



Get started guide for Azure developers

Authors and Contributors

The following resources contributed to this version of this guide:

Author

Glenn Gailey | Microsoft – Senior Content Developer

Summary

The purpose of this document is to provide information that will help quickly get started using Azure services. The target audience is those in a developer role.

© 2016 Microsoft. All rights reserved. This document is for informational purposes only. Microsoft makes no warranties, express or implied, with respect to the information presented here.

Contents

What is Azure?	3
Where do I start?	3
Application hosting.....	3
Azure App Service.....	4
Azure Virtual Machines	5
Azure Functions (serverless)	5
Azure Service Fabric	5
Enhance your applications with Azure services	6
Hosted storage and data access.....	6
Docker support.....	6
Authentication.....	7
Monitoring.....	8
DevOps integration	8
Azure regions	9
Choose the best region for your application and data.....	9
Multi-region apps	9
How do I manage my applications and projects?.....	10
Command-line interfaces and PowerShell	10
Azure portal	10
REST APIs	10
APIs	10
Azure Resource Manager	11
Understanding accounts, subscriptions, and billing	11
What is an Azure account?	11
Manage your subscriptions	12
Administrator roles	12
Resource groups.....	12
Grant access to resources	12
Tags.....	13
Billing	13
Get resource usage data	13
Predict future costs	13
Set up billing alerts.....	13

What is Azure?

Azure is a complete cloud platform that can host your existing application infrastructure, provide compute-based services tailored for your application development needs, or even augment your on-premises applications. Azure integrates the cloud services that you need to develop, test, deploy, and manage your applications—while taking advantage of the efficiencies of cloud computing.

By hosting your applications in Azure, you can start small and easily scale your application as your customer demand grows. Azure also offers the reliability that's needed for high-availability applications, even including failover between different regions. You can easily manage all of your services remotely from one location, the [Azure portal](#), or programmatically by using various APIs and templates.

Who should read this: This guide is an introduction to the Azure platform for application developers. It will provide the guidance and direction that you need to start building new applications in Azure or migrating existing applications to Azure.

Where do I start?

With all the services that Azure offers, it can be a daunting task to figure out which services you need to support your solution architecture. This section highlights the Azure services that developers commonly use. For a list of all Azure services, see the [Azure documentation](#).

You should first decide on how to host your application in Azure. Do you need to manage your entire infrastructure as a virtual machine (VM), or can you use the platform management facilities that Azure provides? Maybe you need a serverless framework to host code execution only?

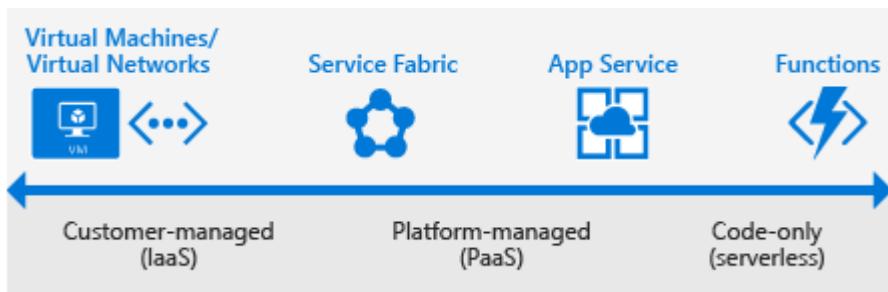
Your application will need cloud storage, which Azure provides several options for. You can take advantage of Azure's enterprise authentication. There are also tools for cloud-based development and monitoring, and most hosting services offer DevOps integration.

Now, let's take a look at some of the specific services that we recommend investigating for your applications.

Application hosting

Azure provides several cloud-based compute offerings to run your application so that you don't have to worry about the infrastructure details. You can easily scale up or scale out your resources as your application usage grows.

Azure offers services that cover the spectrum—from providing infrastructure-as-a-service (IaaS) to give you full control over your application hosting, through platform-as-a-service (PaaS) where the services are hosted for you, to true serverless hosting where all you need to do is write your code.



Azure application hosting options

Azure App Service

When you want the quickest path to publish your web-based projects, consider Azure App Service. App Service makes it easy to extend your web apps to support your mobile clients and publish easily consumed REST APIs. This platform provides authentication by using social providers, traffic-based autoscaling, testing in production, and continuous and container-based deployments.

When you create a new app in App Service, you select one of the following types:

- **Web Apps:** Lets you host websites and web applications that are written in .NET, Java, PHP, Node.js, and Python.
- **Mobile Apps:** Extends Web Apps to support access from mobile devices. It enables authentication with social providers and Azure Active Directory (Azure AD), provides backend storage, and integrates with [Azure Notification Hubs](#) for push notifications.
- **API Apps:** Lets you more securely expose your APIs in the cloud with Swagger metadata so that clients can easily consume them.

Because all three app types share the same App Service runtime, you can host a website, support mobile clients, and expose your APIs in Azure, all from the same project or solution. To learn more about App Service, see [How App Service Works](#).

App Service has been designed with DevOps in mind. It supports various tools for publishing and continuous integration deployments, including GitHub webhooks, Jenkins, Visual Studio Team Services, TeamCity, and others.

You can migrate your existing applications to App Service by using the [online migration tool](#).

When to use: Use App Service when you're migrating existing web applications to Azure, when you need a fully managed hosting platform for your web apps, when you need to support mobile clients, or when you need to expose REST APIs.

Get started: App Service makes it easy to create and deploy your first [web app](#), [mobile app](#), or [API app](#).

Try it now: App Service lets you provision a short-lived app to try the platform without having to sign up for an Azure account. Try the platform and [create your Azure App Service app](#).

Azure Virtual Machines

As an infrastructure-as-a-service (IaaS) provider, Azure lets you deploy to or migrate your application to either Windows or Linux VMs. Together with Azure Virtual Network, Azure Virtual Machines supports the deployment of Windows or Linux VMs to Azure. With VMs, you have total control over the configuration of the machine, and you're responsible for all server software installation, configuration, maintenance, and operating system patches.

Because of the level of control that you have with VMs, you can run a wide range of server workloads on Azure that don't fit into a PaaS model—such as database servers, Windows Server Active Directory, and Microsoft SharePoint. For more information, see the Virtual Machines documentation for either [Linux](#) or [Windows](#).

When to use: Use Virtual Machines when you want full control over your application infrastructure or when you need to migrate on-premises application workloads to Azure without having to make changes.

Get started: Create a new [Linux VM](#) or [Windows VM](#) from the Azure portal.

Azure Functions (serverless)

Maybe you simply want to execute code in response to events or on a schedule, without worrying about building out and managing a whole application or the infrastructure to run your code. [Azure Functions](#) is a "serverless"-style offering that lets you write just the code you need—with code execution that's triggered by HTTP requests, webhooks, cloud service events, or on a schedule. With Functions, you can code in your development language of choice, such as C#, F#, Node.js, Python, or PHP. With consumption-based billing, you pay only for the time that your code executes, and Azure scales as needed.

When to use: Use Azure Functions when you have code that is triggered by other Azure services, by web-based events, or by a schedule. You can also use it when you don't need the overhead of a full project or when you only want to pay for the time that your code runs. To learn more, see [Azure Functions Overview](#).

Get started: Follow the Functions quickstart tutorial to [create your first function](#) from the portal.

Try it now: Azure Functions lets you run your code without having to sign up for an Azure account. Try it now at and [create your first Azure Function](#).

Azure Service Fabric

[Azure Service Fabric](#) is a distributed systems platform that makes it easy to package, deploy, and manage scalable and reliable microservices. It provides a sophisticated runtime for building distributed, scalable microservices. It also provides comprehensive application management capabilities for provisioning, deploying, monitoring, upgrading/patching, and deleting deployed applications. Apps, which run on a shared pool of machines, can start small and grow to massive scale with hundreds or thousands of machines as needed.

Service Fabric supports WebAPI with Open Web Interface for .NET (OWIN) and ASP.NET Core. It provides SDKs for building services on Linux in both .NET Core and Java. To learn more about Service Fabric, see the [Service Fabric learning path](#).

When to use: Service Fabric is a good choice when you're creating an application or rewriting an existing application to use a microservice architecture—or when you need more control over, or direct access to, the underlying infrastructure.

Get started: [Create your first Azure Service Fabric application](#).

Enhance your applications with Azure services

In addition to application hosting, Azure provides service offerings that can enhance the functionality, development, and maintenance of your applications, both in the cloud and on-premises.

Hosted storage and data access

Most applications must store data, so regardless of how you decide to host your application in Azure, consider one or more of the following storage and data services.

- [Azure SQL Database](#): An Azure-based version of the Microsoft SQL Server engine for storing relational tabular data in the cloud. SQL Database provides predictable performance, scalability with no downtime, business continuity, and data protection.

When to use: When your application requires data storage with referential integrity, transactional support, and support for TSQL queries.

Get started: [Create a SQL database in minutes by using the Azure portal](#).

- [Azure Storage](#): Offers durable, highly available storage for blobs, queues, files, and other kinds of nonrelational data. Storage provides the storage foundation for VMs.

When to use: When your app stores nonrelational data, such as key-value pairs (tables), blobs, files shares, or messages (queues).

Get started: Choose from one of these types storage: [blobs](#), [tables](#), [queues](#), or [files](#).

- [Azure DocumentDB](#): A fully managed and scalable NoSQL database service, which features SQL queries over object data. You can access DocumentDB by using existing MongoDB drivers.

When to use: When your application needs to be able to execute SQL queries over JSON documents, or if you're using MongoDB.

Get started: [Build a DocumentDB C# console application](#). If you're a MongoDB developer, see [DocumentDB protocol support for MongoDB](#).

You can use [Azure Data Factory](#) to move existing on-premises data to Azure. If you aren't ready to move data to the cloud, [Hybrid Connections](#) in BizTalk Services lets you connect your App Service hosted app to on-premises resources. You can also connect to Azure data and storage services from your on-premises applications.

Docker support

Docker containers, a form of OS virtualization, let you deploy applications in a more efficient and predictable way. A containerized application works in production the same way as on your development and test systems. You can manage containers by using standard Docker tools. You can use your existing skills and popular open-source tools to deploy and manage container-based applications on Azure.

Azure provides several ways to use containers in your applications.

- [Azure Docker VM extension](#): Lets you configure your VM with Docker tools to act as a Docker host.

When to use: When you want to generate consistent container deployments for your applications on a VM, or when you want to use [Docker Compose](#).

Get started: [Create a Docker environment in Azure by using the Docker VM extension](#).

- [Azure Container Service](#): Lets you create, configure, and manage a cluster of virtual machines that are preconfigured to run containerized applications. To learn more about Container Service, see [Azure Container Service introduction](#).

When to use: When you need to build production-ready, scalable environments that provide additional scheduling and management tools, or when you're deploying a Docker Swarm cluster.

Get started: [Deploy a Container Service cluster](#).

- [Docker Machine](#) (a Linux-based tool): Lets you install and manage a Docker Engine on virtual hosts by using `docker-machine` commands.

When to use: When you need to quickly prototype an app by creating a single Docker host.

- [Custom Docker image for App Service](#): Lets you use Docker containers from a container registry or a customer container when you deploy a web app on Linux.

When to use: When deploying a web app on Linux to a Docker image.

Get started: [Use a custom Docker image for App Service on Linux](#).

Authentication

It's crucial to not only know who is using your applications, but also to prevent unauthorized access to your resources. Azure provides several ways to authenticate your app clients.

- [Azure Active Directory \(Azure AD\)](#): The Microsoft multitenant, cloud-based identity and access management service. You can add single-sign on (SSO) to your applications by integrating with Azure AD. You can access directory properties by using the Azure AD Graph API directly or the Microsoft Graph API. You can integrate with Azure AD support for the OAuth2.0 authorization framework and Open ID Connect by using native HTTP/REST endpoints and the multiplatform Azure AD authentication libraries.

When to use: When you want to provide an SSO experience, work with Graph-based data, or just authenticate domain-based users.

Get started: To learn more, see the [Azure Active Directory developer's guide](#).

- [App Service Authentication](#): When you choose App Service to host your app, you also get built-in authentication support for Azure AD, along with social identity providers—including Facebook, Google, Microsoft, and Twitter.

When to use: When you want to enable authentication in an App Service app by using Azure AD, social identity providers, or both.

Get started: To learn more about authentication in App Service, see [Authentication and authorization in Azure App Service](#).

To learn more about security best practices in Azure, see [Azure security best practices and patterns](#).

Monitoring

With your application up and running in Azure, you need to be able to monitor performance, watch for issues, and see how customers are using your app. Azure provides several monitoring options.

- [Visual Studio Application Insights](#): An Azure-hosted extensible analytics service that integrates with Visual Studio to monitor your live web applications. It gives you the data that you need to continuously improve the performance and usability of your apps, whether they're hosted on Azure or not.

Get started: Follow the [Application Insights tutorial](#).

- [Azure Monitor](#): A service that helps you to visualize, query, route, archive, and act on the metrics and logs that are generated by your Azure infrastructure and resources. Monitor provides the data views that you see in the Azure portal and is a single source for monitoring Azure resources.

Get started: [Get started with Azure Monitor](#).

DevOps integration

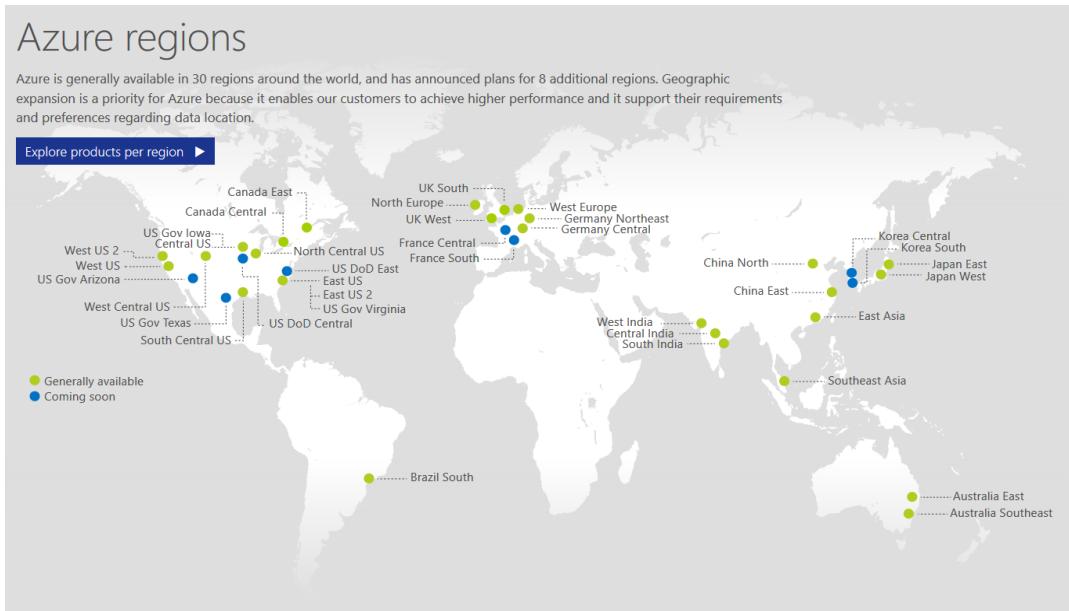
Whether it's provisioning VMs or publishing your web apps with continuous integration, Azure integrates with most of the popular DevOps tools. With support for tools like Jenkins, GitHub, Puppet, Chef, TeamCity, Ansible, VSTS, and others, you can work with the tools that you already have and maximize your existing experience.

Try it now: [Try out several of the DevOps integrations](#).

Get started: To see DevOps options for an App Service app, see [Continuous Deployment to Azure App Service](#).

Azure regions

Azure is a global cloud platform that is generally available in 30 regions around the world. When you provision a service, application, or VM in Azure, you are asked to select a region, which represents a specific datacenter where your application runs or where your data is stored. These regions correspond to specific locations, which are published on the [Azure regions](#) page.



Azure regions map

Choose the best region for your application and data

One of the benefits of using Azure is that you can deploy your applications to a variety of datacenters around the globe. The region that you choose can affect the performance of your application. For example, it's better to choose a region that's closer to a majority of your customers to reduce latency in network requests. You might also want to select your region to meet the legal requirements for distributing your app in certain countries. It's always a best practice to store application data in the same datacenter or in a datacenter as near as possible to the datacenter that is hosting your application.

Multi-region apps

Although it's extremely unlikely, it's possible for an entire datacenter to go offline because of an event such as a natural disaster or Internet failure. It's a best practice to host vital business applications in more than one datacenter to provide maximum availability. This can also reduce latency for local users and provide additional opportunities for flexibility when updating applications.

Some services, such as Virtual Machine and App Services, use [Azure Traffic Manager](#) to enable multi-region support with failover between regions to support high-availability enterprise applications. For an example, see [Azure reference architecture: Web application with high availability](#).

When to use: When you have enterprise and high-availability applications that benefit from failover and replication.

How do I manage my applications and projects?

Azure provides a rich set of experiences for you to create and manage your Azure resources, applications, and projects—both programmatically and in the [Azure portal](#).

Command-line interfaces and PowerShell

Azure provides two ways to manage your applications and services from the command line by using Bash, Terminal, the command prompt, or your command-line tool of choice. Usually, you can perform the same tasks from the command line as in the Azure portal—such as creating and configuring virtual machines, virtual networks, web apps, and other services.

- [Azure Command-Line Interface \(CLI\)](#): Lets you connect to an Azure subscription and program various tasks against Azure resources from the command line.
- [Azure PowerShell](#): Provides a set of modules with cmdlets that enable you to manage Azure resources by using Windows PowerShell.

Azure portal

The Azure portal is a web-based application that you can use to create, manage, and remove Azure resources and services. The Azure portal is located at <https://portal.azure.com>. It includes a customizable dashboard, tooling for managing Azure resources, and an entry way into subscription settings and billing information. For more information, see [Microsoft Azure portal overview](#).

REST APIs

Azure is built on a set of REST APIs that support the Azure portal UI. Most of these REST APIs are also supported to let you programmatically provision and manage your Azure resources and applications from any Internet-enabled device. For the complete set of REST API documentation, see the [Azure REST SDK reference](#).

APIs

In addition to REST APIs, many Azure services also let you programmatically manage resources from your applications by using platform-specific Azure SDKs, including SDKs for the following development platforms:

- [.NET](#)
- [Node.js](#)
- [Java](#)
- [PHP](#)
- [Python](#)
- [Ruby](#)

Services such as [Mobile Apps](#) and [Azure Media Services](#) provide client-side SDKs to let you access services from web and mobile client apps.

Azure Resource Manager

Running your app on Azure likely involves working with multiple Azure services, all of which follow the same life cycle and can be thought of as a logical unit. For example, a web app might use Web Apps, SQL Database, Storage, Azure Redis Cache, and Azure Content Delivery Network services. [Azure Resource Manager](#) lets you work with the resources in your application as a group. You can deploy, update, or delete all the resources in a single, coordinated operation.

In addition to logically grouping and managing related resources, Azure Resource Manager includes deployment capabilities that let you customize the deployment and configuration of related resources. For example, by using Resource Manager, you can deploy and configure an application that consists of multiple virtual machines, a load balancer, and an Azure SQL database as a single unit.

You develop these deployments by using an Azure Resource Manager template, which is a JSON-formatted document. Templates let you define a deployment and manage your applications by using declarative templates, rather than scripts. Your templates can work for different environments, such as testing, staging, and production. For example, by using templates, you can add a button to a GitHub repo that deploys the code in the repo to a set of Azure services with a single click.

When to use: Use Resource Manager templates when you want a template-based deployment for your app that you can manage programmatically by using REST APIs, the Azure CLI, and Azure PowerShell.

Get started: To get started using templates, see [Authoring Azure Resource Manager templates](#).

Understanding accounts, subscriptions, and billing

As developers, we like to dive right into the code and try to get started as fast as possible with making our applications run. We certainly want to encourage you to start working in Azure as easily as possible. This is why Azure offers a [free trial](#)—and some services have a "Try it for free" functionality, like [Azure App Service](#), which doesn't require you to create an account. As fun as it is to just dive into coding and deploying your application to Azure, it's also important to take a little bit of time to understand how Azure works from a standpoint of user accounts, subscriptions, and billing.

What is an Azure account?

To be able to create or work with an Azure subscription, you must have an Azure account. An Azure account is simply an identity in Azure AD or in a directory, such as a work or school organization, that is trusted by Azure AD. If you don't belong to such an organization, you can always create a subscription by using your Microsoft Account, which is trusted by Azure AD. To learn more about integrating on-premises Windows Server Active Directory with Azure AD, see [Integrating your on-premises identities with Azure Active Directory](#).

Every Azure subscription has a trust relationship with an Azure AD instance. This means that it trusts that directory to authenticate users, services, and devices. Multiple subscriptions can trust the same directory, but a subscription trusts only one directory. To learn more, see [How Azure subscriptions are associated with Azure Active Directory](#).

In addition to defining individual Azure account identities, also called *users*, you can also define *groups* in Azure AD. Creating user groups is a good way to manage access to resources in a subscription is by

using role-based access control (RBAC). To learn how to create groups, see [Create a new group in Azure Active Directory preview](#). You can also create and manage groups by using [PowerShell](#).

Manage your subscriptions

A subscription is a logical unit of Azure services that is linked to an Azure account. Each associated account has a role in a subscription. Billing for Azure services is done on a per-subscription basis. For a list of the available subscription offers by type, see [Microsoft Azure Offer Details](#).

Administrator roles

An Azure subscription has multiple account administrator roles, which you can assign at any time.

- **Account Administrator:** This role has full control over the subscription and is the account that is responsible for billing.
- **Service Administrator:** This role has control over all the services in the subscription. By default, this is the same account as the Account Administrator.
- **Co-administrator:** This role has the same access as the Service Administrator, except that it can't change the association of the subscription to an Azure directory.

To learn more about administrator roles, see [How to add or change Azure administrator roles](#).

Resource groups

When you provision new Azure services, you do so in a given subscription. Individual Azure services, which are also called resources, are created in the context of a resource group. Resource groups make it easier to deploy and manage your application's resources. A resource group should contain all the resources for your application that you want to work with as a unit. You can move resources between resource groups and even to different subscriptions. To learn about moving resources, see [Move resources to new resource group or subscription](#).

The Azure Resource Explorer is a great tool for visualizing the resources that you've already created in your subscription. To learn more, see [Use Azure Resource Explorer to view and modify resources](#).

Grant access to resources

When you allow access to Azure resources, it's always a best practice to provide users with the least privilege that's required to perform a given task.

- **Role-based access control (RBAC):** In Azure, you can grant access to user accounts (principals) at a specified scope: subscription, resource group, or individual resources. This lets you deploy a set of resources into a resource group and grant permissions to a specific user or group. This limits access to only the resources that belong to the target resource group. You can also grant access to a single resource, such as a virtual machine or virtual network. To grant access, you assign a role to the user, group, or service principal. There are many predefined roles, and you can also define your own custom roles.

When to use: When you need fine-grained access management for users and groups.

Get started: To learn more, see [Get started with access management in the Azure portal](#).

- **Service principal objects:** In addition to providing access to user principals and groups, you can grant this same access to a service principal.

When to use: When you're programmatically managing Azure resources or granting access for applications. For more information, see [Create Active Directory application and service principal](#).

Tags

Azure Resource Manager lets you assign custom tags to individual resources. Tags, which are key-value pairs, can be helpful when you need to organize resources for billing or monitoring. Tags provide you a way to track resources across multiple resource groups. You can assign tags in the portal, in the Azure Resource Manager template, or programmatically, by using the REST API, the Azure CLI, or PowerShell. You can assign multiple tags to each resource. To learn more, see [Using tags to organize your Azure resources](#).

Billing

In the move from on-premises computing to cloud-hosted services, tracking and estimating service usage and related costs are significant concerns. It's important to be able to estimate what new resources will cost to run on a monthly basis, and also be able to project how the billing will look for a given month based on the current spending.

Get resource usage data

Azure provides a set of Billing REST APIs that give access to resource consumption and metadata information for Azure subscriptions. This gives you the ability to better predict and manage Azure costs. These Billing APIs enable you to track and analyze spending in hourly increments, create spending alerts, and predict future billing based on current usage trends.

Get started: To learn more about using the Billing APIs, see [Azure Billing Usage and RateCard APIs overview](#).

Predict future costs

Although it's challenging to estimate costs ahead of time, Azure has a [pricing calculator](#) that you can use when you estimate the cost of deployed resources. You can also use the Billing blade in the portal and the Billing REST APIs to estimate future costs, based on current consumption.

Get started: See [Azure Billing Usage and RateCard APIs overview](#).

Set up billing alerts

After you've deployed your application or solution on Azure, you can create alerts that send you email when you approach the spending limits that are defined in the alert.

Get started: To learn more, see [Set up billing alerts for your Microsoft Azure subscriptions](#).