virtual Network.

This service wraps your support services and connects them to your app in Web Apps so 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 want even more control. Maybe you want your app to be wrapped in a Virtual Network in order to control access to it.

Perhaps you want it to be called by another app in Web Apps and be a part of your back end.

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.

Snapshot Debugger for .NET

Debugging apps can be difficult, especially if the app is running on production. With Snapshot Debugger you can take a snapshot of your in-production apps when code that you're interested in executes.

The debugger lets you see exactly what went wrong without impacting the traffic of your production application. The Snapshot Debugger can help you dramatically reduce the time it takes to resolve issues that occur in production environments. Additionally, you can use Visual Studio to set snap points to debug step by step.

Automatic OS and .NET Framework patching

Because you're using a fully managed platform, you don't manage your own infrastructure at all and benefit from automatic operating system (OS) and framework patching.

Virtual machines

Hosting your application in a VM in Azure Virtual Machines provides you with a lot of control over how you host your application. However, you're responsible for maintaining the environment, including patching the OS and keeping antivirus programs up to date.

You can use a VM to test the latest preview version of Visual Studio without getting your development machine "dirty."

Azure Functions

With Azure Functions, you can write 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's 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 Storage, Azure Cosmos DB, Azure Service Bus, and even third-party services like Twilio and SendGrid. Using Functions, you can build small pieces of functionality quickly and host them in an elastic environment that automatically manages scaling.

With Azure Functions, it's possible to pay only for functions that run, rather than having to keep compute instances running all month. This is also called serverless because it only requires you to create your application—you don't have to deal with any servers or even scaling of servers.

You can write Azure Functions in .NET, JavaScript, Java, and a growing list of languages.

An application that uses Functions activates a function every time a new image file is uploaded to Azure Blob storage. The function then resizes the image and writes it to another Blob storage account.

Data from the Blob that triggered the function is passed into the function as the myBlob parameter, which includes the Blob URL. Use the outputBlob output binding parameter to specify to which Blob 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 by 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. Weaving together API calls to connectors, you can create a (possibly complex) workflow that can involve resources both in the cloud and on-premises.

Logic Apps has many available connectors to APIs that can connect to Azure SQL Database, Salesforce, 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 and scaled automatically, and you pay for them only when they're running.

The following is an example of a workflow in Logic Apps:

1. The Logic App is activated when an email containing a shipping order 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. Using Twilio, the Logic App sends a text message to the customer's phone indicating that the order was received and the item has been shipped.

```
// Try it out: Get started with
   Azure Logic Apps
```

Azure Batch

If you need to run large-scale batch or high-performance computing (HPC) applications on VMs, you can use Azure Batch. Batch creates and manages a collection of thousands of VMs, installs the applications you want to run, and schedules jobs on the VMs. You don't need to deploy and manage individual VMs or server clusters; Batch schedules, manages, and auto-scales your jobs so you use only the VMs you need.

Batch is a free service, so you only pay for the underlying resources consumed, like VMs, storage, and networking.

Batch is well suited to run parallel workloads at scale, such as financial risk models, media transcoding, VFX and 3D image rendering, engineering simulations, and many other compute-intensive applications. Use Batch to scale out an application or script that you already run on workstations or an on-premises cluster, or develop SaaS solutions that use Batch as a compute platform.

```
// Try it out: Get started on
   Azure Batch with these
   step-by-step tutorials
```

Containers

While much more lightweight, containers are similar to 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.

Scale and orchestrate containers with Azure Kubernetes Service

Azure Kubernetes Service (AKS) makes it simple to create, configure, and manage a cluster of VMs that are preconfigured to run containers. This means you can use your existing skills to manage and deploy applications that run in containers on Azure.

AKS reduces the complexity and operational overhead of managing a Kubernetes cluster by offloading much of that responsibility to Azure. As a hosted Kubernetes service, Azure handles critical tasks like health monitoring and maintenance. In addition, you pay only for the agent nodes within your clusters, not for the masters. As a managed Kubernetes service, AKS provides automated Kubernetes version upgrades and patching, easy cluster scaling, a self-healing hosted control plane (masters), and cost savings, since you only pay for running agent pool nodes.

With Azure handling the management of the nodes in your AKS cluster, there are many tasks that you don't have to perform manually, such as cluster upgrades. Because Azure handles these critical maintenance tasks for you, AKS does not provide direct access (such as with SSH) to the cluster.

```
// More info: Learn how to use
   Azure Kubernetes Service
```

Host containers with Azure Container Instances

You can host your container using Azure Container Instances (ACI). ACI provides fast, isolated compute to meet traffic that comes in spikes, without the need to manage servers. For example, Azure Container Service (ACS) can use the Virtual Kubelet to provision pods inside ACI that start in seconds. This enables ACS to run with just enough capacity for an average workload. As you run out of capacity in your ACS cluster, you can scale out additional pods in ACI without any additional servers to manage. The ACI service is billed per second, per virtual CPU, per gigabyte, or by memory.

```
// More info: Learn more about
   Azure Container Instances
```

Host containers in Azure App Service 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. You can run Docker containers (Linux) and Windows containers on Web App for Containers.

Azure Container Registry

Once you've created a container image to run your application in, you can store that container in Azure Container Registry (ACR). This is a highly available and secure storage service, specifically built to store container images. This is great for storing your private Docker images.

You can also use ACR for your existing container development and deployment pipelines. Use ACR Build to build container images in Azure. You can either build on demand or fully automate builds with source code commit and base image update build triggers.

Azure Service Fabric

Another way to run applications in Azure is with Azure Service Fabric. This is actually the service that runs many of the Azure services inside Microsoft, like Azure SQL Database and Azure App Service. Run your applications in Azure Service Fabric to achieve high availability, run at massive scale, and perform rolling upgrades.

You can use Azure Service Fabric to run .NET microservice-based applications—solutions that consist of many small services that talk to each other and are employed by user interfaces and other components. Service Fabric is ideal for solutions like these because it orchestrates application components together and runs them in a highly available and performant manner.

Azure Service Fabric is unique in that you can run it anywhere. Install Service Fabric on your local development computer, on-premises, or in any cloud—including Azure. You can also use Azure Service Fabric Mesh to run containers on a Service Fabric cluster that Microsoft manages for you as a service. This opens up a lot of possibilities.

It's easy to deploy applications to Azure Service Fabric and manage them with your favorite tools, like Visual Studio and Azure DevOps Services. In addition, Service Fabric recently became open source.

What to use, and when?

Some of the services that run your application in Azure can work together in a solution, while others are more suited to different purposes.

While this can make it difficult to pick the right services, Table 1-1 will help identify which services in Azure are right for your situation.

Table 1-1

	Web Apps*	Web App for Containers*	Mobile Apps*	Functions*	Logic Apps*	Virtual Machines*	Kubernetes Service*	Service Fabric*	Container Instances*	Batch*
Monolithic and N-Tier applications		●				●**			●	●
Mobile app back end			●			●**				
Microservice architecture applications				●			●	●		
Business process orchestration and work flows				●	●					
Compute intensive jobs										●
Run your app anywhere (including on-premises)		●					●	●	●	

* Services with an asterisk have a free tier that you can use to get started at no cost.

** For lifting and shifting existing applications to Azure.

Making your application more performant

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

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, the time it takes for a signal or a request to travel to a user. The farther away users are from your application, the more latency they experience.

Azure Traffic Manager scales across regions, helping to reduce latency and provide users a performant experience, regardless of where they are. Traffic Manager is an intelligent routing mechanism that you put in front of your Web Apps applications. Web Apps acts as endpoints, which Azure Traffic Manager monitors for health and performance.

When users access your application, Traffic Manager routes them to the Web Apps application that is most performant in their proximity.

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

Azure Front Door

Your users might be spread out over the world and at times might be traveling. This can make it difficult to make sure they have a performant experience and that your application is available and secure, regardless of location.

Azure Front Door can help. This service can route traffic from users to the most performant application endpoint for them to improve performance. Azure Front Door can route to endpoints that are available while avoiding endpoints that are down.

Azure Traffic Manager does this as well, but in a different manner than Azure Front Door. Azure Front Door works at OSI layer 7 or the HTTP/HTTPS layer, while Azure Traffic Manager works with DNS. In other words, Azure Front Door works on the application level and Azure Traffic Manager works on the network level. This is a fundamental difference that determines the capabilities of the services.

Because of this difference, Azure Front Door does a lot more than route users to available and performant endpoints.

Azure Front Door allows you to author custom web application firewall (WAF) rules for access control to protect your HTTP/HTTPS workload from exploitation based on client IP addresses, country code, and HTTP parameters.

Additionally, Front Door enables you to create rate limiting rules to battle malicious bot traffic. These are just some of the unique capabilities of Azure Front Door.

Other capabilities of Front Door include:

- **URL-based routing**
  This allows you to route requests for different URLs to different back end pools (applications that receive traffic, like Web Apps). For instance, http://www.contoso.com/users/* goes to one pool, and http://www.contoso.com/products/* goes to another.
- **URL rewrite**
  This enables you to customize the URL that you pass on to the back end pool.
- **SSL termination**
  With this, you can secure your traffic end to end, from the browser to the application in the back end pool.
- **Session affinity**
  When you want users to be sent to the same endpoint every time, session affinity is useful. This is important in cases where session state is saved locally on the back end for a user session.

If you need help choosing between Azure Front Door and Traffic Manager, consider this guidance:

	Azure Traffic Manager	Azure Front Door
You only need routing (performance- or geography-based) and high availability	●	
You need SSL termination (also called SSL offloading)		●
You need application layer features like URL rewriting and WAF		●

Azure Content Delivery Network

One of the Azure services that can help you 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 change the reference to the static files to a different URL.

For example, the reference previously might have been **~/images/image.png**, and it would now be **https://example.azureedge.com/image.png.**

Not only is this easy to do, it also improves the performance of your application in the following ways:

- Offloads serving content from your application. Since it is now served by Content Delivery Network, it frees up processing cycles for your application.
- 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 as well as in mobile and desktop applications. One way to use Content Delivery Network is to serve videos for a mobile app. Since videos can be large, you don't want to store them on the mobile device—and neither do your users. Using Content Delivery Network, the videos are served from the PoP. Since it is close to the user, this also improves performance.

```
// 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 the cache.
2. If the data is not there, get it from the database and store the data in the cache.
3. The next time the app is looks for that 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 industry-leading SLAs. It is highly performant and has advanced options like clustering and geo-replication.

```
// Try it out: Get started with
   Azure Redis Cache
```

Further reading

If you want to learn more about using Azure Kubernetes Service, Azure Container Instances, and other Azure services to create distributed applications, download and read the following free e-books:

```
// Containerize Your Apps
   with Docker and Kubernetes
```

```
// Designing Distributed Systems
```

02 /

Connecting your app with data

What can Azure do for your data?

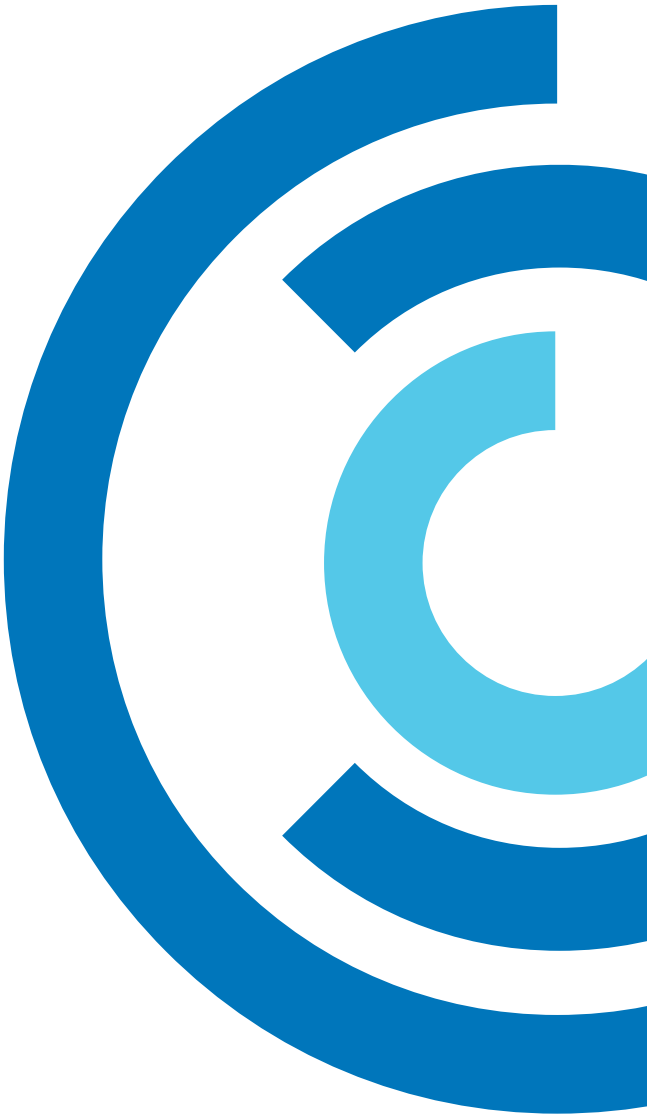
Wherever your data is, Azure will help you unlock its potential. Support rapid growth and save more time for innovation with a portfolio of secure, enterprise-grade database services that support open-source database engines.

Azure database services are fully managed, freeing up valuable time so you can focus on new ways to delight your users and unlock opportunities rather than spending that time managing your database. Enterprise-grade performance with built-in high availability means you can scale quickly and reach global distribution without worrying about costly downtime.

Developers can take advantage of industry-leading innovations, such as built-in security with automatic monitoring and threat detection, automatic tuning for improved performance, and turnkey global distribution. On top of all of this, your investment is protected by financially backed SLAs.

Whatever you build, we'll help you get it to market quickly, distribute it widely, and manage it easily and confidently.

Let's dive in.



Where to store your data

Azure provides many types of data stores that can help you maintain and retrieve data in any scenario. Table 2-1 presents the storage options available in Azure.

All services have a free tier that you can use to get started.

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

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

Table 2-1

	SQL Database*	MySQL*	PostgreSQL*	MariaDB*	Azure Cosmos DB*	Blob*	Table*	Queue*	File*	Disk*	Data Lake Store*	SQL Data Warehouse*
Relational data	●	●	●	●							●	●
Unstructured data					●	●					●	
Semi-structured data					●		●					
Queue messages								●				
Files on disk									●			
High-performance files on disk										●		
Store large data					●	●			●	●	●	●
Store small data	●	●	●	●	●	●	●	●	●	●		
Geographic data replication	●	●	●	●	●	●	●	●	●	●		
Tunable data consistency					●							

* Services with an asterisk have a free tier that you can use to get started at no cost.

Azure Cosmos DB

Azure Cosmos DB is a new kind of database made for the cloud. Its key features include:

- A 99.99 percent SLA (99.999% for read operations) that includes low latencies (less than 10 ms on reads and less than 15 ms on writes)
- Geo-replication, which replicates data to other geographical regions in real time.
- Tunable data consistency levels so you can enable a truly globally distributed data system. You can choose from a spectrum of data consistency models, including strong consistency, session consistency, and eventual consistency.
- Traffic Manager, which sends users to the service endpoint to which they are closest.
- Limitless global scale, so you pay only for the throughput and storage that you need.
- Automatic indexing of data, which removes the need to maintain or tune the database.

In addition to all these features, Azure Cosmos DB offers different APIs with which you can store and retrieve data, including SQL, JavaScript, Gremlin, MongoDB, Azure Table Storage, and Apache Cassandra. Different APIs handle data in different ways. You can use documents as data as well as unstructured tables, graphs, and blobs. You use the API that fits your needs, and Azure Cosmos DB takes care of the rest.

You benefit from cloud-grade performance, scalability, and reliability while using the programming model you're already accustomed to.

```
// Try it out: Get started with
   Azure Cosmos DB
```

Azure SQL Database

If you want to use tables with columns and rows to store data, Azure SQL Database is a great choice. A relational database system similar to on-premises Microsoft SQL Server, SQL Database runs in the cloud—so it's fully managed, performant, scalable, automatically backed up, and includes many advanced features.

With SQL Database, you can do almost everything that you can do with on-premises SQL Server. In fact, new SQL Server features are incorporated first in Azure SQL Database and later in on-premises SQL Server.

You can use SQL Database with your favorite tools, including SQL Server Management Studio and the Entity Framework. Databases in SQL Database are extremely reliable and robust and offer an SLA that guarantees 99.99 percent uptime.

Here are some of the more advanced features in SQL Database:

- Geo-replication, which replicates data to other geographical regions in real time
- Dynamic data masking, which masks sensitive data for certain users at runtime
- Auditing, which provides a complete audit trail of all the actions that happen to the data
- Automatic database tuning, which monitors the performance of your database and tunes it automatically

SQL Database offers several service tiers that are geared toward specific scenarios.

- **General purpose/standard:** This tier offers budget-oriented, balanced, and scalable compute and storage options. Fully managed, with performance comparable to Azure SQL VMs, this tier is the best option for most business workloads.
- **Business Critical/Premium:** This tier offers the highest resilience to failures using several isolated replicas. With consistently high IO, it includes a built-in availability group for high availability. This is the best option for critical Online Transactional Processing (OLTP) (normal CRUD operations) business applications with consistently high IO requirements.
- **Hyperscale:** This tier offers very large database (VLDB) support without the headaches. With a built-for-the-cloud architecture of highly scalable storage and a multilayer cache optimized for very large and demanding workloads, it provides low latency and high throughput regardless of the size of data operations. This is the best tier for very large and demanding workloads with highly scalable storage and read-scale requirements.

Azure databases for MySQL, PostgreSQL, and MariaDB

Azure provides MySQL, PostgreSQL, and MariaDB databases as managed databases, which means that you just spin them up and don't have to worry about any of the underlying infrastructure. Just like Azure SQL Database and Azure Cosmos DB, these databases are universally available, scalable, highly secure, and fully managed.

Each of these databases is suited for slightly different use cases, but in general their functionality overlaps a lot. You would use Azure databases for MySQL, PostgreSQL, and MariaDB when you've already been using one of their on-premises versions and want the advantage of having it run fully managed in the cloud.

Azure Storage

Azure Storage is one of the oldest, most reliable, and most performant services in Azure. Azure Storage offers five types of storage that all benefit from the following shared features:

- Geo-redundancy, which replicates data to different datacenters so you can recover it if a disaster causes an individual datacenter to fail
- Encryption of data at runtime
- Custom domains

The five Azure Storage types are Blob, Table, Queue, File, and Disk (Figure 2-1).

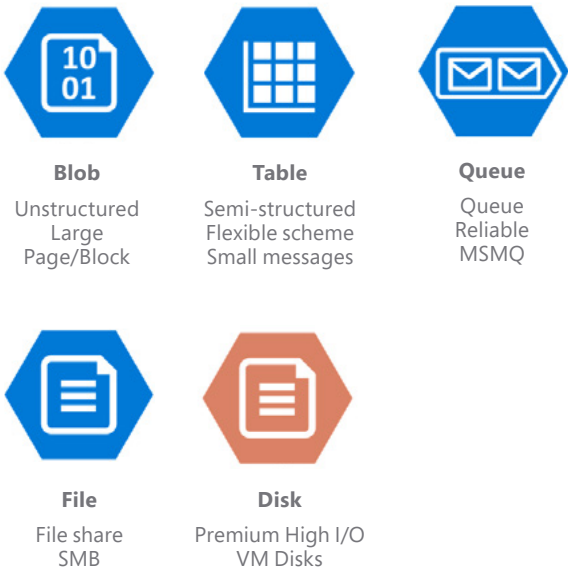


Figure 2-1

Host static websites on Azure Storage

Another exciting feature of Azure Storage is static website hosting. This static websites feature only uses Blob storage as its datastore, and you can use it to host a static website on Azure Storage. All you have to do for your website to run is upload the files of your static website to Blob storage and indicate which file is the default document (like index.html) and which one is the error document (like 404.html). Your website will run quickly for very little cost—in fact, you only pay for the storage you use, and the static website feature doesn't cost anything extra. Additionally, when you use geo-redundancy (which is enabled by default), your website will be up and running even if your primary datacenter fails.

Blob storage

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

There are three types of blobs: Page, Append, and Block Blobs. Page Blobs are optimized for random read and write operations, and 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. Append Blobs are optimized for append operations, such as storing operation logs that can't be updated or deleted.

```
// Try it out: Get started with
   Azure Blob storage
```

Table storage

Azure Table storage is an inexpensive, extremely fast NoSQL key-value store you can use to store data in flexible tables. A table can contain one row describing an order and another row describing customer information. You don't need to define a data schema, making Table storage very flexible.

```
// Try it out: Get started with
   Azure Table storage
```

Queue storage

Azure Queue storage is an unusual type of storage. While it's used to store small messages of data, its main purpose is to serve as a queue. You put messages on the queue and other processes pick them up. This pattern decouples the message sender from the message processor, resulting in performance and reliability benefits. Azure Queue storage is found 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 to share files from. It uses the Server Message Block (SMB) protocol, meaning you can use it with Windows and Linux and access it from either the cloud or on-premises systems. Like the other services in Azure Storage, 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's perfect for use as a drive in a VM that needs high performance to run SQL Server, for instance. 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. The Azure Data Lake Store, on the other hand, is storage for big data applications. You can 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 you can analyze it or transform and move it.

The following are the main characteristics of Azure Data Lake Store:

- 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.
- Data can be stored in any format, without a schema.

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

You can store all of the data that you get from Internet of Things (IoT) devices collecting temperature data, for example, in Data Lake Storage. You can leave the data in the store and then filter through it to create a view of the data per hour or per week. Storing the data in Data Lake Storage is 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

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

Because SQL Data Warehouse runs in Azure, there are many advanced features available to you. One of these features is automatic threat detection, which uses machine learning to understand the patterns of your workload and serve as an alarm system to alert you of a potential breach.

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

```
// Try it out: Create an Azure SQL
   Data Warehouse
```

Azure data analytics solutions

Almost as important as storing data is analyzing it to get insights. Azure provides many services for data analytics scenarios, enabling you to get valuable and actionable insights from your data—no matter how large or small or complex it is.

Azure Data Factory

Moving and transforming data is not a trivial task, but Azure Data Factory can help you to do just that. Within Data Factory, you can create a comprehensive pipeline that performs your complete extraction, transformation, and loading (ETL) process.

Data Factory can reliably move data from on-premises to the cloud, within the cloud, or to on-premises—it doesn't matter where your data sources are. Data Factory also provides many connectors that you can use to easily connect to your data source, like SQL Server, Azure Cosmos DB, Oracle, and many more.

When you move data, you can also filter it before you send it to an end destination, clean it up, or transform it with an activity in the pipeline like the Apache Spark activity. In addition, Azure Data Factory allows you to schedule and monitor pipelines as well as lift and shift your SQL Server Integration Services (SSIS) packages to the cloud.

```
// Try it out: Create a data factory
   using the Azure portal
```

Azure Analysis Services

With Azure Analysis Services, you can create a semantic model of your data that users can access directly with visualization tools like Power BI. Built on the SQL Server Analysis Services tools that run on-premises with SQL Server, the service now runs managed in the cloud. This means that the service is scalable and that data is stored redundantly—and when you aren't using it, you can pause the service to minimize costs.

With Azure Analysis Services, you can provide modeled data directly to users in a very performant way. Users can query millions of records in seconds because the model lives completely in memory and is periodically refreshed.

You can get data into the semantic model from anywhere, including from data sources in the cloud and on-premises. You can use Azure Blob storage,

Azure SQL Database, Azure SQL Data Warehouse, and many other services as data sources for the model. You can also use data sources like on-premises Active Directory, Access databases, and Oracle databases.

```
// Try it out: Create an Azure
   Analysis Services server using
   the Azure portal
```

Azure Data Lake Analytics

Another Azure service for performing data analytics tasks is Azure Data Lake Analytics. With this service, you can analyze, process, and transform potentially massive amounts of data from Azure Storage and Azure Data Lake Store.

Azure Data Lake Analytics allows you to create and submit jobs that query data, analyze it, or transform it. You can write these jobs in U-SQL, which is a SQL-like language, and extend U-SQL with Microsoft R and Python.

You pay for the jobs that you submit and run, and the service scales automatically depending on the power that the jobs need. Azure Data Lake Analytics is typically used for long-running analytics jobs against massive amounts of data.

```
// Try it out: Create your first
   U-SQL script through the
   Azure portal
```

Azure Stream Analytics

You can use the Azure Stream Analytics service to analyze, query, and filter real-time streaming data. For example, when you receive a stream of temperature data from an IoT device, it tells you how warm it is outside. It might provide the same temperature every second for an hour until the temperature changes, but you are only interested in the changes. Azure Stream Analytics can query the data in real time and store only the differential data in an Azure SQL Database.

Stream Analytics can get its data from many services, including Azure Blob storage, Azure Event Hubs, and Azure IoT Hub. You can analyze the data by using a simple SQL-like language or custom code. After querying and filtering the stream of data, Stream Analytics can output the result to many Azure services, including Azure SQL Database, Azure Storage, and Azure Event Hubs.

```
// Try it out: Create a Stream
   Analytics job using the
   Azure portal
```

Azure Time Series Insights

You can use Azure Time Series Insights to get quick insights on large amounts of typically IoT-type data. This service gets data from Azure Event Hubs, IoT Hub, and your own reference inputs, and it retains that data for a specified amount of time.

With Azure Time Series Insights, users can query and analyze data through a visualization tool as soon as it comes in. Time Series Insights not only analyzes data, but also ingests and holds it for a while. This is like Azure Analysis Services, where data lives in-memory in a model for users to query. The key differences are that Time Series Insights is optimized for IoT and time-based data, and it contains its own data visualization tool.

```
// Try it out: Explore a Time Series
   Insights demo environment from
   your browser
```

Azure Databricks

Azure Databricks allows you to run a managed and scalable Databricks cluster in the cloud. Databricks provides a unified analytics platform with a host of tools and capabilities. Within Databricks, you can run optimized versions of Apache Spark to do advanced data analytics.

In addition to Spark-based analytics, Databricks provides interactive notebooks and integrated workflows and workspaces you can use to collaborate with the entire data team, including data scientists, data engineers, and business analysts—all of whom have access to specialized tools for their specific needs.

Databricks is fully integrated with Azure Active Directory, which gives you the ability to implement granular security. With Databricks, you can perform Spark-based data analytics on data that comes from many places, including Azure Storage and Azure Data Lake Store. Databricks also works with data from Azure SQL Data Warehouse, Azure SQL Database, and Azure Cosmos DB. Additionally, you can plug Databricks into Power BI to create and show powerful dashboards.

```
// Try it out: Run a Spark job on
   Azure Databricks using the
   Azure portal
```

Azure HDInsight

Azure HDInsight is a platform within Azure that you can use to run open-source data analytics services. You can also use it to run specialized clusters of your favorite open-source data analytics tools. The advantage of running these tools in Azure HDInsight is that they're managed, which means you don't have to maintain VMs or patch operating systems. Plus, they can scale and easily connect to one another, other Azure services, and on-premises data sources and services.

Most of the specialized open-source data analytics cluster types in Azure HDInsight use Azure Blob storage or Azure Data Lake Store to access or store data, as these services work with the Hadoop File System.

You can run potentially massive specialized clusters of different types, such as an **Apache Hadoop cluster**. This enables you to process and analyze data with Hadoop tools like Hive, Pig, and Oozie.

You can also spin up an **Apache HBase cluster**, which provides a very fast NoSQL database. The data actually lives within Azure Storage or an Azure Data Lake, but HBase provides an abstraction layer on top, which has its own functionality and unique performance.

You can create an **Apache Storm cluster,** which is geared toward analyzing data streams, just like Azure Stream Analytics. In addition, you can have an **Apache Spark cluster**, which provides a framework for processing and analyzing massive amounts of data. HDInsight can also run a cluster for **Microsoft Machine Learning Server** (previously Microsoft R server).

This allows you to run R-based jobs to analyze data. Finally, you can create a cluster that runs **Apache Kafka,** which is a publish-subscribe messaging system used to build applications with queueing mechanisms.

There are more cluster types, as well as tools that you can use within clusters. You can perform almost any data analytics and processing task with a combination of these clusters, and they all run managed in the cloud. Table 2-2 can help you pick the right Azure services for analyzing your data.

```
// Try it out: Extract, transform,
   and load data using Apache Hive
   on Azure HDInsight
```

Table 2-2

	Data Factory*	Analysis Services*	Data Lake Analytics*	Stream Analytics*	Time Series Insights*	Azure Databricks*	Azure HDInsight*
Move data from store to store	●						
Transform data	●	●	●	●	●	●	●
Query and filter streaming data				●		●	●
Provide in-memory semantic model for users		●			●		●
Allow users to query data and create dashboards					●		
Analyze data for later use			●		●		●

* Services with an asterisk have a free tier that you can use to get started at no cost.

Walk-through: Publish an on-premises website to Azure with a SQL database

Before you begin, you will need:

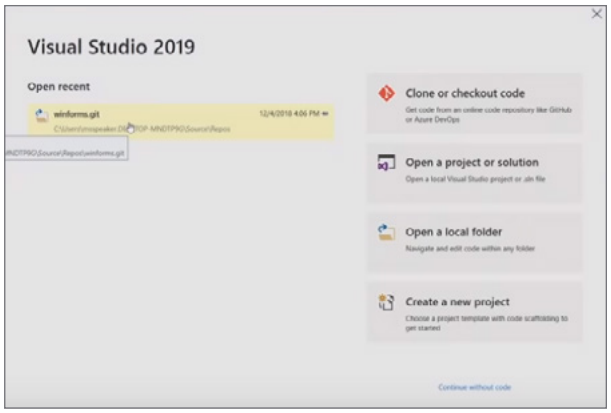
- Microsoft Visual Studio 2017 or later
- An Azure free account in order to follow this demo script
- Tailwind Traders Rewards source code

Walk through: Migrating a .NET app to Azure without code change

When your application has outgrown your local infrastructure, you need to look at other options of meeting demands without burdening your teams.

The Azure cloud offers a variety of platforms and service offerings to host applications. To start, you'll use **Azure App Service** to host the application without any changes to the existing code

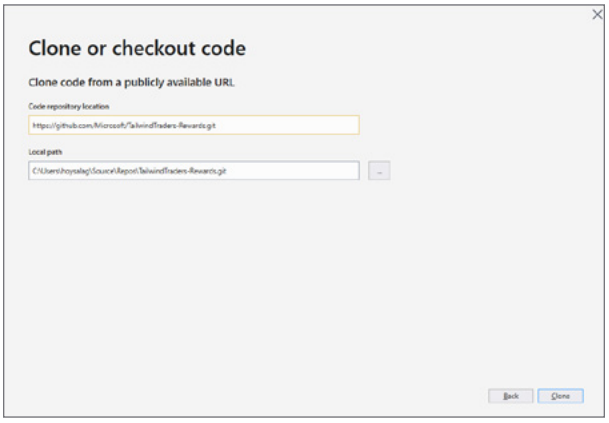
1. Launch **Visual Studio 2019**. You"ll immediately notice the simplified 'open' experience.



Launch Visual Studio

2. Click the **Clone or checkout code** option and enter the **Tailwind Traders Rewards** repository URL (https://github.com/Microsoft/TailwindTraders-Rewards.git) in the Code repository location.

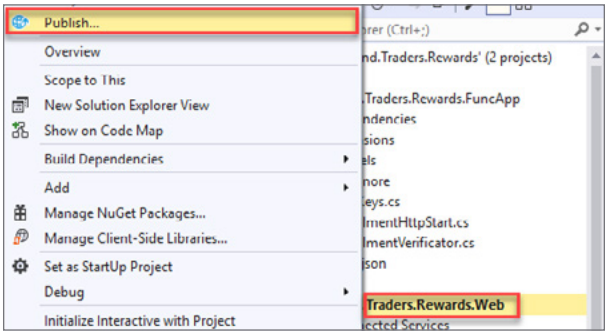
   Click **Clone**. Under **Solutions and folders**, click **Tailwind.Traders.Rewards.sln** to open the solution.



Rewards clone

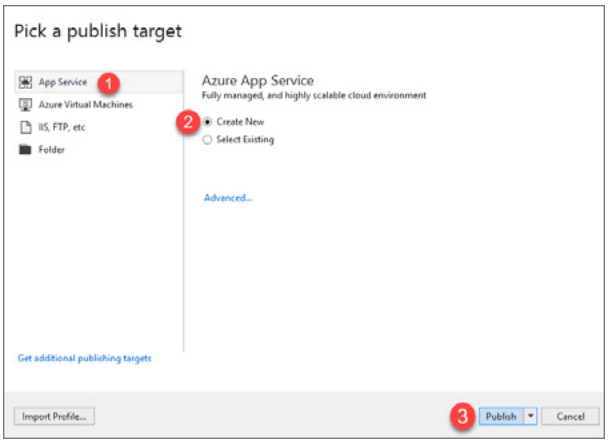
3. Right-click the project **Tailwind.Traders.Rewards.Web** and choose **Publish**. This is the same Publish dialog box that you can use to deploy onto **IIS6** in your local infrastructure.

   Using this Publish dialog, you'll deploy the application to the Azure cloud platform.



Publish app

4. Choose **App Service** as the Publish target. Under the **Azure App Service** window, choose **Create New** and click **Publish**.

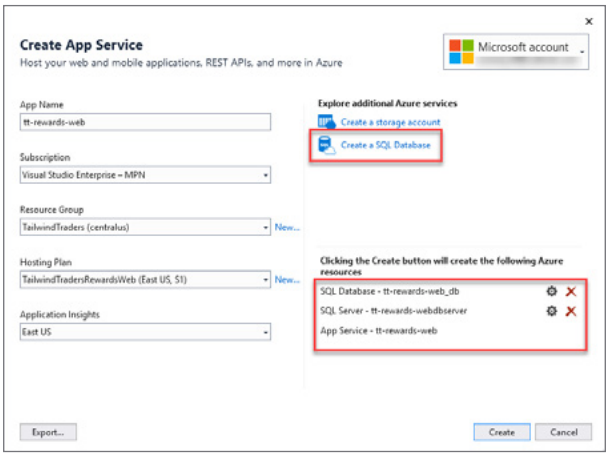


Publish options

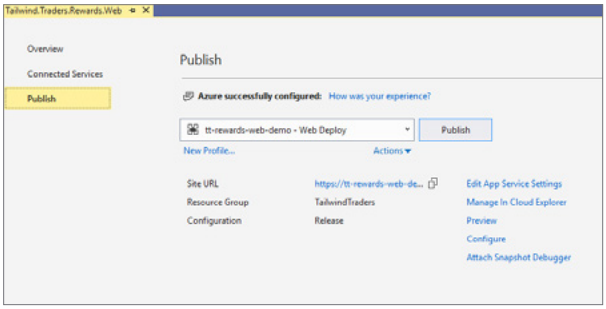
5. In the next window, enter your Azure subscription information, choose the **App Service**, and either choose existing options for **Resource Group, Hosting Plan**, and **Application Insights** or create new ones.

   Click the **Create a SQL database** option on the right side and create a new server and database within the resulting windows.

   Finally, click Create to **create** a publish profile. Alternatively, you can create the Azure SQL database directly on the Azure portal.



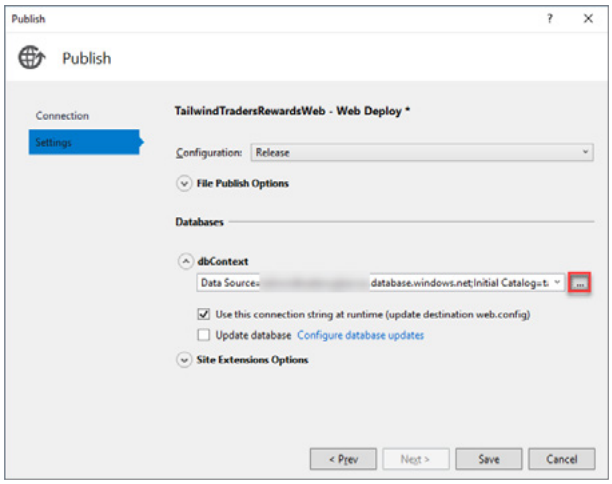
Create profile



Create profile

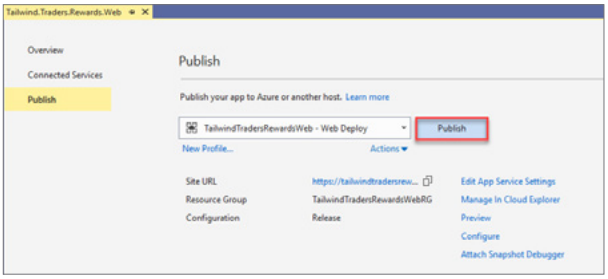
6. Click **Configure** in the Publish window to check the database connection strings. The database connection strings can be populated by selecting the ellipsis button and entering the SQL database details. On clicking **Publish**, the web config file will be updated with this database string, which is pointing to a SQL database.

   When the application is debugged locally, it's the local Internet Information Services (IIS) and the local SQL server that will act, but when the app is published this will be swapped with the created Azure services.



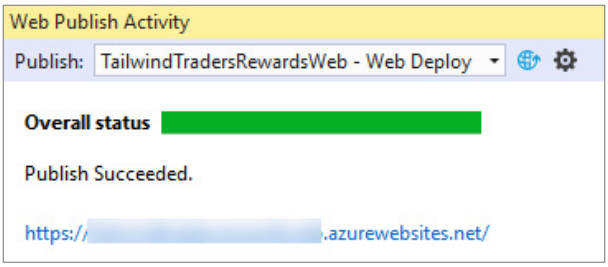
Database connection strings

7. Click **Publish** to deploy the application to Azure App Service and the back end to the SQL database.

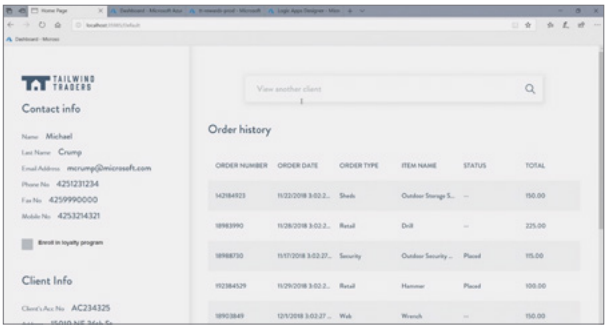


Publish app

8. Once the app is published, you'll see the status as **Publish Succeeded**, and the web app is opened in the browser. The website will now show data from the SQL database.



Publish Succeeded



Web app

Further reading

If you want to learn more about data and data analytics in Azure, you can download and read the following free e-books:

```
// Guide to NoSQL with Azure
   Cosmos DB
```

```
// Azure for Architects
```

```
// Migrating .NET Apps to Azure
```

03 /

Securing your application

How can Azure help secure your app?

Have you ever had a security incident with one of your applications? You might have had one without even knowing it. With Azure, you can protect data, apps, and infrastructure with built-in security services that include security intelligence to help identify rapidly evolving threats early—so you can respond quickly.

Azure can also help you implement a layered, in-depth defense strategy across identity, data, hosts, and networks. With services like the Azure Security Center, you can get an overview of your security status and recommendations for how to improve security.

Most importantly, you'll be notified as soon as there might be a security incident—so you'll always know if there's a threat. This way, you can take immediate steps to secure your assets. In this chapter, we'll dive in to some of them.

Azure Active Directory

An important part of your application's security is authenticating users before they can use it—but 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 (Azure AD) provides all of these things and more out of the box. You store your user identities in Azure AD and have users authenticate against it, redirecting them to your application only after they're authenticated. Azure AD takes care of password management, including resolving common scenarios like forgotten passwords.

Since Azure AD is used by millions of applications every day—including the Azure portal, Outlook.com, and Office 365—it's able to more readily detect and act on malicious behavior. 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 AD would flag this as malicious behavior and ask the user for additional credentials through multifactor authentication.

Azure Key Vault

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

```
// Try it out: Get started with
   Azure Key Vault
```

These secrets can include the credentials in a connection string. Your application would get the connection string from Key Vault instead of from the configuration system. This way, administrators can control the secrets, and developers never need to deal with them. Key Vault also stores SSL and other certificates used to secure the traffic to and from your applications over HTTPS.

Azure Sentinel

To get a good overview of the security status of your organization and all of its users, applications, services, and data, you can use a security information and event manager (SIEM) platform. Azure now offers an AI-powered SIEM in the form of Azure Sentinel.

Use Azure Sentinel to collect data from your organization, including data about users, applications, servers, and infrastructure assets like firewalls and devices running in the cloud and on-premises. It's easy to collect data from your organization with the built-in connectors. As data is being collected, Azure Sentinel detects security threats and minimizes false positives with its smart machine learning algorithms.

When there's a threat, you'll be alerted and can investigate it with AI, utilizing decades of cybersecurity work at Microsoft. You can respond to incidents with Azure Sentinel's built-in workflow orchestration and task automation.

```
// Get started by onboarding
   Azure Sentinel
```

Azure API Management

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

With API Management, you secure an API by requiring users to create a subscription to it. This way, 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's calling your API and block unwanted callers.

Much more than security

While security is critical, Azure API Management offers other capabilities that can help streamline your development and testing workflow, such as test data response mocking, publishing multiple API versions, introducing non-breaking changes safely with revisions, and giving developers access to your API's auto-generated documentation, catalog, and code samples.

```
// Try it out: Get started with
   Azure API Management
```

Azure AD Application Proxy

Azure AD Application Proxy provides single sign-on (SSO) and secure remote access for web applications hosted on-premises. Apps you'd likely want to publish include SharePoint sites, Outlook Web Access, or other line-of-business (LOB) web applications. These on-premises web apps integrate with Azure AD, the same identity and control platform used by Office 365. End users can access your on-premises applications the same way they access Office 365 and other SaaS apps integrated with Azure AD.

Managed Identities for Azure resources

How do you keep credentials out of your code completely? You can start by using Azure Key Vault, but where do you store the credentials to connect to Key Vault? Managed Identities for Azure resources provides a solution.

You can use Managed Identities for a lot of services in Azure, including Azure App Service. You simply enable Managed Identity with a button to inject credentials into your application at runtime, and then use those credentials to access other services like Azure Key Vault. All authentication between services is done on the infrastructure level, meaning your application doesn't have to deal with it and can just use other services.

```
// Try it out: How to use
   Managed Identities for Azure
   resources in App Service and
   Azure Functions
```

Encryption

Default encryption of data

By default, your data is encrypted in Azure when stored 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. All this encryption works automatically, and you don't need to configure anything when you use it.

To help meet your security and compliance requirements, you can use the following features to encrypt data at rest:

- Azure Disk Encryption encrypts Windows and Linux infrastructure as a service (IaaS) VM boot and data volumes using customer-managed keys.
- Azure Storage Service Encryption automatically encrypts data prior to persisting in Azure Storage, and then automatically decrypts the data when you retrieve it.
- Azure client-side encryption supports encrypting data within client applications before uploading to Azure Storage or other endpoints, and then decrypting data when downloading it to the client.
- SQL Transparent Data Encryption (TDE) encrypts SQL Server, Azure SQL Database, and Azure SQL Data Warehouse data files. Data and log files are encrypted using industry-standard encryption algorithms. Pages in a database are encrypted before they're written to disk and decrypted when they're read.
- SQL Always Encrypted encrypts data within client applications prior to storing it in Azure SQL Database. It allows delegation of on-premises database administration to third parties, and maintains separation between those who own and can view the data and those who manage it but should not access it.
- Azure Cosmos DB requires no action from you—user data stored in Azure Cosmos DB in non-volatile storage (solid-state drives) is encrypted by default, and there are no controls to turn it on or off.

Azure Security Center

Azure Security Center provides unified security management and advanced threat protection across hybrid cloud workloads. It offers centralized policy controls to limit exposure to threats and rapidly find and fix vulnerabilities.

In addition, Security Center supports integration with third-party solutions and can be customized with automation and programming capabilities. You can use Security Center to analyze the security state of your compute resources, virtual networks, storage and data services, and applications.

Continuous assessment helps you discover potential security issues, such as systems with missing security updates or exposed network ports. A list of prioritized findings and recommendations can trigger alerts or other guided remediation.

Azure DDoS protection

You've heard about it on the news, and you certainly don't want it to happen to your enterprise: an application is targeted 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 no one can use it anymore. The Azure DDoS protection service offers protection from DDoS attacks through a free tier (Basic) and a paid tier (Standard).

You don't have to do anything to enable the Basic tier—it's automatically enabled for every customer as part of the Azure platform. This service protects your applications against the most common DDoS attacks by performing real-time monitoring and mitigation, and it provides the same defenses used by Microsoft Online Services (MOS).

The Standard tier provides additional mitigation capabilities that are tuned specifically to Azure Virtual Network resources. It's simple to enable, and you don't have to change your applications—everything is done at the network level. Plus, with the Standard tier you can customize the Basic tier protection with your own policies that focus on your specific use cases and applications.

```
// More info: Read more about
   Azure DDoS protection
```

Azure VPN Gateway

One of the many options for connecting Azure to your on-premises network is Azure VPN Gateway. This lets you set up an encrypted Site-to-Site (S2S) VPN connection between an Azure virtual network and your on-premises network.

Because the traffic is encrypted, it's secure—even when it travels over the public internet. VPN Gateway can send encrypted traffic between Azure virtual networks over the Microsoft network. You can also create encrypted Point-to-Site (P2S) connections from your computer to Azure. This way, you have your own private, secured connection to Azure even when you're on the road.

```
// Get started by creating an Azure
   VPN Gateway with PowerShell
```

Azure Application Gateway

Azure Application Gateway is a dedicated virtual appliance that provides an application delivery controller (ADC) as a service. It offers various Layer 7 load balancing capabilities for your application, and allows customers to optimize web farm productivity by offloading CPU-intensive SSL termination to the application gateway. The gateway also provides other Layer 7 routing capabilities, including round-robin distribution of incoming traffic, cookie-based session affinity, URL path-based routing, and the ability to host multiple websites behind a single application gateway.

Azure Web Application Firewall

You need to secure your application against many threats, including SQL injection, Cross-site scripting (XSS), and others defined in the Open Web Application Security Project (OWASP). A WAF from Azure can lend a hand with that. A feature of the Azure Application Gateway service, a WAF provides real-time protection of your application. It detects malicious attacks, as defined in the OWASP core rule set, and blocks those attacks from reaching your application. It also reports on attempted or ongoing attacks so that you can see active threats to your application, providing an extra layer of security.

Azure Network Watcher

Azure Network Watcher is a regional service that enables you to monitor and diagnose conditions at the network level in, to, and from Azure.

Its many diagnostic and visualization tools can help you understand and gain deeper insights into your network in Azure.

Examples include:

- Topology: Provides a network-level view showing the various interconnections and associations between network resources in a resource group.
- Variable packet capture: Captures packet data in and out of a VM. Advanced filtering options and fine-tuned controls, such as the ability to set time and size limitations, provide versatility. The packet data can be stored in a blob store or on the local disk in .cap format.
- IP flow verify: Checks if a packet is allowed or denied based on 5-tuple flow information and packet parameters (destination IP, source IP, destination port, source port, and protocol). If the packet is denied by a security group, the rule and group that denied the packet are returned.

Network security groups

A network security group (NSG) holds a list of security rules that allow or deny network traffic to resources connected to Azure Virtual Networks. NSGs can be associated to subnets, individual VMs (classic-style VMs), or individual network interface controllers (NICs) attached to VMs (Resource Manager–style VMs). When an NSG is associated to a subnet, the rules apply to all resources connected to the subnet. You can restrict traffic even further by also associating an NSG to a VM or NIC.

Azure DNS Private Zones

The DNS is responsible for translating (or resolving) a service name to its IP address. Azure DNS is a hosting service for DNS domains, providing name resolution using the Azure infrastructure. In addition to internet-facing DNS domains, Azure DNS now supports private DNS domains as a preview feature with Azure DNS Private Zones. Security benefits from private DNS zones include the ability to create a split DNS infrastructure. This enables you to create private and public DNS zones with the same names without exposing internal names. In addition, the use of DNS Private Zones removes the need to introduce custom DNS solutions that could increase the overall attack surface with independent updating and management requirements.

```
// More info: Read more about
   DNS Private Zones
```

Cross-premises VPNs

Azure supports two types of cross-premises VPN connections: P2S VPN and S2S VPN. A P2S VPN connection lets you create a secure connection to your virtual network from an individual client computer. This type of connection is established from the client computer, which is useful for telecommuters who want to connect to Azure Virtual Networks from a remote location. A P2S VPN is also useful when you have only a few clients that need to connect to a virtual network. In contrast, an S2S VPN connection is used to connect your on-premises network to an Azure virtual network over an IPsec/IKE (IKEv1 or IKEv2) VPN tunnel. This type of connection requires a VPN device located on-premises that has an externally facing public IP address.

```
// More info: Read more about P2S
   and S2S VPNs
```

Azure ExpressRoute

Azure ExpressRoute lets you extend your on-premises networks into the Microsoft cloud over a secure private connection facilitated by a connectivity provider without traversing the internet. With ExpressRoute, you can establish connections to Microsoft cloud services, such as Azure, Office 365, and Dynamics 365.

Azure Load Balancer

You can use load balancers to increase the availability of applications. Azure supports both external and internal load balancers, which can be used in a public or internal configuration.

In addition, you can configure load balancers to support high availability (HA) ports where an HA ports rule is a variant of a load balancing rule configured on the internal Standard Load Balancer. You can provide a single rule to load balance all TCP and UDP flows arriving on all ports of an internal load balancer.

```
// More info: Read about Load
   Balancer and HA ports rules
```

Logging and monitoring

Azure Log Analytics

Azure Log Analytics helps you collect and analyze data generated by resources in your cloud and on-premises environments. It provides real-time insights by using integrated search and custom dashboards to analyze millions of records across all your workloads and servers regardless of their physical location.

Azure Monitor

Azure Monitor enables basic monitoring for Azure services by collecting metrics, activity logs, and diagnostic logs. The metrics collected provide performance statistics for different resources, including the OS associated with a VM.

The activity log will show you when new resources are created or modified. You can view this data with one of the explorers in the Azure portal and send it to Log Analytics for trending and detailed analysis, or you can create alert rules that will proactively notify you of critical issues.

Azure NSG flow logs

A feature of Network Watcher, Azure NSG flow logs allow you to view information about ingress and egress IP traffic through an NSG. Flow logs can be analyzed to gain information and insights into network traffic and security as well as performance issues related to traffic.

While flow logs target NSGs, they are not displayed in the same way as other logs and are stored only within a storage account.

Azure Monitor Application Insights

Azure Monitor Application Insights is an extensible application performance management (APM) service for web developers on multiple platforms. It includes powerful analytics tools to help you diagnose issues and understand what users do with your app. It works for applications on a variety of platforms hosted on-premises or in the cloud, including .NET, Node.js, and J2EE.

Application Insights integrates with your DevOps process and has connection points to a variety of development tools. It can monitor and analyze telemetry from mobile apps by integrating with Visual Studio App Center.

Azure Security and Compliance Blueprint

The Azure Security and Compliance Blueprint - HIPAA/HITRUST Health Data and AI provides tools and guidance to help deploy a platform as a service (PaaS) environment for compliance with the Health Insurance Portability and Accountability Act (HIPAA) and Health Information Trust Alliance (HITRUST).

This PaaS offering supports ingesting, storing, analyzing, and interacting with personal and non-personal medical records in a secure, multitier cloud environment deployed as an end-to-end solution. The blueprint showcases a common reference architecture that could be applied to use cases beyond healthcare, and is designed to simplify adoption of Azure.

```
// More info: Read more about
   the Azure Security and
   Compliance Blueprint
```

Azure security technical and architectural documentation

Azure maintains a large library of security technical documentation that supplements security information with individual services. White papers, best practices documents, and checklists are included on the Azure Security Information page.

Also covered are core public cloud security topics in diverse areas, including network security, storage security, compute security, identity and access management, logging and auditing, cloud workload protection, PaaS security, and more.

```
// More info: Read more about the
   Azure Security Information page
```

Further reading

Learn more about Azure security in the following free e-books:

```
// Enterprise Cloud Strategy
```

```
// Azure for Architects
```

04 /

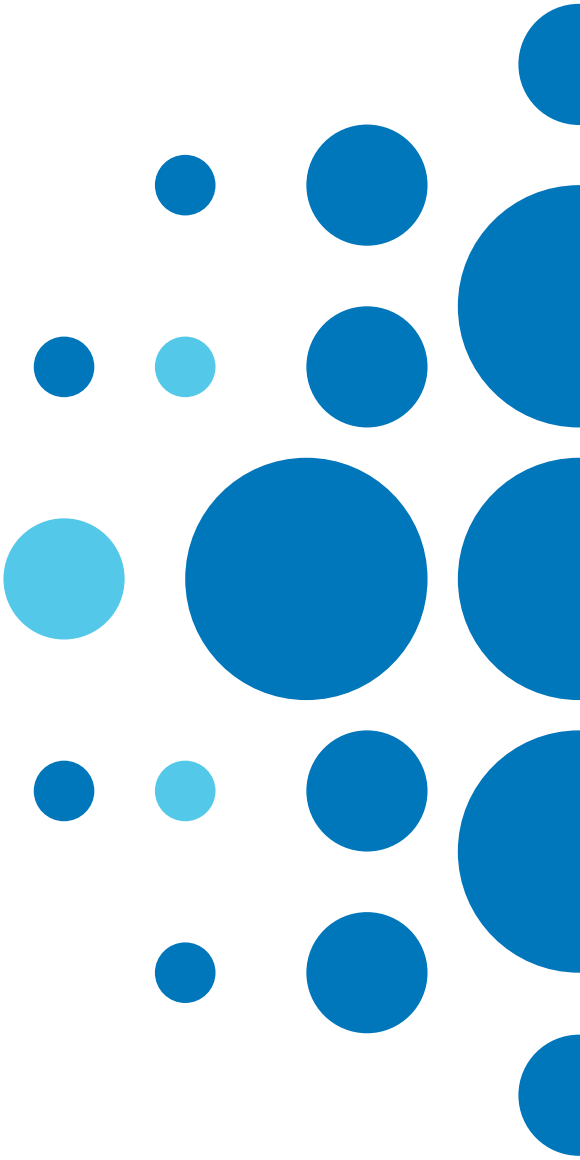
Adding intelligence to your application

How can Azure integrate AI into your app?

AI can give your application an edge over the competition. Just imagine what you can build—apps that translate speech in real time as you're speaking, or an app that helps you identify parts of a motor in a mixed-reality training. The possibilities are endless. But building an AI-powered app seems difficult. How do you create an algorithm that can understand speech, for instance?

Fortunately, you don't have to do everything yourself. Azure is here to help. It provides many AI services that you can just plug into your application, like the Speech Translation service (a service in Azure Cognitive Services) that translates speech in real time.

Using a service like this, you're able to just consume AI. But you can also build your own machine learning algorithms with services like Azure Machine Learning Studio and Azure Machine Learning service.



What to use, and when?

Before diving into the options for AI in Azure, let's look at Table 4-1, which summarizes which services are available and their capabilities.

Table 4-1

	Azure Search*	Azure Cognitive Services*	Azure Bot Service*	Azure Machine Learning Studio*	Azure Machine Learning*	Azure Spatial Anchors*	Azure Remote Rendering*
Move data from store to store	●						
Transform data	●	●	●	●	●	●	●
Query and filter streaming data				●		●	●
Provide in-memory semantic model for users		●			●		●
Users can query data and create dashboards					●		
Analyze data for later use			●		●		●

* Services with an asterisk have a free tier that you can use to get started at no cost.

Azure Search

Azure Search is a common feature, yet it has traditionally been difficult to implement. Azure Search provides a lot of the plumbing to do searches. You just spin up an Azure Search instance, create an index that helps you search, and fill it with data. This means, for example, that you could easily implement Azure Search to help users search your product catalog in an e-commerce application.

There are many options for tweaking Azure Search and great features to make searching easier for your users, such as:

- Geo-search, which lets users explore data based on the proximity of a search result to a physical location.
- Language analyzers from Apache Lucene as well as natural language processing (NLP) from Microsoft, available in 56 languages, which intelligently handle linguistics, including verb tense, gender, irregular plural nouns, word decompounding, and word breaking for languages with no spaces.
- Monitoring and reporting, which provide information on what was searched for and how fast and successful the search was.
- User experience features like sorting and paging search results, intelligent filtering, and providing search suggestions.
- Cognitive Search, which is an AI-first approach to understanding. Cognitive Search is powered by Azure Search with built-in Cognitive Services. It pulls data from almost any source and applies a set of composable cognitive skills that extract knowledge. This knowledge is then organized and stored in an index, enabling new experiences for exploring the data using Azure Search.

Cognitive Search is used by oil and gas companies, whose teams of geologists and other specialists need to understand seismic and geologic data. These teams often have decades of PDFs with pictures of samples along with handwritten field notes. The teams need to connect places, domain experts, and events and then navigate all this information to make key decisions.

Cognitive Search uses Cognitive Services to analyze all this data, extract information, and correlate it—all without the need to write complicated image recognition or Optical Character Recognition (OCR) software.

```
// Try it out: Create your first
   Azure Search index in the portal
```

Cognitive Services

Cognitive Services provides machine learning algorithms, created by Microsoft, and data as a service. For most services, Microsoft has also provided the data to train those algorithms. For some services, you can use your own custom data to train the algorithms.

Cognitive Services provides an exceptionally easy way to incorporate machine learning and AI into your application—by simply calling APIs.

Table 4-2 shows which APIs are currently available. Note that the list keeps growing.

All services have a free tier that you can use to get started.

Each category in the table contains multiple services that you can use by calling an API. Some categories contain custom services, like Custom Vision, Language Understanding, and Bing Custom Search. These custom services provide preconfigured machine learning algorithms, just like the other services, and they also enable you to use your own data to train the model. In addition to these services, you can use the services in the Cognitive Services Labs. The labs contain experimental services that Microsoft is trying out to see if they fit well with customer use cases. One such experimental service is Project Gesture, which enables you to detect gestures like the wave of a hand and weave them into your user experience.

Let's take a closer look at some of the Cognitive Services.

```
// Try it out: Explore
   Cognitive Services
```

Table 4-2

Vision	Speech	Language	Knowledge	Search
Computer Vision	Speech to Text	Text Analytics	QnA Maker	Bing Web Search
Face	Text to Speech	Translator Text		Bing Visual Search
Video Indexer	Speech Translation	Bing Spell Check		Bing Entity Search
Content Moderator	Speaker Recognition	Content Moderator		Bing News Search
Custom Vision		Language Understanding		Bing Custom Search
				Bing Image Search
				Bing Autosuggest
				Bing Video Search
				Bing Local Business Search

* All services have a free tier that you can use to get started.

Language Understanding

Use the Language Understanding (LUIS) service to understand what users are saying to you on social media, in chatbots, or in speech-enabled applications. For example, you can book flights or schedule meetings.

To use the Language Understanding service, give it examples of what you want it to understand, like "Book a flight to Seattle" or "Schedule a meeting at 1pm with Bob," and tell it which words you're looking for. In these examples, you might be looking for the destination of the flight (Seattle) and the time and person for the meeting (1pm and Bob).

After the Language Understanding service creates a machine learning model based on the examples that you put in, it can extract information from natural language that users put in.

```
// Try it out: Create a new app in
   the LUIS portal
```

Custom Vision

With the Custom Vision service, you can detect information in images based on your own training data. Custom Vision works similarly to other Cognitive Services in that it comes with a predefined machine learning algorithm. All you have to do is feed the service with your data.

Let's say you want to create a model that can detect types of rain clouds in the sky, such as cumulus and stratus. To create this model, you upload images of different types of clouds to the Custom Vision portal and give them tags, which tells the service how to train the model. In this example, you would tag an image with "cumulus" or "stratus."

Once you've uploaded enough images, you can train your model. The more images you upload with tags and the more training you do, the more accurate your model will be.

Once you have a model that performs well, you can start using it by making calls to the Custom Vision API and feeding it new images. When you upload a new image, the service tells you if it recognizes it based on the images already uploaded.
Figure 4-1 shows an example of what the API endpoint looks like.

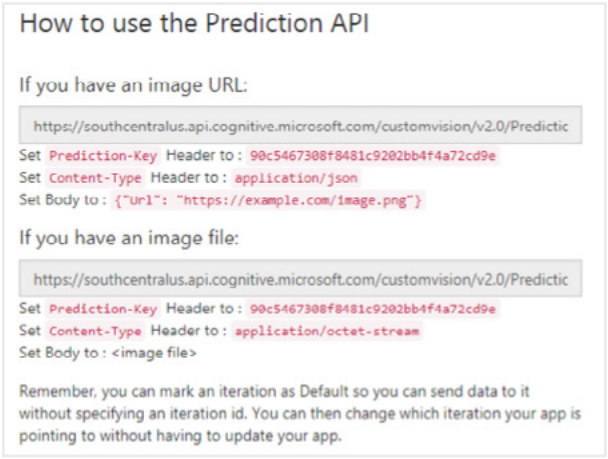


Figure 4-1

Using the Custom Vision service to detect information based on your own model is impressive enough, but Custom Vision can do even more. The model you create when you train the Custom Vision service with your data can be deployed to the "intelligent edge." This means the model and API can run somewhere other than the cloud, like on an on-premises server in a Docker container or on another device, such as your phone.
This offers great flexibility because you don't need

an active internet connection to use the capabilities of the Custom Vision service; you can also run it locally, which provides great performance. In addition, the model you run on the edge isn't very large—only approximately tens of megabytes—because you deploy only the model and API, not the training data.

```
// Try it out: Create your own
   Custom Vision project
```

Video Indexer

The Video Indexer service analyzes the video and audio files you upload to it. This Cognitive Service is also a part of the Media Analytics suite of Azure Media Services. It provides a predefined machine learning algorithm and you provide the data.

In addition to many others, Video Indexer has the following capabilities:

- Creates a transcript of the text in a video. You can refine the transcript manually and train Video Indexer to recognize industry terms like "DevOps."
- Tracks faces and identifies who is in a video and at what points. Video Indexer has the same capability for audio, for which it recognizes who is speaking and when.
- Recognizes visual text in a video, like text on a slide, and makes that part of the transcript.
- Performs sentiment analysis, which identifies when something positive, negative, or neutral is said or displayed.

As the breadth of these functionalities shows, Video Indexer combines many Cognitive Services, like Speech to Text and Speaker Recognition. Cumulatively, these services provide powerful capabilities that make content more discoverable, accessible, and valuable.

You can upload media files to Video Indexer using the Video Indexer portal or the API. Figure 4-2 shows the results of an Azure Friday video that was uploaded to the Video Indexer service.

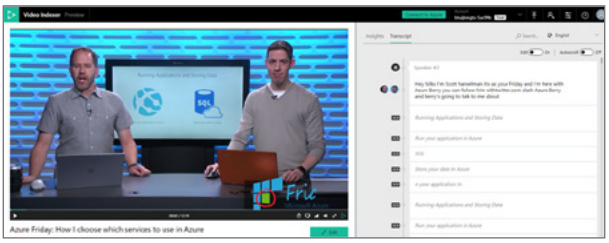


Figure 4-2

As shown in the figure, Video Indexer created a transcript of the audio in the video. The transcript can be edited and even translated into other languages. You can also see that Video Indexer recognized text on the slide behind the speakers and marked it as "OCR." You can skip to that text by clicking it. Video Indexer provides this functionality for individual applications by embedding the Cognitive Insights widget.

```
// Try it out: Upload your first
   video to Video Indexer
```

QnA Maker

The QnA Maker Cognitive Service offers an easy way to create a conversational layer over existing data, like the frequently asked questions (FAQ) pages, support websites, and product manuals. QnA Maker helps you analyze and extract the information and convert it into question-and-answer pairings that can be easily managed. Simply put, QnA Maker allows you to build apps that can provide information to your users in a conversational manner.

With QnA Maker, it's possible to create and manage knowledge bases using the easy-to-use QnA Maker Portal or using REST APIs. We have simplified the bot creation process by allowing you to easily create a bot from your knowledge base—without the need for any code or settings changes. See more details here: Create a QnA bot. Of course, you can also use QnA Maker to create a bot using the Azure Bot Service and augment your QnA bot by adding the Language Understanding Service. To add personality, you can add chit-chat to your bot, and answer commonly asked small talk scenarios out of the box.

You pay only for the hosting of QnA Maker, not for how many times the resulting service gets queried by users.

```
// Try it out: Create your QnA Maker
   knowledge-based service
```

Bing Autosuggest

Bing Autosuggest provides search suggestions while you type. This enables you to give your users a search experience similar to using Bing or Google, in which search results are automated or completed.

Provide the search text character by character to Bing Autosuggest, and it quickly returns search suggestions in JSON format.

For instance, when you input the query text "What should I search for," the service returns the following JSON (see page 59 for larger figure):


```
{
  "_type": "Suggestions",
  "instrumentation": null,
  "queryContext": {
    "originalQuery": "what should I search for"
  },
  "suggestionGroups": [
    {
      "name": "Web",
      "searchSuggestions": [
        {
          "url": "https://www.bing.com/search?q=what+should+i+search+for&FORM=USBAPI",
          "urlPingSuffix": null,
          "displayText": "what should i search for",
          "query": "what should i search for",
          "searchKind": "WebSearch"
        },
        {
          "url": "https://www.bing.com/search?q=what+should+i+search+for+on+bing&
FORM=USBAPI",
          "urlPingSuffix": null,
          "displayText": "what should i search for on bing",
          "query": "what should i search for on bing",
          "searchKind": "WebSearch"
        },
        {
          "url": "https://www.bing.com/search?q=what+should+i+search+for+on+the+internet&
FORM=USBAPI",
          "urlPingSuffix": null,
          "displayText": "what should i search for on the internet",
          "query": "what should i search for on the internet",
          "searchKind": "WebSearch"
        },
        {
          "url": "https://www.bing.com/search?q=what+should+i+search+for+today&
FORM=USBAPI",
          "urlPingSuffix": null,
          "displayText": "what should i search for today",
          "query": "what should i search for today",
          "searchKind": "WebSearch"
        },
        {
          "url": "https://www.bing.com/search?q=what+should+i+search+for+in+dna+raw+data&
FORM=USBAPI",
          "urlPingSuffix": null,
          "displayText": "what should i search for in dna raw data",
          "query": "what should i search for in dna raw data",
          "searchKind": "WebSearch"
        }
      ]
    }
  ]
}
```


This contains all the suggestions. The original search query is contained in the top of the results.

```
// Try it out: Get an API key and
   try out Bing Autosuggest for free
```

```
{
  “_type”: “Suggestions”,
  “instrumentation”: null,
  “queryContext”: {
    “originalQuery”: “what should I search for”
  },
  “suggestionGroups”: [
    {
      “name”: “Web”,
      “searchSuggestions”: [
        {
          “url”: “https://www.bing.com/search?q=what+should+i+search+for&FORM=USBAPI”,
          “urlPingSuffix”: null,
          “displayText”: “what should i search for”,
          “query”: “what should i search for”,
          “searchKind”: “WebSearch”
        },
        {
          “url”: “https://www.bing.com/search?q=what+should+i+search+for+on+bing&
FORM=USBAPI”,
          “urlPingSuffix”: null,
          “displayText”: “what should i search for on bing”,
          “query”: “what should i search for on bing”,
          “searchKind”: “WebSearch”
        },
        {
          “url”: “https://www.bing.com/search?q=what+should+i+search+for+on+the+internet&
FORM=USBAPI”,
          “urlPingSuffix”: null,
          “displayText”: “what should i search for on the internet”,
          “query”: “what should i search for on the internet”,
          “searchKind”: “WebSearch”
        },
        {
          “url”: “https://www.bing.com/search?q=what+should+i+search+for+today&
FORM=USBAPI”,
          “urlPingSuffix”: null,
          “displayText”: “what should i search for today”,
          “query”: “what should i search for today”,
          “searchKind”: “WebSearch”
        },
        {
          “url”: “https://www.bing.com/search?q=what+should+i+search+for+in+dna+raw+data&
FORM=USBAPI”,
          “urlPingSuffix”: null,
          “displayText”: “what should i search for in dna raw data”,
          “query”: “what should i search for in dna raw data”,
          “searchKind”: “WebSearch”
        }
      ]
    }
  ]
}
```

Azure Bot Service

Creating a bot—an application that automatically and autonomously interacts with users—is no trivial task. You need to keep track of the context of your interaction with each user and be ready to respond to a multitude of possible interaction parameters.

The Azure Bot Service enables you to build intelligent, enterprise-grade bots and experiences that can extend your brand and keep you in control of your data. Begin with a simple Q&A bot or build a sophisticated virtual assistant. Use comprehensive open-source SDK and tools to easily connect your bot across popular channels and devices. Give your bot the ability to speak, listen, and understand your users with native integration to Cognitive Services.

Azure Bot Service makes it easy to create a bot and provides the following support:

- Provides a way to host and manage bots you've built using the Microsoft Bot Framework, with a comprehensive open-source SDK and tools for bot development.
- Integrates natively with Cognitive Services.
- Helps you connect your bot to where your customers are, with connectors to channels like Facebook, Slack, Microsoft Teams, Line, Telegram, and more.
- Offers all the benefits of a managed service in Azure, including massive scale and built-in CD, and you pay only for what you use.

An example of a bot you can build with Azure Bot Service is one that provides users with answers to their most frequently asked questions.

You can use this with the QnA Maker Cognitive Service. The interface of the bot can be a chat box on your website. You could also build your own branded virtual assistant using the virtual assistant solution accelerator.

```
// Try it out: Get started
   with chatbots using Azure
   Bot Service
```

Azure Machine Learning Studio

You can add intelligence to your applications with services from Azure such as Cognitive Services. These are based on machine learning algorithms that Microsoft created to use as a service. However, there are other ways to use machine learning in your applications. First, let's talk about what machine learning is.

What is machine learning?

Machine learning is often thought to mean the same thing as AI, but they aren't actually the same. AI involves machines that can perform tasks characteristic of human intelligence. AI can also be implemented by using machine learning, in addition to other techniques.

Machine learning itself is a field of computer science that gives computers the ability to learn without being explicitly programmed. Machine learning can be achieved by using one or multiple algorithm technologies, like neural networks, deep learning, and Bayesian networks.

So what's involved in machine learning? Figure 4-3 shows the basic workflow for using machine learning.

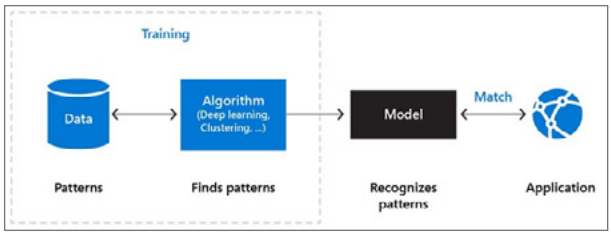


Figure 4-3

The machine learning process works as follows:

- Data contains patterns. You probably know about some of the patterns, like user ordering habits. It's also likely that there are many patterns in data with which you're unfamiliar.
- The machine learning algorithm is the intelligent piece of software that can find patterns in data. This algorithm can be one you create using techniques like deep learning or supervised learning.
- Finding patterns in data using a machine learning algorithm is called "training a machine learning model." The training results in a machine learning model. This contains the learnings of the machine learning algorithm.
- Applications use the model by feeding it new data and working with the results. New data is analyzed according to the patterns found in the data. For example, when you train a machine learning model to recognize dogs in images, it should identify a dog in an image that it has never seen before.

The crucial part of this process is that it is iterative. The machine learning model is constantly improved by training it with new data and adjusting the algorithm or helping it identify correct results from wrong ones.

Using Azure Machine Learning Studio to create models

You can use Azure Machine Learning Studio to create your own custom machine learning models and expose them through web services so that your applications can use them.

Machine Learning Studio is a service in Azure with which you can visually create machine learning projects and experiments, couple datasets, create notebooks, and expose models with web services. The studio itself is a portal that you can use from your web browser and that enables you to create algorithms using a drag-and-drop approach.

In the studio, you can start from scratch or with one of the many experiments that are in the gallery, including one for predicting length of stays in hospitals and another for anomaly detection in real-time data streams. Use these experiments as the basis for a machine learning model or to learn how these cases can be solved.

A machine learning experiment in Machine Learning Studio consists of multiple steps that manipulate data and execute machine learning algorithms on it. Use predefined steps available in the studio to compose a machine learning algorithm.

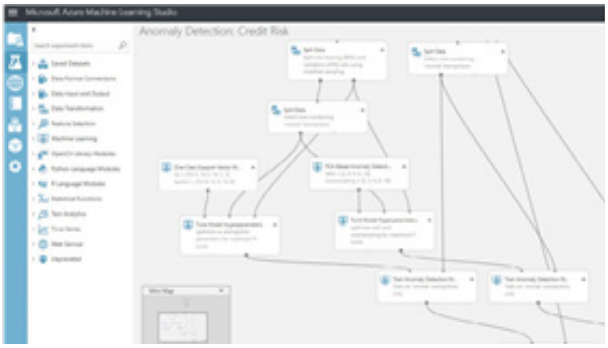


Figure 4-4

Figure 4-4 shows an experiment in Machine Learning Studio, with the workflow to be executed to train a model as well as the categories of predefined steps that can be used in the workflow.

When you've built your experiments and used them on your data to create a machine learning model, you can publish them as web services. When your applications use the web services, they can send data to your model and receive your model's predictions.

```
// Try it out: Sign up to use
   Azure Machine Learning Studio
```

Azure Machine Learning service

You can use Azure Machine Learning service to create data analytics algorithms with open-source tools like Python and the Azure CLI. Just like with

Machine Learning Studio, you can create whatever algorithm you want, providing flexibility for a variety of scenarios, like predictive analytics, data recommendations, and data classification.

With Azure Machine Learning service, you create custom machine learning algorithms from scratch. This is different from the Machine Learning Studio, where you visually create an algorithm by connecting predefined pieces of a machine learning algorithm. Azure Machine Learning service fully supports open-source technologies like Google TensorFlow, PyTorch, and scikit-learn.

Azure Machine Learning service is a complete service that offers start-to-finish capabilities. You can create your algorithm, prepare your data, train the algorithm on it, test and deploy the algorithm, and track and manage it when it's running.

Azure Machine Learning service works with many Azure services that can help create, train, and run your algorithm. You can, for instance, create your algorithm in Jupyter Notebook, train it using Azure Databricks, and deploy it on a Kubernetes container cluster in Azure Kubernetes Service.

```
// Get started with Azure
   Machine Learning service
   by using the Azure portal
```

Developer tooling for AI

Visual Studio Tools for AI

Visual Studio Tools for AI is a free Visual Studio extension. Use it to access a range of AI services and frameworks, including the Microsoft Cognitive Toolkit (CNTK), TensorFlow, Keras, and Caffe2.

Visual Studio Tools for AI allows you to create machine learning algorithms similarly to Azure Machine Learning Studio. You can use languages like Python, C, C++, and C# or leverage one of the many samples in the machine learning experiments gallery.

With Visual Studio Tools for AI, you can create machine learning elements from Visual Studio and take advantage of the power of Visual Studio to debug machine learning algorithms and train machine learning models. From Visual Studio, you can create training jobs that can scale out to many VMs in Azure. You can also monitor training performance and then generate a web service to use the machine learning model in your applications. You can do all this without ever leaving Visual Studio.

```
// Try it out: Download the Visual
   Studio Tools for AI extension
```

AI Toolkit for Azure IoT Edge

Using machine learning models locally on devices (the intelligent edge) delivers a powerful advantage: it enables you to use the local processing power of the device without relying on an internet connection or incurring the latency of a web service call to get your results.

Described earlier in the Cognitive Services section, the Custom Vision service already supports running on the edge. You can expect more services to run on the edge in the future.

To run machine learning models on the edge, you need tooling to help you deploy the models and web services. The AI Toolkit for Azure IoT Edge helps with this tooling by enabling you to package machine learning models in Azure IoT Edge–compatible Docker containers and to expose those models as REST APIs.

The AI Toolkit for Azure IoT Edge contains examples for getting started and is fully open source and available on GitHub.

AI and mixed reality

Your applications are no longer limited to a 2D environment. The world is now your app canvas, backed by spatial intelligence from things like IoT sensors, mixed reality, and computer vision. With Azure mixed-reality services, you can bring data to life in 3D when and where your users need it.

Azure Spatial Anchors

In the world of mixed reality, you can integrate digital information within the context of your physical environment, like a hologram of your favorite game characters on your kitchen counter. With Azure Spatial Anchors, you can place digital content in a physical location and share that with users using your choice of devices and platforms.

For example, people entering a hospital often have difficulty getting to where they need to go. With Azure Spatial Anchors, the hospital can create a mobile app that shows digital information in the physical hospital to guide people to various locations. Within the app on their iOS device, people can use the directional arrows on the hospital's physical information boards to get to their destinations.

Another way Azure Spatial Anchors is being used is in a training application for nurses. Pearson Education has enabled nursing students and professors to practice diagnosing and treating patients in 3D before the pressure of a real case. Students and professors may use HoloLens devices or mobile phones and tablets running iOS or Android.

Azure Spatial Anchors enables you to share digital information and holograms that are positioned in the physical world. It works with apps built on Unity, ARKit, ARCore, and Universal Windows Platform (UWP) and can be used with a HoloLens device, iOS-based devices supporting ARKit, and Android-based devices supporting ARCore.

With Azure Spatial Anchors, you can easily secure your spatial data and give users access through Azure Active Directory. You can also integrate storage, AI, analytics, and IoT services into your spatial application.

```
// Get started by sharing Azure
   Spatial Anchors across sessions
   and devices
```

Azure Remote Rendering

When you use 3D models in scenarios like design reviews and medical procedure plans, you need them to be as detailed as possible. Every detail matters.

Many businesses use complex 3D models containing hundreds of millions of polygons, and edge devices with low or medium graphics-processing power are not capable of rendering them. Traditionally, developers have tried to address this problem using a technique called "decimation." This makes the model simpler by removing polygons so it can display on those devices.

But this loss of detail sacrifices information needed to make the right decision in many situations. With Azure Remote Rendering, 3D models are rendered in the cloud and streamed to devices in real time—with no compromise on visual quality.

This enables you to keep the original quality of the model and interact with the content on edge devices like headsets and mobile phones with every detail intact.

Using events and messages in your application

Modern, globally distributed applications often must deal with large amounts of messages coming in, so they need to be designed with decoupling and scaling in mind. Azure provides several services to help with event ingestion and analysis as well as messaging patterns. These services are also vital for creating intelligent applications that leverage AI.

Azure Service Bus

The core of messaging in Azure is the Azure Service Bus. Service Bus encompasses a collection of services that you use for messaging patterns. The most important services are Azure Service Bus queues and topics.

```
// Get started with Azure Service
   Bus queues
```

Azure Service Bus queues

Azure Service Bus queues decouple systems from one another. For example, a web application receives orders from users and needs to invoke a web service to process the orders. The web service will take too long to process the orders, perhaps up to five minutes.

One way to solve this problem 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 a Service Bus queue. Then the web application informs 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 has processed 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 application's needs.

A Service 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. There are 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 difference between them:

- 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.

Applications can create a subscription on the topic that indicates what type of messages they're 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 Queue storage

Service Bus queues and Azure Queue storage basically do the same thing, but there are differences, as shown in Table 4-3.

Table 4-3

Azure Service Bus queues	Azure Queue storage
Message lifetime >7 days	Message lifetime <7 days
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

Azure Event Hubs can help enterprises capture massive amounts of data to analyze it or transform and move it for later use.

Event Hubs is designed for massive data ingestion. It effortlessly handles millions of messages per second. It retains messages for up to seven days or indefinitely by writing messages to a data store using the Event Hubs Capture feature.

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

```
// Try it out: Get started sending
   messages to Azure Event Hubs
```

Azure IoT Hub

Just like Event Hubs, Azure IoT Hub is built for massive data ingestion. It's specifically geared toward handling the enormous volume of data messages from devices on the Internet of Things, like smart thermostats and 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 it can send messages to devices. It has the ability to manage your complete IoT infrastructure—you can use it to register devices, report their state, manage them by securing and restarting them, and send data to them.

```
// Try it out: Connect your device
   to your IoT hub
```

Azure Event Grid

Azure Event Grid offers a different type of messaging—a fully managed publish and subscribe service that hooks into almost every service in Azure as well as 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 topic for new messages. Event Grid automatically pushes messages to subscribers, making it a real-time, reactive event service.

Services in and outside of Azure publish events when a new blob is added, for example, or when a new user is added to an Azure subscription. Azure Event Grid detects these events and makes them available to event handlers and services that subscribe to the events, as shown in Figure 4-5.

Event handlers can be Azure Functions or Azure Logics Apps, which can then act on the data in the event.

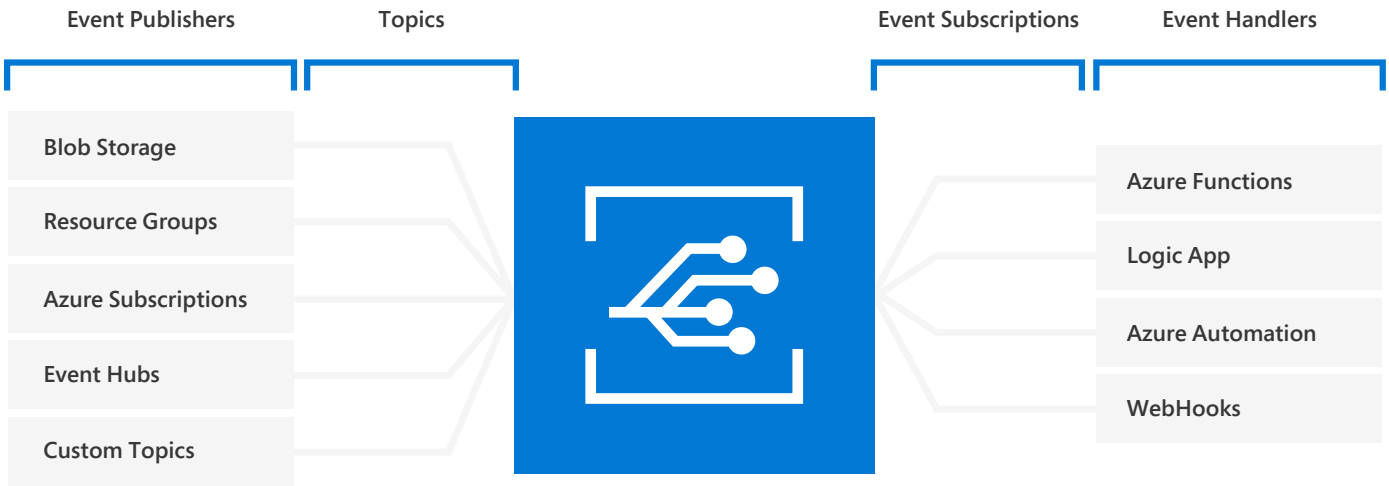


Figure 4-5

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 to be deployed. You just configure it and use it, and you pay only when it's used.

You can use Azure Event Grid if you want an email notification every time a user is added to or deleted from your mailing list in Mailchimp. Azure Event Grid is used to activate an app in Azure Logic Apps and configured to listen to changes to the Mailchimp mailing list. Azure Event Grid then signals Logic Apps to send an email containing the name of a user who has been added or deleted and the action that was performed.

```
// Try it out: Monitor virtual
   machine changes with Azure
   Event Grid and Logic Apps
```

Azure SignalR Service

You can use Azure SignalR Service to add real-time web functionality to your applications. The service is based on ASP.NET Core SignalR and is offered as a standalone, fully managed service in Azure.

SignalR can update connected applications in real time over HTTP, without the need for the applications to poll for updates or submit new HTTP requests. This enables you to create seamless web experiences that update information on the fly. For example, an auction application might use SignalR to refresh the latest bid as soon as it happens, without completely refreshing the page or constantly polling for information.

Hosting a SignalR server yourself is not a trivial task, and it can be difficult to scale and secure properly. When you use the fully managed Azure SignalR Service, setup is easy, and security, availability, performance, and scalability are all managed for you.

```
// Try it out: Create a chat room
   with SignalR
```

What to use, and when?

Azure provides myriad options to perform messaging and to decouple applications. Which one should you use, and when? Table 4-4 summarizes the differences to help you choose.

Further reading

You can learn more about using Azure AI services in your application in this free e-book:

```
// A Developer's Guide
   to Building AI Apps
```

Table 4-4

	SignalR Service*	Event Grid*	Event Hubs*	IoT Hub*	Topics*	Service Bus queues*	Azure Queue storage*
Event ingestion		●	●	●			
Device management				●			
Messaging	●	●	●	●	●	●	●
Multiple consumers	●	●	●	●	●		
Multiple senders	●	●	●	●	●	●	●
Use for decoupling			●	●	●	●	●
Use for publish/subscribe	●	●					
Max message size	64 KB	64 KB	256 KB	256 KB	1 MB	1 MB	64 KB

* Services with an asterisk have a free tier that you can use to get started at no cost.

05 /

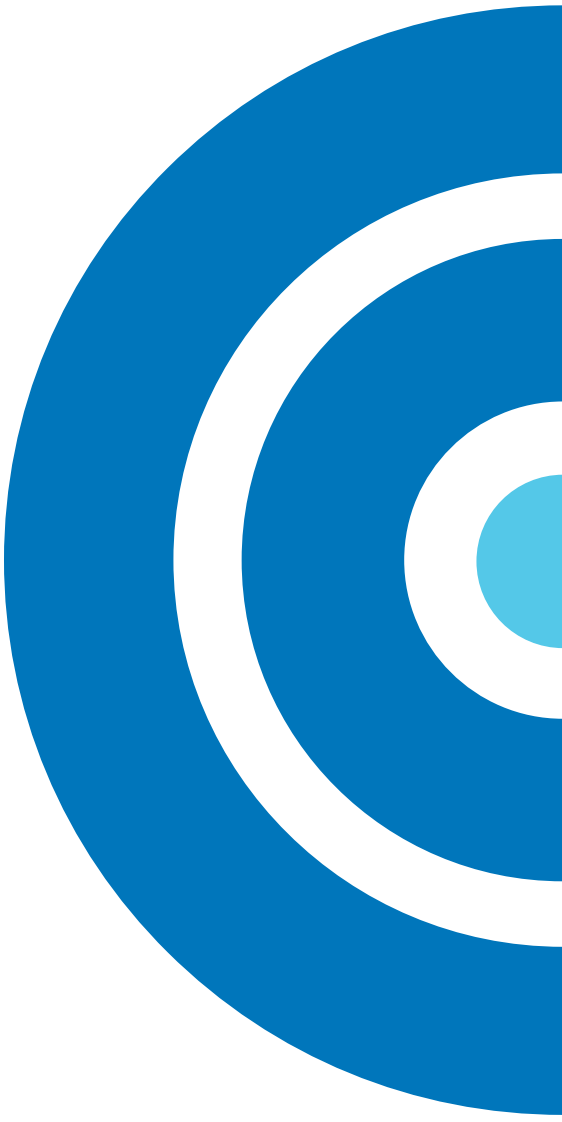
Connect your business with IoT

How can Azure connect, secure, manage, monitor, and control your devices in the cloud?

One of the most exciting things that you get to do as a developer is impact the real world. You can do that with devices like robots, sensors, and microcontrollers. When you create applications with these devices, you can do things like predict when a machine needs maintenance before it actually requires repairs or even create a self-driving car.

Working with devices to impact the real world can be difficult if you develop all the software yourself. Fortunately, Azure provides solutions that can help make this a lot easier. You can, for instance, use Azure IoT Hub to securely ingest messages from sensors and perform device management, like sending messages to devices and resetting them. When you're building anything with a microcontroller (single-chip computers) in it, consider using Azure Sphere, which is a combination of hardware, OS, and cloud services with security built into the silicon.

Let's go through the services in Azure that can help you build amazing IoT applications.



Azure IoT Hub

At the core of Azure IoT is Azure IoT Hub, an open and flexible cloud PaaS that connects, monitors, and manages devices in a secure and scalable manner. We discussed Azure IoT Hub in an earlier chapter, but there's more to learn about it in the context of IoT.

You already know that you can use IoT Hub to ingest massive amounts of messages that typically come from IoT devices, like messages that contain data from temperature sensors. What's more, IoT Hub is unique because it not only receives messages but also sends commands back to devices. It establishes two-way communication with devices and even lets you execute code on devices.

IoT Hub is powerful because it allows you to manage devices in various ways, like sending them a message to reboot themselves or running a startup script. This makes IoT Hub the central service that enables a robust IoT application in Azure. To help provision devices at scale, Azure provides the IoT Hub Device Provisioning Service.

This service enables zero-touch, just-in-time provisioning to the appropriate IoT hub without intervention, allowing you to provision devices in a secure and scalable manner. The service can help you with many device provisioning scenarios, including connecting devices to an IoT hub and running their initial setup scripts, load balancing devices across multiple hubs, and reprovisioning based on a change in the device.

IoT Hub can also connect devices that can run workloads developed in the cloud, including those that run the Azure IoT Edge runtime and modules. Once a device is connected to IoT Hub, the hub has a record of its identity. This enables IoT Hub to send messages and monitor the device; it also allows IoT Hub to secure the device and the communications between them. Devices are required to authenticate to IoT Hub using several industry best-practice security protocols, like X.509 certificates and SAS token-based authentication. You can manage the security of each connected device and revoke privileges if you no longer want a particular device to be connected.

When devices send messages to Azure IoT Hub, you can either store the messages or route them to another service for analysis or action. It's possible,

for instance, to route incoming messages using IoT Hub message routing that offers simplicity, reliability, and scalability without the complexity of building custom routing solutions.

Another option for integrating IoT events into Azure services or business applications is to use Azure Event Grid, a fully managed event routing solution that uses a publish-subscribe model. IoT Hub and Event Grid work together to integrate IoT Hub events into Azure and non-Azure services in near-real time.

You can also create bi-directional communication tunnels using device streams. Azure IoT Hub device streams facilitate the creation of secure bidirectional TCP tunnels for a variety of cloud-to-device communication scenarios.

Example: Azure IoT Hub

A company that provides insights into the movement and usage of trucks is scaling out its business. Previously, the company tracked its assets by using custom code on a native phone app, which called a custom web service, and by polling GPS dongles attached to the trucks. This solution was challenging to maintain because it was difficult to provision new assets with new devices, and the company needed to enroll a new customer that had more than 2,000 assets.

Now the company uses Azure IoT Hub for device management and communication. It can use the IoT Hub Device Provisioning Service to onboard the 2,000 new devices and hook them up to a specific IoT hub for that customer. The phone app now uses Node.js and the Azure IoT Device SDK to interact with IoT Hub. Importantly, the company now has control over the security of its devices and can detect their status and reset them as needed. In addition, the company routes the data from its GPS dongles through Azure Stream Analytics, so only the data of GPS changes is kept. This reduces the data burden because the dongles send their location every second.

Using Azure IoT Hub enabled this company to scale and mature its business by providing first-class security and device management. It also opened new opportunities to do more with devices than the company ever thought possible.

Azure IoT Central

Azure IoT Central is a hosted IoT solutions platform that enables you to create rich IoT applications just by navigating through wizards.

There's no need to perform any coding or in-depth configuration—IoT Central does all that for you, provisioning and configuring everything you need, including Azure IoT Hub.

You get the same capabilities as if you had created the solution from scratch, but without the need for years of programming experience. If you do want more control over certain areas, you can always go deeper and tweak the solution to your needs.

Azure IoT solution accelerators

Azure I oT solution accelerators are a great place to start building your IoT solution.

These comprehensive, customizable templates for common IoT scenarios do everything from monitoring and securing devices to providing a user interface. They also help connect existing and new devices. Figure 5-1 shows an example of a solution template.

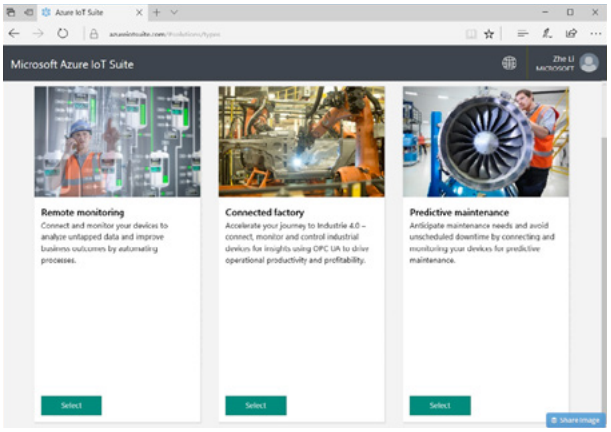


Figure 5-1

Connect and monitor your devices with remote monitoring. Get better visibility into your devices, assets, and sensors, wherever they're located. Collect and analyze real-time device data using a preconfigured remote monitoring solution accelerator that triggers automatic alerts and actions on everything from remote diagnostics to maintenance requests.

There are many more IoT solution accelerators, like those that improve industrial efficiencies with a connected factory, increase equipment reliability with predictive maintenance, and develop and test an IoT solution with device simulation.

Azure IoT Edge

In modern IoT applications, data processing can occur in the cloud or on the device side. Device-side processing is referred to as "edge computing."

You would use edge computing when you don't want to or can't rely on your connection to the cloud, when you want to improve your application performance by eliminating roundtrips to the cloud, or when you can't communicate with the cloud from the device because of security, privacy, or regulatory reasons.

For scenarios such as these, use Azure IoT Edge. Azure IoT Edge is managed from IoT Hub, enabling you to move parts of your workload to the edge. This reduces time spent by devices sending messages to the cloud and allows offline scenarios as well as faster reactions to status changes.

Azure IoT Edge is composed of the following components:

- **IoT Edge modules** are containers that run Azure services, third-party services, or your own code. They're deployed to IoT Edge devices and execute locally on those devices.
- The **IoT Edge runtime** runs on each IoT Edge device and manages the modules deployed to each device.
- **IoT Hub** exposes specific interfaces to remotely monitor and manage IoT Edge devices available through the Azure portal, the Azure CLI, or the SDKs.

These three components work together on devices and in the cloud to run your workloads at the intelligent edge.

It's possible run many Azure services at the edge to help with certain scenarios—and the list of available services keeps growing. Table 5-1 lists just some of them.

Table 5-1

If you want to	Use this on Azure IoT Edge
Build and deploy AI models	Machine learning
Customize computer vision models for your use case	Custom Vision Service
Process real-time streaming data	Stream Analytics
Process events using serverless code	Functions
Deploy a SQL Server database to the edge	SQL Server databases
Comply with Industry 4.0 interoperability standards	OPC Unified Architecture
Build custom logic	Custom module

Once you start using Azure IoT Edge, you'll be able to create fast applications that run machine learning algorithms locally and provide instant feedback on their findings.

Azure Digital Twins

In the world of IoT, you might work with many IoT devices and sensors that function in connection to people and objects. To really understand the data that IoT devices capture, you need to understand the physical environment in which the devices operate.

For instance, if you have a temperature sensor, the temperature data by itself doesn't tell you that much. However, when you know which room the sensor is in, how large the room is, which other devices are in the room and what data they capture, and how many people are in the room and how they're moving, you get a much better picture of what the temperature data means. You can model the physical environment in which your IoT devices reside with Azure Digital Twins. This service enables you to create a graph of data that includes places, people, and things—for instance, an office building that contains a room with people and sensors.

With Azure Digital Twins, you can provide context to data from various sources and relate them to each other—for instance, the temperature and humidity data from sensors in the same room. This allows you to query data in the context of a space rather than from individual sensors.

Azure Digital Twins also lets you to manage permissions to data and devices in the context of the physical world. You can use Azure AD to specify that certain users are able to access data only from a certain physical location.

```
// Get started by finding
   available rooms using Azure
   Digital Twins.
```

Azure Sphere

More and more devices contain microcontrollers (single-chip computers) to make them smarter. This enables scenarios such as a washing machine sending a routine maintenance notification to the owner.

However, it's complex and challenging to secure devices with embedded microcontrollers, as evidenced by the many security incidents with connected devices over the last years.

Azure Sphere provides a solution for securing microcontroller-based devices. The Azure Sphere platform consists of a secure microcontroller chip, OS, and set of cloud services that connect to the microcontroller and update it as it runs. This combination provides the basis for a connected, secure world.

You can use the Visual Studio development tools to develop applications that run on Azure Sphere. This opens up the world of microcontroller development to a much larger group of developers.

```
// Read more about Azure Sphere
```

Learn more about Azure IoT

Azure IoT solutions are easy to use, and there are many related resources, including:

- Azure IoT School: This free online academy provides comprehensive training for Azure IoT, with a variety of courses ranging from beginner to advanced.
- Building IoT Solutions with Azure: This guided online learning experience takes you through all the major Azure IoT concepts at your own pace.
- Azure IoT application page: This resource provides an overview of Azure IoT and examples of how it can be used.
- Azure IoT solution accelerators: Use these templates to get started with Azure IoT.
- Azure IoT Hub: This resource provides an overview of Azure IoT Hub and examples of how it can be used.
- Azure IoT Edge: This resource provides an overview of Azure IoT Edge and an example of how it can be used.
- Azure IoT technical videos: Learn more about IoT on Channel 9.

What to use, and when?

Now that you've read about the available Azure IoT services, how do you know which service to use for your scenario? Table 5-2 shows when you should use each IoT option in Azure.

Further reading

Learn more about using Azure for your IoT solution in the following free e-books:

```
// Developer's Guide to IoT
```

```
// Designed to Disrupt
```

Table 5-2

	Azure IoT Hub	Azure IoT Central	Azure IoT Solution accelerators	Azure IoT Edge
Create an IoT solution with a lot of control and by doing custom coding	●			
Create an IoT solution without worrying about code and management of Azure services		●		
Create an IoT solution for a common scenario with minimal configuration and coding			●	
Run AI workloads locally on IoT devices	●*			●

All services have a free tier you can use to get started.

*Azure IoT Hub is required to manage Azure IoT Edge deployments and devices.

06 /

Where and how to deploy your Azure services

How can Azure deploy your services?

Azure has an option for every type of organization, including those who need Azure to be in their own datacenter. You can deploy your applications either in the public Azure cloud or on-premises in Azure Stack choose how portable your applications should be.

It's also possible to develop apps in containers to deploy them in containers to deploy them on-premises or in another cloud, or by using Azure Resource Manager templates to script your complete infrastructure as code.

Let's explore these options in more detail.

Infrastructure as Code

Infrastructure as Code (IaC) captures environment definitions as declarative code, such as JSON documents, for automated provisioning and configuration. All Azure services introduced in this guide are based on Azure Resource Manager, which you can use to document your environment as IaC thanks to Azure Resource Manager templates. These templates are JSON files that describe what you want to deploy and what the parameters are.

It's easy to create Azure Resource Manager templates in Visual Studio and Visual Studio Code using Azure Resource Group project templates. You can also generate Azure 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 creates the Azure Resource Manager template for the given resource and even generates code for building the resource using the Azure CLI, PowerShell, .NET, and others.

After you have an Azure Resource Manager template, you can deploy it to Azure by using PowerShell, the Azure CLI, or Visual Studio. Or you can automate its deployment in a continuous deployment (CD) pipeline using Azure DevOps.

A great example of deploying resources to the cloud using Azure Resource Manager is the Deploy to Azure button found in many GitHub repositories,

In addition to using Resource Manager for IaC, you can bring your existing skills and tools such as Ansible, Chef, and Terraform to provision and manage Azure infrastructure directly.

Azure Blueprints

It's easy to use Azure Resource Manager templates, resource groups, user identities, and access rights and policies to design and create a complete infrastructure. But how do you keep all of these things together? And how do you keep track of which environments each piece of infrastructure has been deployed to and which version of the artifact is deployed now?

Organize all your infrastructure artifacts with Azure Blueprints. Azure Blueprints provides a mechanism that allows you to create and update artifacts, assign them to environments, and define versions. You can store and manage these artifacts as well as manage their versions and relate them to environments.

This will help you organize your infrastructure and create a context for Azure Resource Manager templates, user identities, resource groups, and policies.

```
// Get started by defining and
   assigning an Azure Blueprint
   in the Azure portal.
```

Containers in Azure

"Containerization" is one of those technology buzzwords flying around in the news. But containers are more than just buzz—they're actually very useful for running your applications. A container is basically a lightweight VM that starts and stops much faster than a traditional VM and is therefore more useful for development, testing, and running applications in production.

The major benefit of containers is that an individual container is always the same. You run a container locally when you develop your app, and then use the same container configuration in the cloud and everywhere else. Your entire team uses the exact same container configuration, so you know that the infrastructure is the same for everybody as it is 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, including Docker. Azure can run and manage containers with Azure Container Instances and Azure Kubernetes Service. You can also run containers in Web App for Containers and in Azure Batch. Table 6-1 shows which service you might choose for various scenarios when using containers.

Table 6-1

	Azure Kubernetes Service	Azure Container Instances	Web App for Containers	Containers on Azure Batch
For production deployments of complex systems (with a container orchestrator)	●			
For running simple configurations (possibly without orchestrator)		●	●	
For long-running workloads on containers	●			●
For short-running workloads on containers		●		●
For orchestrating a system based on containers	●			
Orchestrating with open-source orchestrators Kubernetes	●			
Orchestrating with built-in orchestrator				
Using App Service features like deployment slots			●	

Azure Stack

If you need your applications and data to remain on-premises but 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's Azure in a box.

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

You can run things like Azure App Service and Virtual Machines on Azure Stack. Everything is exactly the same as in the public cloud, except that you're running it on-premises. If you decide to move to the public cloud, you can simply push services from Azure Stack to Azure.

Example: Azure Stack

To help with cruise tasks, a company that offers luxury cruise ship holidays has built various software, including a cabin management application and a passenger management application. The entire cruise ship relies on these applications. In the past, the applications were running on servers carried aboard the cruise ships. The company was forced to do it this way because the cruise ships didn't have a connection to the internet for the whole journey.

The company found that running its applications on-premises was cumbersome, as it had to maintain VMs and operating systems and deal with significant availability problems.

The company now runs its applications on Azure Stack, which runs aboard the cruise ships. Azure Stack provides the same services as Azure, so application deployment and management became much easier. The company also uses Azure App Service to run its applications, which allows it to focus on the applications rather than on maintaining VMs and operating systems. Even better, users enjoy the higher availability that's part of Azure, and therefore, part of Azure Stack.

Where to deploy, and when?

If you want to deploy IaaS-based services (in which you control the OS), consider these options:

- **On-premises or anywhere else** (like your local PC or another cloud), you can use:
  - Azure Stack (where you deploy services like VMs)
  - Any of the Azure container services (as containers can run anywhere)
- **In the public Azure cloud,** you can use:
  - Containers (as containers also run in any of the Azure container services)

If you want to deploy PaaS-based services (where you have less control, but the platform does the heavy lifting), consider these options:

- **On-premises or anywhere else** (like your local PC or another cloud), you can use:
  - Azure Stack (as you can deploy PaaS services like App Service in Azure Stack)
- **In the public Azure cloud,** you can use:
  - Any Azure PaaS service that you script as a Resource Manager template

Further reading

Learn more about deploying your applications to Azure and reducing costs in these free e-books:

```
// Cloud Migration Essentials
```

```
// Making the Most of the
   Cloud Everywhere
```

```
// Effective DevOps
```

```
// Azure for Architects
```

07 /

Share your code, track work, and ship software

How can Azure help you plan smarter, collaborate better, and ship your apps faster?

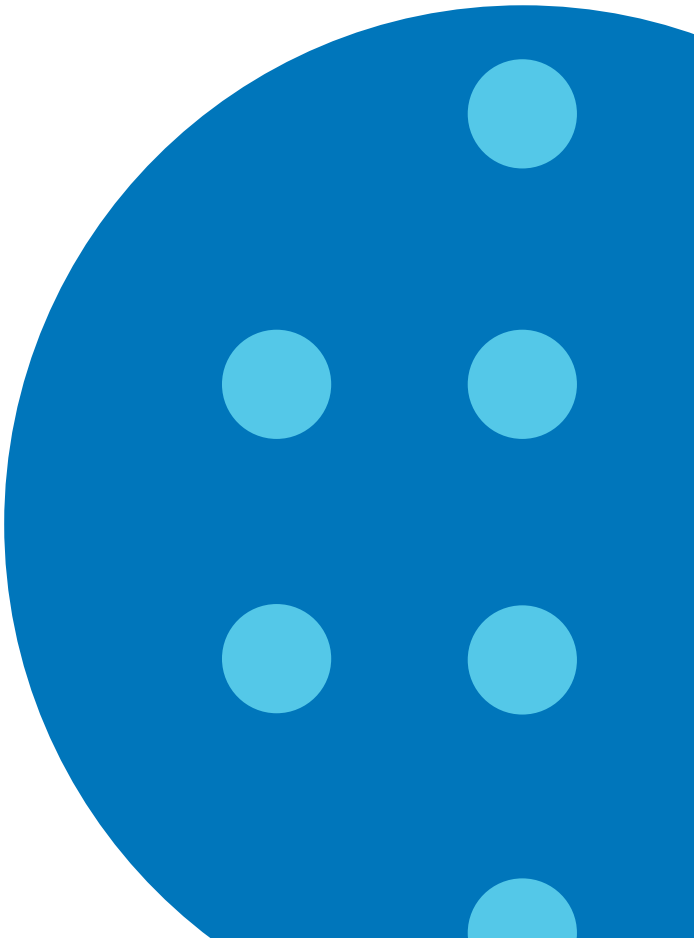
You've spent weekends or nights deploying new versions of your applications. If so, you've probably also spent a lot of time trying to fix the bugs that keep users away from that new version. There's a better way.

Azure DevOps is a set of solutions that can help automate your builds and deployments and automatically test your code and apps before launch.

To help you build, deploy, test, and track your code and applications, Azure DevOps includes:

- Azure Boards
  Use Azure Boards to plan, track, and discuss work across teams.
- Azure Repos
  Use Azure Repos to collaborate on code development with free Git public and private repositories, pull requests, and code review.
- Azure Pipelines
  Use Azure Pipelines to create build and release pipelines that automate builds and deployments.
- Azure Test Plans
  Use Azure Test Plans to improve your overall code quality with manual and exploratory testing services for your apps.
- Azure Artifacts
  Use Azure Artifacts to share code packages (like npm, NuGet, and Maven packages) across your organization.

Let's explore the Azure DevOps services in more detail.



Azure Boards

Planning your work and tracking your progress are important tasks—and Azure Boards can help you complete them.

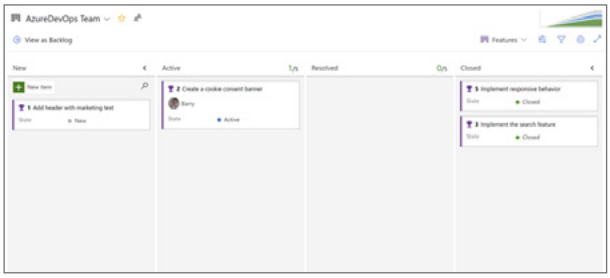


Figure 7-1

In Azure Boards, you can create a complete backlog of work items (like user stories) and plan them in sprints so your team can work iteratively to finish the tasks.

The whole planning system is optimized for working in an agile way. It even includes Kanban boards for managing your work your work (Figure 7-1).

Everything can be customized to work best for your teams, whether using scrum, another agile method, or the Capability Maturity Model Integration (CMMI) process. You can create and manage tasks, features, user stories, bugs, requirements, issues, change requests, and more.

Try customizing your boards and creating charts (like burndown charts or task lists) that show the information you need. You can query work items and progress, and then use these to customize your boards, charts, and lists. From there, share them or pin them on your Azure DevOps dashboard for everyone to see.

```
// Try it out: Start using Azure
   Boards to track issues, tasks,
   and epics
```

Azure Repos

Version control is essential for working together and ensuring that your most important asset—your code—is safely stored. Azure Repos is a set of version control tools for storing your code and sharing it with your team. This is useful both for teams and individual developers. Version control keeps a history of your development so you can review or even roll back to any version of your code.

Choose from the following two systems of version control when using Azure Repos:

Git

This is a widely used version control system among developers and is also the basis for GitHub. Git is a distributed version control system, meaning the complete source code (all versions of all files) is on your machine—which makes it easy to work offline. With Git, the source of truth is essentially on everyone's machine and is synchronized when developers push their code to the Git server (in this case, Azure Repos).

Azure Repos uses standard Git. This means that you can use it with any Git tool and IDE, including Visual Studio and Visual Studio Code as well as Git for Windows, Mac, Eclipse, and IntelliJ.

When you follow the Git workflow, you usually begin by creating your own branch of the code to, for instance, add a feature. Once you finish this, you commit your code to create a pull request for that branch and submit it to the server. Users can see, review, test, and discuss this pull request. Once it's good enough to be pulled into the main branch, the request is accepted, and your development branch can be deleted.

With Azure Repos, you have a rich toolset to support the Git workflow. You can link work items like user stories or bugs to pull requests so you know what each change is about. You can have discussions about committed code and even comment on changes within the code. Azure Repos also enables voting on changes in the code, so a change only gets accepted once everyone on the team agrees to it.

Azure Repos offers free and unlimited private Git repositories.

```
// Get started by learning
   how to code with Git
```

Team Foundation Version Control

Team Foundation Version Control (TFVC) is a centralized version control system that ensures one source of truth is always kept on the server. Developers usually have only one version of each file on their machine, which makes it more difficult to work offline.

With TFVC, you can choose to work with the following workspaces:

Server workspaces: Developers publicly check out files from the server so that only they can make changes to that file. Once they are done, they can check the changes back in and other developers can check out the file to make changes. This eliminates the need to merge changes and removes the possibility of code conflicts.

Local workspaces: Using these, developers each have the latest version of the files on their machine and can change each of them. Once they are done making changes, they check in the changes to the server and resolve conflicts as necessary.

With TFVC on Azure Repos, everyone can download the versions of code branches you create on the server. Azure Repos also provides a rich toolset that allows you to attach work items to code changes. It's also possible to request and perform code reviews, so your team can discuss changes and recommend updates before they're merged into the main branch.

```
// Try it out: Start developing
   and sharing your code in TFVC
   using Visual Studio
```

Azure Pipelines

Once your code is in a repository like Azure Repos, you can start to automate your build and release processes with Azure Pipelines.

Azure Pipelines provides a lot of value in a small amount of time. It enables continuous integration (CI) for compiling and testing code when changes come in, as well as continuous deployment (CD) for deploying apps after changes are compiled and tested successfully. We encourage every organization to explore CI and CD, as these processes improve code quality and reduce deployment efforts.

Azure Pipelines can help with CI and CD by offering build and deployment pipelines. Each contains steps to compile and test your code and deploy it to one or more environments. The beauty of Azure Pipelines is that it works with any type of code, no matter where you store it—from C# on Azure Repos to Java on BitBucket and anything else.

Azure Pipelines works very well with Azure services to deploy your application in an Azure web app, for instance. It also works with any service that runs in any other environment, such as Google Cloud, Amazon, or even on-premises in your own datacenter. If you're already using continuous integration tools like Jenkins or Spinnaker, you can easily bring your existing builds and pipelines to Azure and take advantage of dynamic agent plug-ins to reduce infrastructure requirements and costs.

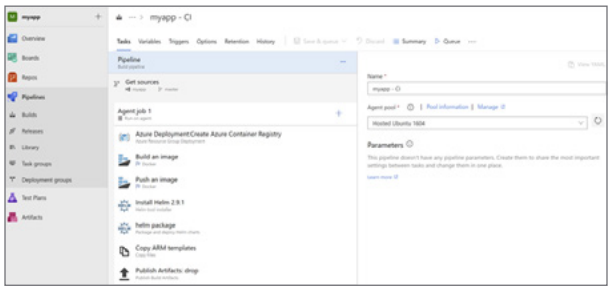


Figure 7-2: Azure Pipelines: build pipeline

There are two ways of working with Azure Pipelines. You can create pipelines using the visual designer in the Azure DevOps portal, or you can use the more advanced YAML-based approach. In this approach, you create a YAML code file, which contains all the steps of the pipeline, and commit that to source control.

The easiest way to get started is to use the visual designer. Let's take a look at an example of a build and release pipeline:

Build pipeline: The build pipeline (or CI pipeline) in Figure 7-2 shows a list of tasks that will be executed when this pipeline runs. The pipeline is configured to run as soon as new changes are committed to Azure Repos. It will take the code in Azure Repos (which is a Node.js app), build a Docker container

image from it, and push that to Azure Container Registry. From there, Helm can use it to compile the image into a package that can be deployed on Azure Kubernetes Service.

Note that you can configure which hosts run your pipelines for you. On the right side of the image, you can see that this particular pipeline will run on a hosted pool of Ubuntu machines. There are also Linux and Windows hosts available, and even a hosted MacOS that you can use to build your iOS apps. This is just one example of a build pipeline. It's possible to create one for every imaginable application. You can also integrate tests, including unit tests and static code tests, into the build pipeline.

Release pipeline: The release pipeline (or CD pipeline) executes as soon as the build pipeline runs successfully, though you can also configure it to be triggered manually. The release pipeline in Figure 7-3 contains nine tasks that first create an Azure Kubernetes Service cluster and then deploy the Helm package that was produced in the build pipeline to the cluster.

Figure 7-3 shows the steps in the development stage. Stages are like environments. You can configure a stage for your development, test, and production environments and so on. You can also configure things like predeployment approvals, which require someone to approve the release of an application into a specific environment (such as the production environment). This means you can automate everything and leave the decision to release into production up to a manager based on test results for previous steps in the pipeline.

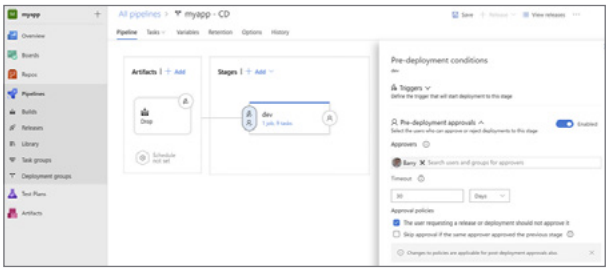


Figure 7-3

Make your pipelines as simple or complex as you want. Ideally, you want to automate as much as you can, from the creation and destruction of your infrastructure to the deployment and testing of your application. Pipeline tasks are available for almost everything, and you can access more tasks as extensions to Azure DevOps in the Visual Studio marketplace.

```
// Get started with Azure
   Pipelines by creating your
   first pipeline
```

Azure Test Plans

To improve the quality of your applications, use Azure Test Plans to define test plans and then create and execute manual and exploratory tests. Azure Test Plans provide the tooling to author tests, execute them, record feedback, and track the test results.

In Azure Test Plans, start by creating a test plan. This contains multiple test suites and test cases. A test case can be an exploratory test, in which the application is explored to see if it works as expected; a guided manual test, in which test steps and expected outcomes are described in detail; or an automated test. You can even record test steps by recording clicks in an application and letting Azure Test Plans automate those clicks into a test. You can also incorporate stress and load tests into your build and release pipelines. Test cases are work items, just like user stories and tasks, and can therefore be scheduled within an iteration.

Once you've created a test (Figure 7-4), a tester can run it. In a manual test, for example, the tester uses the test tool to run through the test steps and record findings, including the screen, the tester's voice, screenshots, and attachments. The tester passes or fails each step of the test.

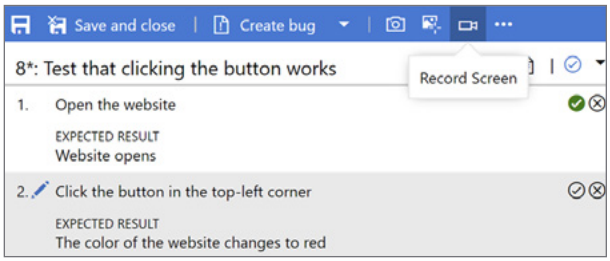


Figure 7-4

From the context of the test, the tester can also create a bug that needs to be solved.

To ensure stakeholders' expectations are in line with your plan, Azure Test Plans also enable you to request feedback for work items like user stories. This enables stakeholders to take a look at what you propose and provide feedback in the form of text, attachments, video, or voice.

Perhaps most importantly, Azure Test Plans provide dashboards and charts on the progress and status of the tests in your project. You can use these to see what the quality of your application is and how it progresses over time. This can help you identify features that aren't ready to be deployed.

```
// Get started with Azure Test
   Plans by creating manual
   test cases
```

Azure Artifacts

Because packages offer functionality that you don't have to build yourself, you probably use a lot of them in your applications. And you likely access them from just as many sources: NuGet, npm, Maven, and more. But what if your team creates packages that you want to only use internally? Where do you host them securely, and how do you share them? Azure Artifacts provides this capability. Azure Artifacts is a package feed that allows you to host packages that you create and secure them for your organization.

You can host all sorts of packages on Azure Artifacts, including NuGet, npm, Maven, Python, and Universal Packages. You can even use the Azure Artifacts feed to store packages from public sources, like nuget.org and npmjs.com. When you store packages from public sources on your feed, you'll be able to keep using them even if they're no longer available on the public feed. This is especially useful for mission-critical packages.

Follow these simple steps to use Azure Artifacts:

1. Create an Azure Artifacts feed.
2. Publish your package to the feed.
3. Consume the feed in your favorite IDE, such as Visual Studio.

```
// Try it now: Get started
   with Python packages in
   Azure Artifacts
```

Further reading

If you want to improve the quality of your software and learn more about automating your build and release processes, download and read these free e-books:

```
// Effective DevOps
```

```
// Continuous Delivery in Java
```

```
// Azure for Architects
```

08 /

Azure in action

Walk-through #1: The Azure portal experience

One of the most important Azure tools is the central hub—the Azure portal. Most things you can do in the Azure portal can also be done through the Azure API, the Azure CLI, and Azure PowerShell.

The Azure portal is a dashboard with tiles. It's easy to create and customize dashboards, and then share them with team members.

Tiles in the Azure portal

Tiles, shown in Figure 8-1, display information for a service or act as a shortcut to a service. They appear throughout the portal in the pages of all the services. They're a useful way to get a quick overview of how a service is doing.

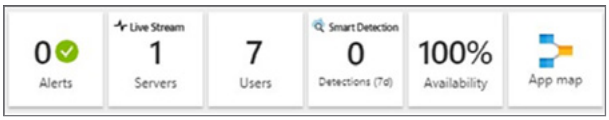


Figure 8-1

The Application Insights service tiles show information like active alerts, live data coming in, active users in the past 24 hours, and availability. You can customize a tile's size and information as well as the appearance of charts by adjusting timelines and displaying data in different formats, such as lines or bars. You can also pin tiles directly to your dashboards so that they're the first thing you see when you enter the portal (Figure 8-2). You can, for instance, pin tiles from the service metrics you use to create a monitoring dashboard to share with your team or display on a physical monitor.

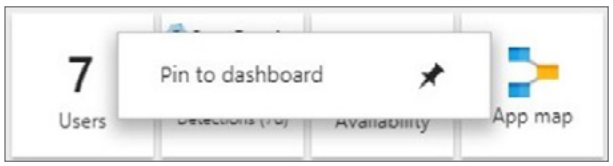


Figure 8-2

Adding services

You can find and add services in the Azure portal in several ways.

To create new services, select the plus sign in the upper-left corner of the portal window. This opens the search box for the marketplace, where you'll find everything from web apps to Linux servers, as shown in Figure 8-3.

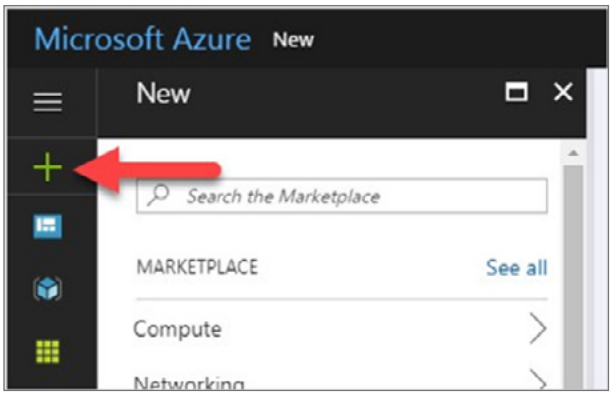


Figure 8-3

When you find the service you want from the search results, as shown in Figure 8-4, a wizard takes you through configuring and deploying it.

.

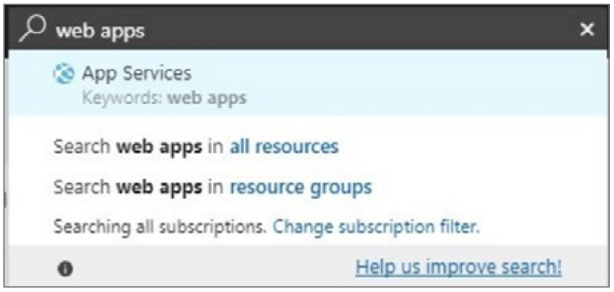


Figure 8-4

You can use the search box at the top of the portal to search through all your resources and go directly to them (Figure 8-5). The favorites menu is in the pane on the left side of the portal.

This menu displays the resource categories, such as Azure App Service, represented by their icons. You can rearrange the icons by dragging them up and down. You can also select which ones you want to see by expanding the favorites menu and selecting the star symbol next to those categories.

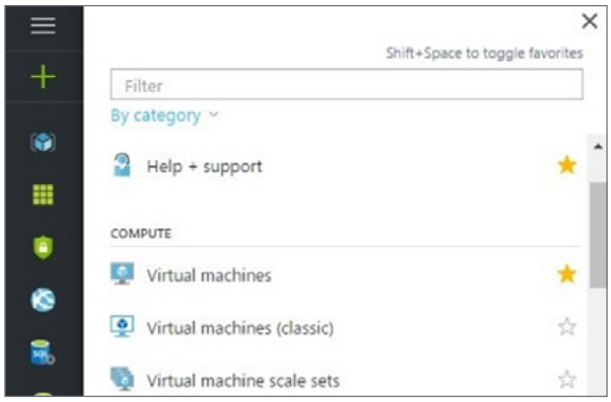


Figure 8-5

Understanding blades

Pages in Azure are also called blades, and you can pin them to your dashboards. When you open a web app, you first see the Overview blade, as shown in Figure 8-6.

This blade provides tools to stop, start, and restart the web app and display 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. For example, if you open the Deployment Slots blade and select Create New Deployment Slot, a new blade appears to the right of the Deployment Slots blade, preserving the context you're in.

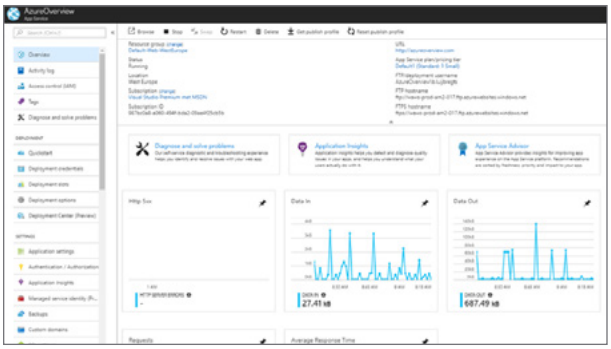


Figure 8-6

Creating a new VM

Let's use the Azure portal to create a new VM. Once we've done so, we'll shut it down and remove it so that you don't continue to pay for it.

A word about resource groups

The VM will be deployed in a resource group, a logical container that holds resources. All Azure resources reside inside resource groups. You can manage the security of a resource group as well as see what the resources in the group cost. It's common practice to bundle related services in a resource group so that they're easier to secure.

1. In the Azure portal, in the upper-left corner, select **Create A New Service**.
2. In the search box, type **Windows Server virtual machine.**
3. Click **Windows Server 2016 Datacenter**.
4. Click **Create**. The Create Virtual Machine Wizard opens.
5. Choose a name for the VM.
6. Choose the disk type. SSD provides a faster VM but is more expensive. For this walk-through, choose **SSD**.
7. Type a username.
8. Select **Password** for the authentication type.
9. Type a password and confirm.
10. In the **Resource Group** box, type a new name.

11. Choose the location of the VM, and then click **OK**.

12. Choose the VM size. There are many sizing options for VMs. VM performance determines the cost. Use the wizard to select how many cores and how much memory you want, and choose options based on that. In addition, there are other features that come with size options, such as:

    - Type of hard drive (SSD or normal HDD).
    - The amount of max input/output operations per second (IOPS). This determines the performance of the VM in a significant way, especially if your applications read and write extensively from and to the hard drive.
    - The amount of data drives that can be installed in the VM.
    - The ability to perform load balancing.
    - The graphics card installed in the VM. This is useful if you need to execute substantial graphics rendering or a heavy computational workload.

13. After you select the size, you can configure additional settings like the virtual network, IP address, and extensions on the machine. For now, leave everything as is and select **OK**.

14. Review the summary, agree to the terms, and then click **Create**.

It usually takes just a few minutes for the VM to be deployed. When you navigate to the VM in the Azure portal, you can configure it further and log in using Remote Desktop Protocol (RDP).

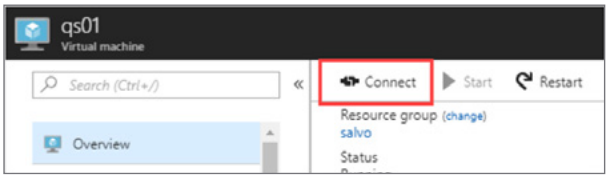


Figure 8-7

To log in to the VM using RDP, click **Connect** in the VM's **Overview** blade in the Azure portal (Figure 8-7). This triggers a download of the RDP file you can use to connect to the VM.

Cleaning up the walk-through resources

When you're finished with the VM, shut it down and remove it by deleting the resource group that we created when we generated the VM. This contains the VM and all other resources that are automatically created. Once 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 web app and database on Azure

In this walk-through, we'll deploy a simple .NET Core application that connects with SQL database. Then, we'll host it in Web Apps.

To follow along, you'll need Git v2 or higher, .NET Core, and Visual Studio Code installed on your device. We'll also use a sample ASP.NET Core MVC application to manage a to-do list.

Creating a web app and database using the Azure portal

To host the .NET Core application, we'll create a new web app in the Azure portal.

1. In the Azure portal, select **Create A New Service**.
2. Search for **Web App**. The **Web App** blade opens. Select **Create**. The **Web App Create** blade opens.
3. Type a name for the web app.
4. Create a new resource group by giving it a name.
5. Leave the OS selection as **Windows**.
6. Select or create an **App Service Plan**, and then select **Create**.

Services like Web Apps run on Azure App Service Plans. App Service Plans are 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 geographic region that you choose. You can, for instance, run your Web Apps application in an App Service Plan of pricing tier S1, which has 1 core and 1.75 GB RAM, as shown in Figure 8-8.

Figure 8-8

You can run as many App Services on an App Service Plan as you want, but note that you need to share resources among all the App Services.

To host the database, we'll create a SQL database. This works the same as a local SQL Server database and now runs fully managed in Azure.

1. In the Azure portal, click **Create A New Service**.
2. Search for **SQL Database** and click it to open the **SQL Database** blade. Click **Create**. The **Create SQL Database** blade opens.
3. Type a database name.
4. Select the resource group that you created for the web app.
5. Leave the source as **Blank database**.
6. Click **Server** to create a new SQL database server.
    a. Type a name for the server.
    b. Type the server admin login. This is the username for the server.
    c. Type the password that you'll use to log on to the server.
    d. Confirm the password.
    e. Choose a location. Choose the same location that you selected for the App Service Plan.
    f. Click **Select** to submit the new server configuration.
7. Select a pricing tier. For development and test purposes, the **Basic** tier is sufficient.
8. Click **Create**. The database will now be created.
9. Navigate to the SQL database and click **Show Database Connection String**.
10. Make note of the connection string because you'll need it later in this tutorial.

Running the .NET Core app locally

Let's run the app locally before we run it in Azure. The app can run locally because by default, it uses a SQLite database, which is a self-contained SQL database engine.

1. Open a command prompt and navigate to a directory you want to use as your source code directory for this project.
2. Run the following commands to get the source code and navigate to the project folder:

```
git clone https://github.com/azure-samples/dotnetcore-sqldb-tutorial

cd dotnetcore-sqldb-tutorial
```

3. The project uses Entity Framework Core to populate its database. To ensure the database is up to date and to run the application locally, execute the following commands:

```
dotnet restore

dotnet ef database update

dotnet run
```

4. The app should now be running, and the URL to the app (such as **http://localhost:5000**) should be in the output in the command window

5. Navigate to that URL in a browser. This will load the application, which will look like that shown in Figure 8-9. Now you can create new to-do items by selecting the **Create New** link.

6. Close the application by closing the command window or pressing **Ctrl+C**.

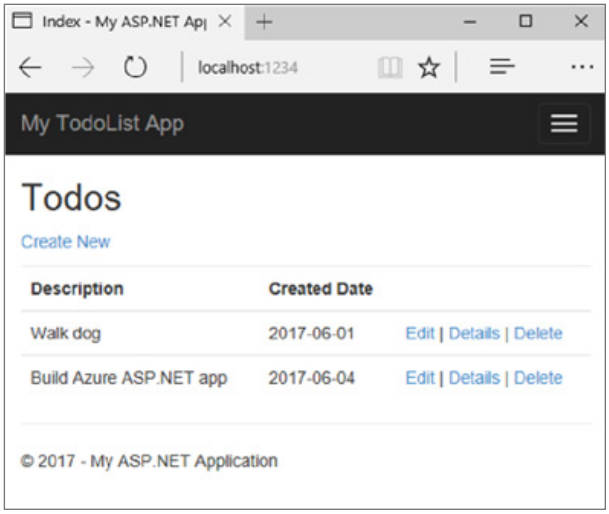


Figure 8-9

Connecting the local web app to the database running in Azure

You now have a working application running locally. Before we deploy it to Azure, we'll change the source code so that it can connect to the SQL database.

1. In your local source code repository, find the **Startup.cs** file and locate the following code:

```
servicesAddDbContext<MyDatabaseContext>
(options => options.UseSqlite
("Data Source=localdatabase.db"));
```

2. Replace the code with the following code, which will connect to the Azure SQL database:

```
// Use SQL Database if in Azure, otherwise,
use SQLite

if(Environment.
GetEnvironmentVariable("ASPNETCORE_
ENVIRONMENT") == "Production")

services.
AddDbContext<MyDatabaseContext>(options =>

options.UseSqlServer(Configuration.
GetConnectionString("MyDbConnection")));
else

services.
AddDbContext<MyDatabaseContext>(options =>

options.UseSqlite("Data
Source=localdatabase.db"));

// Automatically perform database migration
services.BuildServiceProvider().
GetService<MyDatabaseContext>().Database.
Migrate();
```

This code looks at the environment in which it's running and changes its database connection based on that information. When running in the production environment (Azure, in this case), the code will get the connection string for the database from the MyDbConnection variable, which we'll configure in Azure.

The code also runs the Database.Migrate() method, which executes the Entity Framework Core migrations that we previously ran manually.

3. Save your changes and run the following commands to commit the changes to your local Git repository:

```
git add .
git commit -m "connect to SQLDB in Azure"
```

Now we'll configure the connection string variable in Azure.

4. In the Azure portal, navigate to the web app that we created earlier.
5. Navigate to **Application settings**.
6. Create a new connection string named **MyDbConnection**. The value should be the connection string to the SQL database (including username and password) you saved earlier when you created the database.
7. Click **Save**. The application settings in the Azure portal should look like those shown in Figure 8-10.

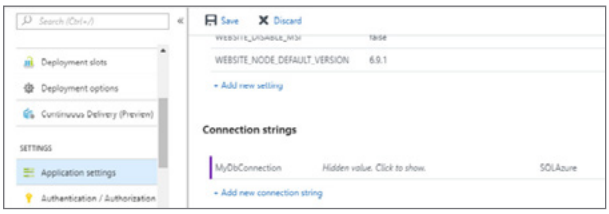
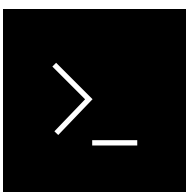


Figure 8-10

Deploying the web app to Azure

We'll use Git to push the application to Azure. To connect the local Git repository to Azure, you must have a deployment user configured on the server (Azure Web App) to authenticate your deployment. The deployment user is account level and is different from your Azure subscription account. You need to configure this deployment user only once.

1. In the Azure portal, navigate to the **Azure Cloud Shell** by selecting the button in the top bar that looks like this:



2. The Azure Cloud Shell enables you to use the Azure CLI in the cloud and manages authentication. When the Cloud Shell is fully loaded, run the following command to create the deployment user. Replace the **<username>** and **<password>** values with ones you create. Make note of the username and password because you'll need them later.

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

3. The command results in a JSON output. If you receive a **'Conflict'. Details: 409** error message, change the username. If you receive a **'Bad Request'. Details: 400** error message, create a stronger password.

   Now we'll push the source code from the local Git repository to the Azure web app.

4. Open the command prompt on your local machine.

5. Add an Azure remote to your local Git repository by using the remote Git URL:

   a. Replace **<username>** with the username you used to create the deployment user.

   b. Replace **<app_name>** with the name of the Azure web app.

   c. Use the URL to run the following command:

```
git remote add azure <deploymentLocalGitUrl>
```

6. Once the remote target is added to the Git repository, you can push your code to it by running the following command. You'll need to enter credentials to be able to push code to Azure. Use the username and password you used to create the deployment user.

```
git push azure master
```

   Pushing the source code to Azure might take a few minutes the first time. Once complete, navigate to the URL of your Azure web app, which will look like this: **http://<app_ name>.azurewebsites.net**

7. Add some to-do items in the application to test its connection to the database.

   Now you have a working application running in Azure.

Walk-through #3: Extending applications with Logic Apps and Cognitive Services

A powerful feature of our application is its ability to analyze the content of to-do items and then automatically create calendar appointments for tasks that include a specific date.

For example, if a user creates a to-do item with the text "family dinner next Friday at 7:00 PM," the application will create a calendar item for that specific Friday at 7:00 PM with the subject "family dinner."

We'll set this up using the Logic Apps feature of Microsoft Azure App Service and the Language Understanding Intelligent Service (LUIS), as follows:

- The .NET Core app writes the to-do item in the SQL database.
- The logic app is triggered by every new row created in the database.
- The logic app takes the to-do item text and passes it to the Language Understanding service.
- The Language Understanding service analyzes the text and creates a calendar item in your Office 365 calendar if the text contains a date and time.

We don't have to change our application to add this functionality. Logic Apps and Cognitive Services are additional services that simply analyze the data that's already there.

Let's get started.

Creating the Language Understanding service

We'll first create the Language Understanding service so that we can use it later in our logic app. We'll keep our model for this example simple—we won't build it out, so it'll be ready for every variation users might need for a date in a to-do item. You can add to the model yourself instead of using the one we create.

1. In the Azure portal, select **Create A New Service**.
2. Search for **Language Understanding** and select it in the search results to open the **Language Understanding** blade. Select **Create**. The **Create Language Understanding** blade opens.
3. Type a name.
4. Select a pricing tier (any will do for this walk-through).
5. Create a new resource group called **datedetection**.
6. Click **Create**.
7. Navigate to the Language Understanding service once it's created.
8. By default, the service opens to the **Quick Start** blade. From here, select **Language Understanding Portal**.
9. If needed, sign in using **Sign In** at the top-right corner.
10. Click **Create new app**.
11. Type a name.
12. Click **Done**.

We're now in the Language Understanding portal and can build a language model. We want the Language Understanding service to understand the phrase "family dinner next Friday at 7 PM." To do that, we'll first add some entities, which are items in the text the service will recognize.

1. Click **Entities**.
2. Click **Manage Prebuilt Entities**.
3. Select **Datetimev2** and **keyPhrase**.
4. Click **Done**. We now have two entities that will recognize text for us.
5. Click **Intents**.
6. Click Create new Intent.
7. Type a name like "Add to-do calendar item," which is the intent we want to detect in the text.
8. Click **Done**.

Now you can enter **utterances**. These are sample texts that represent the intent we want to detect.

9. Enter "family dinner next Friday at 7 PM" to represent the intent of adding a to-do item to the calendar.

10. Because we've already added two entities, the text in the utterance is analyzed and recognized as these entities, as shown in Figure 8-11. The text "family dinner" is recognized as a **keyPhrase**. The text "next Friday at 7 PM" is recognized as **datetimev2**.

11. Let's use this model to train the service and publish it. Select **Train** in the upper-right corner of the screen.

Figure 8-11

- This performs machine learning training and builds a machine learning model based on what we've just entered.
- To test if the service works as expected, type "family dinner next Friday at 7 PM" in the **Test** window next to the **Train** button.

12. Now that we have a working service, we need to publish this model to production. Click **Publish** in the menu (next to the **Train** button) to bring up the **Publish** page.

13. Leave the slot as **Production**.

14. Click **Publish**.

15. The model is now published to production. Scroll down to **Resources and Keys** and make note of the key string you'll find there because we'll need it for our logic app.

Creating the logic app

The logic app we create will be triggered by the new rows of to-do items written in SQL Database. It will then take the value of each to-do item and send it to the Language Understanding service to be analyzed. If the Language Understanding service finds a date in the item, it will create a new calendar event in your Office 365 account.

Let's create the logic app:

1. In the Azure portal, click **Create A New Service**.

2. Search for **Logic App** and click it in the search results to open the **Logic App** blade. Click **Create**. The **Create Logic App** blade opens.

3. Type a name.

4. Select the resource group that you created for the Language Understanding service.

5. Choose a location.

6. Click **Create**.

7. When the logic app is created, you'll see a quick start page that asks if you want to start the logic app from a template (Figure 8-12). Choose **Blank Logic App**.

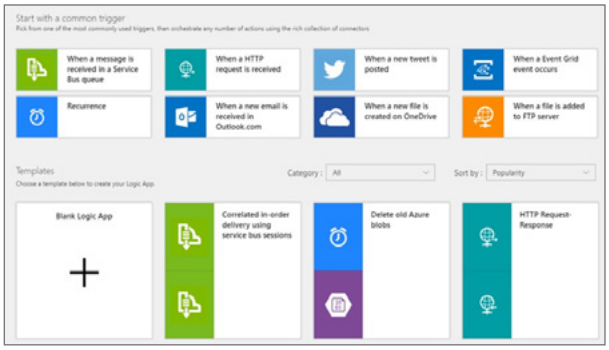


Figure 8-12

We now need to create a trigger for the logic app.

1. Search for SQL.
2. Select the **When an item is created** task. This will ask for the connection to the SQL database (Figure 8-13).

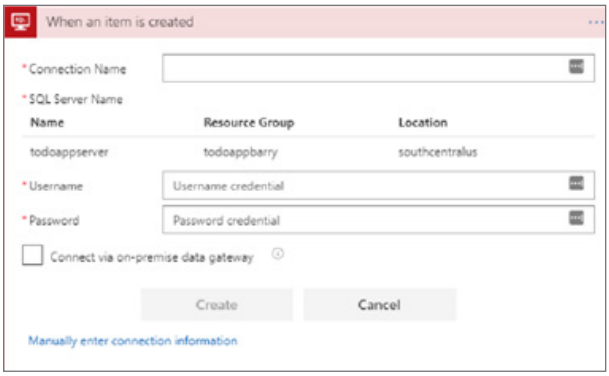


Figure 8-13

3. In this case, the correct SQL Server is already selected because there's only one. You might have to select the appropriate server.
4. Enter a name for the connection, and then type the username and password of the SQL database that we created earlier.
5. Click **Create**. This creates the connection and saves it in your Azure subscription. You can reuse this connection in other logic apps.
6. Select the table that we want to monitor—the **To-do** table.
7. Select an interval and a frequency. Some logic app triggers need to poll to be triggered, whereas others have their information pushed to them.

Now the logic app will be triggered every time we enter a new to-do item.

Next, we'll add another action for the logic app.

1. Click the **plus sign** under the SQL task, and then select **Add an action** to add the next action (Figure 8-14).

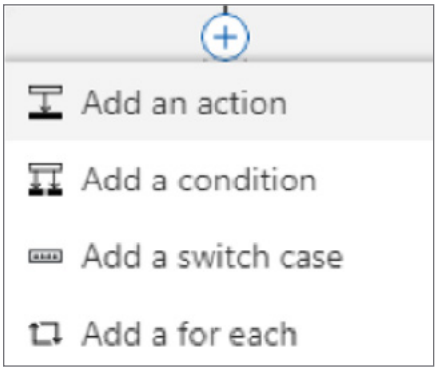


Figure 8-14

2. Search for **LUIS**, which will bring up the Language Understanding service. Select the **LUIS Get prediction action**. It will ask for a connection to a Language Understanding service.

3. Type a name for the connection.
4. Paste in the connection key you saved when we published the Language Understanding model.
5. Click **Create**.
6. Select the App ID that you created in the Language Understanding portal.
7. Select the description from the SQL task as the input for the **Utterance** field.
8. Select the **Add to-do calendar** item for the desired intent. This will output whether the task contains a date.
9. Click the **plus sign**, and then click **Add a condition**. We'll test whether the text contains a date by checking if the desired intent was true. If the text does contain a date, we'll create a calendar event. If it doesn't, we won't do anything.
10. In the condition, select the **Is Desired Intent value** from the Language Understanding task for the value.
11. Leave the **is equal to** statement as is.
12. Add **true** in the value textbox.
13. The condition appears in both the **if true** and **if false** boxes. In the **if true** box, create a new action.
14. Search for LUIS as we did earlier.
15. Select the **Get entity by type** action. This is a Language Understanding action that extracts an entity based on its type from the Language Understanding results.
16. Select the App ID as we did earlier.
17. Select **builtin.datetimev2** for the desired entity.
18. Select the **LUIS Prediction** object for the **luisPredictionObject** field.
19. Below this action, add another one for **Get entity by type**.
20. Select the App ID.
21. Select **builtin.keyPhrase** for the desired entity.
22. Select the **LUIS Prediction** object for the **luisPredictionObject** field.
23. Create a new action while still in the **if true** box. Search for **Office 365** and then select the **Create Event V2** action. This can create an event in your Office 365 calendar.
24. Note that this requires a connection to Office 365. Click **add new connection** and log in with your Office 365 credentials. The logic app will keep your connection in your Azure subscription.
25. Select the calendar to create the event.
26. In the **End Time** and **Start Time** fields, select the **Entity Value** from the action where you filter the **datetimev2** entity.
27. In the **Subject** field, select the **Entity Value** from the action where you filter the **keyPhrase** entity.
28. Save the logic app flow. The **if yes** box should look similar to that shown in Figure 8-15.
29. Navigate to the to-do app URL, which is the URL of your web app from the previous walk-through.

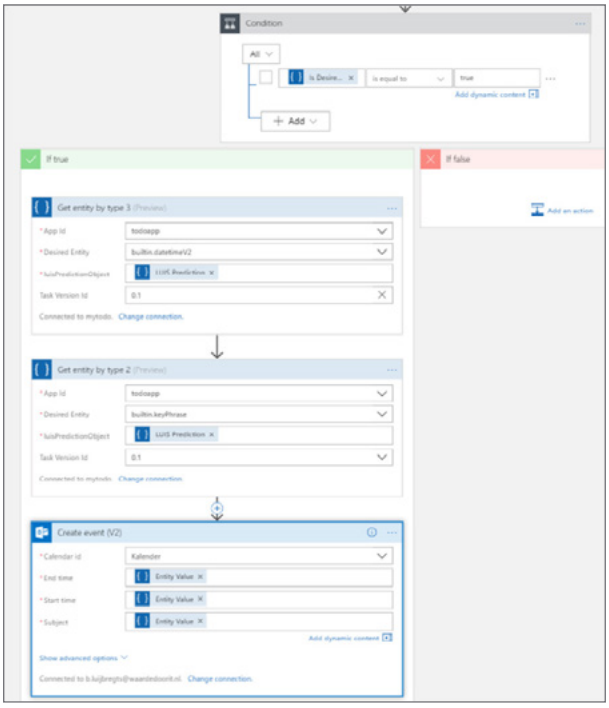


Figure 8-15

30. Create a new to-do item with the text "family dinner next Friday at 7 PM." This should create an event in your calendar.

In addition to checking your calendar, you can see how the logic app ran by reviewing the **Runs History.** You can access the Runs History when you open the logic app from the Azure portal. From the Runs History, you can resubmit the value to run it again through the logic app.

This example shows you can extend an application with Azure services just through configuration, without changing the code.

We've kept this example simple so it's easy to follow in this walk-through. In a real-world scenario, the Language Understanding model should be more robust to be able to understand more utterances. In addition, you could have the logic app trigger on edits of to-do items, not only on their creation.

Walk-through #4: Ready for production

With your application running, you can now use Azure to make it more robust and easier to update.

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 alone, but if you work on a team, you'll need another type of source control, like Azure DevOps Repos or GitHub.

We'll use GitHub to push our code and then link that to our web app so changes are deployed automatically in a continuous delivery pipeline.

Let's get started.

1. Log in from https://github.com/new to create a new repository in GitHub.
2. Type a name for the repository.
3. Leave the other settings as they are (public repository, don't create a README).
4. Create the repository, which should look similar to that shown in Figure 8-16.

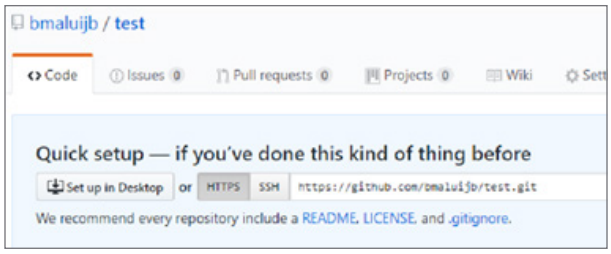


Figure 8-16

5. Use the URL that appears on the screen to set a remote destination for the local Git repository. You can do this in the command window.
6. Use the cd command to change to the directory of the application source code.
7. Run the following command:

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

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

```
git push github
```

With that, the code is in GitHub, and you can share it with your team.

Let's now set up continuous delivery using the Deployment Options feature of Web Apps through the Azure portal. Note that we can also use the Continuous feature in Web Apps directly, but that requires an Azure DevOps Services account.

1. In the Azure portal, go to the web app that hosts the .NET Core to-do app.
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, click **Disconnect**.
4. In **Choose Source**, select **GitHub**.
5. In the **Authorization** section, authorize Azure to use GitHub by selecting **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**.
9. Return to the **Deployment Options** menu. You can now see that GitHub 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. This is shown in Figure 8-17, which illustrates the **Deployment Options** blade. You can also force this process by clicking **Sync**.

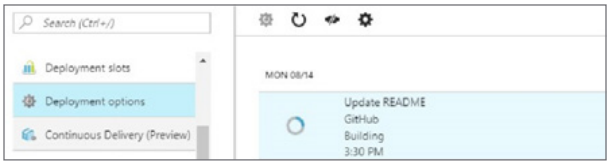


Figure 8-17

Setting up staging environments

Using Azure App Service Web Apps, you can set up a staging slot to test new versions of your application through deployment slots. Deployment slots are App Services with which you can test your code before you promote it to the next slot.

There are deployment slots for staging, load testing, and production, which is always the original App Service—in our example, the .NET Core web app. In fact, you can have as many deployment slots as you want without incurring additional costs.

The deployment slots all run in the same App Service Plan, which is what you pay for. Keep in mind that having additional deployment slots in an App Service Plan will consume resources like CPU and memory.

You create new deployment slots from the **Deployment Slots** menu item in the web app. You need to run 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 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 then swap it with the version in the production slot. This warms up the application before it swaps, resulting in a deployment with no downtime.

Let's see how to create a deployment slot and swap to it.

1. In the Azure portal, go to the web app that hosts the .NET Core app.
2. On the menu bar, click **Deployment Slots**. The **Deployment Slots** blade opens.
3. Click the **plus sign** to create a new deployment slot.
4. Type a name for the slot, for example, "staging."
5. Choose the **.NET Core web app** as the configuration source. This copies the application settings to the new slot.
6. Click **OK** to create the slot, which is similar to the original web app.
7. Set up CD for the slot, just as you did for the web app.
8. Disconnect the CD connection in the original .NET Core web app. This way, when you push new code, it's delivered only into the staging slot.
9. In the .NET Core app, change some text in the **Index.cshtml** file in the **Views/Home** folder.
10. Commit it to Git and push it to GitHub, just like when you deployed the .NET Core 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 .NET Core web app and to the URL of the staging slot, which you can find in the **Overview** blade of the slot.

Now let's put the new version into production.

1. In the Azure portal, go to the .NET Core web app.
2. On the menu bar, select **Deployment Slots** to open the **Deployment Slots** blade.
3. Click **Swap** to open the **Swap** blade. Leave all settings as they are.
4. Click **OK** to initiate the swap.

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

Using diagnostic logs

An efficient way to monitor an app is by using diagnostic logs to see live diagnostic logging from the web app. You can even pipe the logs into the console window. To do this, run the following command in the Azure Cloud Shell:

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

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

Setting up monitoring and alerts

Azure Monitor Application Insights provides another powerful way to track applications. This monitoring tool provides information about your application, such as how many visitors used it, how many exceptions occurred, and where in the code they happened. Unlike diagnostic logs, Application Insights requires a nominal fee.

Let's set up Application Insights.

1. Go to the Azure portal and then to the web app that hosts the .NET Core app.
2. On the menu bar, click **Application Insights**.
3. Select **Create New Resource**.
4. Type a name and select a location for the Application Insights instance.
5. Click **OK**. Application Insights will be deployed and start to collect data for the application.

Now you need to configure your application to send data to Application Insights.

6. In the Visual Studio Code menu, select **Project > Add Application Insights Telemetry...**
7. This opens the Application Insights wizard. Log in with your Azure account.
8. Select an Application Insights pricing plan.
9. Click **Register**. This automatically adds everything you need to the .NET Core project and creates the Application Insights resource in Azure.
10. Build the project and push the changes to GitHub so they're deployed to the web app. When the deployment is complete, the application will send data to Application Insights.

By default, Application Insights performs smart detection. This feature detects when something is wrong—such as a sudden increase in failed requests or when the application is unusually slow—and alerts you. You can also create your own custom events for all sorts of metrics and conditions in the **Alerts** menu of Application Insights, as shown in Figure 8-18.

Figure 8-18

11. To check if Application Insights is working properly, go to the Azure portal, find the Application Insights resource, and select it. You'll see the overview, which shows basic metrics like server response time, page view load time, and number of server requests and failed requests. You should see some data, indicating that Application Insights is working.

Scaling the web app

When you have many users, you need Web Apps to scale up to accommodate increased traffic. When it's not busy, you need it to scale back to save costs. You can do this with the **Automatic Scaling** feature of App Service. You need to run Web Apps in the standard or premium pricing tier to use this feature.

Web Apps has a menu item called **Scale Out,** as shown in Figure 8-19. You can use this to scale out manually or automatically. Scaling out means you add more instances of your application to handle the load.

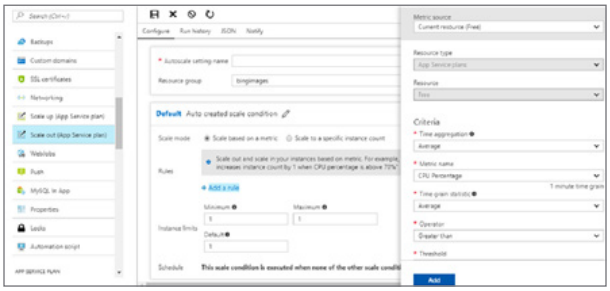


Figure 8-19

Adding Secure Sockets Layer

When an app is ready for production, you need to confirm that it's secure. Besides authentication and authorization, serving the web application over HTTPS is one of the most important security measures you can take. This is because without HTTPS, intruders could see the traffic among your resources and use this information for malicious purposes like signing in to your application. Additionally, HTTPS is a requirement for leading-edge features like service workers.

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

Notifying users about new versions

Your business will benefit from making users aware of new production releases. By extending the continuous integration/continuous delivery (CI/CD) process in Azure builds, you can use a Logic Apps workflow to manage social media communication, like sending out tweets or publishing posts with release notes.

09 /

Summary and resources

In this guide, we introduced the power that Azure can bring to your applications. Using Azure, you can do incredible things with your apps—employ facial and speech recognition, manage your IoT devices in the cloud, scale as much as you want—and pay only for what you use.

You've learned that Azure has services for almost every scenario, so it can help you no matter which programming language you use or what platform you write applications for. We hope you continue to consult this e-book to become better acquainted with the vast range of Azure services and determine which ones best fit your needs.

Thanks to the wealth of prebuilt solutions in Azure, the days of having to write complicated plumbing are over. Free yourself up to work on the things that matter to you by taking advantage of all that Azure offers.

Keep learning with Azure

With your Azure free account, you get all of this—and you won't be charged until you choose to upgrade:

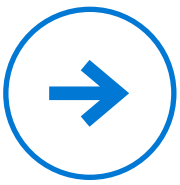
- 12 months of popular free services.
- $200 credit to explore any Azure service for 30 days.
- 25+ services always free.

- Get started with Azure: Watch these short tutorials on how to use Azure and start building projects right away. You can also join our weekly webinar, which provides demos of Azure basics and provides ongoing access to experts.
- Microsoft Learn: Learn new skills and discover the power of Microsoft products with step-by-step guidance. Start your journey today by exploring our learning paths and modules.
- Azure Friday, Take a look at Azure Services and features with the Microsoft engineering team.
- Azure.Source: Keep current on what's happening in Azure, including news and updates, what's now in preview, and what's generally available.
- Azure Tips and Tricks: Browse a collection of useful ideas to help you become more productive with Azure.

Free resources extravaganza

In addition to this guide, there are many other free resources about Azure, including:

- Learn Azure in a Month of Lunches: Practical way to learn Azure from scratch over a month of lunches.
- Azure Serverless Computing Cookbook: E-book about everything serverless in Azure.
- Designing Distributed Systems: E-book on building containerized applications, with hands-on labs on Azure Kubernetes Service.
- Containerize Your Apps with Docker and Kubernetes: Practical guide for moving applications to the cloud with Docker and Kubernetes.
- Guide to NoSQL with Azure Cosmos DB: E-book on building responsive, mission-critical apps with Azure Cosmos DB.
- Effective DevOps: Practical guide for improving collaboration across teams, promoting efficient use of tools, and using the concepts of DevOps.
- Developers Guide to IoT: E-book that provides an overview of Azure IoT services and gets you started.
- Azure for Architects: Comprehensive guide for Azure architects.
- Developer's Guide to Building AI Applications: Practical guide for creating your first intelligent bot with AI.
- Designed to Disrupt: Inspiration and guidance on how transformational changes are possible and how to achieve them.
- Practical Microsoft Azure IaaS: Tips and best practices on how to migrate on-premises systems to the cloud with Azure.
- Enterprise Cloud Strategy: Proven methods for moving your enterprise to a cloud computing strategy.
- Cloud Migration Essentials: E-book on how to simplify your path to the cloud while minimizing risk and impact to your business.
- Making the Most of the Cloud Everywhere: E-book that focuses on unified development and modernization practices in hybrid environments.



http://www.azure.com/free

About the authors

Michael and Barry are passionate about Azure and 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 Michael on Twitter @mbcrump, follow his blog at https://www.michaelcrump.net/, or catch up on a recent post in the Azure Tips and Tricks series.

Barry Luijbregts is an independent software architect and developer with a passion for the cloud and authors courses for Pluralsight.

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

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


Microsoft