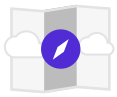
Azure for .NET developers

Learn to use the Azure SDK for .NET. Browse API reference, sample code, tutorials, quickstarts, conceptual articles and more. Know that .NET ♡ Azure.



OVERVIEW
Introduction to Azure and .NET

QUICKSTART
Create an ASP.NET Core web app in Azure

QUICKSTART
Build a serverless function

TUTORIAL
ASP.NET Core and Docker

DEPLOY
Deploy a .NET app with Azure DevOps

TUTORIAL
Authentication end-to-end in App Service

TRAINING
Secure custom APIs with Microsoft Identity

GET STARTED
Azure SDK for .NET

Featured content

Learn to develop .NET apps leveraging a variety of Azure services.

Create web apps

- App Service overview
- Azure Functions overview
- Host a web app with Azure App Service
- Develop, test, and deploy an Azure Function with

Create cloud native apps

- Build cloud native apps using .NET Aspire
- Build your first .NET Aspire app

Create mobile apps

- Consume REST services from Azure in Xamarin apps
- Create a Xamarin.Forms app with .NET SDK and

- Visual Studio
- Publish and manage your APIs with Azure API Management
- ASP.NET Core web app with App Service and Azure SQL Database
- Managed identity with ASP.NET and Azure SQL Database
- Web API with CORS in Azure App Service

- Run and debug a microservice in Kubernetes
- Create and deploy a cloud-native ASP.NET Core microservice
- Deploy and debug multiple containers in AKS
- Dynamic configuration and feature flags using Azure App Config
- Deploy a .NET Core app to Azure Container Registry

- Azure Cosmos DB's API for MongoDB
- Azure Blob storage client library with Xamarin.Forms
- Send push notifications to Xamarin.Forms apps using ASP.NET Core and Azure Notification Hubs
- Add authentication and manage user identities in your mobile apps

Work with storage & data

- Choose the right data storage option
- Use .NET to query an Azure SQL Database or Azure SQL Managed Instance
- Use .NET to query Azure PostgreSQL
- Use the Repository Pattern with Azure Cosmos DB
- Connect to and query an Azure Database for PostgreSQL database
- Persist and retrieve relational data with Entity Framework Core
- Build a .NET Core app with Azure Cosmos DB in Visual Studio Code
- Store application data with Azure Blob storage

Authentication and security

- Microsoft identity platform (Azure AD) overview
- Secure your application by using OpenID Connect and Azure AD
- Secure custom APIs with Microsoft Identity
- Secure an ASP.NET Core web app with the ASP.NET Identity framework
- Add sign-in to Microsoft to an ASP.NET web app
- End-to-end authentication in App Service
- Use Azure Key Vault with ASP.NET Core
- Integrate Azure AD B2C with a web API
- Azure Identity client library for .NET

Messaging on Azure

- Storing messages with Azure queues
- Inter-application messaging with Azure Service Bus
- Streaming big data with Event Hubs
- Building event-based applications with Event Grid
- Use Azure Queue Storage
- Use Azure Service Bus queues
- Ingest data in real-time through Azure Event Hubs
- Route custom events with Event Grid

Diagnostics and monitoring

- Azure Monitor Application Insights quickstart
- Capture and view page load times in your Azure web app

Migration

- Choose an Azure hosting option
- Migrate a .NET web app or service to Azure App Service
- Migrate an ASP.NET app to an Azure VM

Azure SDK for .NET

- Packages
- Authentication for apps
- Logging
- SDK example app
- Tools checklist
- API reference

- Troubleshoot ASP.NET Core on Azure App Service and IIS
- Capture Application Insights telemetry with .NET Core ILogger
- Application Insights for Worker Service applications
- Troubleshoot an app in Azure App Service using Visual Studio

- Migrate a SQL Server database to Azure

.NET and Azure community resources

.NET

.NET documentation

ASP.NET documentation

Webcasts and shows

Azure Friday

The Cloud Native Show

On .NET

.NET Community Standup

On .NET Live

Open Source

Azure SDK for .NET

Microsoft Identity Web

.NET Platform

Are you interested in contributing to the .NET docs? For more information, see our contributor guide.

Introduction to Azure and .NET

Article • 03/24/2023

Azure is a cloud platform designed to simplify the process of building modern applications. Whether you choose to host your applications entirely in Azure or extend your on-premises applications with Azure services, Azure helps you create applications that are scalable, reliable, and maintainable. With extensive support in tools you already use like Visual Studio and Visual Studio Code and a comprehensive SDK library, Azure is designed to make you, the .NET developer, productive right from the start.

Application development scenarios on Azure

You can incorporate Azure into your application in different ways depending on your needs.

- **Application hosting on Azure** - Azure can host your entire application stack from web applications and APIs to databases to storage services. Azure supports a variety of hosting models from fully managed services to containers to virtual machines. When using fully managed Azure services, your applications can take advantage of the scalability, high-availability, and security built in to Azure.

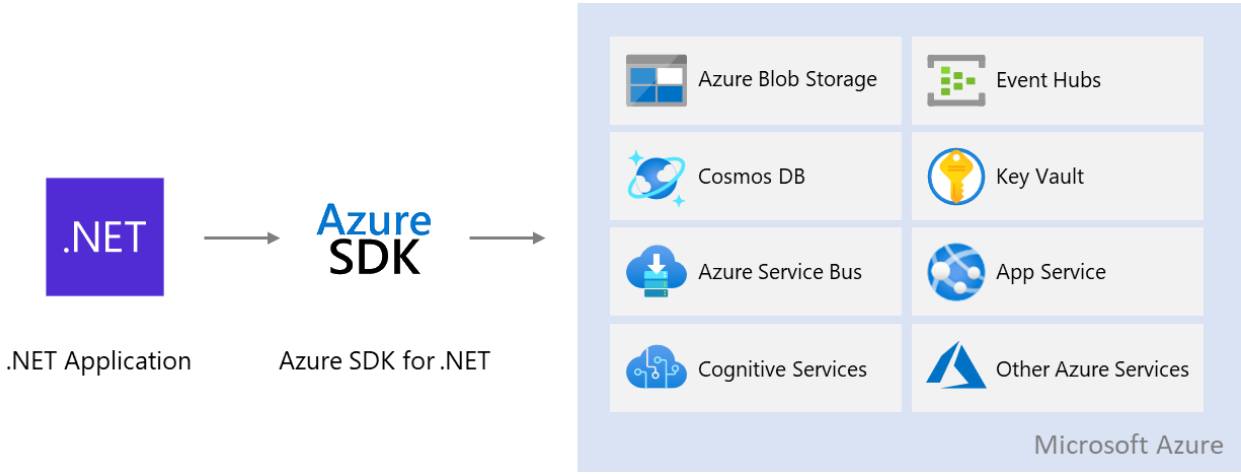
- **Consuming cloud services from applications** - Existing apps can incorporate Azure services to extend their capabilities. This could include adding full-text searching capability with Azure Cognitive Search, securely storing application secrets in Azure Key Vault or adding vision, speech and language understanding capabilities with Azure Cognitive Services. These services are fully managed by Azure and can be easily added to your application without changing your current application architecture or deployment model.

- **Modern serverless architectures** - Azure Functions simplify building solutions to handle event-driven workflows, whether responding to HTTP requests, handling file uploads in Blob storage, or processing events in a queue. You write only the code necessary to handle your event without worrying about servers or framework code. Further, you can take advantage of over 250 connectors to other Azure and third-party services to tackle your toughest integration problems.

Access Azure services from .NET applications

Whether your app is hosted in Azure or on-premises, access to most Azure services is provided through the **Azure SDK for .NET**. The Azure SDK for .NET is provided as a

series of NuGet packages and can be used in both .NET Core (2.1 and higher) and .NET Framework (4.6.1 and higher) applications. The Azure SDK for .NET makes incorporating Azure services into your application as easy as installing the correct NuGet package, instantiating a client object and calling the appropriate methods. More information on the Azure SDK for .NET can be found in the Azure SDK for .NET Overview.



Next steps

Next, learn about the most commonly used Azure services for .NET development.

Key Azure Services for .NET developers

Article • 03/24/2023

While Azure contains over 100 services, the following Azure services are the services you will use most frequently as a .NET developer.

Icon	Service	Description
	**Azure App Service**	Azure App Service is a fully managed platform for hosting web applications and APIs in Azure. It features automatic load balancing and auto-scaling in a highly available environment. You pay only for the compute resources you use and free tiers are available.
	**Azure Container Apps**	Azure Container Apps are fully managed environments that enable you to run microservices and containerized applications on a serverless platform. Applications built on Container Apps provide scalability, resiliency, and other advantages of complex container orchestrators while leaving behind many of the complexities.
	**Azure Functions**	Azure Functions is a serverless compute service that lets you write small, discrete segments of code that can be executed in a scalable and cost-effective manner, all without managing any servers or runtimes. Functions can be invoked by a variety of different events and easily integrate with other Azure services through the use of input and output bindings.
	**Azure SQL**	Azure SQL is a fully managed cloud based version of SQL Server. Azure automatically performs traditional administrative tasks like patching and backups for you and features built-in high availability.
	**Azure Cosmos DB**	Azure Cosmos DB is a fully managed NoSQL database with single digit response times, automatic scaling, and a MongoDB compatible API.
	**Azure Blob Storage**	Azure Blob Storage allows your applications to store and retrieve files in the cloud. Azure Storage is highly scalable to store massive amounts of data and data is stored redundantly to ensure high availability.
	**Azure Service Bus**	Azure Service Bus is a fully managed enterprise message broker supporting both point to point and publish-subscribe integrations. It is ideal for building decoupled applications, queue based load leveling, or facilitating communication between microservices.
	**Azure Key Vault**	Every application has application secrets like connection strings and API keys it must store. Azure Key Vault helps you store and access those secrets securely, in an encrypted vault with restricted access to make sure your secrets and your application are not compromised.

Icon	Service	Description
	**Cognitive Services**	Azure Cognitive Services are a collection of cloud-based services that allow you to add AI based capabilities to your application. Examples include computer vision, speech recognition, language understanding, and anomaly detection.

For the full list of Azure products and services, visit the Azure documentation home page

Next steps

Start configuring your Azure development environment by Creating an Azure Account

Create an Azure account

Article • 03/24/2023

To use Azure, you need an Azure account. Your Azure account is the credential you use to sign into Azure services like the Azure Portal or Cloud Shell.

Option 1: Use monthly Azure credits for Visual Studio subscribers

If you have a Visual Studio subscription, your subscription includes credits for using Azure. Activate your credits by visiting the Monthly Azure credits for Visual Studio subscribers page.

Option 2: Sign up for a free Azure account

You can create an Azure account for free and receive 12 months of popular services for free and a $200 credit to explore Azure for 30 days.

Option 3: Sign up for a pay-as-you-go account

You can also create a pay-as-you-go Azure account. This option includes monthly amounts of select services for free, and charges you for what you use beyond the free limits. There is no upfront commitment and you can cancel anytime.

Option 4: Use a corporate account

If you are using Azure at work, talk to your company's cloud administrator to get your Azure credentials and then sign in to your account with the Azure Portal.

Next steps

After creating an Azure account, be sure to bookmark the Azure Portal in your web browser for easy access to Azure.

Next, you will want to configure Visual Studio or Visual Studio Code for Azure development depending on which IDE you use.

Configure Visual Studio for Azure development

Configure Visual Studio Code for Azure development

Configure Visual Studio for Azure development with .NET

Article • 01/31/2023

Visual Studio includes tooling to help with the development and deployment of applications on Azure. This guide will help you make sure that Visual Studio is properly configured for Azure development.

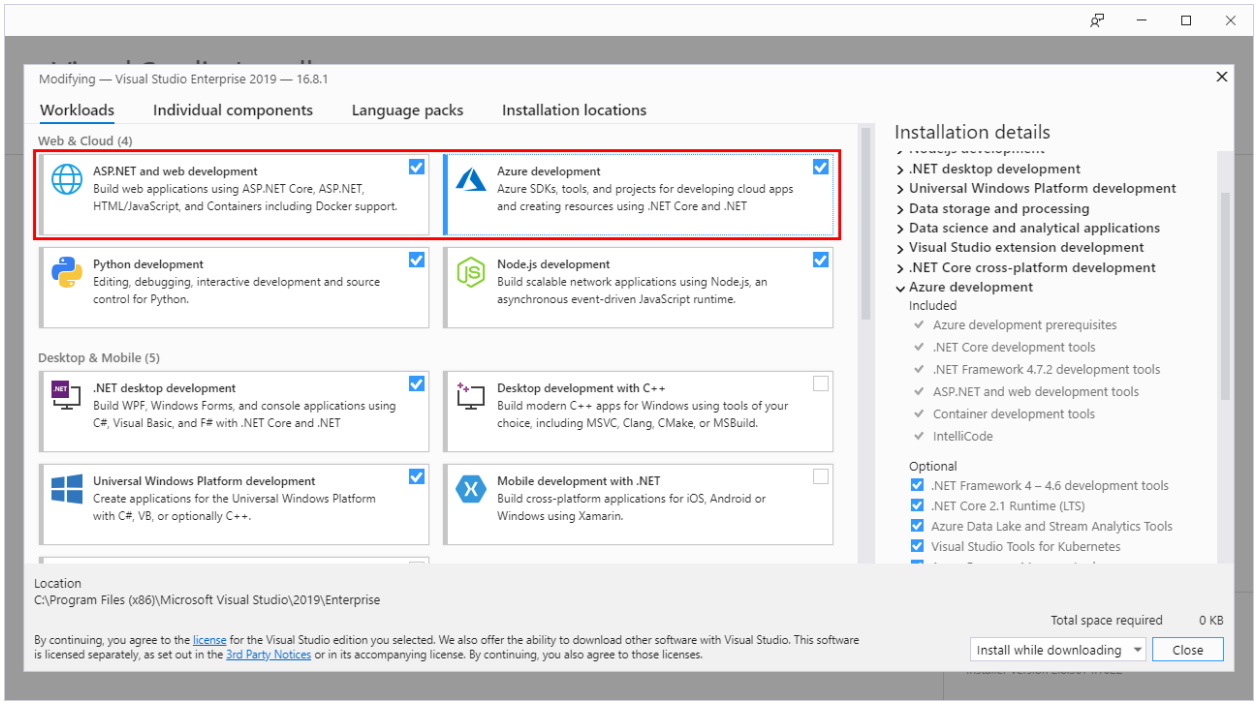
Download Visual Studio

If you already have Visual Studio installed, you can skip this step.

Download Visual Studio

Install Azure workloads

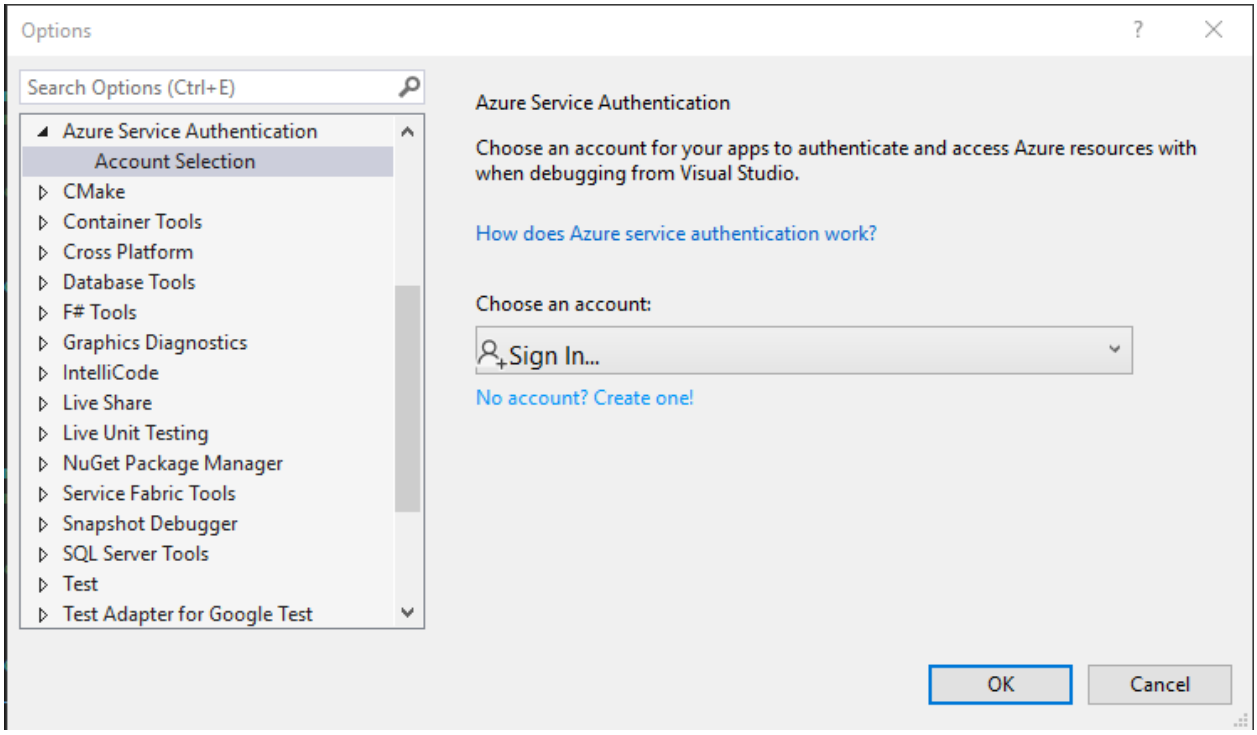
Open Visual Studio Installer and validate that the workloads **Azure development** and **ASP.NET and web development** are installed. If either of these workloads is not installed, select them to be installed.



Authenticate Visual Studio with Azure

When debugging apps through Visual Studio, Visual Studio can use your Azure account to authenticate and access Azure Resources. This account is also used when you publish apps directly from Visual Studio to Azure.

To authenticate your Azure account from Visual Studio, select the **Tools** > **Options** menu to launch the **Options** dialog. Navigate to the **Azure Service Authentication** options and sign in using your Azure account.



Next steps

If you also use Visual Studio Code for development in .NET or any other language, you should configure Visual Studio Code for Azure development. Otherwise, proceed to Installing the Azure CLI.

Configure Visual Studio Code for Azure development

Article • 01/31/2023

If you're using Visual Studio Code, whether for .NET development, for building single page applications using frameworks like Angular, React or Vue, or for writing applications in another language like Python, you'll want to configure Visual Studio Code for Azure development.

Download Visual Studio Code

If you already have Visual Studio Code installed, you can skip this step

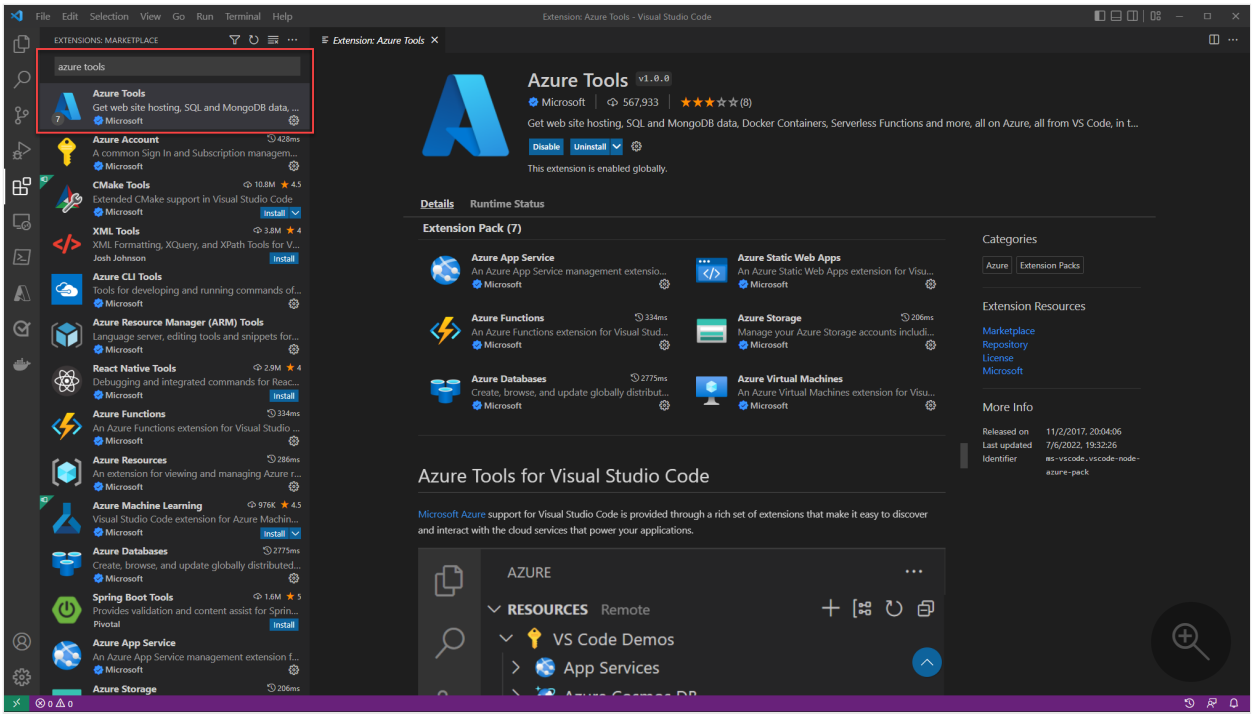
Download Visual Studio Code

Install the Azure Tools Extension Pack

The Azure Tools Extension Pack contains extensions for working with Azure App Service, Azure Functions, Azure Storage, Cosmos DB, and Azure Virtual Machines all in one convenient package.

To install the extension from Visual Studio Code:

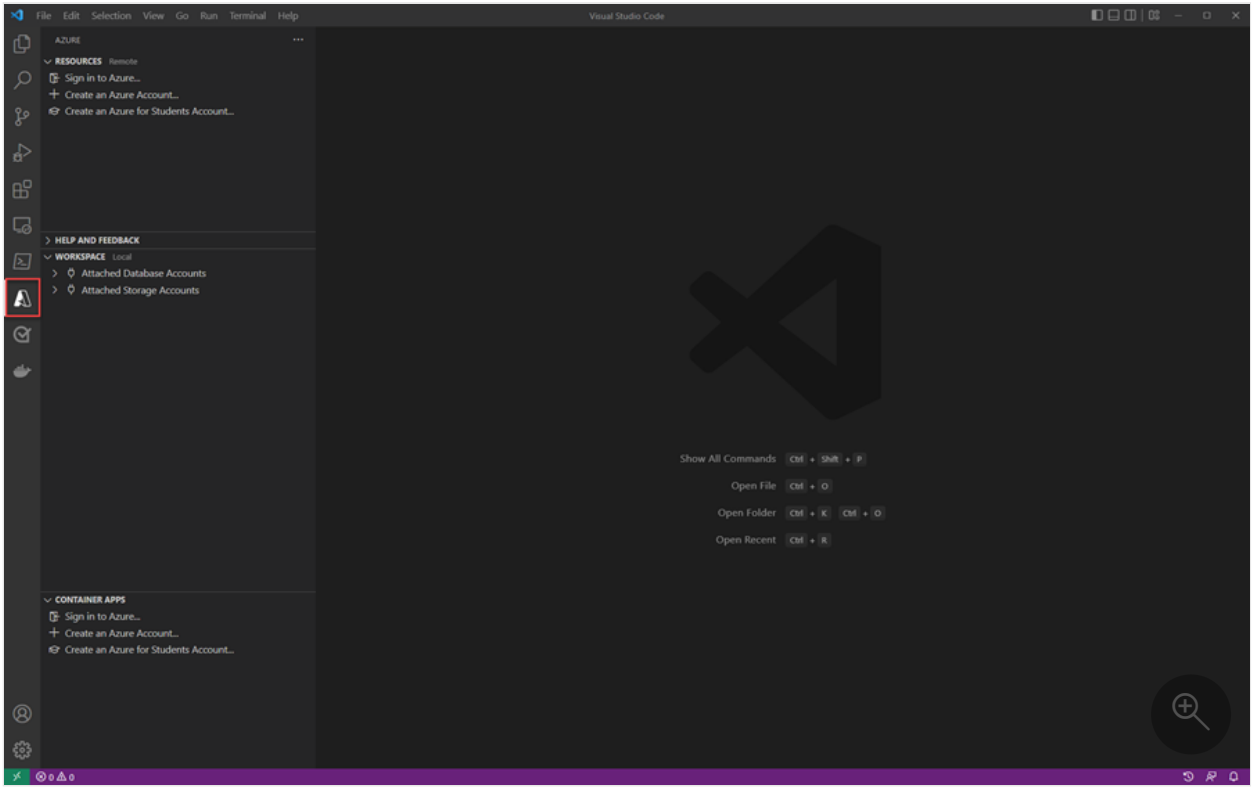
1. Press `Ctrl+Shift+X` to open the **Extensions** window.
2. Search for the *Azure Tools* extension.
3. Select the **Install** button.



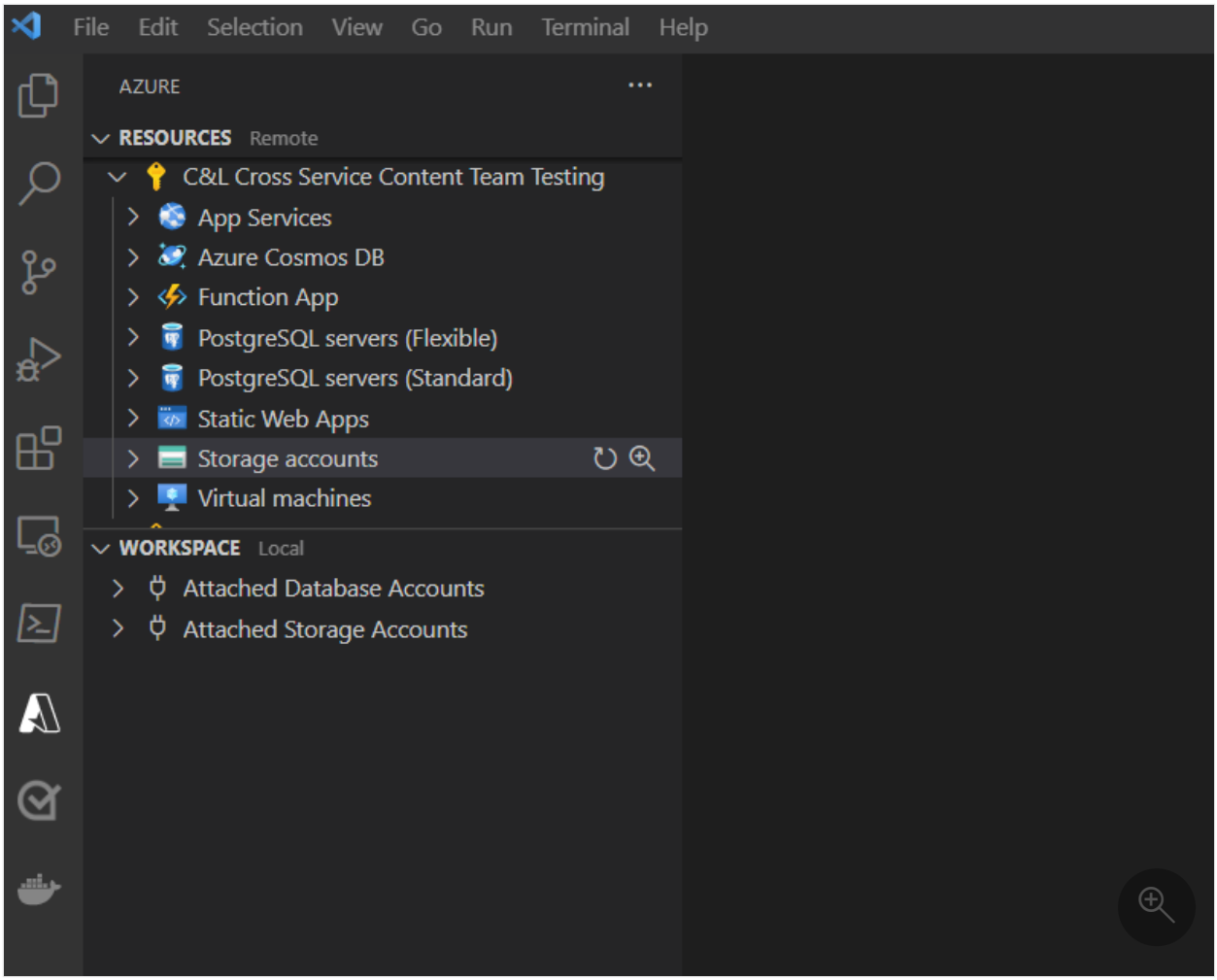
To learn more about installing extensions in Visual Studio Code, refer to the Extension Marketplace document on the Visual Studio Code website.

Sign in to your Azure account with Azure Tools

On the left hand panel, you'll see an Azure icon. Select this icon, and a control panel for Azure services will appear. Choose **Sign in to Azure...** to complete the authentication process for the Azure tools in Visual Studio Code.



After you've signed-in, you'll see all of your existing resources in the **Resources** view. You can create and manage these services directly from Visual Studio Code. You'll also see a **Workspace** view that includes local tasks specific to your workspace and files on your machine, such as attaching to a Database or deploying your current workspace to Azure.



Next steps

Next, you'll want to install the Azure CLI on your workstation.

Install the Azure CLI

Install the Azure CLI

Article • 03/31/2023

In addition to the Azure portal, Azure also offers the Azure CLI as a command-line tool to create and manage Azure resources. The Azure CLI offers the benefits of efficiency, repeatability, and the ability to script recurring tasks.

In practice, most developers use both the Azure portal and the Azure CLI. Whereas the Azure portal is useful when exploring new services and getting an overview of all of the resources in your Azure account, most developers find the Azure CLI to be faster and more efficient. The Azure CLI can often accomplish in a single command what takes multiple steps in the Azure portal. In addition, since Azure CLI commands can be saved to a file, you can ensure that recurrent tasks are run the same way each time.

The Azure CLI is available for Windows, macOS, and Linux.

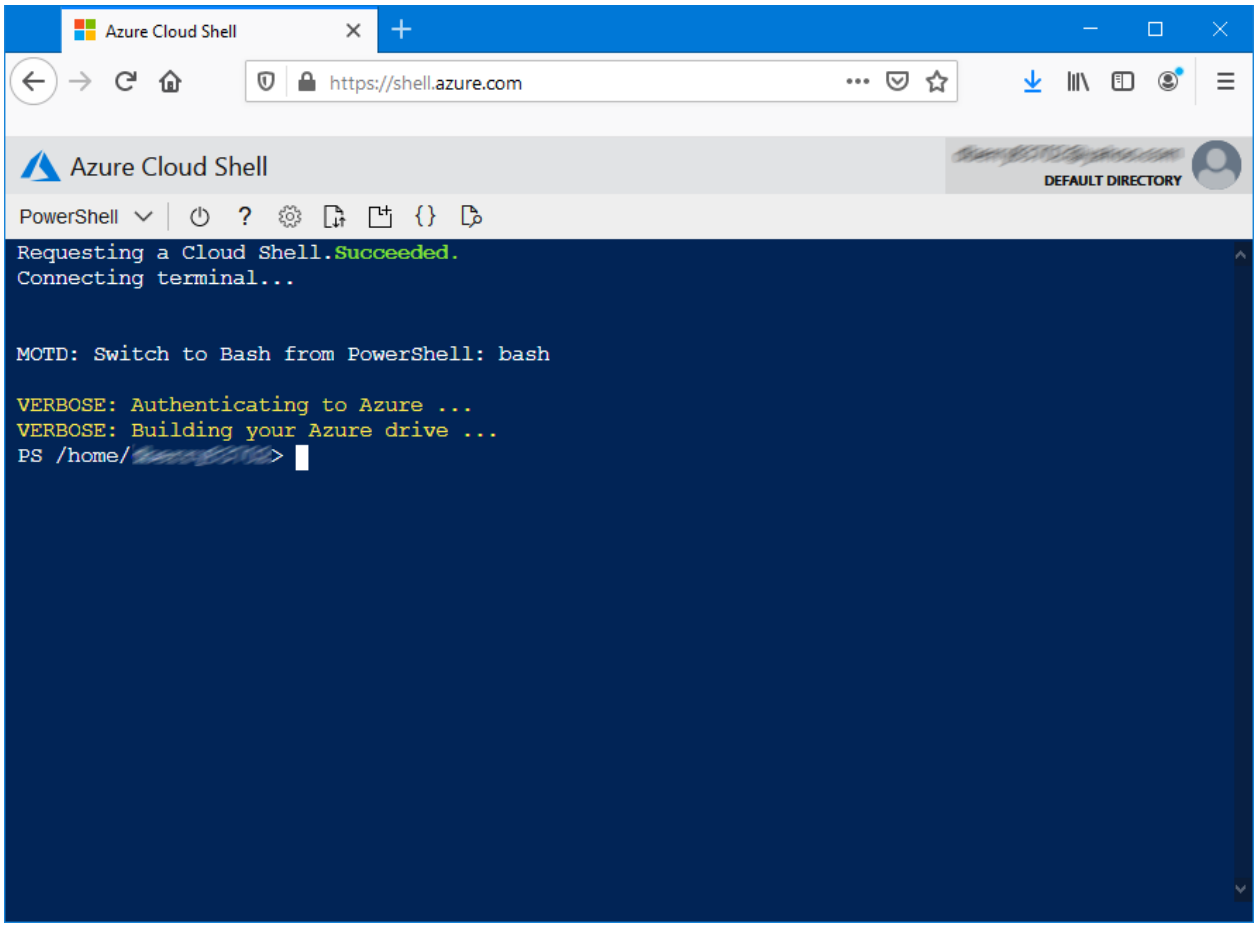
Install the Azure CLI for Windows

Install the Azure CLI for macOS

Install the Azure CLI for Linux

Azure Cloud Shell

You can also use the Azure CLI in the Azure Cloud Shell at https://shell.azure.com. The Azure Cloud Shell is a fully functional, browser-based shell for managing Azure resources. The Azure Cloud Shell is useful when you need a command line environment but are working on a device where you're unable to install the Azure CLI.



Next steps

Next, you'll want to install other Azure tools like Azure Storage Explorer and Azure Data Studio to make you more productive with Azure.

Additional Tools for Azure Developers

Article • 03/31/2023

In addition to configuring your IDE and installing the Azure CLI, multiple other tools and utilities are available to help you be more productive with Azure.

Azure PowerShell

Azure PowerShell is a PowerShell module of cmdlets for managing Azure resource directly from PowerShell, either from the command line or from within PowerShell scripts. Azure PowerShell supports PowerShell features like PowerShell objects and combining commands into pipelines. If you have used PowerShell before or need to write complex automation scripts to manage Azure resources, you will want to install Azure PowerShell.

Install Azure PowerShell

Azure Developer CLI (preview)

Azure Developer CLI (`azd`) is an open-source tool that accelerates the process of building cloud apps on Azure. The CLI provides best practice, developer-friendly commands that map to key stages in your workflow, whether you're working in the terminal, your editor or integrated development environment (IDE), or DevOps.

You can use `azd` with extensible templates that include everything you need to get an application up and running in Azure. These templates include application code, and reusable infrastructure as code assets.

Install Azure Developer CLI

Azure Storage Explorer

Azure Storage Explorer is a free, GUI tool for managing storage resources and data in Azure. You can upload, download and manage blobs and files, as well as manage data in Azure queues, tables and CosmosDB. If you plan on working with any storage resources in Azure, installation of Azure Storage Explorer is recommended. Versions are available for Windows, macOS and Linux.

Download Azure Storage Explorer

Azure Data Studio

Azure Data Studio is a cross-platform database tool for accessing both on-premises and cloud databases. It allows you to edit and execute SQL queries in addition to charting and visualizing result sets. It supports all versions of SQL Server from SQL Server 2014 and later and Azure SQL. If you plan to work with Azure SQL, download and install Azure Data Studio.

Download Azure Data Studio

Next steps

Validate your development environment is set up correctly using the .NET on Azure development environment checklist.

.NET on Azure development environment checklist

Article • 03/24/2023

This checklist is provided to help you make sure you have your development environment correctly configured for .NET development with Azure

Create an Azure account

To access Azure services or run applications in Azure, you need an Azure account.

- If you are a Visual Studio subscriber, you have monthly free Azure credits available to you every month
- Create a free Azure account and receive $200 in credits and select services free for 12 months
- Use an account assigned to you by your company's Azure administrator

Configure your IDE

Popular tools like Visual Studio and Visual Studio Code have extensions available to let you work with Azure right from your IDE.

- Configure Visual Studio for .NET development using Azure
- Configure Visual Studio Code for .NET Development using Azure

Install the Azure CLI

In addition to using the Azure portal, you will want to install the Azure CLI to create and manage Azure resources.

- Install the Azure CLI for Windows
- Install the Azure CLI for macOS
- Install the Azure CLI for Linux

Install additional tools and utilities

These additional tools are designed to make you more productive when working with Azure.

- Install Azure PowerShell if you plan on using PowerShell scripts to create and manage Azure resources.
- Install Azure Developer CLI if you plan on using the Developer CLI to streamline development workflows.
- Install Azure Storage Explorer to upload, download, and manage data in Azure storage resources including blobs, queues, tables, and CosmosDB.
- Install Azure Data Studio if you are working with Azure SQL.

Bookmark the following sites

You will use these sites frequently when doing Azure development. Bookmark them for quick reference.

- Azure Portal - https://portal.azure.com/
- Azure Cloud Shell - https://shell.azure.com/

Assess your .NET applications for migration to Azure

Article • 11/14/2023

This guide describes how to use the application and code assessment for .NET to evaluate how ready your .NET applications are for moving to Azure and what changes you need to make for a successful cloud migration.

What is the application and code assessment for .NET?

The utility is used to assess .NET source code to identify replatforming and migration opportunities to Azure.

It discovers application technology usage through static code analysis, supports effort estimation, and accelerates code replatforming by giving you recommendations on how to overcome any potential issues and make your code cloud-compatible.

Application and code assessment is available in a Visual Studio extension and in a CLI tool.

Application and code assessment for .NET bundles a set of tools, engines, and rules to assess and replatform .NET applications to various Azure targets such as Azure App Service, Azure Kubernetes Service and Azure Container Apps.

When should I use the application and code assessment?

The utility is designed to help organizations modernize their .NET applications in a way that reduces costs and enables faster innovation. It uses advanced analysis techniques to understand the structure and dependencies of any .NET application, and provides guidance on how to refactor and migrate the applications to Azure.

With it you can:

- **Assess the code compatibility with Azure:** get notified about every part of your code that might not work when you move your application to Azure.
- **Get recommendations on refactoring your code:** receive guidance and effort estimates on how to update your code to make it Azure-compatible, what

verifications you should make to ensure your applications will work properly, and how to improve your applications performance, scalability, security, etc.

Supported languages

Application and code assessment for .NET can analyze projects written in the following languages:

- C#
- Visual Basic

Supported project types

It analyzes your code in the following project types:

- ASP.NET
- Class libraries

Supported Azure targets

Currently application identifies potential issues for migration to Azure App Service, AKS, and Azure Container Apps. In the future the tool might have an ability to set the target explicitly and filter the exact issues and recommendations for each target separately.

Next steps

Install the Visual Studio extension or the CLI tool

For information on how to install the Azure Migrate application and code assessment for .NET extension for Visual Studio or CLi tool, see installation instructions.

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For

.NET

.NET feedback

The .NET documentation is open source. Provide feedback here.

Open a documentation issue

Provide product feedback

more information, see our contributor guide.

Install Azure Migrate application and code assessment for .NET

Article • 11/14/2023

Application and code assessment can be installed as a Visual Studio extension or as a .NET command-line tool. When installed as a Visual Studio extension, loaded projects can be analyzed through the context menu in Solution Explorer. The command-line version of the tool provides an interactive step-by-step experience.

Prerequisites

- Windows operating system
- Visual Studio 2022 version 17.1 or later - for the Visual Studio extension
- .NET SDK - for the command-line tool

Install the Visual Studio extension

The application and code assessment for .NET can be installed as a Visual Studio extension, which lets you analyze an open projects in your solution. Use the following steps to install it from inside Visual Studio. Alternatively, you can download and install the extension from the Visual Studio Marketplace.

1. With Visual Studio opened, press the **Extensions** > **Manage Extensions** menu item, which opens the **Manage Extensions** window.
2. In the **Manage Extensions** window, enter "**Azure Migrate**" into the search input box.
3. Select the **Azure Migrate application and code assessment** item, and then select **Download**.
4. Once the extension has been downloaded, close Visual Studio. This starts the installation of the extension.
5. In the VSIX Installer dialog select **Modify** and follow the directions to install the extension.

Install the .NET global tool

Azure Migrate application and code assessment for .NET is also available as a .NET global tool. You can install the tool with the following command.

.NET CLI

```
dotnet tool install -g dotnet-appcat
```

Similarly, to update the tool, use the following command:

.NET CLI

```
dotnet tool update -g dotnet-appcat
```

> **Important**
>
> Installing this tool may fail if you've configured additional NuGet feed sources. Use the `--ignore-failed-sources` parameter to treat those failures as warnings instead of errors.
>
> .NET CLI
>
> ```
> dotnet tool install -g --ignore-failed-sources dotnet-appcat
> ```

Next steps

Use with Visual Studio

For information on how to use and interpret results with Visual Studio, see Use the application and code assessment with Visual Studio.

Use with .NET CLI

For information on how to use and interpret results with the .NET CLI, see Use the application and code assessment with the .NET CLI.

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For

.NET feedback

The .NET documentation is open source. Provide feedback here.

Open a documentation issue

more information, see our contributor guide.

Provide product feedback

Analyze applications with Visual Studio

Article • 11/14/2023

Azure Migrate application and code assessment for .NET helps you to identify any issues your application might have when it is ported to Azure and improve the performance, scalability and security by suggesting modern, cloud-native solutions.

The tool is available as a Visual Studio extension and a CLI tool.

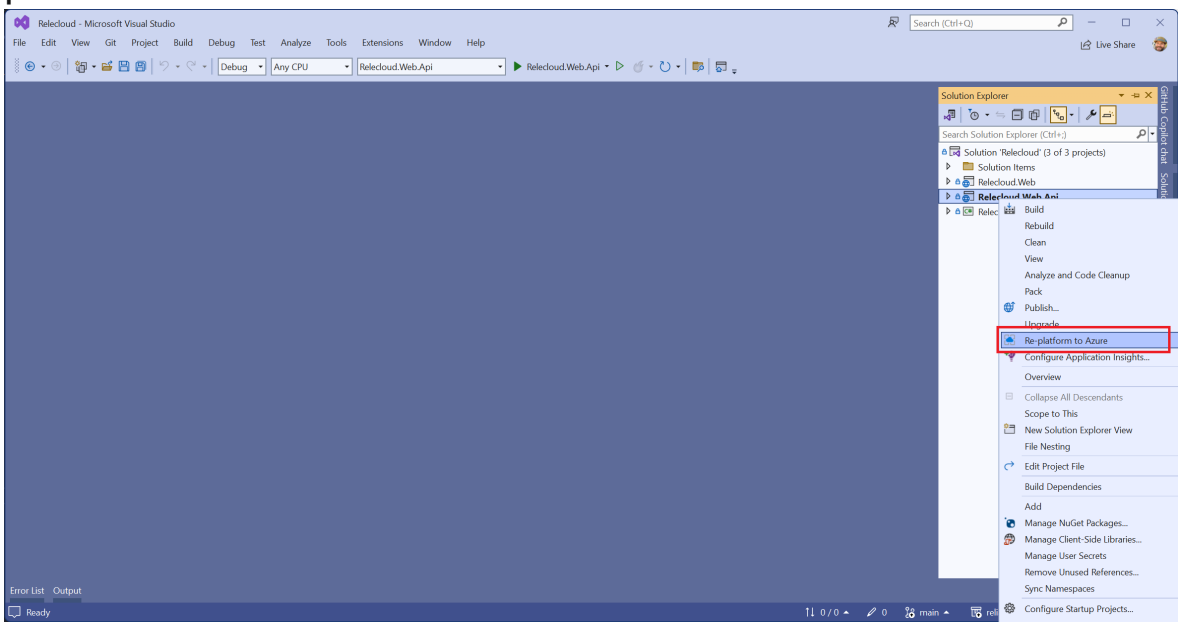
This guide describes how to use the Visual Studio extension to scan your application for possible incompatibilities with Azure.

If you have not installed the Visual Studio extension, please follow these instructions first.

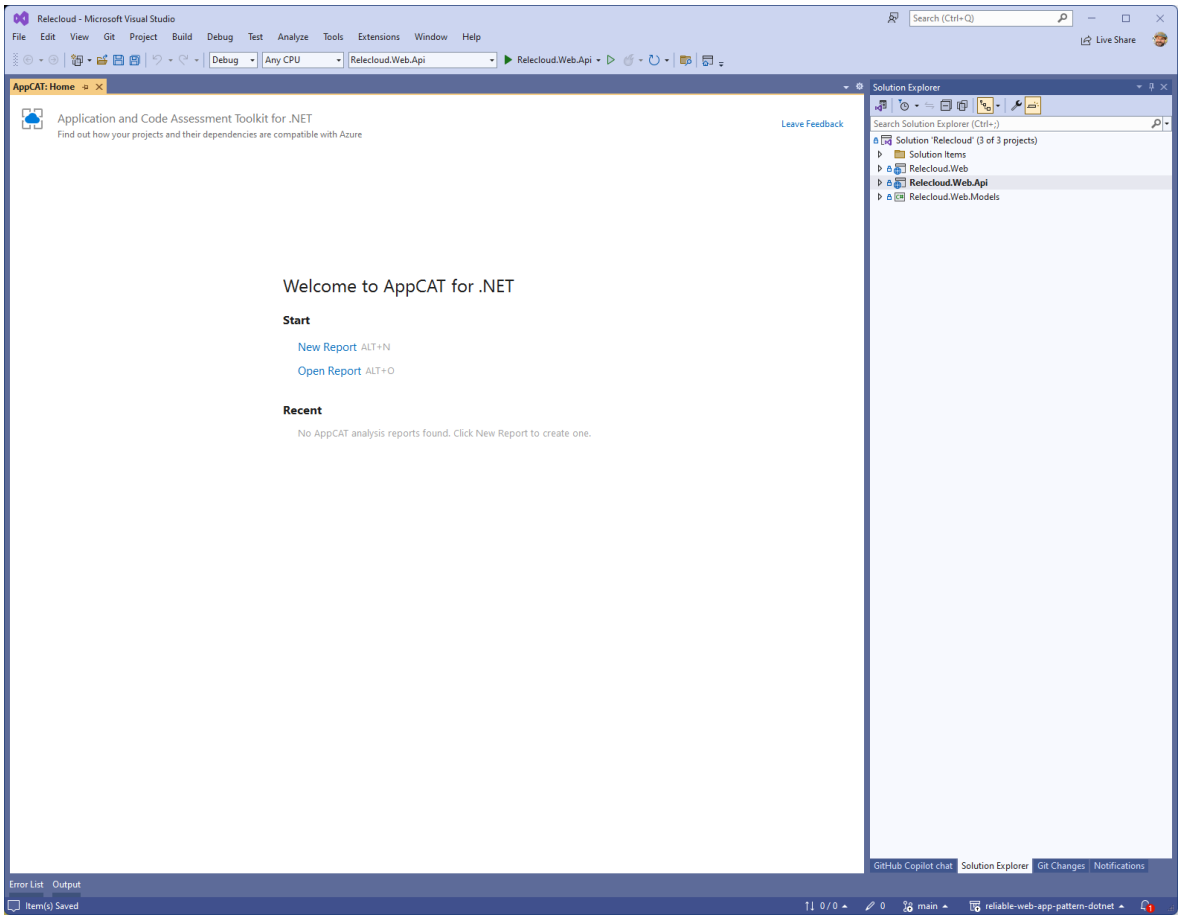
Scan your application

The application and code assessment lets you decide which projects in your solution to scan to identify migration opportunities to Azure. Follow these steps to scan your application.

1. Open the solution containing the projects you want to migrate to Azure in Visual Studio 2022.
2. Right-click on any of the projects in the Solution Explorer window and select **Re-platform to Azure**.

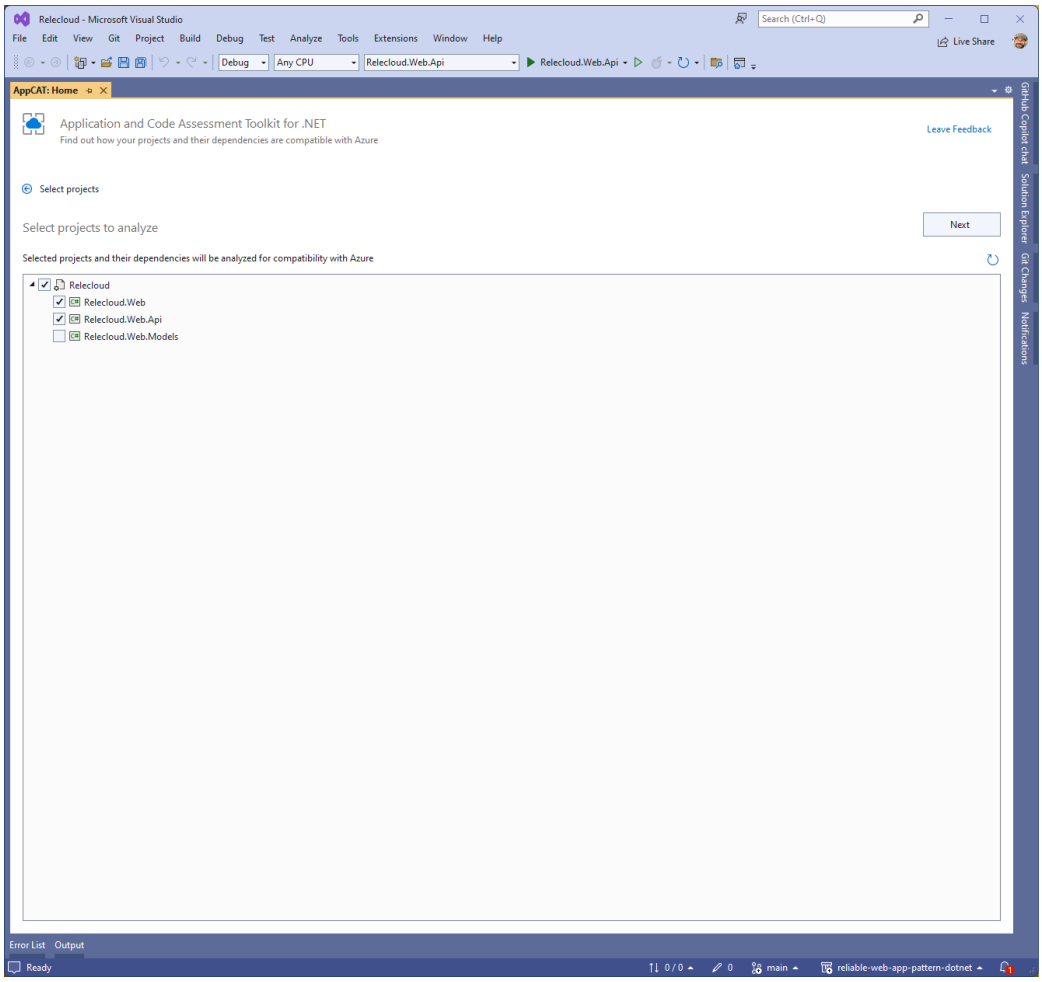


3. The utility will start and give you the option to start a new analysis report or open an existing one. It will also display any recent analysis reports.

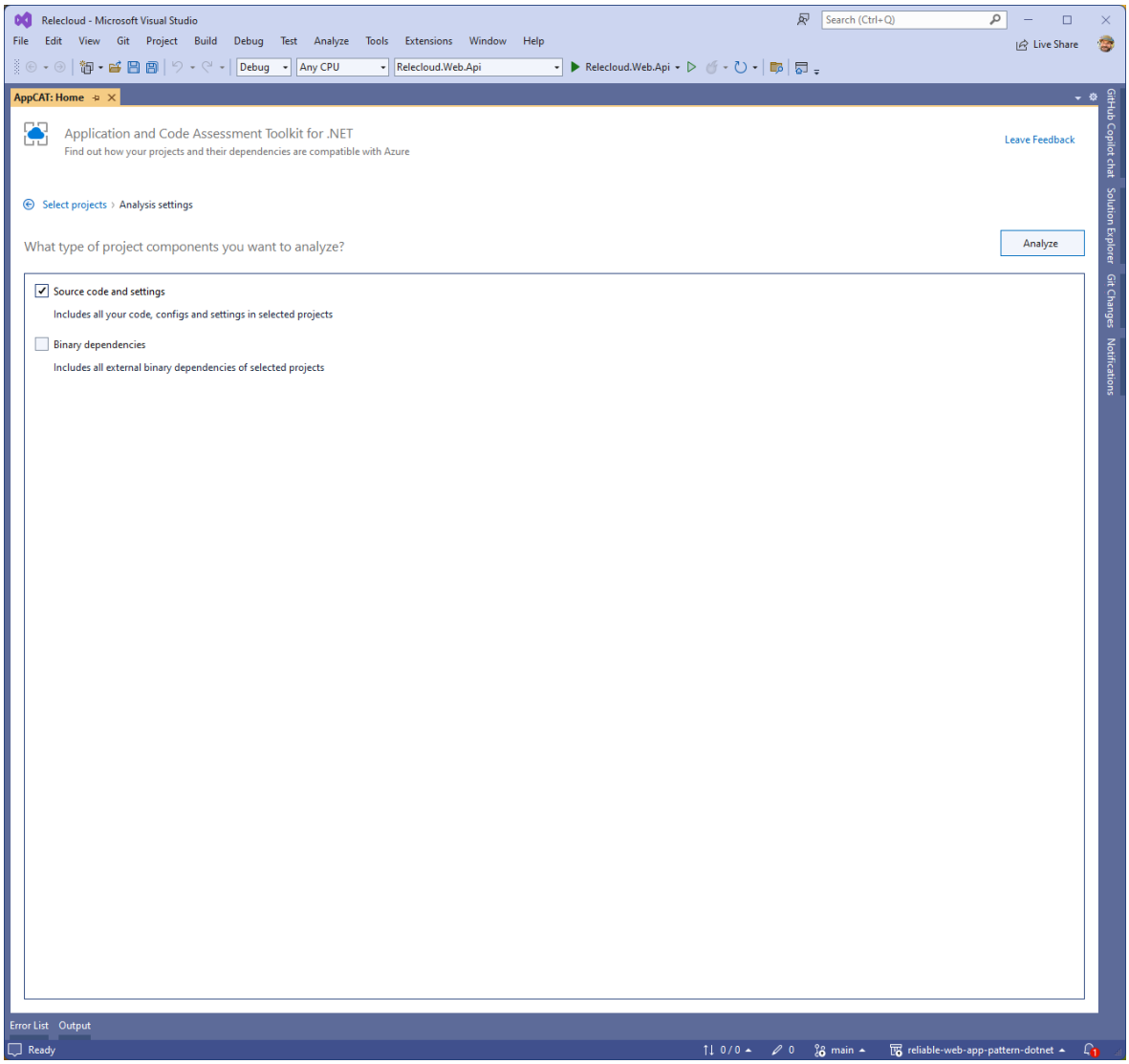


4. Click on **New report** and it will display the projects in your solution in a treeview. It will give you an option to select which projects to analyze. You will find web projects pre-selected for you and you can change the selection by checking or unchecking the boxes next to the projects. When the tool runs, it also analyzes the

dependencies your selected projects has.



5. Click the **Next** button and you'll be presented with the option to analyze **Source code and settings**, **Binary dependencies**, or both.

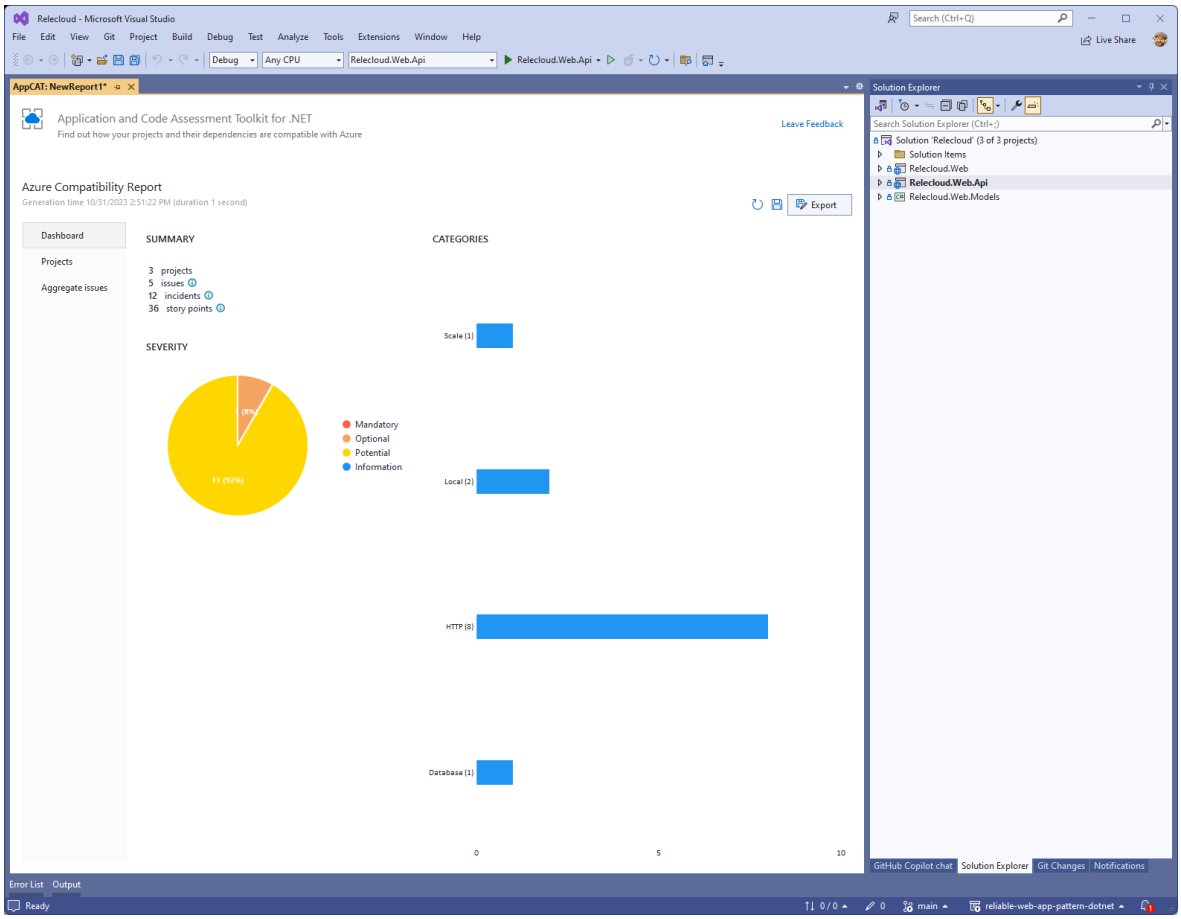


> **Note**
>
> The **source code and settings** option will only scan the source code in the projects you selected on the previous screen. The **Binary dependencies** option will scan any dependencies (such as NuGet packages or referenced dlls) your projects rely on. You can expect to see many more issues identified when **binary dependencies** is selected. Scanning binaries can be valuable because the issues detected may identify potential problem in dependencies, but also might not be as useful because source code is not available for these dependencies, so the issues can't be fixed and, in the case of potential issues, it may not be an issues in your case.

> It might be helpful to generate two different reports: for action items and for your awareness. >

1. Click the **Analyze** button to start the scan. The selected projects are scanned to look for potential issues when migrating to Azure. When finished, you'll see a

dashboard of results.



Next steps

Interpret the results

For information on how to interpret results, see Interpret the analysis results from the Azure Migrate application and code assessment for .NET.

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see our contributor guide.

.NET feedback

.NET is an open source project. Select a link to provide feedback:

Open a documentation issue

Provide product feedback

Analyze applications with the .NET CLI

Article • 11/14/2023

Azure Migrate application and code assessment for .NET helps you to identify any issues your application might have when it is ported to Azure and improve the performance, scalability and security by suggesting modern, cloud-native solutions.

The tool is available as a Visual Studio extension and a CLI tool.

This guide describes how to use the CLI tool to scan your application for possible incompatibilities with Azure.

If you have not installed the .NET CLI tool, please follow these instructions first.

Scan your application

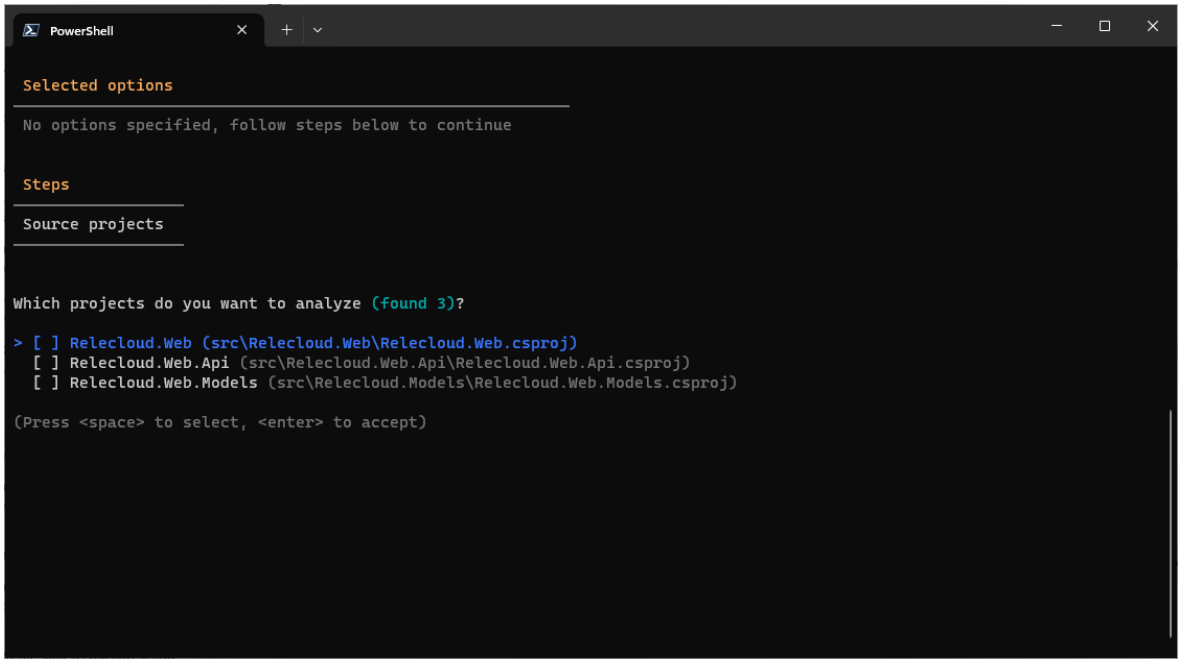
The application and code assessment for .NET CLI tool lets you decide which projects in your solution to scan to identify migration opportunities to Azure. Follow these steps to scan your application.

1. In CLI type `appcat analyze` and press **Enter**.

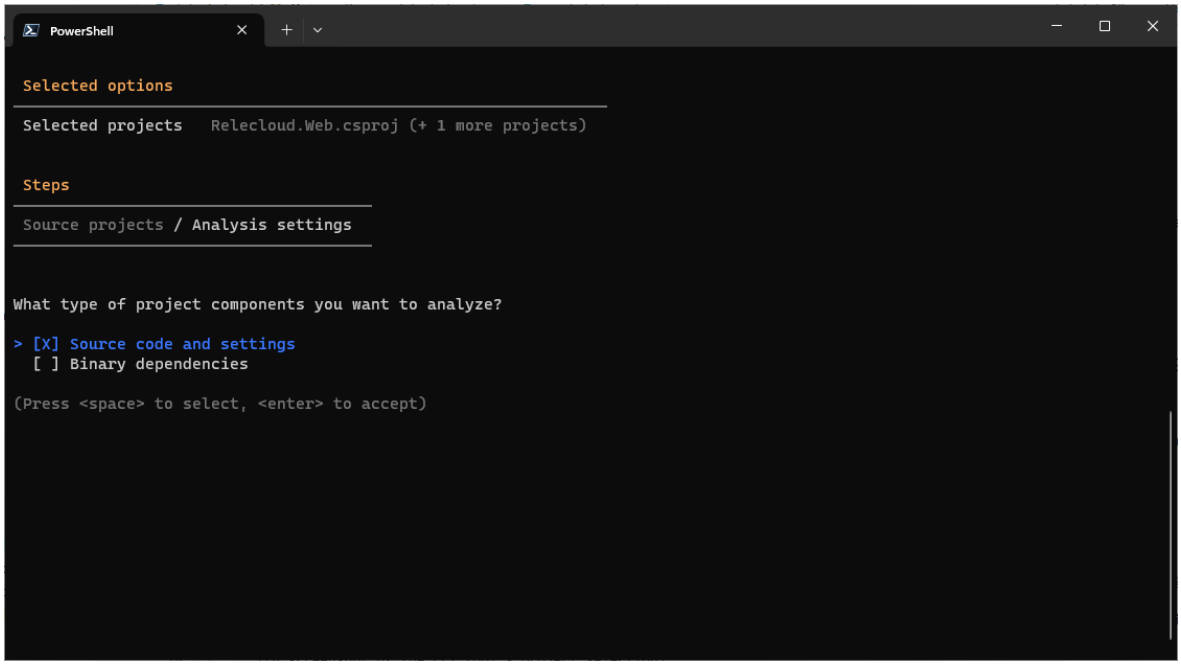
   > **Note**
   >
   > If this is your first time running application and code assessment for .NET, you will see an informational message about telemetry and how to opt-out if you should want to.

2. A screen is presented that allows you to pick the projects in your solution to analyze. Use the arrow keys to highlight individual projects and press **Space** to

select them. Press **Enter** when you're ready.

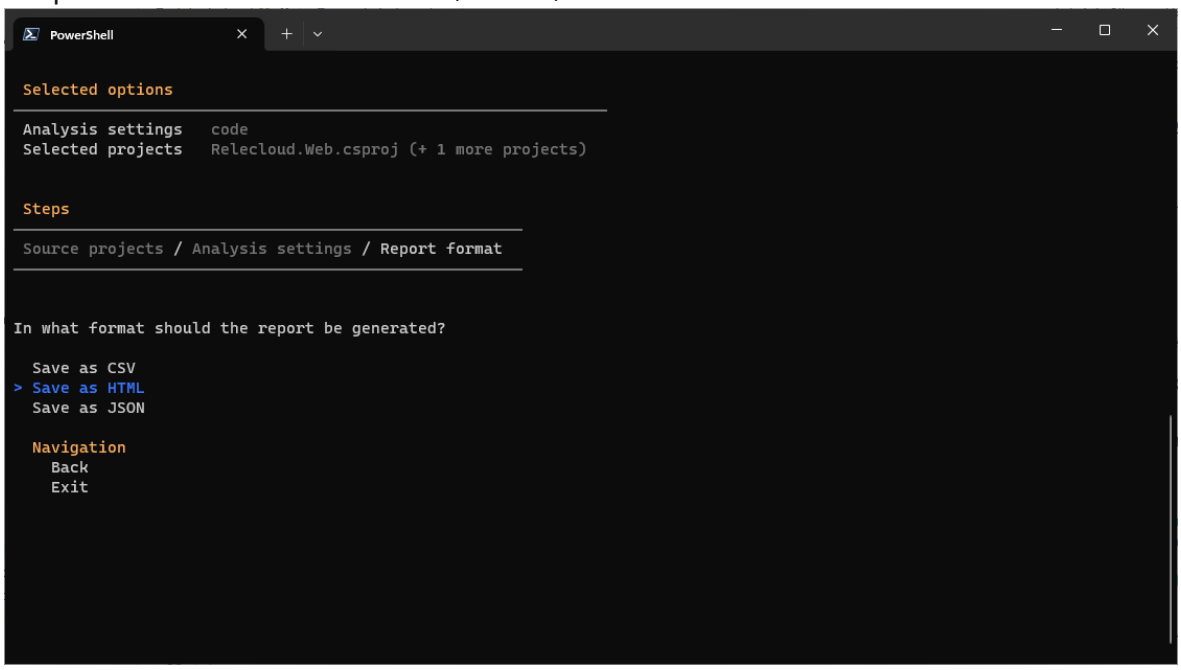


3. Next you'll be presented with the option to analyze **Source code and settings**, **Binary dependencies**, or both. Make your choice and press **Enter**.

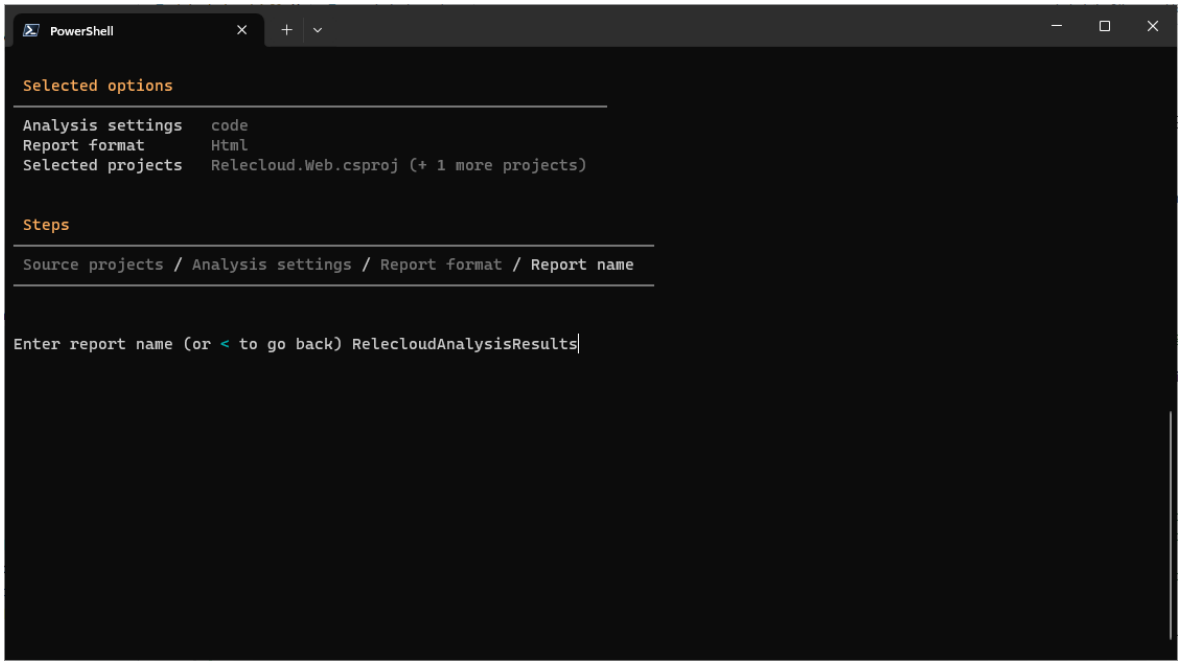


> **Note**
>
> The **source code and settings** option will only scan the source code in the projects you selected on the previous screen. The **Binary dependencies** option will scan any dependencies (such as NuGet packages) your projects rely on. You can expect to see many more issues identified when **binary dependencies** is selected. This option can create some "noise" since it will also identify potential compatibility issues of the binaries that are not necessarily apply to your application.

4. You'll then be prompted to generate a report with the results of the analysis. The output can be formatted as CSV, HTML, or JSON. Press **Enter**.

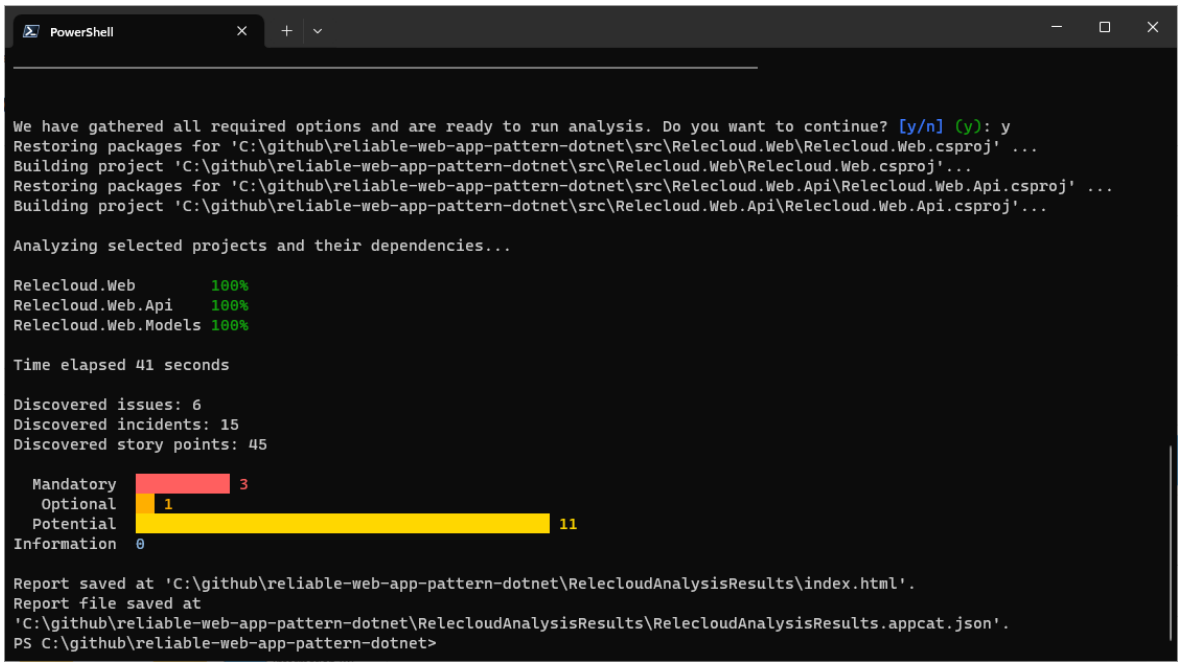


5. You'll be prompted for a report name. Input the name and press **Enter**.



6. Finally, you'll be asked whether you want to perform the scan. Press **y** to continue, or **n** to go back and change options.

7. Once the analysis completes, the report is saved, and a summary of the results are displayed.



Next Steps

Interpret the results

For information on how to interpret results, see Interpret the analysis results from the Azure Migrate application and code assessment for .NET.

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see our contributor guide.

.NET feedback

The .NET documentation is open source. Provide feedback here.

- Open a documentation issue
- Provide product feedback

Interpret the analysis results

Article • 11/14/2023

Both CLI tool and Visual Studio allow you to create HTML, CSV and JSON reports. This section describes how to interpret these reports.

For the purposes of this document we're going to use the HTML report.

Dashboard view

The report presents its results in a dashboard format.

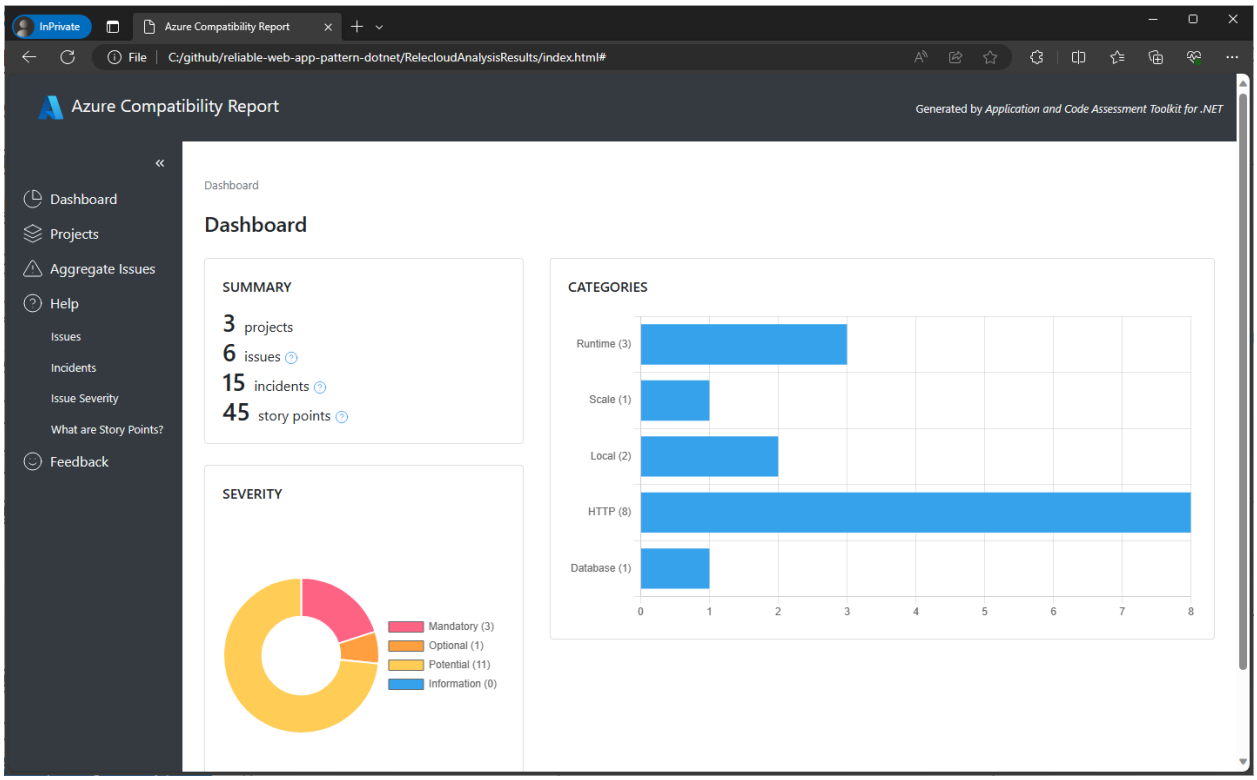
The main dashboard shows a Summary section with the results of the scan, Severity graph and Categories of the issues and incidents.

The **Summary** section of the dashboard contains several terms that are worth defining as you'll see them in other screens.

- **Projects**: the number of projects scanned.
- **Issues**: an incompatibility with Azure or a potential problem.
- **Incidents**: an occurrence of the issue in your code. For example, an issue could be a call to a database that is not accessible from Azure, and the incidents are the times when you make that call in your code. That way one issue might have many incidents (implementation) across your code.
- **Story points**: the estimated effort to fix all of the issues found. This is a relative measure of effort and is not meant to be an exact estimate.

Here are the issue severity classifications:

- **Mandatory** - the issue has to be resolved for the migration to be successful.
- **Optional** - the issue discovered is a real issue and fixing it could improve the application work after migration, however it is not blocking.
- **Potential** - it could or could not be a blocking problem depending on the specifics of your app and the migration scenario, so the tool draws your attention and suggests what checks could be made to ensure the application will work in Azure.
- **Informational** - the issue was raised only for informational purpose and is not required to be resolved.

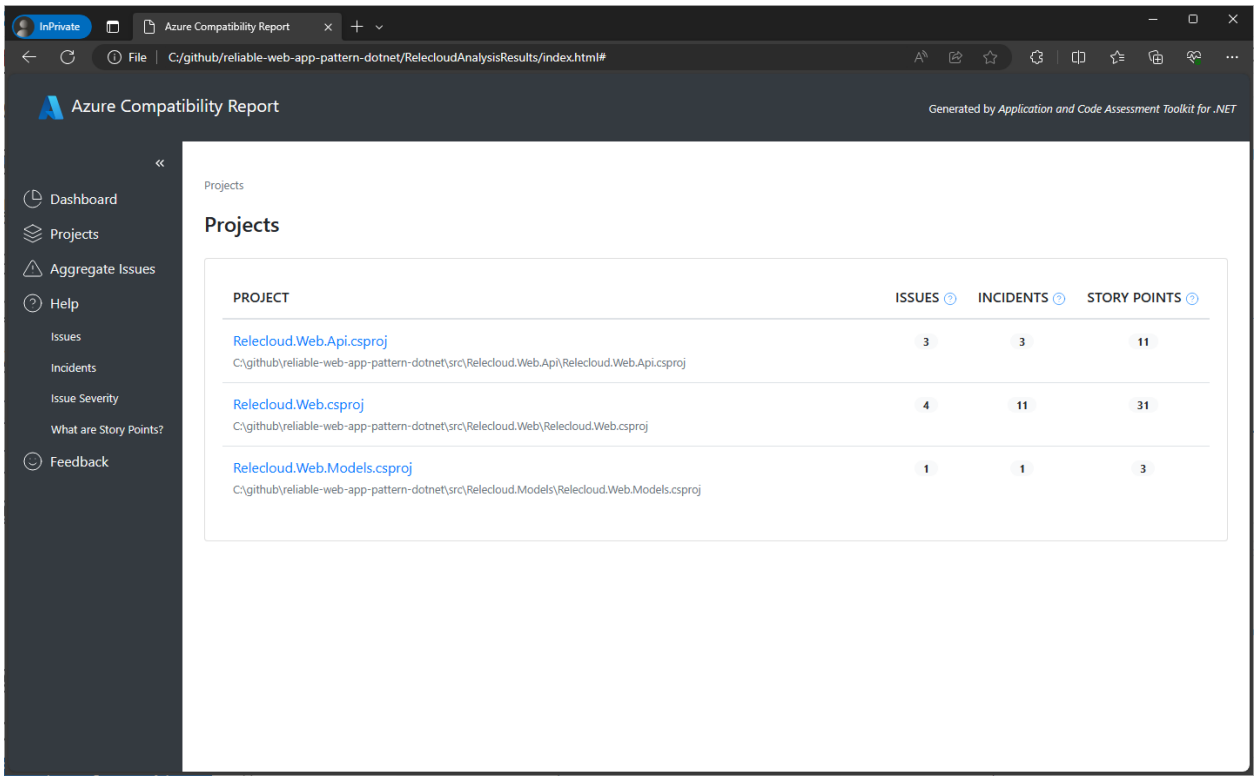


Categories section displays a graph with issues grouped by different categories such as HTTP, database, scaling, and so on.

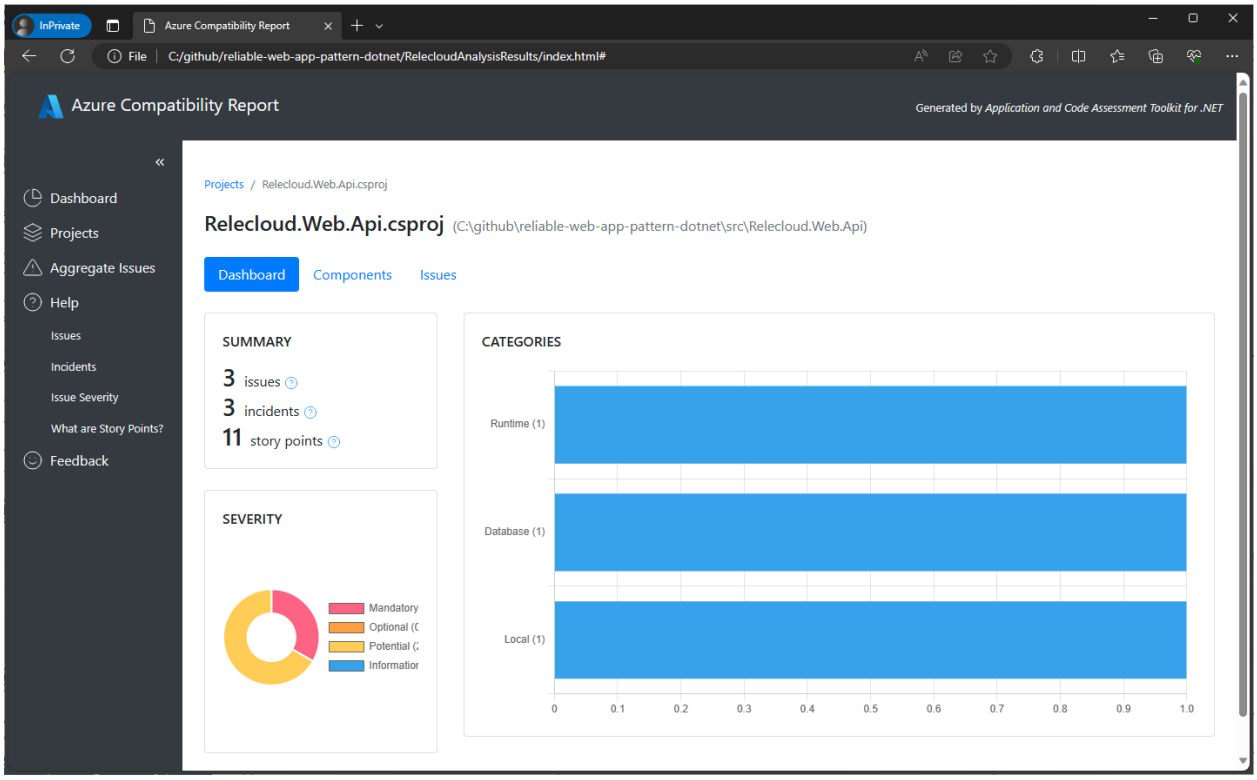
- **Projects**: the number of projects scanned.
- **Issues**: the number of unique encounters of a rule that may need to be addressed.
- **Incidents**: the total number of occurences of all issues found.
- **Story points**: the estimated effort to fix all of the issues found. This is a relative measure of effort and is not meant to be an exact estimate.

Projects view

Click on the **Projects** link below the **Dashboard** on the left side of the report to see the number of issues, incidents, and the estimated effort to fix those incidents by each project scanned.



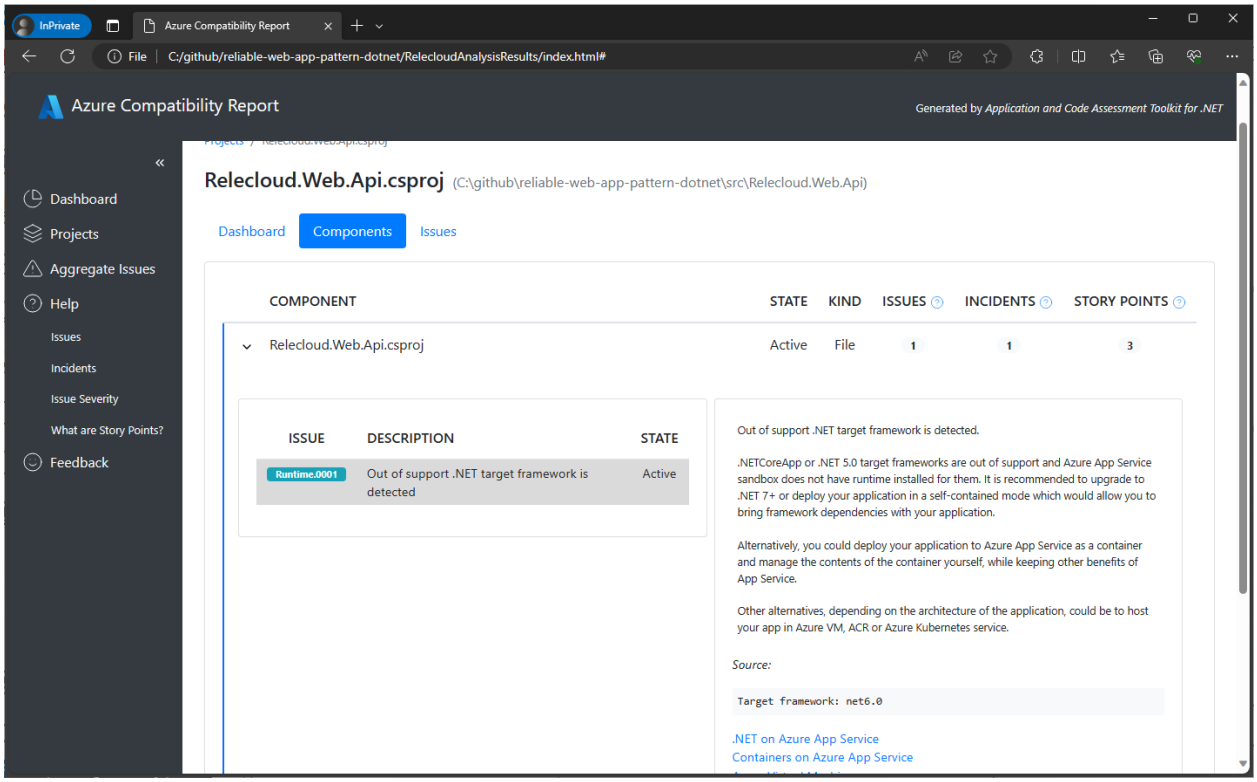
You can drill down to see the issues found in each project by clicking on the project name. This will show a screen similar to the overall dashboard but scoped to the selected project.



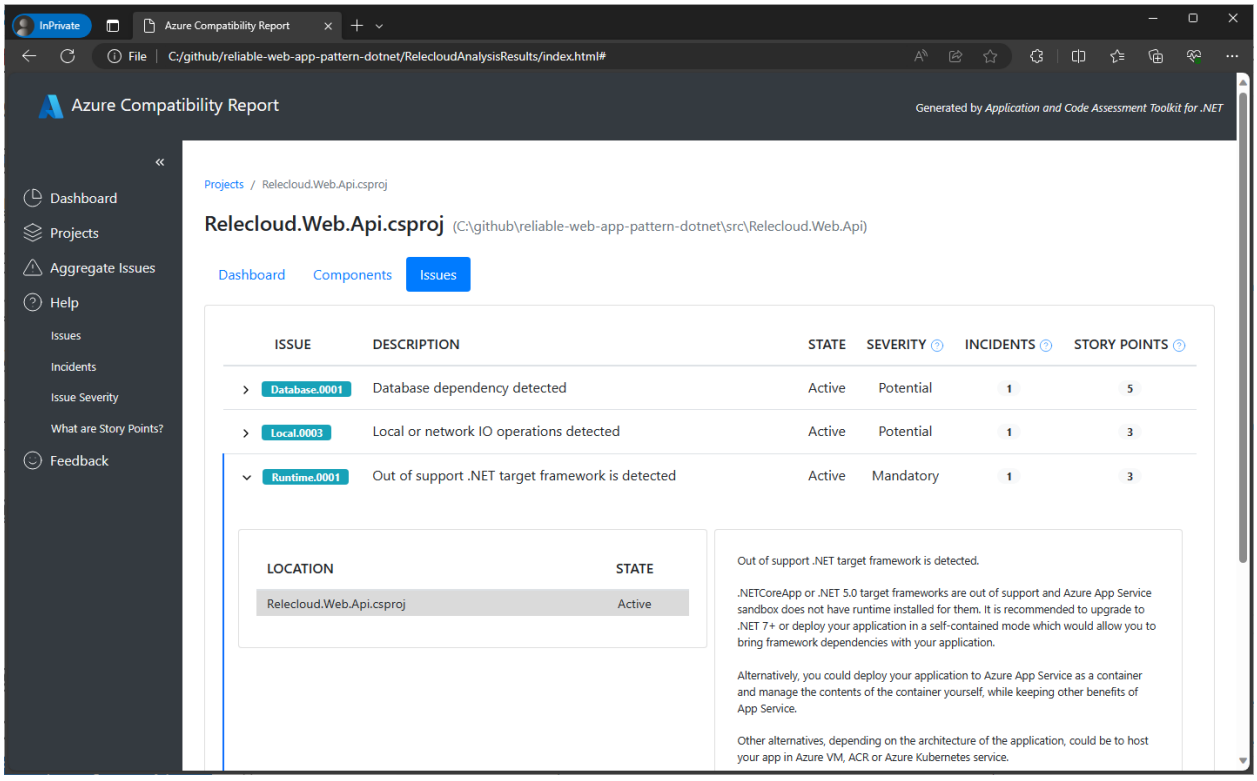
At the top of the project dashboard you'll find 3 tabs: **Dashboard**, **Components**, and **Issues**.

Click on the **Components** tab to see which files the incidents of the issues identified reside in. You can drill down into the file to see the issues that triggered the incident, a

description of the issue, the exact position in the code where the incidents exist, and an estimation of the effort it will take to fix the incidents.



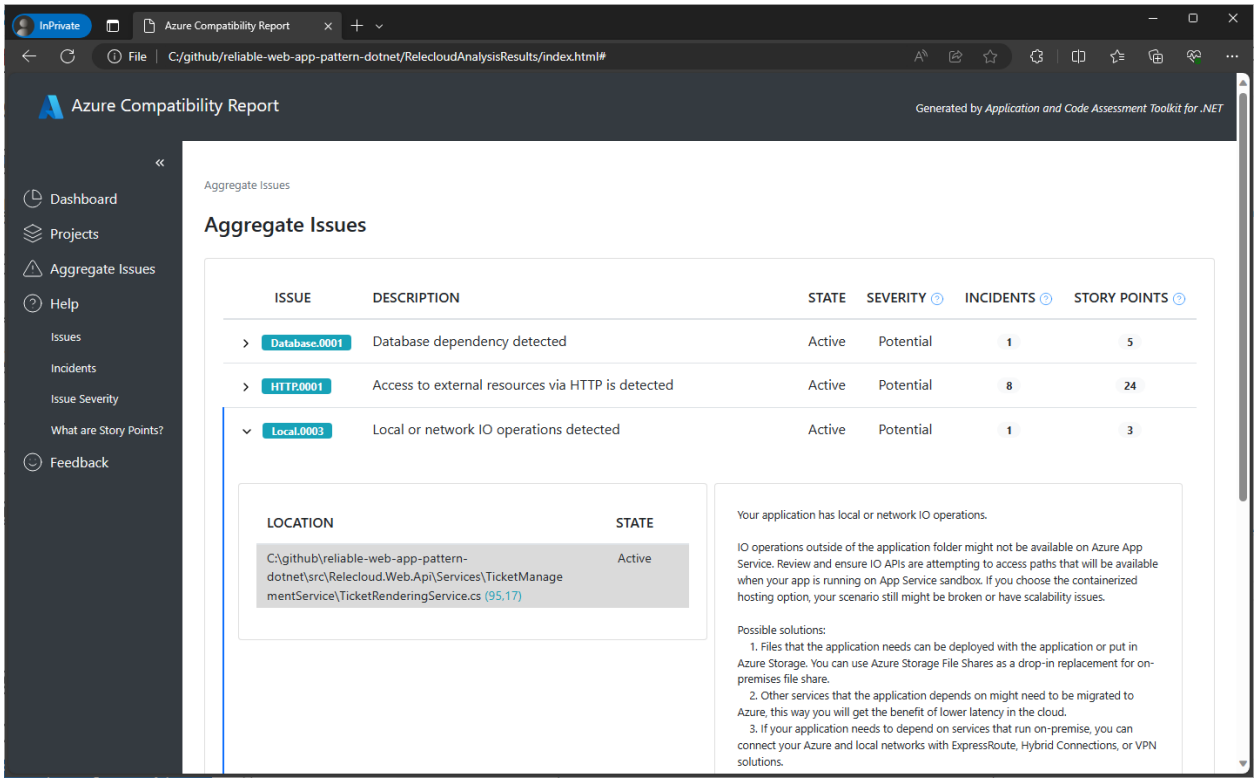
Finally, by clicking on the **Issues** tab, you can see the incidents organized by the issues which triggered them. You can drill down into the issues to see the exact file location that needs to be addressed and the effort to fix.



Aggregate Issues view

Click on the **Aggregate issues** link below the **Projects** on the left side of the report's screen to see the incidents organized by the issues that triggered them. These are all of the issues across all of the projects scanned, including the number of incidents and an estimated story points effort. You can drill down into each issue to see the exact files and locations that needs to be addressed and the effort to fix.

In the section on the right, you will find an explanation for the selected issue with suggestions on how to fix it or the verifications you should make to ensure your application will work properly in Azure. There are also links to the detailed documentation in the bottom part of that section.



Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see our contributor guide.

.NET feedback

The .NET documentation is open source. Provide feedback here.

- Open a documentation issue
- Provide product feedback

Choose the right Azure hosting option

Article • 12/19/2023

This article provides considerations and comparisons between the multiple choices you have in Azure when migrating your existing .NET Framework applications from on-premises to Azure.

The fundamental areas to consider when migrating existing .NET applications to Azure are:

1. Compute choices
2. Database choices
3. Networking and security considerations
4. Authentication and authorization considerations

Compute choices

When migrating existing .NET Framework applications to Azure you have multiple choices. However, since .NET Framework depends on Windows, the following choices are limited to Windows-based compute services.

The following table shows several comparisons and recommendations to help you choose the right compute migration path for your existing .NET application.

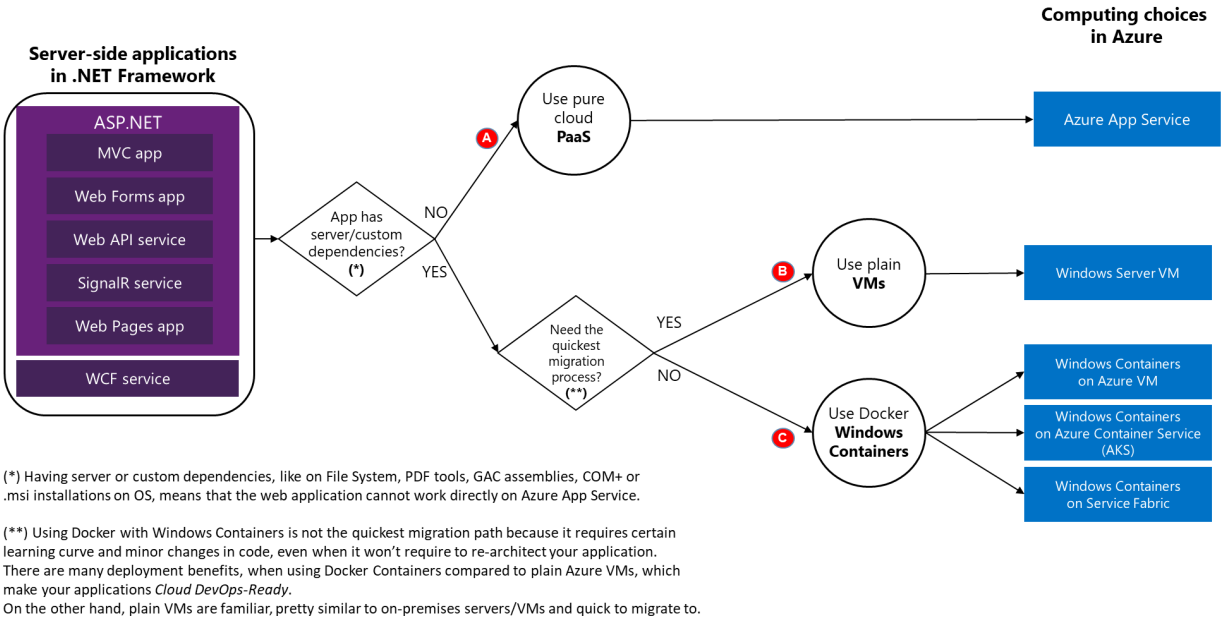
Expand table

	Azure VMs	Azure App Service	Windows Containers
When to use	• Application has strong dependencies on the server and local .msi installations. • You want the easiest application migration path	App has no dependencies on the server, it is just a clean ASP.NET web app (MVC, WebForm) or N-Tier app (Web API, WCf) accessing a database server.	• Application has dependencies on the original server but those dependencies can be included in the Docker Windows image.
Pros & benefits	• Easiest migration path • Familiar environment. Deployment	Ongoing PaaS maintenance, simplest way to manage and scale apps in Azure.	• Prepared for the future, Cloud DevOps-Ready with dependencies

	Azure VMs	Azure App Service	Windows Containers
	environment is a VM, so it's similar to on-premises servers.		included in the app's containers. • Almost no need to refactor .NET /C# code.
Cons	It is IaaS. Maintenance is costly. You have to manage the VM's infrastructure about networking, load-balancer, scale-out, IIS management, and so on.	• Not all apps are supported • Some apps might need to be refactored and even slightly rearchitected, so they support Azure App Service.	• Docker's skills learning curve • Some code and app configuration settings changes
Requirements	Windows Server VM with the same requirements than the app for on-premises	Azure App Service requirements specified in Readiness checks.	• Docker Engine - Enterprise for Windows Server 2019 or • Azure Container Service (AKS) (That is Kubernetes orchestrator) or • Azure Service Fabric orchestrator
How to migrate	See Migrate to Azure Virtual Machines	See Migrate Azure App Service	Follow considerations, scenarios, and walkthroughs explained in the Modernizing existing .NET apps with Azure and Windows Containers eBook

The following flowchart diagram shows a decision tree when planning a migration to Azure for your existing .NET Framework applications. If it's viable, try option A first, but option B is the easiest path to perform.

Azure compute decision tree for existing .NET Framework apps migration



(*) Having server or custom dependencies, like on File System, PDF tools, GAC assemblies, COM+ or .msi installations on OS, means that the web application cannot work directly on Azure App Service.

(**) Using Docker with Windows Containers is not the quickest migration path because it requires certain learning curve and minor changes in code, even when it won't require to re-architect your application. There are many deployment benefits, when using Docker Containers compared to plain Azure VMs, which make your applications *Cloud DevOps-Ready*.
On the other hand, plain VMs are familiar, pretty similar to on-premises servers/VMs and quick to migrate to.

Database choices

When migrating relational databases to Azure you have multiple choices. See Migrate your SQL Server database to Azure to help you choose the right database migration path for your existing .NET application.

Networking and security considerations

When deploying applications to a public cloud like Microsoft Azure, you might want to isolate and secure certain networks by creating network DMZs, such as a DMZ between Azure and on-premises or a DMZ between Azure and the Internet. DMZs can be implemented with Azure Virtual Network.

Azure Virtual networks enable you to:

- Build a hybrid infrastructure that you control
- Bring your own IP addresses and DNS servers
- Secure your connections with an IPsec VPN or ExpressRoute
- Get granular control over traffic between subnets
- Create sophisticated network topologies using virtual appliances
- Get an isolated and highly secure environment for your applications

To get started building your own virtual network, see the Azure Virtual Network documentation.

Authentication and authorization considerations when migrating to Azure

A top concern of any organization moving to the cloud is security. Most companies have invested a substantial amount of time, money, and engineering into designing and developing a security model, and it's important that they're able to leverage existing investments such as identity stores and single sign-on solutions.

Many existing enterprise B2E .NET applications running on-premises use Active Directory for authentication and identity management. Azure AD Connect enables you to integrate your on-premises directories with Azure Active Directory. To get started, see Integrate your on-premises directories with Azure Active Directory.

See Identity requirements for your hybrid identity solution for further planning related to Azure Active Directory.

Other authentication protocol choices are OAuth and OpenID, which are common in consumer-facing applications. When using autonomous identity databases, such as an ASP.NET Identity SQL database wrapped by IdentityServer4 using OAuth, no connectivity to on-premises databases or directories is usually required.

Next steps

Migrate an ASP.NET web application to Azure App Service

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see our contributor guide.

.NET feedback

.NET is an open source project. Select a link to provide feedback:

Open a documentation issue

Provide product feedback

Migrate your .NET web app or service to Azure App Service

Article • 07/30/2022

App Service is a fully managed compute platform service that's optimized for hosting scalable websites and web applications. This article provides information on how to lift-and-shift an existing application to Azure App Service, modifications to consider, and additional resources for moving to the cloud. Most ASP.NET websites (Webforms, MVC) and services (Web API, WCF) can move directly to Azure App Service with no changes. Some may need minor changes while others may need some refactoring.

Ready to get started? Publish your ASP.NET + SQL application to Azure App Service.

Considerations

On-premises resources (including SQL Server)

Verify access to on-premises resources as these may need to be migrated or changed. The following are options for mitigating access to on-premises resources:

- Create a VPN connecting App Service to on-premises resources using Azure Virtual Networks.
- Securely expose on-premises services to the cloud without firewall changes using Azure Relay.
- Migrate dependencies such as a SQL database to Azure.
- Use platform-as-a-service offerings in the cloud to reduce dependencies. For example, rather than connect to an on-premises mail server, consider using SendGrid.

Port Bindings

Azure App Service supports port 80 for HTTP and port 443 for HTTPS traffic.

For WCF, the following bindings are supported:

Binding	Notes
`BasicHttp`	
`WSHttp`	

Binding	Notes
`WSDualHttpBinding`	Web socket support must be enabled.
`NetHttpBinding`	Web socket support must be enabled for duplex contracts.
`NetHttpsBinding`	Web socket support must be enabled for duplex contracts.
`BasicHttpContextBinding`	
`WebHttpBinding`	
`WSHttpContextBinding`	

Authentication

Azure App Service supports anonymous authentication by default and Forms authentication when intended. Windows authentication can be used by integrating with Azure Active Directory and ADFS only. Learn more about how to integrate your on-premises directories with Azure Active Directory.

Assemblies in the GAC (Global Assembly Cache)

This isn't supported. Consider copying required assemblies to the app's *\bin* folder. Custom *.msi* files installed on the server (for example, PDF generators) cannot be used.

IIS settings

Everything traditionally configured via applicationHost.config in your application can now be configured through the Azure portal. This applies to AppPool bitness, enable/disable WebSockets, managed pipeline version, .NET Framework version (2.0/4.0), and so on. To modify your application settings, navigate to the Azure portal, open the blade for your web app, and then select the **Application Settings** tab.

IIS5 Compatibility Mode

IIS5 Compatibility Mode is not supported. In Azure App Service, each web app and all of the applications under it run in the same worker process with a specific set of application pools.

IIS7+ schema compliance

Some elements and attributes are not defined in the Azure App Service IIS schema. If you encounter issues, consider using XDT transforms.

Single application pool per site

In Azure App Service, each web app and all of the applications under it run in the same application pool. Consider establishing a single application pool with common settings or creating a separate web app for each application.

COM and COM+ components

Azure App Service does not allow the registration of COM components on the platform. If your app makes use of any COM components, these need to be rewritten in managed code and deployed with the site or application.

Physical directories

Azure App Service does not allow physical drive access. You may need to use Azure Files to access files via SMB. Azure Blob Storage can store files for access via HTTPS.

ISAPI filters

Azure App Service can support the use of ISAPI Filters, however, the ISAPI DLL must be deployed with your site and registered via web.config.

HTTPS bindings and SSL

HTTPS bindings are not migrated, nor are the SSL certificates associated with your web sites. SSL certificates can be manually uploaded after site migration is completed, however.

SharePoint and FrontPage

SharePoint and FrontPage Server Extensions (FPSE) are not supported.

Web site size

Free sites have a size limit of 1 GB of content. If your site is greater than 1 GB, you must upgrade to a paid SKU. See App Service pricing.

Database size

For SQL Server databases, please check the current SQL Database pricing.

Azure Active Directory (AAD) integration

AAD does not work with free apps. To use AAD, you must upgrade the app SKU. See App Service pricing.

Monitoring and diagnostics

Your current on-premises solutions for monitoring and diagnostics are unlikely to work in the cloud. However, Azure provides tools for logging, monitoring, and diagnostics so that you can identify and debug issues with web apps. You can easily enable diagnostics for your web app in its configuration, and you can view the logs recorded in Azure Application Insights. Learn more about enabling diagnostics logging for web apps.

Connection strings and application settings

Consider using Azure KeyVault, a service that securely stores sensitive information used in your application. Alternatively, you can store this data as an App Service setting.

DNS

You may need to update DNS configurations based on the requirements of your application. These DNS settings can be configured in the App Service custom domain settings.

Azure App Service with Windows Containers

If your app cannot be migrated directly to App Service, consider App Service using Windows Containers, which enables usage of the GAC, COM components, MSIs, full access to .NET FX APIs, DirectX, and more.

See also

- How to determine if your app qualifies for App Service
- Moving your database to the cloud
- Azure web app sandbox details and restrictions

Migrate an ASP.NET Web application to an Azure Virtual Machine

Article • 09/15/2021

This document provides an overview of how to migrate an ASP.NET web application from on-premises to an Azure Virtual Machine.

Quickstart

Learn how to create a virtual machine and publish your app to it: Publish to an Azure VM

Get Started

These tutorials demonstrate the steps to create (or migrate) a virtual machine, publish your web application to it, and other tasks that may be required to support your application in Azure.

- Create a virtual machine for your ASP.NET application in Azure using one of the following options:
  - Create a new virtual machine for ASP.NET Applications
  - Migrate an existing on-premises VMWare virtual machine
  - Migrate an existing on-premises Hyper-V virtual machine
- Publish a cloud service using Visual Studio
- Create a secure virtual network for your VMs
- Create a CI/CD pipeline for your application
- Move to a VM scale set for high availability and scalability

Considerations

Benefits

Virtual machines offer the easiest path to migrate an application from on-premises to the cloud. They enable you to replicate the same environment your application uses on-premises, while removing the need to maintain your own data centers. Virtual Machine Scale Sets provide high availability and scalability for applications running in Virtual Machines.

Virtual Machine Size

Choose the virtual machine size and type that is best optimized for your workload. For more information, see Sizes for Windows virtual machines in Azure.

Maintenance

Just like an on-premises machine, you are responsible for maintaining and updating the virtual machine*. If your application can run in a Platform as a Service (PaaS) environment such as Azure App Service or in a container, that will remove this need.

**Automatic OS upgrades for virtual machine scale sets* is currently available as a Preview service.*

Virtual Networks

Azure Virtual Networks enable you to:

- Build a hybrid infrastructure that you control
- Bring your own IP addresses and DNS servers
- Create an isolated and highly secure environment for your applications
- Connect your VM to your on-premises network using one of several connectivity options
- Integrate your virtual machine into your on-premises network using ExpressRoute

To get started, see the Virtual Network documentation

Active Directory

Many applications use Active Directory for authentication and identity management.

- Azure AD Connect enables you to integrate your on-premises directories with Azure Active Directory. To get started, see Integrate your on-premises directories with Azure Active Directory.
- Alternatively, ExpressRoute enables your application to access your on-premises Active Directory.

SQL Databases

If your application is using an on-premises database, your app will not be able to talk to it by default. You can either:

- Configure a hybrid network that enables your application to access your database running on-premises.
- Migrate your database to the Azure. For more information, see Migrate your SQL Server database to Azure.

High Availability and Scalability

Virtual Machine Scale Sets

You want to make sure that your application is highly available and can scale, migrate your VM image to an Azure Virtual Machine Scale Set to improve the availability and scalability of your application. VM Scale Sets provide the ability to use an existing VM you've already configured or set up a build pipeline to build an image with your application.

To get started, see Deploy your application on virtual machine scale sets.

Centralized Logging

When running your application across multiple instances, consider storing your logs in a centralized location such as Azure Storage.

Next steps

Migrate a SQL Server database to Azure

Migrate a SQL Server database to Azure

Article • 05/12/2022

This article provides a brief outline of two options for migrating a SQL Server database to Azure. Azure has three primary options for migrating a production SQL Server database. This article focuses on the following two options:

1. SQL Server on Azure VMs: A SQL Server instance installed and hosted on a Windows Virtual Machine running in Azure, also known as Infrastructure as a Service (IaaS).
2. Azure SQL Database: A fully managed SQL database Azure service, also known as Platform as a Service (PaaS).

Both come with pros and cons that you will need to evaluate before migrating. The third option is Azure SQL Database managed instances.

Get started

The following migration guides will be useful, depending on which service you use:

- Migrate a SQL Server database to SQL Server in an Azure VM
- Migrate your SQL Server database to Azure SQL Database

Additionally, the following links to conceptual content will help you understand VMs better:

- High availability and disaster recovery for SQL Server in Azure Virtual Machines
- Performance best practices for SQL Server in Azure Virtual Machines
- Application Patterns and Development Strategies for SQL Server in Azure Virtual Machines

And the following links will help you understand Azure SQL Database better:

- Create and manage Azure SQL Database servers and databases
- Database Transaction Units (DTUs) and elastic Database Transaction Units (eDTUs)
- Azure SQL Database resource limits

Choosing IaaS or PaaS

When evaluating where to migrate your database, determine if IaaS or PaaS is more appropriate for you.

Choose SQL Server in Azure VMs if:

- You are looking to "lift and shift" your database and applications with minimal to no changes.
- You prefer having full control over your database server and the VM it runs on.
- You already have SQL Server and Windows Server licenses that you intend to use.

Choose Azure SQL Database if:

- You are looking to modernize your applications and are migrating to use other PaaS services in Azure.
- You do not wish to manage your database server and the VM it runs on.
- You do not have SQL Server or Windows Server licenses, or you intend to let licenses you have expire.

The following table describes differences between each service based on a set of scenarios.

Scenario	SQL Server in Azure VMs	Azure SQL Database
Migration	Requires minimal changes to your database.	May require changes to your database if you use features unavailable in Azure SQL, as determined by the Data Migration Assistant, or if you have other dependencies such as locally installed executables.
Managing availability, recovery, and upgrades	Availability and recovery are configured manually. Upgrades can be automated with VM Scale Sets.	Automatically managed for you.
Underlying OS configuration	Manual configuration.	Automatically managed for you.
Managing database size	Supports up to 256 TB of storage per SQL Server instance.	Supports 8 TB of storage before needing a horizontal partition.
Managing costs	You must manage SQL Server license costs, Windows Server license costs, and VM costs (based on cores, RAM, and storage).	You must manage service costs (based on eDTUs or DTUs, storage, and number of databases if using an elastic pool). You must also manage the cost of any SLA.

To learn more about the differences between the two, see Choose the right deployment option in Azure SQL.

FAQ

- **Can I still use tools such as SQL Server Management Studio and SQL Server Reporting Services (SSRS) with SQL Server in Azure VMs or Azure SQL Database?**

  Yes. All Microsoft SQL tooling works with both services. SSRS is not part of Azure SQL Database, though, and it's recommended that you run it in an Azure VM and then point it to your database instance.

- **I want to go PaaS but I'm not sure if my database is compatible. Are there tools to help?**

  Yes. The Data Migration Assistant is a tool that is used as a part of migrating to Azure SQL Database. The Azure Database Migration Service is a preview service that you can use for either IaaS or PaaS.

- **Can I estimate costs?**

  Yes. The Azure Pricing Calculator can be used for estimating costs for all Azure services, including VMs and database services.

Next steps

Choose the right Azure hosting option

Develop .NET apps that use Azure AI services

Article • 03/13/2024

This article provides documentation, samples, and other resources for learning how to develop applications that use Azure OpenAI Service and other Azure AI Services.

Azure AI reference templates

Azure AI reference templates provide you with well-maintained, easy to deploy reference implementations. These ensure a high-quality starting point for your intelligent applications. The end-to-end solutions provide popular, comprehensive reference applications. The building blocks are smaller-scale samples that focus on specific scenarios and tasks.

End-to-end solutions

Expand table

Link	Description
Get started with the .NET enterprise chat sample using RAG	An article that walks you through deploying and using the Enterprise chat app sample for .NET. This sample is a complete end-to-end solution demonstrating the Retrieval-Augmented Generation (RAG) pattern running in Azure, using Azure AI Search for retrieval and Azure OpenAI large language models to power ChatGPT-style and Q&A experiences.
Build an AI assistant using RAG	This sample is a complete end-to-end solution demonstrating how to design and implement a Q&A AI assistant, which uses the Embeddings API and Completions API in Azure OpenAI Service, as well as the vector database in Azure Cosmos DB.

- Demo video

Building blocks

Expand table

Link	Description
Build a chat app with Azure OpenAI (Python)	A simple Python Quart app that streams responses from ChatGPT to an HTML/JS frontend using JSON Lines over a ReadableStream. (The Python code is provided as a reference and could be adapted to .NET.)
Build a LangChain with Azure OpenAI (Python)	A sample shows how to take a human prompt as HTTP Get or Post input, calculates the completions using chains of human input and templates. This is a starting point that can be used for more sophisticated chains. (The Python code is provided as a reference and could be adapted to .NET.)
Build a ChatGPT Plugin with Azure Container Apps (Python)	A sample for creating ChatGPT Plugin using GitHub Codespaces, VS Code, and Azure. The sample includes templates to deploy the plugin to Azure Container Apps using the Azure Developer CLI. (The Python code is provided as a reference and could be adapted to .NET.)
Azure AI .NET Template Gallery	For the full list of Azure AI templates, visit our gallery. All app templates in our gallery can be spun up and deployed using a single command: *azd up*.
Smart load balancing with Azure Container Apps	This solution is built using the high-performance YARP C# reverse-proxy framework from Microsoft. However, you don't need to understand C# to use it, you can just build the provided Docker image. This is an alternative solution to the API Management OpenAI smart load balancer, with the same logic.
Smart load balancing with Azure API Management	The enterprise solution shows how to create an Azure API Management Policy to seamlessly expose a single endpoint to your applications while keeping an efficient logic to consume two or more OpenAI or any API backends based on availability and priority.

Azure OpenAI

End-to-end solutions

Expand table

Link	Description
Get started with the .NET enterprise chat sample using RAG	An article that walks you through deploying and using the Enterprise chat app sample for .NET. This sample is a complete end-to-end solution demonstrating the Retrieval-Augmented Generation (RAG) pattern running in Azure, using Azure AI Search for retrieval and Azure OpenAI large language models to power ChatGPT-style and Q&A experiences.

Building blocks

Expand table

Link	Description
Vector Similarity Search with Azure Cache for Redis Enterprise (Python)	An article that walks you through using Azure Cache for Redis as a backend vector store for RAG scenarios. (The Python code is provided as a reference and could be adapted to .NET.)
OpenAI solutions with your own data using PostgreSQL (Python)	An article discussing how Azure Database for PostgreSQL Flexible Server and Azure Cosmos DB for PostgreSQL supports the pgvector extension, along with an overview, scenarios, etc. (The Python code is provided as a reference and could be adapted to .NET.)

SDKs and other samples/guidance

Expand table

Link	Description
Azure OpenAI SDK for .NET	The GitHub source version of the Azure OpenAI client library for .NET is an adaptation of OpenAI's REST APIs that provides an idiomatic interface and rich integration with the rest of the Azure SDK ecosystem. It can connect to Azure OpenAI resources or to the non-Azure OpenAI inference endpoint, making it a great choice for even non-Azure OpenAI development.
Azure OpenAI SDK Releases	Links to all Azure OpenAI SDK library packages, including links for .NET, Java, JavaScript and Go.
Azure.AI.OpenAI NuGet package	The NuGet version of the Azure OpenAI client library for .NET.
Get started using GPT-35-Turbo and GPT-4	An article that walks you through creating a chat completion sample.
Completions	A collection of 10 samples that demonstrate how to use the Azure OpenAI client library for .NET to chat, stream replies, use your own data, transcribe/translate audio, generate images, etc.
Streaming Chat Completions	A deep link to the samples demonstrating streaming completions.
OpenAI with Microsoft Entra ID Role based access control	A look at authentication using Microsoft Entra ID.

Link	Description
OpenAI with Managed Identities	An article with more complex security scenarios that require Azure role-based access control (Azure RBAC). This document covers how to authenticate to your OpenAI resource using Microsoft Entra ID.
More samples	A collection of OpenAI samples written in .NET.
More guidance	The hub page for Azure OpenAI Service documentation.

Other Azure AI services

End-to-end solutions

Expand table

Link	Description
Captioning and Call Center Transcription	A repo containing samples for captions and transcriptions in a call center scenario.
Use Form Recognizer to automate a paper based process using the New patient registration with Form Recognizer workshop. (Code)	A complete walkthrough of an Azure AI Document Intelligence scenario in a workshop format.

Building blocks

Expand table

Link	Description
Use Speech to converse with OpenAI	An article detailing how to use Azure AI Speech to converse with Azure OpenAI Service. The text recognized by the Speech service is sent to Azure OpenAI. Speech service then synthesizes the text response from Azure OpenAI.
Translate documents from and into more than 100 different languages	An article showing how to translate local files or network files in many different formats, to more than 100 different languages. Supported formats include HTML, PDF, all Office document formats, Markdown, MHTML, Outlook, MSG, XLIFF, CSV, TSV and plain text.

SDKs and samples/guidance

Expand table

Link	Description
Integrate Speech into your apps with Speech SDK Samples	A repo of samples for the Azure Cognitive Services Speech SDK. Links to samples for speech recognition, translation, speech synthesis, and more.
Azure AI Document Intelligence SDK	Azure AI Document Intelligence (formerly Form Recognizer) is a cloud service that uses machine learning to analyze text and structured data from documents. The Document Intelligence software development kit (SDK) is a set of libraries and tools that enable you to easily integrate Document Intelligence models and capabilities into your applications.
Extract structured data from forms, receipts, invoices, and cards using Form Recognizer in .NET	A repo of samples for the Azure.AI.FormRecognizer client library.
Extract, classify, and understand text within documents using Text Analytics in .NET	The client Library for Text Analytics. This is part of the Azure AI Language service, which provides Natural Language Processing (NLP) features for understanding and analyzing text.
Document Translation in .NET	A quickstart article that details how to use Document Translation to translate a source document into a target language while preserving structure and text formatting.
Question Answering in .NET	A quickstart article to get an answer (and confidence score) from a body of text that you send along with your question.
Conversational Language Understanding in .NET	The client library for Conversational Language Understanding (CLU), a cloud-based conversational AI service, which can extract intents and entities in conversations and acts like an orchestrator to select the best candidate to analyze conversations to get best response from apps like Qna, Luis, and Conversation App.
Analyze images	Sample code and setup documents for the Microsoft Azure AI Image Analysis SDK

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For

.NET feedback

.NET is an open source project. Select a link to provide feedback:

Open a documentation issue

more information, see our contributor guide.

Provide product feedback

Summarize text using Azure AI chat app with .NET

Article • 03/27/2024

Get started with Semantic Kernel by creating a simple .NET 8 console chat application. The application will run locally and use the OpenAI `gpt-35-turbo` model deployed into an Azure OpenAI account. Follow these steps to provision Azure OpenAI and learn how to use Semantic Kernel.

Prerequisites

- .NET 8.0 SDK - Install the .NET 8.0 SDK
- An Azure subscription - Create one for free
- Azure Developer CLI - Install or update the Azure Developer CLI
- Access to Azure OpenAI service.

Deploy the Azure resources

Ensure that you follow the Prerequisites to have access to Azure OpenAI Service as well as the Azure Developer CLI, and then follow the following guide to set started with the sample application.

1. Clone the repository: dotnet/ai-samples
2. From a terminal or command prompt, navigate to the *quickstarts* directory.
3. This provisions the Azure OpenAI resources. It may take several minutes to create the Azure OpenAI service and deploy the model.

Azure Developer CLI

```
azd up
```

> **Note**
>
> If you already have an Azure OpenAI service available, you can skip the deployment and use that value in the *Program.cs*, preferably from an `IConfiguration`.

Troubleshoot

On Windows, you might get the following error message after running `azd up`:

> *postprovision.ps1 is not digitally signed. The script will not execute on the system*

The script **postprovision.ps1** is executed to set the .NET user secrets used in the application. To avoid this error, run the following PowerShell command:

PowerShell

```
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
```

Then re-run the `azd up` command.

Trying Hiking Benefits Summary sample

1. From a terminal or command prompt, navigate to the `semantic-kernel\01-HikeBenefitsSummary` directory.
2. It's now time to try the console application. Type in the following to run the app:

   .NET CLI

   ```
   dotnet run
   ```

   If you get an error message the Azure OpenAI resources may not have finished deploying. Wait a couple of minutes and try again.

Understanding the code

Our application uses the `Microsoft.SemanticKernel` package, which is available on NuGet, to send and receive requests to an Azure OpenAI service deployed in Azure.

The entire application is contained within the **Program.cs** file. The first several lines of code loads up secrets and configuration values that were set in the `dotnet user-secrets` for you during the application provisioning.

C#

```
// == Retrieve the local secrets saved during the Azure deployment
==========
```

```
var config = new ConfigurationBuilder()
    .AddUserSecrets<Program>()
    .Build();
string endpoint = config["AZURE_OPENAI_ENDPOINT"];
string deployment = config["AZURE_OPENAI_GPT_NAME"];
string key = config["AZURE_OPENAI_KEY"];

// Create a Kernel containing the Azure OpenAI Chat Completion Service
Kernel kernel = Kernel.CreateBuilder()
    .AddAzureOpenAIChatCompletion(deployment, endpoint, key)
    .Build();
```

The `Kernel` class facilitates the requests and responses with the help of `AddAzureOpenAIChatCompletion` service.

C#

```
Kernel kernel = Kernel.CreateBuilder()
    .AddAzureOpenAIChatCompletion(deployment, endpoint, key)
    .Build();
```

Once the `Kernel` is created, we read the contents of the file `benefits.md` and create a `prompt` to ask the the model to summarize that text.

C#

```
// Create and print out the prompt
string prompt = $"""
    Please summarize the the following text in 20 words or less:
    {File.ReadAllText("benefits.md")}
    """;
Console.WriteLine($"user >>> {prompt}");
```

To have the model generate a response based off `prompt`, use the `InvokePromptAsync` function.

C#

```
// Submit the prompt and print out the response
string response = await kernel.InvokePromptAsync<string>(prompt, new(new
OpenAIPromptExecutionSettings() { MaxTokens = 400 }));
Console.WriteLine($"assistant >>> {response}");
```

Customize the text content of the file or the length of the summary to see the differences in the responses.

Clean up resources

When you no longer need the sample application or resources, remove the corresponding deployment and all resources.

Azure Developer CLI

```
azd down
```

Next steps

- Quickstart - Build an Azure AI chat app with .NET
- Generate text and conversations with .NET and Azure OpenAI Completions

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see our contributor guide.

.NET feedback

.NET is an open source project. Select a link to provide feedback:

Open a documentation issue

Provide product feedback

Build an Azure AI chat app with .NET

Article • 03/27/2024

Get started with Semantic Kernel by creating a simple .NET 8 console chat application. The application will run locally and use the OpenAI `gpt-35-turbo` model deployed into an Azure OpenAI account. Follow these steps to provision Azure OpenAI and learn how to use Semantic Kernel.

Prerequisites

- .NET 8.0 SDK - Install the .NET 8.0 SDK
- An Azure subscription - Create one for free
- Azure Developer CLI - Install or update the Azure Developer CLI
- Access to Azure OpenAI service.

Deploy the Azure resources

Ensure that you follow the Prerequisites to have access to Azure OpenAI Service as well as the Azure Developer CLI, and then follow the following guide to set started with the sample application.

1. Clone the repository: dotnet/ai-samples
2. From a terminal or command prompt, navigate to the *quickstarts* directory.
3. This provisions the Azure OpenAI resources. It may take several minutes to create the Azure OpenAI service and deploy the model.

   Azure Developer CLI

   ```
   azd up
   ```

> **Note**
>
> If you already have an Azure OpenAI service available, you can skip the deployment and use that value in the *Program.cs*, preferably from an `IConfiguration`.

Troubleshoot

On Windows, you might get the following error message after running `azd up`:

> *postprovision.ps1 is not digitally signed. The script will not execute on the system*

The script **postprovision.ps1** is executed to set the .NET user secrets used in the application. To avoid this error, run the following PowerShell command:

PowerShell

```
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
```

Then re-run the `azd up` command.

Trying HikerAI sample

1. From a terminal or command prompt, navigate to the `semantic-kernel\02-HikerAI` directory.
2. It's now time to try the console application. Type in the following to run the app:

   .NET CLI

   ```
   dotnet run
   ```

   If you get an error message the Azure OpenAI resources may not have finished deploying. Wait a couple of minutes and try again.

Understanding the code

Our application uses the `Microsoft.SemanticKernel` package, which is available on NuGet, to send and receive requests to an Azure OpenAI service deployed in Azure.

The entire application is contained within the **Program.cs** file. The first several lines of code loads up secrets and configuration values that were set in the `dotnet user-secrets` for you during the application provisioning.

C#

```
// == Retrieve the local secrets saved during the Azure deployment
==========
var config = new ConfigurationBuilder().AddUserSecrets<Program>().Build();
string endpoint = config["AZURE_OPENAI_ENDPOINT"];
```

```
string deployment = config["AZURE_OPENAI_GPT_NAME"];
string key = config["AZURE_OPENAI_KEY"];
```

The `AzureOpenAIChatCompletionService` service facilitates the requests and responses.

C#

```
// == Create the Azure OpenAI Chat Completion Service  ==========
AzureOpenAIChatCompletionService service = new(deployment, endpoint, key);
```

Once the `AzureOpenAIChatCompletionService` service is created, we provide more context to the model by adding a system prompt. This instructs the model how you'd like it to act during the conversation.

C#

```
// Start the conversation with context for the AI model
ChatHistory chatHistory = new("""
    You are a hiking enthusiast who helps people discover fun hikes in their
area. You are upbeat and friendly.
    You introduce yourself when first saying hello. When helping people out,
you always ask them
    for this information to inform the hiking recommendation you provide:

    1. Where they are located
    2. What hiking intensity they are looking for

    You will then provide three suggestions for nearby hikes that vary in
length after you get that information.
    You will also share an interesting fact about the local nature on the
hikes when making a recommendation.
    """);
```

Then you can add a user message to the model by using the `AddUserMessage` function.

To have the model generate a response based off the system prompt and the user request, use the `GetChatMessageContentAsync` function.

C#

```


// Add user message to chat history
chatHistory.AddUserMessage("Hi! Apparently you can help me find a hike that
I will like?");

// Print User Message to console
Console.WriteLine($"{chatHistory.Last().Role} >>>
{chatHistory.Last().Content}");
```

```
// Get response
var response = await service.GetChatMessageContentAsync(chatHistory, new
OpenAIPromptExecutionSettings() { MaxTokens = 400 });
```

To maintain the chat history, make sure you add the response from the model.

C#

```
// Add response to chat history
chatHistory.Add(response);

// Print Response to console
Console.WriteLine($"{chatHistory.Last().Role} >>>
{chatHistory.Last().Content}");
```

Customize the system prompt and user message to see how the model responds to help you find a hike that you'll like.

Clean up resources

When you no longer need the sample application or resources, remove the corresponding deployment and all resources.

Azure Developer CLI

```
azd down
```

Next steps

- Quickstart - Get insight about your data from an .NET Azure AI chat app
- Generate text and conversations with .NET and Azure OpenAI Completions

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

Open a documentation issue

Provide product feedback

more information, see our contributor guide.

Get insight about your data from an .NET Azure AI chat app

Article • 03/27/2024

Get started with Semantic Kernel and the `gpt-35-turbo` model, from a simple .NET 8.0 console application. Use the AI model to get analytics and information about your previous hikes. It consists of a simple console application, running locally, that will read the file `hikes.md` and send request to an Azure OpenAI service deployed in your Azure subscription and provide the result in the console. Follow these steps to provision Azure OpenAI and learn how to use Semantic Kernel.

Prerequisites

- .NET 8.0 SDK - Install the .NET 8.0 SDK
- An Azure subscription - Create one for free
- Azure Developer CLI - Install or update the Azure Developer CLI
- Access to Azure OpenAI service.

Deploy the Azure resources

Ensure that you follow the Prerequisites to have access to Azure OpenAI Service as well as the Azure Developer CLI, and then follow the following guide to set started with the sample application.

1. Clone the repository: dotnet/ai-samples
2. From a terminal or command prompt, navigate to the *quickstarts* directory.
3. This provisions the Azure OpenAI resources. It may take several minutes to create the Azure OpenAI service and deploy the model.

   Azure Developer CLI

   ```
   azd up
   ```

> **Note**

If you already have an Azure OpenAI service available, you can skip the deployment and use that value in the *Program.cs*, preferably from an `IConfiguration`.

Troubleshoot

On Windows, you might get the following error message after running `azd up`:

> *postprovision.ps1 is not digitally signed. The script will not execute on the system*

The script **postprovision.ps1** is executed to set the .NET user secrets used in the application. To avoid this error, run the following PowerShell command:

PowerShell

```powershell
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
```

Then re-run the `azd up` command.

Try "Chatting About My Previous Hikes" sample

1. From a terminal or command prompt, navigate to the `semantic-kernel\03-ChattingAboutMyHikes` directory.
2. It's now time to try the console application. Type in the following to run the app:

   .NET CLI

   ```dotnetcli
   dotnet run
   ```

   If you get an error message the Azure OpenAI resources may not have finished deploying. Wait a couple of minutes and try again.

Explore the code

Our application uses the `Microsoft.SemanticKernel` package, which is available on NuGet, to send and receive requests to an Azure OpenAI service deployed in Azure.

The entire application is contained within the **Program.cs** file. The first several lines of code loads up secrets and configuration values that were set in the `dotnet user-secrets` for you during the application provisioning.

C#

```
// == Retrieve the local secrets saved during the Azure deployment
==========
var config = new ConfigurationBuilder().AddUserSecrets<Program>().Build();
string endpoint = config["AZURE_OPENAI_ENDPOINT"];
string deployment = config["AZURE_OPENAI_GPT_NAME"];
string key = config["AZURE_OPENAI_KEY"];
```

The `AzureOpenAIChatCompletionService` service facilitates the requests and responses.

C#

```
// == Create the Azure OpenAI Chat Completion Service  ==========
AzureOpenAIChatCompletionService service = new(deployment, endpoint, key);
```

Once the `AzureOpenAIChatCompletionService` client is created, we read the content of the file `hikes.md` and use it to provide more context to the model by adding a system prompt. This instructs the model how you'd like it to act during the conversation.

C#

```
// Provide context for the AI model
ChatHistory chatHistory = new($"""
    You are upbeat and friendly. You introduce yourself when first saying
hello.
    Provide a short answer only based on the user hiking records below:

    {File.ReadAllText("hikes.md")}
    """);
Console.WriteLine($"{chatHistory.Last().Role} >>>
{chatHistory.Last().Content}");
```

Then you can add a user message to the model by using the `AddUserMessage` function.

To have the model generate a response based off the system prompt and the user request, use the `GetChatMessageContentAsync` function.

C#

```
// Start the conversation
chatHistory.AddUserMessage("Hi!");
Console.WriteLine($"{chatHistory.Last().Role} >>>
{chatHistory.Last().Content}");

chatHistory.Add(await service.GetChatMessageContentAsync(chatHistory, new
OpenAIPromptExecutionSettings() { MaxTokens = 400 }));
```

```
Console.WriteLine($"{chatHistory.Last().Role} >>>
{chatHistory.Last().Content}");
```

To maintain the chat history or context, make sure you add the response from the model to the `chatHistory`. It's time to make our user request about our data again using the `AddUserMessage` and `GetChatMessageContentAsync` function.

C#

```
// Continue the conversation with a question.
chatHistory.AddUserMessage("I would like to know the ratio of the hikes I've
done in Canada compared to other countries.");
Console.WriteLine($"{chatHistory.Last().Role} >>>
{chatHistory.Last().Content}");

chatHistory.Add(await service.GetChatMessageContentAsync(chatHistory, new
OpenAIPromptExecutionSettings() { MaxTokens = 400 }));
Console.WriteLine($"{chatHistory.Last().Role} >>>
{chatHistory.Last().Content}");
```

Customize the system prompt and change the request, asking for different questions (ex: How many times did you hiked when it was raining? How many times did you hiked in 2021? etc.) to see how the model responds.

Clean up resources

When you no longer need the sample application or resources, remove the corresponding deployment and all resources.

Azure Developer CLI

```
azd down
```

Next steps

- Quickstart - Generate images using Azure AI with .NET
- Generate text and conversations with .NET and Azure OpenAI Completions

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see our contributor guide.

.NET is an open source project. Select a link to provide feedback:

Open a documentation issue

Provide product feedback

Extend Azure AI using Tools and execute a local Function with .NET

Article • 03/27/2024

Get started with Semantic Kernel by creating a simple .NET 8 console chat application. The application will run locally and use the OpenAI `gpt-35-turbo` model deployed into an Azure OpenAI account, however using Tool to extend the model capabilities it will call a local function. Follow these steps to provision Azure OpenAI and learn how to use Semantic Kernel.

Prerequisites

- .NET 8.0 SDK - Install the .NET 8.0 SDK
- An Azure subscription - Create one for free
- Azure Developer CLI - Install or update the Azure Developer CLI
- Access to Azure OpenAI service.

Deploy the Azure resources

Ensure that you follow the Prerequisites to have access to Azure OpenAI Service as well as the Azure Developer CLI, and then follow the following guide to set started with the sample application.

1. Clone the repository: dotnet/ai-samples
2. From a terminal or command prompt, navigate to the *quickstarts* directory.
3. This provisions the Azure OpenAI resources. It may take several minutes to create the Azure OpenAI service and deploy the model.

   Azure Developer CLI

   ```
   azd up
   ```

> **Note**
>
> If you already have an Azure OpenAI service available, you can skip the deployment and use that value in the *Program.cs*, preferably from an `IConfiguration`.

Troubleshoot

On Windows, you might get the following error message after running `azd up`:

> *postprovision.ps1 is not digitally signed. The script will not execute on the system*

The script **postprovision.ps1** is executed to set the .NET user secrets used in the application. To avoid this error, run the following PowerShell command:

PowerShell

```powershell
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
```

Then re-run the `azd up` command.

Try HikerAI Pro sample

1. From a terminal or command prompt, navigate to the `semantic-kernel\04-HikerAIPro` directory.
2. It's now time to try the console application. Type in the following to run the app:

   .NET CLI

   ```dotnetcli
   dotnet run
   ```

   If you get an error message the Azure OpenAI resources may not have finished deploying. Wait a couple of minutes and try again.

Understand the code

Our application uses the `Microsoft.SemanticKernel` package, which is available on NuGet, to send and receive requests to an Azure OpenAI service deployed in Azure.

The entire application is contained within the **Program.cs** file. The first several lines of code loads up secrets and configuration values that were set in the `dotnet user-secrets` for you during the application provisioning.

C#

```csharp
var config = new ConfigurationBuilder().AddUserSecrets<Program>().Build();
string endpoint = config["AZURE_OPENAI_ENDPOINT"];
```

```
string deployment = config["AZURE_OPENAI_GPT_NAME"];
string key = config["AZURE_OPENAI_KEY"];
```

The `Kernel` class facilitates the requests and responses with the help of `AddAzureOpenAIChatCompletion` service.

C#

```
// Create a Kernel containing the Azure OpenAI Chat Completion Service
IKernelBuilder b = Kernel.CreateBuilder();

Kernel kernel = b
    .AddAzureOpenAIChatCompletion(deployment, endpoint, key)
    .Build();
```

The function's `ImportPluginFromFunctions` and `CreateFromMethod` are used to define the local function that will be called by the model.

C#

```
// Add a new plugin with a local .NET function that should be available to
the AI model
// For convenience and clarity of into the code, this standalone local
method handles tool call responses. It will fake a call to a weather API and
return the current weather for the specified location.
kernel.ImportPluginFromFunctions("WeatherPlugin",
[
    KernelFunctionFactory.CreateFromMethod(([Description("The city, e.g.
Montreal, Sidney")] string location, string unit = null) =>
    {
        // Here you would call a weather API to get the weather for the
location
        return "Periods of rain or drizzle, 15 C";
    }, "get_current_weather", "Get the current weather in a given location")
]);
```

Once the `kernel` client is created, we provide more context to the model by adding a system prompt. This instructs the model how you'd like it to act during the conversation. Note how the weather is emphasized in the system prompt.

C#

```
ChatHistory chatHistory = new("""
    You are a hiking enthusiast who helps people discover fun hikes in their
area. You are upbeat and friendly.
    A good weather is important for a good hike. Only make recommendations
if the weather is good or if people insist.
    You introduce yourself when first saying hello. When helping people out,
```

```
you always ask them
    for this information to inform the hiking recommendation you provide:

    1. Where they are located
    2. What hiking intensity they are looking for

    You will then provide three suggestions for nearby hikes that vary in
length after you get that information.
    You will also share an interesting fact about the local nature on the
hikes when making a recommendation.
    """);
```

Then you can add a user message to the model by using the `AddUserMessage` functon.

To have the model generate a response based off the system prompt and the user request, use the `GetChatMessageContentAsync` function.

C#

```
chatHistory.AddUserMessage("""
    Is the weather is good today for a hike?
    If yes, I live in the greater Montreal area and would like an easy hike.
I don't mind driving a bit to get there.
    I don't want the hike to be over 10 miles round trip. I'd consider a
point-to-point hike.
    I want the hike to be as isolated as possible. I don't want to see many
people.
    I would like it to be as bug free as possible.
    """);

Console.WriteLine($"{chatHistory.Last().Role} >>>
{chatHistory.Last().Content}");

chatHistory.Add(await service.GetChatMessageContentAsync(chatHistory, new
OpenAIPromptExecutionSettings() { MaxTokens = 400 }));
Console.WriteLine($"{chatHistory.Last().Role} >>>
{chatHistory.Last().Content}");
```

Customize the system prompt and user message to see how the model responds to help you find a hike that you'll like.

Clean up resources

When you no longer need the sample application or resources, remove the corresponding deployment and all resources.

Azure Developer CLI

```
azd down
```

Next steps

- Quickstart - Get insight about your data from an .NET Azure AI chat app
- Generate text and conversations with .NET and Azure OpenAI Completions

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see our contributor guide.

.NET feedback

.NET is an open source project. Select a link to provide feedback:

Open a documentation issue

Provide product feedback

Generate images using Azure AI with .NET

Article • 03/27/2024

Get started with Semantic Kernel by creating a simple .NET 8 console chat application. The application will run locally and use the OpenAI `dell-e-3` model to generate postal card and invite your friends for a hike! Follow these steps to provision Azure OpenAI and learn how to use Semantic Kernel.

Prerequisites

- .NET 8.0 SDK - Install the .NET 8.0 SDK
- An Azure subscription - Create one for free
- Azure Developer CLI - Install or update the Azure Developer CLI
- Access to Azure OpenAI service.

Deploy the Azure resources

Ensure that you follow the Prerequisites to have access to Azure OpenAI Service as well as the Azure Developer CLI, and then follow the following guide to set started with the sample application.

1. Clone the repository: dotnet/ai-samples
2. From a terminal or command prompt, navigate to the *quickstarts* directory.
3. This provisions the Azure OpenAI resources. It may take several minutes to create the Azure OpenAI service and deploy the model.

   Azure Developer CLI

   ```
   azd up
   ```

> **Note**
>
> If you already have an Azure OpenAI service available, you can skip the deployment and use that value in the *Program.cs*, preferably from an `IConfiguration`.

Troubleshoot

On Windows, you might get the following error message after running `azd up`:

> *postprovision.ps1 is not digitally signed. The script will not execute on the system*

The script **postprovision.ps1** is executed to set the .NET user secrets used in the application. To avoid this error, run the following PowerShell command:

PowerShell

```
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
```

Then re-run the `azd up` command.

Trying Generate Hiking Images sample

1. From a terminal or command prompt, navigate to the `semantic-kernel\05-HikeImages` directory.
2. It's now time to try the console application. Type in the following to run the app:

   .NET CLI

   ```
   dotnet run
   ```

   If you get an error message the Azure OpenAI resources may not have finished deploying. Wait a couple of minutes and try again.

Understanding the code

Our application uses the `Microsoft.SemanticKernel` package, which is available on NuGet, to send and receive requests to an Azure OpenAI service deployed in Azure.

The entire application is contained within the *Program.cs* file. The first several lines of code load secrets and configuration values that were set in the `dotnet user-secrets` for you during the application provisioning.

C#

```
// == Retrieve the local secrets saved during the Azure deployment
==========
```

```
var config = new ConfigurationBuilder()
    .AddUserSecrets<Program>()
    .Build();

var config = new ConfigurationBuilder().AddUserSecrets<Program>().Build();
string endpoint = config["AZURE_OPENAI_ENDPOINT"];
string deployment = config["AZURE_OPENAI_GPT_NAME"];
string key = config["AZURE_OPENAI_KEY"];
```

The `AzureOpenAITextToImageService` service facilitates the requests and responses.

C#

```
AzureOpenAITextToImageService textToImageService = new(deployment, endpoint,
key, null);
```

Once the `textToImageService` service is created, we we provide more context to the model by adding a system prompt. A good prompt to generate images requires a clear description: what is in the images, specific color to use, style (drawing, painting, realistic or cartoony). The model will use this prompt to generate the image. To have the model generate a response based off the user request, use the `GenerateImageAsync` function, and specify the size and quality.

C#

```
// Generate the image
string imageUrl = await textToImageService.GenerateImageAsync("""
    A postal card with an happy hiker waving and a beautiful mountain in the
background.
    There is a trail visible in the foreground.
    The postal card has text in red saying: 'You are invited for a hike!'
    """, 1024, 1024);
Console.WriteLine($"The generated image is ready at:\n{imageUrl}");
```

Customize the prompt to personalize the images generated by the model.

Clean up resources

When you no longer need the sample application or resources, remove the corresponding deployment and all resources.

Azure Developer CLI

```
azd down
```

Next steps

- Quickstart - Summarize text using Azure AI chat app with .NET
- Generate text and conversations with .NET and Azure OpenAI Completions

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see our contributor guide.

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

Open a documentation issue

Provide product feedback

Get started with the .NET enterprise chat sample using RAG

Article • 03/19/2024

This article shows you how to deploy and run the Enterprise chat app sample for .NET. This sample implements a chat app using C#, Azure OpenAI Service, and Retrieval Augmented Generation (RAG) in Azure AI Search to get answers about employee benefits at a fictitious company. The employee benefits chat app is seeded with PDF files including an employee handbook, a benefits document and a list of company roles and expectations.

- Demo video

Begin now

By following the instructions in this article, you will:

- Deploy a chat app to Azure.
- Get answers about employee benefits.
- Change settings to change behavior of responses.

Once you complete this procedure,you can start modifying the new project with your custom code.

This article is part of a collection of articles that show you how to build a chat app using Azure Open AI Service and Azure AI Search.

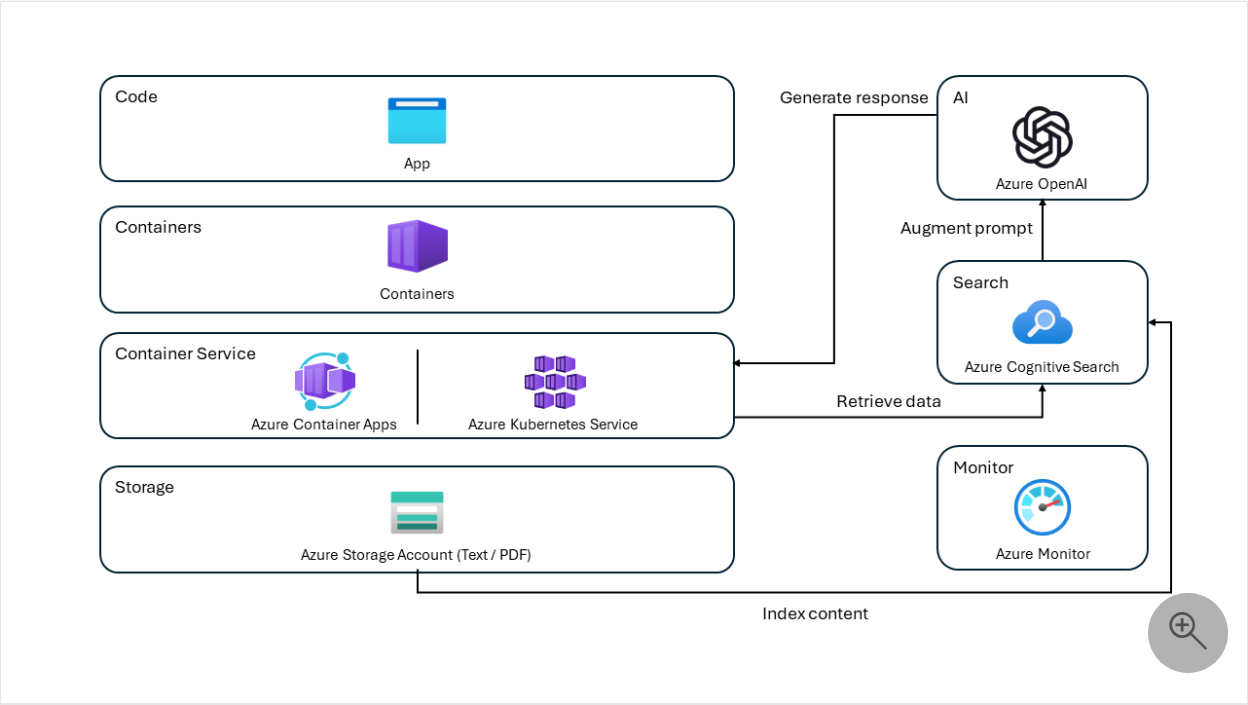
Other articles in the collection include:

- Python
- JavaScript
- Java

Architectural overview

In this sample application, a fictitious company called Contoso Electronics provides the chat app experience to its employees to ask questions about the benefits, internal policies, as well as job descriptions and roles.

The architecture of the chat app is shown in the following diagram:



- **User interface** - The application's chat interface is a Blazor WebAssembly application. This interface is what accepts user queries, routes request to the application backend, and displays generated responses.
- **Backend** - The application backend is an ASP.NET Core Minimal API. The backend hosts the Blazor static web application and is what orchestrates the interactions among the different services. Services used in this application include:
  - **Azure Cognitive Search** – Indexes documents from the data stored in an Azure Storage Account. This makes the documents searchable using vector search capabilities.
  - **Azure OpenAI Service** – Provides the Large Language Models (LLM) to generate responses. Semantic Kernel is used in conjunction with the Azure OpenAI Service to orchestrate the more complex AI workflows.

Cost

Most resources in this architecture use a basic or consumption pricing tier. Consumption pricing is based on usage, which means you only pay for what you use. To complete this article, there will be a charge but it will be minimal. When you are done with the article, you can delete the resources to stop incurring charges.

For more information, see Azure Samples: Cost in the sample repo .

Prerequisites

A development container environment is available with all dependencies required to complete this article. You can run the development container in GitHub Codespaces (in a

browser) or locally using Visual Studio Code.

To follow along with this article, you need the following prerequisites:

Codespaces (recommended)

1. An Azure subscription - Create one for free
2. Azure account permissions - Your Azure Account must have Microsoft.Authorization/roleAssignments/write permissions, such as User Access Administrator or Owner.
3. Access granted to Azure OpenAI in the desired Azure subscription. Currently, access to this service is granted only by application. You can apply for access to Azure OpenAI by completing the form at https://aka.ms/oai/access. Open an issue on this repo to contact us if you have an issue.
4. GitHub account

Open development environment

Begin now with a development environment that has all the dependencies installed to complete this article.

GitHub Codespaces (recommended)

GitHub Codespaces runs a development container managed by GitHub with Visual Studio Code for the Web as the user interface. For the most straightforward development environment, use GitHub Codespaces so that you have the correct developer tools and dependencies preinstalled to complete this article.

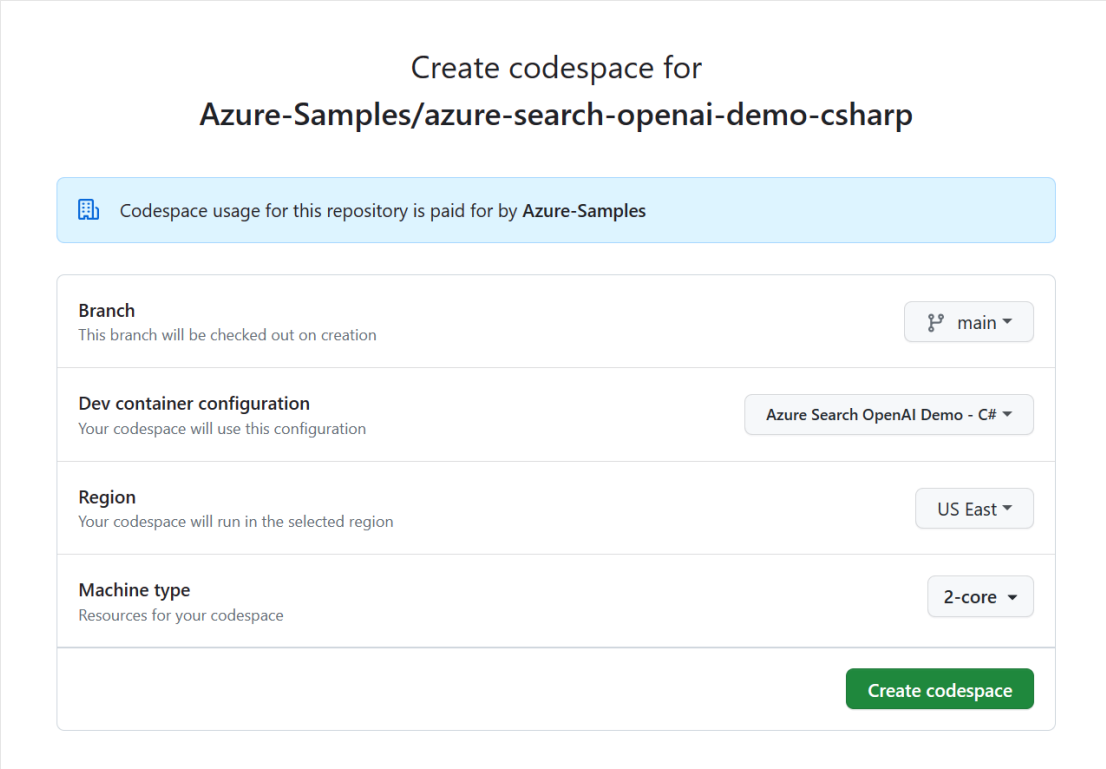
> **Important**
>
> All GitHub accounts can use Codespaces for up to 60 hours free each month with 2 core instances. For more information, see GitHub Codespaces monthly included storage and core hours.

1. Start the process to create a new GitHub Codespace on the `main` branch of the Azure-Samples/azure-search-openai-demo-csharp GitHub repository.
2. Right-click on the following button, and select *Open link in new windows* in order to have both the development environment and the documentation

available at the same time.

Open this project in GitHub Codespaces

3. On the **Create codespace** page, review the codespace configuration settings and then select **Create new codespace**:



4. Wait for the codespace to start. This startup process can take a few minutes.

5. In the terminal at the bottom of the screen, sign in to Azure with the Azure Developer CLI.

```bash
azd auth login
```

6. Copy the code from the terminal and then paste it into a browser. Follow the instructions to authenticate with your Azure account.

7. The remaining tasks in this article take place in the context of this development container.

Deploy and run

The sample repository contains all the code and configuration files you need to deploy a chat app to Azure. The following steps walk you through the process of deploying the sample to Azure.

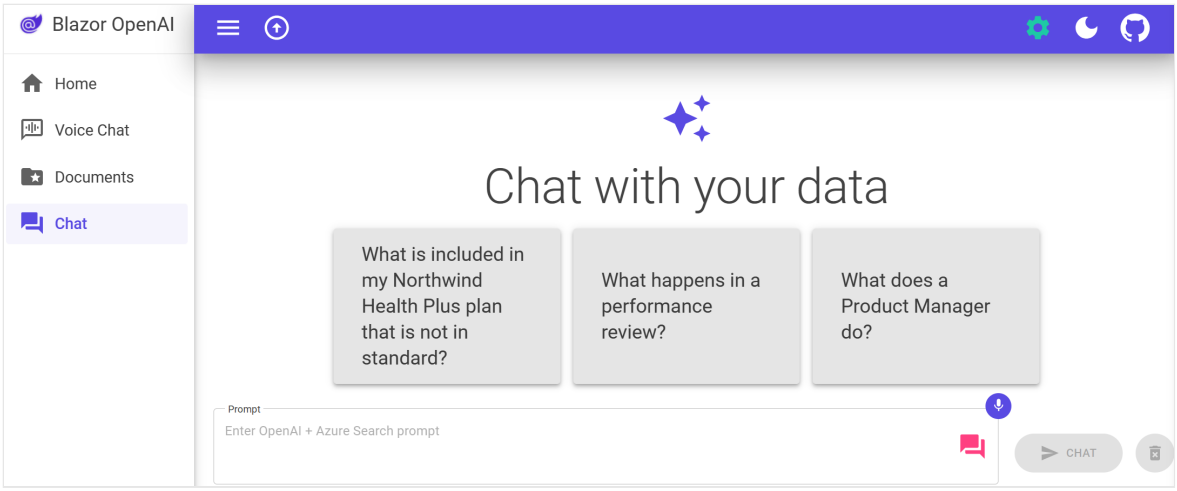
Deploy chat app to Azure

> **ⓘ Important**
>
> Azure resources created in this section incur immediate costs, primarily from the Azure AI Search resource. These resources may accrue costs even if you interrupt the command before it is fully executed.

1. Run the following Azure Developer CLI command to provision the Azure resources and deploy the source code:

   ```
   Bash

   azd up
   ```

2. When you're prompted to enter an environment name, keep it short and lowercase. For example, `myenv`. Its used as part of the resource group name.
3. When prompted, select a subscription to create the resources in.
4. When you're prompted to select a location the first time, select a location near you. This location is used for most the resources including hosting.
5. If you're prompted for a location for the OpenAI model, select a location that is near you. If the same location is available as your first location, select that.
6. Wait until app is deployed. It may take up to 20 minutes for the deployment to complete.
7. After the application has been successfully deployed, you see a URL displayed in the terminal.
8. Select that URL labeled `Deploying service web` to open the chat application in a browser.

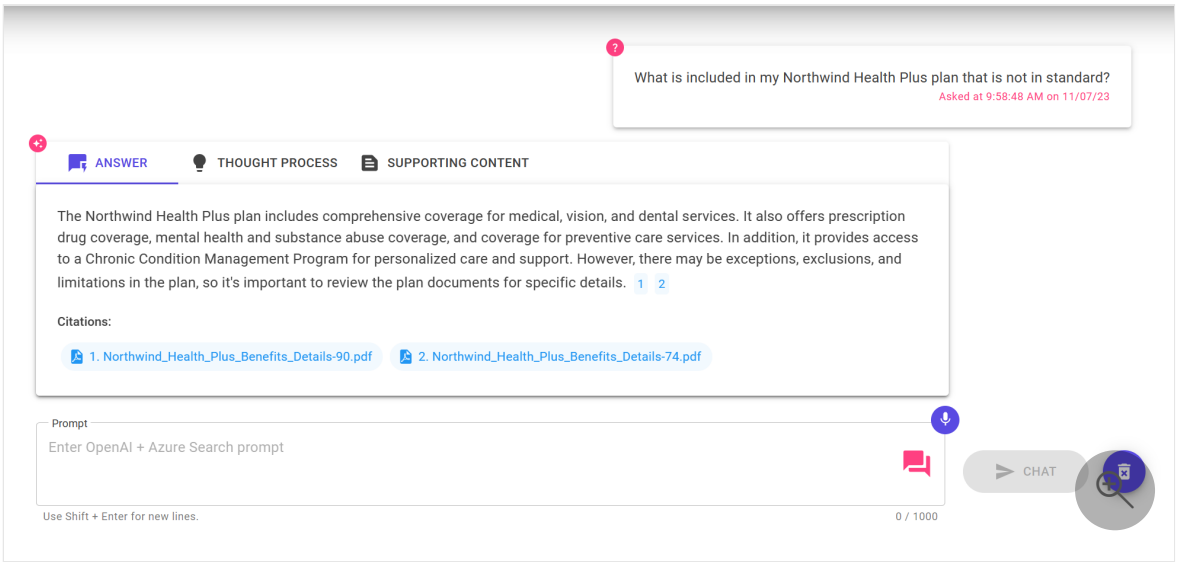


Use chat app to get answers from PDF files

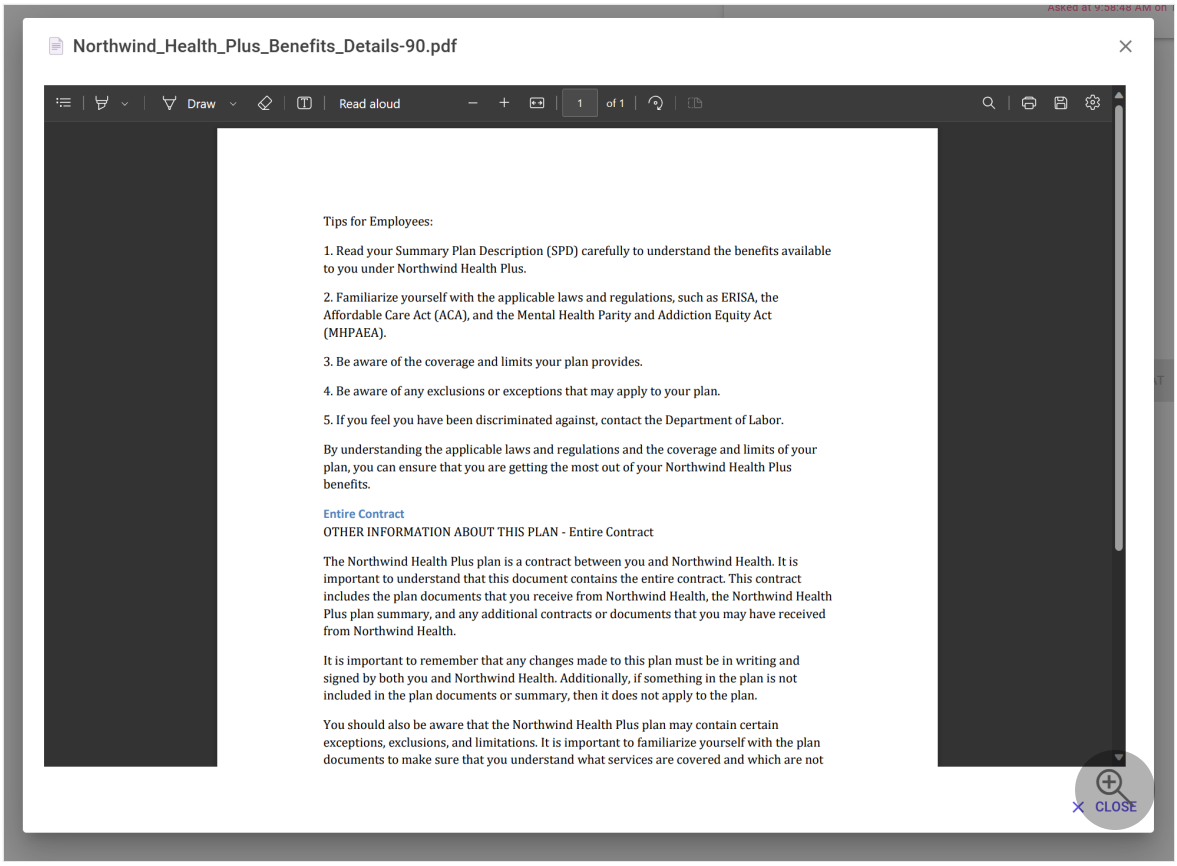
The chat app is preloaded with employee benefits information from PDF files. You can use the chat app to ask questions about the benefits. The following steps walk you through the process of using the chat app.

1. In the browser, navigate to the **Chat** page using the left navigation.

2. Select or enter "What is included in my Northwind Health Plus plan that is not in standard?" in the chat text box.



3. From the answer, select a citation. A pop-up window will open displaying the source of the information.



4. Navigate between the tabs at the top of the answer box to understand how the answer was generated.

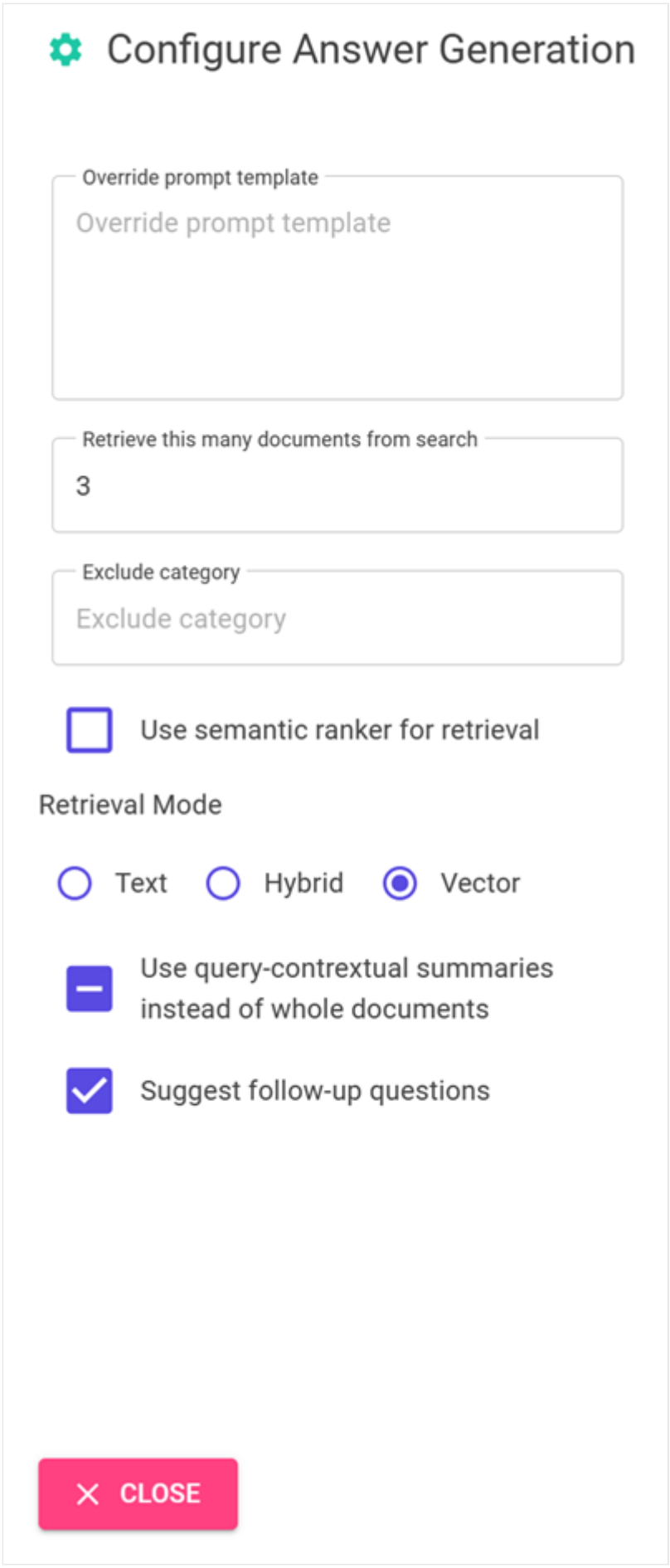
Expand table

Tab	Description
Thought process	This is a script of the interactions in chat. You can view the system prompt (`content`) and your user question (`content`).
Supporting content	This includes the information to answer your question and the source material. The number of source material citations is noted in the **Developer settings**. The default value is **3**.
Citation	This displays the source page that contains the citation.

5. When you're done, navigate back to the answer tab.

Use chat app settings to change behavior of responses

The intelligence of the chat is determined by the OpenAI model and the settings that are used to interact with the model.



Expand table

Setting	Description
Override prompt template	This is the prompt that is used to generate the answer.
Retrieve this many search results	This is the number of search results that are used to generate the answer. You can see these sources returned in the *Thought process* and *Supporting content* tabs of the citation.
Exclude category	This is the category of documents that are excluded from the search results.
Use semantic ranker for retrieval	This is a feature of Azure AI Search that uses machine learning to improve the relevance of search results.
Retrieval mode	**Vectors + Text** means that the search results are based on the text of the documents and the embeddings of the documents. **Vectors** means that the search results are based on the embeddings of the documents. **Text** means that the search results are based on the text of the documents.
Use query-contextual summaries instead of whole documents	When both `Use semantic ranker` and `Use query-contextual summaries` are checked, the LLM uses captions extracted from key passages, instead of all the passages, in the highest ranked documents.
Suggest follow-up questions	Have the chat app suggest follow-up questions based on the answer.

The following steps walk you through the process of changing the settings.

1. In the browser, select the gear icon in the upper right of the page.
2. Check the **Suggest follow-up questions** checkbox and ask the same question again.

   ```
   What is my deductible?
   ```

   The chat returns follow-up question suggestions such as the following:

   - "What is the cost sharing for out-of-network services?"
   - "Are preventive care services subject to the deductible?"
   - "How does the prescription drug deductible work?"
3. In the **Settings** tab, deselect **Use semantic ranker for retrieval**.
4. Ask the same question again.

```
Text
```

```
What is my deductible?
```

5. What is the difference in the answers?

   The response which used the Semantic ranker provided a single answer: `The deductible for the Northwind Health Plus plan is $2,000 per year`.

   The response without semantic ranking returned a less direct answer: `Based on the information provided, it is unclear what your specific deductible is. The Northwind Health Plus plan has different deductible amounts for in-network and out-of-network services, and there is also a separate prescription drug deductible. I would recommend checking with your provider or referring to the specific benefits details for your plan to determine your deductible amount`.

Clean up resources

Clean up Azure resources

The Azure resources created in this article are billed to your Azure subscription. If you don't expect to need these resources in the future, delete them to avoid incurring more charges.

Run the following Azure Developer CLI command to delete the Azure resources and remove the source code:

```
Bash
```

```
azd down --purge
```

Clean up GitHub Codespaces

GitHub Codespaces

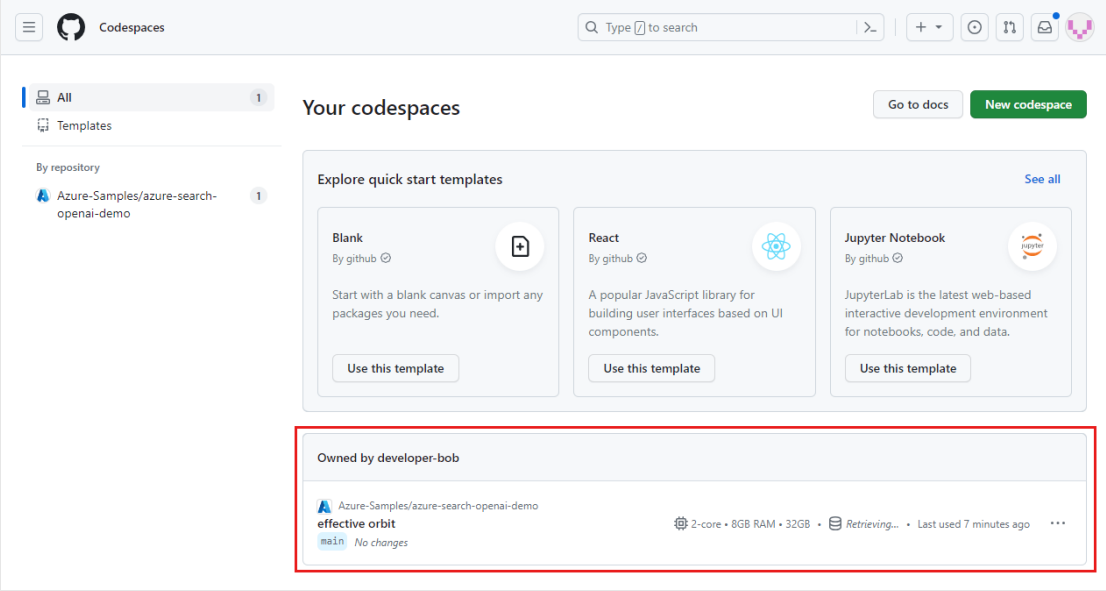
Deleting the GitHub Codespaces environment ensures that you can maximize the amount of free per-core hours entitlement you get for your account.

> **Important**

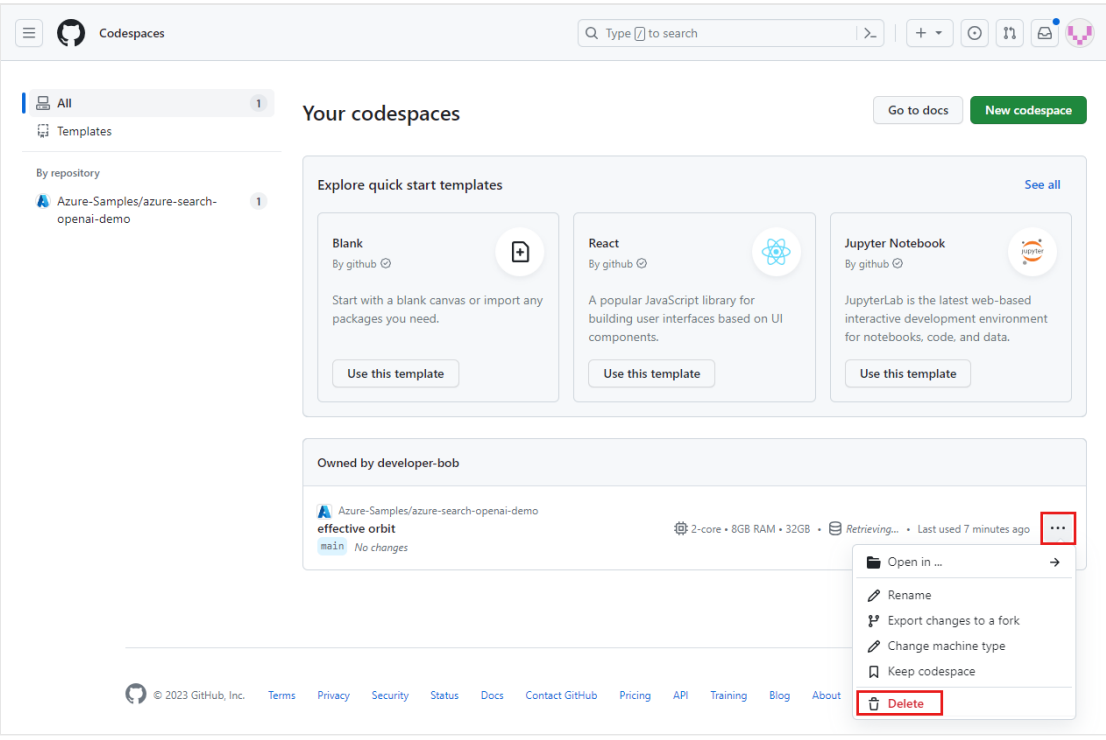
For more information about your GitHub account's entitlements, see [GitHub Codespaces monthly included storage and core hours](#).

1. Sign into the GitHub Codespaces dashboard (https://github.com/codespaces).

2. Locate your currently running codespaces sourced from the Azure-Samples/azure-search-openai-demo-csharp GitHub repository.



3. Open the context menu for the codespace and then select **Delete**.



Get help

This sample repository offers troubleshooting information.

If your issue isn't addressed, log your issue to the repository's Issues.

Next steps

- Enterprise chat app GitHub repository
- Build a chat app with Azure OpenAI best practice solution architecture
- Access control in Generative AI Apps with Azure AI Search
- Build an Enterprise ready OpenAI solution with Azure API Management
- Outperforming vector search with hybrid retrieval and ranking capabilities

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see our contributor guide.

.NET feedback

.NET is an open source project. Select a link to provide feedback:

Open a documentation issue

Provide product feedback

Scale Azure OpenAI for .NET chat using RAG with Azure Container Apps

Article • 03/13/2024

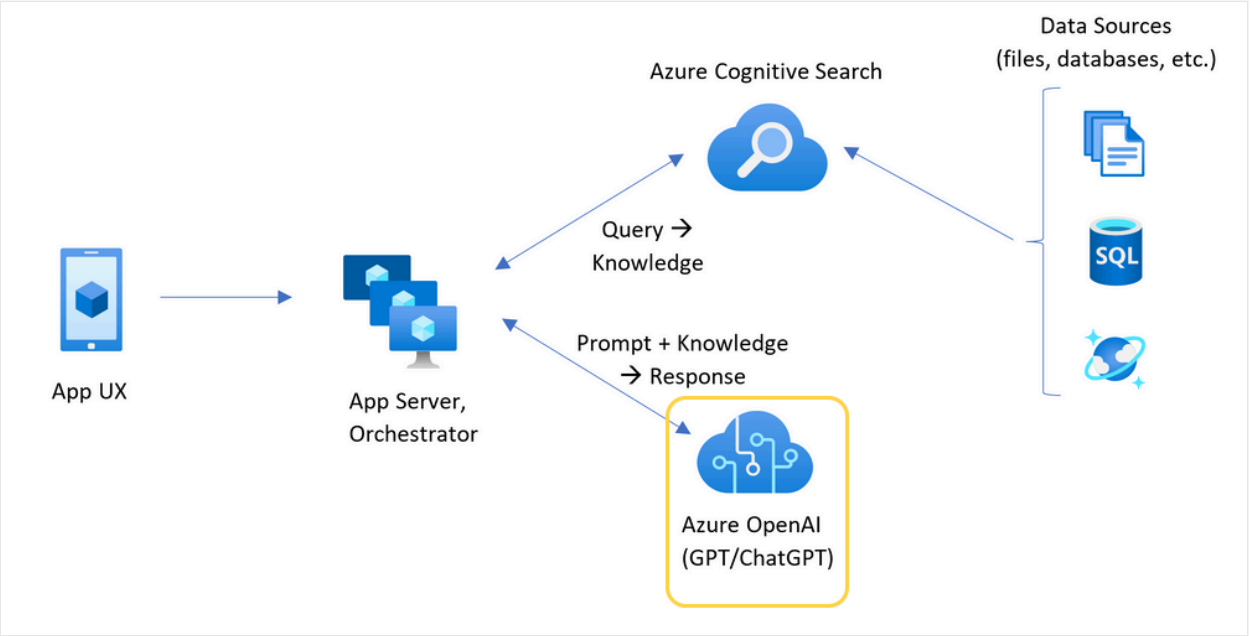
Learn how to add load balancing to your application to extend the chat app beyond the Azure OpenAI token and model quota limits. This approach uses Azure Container Apps to create three Azure OpenAI endpoints, as well as a primary container to direct incoming traffic to one of the three endpoints.

This article requires you to deploy 2 separate samples:

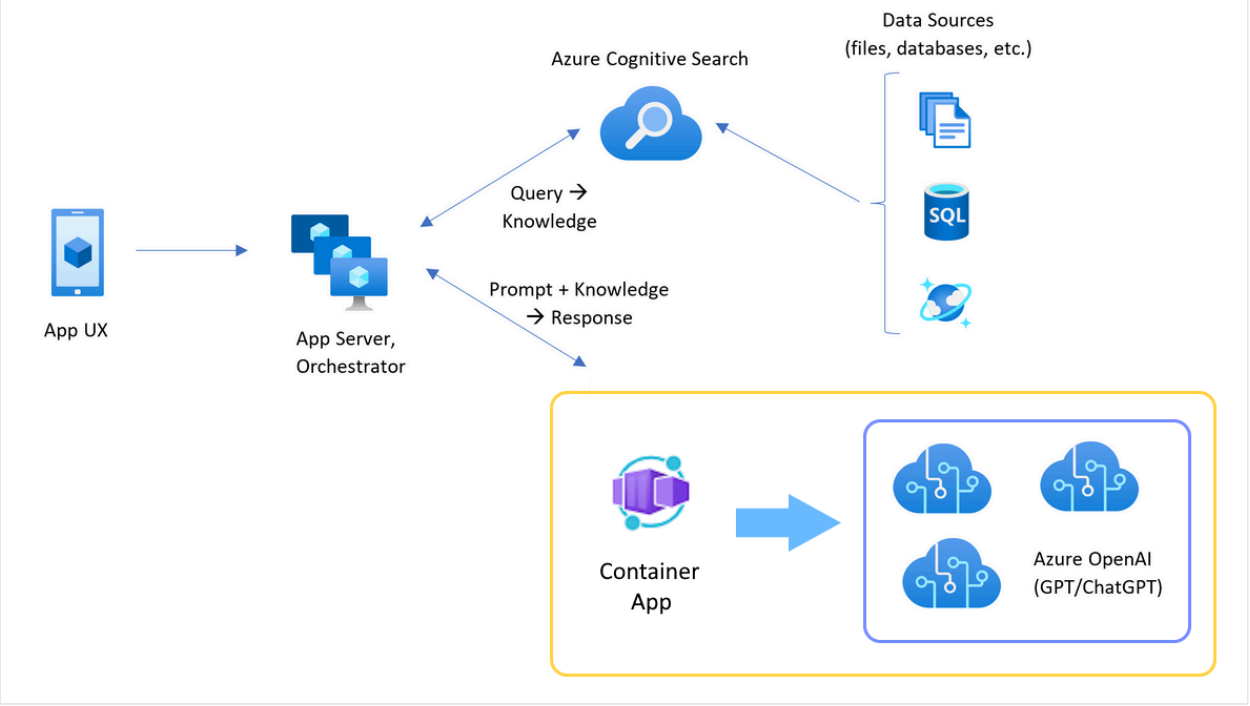
- Chat app
  - If you haven't deployed the chat app yet, wait until after the load balancer sample is deployed.
  - If you have already deployed the chat app once, you'll change the environment variable to support a custom endpoint for the load balancer and redeploy it again.
  - Chat app available in these languages:
    - .NET
    - JavaScript
    - Python
- Load balancer app

Architecture for load balancing Azure OpenAI with Azure Container Apps

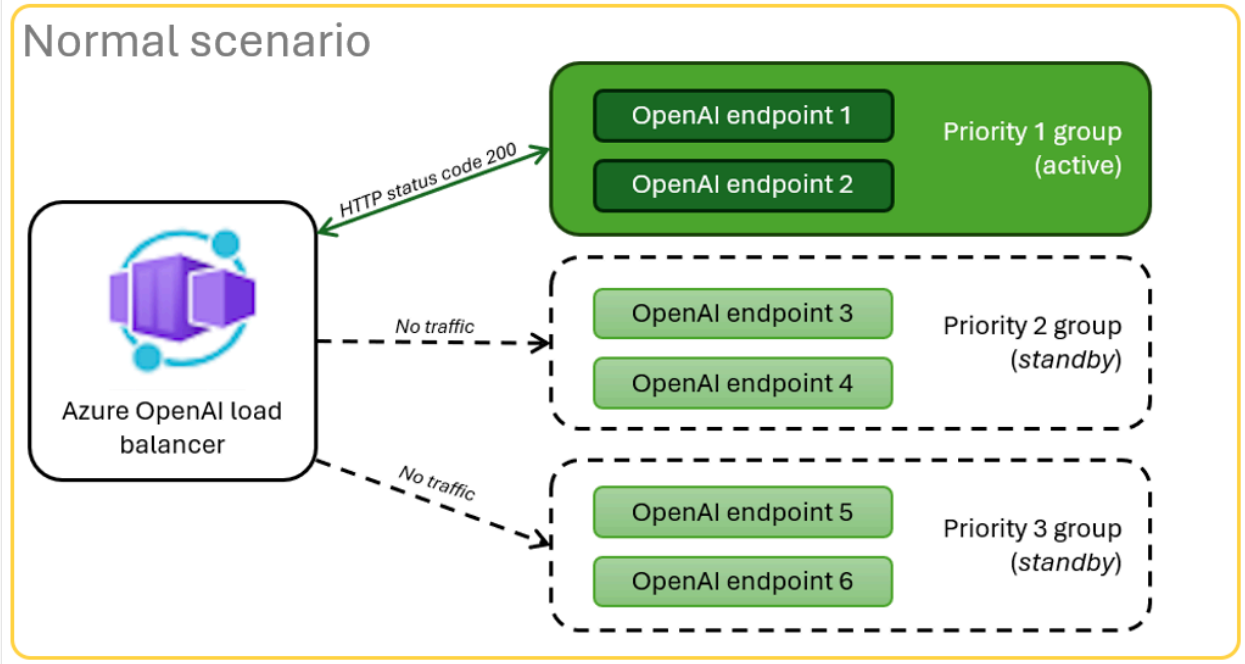
Because the Azure OpenAI resource has specific token and model quota limits, a chat app using a single Azure OpenAI resource is prone to have conversation failures due to those limits.



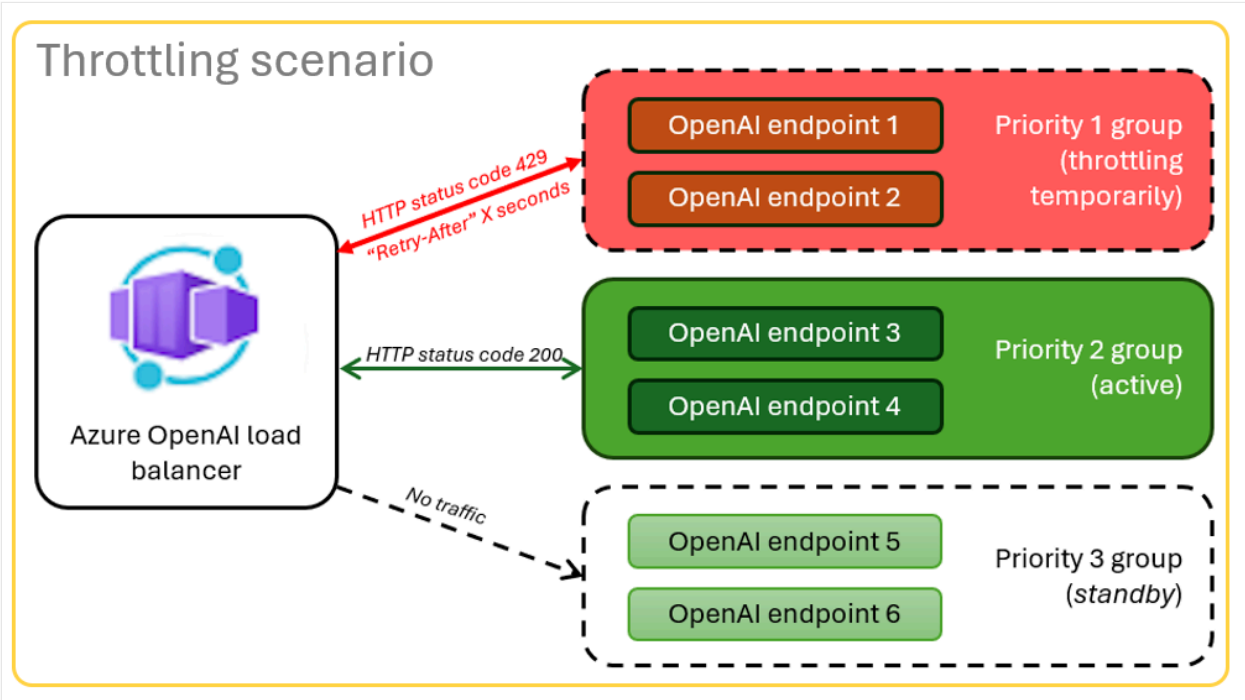
To use the chat app without hitting those limits, use a load balanced solution with Azure Container Apps. This solution seamlessly exposes a single endpoint from Azure Container Apps to your chat app server.



The Azure Container app sits in front of a set of Azure OpenAI resources. The Container app solves two scenarios: normal and throttled. During a **normal scenario** where token and model quota is available, the Azure OpenAI resource returns a 200 back through the Container App and App Server.



When a resource is in a **throttled scenario** such as due to quota limits, the Azure Container app can retry a different Azure OpenAI resource immediately to fullful the original chat app request.



Prerequisites

- Azure subscription. Create one for free
- Access granted to Azure OpenAI in the desired Azure subscription.

Currently, access to this service is granted only by application. You should apply for access to Azure OpenAI.

- Dev containers are available for both samples, with all dependencies required to complete this article. You can run the dev containers in GitHub Codespaces (in a browser) or locally using Visual Studio Code.

Codespaces (recommended)

- Only a GitHub account is required to use CodeSpaces

Open Container apps local balancer sample app

Codespaces (recommended)

GitHub Codespaces runs a development container managed by GitHub with Visual Studio Code for the Web as the user interface. For the most straightforward development environment, use GitHub Codespaces so that you have the correct developer tools and dependencies preinstalled to complete this article.

Open in GitHub Codespaces

Important

All GitHub accounts can use Codespaces for up to 60 hours free each month with 2 core instances. For more information, see GitHub Codespaces monthly included storage and core hours.

Deploy Azure Container Apps load balancer

1. To deploy the load balancer to Azure, sign in to Azure Developer CLI (AZD).

Bash

```
azd auth login
```

2. Finish the sign in instructions.

3. Deploy the load balancer app.

```bash
azd up
```

You'll need to select a subscription and region for the deployment. These don't have to be the same subscription and region as the chat app.

4. Wait for the deployment to complete before continuing.

5. Get the URL at the end of the deployment named **Endpoint**. This is the `CONTAINER_APP_URL` used in the next section.

Redeploy Chat app with load balancer endpoint

These are completed on the chat app sample.

Initial deployment

1. Open the chat app sample's dev container using one of the following choices.

Expand table

Language	Codespaces	Visual Studio Code
.NET		
JavaScript		
Python		

2. Sign in to Azure Developer CLI (AZD).

```bash
azd auth login
```

Finish the sign in instructions.

3. Create an AZD environment with a name such as `chat-app`.

   ```
   Bash

   azd env new <name>
   ```

4. Add the following environment variable, which tells the Chat app's backend to use a custom URL for the OpenAI requests.

   ```
   Bash

   azd env set OPENAI_HOST azure_custom
   ```

5. Add the following environment variable, substituting `<CONTAINER_APP_URL>` for the URL from the previous section. This action tells the Chat app's backend what the value is of the custom URL for the OpenAI request.

   ```
   Bash

   azd env set AZURE_OPENAI_CUSTOM_URL <CONTAINER_APP_URL>
   ```

6. Deploy the chat app.

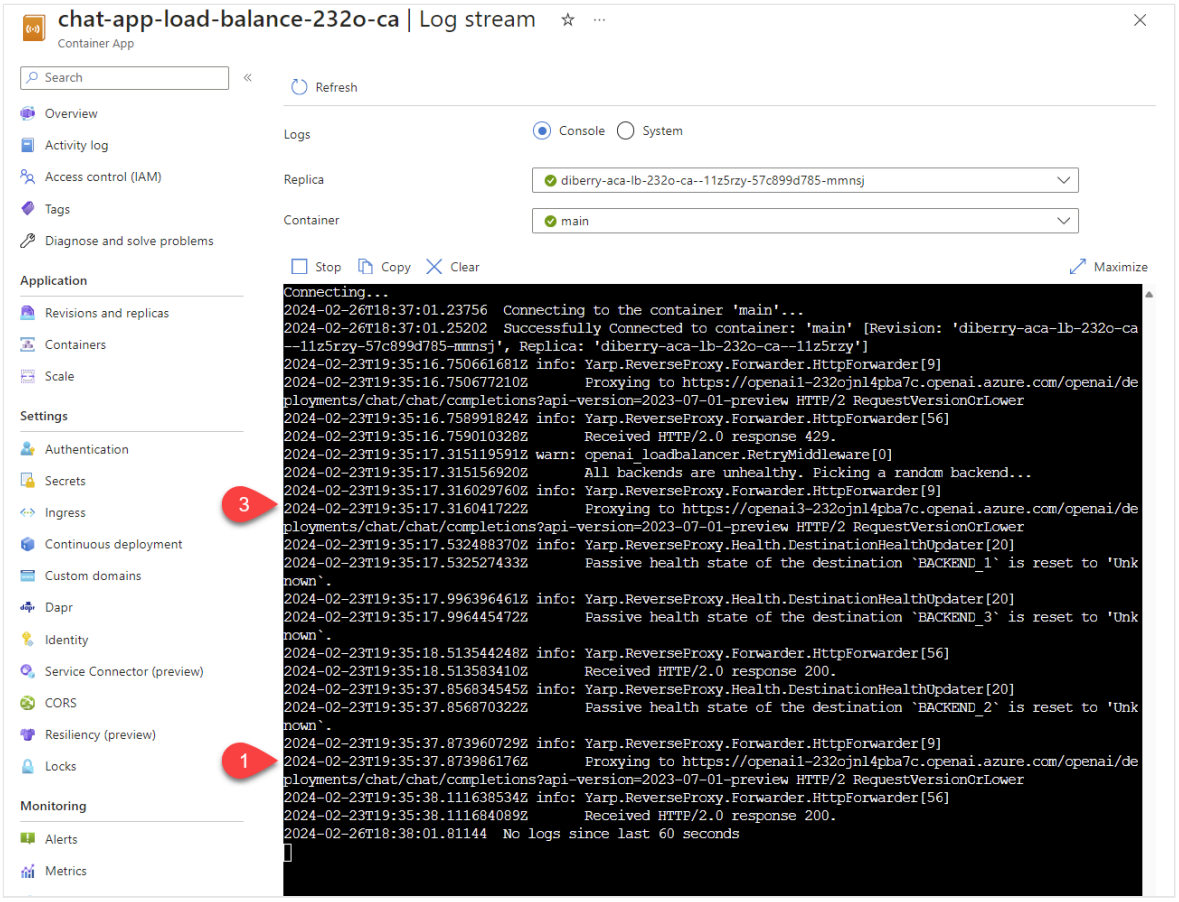
   ```
   Bash

   azd up
   ```

You can now use the chat app with the confidence that it's built to scale across many users without running out of quota.

Stream logs to see the load balancer results

1. In the Azure portal, search your resource group.
2. From the list of resources in the group, select the Container App resource.
3. Select **Monitoring -> Log stream** to view the log.
4. Use the chat app to generate traffic in the log.
5. Look for the logs, which reference the Azure OpenAI resources. Each of the three resources has its numeric identity in the log comment beginning with `Proxying to`

`https://openai3`, where `3` indicates the third Azure OpenAI resource.



6. As you use the chat app, when the load balancer receives status that the request has exceeded quota, the load balancer automatically rotates to another resource.

Configure the tokens per minute quota (TPM)

By default, each of the OpenAI instances in the load balancer will be deployed with 30,000 TPM (tokens per minute) capacity. You can use the chat app with the confidence that it's built to scale across many users without running out of quota. Change this value when:

- You get deployment capacity errors: lower that value.
- Planning higher capacity, raise the value.

1. Use the following command to change the value.

Bash

```
azd env set OPENAI_CAPACITY 50
```

2. Redeploy the load balancer.

Bash

```
azd up
```

Clean up resources

When you're done with both the chat app and the load balancer, clean up the resources. The Azure resources created in this article are billed to your Azure subscription. If you don't expect to need these resources in the future, delete them to avoid incurring more charges.

Clean up chat app resources

Return to the chat app article to clean up those resources.

- .NET
- JavaScript
- Python

Clean upload balancer resources

Run the following Azure Developer CLI command to delete the Azure resources and remove the source code:

Bash

```
azd down --purge --force
```

The switches provide:

- `purge`: Deleted resources are immediately purged. This allows you to reuse the Azure OpenAI TPM.
- `force`: The deletion happens silently, without requiring user consent.

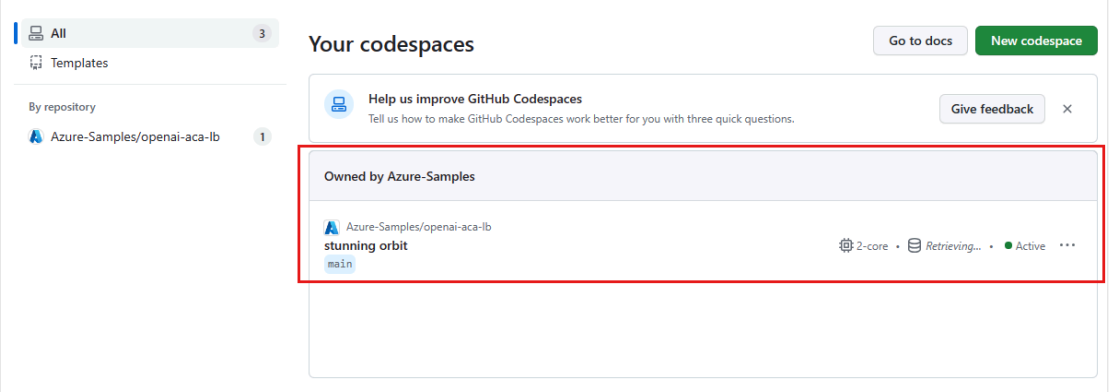
Clean up GitHub Codespaces

GitHub Codespaces

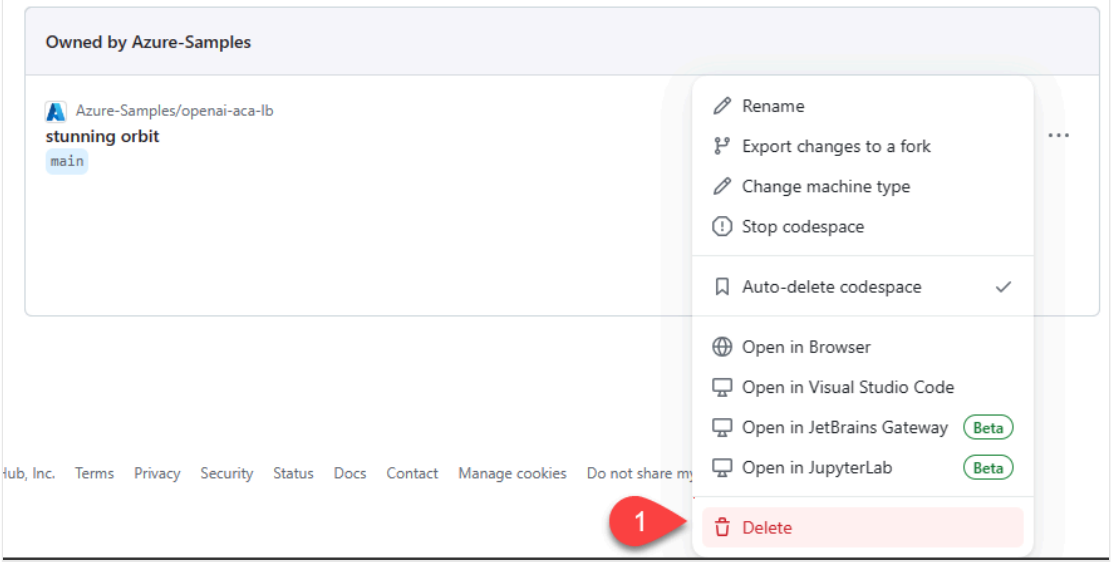
Deleting the GitHub Codespaces environment ensures that you can maximize the amount of free per-core hours entitlement you get for your account.

> **Important**
>
> For more information about your GitHub account's entitlements, see [GitHub Codespaces monthly included storage and core hours].

1. Sign into the GitHub Codespaces dashboard (https://github.com/codespaces).
2. Locate your currently running Codespaces sourced from the azure-samples/openai-aca-lb GitHub repository.



3. Open the context menu for the codespace and then select **Delete**.



Get help

If you have trouble deploying the Azure API Management load balancer, log your issue to the repository's Issues.

Sample code

Samples used in this article include:

- .NET chat app with RAG
- Load Balancer with Azure Container Apps

Next step

- Use Azure Load Testing to load test your chat app

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see our contributor guide.

.NET feedback

.NET is an open source project. Select a link to provide feedback:

Open a documentation issue

Provide product feedback

Overview of the .NET + AI ecosystem

Article • 04/08/2024

.NET can be used with many different libraries and tools that support the development of generative AI applications. This page includes a summary of the services and tools you might need to use in your applications, with links to learn more about each of them.

> **Note**
>
> We recommend using the Semantic Kernel SDK to orchestrate your calls to large language models (LLMs) and manage interactions with the various services mentioned here. Semantic Kernel makes it easy to work with different services without having to learn a different API for each one.

> **Important**
>
> These SDKs and tools are built by a variety of sources. Not all SDKs are maintained by Microsoft. When considering an SDK, be sure to evaluate its quality, licensing, and support to ensure they meet your requirements. Also make sure you review each SDK's documentation for detailed version compatibility information.

Working with models

Today, you can use .NET to access models built by OpenAI, using either the Azure OpenAI SDK or the Semantic Kernel. These models can be hosted by OpenAI or in Azure using the Azure AI services. Preview support is coming soon in Semantic Kernel to work with other models, and you can experiment today using open-source SDKs created by the .NET developer community.

Expand table

NuGet package	Supported models	Maintainer or vendor	Link to docs
Microsoft.SemanticKernel	OpenAI models Azure OpenAI supported models	Semantic Kernel (Microsoft)	Semantic Kernel documentation
Azure OpenAI SDK	Azure OpenAI supported models	Azure SDK for .NET (Microsoft)	Azure OpenAI services documentation

Connect your data using vector stores

To increase relevancy and tailor AI applications for your own data, you'll likely need to work with a vector store. Many services provide a native SDK for .NET, which you can use directly. You can also use Semantic Kernel, which provides an extensible component model that allows you to try different vector stores without needing to learn each SDK.

Expand table

NuGet package	Supported vector store	Maintainer or vendor	Link to docs
Microsoft.SemanticKernel	Supported vector stores	Semantic Kernel (Microsoft)	Semantic Kernel: What is a vector database
Azure.Search.Documents	Azure AI Search	Azure SDK for .NET (Microsoft)	Azure AI Search client library for .NET
Milvus.Client	Milvus Vector Database	Milvus	Install Milvus C# SDK
Qdrant.Client	Qdrant Vector Database	Qdrant	Qdrant .NET SDK

Other options

This article summarized the tools and SDKs in the .NET ecosystem, with a focus on services that provide official support for .NET. Depending on your needs and stage of app development, you might also want to take a look at the open-source options for the ecosystem in the unofficial list of .NET + AI resources. Microsoft is not the maintainer of many of these projects, so be sure to review their quality, licensing, and support.

Next Steps

- What is Semantic Kernel?
- Quickstart - Summarize text using Azure AI chat app with .NET

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see our contributor guide.

.NET is an open source project. Select a link to provide feedback:

Open a documentation issue

Provide product feedback

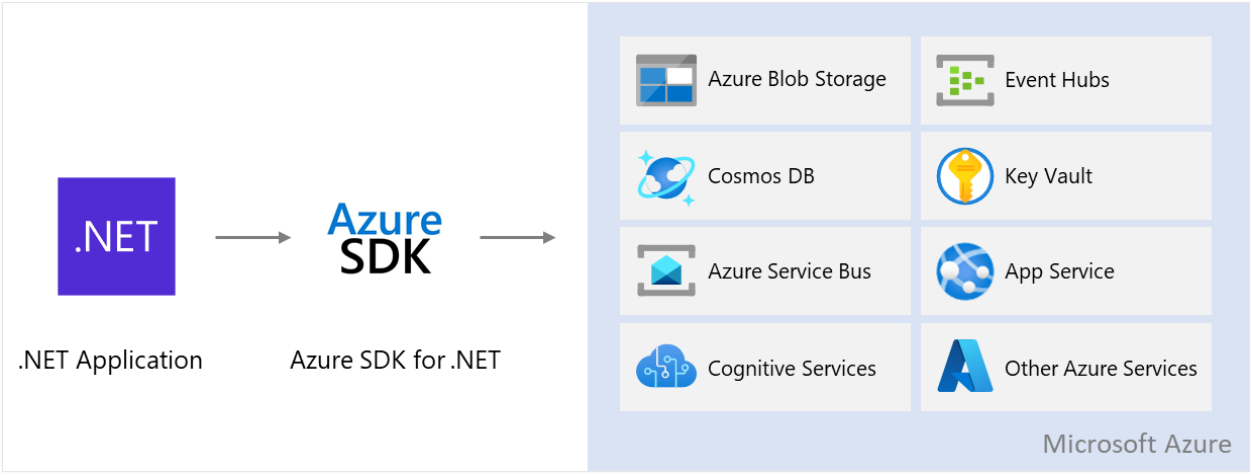
Azure SDK for .NET overview

Article • 03/24/2023

What is the Azure SDK for .NET

The **Azure SDK for .NET** is designed to make it easy to use Azure services from your .NET applications. Whether it is uploading and downloading files to Blob Storage, retrieving application secrets from Azure Key Vault, or processing notifications from Azure Event Hubs, the Azure SDK for .NET provides a consistent and familiar interface to access Azure services.

The Azure SDK for .NET is available as series of NuGet packages that can be used in both .NET Core (2.1 and higher) and .NET Framework (4.7.2 and higher) applications.



Use the Azure SDK for .NET in your applications

To use an Azure SDK package in one of your .NET applications, you want to follow these steps.

1. **Locate the appropriate SDK package** - Use the package list to find the appropriate package for the Azure service you are working with. Be advised that most services have a client package for working with the service and a management package for creating and managing instances of the service. In most cases, you will want the client package. Install this package in your application using NuGet.

2. **Set up authentication for your application** - To access Azure resources, your application will need to have the appropriate credentials and access rights assigned in Azure. Learn how to configure authentication in Authenticating .NET applications to Azure.

3. **Write code using the SDK in your application** - When working with Azure services, your code will first create a client object to work with the service and then call methods on that client object to interact with the service. Both synchronous and asynchronous methods are provided. Examples of using each individual SDK package are provided throughout the Azure documentation.

4. **Configure logging for the SDK (optional)** - If you need to diagnose issues between your application and Azure, you can enable logging in the Azure SDK for .NET.

How to authenticate .NET apps to Azure services using the .NET Azure SDK

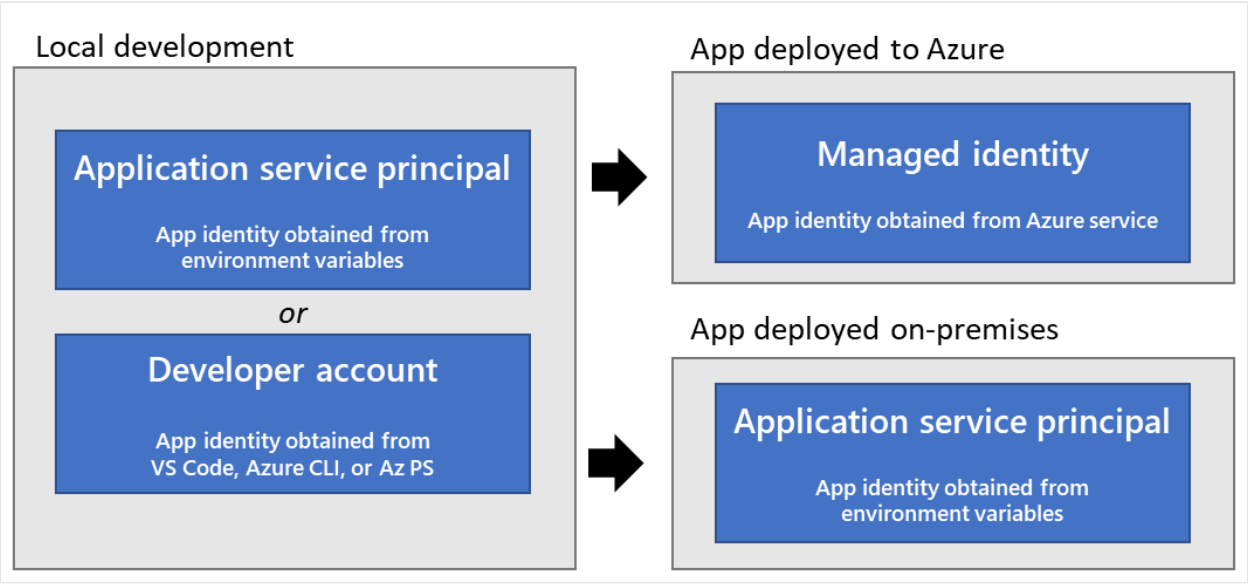
Article • 08/03/2023

When an application needs to access an Azure resource such as storage, key vault, or cognitive services, the application must be authenticated to Azure. This is true for all applications, whether deployed to Azure, deployed on-premises, or under development on a local developer workstation. This article describes the recommended approaches to authenticate an app to Azure when using the Azure SDK for .NET.

Recommended app authentication approach

It is recommended that apps use token-based authentication rather than connection strings when authenticating to Azure resources. The Azure SDK for .NET provides classes that support token-based authentication and allow apps to seamlessly authenticate to Azure resources whether the app is in local development, deployed to Azure, or deployed to an on-premises server.

The specific type of token-based authentication an app should use to authenticate to Azure resources depends on where the app is being run and is shown in the following diagram.



- **When a developer is running an app during local development** - The app can authenticate to Azure using either an application service principal for local development or by using the developer's Azure credentials. Each of these options is discussed in more detail in the section authentication during local development.

- **When an app is hosted on Azure** - The app should authenticate to Azure resources using a managed identity. This option is discussed in more detail below in the section authentication in server environments.
- **When an app is hosted and deployed on-premises** - The app should authenticate to Azure resources using an application service principal. This option is discussed in more detail below in the section authentication in server environments.

DefaultAzureCredential

The DefaultAzureCredential class provided by the Azure SDK allows apps to use different authentication methods depending on the environment they're run in. This allows apps to be promoted from local development to test environments to production without code changes. You configure the appropriate authentication method for each environment and `DefaultAzureCredential` will automatically detect and use that authentication method. The use of `DefaultAzureCredential` should be preferred over manually coding conditional logic or feature flags to use different authentication methods in different environments.

Details about using the `DefaultAzureCredential` class are covered later in this article in the section Use DefaultAzureCredential in an application.

Advantages of token-based authentication

Token-based authentication is strongly recommended over using connection strings when building apps for Azure. Token-based authentication offers the following advantages over authenticating with connection strings.

- The token-based authentication methods described below allows you to establish the specific permissions needed by the app on the Azure resource. This follows the principle of least privilege. In contrast, a connection string grants full rights to the Azure resource.
- Whereas anyone or any app with a connection string can connect to an Azure resource, token-based authentication methods scope access to the resource to only the app(s) intended to access the resource.
- In the case of a managed identity, there is no application secret to store. This makes the app more secure because there's no connection string or application secret than can be compromised.
- The Azure.Identity package in the Azure SDK manages tokens for you behind the scenes. This makes using token based authentication as easy to use as a connection string.

Use of connection strings should be limited to initial proof of concept apps or development prototypes that don't access production or sensitive data. Otherwise, the token-based authentication classes available in the Azure SDK should always be preferred when authenticating to Azure resources.

Authentication in server environments

When hosting in a server environment, each application should be assigned a unique *application identity* per environment the application is run in. In Azure, an app identity is represented by a **service principal**, a special type of *security principal* intended to identify and authenticate apps to Azure. The type of service principal to use for your app depends on where your app is running.

Authentication method	Description
Apps hosted in Azure	Apps hosted in Azure should use a **Managed Identity service principal**. Managed identities are designed to represent the identity of an app hosted in Azure and can only be used with Azure hosted apps. For example, a .NET web app hosted in Azure App Service would be assigned a Managed Identity. The Managed Identity assigned to the app would then be used to authenticate the app to other Azure services. **Learn about auth from Azure-hosted apps**
Apps hosted outside of Azure (for example on-premises apps)	Apps hosted outside of Azure (for example on-premises apps) that need to connect to Azure services should use an **Application service principal**. An Application service principal represents the identity of the app in Azure and is created through the application registration process. For example, consider a .NET web app hosted on-premises that makes use of Azure Blob Storage. You would create an Application service principal for the app using the App registration process. The `AZURE_CLIENT_ID`, `AZURE_TENANT_ID`, and `AZURE_CLIENT_SECRET` would all be stored as environment variables to be read by the application at runtime and allow the app to authenticate to Azure using the Application service principal. **Learn about auth from apps hosted outside of Azure**

Authentication during local development

When an application is run on a developer's workstation during local development, it still must authenticate to any Azure services used by the app. The two main strategies for authenticating apps to Azure during local development are:

Authentication method	Description
Create dedicated application service principal objects to be used during local development	In this method, dedicated **application service principal** objects are set up using the App registration process for use during local development. The identity of the service principal is then stored as environment variables to be accessed by the app when it is run in local development. This method allows you to assign the specific resource permissions needed by the app to the service principal objects used by developers during local development. This makes sure the application only has access to the specific resources it needs and replicates the permissions the app will have in production. The downside of this approach is the need to create separate service principal objects for each developer that works on an application. Learn about auth from Azure-hosted apps
Authenticate the app to Azure using the developer's credentials during local development	In this method, a developer must be signed-in to Azure from either Visual Studio, the Azure Tools extension for VS Code, the Azure CLI, or Azure PowerShell on their local workstation. The application then can access the developer's credentials from the credential store and use those credentials to access Azure resources from the app. This method has the advantage of easier setup since a developer only needs to login to their Azure account from Visual Studio, VS Code or the Azure CLI. The disadvantage of this approach is that the developer's account likely has more permissions than required by the application, therefore not properly replicating the permissions the app will run with in production. Learn about auth from Azure-hosted apps

Use DefaultAzureCredential in an application

`DefaultAzureCredential` supports multiple authentication methods and determines the authentication method being used at runtime. In this way, your app can use different authentication methods in different environments without implementing environment specific code.

The order and locations in which `DefaultAzureCredential` looks for credentials is found at DefaultAzureCredential.

To implement `DefaultAzureCredential`, first add the Azure.Identity and optionally the Microsoft.Extensions.Azure packages to your application. You can do this using either the command line or the NuGet Package Manager.

Command Line

Open a terminal environment of your choice in the application project directory and enter the command below.

terminal

```
dotnet add package Azure.Identity
dotnet add package Microsoft.Extensions.Azure
```

Azure services are generally accessed using corresponding client classes from the SDK. These classes and your own custom services should be registered in the `Program.cs` file so they can be accessed via dependency injection throughout your app. Inside of `Program.cs`, follow the steps below to correctly setup your service and `DefaultAzureCredential`.

1. Include the `Azure.Identity` and `Microsoft.Extensions.Azure` namespaces with a using statement.
2. Register the Azure service using relevant helper methods.
3. Pass an instance of the `DefaultAzureCredential` object to the `UseCredential` method.

An example of this is shown in the following code segment.

c#

```
using Microsoft.Extensions.Azure;
using Azure.Identity;

// Inside of Program.cs
builder.Services.AddAzureClients(x =>
{
    x.AddBlobServiceClient(new Uri("https://<account-
name>.blob.core.windows.net"));
    x.UseCredential(new DefaultAzureCredential());
});
```

Alternatively, you can also utilize `DefaultAzureCredential` in your services more directly without the help of additional Azure registration methods, as seen below.

```c#
using Azure.Identity;

// Inside of Program.cs
builder.Services.AddSingleton<BlobServiceClient>(x =>
    new BlobServiceClient(
        new Uri("https://<account-name>.blob.core.windows.net"),
        new DefaultAzureCredential()));
```

When the above code is run on your local workstation during local development, it will look in the environment variables for an application service principal or at Visual Studio, VS Code, the Azure CLI, or Azure PowerShell for a set of developer credentials, either of which can be used to authenticate the app to Azure resources during local development.

When deployed to Azure this same code can also authenticate your app to other Azure resources. `DefaultAzureCredential` can retrieve environment settings and managed identity configurations to authenticate to other services automatically.

Exploring the sequence of DefaultAzureCredential authentication methods

Internally, `DefaultAzureCredential` implements a chain of credential providers for authenticating applications to Azure resources. Each credential provider is able to detect if credentials of that type are configured for the app. `DefaultAzureCredential` sequentially checks each provider in order and uses the credentials from the first provider that has credentials configured.

The order and locations in which `DefaultAzureCredential` looks for credentials is found at DefaultAzureCredential.

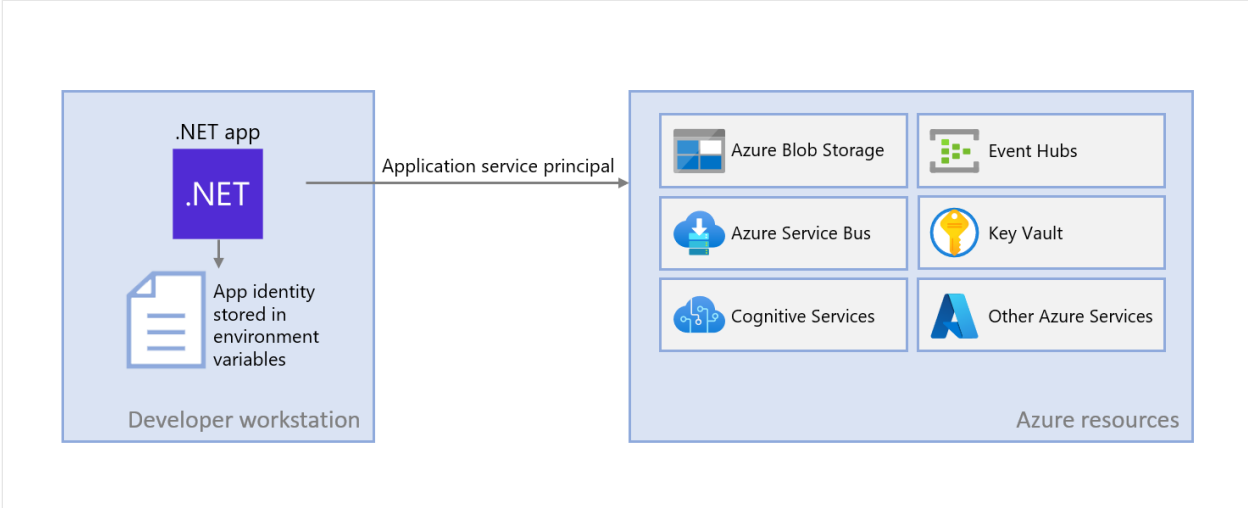
Credential type	Description
Application service principal	`DefaultAzureCredential` reads a set of environment variables to determine if an application service principal (application user) has been set for the app. If so, `DefaultAzureCredential` uses these values to authenticate the app to Azure. This method is most often used in server environments but can also be used when developing locally.

Credential type	Description
Managed Identity	If the application is deployed to an Azure host with Managed Identity enabled, `DefaultAzureCredential` will authenticate the app to Azure using that Managed Identity. Authentication using a Managed Identity is discussed in the Authentication in server environments section of this document. This method is only available when an application is hosted in Azure using a service like Azure App Service, Azure Functions, or Azure Virtual Machines.
Visual Studio	If the developer has authenticated to Azure by logging into Visual Studio, `DefaultAzureCredential` will authenticate the app to Azure using that same account.
Visual Studio Code	If the developer has authenticated to Azure using the Visual Studio Code Azure Account plugin, `DefaultAzureCredential` will authenticate the app to Azure using that same account.
Azure CLI	If a developer has authenticated to Azure using the `az login` command in the Azure CLI, `DefaultAzureCredential` will authenticate the app to Azure using that same account.
Azure PowerShell	If a developer has authenticated to Azure using the `Connect-AzAccount` cmdlet from Azure PowerShell, `DefaultAzureCredential` will authenticate the app to Azure using that same account.
Interactive	If enabled, `DefaultAzureCredential` will interactively authenticate the developer via the current system's default browser. By default, this option is disabled.

Authenticate NET apps to Azure services during local development using service principals

Article • 08/03/2023

When creating cloud applications, developers need to debug and test applications on their local workstation. When an application is run on a developer's workstation during local development, it still must authenticate to any Azure services used by the app. This article covers how to set up dedicated application service principal objects to be used during local development.



Dedicated application service principals for local development allow you to follow the principle of least privilege during app development. Since permissions are scoped to exactly what is needed for the app during development, app code is prevented from accidentally accessing an Azure resource intended for use by a different app. This also prevents bugs from occurring when the app is moved to production because the app was overprivileged in the dev environment.

An application service principal is set up for the app when the app is registered in Azure. When registering apps for local development, it's recommended to:

- Create separate app registrations for each developer working on the app. This will create separate application service principals for each developer to use during local development and avoid the need for developers to share credentials for a single application service principal.
- Create separate app registrations per app. This scopes the app's permissions to only what is needed by the app.

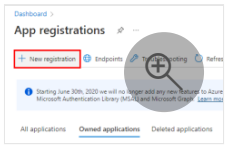
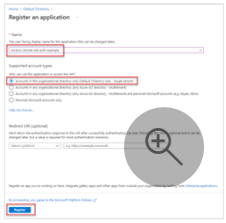
During local development, environment variables are set with the application service principal's identity. The Azure SDK for NET reads these environment variables and uses this information to authenticate the app to the Azure resources it needs.

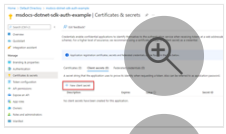
1 - Register the application in Azure

Application service principal objects are created with an app registration in Azure. This can be done using either the Azure portal or Azure CLI.

Azure portal

Sign in to the Azure portal and follow these steps.

Instructions	Screenshot
In the Azure portal: 1. Enter *app registrations* in the search bar at the top of the Azure portal. 2. Select the item labeled **App registrations** under the **Services** heading on the menu that appears below the search bar.	
On the **App registrations** page, select **+ New registration**.	
On the **Register an application** page, fill out the form as follows. 1. **Name** → Enter a name for the app registration in Azure. It is recommended this name include the app name, the user the app registration is for, and an identifier like 'dev' to indicate this app registration is for use in local development. 2. **Supported account types** → *Accounts in this organizational directory only*. Select **Register** to register your app and create the application service principal.	
On the App registration page for your app: 1. **Application (client) ID** → This is the app id the app will use to access Azure during local development. Copy this value to a temporary location in a text editor as you will need it in a future step. 2. **Directory (tenant) id** → This value will also be needed by your app when it authenticates to Azure. Copy this value to a	

Instructions	Screenshot
temporary location in a text editor it will also be needed it in a future step. 3. **Client credentials** → You must set the client credentials for the app before your app can authenticate to Azure and use Azure services. Select *Add a certificate or secret* to add credentials for your app.	
On the Certificates & secrets page, select + **New client secret**.	
The **Add a client secret** dialog will pop out from the right-hand side of the page. In this dialog: 1. **Description** → Enter a value of *Current*. 2. **Expires** → Select a value of *24 months*. Select **Add** to add the secret.	
On the *Certificates & secrets* page, you will be shown the value of the client secret. Copy this value to a temporary location in a text editor as you will need it in a future step. ***IMPORTANT: This is the only time you will see this value.*** Once you leave or refresh this page, you will not be able to see this value again. You may add an additional client secret without invalidating this client secret, but you will not see this value again.	

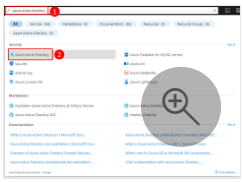
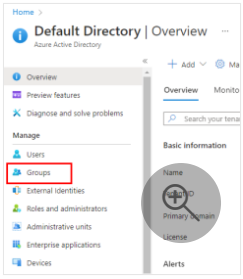
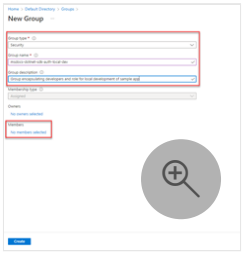
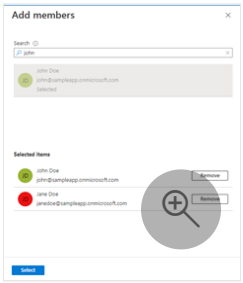
2 - Create an Azure AD security group for local development

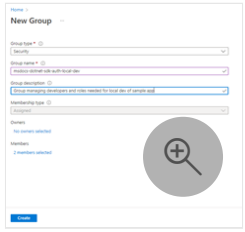
Since there typically multiple developers who work on an application, it's recommended to create an Azure AD group to encapsulate the roles (permissions) the app needs in local development rather than assigning the roles to individual service principal objects. This offers the following advantages.

- Every developer is assured to have the same roles assigned since roles are assigned at the group level.
- If a new role is needed for the app, it only needs to be added to the Azure AD group for the app.

- If a new developer joins the team, a new application service principal is created for the developer and added to the group, assuring the developer has the right permissions to work on the app.

Azure portal

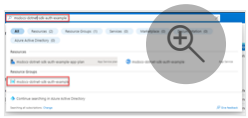
Instructions	Screenshot
Navigate to the Azure Active Directory page in the Azure portal by typing *Azure Active Directory* into the search box at the top of the page and then selecting *Azure Active Directory* from under services.	
On the *Azure Active Directory* page, select **Groups** from the left-hand menu.	
On the *All groups* page, select **New group**.	
On the **New Group** page: 1. **Group type** → **Security** 2. **Group name** → A name for the security group, typically created from the application name. It is also helpful to include a string like *local-dev* in the name of the group to indicate the purpose of the group. 3. **Group description** → A description of the purpose of the group. 4. Select the **No members selected** link under **Members** to add members to the group.	
On the **Add members** dialog box: 1. Use the search box to filter the list of principal names in the list. 2. Select the application service principals for local development for this app. As objects are selected, they will be greyed out and move to the *Selected items* list at the bottom of the dialog. 3. When finished, select the **Select** button.	

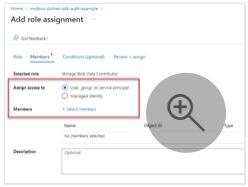
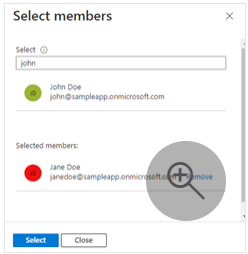
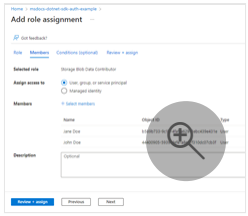
Instructions	Screenshot
Back on the **New group** page, select **Create** to create the group. The group will be created and you will be taken back to the **All groups** page. It may take up to 30 seconds for the group to appear and you may need to refresh the page due to caching in the Azure portal.	

3 - Assign roles to the application

Next, you need to determine what roles (permissions) your app needs on what resources and assign those roles to your app. In this example, the roles will be assigned to the Azure Active Directory group created in step 2. Roles can be assigned a role at a resource, resource group, or subscription scope. This example will show how to assign roles at the resource group scope since most applications group all their Azure resources into a single resource group.

Azure portal

Instructions	Screenshot
Locate the resource group for your application by searching for the resource group name using the search box at the top of the Azure portal. Navigate to your resource group by selecting the resource group name under the *Resource Groups* heading in the dialog box.	
On the page for the resource group, select *Access control (IAM)* from the left-hand menu.	
On the *Access control (IAM)* page: 1. Select the *Role assignments* tab. 2. Select + *Add* from the top menu and then *Add role assignment* from the resulting drop-down menu.	
The *Add role assignment* page lists all of the roles that can be assigned for the resource group. 1. Use the search box to filter the list to a more manageable size. This example shows how to filter for Storage Blob roles.	

Instructions	Screenshot
2. Select the role that you want to assign. Select *Next* to go to the next screen.	
The next *Add role assignment* page allows you to specify what user to assign the role to. 1. Select *User, group, or service principal* under *Assign access to*. 2. Select + *Select members* under *Members* A dialog box will open on the right-hand side of the Azure portal.	
In the *Select members* dialog: 1. The *Select* text box can be used to filter the list of users and groups in your subscription. If needed, type the first few characters of the local development Azure AD group you created for the app. 2. Select the local development Azure AD group associated with your application. Select *Select* at the bottom of the dialog to continue.	
The Azure AD group will now show as selected on the *Add role assignment* screen. Select *Review + assign* to go to the final page and then *Review + assign* again to complete the process.	

4 - Set application environment variables

The `DefaultAzureCredential` object will look for the service principal information in a set of environment variables at runtime. There are multiple ways to configure environment variables when working with .NET depending on your tooling and environment.

Regardless of which approach you choose, you'll need to configure the following environment variables when working with a service principal.

- `AZURE_CLIENT_ID` → The app ID value.
- `AZURE_TENANT_ID` → The tenant ID value.
- `AZURE_CLIENT_SECRET` → The password/credential generated for the app.

Windows

You can set environment variables for Windows from the command line. However, when using this approach the values are accessible to all applications running on that operating system and may cause conflicts if you aren't careful. Environment variables can be set at the user or system level.

Bash

```
# Set user environment variables
setx ASPNETCORE_ENVIRONMENT "Development"
setx AZURE_CLIENT_ID "00000000-0000-0000-0000-000000000000"
setx AZURE_TENANT_ID "11111111-1111-1111-1111-111111111111"
setx AZURE_CLIENT_SECRET "=abcdefghijklmnopqrstuvwxyz"

# Set system environment variables - requires running as admin
setx ASPNETCORE_ENVIRONMENT "Development"
setx AZURE_CLIENT_ID "00000000-0000-0000-0000-000000000000" /m
setx AZURE_TENANT_ID "11111111-1111-1111-1111-111111111111" /m
setx AZURE_CLIENT_SECRET "=abcdefghijklmnopqrstuvwxyz" /m
```

PowerShell can also be used to set environment variables at the user or machine level.

PowerShell

```
# Set user environment variables
[Environment]::SetEnvironmentVariable("ASPNETCORE_ENVIRONMENT",
"Development", "User")
[Environment]::SetEnvironmentVariable("AZURE_CLIENT_ID", "00000000-0000-
0000-0000-000000000000", "User")
[Environment]::SetEnvironmentVariable("AZURE_TENANT_ID", "11111111-1111-
1111-1111-111111111111", "User")
[Environment]::SetEnvironmentVariable("AZURE_CLIENT_SECRET",
"=abcdefghijklmnopqrstuvwxyz", "User")

# Set system environment variables - requires running as admin
[Environment]::SetEnvironmentVariable("ASPNETCORE_ENVIRONMENT",
"Development", "Machine")
[Environment]::SetEnvironmentVariable("AZURE_CLIENT_ID", "00000000-0000-
0000-0000-000000000000", "Machine")
[Environment]::SetEnvironmentVariable("AZURE_TENANT_ID", "11111111-1111-
1111-1111-111111111111", "Machine")
[Environment]::SetEnvironmentVariable("AZURE_CLIENT_SECRET",
"=abcdefghijklmnopqrstuvwxyz", "Machine")
```

5 - Implement DefaultAzureCredential in your application

`DefaultAzureCredential` supports multiple authentication methods and determines the authentication method being used at runtime. In this way, your app can use different authentication methods in different environments without implementing environment specific code.

The order and locations in which `DefaultAzureCredential` looks for credentials is found at DefaultAzureCredential.

To implement `DefaultAzureCredential`, first add the Azure.Identity and optionally the Microsoft.Extensions.Azure packages to your application. You can do this using either the command line or the NuGet Package Manager.

Command Line

Open a terminal environment of your choice in the application project directory and enter the command below.

terminal

```
dotnet add package Azure.Identity
dotnet add package Microsoft.Extensions.Azure
```

Azure services are generally accessed using corresponding client classes from the SDK. These classes and your own custom services should be registered in the `Program.cs` file so they can be accessed via dependency injection throughout your app. Inside of `Program.cs`, follow the steps below to correctly setup your service and `DefaultAzureCredential`.

1. Include the `Azure.Identity` and `Microsoft.Extensions.Azure` namespaces with a using statement.
2. Register the Azure service using relevant helper methods.
3. Pass an instance of the `DefaultAzureCredential` object to the `UseCredential` method.

An example of this is shown in the following code segment.

c#

```
using Microsoft.Extensions.Azure;
using Azure.Identity;

// Inside of Program.cs
builder.Services.AddAzureClients(x =>
{
```

```
    x.AddBlobServiceClient(new Uri("https://<account-
name>.blob.core.windows.net"));
    x.UseCredential(new DefaultAzureCredential());
});
```

Alternatively, you can also utilize `DefaultAzureCredential` in your services more directly without the help of additional Azure registration methods, as seen below.

c#

```
using Azure.Identity;

// Inside of Program.cs
builder.Services.AddSingleton<BlobServiceClient>(x =>
    new BlobServiceClient(
        new Uri("https://<account-name>.blob.core.windows.net"),
        new DefaultAzureCredential()));
```

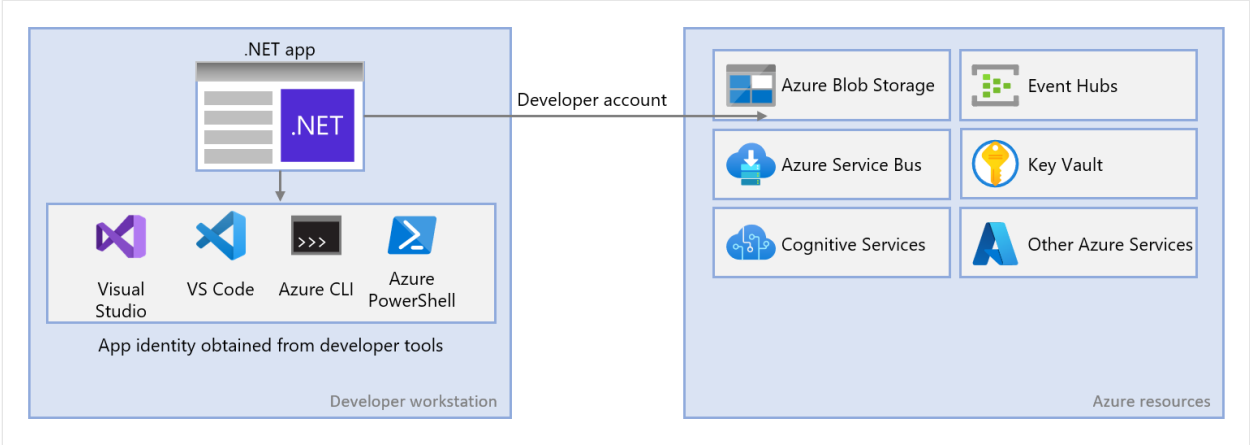
When the above code is run on your local workstation during local development, it will look in the environment variables for an application service principal or at Visual Studio, VS Code, the Azure CLI, or Azure PowerShell for a set of developer credentials, either of which can be used to authenticate the app to Azure resources during local development.

When deployed to Azure this same code can also authenticate your app to other Azure resources. `DefaultAzureCredential` can retrieve environment settings and managed identity configurations to authenticate to other services automatically.

Authenticate .NET apps to Azure services during local development using developer accounts

Article • 08/03/2023

When creating cloud applications, developers need to debug and test applications on their local workstation. When an application is run on a developer's workstation during local development, it still must authenticate to any Azure services used by the app. This article covers how to use a developer's Azure credentials to authenticate the app to Azure during local development.



For an app to authenticate to Azure during local development using the developer's Azure credentials, the developer must be signed-in to Azure from the VS Code Azure Tools extension, the Azure CLI, or Azure PowerShell. The Azure SDK for .NET is able to detect that the developer is signed-in from one of these tools and then obtain the necessary credentials from the credentials cache to authenticate the app to Azure as the signed-in user.

This approach is easiest to set up for a development team since it takes advantage of the developers' existing Azure accounts. However, a developer's account will likely have more permissions than required by the application, therefore exceeding the permissions the app will run with in production. As an alternative, you can create application service principals to use during local development which can be scoped to have only the access needed by the app.

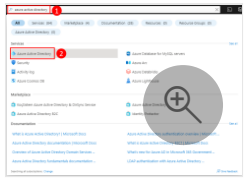
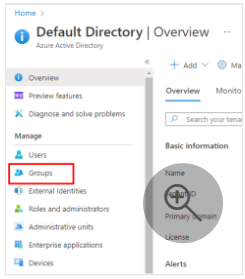
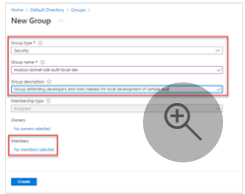
1 - Create Azure AD group for local development

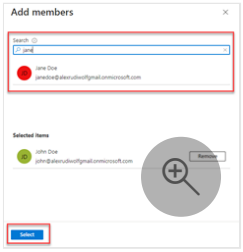
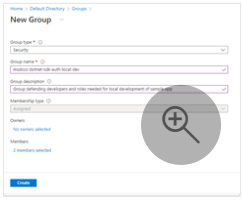
Since there are almost always multiple developers who work on an application, it's recommended to first create an Azure AD group to encapsulate the roles (permissions) the app needs in local development. This offers the following advantages.

- Every developer is assured to have the same roles assigned since roles are assigned at the group level.
- If a new role is needed for the app, it only needs to be added to the Azure AD group for the app.
- If a new developer joins the team, they simply must be added to the correct Azure AD group to get the correct permissions to work on the app.

If you have an existing Azure AD group for your development team, you can use that group. Otherwise, complete the following steps to create an Azure AD group.

Azure portal

Instructions	Screenshot
Navigate to the Azure Active Directory page in the Azure portal by typing *Azure Active Directory* into the search box at the top of the page and then selecting *Azure Active Directory* from under services.	
On the *Azure Active Directory* page, select **Groups** from the left-hand menu.	
On the *All groups* page, select **New group**.	
On the **New Group** page: 1. **Group type** → **Security** 2. **Group name** → A name for the security group, typically created from the application name. It is also helpful to include a string like *local-dev* in the name of the group to indicate the purpose of the group. 3. **Group description** → A description of the purpose of the group. 4. Select the **No members selected** link under **Members** to add members to the group.	

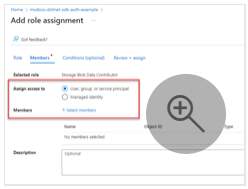
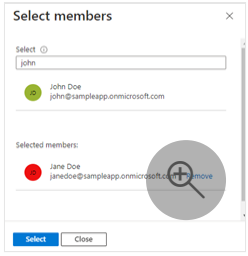
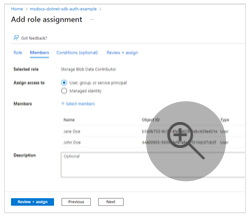
Instructions	Screenshot
On the **Add members** dialog box: 1. Use the search box to filter the list of user names in the list. 2. Select the user(s) for local development for this app. As objects are selected, they will move to the *Selected items* list at the bottom of the dialog. 3. When finished, select the **Select** button.	
Back on the **New group** page, select **Create** to create the group. The group will be created and you will be taken back to the **All groups** page. It may take up to 30 seconds for the group to appear and you may need to refresh the page due to caching in the Azure portal.	

2 - Assign roles to the Azure AD group

Next, you need to determine what roles (permissions) your app needs on what resources and assign those roles to your app. In this example, the roles will be assigned to the Azure Active Directory group created in step 1. Roles can be assigned a role at a resource, resource group, or subscription scope. This example will show how to assign roles at the resource group scope since most applications group all their Azure resources into a single resource group.

Azure portal

Instructions	Screenshot
Locate the resource group for your application by searching for the resource group name using the search box at the top of the Azure portal. Navigate to your resource group by selecting the resource group name under the *Resource Groups* heading in the dialog box.	
On the page for the resource group, select *Access control (IAM)* from the left-hand menu.	
On the *Access control (IAM)* page: 1. Select the *Role assignments* tab.	

Instructions	Screenshot
2. Select + *Add* from the top menu and then *Add role assignment* from the resulting drop-down menu.	
The *Add role assignment* page lists all of the roles that can be assigned for the resource group. 1. Use the search box to filter the list to a more manageable size. This example shows how to filter for Storage Blob roles. 2. Select the role that you want to assign. Select *Next* to go to the next screen.	
The next *Add role assignment* page allows you to specify what user to assign the role to. 1. Select *User, group, or service principal* under *Assign access to*. 2. Select + *Select members* under *Members* A dialog box will open on the right-hand side of the Azure portal.	
In the *Select members* dialog: 1. The *Select* text box can be used to filter the list of users and groups in your subscription. If needed, type the first few characters of the local development Azure AD group you created for the app. 2. Select the local development Azure AD group associated with your application. Select *Select* at the bottom of the dialog to continue.	
The Azure AD group will now show as selected on the *Add role assignment* screen. Select *Review + assign* to go to the final page and then *Review + assign* again to complete the process.	

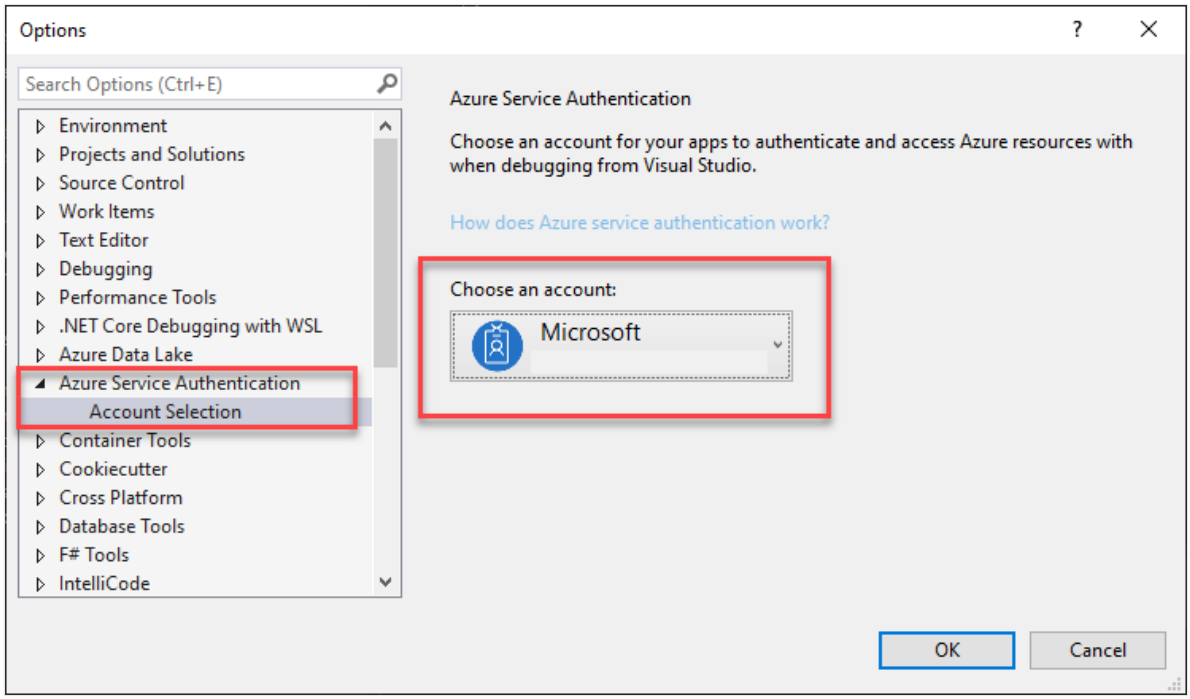
3 - Sign-in to Azure using .NET Tooling

Next you need to sign in to Azure using one of several .NET tooling options. The account you sign into should also exist in the Azure Active Directory group you created and configured earlier.

Visual Studio

On the top menu of Visual Studio, navigate to **Tools** > **Options** to open the options dialog. In the search bar in the upper left, type *Azure* to filter the options. Under the **Azure Service Authentication**, choose **Account Selection**.

Select the drop-down menu under **Choose an account** and choose to add a Microsoft Account. A window will open prompting you to pick an account. Enter the credentials for your desired Azure account, and then select the confirmation.



4 - Implement DefaultAzureCredential in your application

`DefaultAzureCredential` supports multiple authentication methods and determines the authentication method being used at runtime. In this way, your app can use different authentication methods in different environments without implementing environment specific code.

The order and locations in which `DefaultAzureCredential` looks for credentials is found at DefaultAzureCredential.

To implement `DefaultAzureCredential`, first add the Azure.Identity and optionally the Microsoft.Extensions.Azure packages to your application. You can do this using either the command line or the NuGet Package Manager.

Command Line

Open a terminal environment of your choice in the application project directory and enter the command below.

terminal

```
dotnet add package Azure.Identity
dotnet add package Microsoft.Extensions.Azure
```

Azure services are generally accessed using corresponding client classes from the SDK. These classes and your own custom services should be registered in the `Program.cs` file so they can be accessed via dependency injection throughout your app. Inside of `Program.cs`, follow the steps below to correctly setup your service and `DefaultAzureCredential`.

1. Include the `Azure.Identity` and `Microsoft.Extensions.Azure` namespaces with a using statement.
2. Register the Azure service using relevant helper methods.
3. Pass an instance of the `DefaultAzureCredential` object to the `UseCredential` method.

An example of this is shown in the following code segment.

c#

```
using Microsoft.Extensions.Azure;
using Azure.Identity;

// Inside of Program.cs
builder.Services.AddAzureClients(x =>
{
    x.AddBlobServiceClient(new Uri("https://<account-
name>.blob.core.windows.net"));
    x.UseCredential(new DefaultAzureCredential());
});
```

Alternatively, you can also utilize `DefaultAzureCredential` in your services more directly without the help of additional Azure registration methods, as seen below.

c#

```
using Azure.Identity;

// Inside of Program.cs
builder.Services.AddSingleton<BlobServiceClient>(x =>
    new BlobServiceClient(
```

```
        new Uri("https://<account-name>.blob.core.windows.net"),
        new DefaultAzureCredential()));
```

When the above code is run on your local workstation during local development, it will look in the environment variables for an application service principal or at Visual Studio, VS Code, the Azure CLI, or Azure PowerShell for a set of developer credentials, either of which can be used to authenticate the app to Azure resources during local development.

When deployed to Azure this same code can also authenticate your app to other Azure resources. `DefaultAzureCredential` can retrieve environment settings and managed identity configurations to authenticate to other services automatically.

Authenticating Azure-hosted apps to Azure resources with the Azure SDK for .NET

Article • 08/03/2023

When an app is hosted in Azure using a service like Azure App Service, Azure Virtual Machines, or Azure Container Instances, the recommended approach to authenticating an app to Azure resources is to use a managed identity.

A managed identity provides an identity for your app such that it can connect to other Azure resources without the need to use a secret key or other application secret. Internally, Azure knows the identity of your app and what resources it's allowed to connect to. Azure uses this information to automatically obtain Microsoft Entra tokens for the app to allow it to connect to other Azure resources, all without you having to manage any application secrets.

Managed identity types

There are two types of managed identities:

- **System-assigned managed identities** - This type of managed identity is provided by and tied directly to an Azure resource. When you enable managed identity on an Azure resource, you get a system-assigned managed identity for that resource. A system-assigned managed identity is tied to the lifecycle of the Azure resource it's associated with. When the resource is deleted, Azure automatically deletes the identity for you. Since all you have to do is enable managed identity for the Azure resource hosting your code, this is the easiest type of managed identity to use.
- **User-assigned managed identities** - You may also create a managed identity as a standalone Azure resource. This is most frequently used when your solution has multiple workloads that run on multiple Azure resources that all need to share the same identity and same permissions. For example, if your solution had components that ran on multiple App Service and virtual machine instances that all needed access to the same set of Azure resources, creating and using a user-assigned managed identity across those resources would make sense.

This article will cover the steps to enable and use a system-assigned managed identity for an app. If you need to use a user-assigned managed identity, see the article Manage user-assigned managed identities to see how to create a user-assigned managed identity.

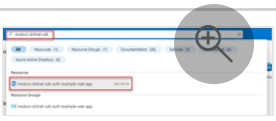
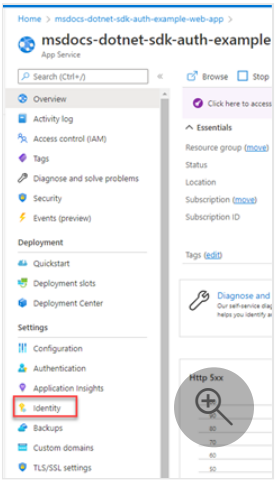
1 - Enable managed identity in the Azure resource hosting the app

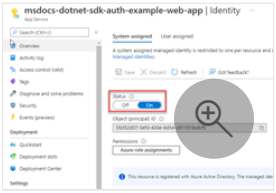
The first step is to enable managed identity on Azure resource hosting your app. For example, if you're hosting a .NET application using Azure App Service, you need to enable managed identity for the App Service web app that is hosting your app. If you were using a virtual machine to host your app, you would enable your VM to use managed identity.

You can enable managed identity to be used for an Azure resource using either the Azure portal or the Azure CLI.

Azure portal

Expand table

Instructions	Screenshot
Navigate to the resource that hosts your application code in the Azure portal. For example, you can type the name of your resource in the search box at the top of the page and navigate to it by selecting it in the dialog box.	
On the page for your resource, select the *Identity* menu item from the left-hand menu. All Azure resources capable of supporting managed identity will have an *Identity* menu item even though the layout of the menu may vary slightly.	

Instructions	Screenshot
On the *Identity* page: 1. Change the *Status* slider to *On*. 2. Click *Save*. A confirmation dialog will verify you want to enable managed identity for your service. Answer *Yes* and managed identity will be enabled for the Azure resource.	

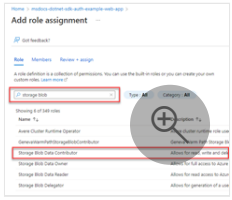
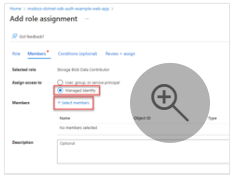
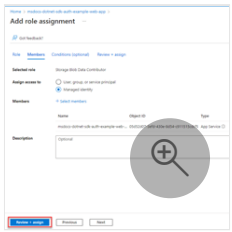
2 - Assign roles to the managed identity

Next, you need to determine what roles (permissions) your app needs and assign the managed identity to those roles in Azure. A managed identity can be assigned roles at a resource, resource group, or subscription scope. This example will show how to assign roles at the resource group scope since most applications group all their Azure resources into a single resource group.

Azure portal

Expand table

Instructions	Screenshot
Locate the resource group for your application by searching for the resource group name using the search box at the top of the Azure portal. Navigate to your resource group by selecting the resource group name under the *Resource Groups* heading in the dialog box.	
On the page for the resource group, select *Access control (IAM)* from the left-hand menu.	
On the *Access control (IAM)* page: 1. Select the *Role assignments* tab. 2. Select + *Add* from the top menu and then *Add role assignment* from the resulting drop-down menu.	

Instructions	Screenshot
The *Add role assignment* page lists all of the roles that can be assigned for the resource group. 1. Use the search box to filter the list to a more manageable size. This example shows how to filter for Storage Blob roles. 2. Select the role that you want to assign. Select *Next* to go to the next screen.	
The next *Add role assignment* page allows you to specify what user to assign the role to. 1. Select *Managed identity* under *Assign access to*. 2. Select + *Select members* under *Members* A dialog box will open on the right-hand side of the Azure portal.	
In the *Select managed identities* dialog: 1. The *Managed identity* dropdown and *Select* text box can be used to filter the list of managed identities in your subscription. In this example by selecting *App Service*, only managed identities associated with an App Service are displayed. 2. Select the managed identity for the Azure resource hosting your application. Select *Select* at the bottom of the dialog to continue.	
The managed identity will now show as selected on the *Add role assignment* screen. Select *Review + assign* to go to the final page and then *Review + assign* again to complete the process.	

3 - Implement DefaultAzureCredential in your application

`DefaultAzureCredential` supports multiple authentication methods and determines the authentication method being used at runtime. In this way, your app can use different authentication methods in different environments without implementing environment specific code.

The order and locations in which `DefaultAzureCredential` looks for credentials is found at DefaultAzureCredential.

To implement `DefaultAzureCredential`, first add the Azure.Identity and optionally the Microsoft.Extensions.Azure packages to your application. You can do this using either the command line or the NuGet Package Manager.

Command Line

Open a terminal environment of your choice in the application project directory and enter the command below.

.NET CLI

```
dotnet add package Azure.Identity
dotnet add package Microsoft.Extensions.Azure
```

Azure services are generally accessed using corresponding client classes from the SDK. These classes and your own custom services should be registered in the `Program.cs` file so they can be accessed via dependency injection throughout your app. Inside of `Program.cs`, follow the steps below to correctly setup your service and `DefaultAzureCredential`.

1. Include the `Azure.Identity` and `Microsoft.Extensions.Azure` namespaces with a using statement.
2. Register the Azure service using relevant helper methods.
3. Pass an instance of the `DefaultAzureCredential` object to the `UseCredential` method.

An example of this is shown in the following code segment.

c#

```
using Microsoft.Extensions.Azure;
using Azure.Identity;

// Inside of Program.cs
builder.Services.AddAzureClients(x =>
{
    x.AddBlobServiceClient(new Uri("https://<account-
name>.blob.core.windows.net"));
    x.UseCredential(new DefaultAzureCredential());
});
```

Alternatively, you can also utilize `DefaultAzureCredential` in your services more directly without the help of additional Azure registration methods, as seen below.

```c#
using Azure.Identity;

// Inside of Program.cs
builder.Services.AddSingleton<BlobServiceClient>(x =>
    new BlobServiceClient(
        new Uri("https://<account-name>.blob.core.windows.net"),
        new DefaultAzureCredential()));
```

When the above code is run on your local workstation during local development, it will look in the environment variables for an application service principal or at Visual Studio, VS Code, the Azure CLI, or Azure PowerShell for a set of developer credentials, either of which can be used to authenticate the app to Azure resources during local development.

When deployed to Azure this same code can also authenticate your app to other Azure resources. `DefaultAzureCredential` can retrieve environment settings and managed identity configurations to authenticate to other services automatically.

Authenticate to Azure resources from .NET apps hosted on-premises

Article • 08/03/2023

Apps hosted outside of Azure (for example on-premises or at a third-party data center) should use an application service principal to authenticate to Azure when accessing Azure resources. Application service principal objects are created using the app registration process in Azure. When an application service principal is created, a client ID and client secret will be generated for your app. The client ID, client secret, and your tenant ID are then stored in environment variables so they can be used by the Azure SDK for .NET to authenticate your app to Azure at runtime.

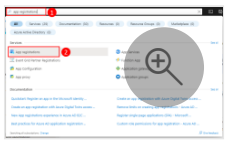
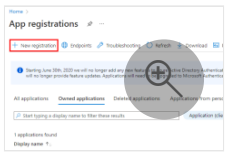
A different app registration should be created for each environment the app is hosted in. This allows environment specific resource permissions to be configured for each service principal and make sure an app deployed to one environment does not talk to Azure resources that are part of another environment.

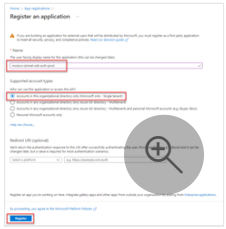
1 - Register the application in Azure

An app can be registered with Azure using either the Azure portal or the Azure CLI.

Azure portal

Sign in to the Azure portal and follow these steps.

Instructions	Screenshot
In the Azure portal: 1. Enter *app registrations* in the search bar at the top of the Azure portal. 2. Select the item labeled **App registrations** under the under **Services** heading on the menu that appears below the search bar.	
On the **App registrations** page, select **+ New registration**.	

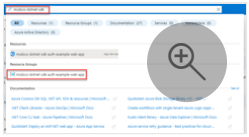
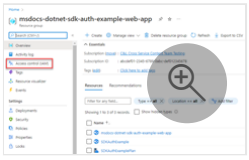
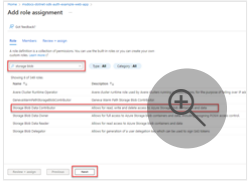
Instructions	Screenshot
On the **Register an application** page, fill out the form as follows. 1. **Name** → Enter a name for the app registration in Azure. It is recommended this name include the app name and environment (test, prod) the app registration is for. 2. **Supported account types** → *Accounts in this organizational directory only*. Select **Register** to register your app and create the application service principal.	
On the App registration page for your app: 1. **Application (client) ID** → This is the app id the app will use to access Azure during local development. Copy this value to a temporary location in a text editor as you will need it in a future step. 2. **Directory (tenant) id** → This value will also be needed by your app when it authenticates to Azure. Copy this value to a temporary location in a text editor it will also be needed it in a future step. 3. **Client credentials** → You must set the client credentials for the app before your app can authenticate to Azure and use Azure services. Select *Add a certificate or secret* to add credentials for your app.	
On the Certificates & secrets page, select **+ New client secret**.	
The **Add a client secret** dialog will pop out from the right-hand side of the page. In this dialog: 1. **Description** → Enter a value of *Current*. 2. **Expires** → Select a value of *24 months*. Select **Add** to add the secret. ***IMPORTANT: Set a reminder in your calendar prior to the expiration date of the secret.*** This way, you can add a new secret prior and update your apps prior to the expiration of this secret and avoid a service interruption in your app.	
On the *Certificates & secrets* page, you will be shown the value of the client secret. Copy this value to a temporary location in a text editor as you will need it in a future step.	

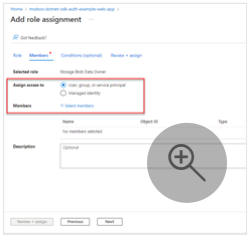
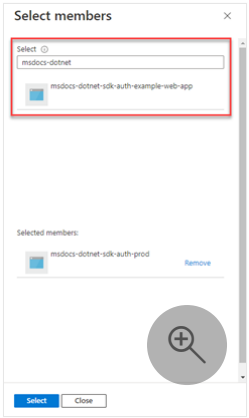
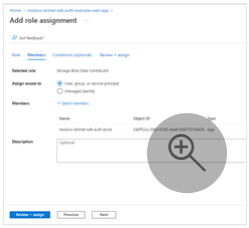
Instructions	Screenshot
IMPORTANT: This is the only time you will see this value. Once you leave or refresh this page, you will not be able to see this value again. You may add an additional client secret without invalidating this client secret, but you will not see this value again.	

2 - Assign roles to the application service principal

Next, you need to determine what roles (permissions) your app needs on what resources and assign those roles to your app. Roles can be assigned a role at a resource, resource group, or subscription scope. This example will show how to assign roles for the service principal at the resource group scope since most applications group all their Azure resources into a single resource group.

Azure portal

Instructions	Screenshot
Locate the resource group for your application by searching for the resource group name using the search box at the top of the Azure portal. Navigate to your resource group by selecting the resource group name under the *Resource Groups* heading in the dialog box.	
On the page for the resource group, select *Access control (IAM)* from the left-hand menu.	
On the *Access control (IAM)* page: 1. Select the *Role assignments* tab. 2. Select + *Add* from the top menu and then *Add role assignment* from the resulting drop-down menu.	
The *Add role assignment* page lists all of the roles that can be assigned for the resource group. 1. Use the search box to filter the list to a more manageable size. This example shows how to filter for Storage Blob roles.	

Instructions	Screenshot
2. Select the role that you want to assign. Select *Next* to go to the next screen.	
The next *Add role assignment* page allows you to specify what user to assign the role to. 1. Select *User, group, or service principal* under *Assign access to*. 2. Select + *Select members* under *Members* A dialog box will open on the right-hand side of the Azure portal.	
In the *Select members* dialog: 1. The *Select* text box can be used to filter the list of users and groups in your subscription. If needed, type the first few characters of the service principal you created for the app to filter the list. 2. Select the service principal associated with your application. Select *Select* at the bottom of the dialog to continue.	
The service principal will now show as selected on the *Add role assignment* screen. Select *Review + assign* to go to the final page and then *Review + assign* again to complete the process.	

3 - Configure environment variables for application

The `DefaultAzureCredential` object will look for service principal credentials in a set of environment variables at runtime. There are multiple ways to configure environment variables when working with .NET depending on your tooling and environment.

Regardless of which approach you choose, you will need to configure the following environment variables when working with a service principal.

- `AZURE_CLIENT_ID` → The app ID value.
- `AZURE_TENANT_ID` → The tenant ID value.
- `AZURE_CLIENT_SECRET` → The password/credential generated for the app.

Windows

You can set environment variables for Windows from the command line. However, when using this approach the values are accessible to all applications running on that operating system and may cause conflicts if you are not careful. Environment variables can be set at either user or system level.

Bash

```
# Set user environment variables
setx ASPNETCORE_ENVIRONMENT "Development"
setx AZURE_CLIENT_ID "00000000-0000-0000-0000-000000000000"
setx AZURE_TENANT_ID "11111111-1111-1111-1111-111111111111"
setx AZURE_CLIENT_SECRET "=abcdefghijklmnopqrstuvwxyz"

# Set system environment variables - requires running as admin
setx ASPNETCORE_ENVIRONMENT "Development"
setx AZURE_CLIENT_ID "00000000-0000-0000-0000-000000000000" /m
setx AZURE_TENANT_ID "11111111-1111-1111-1111-111111111111" /m
setx AZURE_CLIENT_SECRET "=abcdefghijklmnopqrstuvwxyz" /m
```

You can also use PowerShell to set environment variables at either the user or machine level.

PowerShell

```
# Set user environment variables
[Environment]::SetEnvironmentVariable("ASPNETCORE_ENVIRONMENT",
"Development", "User")
[Environment]::SetEnvironmentVariable("AZURE_CLIENT_ID", "00000000-0000-
0000-0000-000000000000", "User")
[Environment]::SetEnvironmentVariable("AZURE_TENANT_ID", "11111111-1111-
1111-1111-111111111111", "User")
[Environment]::SetEnvironmentVariable("AZURE_CLIENT_SECRET",
"=abcdefghijklmnopqrstuvwxyz", "User")

# Set system environment variables - requires running as admin
[Environment]::SetEnvironmentVariable("ASPNETCORE_ENVIRONMENT",
"Development", "Machine")
[Environment]::SetEnvironmentVariable("AZURE_CLIENT_ID", "00000000-0000-
0000-0000-000000000000", "Machine")
[Environment]::SetEnvironmentVariable("AZURE_TENANT_ID", "11111111-1111-
1111-1111-111111111111", "Machine")
[Environment]::SetEnvironmentVariable("AZURE_CLIENT_SECRET",
"=abcdefghijklmnopqrstuvwxyz", "Machine")
```

4 - Implement DefaultAzureCredential in application

`DefaultAzureCredential` supports multiple authentication methods and determines the authentication method being used at runtime. In this way, your app can use different authentication methods in different environments without implementing environment specific code.

The order and locations in which `DefaultAzureCredential` looks for credentials is found at DefaultAzureCredential.

To implement `DefaultAzureCredential`, first add the Azure.Identity and optionally the Microsoft.Extensions.Azure packages to your application. You can do this using either the command line or the NuGet Package Manager.

Command Line

Open a terminal environment of your choice in the application project directory and enter the command below.

terminal

```
dotnet add package Azure.Identity
dotnet add package Microsoft.Extensions.Azure
```

Azure services are generally accessed using corresponding client classes from the SDK. These classes and your own custom services should be registered in the `Program.cs` file so they can be accessed via dependency injection throughout your app. Inside of `Program.cs`, follow the steps below to correctly setup your service and `DefaultAzureCredential`.

1. Include the `Azure.Identity` and `Microsoft.Extensions.Azure` namespaces with a using statement.
2. Register the Azure service using relevant helper methods.
3. Pass an instance of the `DefaultAzureCredential` object to the `UseCredential` method.

An example of this is shown in the following code segment.

c#

```
using Microsoft.Extensions.Azure;
using Azure.Identity;

// Inside of Program.cs
builder.Services.AddAzureClients(x =>
{
    x.AddBlobServiceClient(new Uri("https://<account-
name>.blob.core.windows.net"));
    x.UseCredential(new DefaultAzureCredential());
});
```

Alternatively, you can also utilize `DefaultAzureCredential` in your services more directly without the help of additional Azure registration methods, as seen below.

c#

```
using Azure.Identity;

// Inside of Program.cs
builder.Services.AddSingleton<BlobServiceClient>(x =>
    new BlobServiceClient(
        new Uri("https://<account-name>.blob.core.windows.net"),
        new DefaultAzureCredential()));
```

When the above code is run on your local workstation during local development, it will look in the environment variables for an application service principal or at Visual Studio, VS Code, the Azure CLI, or Azure PowerShell for a set of developer credentials, either of which can be used to authenticate the app to Azure resources during local development.

When deployed to Azure this same code can also authenticate your app to other Azure resources. `DefaultAzureCredential` can retrieve environment settings and managed identity configurations to authenticate to other services automatically.

Create Microsoft Entra credential types using configuration files

Article • 02/01/2024

The `Microsoft.Extensions.Azure` library supports creating different Azure.Core.TokenCredential types from key-value pairs defined in *appsettings.json* and other configuration files. The credential types correspond to a subset of the credential classes in the Azure Identity client library. This article describes the support for different `TokenCredential` types and how to configure the required key-value pairs for each type.

Support for Azure credentials through configuration

The Microsoft.Extensions.Azure library can automatically provide Azure service clients with a `TokenCredential` class by searching *appsettings.json* or other configuration files for credential values using the `IConfiguration` abstraction for .NET. This approach allows developers to explicitly set credential values across different environments through configuration rather than through app code directly.

The following credential types are supported via configuration:

- ClientCertificateCredential
- ClientSecretCredential
- DefaultAzureCredential
- ManagedIdentityCredential
- WorkloadIdentityCredential

Configure Azure credentials

Azure service clients registered with the AddAzureClients method are automatically configured with an instance of `DefaultAzureCredential` if no explicit credential is supplied via the WithCredential extension method. You can also override the global `DefaultAzureCredential` using credential values from configuration files when registering a client to create a specific credential type:

C#

```
builder.Services.AddAzureClients(clientBuilder =>
{
```

```
    // Register BlobServiceClient using credentials from appsettings.json

clientBuilder.AddBlobServiceClient(builder.Configuration.GetSection("Storage
"));

    // Register ServiceBusClient using the fallback DefaultAzureCredential
credentials
    clientBuilder.AddServiceBusClientWithNamespace(
        "<your_namespace>.servicebus.windows.net");
});
```

The associated *appsettings.json* file:

JSON

```
"Storage": {
    "serviceUri": "<service_uri>",
    "credential": "managedidentity",
    "clientId":  "<clientId>"
}
```

The following credential types also support the `AdditionallyAllowedTenants` property, which specifies additional Microsoft Entra tenants beyond the default tenant for which the credential may acquire tokens:

- ClientCertificateCredential
- ClientSecretCredential
- DefaultAzureCredential

Add the wildcard value "*" to allow the credential to acquire tokens for any Microsoft Entra tenant the logged in account can access. If no tenant IDs are specified, this option will have no effect on that authentication method, and the credential will acquire tokens for any requested tenant when using that method.

JSON

```
{
    "additionallyAllowedTenants":  "<tenant-ids-separated-by-semicolon>"
}
```

Create a `ManagedIdentityCredential` type

You can create both user-assigned and system-assigned managed identities using configuration values. Add the following key-value pairs to your *appsettings.json* file to create an instance of Azure.Identity.ManagedIdentityCredential.

User-assigned managed identities

Specify a user-assigned managed identity via a client ID:

JSON

```
{
    "credential": "managedidentity",
    "clientId":  "<clientId>"
}
```

Alternatively, specify a user-assigned managed identity via a resource ID:

JSON

```
{
    "credential": "managedidentity",
    "managedIdentityResourceId":  "<managedIdentityResourceId>"
}
```

The resource ID takes the form `/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.ManagedIdentity/userAssignedIdentities/{identityName}`.

System-assigned managed identities

JSON

```
{
    "credential": "managedidentity"
}
```

Create a `WorkloadIdentityCredential` type

Add the following key-value pairs to your *appsettings.json* file to create an Azure.Identity.WorkloadIdentityCredential:

JSON

```
{
    "credential": "workloadidentity",
    "tenantId":  "<tenantId>",
    "clientId":  "<clientId>",
```

```
    "tokenFilePath": "<tokenFilePath>"
}
```

Create a `ClientSecretCredential` type

Add the following key-value pairs to your *appsettings.json* file to create an Azure.Identity.ClientSecretCredential:

JSON

```
{
    "tenantId":  "<tenantId>",
    "clientId":  "<clientId>",
    "clientSecret": "<clientSecret>"
}
```

Create a `ClientCertificateCredential` type

Add the following key-value pairs to your *appsettings.json* file to create an Azure.Identity.ClientCertificateCredential:

JSON

```
{
    "tenantId":  "<tenantId>",
    "clientId":  "<clientId>",
    "clientCertificate": "<clientCertificate>",
    "clientCertificateStoreLocation": "<clientCertificateStoreLocation>",
    "additionallyAllowedTenants": "<tenant-ids-separated-by-semicolon>"
}
```

> **Note**
>
> The `clientCertificateStoreLocation` and `additionallyAllowedTenants` key-value pairs are optional. If the keys are present and have empty values, they are ignored. If no `clientCertificateStoreLocation` is specified, the default `CurrentUser` is used from the **X509Credentials.StoreLocation** enum.

Create a `DefaultAzureCredential` type

Add the following key-value pairs to your *appsettings.json* file to create an Azure.Identity.DefaultAzureCredential:

JSON

```
{
    "tenantId":  "<tenantId>",
    "clientId":  "<clientId>",
    "managedIdentityResourceId": "<managedIdentityResourceId>"
}
```

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see our contributor guide.

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

Open a documentation issue

Provide product feedback

Additional methods to authenticate to Azure resources from .NET apps

Article • 08/03/2023

This article lists additional methods apps may use to authenticate to Azure resources. The methods on this page are less commonly used and when possible, it is encouraged to use one of the methods outlined in the authenticating .NET apps to Azure using the Azure SDK overview article.

Interactive browser authentication

This method interactively authenticates an application through InteractiveBrowserCredential by collecting user credentials in the default system.

Interactive browser authentication enables the application for all operations allowed by the interactive login credentials. As a result, if you are the owner or administrator of your subscription, your code has inherent access to most resources in that subscription without having to assign any specific permissions. For this reason, the use of interactive browser authentication is discouraged for anything but experimentation.

Example using InteractiveBrowserCredential

The following example demonstrates using an InteractiveBrowserCredential to authenticate with the BlobServiceClient:

C#

```
using Azure.Identity;
using Azure.Storage.Blobs;

var client = new BlobServiceClient(
        new Uri("https://<storage-account-name>.blob.core.windows.net"),
        new InteractiveBrowserCredential());

foreach (var blobItem in client.GetBlobContainers())
{
    Console.WriteLine(blobItem.Name);
}
```

For more exact control, such as setting redirect URIs, you can supply specific arguments to `InteractiveBrowserCredential` such as `redirect_uri`.

Device code authentication

This method interactively authenticates a user on devices with limited UI (typically devices without a keyboard):

1. When the application attempts to authenticate, the credential prompts the user with a URL and an authentication code.
2. The user visits the URL on a separate browser-enabled device (a computer, smartphone, etc.) and enters the code.
3. The user follows a normal authentication process in the browser.
4. Upon successful authentication, the application is authenticated on the device.

For more information, see Microsoft identity platform and the OAuth 2.0 device authorization grant flow.

Device code authentication in a development environment enables the application for all operations allowed by the interactive login credentials. As a result, if you are the owner or administrator of your subscription, your code has inherent access to most resources in that subscription without having to assign any specific permissions. However, you can use this method with a specific client ID, rather than the default, for which you can assign specific permissions.

Authentication with a username and password

This method authenticates an application using previous-collected credentials and the UsernamePasswordCredential object.

This method of authentication is discouraged because it's less secure than other flows. Also, this method is not interactive and is therefore **not compatible with any form of multi-factor authentication or consent prompting.** The application must already have consent from the user or a directory administrator.

Furthermore, this method authenticates only work and school accounts; Microsoft accounts are not supported. For more information, see Sign up your organization to use Azure Active Directory.

C#

```
using Azure.Identity;
using Azure.Storage.Blobs;

var clientId = Environment.GetEnvironmentVariable("AZURE_CLIENT_ID");
var tenantId = Environment.GetEnvironmentVariable("AZURE_TENANT_ID");
var username = Environment.GetEnvironmentVariable("AZURE_USERNAME");
```

```
var password = Environment.GetEnvironmentVariable("AZURE_PASSWORD");

var client = new BlobServiceClient(
        new Uri("https://<storage-account-name>.blob.core.windows.net"),
        new UsernamePasswordCredential(username, password, tenantId,
clientId));

foreach (var blobItem in client.GetBlobContainers())
{
    Console.WriteLine(blobItem.Name);
}
```

Resource management using the Azure SDK for .NET

Article • 01/26/2024

The Azure SDK for .NET management plane libraries will help you create, provision, and manage Azure resources from within .NET applications. All Azure services have corresponding management libraries.

With the management libraries (namespaces beginning with `Azure.ResourceManager`, for example, `Azure.ResourceManager.Compute`), you can write configuration and deployment programs to perform the same tasks that you can through the Azure portal, Azure CLI, or other resource management tools.

Those packages follow the new Azure SDK guidelines, which provide core capabilities that are shared amongst all Azure SDKs, including:

- The intuitive Azure Identity library.
- An HTTP pipeline with custom policies.
- Error handling.
- Distributed tracing.

> **Note**
>
> You may notice that some packages are still pre-release version, phased releases of additional Azure services' management plane libraries are in process. If you are looking for a stable version package for a particular azure resource and currently only a pre-release version is available, please raise an issue in **Azure SDK for .NET Github repo**

Get started

Prerequisites

- An Azure subscription
- A TokenCredential implementation, such as an Azure Identity library credential type.

Install the package

Install the Azure Identity and Azure resource management NuGet packages for .NET. For example:

PowerShell

```
Install-Package Azure.Identity
Install-Package Azure.ResourceManager
Install-Package Azure.ResourceManager.Resources
Install-Package Azure.ResourceManager.Compute
Install-Package Azure.ResourceManager.Network
```

Authenticate the client

The default option to create an authenticated client is to use `DefaultAzureCredential`. Since all management APIs go through the same endpoint, in order to interact with resources, only one top-level `ArmClient` has to be created.

To authenticate with Azure and create an `ArmClient`, instantiate an `ArmClient` given credentials:

C#

```
using Azure.Identity;
using Azure.ResourceManager;
using System;
using System.Threading.Tasks;

// Code omitted for brevity

ArmClient client = new ArmClient(new DefaultAzureCredential());
```

For more information about the `Azure.Identity.DefaultAzureCredential` class, see DefaultAzureCredential Class.

Management SDK Cheat Sheet

To get started with the Azure management SDK for .NET, imagine you have a task to create/list/update/delete a typical Azure service bus namespace, follow these steps:

1. Authenticate to the subscription and resource group that you want to work on.

C#

```
using Azure.Identity;
using Azure.ResourceManager;
using Azure.ResourceManager.ServiceBus;

ArmClient client = new ArmClient(new DefaultAzureCredential());
SubscriptionResource subscription = client.GetDefaultSubscription();
ResourceGroupResource resourceGroup =
    client.GetDefaultSubscription().GetResourceGroup(resourceGroupName);
```

1. Find the corresponding method to manage your Azure resource.

Expand table

Operation	Method
Get a resource with resource identifier	`client.GetServiceBusQueueResource(ResourceIdentifier resourceIdentifier)`
List	`resourceGroup.GetServiceBusNamespaces()`
Index	`resourceGroup.GetServiceBusNamespace(string servicebusNamespaceName)`
Add/Update	`resourceGroup.GetServiceBusNamespaces().CreateOrUpdate(Azure.WaitUntil waitUntil, string name, ServiceBusNamespaceData data)`
Contains	`resourceGroup.GetServiceBusNamespaces().Exists(string servicebusNamespaceName)`
Delete	`client.GetServiceBusQueueResource(ResourceIdentifior resourceIdentifior).Delete()` or `resourceGroup.GetServiceBusNamespace(string servicebusNamespaceName).Delete()`

Remember, all the Azure resources, including the resource group itself, can be managed by their corresponding management SDK using code similar to the above example. To find the correct Azure management SDK package, look for packages named with the following pattern `Azure.ResourceManager.{ResourceProviderName}`.

To learn more about the ResourceIdentifier, please refer to Structured Resource Identifier.

Key concepts

Understanding Azure Resource Hierarchy

To reduce the number of clients needed to perform common tasks and the number of redundant parameters that each of those clients take, we've introduced an object hierarchy in the SDK that mimics the object hierarchy in Azure. Each resource client in the SDK has methods to access the resource clients of its children that are already scoped to the proper subscription and resource group.

To accomplish this, we're introducing three standard types for all resources in Azure:

{ResourceName}Resource class

This type represents a full resource client object that contains a **Data** property exposing the details as a **{ResourceName}Data** type. It also has access to all of the operations on that resource without needing to pass in scope parameters such as subscription ID or resource name. This makes it convenient to directly execute operations on the result of list calls, since everything is returned as a full resource client now.

C#

```
ArmClient client = new ArmClient(new DefaultAzureCredential());
string resourceGroupName = "myResourceGroup";
SubscriptionResource subscription = await
client.GetDefaultSubscriptionAsync();
ResourceGroupResource resourceGroup = await
subscription.GetResourceGroupAsync(resourceGroupName);
await foreach (VirtualMachineResource virtualMachine in
resourceGroup.GetVirtualMachinesAsync())
{
    //previously we would have to take the resourceGroupName and the vmName
from the vm object
    //and pass those into the powerOff method as well as we would need to
execute that on a separate compute client
    await virtualMachine.PowerOffAsync(WaitUntil.Completed);
}
```

{ResourceName}Data class

This type represents the model that makes up a given resource. Typically, this is the response data from a service call such as HTTP GET and provides details about the underlying resource. Previously, this was represented by a **Model** class.

{ResourceName}Collection class

This type represents the operations you can perform on a collection of resources belonging to a specific parent resource. This object provides most of the logical collection operations.

Expand table

Collection Behavior	Collection Method
Iterate/List	GetAll()
Index	Get(string name)
Add	CreateOrUpdate(Azure.WaitUntil waitUntil, string name, {ResourceName}Data data)
Contains	Exists(string name)

In most cases, parent of a resource is **ResourceGroup**, but in some cases, a resource itself has sub resource, for example a **Subnet** is a child of a **VirtualNetwork**. **ResourceGroup** itself is a child of a **Subscription**

Putting it all together

Imagine that our company requires all virtual machines to be tagged with the owner. We're tasked with writing a program to add the tag to any missing virtual machines in a given resource group.

C#

```
// First we construct our armClient
ArmClient client = new ArmClient(new DefaultAzureCredential());

// Next we get a resource group object
// ResourceGroup is a {ResourceName}Resource object from above
SubscriptionResource subscription = await
client.GetDefaultSubscriptionAsync();
ResourceGroupResource resourceGroup =
   await subscription.GetResourceGroupAsync("myRgName");

// Next we get the collection for the virtual machines
// vmCollection is a {ResourceName}Collection object from above
VirtualMachineCollection virtualMachineCollection = await
resourceGroup.GetVirtualMachines();

// Next we loop over all vms in the collection
// Each vm is a {ResourceName}Resource object from above
await foreach(VirtualMachineResource virtualMachine in
virtualMachineCollection)
```

```
{
   // We access the {ResourceName}Data properties from vm.Data
   if(!virtualMachine.Data.Tags.ContainsKey("owner"))
   {
       // We can also access all operations from vm since it is already
scoped for us
       await virtualMachine.AddTagAsync("owner", GetOwner());
   }
}
```

Structured Resource Identifier

Resource IDs contain useful information about the resource itself, but they're plain strings that have to be parsed. Instead of implementing your own parsing logic, you can use a `ResourceIdentifier` object that will do the parsing for you.

Example: Parsing an ID using a ResourceIdentifier object

C#

```
string resourceId = "/subscriptions/aaaaaaaa-bbbb-cccc-dddd-
eeeeeeeeeeee/resourceGroups/workshop2021-
rg/providers/Microsoft.Network/virtualNetworks/myVnet/subnets/mySubnet";
ResourceIdentifier id = new ResourceIdentifier(resourceId);
Console.WriteLine($"Subscription: {id.SubscriptionId}");
Console.WriteLine($"ResourceGroup: {id.ResourceGroupName}");
Console.WriteLine($"Vnet: {id.Parent.Name}");
Console.WriteLine($"Subnet: {id.Name}");
```

However, keep in mind that some of those properties could be null. You can usually tell by the ID string itself which type a resource ID is. But if you're unsure, check if the properties are null.

Example: Resource Identifier Generator

You may not want to manually create the `resourceId` from a pure `string`. Each `{ResourceName}Resource` class has a static method that can help you create the resource identifier string.

C#

```
ResourceIdentifier resourceId =
    AvailabilitySetResource.CreateResourceIdentifier(
        "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
```

```
        "resourceGroupName",
        "resourceName");
```

Manage existing resources

Performing operations on resources that already exist is a common use case when using the management client libraries. In this scenario, you usually have the identifier of the resource you want to work on as a string. Although the new object hierarchy is great for provisioning and working within the scope of a given parent, it is not the most efficient when it comes to this specific scenario.

Here's an example how you can access an `AvailabilitySetResource` object and manage it directly with its resource identifier:

C#

```
using Azure.Identity;
using Azure.ResourceManager;
using Azure.ResourceManager.Resources;
using Azure.ResourceManager.Compute;
using System;
using System.Threading.Tasks;

// Code omitted for brevity

ResourceIdentifier subscriptionId =
    SubscriptionResource.CreateResourceIdentifier("aaaaaaaa-bbbb-cccc-dddd-
eeeeeeeeeeee");

ResourceIdentifier resourceId =
    AvailabilitySetResource.CreateResourceIdentifier(
        subscriptionId.SubscriptionId,
        "resourceGroupName",
        "resourceName");

// We construct a new armClient to work with
ArmClient client = new ArmClient(new DefaultAzureCredential());
// Next we get the specific subscription this resource belongs to
SubscriptionResource subscription =
client.GetSubscriptionResource(subscriptionId);
// Next we get the specific resource group this resource belongs to
ResourceGroupResource resourceGroup = await
subscription.GetResourceGroupAsync(resourceId.ResourceGroupName);
// Finally we get the resource itself
// Note: for this last step in this example, Azure.ResourceManager.Compute
is needed
AvailabilitySetResource availabilitySet = await
resourceGroup.GetAvailabilitySetAsync(resourceId.Name);
```

This approach required a lot of code and three API calls are made to Azure. The same can be done with less code and without any API calls by using extension methods that we've provided on the client itself. These extension methods allow you to pass in a resource identifier and retrieve a scoped resource client. The object returned is a {ResourceName}Resource. Since it hasn't reached out to Azure to retrieve the data yet, calling the `Data` property will throw exception, you can either use `HasData` property to tell if the resource instance contains a data or call the `Get` or `GetAsync` method on the resource to retrieve the resource data.

So, the previous example would end up looking like this:

C#

```
ResourceIdentifier resourceId =
    AvailabilitySetResource.CreateResourceIdentifier(
        "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
        "resourceGroupName",
        "resourceName");
// We construct a new armClient to work with
ArmClient client = new ArmClient(new DefaultAzureCredential());
// Next we get the AvailabilitySet resource client from the armClient
// The method takes in a ResourceIdentifier but we can use the implicit cast
from string
AvailabilitySetResource availabilitySet =
client.GetAvailabilitySetResource(resourceId);
// At this point availabilitySet.Data will be null and trying to access it
will throw exception
// If we want to retrieve the objects data we can simply call get
availabilitySet = await availabilitySet.GetAsync();
// we now have the data representing the availabilitySet
Console.WriteLine(availabilitySet.Data.Name);
```

Check if a Resource exists

If you aren't sure if a resource you want to get exists, or you just want to check if it exists, you can use `Exists()` or `ExistsAsync()` methods, which can be invoked from any `{ResourceName}Collection` class.

`Exists()` returns a `Response<bool>` while `ExistsAsync()` as its async version returns a `Task<Response<bool>>`. In the `Response<bool>` object, you can visit its `Value` property to check if a resource exists. The `Value` is `false` if the resource does not exist and vice versa.

In previous versions of packages, you would have to catch the `RequestFailedException` and inspect the status code for 404. With this new API, we hope that this can boost the

developer productivity and optimize resource access.

C#

```
ArmClient client = new ArmClient(new DefaultAzureCredential());
SubscriptionResource subscription = await
client.GetDefaultSubscriptionAsync();
string resourceGroupName = "myRgName";

try
{
    ResourceGroupResource resourceGroup = await
subscription.GetResourceGroupAsync(resourceGroupName);
    // At this point, we are sure that myRG is a not null Resource Group, so
we can use this object to perform any operations we want.
}
catch (RequestFailedException ex) when (ex.Status == 404)
{
    Console.WriteLine($"Resource Group {resourceGroupName} does not
exist.");
}
```

Now with these convenience methods, we can simply do the following.

C#

```
ArmClient client = new ArmClient(new DefaultAzureCredential());
SubscriptionResource subscription = await
client.GetDefaultSubscriptionAsync();
string resourceGroupName = "myRgName";

bool exists = await
subscription.GetResourceGroups().ExistsAsync(resourceGroupName).Value;

if (exists)
{
    Console.WriteLine($"Resource Group {resourceGroupName} exists.");

    // We can get the resource group now that we know it exists.
    // This does introduce a small race condition where resource group could
have been deleted between the check and the get.
    ResourceGroupResource resourceGroup = await
subscription.GetResourceGroupAsync(resourceGroupName);
}
else
{
    Console.WriteLine($"Resource Group {rgName} does not exist.");
}
```

Examples

Create a resource group

C#

```
// First, initialize the ArmClient and get the default subscription
ArmClient client = new ArmClient(new DefaultAzureCredential());
// Now we get a ResourceGroup collection for that subscription
SubscriptionResource subscription = await
client.GetDefaultSubscriptionAsync();
ResourceGroupCollection resourceGroupCollection =
subscription.GetResourceGroups();

// With the collection, we can create a new resource group with an specific
name
string resourceGroupName = "myRgName";
AzureLocation location = AzureLocation.WestUS2;
ResourceGroupData resourceGroupData = new ResourceGroupData(location);
ResourceGroupResource resourceGroup = (await
resourceGroupCollection.CreateOrUpdateAsync(resourceGroupName,
resourceGroupData)).Value;
```

List all resource groups

C#

```
// First, initialize the ArmClient and get the default subscription
ArmClient client = new ArmClient(new DefaultAzureCredential());
SubscriptionResource subscription = await
client.GetDefaultSubscriptionAsync();
// Now we get a ResourceGroup collection for that subscription
ResourceGroupCollection resourceGroupCollection =
subscription.GetResourceGroups();
// With GetAllAsync(), we can get a list of the resources in the collection
await foreach (ResourceGroupResource resourceGroup in
resourceGroupCollection)
{
    Console.WriteLine(resourceGroup.Data.Name);
}
```

Update a resource group

C#

```
// Note: Resource group named 'myRgName' should exist for this example to
work.
ArmClient client = new ArmClient(new DefaultAzureCredential());
SubscriptionResource subscription = await
client.GetDefaultSubscriptionAsync();
```

```
string resourceGroupName = "myRgName";
ResourceGroupResource resourceGroup = await
subscription.GetResourceGroupAsync(resourceGroupName);
resourceGroup = await resourceGroup.AddTagAsync("key", "value");
```

Delete a resource group

C#

```
ArmClient client = new ArmClient(new DefaultAzureCredential());
SubscriptionResource subscription = await
client.GetDefaultSubscriptionAsync();
string resourceGroupName = "myRgName";
ResourceGroupResource resourceGroup = await
subscription.GetResourceGroupAsync(resourceGroupName);
await resourceGroup.DeleteAsync();
```

For more detailed examples, take a look at samples we have available.

Troubleshooting

- If you have a bug to report or have a suggestion, file an issue via GitHub issues and make sure you add the "Preview" label to the issue.
- If you need help, check previous questions, or ask new ones on StackOverflow using Azure and .NET tags.
- If having trouble with authentication, see the DefaultAzureCredential documentation.

Next steps

More sample code

- Managing Resource Groups
- Creating a Virtual Network
- .NET Management Library Code Samples

Additional Documentation

If you're migrating from the old SDK to this preview, check out this Migration guide.

For more information on Azure SDK, see Azure SDK Releases.

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see our contributor guide.

.NET feedback

.NET is an open source project. Select a link to provide feedback:

Open a documentation issue

Provide product feedback

Dependency injection with the Azure SDK for .NET

Article • 08/14/2023

This article demonstrates how to register Azure service clients from the latest Azure client libraries for .NET for dependency injection in a .NET app. Every modern .NET app starts up by using the instructions provided in a *Program.cs* file.

Install packages

To register and configure service clients from an Azure.-prefixed package:

1. Install the Microsoft.Extensions.Azure package in your project:

   .NET CLI

   ```
   dotnet add package Microsoft.Extensions.Azure
   ```

2. Install the Azure.Identity package to configure a `TokenCredential` type to use for authenticating all registered clients that accept such a type:

   .NET CLI

   ```
   dotnet add package Azure.Identity
   ```

For demonstration purposes, the sample code in this article uses the Key Vault Secrets, Blob Storage, and Service Bus libraries. Install the following packages to follow along:

.NET CLI

```
dotnet add package Azure.Security.KeyVault.Secrets
dotnet add package Azure.Storage.Blobs
dotnet add package Azure.Messaging.ServiceBus
```

Register clients and subclients

A service client is the entry point to the API for an Azure service – from it, library users can invoke all operations the service provides and can easily implement the most common scenarios. Where it will simplify an API's design, groups of service calls can be organized around smaller subclient types. For example, `ServiceBusClient` can register

additional `ServiceBusSender` subclients for publishing messages or `ServiceBusReceiver` subclients for consuming messages.

In the *Program.cs* file, invoke the AddAzureClients extension method to register a client for each service. The following code samples provide guidance on application builders from the `Microsoft.AspNetCore.Builder` and `Microsoft.Extensions.Hosting` namespaces.

WebApplicationBuilder

C#

```
using Azure.Identity;
using Azure.Messaging.ServiceBus;
using Azure.Messaging.ServiceBus.Administration;
using Microsoft.Extensions.Azure;

WebApplicationBuilder builder = WebApplication.CreateBuilder(args);

List<string> queueNames = await GetQueueNames();

builder.Services.AddAzureClients(clientBuilder =>
{
    // Register clients for each service
    clientBuilder.AddSecretClient(new Uri("<key_vault_url>"));
    clientBuilder.AddBlobServiceClient(new Uri("<storage_url>"));
    clientBuilder.AddServiceBusClientWithNamespace(
        "<your_namespace>.servicebus.windows.net");
    clientBuilder.UseCredential(new DefaultAzureCredential());

    // Register a subclient for each Service Bus Queue
    foreach (string queue in queueNames)
    {
        clientBuilder.AddClient<ServiceBusSender,
ServiceBusClientOptions>(
            (_, _, provider) => provider.GetService<ServiceBusClient>()
                    .CreateSender(queue)).WithName(queue);
    }
});

WebApplication app = builder.Build();

async Task<List<string>> GetQueueNames()
{
    // Query the available queues for the Service Bus namespace.
    var adminClient = new ServiceBusAdministrationClient
        ("<your_namespace>.servicebus.windows.net", new
DefaultAzureCredential());
    var queueNames = new List<string>();

    // Because the result is async, the queue names need to be captured
```

```
    // to a standard list to avoid async calls when registering. Failure
to
    // do so results in an error with the services collection.
    await foreach (QueueProperties queue in
adminClient.GetQueuesAsync())
    {
        queueNames.Add(queue.Name);
    }

    return queueNames;
}
```

In the preceding code:

- Key Vault Secrets, Blob Storage, and Service Bus clients are registered using the AddSecretClient, AddBlobServiceClient and AddServiceBusClientWithNamespace, respectively. The `Uri`- and `string`-typed arguments are passed. To avoid specifying these URLs explicitly, see the Store configuration separately from code section.
- DefaultAzureCredential is used to satisfy the `TokenCredential` argument requirement for each registered client. When one of the clients is created, `DefaultAzureCredential` is used to authenticate.
- Service Bus subclients are registered for each queue on the service using the subclient and corresponding options types. The queue names for the subclients are retrieved using a separate method outside of the service registration because the `GetQueuesAsync` method must be run asynchronously.

Use the registered clients

With the clients registered, as described in the Register clients and subclients section, you can now use them. In the following example, constructor injection is used to obtain the Blob Storage client in an ASP.NET Core API controller:

C#

```
[ApiController]
[Route("[controller]")]
public class MyApiController : ControllerBase
{
    private readonly BlobServiceClient _blobServiceClient;

    public MyApiController(BlobServiceClient blobServiceClient)
    {
        _blobServiceClient = blobServiceClient;
    }
```

```
    [HttpGet]
    public async Task<IEnumerable<string>> Get()
    {
        BlobContainerClient containerClient =
            _blobServiceClient.GetBlobContainerClient("demo");
        var results = new List<string>();

        await foreach (BlobItem blob in containerClient.GetBlobsAsync())
        {
            results.Add(blob.Name);
        }

        return results.ToArray();
    }
}
```

Store configuration separately from code

In the Register clients and subclients section, you explicitly passed the `Uri`-typed variables to the client constructors. This approach could cause problems when you run code against different environments during development and production. The .NET team suggests storing such configurations in environment-dependent JSON files. For example, you can have an *appsettings.Development.json* file containing development environment settings. Another *appsettings.Production.json* file would contain production environment settings, and so on. The file format is:

JSON

```
{
  "AzureDefaults": {
    "Diagnostics": {
      "IsTelemetryDisabled": false,
      "IsLoggingContentEnabled": true
    },
    "Retry": {
      "MaxRetries": 3,
      "Mode": "Exponential"
    }
  },
  "KeyVault": {
    "VaultUri": "https://mykeyvault.vault.azure.net"
  },
  "ServiceBus": {
    "Namespace": "<your_namespace>.servicebus.windows.net"
  },
  "Storage": {
    "ServiceUri": "https://mydemoaccount.storage.windows.net"
```

```
  }
}
```

You can add any properties from the ClientOptions class into the JSON file. The settings in the JSON configuration file can be retrieved using IConfiguration.

WebApplicationBuilder

C#

```
builder.Services.AddAzureClients(clientBuilder =>
{
    clientBuilder.AddSecretClient(
        builder.Configuration.GetSection("KeyVault"));

    clientBuilder.AddBlobServiceClient(
        builder.Configuration.GetSection("Storage"));

    clientBuilder.AddServiceBusClientWithNamespace(
        builder.Configuration["ServiceBus:Namespace"]);

    clientBuilder.UseCredential(new DefaultAzureCredential());

    // Set up any default settings
    clientBuilder.ConfigureDefaults(
        builder.Configuration.GetSection("AzureDefaults"));
});
```

In the preceding JSON sample:

- The top-level key names, `AzureDefaults`, `KeyVault`, `ServiceBus`, and `Storage`, are arbitrary. All other key names hold significance, and JSON serialization is performed in a case-insensitive manner.
- The `AzureDefaults.Retry` object literal:
  - Represents the retry policy configuration settings.
  - Corresponds to the Retry property. Within that object literal, you find the `MaxRetries` key, which corresponds to the MaxRetries property.
- The `KeyVault:VaultUri`, `ServiceBus:Namespace`, and `Storage:ServiceUri` key values map to the `Uri`- and `string`-typed arguments of the Azure.Security.KeyVault.Secrets.SecretClient.SecretClient(Uri, TokenCredential, SecretClientOptions), Azure.Messaging.ServiceBus.ServiceBusClient.ServiceBusClient(String), and Azure.Storage.Blobs.BlobServiceClient.BlobServiceClient(Uri, TokenCredential, BlobClientOptions) constructor overloads, respectively. The `TokenCredential`

variants of the constructors are used because a default `TokenCredential` is set via the Microsoft.Extensions.Azure.AzureClientFactoryBuilder.UseCredential(TokenCredential) method call.

Configure multiple service clients with different names

Imagine you have two storage accounts: one for private information and another for public information. Your app transfers data from the public to the private storage account after some operation. You need to have two storage service clients. To differentiate those two clients, use the WithName extension method:

C#

```
builder.Services.AddAzureClients(clientBuilder =>
{
    clientBuilder.AddBlobServiceClient(
        builder.Configuration.GetSection("PublicStorage"));

    clientBuilder.AddBlobServiceClient(
            builder.Configuration.GetSection("PrivateStorage"))
        .WithName("PrivateStorage");
});
```

Using an ASP.NET Core controller as an example, access the named service client using the IAzureClientFactory<TClient> interface:

C#

```
public class HomeController : Controller
{
    private readonly BlobServiceClient _publicStorage;
    private readonly BlobServiceClient _privateStorage;

    public HomeController(
        BlobServiceClient defaultClient,
        IAzureClientFactory<BlobServiceClient> clientFactory)
    {
        _publicStorage = defaultClient;
        _privateStorage = clientFactory.CreateClient("PrivateStorage");
    }
}
```

The unnamed service client is still available in the same way as before. Named clients are additive.

Configure a new retry policy

At some point, you may want to change the default settings for a service client. For example, you may want different retry settings or to use a different service API version. You can set the retry settings globally or on a per-service basis. Assume you have the following *appsettings.json* file in your ASP.NET Core project:

JSON

```
{
  "AzureDefaults": {
    "Retry": {
      "maxRetries": 3
    }
  },
  "KeyVault": {
    "VaultUri": "https://mykeyvault.vault.azure.net"
  },
  "ServiceBus": {
    "Namespace": "<your_namespace>.servicebus.windows.net"
  },
  "Storage": {
    "ServiceUri": "https://store1.storage.windows.net"
  },
  "CustomStorage": {
    "ServiceUri": "https://store2.storage.windows.net"
  }
}
```

You can change the retry policy to suit your needs like so:

C#

```
builder.Services.AddAzureClients(clientBuilder =>
{
    // Establish the global defaults
    clientBuilder.ConfigureDefaults(
        builder.Configuration.GetSection("AzureDefaults"));
    clientBuilder.UseCredential(new DefaultAzureCredential());

    // A Key Vault Secrets client using the global defaults
    clientBuilder.AddSecretClient(
        builder.Configuration.GetSection("KeyVault"));

    // A Blob Storage client with a custom retry policy
    clientBuilder.AddBlobServiceClient(
            builder.Configuration.GetSection("Storage"))
        .ConfigureOptions(options => options.Retry.MaxRetries = 10);

    clientBuilder.AddServiceBusClientWithNamespace(
            builder.Configuration["ServiceBus:Namespace"])
```

```
        .ConfigureOptions(options => options.RetryOptions.MaxRetries = 10);

    // A named storage client with a different custom retry policy
    clientBuilder.AddBlobServiceClient(
            builder.Configuration.GetSection("CustomStorage"))
        .WithName("CustomStorage")
        .ConfigureOptions(options =>
        {
            options.Retry.Mode = Azure.Core.RetryMode.Exponential;
            options.Retry.MaxRetries = 5;
            options.Retry.MaxDelay = TimeSpan.FromSeconds(120);
        });
});
```

You can also place retry policy overrides in the *appsettings.json* file:

JSON

```
{
  "KeyVault": {
    "VaultUri": "https://mykeyvault.vault.azure.net",
    "Retry": {
      "maxRetries": 10
    }
  }
}
```

See also

- Dependency injection in ASP.NET Core
- Configuration in .NET
- Configuration in ASP.NET Core

Thread safety and client lifetime management for Azure SDK objects

Article • 06/24/2023

This article helps you understand thread safety issues when using the Azure SDK. It also discusses how the design of the SDK impacts client lifetime management. You'll learn why it's unnecessary to dispose of Azure SDK client objects.

Thread safety

All Azure SDK client objects are thread-safe and independent of each other. This design ensures that reusing client instances is always safe, even across threads. For example, the following code launches multiple tasks but is thread safe:

C#

```
var client = new SecretClient(
    new Uri("<secrets_endpoint>"), new DefaultAzureCredential());

foreach (var secretName in secretNames)
{
    // Using clients from parallel threads
    Task.Run(() => Console.WriteLine(client.GetSecret(secretName).Value));
}
```

Model objects used by SDK clients, whether input or output models, aren't thread-safe by default. Most use cases involving model objects only use a single thread. Therefore, the cost of implementing synchronization as a default behavior is too high for these objects. The following code illustrates a bug in which accessing a model from multiple threads could cause an undefined behavior:

C#

```
KeyVaultSecret newSecret = client.SetSecret("secret", "value");

foreach (var tag in tags)
{
    // Don't use model type from parallel threads
    Task.Run(() => newSecret.Properties.Tags[tag] = CalculateTagValue(tag));
}

client.UpdateSecretProperties(newSecret.Properties);
```

To access the model from different threads, you must implement your own synchronization code. For example:

C#

```
KeyVaultSecret newSecret = client.SetSecret("secret", "value");

// Code omitted for brevity

foreach (var tag in tags)
{
    Task.Run(() =>
    {
        lock (newSecret)
        {
            newSecret.Properties.Tags[tag] = CalculateTagValue(tag);
        }
    );
}

client.UpdateSecretProperties(newSecret.Properties);
```

Client lifetime

Because Azure SDK clients are thread-safe, there's no reason to construct multiple SDK client objects for a given set of constructor parameters. Treat Azure SDK client objects as singletons once constructed. This recommendation is commonly implemented by registering Azure SDK client objects as singletons in the app's Inversion of Control (IoC) container. Dependency injection (DI) is used to obtain references to the SDK client object. The following example shows a singleton client object registration:

C#

```
var builder = Host.CreateApplicationBuilder(args);

var endpoint = builder.Configuration["SecretsEndpoint"];
var blobServiceClient = new BlobServiceClient(
    new Uri(endpoint), new DefaultAzureCredential());

builder.Services.AddSingleton(blobServiceClient);
```

For more information about implementing DI with the Azure SDK, see Dependency injection with the Azure SDK for .NET.

Alternatively, you may create an SDK client instance and provide it to methods that require a client. The point is to avoid unnecessary instantiations of the same SDK client

object with the same parameters. It's both unnecessary and wasteful.

Clients aren't disposable

Two final questions that often come up are:

- Do I need to dispose of Azure SDK client objects when I'm finished using them?
- Why aren't HTTP-based Azure SDK client objects disposable?

Internally, all Azure SDK clients use a single shared `HttpClient` instance. The clients don't create any other resources that need to be actively freed. The shared `HttpClient` instance persists throughout the entire application lifetime.

C#

```
// Both clients reuse the shared HttpClient and don't need to be disposed
var blobClient = new BlobClient(new Uri(sasUri));
var blobClient2 = new BlobClient(new Uri(sasUri2));
```

It's possible to provide a custom instance of `HttpClient` to an Azure SDK client object. In this case, you become responsible for managing the `HttpClient` lifetime and properly disposing of it at the right time.

C#

```
var httpClient = new HttpClient();

var clientOptions = new BlobClientOptions()
{
    Transport = new HttpClientTransport(httpClient)
};

// Both clients would use the HttpClient instance provided in clientOptions
var blobClient = new BlobClient(new Uri(sasUri), clientOptions);
var blobClient2 = new BlobClient(new Uri(sasUri2), clientOptions);

// Code omitted for brevity

// You're responsible for properly disposing httpClient some time later
httpClient.Dispose();
```

Further guidance for properly managing and disposing of `HttpClient` instances can be found in the HttpClient documentation.

See also

- Dependency injection with the Azure SDK for .NET
- Dependency injection in .NET

Logging with the Azure SDK for .NET

Article • 04/09/2024

The Azure SDK for .NET's client libraries include the ability to log client library operations. This logging allows you to monitor I/O requests and responses that client libraries are making to Azure services. Typically, the logs are used to debug or diagnose communication issues. This article describes the following approaches to enable logging with the Azure SDK for .NET:

- Enable logging with built-in methods
- Configure custom logging
- Map to ASP.NET Core logging

> **Important**
>
> This article applies to client libraries that use the most recent versions of the Azure SDK for .NET. To see if a library is supported, see the list of Azure SDK latest releases. If your app is using an older version of an Azure SDK client library, refer to specific instructions in the applicable service documentation.

Log information

The SDK logs each HTTP request and response, sanitizing parameter query and header values to remove personal data.

HTTP request log entry:

- Unique ID
- HTTP method
- URI
- Outgoing request headers

HTTP response log entry:

- Duration of I/O operation (time elapsed)
- Request ID
- HTTP status code
- HTTP reason phrase
- Response headers
- Error information, when applicable

HTTP request and response content:

- Content stream as text or bytes depending on the `Content-Type` header.

> **Note**
>
> Content logging is disabled by default. To enable it, see Log HTTP request and response bodies. This capability applies only to libraries using HTTP to communicate with an Azure service. Libraries based on alternative protocols, such as AMQP, don't support content logging. Unsupported examples include libraries for Azure services such as Event Hubs, Service Bus, and Web PubSub.

Event logs are output usually at one of these three levels:

- Informational for request and response events
- Warning for errors
- Verbose for detailed messages and content logging

Enable logging with built-in methods

The Azure SDK for .NET's client libraries log events to Event Tracing for Windows (ETW) via the System.Diagnostics.Tracing.EventSource class, which is typical for .NET. Event sources allow you to use structured logging in your app with minimal performance overhead. To gain access to the event logs, you need to register event listeners.

The SDK includes the Azure.Core.Diagnostics.AzureEventSourceListener class, which contains two static methods that simplify comprehensive logging for your .NET app: `CreateConsoleLogger` and `CreateTraceLogger`. Each of these methods accepts an optional parameter that specifies a log level. If the parameter isn't provided, the default log level of `Informational` is used.

Log to the console window

A core tenet of the Azure SDK for .NET client libraries is to simplify the ability to view comprehensive logs in real time. The `CreateConsoleLogger` method allows you to send logs to the console window with a single line of code:

C#

```
using AzureEventSourceListener listener =
    AzureEventSourceListener.CreateConsoleLogger();
```

Log to diagnostic traces

If you implement trace listeners, you can use the `CreateTraceLogger` method to log to the standard .NET event tracing mechanism (System.Diagnostics.Tracing). For more information on event tracing in .NET, see Trace Listeners.

This example specifies a log level of verbose:

C#

```
using AzureEventSourceListener listener =
    AzureEventSourceListener.CreateTraceLogger(EventLevel.Verbose);
```

Configure custom logging

As mentioned above, you need to register event listeners to receive log messages from the Azure SDK for .NET. If you don't want to implement comprehensive logging using one of the simplified methods above, you can construct an instance of the `AzureEventSourceListener` class. Pass that instance a callback method that you write. This method will receive log messages that you can process however you need to. In addition, when you construct the instance, you can specify the log levels to include.

The following example creates an event listener that logs to the console with a custom message. The logs are filtered to those events emitted from the Azure Core client library with a level of verbose. The Azure Core library uses an event source name of `Azure-Core`.

C#

```
using Azure.Core.Diagnostics;
using System.Diagnostics.Tracing;

// code omitted for brevity

using var listener = new AzureEventSourceListener((e, message) =>
    {
        // Only log messages from "Azure-Core" event source
        if (e.EventSource.Name == "Azure-Core")
        {
            Console.WriteLine($"{DateTime.Now} {message}");
        }
    },
    level: EventLevel.Verbose);
```

Map to ASP.NET Core logging

The AzureEventSourceLogForwarder service enables you to use the standard ASP.NET Core logging configuration for logging. The service forwards log messages from Azure SDK event sources to ILoggerFactory.

The following table depicts how the Azure SDK for .NET `EventLevel` maps to the ASP.NET Core `LogLevel`.

Expand table

Azure SDK `EventLevel`	**ASP.NET Core `LogLevel`**
`Critical`	`Critical`
`Error`	`Error`
`Informational`	`Information`
`Warning`	`Warning`
`Verbose`	`Debug`
`LogAlways`	`Information`

Logging with client registration

Using the Azure Service Bus library as an example, complete the following steps:

1. Install the Microsoft.Extensions.Azure NuGet package:

   .NET CLI

   ```
   dotnet add package Microsoft.Extensions.Azure
   ```

2. In *Program.cs*, register the Azure SDK library's client via a call to the AddAzureClients extension method:

   C#

   ```
   using Azure.Identity;
   using Microsoft.Extensions.Azure;

   // code omitted for brevity

   builder.Services.AddAzureClients(azureBuilder =>
   ```

```
{
    azureBuilder.AddServiceBusClient(
        builder.Configuration.GetConnectionString("ServiceBus"));
    azureBuilder.UseCredential(new DefaultAzureCredential());
});
```

In the preceding sample, the `AddAzureClients` method:

- Registers the following objects with the dependency injection (DI) container:
  - Log forwarder service
  - Azure Service Bus client
- Sets the default token credential to be used for all registered clients.

3. In *appsettings.json*, change the Service Bus library's default log level. For example, toggle it to `Debug` by setting the `Logging:LogLevel:Azure.Messaging.ServiceBus` key as follows:

JSON

```
{
    "ConnectionStrings": {
        "ServiceBus": "<connection_string>"
    },
    "Logging": {
        "LogLevel": {
            "Default": "Information",
            "Microsoft.AspNetCore": "Warning",
            "Azure.Messaging.ServiceBus": "Debug"
        }
    },
    "AllowedHosts": "*"
}
```

Since the `Logging:LogLevel:Azure.Messaging.ServiceBus` key is set to `Debug`, Service Bus client events up to `EventLevel.Verbose` will be logged.

Logging without client registration

There are scenarios in which registering an Azure SDK library's client with the DI container is either impossible or unnecessary:

- The Azure SDK library doesn't include an `IServiceCollection` extension method to register a client in the DI container.
- Your app uses Azure extension libraries that depend on other Azure SDK libraries. Examples of such Azure extension libraries include:
  - Azure Key Vault key encryptor for DataProtection

- Azure Key Vault secrets configuration provider
- Azure Blob Storage key store for DataProtection

In these scenarios, complete the following steps:

1. Install the Microsoft.Extensions.Azure NuGet package:

.NET CLI

```
dotnet add package Microsoft.Extensions.Azure
```

2. In *Program.cs*, register the log forwarder service as a singleton in the DI container:

C#

```
using Azure.Identity;
using Microsoft.AspNetCore.DataProtection;
using Microsoft.Extensions.Azure;
using Microsoft.Extensions.DependencyInjection.Extensions;

var builder = WebApplication.CreateBuilder(args);
builder.Services.AddRazorPages();
builder.Services.TryAddSingleton<AzureEventSourceLogForwarder>();

builder.Services.AddDataProtection()
    .PersistKeysToAzureBlobStorage("<connection_string>", "
<container_name>", "keys.xml")
    .ProtectKeysWithAzureKeyVault(new Uri("<uri>"), new
DefaultAzureCredential());
```

3. Fetch the log forwarder service from the DI container and invoke its Start method. For example, using constructor injection in an ASP.NET Core Razor Pages page model class:

C#

```
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.Extensions.Azure;

public class IndexModel : PageModel
{
    public IndexModel(AzureEventSourceLogForwarder logForwarder) =>
        logForwarder.Start();
```

4. In *appsettings.json*, change the Azure Core library's default log level. For example, toggle it to `Debug` by setting the `Logging:LogLevel:Azure.Core` key as follows:

JSON

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning",
      "Azure.Core": "Debug"
    }
  },
  "AllowedHosts": "*"
}
```

Since the `Logging:LogLevel:Azure.Core` key is set to `Debug`, Azure Core library events up to `EventLevel.Verbose` will be logged.

For more information, see Logging in .NET Core and ASP.NET Core.

Logging using Azure.Monitor.OpenTelemetry.AspNetCore

The Azure Monitor OpenTelemetry distro, starting with version `1.2.0`, supports capturing logs coming from Azure client libraries. You can control logging using any of the configuration options discussed in Logging in .NET Core and ASP.NET Core.

Using the Azure Service Bus library as an example, complete the following steps:

1. Install the Azure.Monitor.OpenTelemetry.AspNetCore NuGet package:

   .NET CLI

   ```
   dotnet add package Azure.Monitor.OpenTelemetry.AspNetCore
   ```

2. Create or register the library's client. The distro supports both cases.

   C#

   ```
   await using var client = new ServiceBusClient("<connection_string>");
   ```

3. In *appsettings.json*, change the Service Bus library's default log level. For example, toggle it to `Debug` by setting the `Logging:LogLevel:Azure.Messaging.ServiceBus` key as follows:

   JSON

```
{
    "ConnectionStrings": {
        "ServiceBus": "<connection_string>"
    },
    "Logging": {
        "LogLevel": {
            "Default": "Information",
            "Microsoft.AspNetCore": "Warning",
            "Azure.Messaging.ServiceBus": "Debug"
        }
    },
    "AllowedHosts": "*"
}
```

Since the `Logging:LogLevel:Azure.Messaging.ServiceBus` key is set to `Debug`, Service Bus client events up to `EventLevel.Verbose` will be logged.

Log HTTP request and response bodies

> **Note**
>
> This capability applies only to libraries using HTTP to communicate with an Azure service. Libraries based on alternative protocols, such as AMQP, don't support content logging. Unsupported examples include libraries for Azure services such as Event Hubs, Service Bus, and Web PubSub.

When troubleshooting unexpected behavior with a client library, it's helpful to inspect the following items:

- The HTTP request body sent to the underlying Azure service's REST API.
- The HTTP response body received from the Azure service's REST API.

By default, logging of the aforementioned content is disabled. To enable logging of the HTTP request and response bodies, complete the following steps:

1. Set the client options object's IsLoggingContentEnabled property to `true`, and pass the options object to the client's constructor. For example, to log HTTP requests and responses for the Azure Key Vault Secrets library:

```
C#

var clientOptions = new SecretClientOptions
{
    Diagnostics =
```

```
    {
        IsLoggingContentEnabled = true,
    }
};
var client = new SecretClient(
    new Uri("https://<keyvaultname>.vault.azure.net/"),
    new DefaultAzureCredential(),
    clientOptions);
```

2. Use your preferred logging approach with an event/log level of verbose/debug or higher. Find your approach in the following table for specific instructions.

Expand table

Approach	Instructions
Enable logging with built-in methods	Pass `EventLevel.Verbose` or `EventLevel.LogAlways` to `AzureEventSourceListener.CreateConsoleLogger` or `AzureEventSourceListener.CreateTraceLogger`
Configure custom logging	Set the `AzureEventSourceListener` class' `level` constructor parameter to `EventLevel.Verbose` or `EventLevel.LogAlways`
Map to ASP.NET Core logging	Add `"Azure.Core": "Debug"` to *appsettings.json*

Next steps

- Enable diagnostics logging for apps in Azure App Service
- Review Azure security logging and auditing options
- Learn how to work with Azure platform logs
- Read more about .NET logging and tracing

Pagination with the Azure SDK for .NET

Article • 06/23/2023

In this article, you'll learn how to use the Azure SDK for .NET pagination functionality to work efficiently and productively with large data sets. Pagination is the act of dividing large data sets into pages, making it easier for the consumer to iterate through smaller amounts of data. Starting with C# 8, you can create and consume streams asynchronously using Asynchronous (async) streams. Async streams are based on the IAsyncEnumerable<T> interface. The Azure SDK for .NET exposes an implementation of `IAsyncEnumerable<T>` with its `AsyncPageable<T>` class.

All of the samples in this article rely on the following NuGet packages:

- Azure.Security.KeyVault.Secrets
- Microsoft.Extensions.Azure
- Microsoft.Extensions.Hosting
- System.Linq.Async

For the latest directory of Azure SDK for .NET packages, see Azure SDK latest releases.

Pageable return types

Clients instantiated from the Azure SDK for .NET can return the following pageable types.

Type	Description
Pageable<T>	A collection of values retrieved in pages
AsyncPageable<T>	A collection of values asynchronously retrieved in pages

Most of the samples in this article are asynchronous, using variations of the `AsyncPageable<T>` type. Using asynchronous programming for I/O-bound operations is ideal. A perfect use case is using the async APIs from the Azure SDK for .NET as these operations represent HTTP/S network calls.

Iterate over `AsyncPageable` with `await foreach`

To iterate over an `AsyncPageable<T>` using the await foreach syntax, consider the following example:

C#

```
async Task IterateSecretsWithAwaitForeachAsync()
{
    AsyncPageable<SecretProperties> allSecrets =
client.GetPropertiesOfSecretsAsync();

    await foreach (SecretProperties secret in allSecrets)
    {
        Console.WriteLine($"IterateSecretsWithAwaitForeachAsync:
{secret.Name}");
    }
}
```

In the preceding C# code:

- The SecretClient.GetPropertiesOfSecretsAsync method is invoked and returns an `AsyncPageable<SecretProperties>` object.
- In an `await foreach` loop, each `SecretProperties` is asynchronously yielded.
- As each `secret` is materialized, its `Name` is written to the console.

Iterate over `AsyncPageable` with `while`

To iterate over an `AsyncPageable<T>` when the `await foreach` syntax isn't available, use a `while` loop.

C#

```
async Task IterateSecretsWithWhileLoopAsync()
{
    AsyncPageable<SecretProperties> allSecrets =
client.GetPropertiesOfSecretsAsync();

    IAsyncEnumerator<SecretProperties> enumerator =
allSecrets.GetAsyncEnumerator();
    try
    {
        while (await enumerator.MoveNextAsync())
        {
            SecretProperties secret = enumerator.Current;
            Console.WriteLine($"IterateSecretsWithWhileLoopAsync:
{secret.Name}");
        }
    }
    finally
    {
        await enumerator.DisposeAsync();
    }
}
```

In the preceding C# code:

- The SecretClient.GetPropertiesOfSecretsAsync method is invoked, and returns an `AsyncPageable<SecretProperties>` object.
- The AsyncPageable<T>.GetAsyncEnumerator method is invoked, returning an `IAsyncEnumerator<SecretProperties>`.
- The MoveNextAsync() method is invoked repeatedly until there are no items to return.

Iterate over `AsyncPageable` pages

If you want control over receiving pages of values from the service, use the AsyncPageable<T>.AsPages method:

C#

```
async Task IterateSecretsAsPagesAsync()
{
    AsyncPageable<SecretProperties> allSecrets =
client.GetPropertiesOfSecretsAsync();

    await foreach (Page<SecretProperties> page in allSecrets.AsPages())
    {
        foreach (SecretProperties secret in page.Values)
        {
            Console.WriteLine($"IterateSecretsAsPagesAsync: {secret.Name}");
        }

        // The continuation token that can be used in AsPages call to resume
enumeration
        Console.WriteLine(page.ContinuationToken);
    }
}
```

In the preceding C# code:

- The SecretClient.GetPropertiesOfSecretsAsync method is invoked and returns an `AsyncPageable<SecretProperties>` object.
- The AsyncPageable<T>.AsPages method is invoked and returns an `IAsyncEnumerable<Page<SecretProperties>>`.
- Each page is iterated over asynchronously, using `await foreach`.
- Each page has a set of Page<T>.Values, which represents an `IReadOnlyList<T>` that's iterated over with a synchronous `foreach`.

- Each page also contains a Page<T>.ContinuationToken, which can be used to request the next page.

Use `System.Linq.Async` with `AsyncPageable`

The System.Linq.Async package provides a set of LINQ methods that operate on IAsyncEnumerable<T> type. Because `AsyncPageable<T>` implements IAsyncEnumerable<T>, you can use `System.Linq.Async` to query and transform the data.

Convert to a `List<T>`

Use `ToListAsync` to convert an `AsyncPageable<T>` to a `List<T>`. This method might make several service calls if the data isn't returned in a single page.

C#

```
async Task ToListAsync()
{
    AsyncPageable<SecretProperties> allSecrets =
        client.GetPropertiesOfSecretsAsync();

    List<SecretProperties> secretList = await allSecrets.ToListAsync();

    secretList.ForEach(secret =>
        Console.WriteLine($"ToListAsync: {secret.Name}"));
}
```

In the preceding C# code:

- The SecretClient.GetPropertiesOfSecretsAsync method is invoked and returns an `AsyncPageable<SecretProperties>` object.
- The `ToListAsync` method is awaited, which materializes a new `List<SecretProperties>` instance.

Take the first N elements

`Take` can be used to get only the first `N` elements of the `AsyncPageable`. Using `Take` will make the fewest service calls required to get `N` items.

C#

```
async Task TakeAsync(int count = 30)
{
    AsyncPageable<SecretProperties> allSecrets =
```

```
        client.GetPropertiesOfSecretsAsync();

    await foreach (SecretProperties secret in allSecrets.Take(count))
    {
        Console.WriteLine($"TakeAsync: {secret.Name}");
    }
}
```

More methods

`System.Linq.Async` provides other methods that provide functionality equivalent to their synchronous Enumerable counterparts. Examples of such methods include `Select`, `Where`, `OrderBy`, and `GroupBy`.

Beware client-side evaluation

When using the `System.Linq.Async` package, beware that LINQ operations are executed on the client. The following query would fetch *all* the items just to count them:

C#

```
// ⚠ DON'T DO THIS! 😳
int expensiveSecretCount =
    await client.GetPropertiesOfSecretsAsync()
        .CountAsync();
```

> ⚠ **Warning**
>
> The same warning applies to operators like `Where`. Always prefer server-side filtering, aggregation, or projections of data if available.

As an observable sequence

The System.Linq.Async package is primarily used to provide observer pattern capabilities over `IAsyncEnumerable<T>` sequences. Asynchronous streams are pull-based. As their items are iterated over, the next available item is *pulled*. This approach is in juxtaposition with the observer pattern, which is push-based. As items become available, they're *pushed* to subscribers who act as observers. The `System.Linq.Async` package provides the `ToObservable` extension method that lets you convert an `IAsyncEnumerable<T>` to an IObservable<T>.

Imagine an `IObserver<SecretProperties>` implementation:

```
C#

sealed file class SecretPropertyObserver : IObserver<SecretProperties>
{
    public void OnCompleted() =>
        Console.WriteLine("Done observing secrets");

    public void OnError(Exception error) =>
        Console.WriteLine($"Error observing secrets: {error}");

    public void OnNext(SecretProperties secret) =>
        Console.WriteLine($"Observable: {secret.Name}");
}
```

You could consume the `ToObservable` extension method as follows:

```
C#

IDisposable UseTheToObservableMethod()
{
    AsyncPageable<SecretProperties> allSecrets =
        client.GetPropertiesOfSecretsAsync();

    IObservable<SecretProperties> observable = allSecrets.ToObservable();

    return observable.Subscribe(
        new SecretPropertyObserver());
}
```

In the preceding C# code:

- The SecretClient.GetPropertiesOfSecretsAsync method is invoked and returns an `AsyncPageable<SecretProperties>` object.
- The `ToObservable()` method is called on the `AsyncPageable<SecretProperties>` instance, returning an `IObservable<SecretProperties>`.
- The `observable` is subscribed to, passing in the observer implementation, returning the subscription to the caller.
- The subscription is an `IDisposable`. When it's disposed, the subscription ends.

Iterate over pageable

`Pageable<T>` is a synchronous version of `AsyncPageable<T>` that can be used with a normal `foreach` loop.

```
```

C#

```
void IterateWithPageable()
{
    Pageable<SecretProperties> allSecrets = client.GetPropertiesOfSecrets();

    foreach (SecretProperties secret in allSecrets)
    {
        Console.WriteLine($"IterateWithPageable: {secret.Name}");
    }
}
```

> **ⓘ Important**
>
> While this synchronous API is available, use the asynchronous API alternatives for a better experience.

See also

- Dependency injection with the Azure SDK for .NET
- Thread safety and client lifetime management for Azure SDK objects
- System.Linq.Async
- Task-based asynchronous pattern

Unit testing and mocking with the Azure SDK for .NET

Article • 12/05/2023

Unit testing is an important part of a sustainable development process that can improve code quality and prevent regressions or bugs in your apps. However, unit testing presents challenges when the code you're testing performs network calls, such as those made to Azure resources. Tests that run against live services can experience issues, such as latency that slows down test execution, dependencies on code outside of the isolated test, and issues with managing service state and costs every time the test is run. Instead of testing against live Azure services, replace the service clients with mocked or in-memory implementations. This avoids the above issues and lets developers focus on testing their application logic, independent from the network and service.

In this article, you learn how to write unit tests for the Azure SDK for .NET that isolate your dependencies to make your tests more reliable. You also learn how to replace key components with in-memory test implementations to create fast and reliable unit tests, and see how to design your own classes to better support unit testing. This article includes examples that use Moq and NSubstitute, which are popular mocking libraries for .NET.

Understand service clients

A service client class is the main entry point for developers in an Azure SDK library and implements most of the logic to communicate with the Azure service. When unit testing service client classes, it's important to be able to create an instance of the client that behaves as expected without making any network calls.

Each of the Azure SDK clients follows mocking guidelines that allow their behavior to be overridden:

- Each client offers at least one protected constructor to allow inheritance for testing.
- All public client members are virtual to allow overriding.

> **Note**
>
> The code examples in this article use types from the **Azure.Security.KeyVault.Secrets** library for the Azure Key Vault service. The

concepts demonstrated in this article also apply to service clients from many other Azure services, such as Azure Storage or Azure Service Bus.

To create a test service client, you can either use a mocking library or standard C# features such as inheritance. Mocking frameworks allow you to simplify the code that you must write to override member behavior. (These frameworks also have other useful features that are beyond the scope of this article.)

Non-library

To create a test client instance using C# without a mocking library, inherit from the client type and override methods you're calling in your code with an implementation that returns a set of test objects. Most clients contain both synchronous and asynchronous methods for operations; override only the one your application code is calling.

> **Note**
>
> It can be cumbersome to manually define test classes, especially if you need to customize behavior differently for each test. Consider using a library like Moq or NSubstitute to streamline your testing.

C#

```
using Azure.Security.KeyVault.Secrets;
using Azure;
using NSubstitute.Routing.Handlers;

namespace UnitTestingSampleApp.NonLibrary;

public sealed class MockSecretClient : SecretClient
{
    AsyncPageable<SecretProperties> _pageable;

    // Allow a pageable to be passed in for mocking different responses
    public MockSecretClient(AsyncPageable<SecretProperties> pageable)
    {
        _pageable = pageable;
    }

    public override Response<KeyVaultSecret> GetSecret(
        string name,
        string version = null,
        CancellationToken cancellationToken = default)
        => throw new NotImplementedException();
```

```
    public override Task<Response<KeyVaultSecret>> GetSecretAsync(
        string name,
        string version = null,
        CancellationToken cancellationToken = default)
        => throw new NotImplementedException();

    // Return the pageable that was passed in
    public override AsyncPageable<SecretProperties>
GetPropertiesOfSecretsAsync
        (CancellationToken cancellationToken = default)
        => _pageable;
}
```

Service client input and output models

Model types hold the data being sent and received from Azure services. There are three types of models:

- **Input models** are intended to be created and passed as parameters to service methods by developers. They have one or more public constructors and writeable properties.
- **Output models** are only returned by the service and have no public constructors or writeable properties.
- **Round-trip models** are less common, but are returned by the service, modified, and used as an input.

To create a test instance of an input model, use one of the available public constructors and set the additional properties you need.

C#

```
var secretProperties = new SecretProperties("secret")
{
    NotBefore = DateTimeOffset.Now
};
```

To create instances of output models, a model factory is used. Azure SDK client libraries provide a static model factory class with a `ModelFactory` suffix in its name. The class contains a set of static methods to initialize the library's output model types. For example, the model factory for `SecretClient` is `SecretModelFactory`:

C#

```
KeyVaultSecret keyVaultSecret = SecretModelFactory.KeyVaultSecret(
```

```
    new SecretProperties("secret"), "secretValue");
```

> **Note**
>
> Some input models have read-only properties that are only populated when the model is returned by the service. In this case, a model factory method will be available that allows setting these properties. For example, **SecretProperties**.

C#

```
// CreatedOn is a read-only property and can only be
// set via the model factory's SecretProperties method.
secretPropertiesWithCreatedOn = SecretModelFactory.SecretProperties(
    name: "secret", createdOn: DateTimeOffset.Now);
```

Explore response types

The Response class is an abstract class that represents an HTTP response and is returned by most service client methods. You can create test `Response` instances using either a mocking library or standard C# inheritance.

Non-library

The `Response` class is abstract, which means there are many members to override. Consider using a library to streamline your approach.

C#

```
using Azure.Core;
using Azure;
using System.Diagnostics.CodeAnalysis;

namespace UnitTestingSampleApp.NonLibrary;

public sealed class MockResponse : Response
{
    public override int Status => throw new NotImplementedException();

    public override string ReasonPhrase => throw new
NotImplementedException();

    public override Stream? ContentStream
    {
        get => throw new NotImplementedException();
        set => throw new NotImplementedException();
```

```
    }
    public override string ClientRequestId
    {
        get => throw new NotImplementedException();
        set => throw new NotImplementedException();
    }

    public override void Dispose() =>
        throw new NotImplementedException();
    protected override bool ContainsHeader(string name) =>
        throw new NotImplementedException();
    protected override IEnumerable<HttpHeader> EnumerateHeaders() =>
        throw new NotImplementedException();
    protected override bool TryGetHeader(
        string name,
        [NotNullWhen(true)] out string? value) =>
        throw new NotImplementedException();
    protected override bool TryGetHeaderValues(
        string name,
        [NotNullWhen(true)] out IEnumerable<string>? values) =>
        throw new NotImplementedException();
}
```

Some services also support using the Response<T> type, which is a class that contains a model and the HTTP response that returned it. To create a test instance of `Response<T>`, use the static `Response.FromValue` method:

Non-library

C#

```
KeyVaultSecret keyVaultSecret = SecretModelFactory.KeyVaultSecret(
    new SecretProperties("secret"), "secretValue");

Response<KeyVaultSecret> response = Response.FromValue(keyVaultSecret,
new MockResponse());
```

Explore paging

The Page<T> class is used as a building block in service methods that invoke operations returning results in multiple pages. The `Page<T>` is rarely returned from APIs directly but is useful to create the `AsyncPageable<T>` and `Pageable<T>` instances in the next section. To create a `Page<T>` instance, use the Page<T>.FromValues method, passing a list of items, a continuation token, and the `Response`.

The `continuationToken` parameter is used to retrieve the next page from the service. For unit testing purposes, it should be set to `null` for the last page and should be nonempty for other pages.

Non-library

C#

```
Page<SecretProperties> responsePage = Page<SecretProperties>.FromValues(
    new[] {
        new SecretProperties("secret1"),
        new SecretProperties("secret2")
    },
    continuationToken: null,
    new MockResponse());
```

AsyncPageable<T> and Pageable<T> are classes that represent collections of models returned by the service in pages. The only difference between them is that one is used with synchronous methods while the other is used with asynchronous methods.

To create a test instance of `Pageable` or `AsyncPageable`, use the FromPages static method:

Non-library

C#

```
Page<SecretProperties> page1 = Page<SecretProperties>.FromValues(
    new[]
    {
        new SecretProperties("secret1"),
        new SecretProperties("secret2")
    },
    "continuationToken",
    new MockResponse());

Page<SecretProperties> page2 = Page<SecretProperties>.FromValues(
    new[]
    {
        new SecretProperties("secret3"),
        new SecretProperties("secret4")
    },
    "continuationToken2",
    new MockResponse());

Page<SecretProperties> lastPage = Page<SecretProperties>.FromValues(
    new[]
```

```
    {
        new SecretProperties("secret5"),
        new SecretProperties("secret6")
    },
    continuationToken: null,
    new MockResponse());

Pageable<SecretProperties> pageable = Pageable<SecretProperties>
    .FromPages(new[] { page1, page2, lastPage });

AsyncPageable<SecretProperties> asyncPageable =
AsyncPageable<SecretProperties>
    .FromPages(new[] { page1, page2, lastPage });
```

Write a mocked unit test

Suppose your app contains a class that finds the names of keys that will expire within a given amount of time.

C#

```
using Azure.Security.KeyVault.Secrets;

public class AboutToExpireSecretFinder
{
    private readonly TimeSpan _threshold;
    private readonly SecretClient _client;

    public AboutToExpireSecretFinder(TimeSpan threshold, SecretClient
client)
    {
        _threshold = threshold;
        _client = client;
    }

    public async Task<string[]> GetAboutToExpireSecretsAsync()
    {
        List<string> secretsAboutToExpire = new();

        await foreach (var secret in _client.GetPropertiesOfSecretsAsync())
        {
            if (secret.ExpiresOn.HasValue &&
                secret.ExpiresOn.Value - DateTimeOffset.Now <= _threshold)
            {
                secretsAboutToExpire.Add(secret.Name);
            }
        }

        return secretsAboutToExpire.ToArray();
```

```
    }
}
```

You want to test the following behaviors of the `AboutToExpireSecretFinder` to ensure they continue working as expected:

- Secrets without an expiry date set aren't returned.
- Secrets with an expiry date closer to the current date than the threshold are returned.

When unit testing you only want the unit tests to verify the application logic and not whether the Azure service or library works correctly. The following example tests the key behaviors using the popular xUnit library:

Non-library

C#

```
using Azure;
using Azure.Security.KeyVault.Secrets;

namespace UnitTestingSampleApp.NonLibrary;

public class AboutToExpireSecretFinderTests
{
    [Fact]
    public async Task DoesNotReturnNonExpiringSecrets()
    {
        // Arrange
        // Create a page of enumeration results
        Page<SecretProperties> page =
Page<SecretProperties>.FromValues(new[]
        {
            new SecretProperties("secret1") { ExpiresOn = null },
            new SecretProperties("secret2") { ExpiresOn = null }
        }, null, new MockResponse());

        // Create a pageable that consists of a single page
        AsyncPageable<SecretProperties> pageable =
            AsyncPageable<SecretProperties>.FromPages(new[] { page });

        var clientMock = new MockSecretClient(pageable);

        // Create an instance of a class to test passing in the mock
client
        var finder = new AboutToExpireSecretFinder(TimeSpan.FromDays(2),
clientMock);

        // Act
        string[] soonToExpire = await
```

```
finder.GetAboutToExpireSecretsAsync();

        // Assert
        Assert.Empty(soonToExpire);
    }

    [Fact]
    public async Task ReturnsSecretsThatExpireSoon()
    {
        // Arrange

        // Create a page of enumeration results
        DateTimeOffset now = DateTimeOffset.Now;
        Page<SecretProperties> page =
Page<SecretProperties>.FromValues(new[]
        {
            new SecretProperties("secret1") { ExpiresOn = now.AddDays(1)
},
            new SecretProperties("secret2") { ExpiresOn = now.AddDays(2)
},
            new SecretProperties("secret3") { ExpiresOn = now.AddDays(3)
}
        },
        null, new MockResponse());

        // Create a pageable that consists of a single page
        AsyncPageable<SecretProperties> pageable =
            AsyncPageable<SecretProperties>.FromPages(new[] { page });

        // Create a client mock object
        var clientMock = new MockSecretClient(pageable);

        // Create an instance of a class to test passing in the mock
client
        var finder = new AboutToExpireSecretFinder(TimeSpan.FromDays(2),
clientMock);

        // Act
        string[] soonToExpire = await
finder.GetAboutToExpireSecretsAsync();

        // Assert
        Assert.Equal(new[] { "secret1", "secret2" }, soonToExpire);
    }
}
```

Refactor your types for testability

Classes that need to be tested should be designed for dependency injection, which allows the class to receive its dependencies instead of creating them internally. It was a

seamless process to replace the `SecretClient` implementation in the example from the previous section because it was one of the constructor parameters. However, there might be classes in your code that create their own dependencies and aren't easily testable, such as the following class:

C#

```
public class AboutToExpireSecretFinder
{
    public AboutToExpireSecretFinder(TimeSpan threshold)
    {
        _threshold = threshold;
        _client = new SecretClient(
            new Uri(Environment.GetEnvironmentVariable("KeyVaultUri")),
            new DefaultAzureCredential());
    }
}
```

The simplest refactoring you can do to enable testing with dependency injection would be to expose the client as a parameter and run default creation code when no value is provided. This approach allows you to make the class testable while still retaining the flexibility of using the type without much ceremony.

C#

```
public class AboutToExpireSecretFinder
{
    public AboutToExpireSecretFinder(TimeSpan threshold, SecretClient client
= null)
    {
        _threshold = threshold;
        _client = client ?? new SecretClient(
            new Uri(Environment.GetEnvironmentVariable("KeyVaultUri")),
            new DefaultAzureCredential());
    }
}
```

Another option is to move the dependency creation entirely into the calling code:

C#

```
public class AboutToExpireSecretFinder
{
    public AboutToExpireSecretFinder(TimeSpan threshold, SecretClient
client)
    {
        _threshold = threshold;
        _client = client;
    }
```

```
}

var secretClient = new SecretClient(
    new Uri(Environment.GetEnvironmentVariable("KeyVaultUri")),
    new DefaultAzureCredential());
var finder = new AboutToExpireSecretFinder(TimeSpan.FromDays(2),
secretClient);
```

This approach is useful when you would like to consolidate the dependency creation and share the client between multiple consuming classes.

Understand Azure Resource Manager (ARM) clients

In ARM libraries, the clients were designed to emphasize their relationship to one another, mirroring the service hierarchy. To achieve that goal, extension methods are widely used to add additional features to clients.

For example, an Azure virtual machine exists in an Azure resource group. The `Azure.ResourceManager.Compute` namespace models the Azure virtual machine as `VirtualMachineResource`. The `Azure.ResourceManager` namespace models the Azure resource group as `ResourceGroupResource`. To query the virtual machines for a resource group, you would write:

C#

```
VirtualMachineCollection virtualMachineCollection =
resourceGroup.GetVirtualMachines();
```

Because the virtual machine-related functionality such as `GetVirtualMachines` on `ResourceGroupResource`, is implemented as extension methods, it's impossible to just create a mock of the type and override the method. Instead, you'll also have to create a mock class for the "mockable resource" and wire them together.

The mockable resource type is always in the `Mocking` sub-namespace of the extension method. In the preceding example, the mockable resource type is in the `Azure.ResourceManager.Compute.Mocking` namespace. The mockable resource type is always named after the resource type with "Mockable" and the library name as prefixes. In the preceding example, the mockable resource type is named `MockableComputeResourceGroupResource`, where `ResourceGroupResource` is the resource type of the extension method, and `Compute` is the library name.

One more requirement before you get the unit test running is to mock the `GetCachedClient` method on the resource type of the extension method. Completing this step hooks up the extension method and the method on the mockable resource type.

Here's how it works:

Non-library

C#

```
using Azure.Core;

namespace UnitTestingSampleApp.ResourceManager.NonLibrary;

public sealed class MockMockableComputeResourceGroupResource :
MockableComputeResourceGroupResource
{
    private VirtualMachineCollection _virtualMachineCollection;
    public
MockMockableComputeResourceGroupResource(VirtualMachineCollection
virtualMachineCollection)
    {
        _virtualMachineCollection = virtualMachineCollection;
    }

    public override VirtualMachineCollection GetVirtualMachines()
    {
        return _virtualMachineCollection;
    }
}

public sealed class MockResourceGroupResource : ResourceGroupResource
{
    private readonly MockableComputeResourceGroupResource
_mockableComputeResourceGroupResource;
    public MockResourceGroupResource(VirtualMachineCollection
virtualMachineCollection)
    {
        _mockableComputeResourceGroupResource =
            new
MockMockableComputeResourceGroupResource(virtualMachineCollection);
    }

    internal MockResourceGroupResource(ArmClient client,
ResourceIdentifier id) : base(client, id)
    {}

    public override T GetCachedClient<T>(Func<ArmClient, T> factory)
where T : class
    {
        if (typeof(T) == typeof(MockableComputeResourceGroupResource))
```

```
            return _mockableComputeResourceGroupResource as T;
        return base.GetCachedClient(factory);
    }
}

public sealed class MockVirtualMachineCollection :
VirtualMachineCollection
{
    public MockVirtualMachineCollection()
    {}

    internal MockVirtualMachineCollection(ArmClient client,
ResourceIdentifier id) : base(client, id)
    {}
}
```

See also

- Dependency injection in .NET
- Unit testing best practices
- Unit testing C# in .NET using dotnet test and xUnit

Configure a proxy server when using the Azure SDK for .NET

Article • 03/24/2023

If your organization requires the use of a proxy server to access internet resources, you will need to set an environment variable with the proxy server information to use the Azure SDK for .NET.

Configuration using environment variables

Depending on if your proxy server uses HTTP or HTTPS, you will set either the environment variable `HTTP_PROXY` or `HTTPS_PROXY` respectively. The proxy server URL has the form `http[s]://[username:password@]<ip_address_or_hostname>:<port>/` where the `username:password` combination is optional. To get the IP address or hostname, port and credentials for your proxy server, consult your network administrator.

The following examples show how to set the appropriate environment variables in command shell (Windows) and bash (Linux/Mac) environments. Setting the appropriate environment variable will then cause the Azure SDK for .NET to use the proxy server at run time.

cmd

Windows Command Prompt

```
rem Non-authenticated HTTP server:
set HTTP_PROXY=http://10.10.1.10:1180

rem Authenticated HTTP server:
set HTTP_PROXY=http://username:password@10.10.1.10:1180

rem Non-authenticated HTTPS server:
set HTTPS_PROXY=http://10.10.1.10:1180

rem Authenticated HTTPS server:
set HTTPS_PROXY=http://username:password@10.10.1.10:1180
```

Azure SDK for .NET package index

Article • 03/01/2023

Libraries using Azure.Core

All libraries

Libraries using Azure.Core

Name	Package	Docs	Source
Anomaly Detector	NuGet 3.0.0-preview.7	docs	GitHub 3.0.0-preview.7
App Configuration	NuGet 1.2.0 NuGet 1.3.0-beta.1	docs	GitHub 1.2.0 GitHub 1.3.0-beta.1
Attestation	NuGet 1.0.0	docs	GitHub 1.0.0
Azure Object Anchors Conversion	NuGet 0.3.0-beta.6	docs	GitHub 0.3.0-beta.6
Azure Remote Rendering	NuGet 1.1.0	docs	GitHub 1.1.0
Brokered Authentication	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Cognitive Search	NuGet 11.4.0 NuGet 11.5.0-beta.2	docs	GitHub 11.4.0 GitHub 11.5.0-beta.2
Communication Call Automation	NuGet 1.0.0	docs	GitHub 1.0.0
Communication Chat	NuGet 1.1.2	docs	GitHub 1.1.2
Communication Common	NuGet 1.2.1 NuGet 2.0.0-beta.1	docs	GitHub 1.2.1 GitHub 2.0.0-beta.1
Communication Email	NuGet 1.0.0	docs	GitHub 1.0.0
Communication Identity	NuGet 1.2.0	docs	GitHub 1.2.0
Communication Network Traversal	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1

Name	Package	Docs	Source
Communication Phone Numbers	NuGet 1.1.0	docs	GitHub 1.1.0
Communication Rooms	NuGet 1.0.0	docs	GitHub 1.0.0
Communication SMS	NuGet 1.0.1	docs	GitHub 1.0.1
Confidential Ledger	NuGet 1.1.0	docs	GitHub 1.1.0
Container Registry	NuGet 1.1.0	docs	GitHub 1.1.0
Content Safety	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Conversational Language Understanding	NuGet 1.1.0	docs	GitHub 1.1.0
Core - Client - AMQP	NuGet 1.3.0	docs	GitHub 1.3.0
Core - Client - Core	NuGet 1.33.0	docs	GitHub 1.33.0
Cosmos DB	NuGet 4.0.0-preview3	docs	GitHub 4.0.0-preview3
Data Movement	NuGet 12.0.0-beta.2	docs	GitHub 12.0.0-beta.2
Dev Center	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Device Update	NuGet 1.0.0	docs	GitHub 1.0.0
Digital Twins - Core	NuGet 1.4.0	docs	GitHub 1.4.0
Document Translation	NuGet 1.0.0	docs	GitHub 1.0.0
Event Grid	NuGet 4.17.0	docs	GitHub 4.17.0
Event Hubs	NuGet 5.9.2	docs	GitHub 5.9.2
Event Hubs - Event Processor	NuGet 5.9.2	docs	GitHub 5.9.2
FarmBeats	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Form Recognizer	NuGet 4.0.0 NuGet 4.1.0-beta.1	docs	GitHub 4.0.0 GitHub 4.1.0-beta.1
Health Insights Cancer Profiling	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1

Name	Package	Docs	Source
Health Insights Clinical Matching	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Identity	NuGet 1.9.0	docs	GitHub 1.9.0
Image Analysis	NuGet 0.11.1-beta.1		GitHub 0.11.1-beta.1
Key Vault - Administration	NuGet 4.3.0	docs	GitHub 4.3.0
Key Vault - Certificates	NuGet 4.5.1	docs	GitHub 4.5.1
Key Vault - Keys	NuGet 4.5.0	docs	GitHub 4.5.0
Key Vault - Secrets	NuGet 4.5.0	docs	GitHub 4.5.0
Load Testing	NuGet 1.0.1	docs	GitHub 1.0.1
Maps Common	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Maps Geolocation	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Maps Render	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Maps Route	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Maps Search	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Media Analytics Edge	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Metrics Advisor	NuGet 1.1.0	docs	GitHub 1.1.0
Mixed Reality Authentication	NuGet 1.2.0	docs	GitHub 1.2.0
Monitor Ingestion	NuGet 1.0.0	docs	GitHub 1.0.0
Monitor Query	NuGet 1.2.0	docs	GitHub 1.2.0
OpenAI Inference	NuGet 1.0.0-beta.5	docs	GitHub 1.0.0-beta.5
OpenTelemetry AspNetCore	NuGet 1.0.0-beta.4	docs	GitHub 1.0.0-beta.4

Name	Package	Docs	Source
OpenTelemetry Exporter	NuGet 1.0.0-beta.12	docs	GitHub 1.0.0-beta.12
Personalizer	NuGet 2.0.0-beta.2	docs	GitHub 2.0.0-beta.2
Purview Account	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Purview Administration	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Purview Catalog	NuGet 1.0.0-beta.4	docs	GitHub 1.0.0-beta.4
Purview Scanning	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Purview Share	NuGet 1.0.3-beta.20	docs	GitHub 1.0.3-beta.20
Purview Share	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Purview Workflow	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Question Answering	NuGet 1.1.0	docs	GitHub 1.1.0
Schema Registry	NuGet 1.3.0 NuGet 1.4.0-beta.1	docs	GitHub 1.3.0 GitHub 1.4.0-beta.1
Schema Registry - Avro	NuGet 1.0.0	docs	GitHub 1.0.0
Service Bus	NuGet 7.15.0	docs	GitHub 7.15.0
Storage - Blobs	NuGet 12.16.0 NuGet 12.17.0-beta.1	docs	GitHub 12.16.0 GitHub 12.17.0-beta.1
Storage - Blobs Batch	NuGet 12.13.0 NuGet 12.14.0-beta.1	docs	GitHub 12.13.0 GitHub 12.14.0-beta.1
Storage - Blobs ChangeFeed	NuGet 12.0.0-preview.35	docs	GitHub 12.0.0-preview.35

Name	Package	Docs	Source
Storage - Files Data Lake	NuGet 12.14.0 NuGet 12.15.0-beta.1	docs	GitHub 12.14.0 GitHub 12.15.0-beta.1
Storage - Files Share	NuGet 12.14.0 NuGet 12.15.0-beta.1	docs	GitHub 12.14.0 GitHub 12.15.0-beta.1
Storage - Queues	NuGet 12.14.0 NuGet 12.15.0-beta.1	docs	GitHub 12.14.0 GitHub 12.15.0-beta.1
Synapse - AccessControl	NuGet 1.0.0-preview.5	docs	GitHub 1.0.0-preview.5
Synapse - Artifacts	NuGet 1.0.0-preview.17	docs	GitHub 1.0.0-preview.17
Synapse - Managed Private Endpoints	NuGet 1.0.0-beta.5	docs	GitHub 1.0.0-beta.5
Synapse - Monitoring	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Synapse - Spark	NuGet 1.0.0-preview.8	docs	GitHub 1.0.0-preview.8
Tables	NuGet 12.8.0	docs	GitHub 12.8.0
Text Analytics	NuGet 5.3.0	docs	GitHub 5.3.0
Text Translation	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Video Analyzer Edge	NuGet 1.0.0-beta.6	docs	GitHub 1.0.0-beta.6
Vision Core	NuGet 0.10.0-beta.1		GitHub 0.10.0-beta.1
Web PubSub	NuGet 1.3.0	docs	GitHub 1.3.0
Web PubSub Client	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Azure client library integration for ASP.NET Core	NuGet 1.6.3	docs	GitHub 1.6.3
Blob Storage Key Store for .NET Data Protection	NuGet 1.3.2	docs	GitHub 1.3.2

Name	Package	Docs	Source
CloudNative CloudEvents with Event Grid	NuGet 1.0.0	docs	GitHub 1.0.0
Core - Client - Spatial	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Core - Client - Spatial Newtonsoft Json	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1
Functions extension for Azure Tables	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Key Encryptor for .NET Data Protection	NuGet 1.2.2	docs	GitHub 1.2.2
Secrets Configuration Provider for .NET	NuGet 1.2.2	docs	GitHub 1.2.2
Storage - Common	NuGet 12.15.0 NuGet 12.16.0-beta.1	docs	GitHub 12.15.0 GitHub 12.16.0-beta.1
WebJobs Extensions - Event Grid	NuGet 3.3.0	docs	GitHub 3.3.0
WebJobs Extensions - Event Hubs	NuGet 5.4.0	docs	GitHub 5.4.0
WebJobs Extensions - Service Bus	NuGet 5.11.0	docs	GitHub 5.11.0
WebJobs Extensions - SignalR Service	NuGet 1.11.0	docs	GitHub 1.11.0
WebJobs Extensions - Storage	NuGet 5.1.3	docs	GitHub 5.1.3
WebJobs Extensions - Web PubSub	NuGet 1.5.0	docs	GitHub 1.5.0
Cosmos DB for PostgreSQL	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
IotFirmwareDefense	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Resource Management - Advisor	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Agrifood	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Alerts Management	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1

Name	Package	Docs	Source
Resource Management - Analysis	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Api Management	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - App Compliance Automation	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - App Configuration	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1
Resource Management - App Platform	NuGet 1.0.2 NuGet 1.1.0-beta.2	docs	GitHub 1.0.2 GitHub 1.1.0-beta.2
Resource Management - App Service	NuGet 1.0.2 NuGet 1.1.0-beta.1	docs	GitHub 1.0.2 GitHub 1.1.0-beta.1
Resource Management - Application Insights	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Arc ScVmm	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Attestation	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Authorization	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Automanage	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1
Resource Management - Automation	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Azure Stack HCI	NuGet 1.0.2	docs	GitHub 1.0.2
Resource Management - Azure VMware Solution	NuGet 1.1.1 NuGet 1.2.0-beta.1	docs	GitHub 1.1.1 GitHub 1.2.0-beta.1

Name	Package	Docs	Source
Resource Management - Backup	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1
Resource Management - Batch	NuGet 1.2.0	docs	GitHub 1.2.0
Resource Management - Billing	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Billing Benefits	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Resource Management - Blueprint	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Bot Service	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Change Analysis	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Chaos	NuGet 1.0.0-beta.4	docs	GitHub 1.0.0-beta.4
Resource Management - Cognitive Search	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Cognitive Services	NuGet 1.3.0	docs	GitHub 1.3.0
Resource Management - Communication	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Compute	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Confidential Ledger	NuGet 1.0.1 NuGet 1.1.0-beta.2	docs	GitHub 1.0.1 GitHub 1.1.0-beta.2
Resource Management - Confluent	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1

Name	Package	Docs	Source
Resource Management - Connected VMware vSphere	NuGet 1.0.0-beta.7	docs	GitHub 1.0.0-beta.7
Resource Management - Consumption	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Container Apps	NuGet 1.0.3 NuGet 1.1.0-beta.2	docs	GitHub 1.0.3 GitHub 1.1.0-beta.2
Resource Management - Container Instances	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Container Registry	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Container Service	NuGet 1.1.0 NuGet 1.2.0-beta.2	docs	GitHub 1.1.0 GitHub 1.2.0-beta.2
Resource Management - Content Delivery Network	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Core	NuGet 1.6.0	docs	GitHub 1.6.0
Resource Management - Cosmos DB	NuGet 1.3.0 NuGet 1.4.0-beta.2	docs	GitHub 1.3.0 GitHub 1.4.0-beta.2
Resource Management - Costmanagement	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Resource Management - Customer Insights	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Data Box	NuGet 1.0.2	docs	GitHub 1.0.2
Resource Management - Data Box Edge	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Data Factory	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2

Name	Package	Docs	Source
Resource Management - Data Lake Analytics	NuGet 1.0.0 NuGet 1.1.0-beta.2	docs	GitHub 1.0.0 GitHub 1.1.0-beta.2
Resource Management - Data Lake Store	NuGet 1.0.0 NuGet 1.1.0-beta.2	docs	GitHub 1.0.0 GitHub 1.1.0-beta.2
Resource Management - Data Migration	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Data Protection	NuGet 1.1.2 NuGet 1.2.0-beta.1	docs	GitHub 1.1.2 GitHub 1.2.0-beta.1
Resource Management - Data Share	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Datadog	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Deployment Manager	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Desktop Virtualization	NuGet 1.1.0	docs	GitHub 1.1.0
Resource Management - Dev Center	NuGet 1.0.0	docs	GitHub 1.0.0
Resource Management - Dev Spaces	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Device Provisioning Services	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Device Update	NuGet 1.0.0-beta.9	docs	GitHub 1.0.0-beta.9
Resource Management - DevTest Labs	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Digital Twins	NuGet 1.2.0 NuGet 1.3.0-beta.1	docs	GitHub 1.2.0 GitHub 1.3.0-beta.1

Name	Package	Docs	Source
Resource Management - DNS	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - DNS Resolver	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Dynatrace	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Edge Order	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Elastic	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - ElasticSan	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Energyservices	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Resource Management - Event Grid	NuGet 1.0.1 NuGet 1.1.0-beta.2	docs	GitHub 1.0.1 GitHub 1.1.0-beta.2
Resource Management - Event Hubs	NuGet 1.0.0 NuGet 1.1.0-beta.3	docs	GitHub 1.0.0 GitHub 1.1.0-beta.3
Resource Management - Extended Location	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Fluid Relay	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Front Door	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Graph Services	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1

Name	Package	Docs	Source
Resource Management - Guest Configuration	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - HDInsight	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Health Bot	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Healthcare APIs	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Hybrid Compute	NuGet 1.0.0-beta.4	docs	GitHub 1.0.0-beta.4
Resource Management - Hybrid Connectivity	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Hybrid Container Service	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Resource Management - Hybrid Data	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Hybrid Kubernetes	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - IoT Central	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - IoT Hub	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Key Vault	NuGet 1.1.0 NuGet 1.2.0-beta.2	docs	GitHub 1.1.0 GitHub 1.2.0-beta.2
Resource Management - Kubernetes Configuration	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1

Name	Package	Docs	Source
Resource Management - Kusto	NuGet 1.3.0 NuGet 1.4.0-beta.1	docs	GitHub 1.3.0 GitHub 1.4.0-beta.1
Resource Management - Lab Services	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Load Testing	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Log Analytics	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Logic	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Machine Learning	NuGet 1.1.1 NuGet 1.2.0-beta.1	docs	GitHub 1.1.1 GitHub 1.2.0-beta.1
Resource Management - Machine Learning Compute	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Maintenance	NuGet 1.1.2 NuGet 1.2.0-beta.1	docs	GitHub 1.1.2 GitHub 1.2.0-beta.1
Resource Management - Managed Grafana	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Managed Network	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Managed Service Identity	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Managed Services	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Managednetworkfabric	NuGet 1.0.0-beta.1		GitHub 1.0.0-beta.1

Name	Package	Docs	Source
Resource Management - Management Partner	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Maps	NuGet 1.0.1 NuGet 1.1.0-beta.2	docs	GitHub 1.0.1 GitHub 1.1.0-beta.2
Resource Management - Marketplace	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Marketplace Ordering	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Media	NuGet 1.2.0 NuGet 1.3.0-beta.1	docs	GitHub 1.2.0 GitHub 1.3.0-beta.1
Resource Management - Mixed Reality	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Mobile Network	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Resource Management - Monitor	NuGet 1.2.0 NuGet 1.3.0-beta.2	docs	GitHub 1.2.0 GitHub 1.3.0-beta.2
Resource Management - MySQL	NuGet 1.0.1 NuGet 1.1.0-beta.2	docs	GitHub 1.0.1 GitHub 1.1.0-beta.2
Resource Management - NetApp Files	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Network	NuGet 1.3.0	docs	GitHub 1.3.0
Resource Management - Network Function	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Networkcloud	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Resource Management - New Relic Observability	NuGet 1.0.0	docs	GitHub 1.0.0

Name	Package	Docs	Source
Resource Management - Nginx	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Resource Management - Notification Hubs	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Orbital	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Paloaltonetworks.Ngfw	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Resource Management - Peering	NuGet 1.1.1 NuGet 1.2.0-beta.1	docs	GitHub 1.1.1 GitHub 1.2.0-beta.1
Resource Management - Policy Insights	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - PostgreSQL	NuGet 1.1.1 NuGet 1.2.0-beta.1	docs	GitHub 1.1.1 GitHub 1.2.0-beta.1
Resource Management - Power BI Dedicated	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Private DNS	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Provider Hub	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1
Resource Management - Purview	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Quantum	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Qumulo	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1
Resource Management - Quota	NuGet 1.0.0	docs	GitHub 1.0.0

Name	Package	Docs	Source
Resource Management - Recovery Services	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1
Resource Management - Redis	NuGet 1.2.0	docs	GitHub 1.2.0
Resource Management - Redis Enterprise	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Redis Enterprise	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Resource Management - Relay	NuGet 1.1.1 NuGet 1.2.0-beta.1	docs	GitHub 1.1.1 GitHub 1.2.0-beta.1
Resource Management - Reservations	NuGet 1.3.0 NuGet 1.4.0-beta.1	docs	GitHub 1.3.0 GitHub 1.4.0-beta.1
Resource Management - Resource Graph	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Resource Health	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1
Resource Management - Resource Mover	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Resources	NuGet 1.6.0	docs	GitHub 1.6.0
Resource Management - Security	NuGet 1.1.0 NuGet 1.2.0-beta.2	docs	GitHub 1.1.0 GitHub 1.2.0-beta.2
Resource Management - Security DevOps	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Security Insights	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Selfhelp	NuGet 1.0.0	docs	GitHub 1.0.0

Name	Package	Docs	Source
Resource Management - Service Bus	NuGet 1.0.0 NuGet 1.1.0-beta.3	docs	GitHub 1.0.0 GitHub 1.1.0-beta.3
Resource Management - Service Fabric	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Service Fabric Managed Clusters	NuGet 1.0.1 NuGet 1.1.0-beta.2	docs	GitHub 1.0.1 GitHub 1.1.0-beta.2
Resource Management - Service Linker	NuGet 1.0.2 NuGet 1.1.0-beta.1	docs	GitHub 1.0.2 GitHub 1.1.0-beta.1
Resource Management - Service Networking	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Resource Management - SignalR	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Site Recovery	NuGet 1.0.0	docs	GitHub 1.0.0
Resource Management - SQL	NuGet 1.2.0 NuGet 1.3.0-beta.1	docs	GitHub 1.2.0 GitHub 1.3.0-beta.1
Resource Management - SQL Virtual Machine	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Storage	NuGet 1.1.1 NuGet 1.2.0-beta.1	docs	GitHub 1.1.1 GitHub 1.2.0-beta.1
Resource Management - Storage Cache	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Storage Mover	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Storage Pool	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1

Name	Package	Docs	Source
Resource Management - Storage Sync	NuGet 1.1.1 NuGet 1.2.0-beta.1	docs	GitHub 1.1.1 GitHub 1.2.0-beta.1
Resource Management - Stream Analytics	NuGet 1.1.1 NuGet 1.2.0-beta.1	docs	GitHub 1.1.1 GitHub 1.2.0-beta.1
Resource Management - Subscriptions	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Support	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Synapse	NuGet 1.1.1 NuGet 1.2.0-beta.1	docs	GitHub 1.1.1 GitHub 1.2.0-beta.1
Resource Management - Traffic Manager	NuGet 1.1.0	docs	GitHub 1.1.0
Resource Management - Voice Services	NuGet 1.0.0	docs	GitHub 1.0.0
Resource Management - Web PubSub	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Workload Monitor	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Workloads	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1

All libraries

Name	Package	Docs	Source
Anomaly Detector	NuGet 3.0.0-preview.7	docs	GitHub 3.0.0-preview.7

Name	Package	Docs	Source
App Configuration	NuGet 1.2.0 NuGet 1.3.0-beta.1	docs	GitHub 1.2.0 GitHub 1.3.0-beta.1
Attestation	NuGet 1.0.0	docs	GitHub 1.0.0
Azure Object Anchors Conversion	NuGet 0.3.0-beta.6	docs	GitHub 0.3.0-beta.6
Azure Remote Rendering	NuGet 1.1.0	docs	GitHub 1.1.0
Brokered Authentication	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Cognitive Search	NuGet 11.4.0 NuGet 11.5.0-beta.2	docs	GitHub 11.4.0 GitHub 11.5.0-beta.2
Communication Call Automation	NuGet 1.0.0	docs	GitHub 1.0.0
Communication Chat	NuGet 1.1.2	docs	GitHub 1.1.2
Communication Common	NuGet 1.2.1 NuGet 2.0.0-beta.1	docs	GitHub 1.2.1 GitHub 2.0.0-beta.1
Communication Email	NuGet 1.0.0	docs	GitHub 1.0.0
Communication Identity	NuGet 1.2.0	docs	GitHub 1.2.0
Communication Network Traversal	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1

Name	Package	Docs	Source
Communication Phone Numbers	NuGet 1.1.0	docs	GitHub 1.1.0
Communication Rooms	NuGet 1.0.0	docs	GitHub 1.0.0
Communication SMS	NuGet 1.0.1	docs	GitHub 1.0.1
Confidential Ledger	NuGet 1.1.0	docs	GitHub 1.1.0
Container Registry	NuGet 1.1.0	docs	GitHub 1.1.0
Content Safety	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Conversational Language Understanding	NuGet 1.1.0	docs	GitHub 1.1.0
Core - Client - AMQP	NuGet 1.3.0	docs	GitHub 1.3.0
Core - Client - Core	NuGet 1.33.0	docs	GitHub 1.33.0
Cosmos DB	NuGet 4.0.0-preview3	docs	GitHub 4.0.0-preview3
Data Movement	NuGet 12.0.0-beta.2	docs	GitHub 12.0.0-beta.2
Dev Center	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Device Update	NuGet 1.0.0	docs	GitHub 1.0.0
Digital Twins - Core	NuGet 1.4.0	docs	GitHub 1.4.0
Document Translation	NuGet 1.0.0	docs	GitHub 1.0.0

Name	Package	Docs	Source
Event Grid	NuGet 4.17.0	docs	GitHub 4.17.0
Event Hubs	NuGet 5.9.2	docs	GitHub 5.9.2
Event Hubs - Event Processor	NuGet 5.9.2	docs	GitHub 5.9.2
FarmBeats	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Form Recognizer	NuGet 4.0.0 NuGet 4.1.0-beta.1	docs	GitHub 4.0.0 GitHub 4.1.0-beta.1
Health Insights Cancer Profiling	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Health Insights Clinical Matching	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Identity	NuGet 1.9.0	docs	GitHub 1.9.0
Image Analysis	NuGet 0.11.1-beta.1		GitHub 0.11.1-beta.1
Key Vault - Administration	NuGet 4.3.0	docs	GitHub 4.3.0
Key Vault - Certificates	NuGet 4.5.1	docs	GitHub 4.5.1
Key Vault - Keys	NuGet 4.5.0	docs	GitHub 4.5.0
Key Vault - Secrets	NuGet 4.5.0	docs	GitHub 4.5.0
Load Testing	NuGet 1.0.1	docs	GitHub 1.0.1

Name	Package	Docs	Source
Maps Common	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Maps Geolocation	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Maps Render	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Maps Route	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Maps Search	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Media Analytics Edge	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Metrics Advisor	NuGet 1.1.0	docs	GitHub 1.1.0
Mixed Reality Authentication	NuGet 1.2.0	docs	GitHub 1.2.0
Monitor Ingestion	NuGet 1.0.0	docs	GitHub 1.0.0
Monitor Query	NuGet 1.2.0	docs	GitHub 1.2.0
OpenAI Inference	NuGet 1.0.0-beta.5	docs	GitHub 1.0.0-beta.5
OpenTelemetry AspNetCore	NuGet 1.0.0-beta.4	docs	GitHub 1.0.0-beta.4
OpenTelemetry Exporter	NuGet 1.0.0-beta.12	docs	GitHub 1.0.0-beta.12

Name	Package	Docs	Source
Personalizer	NuGet 2.0.0-beta.2	docs	GitHub 2.0.0-beta.2
Purview Account	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Purview Administration	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Purview Catalog	NuGet 1.0.0-beta.4	docs	GitHub 1.0.0-beta.4
Purview Scanning	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Purview Share	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Purview Workflow	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Question Answering	NuGet 1.1.0	docs	GitHub 1.1.0
Schema Registry	NuGet 1.3.0 NuGet 1.4.0-beta.1	docs	GitHub 1.3.0 GitHub 1.4.0-beta.1
Schema Registry - Avro	NuGet 1.0.0	docs	GitHub 1.0.0
Service Bus	NuGet 7.15.0	docs	GitHub 7.15.0
Storage - Blobs	NuGet 12.16.0 NuGet 12.17.0-beta.1	docs	GitHub 12.16.0 GitHub 12.17.0-beta.1

Name	Package	Docs	Source
Storage - Blobs Batch	NuGet 12.13.0 NuGet 12.14.0-beta.1	docs	GitHub 12.13.0 GitHub 12.14.0-beta.1
Storage - Blobs ChangeFeed	NuGet 12.0.0-preview.35	docs	GitHub 12.0.0-preview.35
Storage - Files Data Lake	NuGet 12.14.0 NuGet 12.15.0-beta.1	docs	GitHub 12.14.0 GitHub 12.15.0-beta.1
Storage - Files Share	NuGet 12.14.0 NuGet 12.15.0-beta.1	docs	GitHub 12.14.0 GitHub 12.15.0-beta.1
Storage - Queues	NuGet 12.14.0 NuGet 12.15.0-beta.1	docs	GitHub 12.14.0 GitHub 12.15.0-beta.1
Synapse - AccessControl	NuGet 1.0.0-preview.5	docs	GitHub 1.0.0-preview.5
Synapse - Artifacts	NuGet 1.0.0-preview.17	docs	GitHub 1.0.0-preview.17
Synapse - Managed Private Endpoints	NuGet 1.0.0-beta.5	docs	GitHub 1.0.0-beta.5
Synapse - Monitoring	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Synapse - Spark	NuGet 1.0.0-preview.8	docs	GitHub 1.0.0-preview.8

Name	Package	Docs	Source
Tables	NuGet 12.8.0	docs	GitHub 12.8.0
Text Analytics	NuGet 5.3.0	docs	GitHub 5.3.0
Text Translation	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Video Analyzer Edge	NuGet 1.0.0-beta.6	docs	GitHub 1.0.0-beta.6
Vision Core	NuGet 0.10.0-beta.1		GitHub 0.10.0-beta.1
Web PubSub	NuGet 1.3.0	docs	GitHub 1.3.0
Web PubSub Client	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Azure client library integration for ASP.NET Core	NuGet 1.6.3	docs	GitHub 1.6.3
Blob Storage Key Store for .NET Data Protection	NuGet 1.3.2	docs	GitHub 1.3.2
CloudNative CloudEvents with Event Grid	NuGet 1.0.0	docs	GitHub 1.0.0
Core - Client - Spatial	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Core - Client - Spatial Newtonsoft Json	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1

Name	Package	Docs	Source
Functions extension for Azure Tables	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Key Encryptor for .NET Data Protection	NuGet 1.2.2	docs	GitHub 1.2.2
Secrets Configuration Provider for .NET	NuGet 1.2.2	docs	GitHub 1.2.2
Storage - Common	NuGet 12.15.0 NuGet 12.16.0-beta.1	docs	GitHub 12.15.0 GitHub 12.16.0-beta.1
WebJobs Extensions - Event Grid	NuGet 3.3.0	docs	GitHub 3.3.0
WebJobs Extensions - Event Hubs	NuGet 5.4.0	docs	GitHub 5.4.0
WebJobs Extensions - Service Bus	NuGet 5.11.0	docs	GitHub 5.11.0
WebJobs Extensions - SignalR Service	NuGet 1.11.0	docs	GitHub 1.11.0
WebJobs Extensions - Storage	NuGet 5.1.3	docs	GitHub 5.1.3
WebJobs Extensions - Web PubSub	NuGet 1.5.0	docs	GitHub 1.5.0
Cosmos DB for PostgreSQL	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
IotFirmwareDefense	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Resource Management - Advisor	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3

Name	Package	Docs	Source
Resource Management - Agrifood	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Alerts Management	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Analysis	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Api Management	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - App Compliance Automation	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - App Configuration	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1
Resource Management - App Platform	NuGet 1.0.2 NuGet 1.1.0-beta.2	docs	GitHub 1.0.2 GitHub 1.1.0-beta.2
Resource Management - App Service	NuGet 1.0.2 NuGet 1.1.0-beta.1	docs	GitHub 1.0.2 GitHub 1.1.0-beta.1
Resource Management - Application Insights	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3

Name	Package	Docs	Source
Resource Management - Arc ScVmm	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Attestation	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Authorization	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Automanage	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1
Resource Management - Automation	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Azure Stack HCI	NuGet 1.0.2	docs	GitHub 1.0.2
Resource Management - Azure VMware Solution	NuGet 1.1.1 NuGet 1.2.0-beta.1	docs	GitHub 1.1.1 GitHub 1.2.0-beta.1
Resource Management - Backup	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1
Resource Management - Batch	NuGet 1.2.0	docs	GitHub 1.2.0
Resource Management - Billing	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1

Name	Package	Docs	Source
Resource Management - Billing Benefits	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Resource Management - Blueprint	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Bot Service	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Change Analysis	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Chaos	NuGet 1.0.0-beta.4	docs	GitHub 1.0.0-beta.4
Resource Management - Cognitive Search	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Cognitive Services	NuGet 1.3.0	docs	GitHub 1.3.0
Resource Management - Communication	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Compute	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1

Name	Package	Docs	Source
Resource Management - Confidential Ledger	NuGet 1.0.1 NuGet 1.1.0-beta.2	docs	GitHub 1.0.1 GitHub 1.1.0-beta.2
Resource Management - Confluent	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Connected VMware vSphere	NuGet 1.0.0-beta.7	docs	GitHub 1.0.0-beta.7
Resource Management - Consumption	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Container Apps	NuGet 1.0.3 NuGet 1.1.0-beta.2	docs	GitHub 1.0.3 GitHub 1.1.0-beta.2
Resource Management - Container Instances	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Container Registry	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Container Service	NuGet 1.1.0 NuGet 1.2.0-beta.2	docs	GitHub 1.1.0 GitHub 1.2.0-beta.2

Name	Package	Docs	Source
Resource Management - Content Delivery Network	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Core	NuGet 1.6.0	docs	GitHub 1.6.0
Resource Management - Cosmos DB	NuGet 1.3.0 NuGet 1.4.0-beta.2	docs	GitHub 1.3.0 GitHub 1.4.0-beta.2
Resource Management - Costmanagement	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Resource Management - Customer Insights	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Data Box	NuGet 1.0.2	docs	GitHub 1.0.2
Resource Management - Data Box Edge	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Data Factory	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Resource Management - Data Lake Analytics	NuGet 1.0.0 NuGet 1.1.0-beta.2	docs	GitHub 1.0.0 GitHub 1.1.0-beta.2
Resource Management - Data Lake Store	NuGet 1.0.0 NuGet 1.1.0-beta.2	docs	GitHub 1.0.0 GitHub 1.1.0-beta.2

Name	Package	Docs	Source
Resource Management - Data Migration	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Data Protection	NuGet 1.1.2 NuGet 1.2.0-beta.1	docs	GitHub 1.1.2 GitHub 1.2.0-beta.1
Resource Management - Data Share	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Datadog	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Deployment Manager	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Desktop Virtualization	NuGet 1.1.0	docs	GitHub 1.1.0
Resource Management - Dev Center	NuGet 1.0.0	docs	GitHub 1.0.0
Resource Management - Dev Spaces	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Device Provisioning Services	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Device Update	NuGet 1.0.0-beta.9	docs	GitHub 1.0.0-beta.9
Resource Management - DevTest Labs	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1

Name	Package	Docs	Source
Resource Management - Digital Twins	NuGet 1.2.0 NuGet 1.3.0-beta.1	docs	GitHub 1.2.0 GitHub 1.3.0-beta.1
Resource Management - DNS	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - DNS Resolver	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Dynatrace	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Edge Order	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Elastic	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - ElasticSan	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Energyservices	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Resource Management - Event Grid	NuGet 1.0.1 NuGet 1.1.0-beta.2	docs	GitHub 1.0.1 GitHub 1.1.0-beta.2

Name	Package	Docs	Source
Resource Management - Event Hubs	NuGet 1.0.0 NuGet 1.1.0-beta.3	docs	GitHub 1.0.0 GitHub 1.1.0-beta.3
Resource Management - Extended Location	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Fluid Relay	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Front Door	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Graph Services	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Resource Management - Guest Configuration	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - HDInsight	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Health Bot	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1

Name	Package	Docs	Source
Resource Management - Healthcare APIs	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Hybrid Compute	NuGet 1.0.0-beta.4	docs	GitHub 1.0.0-beta.4
Resource Management - Hybrid Connectivity	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Hybrid Container Service	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Resource Management - Hybrid Kubernetes	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - IoT Central	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - IoT Hub	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Key Vault	NuGet 1.1.0 NuGet 1.2.0-beta.2	docs	GitHub 1.1.0 GitHub 1.2.0-beta.2
Resource Management - Kubernetes Configuration	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1

Name	Package	Docs	Source
Resource Management - Kusto	NuGet 1.3.0 NuGet 1.4.0-beta.1	docs	GitHub 1.3.0 GitHub 1.4.0-beta.1
Resource Management - Lab Services	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Load Testing	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Log Analytics	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Logic	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Machine Learning	NuGet 1.1.1 NuGet 1.2.0-beta.1	docs	GitHub 1.1.1 GitHub 1.2.0-beta.1
Resource Management - Machine Learning Compute	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Maintenance	NuGet 1.1.2 NuGet 1.2.0-beta.1	docs	GitHub 1.1.2 GitHub 1.2.0-beta.1

Name	Package	Docs	Source
Resource Management - Managed Grafana	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Managed Network	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Managed Service Identity	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Managed Services	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Managednetworkfabric	NuGet 1.0.0-beta.1		GitHub 1.0.0-beta.1
Resource Management - Management Partner	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Maps	NuGet 1.0.1 NuGet 1.1.0-beta.2	docs	GitHub 1.0.1 GitHub 1.1.0-beta.2
Resource Management - Marketplace	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Marketplace Ordering	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1

Name	Package	Docs	Source
Resource Management - Media	NuGet 1.2.0 NuGet 1.3.0-beta.1	docs	GitHub 1.2.0 GitHub 1.3.0-beta.1
Resource Management - Mixed Reality	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Mobile Network	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Resource Management - Monitor	NuGet 1.2.0 NuGet 1.3.0-beta.2	docs	GitHub 1.2.0 GitHub 1.3.0-beta.2
Resource Management - MySQL	NuGet 1.0.1 NuGet 1.1.0-beta.2	docs	GitHub 1.0.1 GitHub 1.1.0-beta.2
Resource Management - NetApp Files	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Network	NuGet 1.3.0	docs	GitHub 1.3.0
Resource Management - Network Function	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Networkcloud	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Resource Management - New Relic Observability	NuGet 1.0.0	docs	GitHub 1.0.0

Name	Package	Docs	Source
Resource Management - Nginx	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Resource Management - Notification Hubs	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Orbital	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Paloaltonetworks.Ngfw	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Resource Management - Peering	NuGet 1.1.1 NuGet 1.2.0-beta.1	docs	GitHub 1.1.1 GitHub 1.2.0-beta.1
Resource Management - Policy Insights	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - PostgreSQL	NuGet 1.1.1 NuGet 1.2.0-beta.1	docs	GitHub 1.1.1 GitHub 1.2.0-beta.1
Resource Management - Power BI Dedicated	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Private DNS	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1

Name	Package	Docs	Source
Resource Management - Provider Hub	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1
Resource Management - Purview	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Quantum	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Qumulo	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1
Resource Management - Quota	NuGet 1.0.0	docs	GitHub 1.0.0
Resource Management - Recovery Services	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1
Resource Management - Redis	NuGet 1.2.0	docs	GitHub 1.2.0
Resource Management - Redis Enterprise	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Redis Enterprise	NuGet 1.0.0-beta.1	docs	GitHub 1.0.0-beta.1
Resource Management - Relay	NuGet 1.1.1 NuGet 1.2.0-beta.1	docs	GitHub 1.1.1 GitHub 1.2.0-beta.1

Name	Package	Docs	Source
Resource Management - Reservations	NuGet 1.3.0 NuGet 1.4.0-beta.1	docs	GitHub 1.3.0 GitHub 1.4.0-beta.1
Resource Management - Resource Graph	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Resource Health	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1
Resource Management - Resource Mover	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Resources	NuGet 1.6.0	docs	GitHub 1.6.0
Resource Management - Security	NuGet 1.1.0 NuGet 1.2.0-beta.2	docs	GitHub 1.1.0 GitHub 1.2.0-beta.2
Resource Management - Security DevOps	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Security Insights	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Selfhelp	NuGet 1.0.0	docs	GitHub 1.0.0

Name	Package	Docs	Source
Resource Management - Service Bus	NuGet 1.0.0 NuGet 1.1.0-beta.3	docs	GitHub 1.0.0 GitHub 1.1.0-beta.3
Resource Management - Service Fabric	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Service Fabric Managed Clusters	NuGet 1.0.1 NuGet 1.1.0-beta.2	docs	GitHub 1.0.1 GitHub 1.1.0-beta.2
Resource Management - Service Linker	NuGet 1.0.2 NuGet 1.1.0-beta.1	docs	GitHub 1.0.2 GitHub 1.1.0-beta.1
Resource Management - Service Networking	NuGet 1.0.0-beta.2	docs	GitHub 1.0.0-beta.2
Resource Management - SignalR	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Site Recovery	NuGet 1.0.0	docs	GitHub 1.0.0
Resource Management - SQL	NuGet 1.2.0 NuGet 1.3.0-beta.1	docs	GitHub 1.2.0 GitHub 1.3.0-beta.1
Resource Management - SQL Virtual Machine	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1

Name	Package	Docs	Source
Resource Management - Storage	NuGet 1.1.1 NuGet 1.2.0-beta.1	docs	GitHub 1.1.1 GitHub 1.2.0-beta.1
Resource Management - Storage Cache	NuGet 1.1.0 NuGet 1.2.0-beta.1	docs	GitHub 1.1.0 GitHub 1.2.0-beta.1
Resource Management - Storage Mover	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Storage Pool	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Storage Sync	NuGet 1.1.1 NuGet 1.2.0-beta.1	docs	GitHub 1.1.1 GitHub 1.2.0-beta.1
Resource Management - Stream Analytics	NuGet 1.1.1 NuGet 1.2.0-beta.1	docs	GitHub 1.1.1 GitHub 1.2.0-beta.1
Resource Management - Subscriptions	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Support	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1

Name	Package	Docs	Source
Resource Management - Synapse	NuGet 1.1.1 NuGet 1.2.0-beta.1	docs	GitHub 1.1.1 GitHub 1.2.0-beta.1
Resource Management - Traffic Manager	NuGet 1.1.0	docs	GitHub 1.1.0
Resource Management - Voice Services	NuGet 1.0.0	docs	GitHub 1.0.0
Resource Management - Web PubSub	NuGet 1.0.1 NuGet 1.1.0-beta.1	docs	GitHub 1.0.1 GitHub 1.1.0-beta.1
Resource Management - Workload Monitor	NuGet 1.0.0-beta.3	docs	GitHub 1.0.0-beta.3
Resource Management - Workloads	NuGet 1.0.0 NuGet 1.1.0-beta.1	docs	GitHub 1.0.0 GitHub 1.1.0-beta.1
Azure.Communication.Administration	NuGet 1.0.0-beta.3		
Azure.Communication.Calling	NuGet 1.0.0-beta.36		
Azure.Communication.CallingServer	NuGet 1.0.0-beta.3		
Azure.Core.Expressions.DataFactory	NuGet 1.0.0-beta.3		
Data Movement - Blobs	NuGet 12.0.0-beta.2		
IoT Edge Function	NuGet 3.5.3		
IoT Models Repository	NuGet 1.0.0-preview.5		

Name	Package	Docs	Source
Microsoft.Azure.Devices.Authentication	NuGet 2.0.0-preview001		
Microsoft.Azure.Functions.Worker.Extensions.Kusto	NuGet 1.0.7-Preview		
Microsoft.Azure.IoT.Edge.Module	NuGet 3.2.0		
Microsoft.Azure.WebJobs.CosmosDb.ChangeProcessor	NuGet 1.0.4		
Microsoft.Azure.WebPubSub.AspNetCore	NuGet 1.0.0		
Microsoft.Azure.WebPubSub.Common	NuGet 1.2.0		
Quantum Jobs	NuGet 1.0.0-beta.3		
Time Series Insights	NuGet 1.0.0-beta.1		
Unknown Display Name	NuGet 1.0.0		
Unknown Display Name	NuGet 0.14.0-preview01		
Unknown Display Name	NuGet 0.14.0-preview01		
Unknown Display Name	NuGet 1.0.0-preview2		
Unknown Display Name	NuGet 0.3.1-preview		
Unknown Display Name	NuGet 1.5.0-beta.1		
Unknown Display Name	NuGet 1.0.0-beta.1		
Unknown Display Name	NuGet 1.0.0-beta.1		
Unknown Display Name	NuGet 1.0.2		

Name	Package	Docs	Source
Unknown Display Name	NuGet 1.0.2		
Unknown Display Name	NuGet 0.14.0-preview01		
Unknown Display Name	NuGet 0.3.1-preview		
Unknown Display Name	NuGet 0.15.0		
Unknown Display Name	NuGet 0.2.3		
Unknown Display Name	NuGet 0.3.4		
Anomaly Detector	NuGet 1.0.0		GitHub 1.0.0
App Configuration Provider	NuGet 5.2.0 NuGet 5.3.0-preview		GitHub 5.2.0
App Service	NuGet 0.2.2-alpha		
App Service - API Apps Common	NuGet 0.9.36		
App Service - API Apps Service	NuGet 0.9.64		
Application Insights	NuGet 0.9.0-preview		
Auto Suggest	NuGet 2.0.0		
Auto Suggest	NuGet 2.1.0-preview.1		GitHub 2.1.0-preview.1
AutoRest Common	NuGet 2.4.48		GitHub 2.4.48
Azure Stack - Azure Consistent Storage	NuGet 0.10.8-preview		

Name	Package	Docs	Source
Azure.FX	NuGet 0.0.22-alpha		
Azure.FX.Templates	NuGet 1.2.0		
Batch	NuGet 16.0.0		GitHub 16.0.0
Batch - Apps Cryptography	NuGet 1.1.1.4		
Batch - Conventions Files	NuGet 4.0.0		GitHub 4.0.0
Batch - File Staging	NuGet 9.0.0		GitHub 9.0.0
Client Runtime	NuGet 2.3.24		GitHub 2.3.24
Client Runtime - Azure	NuGet 3.3.19		GitHub 3.3.19
Client Runtime - Azure Authentication	NuGet 2.4.1		GitHub 2.4.1
Client Runtime - Azure Test Framework	NuGet 1.7.7		GitHub 1.7.7
Client Runtime - ETW	NuGet 2.1.3		GitHub 2.1.3
Client Runtime - Log4Net	NuGet 2.1.4		GitHub 2.1.4
Code Analyzers for Durable Functions	NuGet 0.5.0		GitHub 0.5.0
Commerce Usage Aggregates	NuGet 1.5.3		
Common	NuGet 2.2.1		
Common - Authentication	NuGet 1.7.0-preview		
Common - Dependencies	NuGet 1.0.0		

Name	Package	Docs	Source
Computer Vision	NuGet 7.0.1		GitHub 7.0.1
Content Moderator	NuGet 2.0.0		
Cosmos DB	NuGet 3.35.1	docs	GitHub 3.35.1
Custom Image Search	NuGet 2.1.0-preview.1		GitHub 2.1.0-preview.1
Custom Image Search	NuGet 2.0.0		
Custom Search	NuGet 2.1.0-preview.1		GitHub 2.1.0-preview.1
Custom Search	NuGet 2.0.0		
Custom Vision Prediction	NuGet 2.0.0		GitHub 2.0.0
Custom Vision Training	NuGet 2.0.0 NuGet 2.1.0-preview		GitHub 2.0.0
Data Lake Analytics	NuGet 1.4.211011		
Data Movement	NuGet 2.0.4		GitHub 2.0.4
DCAP	NuGet 1.12.0		GitHub 1.12.0
Digital Twins Client	NuGet 1.0.0-preview-001		
Digital Twins Service	NuGet 1.0.0-preview-001		

Name	Package	Docs	Source
Entity Search	NuGet 2.1.0-preview.1		GitHub 2.1.0-preview.1
Entity Search	NuGet 2.0.0		
Event Hubs - Service Fabric Processor	NuGet 0.5.4		GitHub 0.5.4
Face	NuGet 2.8.0-preview.3		GitHub 2.8.0-preview.3
Feature Management	NuGet 2.5.1	docs	
Functions - Extensions	NuGet 1.1.0		GitHub 1.1.0
Functions extension for Azure Mobile Apps	NuGet 3.0.0-beta8		GitHub 3.0.0-beta8
Functions extension for Blob Storage	NuGet 5.1.3	docs	
Functions extension for Cosmos DB	NuGet 4.3.0		GitHub 4.3.0
Functions extension for DocumentDB	NuGet 1.3.0		GitHub 1.3.0
Functions extension for Durable Task Framework	NuGet 2.9.6	docs	GitHub 2.9.6
Functions extension for HTTP	NuGet 3.2.0		GitHub 3.2.0
Functions extension for IoT Edge	NuGet 1.0.7		GitHub 1.0.7
Functions extension for Kafka	NuGet 3.8.0		GitHub 3.8.0
Functions extension for Notification Hubs	NuGet 1.3.0		GitHub 1.3.0
Functions extension for RabbitMQ	NuGet 2.0.3		GitHub 2.0.3

Name	Package	Docs	Source
Functions extension for script abstractions	NuGet 1.0.0-preview		
Functions extension for SendGrid	NuGet 3.0.3		GitHub 3.0.3
Functions extension for Sources	NuGet 3.0.37		GitHub 3.0.37
Functions extension for Storage Queues	NuGet 5.1.3	docs	
Functions extension for Twilio	NuGet 3.0.2		GitHub 3.0.2
Functions extension metadata generator	NuGet 4.0.1		GitHub 4.0.1
Functions item template pack for Microsoft Template Engine	NuGet 4.0.2529		GitHub 4.0.2529
Functions OpenAPI app settings deserialization library	NuGet 1.4.0 NuGet 2.0.0-preview2		
Functions OpenAPI document and Swagger UI renderer library	NuGet 1.4.0 NuGet 2.0.0-preview2		
Functions project template pack for Microsoft Template Engine	NuGet 4.0.2529		GitHub 4.0.2529
Functions runtime assemblies for App Insights logging	NuGet 3.0.35		GitHub 3.0.35
Functions runtime assemblies for logging	NuGet 4.0.3		GitHub 4.0.3
Functions runtime assemblies for Microsoft.Azure.WebJobs.Host	NuGet 3.0.37		GitHub 3.0.37
Functions timers and file triggers	NuGet 5.0.0		GitHub 5.0.0
Gallery	NuGet 2.6.2-preview		

Name	Package	Docs	Source
Hadoop Client	NuGet 1.5.13		
HDInsight - Job	NuGet 3.0.0-preview.3		GitHub 3.0.0-preview.3
Image Search	NuGet 2.1.0-preview.1		GitHub 2.1.0-preview.1
Image Search	NuGet 2.0.0		
Information Protection	NuGet 1.7.147		
IoT Plug and Play - Devices Client	NuGet 1.41.2 NuGet 2.0.0-preview005		
Kinect Developer Kit	NuGet 1.0.1		
Kinect Developer Kit	NuGet 1.4.1		GitHub 1.4.1
Kusto Data	NuGet 9.3.1	docs	GitHub 9.3.1
Kusto Ingest	NuGet 9.3.1	docs	GitHub 9.3.1
Local Search	NuGet 1.0.0-preview.1		GitHub 1.0.0-preview.1
Local Search	NuGet 0.9.0-preview		
LUIS Authoring	NuGet 3.1.0 NuGet 3.2.0-preview.5		GitHub 3.1.0
LUIS Runtime	NuGet 3.1.0-preview.1		GitHub 3.1.0-preview.1

Name	Package	Docs	Source
Media Live Video Analytics Edge	NuGet 1.0.4-preview.1		GitHub 1.0.4-preview.1
Microsoft.Azure.Amqp	NuGet 2.6.2		
Microsoft.Azure.Devices	NuGet 1.38.1 NuGet 2.0.0-preview005		
Microsoft.Azure.Devices.Client.PCL	NuGet 1.0.16		
Microsoft.Azure.Devices.ProtocolGateway.Core	NuGet 2.0.1		
Microsoft.Azure.Devices.ProtocolGateway.IotHubClient	NuGet 2.0.1		
Microsoft.Azure.Devices.ProtocolGateway.Providers.CloudStorage	NuGet 2.0.1		
Microsoft.Azure.Devices.Provisioning.Client	NuGet 1.19.1 NuGet 2.0.0-preview005		
Microsoft.Azure.Devices.Provisioning.Security.Tpm	NuGet 1.14.1 NuGet 1.15.0-preview-001		
Microsoft.Azure.Devices.Provisioning.Service	NuGet 1.18.1 NuGet 2.0.0-preview005		
Microsoft.Azure.Devices.Provisioning.Transport.Amqp	NuGet 1.16.1 NuGet 1.17.0-preview-001		

Name	Package	Docs	Source
Microsoft.Azure.Devices.Provisioning.Transport.Http	NuGet 1.15.1 NuGet 1.16.0-preview-001		
Microsoft.Azure.Devices.Provisioning.Transport.Mqtt	NuGet 1.17.1 NuGet 1.18.0-preview-001		
Microsoft.Azure.Devices.Shared	NuGet 1.30.1 NuGet 1.31.0-preview-001		
Microsoft.Azure.Functions.Worker	NuGet 1.16.0		
Microsoft.Azure.Functions.Worker.Sdk	NuGet 1.11.0		
Microsoft.Azure.uamqp	NuGet 1.2.11		
Microsoft.Azure.umqtt	NuGet 1.1.11		
Mobile Apps	NuGet 2.0.3		
Mobile Server - Cross Domain	NuGet 2.0.3		
Mobile Service - Resource Broker	NuGet 1.0.2.1		
News Search	NuGet 2.1.0-preview.1		GitHub 2.1.0-preview.1
News Search	NuGet 2.0.0		
Notification Hubs	NuGet 4.1.0		GitHub 4.1.0

Name	Package	Docs	Source
Personalizer	NuGet 1.0.0		GitHub 1.0.0
Power BI	NuGet 3.20.1		GitHub 3.20.1
Relay	NuGet 3.0.1	docs	GitHub 3.0.1
Search - Common	NuGet 10.1.0		GitHub 10.1.0
Search - Data	NuGet 10.1.0		GitHub 10.1.0
Search - Service	NuGet 10.1.0		GitHub 10.1.0
Service Bus	NuGet 6.2.2		
SignalR	NuGet 1.21.4		GitHub 1.21.4
SignalR - ASP.NET	NuGet 1.21.4		GitHub 1.21.4
SignalR - Benchmark	NuGet 1.0.0-preview1-10415		GitHub 1.0.0-preview1-10415
SignalR - Protocols	NuGet 1.21.4		GitHub 1.21.4
SignalR - Serverless Protocols	NuGet 1.9.0		GitHub 1.9.0
SignalR Management	NuGet 1.21.4		GitHub 1.21.4
Speech	NuGet 1.29.0		
Speech Remote Conversation	NuGet 1.29.0		
Speech Xamarin iOS	NuGet 1.25.0		

Name	Package	Docs	Source
Spell Check	NuGet 4.1.0-preview.1		GitHub 4.1.0-preview.1
Spring Cloud Client	NuGet 2.0.0-preview.3		
Storage - Files Data Lake	NuGet 2.0.0-alpha	docs	GitHub 2.0.0-alpha
Storage APIs for Microsoft.Azure.WebJobs.Host	NuGet 5.0.0		GitHub 5.0.0
Supporting library for Microsoft.Azure.WebJobs	NuGet 3.0.37		GitHub 3.0.37
Supporting library for Microsoft.Azure.WebJobs.Extensions.OpenApi	NuGet 1.4.0 NuGet 2.0.0-preview2		
Supporting library for testing Microsoft.Azure.WebJobs.Host	NuGet 3.0.37		GitHub 3.0.37
Synapse Analytics	NuGet 0.1.0-preview		GitHub 0.1.0-preview
Tables	NuGet 2.1.2		
Video Search	NuGet 2.1.0-preview.1		GitHub 2.1.0-preview.1
Video Search	NuGet 2.0.0		
Vision Content Moderator	NuGet 2.1.0-preview.1		GitHub 2.1.0-preview.1
Visual Search	NuGet 2.1.0-preview.1		GitHub 2.1.0-preview.1
Visual Search	NuGet 2.0.0		

Name	Package	Docs	Source
Web - Redis Output Cache Provider	NuGet 4.0.0		GitHub 4.0.0
Web - Redis Session State Provider	NuGet 5.0.0		GitHub 5.0.0
Web Search	NuGet 2.1.0-preview.1		GitHub 2.1.0-preview.1
Web Search	NuGet 2.0.0		
WebSites - DataProtection	NuGet 0.1.78-alpha		
WindowsAzure Common	NuGet 1.4.1		
WindowsAzure Common - Dependencies	NuGet 1.1.1		
WindowsAzure Common - Tracing ETW	NuGet 1.0.0		
WindowsAzure Common - Tracing Log4Net	NuGet 1.0.0		
WindowsAzure Media Services	NuGet 4.2.0	docs	GitHub 4.2.0
Core - Client - Newtonsoft Json	NuGet 1.0.0		
Cosmos DB - BulkExecutor	NuGet 2.5.1-preview		GitHub 2.5.1-preview
Cosmos DB - Direct	NuGet 3.31.3		GitHub 3.31.3
Cosmos DB - Encryption	NuGet 2.0.2		GitHub 2.0.2
Cosmos DB - Encryption	NuGet 1.0.0-preview05		
Extensions - Caching Cosmos	NuGet 1.5.0		GitHub 1.5.0

Name	Package	Docs	Source
Functions extension for Application Insights	NuGet 1.0.0-preview4		
Functions extension for Authentication Events	NuGet 1.0.0-beta.3		
Functions extension for Azure SQL and SQL Server	NuGet 3.0.253-preview		
Functions extension for Durable Task Framework - isolated worker	NuGet 1.0.2		
Functions extension for Storage Timers	NuGet 1.0.0		
Microsoft.Azure.Cosmos.Templates	NuGet 1.0.0		
Microsoft.Azure.Functions.Analyzers	NuGet 1.0.0		
Microsoft.Azure.Functions.Authentication.WebAssembly	NuGet 1.0.0 NuGet 1.0.1-preview		
Microsoft.Azure.Functions.Worker.ApplicationInsights	NuGet 1.0.0-preview4		
Microsoft.Azure.Functions.Worker.Core	NuGet 1.13.0		
Microsoft.Azure.Functions.Worker.Extensions.Abstractions	NuGet 1.2.0		
Microsoft.Azure.Functions.Worker.Extensions.ApplicationInsights	NuGet 1.0.0-preview4		
Microsoft.Azure.Functions.Worker.Extensions.CosmosDB	NuGet 4.0.1 NuGet 4.3.1-preview1		
Microsoft.Azure.Functions.Worker.Extensions.EventGrid	NuGet 3.2.1 NuGet 3.3.0-preview1		

Name	Package	Docs	Source
Microsoft.Azure.Functions.Worker.Extensions.EventHubs	NuGet 5.4.0 NuGet 5.4.1-preview1		
Microsoft.Azure.Functions.Worker.Extensions.Http	NuGet 3.0.13		
Microsoft.Azure.Functions.Worker.Extensions.Kafka	NuGet 3.8.0		
Microsoft.Azure.Functions.Worker.Extensions.OpenApi	NuGet 1.4.0 NuGet 2.0.0-preview2		
Microsoft.Azure.Functions.Worker.Extensions.RabbitMQ	NuGet 2.0.3		
Microsoft.Azure.Functions.Worker.Extensions.SendGrid	NuGet 3.0.3		
Microsoft.Azure.Functions.Worker.Extensions.ServiceBus	NuGet 5.11.0		
Microsoft.Azure.Functions.Worker.Extensions.SignalRService	NuGet 1.10.0		
Microsoft.Azure.Functions.Worker.Extensions.Sql	NuGet 3.0.181-preview		
Microsoft.Azure.Functions.Worker.Extensions.Storage	NuGet 5.1.2 NuGet 5.1.3-preview1		
Microsoft.Azure.Functions.Worker.Extensions.Storage.Blobs	NuGet 5.1.2 NuGet 5.1.3-preview1		
Microsoft.Azure.Functions.Worker.Extensions.Storage.Queues	NuGet 5.1.2 NuGet 5.1.3-preview1		
Microsoft.Azure.Functions.Worker.Extensions.Storage.Tables	NuGet 1.0.0-preview1		

Name	Package	Docs	Source
Microsoft.Azure.Functions.Worker.Extensions.Tables	NuGet 1.0.0 NuGet 1.2.0-preview1		
Microsoft.Azure.Functions.Worker.Extensions.Timer	NuGet 4.2.0		
Microsoft.Azure.Functions.Worker.Extensions.Warmup	NuGet 4.0.2		
Microsoft.Azure.Functions.Worker.Grpc	NuGet 1.11.0		
Microsoft.Azure.Functions.Worker.ItemTemplates	NuGet 4.0.2529		
Microsoft.Azure.Functions.Worker.ProjectTemplates	NuGet 4.0.2529		
Microsoft.Azure.Functions.Worker.Sdk.Analyzers	NuGet 1.1.2		
Microsoft.Azure.Functions.Worker.Sdk.Generators	NuGet 1.1.0-preview4		
Microsoft.Azure.WebJobs.CosmosDb.Mongo	NuGet 1.0.4		
Microsoft.Azure.WebJobs.Extensions.Kusto	NuGet 1.0.7-Preview		
Service Bus - Message ID plugin	NuGet 2.0.0		
SQL Database Elastic Scale Client	NuGet 2.3.0		GitHub 2.3.0
SQL Database Elastic Scale Service SplitMerge	NuGet 1.2.0		
SQL Database Jobs	NuGet 0.8.3362.1		
Hyak Common	NuGet 1.2.2		
Hyak Common - Tracing Etw	NuGet 1.0.2		

Name	Package	Docs	Source
Hyak Common - Tracing Log4Net	NuGet 1.0.2		
Microsoft.Azure.SignalR.Emulator	NuGet 1.1.0 NuGet 1.0.0-preview1-10809		
Template	NuGet 1.0.2-preview1		
Test HttpRecorder	NuGet 1.13.3		GitHub 1.13.3

Reference

Reference

Services

Alerts Management
Analysis
API Management
App Configuration
App Platform
App Service
Application Insights
Attestation
Authorization
Auto Suggest
Automanage
Automation
Azure VMware Solution
Batch
Billing
Bot Service
Change Analysis
Chaos
Cognitive Services
Communication
Compute
Confidential Ledger
Confluent
Connected VMware vSphere
Consumption
Container Apps
Container Instances
Container Registry
Container Service
Container Service Fleet
Content Delivery Network
Core
Cosmos DB
Cosmos DB for PostgreSQL
Cost Management
Data Box
Data Box Edge
Data Factory
Data Lake Analytics
Data Lake Store
Data Protection
Data Share
Defender EASM
Desktop Virtualization
Dev Center
DevTest Labs
DNS
DNS Resolver
Dynatrace
Edge Order
Elasticsan
Event Grid
Event Hubs
Extended Location
FarmBeats
Fluid Relay
Front Door
Grafana
Graph Services
Guest Configuration
HDInsight
Health Bot
Healthcare APIs
Hybrid Container Service
IoT
Key Vault
Kubernetes Configuration
Kusto
Lab Services
Load Testing
Logic Apps
Machine Learning
Maintenance
Managed Network Fabric
Managed Service Identity
Managed Services
Maps
Marketplace
Marketplace Ordering
Media
Metrics Advisor
Mixed Reality
Mobile Network
Mobile Services
Monitor
MySQL
NetApp Files
Network
Network Analytics
New Relic Observability
Nginx
Notification Hubs
Operator Nexus - Network Cloud
Orbital
Palo Alto Networks
Peering
Policy Insights

PostgreSQL
Private DNS
Provider Hub
Purview
Qumulo
Quota
Recovery Services
Redis
Relay
Reservations
Resource Graph
Resource Health
Resource Manager
Resource Mover
Schema Registry
Search
Security
Security Insights
Self Help
Service Bus
Service Fabric
Service Linker
Service Networking
SignalR
Sphere
SQL
SQL Virtual Machine
Stack HCI
Storage
Stream Analytics
Subscriptions
Support
Synapse
Tables
Traffic Manager
unknown
Voice Services
Web PubSub
Workloads
Other