



**Build Enterprise and Cloud
Applications with
Microsoft Azure**

Build Enterprise and Cloud Applications with Microsoft Azure

© 2024 Aptech Limited

All rights reserved.

No part of this book may be reproduced or copied in any form or by any means – graphic, electronic or mechanical, including photocopying, recording, taping, or storing in information retrieval system or sent or transferred without the prior written permission of copyright owner Aptech Limited.

All trademarks acknowledged.

APTECH LIMITED

Contact E-mail: ov-support@onlinevarsity.com

First Edition - 2024



Onlinevarsity



Preface

The book **Build Enterprise and Cloud Applications with Microsoft Azure** is a detailed guide for enterprise application development on the Azure cloud platform. It begins with an introduction to Azure Cloud Computing, then progresses through Microsoft Azure Cloud Services, management, and monitoring tools. The book dives into practical aspects such as using Entity Framework, managing data, and implementing Web API and RESTful APIs for enterprise solutions. Security, hosting, and consumption of Web APIs, alongside WCF Services, are extensively explored. It highlights the deployment of serverless Web services, API management, and the importance of Continuous Integration and Continuous Delivery (CI/CD). The book concludes with strategies for traffic management and deploying applications using Azure App Service Apps, all through the lens of .NET technologies and tools such as Visual Studio 2022.

This book is the result of a concentrated effort of the Design Team, which is continuously striving to bring you the best and the latest in Information Technology. The process of design has been a part of the ISO 9001 certification for Aptech-IT Division, Education Support Services. As part of Aptech's quality drive, this team does intensive research and curriculum enrichment to keep it in line with industry trends.

We will be glad to receive your suggestions.

DesignTeam



**MANY
COURSES
ONE
PLATFORM**

Table of Contents

Sessions

- Session 1: Introduction to Azure Cloud Computing**
- Session 2: Understanding Microsoft Azure Cloud Services**
- Session 3: Azure Management Tools**
- Session 4: Azure Monitoring**
- Session 5: Data Access Mechanism in Azure**
- Session 6: Working with Data in Azure Applications**
- Session 7: Design and Implementation of Web API**
- Session 8: APIs and RESTful APIs for Enterprise Applications**
- Session 9: Azure Security Features**
- Session 10: Hosting and Consuming Web API**
- Session 11: Introduction to WCF Services**
- Session 12: Implementing Azure Serverless Web Services and API Management**
- Session 13: CI/CD**
- Session 14: Implementing Traffic Management and Monitoring Strategies for Web Services**
- Session 15: Deploying Web Applications and Azure App Service Apps**



Onlinevarsity App for Android devices

Download from **Google Play Store**



SESSION 01

INTRODUCTION TO AZURE CLOUD COMPUTING

Overview

This session provides an introduction to cloud computing with its benefits. It discusses distributed applications and the concept of service-oriented architecture. It then provides an introduction to Web services as well as Microsoft Azure cloud services.

Learning Objectives

In this session, students will learn to:

- Explain cloud computing
- Explain Microsoft Azure cloud services
- Define features of Azure cloud computing

1.1 Overview of Cloud Services and Cloud Computing

Cloud computing is an approach that enables convenient and on-demand access through the Internet to computing capabilities and resources. By using cloud computing, computing services such as databases, software, storage, servers, and networking can be delivered on the Internet, in other words over 'the cloud'. Such computing services are provided by companies, which are known as cloud providers. In general, these companies charge for the services depending on usage. Services provided by these companies through the Internet are called cloud services.

Figure 1.1 represents the idea of cloud computing graphically.



Figure 1.1: Cloud Computing

In cloud computing, data for the applications that are served over the Internet reside on servers hosted across different data centers, which are spread over multiple geographical locations. In other words, the consumer of the applications and services does not host them locally but uses them over the Internet from a service provider. A data center is a dedicated storage space that is used to store computers and various related components.

Consider an example where a company has to deploy a custom application for internal users, such as employees and contract staff. Currently, there are about 10,000 users who will be accessing the application. Over the years, the company expands and hires another 50,000 users. Overall, there is an impact on the application performance as it is unable to cater to large number of users because of hardware limitations. The company has not upgraded the hardware for many years. The company then, researches some workarounds and solutions for this without incurring heavy expenditures.

Based on their research and findings, the company deploys the same application on the Internet in a cloud environment. Now, the IT department does not have to worry about the application performance being affected by hardware limitations, as there will not be any.

This is one of the key advantages of cloud computing. An end-user is unaware of the deployment, as the process of accessing the application remains the same. The hardware requirements for the application can automatically be scaled up and down depending on the usage of the application in the cloud environment. The IT department does not have to worry about application failure due to

hardware limitations.

The cloud computing approach helps small, medium, and big enterprises to build, deploy, and manage any kind of service from creating a small and simple Website to working with large server workloads. Figure 1.2 shows how various devices can be used with cloud computing.

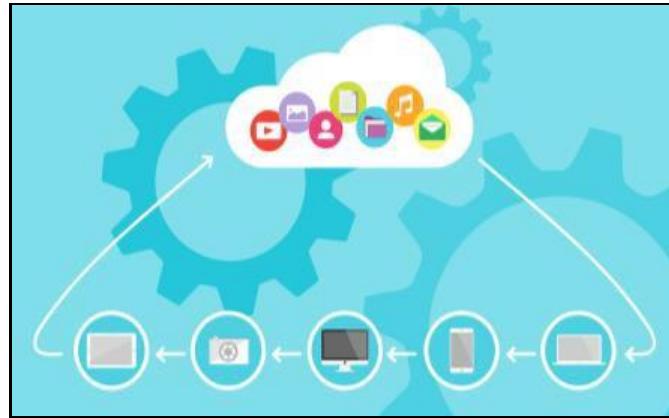


Figure 1.2: Cloud Computing

An example of a commonly used cloud computing application can be a social networking application, such as Twitter. The application is run from the servers hosted on the cloud and the data is also maintained on the cloud servers. Other such applications are Facebook, Gmail, Flickr, and Dropbox.

1.1.1 Why Use Cloud Computing?

The word ‘cloud’ is used as a representation of the Internet because a pictorial representation of the Internet has always been the cloud. In the programming world, ‘cloud’ refers to **cloud computing**, which is derived from the concept of utility computing. Utility computing is a concept that allows users to use-and-pay computing resources. Over the last few decades, this concept of using and paying resources on a metered basis gave birth to the concept of cloud computing.

There are three core reasons that have driven organizations to use cloud computing, as follows:

Economical

- More economical and cheaper than hosting applications on local infrastructure in an organization
- To host an application, an organization's IT department must procure hardware, which comes with a reasonable cost

- When hardware becomes obsolete, organizations must purchase new hardware
- Cloud computing, on the other hand, minimizes purchase of new hardware because infrastructure is managed by service providers, such as Amazon and Microsoft

Scalability

- Offers unlimited scalability
- If a cloud-hosted application requires more hardware, such as storage, the IT department of an organization can scale it within a few minutes
- Similarly, if hardware requires to be scaled down, it can be done without impacting the application or causing the down time

Deployment

- Offers quick deployment
- In the local environment, when an application is developed, it must be tested and then, deployed
- Both testing and production environments may differ in terms of hardware and resources
- This mismatch can cause application deployment failure in the production environment
- Cloud computing allows developers to use similar hardware resources so that deployment does not fail

Benefits:

Most modern cloud computing platforms including Microsoft Azure have following distinct characteristics:

Access to infinite resources

- Cloud computing platforms endow end users with an illusion of providing infinite capacity of computing and storage of resources. End users do not have to plan much about storage, usage of computing, or infrastructural resources.
- Once a company or organization deploys its own storage on a cloud computing platform, it can leverage large datacenters that can span across the globe.

Scale on demand

- In the past, users had to wait for resources to be synchronized with servers and their hardware. Cloud computing platforms allow adding resources only when required, which is, when there is a demand for it. With this characteristic, cost can be economized as well as the time to acquire.

Pay-for-play

- Core concept of cloud computing is about paying to use computing resources. This means all initial investment, setup fees, and reservation, are minimized and only the software and hardware fees are incurred.
- Thus, cloud computing platforms allow reducing up front Capital Expenditure (CapEx) costs and incur only the **Operating Expenditure (OpEx) cost**. Hence, user's expenditure is only for what they use.

High availability and agreements

- Most cloud computing platform providers sign a Service Level Agreement (SLA) for storage and other computing outages. By signing SLA, platform providers will guarantee a set level of uptime and if they cannot meet the SLA, they will then provide a refund. The SLA that Microsoft Azure platform provides takes care of both its hosting and its storage.

Geographically distributed data centers

- This characteristic of cloud computing spreads the datacenter globally in different geographies. When data centers are geographically distributed, users can make maximum use of load balancing, network latency, edge caching, as well as take care of all legal or regulatory hassles. Network latency refers to a delay that happens in data communication over a network. Edge caching involves distributing content from a local Web server to a caching server nearer to the user.

1.1.2 Cloud Services Models

Cloud computing services are categorized into three different categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

- **Infrastructure-as-a-Service (IaaS)** - This is the simplest type of cloud computing service. Using IaaS, a user can rent IT infrastructure from a cloud provider. Some examples of such infrastructure are networks, storage, servers, Virtual Machines (VMs), and operating systems. The cloud provider would charge the user on a pay-as-you-go basis. Figure 1.3 represents a simple IaaS architecture. IaaS provides hassle-free hardware resources without having a physical data center, for example, it includes network-attached storage, virtual machines, and load-balancer settings. Amazon is

a popular IaaS provider with services such as Elastic Cloud 2 (EC2) and S3.

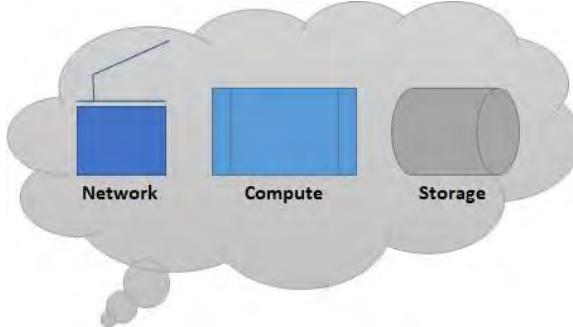


Figure 1.3: Infrastructure as a Service

- **Platform-as-a-Service (PaaS)** – In this type of service, there is an on-demand environment to develop, test, deliver, and manage software applications. Such an environment makes it simpler for users to promptly create Web apps or mobile apps. There is no burden on users to set up or manage the core infrastructure of the network, storage, servers, or databases required for the development.

Microsoft Azure is a popular example of a PaaS product. Microsoft Azure is a cloud offering by Microsoft and is used for building, deploying, and managing services and applications. Figure 1.4 represents commonly used services and products in Azure PaaS. Organizations typically use PaaS for development framework, analytics or business intelligence, security, and scheduling.



Figure 1.4: Azure Platform as a Service

Software-as-a-Service (SaaS) – In this type of model, the software subscription is leased to the consumer. For example, Microsoft provides Office 365 as a SaaS model. A popular SaaS service is Gmail where an e-mail client is offered as an Internet service without requiring any local installation. Anyone, anywhere, can use Gmail, provided he/she has an account.

1.2 Microsoft Azure Cloud Services

On October 27, 2008, a cloud computing platform created by Microsoft was first unveiled at the Professional Developers Conference. It was then called as Windows Azure. However, Windows Azure became commercially available only from 2010 onwards.

In its initial launch, the Azure platform had following components:

- Cloud OS
- SQL Azure
- Microsoft .NET Services

Over the years, it has undergone an overhaul and includes a variety of components now. It was renamed from Windows Azure to Microsoft Azure in 2014.

1.2.1 Microsoft Azure – Definition

Microsoft Azure is used for building, deploying, and managing services and applications. It uses a network of data centers connected globally to accomplish such tasks. It provides PaaS as well as IaaS services and supports various frameworks, programming languages, and tools which are Microsoft-specific as well as third-party software.

1.2.2 Requirement for Microsoft Azure

The requirement for Microsoft Azure arose because while there were a variety of SaaS products, there were hardly any PaaS products that could be beneficial for companies. Microsoft Azure simplifies IT-based management and reduces upfront and regular expenses. It is used to prepare, allocate, and upgrade Web applications instead of using expensive on-site resources.

Microsoft Azure OS works as an integral part of the Azure Services Platform that covers different and separate applications, storage, desktop environment, security, and so on. This platform also supports Microsoft standards, programming languages, platforms, and protocols.

1.2.3 Azure Platform and Cloud OS

Microsoft Azure was conceptualized as part of the cloud OS. The Microsoft Azure cloud computing platform is built as an integrated platform that can cater to the building, deployment, and management of applications. This open and flexible cloud platform can also cater to workloads hosted on a global network of Microsoft-managed datacenters.

Microsoft Azure platform architecture includes several components, of which following are the most important:

- **Compute:** provides a very large-scale hosting and processing environment for applications. Cloud services, Websites, Mobile Services, and so on.
- **Data Services:** focuses on scalable storage services such as blobs, queues, and tables. It includes SQL Database and Microsoft Azure storage.
- **App Services:** provides a variety of services such as authentication, service bus, caching, and so on.

1.2.4 Different Features of Azure

Table 1.1 provides a list of features of Microsoft Azure.

Feature	Description
Service Hosting	Allows creating own server-side applications, such as Websites, computation services, and hosting them using Microsoft Azure. Note that in its current version, a code that requires administrative privileges on the machine will not be supported by Azure.
Service Management	Allows using the Microsoft Azure in-built fabric controller that helps in dealing with application monitoring and management. Microsoft Azure automatically monitors and maintains software and hardware-related upgrades or failures.
Storage	Allows a unique storage provisioning and management service. It offers volume management built natively for containers. With the latest update, you can deploy Azure Container Storage on Azure Linux container host.
Windows Virtual Desktop	Allows you to get the best virtual desktop with the full Windows 10 and an optimized Office 365 ProPlus experience. It enables you to stream the full desktop or apps to any device. With this, you can deploy and scale your Windows desktops and apps on Azure in minutes.
Development Tools	Allows using many of its in-built development tools, such as Application Programming Interfaces (APIs) for logging and reporting errors, tools for deploying applications to cloud simulators, and other tools to read and update service configuration files.

Table 1.1: Features of Microsoft Azure

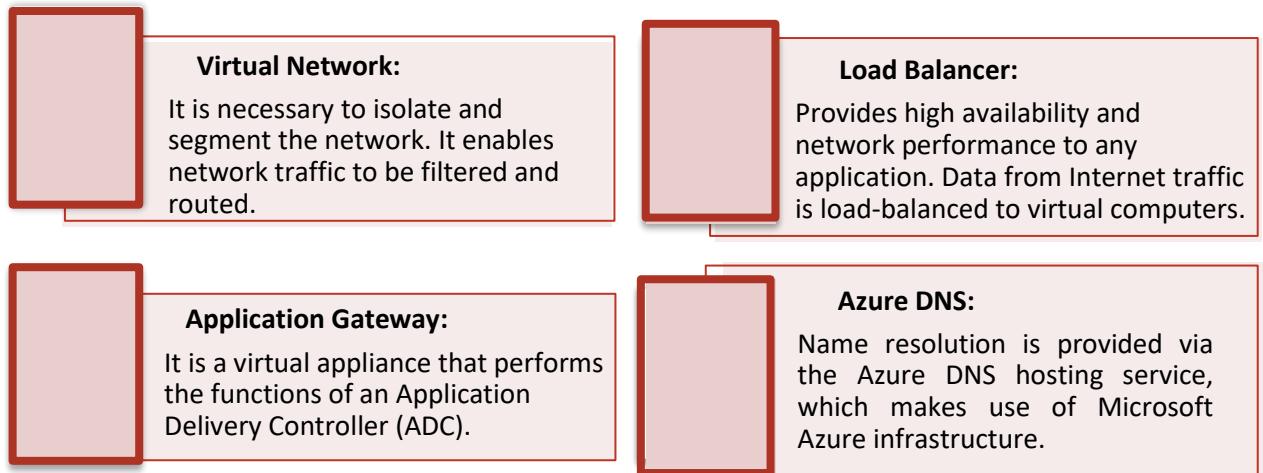
1.2.5 Compute

It offers computing services such as app hosting, development, and deployment through the Azure Platform. It consists of following components:

- Virtual Machine: Deploy any language, any workload on any operating system.
- Virtual Machine Scale Sets: Create thousands of similar virtual machines in a matter of seconds.
- Azure Container Registry: This service manages and stores container images for all Azure deployments.
- Azure Container Service: Create an Azure-friendly container hosting solution. To grow and organize applications, **Kube**, **DC/OS**, **Swarm**, or **Docker** are employed.
- Batch: Batch processing makes scaling to tens, hundreds, or thousands of virtual computers and running computer pipelines easier.
- Service Fabric: Simplify the design of microservice-based applications and their lifecycle management. All of the major programming languages are supported, including Java, PHP, Node.js, Python, and Ruby.

1.2.6 Networking

Azure Networking is a communication protocol for connecting multiple resources via the Internet. Microsoft provides various services and tools under Azure that make your network strong and easy to manage.



1.2.7 Storage

Azure Store is a cloud storage alternative for modern applications. It was designed to meet the scalability requirements of its users. It has the capacity to store and process hundreds of terabytes of data. It is made up of following components:

- Blob Storage: Azure Blob storage is a cloud-based service for storing unstructured data as objects/blobs. It is possible to store any text or binary

- data, such as a document, a media file, or an application installation.
- Queue Storage: It enables cloud-based communication between application components. It uses asynchronous messaging to communicate between application components.
- File Storage: Legacy apps can be moved to Azure File Storage. To connect to Azure quickly and without the requirement for costly rewrites, it relies on file sharing.
- Table Storage: Semi-structured NoSQL data is stored in Azure Table Storage in the cloud. It stores keys and attributes without regard to their schema.

1.2.8 Web and Mobile Services

Following are Web and Mobile Services offered by Microsoft Azure:

- Web Apps: Web Apps allow you to design and host Websites in your preferred programming language without having to worry about the infrastructure.
- Mobile Apps: Consumers can use the Mobile Apps Service to design mobile apps that are extremely scalable and available from anywhere in the world.
- API Apps: API apps make developing, hosting, and using APIs in the cloud and on-premises much easier.

1.2.9 Containers

Containers are a sort of OS virtualization. From a small microservice or software process to a huge application, a single container may handle it all. All of the essential executables, binary code, libraries, and configuration files are stored in a container. Containers, unlike server and machine virtualization, do not include operating system images. As a result, they are more portable, light, and have lower overhead. Several containers can be deployed as one or more container clusters in bigger application deployments.

Containers make it easier to build, test, deploy, and redeploy software in a range of environments, ranging from a developer's laptop to an on-premises data center and even the cloud. Containers have a number of benefits, including:

- The overhead is reduced
- Better portability
- An operation with a higher level of consistency
- Increased effectiveness
- App development has improved

1.2.10 Database

SQL and NoSQL technologies are included in the Database as a Service (DBaaS) category. Azure Cosmos DB and Azure Database for PostgreSQL are among the databases available. It is made up of following components:

- SQL Database: It is a Microsoft cloud-based relational database service based on Microsoft SQL Server, the industry's most popular engine.
- Document DB: It is a fully managed NoSQL database service that is built for speed, predictability, and ease of use.
- Redis Cache: It is a sophisticated key-value storage system that is both secure and reliable. Strings, hashes, lists, and other data structures are stored here.

1.2.11 Data + Analytics

Microsoft Azure has a number of services for analyzing data. One of the most efficient approaches is to store data in Azure Data Lake Storage Gen2 and then, process it using Spark on Azure Databricks.

Microsoft's Azure Stream Analytics (ASA) solution provides real-time data analytics. Stock trading analysis, fraud detection, embedded sensor analysis, and online clickstream analytics are just a few examples. The Stream Analytics Query Language, a variation of T-SQL, is used by ASA. As a result, anyone who knows SQL will find it quite simple to learn how to develop tasks for Stream Analytics.

Data transformation applications may be built using Azure Data Lake Analytics using a variety of languages, including Python, R, .NET, and U-SQL.

Note: U-SQL is the Microsoft big data query language of the Azure Data Lake Analytics service.

Data Lake Analytics is ideal for handling petabytes of data. Data Lake Analytics links to Azure-based data sources, such as Azure Data Lake Storage and does real-time analytics depending on your code's specifications.

1.2.12 AI + Cognitive Services

The ability of a machine to replicate intelligent human behavior is known as Artificial Intelligence (AI). Thanks to artificial intelligence, machines can scan photographs, comprehend voice, communicate in natural ways, and generate data-driven predictions.

Microsoft Cognitive Services is a set of AI services and APIs meant to make it easier for developers to integrate AI capabilities into their apps without having to start from scratch.

Developers just choose the Cognitive Services APIs that are best suited to their requirements.

You can use Microsoft Cognitive Services to:

- Analyze photos and generate a description of what they include, among other things.
- Execute translations in a variety of languages.
- Recognize people and emotions via machine vision.
- Images that you recognize.

1.2.13 Internet of Things (IoT)

IoT is a combination of managed and platform services that connect, monitor, and regulate billions of IoT assets at the edge and in the cloud. It also covers security and operating systems for devices and equipment, as well as data and analytics to help businesses build, implement, and manage IoT applications.

A typical IoT device consists of a circuit board with sensors connected to the Internet through Wi-Fi. Consider following scenarios:

- On a remote oil pump, a pressure sensor.
- Sensors for temperature and humidity in an air conditioner.
- In an elevator, there is an accelerometer.
- In a room, there are presence sensors.

1.2.14 Security + Identify Services

It allows for the detection and mitigation of cloud security threats. It also makes it easier to keep track of encryption keys and other sensitive data. It is made up of following components:

- Key Vault: Azure Key Vault enables you to secure cryptographic keys and generate secrets for cloud apps and services.
- Azure Active Directory: Azure Active Directory is identity management and directory service. This includes features such as multi-factor authentication and device registration, among others.
- Azure AD B2C: Azure AD B2C is a consumer-facing online and mobile app identity management solution in the cloud. It enables hundreds of millions of client IDs to be scaled.

1.2.15 Advantages of Microsoft Azure

Following are some advantages of Microsoft Azure:

- Pricing: Microsoft Azure offers pay-per-use pricing. The consumer of Microsoft Azure is charged depending upon the usage of the cloud infrastructure. Each component in the cloud environment, such as storage and bandwidth, is charged separately and billing is done on the actual usage.

- Scalability: Microsoft Azure offers agility in the IT infrastructure. Most businesses suffer from application performance when their business grows. To gain application performance, businesses have to purchase new hardware to ensure sustainability. Microsoft Azure, on the other hand, scalability can be achieved within a few minutes by scaling the IT hardware in the cloud environment. The businesses pay only for what they use.
- Availability: Microsoft Azure provides assured availability with 99.95% uptime.
- Manageability: Microsoft Azure uses Fabric Controller that maintains the instances on which an application is running. It performs a number of tasks, such as updating the operating system with the updates and patches and recovering an instance in case of a crash.
- Integration: The Connect feature allows developers to integrate data and users from their local infrastructure to the Microsoft Azure applications.

1.3 Azure Availability and Service Level Agreements (SLAs)

Microsoft Azure is a measure of how consistently and reliably Azure services are accessible to users. Azure, as any cloud service provider, aims for high availability to ensure that services are operational and accessible for users whenever they are required.

SLA stands for Service Level Agreement. It is a commitment made by Microsoft Azure to its customers regarding the level of service they can expect. SLAs typically include guarantees on uptime, performance, and other service-related aspects. For instance, an SLA might specify that a particular Azure service will be available 99.9% of the time in a given month.

Azure's SLAs outline the guaranteed uptime for various services and provide compensation in case Azure fails to meet these commitments. These SLAs help customers understand the reliability they can expect and provide a basis for holding Azure accountable for service disruptions beyond the agreed-upon thresholds.

Customers often rely on SLAs to ensure that their applications and services hosted on Azure have a certain level of reliability and uptime, enabling them to plan their operations accordingly.

1.4 Features and Functionalities of Azure Cloud Computing

Various product applications and services are built, deployed, and managed by Microsoft. This is made easier by having a huge network of data centers that provide secure and reliable storage. In recent years, these have been extended further to provide core infrastructure and foundational technologies for Microsoft

products and online services, including Bing, MSN, Office 365, Skype, and OneDrive.

Hundreds of thousands of servers, content distribution networks, edge-computing nodes, and fiber optic networks comprise the cloud-based infrastructure at the data centers.

Microsoft Azure is one such product that is built by Microsoft based on the cloud-based infrastructure. It is Microsoft's cloud platform product. It is a set of powerful cloud services that meet enterprise development requirements. It helps to build, manage, and deploy applications on a massive, global network. Microsoft Azure enables us to work with different programming languages, frameworks, and tools (including third-party products).

Features and functionalities of these cloud services of Azure are accessible through the Microsoft Azure portal. Various Azure components and services can be created and managed using the portal. To work with Azure, a developer must understand various components of the portal. The portal allows accessing the components based on a subscription.

Using the portal, one can also gain access to cloud service deployment and management tasks. The portal also has a reporting mechanism in which it displays a dashboard with status information depicting the overall status of the developer's deployments and accounts. The portal is refreshed regularly to display the active status. It is important to create an account on the Microsoft Azure portal prior to using the portal.

An Azure subscription grants a developer or user access to Azure services and the Azure portal.

Each subscription has two components:

- The Azure account, through which resource usage is reported and services will be billed as and when they are used.
- The subscription itself, which governs access to and use of the Microsoft Azure services that are subscribed. The subscription holder manages all the cloud services offered by Azure through the Azure portal.

1.5 Summary

- ✓ Cloud computing is an approach that enables convenient and on-demand access through the Internet to computing capabilities and resources.
- ✓ Cloud computing platforms provide different kinds of services, depending on the delivery model that they deploy, such as IaaS, PaaS, and SaaS.
- ✓ Microsoft Azure is a cloud computing platform created by Microsoft and is used for building, deploying, and managing services and applications.
- ✓ Microsoft Azure OS works as an integral part of the Azure Services Platform that covers different and separate application, storage, desktop environment, security, and so on.
- ✓ Microsoft Azure platform also supports Microsoft standards, programming languages, platforms, and protocols.

1.6 Test Your Knowledge

1. Which of these statements about Cloud computing is true?
 - A. Cloud computing focuses only on scalable storage services. It includes SQL Database and Microsoft Azure storage
 - B. Cloud computing is defined as an integral part of Azure Services Platform covering application, storage, desktop environment, security, and so on
 - C. Cloud computing is an approach that enables convenient and on-demand access through the Internet to computing
 - D. Cloud computing is an in-house IT based management approach and reduces upfront and regular expenses
2. Which of the following is a popular Cloud computing service?
 - A. Software as a Service
 - B. Cloud as a Service
 - C. Coding as a Service
 - D. Scheduling as a Service
3. Which Azure service supports multiple programming languages and simplifies microservice-based application design?
 - A. Virtual Machine Scale Sets
 - B. Azure Container Service
 - C. Batch
 - D. Service Fabric
4. What are the two main components of an Azure subscription?
 - A. Azure services and Azure Resource Manager
 - B. Azure Resource Group and Azure account
 - C. Azure account and the subscription itself
 - D. Azure Portal and Azure Marketplace
5. Which of these is not a Microsoft Azure platform architecture component?
 - A. Data Services
 - B. App Services
 - C. Compute
 - D. Troubleshooting services

1.6.1 Answers to Test Your Knowledge

1. C
2. A
3. D
4. C
5. D

Try It Yourself

- Explore Azure Services:
 1. Sign up for a free Azure account.
 2. Experiment with deploying a virtual machine and creating a Web app.
- Experiment with Azure Features:
 1. Set up a database using Azure SQL Database.
 2. Create a storage account in your existing Azure account and create a container in it. Name this new container as – ‘Container.demo’.



SESSION 02

UNDERSTANDING MICROSOFT AZURE CLOUD SERVICES

Overview:

This session describes the features and functionalities of the Microsoft Azure cloud platform. It also explains data and data access technologies. Further, the session explores different Azure cloud services.

Learning Objectives

In this session, students will learn to:

- Define Virtual Machines (VMs)
- Explain how to configure and manage VMs
- Describe data access technologies available under the Azure platform
- Explain Data and Data Access

2.1 Virtual Machines

A VM is essentially a software emulation of a computer system. It operates like a standalone computer within another physical machine. By using special software called a hypervisor, you can create and run multiple VMs on a single physical server or computer.

Each VM runs its own Operating System (OS) and behaves as if it were an independent computer, complete with its own CPU, memory, storage, and network interfaces. It allows simultaneous operation of multiple operating systems on a single machine, streamlining resource management, software testing, diverse application usage, and creating secure isolated environments.

Figure 2.1 represents a Virtual Machine graphically.



Figure 2.1: Graphical Representation of a Virtual Machine

VMs offer flexibility, scalability, and cost-effectiveness, enabling users to consolidate multiple servers onto a single physical machine, thereby optimizing hardware utilization and reducing infrastructure costs.

2.1.1 Virtual Machines in Azure

In Microsoft Azure, VMs are a core service providing scalable computing resources in the cloud. They allow you to create and manage virtualized instances of Windows or Linux-based operating systems in Azure's data centers.

2.2 Azure Portal

The Azure portal is an integrated Web console that provides an alternative to command-line tools. The Azure portal allows developers to manage their Azure subscriptions using a graphical user interface. Developers can create, manage, and monitor everything from simple Web applications to complex cloud deployments. Also, they can create custom dashboards for an organized resource view and then, adjust accessibility settings for optimal use.

The Azure portal provides fault tolerance and continuous availability. It is presented in each Azure data center. This makes the Azure portal resilient to individual datacenter failures and avoids network slowdowns due to proximity to users. The Azure portal is constantly updated and requires no downtime for maintenance.

Developers can access the Azure Portal through either one of following links:

<https://portal.azure.com/>
<https://azure.microsoft.com/>

To begin using the portal, developers can perform following steps:

- 1) Log in to the Azure Portal with your credentials. If one does not yet have an account at the portal, create it using these steps:

- i. To create a free trial account, click Free Account (on top right), Try Azure for Free (center of the page), or Start free (bottom right). Refer to Figure 2.2.

Free Trial accounts are valid for 30 days. Alternatively, you can sign up for a paid subscription to use Azure.

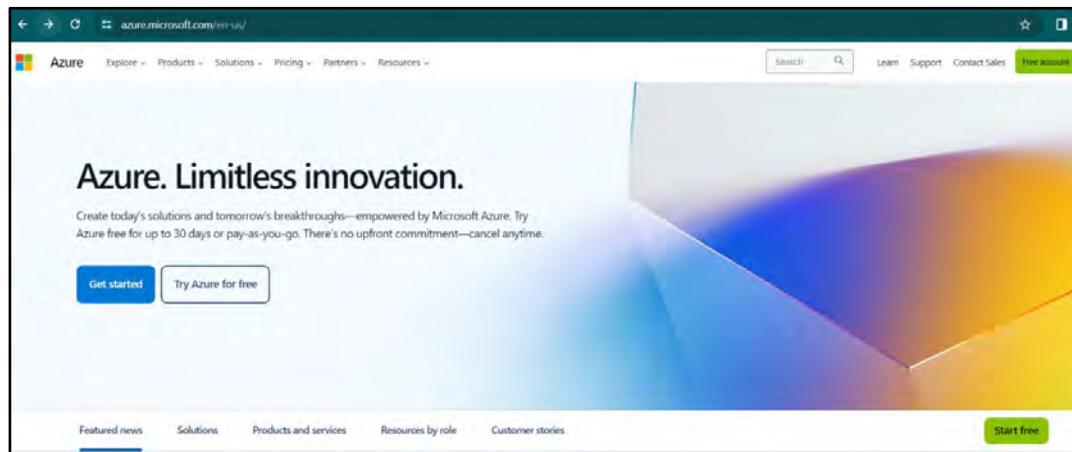


Figure 2.2: Azure Portal

When developers click either one, they will be routed to the page where they can sign up. Refer to Figure 2.3.

- ii. Click Start Free.

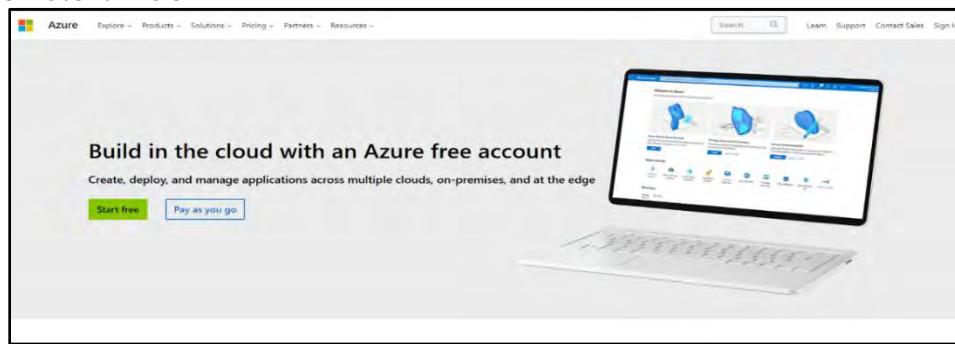


Figure 2.3: Account Creation Page

Clicking this option redirects developers to a new page where they are asked to sign in with their Microsoft Azure credentials or else, create a new Microsoft Azure account.

- iii. Click Create one on the screen shown in Figure 2.4.

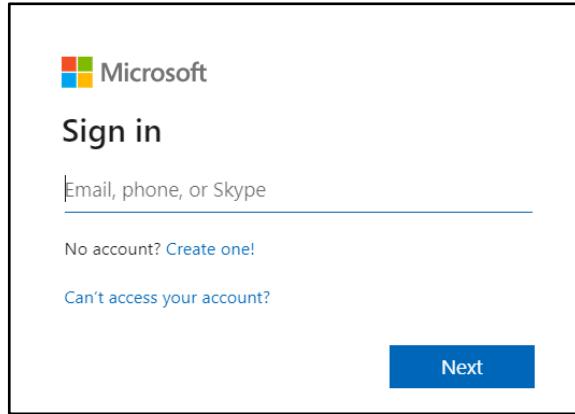


Figure 2.4: Azure Account Sign Up/Creation Page

Developers will be asked to specify an email id to create an Azure account.

- iv. Enter email id and click Next. Then, enter the password. Refer to Figure 2.5.

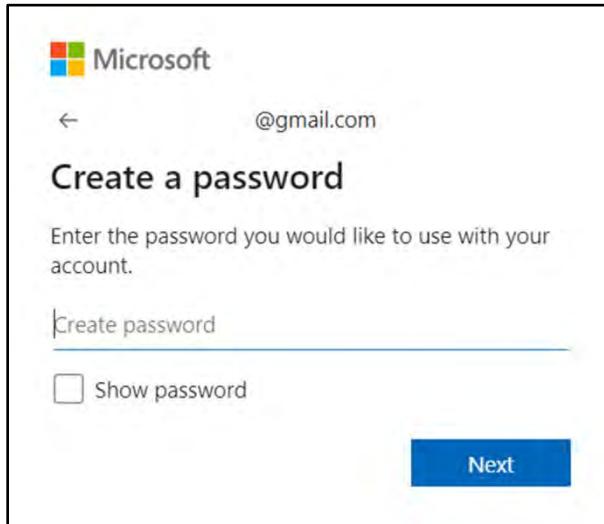


Figure 2.5: Enter Password

Ensure passwords must have at least eight characters and contain at least two of the following: uppercase letters, lowercase letters, numbers, and symbols.

- v. Once password is entered, an Azure account is successfully created. Developers will be able to login to Microsoft account using the credentials. Then, they will land on the Azure Portal as shown in Figure 2.6.



Figure 2.6: Azure Landing Page

Layout and Customization

At the top left corner, one can see a hamburger icon. This is the portal menu. When the portal menu is in flyout mode, it is hidden until required. One should click the menu icon for opening and closing the menu, as shown in Figure 2.7.



Figure 2.7: Flyout Mode Menu

The portal menu is always visible when one chooses docked mode. One can collapse the menu to provide more workspace, as shown in Figure 2.8.

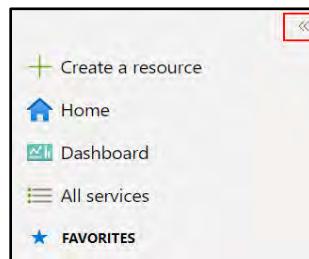


Figure 2.8: Docked Mode

The Appearance + startup views panel consists of two sections. The Appearance section lets developers choose the menu behavior, color theme, and determine

whether to use a high-contrast theme or not. The Welcome view allows them to customize what is seen after first sign-in to the Azure portal. Refer to Figure 2.9.

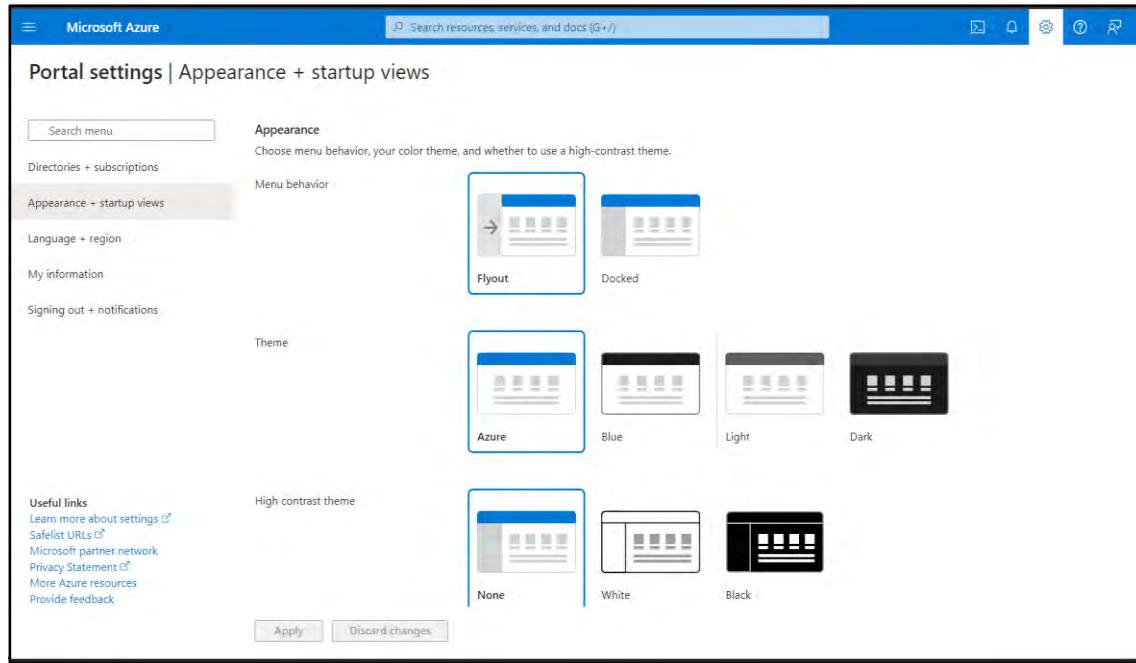


Figure 2.9: Welcome View

Azure Dashboard

The dashboard provides a focused view of developer's most important subscription resources. Developers can create additional dashboards for use or publish customized dashboards and share them with others in their organization.

Dashboards are private by default when first created, but one can publish dashboards and share them with others in the organization.

1. First, sign in to the Azure portal.
2. Select Dashboard from the Azure portal menu.
3. Click Create, as shown in Figure 2.10.

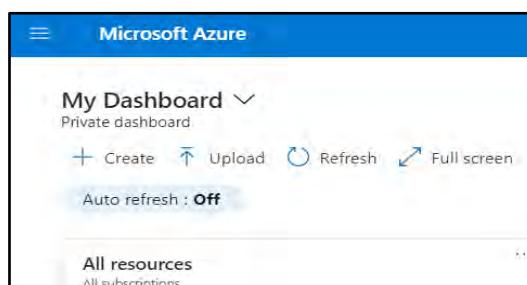


Figure 2.10: Dashboard

4. Select Custom as shown in Figure 2.11.

Figure 2.11: Create Dashboard

5. In the dashboard label, select the text My Dashboard and enter a name that will help developers easily identify their custom dashboard, as shown in Figure 2.12.

Figure 2.12: Dashboard Customization

6. To save the dashboard, select Done on the page title.

2.2.1 Create, Deploy, and Work with Resources

Azure provides a manageable item named Resource. Following are some of the common resources:

- Virtual Machine
- Storage account
- WebApp Database
- Virtual network

Resources can be deployed, updated, or deleted for a solution as a group with the help of Azure Resource Manager. This is done using a single and coordinated operation. The same template can be used for deployment, testing, staging, and production. Resources can be managed post-deployment using the Resource Manager. Managing includes security, auditing, and tagging features.

Perform following steps to create a resource:

1. Sign in to the portal, if not done already.
2. Click + Create a resource option from the left panel. Create resources from the Dashboard, as shown in Figure 2.13.

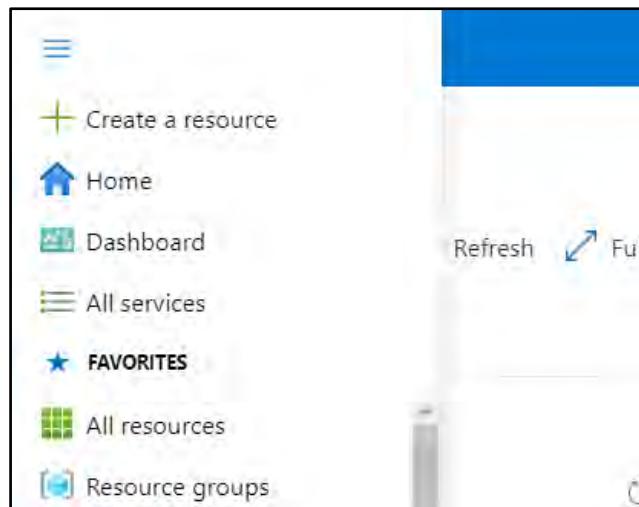


Figure 2.13: Microsoft Azure Dashboard

This will load the new resource window, as displayed in Figure 2.14. Resources can be added or managed from this window.

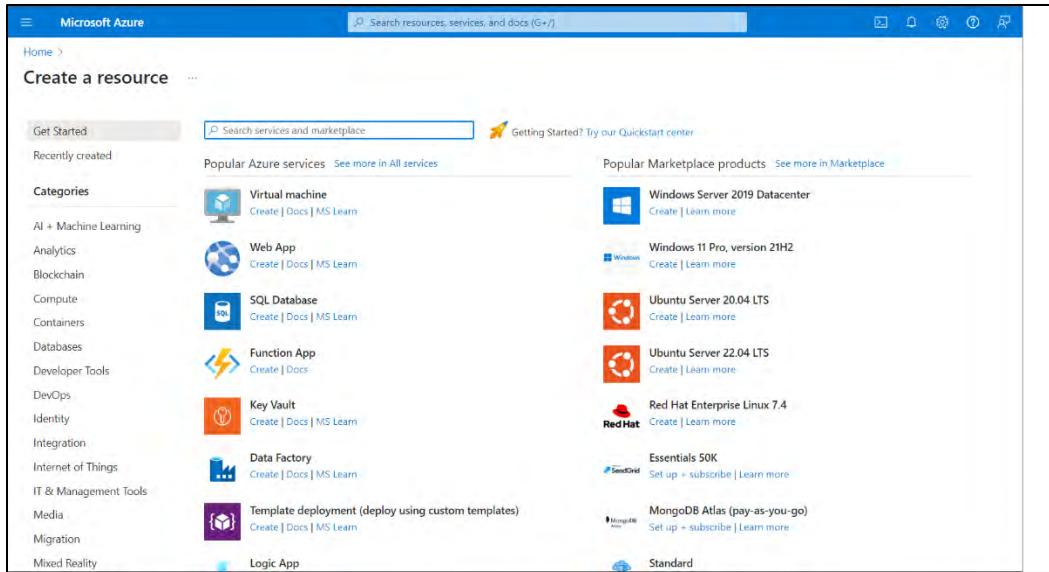


Figure 2.14: New Resource Page

3. Now, select any of the menu and sub-menu options from this window.

The most commonly used resources are added to this window: Windows VM, SQL Server, Cosmos DB, and DevOps. Upon selecting an option, resources are quickly added to the account.

Creating a VM: Within the Azure Portal, you can create a new virtual machine. You specify details such as the desired operating system, compute power (CPU, RAM, and so on), storage configuration, networking settings, and more.

Resource Group: Azure organizes resources into Resource Groups. When creating a VM, you specify the resource group where the VM will reside, allowing for easier management and grouping of related resources.

Perform following steps to create resource groups:

1. Sign in to the Azure portal, if not done already.
2. Select Resource groups, as shown in Figure 2.15.

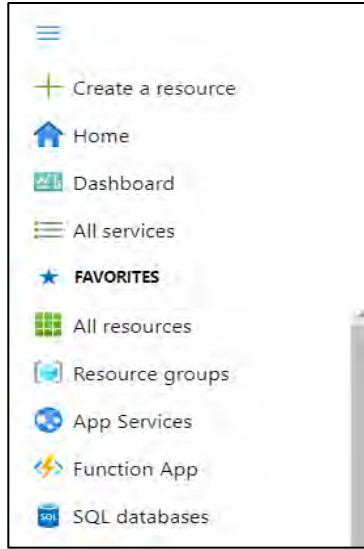


Figure 2.15: Select Resource Group

3. Select Create, as shown in Figure 2.16.

A screenshot of the Microsoft Azure Resource groups page. The page title is "Resource groups". It shows a toolbar with "Create", "Manage view", "Refresh", "Export to CSV", "Open query", and "Assign tags". Below the toolbar are filter options: "Filter for any field...", "Subscription equals all", "Location equals all", and "Add filter".

Figure 2.16: Select Create

4. Enter following values:
 - Subscription: Select your Azure subscription.
 - Resource group: Enter a new resource group name.
 - Region: Select an Azure location, such as Central US.

Refer to Figure 2.17.

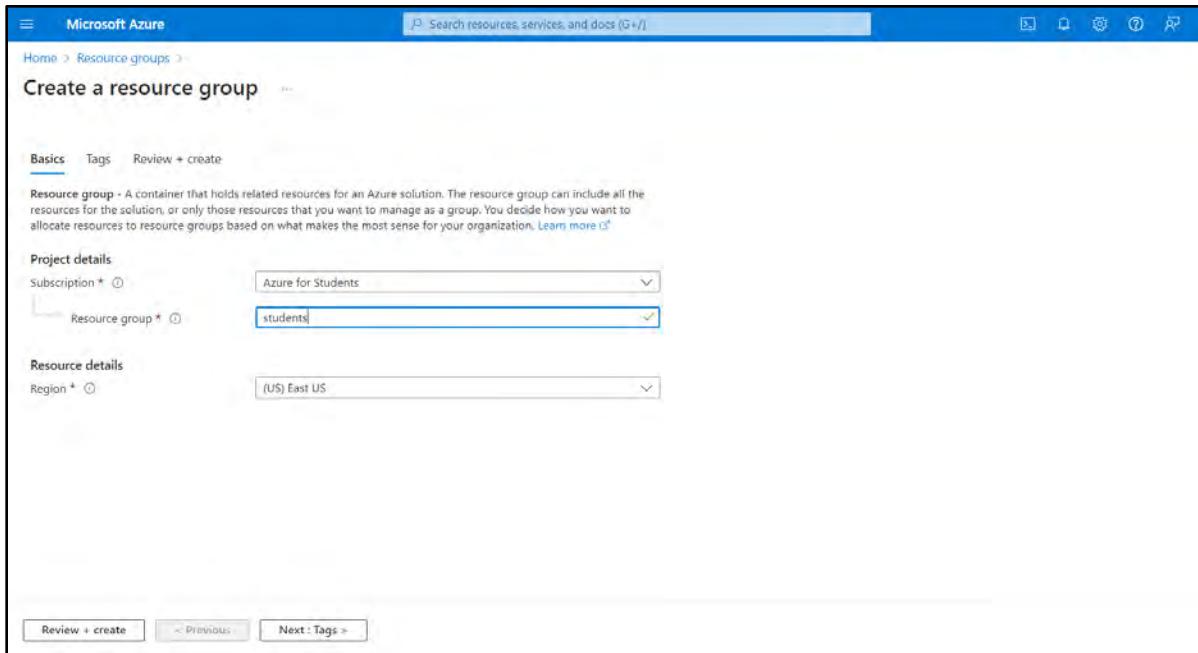


Figure 2.17: Create Resource Groups

Azure Infrastructure: Once created, the Azure infrastructure provisions the VM by allocating the necessary hardware resources from Azure's data centers. The VM is assigned a public IP address for Internet access if required.

Access and Configuration: Azure provides various ways to access and configure your VM, including Remote Desktop Protocol (RDP) for Windows VMs or Secure Shell (SSH) for Linux VMs. Users can remotely connect to their VMs and install software, configure settings, and manage the operating system as required.

Scaling and Monitoring: Azure offers features to scale VMs vertically (increasing or decreasing resources within a VM) or horizontally (adding or removing VM instances). Additionally, Azure's monitoring tools allow developers to track VM performance, set up alerts, and manage resource utilization.

Networking and Security: Azure provides networking capabilities to connect VMs to virtual networks, establish firewalls, set up load balancers for high availability, and implement security measures such as network security groups and encryption.

Pricing and Billing: Azure VMs operate on a pay-as-you-go model, where you are charged based on the resources used, such as compute power, storage, and networking. Azure provides various pricing options, including options for reserved instances and cost management tools to monitor spending.

Overall, Azure VMs offer flexibility, scalability, and robust features for hosting applications, running workloads, and building complex infrastructures in the cloud.

2.3 Configuring and Managing VM in Azure Cloud

Configuring and managing a VM involves several steps, from creation to ongoing maintenance.

The step-by-step process to create a VM is described as follows:

- 1) Navigate to Virtual Machine by clicking Virtual Machines. This will take developers to the Virtual Machine section where they can manage their VMs, as shown in Figure 2.18.

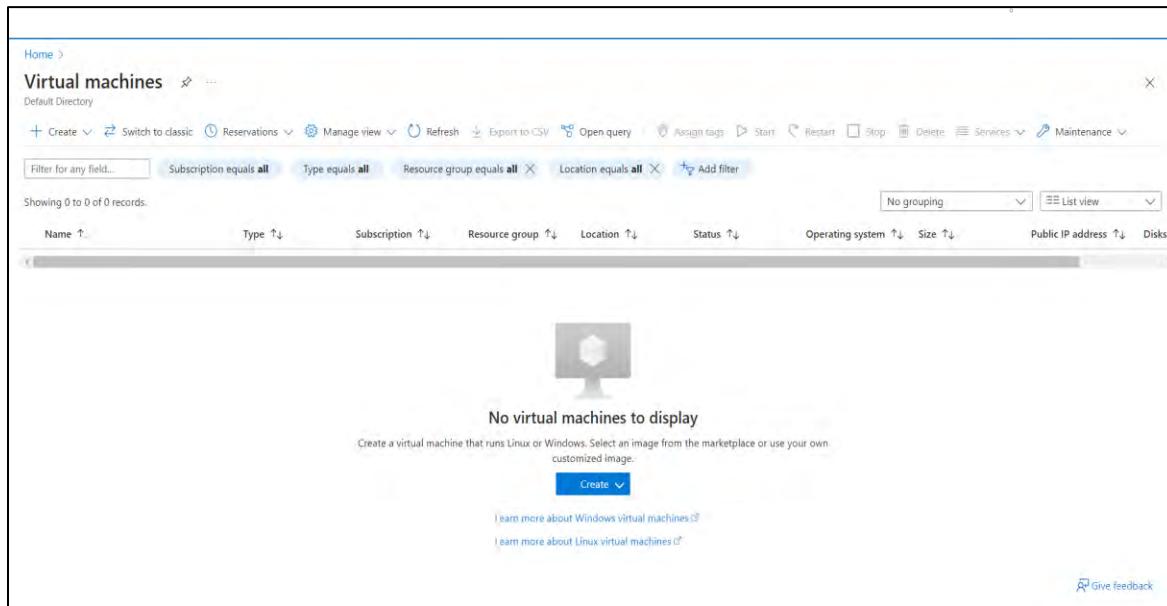


Figure 2.18: Virtual Machine Page

- 2) Click Create and then, select Azure Virtual Machine to create a new VM (Refer to Figure 2.19).

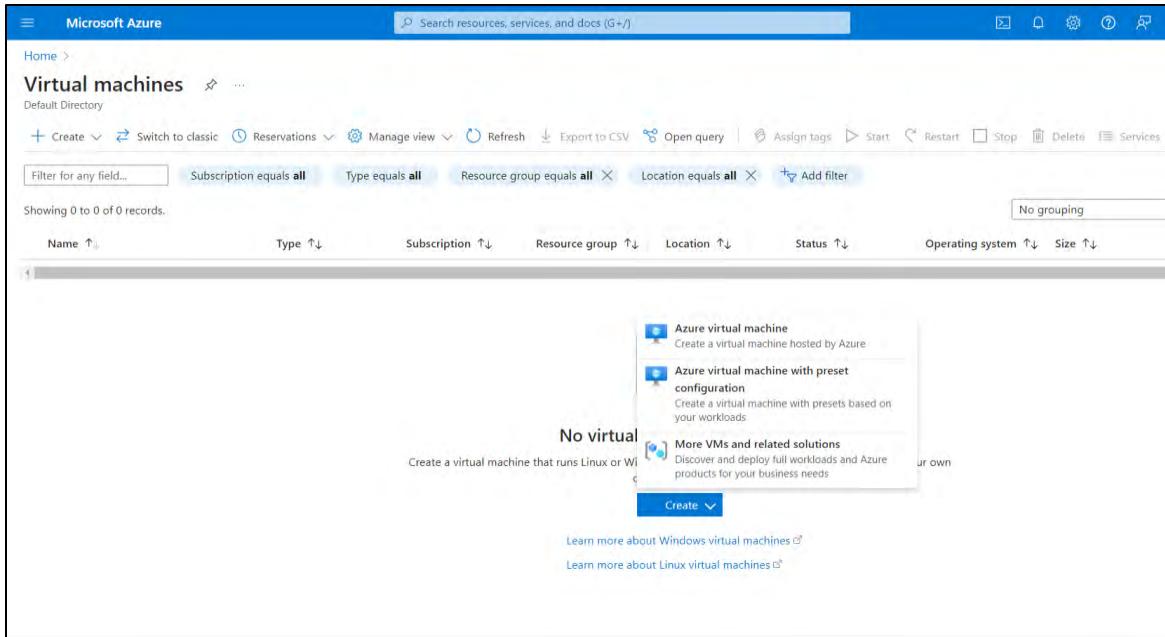


Figure 2.19: Create a Virtual Machine

- 3) Fill the Project details and Instance details such as:
 - Subscription - Choose the subscription.
 - Resource Group - Create a new resource group or select an existing one.
 - Virtual Machine Name - Give a name to the VM.
 - Region - Choose the data center region which is closest.
 - Availability options - Choose the availability preferences.
 - Security type - Select the security type for the VM.
 - Image - Select the operating system image to be used.
 - Size - Select the appropriate configuration for the VM based on CPU, memory, and Storage requirements.

Refer to Figure 2.20.

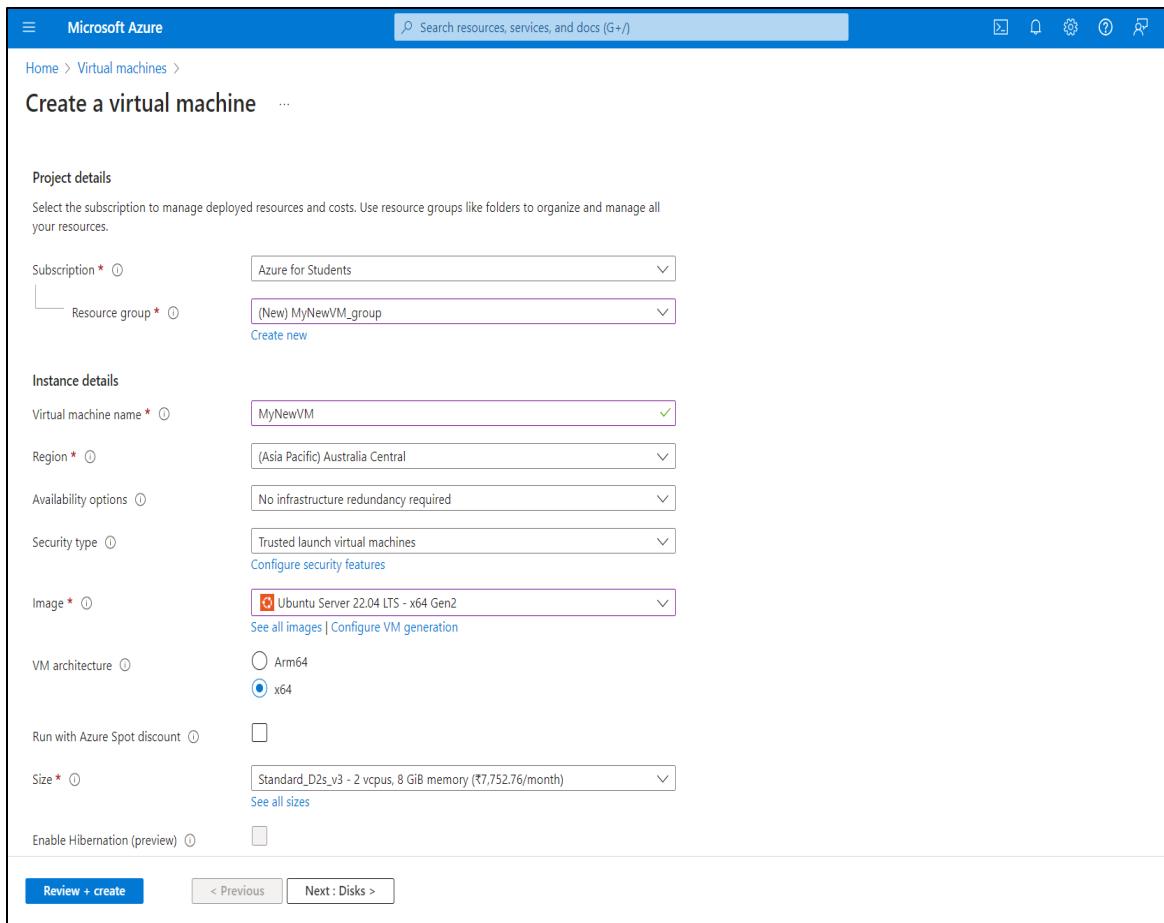


Figure 2.20: Project Details and Instance Details

- 4) Configure the VM. The step-by-step process for Authentication and security of the VM is as follows:
 - Under Authentication type, select either an SSH public key or a password.
 - According to developer's choice, they can provide the required credentials (a username and password).
 - Configure inbound port rules to allow remote access via Secure Shell (SSH), HyperText Transfer protocol (HTTP) and Hypertext Transfer Protocol Secure (HTTPS).

Refer to Figure 2.21.

The screenshot shows the 'Create a virtual machine' wizard in the Microsoft Azure portal. The current step is 'Administrator account'. Under 'Authentication type', 'Password' is selected. The 'Username' field contains 'AzureUser'. The 'Password' and 'Confirm password' fields both contain '*****'. In the 'Inbound port rules' section, 'Allow selected ports' is chosen, and the selected ports are 'HTTP (80), HTTPS (443), SSH (22)'. A note states: 'All traffic from the internet will be blocked by default. You will be able to change inbound port rules in the VM > Networking page.' Navigation buttons at the bottom include 'Review + create', '< Previous', and 'Next : Disks >'.

Figure 2.21: Administrator Account and Inbound Port Rules

- 5) Configure Disks and Storage. The step-by-step process for configuring Disks and Storage aspects of the VM is as follows:
- Select the OS disk size according to your requirements.
 - Choose the OS disk type (Standard HDD, Standard SSD, or Premium SSD). Refer to Figure 2.22.

The screenshot shows the 'Create a virtual machine' wizard in the Microsoft Azure portal, specifically the 'Disks' tab. It shows the configuration for the OS disk. The 'OS disk size' is set to 'Image default (30 GiB)', 'OS disk type' is 'Premium SSD (locally-redundant storage)', and 'Delete with VM' is checked. Other options like 'Key management' and 'Enable Ultra Disk compatibility' are also present. A note indicates: 'Encryption at host is not registered for the selected subscription.' Navigation buttons at the bottom include 'Basics', 'Disks', 'Networking', 'Management', 'Monitoring', 'Advanced', 'Tags', and 'Review + create'.

Figure 2.22: Configuration of Disks and Storage

- 6) Configure the network. The step-by-step process for Network Configuration of the VM is as follows:
- Select an existing virtual network and subnet, or create a new one.
 - Assign a Public IP address if required.
 - Configure network security groups to control traffic.
 - Select inbound rules for traffic.

Refer to Figure 2.23.

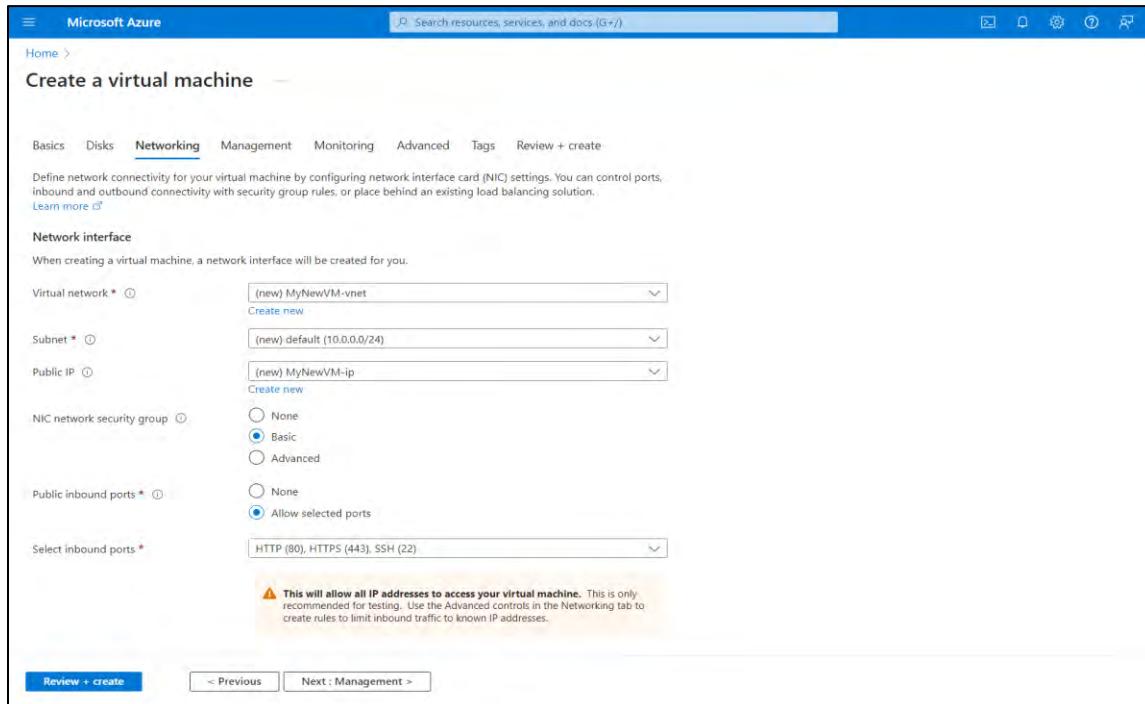


Figure 2.23: Configuration of Network

- 7) Click Review + create. Developers will get a validation pass as shown in Figure 2.24.

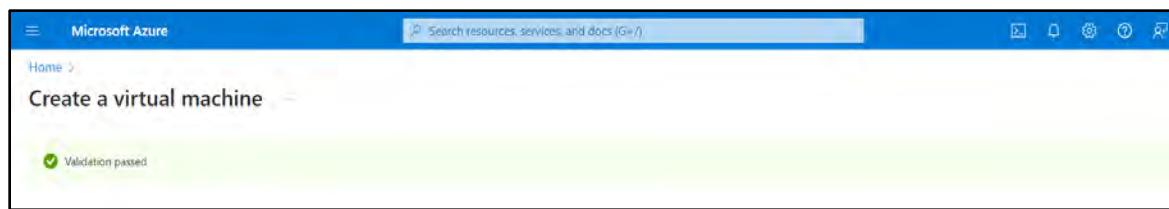


Figure 2.24: Validation Pass

- 8) After validation, click Create. The Virtual Machine will be ready to use.

Resource Management

Let us look at how resource management can be done.

a. Monitoring and Scaling:

- Use Azure Monitor to track VM performance, set up alerts, and manage resource utilization.
- Scale VMs vertically (adjust resources within a VM) or horizontally (add or remove VM instances) as required.

b. Resource Group Management:

- Organize VMs and associated resources within resource groups for easier management.

Cost Management

Developers can perform cost analysis and monitor resource consumption and costs for the VM by using Azure Cost Management tools.

They should consider using reserved instances or cost optimization strategies.

Backup and Disaster Recovery

To perform backup and disaster recovery processes, developers can follow these steps:

a. Backup Policies:

Set up backup policies to regularly back up VM data.

Establish disaster recovery plans to ensure business continuity.

Regular Maintenance

To perform regular maintenance, developers can follow these steps:

a. OS Updates and Patches:

Keep the operating system and installed software up-to-date with patches and updates.

b. Performance Optimization:

Regularly review and optimize VM configurations for performance.

Documentation and Best Practices

Maintain documentation detailing configurations, access credentials, and key settings.

Follow best practices provided by Azure documentation and security guidelines. Managing a VM in Azure involves continuous monitoring, optimization, and adherence to security practices to ensure efficient operation and safeguarding of resources and data. The Azure Portal offers a user-friendly interface for performing most of these tasks.

2.4 Data and Data Access Technologies

Microsoft Azure provides a variety of data access technologies to meet the changing industry requirements. These technologies include SQL database resources, Cosmos DB, data warehouses, and data factories. By using Microsoft Azure database resources, applications ranging from small dynamic Websites to large-scale data warehouses and data mining applications can be built.

An Azure SQL database helps to drive applications that deal with small to medium chunks of data. Data warehouses and data mining applications use an Azure SQL data warehouse and Azure data factory to handle large chunks of data. Azure Cosmos DB is also another technology in use that also supports NoSQL database services, such as MongoDB or Microsoft's own NoSQL Database Cosmos DB.

Figure 2.25 shows the database services provided by Microsoft Azure.

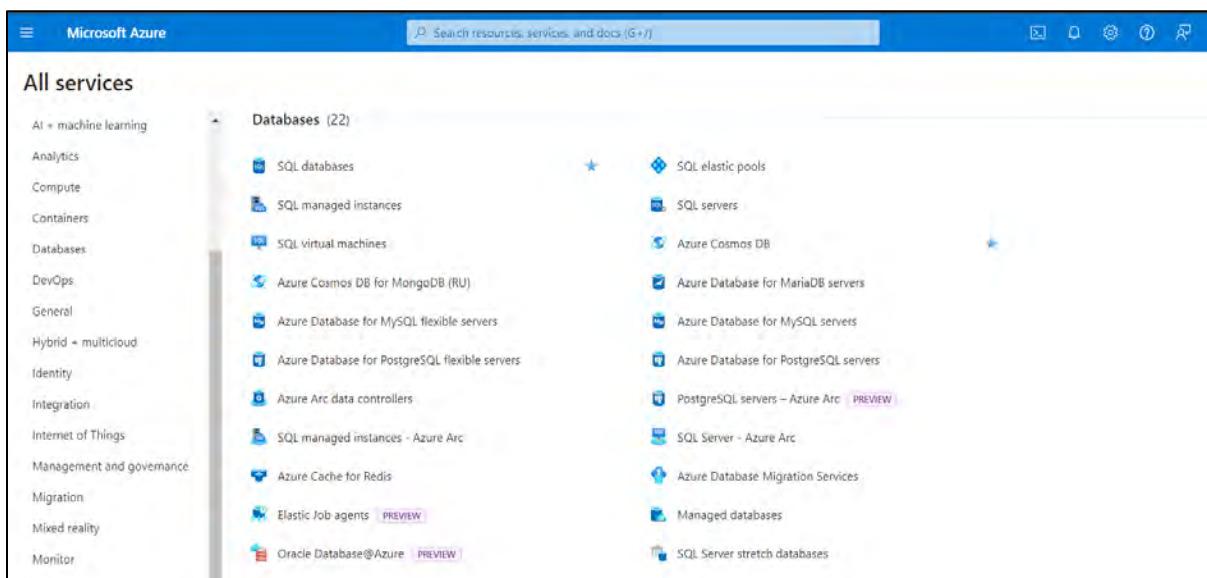
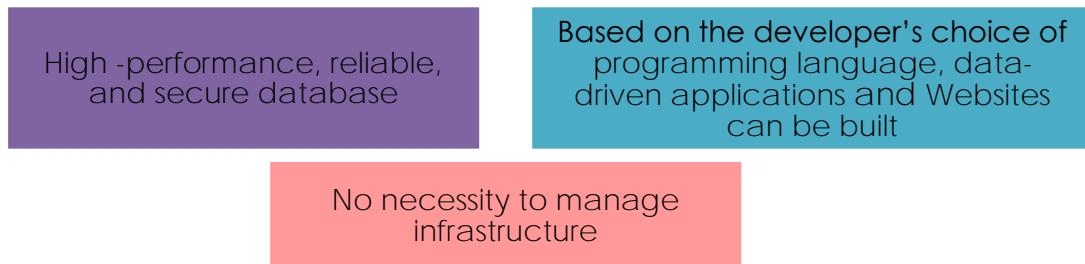


Figure 2.25: Azure Database Services

2.4.1 Azure SQL Database

Microsoft SQL Server database engine provides a relational Database-As-A Service (DBaaS) named as Azure SQL Database.

Following are some of its features:



Following deployment options are provided by Azure SQL database:

Single Database:

- Operates autonomously with dedicated resources.
- Managed through a logical server.
- Optimized for contemporary cloud-native applications.
- Offers Hyperscale and serverless alternatives.

Elastic Pool:

- Forms a cluster of databases sharing resources.
- Managed through a logical server.
- Ideal for multi-tenant SaaS applications with seamless database movement.
- Presents a cost-effective solution for managing performance across multiple databases with varying usage patterns.
- Hyperscale elastic pools are currently available in preview.

2.4.2 Azure Cosmos DB

Microsoft offers Azure Cosmos DB, which is a globally distributed, multi-model database.

Cosmos DB also offers following services:

- Throughput
- Latency
- Availability
- Consistency assurance with Service Level Agreements (SLAs)

Azure Cosmos DB helps to elastically and independently scale throughput and storage across various geographic regions that use Azure with a click of a button.

Throughput is a measure of a system's overall performance in sending data through all its components.

2.4.3 Azure SQL Data Warehouse

Azure SQL Data Warehouse is a cloud-based Enterprise Data Warehouse (EDW). Complex queries are run using Massively Parallel Processing (MPP) across petabytes of data. A key component of a big data solution is the SQL Data Warehouse.

Simple PolyBase Transact-SQL queries are used to import Big Data into SQL Data Warehouse. High-performance analytics is then run using the power of MPP.

2.4.4 Azure Data Factory

Data movement and data transformation can be managed and automated within data-driven workflows in the cloud. This can be done using a cloud-based, data integration service called the Azure Data Factory.

Following tasks are performed using Azure Data Factory:

- Data from various data stores are absorbed using data-driven workflows (called pipelines). These workflows are created and scheduled by Azure Data Factory.
- Data is processed or transformed using compute services. These include Azure HDInsight Hadoop, Spark, Azure Data Lake Analytics, and Azure Machine Learning.
- Data is published to data stores, such as Azure SQL Data Warehouse to be utilized by Business Intelligence (BI) applications.

Azure Redis Cache, used as a cache, utilizes the popular open-source cache called Redis. Some systems rely heavily on backend data stores and their performance gets affected on account of that. By temporarily copying frequently accessed data to fast storage, the performance of these systems can be improved. The fast storage should be closely located to the application. Using Redis cache helps to use the in-memory cache as a fast storage rather than taking it from a disk by a database.

Azure Redis Cache also plays following roles:

- A data structure store.
- Distributed non-relational database.
- Message broker.

Performance of the application is improved due to the efficient Redis engine performance, which is marked by a high throughput and a low latency.

Apart from the mentioned four database resources, Azure also provides some more database resources. These are as follows:



2.5 Advanced Azure Cloud Services

Using the Azure platform, existing applications can be hosted, the development of new applications can be streamlined and even on-premises applications can be enhanced. Azure integrates cloud services required to develop, test, deploy, and manage applications by leveraging benefits offered through cloud computing.

2.5.1 Compute Services

Compute acts as a facilitating model, which handles all the computing assets that any application runs. Azure is a well-known computing service, especially for cloud-based applications. It provides computing resources such as storage, processors, memory, networking, and operating systems.

Azure supports a wide range of computing solutions, some of which are as follows:

- Virtual Machine
- Container Instances
- Kubernetes Services

Virtual Machine

Azure VM gives the flexibility of virtualization without buying and maintaining the physical hardware that runs it. Developers can determine the size of the Virtual Machine based on the workload that they want to run. The size developers choose then, determines factors such as processing power, memory, and storage capacity.

The Azure VM enables the deployment of different services (Windows or Linux) within the Azure cloud. When developers implement a virtual machine, each VM will have an associated OS and data disk.

Container Instances

Azure Container Instances are one of the simplest ways to run containers on Azure. A CI does not require any sort of middleware, orchestration, or clusters. This means that running containers is as simple as creating any other service using the Azure Portal, the Azure CLI, PowerShell, the Azure Cloud Shell, or ARM templates. Azure Container Instances enables exposing user's container groups directly to the Internet with the IP address and a Fully Qualified Domain Name (FQDN).

Kubernetes Services

Azure Kubernetes Services (AKS) Engine is an open-source tool for creating Kubernetes clusters on Azure. It is useful for Azure regions in which AKS is not available. It accepts a configuration file and then interacts with Azure Resource Manager to create a cluster of virtual machines and the network infrastructure to go with it. AKS Engine is IaaS.

2.5.2 Azure Storage Services

Azure Storage Account is a fundamental component of Microsoft Azure's cloud platform, providing scalable and secure storage solutions for various data types. Data in Azure Storage is accessible across the globe over HTTP or HTTPS. The storage service encrypts all data written in imitation of an Azure storage account.

Azure storage provides you with control over who can get an entry as per data sensitivity.

Azure Storage includes following data services:

- Azure Blobs
- Azure Files
- Azure Queues
- Azure Tables
- Azure Disks

Azure Blobs

Blob Storage is Azure's provider for storing binary large objects or blobs that are usually composed of unstructured records inclusive of text, images, and videos, alongside their metadata. Blobs are stored in listing systems called containers.

The blob service enables users access to following features:

- Images and documents can be directly added to the server.
- Files that will be accessed globally can be stored.
- Videos and audio can be streamed.
- Data can be stored for use during backup, disaster recovery, and archiving.
- Data used by on-premises or Azure-hosted service can be stored.

Azure Files

Azure Files provides a Server Message Block (SMB) share that can be mounted as part of a container's file system for persisting data outside the container. SMB is better suited for random read and writes to the file system; most container environments support using Azure File storage.

Benefits of Azure Files are as follows:

- Easy to manage
- Secure storage
- Cross-platform support
- Highly Scalable
- Hybrid Access

Azure Queues

Queue Storage allows you to separate your elements and have authentic asynchronous communication. In Azure Queue Storage, the number of queues is barely restricted by the capability of the storage account. Azure queues is to allow communication between different parts of an Azure application.

Queues and messages are often created programmatically or using the Storage Explorer tool.

Queue storage consists of following elements:

- Storage Account

- Queue
- Message

Azure Tables

Structured NoSQL data is stored in the cloud using Azure Table storage. Azure Table storage provides a key/attribute store with a schemaless design. Data can be adapted as per the application requirements as table storage is schemaless. Table storage data is fast to access and cost-effective.

Compared to traditional SQL, it is considerably lower in cost.

Table storage is used to maintain flexible datasets. These include any applications used by the Web, email address books, information regarding a device, or any metadata that is required by a service being used. A table can include any number of entities. Also, depending on the capacity of the storage amount, any number of tables can be added.

Uses of Table storage are as follows:

- Web scale applications are served by storing TBs of structured data.
- Datasets that do not require complex joins, foreign keys, or stored procedures can be stored. For easy access, these can be denormalized.
- Data can be queried using a clustered index.
- Data protocol and LINQ queries with Windows Communication Foundation (WCF) Data Service .NET Libraries can be used to access data.

Azure Disks

Azure Managed Disks are block-level storage volumes run by Azure and used with Virtual Machines. Managed disks are like physical disks on a local server, but are virtualized. Managed Disks are coordinated with availability sets to make sure that the virtual machine disks in an availability set are separated from one another to cast aside any point of error.

2.5.3 Network Services

Azure's network services provide a variety of networking features that can be used together or separately. The network services are used for communication between servers, resources, and so on. Azure enables developers to connect their cloud and on-premises infrastructure and services. Azure provides Virtual Networks (VNets), gateways, connectivity services such as DNS, application protection services such as firewalls, network security groups, and application delivery services such as CDNs, traffic managers, load balancers, and network monitoring services such as Azure Monitor.

Developers can use these network services in Azure or a combination of them to connect Azure resources and on-premises resources.

Networking services in Azure include:

- Virtual Network (Vnet)
- Traffic Manager

2.5.4 App Services

If developers want the fastest way to publish their Web project, they should consider using Azure App Service. This platform makes it easy to upgrade Web applications for mobile use and smoothly share them as simple REST APIs. It has features for social logins based on traffic, and it supports scaling as required, testing in real-world conditions, ongoing updates, and deploying using containers.

App Services are of following types:

Web Apps:

Enables hosting of Websites and network applications written in .NET, Java, PHP, Node.js, and Python.

Mobile Apps:

Extends the Web app for access on mobile devices and provides certification Server storage and storage with social providers and Azure Active Directory (Azure AD) Integrates with Azure notification Hubs for push notifications.

API Apps:

Exposes APIs securely in the cloud using Swagger metadata. It can be easily consumed by clients.

2.5.5 Database Service

As organizations embrace cloud computing and move applications to the cloud, Azure SQL databases provide everything a database has to offer.

Azure SQL Database is Microsoft's highly scalable, multi-tenant and highly available Platform as a Service (PaaS) or Database as a Service (DbaaS) offering. Microsoft stands for 'OS', storage, networking, virtualization, Servers, installations, upgrades, infrastructure management, and maintenance.

Azure SQL Database has the alternative deployments:

Elastic Pool

You can deploy database pools with a common set of resources managed through logical servers.

Managed Instance

This is for local clients. If you already have an instance of SQL Server in your local datacenter and want to migrate to Azure with minimal application changes and maximum compatibility. Then, the new instance is moved to the managed instance.

Single Database

You can deploy a single database in Azure with your own set of resources managed through a logical server.

2.5.6 Testing the Cloud Service

In organizations, testing normally involves performance testing, production service monitoring, general, and load testing.

Necessity for Cloud Testing:

The software testing carried out by traditional approaches incurs a high cost to simulate user activity from various geographical regions. Testing the firewalls and load balancers involves maintenance of hardware and software.

Just as the local environment, a number of bugs can exist in cloud-based applications. The solution must be tested in an environment other than the production environment.

Limitations

Test results are not likely to be correct because of the dissimilar performance of the networks from different service providers. In many cases, service virtualization may help in imitating a few performance behaviors required for precise and an-in-depth testing. In addition, the initial setup cost for switching to cloud is quite high. This is because it includes changing a few test cases as per the chosen cloud environment.

Figure 2.26 displays the steps in cloud testing.

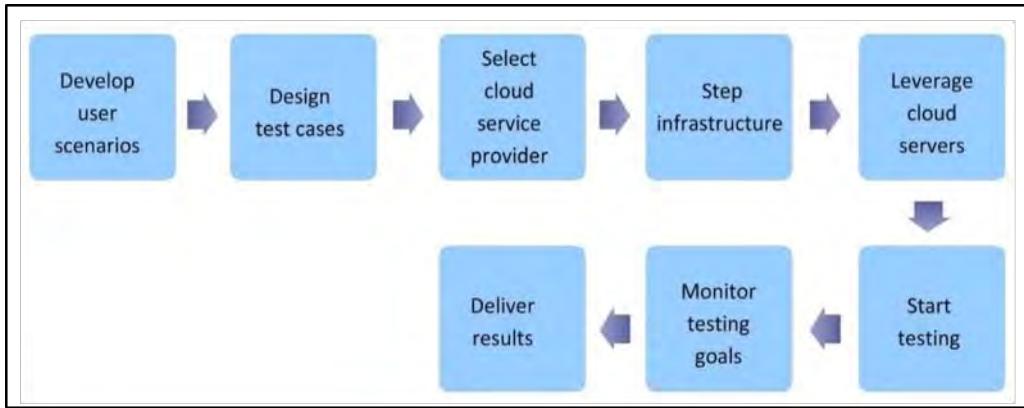


Figure 2.26: Steps in Cloud Testing

Successful testing can be done if following are taken care of:

- Understanding of the elasticity model/dynamic configuration method used by the platform provider.
- Reviewing and keeping oneself updated about the provider's changing services and SLAs.
- Interacting with the service provider as an on-going operations partner if producing Commercial Off-The-Shelf (COTS) software.

2.6 Application Development in Azure

Microsoft Azure provides some valuable features for robust application development. Following are the features:

- Allows building and hosting Web applications based on the choice of a programming language without the necessity to manage the infrastructure; for example, .NET, Python, Java, PHP, or Ruby.
- Provides auto-scaling.
- Ensures maximum availability.
- Supports even Linux apart from Windows.
- Provides automated deployments from GitHub, Visual Studio Team Services, or any Git repo.
- Hosts Web applications, REST APIs, and mobile backends.
- Runs and scales applications easily on a Windows-based environment.

2.7 Summary

- ✓ Microsoft Azure is a set of powerful cloud services by Microsoft that meet enterprise development requirements.
- ✓ Azure helps to build, manage, and deploy applications on a massive, global network.
- ✓ Microsoft Azure enables developers to work with different programming languages, frameworks, and tools (including third-party products).
- ✓ Users can create paid or free accounts and obtain subscriptions to access the Azure portal.
- ✓ Microsoft Azure portal defines the access to the components based on the subscription.
- ✓ Using Microsoft Azure, Windows Virtual Machines can be created for use as Web Server or Hosted Database Server.
- ✓ Azure database services allow developers to use a variety of database or data sources as per requirements.
- ✓ Azure's diverse technologies, from SQL databases to Cosmos DB, cater to requirements from small Websites to large-scale warehouses.

2.8 Test Your Knowledge

1. What is a VM?
 - A. A physical computer
 - B. A software emulation of a physical computer
 - C. A networking protocol
 - D. A programming language
2. Which task is NOT part of managing a VM?
 - A. Starting and stopping the VM
 - B. Configuring network settings
 - C. Writing code for the VM
 - D. Backing up the VM
3. Which Azure service is primarily used for structured query language-based data storage?
 - A. Azure Cosmos DB
 - B. Azure Blob Storage
 - C. Azure SQL Database
 - D. Azure Data Lake Storage
4. What does data access involve?
 - A. Creating data
 - B. Deleting data
 - C. Retrieving, manipulating, or transferring data
 - D. All of these
5. What does a hypervisor do in the context of VMs?
 - A. Manages VM configurations
 - B. Emulates physical hardware
 - C. Provides networking for VMs
 - D. All of these

2.8.1 Answers to Test Your Knowledge

1. A
2. C
3. C
4. C
5. B

Try It Yourself

1. Create your account on Azure Portal.
2. Make changes in appearance of the Azure Portal.
3. Make a new custom dashboard.
4. Create a new Virtual Machine.
5. Add a resource.



SESSION 03

AZURE MANAGEMENT TOOLS

Overview

In this session, various management tools that are part of Microsoft Azure are covered. It describes how to use Azure Command-Line Interface (CLI). It also describes how to use Azure PowerShell and Cloud Shell. Further, the session outlines the benefits of Azure Resource Manager.

Learning Objectives

In this session, students will learn to:

- Describe how to use Azure CLI
- Define Azure PowerShell
- Describe Azure Cloud Shell
- Define Azure Resource Manager

3.1 Azure Command-Line Interface (CLI)

Azure CLI is Microsoft's cross-platform command line utility or platform for managing Azure resources. Developers can use it from their Azure Cloud Shell browser or install it on macOS, Linux, or Windows and then, run it from the command line. The Azure CLI is easy to get started with and is best used to create automation scripts that work with Azure. Resource managers using the Azure CLI can create virtual machines in Azure.

Install Azure CLI on Windows

The current version of Windows-based CLI is 2.57. One can install the Azure CLI by using an MSI or Zip installer available at following link:

<https://learn.microsoft.com/en-us/cli/azure/install-azure-cli-windows?tabs=azure-cli>

This gives access to the CLI through the Windows Command Prompt or PowerShell.

To sign in, run the `az login` command.

If developers have a default browser set, Azure initiates an authorization code flow and opens the browser to load an Azure sign-in page. Developers can sign in there with their Azure credentials.

3.1.1 Features of Azure CLI

Azure CLI provides many great features for managing Azure resources.

CrossPlatform	One can install Azure CLI on Linux, Mac, or Windows systems. Developer can also run the Azure CLI in a Docker container.
Multi-shell environment	Azure CLI runs in multi-shell environments such as Windows Command Prompt, Bash, and PowerShell.
Bashlike syntax	If developer is a Linux administrator, the Azure CLI syntax is similar to writing Bash scripts. These similarities make the Azure CLI more natural and easier to learn.
Automation	One can automate Azure management tasks such as creating, modifying, or deleting resources by scripting using the Azure CLI.
OpenSource	Microsoft hosts the Azure CLI source code on GitHub, and community members can contribute to the project.

3.1.2 Deploy and Configure Resources with CLI

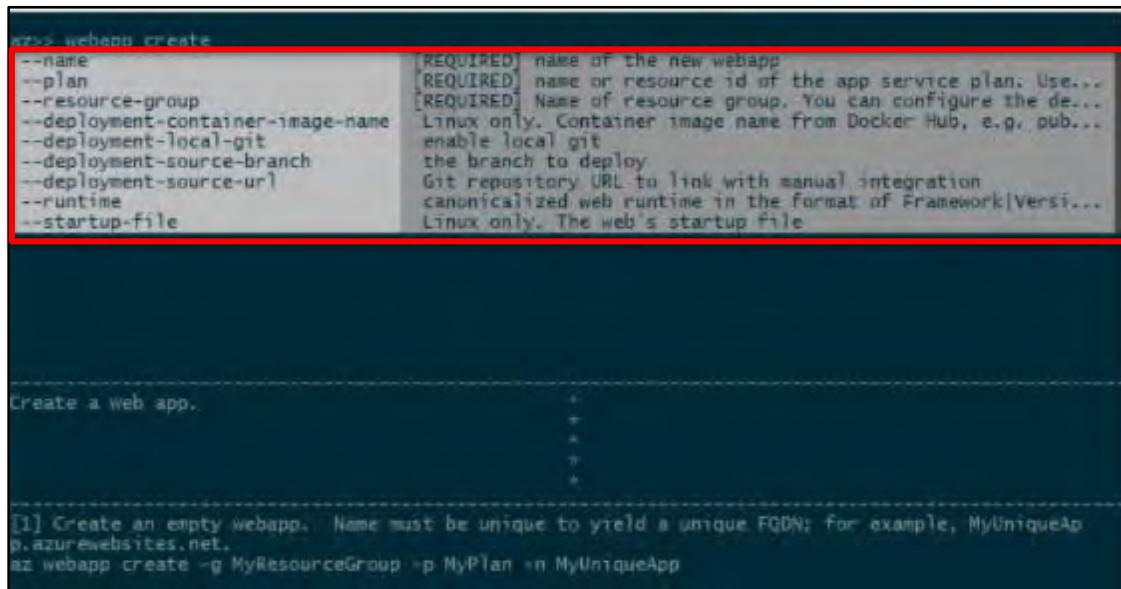
Developers can manage the Resource group setup using the CLI, such as creating a new resource group, deleting a resource group, deploying Azure Resource Manager (ARM), and so on. Table 3.1 lists a few commands that can be used with the Azure CLI.

Commands	Managing Resource Groups
<code>az group create</code>	Create a new resource group.
<code>az group delete</code>	Delete a resource group.
<code>az group deployment</code>	Manage Azure Resource Manager deployments.

Table 3.1: CLI Commands

3.1.3 Using Interactive Mode

Developers can use the Azure CLI interactively by running the `az interactive` command. This mode takes developers to an interactive shell with autocomplete command descriptions. Figure 3.1 shows how the Azure CLI Interactive mode prompts the user with further commands after `webapp create` has been typed.

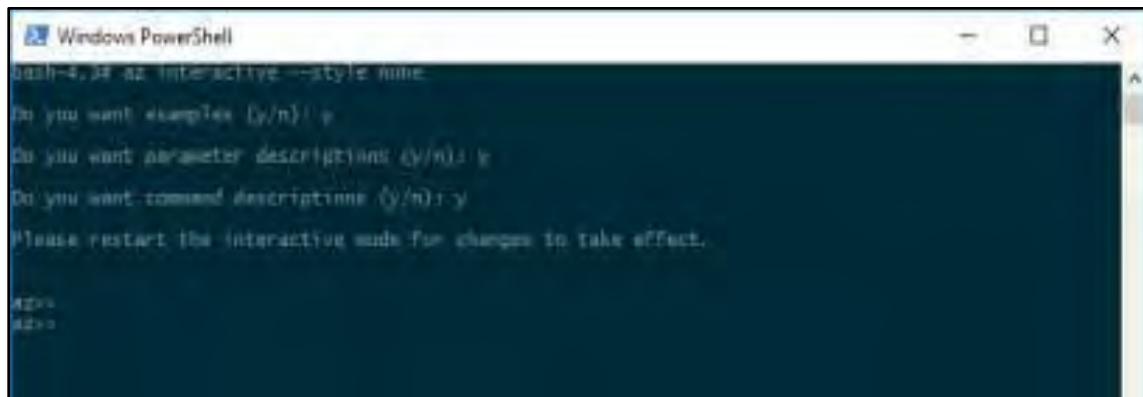


The screenshot shows a terminal window with the Azure CLI. The command `az webapp create` has been entered, and the terminal is displaying detailed help information for each parameter. A red box highlights the parameter descriptions and command examples. The terminal also shows a command example at the bottom:

```
az webapp create -g MyResourceGroup -p MyPlan -n MyUniqueApp
```

Figure 3.1: Azure CLI Interactive

Interactive mode optionally displays command descriptions and more. Parameter descriptions and command examples can be enabled or disabled using F1 key. Refer to Figure 3.2.



The screenshot shows a Windows PowerShell window. The command `az interactive --style none` has been run. The terminal then asks three questions using the F1 key: "Do you want examples? (y/n):", "Do you want parameter descriptions? (y/n):", and "Do you want command descriptions? (y/n):". The user has responded "y" to all three. The terminal concludes with the message "Please restart the interactive mode for changes to take effect." At the bottom, there are two "az>>" prompts.

Figure 3.2: Enabling Command Descriptions

One can toggle display of default options ON or OFF with F2 as shown in Figure 3.3.

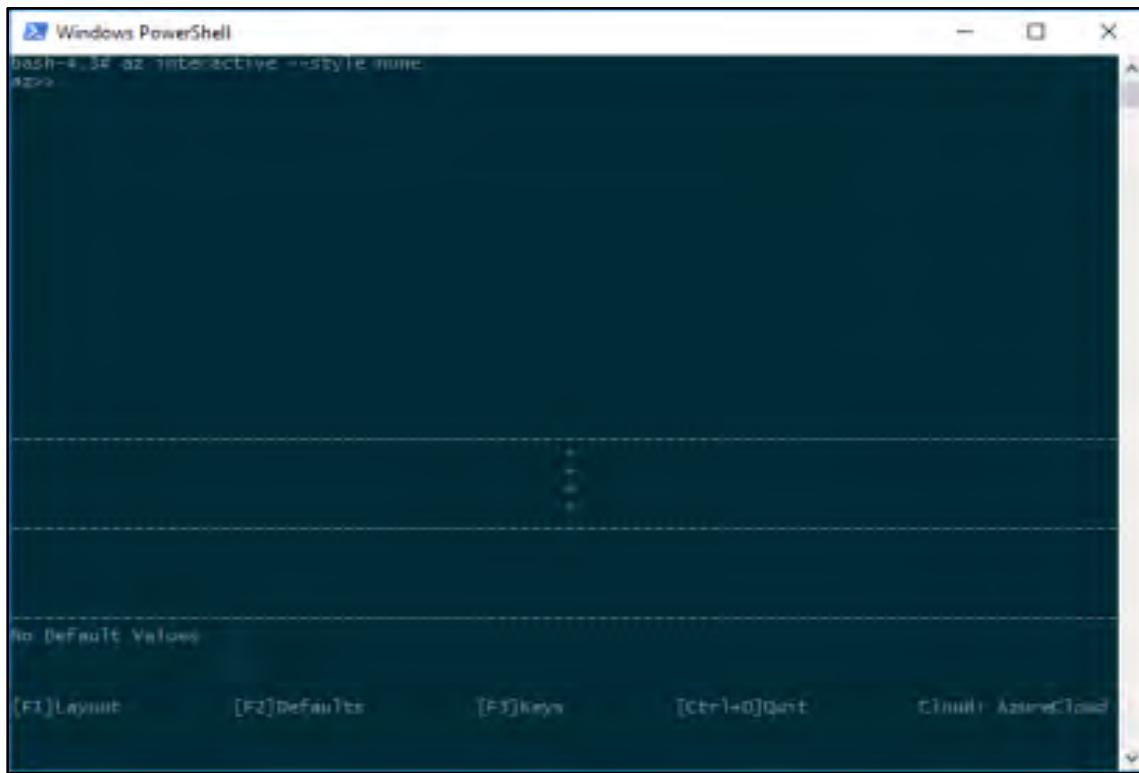


Figure 3.3: Default Options

Clicking F3 results in display of key gestures, as shown in Figure 3.4.

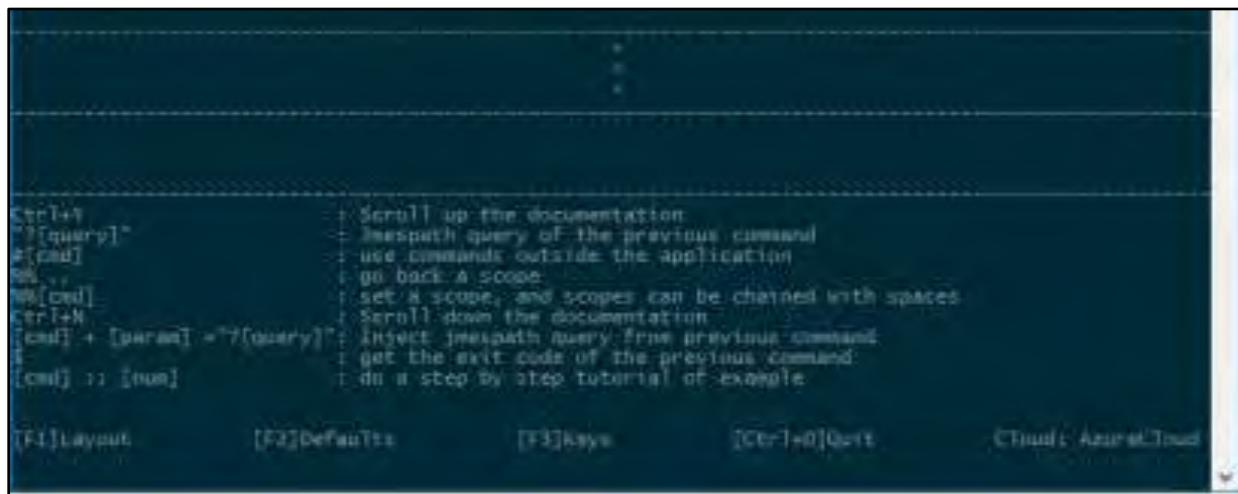


Figure 3.4: Keys

3.2 Azure PowerShell

Azure PowerShell is designed to manage Azure resources from the command line. One can use Azure PowerShell to create automated tools that utilize the Azure Resource Manager model. One can also try using Azure Cloud Shell in the browser or install it on the local machine.

3.2.1 Uses of Azure PowerShell

Azure PowerShell provides a complete set of commands for managing Azure resources through the PowerShell command line. Administrators can use cmdlets to perform complex tasks such as creating virtual machines and managing other hosting resources. Azure PowerShell cmdlets can also be used programmatically, allowing to schedule and automate various tasks.

The naming convention for Azure PowerShell cmdlets is VerbNoun, which is the PowerShell standard. Nouns indicate the resource type and verbs describe the operation (for example New, Get, Set, and Remove) (for example AzVM, AzKeyVaultCertificate, AzFirewall, AzVirtualNetworkGateway). In Azure PowerShell, nouns always start with the letter Az.

3.2.2 Deploy and Configure Resources with PowerShell

Deploying a Bicep file or ARM template requires writing access to the resource being deployed and access to all Microsoft operations. Bicep refers to the Domain-Specific Language (DSL) that uses declarative syntax in order to deploy Azure resources. In a Bicep file, developers can define the infrastructure to be deployed to Azure. They can then use that file throughout the development lifecycle to deploy the infrastructure over and over.

In PowerShell:

To establish to a resource group, utilize New-AzResourceGroupDeployment:

```
New-AzResourceGroupDeployment -ResourceGroupName  
<resource-group-name> -TemplateFile <path-to-template>
```

To establish to a subscription, utilize New-AzSubscriptionDeployment which is also known as the New-AzDeployment cmdlet:

```
New-AzSubscriptionDeployment -Location <location> -  
TemplateFile <path-to-template>
```

To establish to a management group, utilize New-AzManagementGroupDeployment.

```
New-AzManagementGroupDeployment -Location <location> -TemplateFile <path-to-template>
```

To deploy to a tenant, use New-AzTenantDeployment.

```
New-AzTenantDeployment -Location <location> -TemplateFile <path-to-template>
```

3.2.3 Interactive Mode

Table 3.2 lists the types of resources that are useful to find usual commands along with the Azure PowerShell module and noun prefixes to use with Get-Command.

Resource Type	Azure Powershell Module	Noun Prefix
Resource group	Az.Resources	AzResourceGroup
Virtual machines	Az.Compute	AzVM
Storage accounts	Az.Storage	AzStorageAccount
Key Vault	Az.KeyVault	AzKeyVault
Web applications	Az.Websites	AzWebApp
SQL databases	Az.Sql	AzSqlDatabase

Table 3.2: Types of Resources and Azure PowerShell Modules

3.3 Azure Cloud Shell

Azure Cloud Shell is an interactive browser-based shell for managing Azure resources. This gives the flexibility to choose the best shell for the job. Linux users can choose Bash and Windows users can choose PowerShell.

On the Azure portal, click the Cloud Shell symbol as shown in Figure 3.5.



Figure 3.5: Azure Portal

3.3.1 Features (Bash or PowerShell)

Features provided by Azure Cloud Shell include:

- Browser-based shell experience

Cloud Shell provides access to a browser-based CLI built with Azure management in mind. Cloud Shell lets developers work without being tied to their local computer, which only the cloud can provide.

- Choose Preferred Shell

Users can choose between Bash and PowerShell from the Shell drop-down list.

1. Choose Cloud Shell, as shown in Figure 3.6.

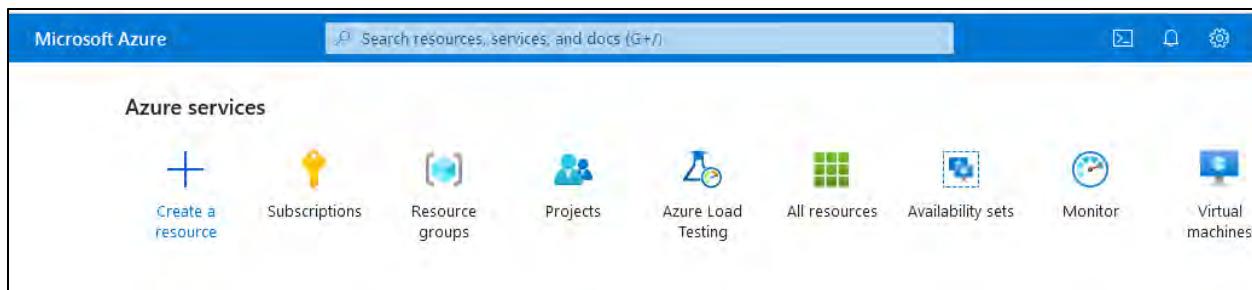


Figure 3.6: Cloud Shell Option on Azure Portal

2. Choose PowerShell as shown in Figure 3.7.

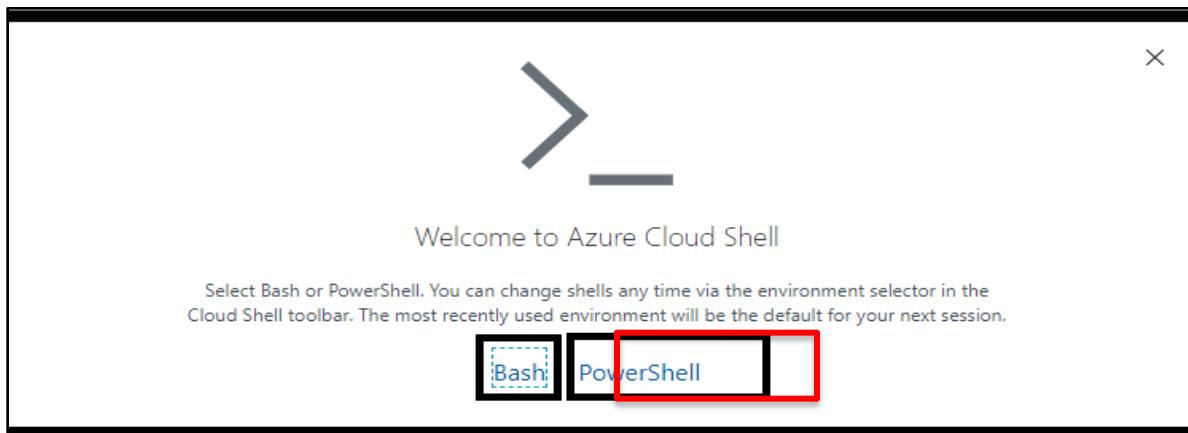


Figure 3.7: Azure Cloud Shell Toolbar

After the first run, developers can switch between Bash and PowerShell using the shell type drop-down as shown in Figure 3.8.



Figure 3.8: Azure Cloud Shell Terminal

- **Secure auto-authentication**

Cloud Shell securely and automatically verifies account access to Azure CLI and Azure PowerShell.

- **Integrated Cloud Shell Editor**

Cloud Shell provides an integrated graphical text editor based on the open-source Monaco editor. Simply execute the code to create and edit the configuration file. Cloud shell is easy to deploy via Azure CLI or Azure PowerShell.

3.4 Azure Resource Manager

Azure Resource Manager is a deployment and management service for Azure. It provides a layer of control to create, update, and delete resources in the Azure account. Secure and organize resources after deployment with management features such as access control, lockout, and tagging.

When a user submits a request in an Azure tool, API, or SDK, the Resource Manager receives the request, authenticates, and authorizes requests. Resource Manager sends a request to an Azure service that performs the requested operation. Since all requests are handled through the same API, developers can see consistent results and functionality across different tools.

All features available in the portal are also available through PowerShell, Azure CLI, REST API, and client SDK.

3.4.1 Benefits

Benefits of using Resource Manager are as follows:

Manage infrastructure with declarative templates rather than scripts.

Instead of handling resources separately, deploy, manage, and monitor each resource as a group.

Redeploy solutions throughout the development lifecycle and ensure that resources are deployed in a consistent state.

Define dependencies between resources to ensure they are deployed in the correct order.

Apply access control to all services because Azure role-based access control (Azure RBAC) is natively integrated into the management platform.

3.4.2 ARM Template Best Practices

There are a few best practices for developers to bear in mind when building ARM templates. These practices can aid in administration and guarantee that one's resources are properly set up.

Define dependencies between resources to ensure they are deployed in the correct order.

Apply access control to all services because Azure Role-Based Access Control (Azure RBAC) is natively integrated into the management platform.

Template limits

Developers should try to keep templates around 4 MB in size, with parameters not greater than 64 KB. These restrictions should be applied to the size of the final template, which should contain any iterative resource declarations, variables, or parameters.

Developers should also note that templates have certain strict limitations that are as follows:

- In a template expression, there are only 24,576 characters
- 64 different output values

- There are 800 resources available (including copy count)
- There are 256 variables and 256 parameters allowed

Resource Groups

Metadata about resources is stored in any resource grouping that developers create. This information is kept in the same place as the group. This implies that until a group region's access is restored, developers would not be able to update or administer those resources.

They should keep a duplicate of metadata in the failover region if one must build a failover for resource groups. Even if the original group is unreachable, one can still have control.

Consider YAML

The JSON format is used for ARM templates. This approach is appropriate for simple templates, but as the number of resources and deployment complexities increase, these templates become less human-readable. Some Azure resources, for example, require over a hundred lines of code to declare, on account of unending nested brackets.

This makes it difficult to decipher the template definitions.

While this format cannot be changed, developers can build their templates in YAML and then, convert them to JSON when ready to launch. YAML may help make templates more readable by allowing developers to specify why they are specifying resources, the way they are. This option to leave comments is handy if one has a large team defining templates.

Avoid Linked Templates Unless Necessary

Azure Resource Manager has a feature called linked templates that allows developers to divide their resources into distinct templates.

After that, all of the templates are connected and assigned to a single deployment source.

This linkage is good; however, it necessitates either public access to templates or access to the deployment workflow. One can provide this latter access using a SAS token, but systems will still be vulnerable.

Furthermore, while working with connected templates, one can only use parameter URIs or parameter objects. Developers cannot mix formats, reducing their ability to manage sensitive information inside the template.

Rather than attempting to work around these limitations, standard templates should be considered. Pipelines may be used to manage deployment and

guarantee that dependent templates are delivered together. This gives the same result as connected templates, but with more security.

3.4.3 Azure Resource Manager with Cloud Volumes ONTAP

On AWS, Azure, and Google Cloud, NetApp Cloud Volumes ONTAP, the premier enterprise-grade storage management solution, provides safe, proven storage management services. Cloud Volumes ONTAP offers up to 368 TB of capacity and a wide range of use cases, including file services, databases, DevOps, and any other corporate application. This is because of a powerful set of capabilities such as high availability, data protection, storage efficiency, Kubernetes integration, and many more.

Cloud Volumes, in particular, ONTAP offers Cloud Manager, the user interface, and APIs for administration, automation, and orchestration, enabling hybrid and multi-cloud architectures and allowing developers to handle storage pools as a single piece in the infrastructure as code setup.

3.5 Summary

- ✓ The Azure Portal is a user-friendly Web interface for managing Azure subscriptions, replacing command-line tools.
- ✓ Users can connect and manage Azure resources using commands, such as Azure CLI, Windows PowerShell, and Azure Cloud Shell.
- ✓ Azure CLI is available for all OS such as Windows, Linux, Mac, and so on.
- ✓ Using the Azure Cloud Shell is the easiest and fastest way to manage Azure resources.
- ✓ Azure PowerShell manages Azure resources from the command line and can create automated tools that utilize the Azure Resource Manager model.
- ✓ With the help of ARM, template developers can deploy and manage one's Azure resources in a simple manner.

3.6 Test Your Knowledge

1. What is the primary role of Azure Portal in Microsoft Azure?
 - A. Graphical Azure service management
 - B. Command-line interface
 - C. Database access
 - D. Infrastructure visualization
2. How does Azure CLI manage Azure services?
 - A. SQL query execution
 - B. Graphical deployment
 - C. Command-line interaction
 - D. Visualization creation
3. What is Azure PowerShell mainly used for?
 - A. Azure database scripting
 - B. Linux-based services
 - C. Managing Azure services via scripts
 - D. Virtualization tool
4. What defines Azure Cloud Shell in Azure?
 - A. Physical server
 - B. Local emulator
 - C. Web-based shell
 - D. Standalone application
5. What best characterizes Azure Resource Manager in Azure?
 - A. Exclusive virtual network tool
 - B. Resource management framework
 - C. Database management system
 - D. Service usage dashboard

3.6.1 Answers to Test Your Knowledge

1. A
2. C
3. C
4. C
5. B

Try It Yourself

1. Create a Resource Group:

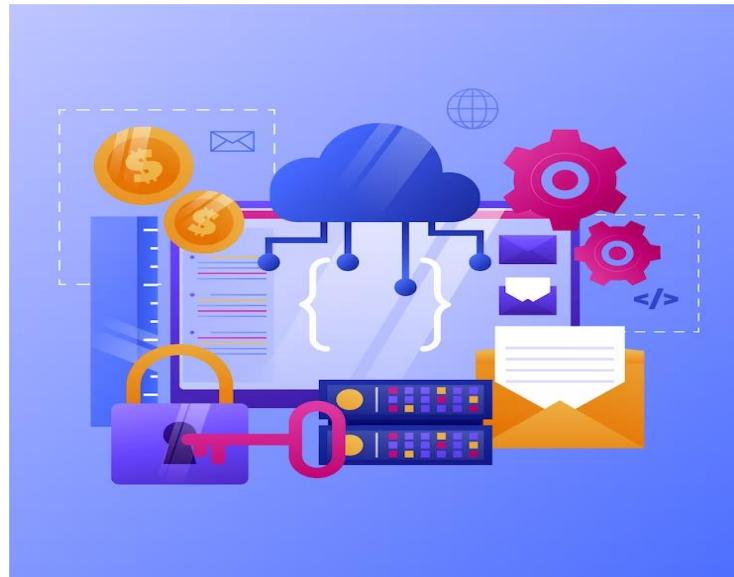
Practice creating a resource group in the Azure portal to organize your resources effectively. Experiment with different names and configurations to understand how resource groups function.

2. Deploy Resources:

Explore deploying various resources such as virtual machines or storage accounts within your resource group. Adjust settings and configurations to learn how to customize deployments to suit your requirements.

3. Check Permissions:

Dive into Azure's permissions settings to understand how access control works. Experiment with assigning roles to users and groups and verify their permissions to ensure they align with your resource management strategy.



SESSION 4

AZURE MONITORING

Learning Objectives

In this session, students will learn to:

- Identify Azure monitoring tools
- Explain the metrics in Azure monitoring tools
- Explain Azure monitoring analytics
- Describe Azure application performance
- Explain altering, log integration, networking, and infrastructure in Azure monitoring

4.1 Introduction to Azure Monitoring

Azure Monitor can monitor these types of resources in Azure, other clouds, or on-premises:

- Applications
- Virtual machines
- Guest operating systems
- Containers including Prometheus metrics
- Databases
- Security events in combination with Azure Sentinel
- Networking events and health in combination with Network Watcher
- Custom sources that use the APIs to get data into Azure Monitor

Figure 4.1 depicts a high level architecture of Azure Monitoring.

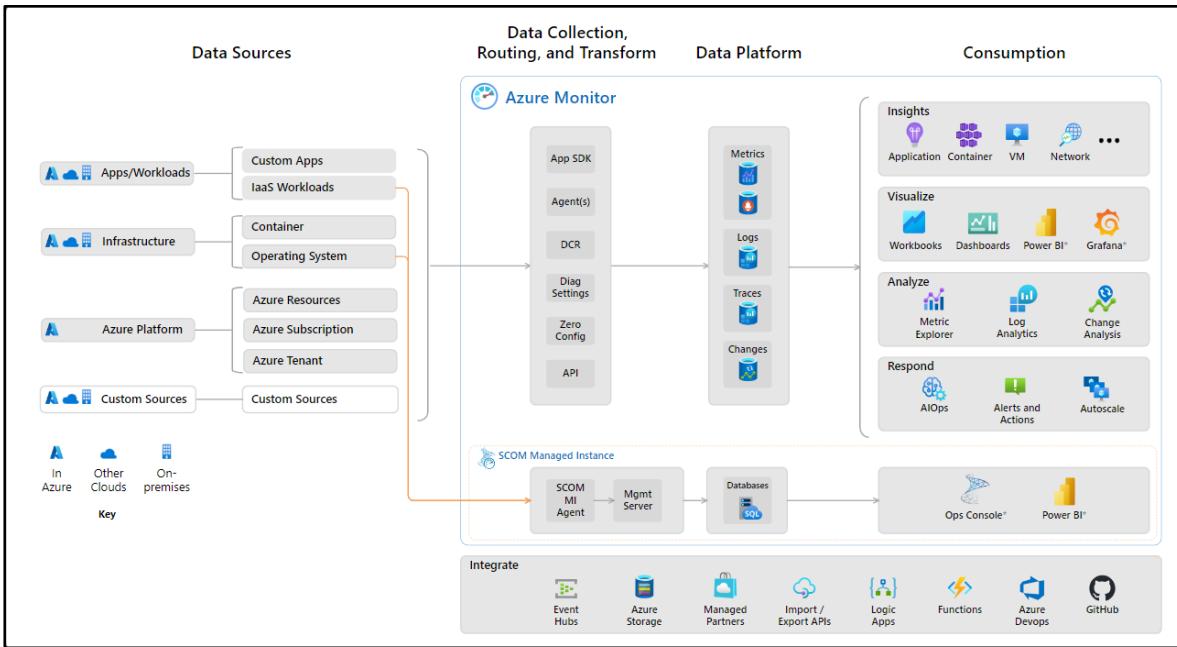


Figure 4.1: High Level Architecture of Azure Monitoring

Azure Monitoring is vital in cloud computing, offering tools to track, manage, and optimize application and infrastructure performance on Azure. Its significance in cloud computing can be understood through several key aspects:

Performance Management: Azure Monitoring allows users to collect metrics and logs from various Azure services and resources. These metrics provide valuable insights into the performance of applications, databases, virtual machines, and other resources. Monitoring performance helps in identifying bottlenecks, optimizing resource utilization, and ensuring efficient operations.

Proactive Issue Detection: By setting up alerts based on predefined thresholds or patterns within monitoring data, Azure Monitoring enables proactive identification of potential issues. This proactive approach allows users to address problems before they impact the performance or availability of services, ensuring a seamless user experience.

Troubleshooting and Diagnostics: The ability to collect and analyze logs from different sources, including applications and infrastructure, facilitates effective troubleshooting and diagnostics. It helps in root cause analysis, identifying the source of issues, and expediting resolution, thereby minimizing downtime and enhancing reliability.

Optimization and Scalability: Monitoring data collected by Azure Monitoring assists in making informed decisions about resource allocation, scaling strategies,

and optimizations. Users can identify underutilized resources or anticipate the requirement for scaling based on performance trends, ensuring efficient resource utilization and cost-effectiveness.

Security and Compliance: Monitoring tools within Azure provide insights into security-related events and anomalies. By monitoring logs and metrics, users can detect potential security threats, unauthorized access, or unusual activities, contributing to a more secure environment. Additionally, it helps in meeting compliance requirements by tracking and auditing activities.

Cost Management: Azure Monitoring offers insights into resource usage and associated costs. By monitoring resource metrics, users can optimize allocation, cut costs, and improve cloud cost management.

Service Health and Reliability: Azure Monitoring includes features such as Azure Service Health that provide real-time information about the status of Azure services, planned maintenance, or service incidents. This information helps users stay informed about potential impacts on their resources and take necessary precautions to maintain service reliability.

4.2 Metrics in Azure Monitoring Tools

In Azure Monitoring, metrics play a pivotal role in tracking, assessing, and managing the performance, health, and behavior of various Azure resources. Metrics provide quantitative data points that offer insights into the utilization, availability, and overall status of these resources.

The step-by-step process to create a Metrics chart is described as follows:

- 1) Developers must launch the Azure portal (portal.azure.com) and log in with their credentials. This will take developers to Azure Portal as shown in Figure 4.2.

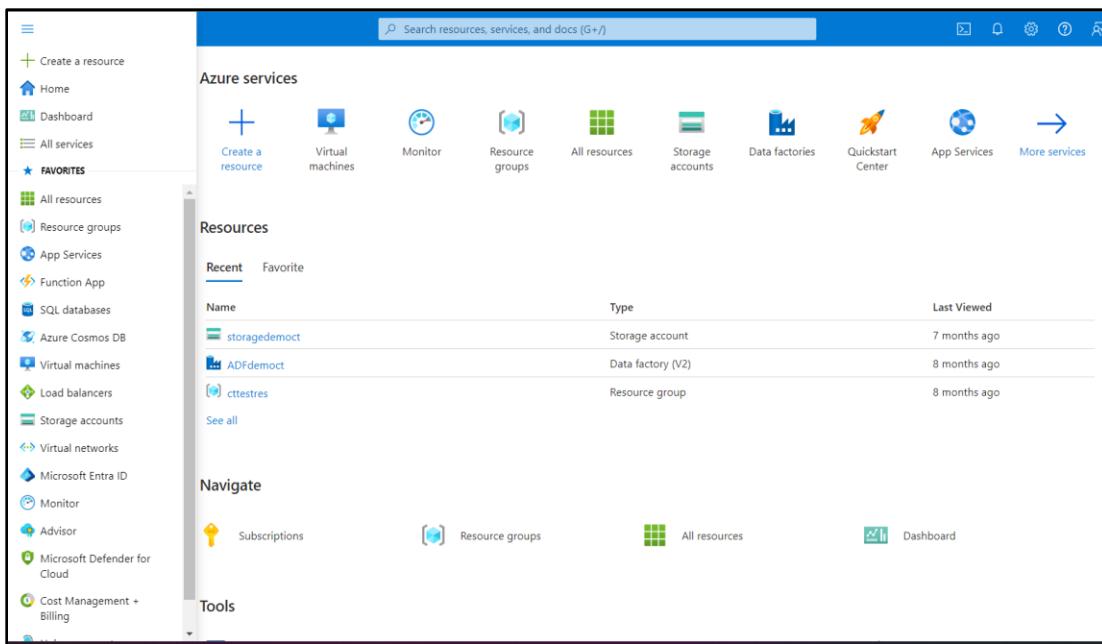


Figure 4.2: Azure Portal

- 2) Navigate to Azure Monitor service by clicking **Monitor**. This will display the Monitor Services Page as shown in Figure 4.3.

The screenshot shows the 'Monitor | Overview' page. The left sidebar has sections for 'Overview', 'Activity log', 'Alerts', 'Metrics', 'Logs', 'Change Analysis', 'Service health', and 'Workbooks'. Under 'Insights', there are sections for 'Applications', 'Virtual Machines', 'Storage accounts', 'Containers', 'Networks', 'SQL (preview)', 'Azure Cosmos DB', and 'Key Vaults'. The main content area has tabs for 'Overview' (selected) and 'Tutorials'. It features a 'Curated monitoring views' section with cards for 'Application insights', 'Container Insights', 'VM Insights', and 'Network Insights'. Below this is a 'Detection, triage, and diagnosis' section with cards for 'Metrics', 'Logs', 'Alerts', and 'Workbooks'. A note at the top right says: 'The Log Analytics agents, used by VM Insights, won't be supported as of August 31, 2024. Plan to migrate to VM Insights on Azure Monitor agent prior to this date.' A 'Diagnostics Settings' link is also present.

Figure 4.3: Azure Monitor Services Page

- 3) In the Azure Monitor menu, click **Metrics** to view the metrics available for the Azure resources (Refer to Figure 4.4).

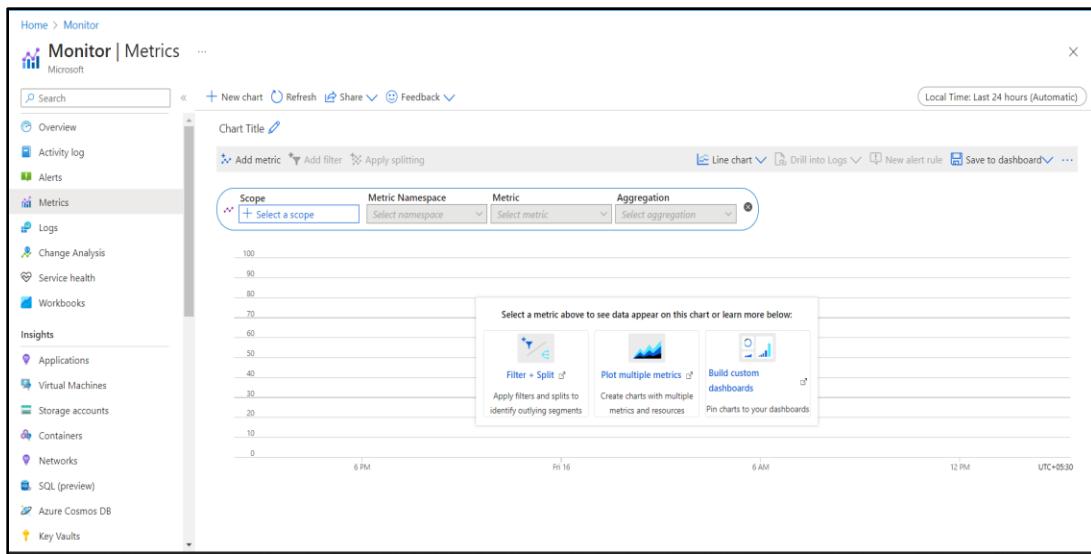


Figure 4.4: Metrics Page

- 4) Click **Select a scope**, then select the Resource type and location (Refer to Figure 4.5).

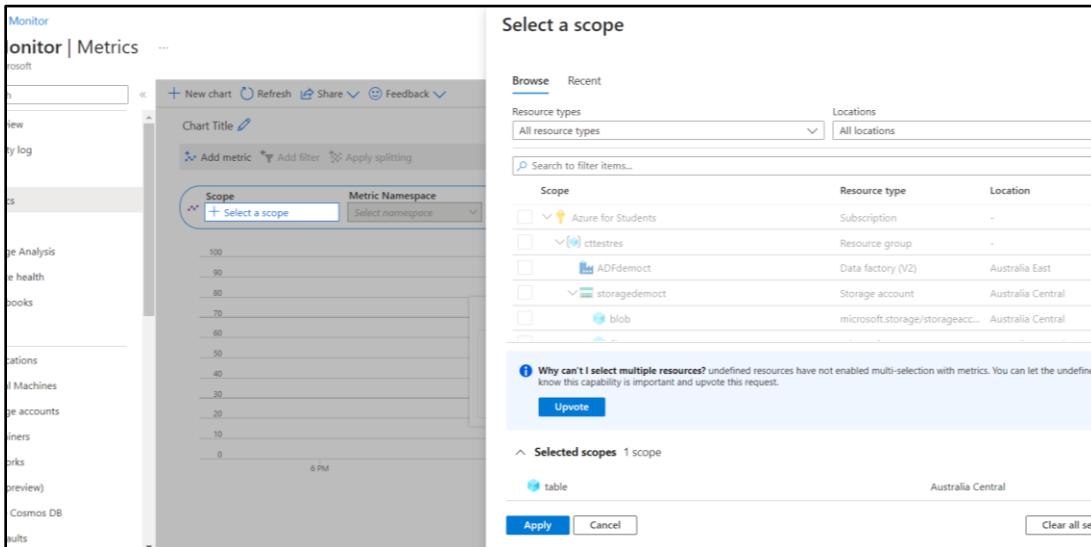


Figure 4.5: Select a Scope

- 5) Click **Apply** and the metric chart will appear as per the filter, selected resource type, and location. Refer to Figure 4.6.

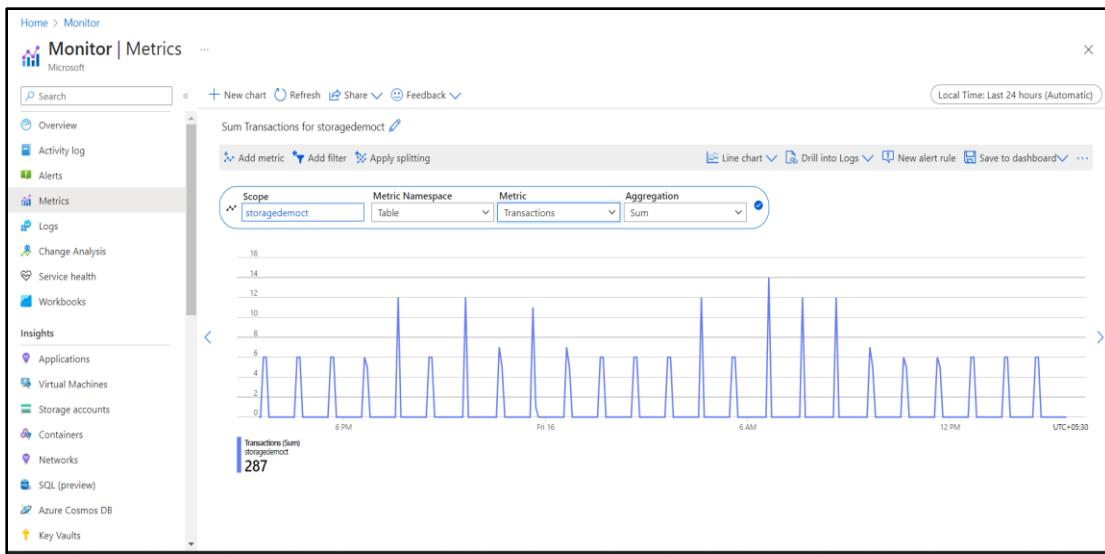


Figure 4.6: Azure Metrics Chart

4.2.1 Metrics Types

Azur Monitor Metrics in Microsoft Azure includes various types of metrics, each tailored for specific monitoring objectives.

- **Native Metrics:** These metrics utilize Azure Monitor's tools for analysis and alerting, delivering immediate insights directly within the Azure platform. These can be further classified into:
 - **Platform Metrics:** Azure services generate platform-specific metrics. For instance, compute resources such as Virtual Machines provide metrics on CPU usages, memory utilization, disk I/O, and network traffic.
 - **Custom Metrics:** Users can define and collect custom metrics specific to their applications or resources to monitor unique performance indicators.
- **Prometheus metrics:** Prometheus metrics are gathered from Kubernetes clusters, including Azure Kubernetes Service (AKS), and utilize industry-standard tools such as PromQL and Grafana for analysis and alerting.

4.2.2 Data Collection

Data collection in Azure Monitoring can be categorized as:

Azure Monitor

This centralizes the collection of metrics across Azure services. It continuously gathers metric data from Azure resources and stores it for analysis and monitoring.

Resource Providers

Each Azure service has its own set of metrics. Resource providers within Azure collect and expose these metrics to Azure Monitor.

4.2.3 Metric Granularity and Retention

Metrics can be collected at various intervals, ranging from seconds to hours, depending on the resource and metric type. Users can configure the granularity based on their monitoring requirements.

Azure retains metric data for a specified duration, allowing users to analyze historical trends and performance patterns. Retention periods may vary for different services.

4.2.4 Visualization and Analysis

Azure Monitoring offers visualization tools for creating user-friendly charts and dashboards that display metrics. These visualizations help monitor real-time performance and track trends over time, as discussed in the previous section 4.2.

Users can analyze metric data, spot anomalies, and derive insights by examining trends, patterns, and correlations. This analysis helps in making informed decisions about resource optimization and performance improvements.

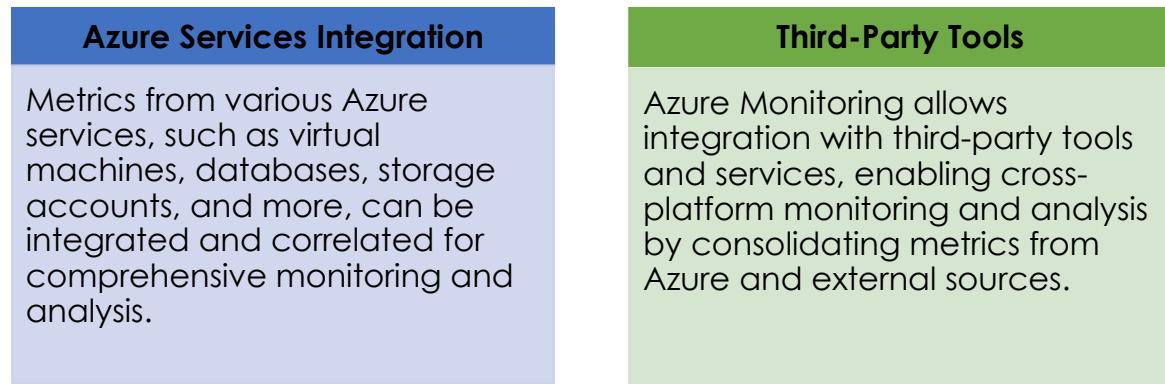
4.2.5 Alerting and Automation

Azure Monitoring allows users to define alert rules based on metric thresholds or conditions. When a metric breaches the set threshold, it triggers alerts to notify users or automated actions to address potential issues.

Actions can be automated based on metrics, such as scaling resources up or down, triggering remediation scripts, or initiating other workflows to maintain system health.

4.2.6 Integration with Azure Services

Azure Monitoring also supports integration as follows:



4.3 Azure Monitoring – Analytics

Figure 4.7 depicts components of Application Insights.

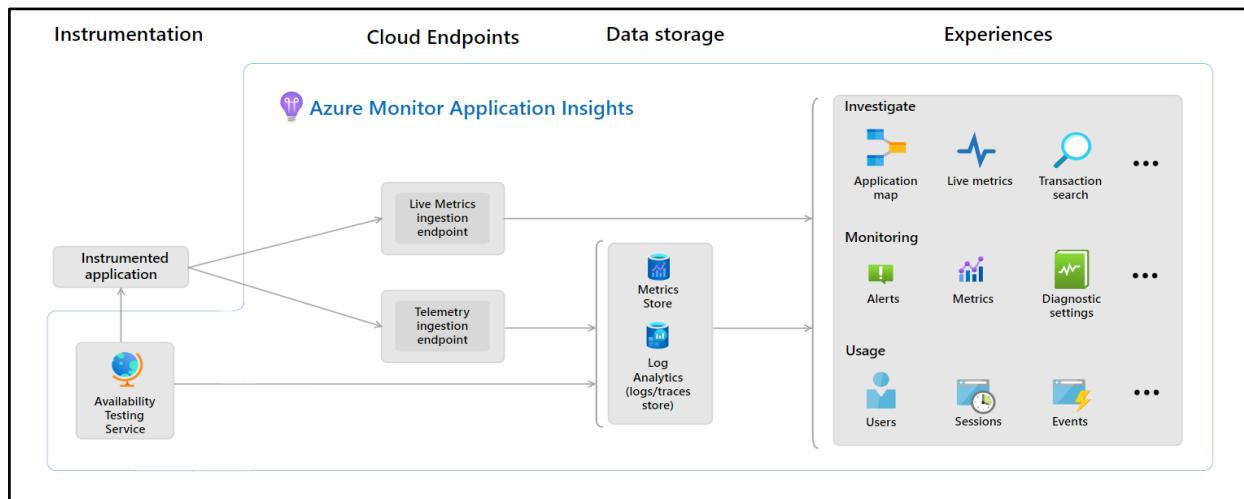


Figure 4.7: Components of Application Insights

Azure Monitoring Analytics refers to the analytical capabilities provided within Azure Monitor for processing and gaining insights from log and metric data collected from various Azure resources. It encompasses several features and tools designed to analyze and derive valuable insights from monitoring data.

4.3.1 Log Analytics

Azure Monitor utilizes Log Analytics as a part of its analytics suite. Log Analytics provides a powerful query language, Kusto Query Language (KQL), enabling users to query and analyze large volumes of log data collected from Azure resources, applications, and custom logs. This allows for in-depth analysis, troubleshooting, and trend identification.

The step-by-step process for log analytics is described as follows:

- 1) Sign in to Azure portal and navigate to Azure Monitor services. The Monitor Services Page is displayed. Refer to earlier Figures 4.2 and 4.3.
- 2) In the Azure Monitor menu, click **Logs** as shown in Figure 4.8.

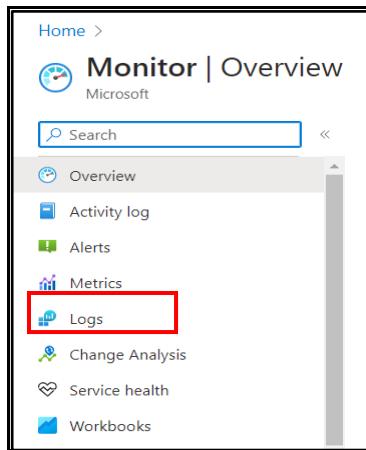


Figure 4.8: Logs Section in Azure Monitor Services Page

- 3) This will lead to Queries Page as shown in Figure 4.9.

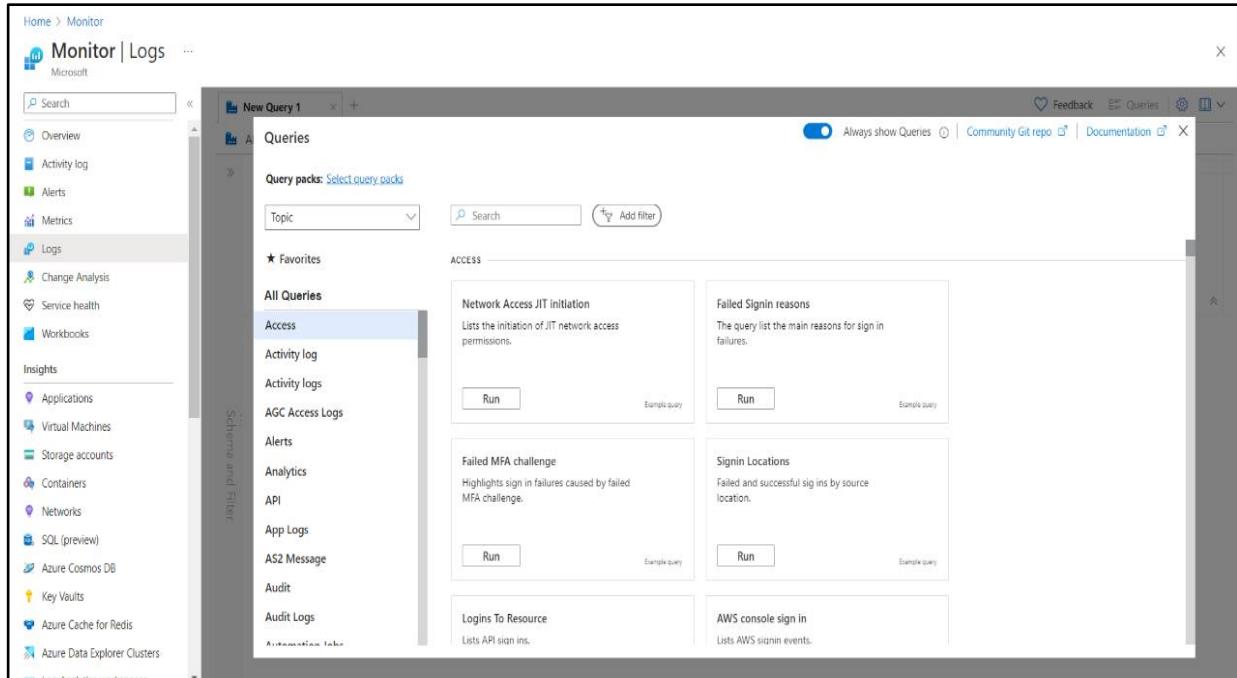


Figure 4.9: Queries Page

- 4) Search the query and load it to the editor by clicking **Load to editor** as shown in Figure 4.10.

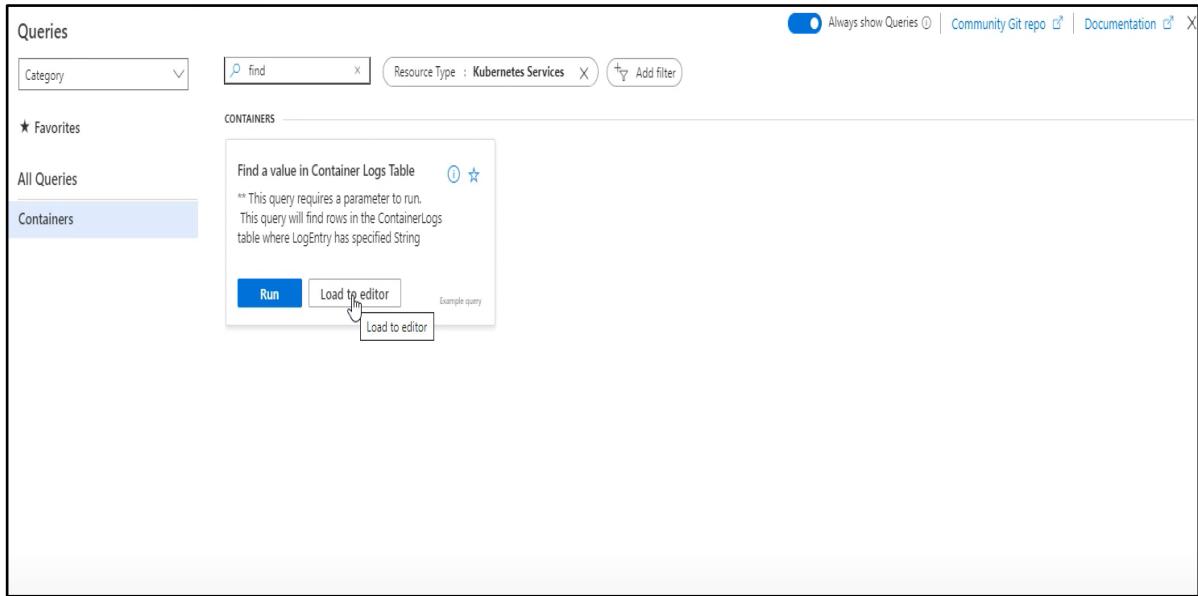
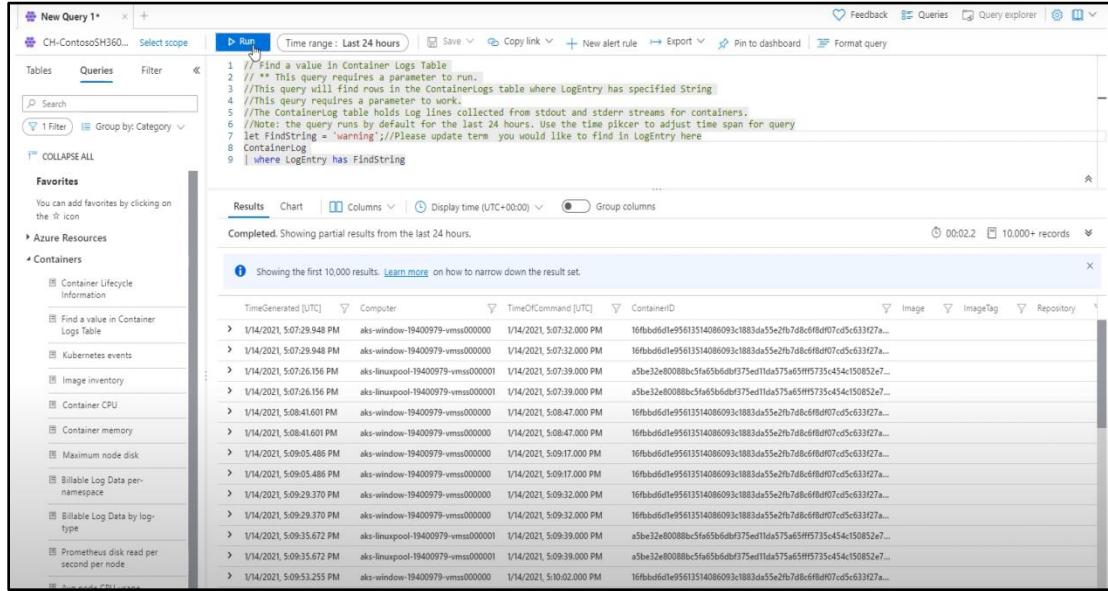


Figure 4.10: Query Load to Editor

- 5) Edit or write the query by using Kusto Query Language (KQL). Then, click **Run** as shown in Figure 4.11.



The screenshot shows the Azure Kusto Query Editor interface. At the top, there's a toolbar with buttons for Run, Save, Copy link, New alert rule, Export, Pin to dashboard, and Format query. The Run button is highlighted with a yellow box. Below the toolbar, the query text is displayed:

```

1 // Find a value in ContainerLogs Table
2 // This query will result in a table to run.
3 // This query will find rows in the ContainerLogs table where LogEntry has specified String.
4 // This query requires a parameter to work.
5 // The ContainerLogs table holds log lines collected from stdout and stderr streams for containers.
6 // Note: the query runs by default for the last 24 hours. Use the time picker to adjust time span for query
7 let FindString = "warning"; //Please update term you would like to find in LogEntry here
8 ContainerLog
9 | where LogEntry has FindString

```

Below the query text, the Results section shows the first 10,000 results. The table has columns: TimeGenerated [UTC], Computer, TimeOfCommand [UTC], ContainerID, Image, ImageTag, and Repository. The results list various log entries from AKS nodes and Linux pools, each with a timestamp, node name, command time, container ID, image, tag, and repository.

Figure 4.11: Edit or Write Query

- 6) The result of the query will appear. Then, analyze the log by applying some filters and time range.

4.3.2 Custom Queries and Filters

Users can create custom queries using KQL to filter, aggregate, and correlate log data. These queries can extract specific information, identify patterns, perform statistical analysis, and visualize results to derive meaningful insights from log data.

4.3.3 Advanced Analytics

Azure Monitoring Analytics offers advanced analytical capabilities, including machine learning-based anomaly detection. It can automatically detect irregular patterns or deviations from normal behavior within log data, aiding in proactive issue identification and resolution.

4.3.4 Visualizations and Dashboards

The analytics capabilities within Azure Monitoring allow users to visualize query results through interactive charts, graphs, and dashboards. These visualizations provide a clear representation of trends, patterns, and anomalies within log data, aiding in easier interpretation and decision-making.

4.3.5 Integration with Other Azure Services

Azure Monitoring Analytics integrates seamlessly with other Azure services such as Azure Sentinel (cloud-native SIEM), Azure Data Explorer, Azure Machine Learning,

and more. This integration allows for advanced analytics, correlation of data across services, and leveraging additional AI/ML capabilities for deeper insights and threat detection.

4.3.6 Cross-Resource Analysis

Users can perform cross-resource analysis by querying and correlating data from multiple Azure resources or services. This capability helps in identifying dependencies, understanding system-wide behaviors, and troubleshooting complex issues that span across multiple components.

4.3.7 Alerting and Automation

Analytics within Azure Monitoring can be used to create queries that trigger alerts based on specific conditions or patterns found in log data. These alerts can be configured to notify users or trigger automated responses for timely intervention.

Azure Monitoring Analytics empowers users to perform in-depth analysis of log data, enabling proactive monitoring, troubleshooting, performance optimization, and security threat detection within Azure environments. It forms a crucial part of Azure Monitor, providing the tools required to extract actionable insights from the vast amount of data generated by Azure resources.

4.4 Azure Application Performance

Azure Monitoring offers a specialized focus on application performance through a service called 'Azure Application Insights'. This service is integrated into Azure Monitoring and is specifically designed to monitor and analyze the performance and behavior of applications hosted on Azure.

4.4.1 Application insights

Application insights can be useful for following tasks:

Performance Monitoring

Azure Application Insights collects telemetry data, including request rates, response times, failure rates, and dependencies (such as calls to databases or external services), providing a comprehensive view of application performance.

Gaining Code-Level Insights

Application Insights offers insights into the application's code behavior, identifying performance bottlenecks, exceptions, and slow dependencies.

Developers can trace and analyze the code path to diagnose issues more effectively.

Availability Tests

Azure Application Insights includes features to perform availability tests, checking endpoints at regular intervals to ensure that applications are responsive and available to users.

End-to-End Transaction Tracing

Application Insights enables end-to-end transaction tracing, allowing users to track a user request across various components and services to identify where performance issues might occur.

User Behavior Analysis

Application Insights tracks user interactions, user flows, and usage patterns within the application, providing insights into how users interact with the application and potential performance impacts on user experience.

Performance Impact Analysis

Performance Impact Analysis correlates user behavior with application performance metrics to understand the impact of user actions on the application's overall performance.

Alert Rules

Users can set up alerts based on application performance metrics, such as response time exceeding a defined threshold. This allows for proactive identification and resolution of performance issues.

Diagnostics and Profiling

Application Insights supports diagnostics and profiling tools that enable deeper analysis of application behavior, performance, and exceptions, aiding in rapid troubleshooting.

Integration with DevOps

Application Insights integrate seamlessly with various development tools, including Visual Studio and Azure DevOps, allowing developers to access performance data during the development lifecycle for continuous improvement.

Custom Telemetry

Application Insights allow users to track custom telemetry data, enabling monitoring of specific application behaviors or metrics relevant to unique business requirements.

Integration with Azure Services

It integrates with other Azure services, enabling comprehensive monitoring by correlating application performance data with metrics from other Azure resources.

4.5 Alerting

Alerting within Azure Monitoring enables users to set up notifications based on defined conditions or thresholds, allowing proactive identification and timely response to potential issues or anomalies in the Azure environment.

4.5.1 Types of Alerts

Metric-Based Alerts

Users can create alerts based on specific metric values exceeding or falling below predefined thresholds. For instance, CPU usage exceeding 90% for more than five minutes can trigger an alert.

Log-Based Alerts

Alerts can also be configured based on log data using queries in the KQL. Developers can define alert rules based on log analytics results to detect specific conditions or patterns within log data.

Activity Log Based Alerts

Activity Log based Alerts are based on auditing logs that track all actions occurring on resources. Activity log alerts are triggered based on specific conditions when new activity log events occur. Resource Health alerts and Service Health alerts are types of activity log alerts that provide notifications about the health status of the services and resources.

Smart Detection Alerts

Smart detection on an Application Insights resource automatically detects potential performance issues and failure anomalies in the developer's Web application, providing warnings. Developers can utilize smart detection on their Application Insights resource to establish alert rules for various smart detection modules.

Prometheus Alerts

Prometheus alerts are utilized to trigger notifications based on Prometheus metrics stored in Azure Monitor managed services for Prometheus. The alert rules are defined using the PromQL open-source query language.

4.5.2 Alert Conditions and Criteria

Alert conditions and criteria are of two types:

Thresholds

Users can set thresholds for metrics or log queries, specifying when an alert should be triggered. These thresholds can be based on values such as CPU usage, response times, error rates, and so on.

Time Windows

Users can define the duration for which a metric is required to breach the threshold to trigger an alert, avoiding false positives due to temporary spikes.

A breakdown of the alerting mechanism is now explained.

4.5.3 Alert Rules Creation

The step-by-step process to create an Alert Rule is described as follows:

- 1) Sign in to the Azure portal and navigate to Azure Monitor service by clicking **Monitor**. The Monitor Services page is displayed. Refer to earlier Figures 4.2 and 4.3.
- 2) In the Azure Monitor menu, click **Alerts** and an alert page will appear as shown in Figure 4.12.

The screenshot shows the Azure Monitor | Alerts page. The left sidebar lists various monitoring services: Overview, Activity log, Alerts (selected), Metrics, Logs, Change Analysis, Service health, Workbooks, Insights (Applications, Virtual Machines, Storage accounts, Containers, Networks, SQL (preview), Azure Cosmos DB, Key Vaults), and Prometheus rule groups. The main content area has a search bar and filters for Subscription (Azure for Students), Time range (Past 24 hours), Alert condition (Fired), Severity (all), and Add filter. Below these are summary counts for Total alerts (0), Critical (0), Error (0), Warning (0), Informational (0), and Verbose (0). A large central message says "No alerts found" with a speech bubble icon containing an exclamation mark. At the bottom, it says "Try changing your search or choose a different scope level if you don't see what you're looking for."

Figure 4.12: Alert Page

- 3) Click **Create** and select the Alert rule as shown in Figure 4.13.

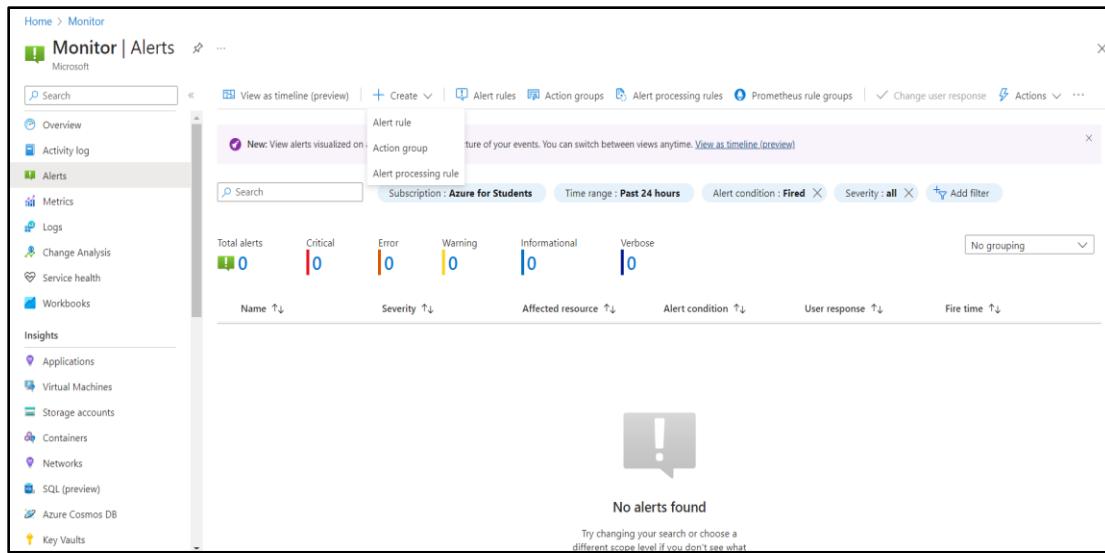


Figure 4.13: Select Alert Rule

- 4) Click **Select Scope** and select the Resource type and location. Then, click **Apply**. Refer to Figure 4.14.

Figure 4.14: Select a Resource

- 5) Click **Condition** and Select a suitable signal. Then, click **Apply** as shown in Figure 4.15.

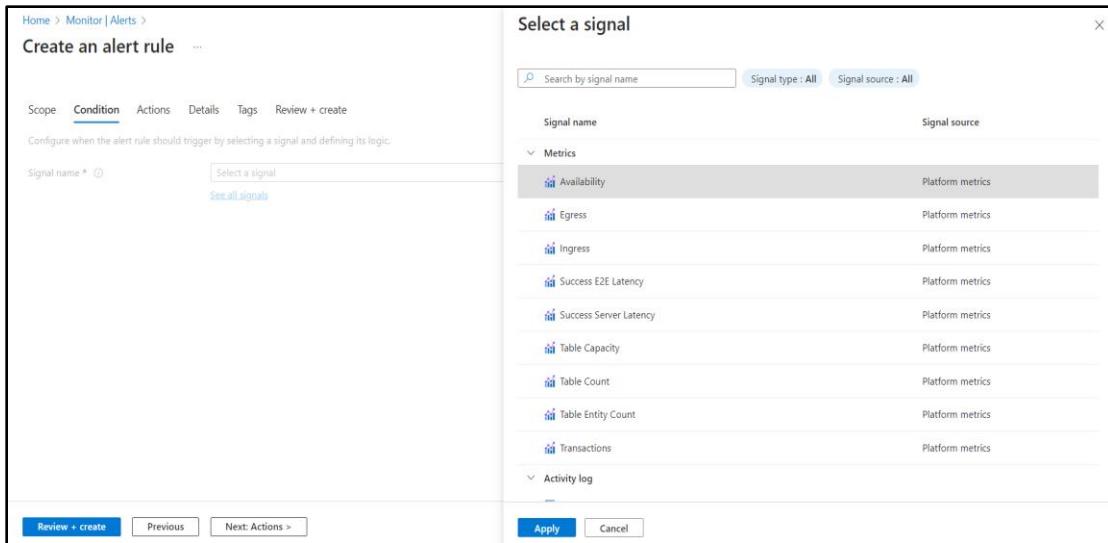


Figure 4.15: Select a Signal

6) Set the Alert logic and set Split by dimension. Refer to Figure 4.16.

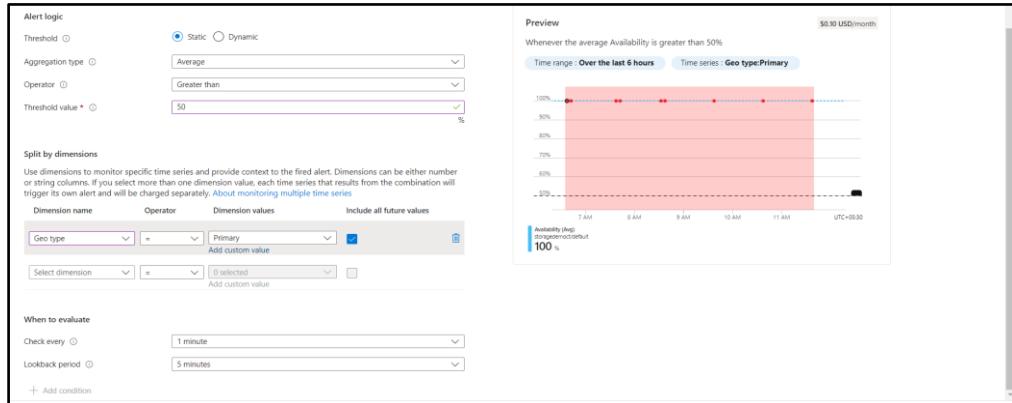


Figure 4.16: Set Alert Logic and Set Split by Dimensions

7) Click **Action** and the action tab will open where developer can select or create the required action groups. Refer to Figure 4.17.

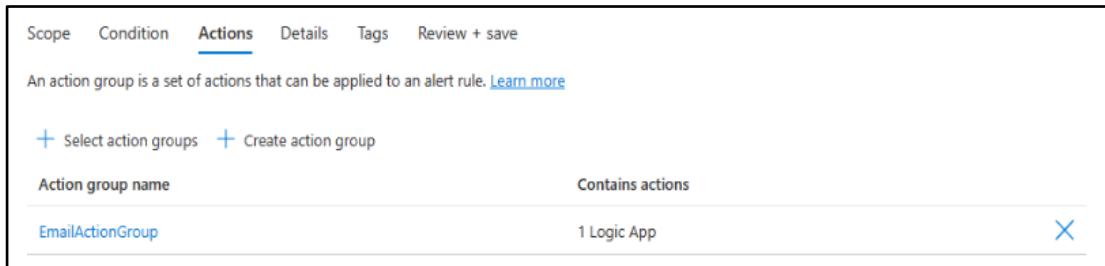


Figure 4.17: Action Page

- 8) Click **Details** and the Details tab will open, where developer can define the project details and Alert rule details as shown in Figure 4.18.

The screenshot shows the 'Details' tab of the Azure portal for creating an alert rule. The top navigation bar includes 'Scope', 'Condition', 'Actions', 'Details' (which is underlined in blue), 'Tags', and 'Review + create'. The main area is divided into two sections: 'Project details' and 'Alert rule details'. In 'Project details', there is a dropdown for 'Subscription' set to 'Azure for Students' and a dropdown for 'Resource group' set to 'cttestres', with a 'Create new' link below it. In 'Alert rule details', there is a dropdown for 'Severity' set to '3 - Informational', a text input for 'Alert rule name' containing 'Test Rule', and a text input for 'Alert rule description' which is empty. Below these fields is a collapsed section titled 'Advanced options'. At the bottom of the page are three buttons: 'Review + create' (in blue), 'Previous', and 'Next: Tags >'. The entire interface has a light gray background with white and light blue UI elements.

Figure 4.18: Details Page

- 9) Click **Tags**. The tag section opens where you can set any required tags on the alert rules resources. Refer to Figure 4.19.

The screenshot shows a 'Tags' page with the following interface elements:

- Top navigation:** Scope, Condition, Actions, Details, **Tags**, Review + create.
- Text description:** Tags are name and value pairs that enable you to categorize resources and view consolidated billing by applying the same tag to multiple resources and resource groups. [Learn more about using tags](#)
- Note:** Note that if you later change resource settings on other tabs, your tags will be automatically updated.
- Table header:** Name ⓘ and Value ⓘ.
- Table body:** A single row with empty input fields for Name and Value.
- Bottom navigation:** Review + create (highlighted in blue), Previous, Next: Review + create >.

Figure 4.19: Tags Page

- 10) Click **Review + create**, then, click **Create**. Refer to Figure 4.20.

The screenshot shows the 'Create an alert rule' interface in the Azure portal. The top navigation bar includes 'Home > Create an alert rule ...'. Below the navigation are tabs: Scope, Condition, Actions, Details, Tags, and Review + create (which is underlined). A summary section on the left lists 'Metric alert rule', '1 Condition', and links to 'Terms of use' and 'Privacy statement'. To the right, it shows 'Total pricing' at '0.10 USD/month' with a 'Pricing' link. The main content area is divided into sections: 'Scope' (Resource: Azure for Students > cttestres > storagedemoct/table), 'Condition' (Signal name: Availability, Geo type: Include * Greater than Average, Threshold value: 50, Lookback period: 5 minutes, Check every: 1 minute), 'Details' (Project details: Subscription: Azure for Students, Resource group: cttestres, Region: global), and Alert rule details: Alert rule name: Test Rule, Alert rule description: 3 - Informational. At the bottom are 'Create' and 'Previous' buttons.

Figure 4.20: Review + Create Page

4.5.4 Notification and Action Options

Notification and Action Options in Azure are now explained here.

Notification Channels

Azure supports various notification channels such as email, SMS, Webhook, Azure Application Insights, Azure Logic Apps, and so on. Users can choose their preferred channel to receive alerts.

Action Groups

Action groups define a set of notifications or actions to take when an alert is triggered. This could include sending notifications to specific individuals or groups, invoking automated remediation scripts, scaling resources, or initiating other workflows.

4.5.5 Integration with Monitoring Services

Integration with Monitoring Services in Azure includes two types:

Azure Services Integration

Alerting in Azure Monitoring integrates with various Azure services, allowing users to set up alerts for different resources such as Virtual Machines, Azure SQL Databases, Azure App Service and so on, based on their specific metrics.

Cross-Resource Alerts

Users can create alerts that span multiple Azure resources, enabling a holistic approach to monitoring and alerting across interconnected components.

4.5.6 Alert Lifecycle Management

Alert Lifecycle Management in Azure comprises alert monitoring and alert history and insights.

Alert Monitoring

Users can monitor active alerts, review their status, and acknowledge them within the Azure portal or through APIs, ensuring they are aware of ongoing issues.

Alert History and Insights

Azure Monitoring maintains a history of alerts triggered, acknowledged, or resolved, allowing users to review past incidents for analysis and improvement.

4.5.7 Dynamic Thresholds and Machine Learning

Dynamic Thresholds and Machine Learning are useful for:

➤ Dynamic Thresholds

Some Azure services offer dynamic thresholds that adapt to changing patterns and behavior, allowing for more accurate alerting based on current conditions rather than static thresholds.

➤ Machine Learning-Based Alerts

Azure also provides machine learning-based anomaly detection for certain metrics, automatically identifying abnormal behavior and triggering alerts.

4.6 Log Integration

In Azure, log integration refers to the process of collecting, storing, analyzing, and acting upon log data generated by various Azure resources, applications, and services. Azure provides several tools and services to facilitate log integration.

4.6.1 Azure Monitor

Application of Azure Monitor includes the following:

Log Analytics

Azure Monitor leverages Log Analytics as a centralized platform for collecting and analyzing log data. It supports ingestion of logs from diverse sources, including Azure services, custom applications, operating systems, and third-party sources.

Log Data Collection

Azure Monitor collects logs in various formats such as text, JSON, and XML. It allows users to configure log data ingestion from different Azure resources and services into Log Analytics workspaces.

4.6.2 Azure Log Analytics

Azure Log Analytics includes the following:

Data Sources

Log Analytics can collect logs from Azure resources such as Virtual Machines, Azure App Service, Azure Kubernetes Service (AKS), Azure SQL Database, Azure Active Directory, and more. It also supports non-Azure sources through agents or custom log ingestion methods.

Log Query Language

Log Analytics uses Kusto Query Language (KQL), enabling users to query log data efficiently, perform complex analytics, filter logs based on specific criteria, and derive actionable insights.

4.6.3 Log Integration Tools

There are various Log Integration Tools such as:

Azure Monitor Agents	APIs and SDKs
Azure provides agents that can be installed on virtual machines or servers to collect and send logs to Log Analytics. These agents facilitate log integration from on-premises environments and hybrid setups.	Azure offers APIs and software development kits (SDKs) allowing developers to programmatically send log data to Log Analytics from custom applications, services, or third-party tools.

4.6.4 Log Retention and Storage

Retention Policies and Storage and Archiving are two areas under Log Retention and Storage.

Retention Policies	Storage and Archiving
<p>Users can define retention policies in Log Analytics to specify how long log data should be retained. This helps in compliance with regulatory requirements and enables historical analysis.</p>	<p>Azure provides options to archive log data to cost-effective storage solutions such as Azure Blob Storage for long-term retention while still allowing querying and analysis.</p>

4.6.5 More Aspects Under Azure Log Analytics

Table 4.1 lists more aspects under Azure Log Analytics.

Parameters		Description
Visualization and Analysis	Log Analytics Workbooks	Users can create custom dashboards and workbooks in Log Analytics to visualize log data through charts, tables, and custom views, facilitating easy interpretation and analysis.
	Alerting and Insights	Log data analyzed in Log Analytics can trigger alerts based on specific log queries or patterns, allowing proactive identification of issues or anomalies.
Security and Compliance	Security Analytics	Log integration enables monitoring and analysis of security-related logs, aiding in threat detection, incident response, and compliance adherence.
	Auditing and Governance	Log integration supports auditing and governance

Parameters		Description
		by providing visibility into user activities, resource changes, and operational logs for compliance purposes.

Table 4.1: More Aspects under Azure Log Analytics

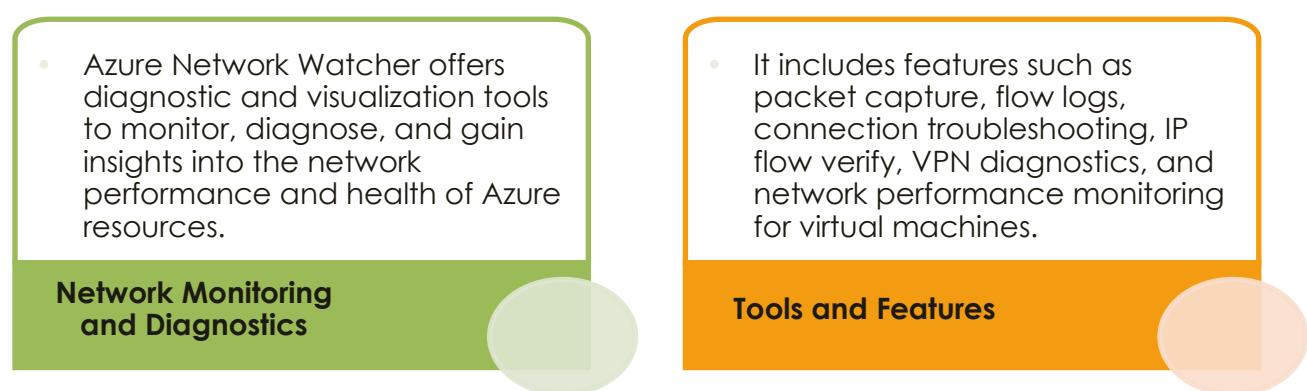
4.7 Networking

Networking monitoring within Azure involves overseeing the performance, security, and health of networking components and services to ensure reliable connectivity and optimal functioning.

Azure provides various tools and services for monitoring networking aspects:

4.7.1 Azure Network Watcher

Azure Network Watcher is used for the following:



4.7.2 Network Performance Monitoring

Azure Monitoring collects network-related metrics, including latency, throughput, and packet loss, providing insights into the performance of virtual networks, subnets, and network interfaces. These metrics can be accessed and visualized through Azure Monitor, allowing users to create custom dashboards and set up alerts based on network performance thresholds.

Azure Network Watcher enables the collection of flow logs, which capture network traffic data (source, destination, ports, protocols) for analysis and visualization. Flow logs can be utilized for security analysis, detecting anomalies, and identifying potential security threats or unusual network behavior.

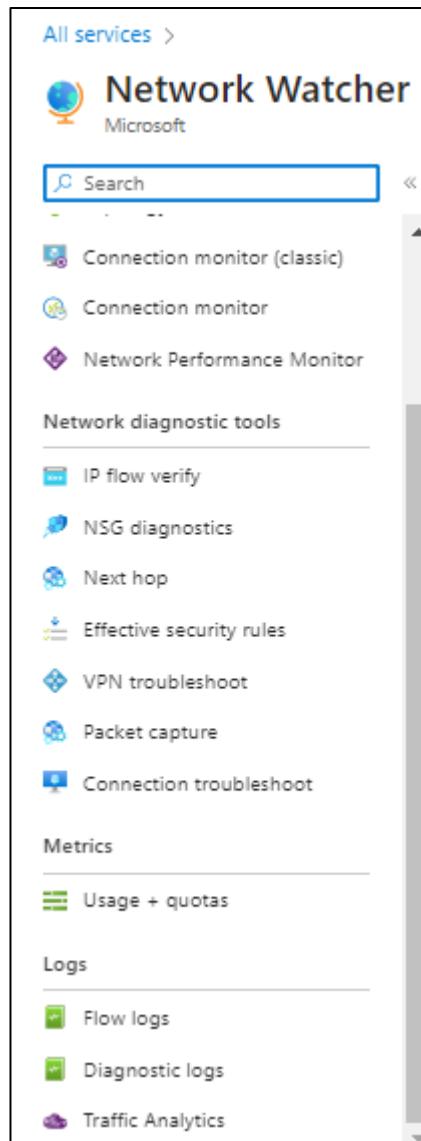


Figure 4.21: Network Watcher

4.7.3 Virtual Network Gateways

Azure also provides insights into the performance and health of VPN gateways and ExpressRoute connections, allowing monitoring of connectivity and data transfer between on-premises and Azure resources.

4.8 Infrastructure

In Azure Monitoring, infrastructure monitoring refers to the process of overseeing and managing the health, performance, and availability of the underlying resources and components that make up an Azure environment. This includes various services, virtual machines, storage, databases, and more. Table 4.2 lists infrastructure monitoring that is handled within Azure.

Parameters		Description
Azure Resource Health	Resource Status and Health	Azure provides Resource Health services, offering real-time visibility into the health and availability status of Azure services and resources. It helps identify whether the issues are caused by the underlying Azure platform or user configurations.
Azure Resource Metrics	Performance Metrics	Azure Monitoring collects and maintains metrics for various Azure resources. These metrics include CPU usage, memory utilization, disk I/O, network traffic, and other performance indicators. Users can visualize and analyze these metrics to monitor resource performance and health.
	Azure Monitor Integration	These metrics are integrated into Azure Monitor, allowing users to create custom dashboards, set up alerts based on specific thresholds, and conduct historical analysis to optimize resource performance.
Virtual Machine Monitoring	Performance Insights for virtual machines	Azure Monitoring provides detailed performance insights, including CPU usage, disk performance, memory utilization, and network activity. Users can analyze these metrics to optimize VM performance.
	Integration with Insights and Logs	Virtual machine diagnostics can be enhanced by integrating with Azure Application Insights and Log Analytics for deeper insights into application performance and system behavior.
Storage and Database Monitoring	Storage Metrics	Azure storage services offer metrics related to throughput, latency, and capacity utilization. Monitoring these metrics helps ensure efficient data storage and access.
	Database Metrics	Azure databases, such as Azure SQL Database or Cosmos DB, provide performance metrics related to query performance, connections, and resource utilization. Monitoring these metrics aids in optimizing database performance and scalability.
Automation and Remediation	Azure Automation	Using Azure Automation, users can set up automated actions and remediation

Parameters		Description
		workflows triggered by specific infrastructure events or alerts, ensuring proactive response to issues.

Table 4.2: Infrastructure Monitoring

4.9 Summary

- ✓ Azure Monitoring is vital for cloud, tracking, managing, and optimizing performance, health, and availability.
- ✓ Metrics offer granularity and retention options, with visualization and analysis tools for real-time monitoring.
- ✓ Azure Monitoring Analytics utilizes Log Analytics with Kusto Query Language for log and metric data analysis.
- ✓ Azure Application Insights focuses on application performance, providing insights into performance, availability, user behavior, and integration with development tools.
- ✓ Networking, infrastructure, and automation aspects are also covered in Azure Monitoring.

4.10 Test Your Knowledge

1. What do Azure monitoring tools primarily focus on?
 - A. Resource creation
 - B. Performance, availability, and health
 - C. Cost management
 - D. User authentication
2. What do metrics in Azure monitoring tools represent?
 - A. Billing information
 - B. Server locations
 - C. Data encryption levels
 - D. Performance and behavior data points
3. What is the primary goal of Azure monitoring analytics?
 - A. Resource provisioning
 - B. Predictive maintenance
 - C. User authentication
 - D. Invoice generation
4. What aspects are typically monitored for Azure application performance?
 - A. Weather conditions
 - B. Stock market trends
 - C. Response times, resource utilization, and error rates
 - D. Political events
5. In Azure monitoring, what does alerting primarily involve?
 - A. Encrypting data
 - B. Setting up notifications based on predefined conditions
 - C. Code optimization
 - D. User authentication processes

4.10.1 Answers to Test Your Knowledge

1. B
2. D
3. B
4. C
5. B

Try It Yourself

1. Monitor Resource Metrics, Choose any resource from your Azure environment, such as a Virtual Machine or a Storage Account. What is the average CPU Usage of your selected resource in the last 24 hours?
2. In the Azure Portal, find and access the Log Analytics service. What are the top three log entries based on a specific condition (such as, status or severity) for your chosen resource type?
3. In the Azure Portal, create an alert rule and set up alerts for critical metrics.



SESSION 05

DATA ACCESS MECHANISM IN AZURE

Overview

This session provides an overview of Entity Framework (EF) which is used for connecting to and working with a database. The session also covers the design approach of Entity Framework and describes Language Integrated Query (LINQ). Finally, the session covers complex queries in LINQ.

Learning Objectives

In this session, students will learn to:

- Describe Entity Framework
- Define database operations in entities
- Explain Query Data and complex queries in LINQ

5.1 Entity Framework

It provides a set of tools and libraries that enable developers to work with relational databases using .NET objects. EF aims to simplify the process of database interaction by allowing developers to work with data in terms of objects and classes, rather than directly dealing with database-specific SQL queries.

Table 5.1 summarizes the version history of EF.

Year	Version (Along with Key New Features)
2008	EF 1.0 was released as part of .NET Framework 3.5 SP1.
2010	EF 4.0 was released (POCO, foreign keys, lazy loading, templates).
2011	EF 4.1 was released (DbContext and code first).
2012	EF 4.3 was released (migrations). It also became open source in this year. EF 5.0 was released (enums, spatial, and Table Valued Functions)
2013	EF 6.0 was released (async, connection resiliency, custom conventions, and sprocs).
2014	EF 6.1 was released.
2016	EF7 was renamed to EF Core. EF Core 1.0 was released (mixed-eval, shadow state properties, unique constraints, sequences, batching, and attach graph APIs). EF Core 1.1 was released (memory-optimized tables).
2017	EF Core 2.0 was released (table splitting, owned entities, global query filters, and function mapping). EF 6.2 was released.
2018	EF Core 2.1 was released (lazy loading, value converters, keyless entity types, group by, constructor parameters, and seeding). EF Core 2.2 was released (spatial, owned collections, and query tags).
2019	EF Core 3.0 was released (Azure Cosmos DB, await foreach, nullable reference types, single server-eval query, and interceptors). EF 6.3 was released (.NET Core). EF 6.4 was released. EF Core 3.1 was released.
2020	EF Core 5.0 was released (many-to-many, TPT, collations, TVFs, filtered include, property bags, table rebuilds, exclude from migrations, and change-tracking proxies).

Year	Version (Along with Key New Features)
2021	EF Core 6.0 was released (perf improvements, compiled models, migration bundles, temporal tables, and pre-convention configuration).
2022	EF Core 7.0 was released (TPC, JSON columns, value objects, bulk updates, templates, sprocs, conventions, trimming support, entity splitting, group by improvements, raw SQL improvements, and migrations improvements).
2023	EF Core 8.0 was released (support for raw SQL queries for unmapped types, lazy-loading improvements, and support for TimeOnly and DateOnly SQL Server data types).

Table 5.1: Entity Framework Version History

EF increases productivity by reducing the task of tracking data utilized in applications. It serves as the fundamental means of interaction between relational databases and Microsoft .NET applications. It makes mapping easier between software objects and a relational database's tables and columns. An Object-Relational Mapping (ORM) helps to build database connections and execute commands. It also helps with query results and automatically creates results as application objects. Further, an ORM enables tracking modifications made to objects. When required, these modifications can persist in the database. EF is a powerful and versatile ORM framework developed by Microsoft for .NET applications.

EF supports various types of queries, which are converted into SQL queries for underlying databases. One of these types is Language-Integrated Query (LINQ). LINQ is one of the most powerful features that was first introduced in .NET Framework 3.5. LINQ can be used with C# or Visual Basic to query different data sources.

5.1.1 Key Features of EF

Some key features of EF are given as follows:

- **ORM:** EF allows developers to work with data in terms of strongly-typed .NET objects. It automatically maps these objects to the corresponding database tables. This abstraction makes it easier for developers to interact with databases using familiar object-oriented programming paradigms.
- **Database Independence:** Entity Framework supports various database engines, including Microsoft SQL Server, MySQL, PostgreSQL, and more.

This enables database-independent coding, facilitating easy database switches or multi-database support in applications.

- **Code-First and Database-First Approaches:** EF supports both Code-First and Database-First development approaches. In Code-First, developers define the data model using C# or VB.NET classes. The database schema is automatically generated based on these classes. In Database-First, the database schema is defined first, and then EF generates the corresponding .NET classes.
- **LINQ Integration:** Entity Framework integrates seamlessly with LINQ. It allows developers to write queries against the data model using a strongly-typed syntax. This enhances code readability and maintainability.
- **Change Tracking:** EF automatically tracks changes made to objects. It can generate the appropriate SQL statements to update the database accordingly. This simplifies the process of persisting changes to the database.
- **Lazy Loading and Eager Loading:** EF supports lazy loading, which means that related data is loaded from the database only when it is accessed. It also supports eager loading, where developers can specify in advance which related data should be loaded along with the main entity.
- **Migrations:** Entity Framework includes a migration feature that helps manage changes to the database schema over time. Migrations enable developers to evolve the database schema along with the application's code.

Figure 5.1 depicts where actually the Entity Framework fits in an application.

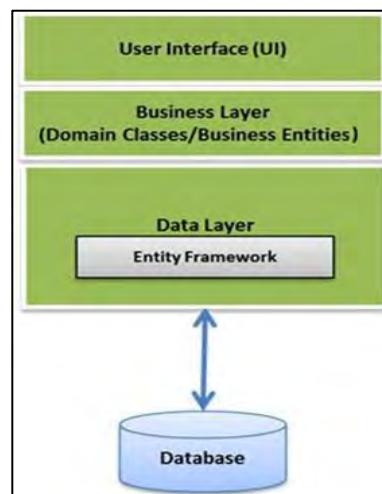


Figure 5.1: ASP.NET Multi-tier Application with Entity Framework

5.1.2 Entity Framework Architecture

Figure 5.2 shows the overall architecture of EF.

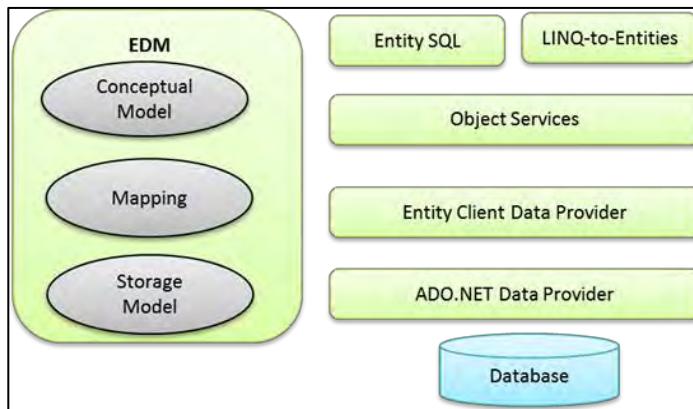


Figure 5.2: Entity Framework Architecture

Entity Data Model (EDM):

EDM consists of three primary parts, which are as follows:

1. Conceptual Model: The conceptual model comprises model classes and their relationships. It is not connected to the database table design.
2. Mapping: Mapping consists of how the storage model is mapped to the conceptual model.
3. Storage Model: The storage model is the database design model which comprises tables, stored procedures, views, keys, and their relationships.

LINQ-to-Entities:

LINQ is a collection of extension methods for classes that implement `IEnumerable` and `IQueryable` interfaces. LINQ-to-Entities (L2E) creates queries for the object model. The query returns entities described in the conceptual model.

Entity SQL:

Entity SQL is a different query language for EF 6 only, which resembles L2E.

Object Services:

An object service is a primary entry point for retrieving information from the database and returning the same. It manages the translation of information from the entity client data provider to the entity object structure.

Entity Client Data Provider:

The entity client data provider translates L2E or Entity SQL queries to an SQL query so that the database can comprehend the query. It converses with the ADO.NET data provider, which retrieves data from the database.

ADO.NET Data Provider:

The ADO.NET Data Provider converses with the database using ADO.NET.

5.1.3 How does Entity Framework Work?

EF is capable of mapping database schema to the domain entity classes. It interprets and performs LINQ queries on SQL. It also tracks modifications made to entities and saves changes in the database. Following steps indicate how EF works in an application:

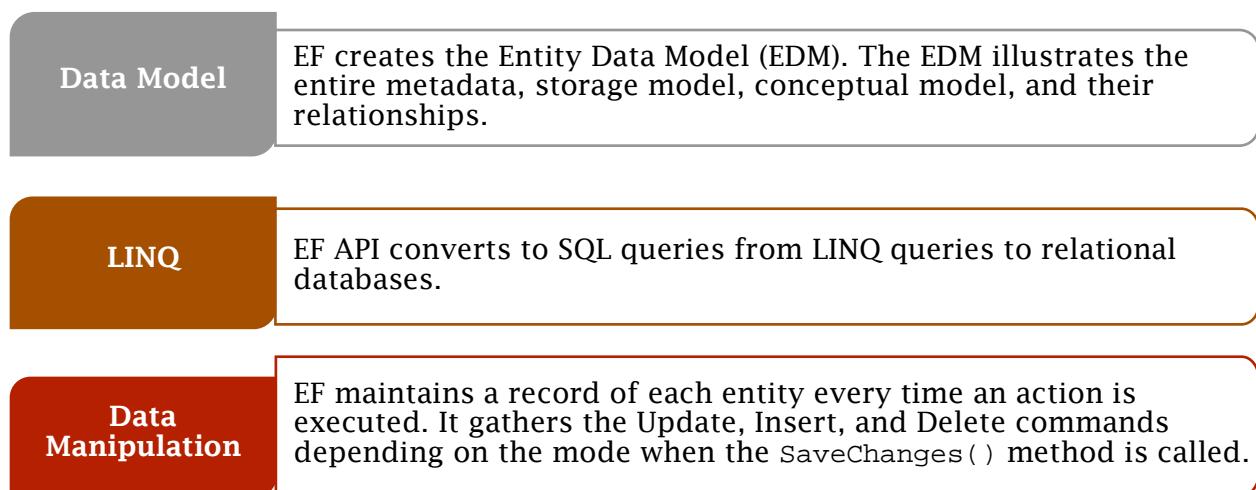


Figure 5.3 shows the Entity Framework API workflow.

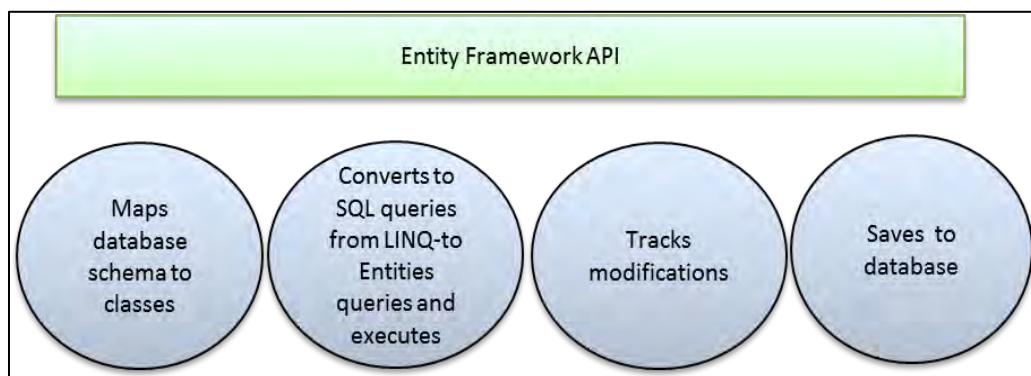


Figure 5.3: Entity Framework Workflow

5.2 Design Approach in Entity Framework

EF offers three methods to build an entity model. Each method has its own advantages and disadvantages.

EF supports three design approaches namely, code-first, model-first, and database-first. Each of them has their own uses.

5.2.1 Code-First Approach

With the Code-First approach, the database is generated from the entity class. A Model Designer is not required for this approach. When entity classes are ready to use the DbContext object, the database is generated depending on the entity and the relationship between them. In this approach, entities are known as Plain Old CLR Objects (POCO) Entities.

Code-First consists of domain classes. The domain classes are items of the business domain. EF has a context, which manages the interaction between classes and the database. Code-First adds a model builder, which examines the classes managed by the context. It uses a set of rules to determine relationships between classes, define a model, and map that model to the database. It also determines how that model should be mapped to the database. Code-First can also update the database using a feature called Code-First migration.

5.2.2 Model-First Approach:

The EF Designer is used for the Model-First approach. Using this approach, a user can build entity models and the relationship between them. The database is created based on these models, depending on their relationship. Model-First is suitable for new projects where the database has not been created. The model is first designed in an EF designer, then the SQL query is generated. This, in turn, creates a database schema to match the model, and the SQL is executed to create the schema in the database. The EDMX file saves the model, which can be modified and displayed in the EF Designer. The classes are created from the EDMX file and one can interact with these classes in an application.

5.2.3 Database-First Approach:

On using the Database-First approach, the model or entity is generated based on the existing database. The database is designed before any code is created. The Database-First approach is recommended for projects where the database already exists, especially for large applications with extensive data. In some scenarios, associations, relationships, and naming conventions may not align with user preferences. In such cases, the Code-First approach is recommended.

The Database-First approach (an alternative to Code-First and Model-First approaches) builds model codes, including properties, classes, and DbContext. These model codes are created from the existing database and the classes act as a link between the controller and the database. When using this approach, EF is first created from the existing database. All functionalities such as database/model sync and generation of code are the same as in the Model-First approach.

5.3 Database Operation in Entity Framework

EF is a well-planned and executed ORM tool. It is available with .NET Framework and has quite a few years of stabilization and feature development. As an ORM, EF reduces disparity between object-oriented and relational worlds. Developers are enabled to write applications interacting with data present in relational databases. This is done using well-typed .NET objects representing the application's domain. The requirement for a huge chunk of data access 'plumbing' that used to be performed earlier, is now eliminated.

EF implements rich mapping capabilities, including support for:

One-to-one relationships, one-to-many relationships, and many-to-many relationships

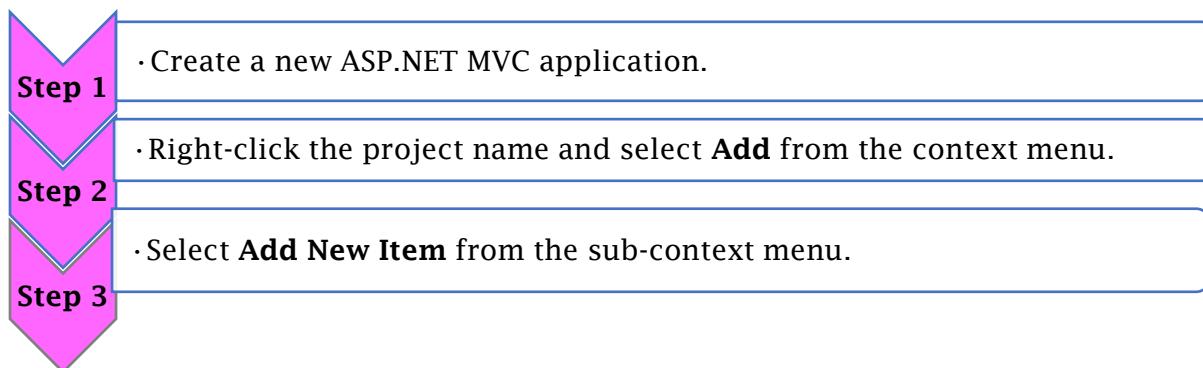
Inheritance - table per type, table per concrete class, and table per hierarchy

Stored procedures

Complex types

5.3.1 Connecting to a Database and Creating an Entity Data Model

Following are the steps to connect to Azure SQL Database from Entity Framework:



- Step 4 • Select **ADO.NET Utility Model** from the **Add New Item** window.
- Step 5 • Select **ADO.NET Entity Data Model** from the template list and name it as **StudentDBModel**.
- Step 6 • Click **Add**. This displays the **Entity Data Model Wizard**, as shown in Figure 5.4.

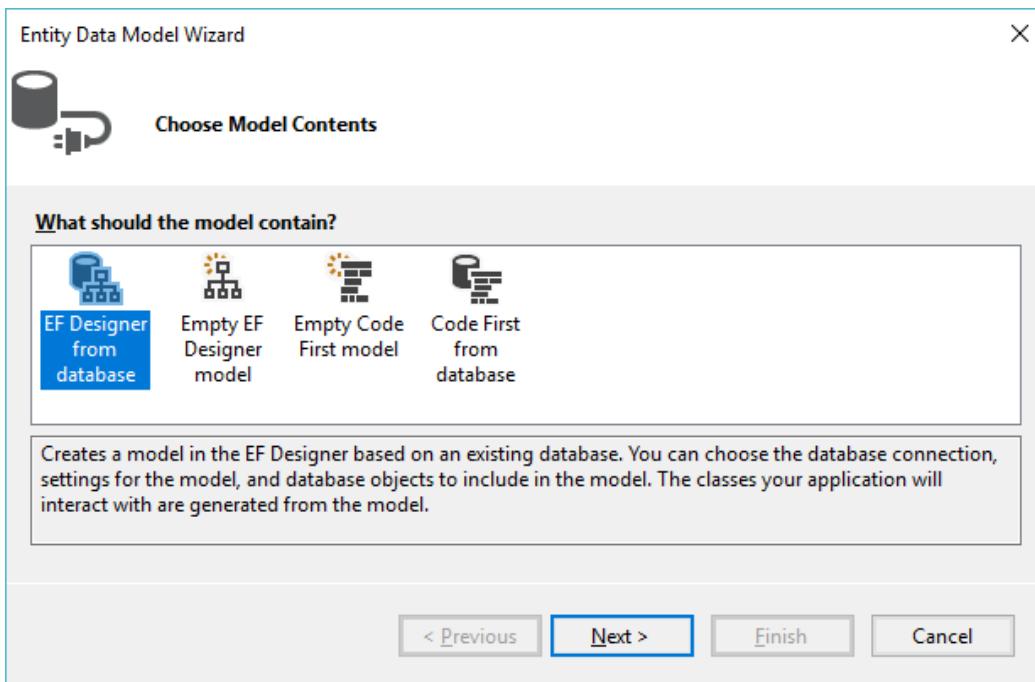


Figure 5.4: Entity Data Model Wizard

- Step 7 • Select **EF Designer from database** option and click **Next**, which displays the **Database connection** option.

Step 8

- Click **New Connection**, which displays a **Data source** selection option.

Step 9

- Select **Microsoft SQL Server** from the list and click **Continue**.

Step 10

- In the **Connection Properties** window, provide Azure Database Server name as the server name along with a suitable Username and Password. The connection properties and options are shown in Figure 5.5.

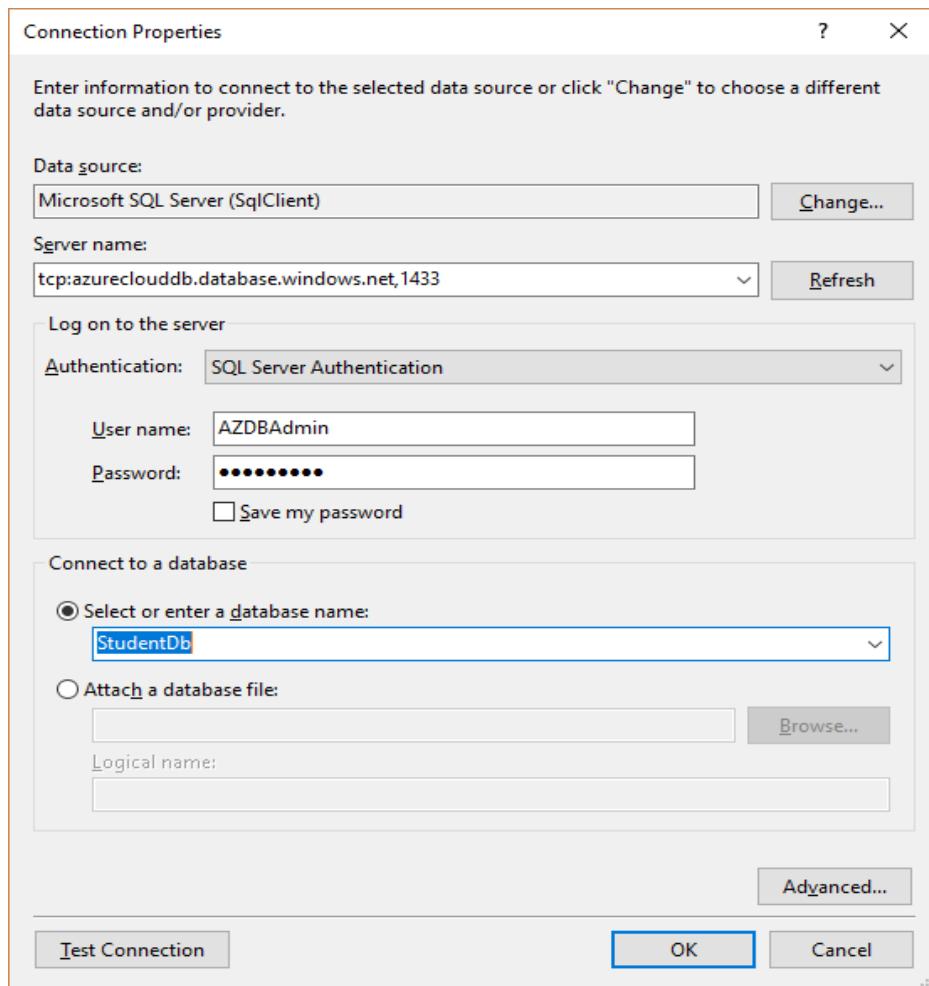


Figure 5.5: Connection Properties Option Window

Step 11

- Click **Test Connection** to test if the database is working fine. Click **OK** to proceed.

Step 12

- In the **Next** window, select **Option 2** to 'Include sensitive data in the connection string' and then, click **Next**.

Step 13

- Select **Entity Framework 6.0** and click **Next**.

Step 14

- Ensure that all necessary tables are selected as shown in Figure 5.6 and then, click **Finish**.

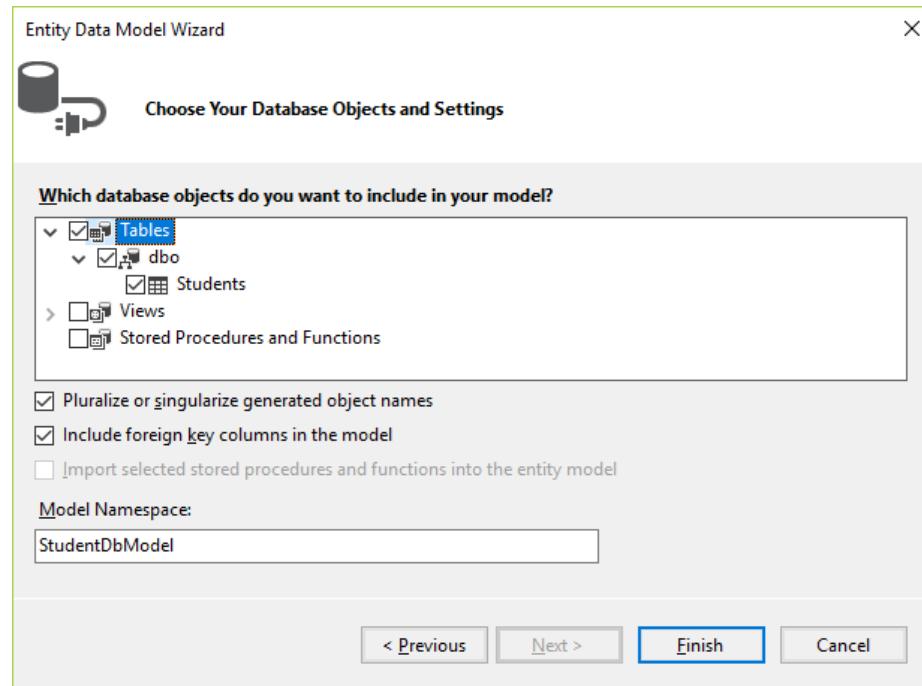


Figure 5.6: Database Objects in Entity Framework

The StudentDBModel diagram on the screen represents the Student table from the database, as shown in Figure 5.7.

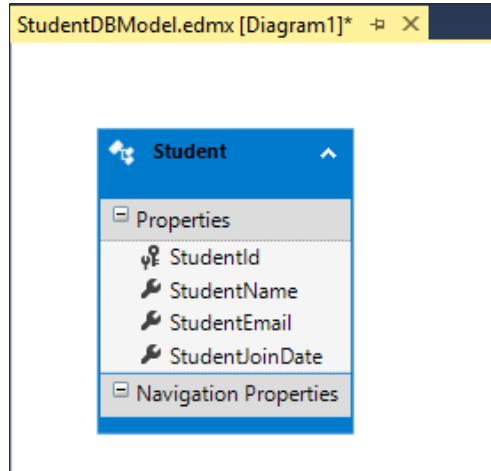


Figure 5.7: Entity Data Model Diagram

The data models and DbContext-related codes that are auto-generated can be seen in the Solution Explorer, as shown in Figure 5.8.

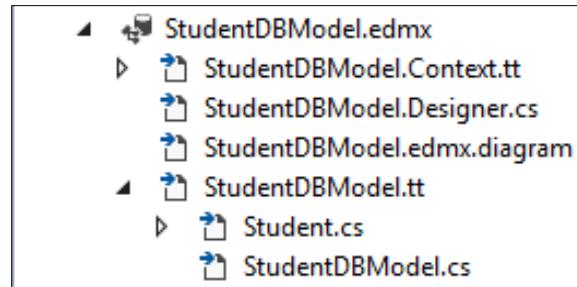


Figure 5.8: Entity Data Models in Solution Explorer

The auto-generated Student model class is shown in Code Snippet 1.

Code Snippet 1:

```
public partial class Student {
    public int StudentId { get; set; } public string
    StudentName { get; set; } public string StudentEmail
    { get; set; }
    public System.DateTime StudentJoinDate { get; set; }
}
```

5.3.2 Inserting a Record Using DbContext

Following are the steps to implement the insert or create functionality of Student database using EF and ASP.NET MVC:

Step 1

Right-click Controllers folder and select Add New Scaffolded Item. In the Add Scaffold dialog box, select MVC 5 Controller with views, using Entity Framework option. This option will generate controller and views for creating, updating, deleting, and displaying the data in the model.

Step 2

Select Student for the model class, and select StudentDBEntities for the context class. Name the controller as StudentController.

Under the Controllers folder, a StudentController class will be added.

The auto-generated views can be seen in Figure 5.9.

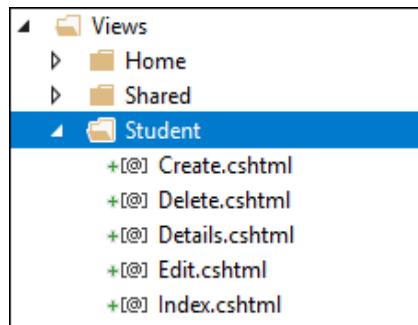


Figure 5.9: Auto-generated Views

Step 3

Run the application and in the browser address bar, add /Student to the URL. Add data to the Web page as shown in Figure 5.10.

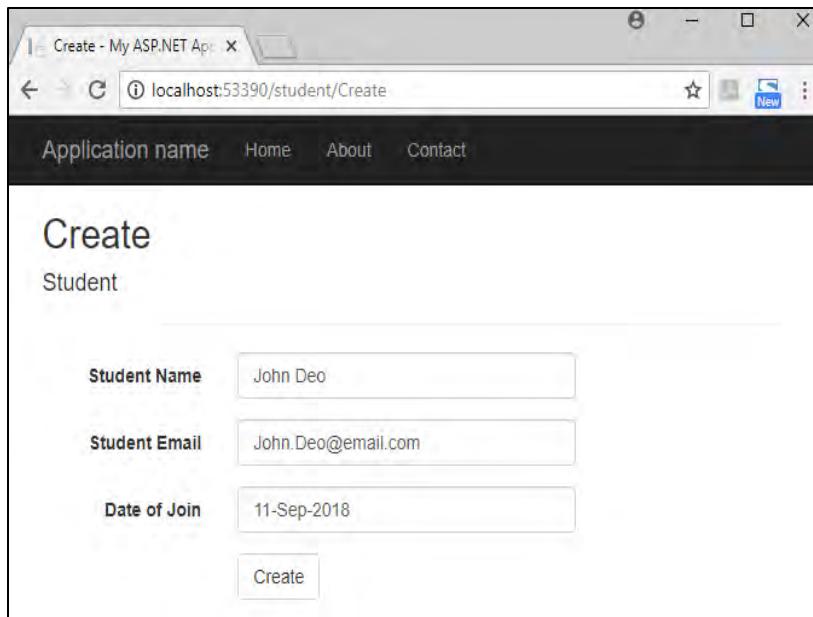


Figure 5.10: Output

Similarly, developers can test the Edit and Delete functionalities of the application. This example showed how easy it is to create a fully functional CRUD application using Entity Framework within a few steps. This improves productivity and reduces chances of errors since most of the code is auto-generated.

5.4 Query Data by Using LINQ

LINQ-to-Entities queries work on the entity set (DbSet type properties) to connect with the information in the database. Developers should use LINQ method syntax or query syntax when executing queries on the EDM.

Code Snippet 2 depicts a LINQ-to-Entities query, which retrieves information from the Student table in the database.

Code Snippet 2:

```
var query = _context.Students  
    .Where(s => s.StudentName == "John Dill")  
    .FirstOrDefault<Student>();
```

5.4.1 Inserting a Record Using DbContext

Eager loading is the method in which a specific type of entity also loads entities that are related as a part of the query. This erases the requirement to execute a

different query for related entities. Use the `Include()` method to execute eager loading.

5.4.2 Lazy Loading in Entity Framework

Lazy loading is the process wherein a specific entity or a group of entities are loaded from database automatically when a property indicating the entity/entities is accessed for first time. For executing lazy loading, create instances of proxy types which are derived and then, override virtual properties to also include the loading hook.

Consider an example where the `StudentAddress` entity is part of the `Student` entity. In lazy loading, the `Student` entity data from the database is first loaded by the context. Subsequently, the `StudentAddress` entity is loaded when the `StudentAddress` property is accessed.

5.5 Complex Query in LINQ

One of the most important concepts in LINQ-to-Entities is that it is a declarative language. The focus is on defining the information required and not on how to collect the required information. This approach enhances readability and abstraction by allowing developers to concentrate more on the intent of the query and less on the implementation details.

5.5.1 Standard Query Operators

Standard query operators are listed in Table 5.2.

Operation	Standard Query Operators
Set	<code>Distinct</code> , <code>Except</code> , <code>Intersect</code> , <code>Union</code>
Quantifiers	<code>All</code> , <code>Any</code> , <code>Contains</code>
Elements	<code>ElementAt</code> , <code>ElementAtOrDefault</code> , <code>First</code> , <code>FirstOrDefault</code> , <code>Last</code> , <code>LastOrDefault</code> , <code>Single</code> , <code>SingleOrDefault</code>
Filtering	<code>Where</code> , <code>OfType</code>
Sorting	<code>OrderBy</code> , <code>OrderByDescending</code> , <code>ThenBy</code> , <code>ThenByDescending</code> , <code>Reverse</code>
Aggregation	<code>Aggregate</code> , <code>Average</code> , <code>Count</code> , <code>LongCount</code> , <code>Max</code> , <code>Min</code> , <code>Sum</code>
Join	<code>GroupJoin</code> , <code>Join</code>
Grouping	<code>GroupBy</code> , <code>ToLookup</code>

Operation	Standard Query Operators
Projection	Select, SelectMany
Equality	SequenceEqual
Concatenation	Concat
Generation	DefaultEmpty, Empty, Range, Repeat

Table 5.2 Standard Query Operations

Code Snippet 3 depicts a sample complex query in LINQ. It retrieves student records in an ordered fashion.

Code Snippet 3:

```
var studentOrderedList =
    from student in _context.Students orderby
    student.StudentName select student;
```

The query is considered complex because it uses orderby query operator but is otherwise easy to create.

5.6 Summary

- ✓ Entity Framework (EF) is a .NET ORM framework facilitating object-oriented data access by mapping database tables to .NET objects.
- ✓ EF offers three varieties of design approach depending on the requirement, which are Code-First, Database-First, and Model-First.
- ✓ LINQ is one of the most powerful features that was first introduced in .NET Framework 3.5 and can be used with C# to query different data sources.
- ✓ Developers can write C# code to specify a LINQ query which automatically generates an SQL query by EF.
- ✓ LINQ also supports complex queries such as aggregate functions, joins, and paging which can be achieved by writing C# code.

5.7 Test Your Knowledge

1. What does Entity Framework (EF) primarily contribute to developer productivity?
 - A. Graphical user interface
 - B. Automatic generation of database commands
 - C. Code obfuscation
 - D. Hardware optimization
2. Which design approaches does EF offer for modeling entities and relationships?
 - A. Front-End, Back-End, and Full-Stack
 - B. Code-First, Database-First, and Model-First
 - C. Agile, Waterfall, and Scrum
 - D. UI-First, Logic-First, and Data-First
3. In Entity Framework, what allows developers to write C# code for LINQ queries that automatically generate SQL queries?
 - A. SQL Server Management Studio
 - B. Entity Query Language (EQL)
 - C. Dynamic SQL
 - D. LINQ to SQL
4. What kind of queries can be achieved by writing C# code using LINQ in Entity Framework?
 - A. Basic CRUD operations
 - B. Complex queries such as aggregate functions, joins, and paging
 - C. SQL injection queries
 - D. Only simple SELECT queries
5. Assuming a table named `Flowers` exists in a database, identify the correct code that depicts a LINQ-to-Entities query to retrieve information from `Flowers`.

A.	<pre>var query = _context.Flowers .Where(s => s.FlowerName == "Geranium") .FirstOrDefault<Flower>();</pre>	C.	<pre>var query = _context.Flowers .Where(s => s.FlowerName == "Geranium").FirstOrDefault<Flow er>();</pre>
B.	<pre>query = _context.Flowers .Where(s => s.FlowerName == "Geranium").FirstOrDefault<Flo wer>();</pre>	D.	<pre>var query = _context.Flowers .Where(s => s.FlowerName == "Geranium").FirstOrDefault<Flow er>();</pre>

5.7.1 Answers to Test Your Knowledge

1. B
2. B
3. D
4. B
5. D

Try It Yourself

1. Create an Entity Data Model (EDM) in Visual Studio 2022 for any sample database that is created using Azure SQL.
2. Test the Edit and Delete functionalities of the application (as discussed in section 5.3.2) using Entity Framework.



SESSION 06

WORKING WITH DATA IN AZURE APPLICATIONS

Overview

This session explains various data ingestion techniques, Azure Data Service and the implementation of data caching services in Microsoft Azure. It describes how to use transactions and also defines distributed transactions in SQL Azure. It explains the importance of Content Delivery Networks (CDNs) and the functionalities of Azure services for Master Data Processing.

Learning Objectives

In this session, students will learn to:

- Explain steps involved to ingest data from various sources
- Summarize how various Azure databases can be used
- Explain the implementation of data caching services in Azure
- Explain how to use transactions
- Define distributed transactions in SQL Azure
- Explain the role of Content Delivery Networks (CDNs)
- Explain Data Processing using Azure services

6.1 Data Ingestion into Azure

Data ingestion into Azure involves the transfer and loading of data from diverse sources into Azure storage or data services. This process is crucial for organizations to utilize Azure's capabilities for analysis, storage, and processing.

Let us explore the process and the methods/tools involved.

The Azure portal provides fault tolerance and continuous availability. It is presented in each Azure data center. This makes the Azure portal resilient to individual data.

6.1.1 Process of Data Ingestion into Azure

The complete process of data management and ingestion into Azure is as follows:

a. Data Source Identification:

Identify various sources such as databases, files, IoT devices, streaming platforms, and so on containing the data to be ingested into Azure.

b. Data Preparation and Transformation:

- Clean, format, and transform the data to ensure compatibility with Azure services and the target data model.
- Resolve any schema differences or inconsistencies between the source and the destination.

c. Selection of Ingestion Method:

- Choose appropriate ingestion methods based on factors such as data volume, velocity, variety, and latency requirements.
- Select tools or services that support the desired ingestion method(s) effectively.

d. Data Ingestion Methods and Tools:

- Azure Data Factory: Offers a managed cloud service for orchestrating data movement and transformation workflows across various sources and Azure services.
- Azure Blob Storage and Azure Data Lake Storage: Supports direct upload of files or data from on-premises or cloud-based sources.
- Azure Event Hubs and Azure IoT Hub: Enable real-time streaming data ingestion for IoT devices and event-driven applications.
- Azure SQL Database Migration Service: Facilitates database migration to Azure SQL Database with minimal downtime.
- Azure Data Box: Physical devices used to securely transfer large volumes of data to Azure.

e. Data Compatibility and Transformation:

- Ensure data compatibility by transforming data formats, structures, and schemas as required.
- Use tools such as Azure Data Factory or Azure Databricks for data transformation tasks.

f. Data Validation and Monitoring:

- Validate the ingested data to ensure accuracy, completeness, and consistency.
- Implement monitoring and logging mechanisms to track the data ingestion process and detect any issues or errors.

g. Data Storage and Accessibility:

- Store the ingested data in Azure services such as Azure SQL Database, Azure Blob Storage, Azure Data Lake Storage, or Azure Cosmos DB based on the specific requirements and use cases.
- Configure access permissions and data security measures as per the organization's policies.

6.1.2 Ingesting Data in Azure Data Explorer

The pre-requisites to ingest or consume data in Azure Data Explorer are as follows:

- Open Azure Data Explorer, select My cluster, and complete the necessary fields to create a cluster. Refer to Figure 6.1.

The screenshot shows the Azure Data Explorer interface. On the left, there's a sidebar with icons for Home, Query, and Dashboards, and a selected 'My cluster' icon. The main area is titled 'John_Cluster1'. It includes a brief description: 'Use the free cluster to try out Azure Data Explorer. Load your data and quickly get answers to your questions for free.' Below this is a 'Learn more' link and two buttons: 'Upgrade to Azure Cluster' (blue) and 'Delete' (grey). The 'Cluster details' section contains the following information:

- Cluster URI: <https://kyc-504tj9ew79hebekeck.southcentralus.kusto.windows.net>
- Cluster location: North America
- Data ingestion URI: <https://ingest-kyc-504tj9ew79hebekeck.southcentralus.kusto.windows.net>
- Policies: Terms of service and Microsoft privacy policy

At the bottom, there's an 'Actions' section with three buttons: 'Ingest data', 'Query data', and 'Create database'.

Figure 6.1: Creating Cluster

- Navigate to 'My cluster' and establish a database under the previously created cluster. Refer to Figure 6.2.

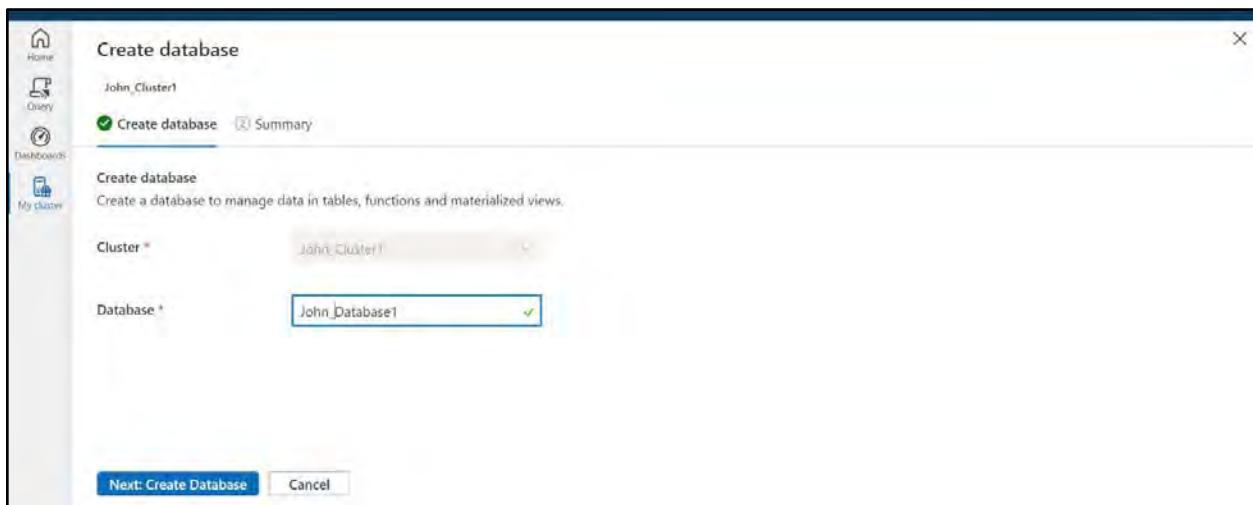


Figure 6.2: Creating Database

- Open Azure Portal and create a storage account. Refer to Figures 6.3 and 6.4.

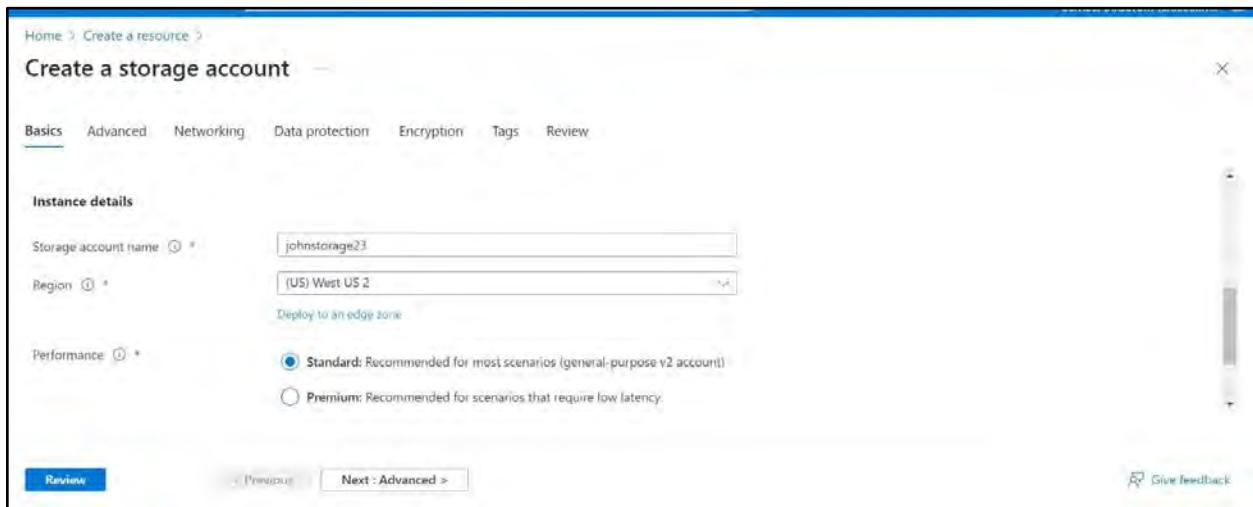


Figure 6.3: Creating Storage Account

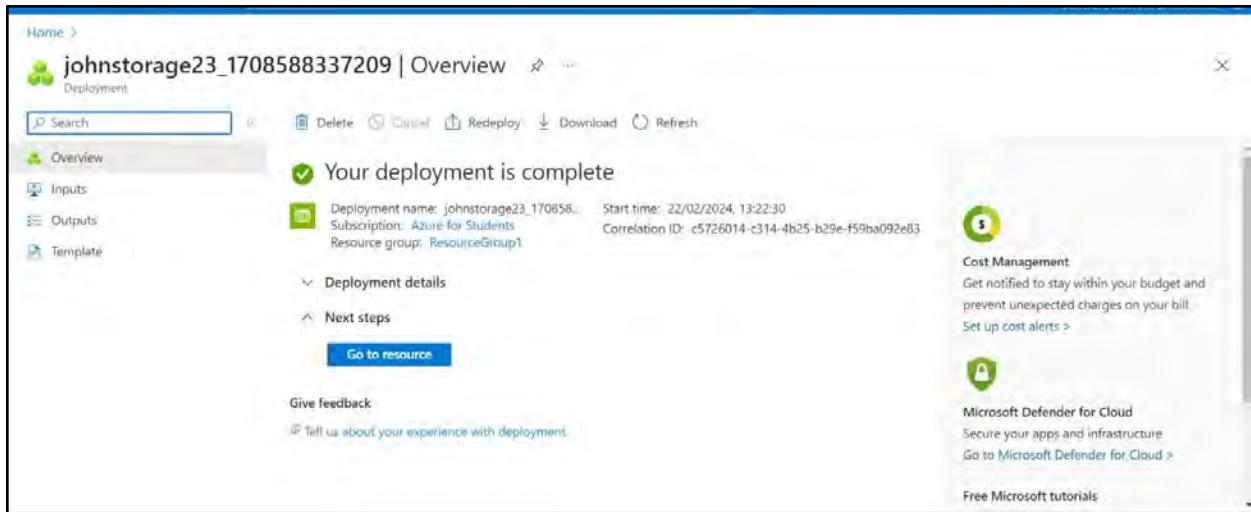


Figure 6.4: Completion of Storage Account Deployment

Now, ingesting data into Azure Data Explorer can be initiated.

Step 1: Launch Azure Data Explorer and navigate to the Query section on the left. Right-click the previously created database. A user interface similar to Figure 6.5 will appear. Select Get data.

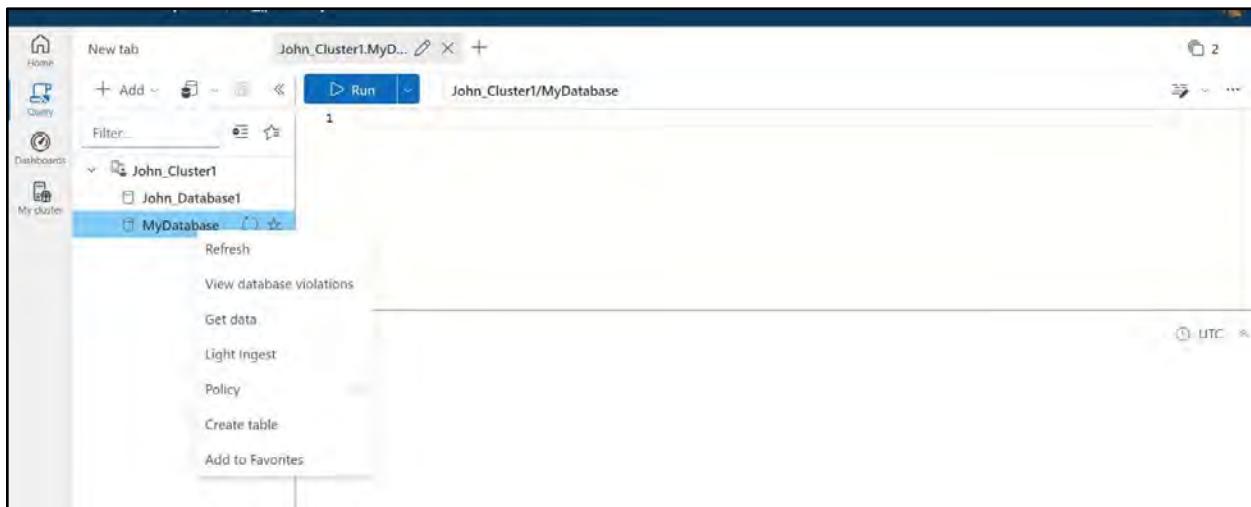


Figure 6.5: Getting Data

Step 2: Select a data source for data ingestion; in this case, opt for 'Local file.' Refer to Figure 6.6.

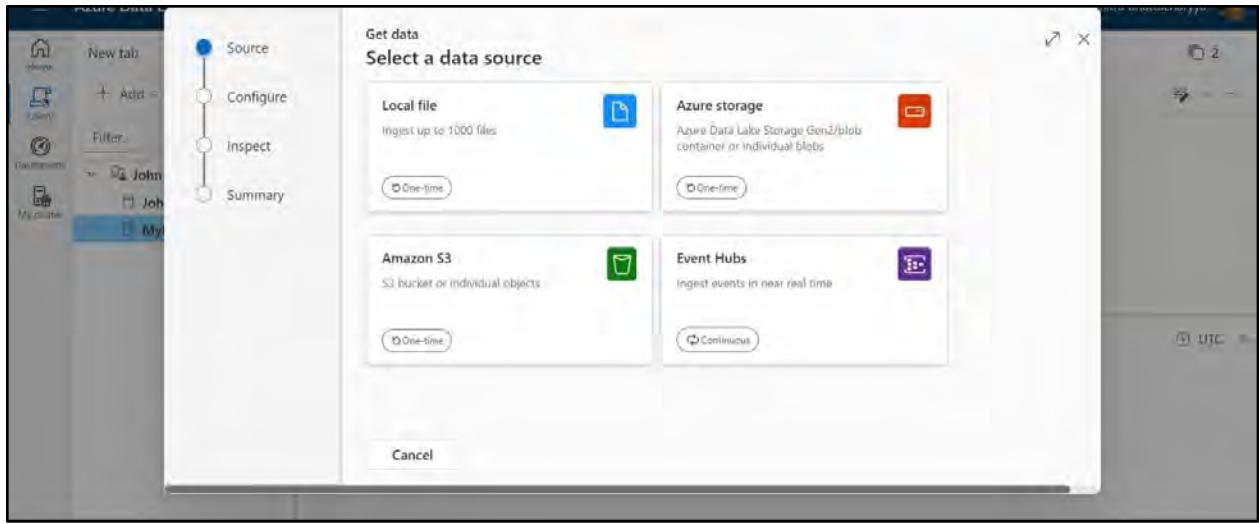


Figure 6.6: Selecting Data Source

Step 3: Click New table and assign an appropriate name to your table. Drag or upload the data on the right and proceed by clicking s. Refer to Figure 6.7 for visual guidance.

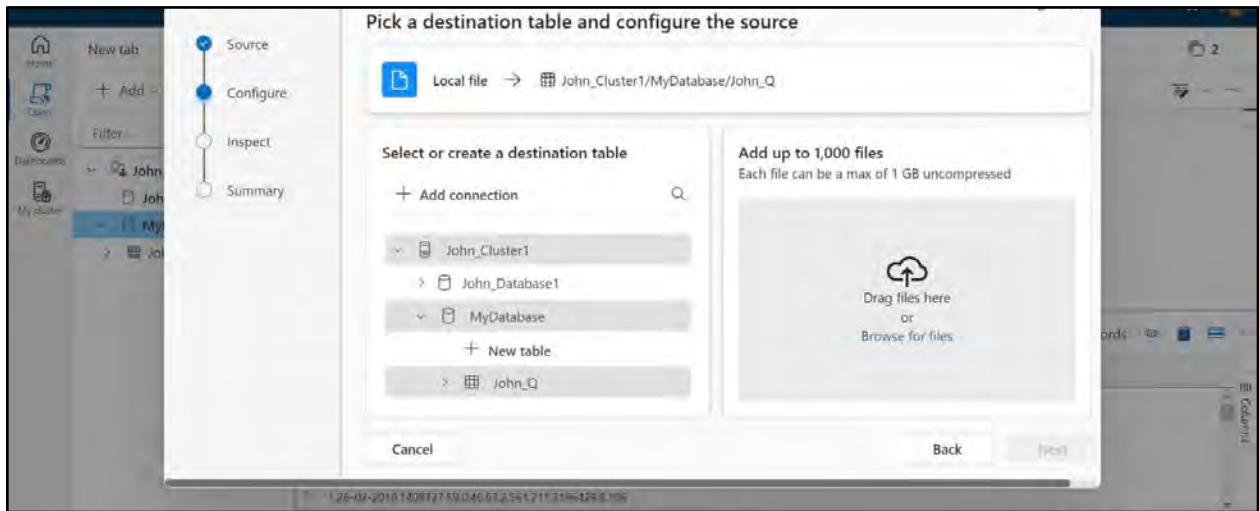


Figure 6.7: Configuration of Source

Step 4: Verify the data preview to ensure that correct data has been uploaded. If accurate, click Finish; otherwise, click Back and repeat the preceding step. Refer to Figure 6.8.

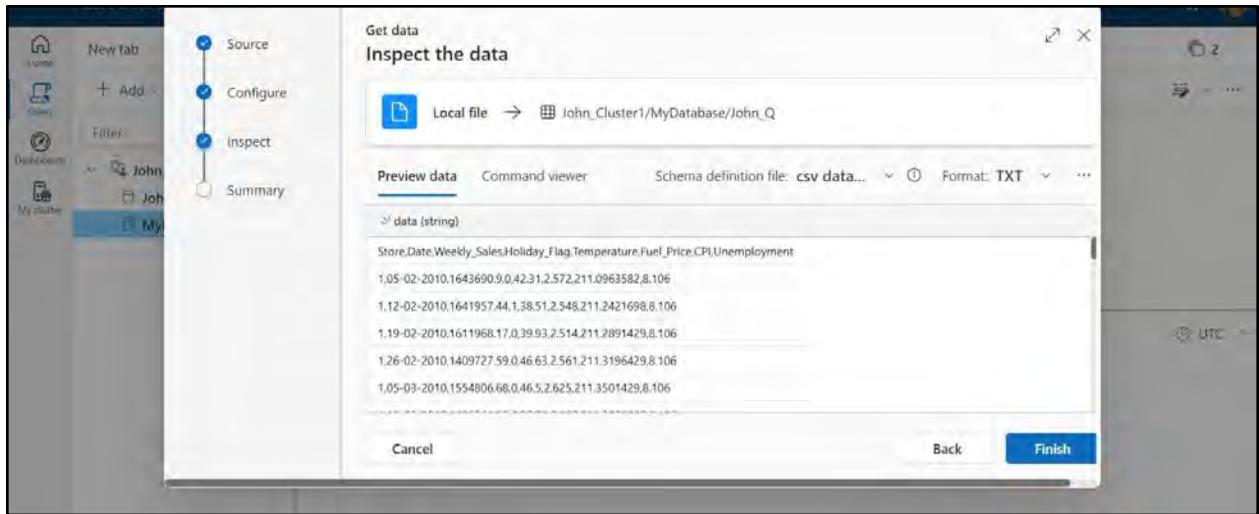


Figure 6.8: Data Inspection

Upon successful ingestion of the data, an interface resembling Figure 6.9 will emerge.

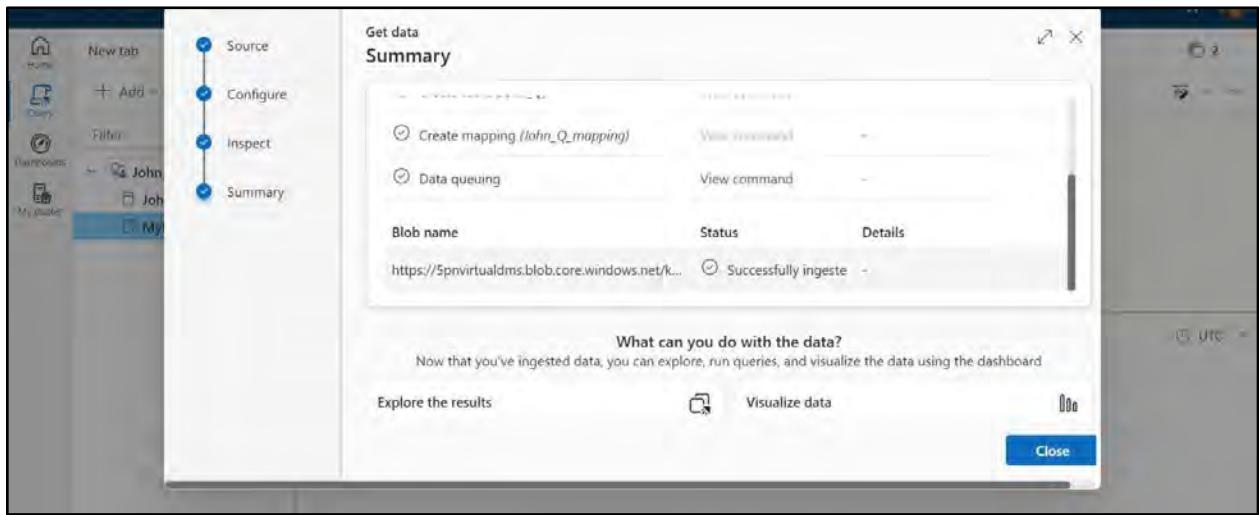


Figure 6.9: Ingestion Completion

Step 6: To examine the ingested data, click table name (in this instance, 'John_Q') and select Run, as illustrated in Figure 6.10.

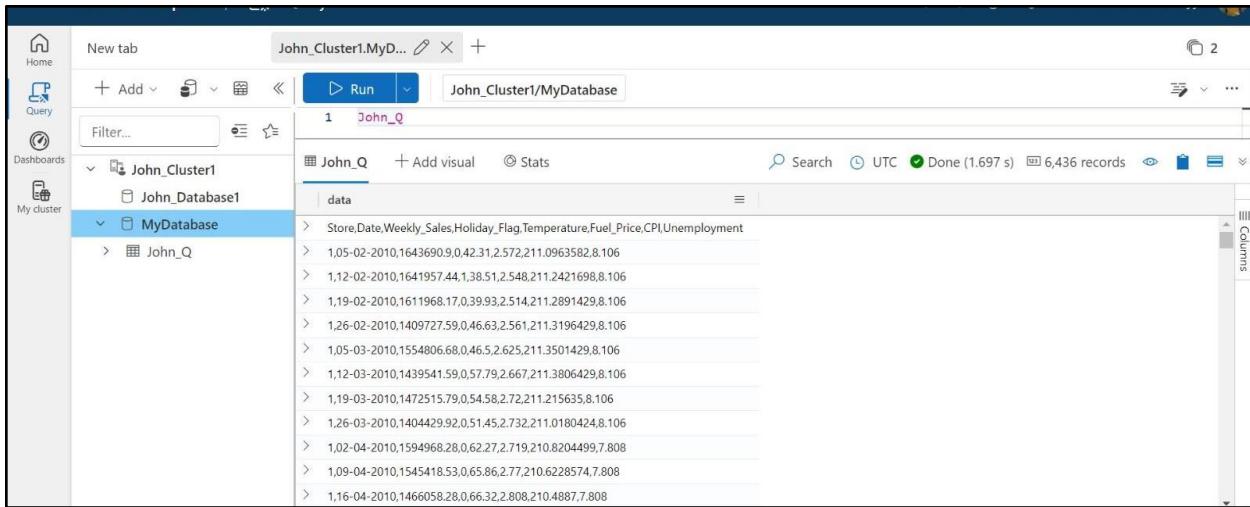


Figure 6.10: Running the Ingested Data

Similarly, developers can consider the prospect of ingesting data from other data sources into Azure Data Explorer beyond local files, as demonstrated in the previous example.

6.2 Azure Data Service

Azure Data Service is a managed service that extends the platform with so-called common capabilities (shared functionalities). Due to the importance of data in today's digital world, it is a separate service and separates from building 4 GB blocks of Azure applications.

- **Storage:** This category contains a total of five different services. Blob storage (unstructured data storage), table storage (key-value pair-based NoSQL storage), queue storage (for message processing), file storage, and Disk storage (premium storage).
- **SQL Database as a Service:** This category includes three fully managed databases as a service: SQL Server, MySQL, and PostgreSQL. This category also includes special offers such as SQL Server DWH, SQL Server Stretch DB, and SQL Server Elastic DB. All special offers are further developments of SQL Server as a Service and address specific cloud workloads.
- **NoSQL database as a service:** This category includes NoSQL databases that are fully managed by services such as Azure CosmosDB. NoSQL databases are used to store semi-structured data. NoSQL databases separate the storage of key values, plots, and document data. Developers can specify the type of storage to use when creating the database.

- Big Data: This category includes a fully managed implementation of Apache Hadoop with Azure HDInsight. In addition, implementations for Apache Storm, Apache Spark, Apache Kafka, and Microsoft R Server (with various levels of development) are available.
- Analytics: This category includes data analytics and tools such as Azure Stream Analytics, Azure Data Lake Analytics, and Azure Data Factory.
- AI: This category includes fully managed Azure Machine Learning (Azure ML) services that make it easy to build, deploy, and share predictive analytics solutions, and some ready-to-use out-of-the-box solutions (Microsoft Cognitive Services).
- Visualization: This category is a special case because the service offered (Microsoft Power BI) is strictly an Azure service, but only as a SaaS.

6.2.1 Azure SQL Database

Azure SQL Database is a managed cloud database service provided by Microsoft as a part of Azure services. It operates as a Platform as a Service (PaaS), taking care of database management tasks such as upgrades, patches, backups, and monitoring without requiring user intervention. It supports versatile storage for structured, semi-structured, and non-relational data. It incorporates built-in intelligence to learn application patterns, optimizing performance, reliability, and data protection.

Its key capabilities include:

- Learns and adapts to the data access patterns of the host application for enhanced performance and reliability.
- Allows scaling on demand.
- Manages and monitors multi-tenant applications, providing isolation benefits with one customer per database.
- Integrates with open-source tools such as cheetah, SQL-CLI, Visual Studio Code, as well as Microsoft tools such as Visual Studio, SQL Server Management Studio, Azure Management Portal, PowerShell, and REST APIs.
- Ensures data protection through encryption, authentication, access control, continuous monitoring, and auditing.

6.2.2 Microsoft Azure Cosmos DB

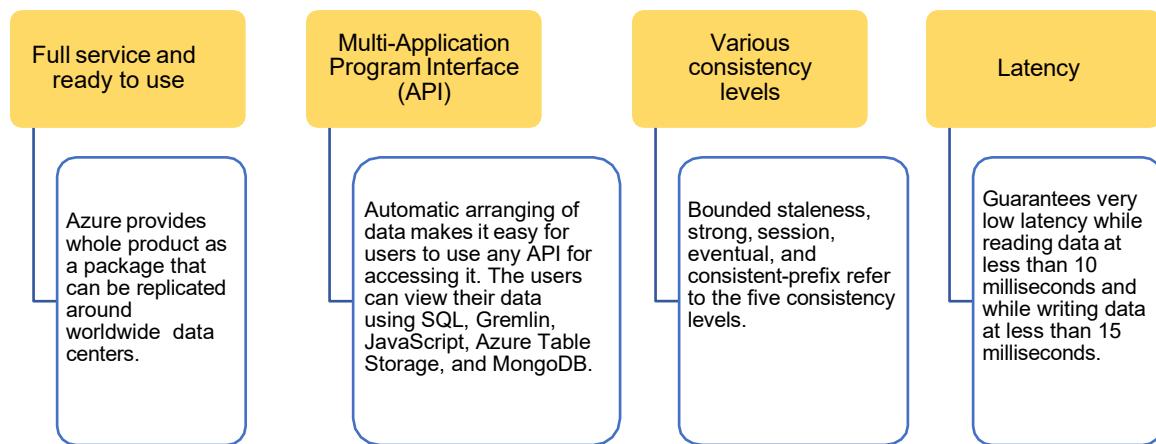
Microsoft Azure Cosmos DB provides database service across the world. It allows managing data, which is stored in data centers across the globe. It also provides tools required for scaling both computational resources as well as global distribution patterns.

Following are some of its features:

- Supports document, key-value, relational, and graph models using one backend
- Does not rely on any schemas
- Uses query language similar to SQL
- Supports ACID transactions

Cosmos is classified as a NewSQL type of database as it uses query language and also supports ACID transactions. However, it does not include a relational data model.

Following are some of its benefits:



6.2.3 Azure Data Lake Storage

Azure Data Lake Storage (ADLS), or Azure Data Lake Store, is a secure and scalable platform for high-performance analytics workloads. It offers cost-effective tiered storage, policy management, and compatibility with Hadoop Distributed File System (HDFS) tools. ADLS ensures security with single sign-on and access controls. This makes it a reliable solution for seamlessly integrating and managing large volumes of organizational data.

Following are some of its features:



Limitless Scale and Data Durability: Achieve unparalleled scalability and 16 9s of data durability with automatic geo-replication.



Highly Secure Storage: Benefit from a highly secure storage solution with adaptable protection mechanisms, covering data access, encryption, and network-level control.



Unified Storage Platform for Analytics: Utilize a single storage platform for seamless ingestion, processing, and visualization, supporting popular analytics frameworks.



Cost Optimization: Optimize costs through independent scaling of storage and compute, efficient lifecycle policy management, and object-level tiering.

6.2.4 Azure Synapse Analytics

Azure Synapse is a tool that helps companies quickly understand their data. It blends powerful technologies for organizing big data and traditional databases. It includes features such as Data Explorer for analyzing logs and time-based data, pipelines for smoothly moving and transforming data. It easily connects with other Microsoft services such as Power BI, CosmosDB, and AzureML. It is a one-stop solution for businesses to get insights from their data without the hassle.

Following are some of its features:



Limitless Scalability for Rapid Insights: Achieve rapid insights from diverse data sources, spanning data warehouses and big data analytics systems.



Enhanced Intelligent Apps with ML: Transform data into powerful insights using machine learning models to enhance intelligent applications.



Unified Analytics Development: Streamline development with a unified experience, significantly reducing project timelines for end-to-end analytics solutions.



Cutting-edge Security and Privacy: Enjoy advanced security features, including column-level and row-level security, and dynamic data masking.

6.2.5 Azure Databricks

Azure Databricks is a unified analytics platform designed for developing, deploying, and managing enterprise-grade data, analytics, and AI solutions at scale. It efficiently handles infrastructure deployment on your cloud account, when integrated with cloud storage and security. The platform provides tools for seamless connection of data sources, enabling processing, storage, analysis, and monetization of datasets, supporting solutions from BI to generative AI.

Following are some of its features:

Reliable data engineering: Reliable data engineering for big data processing in both batch and streaming.



Analytics for all your data: Get analytics for all your data, ensuring insights from the most recent and complete datasets.



Collaborative data science: Simplify and speed up collaborative data science on large datasets.



Rooted in open source: Built on a fast Apache Spark foundation, sticking to open source roots.



6.2.6 Applications of Various Azure Databases

Applications of various Azure Databases have been listed in table 6.1 along with their specific strengths and scenarios where they excel.

Azure Service	Unique Applications and Optimal Use Cases	Specific Strengths	Scenarios Where They Excel
Azure SQL Database	Transactional Applications: Line-of-business Applications	Fully managed relational database service, Scalability, High Availability, and security	<ul style="list-style-type: none">Applications requiring structured data management and transactions.Business-critical applications with ACID compliance.
Azure Cosmos DB	Globally Distributed Applications, IoT, Gaming, and Mobile Applications	Multi-model database with low-latency, high availability, and elastic scalability	<ul style="list-style-type: none">Applications requiring low-latency access.Globally diverse data models and high throughput requirements.
Azure Data Lake Storage	Big Data Analytics: Data archiving and Backup	Scalable, secure storage for structured, semi-structured, and unstructured data	<ul style="list-style-type: none">Storing and analyzing large volumes of diverse data types.Use cases requiring compatibility with big data analytics tools.
Azure Synapse Analytics	Enterprise Data Warehousing, Big Data Analytics and Data Integration	Integration of data warehousing and big data analytics - Exploration and insights from diverse datasets	<ul style="list-style-type: none">Unified analytics platform for large-scale data integration and analytics.

Azure Service	Unique Applications and Optimal Use Cases	Specific Strengths	Scenarios Where They Excel
			<ul style="list-style-type: none"> • Use cases requiring SQL and spark-based analytics.
Azure Databricks	Collaborative Data Science and Analytics, Real-time Data Processing and Machine Learning	Apache Spark-based analytics platform - Collaborative and scalable environment for data engineering and analytics	<ul style="list-style-type: none"> • Real-time analytics on streaming data. • Collaborative and distributed data processing for machine learning and analytics.

Table 6.1: Applications of Various Azure Databases

6.3 Data Caching Services in Azure

Data that is present somewhere remote is stored locally for frequent use through a technique called caching. This stored data can be reused quickly again whenever required. Data caching avoids the repetitive method of getting the same data from the server and executing the same logic again and again, in the case of a user request.

Caching, which means performance, helps in the following:

- Reduce network traffic [Http call to server]
- Reduce server round trips [hosting server or database server or any other server]
- Ensure not to use the same data binding logic

6.3.1 Caching Static Content:

Static content is often used to design any Website. Content that is static and does not change dynamically is called static content, such as images, Cascading Style Sheets (CSS), and JavaScript files.

If these files are heavy, they can consume around 60 percent of the total time to load from the server. Loading takes more time when the static content is more.

To share common data with multiple action methods and to add this type of data to the cache, developers should use `HttpContext.Cache` for setting or getting the data. It is only during the first time that the downloading process occurs. In ASP.NET MVC, it is possible to cache the downloaded static contents. This downloaded content is stored in memory and when the user accesses the same

page again, the memory caches provide the content.

6.3.2 Cache Policies

A cache policy is a set of rules that determine if a cached copy of the requested resource can be used for a request. Client cache requirements are specified in the applications, but an effective cache policy is determined by the following:

- Client cache requirements
- Server's content expiration requirements
- Server's re-validation requirements

Using client cache policy and server requirements, it can be ensured that the most conservative cache policy is achieved. This ensures that the client receives up-to-date content.

6.3.3 Including Policy Expirations

Cache data is often temporary. The cache expiration policy represents a set of eviction and expiration details for a specific cache entry. There are two types of expiration policies available for cache expiration. These are given as follows:

Absolute Expiration

Specifies a fixed duration for cache validity, and the entry expires after that duration.

Sliding Expiration

Dynamically extends the expiration time for a cache entry based on access patterns.

Code Snippet 1 represents the caching of `students` for 23 Hours and 59 Minutes. On completion of the time, the cache is refreshed and it will be reassigned.

Code Snippet 1:

```
private const string studentKey = "studentListCacheKey";
CacheItemPolicy cachePolicy = new CacheItemPolicy();
cachePolicy.AbsoluteExpiration =
DateTime.Now.AddHours(23).AddMinutes(59);
cache.Add(studentkey, studentList, cachePolicy);
```

6.3.4 Using Cache Dependency

The validity of a cache item is based on cache dependencies. It is based on a file or directory on the Web server or another cache item. Any change in the

dependency object invalidates and removes the cached item automatically.

A dependency is created by first creating a `CacheDependency` object. This object mentions the file, directory, or cache item that the dependency will be based upon. The `Cache.Insert()` method is used to add the dependent cache item. The `CacheDependency` object is used as the parameter.

Table 6.2 shows three types of cache dependencies.

Type of Cache Dependency	Description
Cache dependency on cached items	Indicates dependence of cached items on other cached items. For example, removing item X from the cache allows removing item Y also from the cache.
Cache dependency on a file	Indicates cache is dependent on a file. Any update in the file contents leads to the expiration of the cache and refilling of the content again.
Cache dependency on SQL	Indicates that until the table entries do not change, the data will be present in the cache. Any change to the content would expire the cache. SQL Cache dependency is enabled using the <code>aspnet_Regsql.exe</code> command-line tool.

Table 6.2: Cache Dependency Types and Descriptions

6.3.5 Query Notifications in ADO.NET

Any change in data notifies an application. This is called query notification. This is mostly used when applications provide a cache of information from the database and are notified during a change in the source data. For example, a Web application.

ADO.NET helps to implement query notifications using following classes:

6.4 Using Transactions

A transaction is a single unit of work that means either all or none are performed. If it runs smoothly, all data operations are performed and their output is written to the database. If there is an error/exception, that transaction must be rolled back. If this happens, all data modifications/operations are withdrawn and none of them are executed.

6.4.1 Properties of Transaction

A transaction has following four standard properties, usually called by the acronym ACID:

Atomicity	Assures successful completion of work unit and the operations used. If the transaction fails then, operations are rolled back to their previous state.
Consistency	Assures that when a transaction is successful, the state of the database changes.
Isolation	Assures independent and transparent transactions.
Durability	Assures that when a system fails, a committed transaction remains.

6.4.2 Database Transaction

Any transaction in software development refers to a database transaction. In this type of transaction, an array of data manipulation statements (insert/update/delete) are executed together. Each statement either executes successfully or fails. This ensures that the database is in a consistent mode. Any database state change is represented by a database transaction.

6.4.3 Local Transaction

On a single data source/database, an array of data manipulation statements is executed together. This is called a local transaction. This type of transaction is directly done by a database and is a single phase. A Lightweight Transaction Manager (LTM) is provided by `System.Transactions` to manage this type of transaction. It behaves as a gateway in a transaction. The `System.Transactions` start and handle all transactions. Any change/barrier in the transaction nature due to predefined rules, the Microsoft Distributed Transaction Coordinator (MSDTC) distributed transaction receives a fallback transaction.

6.5 Distributed Transactions in SQL Azure

Some transactions include more than one data source. This is called a distributed or elastic transaction. All the affected data sources are mitigated in case of a transaction failure. Microsoft Distributed Transaction Coordinator (MSDTC) uses the classes present in `System.Transactions` for managing distributed transactions on the cloud. A two-phase commit procedure is implemented for

maintaining atomicity across the databases.

6.5.1 Elastic Database Transactions with Azure SQL Database

Some transactions use various databases in SQL DB. In Azure SQL DB, elastic database transactions enable a .NET application utilizing ADO .NET to employ database transactions for SQL DB.

6.5.2 Using the System.Transactions Namespace

ADO.NET is linked with the `System.Transactions` framework. To implement a distributed transaction, there is no change.

However, a `TransactionScope` object is instantiated to create a transaction. Internally, in its constructor, a transaction is created by the `TransactionScope` object and is assigned to the `System.Transactions.Transaction.Current` property. This type of transaction is termed the current or ambient transaction. `TransactionScope` represents a disposable object. When the `Dispose()` method is called, this transaction ends.

To be part of the transaction, ADO.NET connections must be opened after the `TransactionScope` object is instantiated and before it is disposed of. This is usually observed when opening single or multiple database connections.

The transaction will remain committed if the `TransactionScope.Complete()` method is called before the `Dispose` method. Otherwise, it will be called off.

Code Snippet 2 displays an example of using `System.Transactions`. A transaction in the current thread is called an ambient transaction. An ambient transaction is opened by the `TransactionScope` class. NET. The transaction involves all the connections that are opened within the `TransactionScope`. A distributed transaction occurs when various databases participate. The scope is set as complete to control the outcome of the transaction. This indicates a commitment.

Code Snippet 2:

```
using (var scope = new TransactionScope()){
    using (var conn1 = new SqlConnection(connStrDb1))
        conn1.Open();
    SqlCommand cmd1 = conn1.CreateCommand();
    cmd1.CommandText = string.Format("INSERT INTO T1
values(1)");
```

```

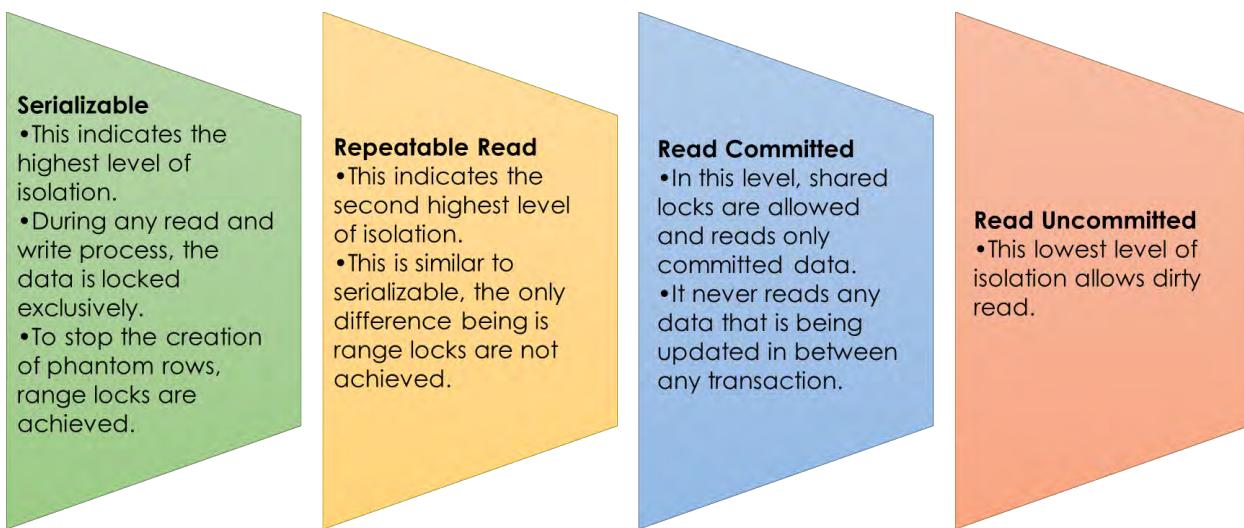
        cmd1.ExecuteNonQuery();
    }
    using (var conn2 = new SqlConnection(connStrDb2))
    {
        conn2.Open();
        var cmd2 = conn2.CreateCommand();
        cmd2.CommandText = string.Format("INSERT INTO T2
            values(2)");
        cmd2.ExecuteNonQuery();
    }
    scope.Complete();
}

```

6.5.3 Transaction Isolation Level

A locking mechanism isolates one transaction from another. This locking mechanism is introduced by the Transaction Management System. The behavior of this locking policy depends on the level of isolation set for each transaction.

Following are the four important isolation levels:



6.6 Content Delivery Networks (CDNs)

Web content can be delivered to the users using a distributed network of services, called Content Delivery Network (CDN). Cached content is stored in Point-of-Presence (POP) locations on edge servers. These will be located near end-users to reduce latency.

High-bandwidth content is delivered to the users by Azure CDN. Content is cached at physical nodes that are strategically placed across the world. Dynamic content that cannot be cached is accelerated by Azure CDN. CDN POPs perform this by leveraging various network optimizations. For example, optimizing a route to bypass Border Gateway Protocol (BGP).

Azure CDN helps to deliver Website assets. Following are its benefits:

Good performance and easy to use	Massive scaling	Handling the user request and content
Very useful for an application when various round-trips are required to add the content.	Helps in handling instantaneous high loads. For example, start of a product launch event.	The user requests are assigned and content is sent directly from edge servers. This prevents traffic from being formed at the origin server.

6.7 Capabilities and Functionalities of Azure Services for Master Data Processing

Master data processing involves handling and analyzing vast amounts of data to derive insights, both in batch and real-time scenarios.

6.7.1 Azure HDInsight

Its functionalities include:

- Hadoop and Spark-Based Analytics: Azure HDInsight provides managed clusters for Hadoop, Spark, HBase, and other big data frameworks.
- Batch Processing: Enables the processing of large volumes of data in batches using technologies such as MapReduce for distributed processing.
- Scalability: Scales clusters based on demand, allowing processing of extensive datasets efficiently.
- Machine Learning Integration: Allows integration with machine learning frameworks for predictive analytics on large datasets.

Some of its Use Cases are as follows:

- Batch Processing of Big Data: Ideal for scenarios requiring massive-scale batch processing, such as log processing, ETL (Extract, Transform, Load) tasks, and historical data analysis.

- Machine Learning and Predictive Analytics: Utilized for training machine learning models on large datasets.

6.7.2 Azure Databricks

Its functionalities include:

- Apache Spark-Based Platform: Offers a collaborative environment built on Apache Spark, enabling interactive querying, data visualization, and machine learning.
- Unified Data Processing: Combines data engineering, data science, and analytics in a unified platform.
- Real-time and Batch Processing: Supports both real-time and batch processing through its spark-streaming capabilities.
- Scalability and Collaboration: Allows teams to collaborate on data engineering and analysis tasks in a scalable manner.

Some of its Use Cases are as follows:

- Real-time Data Processing: Suited for scenarios requiring both real-time and historical data analysis, such as fraud detection, IoT data processing, and real-time analytics.
- Data Engineering and Collaboration: Ideal for collaborative data engineering tasks such as data cleaning, transformation, and feature engineering for machine learning.

6.7.3 Azure Stream Analytics:

Its functionalities include:

- Real-Time Streaming Analytics: Enables processing and analyzing streaming data from various sources such as IoT devices, sensors, social media, and so on.
- Scalability and Low Latency: Scales dynamically to handle high throughput with low latency, making it suitable for real-time decision-making.
- Integration with Azure Services: Easily integrates with other Azure services for data visualization, storage, and event-driven actions.

Some of its Use Cases are as follows:

- Real-time Insights and Alerts: Used for scenarios requiring immediate insights from streaming data, such as monitoring systems, detecting anomalies, and triggering alerts.
- Continuous Data Processing: Suited for applications that demand continuous data processing and analysis, such as clickstream analysis and sentiment analysis in social media feeds.

6.8 Summary

- ✓ Data Ingestion into Azure involves transferring and loading diverse data from sources.
- ✓ Azure Data Service includes a number of features such as Data Storage, SQL Database as a Service, NoSQL database as a service, Big Data, and so on.
- ✓ Data Caching in Azure enhances performance by storing frequently accessed data in memory, reducing latency.
- ✓ Distributed Transactions in SQL Azure ensures transactional consistency across multiple databases or instances in SQL Azure.
- ✓ Azure Storage is a scalable object store that is massive for data objects. It is a file system service for the cloud and a reliable messaging store.
- ✓ Content Delivery Network (CDN) is a distributed network of services that can be used to deliver Web content to the users.
- ✓ Azure Services Capabilities encompass a wide range, from Azure HDInsight and Databricks for big data processing to Stream Analytics for real-time insights, each tailored for specific data processing requirements within Azure.

6.9 Test Your Knowledge

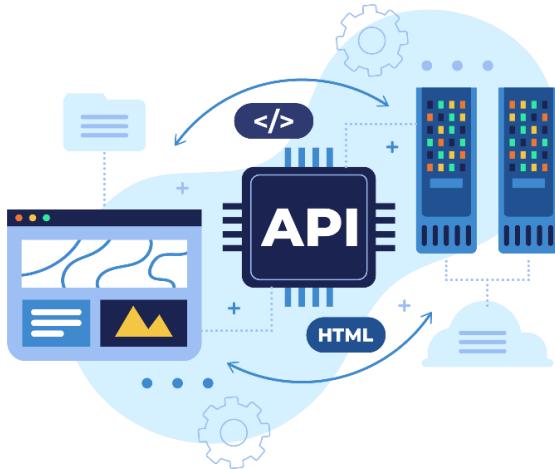
1. Which Azure service is commonly used for orchestrating data movement and transformation workflows across various sources and Azure services during the data ingestion process?
 - A. Azure Data Lake Storage
 - B. Azure Data Factory
 - C. Azure Blob Storage
 - D. Azure Event Hubs
2. Which Azure database service is best suited for globally distributed applications and offers multi-model capabilities for diverse data models?
 - A. Azure SQL Database
 - B. Azure Cosmos DB
 - C. Azure Database for PostgreSQL
 - D. Azure Database for MySQL
3. What is the primary purpose of data caching services such as Azure Cache for Redis within the Azure ecosystem?
 - A. Ensuring data durability
 - B. Reducing latency and improving application responsiveness
 - C. Enabling real-time data analytics
 - D. Streamlining data transformation processes
4. What is the main purpose of distributed transactions in SQL Azure?
 - A. Optimizing query performance in large-scale databases
 - B. Enforcing data security and encryption
 - C. Ensuring transactional consistency across multiple databases or instances
 - D. Managing schema changes and data migration seamlessly
5. Which Azure service is specifically tailored for real-time streaming analytics on data from various sources, such as IoT devices or social media feeds?
 - A. Azure Databricks
 - B. Azure HDInsight
 - C. Azure Stream Analytics
 - D. Azure Synapse Analytics

6.9.1 Answers to Test Your Knowledge

- 1. B
- 2. B
- 3. B
- 4. C
- 5. C

Try It Yourself

1. Download a sample dataset and perform cleaning, formatting, and transformation using Azure Data Factory.
2. Perform one-time data ingestion on a sample dataset.



SESSION 07

DESIGN AND IMPLEMENTATION OF WEB API

Overview:

This session introduces ASP.NET Web API and its features. It explains how to implement it and how to perform routing in Web API. It also explains dependency injection and elaborates on how to create OData Services.

Learning Objectives

In this session, students will learn to:

- Describe ASP.NET Web API
- Explain how to implement Web API with and without authentication
- Describe how to perform routing in Web API
- Explain how to implement database operations using Web API
- Describe dependency injection
- Explain how asynchronous and synchronous actions are implemented in Web API
- Describe OData services

7.1 Overview of Web API

Due to an increase in usage of smartphones and other similar devices to access services over the Web, a technology that simplifies providing services to a variety of clients is required. ASP.NET Web API is one such technology provided by Microsoft that allows clients to create Web services targeting diverse clients. Using ASP.NET Web API, client requests can be handled, and responses can be sent back using content types, such as JavaScript Object Notation (JSON) or XML.

ASP.NET Web API is a powerful framework provided by Microsoft, enabling the creation of HTTP services.

These are easily accessible to a wide range of clients, including Web browsers and mobile devices.

Ideal for building RESTful applications on the .NET Framework, it supports responses in JSON and XML formats, catering to diverse client requirements. Beyond its basic functionality, ASP.NET Web API offers robust features such as authentication and authorization schemes. This also includes OAuth and token-based authentication, to secure Web applications effectively. It employs a flexible routing system to direct HTTP requests to the appropriate controllers, enhancing API usability and maintainability.

Moreover, the framework facilitates action filters for implementing cross-cutting concerns such as logging and exception handling and supports dependency injection for creating de-coupled, easily testable code. With built-in support for content negotiation, ASP.NET Web API ensures that services can deliver data in the most suitable format for each client, further extending its versatility.

Integration with ASP.NET Core marks a step forward, streamlining the development of Web applications and services within a unified framework. This comprehensive feature set makes ASP.NET Web API a preferred choice for developers aiming to deliver high-quality, scalable, and secure Web services.

7.1.1 Evolution of Web API

The most basic rule for an API evolution is that the externally observable behavior of an API (from the perspective of the clients) cannot be modified, once the API has been published. Already little modification of the API might break some of the clients consuming the API. It is not possible to update each consumer or at least unrealistic since they are under the control of different owners. Thus, longevity and stability are crucial aspects of published APIs.

The restriction imposed through this rule would possibly sound intense and even counter-intuitive since APIs regularly evolved the use of an Agile development approach. Agile approaches are primarily based on feedback loops and the concept of many incremental adjustments within the software. The Agile approach of development still applies to new or unpublished APIs.

Before the publication of the API, any modification can be done in an Agile manner. When APIs are published, they become available for consumers and it must be assumed that the consumers build apps relying on the APIs. Published APIs cannot be modified in an Agile manner. At least, APIs want to stay backward and forward compatible, so that old clients do not break and new clients can use the new and improved features.

7.1.2 ASP.NET Web API

The attributes of Model View Controller (MVC) which include routing, controllers, filters, action results, model binders, Inversion of Container (IOC), and dependency injection are part of the ASP.NET Web API. While it is deemed to be similar to ASP.NET MVC, it is considered to be a part of the core ASP.NET platform and not the MVC Framework.

Figure 7.1 shows the framework of ASP.NET Web API.

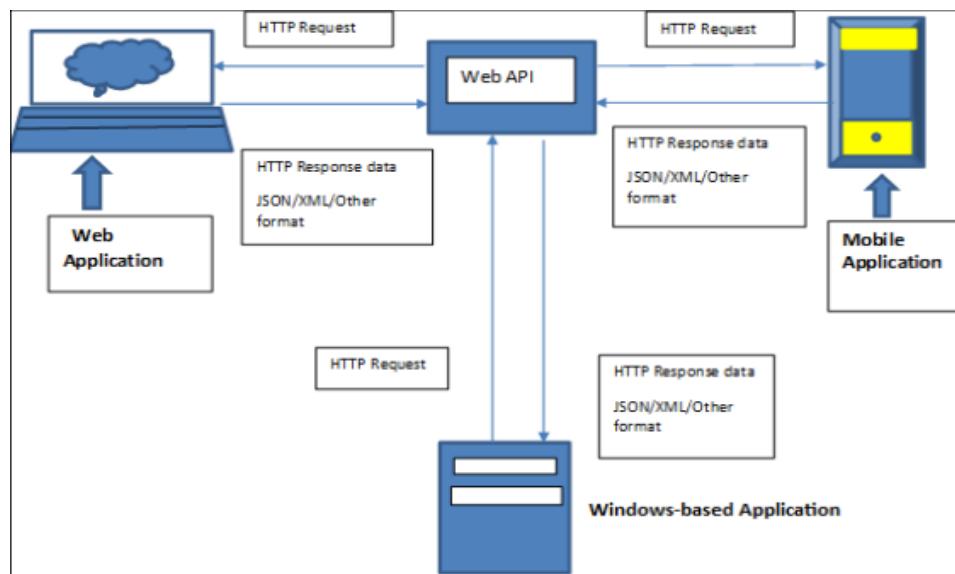


Figure 7.1: ASP.NET Web API

Some important features of ASP.NET Web API are as follows:

Simple Programming Model: Enables easy access and updates HTTP requests and responses in an ASP.NET Web API application. In an ASP.NET Web API application, the similar programming model is utilized on the server as well as on the client. The client may be any .NET application.

Content Negotiation: Enables client and server to negotiate data format that the service returns as a response. By default, an ASP.NET Web API provides support for JSON, XML, and URL-encoded formats. It also provides flexibility to extend the default data format by adding custom formats or by replacing the default negotiation strategy with a custom one.

Request Routing: Provides a routing module that is accountable for mapping request URLs to the precise controller action. Using the routing module, custom routes can be defined in a centralized location without any additional configurations.

Filters: Provides filters to carry out certain logic just before and after an action method of an API, the controller is invoked. For example, a filter can be used to analyze the time that a method takes to process a request.

Flexible Hosting: Provides flexibility to be hosted with other .NET applications, such as ASP.NET MVC and ASP.NET WebForms. ASP.NET Web API additionally helps self-hosting, where the application is hosted within a process that runs a console application. Therefore, an ASP.NET Web API application does not require a Web server such as IIS to be hosted for providing services.

7.1.3 Type of Request and Response

HTTP Request

HTTP requests are used to communicate over the Web. When a user enters a URL in the address bar of a browser, an HTTP request is sent to an application running on the server represented by the URL.

The application processes the request and returns an HTTP response. The browser, on receiving the response, displays the response to the user. HTTP provides different types of request methods based on the type of operations that the request requires. The four essential HTTP methods are as follows:

Get:	Post:	Put:	Delete:
The HTTP get method requests the server to retrieve a resource.	The HTTP post method requests the server that the target resource must process the data contained in the request.	The HTTP put method requests the server to create or update the data contained in the request.	The HTTP delete method requests the server to delete a resource.

HTTP request headers contain following information:

Host:	Specifies the domain name of the application whose resource is being accessed.
Accept:	Specifies the content type, also known as Multipurpose Internet Mail Extensions (MIME) type that the request expects as the response.
Accept-Language:	Requests the server to create or update a request.
Connection:	Specifies whether the server should use the same connection (keep-alive value indicates this) for HTTP communication instead of creating a new
Keep Alive:	Specifies the period in seconds for which the server has to use the identical connection for HTTP communication.

HTTP Response

When a server gets an HTTP request, it sends back an HTTP response. This response may be the output of the requested resource or an HTTP status code to signify that the requested resource is not available.

Similar to an HTTP request, an HTTP response contains response headers and a response body.

The HTTP response specifies the HTTP response status code along with following key HTTP response headers:

Date:	Specifies the date and time whilst the response is being dispatched from the server.
Server:	Specifies the server that is managing the request-response exchange.
Last modified:	Specifies the date and time at which the resource was last changed.
Content length:	Specifies the size of the response body in a decimal number of octets.
Content-type:	Specifies the sort of content that the response contains.

7.2 Implementing Web API

Following are the steps to create a Web API application using Microsoft Visual Studio 2022:

1. Create a new project as shown in Figure 7.2.

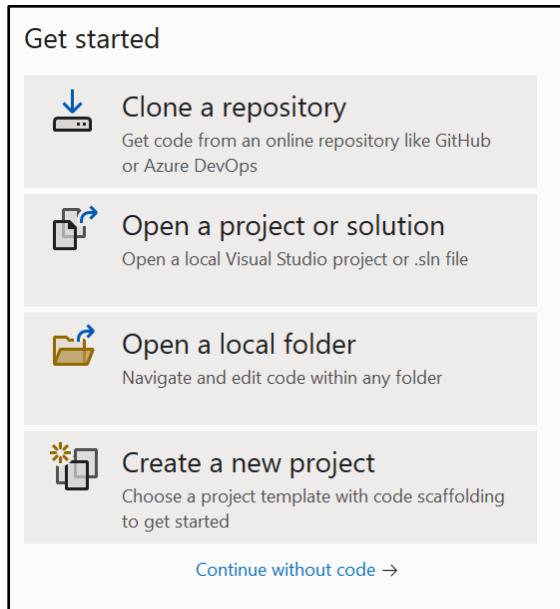


Figure 7.2: Creating a New Project

2. Search ASP.NET Core. After selecting ASP.NET Core Web API template, click **Next**, as shown in Figure 7.3.

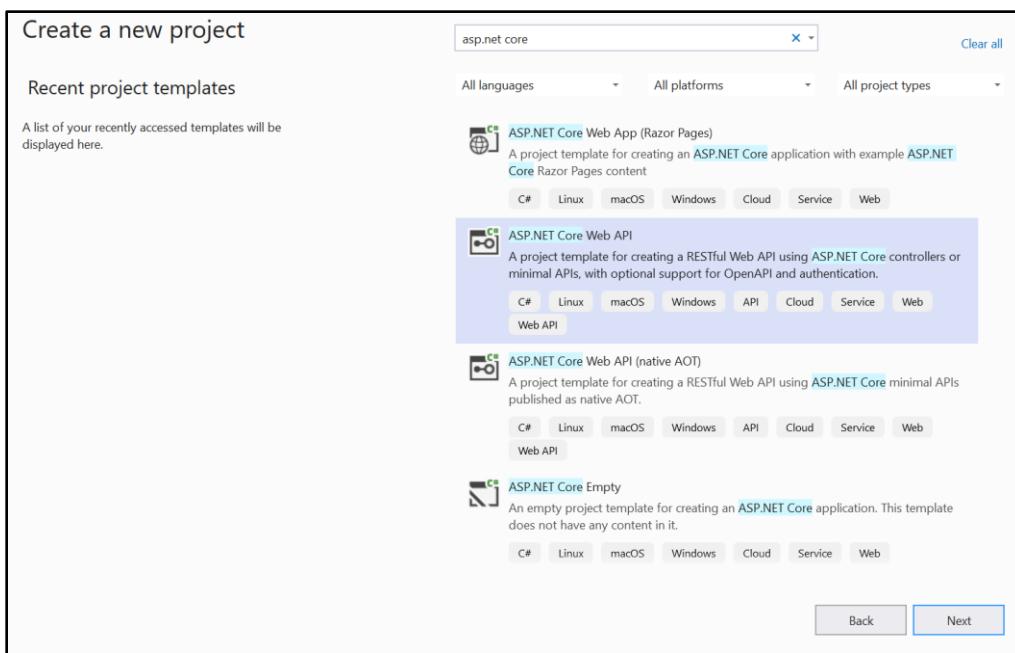


Figure 7.3: Selecting Template

3. Add a project name and location in the text boxes as shown in Figure 7.4.

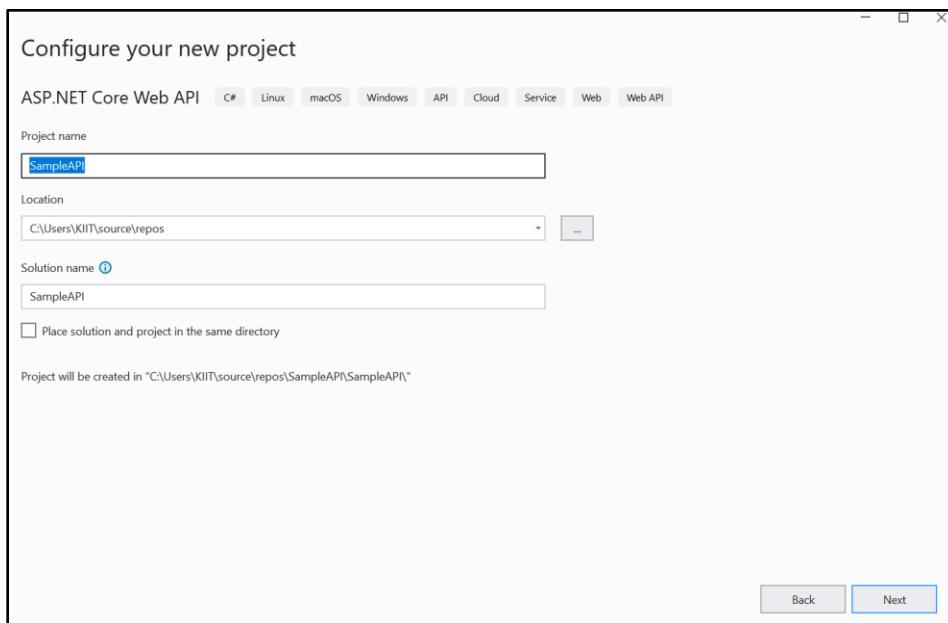


Figure 7.4: Naming the Project

An API project is successfully created. The project structure is generated in Solution Explorer as shown in Figure 7.5.

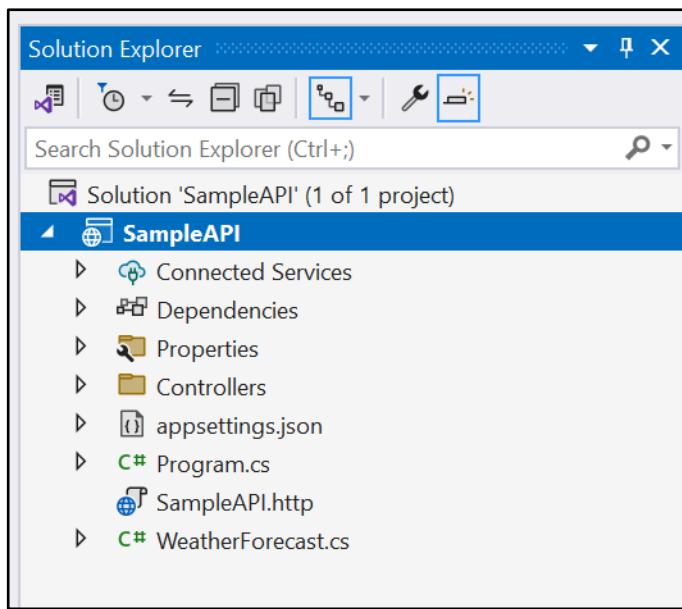


Figure 7.5: Solution Explorer Showing Project Structure

4. Run this API project without making any modifications to the auto generated files in the project. Figure 7.6 shows the output in the browser.

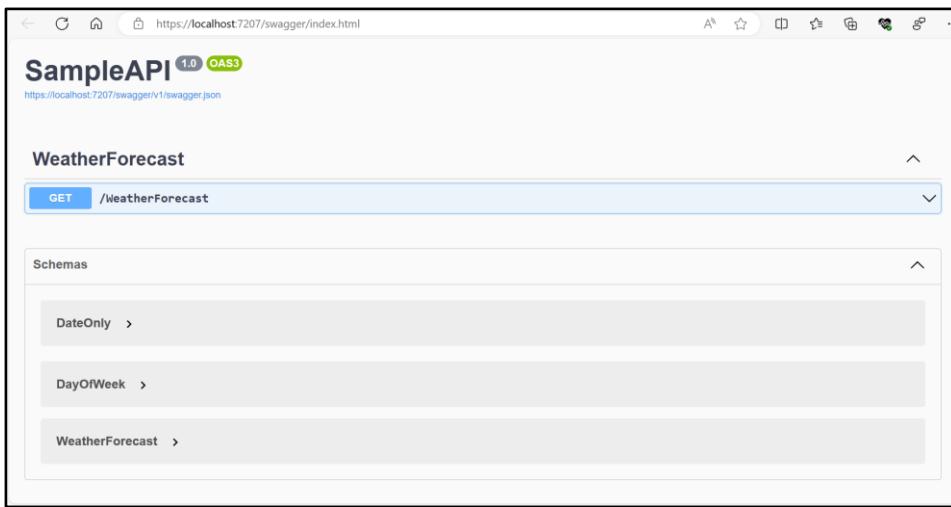


Figure 7.6: Output

The output of the Weather API is displayed on Swagger, which is an open-source software tool to design, build, and utilize RESTful Web API.

Following steps will explain how to create Web API for Get HTTP method:

Step 1: Create a new .NET Class Library by right-clicking the solution in Solution Explorer and selecting **Add → New Project** called Sample.Data.

Step 2: Add three folders; Models, Interface, and Repository in **Sample Data Class** library. Refer to Figure 7.7.

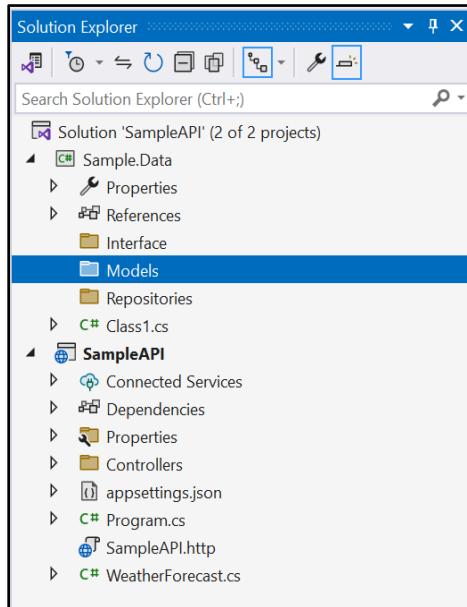


Figure 7.7: Creating Class Library with Three Folders

Step 3: Create a Model class in the Model folder called `Customer`. Refer to Code Snippet 1.

Code Snippet 1:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Sample.Data.Model
{
    public class Customer{
        public int CustomerId { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Address { get; set; }
    }
}
```

Step 4: In the Interface folder, create an interface called `ICustomer`. Refer to Code Snippet 2.

Code Snippet 2:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Sample.Data.Model;
namespace Sample.Data.Interface{
    internal interface ICustomer
    {
        List<Customer> GetAllCustomer();
        Customer GetCustomer(int id);
    }
}
```

Step 5: In the Repository folder, create a class called `CustomerRepo` and implement the `ICustomer` interface in it as shown in Code Snippet 3. This class will interact with the database.

Code Snippet 3:

```
using Sample.Data.Interface;
using Sample.Data.Model;
using System.Collections.Generic;
using System.Linq;
namespace Sample.Data.Repository
{
    public class CustomerRepo:ICustomer
    {
        List<Customer>lisCustomers=newList<Customer>
        {
            new Customer{CustomerId=1,
                FirstName="FirstName1",
                LastName="LastName1", Address=" Austin "
            },
            new Customer{CustomerId=2,
                FirstName="FirstName2",
                LastName="LastName2", Address=" Austin "
            },
            new Customer{CustomerId=3,
                FirstName="FirstName3",
                LastName="LastName3", Address=" Austin "
            },
            new Customer{CustomerId=4,
                FirstName="FirstName4",
                LastName="LastName4", Address=" Austin "
            },
            new Customer{CustomerId=5,
                FirstName="FirstName5",
                LastName="LastName5",Address="Austin"
            },
        };
        public List<Customer> GetAllCustomer()
        {
            returnlisCustomers;
        }
        publicCustomerGetCustomer(intid)
        {
            returnlisCustomers.FirstOrDefault(x=>
                x.CustomerId==id);
        }
    }
}
```

Here, the code has dummy data in the form of FirstName1, LastName1, and so on. In an actual example, you just replace with proper customer names.

Step 6: Create a controller in the controller folder called CustomerController. Add a new controller by right-clicking the controller folder as shown in Figure 7.8. Name it as shown in Figure 7.9.

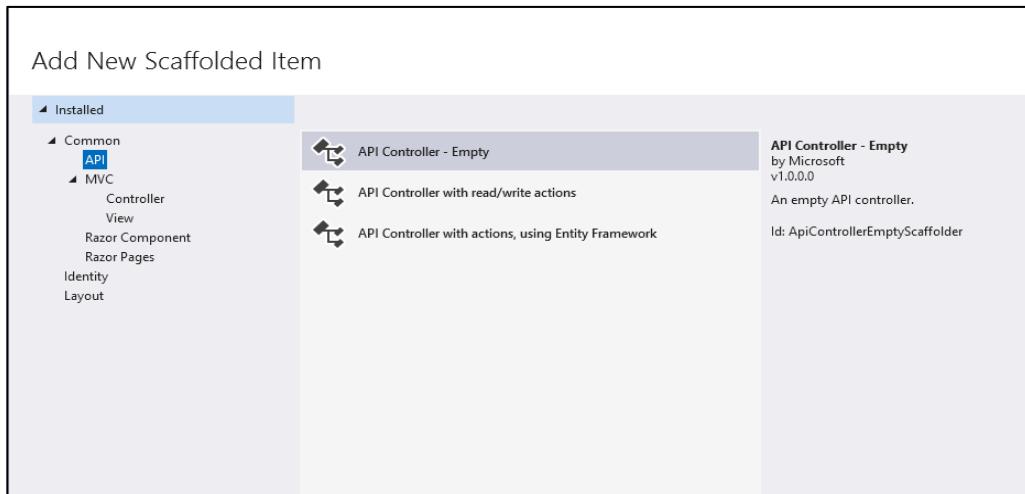


Figure 7.8: Selecting Controller

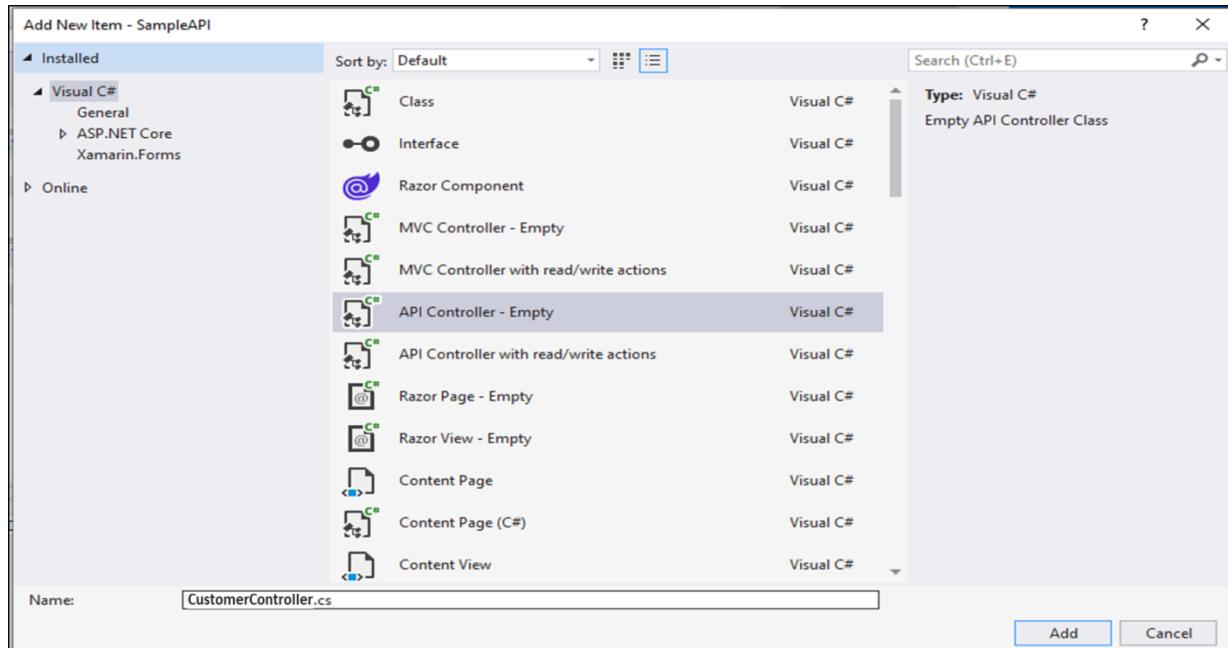


Figure 7.9: Naming the Controller

Step 7: Right-click Sample API – Dependencies and add a project reference as shown in Figure 7.10.

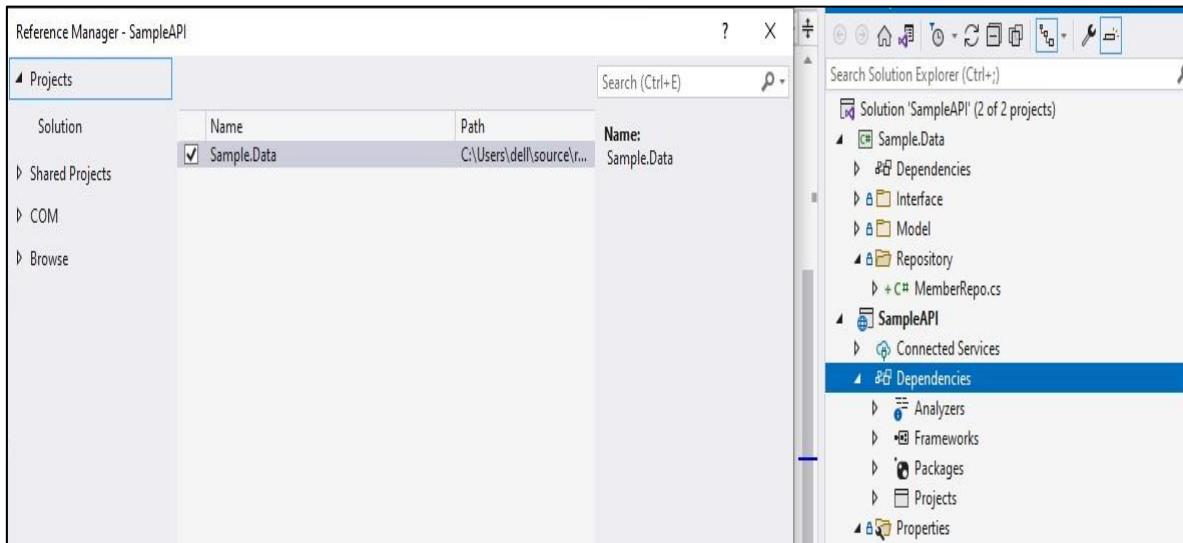


Figure 7.10: Adding Dependencies

Step 8: Execute Customer API Project and the screen shown in Figure 7.11 will appear.

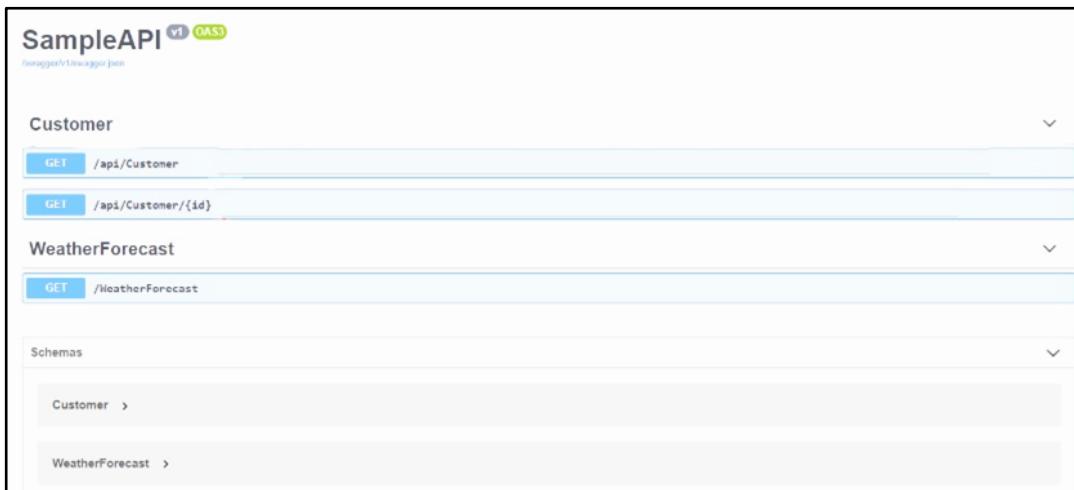


Figure 7.11: Output Showing Options

Step 9: Click the GET button on the left of the api/Customer box as shown in Figure 7.11. The outcome will be as shown in Figure 7.12.

The screenshot shows a browser window with the URL <https://localhost:7207/swagger/index.html>. The page displays a "Server response" section for a "GET /weatherforecast" request. The response code is 200, and the response body is a JSON array of five weather forecast entries. Each entry includes a date, temperature in Celsius, temperature in Fahrenheit, and a summary. The JSON is as follows:

```
[{"date": "2024-02-24", "temperatureC": 18, "temperatureF": 64, "summary": "Mild"}, {"date": "2024-02-25", "temperatureC": 48, "temperatureF": 118, "summary": "Warm"}, {"date": "2024-02-26", "temperatureC": 2, "temperatureF": 35, "summary": "Hot"}, {"date": "2024-02-27", "temperatureC": -20, "temperatureF": -3, "summary": "Scorching"}, {"date": "2024-02-28", "temperatureC": null, "temperatureF": null, "summary": null}]
```

At the bottom of the response body, there are "Copy" and "Download" buttons. Below the response body, the "Response headers" section shows:

```
content-type: application/json; charset=utf-8
date: Fri, 23 Feb 2024 12:33:11 GMT
```

Figure 7.12: Outcome of GET

Similarly, Web APIs can be created for HTTP methods POST, PUT, and DELETE.

7.2.1 Using Third-Party Tool for Testing Web API

Once the Web APIs are developed, they are tested using third-party tools. Among all tools, Postman is one of the best tools for API testing.

To download the Postman application, use following link:

<https://www.getpostman.com/apps>

Once downloaded and installed, open Postman. Following steps depict how to test the APIs:

Step 1: Copy the Web API URL of HTTP Get method (for example: <https://localhost:7207/swagger/index.html>) from the browser and paste it into the Postman URL Tab as shown in Figure 7.13.

The screenshot shows the Postman interface with a 'GET New Request' selected. The URL is set to `https://localhost:7207/swagger/index.html`. The 'Body' tab is selected, showing the raw JSON response:

```

1 <!-- HTML for static distribution bundle build -->
2 <!DOCTYPE html>
3 <html lang="en">
4
5 <head>
6   <meta charset="UTF-8">
7   <title>Swagger UI</title>
8   <link rel="stylesheet" type="text/css" href="/swagger-ui.css">

```

Figure 7.13: Web API Testing in Postman - Get Method

Step 2: Test the POST method to create a region. To do this, copy the POST method and paste it into the Postman URL or edit the headers. Change the HTTP request to POST in the body part of the request. JSON data has to be passed as raw content for the creation of a region. The details are shown in Figure 7.14.

The screenshot shows the Postman interface with a 'POST New Request' selected. The URL is set to `https://localhost:7207/swagger/index.html`. The 'Headers' tab is selected, showing the following configuration:

Key	Value	Description	... Bulk Edit	Presets
Region	US East			
Key	Value	Description		

Figure 7.14: Web API Testing in Postman - Post Method

Step 3: Once all the information is set, click **b**. This creates the customer and displays the Success message.

Step 4: To test if the record is created, use the Get method to call. The left of the window displays the history of recent calls.

7.3 Hosting Web API in Microsoft Azure Web App

If developers want the fastest way to publish their Web project, they should consider using Azure App Service. The hosting service is reliable and can grow and fix itself. A resource group is made when the first ASP.NET Web app that uses this service works well. This group has a service plan, an Azure Web app, and the application that has been used.

Steps to publishing a Web API application to the Azure App service are as follows:

Step 1: Assuming that an application StudentServices has been created, open it in Visual Studio 2022. Right-click the StudentServices project and choose **Publish** from the menu that appears. This will bring you to the profile page for Publish.

Step 2: As there is already a publish profile, a new one should be made for Web Apps publish. The Publish profiles can be seen in Figure 7.15. To make a profile, click the link that says **New Profile**.

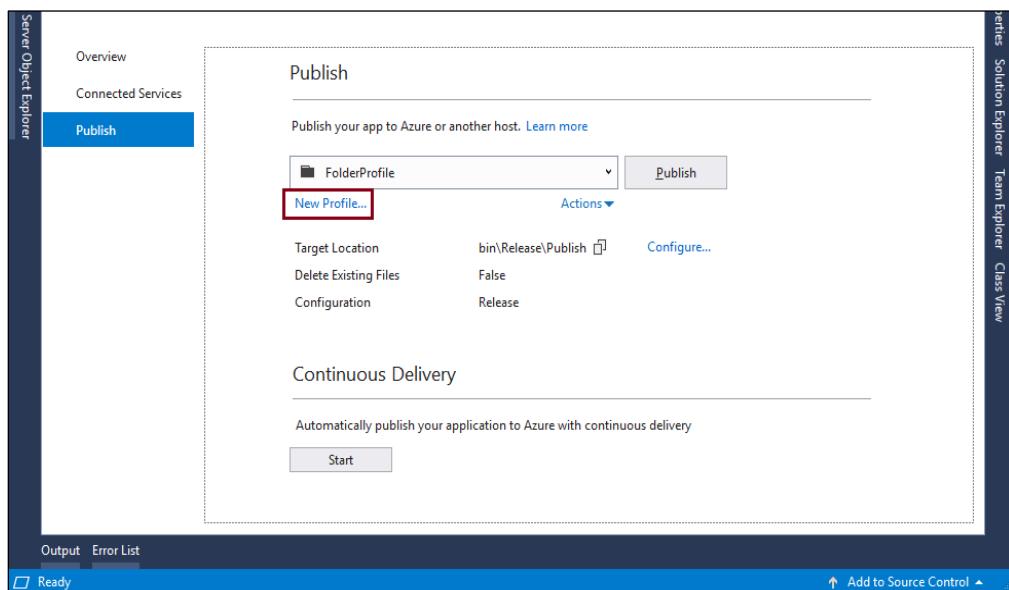


Figure 7.15: Publish Profiles

Step 3: To make an Azure Web App, choose the **App Service** from the context menu for the Publish option. Make sure to choose the 'New' option. Figure 7.16 shows the choices for the Web App.

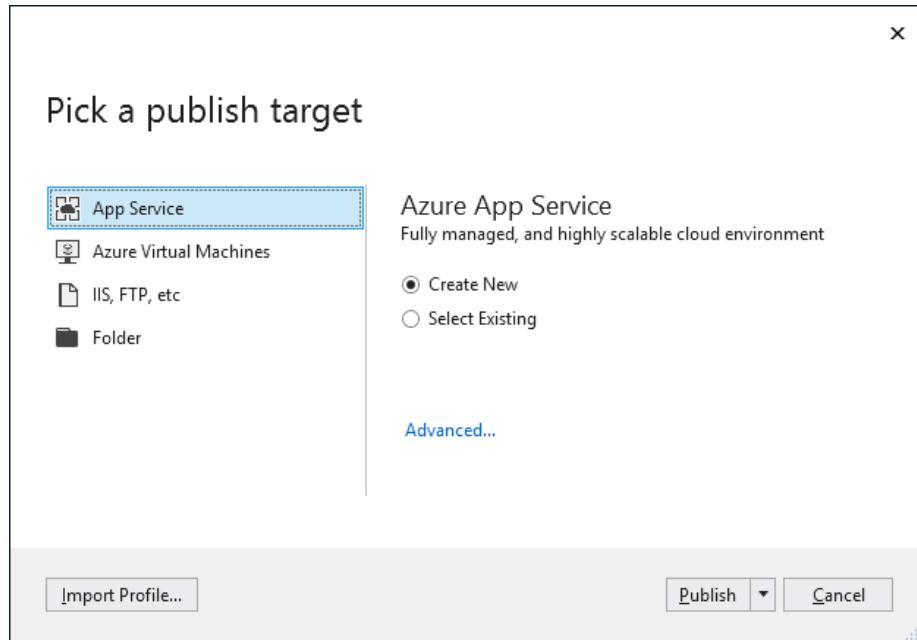


Figure 7.16: Web App Publish Profiles

Step 4: In the next window, which is called 'Create App Service,' choose **Already have an account? Sign in**. Sign in with your Azure account and the Account will appear in the upper right corner. Choose the account from the drop-down menu next to the Account. This will make a Web application with a name you choose for you automatically. Figure 7.17 shows how to change the name of the app.

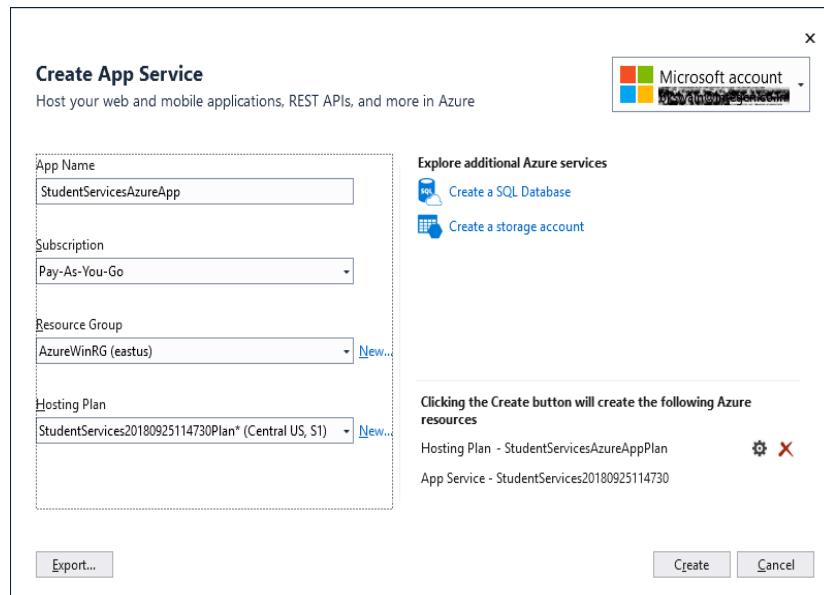


Figure 7.17: Web App Settings

Step 5: Select **Create**. This will make the Publish profile and create the Web App and Web API automatically. After the publish operation is done, the Web API application will open in the default browser by itself. The address is <http://studentservicesazureapp.azureWebsites.net>.

Step 6: You can test the API using the Postman app.

7.4 Routing in Web API

The routing procedures of ASP.NET MVC are similar to that of Web API. While the HTTP method is used in Web API, the URI method is used in ASP.NET MVC for selection of an action. Web API transfers a HTTP request to a specific action method on a controller.

7.4.1 Default Routing Using Routing Tables

A controller is a type of class found in ASP.NET Web API, which can manage HTTP requests. The public methods of controllers are known as action methods or actions. Within the framework of Web API, when a request arrives, it is routed to an action. The framework uses a routing table to decide which of the actions are required to be called. A Visual Studio project has a template for Web API that creates a default route.

Code for the same is depicted in Code Snippet 4.

Code Snippet 4:

```
public static void Register(HttpConfiguration config)
{
    // Web API configuration and services and routes
    config.MapHttpAttributeRoutes();
    config.Routes.MapHttpRoute(
        name: "DefaultApi",
        routeTemplate: "api/{controller}/{id}", defaults: new { id =
RouteParameter.Optional
    }
```

The route mentioned in Code Snippet 4 is part of the WebApiConfig.cs file, which is under the App_Start directory as shown in Figure 7.18.

This assumes that a project named StudentServices has been created.

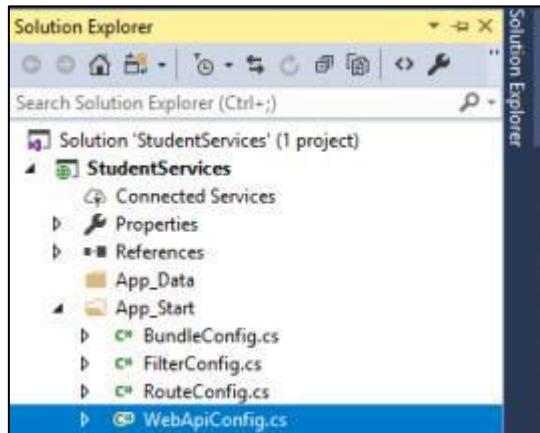


Figure 7.18: Project Named 'StudentServices'

Each item in the routing table has a route template embedded in it. For Web API, the default template is 'api/{controller}/{id}'. The term 'api' refers to the actual path segment while {controller} and {id} are the placeholder variables.

As discussed, when an HTTP request arrives, its URI is compared with the route templates of the routing table. If the comparison fails, a 404 error is displayed. For example, following URIs compare the (api/{controller}/{id}) format as defined in the default routes in Code Snippet 4.

```
/students Returns the list of students  
/api/students/1 Returns student for Student id 1
```

There are two types of routing supported by Web APIs:

- Convention-based Routing
- Attribute Routing

7.4.2 Convention-Based Routing

Route templates are used by Web API to determine which controller and action method to execute rather than concrete physical files. At least one route template must be added into the route table in order to handle various HTTP requests. Using the default routing template, the Web API chooses an action using an HTTP method as shown in Code Snippet 5.

Code Snippet 5:

```
routes.MapHttpRoute( name:"DefaultApi",  
    routeTemplate: "api/{controller}/{id}",  
    defaults:new{id=RouteParameter.Optional}  
);
```

7.4.3 Attribute Routing:

Attribute routing uses [Route()] attribute to define routes. Attribute routing provides flexibility for the URLs used in Web API 2. As shown in Code Snippet 6, the description of hierarchies of resources is one such example of the creation of URLs.

Attribute routing gives you more control over the URLs in your Web API. For example, you can easily create URLs that describe hierarchies of resources.

Code Snippet 6:

```
[Route ("students/{studentId}/marks")]
public IEnumerable<Student>GetOrdersByStudent(intstudentId)
{
    ...
}
```

7.5 Implementing Database Operation Using Web API

Implementation of the HTTP GET (by id), PUT, and DELETE methods are now explained in detail.

Following are steps to implement the DELETE method:

Step 1: Retrieve the student information by Id. Code Snippet 7 represents the GET method by Id.

Code Snippet 7:

```
[HttpGet] [Route ("api/Student/GetById/{id}")]
public IHttpActionResult GetById(int id) {
    var student=_context.Students.SingleOrDefault( e => e.StudentId
    == id);
    return Ok(student);
```

Step 2: Implement the Edit method. For this, first, create a new Model class in the Models folder for editing purposes. Name it as **EditStudentModel**. Add properties and validation attributes as shown in Code Snippet 8.

Code Snippet 8:

```
using System;
using System.ComponentModel.DataAnnotations;

namespace StudentServices.Models{
    public class EditStudentModel {
        [Required(ErrorMessage="StudentId Is required")]
        [Display(Name = "Student Id")]

        public int StudentId { get; set; }
        [Required(ErrorMessage="Student Name is required")]
        [Display(Name = "Student Name")]
        public string StudentName { get; set; }
        [Required(ErrorMessage="Student Email is required")]
        [Display(Name = "Student Email")]
        public string StudentEmail{get;set;}
        [Display(Name = "Date of Join")]
        public DateTime? StudentJoinDate{get;set;}
    }
}
```

Step 3: Add a new method in the Student Controller named `Modify()`. Add the code given in Code Snippet 9 for the modification method. Note that the `Modify()` method is marked as the HTTP PUT method.

Code Snippet 9:

```
[[HttpPost]
public IHttpActionResult Modify(EditStudentModel studentModel){
    var student = new Student(){
        StudentId=studentModel.StudentId,
        StudentName = studnentModel.StudentName, StudentEmail
        = studentModel.StudentEmail,
        StudentJoinDate=studentModel.StudentJoinDate
    };
    _context.Entry(student).State=EntityState.Modified;
    _context.SaveChanges();
    return Ok("Success");
}}
```

Step 4: Add the codes in Code Snippet 10 to implement the Delete method.

Code Snippet 10:

```
[HttpDelete]
public IHttpActionResult Delete(int id){
    var student = _context.Students.SingleOrDefault(e => e.StudentId
    == id);
    _context.Students.Remove(student ?? throw new
    InvalidOperationException());
    _context.SaveChanges();
    return Ok("Success");
}
```

Step 5: Use Postman to test each HTTP methods. Table 7.1 shows how each HTTP method is implemented.

HTTP Method	URI Path	Action	Parameter/Body
POST	api/Student	Create()	Student Json object as the body
GET	api/Student	GetAll()	none
GET	api/Student/1	GetById()	1
PUT	api/Student	Modify()	Student Json object as the body
DELETE	api/Student/1	Delete()	1

Table 7.1: HTTP Method Implementation

7.6 Implementing Identity for Authentication

When it comes to enterprise-level applications, an aspect that requires prime concern is security. Security is considered extremely important when exposing business data or processes via other services.

Regarding security for Web API, there is no systematized approach to ensure it. Web API allows users to develop their security methods depending on their application. It is vital to understand two important phases of Web API security implementation, which are as follows:

Authentication:

This method is used to validate the identity of the user.

Authorization:

This method is used to determine if a given user is permitted to carry out a task.

To understand these phases/processes, consider an example. The user, John uses his unique username and password to log in. This password is then utilized by the server to validate John, then this is authentication phase. Once John has successfully logged in, consider that he can access a particular file, but he cannot edit the file. Then, it is an authorization phase.

Figure 7.19 represents authentication and authorization in Web API.

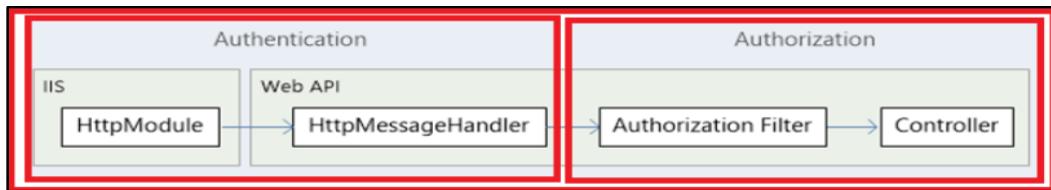


Figure 7.19: Authentication and Authorization

7.6.1 Authentication

In Web API, authentication is performed in the host. To authenticate users, Internet Information Server (IIS) employs HTTP modules. Users have the provision to select authentication modules, which are configured in IIS or ASP.NET for their projects. If custom authentication is required, they can create HTTP modules to suit their requirements.

The host generates a principal called the `IPrincipal` object for authenticated users. This object helps in identifying the security context that has been chosen. This object is then appended to the current thread using `Thread.CurrentPrincipal`. The principal has also an `Identity` object, which holds all the data regarding the user.

When authentication for the user is successful, the `Identity.IsAuthenticated` property provides value 'true'. Otherwise, it provides the value 'false'.

7.6.2 Authorization

While authentication takes place in the host, authorization takes place near the controller. Authorization employs filters, which are applied before the controller action. For example, an employer can filter data with employee names, which returns data related to the employee.

When authorization is not successful, the filters are not applied and an error message is displayed. To obtain the current principal from a controller action, view the `ApiController.User` property. The `AuthorizeAttribute` filter is an integral part of Web API. This helps in ascertaining if the authentication is successful. If unsuccessful, it displays HTTP error code 401.

This filter can be utilized globally or at the controller or an individual level as follows:

Globally:

If it is necessary to prevent access for all Web API controllers, then, ensure that the `AuthorizeAttribute` filter is included in the global filter list within the `Global Register()` method.

Controller:

If it is necessary to prevent access for a particular controller, then, ensure that the filter is included as an attribute to the controller before the Controller class definition.

Action:

If it is necessary to prevent access for particular actions, then, ensure that the attribute is included in the action method.

7.6.3 Identity Authentication in Web API

Providing security to Web APIs is important so that developers can restrict users from accessing them. Following are the steps to create a Web API with Open Web Interface for .NET (OWIN) token-based authentication:

Step 1: Create a new ASP.NET Web application and give the name as StudentServices. Click **OK** and select the Application type as shown in Figure 7.20. Also, observe the Authentication type drop-down.

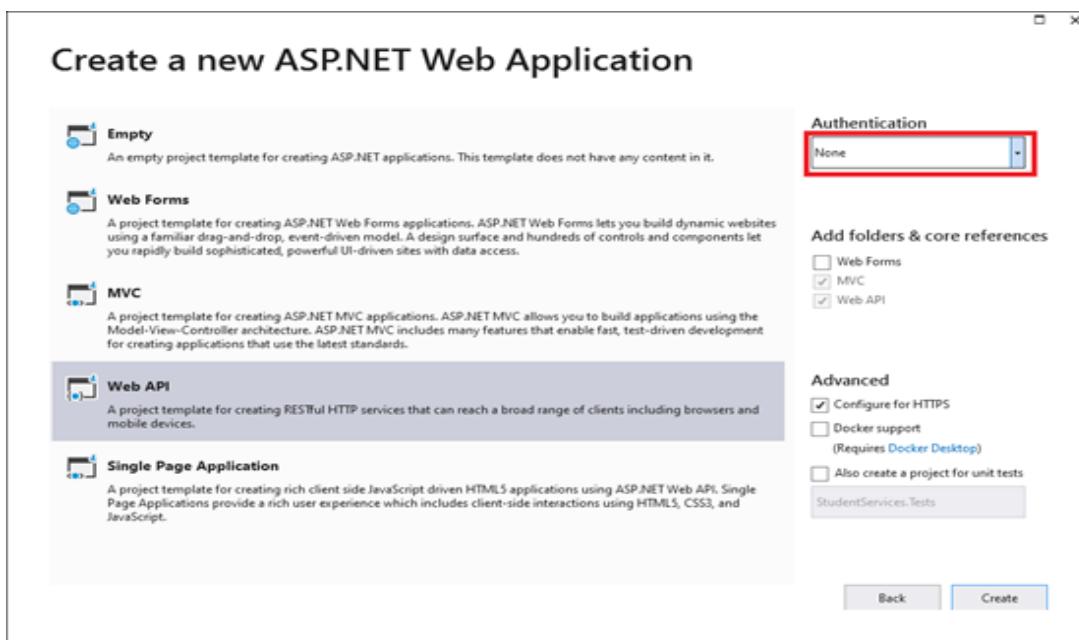


Figure 7.20: Application Type

Step 2: Click the Authentication drop-down to implement the authentication. A window with different authentication options appears as shown in Figure 7.21.

Select Individual User Account and click **OK**. It will return to the previous window.

Click **OK** again to complete the process.

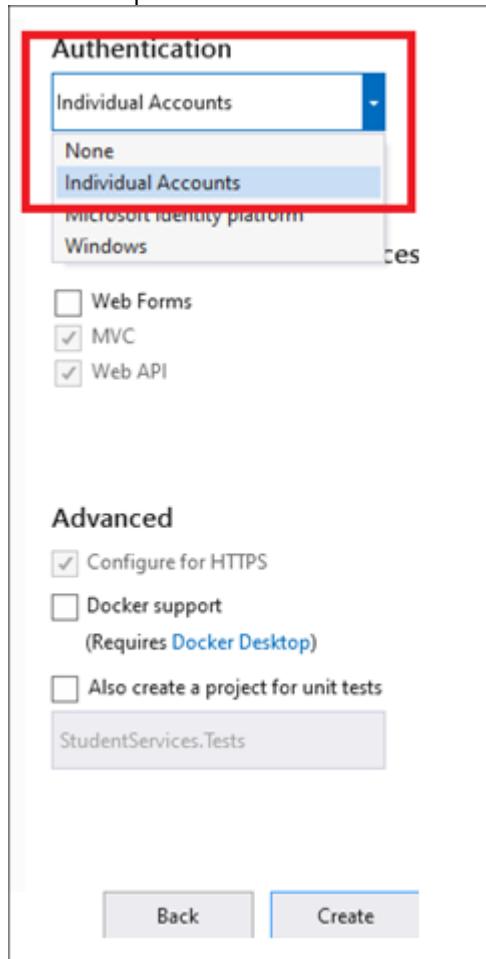


Figure 7.21: Authentication Options

Step 3: Once the application is created, expand the Solution Explorer. The AccountController can be seen in the Controllers folder as shown in Figure 7.22.

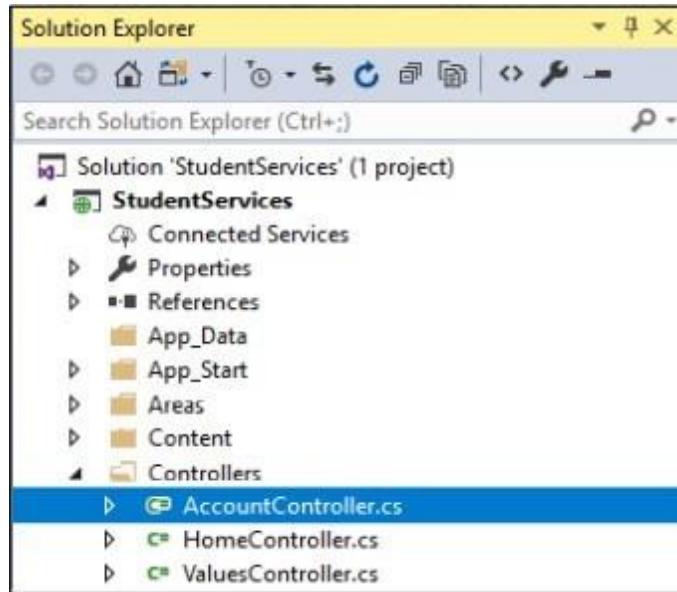


Figure 7.22: Account Controller

Step 4: Open web.config to update the Connection string. By default, OWIN authentication creates an authentication database inside the Data folder. Instead, create authentication tables in the Azure SQL Database. To achieve this, the connectionStrings section must be replaced as given in Code Snippet 11.

In Code Snippet 11, the database connection name is DefaultConnection.

Code Snippet 11:

```
<connectionStrings>
  <addname="DefaultConnection" connectionString="Data
Source=tcp:azureclouddb.database.windows.net,1433;
initial catalog=StudentDb;user id=AZDBAdmin;
password=AzDb@2016;MultipleActiveResultSets=True;" 
providerName="System.Data.SqlClient" />
</connectionStrings>
```

Step 5: Run the application. It can be observed that the default MVC application will also load, which is an API helper application. Click the API link to load the API helper page. The account-related API methods are listed as shown in Figure 7.23.

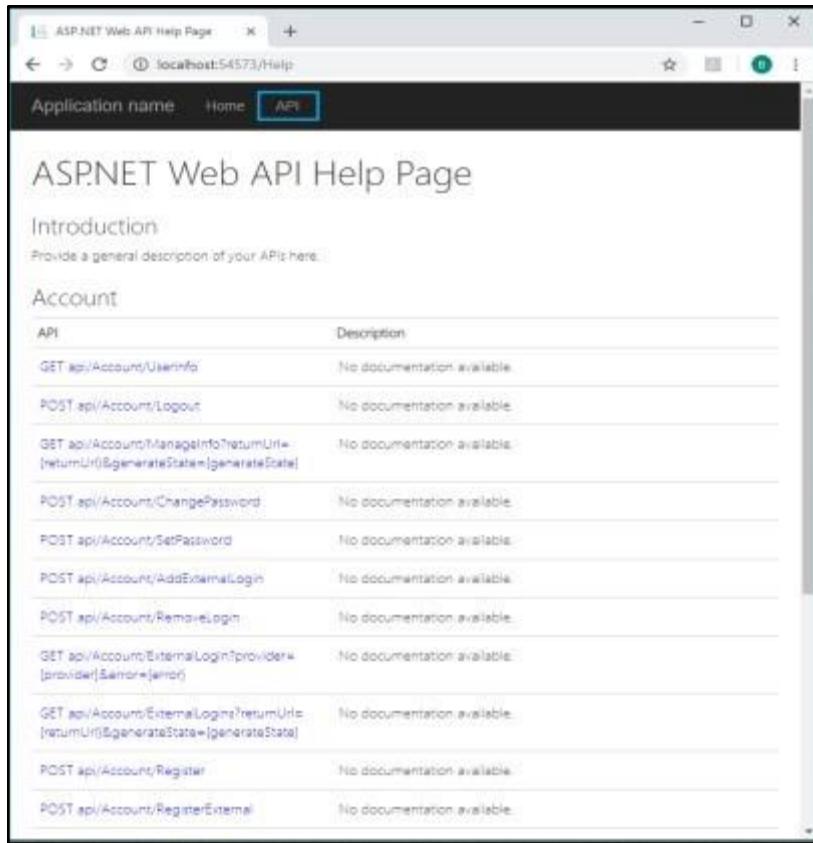


Figure 7.23: Account API Method

Both the Register method and Logout method can be seen, but not the Login method. The Login method works in OWIN authentication as follows:

Step 6: Stop the application and open the Startup.Auth.cs file inside the App_Start folder. Here, the code shown in Code Snippet 12 can be seen. A setting for a token generation - TokenEndpointPath can also be found here.

Code Snippet 12:

```
OAuthOptions=new OAuthAuthorizationServerOptions{
    TokenEndpointPath = new PathString("/Token"),
    Provider=new ApplicationOAuthProvider(public ClientId),
    AuthorizeEndpointPath = new
    PathString("/api/Account/ExternalLogin"),
    AccessTokenExpires=TimeSpan.FromDays(14),
    //In production mode set AllowInsecureHttp=false
    AllowInsecureHttp = true
};
```

By default, this path or URL is not explorable, so it is not visible to the API helper class. However, this path can be called. In this case, the path is <http://localhost:54573/Token>

Use Postman to invoke this API method to generate the Token from OWIN.

To implement authentication and authorization, it is important to first register the user with a unique user id and password and then, attempt to log in with the same.

Step 7: Execute the application again. Open the Postman application or browser extension to register the user. Select POST from the drop-down. Figure 7.24 shows the output displaying the URL to post the user registration information. Pass the JSON user object shown in Code Snippet 13 to the POST method.

Code Snippet 13:

```
{  
  "Email": "alex.g@gmail.com", "Password": "myPass@123",  
  "ConfirmPassword": "myPass@123"  
}
```

Figure 7.24 shows the POST method running in the Postman application. Observe all the highlighted sections. The response code is 200, which means the API is executed successfully and the user is created.

The screenshot shows the Postman interface for a POST request to `https://localhost:44338/api/Authenticate/register`. The request body is set to `raw` and `JSON`, containing the following JSON payload:

```
1 {  
2   "Email": "alex.g@gmail.com",  
3   "Password": "myPass@123",  
4   "ConfirmPassword": "myPass@123"  
5 }
```

The response tab shows a successful 200 OK status with the message "User created successfully".

Figure 7.24: Account Registration Testing Using Postman

Step 8: Log in to the API by calling the /Token method. To do this, the developer should provide the full path as <http://localhost:54573/Token>.

The parameter and HTTP method types are shown in Figure 7.25.

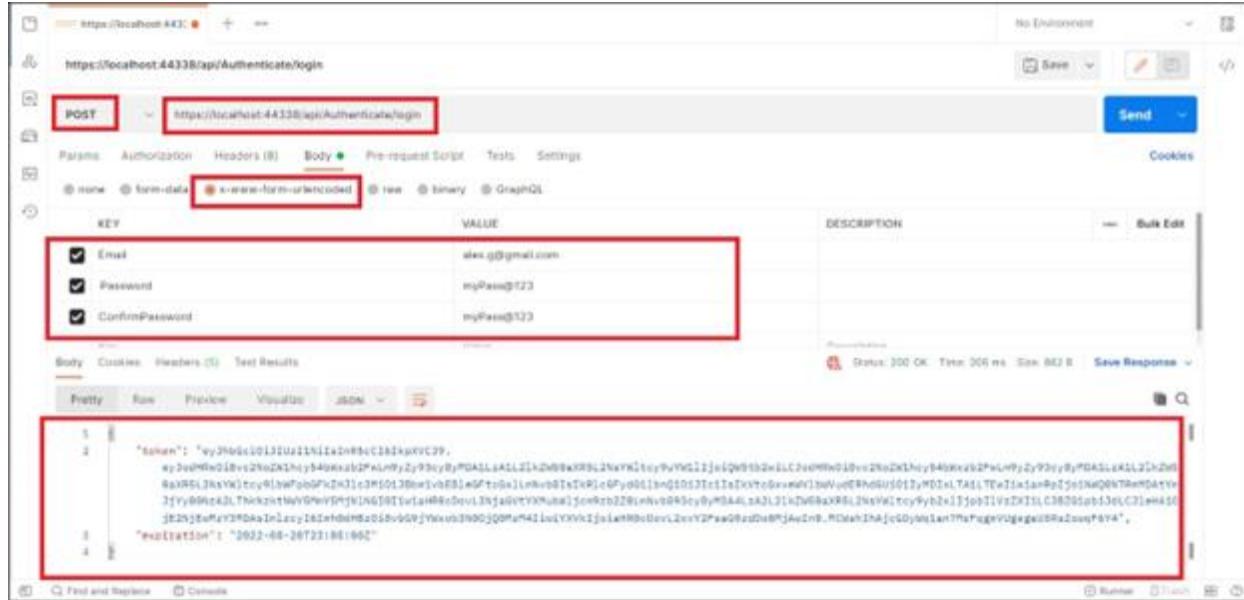


Figure 7.25: Account Login and Token Generation Testing Using Postman

It can be observed that the response represents the encoded Access Token value, Expiry date, Token Type, and so on. This Access Token is important as it requires to be sent as a header object for a further authorized API call.

The login information is stored in auto-generated tables in the Azure database. To see the tables, log in to Azure SQL Server using SQL Server Management Studio (SSMS). When the developer expands the database tables, he/she can see that there are six tables created.

Out of the six, five tables begin with the prefix AspNet. These tables are related to authentication. Figure 7.26 represents the table structure.

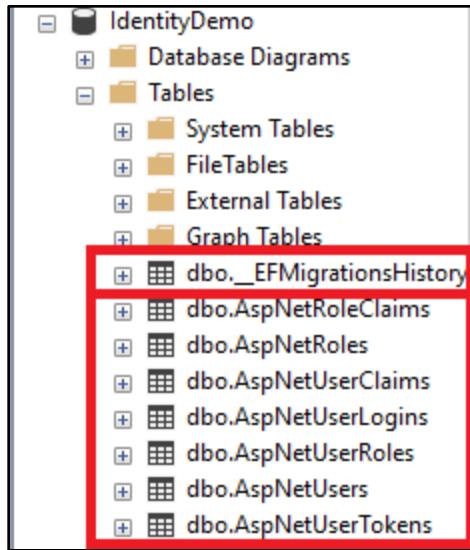


Figure 7.26: Auto-generated Tables in Azure SQL Server Database

7.7 Overview of CORS and XSRF

Cross-Origin Resource Sharing (CORS) is a World Wide Web Consortium (W3C) standard. It is used to make a server ease the same-origin policy. The same-origin policy is a restriction on Web pages that curb them from sending AJAX requests to another domain due to browser security issues. This restricts a malicious site from accessing confidential information from another site. Despite this, a requirement may arise wherein other sites should call the developed Web API. This is where CORS comes in handy. CORS allows a server to accept a few cross-origin requests while declining others.

Earlier, JavaScript Object Notation with Padding (JSONP) was used for the same purpose, but now CORS is considered more reliable and flexible. Let us now understand how to enable CORS in the WebService app.

Steps to enable it are as follows:

Step 1: Add the CORS NuGet package. To do this, open Visual Studio 2022, go to the Tools menu, and choose Library Package Manager Package Manager Console. Enter the command `Install-Package Microsoft.AspNet.WebApi.Cors`

Once the command is executed, the latest version of the package is installed. It also updates all the associated entities, for example, the core Web API libraries. In case a particular version is required, use the `-Version` flag. However, note that Web API 2.0 or later is essential for the CORS package.

Step 2: In the file App_Start/WebApiConfig.cs, insert the code given in Code Snippet 14 to the WebApiConfig.Register method.

Code Snippet 14:

```
using System.Web.Http;
namespace WebService{
public static class WebApiConfig {
public static void Register(HttpConfiguration config)
{
// This line of code will enable CORS
config.EnableCors();
config.Routes.MapHttpRoute(
name: "DefaultApi",
routeTemplate: "api/{controller}/{id}", defaults: new { id =
RouteParameter.Optional }
);
}
}
```

Step 3: Add the [EnableCors] attribute to the StudentController class as shown in Code Snippet 15.

Code Snippet 15:

```
using StudentServices.Models;
using System.Linq;
using System.Web.Http;
namespace StudentServices.Controllers {
[EnableCors(origins:"*",headers:"*",methods:"*")]
[Authorize]
public class StudentController:ApiController{
//Code snippet for Student APIs
}
}
```

For the origin parameter, use the URI where the WebClient application was deployed. This permits cross-origin requests arriving from the WebClient and prohibits other requests that are of cross-domain type.

XSRF is popularly known as **Cross-Site Request Forgery (CSRF)**. It can be defined as an attack on Web-hosted applications in which a malicious Web application takes control of the communication between a browser and a Web application. The chances of such attacks are very likely because, in their communication with a Website, Web browsers automatically forward some sort of authentication tokens. Such attacks are also called a one-click attack or session riding as they

exploit the user's formerly authenticated session.

XSRF has a high chance when Web applications employ cookies to authenticate the user. This is because of following reasons:

- A Web application utilizes cookies, which are in turn saved by the browsers.
- A saved cookie has session cookies for users who are authenticated successfully.
- Cookies stored by the browsers are sent to the Web application with each request.

This is done irrespective of the request generation process.

However, it is important to note that XSRF is not just confined to exploiting cookies. Even the basic authentication process can be prone to such attacks. This is because, once the user logs in using basic authentication, the browser continues to forward the credentials till that particular session persists.

7.8 Implementing Asynchronous and Synchronous Actions

Synchronous/asynchronous API return data for requests either immediately or at a later time, respectively. They allow us to make immediate or scheduled requests for resources, data, or services when available.

The synchronous and asynchronous nature of an API depends on the timing of the request to the return of data. In synchronous APIs, there is an immediate return of data. The application sends requests for data and waits until a value is returned.

In asynchronous APIs, the availability of a resource, service, or data store may not be immediate. This kind of API can provide a callback to the requester when the requested resource is ready. They provide better functionality.

API can be synchronous where data or service availability, resources, and connectivity are high and low latency is a requirement or it can be asynchronous where data, service availability, and connectivity are low with demand.

7.9 Creating OData Services

Open Data Protocol (OData) is a type of data access protocol for the Web. It provides a consistent method for querying and controlling data sets via CRUD operations. These operations include delete, read, update, and create.

OData offers a standard method for querying information and is built on REST. The standard queries are operations that limit the number of returned objects, perform paging, count returned elements, select objects based on certain conditions,

and so on. ASP.NET Web API supports both versions of the protocol, that is, v3 and v4. It also supports a v4 endpoint that could be executed beside a v3 endpoint.

7.9.1 Creating OData Endpoint Using ASP.NET Web API

Following are the steps to create an OData service based on the existing student entity data model:

Step 1: Create a new ASP.NET Web Application or add a new project to the previous solution. Assign the name `StudentOData`. Choose an Empty template and select the Web API option as shown in Figure 7.27.

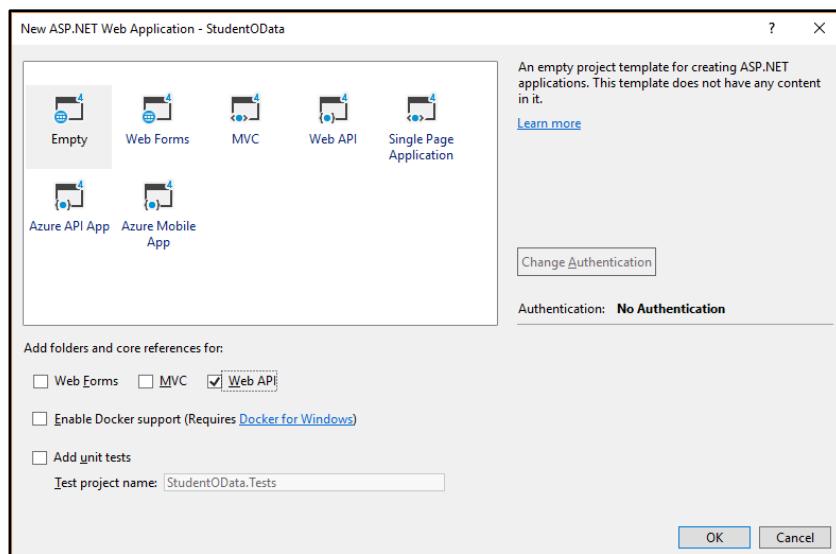


Figure 7.27: Web API for OData Services

Step 2: Install OData and Entity Framework from NuGet by executing the following NuGet command in NuGet Console:

```
Install-Package Microsoft.AspNet.Odata  
Install-Package EntityFramework  
Install-Package Microsoft.AspNet.WebApi.Tracing  
Update-Package Microsoft.AspNet.WebApiWebHost
```

Go to Tools menu and select NuGet Package Manager Console.

Step 3: Add a model class `StudentModel` in the Models folder as shown in Code Snippet 16.

Code Snippet 16:

```
public class StudentModel{  
    public int StudentId { get; set; }  
    public string StudentName { get; set; }  
    public string StudentEmail { get; set; }  
    public DateTime StudentJoinDate { get; set; }  
}
```

Step 4: Add a new Connection String parameter in `web.config` within as given in Code Snippet 17.

Code Snippet 17:

```
<connectionStrings>  
    <add name="StudentODataContext" connectionString="Data  
    Source=tcp:azureclouddb.database.windows.net,1433; initial  
    catalog=StudentDb; user id=billsmith; password=Dracula99;  
    MultipleActiveResultSets=True;"  
    providerName="System.Data.SqlClient" />  
</connectionStrings>
```

Step 5: Configure the OData route. Open the `WebApiConfig.cs` file from the `App_Start` folder for registering the route of the OData service and add the code, as shown in Code Snippet 18.

Code Snippet 18:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web.Http;  
using System.Web.Http.OData.Builder;  
using System.Web.Http.OData.Extensions;  
using StudentOData.Models;  
namespace StudentOData{  
    public static class WebApiConfig{  
        public static void Register(HttpConfiguration config){  
            ODataConventionModelBuilder builder = new  
            ODataConventionModelBuilder();  
            builder.EntitySet<StudentModel>("StudentModels");  
            config.Routes.MapODataServiceRoute("odata", "odata",  
            builder.GetEdmModel());  
        }  
    }  
}
```

Step 6: To add a controller, right-click the Controllers folder and select Add Scaffolded item. Name the class StudentModelsController. Refer to Figure 7.28.

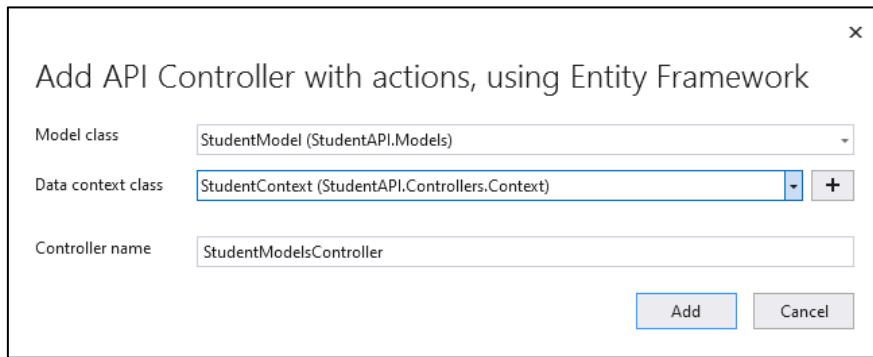


Figure 7.28: Adding API Controller with Actions

Implement the OData functionality using StudentDbContext as given in Code Snippet 19.

Code Snippet 19:

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Data.Entity.Infrastructure;
using System.Linq;
using System.Net.Http;
using System.Web.Http;
using System.Web.Http.ModelBinding;
using System.Web.Http.OData;
using System.Web.Http.OData.Routing;
using StudentOData.Models;
namespace StudentOData.Controllers {

    /*The WebApiConfig class may require additional changes to add a
    route for this controller. Merge these statements into the
    Register method of the WebApiConfig class as applicable. Note
    that OData URLs are case sensitive.*/

    public class StudentModelsController : ODataController {
        private StudentODataContext db = new StudentODataContext();
        // GET: odata/StudentModels [EnableQuery]
        public IQueryable<StudentModel> GetStudentModels() {
            return db.StudentModels;
        }
    }
}
```

```

// POST: odata/StudentModels
public IHttpActionResult Post(StudentModel studentModel) {
    if (!ModelState.IsValid) {
        return BadRequest(ModelState);
    }
    db.StudentModels.Add(studentModel);
    db.SaveChanges();
    return Created(studentModel);
}
protected override void Dispose(bool disposing) {
    if (disposing) {
        db.Dispose();
    }
    base.Dispose(disposing);
}

```

Step 7: Execute the application. This displays the OData properties of Students as shown in Figure 7.29.

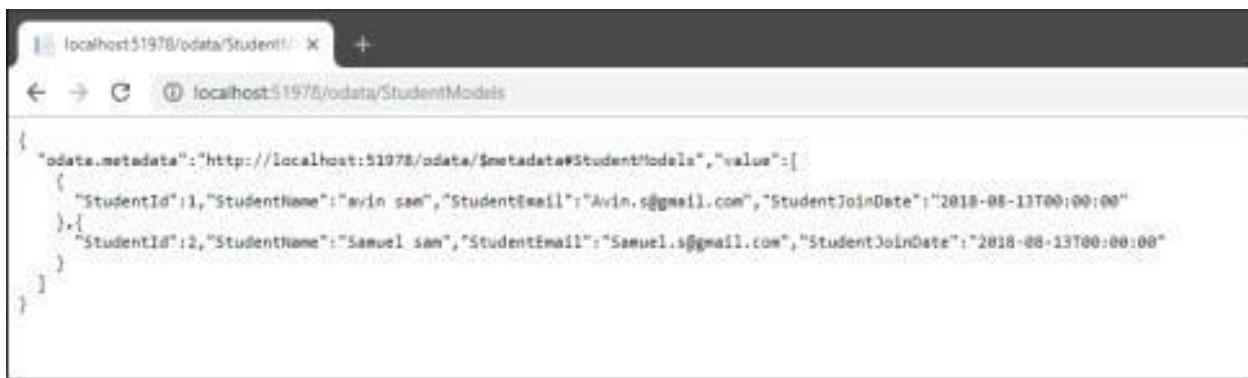


Figure 7.29: OData Properties of Students

Step 8: Use Postman to verify the OData Services. The URL and data are given in Figure 7.30.

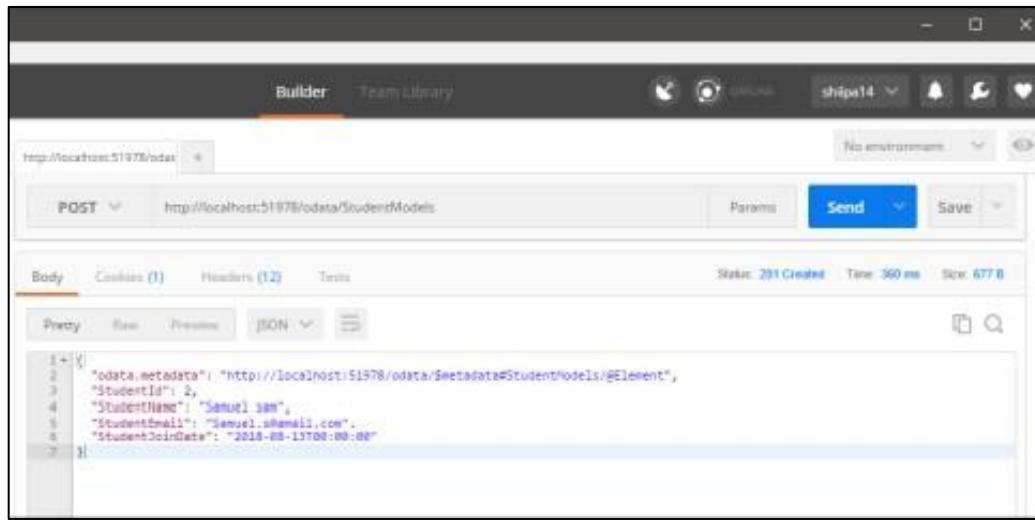


Figure 7.30: URL and Data of Students

7.10 Summary

- ✓ ASP .NET Web API supports HTTP requests, such as GET, POST, PUT, and DELETE.
- ✓ The attributes of the Model View Controller (MVC), which include Routing, Controllers, Filters, Action results, Model binders, Inversion of Container (IOC), and Dependency injection are part of ASP.NET Web API.
- ✓ ASP.NET Web API supports OData functionality to implement easy CRUD functionality in databases.
- ✓ Routing in ASP .NET Web API works based on the routing table or action-based or attribute-based routing.
- ✓ ASP.NET Web API supports both synchronous and asynchronous methods of invocation.

7.11 Test Your Knowledge

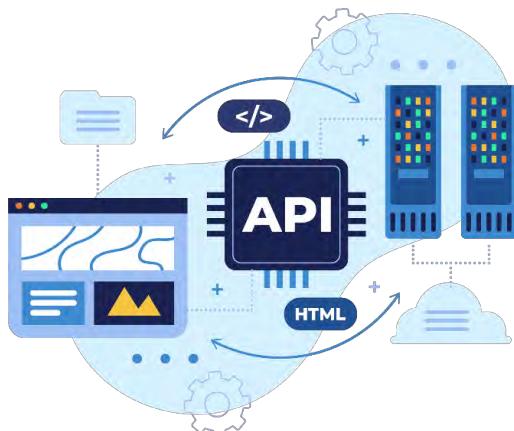
1. Which of the following is a way to define routes in Web API?
 - A. Using HTML
 - B. Using SQL
 - C. Using the [Route] attribute
 - D. Using JavaScript
2. What is Entity Framework commonly used for in the context of Web API?
 - A. Front-end Development
 - B. Database Operations
 - C. Server-side Caching
 - D. CSS Styling
3. What keyword is commonly used in C# to enable asynchronous programming in Web API?
 - A. sync
 - B. wait
 - C. async
 - D. pause
4. What is OData primarily used for in Web API?
 - A. Styling Web pages
 - B. Consuming RESTful APIs
 - C. Creating database tables
 - D. Generating HTML forms

7.11.1 Answers to Test Your Knowledge

1. C
2. B
3. C
4. B

Try It Yourself

- 1) Retrieve book details from the API: <https://simple-books-api.glitch.me> using the GET function.
- 2) Make an ASP.NET Web API with Authentication as 'Individual Account' or 'Microsoft Identity Platform'.
- 3) Create a Web API for HTTP method 'DELETE'.
- 4) Publish the Web API to Azure App Service.



SESSION 08

APIS AND RESTFUL APIS FOR ENTERPRISE APPLICATIONS

Overview

APIs play a crucial role in enabling communication and integration between different software systems, and they are especially essential in enterprise applications where various services and components work together seamlessly.

When it comes to designing APIs for enterprise applications, RESTful APIs are a popular choice due to their simplicity, scalability, and compatibility with Web standards. This introduces these APIs.

Learning Objectives

In this session, students will learn to:

- Explain OAuth 2.0 or JWT authentication within an API framework
- Illustrate how to set up Role-Based Access Control (RBAC)
- Explain different versioning strategies
- Elaborate on HATEOAS

8.1 Overview of OAuth 2.0 or JWT Authentication

OAuth 2.0 and JSON Web Token (JWT) are both widely used in the context of API authentication and authorization. They serve different purposes, and in some cases, they are even used together. Let us discuss each of them and how they contribute to ensuring secure access to resources within an API framework. Figure 8.1 depicts API authentication.

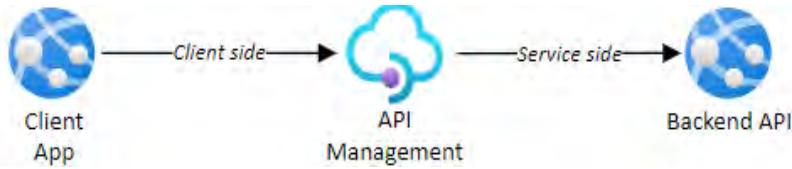


Figure 8.1: API Authentication

8.1.1 OAuth 2.0

Open Authorization 2.0 (OAuth 2.0) is an open standard framework for access delegation, commonly used for securing Web applications and APIs. It provides a way for users to grant third-party applications limited access to their resources without sharing their credentials. OAuth 2.0 is widely adopted for authorization and authentication in various scenarios, including social login, mobile app access, and API security.

Following are key aspects of OAuth 2.0:

Components:

- Resource Owner (User)

The entity that can grant access to a protected resource. It is typically the end-user.

- Client

The application that is requesting access to the user's resources. This could be a Web or mobile application.

- Authorization Server

The server is responsible for authenticating the user and obtaining their consent to grant access to the client. It issues access tokens.

- Resource Server

The server hosting the protected resources that the client wants to access. It validates access tokens and serves the requested resources.

OAuth 2.0 Grant Types:

- Authorization Code Grant

Used for Web applications where the client can securely store a client secret.

Involves a two-step process. These are namely, obtaining an authorization code and exchanging it for an access token.

- Implicit Grant

Designed for mobile or single-page applications where keeping a client secret is not feasible. The access token is obtained directly without exchanging an authorization code.

- Resource Owner Password Credentials Grant

Allows the client to directly obtain the user's credentials and exchange them for an access token. Generally discouraged due to security concerns unless the client is highly trusted.

- Client Credentials Grant

Used when the client is acting on its own behalf and not on behalf of a user. Directly exchanges client credentials for an access token.

OAuth 2.0 Workflow:

- Authorization Request

The client requests authorization from the user by re-directing them to the authorization server.

- User Authorization

The user authenticates and grants permission to the client.

- Token Request

The client exchanges the authorization grant for an access token.

- Accessing Protected Resource

The client uses the access token to access the protected resource on the resource server.

OAuth 2.0 Tokens:

Access Token:

Represents the authorization granted to the client.
Provides access to specific resources for a specific duration.

Refresh Token:

Used to obtain a new access token without requiring the user to re-authenticate.
Generally, has a longer lifespan than access tokens.

Security:

HTTPS Usage:

All communication should be done over HTTPS to secure data transmission.

Token Validation:

Resource servers should validate tokens before processing requests.

Token Request:

The client exchanges the authorization grant for an access token.

Token Expiry and Refresh:

Access tokens should have a limited lifespan, and refresh tokens should be used to obtain new access tokens.

8.1.2 JWT Authentication

JSON Web Token (JWT) authentication is a method of securing APIs by using JSON-encoded tokens to represent the authentication and authorization of a user. JWT is widely used due to its simplicity, compactness, and the ability to carry information in a self-contained manner. It is commonly employed in stateless authentication scenarios where the server does not store session data.

Key Components of JWT are as follows:

- Header

The header typically consists of two parts: the type of the token, which is JWT, and the signing algorithm being used, such as HMAC SHA256 or RSA.

Example Header:

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

- Payload

The payload contains the claims. Claims are statements about an entity (typically, the user) and additional data.

Example Payload:

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "admin": true  
}
```

Common Claims:

sub (subject): Identifies the principal subject of the token.

iss (issuer): Identifies the issuer of the token.

exp (expiration time): Specifies the expiration time after which the token must not be accepted.

aud (audience): Identifies the recipients for which the token is intended.

iat (issued at): Specifies the time at which the token was issued.

Example Signature:

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  secret  
)
```

JWT Authentication Workflow:

- User Authentication

When a user logs in, the server generates a JWT containing relevant user information.

- Token Issuance

The server signs the JWT with a secret key or a private key (in the case of RSA), creating the final JWT.

- Token Response

The JWT is sent to the client as part of the authentication response.

- Server Verification

The server verifies the integrity of the JWT using the stored secret key or public key.

- User Authorization

The server extracts and decodes the claims in the payload to determine the user's identity and authorization.

Security:

Token Expiry:

Implement short-lived tokens, and regularly refresh them using refresh tokens or by re-authenticating.

Secure Transmission:

Always use HTTPS to ensure secure transmission of JWTs between the client and the server.

Keep Secrets Secure:

Protect the secret key used for signing JWTs. If using public-key cryptography, secure the private key.

Token Scope and Validation:

Validate the claims in the JWT, such as expiration time and issuer, to ensure the token's integrity.

8.2 Setting up Role-Based Access Control (RBAC)

Setting up Role-Based Access Control (RBAC) involves defining roles, assigning permissions to those roles, associating roles with users, and implementing access controls based on these roles.

Here is a step-by-step guide to illustrate how to set up RBAC:

Step 1: Identify Roles and Permissions

Identify different roles that exist within your organization or system. Roles can be based on job functions, responsibilities, or any other criteria relevant to your organization.

Step 2: Define Permissions

For each role, determine the specific permissions or actions that users assigned to that role should be allowed to perform. These permissions should be granular and well-defined to ensure that users only have access to the resources necessary for their role.

Step 3: Assign Permissions to Roles

Once permissions are defined, assign them to the appropriate roles. This step essentially creates a mapping between roles and the permissions associated with each role. Refer to Figure 8.2.

The screenshot shows the Azure portal interface for managing role assignments. On the left, there's a navigation bar with 'Add', 'Download role assignments', 'Edit columns', 'Refresh', and a 'Ref' button. Below the navigation are tabs: 'Check access' (which is selected), 'Role assignments', 'Roles', 'Deny assignments', and 'Classic admin'. Under 'Check access', there's a section titled 'My access' with a 'View my access' button. A search bar labeled 'Find' is followed by a dropdown menu set to 'User, group, or service principal' with the search term 'tom'. Below it is a list item: 'Tom Doe tomdoe@adammarczak620.onmicrosoft.com'. On the right, a 'Reader' role is selected. The 'Assign access to' dropdown is set to 'User, group, or service principal'. A 'Select' button is present with a search bar 'Search by name or email address'. Below it is a list of users: Adam Marczak (with a blue profile icon), Alice Jones (with a black profile icon), and Azure ATP adammarczak620 Administrators (with a blue profile icon). At the bottom, there's a note about selected members: 'Selected members: No members selected. Search for and add one or more members you want to assign to the role for this resource.' and a link 'Learn more about RBAC'.

Figure 8.2: Assigning Permission to The Roles

Example mapping:

Role	Permissions
<hr/>	
Administrator	CRUD on all resources
Manager	Read and Update on employee data
Employee	Read on company policies
Customer	Create and Update own profile

Step 4: Assign Users to Roles

Assign users to the roles that best correspond to their job responsibilities or functions within the organization. Each user may be assigned to one or more roles, depending on their access requirement. Refer to Figure 8.3.

Figure 8.3: Implementation of RBAC- Role based Access Control

Example Assignments:

User	Assigned Roles

John Doe	Administrator
Jane Smith	Manager, Employee
Alice Brown	Employee, Customer

Step 5: Implement RBAC in the System

Implement RBAC within your system or application using the roles, permissions, and user-role assignments defined earlier. This typically involves configuring access controls, such as user authentication and authorization mechanisms, based on the RBAC model.

Step 6: Regularly Review and Update

RBAC is not a one-time setup. It requires ongoing maintenance to ensure that roles, permissions, and user-role assignments remain up-to-date and aligned with the organization's requirements. Regularly review and update RBAC configurations as necessary.

Step 7: Audit and Monitoring

Implement auditing and monitoring mechanisms to track user access and activities within the system. This helps detect any unauthorized access attempts or suspicious activities and ensures compliance with security policies and regulations.

Step 8: Enforce Least Privilege Principle

Follow the principle of least privilege, which means granting users only the minimum level of access required to perform their job functions. This helps minimize the potential impact of security breaches or insider threats.

Step 9: Provide Administrative Tools

Provide administrative tools or interfaces for managing roles, permissions, and user-role assignments. This allows administrators to easily add, modify, or revoke access rights as required.

The way you control access to resources using Azure RBAC is to assign Azure roles. This is a key concept to understand – it is how permissions are enforced. A role assignment consists of three elements: security principal, role definition, and scope.

- Security Principal

A security principal is an object that represents a user, group, service principal, or managed identity that is requesting access to Azure resources. You can assign a role to any of these security principals.

- Role Definition

A role definition is a collection of permissions. It is typically just called a role. A role definition lists the actions that can be performed, such as read, write, and delete. Roles can be high-level, such as owner, or specific, such as virtual machine reader.

- Scope

Scope is the set of resources that the access applies to. When you assign a role, you can further limit the actions allowed by defining a scope. This is helpful if you want to make someone a Website Contributor, but only for one resource group.

- Role Assignments

A role assignment is the process of attaching a role definition to a user, group, service principal, or managed identity at a particular scope for the purpose of granting access. Access is granted by creating a role assignment, and access is revoked by removing a role assignment.

8.3 Versioning Strategies

Versioning strategies are crucial in API design and maintenance, as they determine how changes and updates are managed while ensuring backward compatibility and a positive user experience.

Let us examine versioning strategies along with their implications on API maintenance and user experience.

8.3.1 URI Versioning

Description: In this strategy, the version is included as part of the URI path.

Example: <https://api.example.com/v1/resource>

Implications:

Pros: Clear and explicit versioning, easy to implement, allows for multiple versions to co-exist.

Cons: Can clutter the URI, leading to longer and less readable URLs. Requires clients to update URLs when switching versions.

8.3.2 Query Parameter Versioning

Description: The version is specified as a query parameter in the URL.

Example: <https://api.example.com/resource?version=v1>

Implications:

Pros: Keeps the base URI clean, and allows for easy version switching without modifying the base URL.

Cons: May not be as explicit as URI versioning. Caching mechanisms might not differentiate between different versions.

8.3.3 Header Versioning

Description: The version information is included in the request header.

Example: GET /resource HTTP/1.1\nHost: api.example.com\nAccept: application/json\nX-API-Version: v1

Implications:

Pros: Keeps the URI clean, provides a centralized way to manage version information.

Cons: Requires clients to explicitly specify the version in the request header, which can be less intuitive.

8.3.4 Media Type Versioning

Description: Different versions of the API are represented by different media types.

Example: Accept: application/vnd.example. v1+json

Implications:

Pros: Allows clients to request specific API versions using content negotiation, and keeps URLs clean.

Cons: Requires clients to understand and correctly specify media types, and may be less intuitive for developers.

8.3.5 Semantic Versioning

Description: Versions are represented using a three-part format- MAJOR.MINOR.PATCH.

Example: 1.0.0

Implications:

Pros: Provides clear guidelines for version incrementation based on the nature of changes (major, minor, and patch).

Cons: Does not inherently specify how version information should be conveyed in requests, and should be combined with other versioning strategies.

8.3.6 Date Versioning

Description: Versions are represented by the date of release.

Example: 2024-01-01

Implications:

Pros: Provides a chronological way to track API changes, and is easy to understand.

Cons: May not indicate the nature of changes in a version (example, breaking changes vs. minor updates).

Implications on API Maintenance and User Experience:

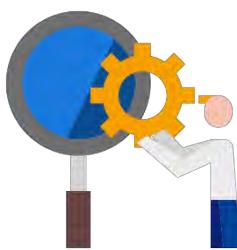
Maintenance:	User Experience:
<p>Different versioning strategies have different implications for API maintenance. URI versioning may require more effort to manage as APIs evolve. Whereas, header or media type versioning can provide more flexibility in introducing changes without affecting the URI structure.</p>	<p>A clear versioning strategy contributes to a positive user experience. Strategies such as URI or query parameter versioning make versioning explicit but may result in longer URLs. On the other hand, strategies such as header or media type versioning keep the URLs clean but may require users to understand and correctly use request headers or media types. Semantic versioning helps users understand the nature of changes but requires them to be combined with other versioning strategies for implementation. Ultimately, the chosen strategy should balance clarity, flexibility, and ease of use for API consumers.</p>

8.4 HATEOAS and its Role in Creating APIs

Hypermedia as the Engine of Application State (HATEOAS) is a constraint in the REST architectural style that emphasizes the use of hypermedia links to navigate the API dynamically. In simpler terms, HATEOAS allows clients to interact with an API by following links provided by the server, rather than having prior knowledge of API endpoints.

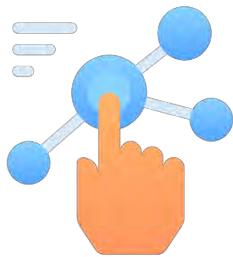
8.4.1 Working and Role of HATEOAS in creating Self-Discoverable and Dynamic APIs

Self-Discoverability:



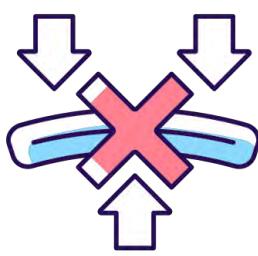
With HATEOAS, APIs are designed to be self-descriptive. Each response from the server includes hypermedia links that guide clients on what actions they can take next. Clients do not require to rely on out-of-band information (such as API documentation) to understand how to interact with the API. Instead, they can dynamically discover available resources and actions by following links embedded in API responses.

Dynamic Interaction:



HATEOAS enables dynamic interaction with the API. Clients can navigate through different resources and perform actions based on the links provided in the API responses. It allows for more flexible and adaptable client-server interactions. This is because clients can respond to changes in the API's structure or behavior without requiring modifications to their code.

Reduced Coupling:



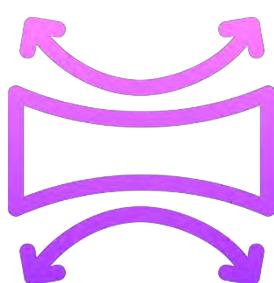
HATEOAS promotes loose coupling between clients and servers. Clients interact with the API based on the links provided by the server, rather than hard-coding specific endpoints or assuming a fixed API structure. This makes the API more resilient to changes. It also allows for easier evolution and versioning. This is because the server can modify resource URLs or introduce new functionality without breaking existing clients.

Improved API Usability:



By providing hypermedia links in API responses, HATEOAS enhances the usability of APIs. Clients can navigate through different resources and perform actions in a more intuitive and self-explanatory manner. This can simplify client development and integration efforts, as well as improve the overall user experience when interacting with the API.

Enhanced Flexibility:



HATEOAS allows APIs to evolve over time without disrupting client implementations. Servers can add, remove, or modify resources and actions, and clients can adapt to these changes dynamically by following the links provided in API responses. This flexibility enables agile development practices and supports the evolution of complex distributed systems.

8.5 Summary

- ✓ OAuth 2.0 or JWT authentication within an API framework enables secure access control and user authentication.
- ✓ RBAC defines user permissions based on roles, managing access within an organization's system or application.
- ✓ Different versioning strategies impact API maintenance and user experience by shaping update management and accessibility over time.
- ✓ HATEOAS enables dynamic APIs by embedding hypermedia links, allowing clients to navigate resources without prior knowledge.

8.6 Test Your Knowledge

1. What is the primary purpose of OAuth 2.0 in API security?
 - A. To encrypt API requests and responses
 - B. To provide a framework for access delegation without sharing credentials
 - C. To generate and manage API keys
 - D. To monitor API usage and rate limit requests
2. Which of the following is NOT a common component of JSON Web Token (JWT)?
 - A. Header
 - B. Payload
 - C. Signature
 - D. Footer
3. In Role-Based Access Control (RBAC), what does the term 'role' refer to?
 - A. A specific permission, such as read or write access
 - B. A unique identifier for each user
 - C. A collection of permissions assigned based on job functions or responsibilities
 - D. The process of logging user activities
4. What is the role of HATEOAS in RESTful APIs?
 - A. It encrypts API communications
 - B. It provides a directory of API services
 - C. It enables dynamic discovery of API resources through hypermedia links
 - D. It ensures high performance of APIs
5. Which of the following statements best describes the role of HATEOAS in RESTful APIs?
 - A. HATEOAS restricts the API to a fixed set of operations that cannot be changed or extended.
 - B. HATEOAS requires clients to have prior knowledge of all API endpoints before they can interact with the API.
 - C. HATEOAS allows clients to dynamically discover and interact with resources through hypermedia links provided by the API.
 - D. HATEOAS mandates the use of JSON Web Tokens (JWT) for authentication and authorization in RESTful APIs.

8.6.1 Answers to Test Your Knowledge

1. B
2. D
3. C
4. C
5. C

Try It Yourself

- Identify key considerations when designing and implementing RESTful APIs for enterprise applications, highlighting their importance in modern software development practices. Present this through a PowerPoint presentation.
- Implement Role-Based Access Control (RBAC) for an enterprise application scenario, defining roles, permissions, and user-role assignments, and illustrate how it enhances security and access management within the system.
- Perform a task of URI versioning in a REST API including the version number in the URI path.



SESSION 09

AZURE SECURITY FEATURES

Overview

This session provides an introduction to security features of Azure. It discusses how to implement secure networks with Azure. It discusses malware protection and the features Azure delivers. It discusses Identity and Access Management (IAM). It explains how to configure MFA for Microsoft Entra ID users and administrators. It discusses the use of Azure Firewall and NSGs to control network traffic and protect resources. It also explains configuration and monitoring of Distributed Denial of Service (DDoS) protection for developer resources.

Learning Objectives

In this session, students will learn to:

- Explain network security fundamentals and components
- Define Key Logs and Key Vault in Azure
- Outline the importance of malware protection and access management
- Illustrate Multi-Factor Authentication (MFA) and its significance
- Explain Azure Firewall and Network Security Groups (NSGs)
- Define DDoS attacks and Azure's protection services
- Define and configure secure virtual networks in Azure
- Outline the purpose and functionality of Defender for Identity

9.1 Introduction

Azure Security refers to the security tools and features available on the Microsoft Azure Cloud Platform. According to Microsoft, the means to ensure cloud services include various physical infrastructure and operating controls in the Azure platform itself as well as through third-party solutions that developers can deploy into their Azure subscriptions.

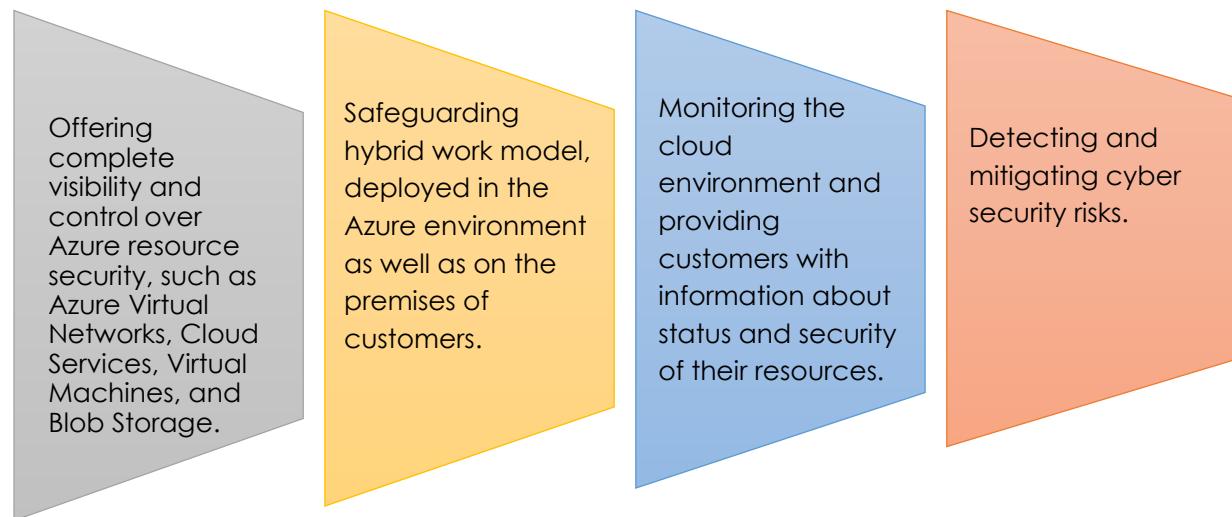
Built-in security features available in the Azure platform are categorized into these functional areas: Operations, Applications, Storage, Networking, Compute, and Identity.

Operations	Microsoft Sentinel, Microsoft Defender for Cloud, Azure Resource Manager, Application Insights, Azure Monitor, and Azure Advisor are some of the operation-based tools and solutions.
Applications	Web Application firewall, Authentication and authorization in Azure App Service, Layered Security Architecture, Web server diagnostics and application diagnostics are some of the application-based security tools and features.
Storage	Azure Role-Based Access Control (Azure RBAC), Shared Access Signature, Encryption in Transit, Encryption at rest, Storage Analytics, and enabling Browser-Based Clients Using CORS are some of the storage-based security tools and capabilities as part of the Azure platform.
Networking	Network Layer Controls, Azure Virtual Network, Azure Private Link, VPN Gateway, Express Route, and Application Gateway are some of the network-based security tools and capabilities as part of the Azure platform.
Compute	Antimalware and Antivirus, Hardware Security Module, Virtual machine backup, and Azure Site Recovery some of the tools and capabilities under Compute in the Azure platform.
Identity and Access Management	Secure Identity and Secure Apps and data are features under Identity.

9.1.1 Microsoft Defender for Cloud

Microsoft Defender for Cloud, formerly called Azure Security Center, is a unified security management system that can be availed by Azure customers.

Benefits of Microsoft Defender for Cloud are as follows:



Additionally, Microsoft Defender for Cloud helps in addressing following security concerns and issues:

Ever Changing Workloads

The Azure platform provides a bunch of services and allows customers to do customization in the cloud as per their requirements. The Azure Security Center makes it easier to consistently implement security standards and best practices that ensure protection against any implemented services.

Control Sophisticated Tasks

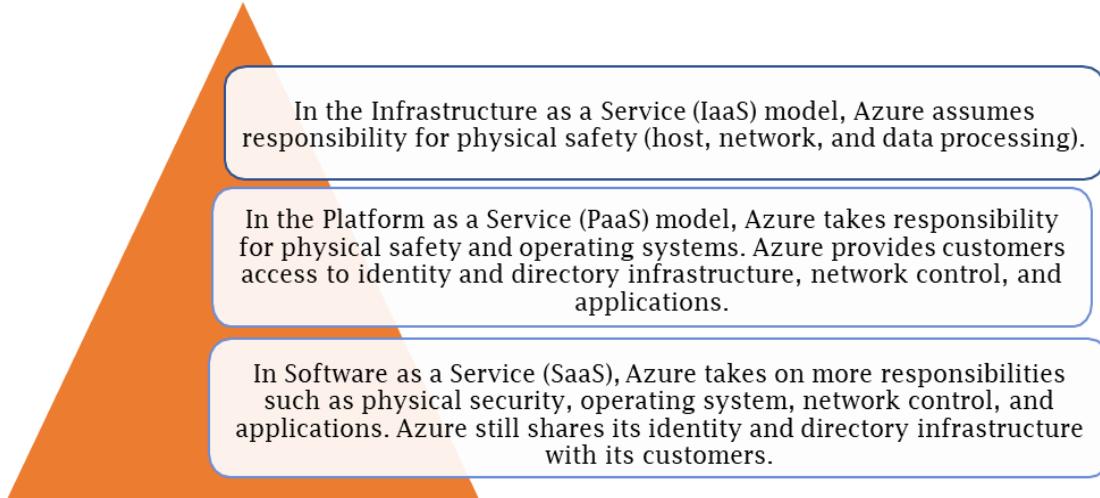
The public cloud is filled with diverse types of data workloads. Most people upload data to the public cloud so that they can access it at anytime, anywhere. Azure cloud customers can do this too and secure their workloads as well, but doing so may reveal additional threats if security best practices are not followed. Azure Security Center can assist with this task and ensure protection from such threats.

Security Skills Shortage

A large number of security concerns can overwhelm administrators, especially if they are inexperienced. However, the Azure Security Center can assist administrators in dealing with such attacks efficiently.

9.1.2 Azure Security Process

Azure Security is mostly a joint effort and shared responsibility between Azure and customers, except for scenarios where customers bear all responsibilities. Based on the cloud service model being used, ownership of who is responsible for managing the security of the application or service varies.



9.2 Secure Networks

Network security involves protecting resources from unauthorized access or attacks by applying controls to network traffic. Azure includes a robust network infrastructure to support application and service connectivity requirements. Network connectivity is possible between resources hosted on Azure, on-premises resources and resources hosted on Azure, and between the Internet and Azure.

Following are some options for network security:

Azure Networking	Azure requires a Virtual Memory to be connected to an Azure virtual network. A virtual network is a logical configuration built on top of a physical Azure network fabric. Each virtual network is secluded from all other virtual networks.
Network Access Control	Azure supports various sorts of network access control, such as: Network layer Control, Route control and Forced tunneling, and Virtual network security appliances.
Network Layer Control	Network access control limits virtual machine communication to the essential systems. Other communication attempts are blocked.
Network Security Rules (NSGR)	Network security groups include features such as Augmented Security Rules, Service tags, Application Security groups. They simplify management and reduce the potential for misconfiguration.
Service Endpoints	Service endpoints are another way to apply control over traffic. Communication with supported services over direct connections can be limited to virtual networks only. Traffic from the virtual network to a given Azure service remains on the Microsoft Azure backbone.

Route Control and Forced Tunneling

The ability to control routing behavior in a virtual network is very important. If routing is misconfigured, applications and services hosted on virtual machines can connect to unauthorized devices, including systems owned and controlled by potential attackers.

Azure Firewall

Azure Firewall is a managed, cloud-based network security service that guards one's Azure Virtual Network resources.

9.2.1 Some Best Practices for Azure Security

Following are some security best practices that can be used with Azure platform to meet project objectives:

- Constantly upgrade Azure subscription to Microsoft Defender for Cloud Standard to gain access to premium features such as detecting and fixing security bugs, detecting threats with analytics and intelligence, and responding quickly to an attack.
- Always keep keys in the Azure Key Vault. This vault is intended to store database credentials, passwords, and other sensitive data or information.
- Always prefer Azure Multi-factor Authentication (MFA), particularly for administrative accounts.
- Protect virtual hard disk files with encryption.
- Place Azure Virtual Machines VMs on Azure virtual networks to connect the machine to other networked devices.
- Install and configure a Web application firewall.
- To prevent and mitigate Distributed Denial Of Service (DDoS) attacks, use Azure's DDoS services.
- Examine the Microsoft Defender for Cloud dashboard on a regular basis. The dashboard offers a centralized view of a developer's Azure resources and suggests actions.

9.3 Key Logs

Azure Key Vault allows the developer to securely store keys, secrets, and certificates based on the vault and keys in their Azure subscription.

Items that developers can place in their vaults include:



9.4 Malware Protection

Microsoft Antimalware for Azure is a free, real-time protection solution that helps detect and remove viruses, spyware, and other malware. It generates alerts whenever either known malicious or unwanted software attempts to install or run on Azure systems. Microsoft Antimalware for Azure is a single-agent solution for several applications and client settings intended to operate in the background without human intervention. Developers can deploy protection according to the requirements of their application workload.

Following features are available for Microsoft Antimalware:

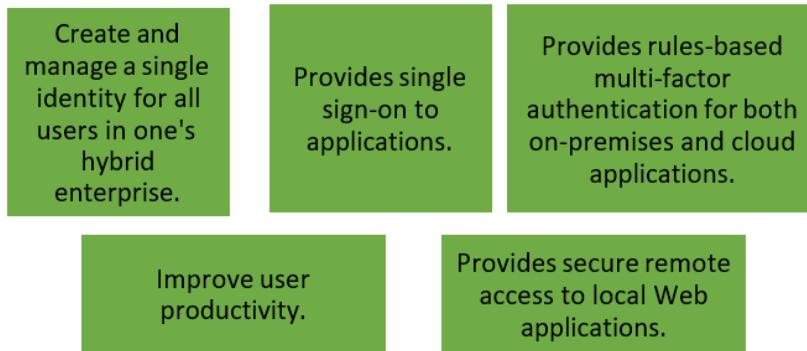
- **Real-time protection:** It monitors activity in Cloud Services and Virtual Machines to detect and block malware execution.
- **Scheduled scanning:** It occasionally scans to detect malware with actively running programs.
- **Malware remediation:** It automatically takes action on detected malware, such as deleting or quarantining malicious files or cleaning up malicious registry entries.
- **Signature updates:** It automatically installs the latest protection signatures (virus definitions) to ensure protection is up-to-date on a predetermined frequency.
- **Antimalware Platform updates:** It automatically updates the Microsoft Antimalware Platform.
- **Antimalware Engine updates:** It automatically updates the Microsoft Antimalware Engine.
- **Active protection:** It reports metadata about detected threats and suspicious resources to Microsoft Azure; the developer can quickly respond to the changing threat landscape and use Microsoft Active Protection (MAPS) to ensure real-time synchronous delivery of signatures through the system.
- **Samples reporting:** It provides and reports samples to the Microsoft Antimalware service to help refine the service and troubleshoot.
- **Antimalware event collection:** It records antimalware service status, suspicious actions, and remedial actions in the operating system's event log and collects them into the customer's Azure storage account.
- **Exclusions:** Applications and services administrators can configure exclusions for files, processes, and drives.

9.5 Access Management

Microsoft Azure identity and access management solutions can help IT professionals secure access to applications and resources in enterprise data centers and clouds. This includes additional levels of verification, such as multi-factor authentication and conditional access policies. Monitoring suspicious

activity with advanced security reporting, auditing, and alerting helps mitigate potential security issues.

With Azure IAM, developer can:



9.5.1 Microsoft Azure Identity

Identity serves as the foundation for a huge percentage of security assurances in a cloud-based architecture. However, due to their accessibility to a large volume of connection requests and risks across many customers, cloud-based identity solutions such as Microsoft Entra ID can provide additional security capabilities that legacy identity services cannot.

Here are some best practices that can be determined with Microsoft Azure identity and authentication:

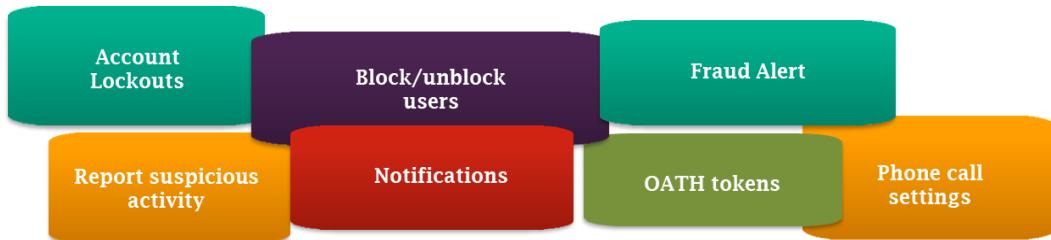
- **A centralized enterprise directory**
Create a single enterprise directory to manage in-house employee identities and enterprise operations.
- **Synchronized identity systems**
Connect cloud identity to the existing identity systems.
- **Cloud provider identity source for third parties**
Rather than incorporating third parties into an on-premises directory, it is always good to use cloud identity services to host non-employee transactions such as vendors, partners, and customers.
- **Multi-factor or password less authentication for administrative accounts**
Over time, all users must be converted to password less authentication or MFA.
- **Block legacy authentication**
For Internet-facing services, disable insecure legacy protocols.

- **In cloud identity providers, there are no on-premises admin accounts**
It is never a good practice to synchronize extremely privileged Domain Active Directory accounts to Microsoft Entra ID such as Domain, Enterprise, and Schema admins.
- **Modern password protection**
Provide modern, effective safeguards for accounts that cannot be password-less.
- **Credential management on multiple platforms**
For all platforms such as Windows, Linux, and cloud services, use a single identity provider. Microsoft Entra ID can authenticate several platforms such as:
 - Google Services
 - Azure
 - Office 365
 - Windows
 - Linux
 - Third-party SaaS providers and many more
- **Users with Zero Trust should be given Conditional Access**
To promote a Zero Trust strategy, all users' authentication must include the standard measure and regulation of key security attributes.
- **Attacks should be prevented**
One should train users regarding threats and vulnerabilities on a regular basis to educate and empower them. People are an important part of the defense strategy; therefore, one should make sure they have the skills and knowledge necessary to avoid and resist attacks. This procedure will lower overall organizational risk.

9.6 Configuration of MFA for Microsoft Entra ID Users and Administrators

Customizing the end-user experience in Microsoft Entra multi-factor authentication involves configuring settings such as account lockout thresholds, fraud alerts, and notifications.

Available settings for Microsoft Entra multi-factor authentication include:



9.6.1 Account Lockout

The account lockout settings control how many failed attempts are allowed before an account is locked out. This applies specifically when entering a PIN code in the MFA prompt using the on-premises MFA Server.

Perform following steps to set up account lockout settings:

1. Log in to the Microsoft Entra admin center with Authentication Policy Administrator rights. Refer to Figure 9.1.

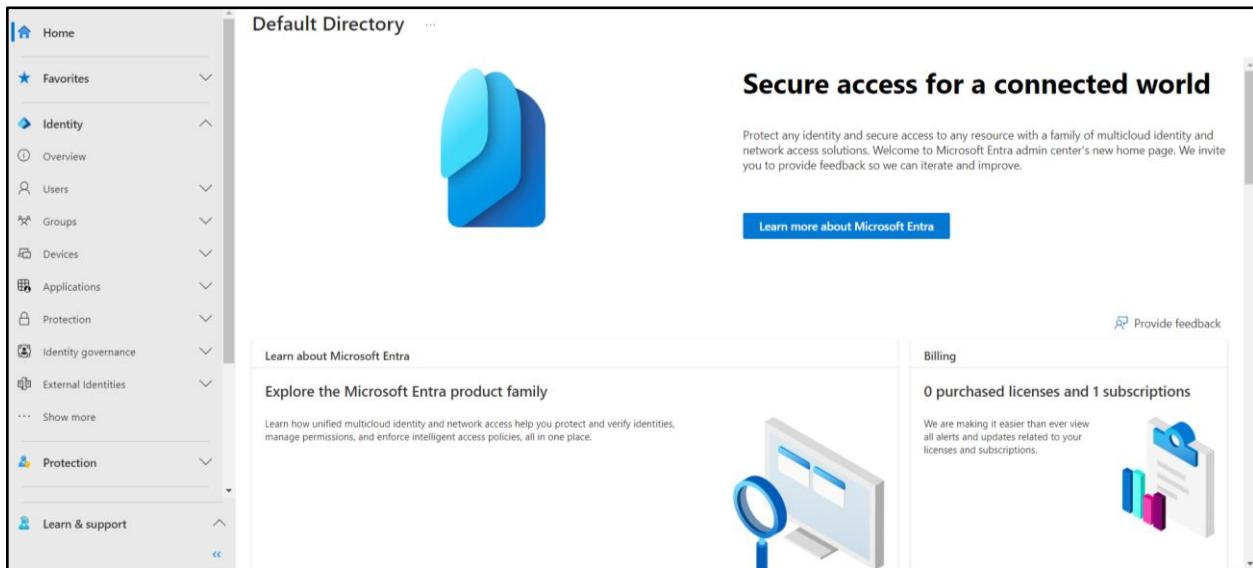


Figure 9.1: Microsoft Entra Admin Centre Interface

2. Go to **Protection**→**Multi-factor authentication**→**Account lockout**. Refer to Figure 9.2.

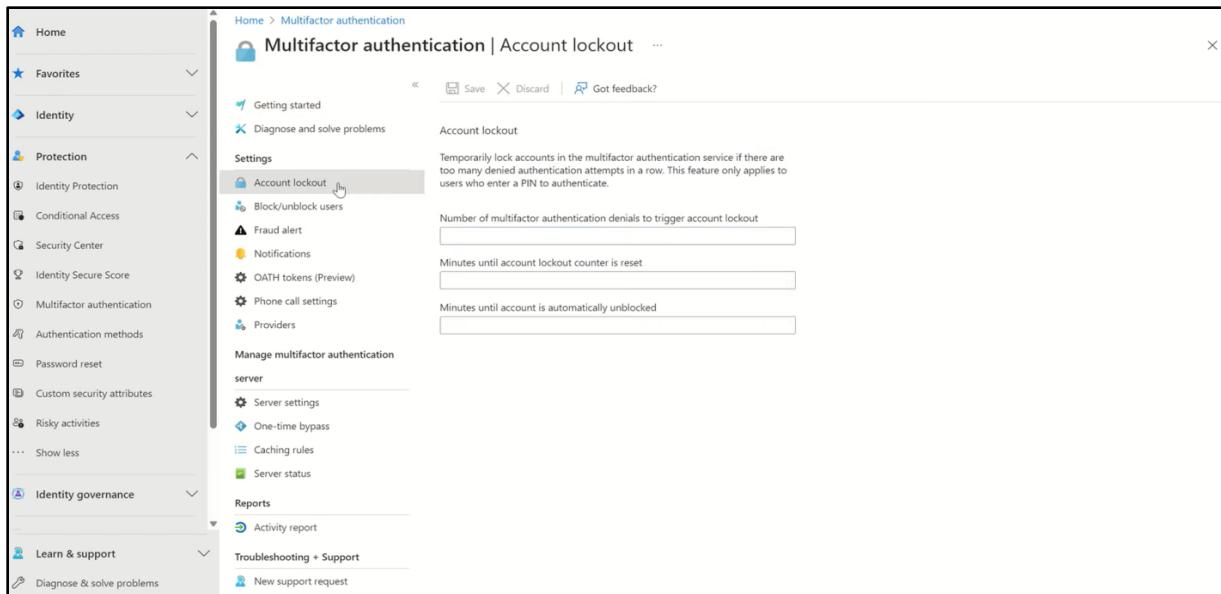


Figure 9.2: Account Lockout

3. Input the appropriate values for one's environment and click **Save**.

9.6.2 Block and Unblock Users

In case a user's device is lost or stolen, Microsoft Entra multi-factor authentication attempts for the associated account can be blocked. Authentication attempts for blocked users are automatically denied, and the block remains effective for 90 days from the time of blocking.

Perform following steps to block/unblock users:

1. Navigate to **Protection** → **Multi-factor authentication** → **Block/unblock users**. Refer to Figure 9.3.

The screenshot shows the 'Multifactor authentication | Block/unblock users' page. On the left, there's a navigation sidebar with sections like Home, Favorites, Identity, Protection, and Learn & support. Under 'Protection', 'Multifactor authentication' is selected. The main content area has a header 'Multifactor authentication | Block/unblock users'. Below it, there's a 'Block/unblock users' section with a note: 'A blocked user will not receive multifactor authentication requests. Authentication attempts for that user will be automatically denied. A user will remain blocked for 90 days from the time they are blocked. To manually unblock a user, click the "Unblock" action.' There are tabs for 'Blocked users' (selected) and 'Unblocked users'. The 'Blocked users' table has columns for User, Reason, Date, and Action. It shows 'No results'. On the right, there's a 'Block a user' modal with fields for 'User *' (with placeholder 'username@domain.com') and 'Reason *' (with placeholder 'Example: "Lost phone"').

Figure 9.3: Block/Unblock Users

2. Press **Add** to block a user. Enter the username in the format `username@domain.com` and provide a comment in the **Reason** box. Refer to Figure 9.4.

This screenshot is similar to Figure 9.3, but the 'Add' button in the top right of the main content area is highlighted with a cursor, indicating the next step. The rest of the interface and modal are identical to Figure 9.3.

Figure 9.4: Adding User to Block

3. Click **OK** to block the user. The user will be blocked.

9.6.3 Fraud Alert

The Fraud Alert feature allows users to report any attempts to fraudulently access their resources. If users receive an unfamiliar and suspicious MFA prompt, they can report fraudulent attempt using the Microsoft Authenticator app or their phone.

Configurable options for fraud alert include:

1. Automatic User Block:

- If a user reports fraud, Microsoft Entra MFA blocks attempts for that user's account for 90 days or until an administrator unblocks it.
- Administrators can review sign-ins through the sign-in report and take necessary action to prevent future fraud. Subsequently, the administrator can unblock the user's account.

2. Fraud Reporting Code during Greeting:

- During a phone call for multi-factor authentication, users typically press # to confirm sign-in. To report fraud, users enter a code before pressing # (default is 0, customizable).
- If automatic blocking is enabled, after entering 0# to report fraud, users are required to press 1 to confirm the account blocking.

Enabling and Configuring Fraud Alerts

1. Log in to the Microsoft Entra admin center with Authentication Policy Administrator access. Refer to earlier Figure 9.1.

2. Go to **Protection** → **Multi-factor authentication** → **Fraud alert**. Refer to Figure 9.5.

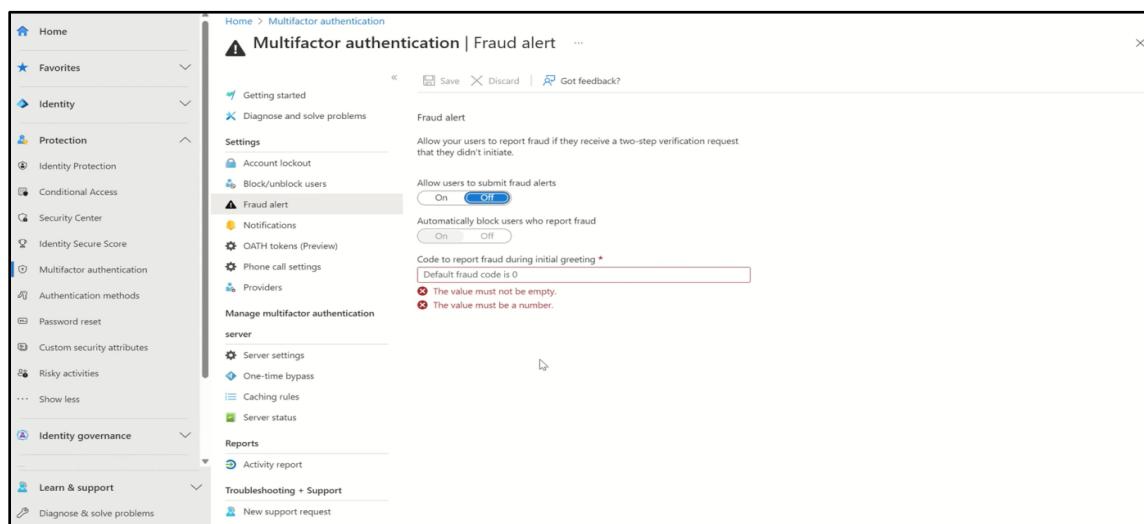


Figure 9.5: Fraud Alert

3. Turn ON **Allow users to submit fraud alerts**. Adjust settings to automatically block users who report fraud or Code to report fraud during initial greeting as required. Refer to Figure 9.6.

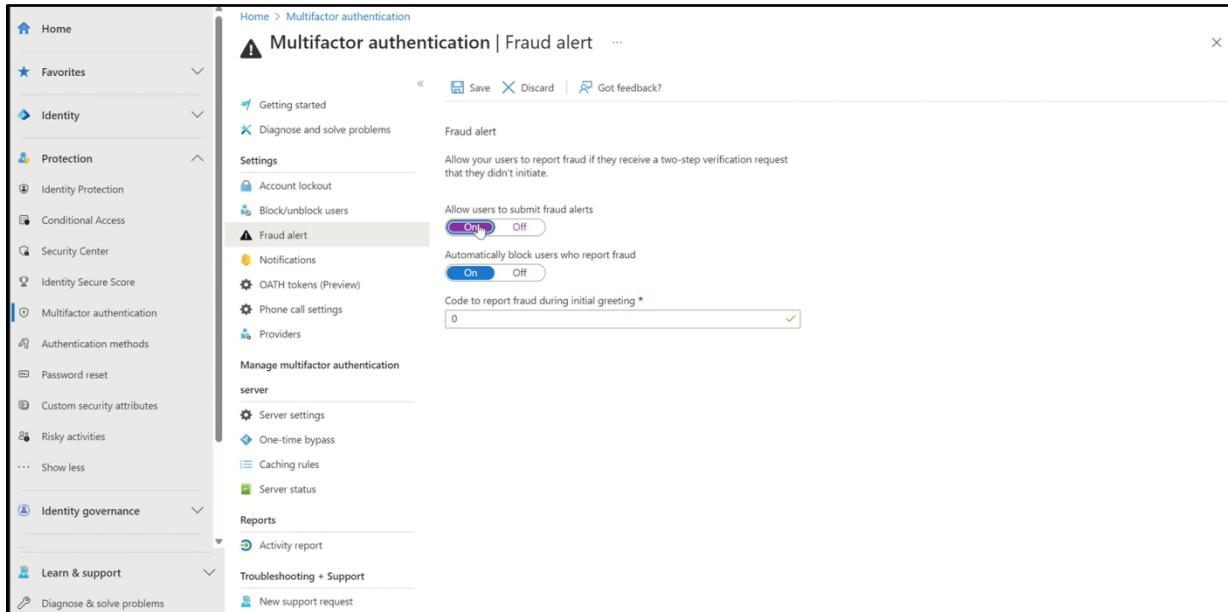


Figure 9.6: Allowing Users to Submit Fraud Alerts

4. Click **Save** to apply the changes.

Similarly, developers can try the remaining available settings for Microsoft Entra multi-factor authentication.

9.7 Azure Firewall and Network Security Groups (NSGs)

Azure Firewall acts as a vigilant security guard for Microsoft Azure cloud resources, carefully monitoring and controlling incoming and outgoing traffic. It allows only authorized and safe traffic while blocking harmful or unauthorized access.

Key features of Azure Firewall include:

Filtering Rules

Users define permissible traffic through detailed rules based on factors such as IP addresses, ports, and website categories.

High

Azure Firewall ensures uninterrupted operation with automatic capacity adjustments and backup systems, providing exceptional reliability.

Deep

It conducts thorough examinations of network traffic, including encrypted packets, to identify and mitigate potential threats such as malware.

Cloud

Seamlessly expands with the growth of cloud environments, accommodating increased traffic volume and ensuring consistent

Azure Firewall is available in three Stock Keeping Unit (SKUs):

1. Standard:

- Ideal for most organizations.
- Comprehensive Filtering: Allows detailed rules and controls based on criteria such as IP addresses, ports, and Website categories.
- Threat Intelligence: Capable of detecting and responding to emerging threats effectively.
- Stateful Inspection: Monitors the state of connections to ensure only legitimate traffic is allowed.

2. Premium:

- Well-suited for highly sensitive environments.
- Advanced Threat Detection: Identifies and mitigates sophisticated attacks to keep sensitive data secure.
- Basic DDoS Protection: Helps prevent disruptions by blocking or mitigating malicious traffic.

3. Basic:

- Useful for low-risk deployments.
- Simple Network Configurations: Suited for organizations with straightforward setups that do not require advanced security features.
- Basic Traffic Control: Provides simple rules for allowing or blocking traffic based on criteria such as IP addresses and ports.

Network Security Groups (NSGs)

Network Security Groups (NSGs) in Azure serve as powerful guardians, orchestrating inbound and outbound network traffic for diverse Azure resources.

Let us explore the key facets that define NSG rules, shaping a robust security framework.

1. Rule Composition:

- Name: Unique identifiers, up to 80 characters, ensuring clarity and

- conformity.
- Priority: Numeric values (100 to 4096) orchestrating rule processing, where lower values signify higher precedence.
 - Source/Destination: Versatile options such as any, specific IP addresses, CIDR blocks, service tags, or application security groups.
 - Protocol: Offers flexibility with choices such as Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Internet Control Message Protocol (ICMP), Encapsulating Security Payload (ESP), Authentication Header (AH), and so on.
 - Direction: Determines if the rule pertains to inbound or outbound traffic.
 - Port Range: Specifies individual or ranges of ports.
 - Action: Clearly defined as allow or deny.

2. Rule Processing Dynamics:

- Rules are meticulously assessed based on five-tuple information (source, source port, destination, destination port, and protocol).
- Ensures exclusivity by disallowing two rules with identical priority and direction.
- Utilizes flow records for managing existing connections, establishing NSGs as stateful guardians.
- Rule modifications impact new connections, ensuring existing connections remain uninterrupted.

9.8 Configuration and Monitoring of Distributed Denial of Service (DDoS) Protection

Distributed Denial of Service (DDoS) attacks pose significant threats to the availability and security of applications in the cloud. These attacks aim to overwhelm an application's resources, causing it to be inaccessible to genuine users. DDoS attacks can target any Internet-accessible endpoint.

Azure DDoS Protection coupled with good application design, offers robust defenses against such attacks. Refer to Figure 9.7 for DDoS protection architecture. It is designed to automatically adapt and safeguard specific Azure resources within a virtual network. Enabling protection is straightforward, applicable to both new and existing virtual networks, and it does not necessitate changes to the applications or resources.

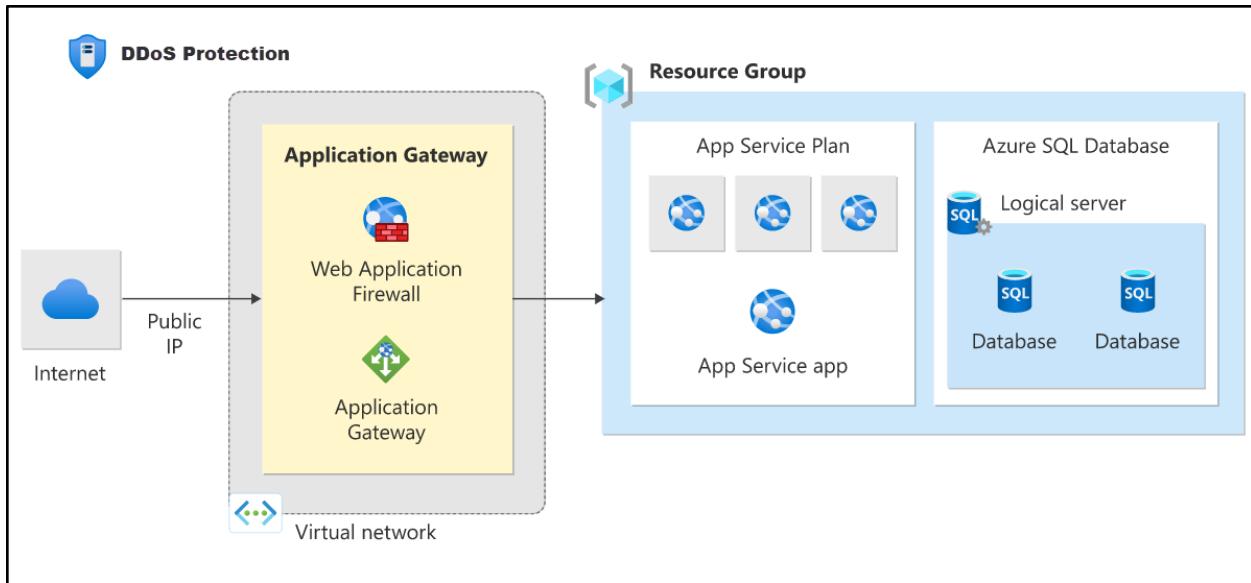


Figure 9.7: DDoS Protection Architecture

9.8.1 DDoS Network Protection

Azure DDoS Network Protection, paired with good application design, offers advanced features to shield against DDoS attacks. It is automatically adjusted to safeguard the specific Azure resources within a virtual network.

To create a DDoS protection plan, follow these steps:

1. Click **Create a resource** in the upper left corner of the Azure portal. Refer to Figure 9.8.

The screenshot shows the Microsoft Azure Portal interface. At the top, there's a navigation bar with 'Azure services' and several icons: Create a resource, Monitor, App Services, Application Insights, All resources, Resource groups, Microsoft Entra ID, Azure AD B2C, Azure Cosmos DB, and More services. Below this is a 'Resources' section with tabs for 'Recent' and 'Favorite'. A table lists recent resources with columns for Name, Type, and Last Viewed.

Name	Type	Last Viewed
app	Application Insights	an hour ago
MyNewVM_group	Resource group	an hour ago
TestM1-nsq	Network security group	21 hours ago
johnstorage23	Storage account	21 hours ago
ADFdemotc	Data factory (V2)	2 days ago
storagedemotc	Storage account	2 days ago
ResourceGroup1	Resource group	6 days ago
Azure for Students	Subscription	a week ago
cttestres	Resource group	2 weeks ago

Figure 9.8: Microsoft Azure Portal Interface

2. Search for the term '**DDoS**'. When '**DDoS protection plan**' appears in the search results, click it. Refer to Figure 9.9.

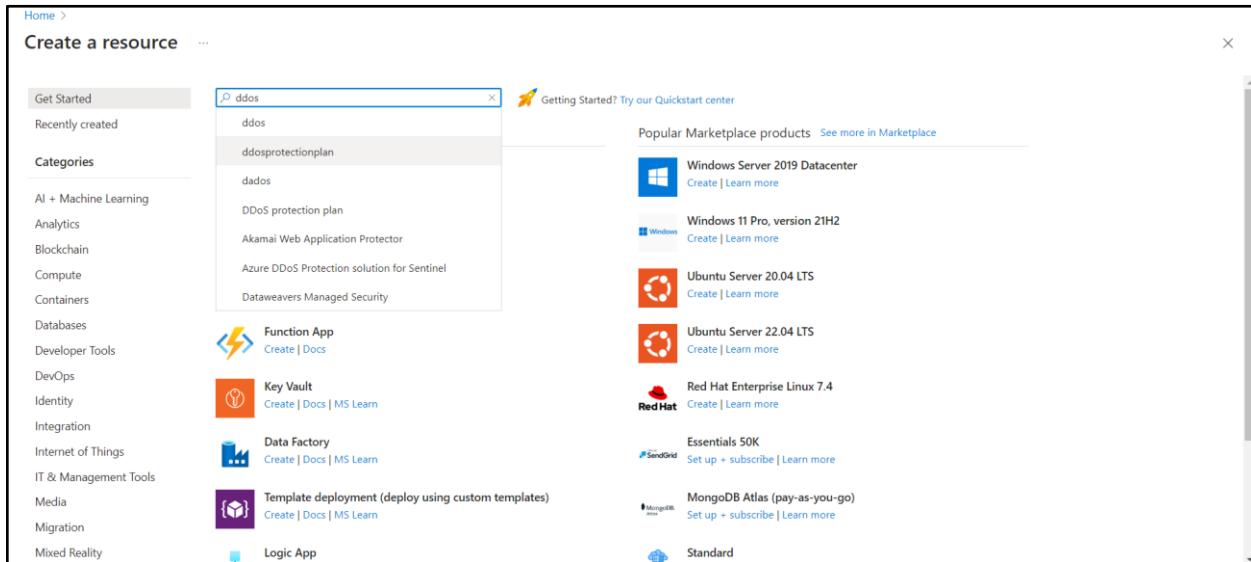


Figure 9.9: Searching DDoS Protection Plan

3. Click **Create**. Refer to Figure 9.10.

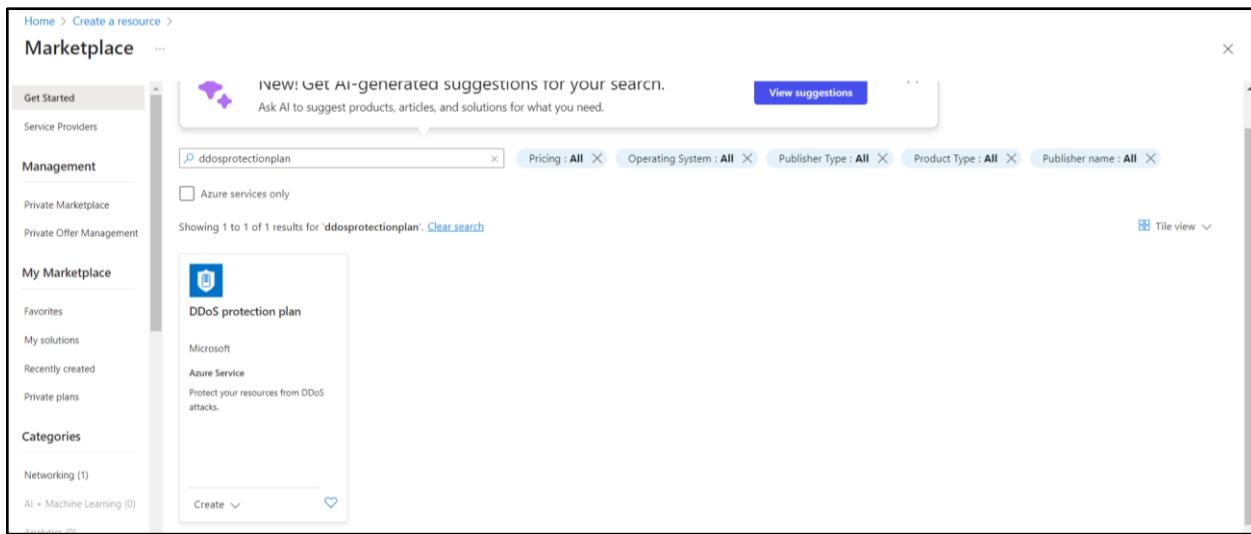


Figure 9.10: Creating DDoS Protection Plan

4. Enter or choose the required values as given in Table 9.1. Refer to Figure 9.11 for instance.

Setting	Value
Subscription	Select the subscription.
Resource group	Select Create new and enter MyResourceGroup.
Name	Enter 'MyDdosProtectionPlan'.
Region	Enter 'East US'.

Table 9.1: Values Required to Create DDoS Protection Plan

The screenshot shows the 'Create a DDoS protection plan' wizard in the 'Basics' tab. It includes fields for Project details (Subscription: Azure for Students, Resource group: (New) MyResourceGroup, both marked with red asterisks), Instance details (Name: MyDdosProtectionPlan, Region: East US, both marked with red asterisks), and a note about creating a single plan across subscriptions. Navigation buttons at the bottom include 'Review + create' (highlighted in blue), '< Previous', 'Next : Tags >', and 'Download a template for automation'.

Figure 9.11: Entering DDoS Protection Plan Details

5. Click '**Review + create**' and then, click '**Create**' to complete the process. Refer to Figures 9.12 and 9.13.

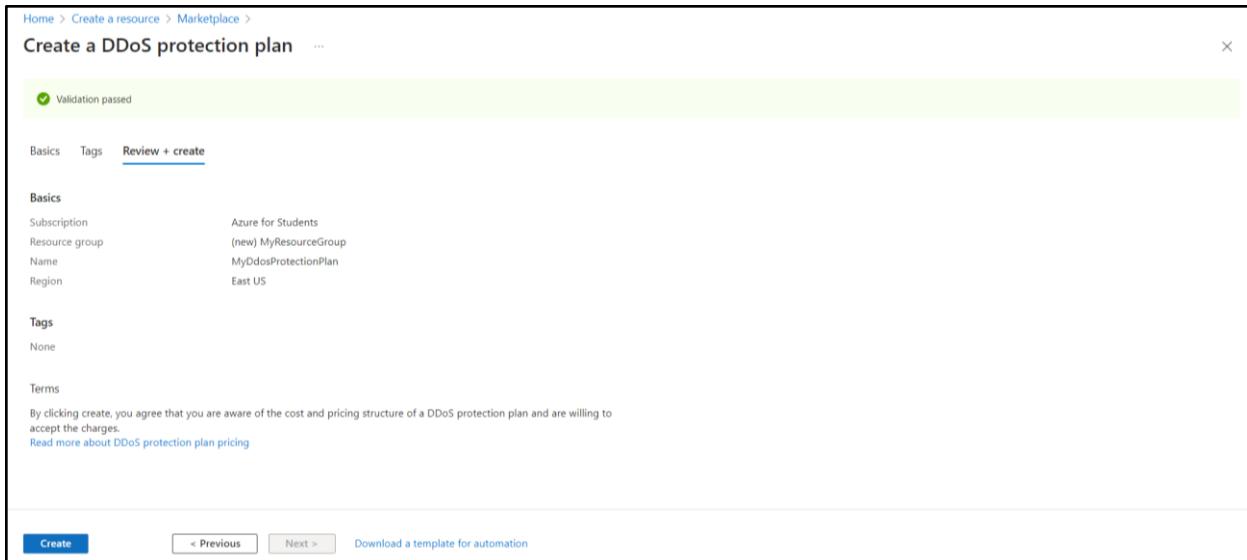


Figure 9.12: Completing DDoS Protection Plan Creation

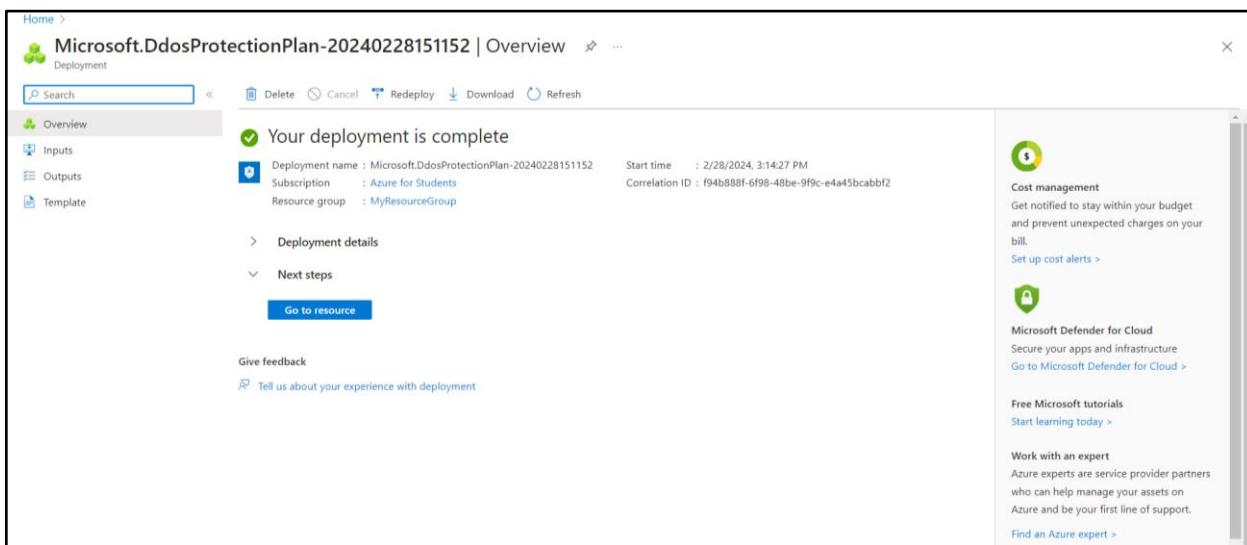


Figure 9.13: DDoS Protection Plan Deployment Success

To enable Azure DDoS Network Protection for a new virtual network, follow these steps:

1. Select **Create a resource** in the upper left corner of the Azure portal. Refer to Figure 9.8.
2. Choose **Networking**, then select **Virtual network**. Refer to Figure 9.14.

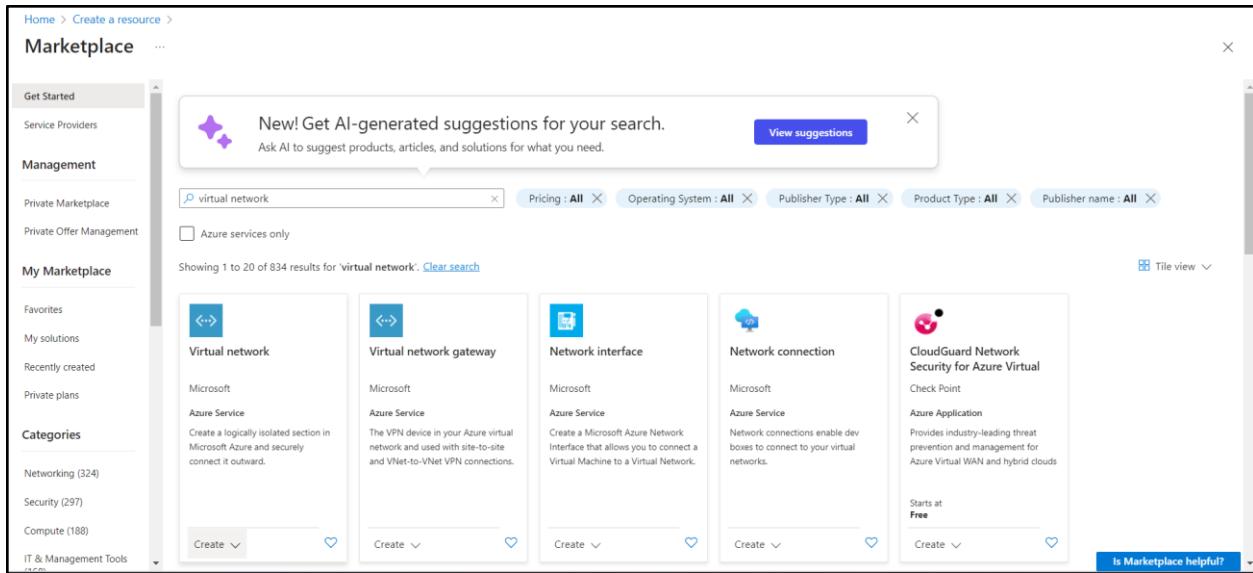


Figure 9.14: Searching Virtual Network

3. Enter or select the required values as shown in Table 9.2 and click **Next**. Refer to Figure 9.15.

Setting	Value
Subscription	Select the subscription.
Resource group	Select Use existing, and then select MyResourceGroup.
Name	Enter 'MyVnet'.
Region	Enter 'East US'.

Table 9.2: Values Required to Create Virtual Network

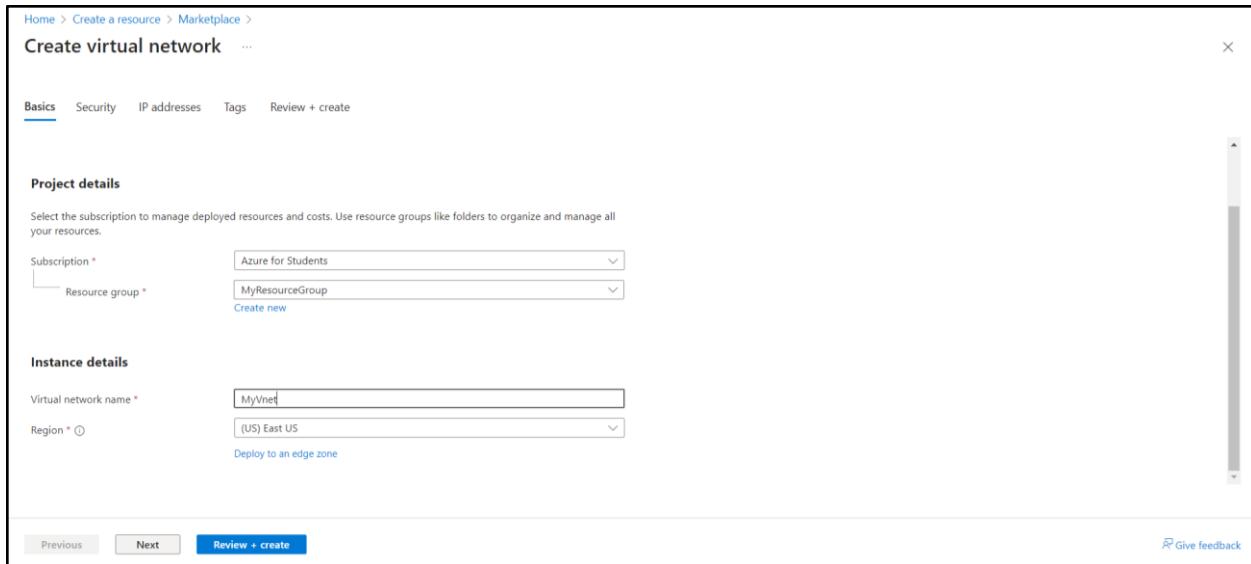


Figure 9.15: Entering Virtual Network Details

4. In the Security pane, click **Enable** on the Azure DDoS Network Protection radio. Refer to Figure 9.16.

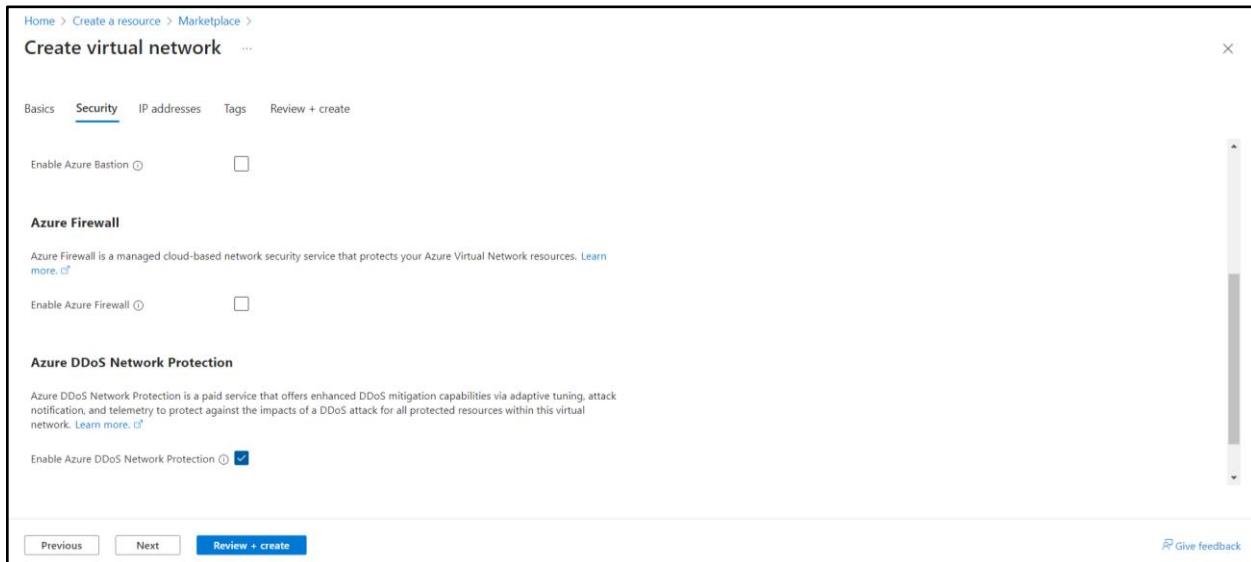


Figure 9.16: Enabling Azure DDoS Network Protection

5. From the DDoS protection plan pane, select '**MyDdosProtectionPlan**'. The plan can be in the same or a different subscription, but both subscriptions must be associated with the same Microsoft Entra tenant. Refer to Figure 9.17.

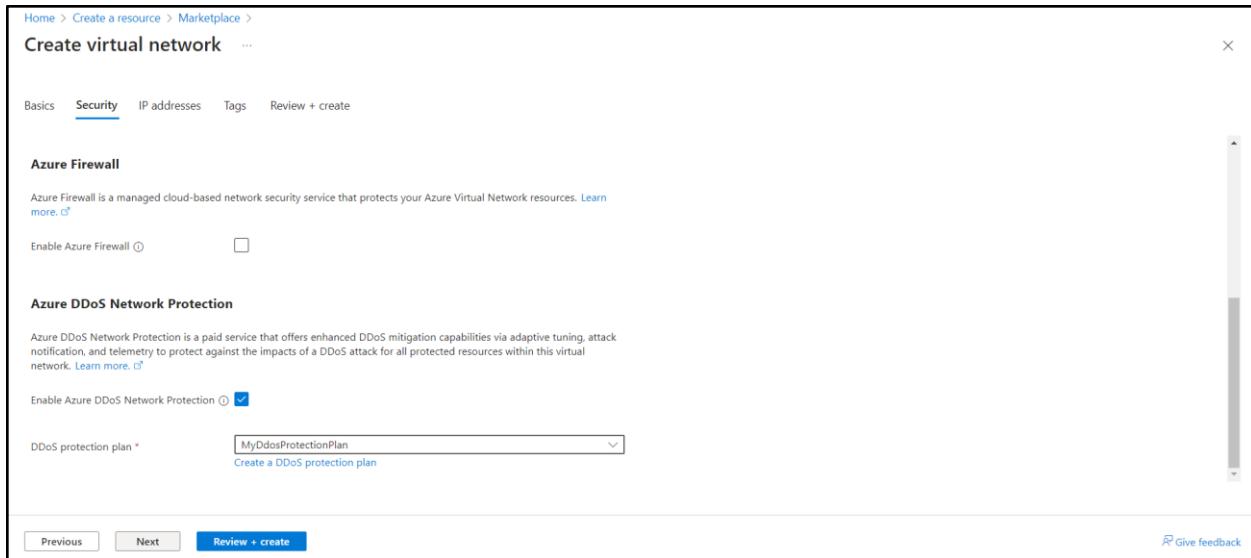


Figure 9.17: Selecting DDoS Protection Plan

6. Click '**Next**'. In the IP address pane, choose **Add IPv4 address space**, enter the necessary values as mentioned in Table 9.3, and then, click **Add**. Delete the subnets which are not in the range. Refer to Figure 9.18.

Setting	Value
IPv4 address space	Enter 10.1.0.0/16.
Subnet name	Under Subnet name, select the Add subnet link and enter mySubnet.
Subnet address range	Enter 10.1.0.0/24.

Table 9.3: Values required to Add Subnet

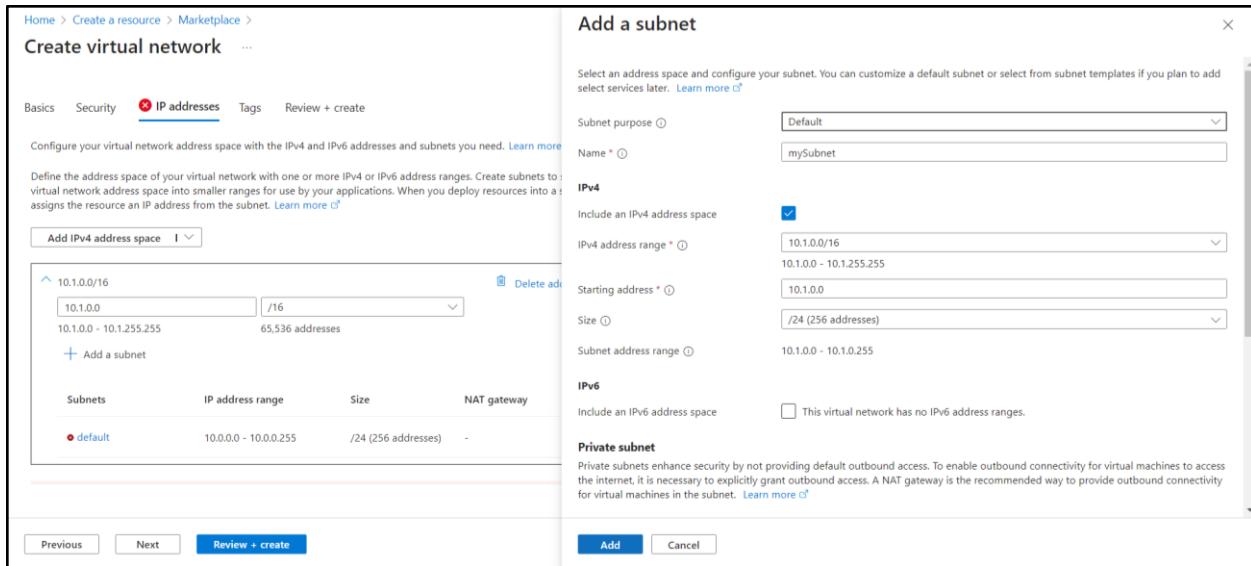


Figure 9.18: Entering Subnet Details

7. Click '**Review + create**' and then, click '**Create**' to complete the process. Refer to Figures 9.19 and 9.20.

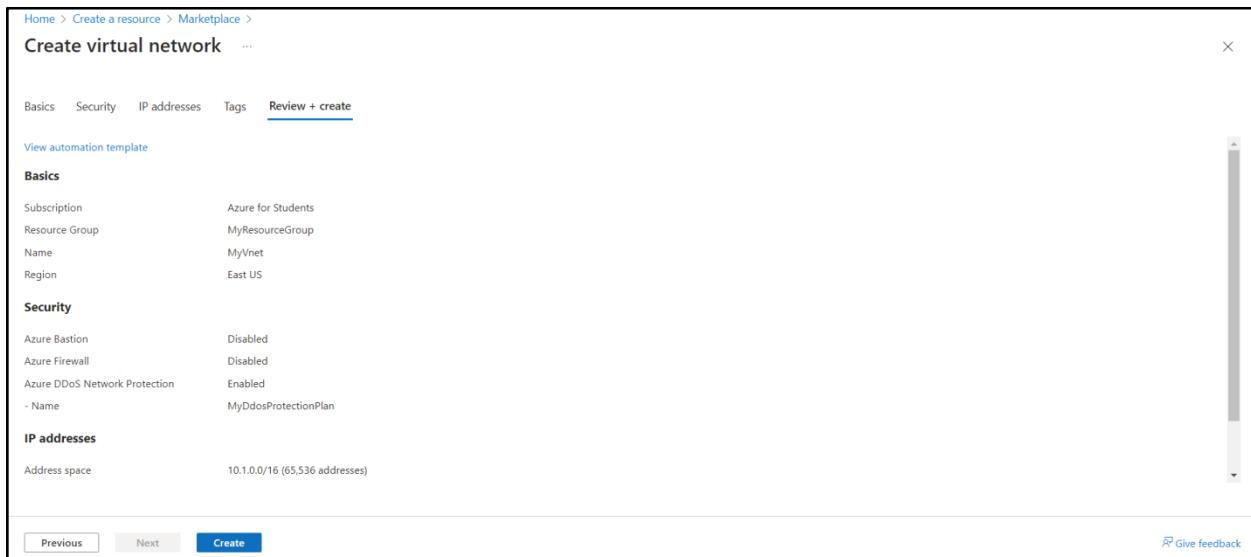


Figure 9.19: Completing Azure DDoS Network Protection for New Virtual Network

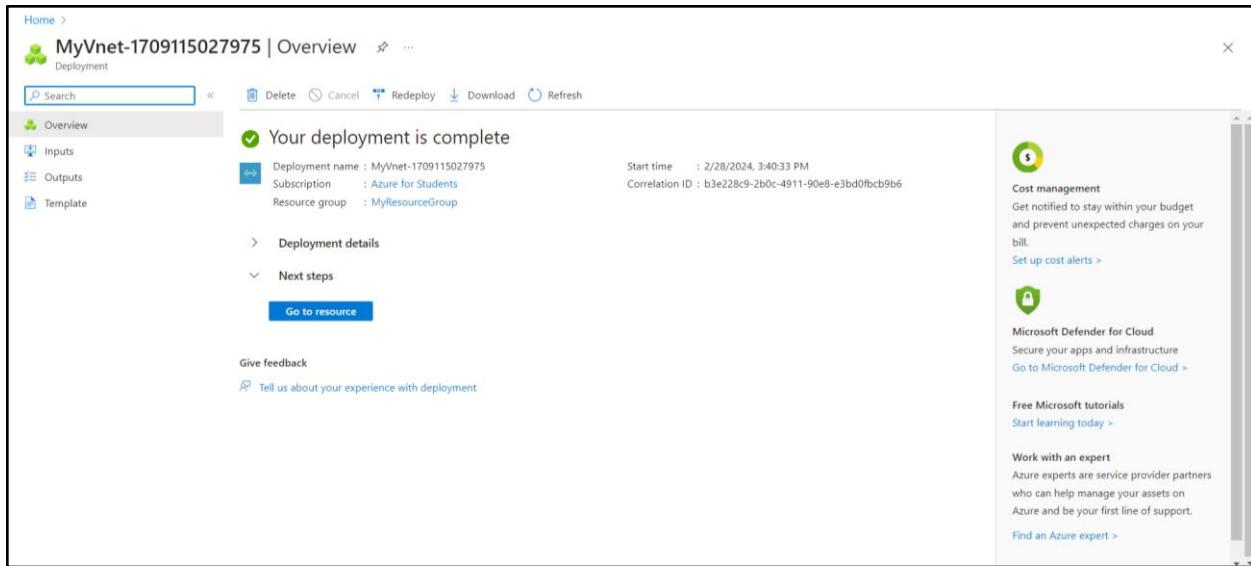


Figure 9.20: Virtual Network Deployment Success

9.8.2 DDoS IP Protection

DDoS IP Protection follows a pay-per-protected IP model. It shares the core engineering features of DDoS Network Protection but adds extra services such as DDoS rapid response support, cost protection, and discounts on Web Application Firewall (WAF). To set up DDoS IP Protection, use Azure PowerShell and follow the given procedure.

To enable DDoS IP Protection for a public IP address, refer to Code Snippet 1.

Code Snippet 1:

```
#Creates the resource group
New-AzResourceGroup -Name MyResourceGroup -Location eastus

#Creates the IP address and enables DDoS IP Protection
New-AzPublicIpAddress -Name myPublicIP -
    ResourceGroupNameMyResourceGroup -Sku Standard -Location "East US" -AllocationMethod Static -DdosProtectionMode Enabled
```

This PowerShell script creates a resource group named 'MyResourceGroup' in the 'East US' location and then, creates a public IP address named 'myPublicIP' within that resource group. It specifies the SKU as 'Standard,' allocation method as 'Static,' and enables DDoS IP Protection with the 'DdosProtectionMode Enabled' parameter.

To enable DDoS IP Protection for an existing public IP address, use the PowerShell

code given in Code Snippet 2.

Code Snippet 2:

```
#Gets the public IP address  
$publicIp = Get-AzPublicIpAddress -Name myPublicIP  
-ResourceGroupNameMyResourceGroup  
  
#Enables DDoS IP Protection for the public IP address  
$publicIp.DdosSettings.ProtectionMode = 'Enabled'  
  
#Updates public IP address  
Set-AzPublicIpAddress -PublicIpAddress $publicIp
```

This script retrieves the existing public IP address named 'myPublicIP' from the resource group 'MyResourceGroup,' sets its DDoS protection mode to 'Enabled,' and then updates the public IP address with the new DDoS protection setting using the 'Set-AzPublicIpAddress' cmdlet.

To check the details of their public IP address and verify that DDoS IP Protection is enabled, one can use PowerShell code given in Code Snippet 3.

Code Snippet 3:

```
#Gets the public IP address  
$publicIp = Get-AzPublicIpAddress -Name myPublicIP -  
ResourceGroupNameMyResourceGroup  
  
#Checks the status of the public IP address  
$protectionMode = $publicIp.DdosSettings.ProtectionMode  
  
#Returns the status of the public IP address  
$protectionMode
```

This script retrieves the existing public IP address named 'myPublicIP' from the resource group 'MyResourceGroup'. It then checks status of its DDoS protection mode and then, returns the status. If DDoS IP Protection is enabled, it will return 'Enabled.'

9.9 Defender for Identity

Microsoft Defender for Identity is a cloud-based security solution designed to fortify identity monitoring across organizations utilizing the Azure environment. It was formerly known as Azure Advanced Threat Protection (Azure ATP).

Its purpose-driven design and multifaceted functionality contribute to a resilient security framework. This design makes it an indispensable component in safeguarding identities across modern, hybrid environments within the Azure ecosystem.

This section delves into the purpose and functionality of Defender for Identity, emphasizing its integration with Microsoft Defender XDR and its role in safeguarding both on-premises and cloud identities.

Purpose of Defender for Identity are as follows:

1. Detecting Advanced Threats:

Defender for Identity serves as a vigilant guardian. It identifies and investigates advanced threats targeting the organization by using signals from both on-premises Active Directory and cloud identities. It plays a crucial role in identifying suspicious activities throughout the cyber-attack kill chain.

2. Proactive Security Posture Assessments:

Defender for Identity aids organizations in proactively assessing their security posture by providing insights into identity configurations and suggesting security best practices. This includes features such as Lateral Movement Paths, offering a clear understanding of potential risks and security assessments available through Microsoft Secure Score.

Functionality of Defender for Identity includes:

➤ Protecting User Identities:

- Insights and Best Practices: Defender for Identity offers security reports and user profile analytics, reducing the organizational attack surface by making it harder for attackers to compromise user credentials.
- Proactive Identity Posture Assessment: The solution allows organizations to identify and resolve security issues before exploitation, providing a clear view of the identity security posture.

➤ Detecting Threats across Modern Environments:

- Complete Visibility: Utilizing data from domain controllers, Active Directory Federation Services (AD FS), and Active Directory Certificate Services (AD CS), Defender for Identity ensures a comprehensive view of the identity environment.
- Sensor Monitoring: Default sensor monitoring for domain controllers and specific sensor types for AD FS/AD CS servers ensure complete identity monitoring.

3. Identifying Suspicious Activities:

- Advanced Threat Identification: Defender for Identity actively identifies rogue users, compromised credentials, lateral movements, and potential domain dominance activities throughout the cyber-attack kill chain.
- Detailed Threat Analysis: Offering a detailed threat analysis, it spots attackers' attempts at reconnaissance, credential compromise, lateral movement, and domain dominance.

4. Investigation and Response:

- Reduced Alert Noise: Minimizing alert noise, Defender for Identity provides a prioritized list of relevant security alerts in a real-time organizational attack timeline.
- Integration with Microsoft Defender XDR: It is seamlessly integrated with Microsoft Defender XDR. The solution enhances security by correlating data from different domains, ensuring greater visibility and accuracy.

9.10 Summary

- ✓ Azure Security is mostly a collaborative effort between Azure and its customers.
- ✓ Azure ensures physical security in IaaS (host, network, and data processing). Azure also handles physical security and operating systems in PaaS.
- ✓ Azure takes on additional responsibilities in the SaaS cloud model, such as physical security, operating system, network control, and applications.
- ✓ Items that developers can put in their vaults can include: Client Secrets, API keys, Passwords, Certificates, Keys to decryption/encryption, and Connection Strings.
- ✓ Microsoft Antimalware comes with following features: Scheduled scanning, malware remediation, signature update, updates to the Microsoft Antimalware Platform, updates to the Microsoft Antimalware Engine, Active protection, Antimalware event collection, and Exclusions.
- ✓ Developers may use Azure IAM to perform several identity and access management tasks such as creating and managing a single identity for all users in their hybrid company, allowing for single sign-on to applications, and more.
- ✓ Multi-Factor Authentication (MFA) enhances security by requiring users to verify their identity through multiple authentication methods.
- ✓ Azure Firewall controls and monitors both incoming and outgoing traffic, while Network Security Groups (NSGs) provide a rule-based approach for managing network traffic effectively.
- ✓ Distributed Denial of Service (DDoS) Protection safeguards Azure resources by dynamically adapting to counteract DDoS attacks, ensuring both availability and security.
- ✓ Microsoft Defender for Identity safeguards organizations by actively detecting advanced threats, offering security insights, and ensuring complete visibility across on-premises and cloud identities in Azure.

9.11 Test Your Knowledge

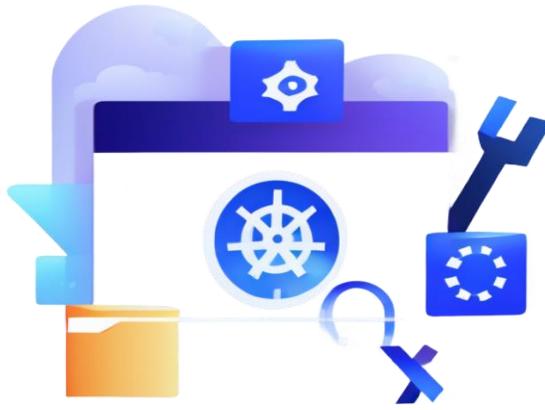
1. Which of the following categories include features of Microsoft antimalware such as service status, suspicious actions, and remedial actions in the operating system's event log?
 - A. Exclusions
 - B. Samples reporting
 - C. Antimalware event collection
 - D. Antimalware engine updates
2. Which of the following can a developer not do with Azure IAM?
 - A. Create and manage a single identity
 - B. Boost user productivity
 - C. Allows for single sign-on to applications
 - D. Update the Microsoft Antimalware
3. Which of the following options is not a feature of Microsoft Antimalware?
 - A. Active Protection
 - B. Microsoft Antimalware Engine
 - C. Connection Strings
 - D. Signature Update
4. Which of the following additional responsibilities does Azure take care of in SaaS?
 - A. Physical Security
 - B. Network Control
 - C. Applications
 - D. All of these
5. Which of the following cannot be put into the Vault by the developer?
 - A. Passwords
 - B. Application
 - C. Code
 - D. None of these

9.11.1 Answers to Test Your Knowledge

- 1. C
- 2. D
- 3. C
- 4. D
- 5. C

Try It Yourself

1. Perform following settings for Microsoft Entra multi-factor authentication:
 - a. Report suspicious activity
 - b. Notifications
 - c. OATH tokens
 - d. Phone call settings
2. Enable DDoS IP Protection for a public IP address.



SESSION 10

HOSTING AND CONSUMING WEB API

Overview

This session describes an overview of hosting Web API in IIS, self-hosting, and hosting on Kestrel Web server. It also describes how to host Web API in Microsoft Azure Web app and on container services. It provides information on methods of consuming Web API in HTTP Client and explains how to use Swagger.

Learning Objectives

In this session, students will learn to:

- Define how to host Web API on IIS, self-host, and Kestrel Web Server
- Define how to host Web API on Microsoft Azure Web app service on Windows
- Explain how to host Web API on containers services
- Explain ways of consuming Web API in HTTP Client and in Postman
- Explain how to use Swagger

10.1 Hosting Web API in IIS, Self-host in a Process, or Host on Kernel

A Web API/Website project, subsequent to creation, should be deployed or published onto a Web server. This would enable users to access the application and utilize it. When deploying files of the application, developers must have a strategy in mind. As the process is complicated and does not involve merely copying application files from one to another server, various tasks should be performed. Developers can choose the appropriate process according to the deployment type.

An understanding of the basic available deployment options for Web applications would help a developer to work towards selecting an appropriate strategy for deployment. Visual Studio 2022 offers strong support for the packaging and deployment of Web applications and services.

10.1.1 Installing IIS on Windows Virtual Machine

Hosting a Web API on the IIS involves setting up IIS and publishing the Web API application on it. Following are the steps to install IIS on a Windows VM:

- 1) Log in to Microsoft Azure and click **All resources** from the right panel. The **All resources** page is loaded, as shown in Figure 10.1.

The screenshot shows the Azure 'All resources' dashboard. At the top, there are filter options: 'Subscription equals all', 'Resource group equals all', 'Type equals all', and 'Location equals all'. Below these are two buttons: 'Unsecure resources' (0 results) and 'Recommendations' (0 results). The main table lists the following resources:

Name	Type	Resource group	Location
NetworkWatcher_westeu	Network Watcher	NetworkWatcherRG	West Europe
TestResource-vnet	Virtual network	TestResource	West Europe
TestVM1	Virtual machine	TestResource	West Europe
TestVM1-ip	Public IP address	TestResource	West Europe
TestVM1-nsg	Network security group	TestResource	West Europe
testvm1224	Network Interface	TestResource	West Europe
TestVM1_OsDisk_1_a2159e44f5d4478eb74cb367d0d8ca90	Disk	TESTRESOURCE	West Europe

Figure 10.1: Azure All Resources Panel

- 2) Click **TestVM1-nsg**, which is of type Network security group. It loads the group's Manage page.
- 3) In the **Network Security Group Manage** page, from the left panel, select Inbound security rules. This loads the default Inbound (incoming request) security settings for defining a setting for Remote Desktop (RDP).
- 4) Click the **+ Add** button. A Settings page is displayed, as shown in Figure 10.2.

Figure 10.2: Adding Inbound Security Rules

- 5) Provide information with **Port ranges** set to **3389** and **Name** as **rdp-access** and click **Add**. Now, Windows VM is ready for login.
- 6) Open **Windows VM** and copy the **Public IP** for login. To login into **Windows VM**, the login credentials that were assigned while creating the VM are required.
- 7) Open **Remote Desktop** and paste the IP address with the port. The IP address will look like 40.114.1xx.xxx:3389. Note 3389 port number added at the end.

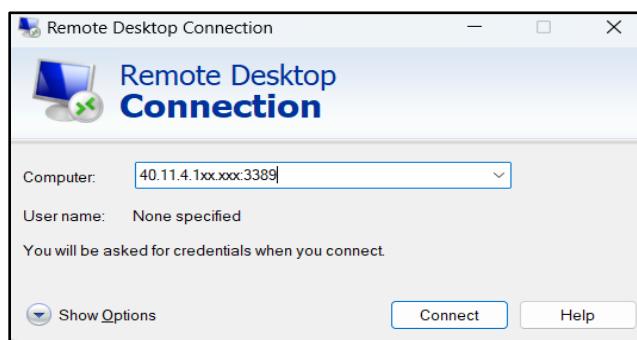


Figure 10.3: Log in to Windows VM by Using RDP

- 8) Click **Connect**, as shown in Figure 10.3. This will prompt for user ID and password. Correct credentials should be provided for logging into Windows VM. After login, the VM loaded into the remote desktop windows can be seen.

- 9) From the **Start** menu, run **Server Manager** to install IIS. The Server Manager application is displayed.
- 10) Select **Add Roles and Features** from the main window. From the **Add Roles and Features Wizard** window click **Next**, which will load the second step of the wizard.
- 11) Select **Role-based or Feature-based installation** option button as shown in Figure 10.4.

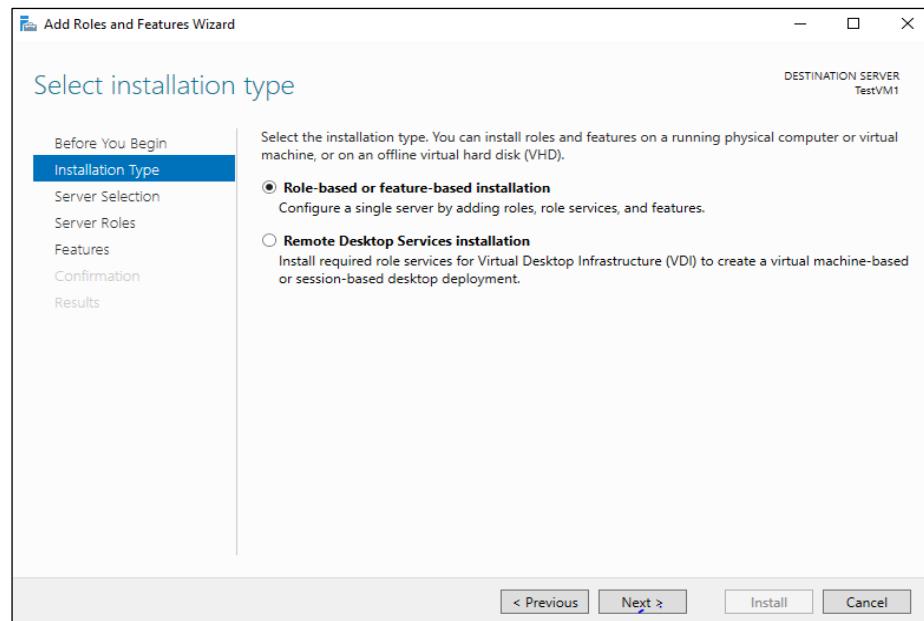


Figure 10.4: Add Roles and Features

- 12) Click **Next**, which will load the Destination Server Selection window. By default, the same server name is selected. Click **Next** to proceed to the Server Role selection window. Figure 10.5 shows the list of roles and features.

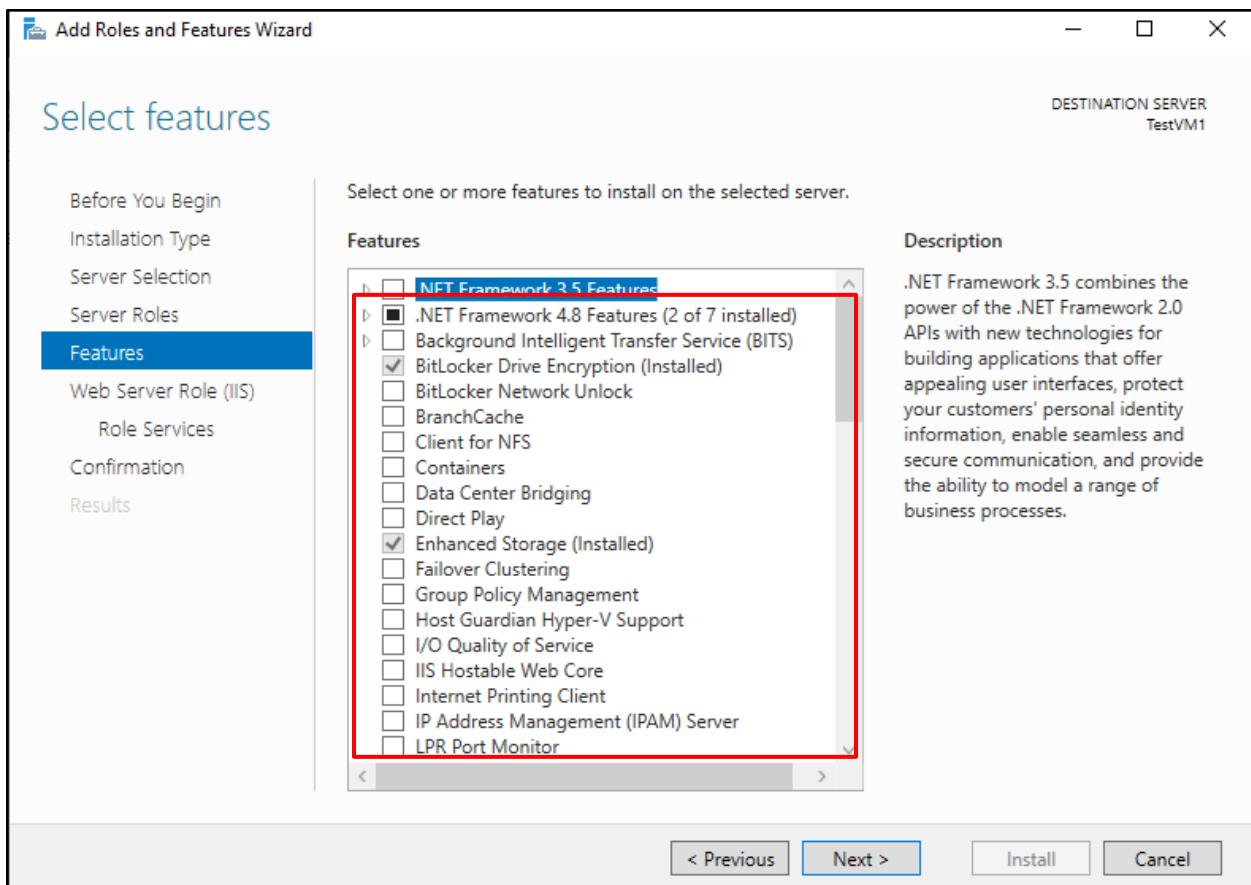


Figure 10.5: Window Server Roles and Features

- 13) Scroll down the list to locate the **Web Server (IIS)** check box, which will bring up a confirmation window. Click **Add Feature** and return to the **Main Wizard** window. Now, click **Next** to load the feature window.
- 14) Expand .NET Framework 4.8 features and click the ASP.NET 4.8 check box as shown in Figure 10.6.

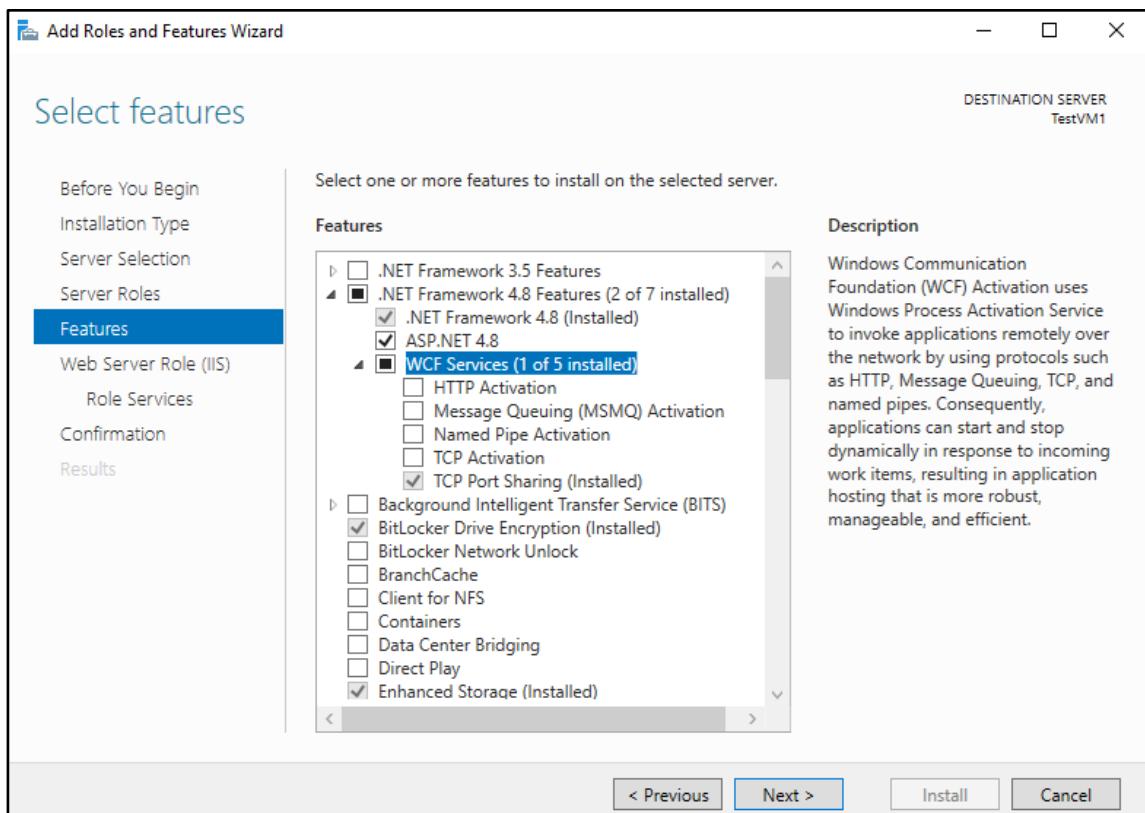


Figure 10.6: .NET Framework 4.8 Options

- 15) Click **Next**. The next window mentions the summary of what has been selected. Again, click **Next** to proceed further.

The next window will be displayed to install the **Roles and Services** into IIS.

- 16) Install ASP.NET **Application Development** feature. To install, scroll down the list and check ASP.NET 4.8, which will bring up a confirmation window Refer to Figure 10.7.

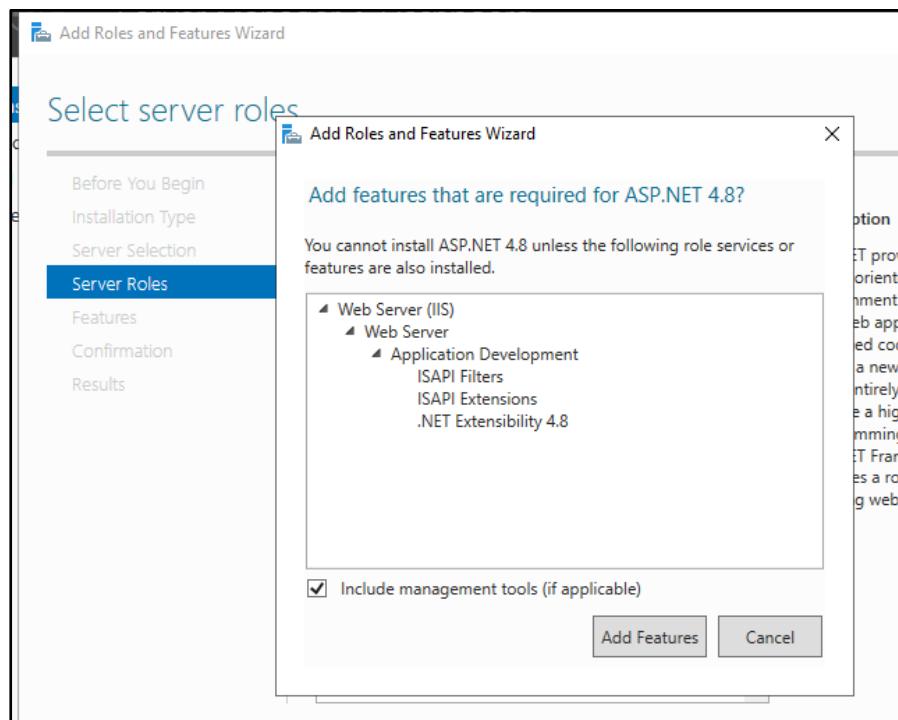


Figure 10.7: Confirmation Window

- 17) In the pop-up window, click **Add Features**; this will check all the dependent features required to run ASP.NET in IIS. The final option is shown in Figure 10.8.

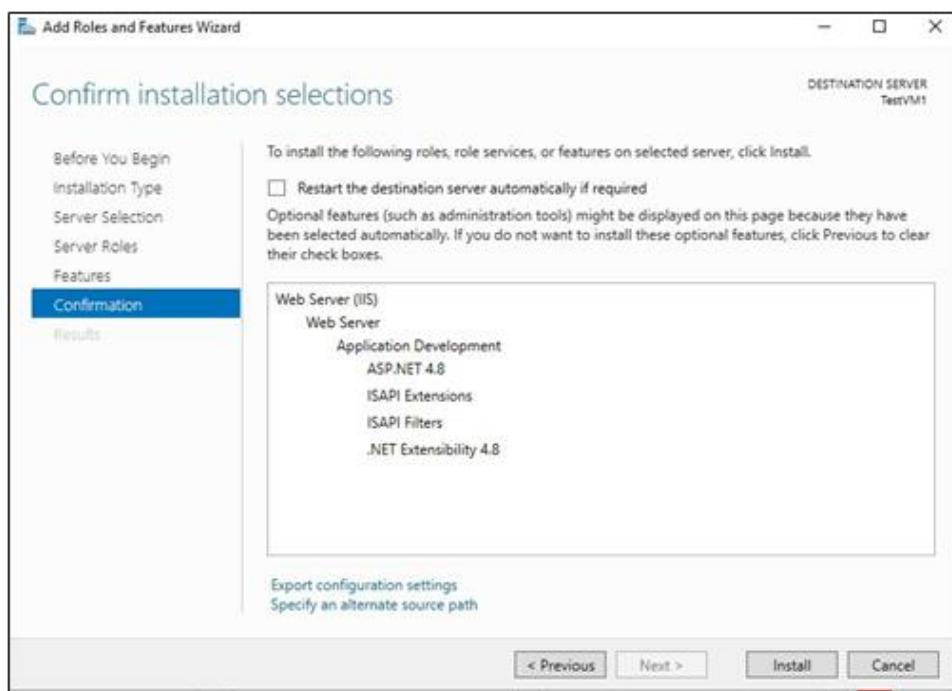


Figure 10.8: Selected Application Development Features

- 18) Click **Next**. It will load the final summary feature installation confirmation window. Select **Install** and this will start installing the dependencies for ASP.NET 4.8 and subsequently install IIS.

Now, the IIS is successfully installed in the Window VM. To check if the IIS is installed properly or not, use the VM's Public IP address and browse it using any browser. It will load the IIS welcome page.

10.1.2 Hosting a Web API on IIS

After installing IIS on the VM, developers can now publish an existing Web API application and host it on the VM-based IIS.

Following steps should be performed to publish:

- 1) Open the StudentServices application that was made in an earlier session.
- 2) To display different ways to publish window, as shown in Figure 10.9, right-click the StudentServices project and choose **Publish** from the menu that pops up.
- 3) After choosing Folder from the menu on the left, click **Publish** to put the application in the /bin/Release/Publish directory. Refer to Figure 10.9.

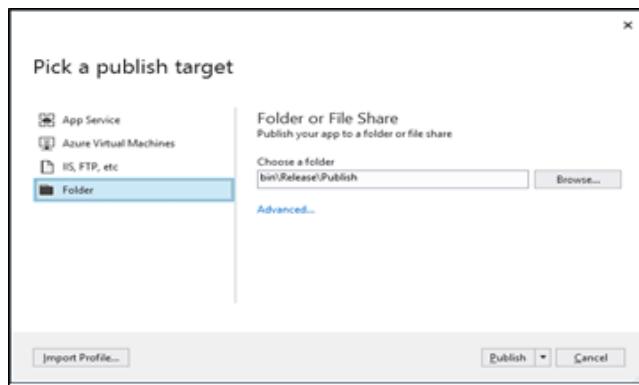


Figure 10.9: Publish Application in Folder

- 4) Explore the files in the /bin/Release/Publish directory and make a.zip file. The zip file will make it easier to copy the content to the Windows VM to be used.
- 5) Copy the zip file and paste it into the wwwroot directory of the Windows VM server. Most of the time, the wwwroot folder can be found in the path C:\inetpub\wwwroot. Now, delete the content that is already there and

unzip the content that is in wwwroot. Make sure all files are unzipped directly in the wwwroot folder.

Now, developers can use any browser to navigate to the VM's public IP. This will bring up the page for API helpers. If one clicks the API link in the top menu, the API helper page will be displayed.

- 6) Use **Postman** application to test the API (section 10.4.2 describes Postman in detail).

Note: There may arise an issue in connecting to the Database from the Web API application. In case there is any such issue, copy the Connection String from Azure SQL Database settings and replace it in web.config. A Connection string example is given in Code Snippet 1.

Code Snippet 1:

```
Server=tcp:azureclouddb.database.windows.net,1433;Initial  
Catalog=StudentDb;Persist Security Info=False;User  
ID=billsmith;Password=Dracula99;MultipleActiveResultSets=False;  
Encrypt=True;TrustServerCertificate=False;Connection  
Timeout=30;
```

10.1.3 Self-hosting a WebAPI

Self-hosting separates the Web API hosting from the primary (or client) application. As a result, the actual Web API hosting occurs on a server-based console application or Windows Service. During development, the self-hosting project is a console program. Therefore, a Program.cs file is generated in the project and associated with it. This Program.cs file contains the primary method, which is the entry point for the application - the primary mechanism used for creating self-hosting Web API code.

The registration of virtual configuration upon the launch of the hosting application, which may be a Windows service or application, is a vital requirement.

10.1.4 Using Kestrel for WebAPI Hosting

The open-source Web server that comes with ASP.NET templates is Kestrel. It can be used to host ASP.NET applications on all operating systems. Kestrel can be used on its own or with a reverse proxy server (for example, Apache, Nginx, and IIS). A reverse proxy server gets HTTP requests from the Internet, processes them, and then, sends them to Kestrel.

In a reverse proxy scenario, many applications on the same server share the same IP address and port. Kestrel does not let the same IP address and port be used by more than one application. If Kestrel is set up to listen on a port, it can handle all traffic going through that port, no matter what the host header says. A reverse proxy that shares ports can also send requests to Kestrel on a different IP address and port.

Azure Application Service Web Apps (or just Web Apps) is a service that hosts mobile backends, REST APIs, and Web apps. It can be made with any programming language, such as .NET Core, Python, Java, or PHP. On a Windows platform, the apps tend to work well and scale well.

10.2 Hosting Web API in Microsoft Azure Web App

The hosting service is reliable and can grow and fix itself. A resource group is made when the first ASP.NET Web app that uses this service works well. This group has a service plan, an Azure Web app, and the application that has been used.

Steps to publishing a Web API application to the Azure App service are as follows:

- 1) Open the application called StudentServices that was made. Right-click the StudentServices project and choose **Publish** from the menu that appears. This will bring you to the profile page for Publish.

As there is already a publish profile, a new one should be made for Web Apps publish. The Publish profiles can be seen in Figure 10.10.

- 2) To make a profile, click the link that says **Add a publish profile**.

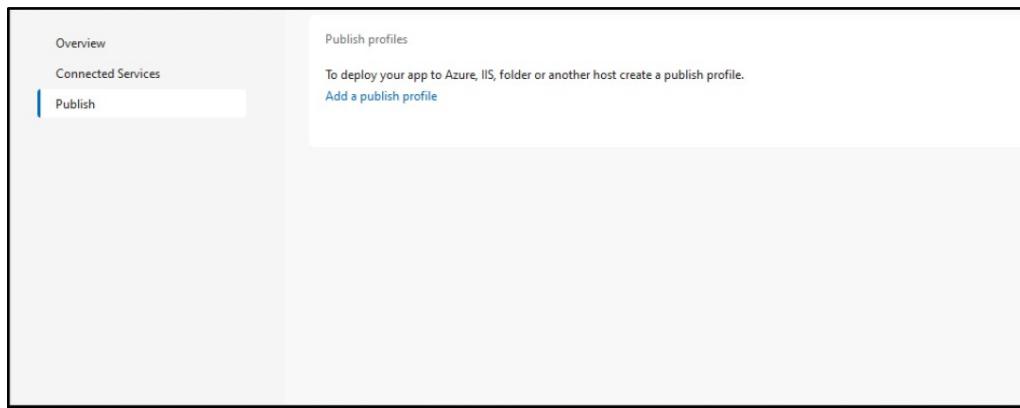


Figure 10.10: Publish Profiles

- 3) To make an Azure Web App, choose the App Service from the context menu for the Publish option. Make sure to choose the **New** option.

Figure 10.11 shows the choices for the Web App.

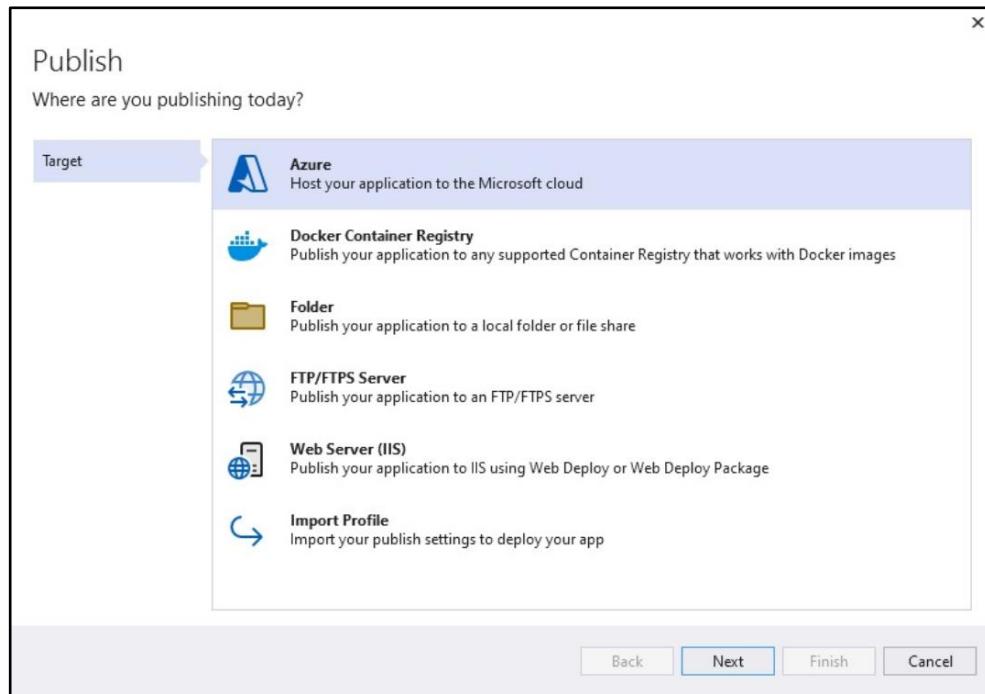


Figure 10.11: Web App Publish Profiles

- 4) In the next window, which is named 'Create App Service,' choose **Already have an account? Sign in**. Sign in with your Azure account and the Account will appear in the upper right corner. Choose the account from the drop-down menu next to the Account. This will generate a Web application automatically, having a name chosen by you. Figure 10.12 shows how to change the name of the app.

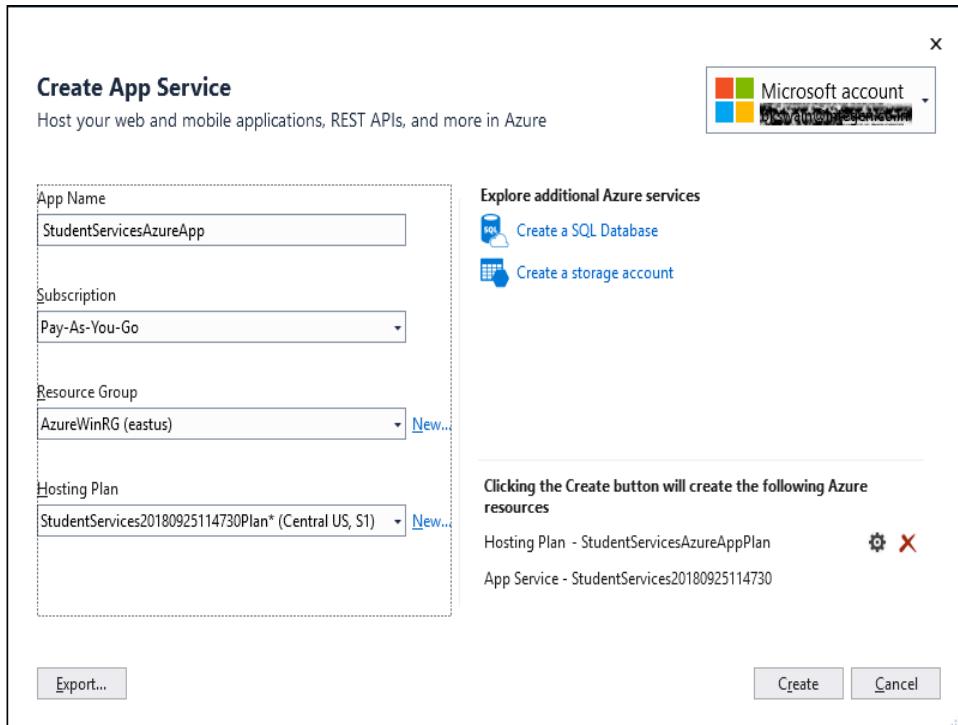


Figure 10.12: Web App Settings

- 5) Select **Create**. This will generate the Publish profile and create the Web App and Web API automatically. After the publish operation is done, the Web API application will open in the default browser by itself. The address is <http://studentservicesazureapp.azureWebsites.net>.
- 6) You can test the API using the Postman app.

10.3 Hosting a Web API on Container Service

Windows Containers can be used as a means for automating the deployment process of .NET applications. They help in delivering these applications as independent and portable containers that can run on any platform, such as Cloud, Linux, and Windows.

The salient feature of a container is that it retains the same environment across various deployments. A developer can debug an application on one system and deploy the same on another without any change in the running environment. The container service is established on the Docker platform, the main open-source platform that facilitates automatic deployment.

An application usually consists of one or more containerized components, also known as microservices. If there are multiple such components, it is essential to organize them such that the application performs as intended. This process of organizing is called container orchestration. It involves controlling the containers' lifecycle, specifically in a big and dynamic environment. It helps in automating the deployment, availability, scaling, and moving containers. This is possible using one of the famous container orchestration tools such as Docker Swarm or Kubernetes.

The importance of a modern API and the use of a Docker container for hosting and running the API in the cloud can be described as follows:

Importance of Containers

The use of containers increases deployment speed. This is because all dependencies of an application are within the container itself.

Application with Dependencies

At times, dependencies may get installed on a VM. Thus, when applications are deployed on that VM, there is a possibility of an impedance mismatch between the container and the dependency on virtual machine. This mismatch refers to issues arising while representing application data. To minimize this risk, the dependency can be kept along with the container.

Deployment of Web Application with its Dependencies in a Container

A specific version of IIS and the ASP.NET framework are usually packaged. When it is rolled out as a container, the developer is confident that the Web application will perform as per the expectations, since the same version of IIS and ASP.NET will be used for developing and debugging.

10.4 Consuming Web API

One can consume a Web API using an `HttpClient` or using Postman.

10.4.1 HttpClient

HTTP requests and responses from a URL are sent and received with the help of a base class. This is a feature of the `HttpClient` class. This is an async feature that the .NET Framework can handle. Modern HTTP clients such as `HttpClient` are used by .NET applications. Consider a situation where an ASP.NET Web API service is set up to show off a particular feature. Use a client-side script (jQuery or JavaScript) to call the POST, PUT, GET, and DELETE operations in a Web application. Use `HttpClient` to connect a desktop or console application to the same Web API.

There are several ways to use the ASP.NET Web API with the HttpClient class.

Table 10.1 shows these steps, how to do them, and their explanations.

Method	Description
GetAsync	GET request is sent to a specified Uniform Resource Identifier (URI).
PostAsync	POST request is sent to a specified URI.
PutAsync	PUT request is sent to a specified URI.
DeleteAsync	DELETE request is sent to a specified URI.

Table 10.1: HttpClient Class Methods

Following are steps to implement Web API with MVC Application using HttpClient class:

Step 1: Use Visual Studio 2022 to create an MVC Web Application, say StudentApp. Add StudentModel class into **Models** folder. The model code is given in Code Snippet 2.

Code Snippet 2:

```
public class StudentModel{
    public int SID { get; set; }
    public string SName { get; set; }
    public string SEmail { get; set; }
    public DateTime JoiningDate { get; set; }
}
```

Step 2: Place a controller class called StudentController in the **Controller** folder.

Step 3: Inside the StudentController, right-click the Index () method and from the context menu, select **Add View** option. This time, the view will be created as Razor page. The setting is shown in Figure 10.13.

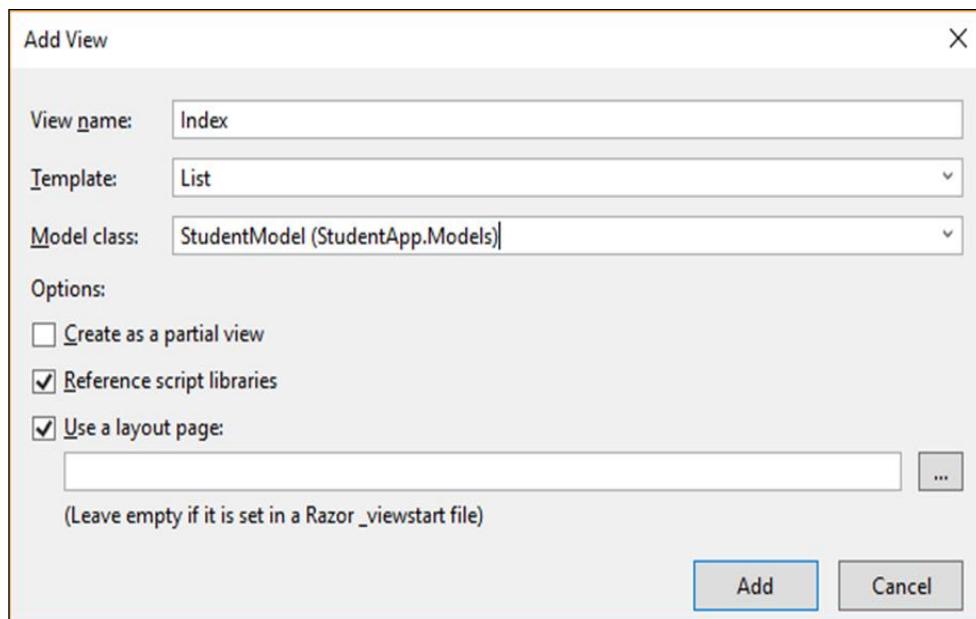


Figure 10.13: MVC View Scaffolding

Code Snippet 3 displays the auto-generated code in **Index.cshtml**.

Code Snippet 3:

```
@model IEnumerable<StudentApp.Models.StudentModel>
@{
    ViewBag.Title = "Index";
}
<h2>Index</h2>
<p>@Html.ActionLink("Create New", "Create")</p>
<table class="table">
<tr>
    <th>
        @Html.DisplayNameFor(model => model.StudentID)
    </th>
    <th>
        @Html.DisplayNameFor(model => model.StudentName)
    </th>
    <th>
        @Html.DisplayNameFor(model => model.StudentEmail)
    </th>
    <th>
        @Html.DisplayNameFor(model => model.JoiningDate)
    </th>
    <th>
        </th>
</tr>
@foreach (var item in Model) {
```

```

<tr>
    <td>
        @Html.DisplayFor(modelItem => item.StudentID)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.StudentName)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.StudentEmail)
    </td>
    <td>
        @Html.DisplayFor(modelItem => item.JoiningDate)
    </td>
    <td>
        @Html.ActionLink("Edit", "Edit", new { /* id=item.PrimaryKey */ }) | 
        @Html.ActionLink("Details", "Details", new { /* id=item.PrimaryKey */ }) | 
        @Html.ActionLink("Delete", "Delete", new { /* id=item.PrimaryKey */ })
    </td>
</tr>
}
</table>

```

In order to call the Web API application from MVC application, an additional reference named `WebApi.Client` is required. Use NuGet Package Manager to add this new DLL reference.

Follow these steps to add the reference:

Step 4: Using NuGet Package Manager, install `Microsoft.AspNet.WebApi.Client`.

Step 5: Go to the `StudentController` file and add the code to the `Index()` method, as shown in Code Snippet 4.

Code Snippet 4:

```

public ActionResult Index() { web
    HttpClient WebHttpClient = new HttpClient();
    WebHttpClient.BaseAddress = new
        Uri("http://studentservicesazureapp.azurewebsites.net/
api/");
    WebHttpClient.DefaultRequestHeaders.Clear();
    WebHttpClient.DefaultRequestHeaders.Accept.Add(new
        MediaTypeWithQualityHeaderValue("application/json"));
}

```

```

IEnumerable<Models.StudentModel> studentList;
HttpResponseMessage response =
WebHttpClient.GetAsync("Student").Result;
studentList =
response.Content.ReadAsAsync<IEnumerable<Models.StudentModel>>()
.Result;
return View(studentList);
}

```

Step 6: Run the application by pressing F5 and it can be seen that the browser is loaded with data from the database, which are actually loaded via Web API, as shown in Figure 10.14.

Note: In Code Snippet 4, the API path used is the hosted WebApp <http://studentservicesazureapp.azurewebsites.net/api/>.

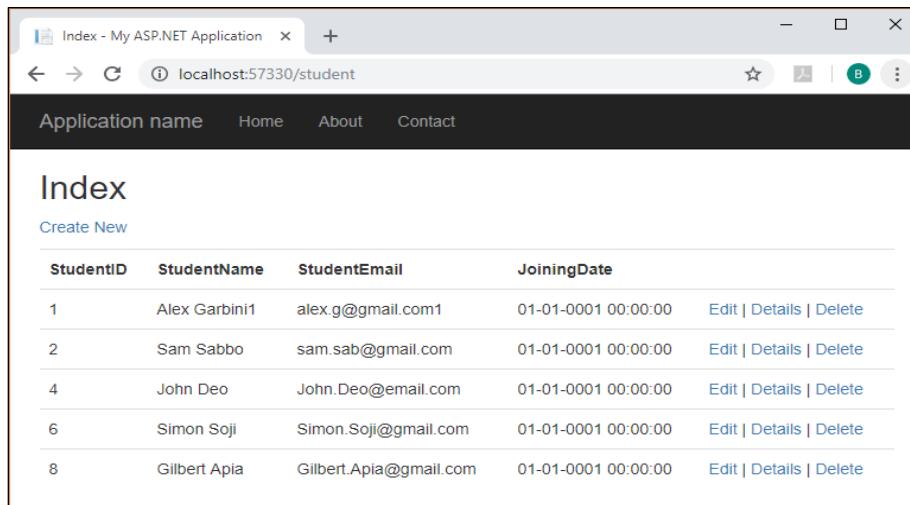


Figure 10.14: MVC Application Using HttpClient

10.4.2 Postman

After Web APIs are developed in real-world application development, they are tested using third-party tools. Postman is one of the most excellent tools available for API testing. Developers can visit <https://www.getpostman.com/apps> to download the program. After downloading and installing the Postman application, launch it.

Following steps illustrate how to test Web APIs with Postman:

Step 1: Paste the Web API URL (<http://localhost:59502/api/Student/GetAll>) into the Postman URL Tab. Figure 10.15 shows how to do this.

The screenshot shows the Postman application interface. In the top navigation bar, the URL `http://localhost:59502/api/Student/GetAll` is entered under the 'URL' field. The request method is set to 'GET'. Below the URL, there are tabs for 'Params', 'Authorization', 'Headers (8)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Body' tab is currently selected. A table under 'Query Params' shows one entry: 'Key' with 'Value'. On the right side of the interface, there is a 'Response' section featuring a cartoon character of an astronaut pointing at a star, with the text 'Click Send to get a response' below it. The bottom of the screen shows various toolbars and status indicators.

Figure 10.15: Web API Testing in Postman-Get Method

Step 2: Try creating a Student with the POST method. To do this, copy and paste the POST method into the Postman URL. In the body of the request, change the HTTP request to POST. For the Student to be made, the raw content, which is JSON data, has to be passed. Figure 10.16 shows the specifics.

The screenshot shows the Postman application interface. In the top navigation bar, the URL `http://localhost:59502/api/Student/GetAll` is entered under the 'URL' field. The request method is set to 'POST'. Below the URL, there are tabs for 'Params', 'Authorization', 'Headers (8)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Body' tab is currently selected. Under 'Body', the 'raw' radio button is selected, and the JSON content is:

```
[{"StudentName": "Simon Soji", "StudentEmail": "simon_soji@gmail.com", "StudentJoinDate": "2024-02-08"}]
```

. The 'JSON' tab is also visible. At the bottom of the 'Body' section, there is a preview of the response:

```
1 {  
2     "status": "success",  
3     "data": [  
4         {"StudentName": "Simon Soji",  
5          "StudentEmail": "simon_soji@gmail.com",  
6          "StudentJoinDate": "2024-02-08",  
7          "Id": 6148  
8     ],  
9     "message": "Successfully! Record has been added."  
10 }
```

 The status bar at the bottom indicates 'Status: 200 OK Time: 1033 ms Size: 578 B'. The bottom of the screen shows various toolbars and status indicators.

Figure 10.16: Web API Testing in Postman-Post Method

Step 3: Once all the information is set, click **Send**. This creates the student and displays the Success message.

Step 4: Use the method to call the Get method to see if the record was made. The history of recent calls is shown on the left of the window.

10.4.3 Swagger

Various methods associated with a Web API are interesting to a developer. Swagger is a language-agnostic specification used to describe REST APIs. Swagger helps in solving the problem of creating help pages as well as suitable documentation for Web APIs. Swagger is also referred to as OpenAPI as the Swagger project was donated to the OpenAPI Initiative. The name OpenAPI is preferred. Some benefits of Swagger are API discoverability, client SDK generation, and interactive documentation.

Swagger enables computers as well as humans to recognize the competencies of a service. It functions without any form of direct access to the implementation such as network access, source code, and documentation. It is aimed at minimizing the workload that is required for connecting disassociated services. At the same time, it is aimed at reducing the time required to precisely document a service.

Swagger specification is fundamental to Swagger flow. By default, this is a document of name swagger.json. The Swagger tool chain or its third-party implementations can generate it depending on the service. The document explains the proficiencies of the particular API and also how to access it using HTTP.

Swagger UI presents a Web-based UI, which offers details on the service by using the Swagger specification that is generated. Nswag and Swashbuckle both contain Swagger UI in an embedded version. Thus, a middleware registration call can be used to host it on the ASP.NET Core app.

10.5 Secure Hosted API

When making an app that gives users access to restricted resources, developers must know how to stop people from getting in without permission. Developers should use Azure Active Directory (Azure AD) to make this happen. Using OAuth 2.0 bearer access tokens is a simple and flexible way to protect a Web API with a few lines of code. Developers of ASP.NET Web applications can use the OWIN

middleware in the .NET Framework, which Microsoft builds with the help from the community.

Here is a list of steps that must be done to register the directory:

- Register both the API and a backend application with Azure Active Directory.
- Represent an API client and register another app (client-app) in Azure Active Directory.
- In Azure Active Directory, let client-app call backend-app by permitting it to do so.
- Set up OAuth 2.0 authorization in the Developer Console.
- To check if an OAuth token is valid, add the validate-jwt policy to all requests that come in.

10.6 Summary

- ✓ Visual Studio helps a lot with packaging and deploying Web services and applications.
- ✓ Publishing a Web API application to IIS on an Azure VM is one way to host it.
- ✓ In self-hosting, the Web API hosting is separate from the main application, also called the client.
- ✓ On the Kestrel server, you can also host Web API apps.
- ✓ With Azure Container Service, you can quickly set up a Kubernetes, DC/OS, or Docker Swarm cluster that is ready for production.
- ✓ Swagger is a language-independent description of REST APIs. Swagger is a tool that makes it easier to make help pages and good documentation for Web APIs.

10.7 Test Your Knowledge

1. What is the most important Security Rule for Remote Desktop Connection?
 - A. Inbound Security Rule
 - B. Outbound Security Rule
 - C. Incoming Security Rule
 - D. Both Inbound and Outbound Security Rules
2. Which publish setting is required in order to publish an Azure Web App?
 - A. Azure Virtual Machine
 - B. App Service Publish
 - C. IIS and FTP Public
 - D. Folder Publish
3. Which of these services can be managed by using Remote Desktop?
 - A. Azure Windows Virtual Machine
 - B. Internet Information Services (IIS)
 - C. Azure SQL Server
 - D. Both Azure Windows Virtual Machine and IIS
4. Which of the following can be done by the HTTP client?
 - A. HTTP GET
 - B. HTTP POST
 - C. HTTP PUT
 - D. All of these
5. What is Kestrel?
 - A. It is the default open-source Web server that is provided with ASP.NET Core
 - B. It is the Web protocol that is used with ASP. NET
 - C. It is a third-party API testing tool
 - D. None of these

10.7.1 Answers to Test Your Knowledge

1. A
2. B
3. D
4. D
5. A

Try It Yourself

- Install IIS on Windows Virtual Machine.
- Host a Web API on IIS.
- Host Web API in Microsoft Azure Web.
- Implement Web API with MVC application using HttpClient class.
- Test API with Postman.



SESSION 11

INTRODUCTION TO WCF SERVICES

Overview

This session introduces WCF services. The session explains how to create, deploy, and configure WCF services. The session also describes how to create and configure bindings for WCF services. It outlines the process of generating proxies using SvcUtil and by creating a Service Reference.

Learning Objectives

In this session, students will learn to:

- Explain WCF services
- Define the process of creating and deploying WCF services
- Explain the process of configuring WCF services
- Explain how to create and configure bindings for WCF services
- Explain how to integrate with the Azure service bus relay
- Explain the process to generate proxies using SvcUtil
- Illustrate how to generate proxies by creating a Service Reference
- Explain high-availability solutions for WCF services to ensure business continuity

11.1 Overview of WCF Services

When applications are being developed for client/server architecture, one of the most-suited technologies available in .NET is Windows Communication Foundation (WCF) framework. It is the most up-to-date communication infrastructure made available by Microsoft. It is also one of the best solutions for building distributed applications based on Service-Oriented Architecture (SOA). SOA emphasizes building software components (services) that are loosely

coupled, reusable, and independently deployable. WCF services encapsulate specific business functionalities and expose them to clients over the network.

WCF is a unified .NET Framework for building service-oriented applications. WCF was first introduced in .NET Framework 3.0 and then, extended in .NET 3.5 and .NET 4. WCF is the foundation for other Microsoft-distributed technologies.

WCF provides the same functionality as a Web service, but the only difference is Web services use HyperText Transfer Protocol (HTTP) for communication, whereas WCF can use any protocol for communication. So, it becomes easier to communicate with components of other languages. WCF promotes development of applications based on the principles of SOA.

The purpose of WCF services is to simplify the development of distributed applications by providing a flexible, extensible, and secure communication framework. WCF abstracts the complexities of communication protocols and infrastructure, allowing developers to focus on implementing business logic and building robust, interoperable services.

One of the main advantages of using WCF is it can also support proprietary protocols. WCF supports protocols and transports protocols such as SOAP with HTTP, TCP/IP, and Named pipes.

SOAP is a lightweight protocol used to exchange data over distributed environments. SOAP uses XML for its message formatting and usually relies on HTTP for message negotiation and transmission. It was created by Microsoft and adopted by W3C as a standard. SOAP is used as the underlying layer for ASP.NET Web services (ASMX) and for WCF. SOAP-based services have been used for more than a decade. WCF can exchange data and information using formats such as SOAP, XML, and JSON.

SOAP is now maintained by the W3C. It is also known as Remote Procedure Call (RPC) protocol. It was designed as a protocol specification for invoking methods on servers, services, and objects. It was developed as language independent way and multi-environment for exchanging structured data between services.

Figure 11.1 shows how a request is sent to a WCF service and how the service responds to the request.

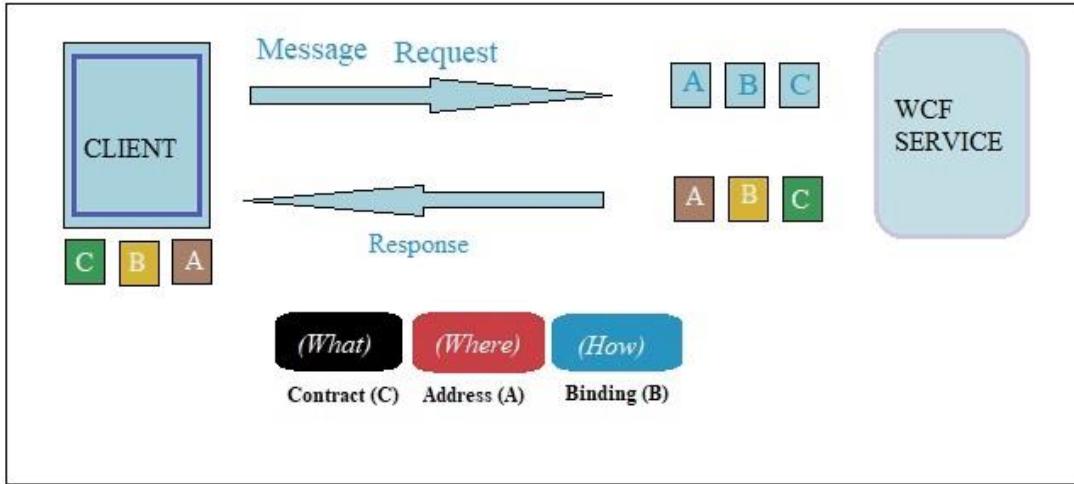


Figure 11.1: Request in WCF Service

11.1.1 ABC of WCF Services

As shown in Figure 11.1, a client sends a request to the service. The request goes as a message with one or more endpoints. A service endpoint defines how the service is exposed to the clients.

The endpoint comprises A-Address, B-Binding, and C-Contract, each of which are defined as follows:

Address	Binding	Contract
Specifies where the service resides. The address is a Uniform Resource Locator (URL) that is used by the client applications to locate the service.	Specifies how clients should communicate with the service. The binding specifies the message encoding, transport type, security modes, session support, and other protocols.	Specifies the operations supported by the endpoint. The contract must match one of the contract interfaces implemented by service class.

Let us explore these in some more detail along with more terms such as endpoints and hosting.

Bindings

- Define how communication occurs between the client and the service.
- They specify protocols, encoding formats, and other communication settings.
- WCF provides a variety of predefined bindings (for example, BasicHttpBinding or NetTcpBinding) suitable for different communication scenarios.

Contracts

- Contracts define the interface of a WCF service. Service contracts define the operations offered by the service.
- Data contracts define the data types exchanged between the client and the service.
- Message contracts define the structure of messages exchanged between the client and the service.

Endpoints

- Represent the connection points through which clients can access WCF services.
- Consists of an address, a binding, and a contract.
- WCF services can expose multiple endpoints to support different communication protocols and message formats.

Hosting

- WCF services can be hosted in different environments, including Internet Information Services (IIS), Windows services, self-hosted applications, and Windows Process Activation Service (WPAS).
- Hosting provides the infrastructure necessary for clients to discover and communicate with the service.

11.1.2 Features and Components of WCF

Some of the features provided by WCF include:

Security features such as authentication and authorization.

Performance tuning features such as throttling, concurrency, and load balancing.

Hosting environments such as IIS, Windows Services, and Self-hosting.

The WCF framework combines benefits of various .NET technologies such as Remoting, Web services, Enterprise services, WSE Standards, and Messaging services into a single unified model.

Three main components of WCF Service are:



A real-world analogy will help understand these better. Consider that a user wants to go to Boise, Idaho. What steps will the user require to take? The user may first decide on the travel mode and then, book tickets through a travel agent or at a railway station, airport, or sea port. With regards to WCF, in an endpoint, this real-life scenario can be mapped with following elements:

- **Binding** - This can be similar to determining the mode of travel that is, by air, train, or road. Similarly, with respect to endpoints, binding refers to determining the protocol to implement a service.
- **Address** - This can be related to the address of the travel agent or railway station or airport to book the ticket. Similarly, in endpoints, an address refers to the URL at which a service is offered. In WCF, user requests are serviced using a service class, which is predefined in the application. User can connect to these service classes using one or more endpoints. These services might be running on a remote machine or in some rare cases on the local machine.
- **Contract** - This can be related to the purpose of using the binding and address elements, that is, to reach Boise. Similarly, in endpoints, binding, and address fulfill the service requested by a client. Contracts define an important part of the architecture. The data contract consists of parameters that make up a message which a service can use. Message contracts use protocols such as SOAP, which help to acquire a finer control over the message. A service contract consists of declarations of methods that are used in them. These methods are declared in an interface, which can be created using programming languages such as Visual Basic or C#.

11.1.3 Policies and Bindings in WCF

Policies and bindings apply the conditions required to communicate with a service. For example, the binding specifies the transport protocol, such as HTTP or TCP, to be used for communication. Security requirements and other conditions are defined by policies.

There are various kinds of bindings that can be used based on requirements, which are as follows:



Endpoints are Universal Resource Identifiers (URI) which are exposed to the outside world. The client can connect to these WCF services from these endpoints. There are various hosts that are used to run a WCF service. The selection of a host entirely depends upon the kind of application.

Some of the popularly used hosts include:

Console Application	Can be a simple console application host.
Windows Service	WCF service can be controlled by the Service Control Manager.
IIS	WCF services can be hosted in IIS provided the service exposes at least one HTTP endpoint.
Microsoft Azure	WCF services can be hosted on Microsoft Azure.

11.1.4 Differences Between ASP.NET Web API and WCF

There are situations in which ASP.NET Web API does not provide a solution or in which WCF provides a better infrastructure. Table 11.1 lists the main differences between Web API and WCF.

Web API	WCF
It supports only HTTP and allows accessing from various mobile devices, browsers, and so on.	It supports transport protocols such as TCP, HTTP, UDP, and custom transports. It also allows switching between them.
It enables building Web APIs that support XML and JSON.	Binary, text, and Message Transmission Optimization Mechanism (MTOM) encoding is supported by WCF. MTOM is a method of efficiently sending binary data to and from Web services.

Web API	WCF
It uses basic protocol and formats such as XML, SSL HTTP, WebSockets, SSL, jQuery, and JSON.	It supports building services with WS-* standards such as message security, transactions, and reliable messaging.
It allows describing a Web API such as autogenerated HTML.	It allows describing WCF SOAP services in WSDL.
It ships with .NET Framework, but is also available as open-source.	It ships with the .NET Framework.

Table 11.1: Differences Between Web API and WCF

11.2 Creating and Deploying WCF Services

Before creating a service, the developer must answer following questions:

- What functionality will be provided by the service?
- Where is the service located?
- How can one send messages to the service and how to invoke it?

11.2.1 Creating WCF Services

Follow these steps to create a WCF service:

Step 1

Start Visual Studio 2022 IDE with a WCF Service Library project. In this case, name it as StudentWCFServices.

Step 2

Provide the path to store the application and click **OK**. Figure 11.2 represents the WCF project creation.

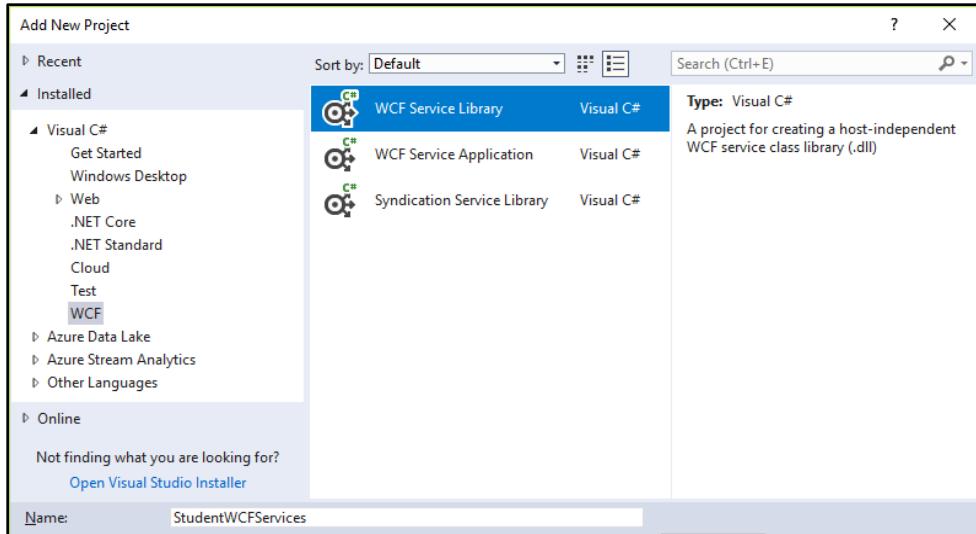


Figure 11.2: Creation of WCF Application

By default, the project creates a `Service1.cs` class and an `IService1.cs` interface. Figure 11.3 shows these in the Solution Explorer. Remove these default files.

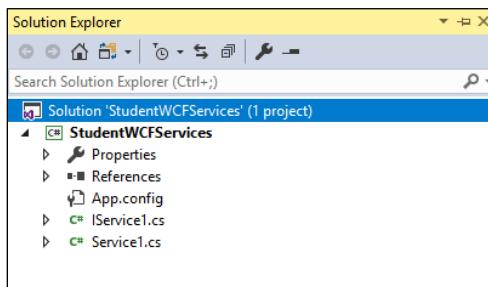


Figure 11.3: WCF Application in Solution Explorer

Step 3

Add a new folder **Models** and a Model class file named `studentModel` in the folder. Code Snippet 1 represents code to be added inside `studentModel` class. `System.Runtime.Serialization` namespace is used in the `using` section to make this model `Serializable`. The model class is marked as `DataContract` and each property is marked as `DataMember`.

Code Snippet 1:

```
using System;
using System.Runtime.Serialization;
namespace StudentWCFServices.Models {
```

```
[DataContract]
public class StudentModel {
    [DataMember]
    public int StudentID { get; set; }
    [DataMember]
    public string StudentName { get; set; }
    [DataMember]
    public string StudentEmail { get; set; }
    [DataMember]
    public DateTime? JoiningDate { get; set; }
}
}
```

Step 4

Right-click the project and add an interface `IStudentService.cs`. Add code to the interface, as shown in Code Snippet 2. Note that `System.ServiceModel` namespace is added in the using section, `ServiceContract` attribute is added at the interface level and `OperationContract` at the method level.

Code Snippet 2:

```
using StudentWCFServices.Models;
using System.Collections.Generic;
using System.ServiceModel;
namespace StudentWCFServices {
    [ServiceContract]
    public interface IStudentService {
        [OperationContract]
        IList<StudentModel> GetAll(double amount);
        [OperationContract]
        bool Create(StudentModel studentModel);
    }
}
```

Step 5

Add a new Data folder to the project, right-click it, and select **Add→New Item→ADO.NET Entity Model**. Now, follow the wizard instructions. This will create the `StudentDbEntities`.`DbContext` and add the connection string into the `App.config` file.

Step 6

Now, add a class file and name it as `StudentService.cs` to inherit from `IStudentService`.

This class will implement the two methods added to the interface. Code Snippet 3 represents implementation of StudentService class file.

Code Snippet 3:

```
using StudentWCFServices.Data;
using StudentWCFServices.Models;
using System.Collections.Generic;
using System.Linq;
namespace StudentWCFServices {
    public class StudentService : IStudentService {
        private readonly StudentDbEntities _context
            = new StudentDbEntities();
        public IList<StudentModel> GetAll(double amount) {
            IList<StudentModel> studentModels = new List<StudentModel>();
            var students = _context.Students.ToList();
            foreach (var student in students) {
                var studentModel = new StudentModel() {
                    StudentID = student.StudentID,
                    StudentName = student.StudentName,
                    StudentEmail = student.StudentEmail,
                    JoiningDate = student.JoiningDate
                };
                studentModels.Add(studentModel);
            }
            return studentModels;
        }
        public bool Create(StudentModel studentModel) {
            var student = new Student()
            {
                StudentName = studentModel.StudentName,
                StudentEmail = studentModel.StudentEmail,
                JoiningDate = studentModel.JoiningDate
            };
            _context.Students.Add(student);
            _context.SaveChanges();
            return true;
        }
    }
}
```

This Service class file implements the GetAll() and Create() methods of Student. Now, the WCF service is ready to run.

Step 7

Next step is to add the Binding contract into the App.config file. The Binding contract is given in Code Snippet 4. This code is added between the <system.serviceModel> and </system.serviceModel> tags. It is required to remove the Binding contract related to the default service added during the creation of an application.

Code Snippet 4:

```
<services>
<service name="StudentWCFServices.StudentService">
<host>
<baseAddresses>
<add
baseAddress="http://localhost:8733/Design_Time_Addresses/Studen
tWCFServices/StudentService/" />
</baseAddresses>
</host>
<!-- Service Endpoints -->
<!-- Unless fully qualified, address is relative to base
address supplied above -->
<endpoint address="" binding="basicHttpBinding"
contract="StudentWCFServices.IStudentService">
<!-- Upon deployment, following identity element should be
removed or replaced to reflect the identity under which the
deployed service runs. If removed, WCF will infer an
appropriate identity automatically. -->
<identity>
<dns value="localhost" />
</identity>
</endpoint>
<!-- Metadata Endpoints -->
<!-- The Metadata Exchange endpoint is used by the service to
describe itself to clients. -->
<!-- This endpoint does not use a secure binding and should be
secured or removed before deployment -->
<endpoint address="mex" binding="mexHttpBinding"
contract="IMetadataExchange" />
</service>
</services>
```

Step 8

Running the application will load the WCF Test Client which is by default embedded into Visual Studio for easy testing of WCF services. Figure 11.4 shows the WCF Test Client.

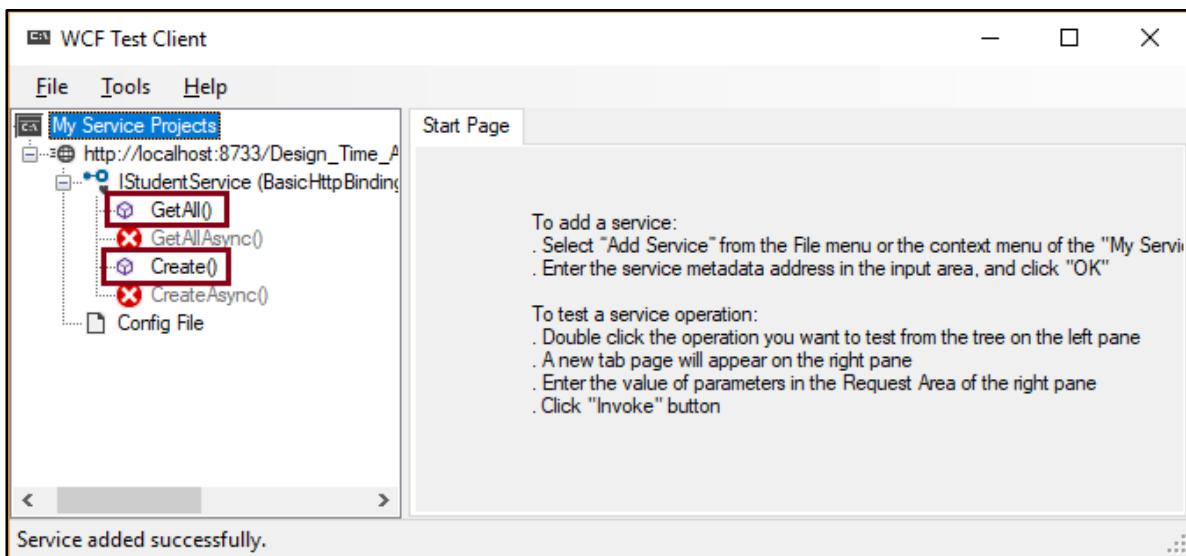


Figure 11.4: WCF Test Client

Step 9

Two service methods added in the service are listed to the left of the menu. Double-click GetAll() method. This will load the UI to execute the GetAll() method.

Step 10

Click **Invoke** to execute the method. Figure 11.5 shows the execution of GetAll() method.

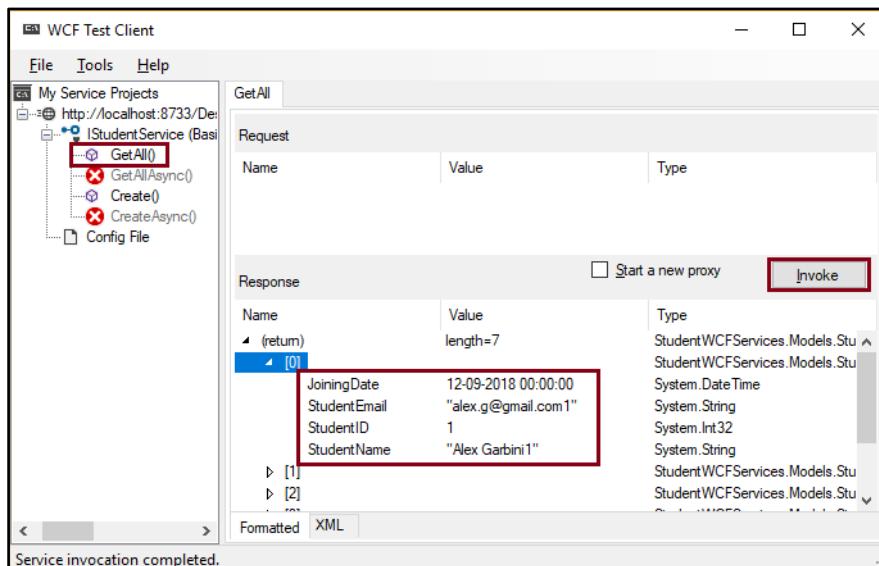


Figure 11.5: Execution of GetAll() Method

Step 11

Now, double-click `Create()` method. This will load a dynamic form to input data to the Student Model with all the student information.

Step 12

After providing all the data in the dynamic form, click **Invoke** to execute the method. Figure 11.6 shows the execution of `Create()` method.

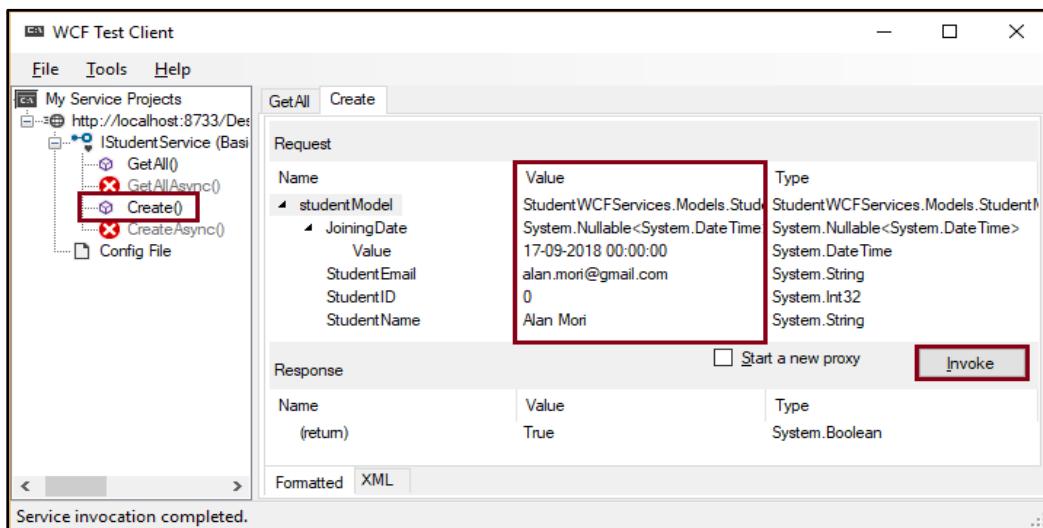


Figure 11.6: Execution of Create() Method

11.3 Configuring WCF Services

At the deployment point, the configuration file configured with WCF service gives the flexibility of providing endpoint and service behavior data.

.NET Framework configuration technology is used to configure a WCF service.

Adding XML elements to the `web.config` file for an IIS site that hosts a WCF service is very common. These elements make it possible to change details on a machine-by-machine basis. The changes can be endpoint addresses (the actual addresses used in communicating with the service). Along with this, WCF also includes various system-provided elements that provides a quick selection of the basic features for a service.

Endpoints are major infrastructure for communication in a WCF service. Endpoints provide clients access to the functionality offered by a WCF service.

Each endpoint consists of four properties, which are as follows:

- Address

- Binding
- Contract
- Behaviors

Basic usage for each of these in the configuration through web.config are as follows:

Address

```
<endpoint address="http://localhost/MyService"
binding="..." contract="..."/>
```

Here, in this example, the address specifies the location of the endpoint.

Binding

```
<endpoint address="http://localhost/MyService"
binding="wsHttpBinding" contract="..."/>
```

Here, binding wsHttpBinding specifies the binding policy to be used.

Contract

```
<endpoint address="http://localhost/MyService"
binding="wsHttpBinding" contract="IMinfo"/>
```

Here, the contract IMinfo specifies contract service that must be accessed by the host.

Web Services Description Language (WSDL) is used to define the Web service and Web service contract. It is an XML document, which contains custom tags used to define the elements and attributes of the service. WSDL also defines a standard manner in which a Web service can be accessed.

Defining the Contracts

Following syntax helps to create a ServiceContract and OperationContract in code:

```
[ServiceContract]
public interface <Name of interface>
{
}
[OperationContract]
void Operation1();
```

Syntax for New Bindings

Following syntax helps to create a new binding instance:

```
<specify binding type> binding = new <specify binding type>();
```

Following syntax helps to create endpoints in code:

```
Uri MyUri = new Uri("<specify address>");  
<specify binding type> binding = new <specify binding type>();  
ServiceHost.AddEndpoint(typeof(<specify the service interface>), binding, MyUri);
```

11.4 Creating and Configuring Bindings for WCF Services

To specify the communication details that are required to connect to the endpoints of a WCF service, objects called bindings are used. A well-defined binding is required for every endpoint in a WCF service. Developers must understand different types of communication details that are defined by the bindings, what kind of bindings are enclosed in WCF, and so on.

11.4.1 WCF Bindings

There can be basic or complex information in a binding. Only the transport protocol such as HTTP is defined in the most basic binding. This is used to connect to the endpoint. Most commonly, the information enclosed in a binding about how to connect to an endpoint is categorized into one of the following:

Protocols	Finds out the security mechanism that is employed: either the capability of reliable messaging or transaction context flow settings.
Encoding	Encodes the information in the message (for example, text or binary).
Transport	Finds out the underlying transport protocol to employ (for example, TCP or HTTP).

11.4.2 System-Provided WCF Bindings

Certain settings may not be compatible with others when there is complex information in a binding. This is the reason why WCF has a set of

system-provided bindings and the design of these bindings are such that they deal with the requirements of the application.

Some examples of system-provided bindings are as follows:

- **BasicHttpBinding:** This is appropriate for connecting to Web services that adapts to the WS-I Basic Profile description (for instance, ASP.NET Web services-based services).
- **WSHttpBinding:** This is a suitable interoperable binding for connecting to endpoints that adapts to the WS-* protocols.
- **NetMsmqBinding:** This binding uses the .NET Framework, to create connections between the queued message with other WCF endpoints.
- **NetTcpBinding:** This binding is suitable for using in a local network, offering higher performance than HTTP bindings.
- **NetNamedPipeBinding:** This connects to other WCF endpoints on the same machine, using the .NET Framework.

11.5 Integrating with the Azure Service Bus Relay

Azure Service Bus is a message broker that acts as a medium for messaging. It sits between many cloud app components or between the cloud and on-premises applications, enabling them to transmit information through messages. A Service Bus works on the basis of a multi-tenant concept. According to the multi-tenant concept, it is possible for the same service to be used by many users. Application developers will create a namespace which includes one or more specified communication mechanisms.

11.5.1 Azure Service Bus Relay

Enterprise applications often consist of various features in the form of components, services, and so on. They are often not simple or straightforward to work with. It is not an easy task to consolidate different components of an enterprise application into a single system even if all of them occupy the same location. It gets even more difficult if any of the components reside on a cloud.

Hybrid applications are enterprise applications based on Microsoft Azure. These applications include Web and worker roles, allowing data storage in SQL Database, and optionally, communicating with third-party service providers for authenticating or completing other tasks. It is possible that these applications may use some local components that are difficult to migrate to the cloud.

The Service Bus Relay is useful in building cloud hybrid applications that function in an Azure data center and on-premises environment. The Service Bus Relay is helpful in cloud hybrid applications. It considers WCF Web services already present and prepares them to access cloud-based solutions without disturbing security terms or infrastructure.

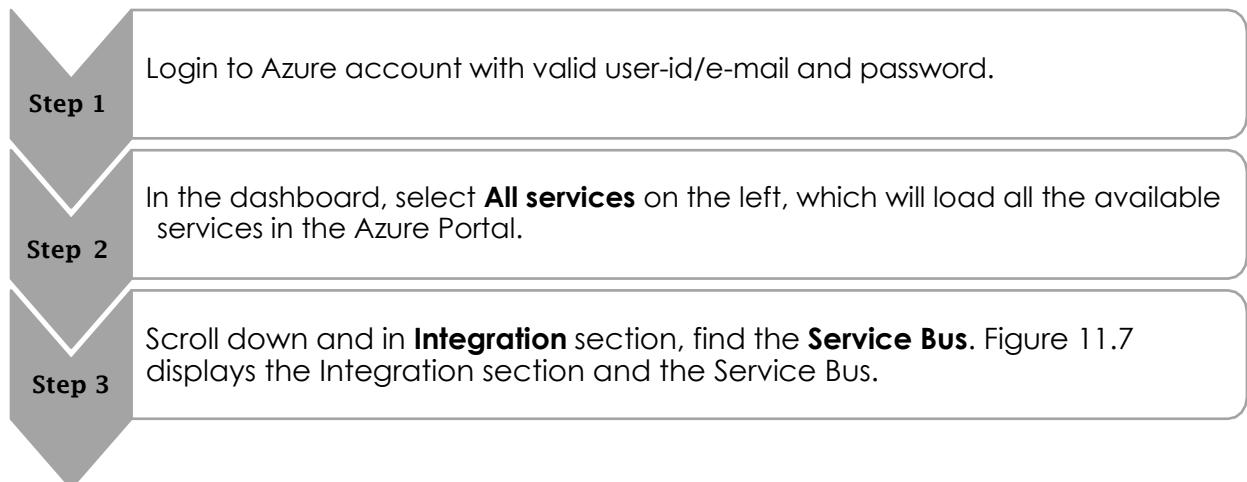
It is possible to host the WCF services using the Service Bus Relay in the existing enterprise environment. The Service Bus functioning in Azure can be assigned with the incoming requests and sessions that these WCF services receive. These services are then revealed to the application code executing in Azure or to mobile workers or extranet partner environments.

Two types of Messaging Capabilities by Service Bus include:

Brokered:	Relayed:
This offers subscriptions, topics, and queues that are factors of asynchronous messaging.	This offers response/request messaging, peer-to-peer messaging, and direct one-way messaging.

11.5.2 Creating Azure Service Bus

Following steps are performed to create a Service Bus in Azure:



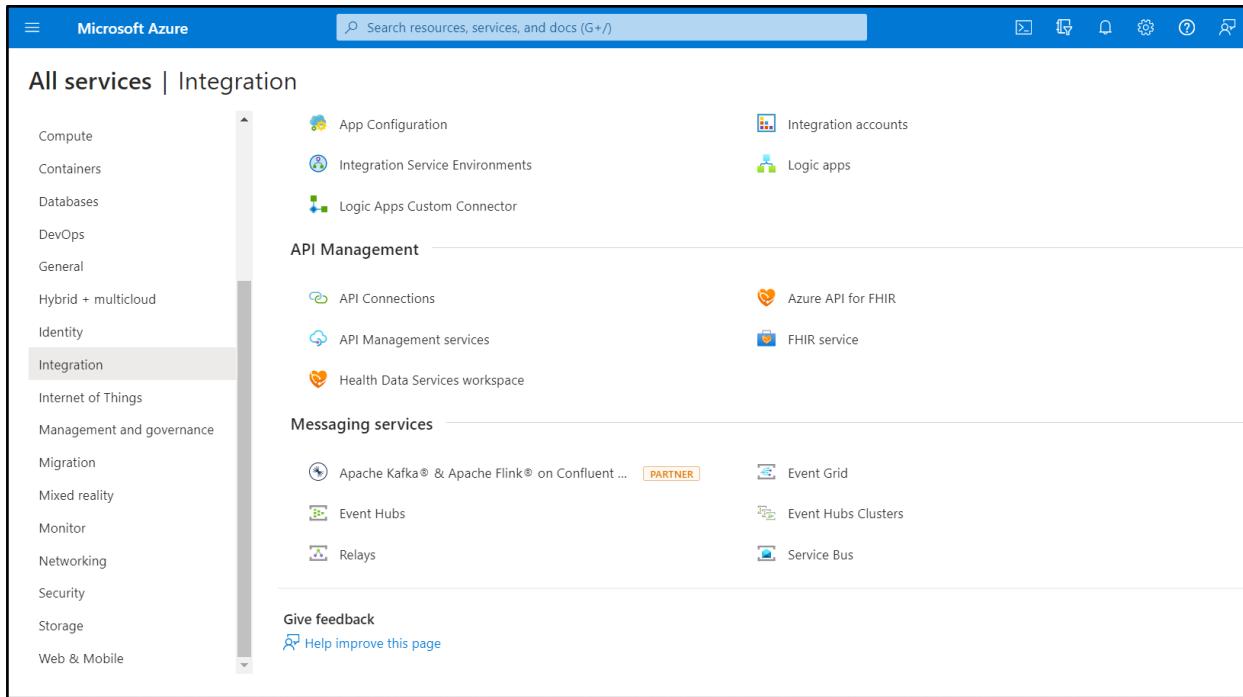


Figure 11.7: Integration Section and Service Bus

Step 4

If the user is creating Service Bus for the first time, the page does not have any Service Bus page empty. Click **+Add** link or **Create Service Bus namespace** button to create a new Service Bus. Figure 11.8 displays the Service Bus page.

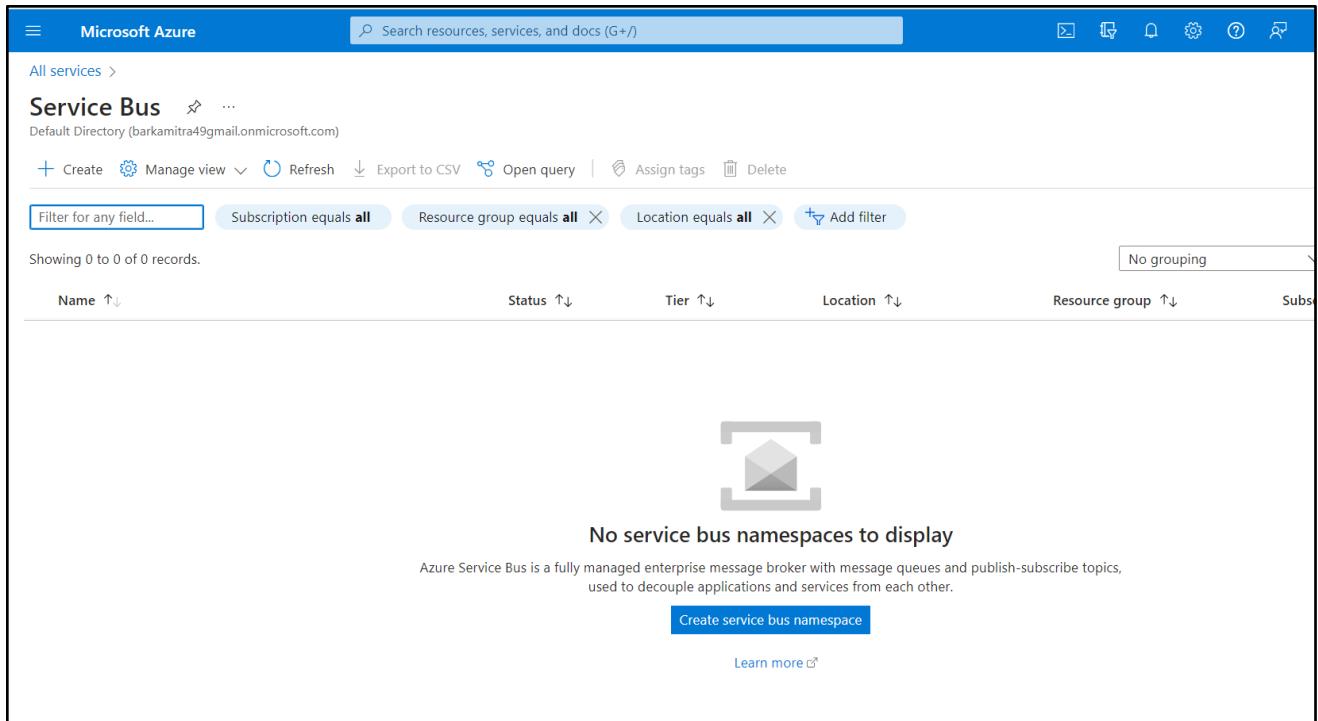
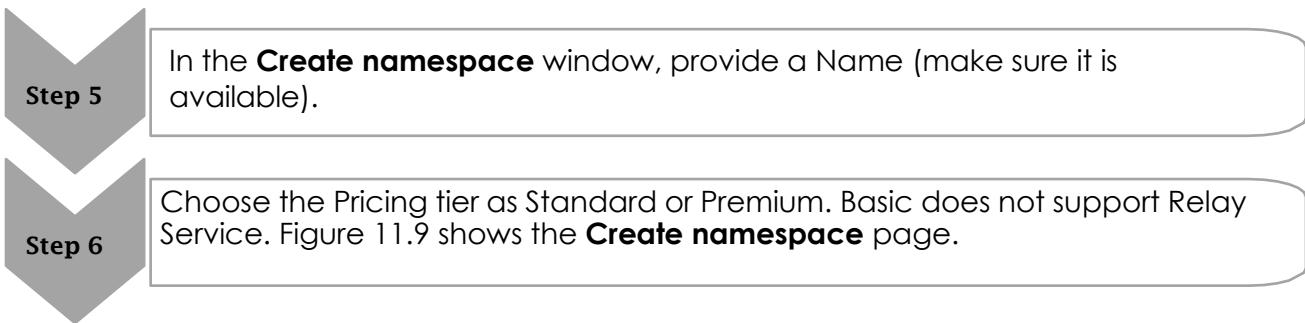


Figure 11.8: Service Bus Page



The screenshot shows the 'Create namespace' page in the Microsoft Azure portal. At the top, there's a blue header bar with the Microsoft Azure logo and a search bar. Below the header, the URL 'All services > Service Bus >' is visible. The main title is 'Create namespace' with a 'Service Bus' icon. Below the title, there are tabs: 'Basics' (which is selected), 'Advanced', 'Networking', 'Tags', and 'Review + create'. The 'Project Details' section asks to select a subscription to manage resources. A dropdown menu shows 'Azure for Students' is selected. The 'Instance Details' section requires entering a namespace name, location, and pricing tier. The namespace name 'studentwcf' is entered, resulting in the full URL 'studentwcf.servicebus.windows.net'. The location is set to 'Australia Central' and the pricing tier is 'Standard (~\$10 USD per 12.5M Operations per Month)'. Navigation buttons at the bottom include 'Review + create', '< Previous', and 'Next: Advanced >'.

Figure 11.9: Create Namespace Page

- Step 7** Once the Namespace is created, user is automatically redirected to the Service Bus List page and will get a notification. Refresh the list page to see the created Service Bus.
- Step 8** After completion of entering all the information, click **Create**.
- Step 9** From the List page, click the newly created namespace, which will load the Namespace property page.
- Step 10** On the left of the namespace page, click **Shared access policies**. By default, a shared access policy should be created under it. Select the default access policy to see the attributes. Figure 11.10 shows the Shared access policies page.

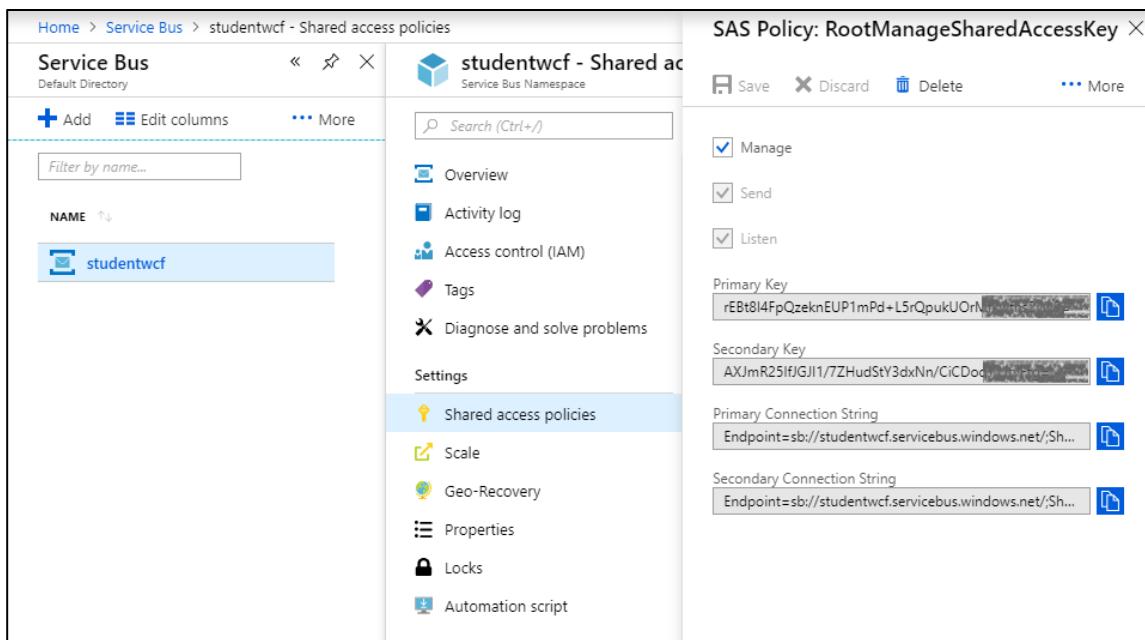


Figure 11.10: Shared Access Policies

These keys and connection strings are an important part of the Service Bus. This can later be used in Key and Connection strings in WCF Application. Copy the Keys and Connection Strings and store them in a safe place.

11.5.3 Integration of Azure Service Bus Relay in WCF

To integrate Azure Service Bus Relay, follow these steps:

Step 1

Open WCF application and install the Service Bus NuGet package named Microsoft Azure Service Bus.

Step 2

Search for Service Bus and select the Microsoft Azure Service Bus item.

Step 3

Open the App.config file and add a new <endpoint> inside the <service> tag.

Code Snippet 5 shows the endpoint declaration.

Code Snippet 5:

```
<endpoint contract="StudentWCFservices.IStudentService">
    binding="netTcpRelayBinding"
    address="sb://studentwcf.servicebus.windows.net/solver"
```

```
behaviorConfiguration="sbTokenProvider"/>
```

Step 4

Add the `<endpointBehaviors>` inside `<behaviors>`. Code Snippet 6 shows the `endpointBehavior` declaration.

Code Snippet 6:

```
<endpointBehaviors>
<behavior name="sbTokenProvider">
<transportClientEndpointBehavior>
<tokenProvider>
<sharedAccessSignaturekeyName="RootManageSharedAccessKey"
key="rEBt8I4FpQzeknEUP1mPd+L5rQpukUxxxxxxxxxxxxxx" />
</tokenProvider>
</transportClientEndpointBehavior>
</behavior>
</endpointBehaviors>
```

11.6 Generating Proxies Using SvcUtil

Using `Svcutil.exe`, one can download metadata from the functioning services and save them to the local files. `Svcutil.exe` makes an effort to recover metadata with the help of WS-MetadataExchange and XML Web Service Directory for HTTP and HTTPS URL schemes and it uses only WS-MetadataExchange to retrieve metadata for all other URL schemes.

One can find the `ServiceModel Metadata Utility Tool`, specifically under `Bin` in the local Windows SDK installation location.

`Svcutil.exe` utilizes the bindings specified in the `MetadataExchangeBindings` class by default. One must specify a client endpoint in the configuration file for `Svcutil.exe` (`svcutil.exe.config`) while configuring the binding utilized for WS-MetadataExchange. The client endpoint should use the `IMetadataExchange` contract and should have the same name as that of Uniform Resource Identifier (URI) scheme of the metadata endpoint address.

`Svcutil.exe` generates a `BasicHttpBinding` for a service with a `BasicHttpContextbinding` endpoint with the attribute `allowCookies` set to true. These cookies are utilized for context on the server. When the service uses these cookies, one can alter the configuration manually to use a context binding to manage context on the client.

Code Snippet 7 represents the syntax to use `svutil`.

Code Snippet 7:

```
svutil.exe [ /t:code ] <metadataDocumentPath>* |<url>* |<epr>
```

Code Snippet 8 represents a sample command using `svutil`.

Code Snippet 8:

```
svutil.exe /t:metadata  
http://localhost:8733/Design_Time_Addresses/StudentWCFServices/S  
tudentService/mex /d:E:\ StudentServiceMeta
```

11.7 Generating Proxies by Creating a Service Reference

Communication between a client and a Web service happens using SOAP messages. These enclose the input and output parameters as XML. SOAP messages are sent over a network after a proxy class maps parameters to XML elements. This is how a proxy class releases one from communicating with the Web service at the SOAP level and makes it possible to invoke Web service methods in a development environment that supports SOAP and Web service proxies.

Following steps are performed to generate WCF client proxy within Visual Studio 2022, using **Add Service Reference** feature (Service must be running to generate the WCF client proxy):

Step 1

Right-click the project containing the WCF client proxy when the service is running and choose **Add Service Reference**.

Step 2

Type in the URL to the service that is required to call, in the **Add Service Reference** Dialog box and click **Go**. This displays a list of services that are available at the specified address in the dialog.

Step 3

To view the contracts and operations available, double-click the service, assign a namespace for the generated code, and then, click **OK**.

11.8 High Availability Solutions for WCF Services

High Availability (HA) is a system's capability to provide services to end users without going down for a specified period of time. High availability minimizes or (ideally) eliminates service downtime regardless of what incident the company runs into (a power outage, hardware failure, unresponsive apps, lost connection with the cloud provider, and so on).

In WCF, high-availability refers to the capacity to sustain service accessibility and functionality despite various faults or disturbances, minimizing downtime and ensuring uninterrupted service for clients.

Some high-availability solutions and their purpose in WCF are as follows:

Fault-Tolerance

WCF services must be fault-tolerant, meaning they can handle and recover gracefully from hardware failures, software errors, network issues, or unexpected exceptions. Techniques such as retry policies, circuit breakers, and graceful degradation aid in achieving fault tolerance.

Redundancy

Redundancy is crucial for high availability. It involves deploying multiple instances of WCF services across distributed environments such as servers or data centers. If one instance fails, others can seamlessly take over, minimizing disruptions.

Load Balancing

Load balancing distributes incoming client requests across multiple instances of WCF services to optimize resource utilization and prevent overload. Load balancers monitor instance health and route traffic accordingly, avoiding unhealthy or overloaded instances.

Failover Mechanisms

Failover mechanisms detect and respond to failures by redirecting client requests from failed or unhealthy instances to healthy ones. This may involve automatic rerouting via load balancers or clustering and replication techniques.

Statelessness

Designing WCF services to be stateless reduces reliance on specific instances. Stateless services do not retain client state between requests, allowing any instance to handle incoming requests without session affinity or sticky sessions.

Monitoring and Alerting

Effective monitoring and alerting systems continuously monitor service health and performance, alerting administrators to potential issues before they disrupt service availability. Proactive monitoring enables quick intervention and problem resolution.

Scalability

Scalability is key for handling increasing demand while maintaining performance. Horizontal scaling adds more instances, while vertical scaling increases resources. Both approaches ensure continuous service availability under heavy loads.

11.9 Summary

- ✓ WCF is a unified .NET Framework for building service-oriented applications.
- ✓ WCF simplifies building distributed applications through a unified programming model, defining contracts, bindings, endpoints, and supporting various hosting environments.
- ✓ A service endpoint defines how the service is exposed to the clients. The endpoint of WCF is called as ABC, which stands for A-Address, B-Binding, and C-Contract.
- ✓ Among various kinds of bindings that can be used in WCF, major bindings are BasicHttpBinding, NetTcpBinding, NetMsmqBinding, and WSHttpBinding.
- ✓ Developers can use WCF Test Client which is a default, embedded into Visual Studio for easy testing of the WCF Services.
- ✓ WCF client proxy can be generated manually by using the Servicemodel Metadata Utility Tool (SvcUtil.exe).
- ✓ WCF client proxy can also be generated within Visual Studio using the Add Service Reference feature.
- ✓ High availability solutions for WCF services ensure uninterrupted service through fault tolerance, redundancy, load balancing, failover, statelessness, monitoring, and scalability.

11.10 Test Your Knowledge

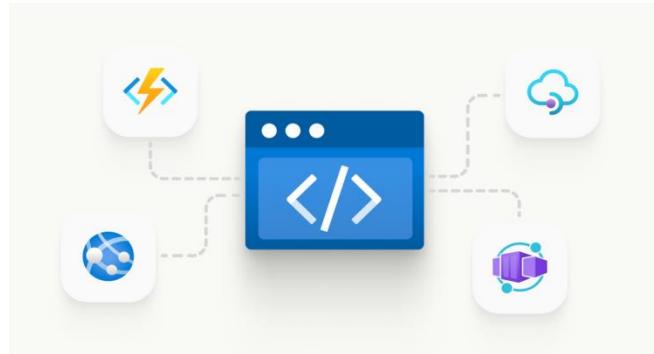
1. What does ABC stand for with respect to WCF?
 - A. Address, Behavior, and Contract
 - B. Address, Binding, and Controller
 - C. Address, Binding, and Contract
 - D. None of these
2. Which host can be used to run a WCF service?
 - A. Console Application
 - B. Microsoft Azure
 - C. Windows Service
 - D. All of these
3. What is the possible way to generate WCF client proxy?
 - A. By using SvcUtil.exe
 - B. By using Service Reference
 - C. By using both SvcUtil.exe and Service Reference
 - D. None of these
4. What is not true about WCF services?
 - A. WCF supports protocols such as TCP, HTTP, UDP, and custom transports
 - B. WCF is the replacement of Web API
 - C. It allows describing WCF SOAP services in WSDL
 - D. WCF Service allows messaging through SOAP
5. Identify the correct syntax for binding in a WCF service.
 - A. <endpoint binding=http://localhost/MyService
binding="wsHttpBinding" contract="..."/>
 - B. <binding
address="http://localhost/MyService" binding="wsHttpBinding"
contract="..."/>
 - C. <endpoint=http://localhost/MyService binding="wsHttpBinding"
contract="..."/>
 - D. <endpoint address=http://localhost/MyService
binding="wsHttpBinding" contract="..."/>

11.10.1 Answers to Test Your Knowledge

1. B
2. D
3. C
4. B
5. C

Try It Yourself

- Experiment with different error handling techniques in your WCF service, such as fault contracts, global error handlers, and exception shielding.
- Create a WCF service that uses message contracts to define the structure of SOAP messages exchanged between clients and the service.



SESSION 12

IMPLEMENTING AZURE SERVERLESS WEB SERVICES AND API MANAGEMENT

Overview

This session presents an overview on creating, designing, and deploying logic apps for automating processes in businesses. It explains the implementation of Azure Functions, WebJobs, and fabric-based Web services. It deals with security enhancement and throws light on implementing policies as well as Swagger services.

Learning Objectives

In this session, students will learn to:

- Define serverless computing
- Explain Azure Logic Apps, Azure Functions, and Webjobs
- Explain Fabric-based Web services
- Identify different ways to secure Web Services
- Define and implement Azure policies
- Define an API Interface
- Explain Swagger managed services
- Explain the foundational principles of API design
- Explain the concept of event-driven architectures
- Implement utilization of Azure Functions in event driven architectures

12.1 Working with Azure Logic Apps and API Apps

Serverless computing in Microsoft Azure offers abstraction of servers, infrastructure, and operating systems for developers. With serverless computing, developers eliminate the overhead of provisioning and managing servers, hence they can focus on their app development. Some of the products and services available

under serverless computing include Azure Logic Apps, Azure Functions, and so on.

12.1.1 What is Azure Logic Apps?

Azure Logic Apps represents a cloud service aimed at simplifying the process of creating automated scalable workflows. A series of steps or tasks is known as a workflow. Azure Logic apps provides a visual designer to model and automate the process as a workflow. It can incorporate data and apps through on-premises systems and cloud services.

Logic apps help in simplifying design process and creating scalable solutions useful for data integration, app integration, Enterprise Application Integration (EAI), and Business-to-Business (B2B) communication. It can be carried out in the cloud or on-premises, or if required, on both.

Logic apps can be used for automation of workloads such as:

Managing and routing orders through cloud services and on-premises systems.

Notifying events that happen in apps, systems, and services by e-mail using Office 365.

Monitoring when records are created or changed in Salesforce.

Developers can select from several services in order to build enterprise integration solutions with Azure logic apps. Some of these services include Azure Service Bus, functions, and storage; Office 365, Salesforce, Dynamics, SQL Server, Oracle, file shares, and so on.

Each connector provides a set of operations that may be categorized as 'actions' and 'triggers'. To create logic apps that would safely retrieve and manage data in real time, connectors offer actions, triggers, or both. Thus, connectors facilitate users to connect their accounts and make use of a set of ready-made **actions** and **triggers** to build their apps and workflows, thereby increasing productivity.

Actions

Actions are changes that are driven by a user. Consider that a developer wants to delete data in a SQL database. To do this, the developer can use an action. Actions directly map to operations.

Triggers

Certain connectors have a trigger. This means that an event from the connector will fire a logic app and pass in any data as part of the trigger.

Every logic app starts with a trigger, which begins the logic app workflow and passes some data as part of that trigger. In other words, a trigger is always the first step in a logic app. Several connectors provide triggers that can notify the app when specific events occur. Some examples of common triggers are as follows:

Recurrence of a task - run every hour

- When an HTTP request is received
- When an item is added to a queue
- When an email is received

When an event is received, it causes the logic app to run and actions in the workflow are executed. Developers can also access data from the trigger throughout the workflow.

12.1.2 Services in Azure Logic Apps

Logic apps and the Enterprise Integration Pack (EIP) offer following features that can be used for implementing and scaling up to advanced integration scenarios:

Processing XML messages

Building based on following products and services:

Azure Service Bus

Azure Functions

Azure API management

Exchanging messages with EDIFACT, AS2, and X12 protocols

Processing flat files

12.1.3 Custom API Apps

Connectors are primarily Web APIs, which utilize JSON for data exchange format, swagger metadata format for documentation, and REST for pluggable interfaces. Connectors are REST APIs; they correspond using HTTP endpoints. Thus, to build connectors, any language such as Java, .NET, or Node.js can be used.

Azure Logic apps provide over 100+ built-in connectors that can be used in logic app workflows. However, there may arise scenarios where developers must call APIs and services which are not available as connectors. In such cases, custom API apps can be created for use in logic apps.

Following are some reasons why developers may require to create custom APIs to call from logic app workflows:

To expand present data integration and system integration workflows

To help consumers use the service for handling specialized and particular tasks

To develop scope and use for service

It is possible to host these customized APIs on Azure App service, which represents a Platform-As-A-Service (PaaS) offering a highly scalable hosting. It also offers one of the simplest and most remarkable method for API hosting. To handle the working of custom APIs with logic apps, the customized API can specify actions for carrying out particular tasks in logic app workflows. Also, the customized API can work as a trigger; it can begin a logic app workflow if an event or new data satisfies a particular condition.

12.2 Designing and Implementing Azure Functions and WebJobs

Azure Functions is a core component of serverless computing in Azure. It is a serverless compute service that helps developers to run code on-demand without requiring to provision or manage infrastructure.

Azure Functions are helpful to run small pieces of code in an easy manner on the cloud. Developers can write specific code to solve a particular problem and do not have to concern themselves regarding infrastructure required for running it.

By adding Functions, developers can enhance productivity of the developed code. The choice of language is up to the developer.

For example, developers can choose Java, Node.js, F#, C#, or PHP. A developer can pay just for the time for which the code runs and allow Azure to manage scalability.

Another feature of Azure platform that works similar to Functions is WebJobs. The WebJobs feature of Azure App service allows running a script or programs in a context same as a Web app, mobile app, or an App service Web app.

WebJobs enable developers to run programs or scripts in their Website as background processes. It runs and scales as part of Azure Websites. The WebJobs SDK framework has been devised for WebJobs. It makes it easier to generate code that has been written as response to events in Azure services. For instance, creation of a thumbnail image could be a response to create an image blob in Azure storage. The WebJobs SDK can be deployed to a WebJob, as it runs in the

form of a .NET console application.

Ideally, WebJobs and WebJobs SDK should work together; however, they can work separately as well. A WebJob is capable of running any script or program that is running in the App service sandbox. Similarly, a WebJobs SDK console application is capable of running wherever console applications are running, for example, on-premises servers.

12.2.1 Creating Azure Functions

Following steps specify how to use Azure Functions to create a serverless function that runs based on a defined schedule:

Step 1

Log on to the Azure Portal and open Desktop.

Step 2

Click **+ Create a resource** from the top corner of the left menu. This will load the Marketplace menu.

Step 3

Select **Compute** from the Marketplace menu and Function App from the sub menu. Figure 12.1 shows the option to create the Function app.

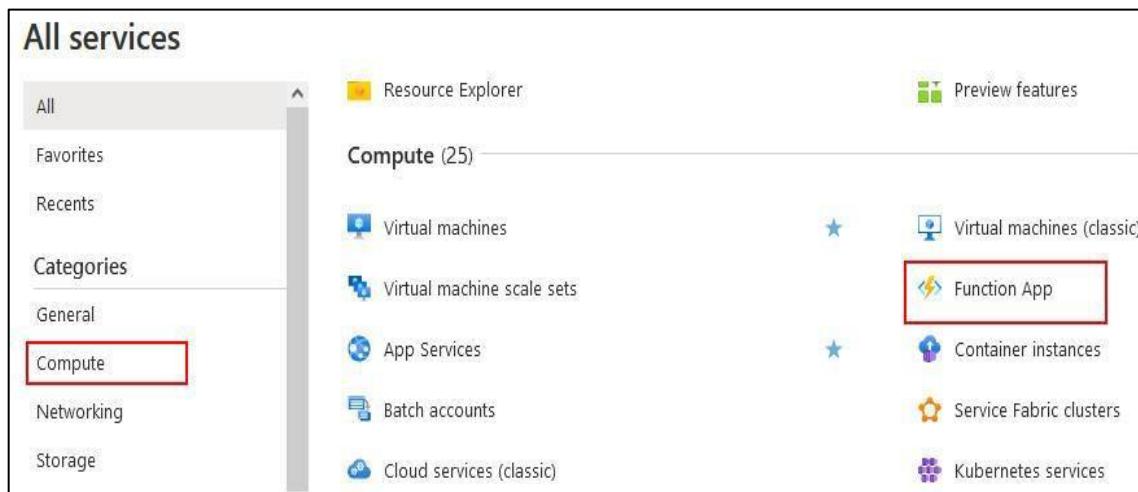


Figure 12.1: Option to Create the Function App

Step 4

Add the properties, as shown in Figure 12.2. Create an Azure Function that will validate first name and last name to convert it into a full name. Once all required information is provided, click **Create**.

The screenshot shows the 'Create Function App' step of the Azure portal. It includes fields for Resource Group (TestResource), Function App name (funcfullname), Publish (Code selected), Runtime stack (.NET), Version (6), Region (Central US), and Operating system (Windows selected). The 'Review + create' button is at the bottom.

Figure 12.2: Azure Function Properties

Step 5

After the function is created successfully, visit **All Resources** from the left menu, locate the function, and click to view the function dashboard. Figure 12.3 shows the default setting.

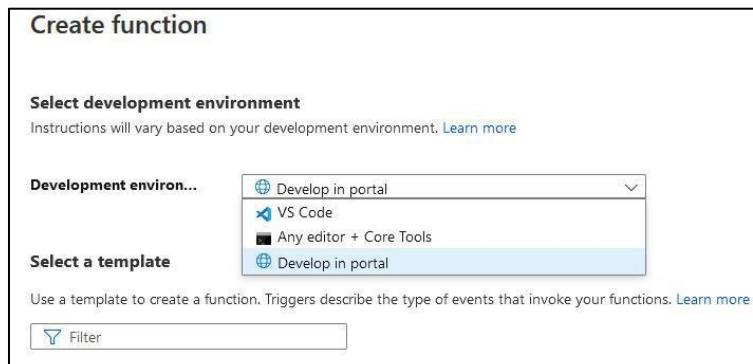


Figure 12.3: Function Settings

Step 6

In the function dashboard, from the left menu, click **Functions +** link to create a new function.

Step 7

Select Visual Studio Code to implement the code in VS Code. In this case, developers will have to install appropriate extensions to use Azure Functions with Visual Studio Code.

<https://marketplace.visualstudio.com/items?itemName=ms-azuretools.vscode-azurefunctions>

Alternatively, developers can use Visual Studio 2022 IDE to use templates to create Azure functions. Refer to Figure 12.4.

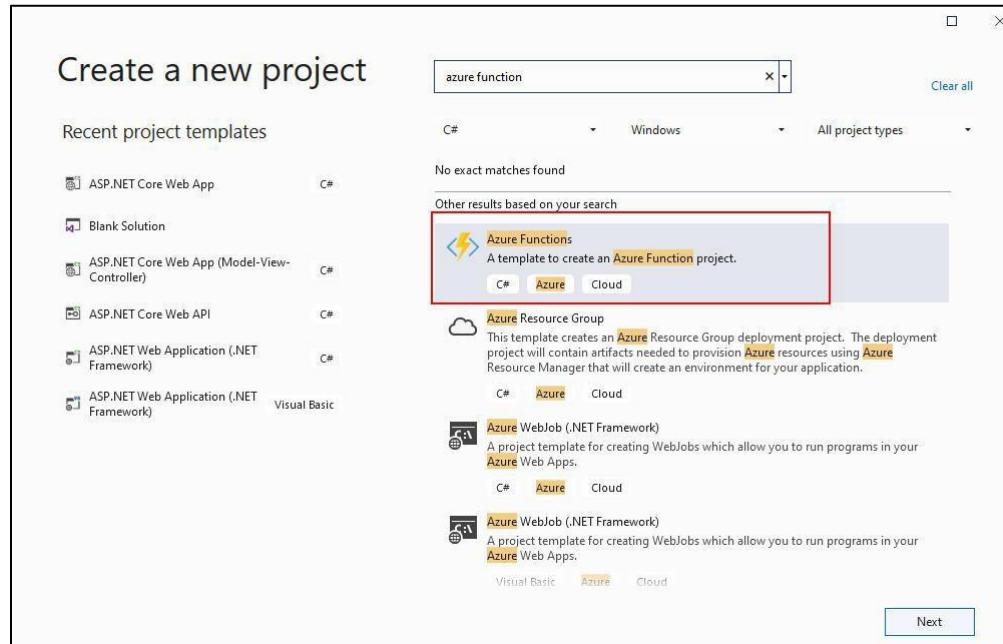


Figure 12.4: Visual Studio IDE Template

Figures 12.5 and 12.6 depict the steps in the Wizard that help to create the function in Visual Studio IDE.

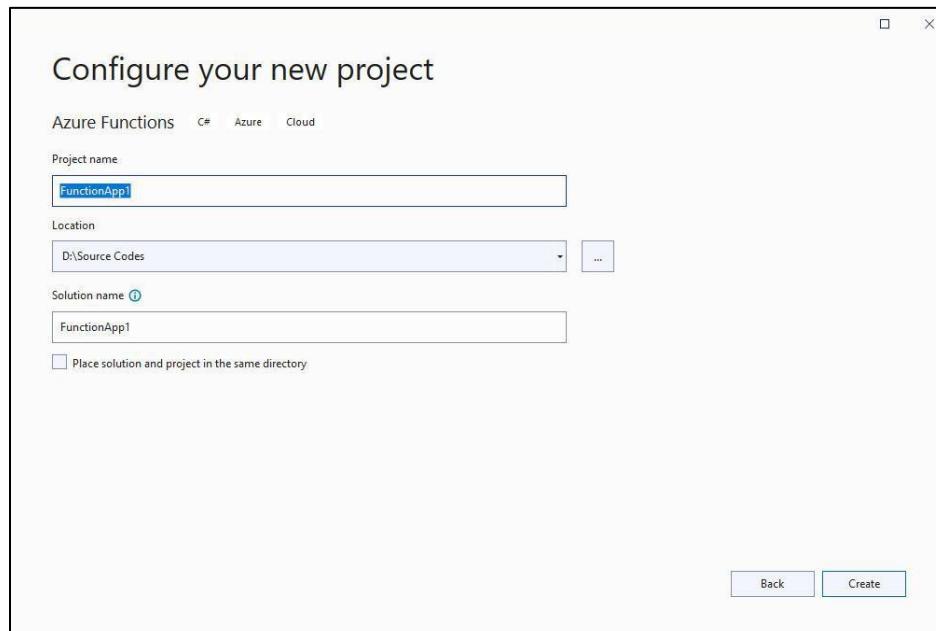


Figure 12.5: Configuring New Function

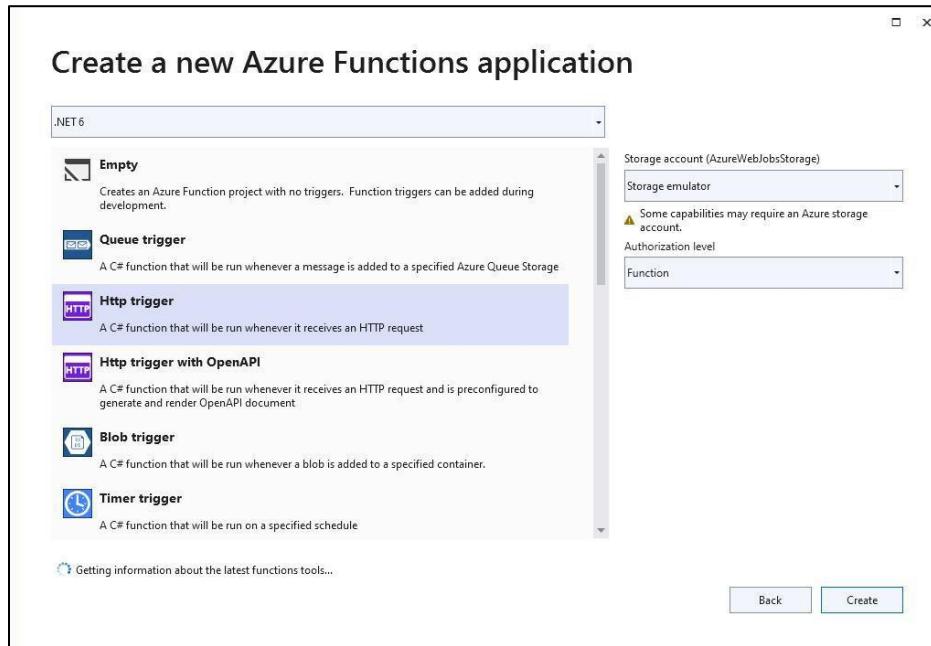


Figure 12.6: Selecting Trigger Type

Step 10

Click **Create**.

Thus, by following these step-by-step instructions, developers can create and implement their first Azure Function.

12.3 Designing and Implementing Service Fabric-based Web Services

Azure service Fabric presents a systems platform that is distributed. It simplifies packaging, deploying, and managing of reliable and scalable containers and microservices. In addition, Service Fabric focuses on important issues that are faced in the development and management of native cloud applications.

Issues related to complicated infrastructure can be avoided with the help of Azure Service Fabric. By doing so, developers as well as administrators can concentrate on implementation of workloads with high demand and which are mission-critical, and make them dependable, scalable, and easy to handle. The Service Fabric presents as a platform of future generation to build and handle tier-1, enterprise-class, and cloud-scale apps running in containers.

12.3.1 Microservices Applications

Service Fabric helps in building and managing applications that are dependable and scalable. These apps comprise microservices running at high density on machines that belong to a shared pool. This is also known as a cluster.

Service Fabric offers a high-level and lightweight runtime, which can be used for creating scalable, stateless, distributed, and stateful microservices that run in containers. In addition, it offers a detailed application management ability for providing, monitoring, deploying, upgrading, or patching, and deleting deployed applications, which also includes containerized services.

There are various Microsoft services that are powered by Service Fabric. Some examples are Azure SQL, Microsoft Power BI, Azure Cosmos DB, Azure IoT Hub, Azure Event hubs, Skype for business, as well as several core Azure services. Service Fabric is designed for building cloud native services, which start small, and when required, grow to a considerable scale that can accommodate a large number of machines.

12.3.2 Container Service and Orchestration

In Service Fabric, which acts as Microsoft's container organizer, microservices are deployed within machines in a cluster. There are several ways to develop microservices with the use of ASP.NET Core and Service Fabric programming models. Essentially, two services can be mixed as services and processes for the same application containers. As a container organizer, Service Fabric is the best choice when a requirement involves only deploying and managing containers.

12.3.3 State Management of Microservices for Service Fabric

Applications can be created using Service Fabric, which include containers or microservices. There are stateless microservices; these services cannot have a mutable state outside of a request as well as its response. An example is Web proxies and protocol gateways, and Azure Cloud Services worker roles.

On the other hand, stateful microservices can have an authoritative and mutable state outside of the request and in the response from the service. An example is databases, user accounts, devices, queues, and shopping carts. Of late, various applications of the Internet make use of a combination of stateful and stateless microservices.

Stateful services can be created using containerized stateful services or built-in programming models. The main focus of Service Fabric is to do so.

12.4 Secure Web Services Using Certificates, Azure Active Directory, and OAuth

To protect Web apps, API apps, mobile app back end, and Azure functions, the Azure App service can be used. In addition, the built-in App service features can be used for providing additional security. App service includes platform components, such as Azure VMs, network connections, storage, Web frameworks, and integration and management features. These components have strong and active security. On a regular basis, the App Service is thoroughly verified for conformity.

12.4.1 Securing Using Certificates

For an application on Azure Website that requires a certificate, developers can upload their certificate on the Azure Website in the certificates collection. It can then, be utilized in the Web application from the personal certificate store of their site.

12.4.2 Azure Active Directory

When creating an application for giving access to resources that are protected, developers should be aware of methods to avoid unwarranted access. For this purpose, the Azure Active Directory (Azure AD) can be used. It is an easy and open way to offer protection to a Web API by the use of OAuth 2.0 bearer access tokens having just a few lines of code. Developers can thus use community-driven OWIN middleware Microsoft implementation included in .NET Framework, in ASP.NET Web apps.

Following is an overview of the steps to achieve the same:

- Represent the API and register an application (backend-app) in Azure AD.
- Represent a client application for calling the API and register another application (client-app) in Azure AD.
- Grant permissions for allowing client-app to call backend-app, in Azure AD.
- Configure Developer Console for using OAuth 2.0 user authorization.
- Add validate-jwt policy for validating OAuth token for all incoming requests.

12.4.3 Securing Using OAuth

OAuth 2.0 is supported by several APIs. It helps in securing the API and also makes sure that access is available only to authenticated users, and only to the extent to which they are entitled for accessing resources. For the use of such APIs with

Azure API management's interactive developer console, the service enables the user to configure their service instance in order to work with OAuth 2.0 enabled API. The configuration of each OAuth 2.0 provider can differ even though there are similar steps to follow. Also, the pieces of information required for configuring OAuth 2.0 in the API management service instance are similar.

12.4.4 How to Provision an API Function?

Azure Functions is an ideal platform for small-scale development. It provides an array of triggers and binding types that connect input and output data, thus making the programming relatively easy. Azure Functions can be hosted on a serverless consumption plan that scales automatically and charges you only when your Function is in use. This simplifies the process of building APIs. Each API call is handled by a function that automatically scales and spins it up. If the user uses Functions for API, then the user only pays for what they use.

One of the key prerequisites for creating serverless API using Azure is an Azure subscription. In case the user does not have a subscription then, they can create a free/paid Azure account to begin creating a serverless API.

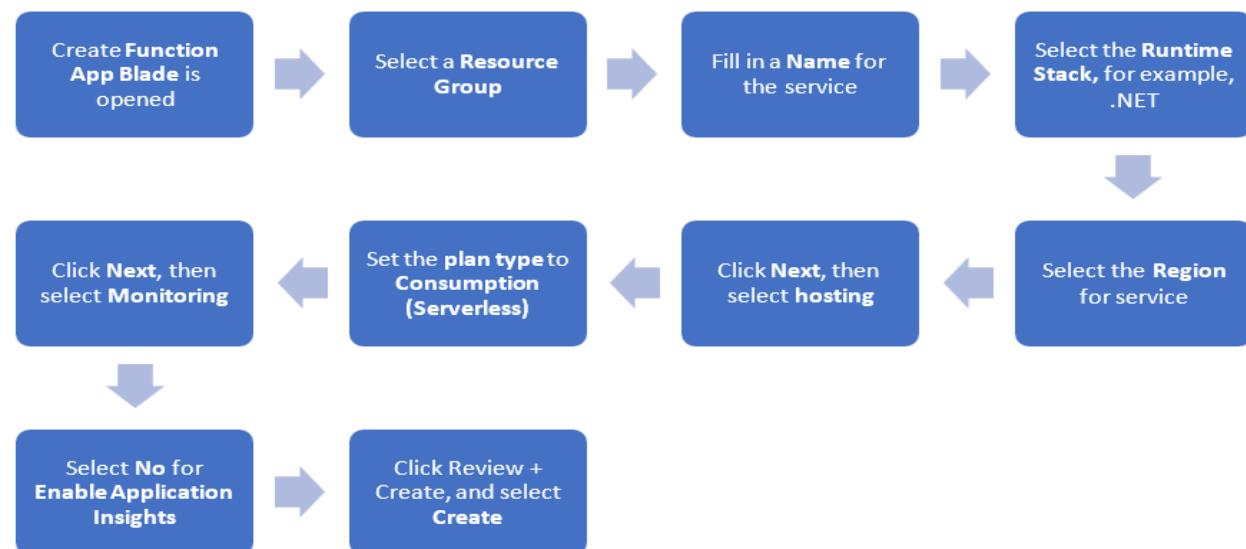
Building a Serverless API

Steps to create an Azure Function in the Azure portal and then, convert it into an API are as follows:

First, click the **Create a Resource** button which is a plus-sign button in the top left corner.

Next, search for Functions, select the **Function App**, and then, click **Create**.

Perform rest of the actions as listed here:



Create Function App

The function app enables developers to group functions as a logical unit for easier resource management, deployment, and sharing. It allows developers to run code in a serverless environment without creating a VM or publishing a Web application.

Azure Function Creation in the Azure portal

After creating the Function App, navigate to it on the Azure Portal. The Function App includes one or more functions. To create a new function, navigate to the Function blade in the Function App in the Azure portal.

To add a function, click **Create**. A new **Create function** blade will open. Perform following actions:

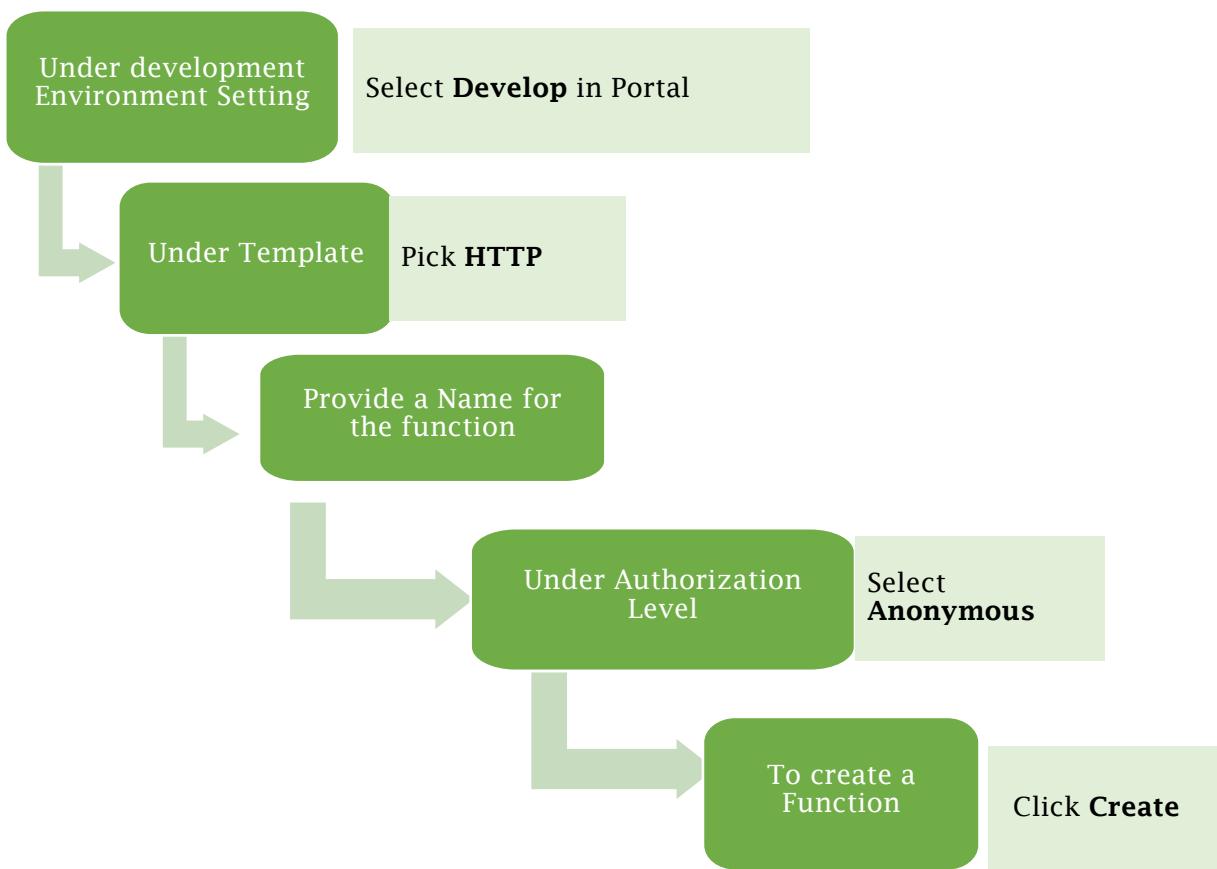


Figure 12.7 depicts creation of function in the portal.

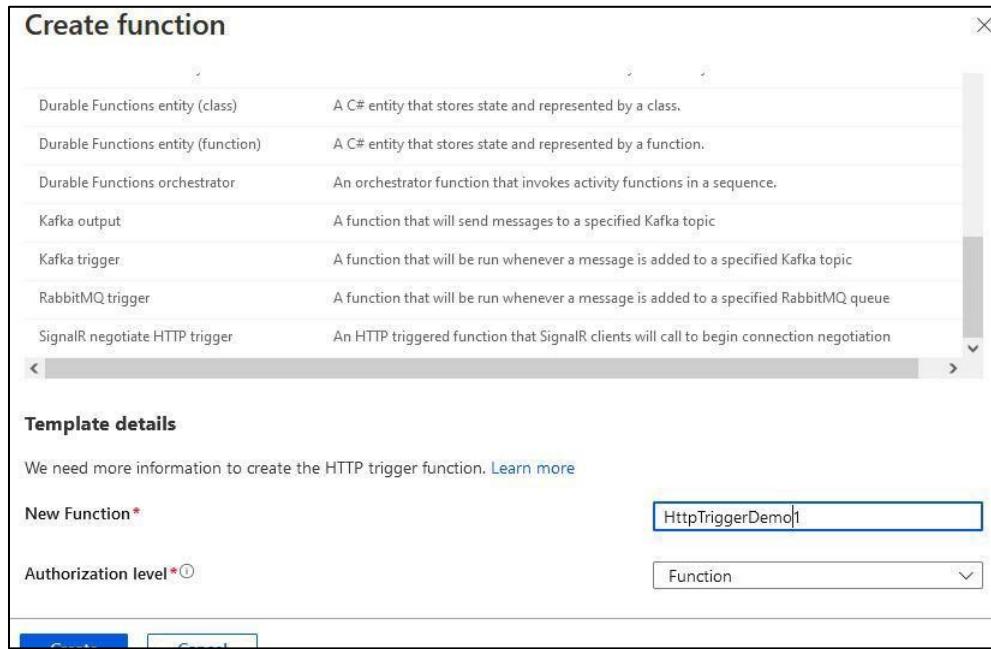


Figure 12.7: Creating a Function Using Azure Portal

The function is opened in the portal once it is created. Developers can turn it into an API to try it out.

1. Click the **Integration** menu in the Function in the Azure portal. The triggers, inputs, and outputs will be displayed. The HTTP trigger is the trigger for the function.
2. To edit the trigger, click **HTTP (req)**.

By default, the function would be triggered by navigating it to the endpoint and performing an HTTP operation such as a GET or POST against it. This is the behavior of most of the APIs. However, to make it more of a REST API, the user might change the way to fire HTTP requests at the function.

- a) To add a Route template to a function, type '**customer/{customerName}**', and an HTTP trigger will now be exposed as **functionurl/api/customer**
- b) Click **Save** to save the changes.

Figure 12.8 shows how to edit trigger in the Azure portal.

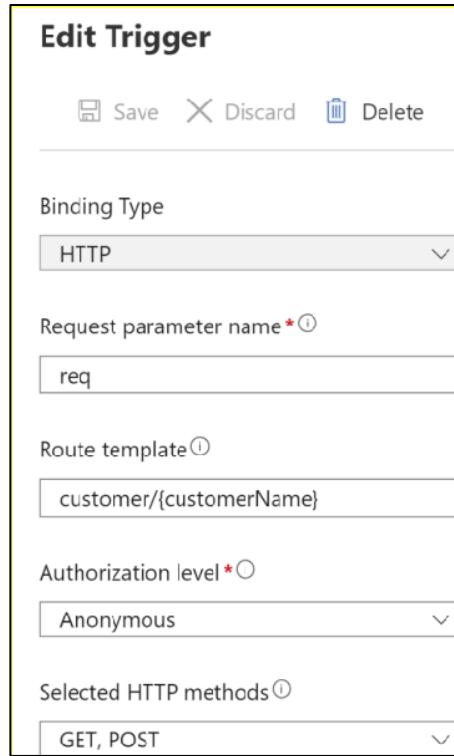


Figure 12.8: Edit Trigger in the Azure Portal

3. To edit the code of the Function, click the **Code + Test menu**. The code editor in the portal will be displayed.

The Function code is auto-generated using standard **HTTP trigger** template.

The two changes required to change this code are as follows:

- The **Run** method will now include a **string customerName** that catches **the customerName parameter** of the API route.
- **String name** will now come from the **customerName parameter**.

Code Snippet 1 shows the code that should be added in the function.

Code Snippet 1:

```
using System.Net;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Primitives;
using Newtonsoft.Json;
public static async Task<IActionResult> Run(HttpContext
req, string customerName, ILogger log)
{
    log.LogInformation("C# HTTP trigger function")
```

```

        processed a request.");
        string name = customerName;
        string requestBody = await new
            StreamReader(req.Body).ReadToEndAsync();
        dynamic data =
            JsonConvert.DeserializeObject(requestBody);
        name = name ?? data?.name;
        string responseMessage =
            string.IsNullOrEmpty(name) ? "This HTTP
                triggered function executed successfully.
                Pass a name in the query string or in the
                request body for a personalized response.":
            $"Hello, {name}. You are a customer.";
        return new OkObjectResult(responseMessage);
    }
}

```

4. To compile the changes and save the code, click **Save**.
5. To open the test window, select **Test/Run**.
6. Now, fill in a value for the customerName Query parameter.
7. To execute the code, click **Run**.
8. If the call is successful, then the entered value will be displayed in the Output tab.

Figure 12.9 shows how to edit code and test in the Azure portal.

The screenshot shows the Azure portal interface for a function named 'Customer'. On the left, the code editor displays the following C# code:

```

1 #r "Newtonsoft.Json"
2
3 using System.Net;
4 using Microsoft.AspNetCore.Mvc;
5 using Microsoft.Extensions.Primitives;
6 using Newtonsoft.Json;
7
8 public static async Task<IActionResult> Run(HttpContext req, string cust
9 {
10     log.LogInformation("C# HTTP trigger function processed a request.");
11
12     string name = customerName;
13
14     string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
15     dynamic data = JsonConvert.DeserializeObject(requestBody);
16     name = name ?? data?.name;
17
18     string responseMessage = string.IsNullOrEmpty(name)
19     ? "This HTTP triggered function executed successfully. Pass a na
20     : $"Hello, {name}. You are a customer.";
21
22     return new OkObjectResult(responseMessage);
23 }

```

On the right, the 'Test/Run' interface is shown with the following configuration:

- Input** tab is selected.
- HTTP method**: POST
- Key**: master (Host key)
- Query** section:

Name	Value
customerName	Harry
- Headers** section: + Add header
- Body** section:


```

1 {
2   "body": ""
3 }
```
- Run** and **Close** buttons at the bottom.

Figure 12.9: Code and Test in Azure Portal

9. Click the **Get Function URL** tab and **copy** the URL, to call the Function similar to an API.
10. In the new browser window, paste the URL and replace the {customerName} at the end of the URL with a value. The URL will be displayed as follows:

<https://functionappname.azurewebsites.net/api/customer/mynewcustomer>

When the value will be submitted, the value of the customer parameter will be displayed in the result. Figure 12.10 shows calling a function such as an API in the Azure portal.

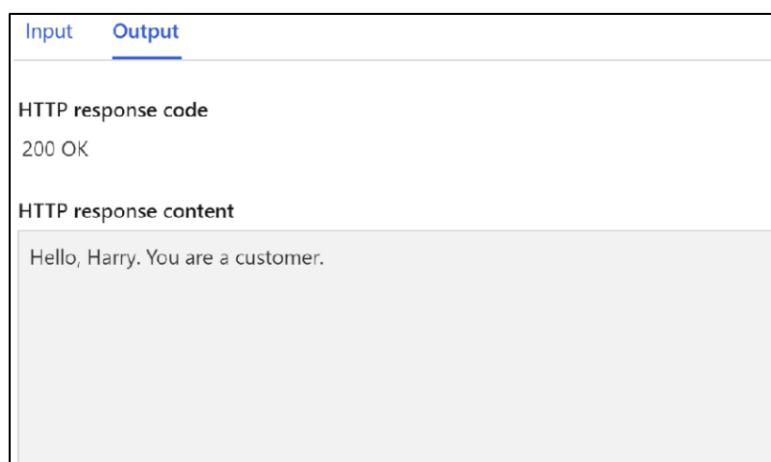


Figure 12.10: Calling a Function

To create a complete API, multiple functions can be created that can have their own purpose and can be exposed through other URLs.

12.4.5 Testing the API

The Serverless Architecture makes it simple for serving business logic, but sometimes, some of the characteristics can create challenges for testing.

Some of the challenges are as follows:

- The Serverless Architecture is a union of separate, distributed services. It must be tested both independently and together.
- The Serverless Architecture is dependent on the Internet and cloud services. These services are sometimes hard to match locally.

The Serverless Architecture features asynchronous and event-driven workflows that are difficult to emulate entirely. The principles to overcome the challenges are as follows:

- Write the business logic so that it is separate from the Function as a Service (FaaS) provider. This will make it provider-independent, reusable, and easily

testable.

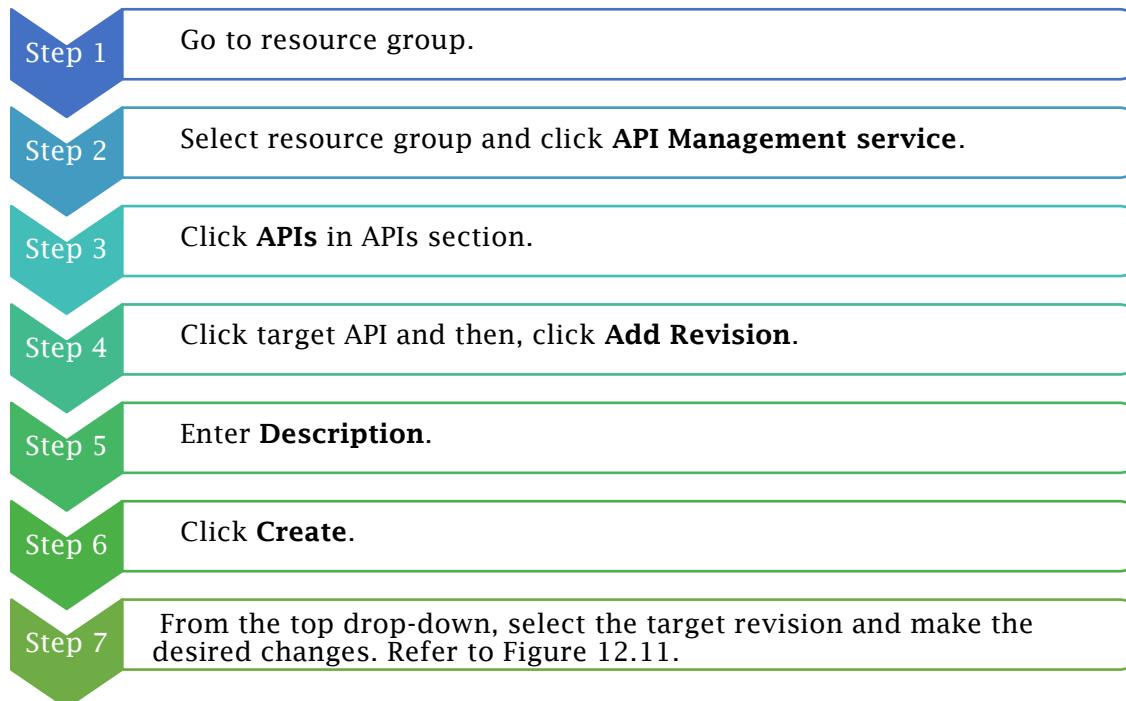
- By writing the business logic separately, it becomes easy to write the traditional Unit Tests to ensure the logic is working properly.
- The integration test is written to verify if the integration is working properly with other services.

12.4.6 Create Revision

Revision allows the user to make changes to the APIs in a controlled and safe manner.

- Create a new revision to make changes.
- The APIs can be then edited and tested without disturbing the API consumers. Then, the revision can be made current when it is ready.

Steps to create a revision are shown as follows:



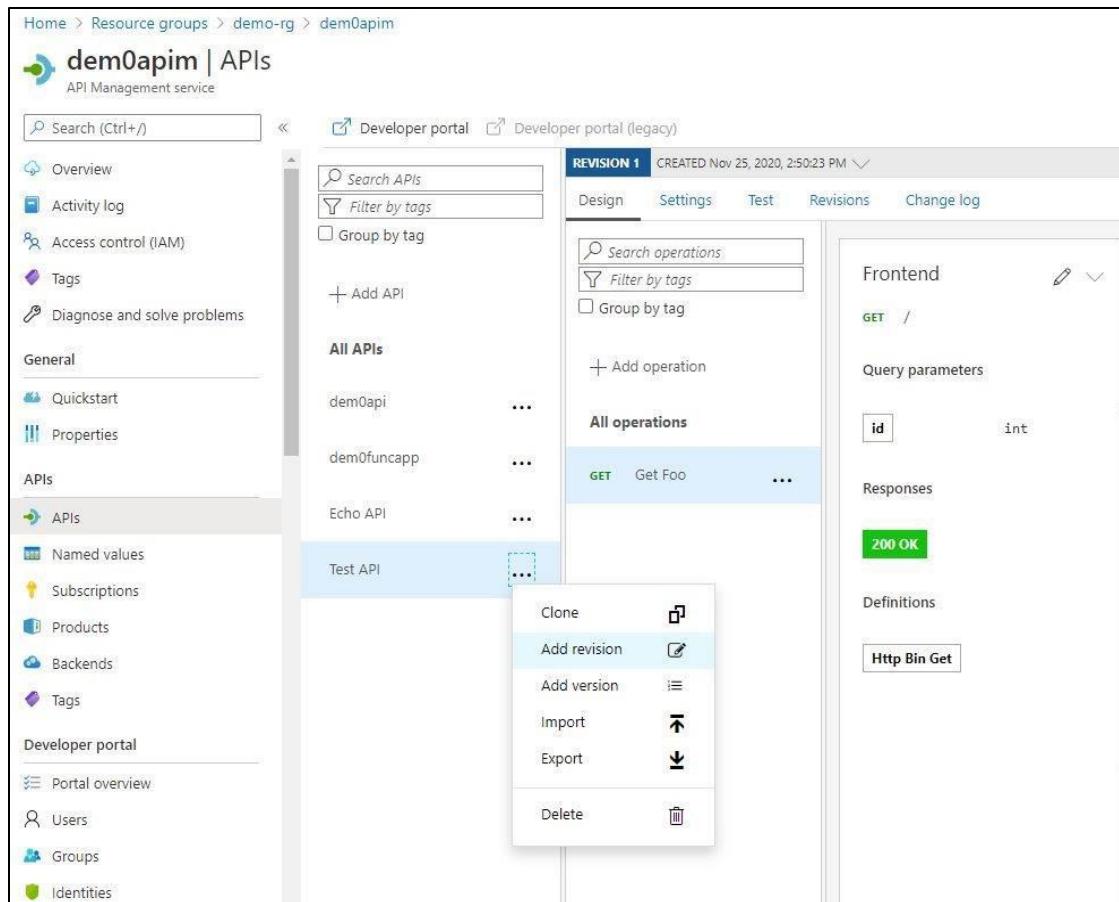


Figure 12.11: Create Revision

12.5 Defining and Implementing Policies

An Azure policy presents a service in Azure, which can be used for creating, assigning, and managing policies. With such a policy, IT issues can be handled and avoided with policy definitions. The policies lay different rules and consequences for the resources. In Azure policy usage, resources are in compliance with service level agreements and corporate standards.

Azure policy runs verifications of the resources. It also scans for the ones not in compliance with the policies created. For instance, there can be a policy for allowing only a particular SKU size of virtual machines in the given scenario.

Azure policy can be used to enforce policies for regular tasks such as to create, assign, and manage policies throughout an organization. Following are some examples:

- A policy can be assigned for applying a condition for resources to be created in the future.
- An initiative definition can be made, which is assigned for tracking

- compliance for various resources.
- A non-compliant resource/denied resource can be resolved.
- A new policy can be implemented throughout an organization.

12.6 Defining an API Interface Using Azure Portal and Swagger

Azure API Management (APIM) offers help to organizations for publishing APIs to internal, external, and partner developers. This is done in order to unlock the capacity of the data as well as services. The recent trend in businesses world-wide is to look for opportunities to expand operations digitally, create new channels, and make new customers while establishing stronger relations with existing ones.

API Management, thus, offers core competencies for a successful API program by engaging developers, improving business insights, security, analytics, and protection. APIM can be used at the backend for launching a complete API program based on it.

APIM offers the facility for creating developer portal and API gateway. This is secure, scalable, and will work with any API or host. In addition, it would also offer an insight of the existing APIs. It is a continuous challenge for API developers to better manage the boarding process and documentation, and at the same time provide the simplest integration experience to the end users of the APIs. APIM presents a developer portal for managing the subscription of services. It thus provides an outstanding experience for the developer.

12.6.1 Using Swagger in Azure

Swagger represents an open source framework that is powerful and is assisted by a significant ecosystem of tools. It helps in building, designing, documenting, and consuming RESTful APIs. Swagger works as follows: Developers should 'swagger-enable' their API. This can be done in .NET by taking in a Nuget package called Swashbuckle. This autogenerated a Swagger definition file for the API in JSON format. The file can be easily accessed with a URL on the site of the API. Code generation tools can be used for reading the Swagger definition file as well as auto-generating client SDK code in any of the supported languages.

Swagger auto-generates client SDKs and definition file. It also auto-generates a UI for finding and testing the APIs using Swagger UI. It is an open source that defines the method to describe the look of a Web API in a simple text file. Further, computers can read the file with extension *.yaml or *.JSON and it can be used for generating actual server API code as well as client code. Reverse engineering of a present API is also supported by Swagger for producing a JSON Swagger file. The Swagger standard has recently been adopted and renamed as OpenAPI.

Microsoft ASP.NET MVC Web API server stub code can be generated by Swagger.IO from the yaml in the editor. The code generated can be downloaded at a later time. Subsequently, the developer should implement the internal logic for returning a favorable result. When the generated .NET ASP.NET MVC code is deployed to an Azure App Service, it is called an Azure API app.

12.7 Manage Running Services (Using Logging, Disaster Recovery, and Multiple Regions)

By using Azure management and governance tools, system administrators and developers can maintain secure resources that are compliant. This is ensured for both in the cloud and also on-premises. The services can be used to monitor infrastructure and applications, update apps, provide and configure resources, create backup resources, build disaster recovery, automate processes, and so on.

12.7.1 Azure Logging

In Azure, there is a comprehensive list of security auditing and logging options that can be configured. These help in identifying lapses in security mechanisms and policies.

Logs offer data for keeping applications up and running. Thus, troubleshooting of issues that occurred previously can be done and at the same time, prevention of new ones. This helps in improving the performance and maintainability of an application. In addition, actions can be automated so that manual intervention is not required. Following are the types of Azure logs:

Control/Management Logs	Data Plane Logs	Processed Events
Control/management logs give details on the Azure Resource Manager Operations such as CREATE, DELETE, and UPDATE.	Data plane logs give details on events that are presented as a part of Azure resource usage. For example, the Windows event system log, security logs, diagnostics logs configured through Azure Monitor, and application logs in a VM.	Processed events give details on analyzed events or alerts processed on behalf of the user. For example, Azure Security Center alerts. In this, the Azure Security Center completes the processing and analysis of the subscription. It thus provides brief security alerts.

12.7.2 Disaster Recovery Using Multiple Regions

Azure Site Recovery plans and handles disaster recovery for Azure VMs and physical servers as well as on-premises VMs. It is essential for organizations to include a strategy of Business Continuity and Disaster Recovery (BCDR). The BCDR should ensure safety of data and at the same time maintain workloads and apps as up and running in the event of a predicted/unpredicted outage. Azure Recovery Services supports BCDR strategy in the following ways:

➤ Site Recovery Service

Site Recovery aids in ensuring business continuity. This is done by maintaining workloads and business apps as running at the time of outages. Workloads that are running on physical as well as VMs are replicated by site recovery, from a primary site to a secondary location. If there is an outage at a primary site, there is failover to the secondary location from where apps can be accessed. When the primary location starts again, it can be used again.

Replication is managed by Site Recovery for the following:

- For Azure VMs that replicate involving Azure regions
- For on-premises VMs as well as physical servers that replicate to Azure or alternatively to a secondary site

Azure site recovery service supports disaster recovery strategy. This is done as it manages and plans replication and failover, as well as fallback of on-premises machines and Azure VMs.

Following high-level steps can be used for setting up disaster recovery to a secondary Azure region for Azure VMs.

- Creating a recovery services vault
- Verifying target resource settings
- Setting up an outbound access for VMs
- Enabling replication for a VM

➤ Backup Service

Azure Backup service helps in keeping data safe and also recoverable. This is done by taking a backup to Azure.

Serverless computing in Microsoft Azure offers abstraction of servers, infrastructure, and operating systems for developers.

Products and services available under serverless computing include Azure Logic Apps, Azure Functions, and so on.

Azure Logic apps help to design and build scalable solutions for application

integration, data integration, and system integration.

Azure Functions are a powerful solution to run small pieces of code or functions on the cloud.

Service Fabric and microservices are referred to as modern computing, which help in building distributed, scalable, stateless, and stateful microservices running in container services.

Azure provides a secure environment with multiple security features such as SSL certificate, Active Directory services, and easy integration of OAuth authentication.

12.8 Mastering API Design Principles

Let us look at some design principles for API design.

12.8.1 RESTful API Design

Representational State Transfer (REST) is an architectural style for designing networked applications. It relies on stateless, client-server communication, where each message contains all the information necessary to understand the request. RESTful APIs are designed around resources, which are any kind of object, data, or service that can be accessed by the client.

Key Principles:

Resource Identification: Uses Uniform Resource Identifiers (URI) to identify resources uniquely.

Uniform Interface:

REST APIs have a uniform interface, simplifying the architecture and improving scalability and performance. This includes using standard HTTP methods (GET, POST, PUT, and DELETE) in a consistent way.

Statelessness:

Each request from client to server must contain all the information required to understand and complete the request. No client context is stored on the server between requests.

Cacheable:

Responses should be implicitly or explicitly labeled as cacheable or non-cacheable, to prevent clients from reusing stale or inappropriate data.

Layered System:

A client cannot ordinarily figure out whether it is connected directly to the end server or to an intermediary along the way.

12.8.2 Versioning

API versioning is crucial for maintaining and evolving APIs over time without disrupting existing clients. There are several strategies for versioning:

Uniform Resource Identifiers (URI) Versioning:

Including the version number in the API path (example, /api/v1/resource).

Parameter Versioning:

Using a request parameter to specify the version (example, /api/resource?version=1).

Header Versioning:

Specifying the version in a custom HTTP header (example, Accept-version: v1).

Media Type Versioning:

Using the Accept header to specify version as part of the content type (example, Accept: application/vnd.myapi.v1+json).

Each strategy has its advantages and disadvantages and the choice depends on the specific requirements and constraints of the API and its consumers.

12.8.3 Content Negotiation

Content negotiation allows clients to indicate the format of the response they wish to receive (example, JSON, XML, and so on) and enables the server to serve different formats based on the client's request. This is typically achieved using the Accept HTTP header.

Use of Accept Header: Clients specify their preferred response format (example, Accept: application/json or Accept: application/xml).

Server Response: The server evaluates the Accept header and responds with the Content-Type header indicating the format of the returned data.

Fallback Mechanism: Implement a fallback mechanism for clients that do not specify a preference, ensuring a default response format is always available.

12.9 Implementing Event-Driven Architectures Using Azure Functions

Event-Driven Architecture (EDA) is a design paradigm in which the flow of the program is determined by events such as user actions, sensor outputs, or message passing between processes. In an EDA, components of the application react to events by processing data, communicating with other services, or performing other actions in response. This architecture enables highly responsive, scalable, and flexible systems that can easily adapt to changes in requirements or workloads.

Azure Functions is a serverless compute service that enables the user to run event-driven code without having to explicitly provision or manage infrastructure. It fits perfectly into the EDA paradigm by allowing developers to focus on the application logic rather than the underlying hardware or platform.

12.9.1 How Azure Functions Facilitate EDA?

Azure Functions facilitate EDA in following ways:

Triggers and Bindings: Azure Functions supports a variety of triggers (example, HTTP requests, timer events, changes in data, or messages from other services) that can automatically execute the user's code in response to external events. Bindings allow for declarative connections to data sources, simplifying the task of sending or receiving data from your function.

Integration with Azure Services: Azure Functions seamlessly integrates with other Azure services such as Azure Event Grid, Azure Event Hubs, Azure Service Bus, and Azure Storage queues. This makes it easy to build complex event-driven workflows that span across various Azure services and resources.

Scalability: Azure Functions automatically scales based on the number of incoming event triggers, making it well-suited for applications with variable workloads. This means that the created application can handle a large number of events without requiring manual scaling operations.

Development Flexibility: A user can develop Azure Functions using multiple programming languages such as C#, Java, JavaScript, TypeScript, Python, and PowerShell. This flexibility allows developers to use their preferred language to implement the logic responding to events.

Serverless Model: With Azure Functions, the user only pays for the compute time the functions consume, which is ideal for event-driven applications that may have sporadic or unpredictable workloads. The serverless model also removes the necessity for managing servers, allowing developers to concentrate on writing code.

12.10 Implementation of Event-driven Applications Using Azure Functions

Step 1:

In Visual Studio 2022, start with an empty Azure Function application and install the ‘Microsoft.Azure.WebJobs.Extensions.EventGrid’ package using NuGet package manager. Refer to Figure 12.12.

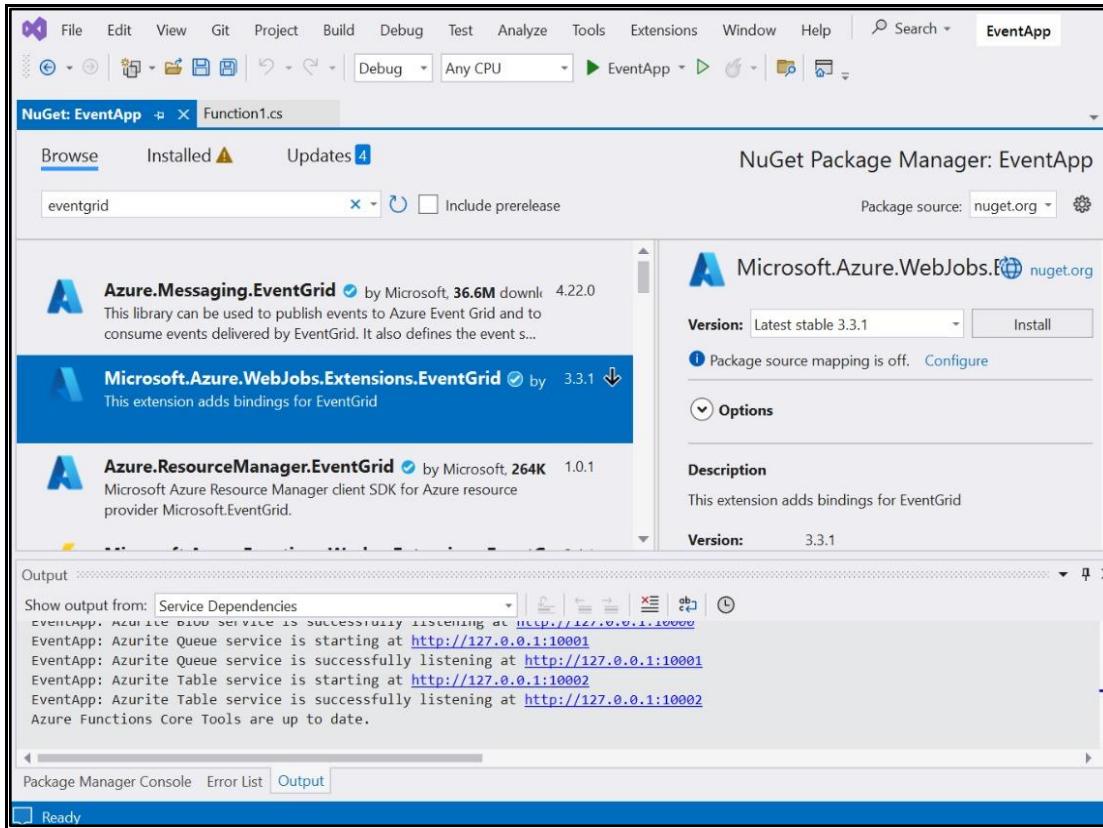


Figure 12.12: Installing ‘Microsoft.Azure.WebJobs.Extensions.EventGrid’ Package

Step 2:

Add a new item, select Azure Functions and name it as ‘EventHandler.cs’. For the newly created Azure Function, pick ‘Event Grid Trigger’ and the user will see an existing template as shown in Figures 12.13 and 12.14.

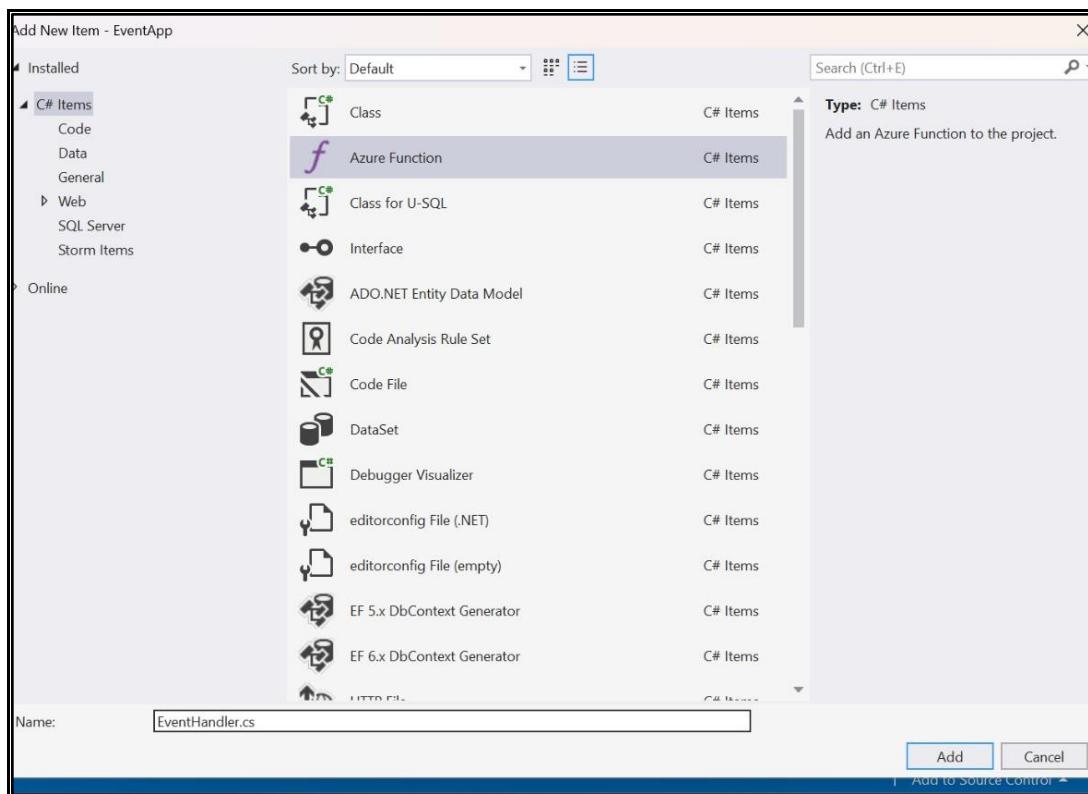


Figure 12.13: Creating an Azure Function

The screenshot shows the Visual Studio code editor with the file 'Function1.cs' open. The title bar says 'EventApp'. The code editor displays the following C# code:

```
// Default URL for triggering event grid function in the local environment.  
// http://localhost:7071/runtime/webhooks/EventGrid?functionName={functionname}  
  
using System;  
using Azure.Messaging;  
using Microsoft.Azure.Functions.Worker;  
using Microsoft.Extensions.Logging;  
  
namespace EventApp  
{  
  
    public class EventHandler  
    {  
        private readonly ILogger<EventHandler> _logger;  
  
        public EventHandler(ILogger<EventHandler> logger)  
        {  
            _logger = logger;  
        }  
    }  
}
```

The status bar at the bottom shows 'No issues found' and 'Ln: 1 Ch: 1 SPC CRLF'.

Figure 12.14: Template for Event Grid Trigger

Step 3:

Publish the newly created Azure Function directly to the existing Azure subscription as shown in Figure 12.15.

This step will start building our created application.

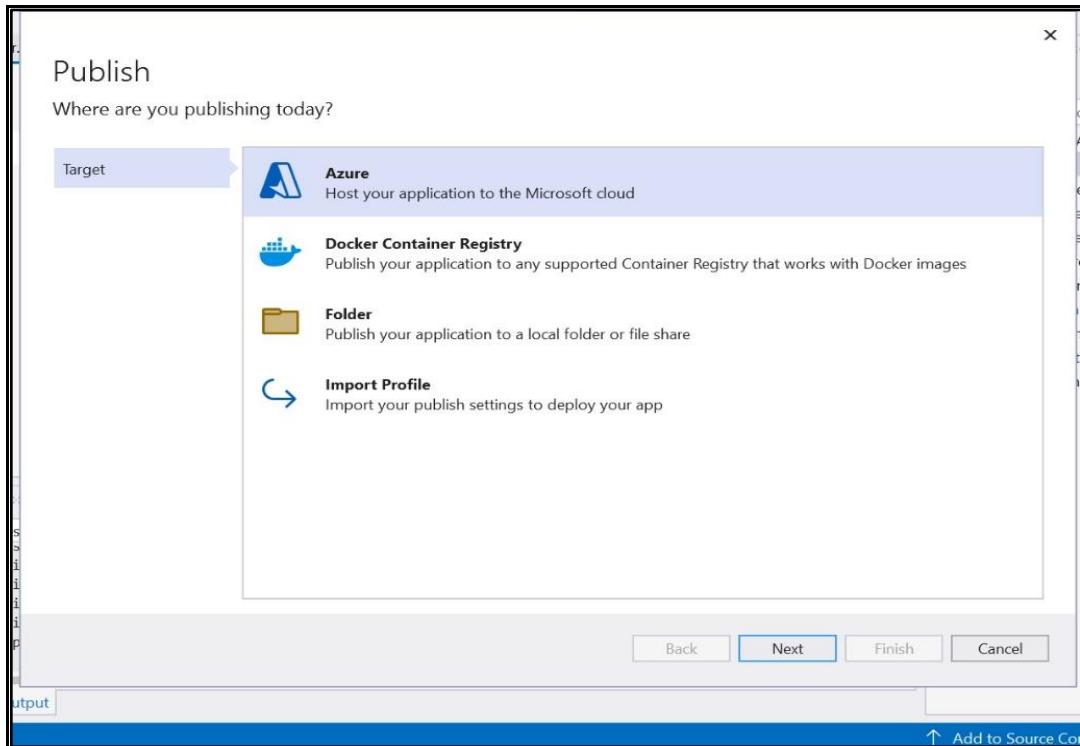


Figure 12.15: Publish and Build Function

Step 4:

Now, on Microsoft Azure cloud, go to the created resource group and storage account, add the created Azure function and create an event subscription to it as shown in Figures 12.16 and 12.17.

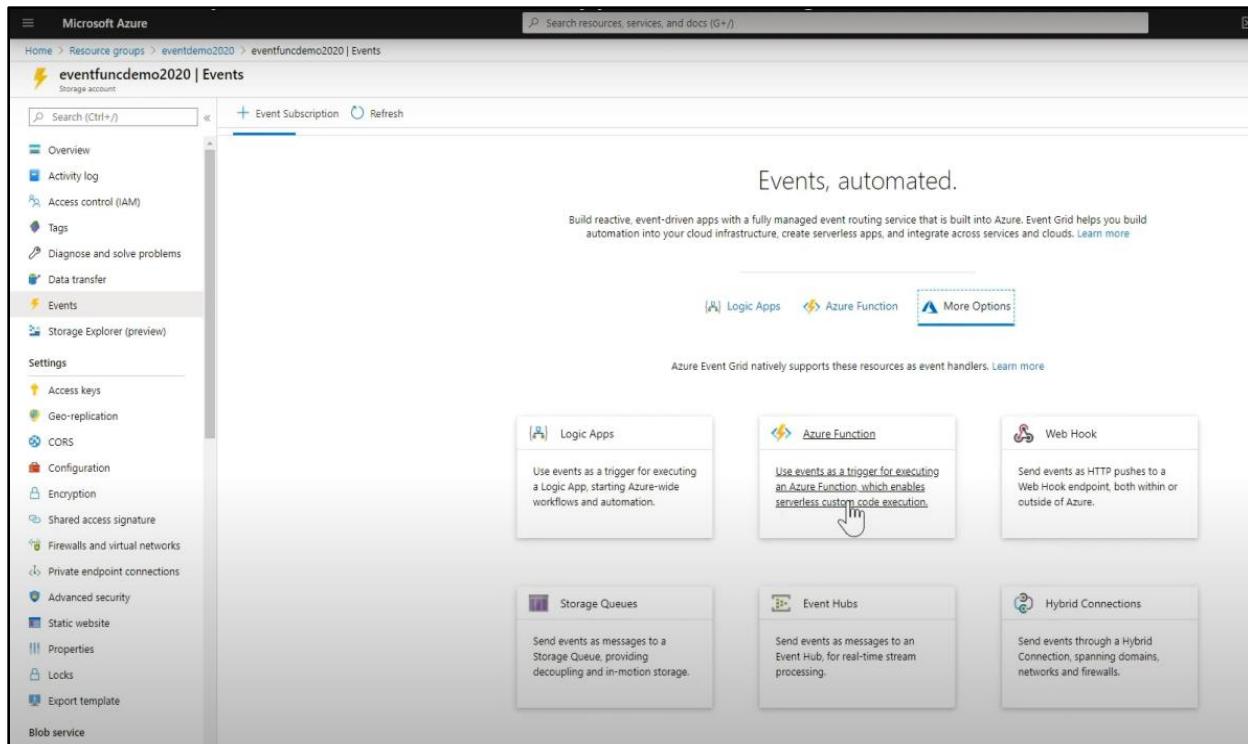


Figure 12.16: Created Event on Azure Cloud

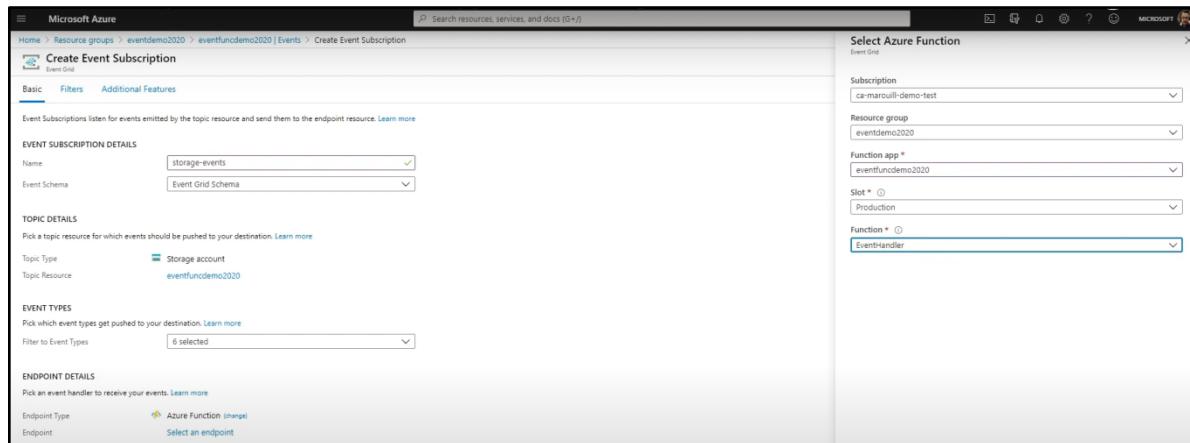


Figure 12.17: Creating an Event Subscription

Step 5:

Utilize Azure Monitor and Application Insights to track the performance and health of the functions which have been deployed. Also, analyze execution logs and metrics to ensure the functions are performing as expected and troubleshoot any issues if any. This concludes implementation of event-driven Applications Using Azure Functions.

12.11 Summary

- ✓ Serverless computing in Microsoft Azure offers abstraction of servers, infrastructure, and operating systems for developers.
- ✓ Products and services available under serverless computing include Azure Logic Apps, Azure Functions, and so on.
- ✓ Azure Logic apps help to design and build scalable solutions for application integration, data integration, and system integration.
- ✓ Azure Functions are a powerful solution to run small pieces of code or functions on the cloud.
- ✓ Service Fabric and microservices are referred to as modern computing, which help in building distributed, scalable, stateless, and stateful microservices running in container services.
- ✓ Azure provides a secure environment with multiple security features such as SSL certificate, Active Directory services, and easy integration of OAuth authentication.
- ✓ Mastering API design involves creating RESTful services with clear resource identification and stateless interactions.
- ✓ Implementing thoughtful versioning strategies for long-term evolution and employing content negotiation to dynamically serve different data formats. These are based on client preferences, all while ensuring consistency, scalability, and adherence to HTTP standards for optimal performance and user experience.
- ✓ Azure Functions facilitate event-driven architectures by providing a serverless platform that automatically executes code in response to events, offering seamless integration with Azure services.
- ✓ Automatic scalability, and flexible development across multiple programming languages. This approach allows for the building of responsive, scalable applications with minimal overhead.

12.12 Test Your Knowledge

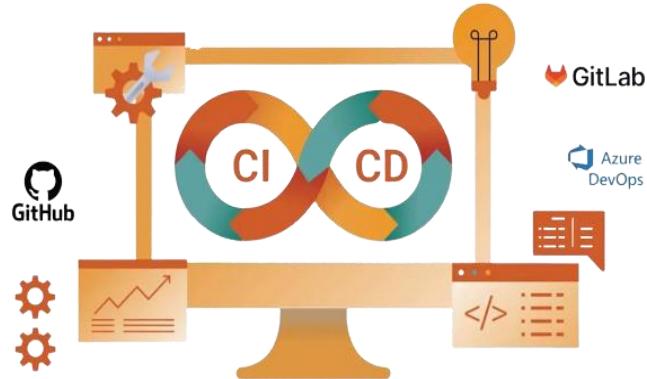
1. Which of the following Azure Services can be integrated with Logic apps?
 - A. Azure Service Bus
 - B. Azure Functions
 - C. Azure Storage
 - D. All of these
2. Which of the following statements is false about Azure Functions?
 - A. Azure Function helps to run small pieces of code on the cloud
 - B. Only C# code is used to write Azure Function
 - C. Azure Functions is a serverless application
 - D. None of these
3. Which of the following are part of Azure security?
 - A. Security certificate
 - B. Azure AD authentication
 - C. OAuth authentication
 - D. All of these
4. Which of the following is not a main category of Azure logging service?
 - A. Control or management logs
 - B. Data plane logs
 - C. Azure SQL database logs
 - D. Backup service logs
5. Which of the following statements regarding Service Fabric are not correct?
 - A. Service Fabric helps in building and managing applications that are dependable and scalable
 - B. Service Fabric offers a high-level and lightweight runtime, which can be used for creating scalable, stateless, distributed, and stateful microservices that run in containers
 - C. As a container organizer, Service Fabric is not recommended when a requirement involves only deploying and managing containers
 - D. Some services powered by Service Fabric are Azure SQL, Microsoft Power BI, Azure Cosmos DB, Azure IoT Hub, Azure Event hubs, Skype for business, as well as several core Azure services

12.12.1 Answers to Test Your Knowledge

1. D
2. B
3. D
4. C
5. C

Try It Yourself

- Create an Azure Function triggered by an HTTP request that returns a simple 'Hello, World!' message.
- Create a serverless function that simulates an asynchronous, event-driven workflow, such as processing an image upload and write a unit test to verify its functionality independently from the cloud environment.



SESSION 13

CI/CD

Overview

This session introduces Continuous Integration and Continuous Delivery (CI/CD) and its pipeline mechanism, focusing on the Azure CI/CD pipeline. It explains how to create and release Azure pipelines. Furthermore, it discusses how integrating Azure DevOps with LambdaTest enables running tests on applications. The session also covers setting up artifacts for efficient software delivery. It delves into Infrastructure as Code (IaC) concepts using Azure Resource Manager templates, Terraform, or similar declarative scripting languages.

Learning Objectives

In this session, students will learn to:

- Explain CI/CD and its pipeline mechanism
- Define the Azure CI/CD pipeline
- Explain the advantages of Azure Pipelines
- Define how Azure release pipelines works
- Outline the process to integrate Azure DevOps with LambdaTest to run test on applications
- Explain the role and importance of artifact repositories in CI/CD
- Explain Infrastructure as Code (IaC)

13.1 What is CI/CD?

CI/CD is a way to regularly supply apps to clients by introducing automation into the tiers of app improvement. The most important standards attributed to CI/CD are non-stop integration, non-stop delivery, and non-stop deployment. CI/CD is an approach to the issues that new code integration can cause for development and operations teams.

CI/CD introduces ongoing automation and non-stop tracking during the lifecycle of apps, from integration and testing phases to delivery and deployment. These linked practices are commonly known as a 'CI/CD pipeline' and are supported by agile development and operations teams using either a DevOps or Site Reliability Engineering (SRE) method.

'CI' in CI/CD refers to non-stop integration, that is, an automation technique for developers. Success in CI happens when any code changes in an application result in fresh builds, tests, and merges into a shared repository. The integration process is continual and round-the-clock. It is an approach to the hassle of getting too many branches of an app in improvement right away that could synchronize content with each other.

'CD' in CI/CD pertains to non-stop delivery and/or non-stop deployment (these terms are used interchangeably these days). Non-stop delivery/deployment involves automating the later stages during the lifecycle of application development.

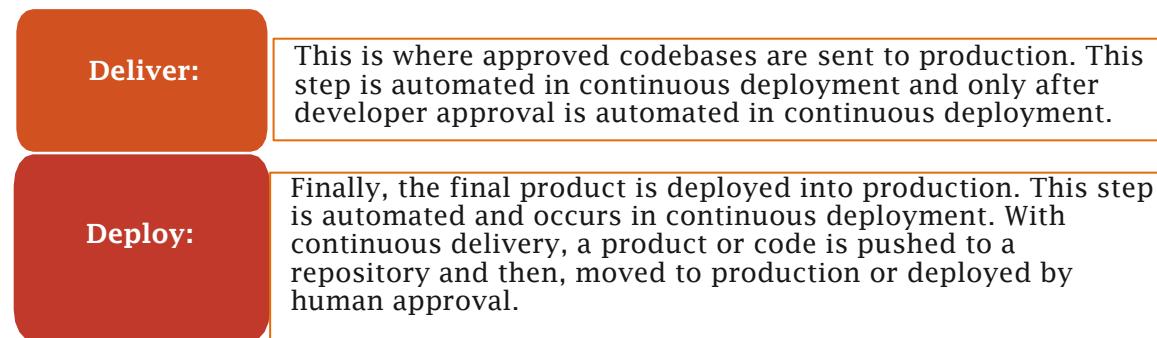
While CI indicates the mechanism to build and test code automatically, CD is the process that deploys all code changes within a build to the testing or staging environment. CD makes it possible to release builds to the production environment whenever required. By enabling teams to deploy as and when required, CD reduces time to make a product market ready.

In Continuous delivery, code changes made by development teams are debugged, tested, and uploaded to a source code repository such as GitHub. This code can be deployed to production environments.

Continuous deployment takes care of automating code changes from the shared repository to production and addresses the hassle of slow app deliveries.

13.1.1 Stages of a CI/CD Pipeline

Following are the key stages in a CI/CD pipeline:



13.1.2 Tools for CI/CD

CI/CD is today considered as the backbone of a DevOps methodology because it enables collaboration between developers and IT operations teams to deploy and deliver software.

Several readymade CI/CD tools are available today that enable teams to automate development, deployment, and testing tasks in the lifecycle of application development. Popular tools include Jenkins, CircleCI, GitLab, and Azure Pipelines.

13.1.3 Azure Tools for CI/CD

As a CD tool, Azure DevOps is quite adaptable and offers customizable pipelines. The one disadvantage of Azure DevOps Pipelines is that it requires a significant amount of scripting during the setup and configuration process.

Azure DevOps provides assistance with Azure, Kubernetes, and VM-based resources. Their GitOps functionality is limited to pipelines, not releases. There are no validation capabilities and unscheduled downtime is accomplished through a plugin. Software delivery metrics such as build history and deployment status are available, but they are limited to determining deployment speed.

Azure DevOps supports the creation of Docker images and makes it simple to deploy and run Docker containers on Azure.

A DevOps artifact is a document, file, service, code, or component that is created as part of the software development process. Artifacts are useful to give information to stakeholders, and others who are involved in the process. They can also be useful during continuous development.

Microsoft and Google recently partnered with Codefresh to become the preferred third-party CI/CD solution and with a recent GitHub acquisition, Microsoft already sells two competing products: GitHub Actions and Azure DevOps.

Features of Azure DevOps include:

- Limited GitOps Functionality
- Granular RBAC and SSO
- Limited Infrastructure Provisioners
- SaaS and On-Premise
- Cloud-Native and Traditional App Support

For integration, testing, and deployment, efficient CI/CD pipelines use open source tools. The CI/CD process is always considered for the perfect execution of the software development projects. Jenkins is the most commonly used open-source tool that supports building, deploying, and automating software development projects.

When working in a cloud environment, the team uses a container such as Docker for packaging and delivering applications, as well as Kubernetes for orchestration. Although Kubernetes is not specifically designed for the CI/CD pipeline, it is used in numerous CI/CD workflows.

No matter what CI/CD tools, plugins, or configurations are used in the pipeline, the primary goal should be to streamline and automate the software development process.

13.1.4 Setup and Configuration

Azure DevOps produces a CI/CD pipeline within Azure Pipelines. One can generate a new Azure DevOps organization or they can use an existing organization. Azure DevOps also generates Azure resources within the Azure subscription of developer's choice.

Developers can perform following steps to create a new DevOps organization:

- 1) Go to Azure DevOps Services (dev.azure.com) and click **Start free** as shown in Figure 13.1.

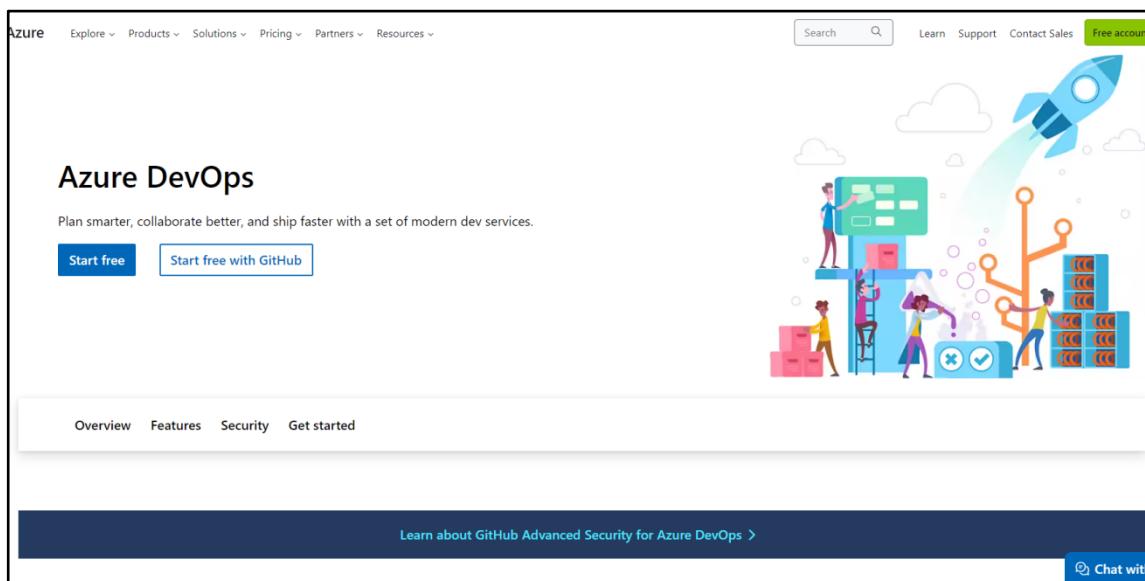


Figure 13.1: Azure DevOps

- 2) Sign in by using a Microsoft account. If one does not have a Microsoft account, select **Create One!** and finish the steps. Then, developer will be redirected to **Get started Page** as shown in Figure 13.2.

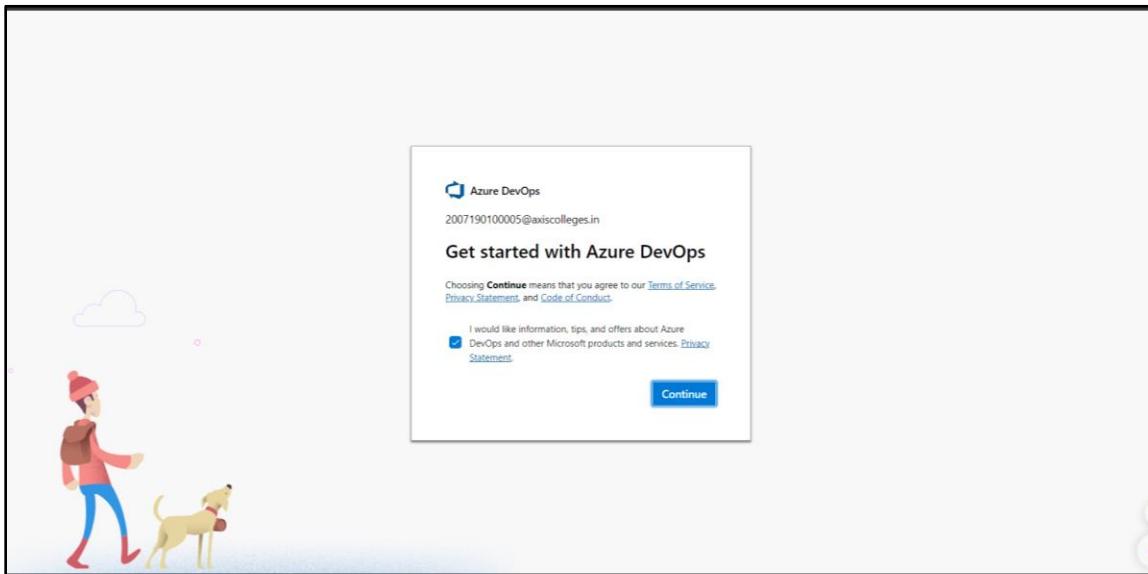


Figure 13.2: Get started with Azure DevOps

- 3) Once login is done, the organization creation page will appear. Refer to Figure 13.3.

Then, enter following details:

- Enter the Name of DevOps organization
- Select nearest location where to host the project
- Complete the captcha.

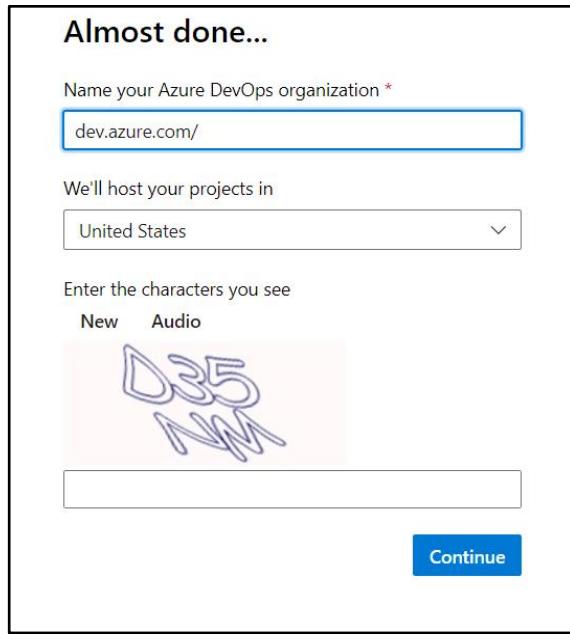


Figure 13.3: Organization Creation Page

- 4) Click **Continue** and **Create a Project** page will appear as shown in Figure 13.4.

The screenshot shows the 'Create a project to get started' page. It features a cartoon illustration of a person sitting on the ground with a dog, surrounded by clouds. The form includes fields for 'Project name*', 'Description', and 'Visibility'. The 'Private' option is selected, highlighted with a blue border. A note at the bottom states that public projects are disabled for the organization.

Figure 13.4: Create a Project

- 5) Fill in following details:
 - In the **Project Name** field, enter the name of the project.
 - In the **Description** field, enter the description of the project.
 - Under **Visibility**, choose whether to make the project public or private.

- Click **Advanced** as shown in Figure 13.5.

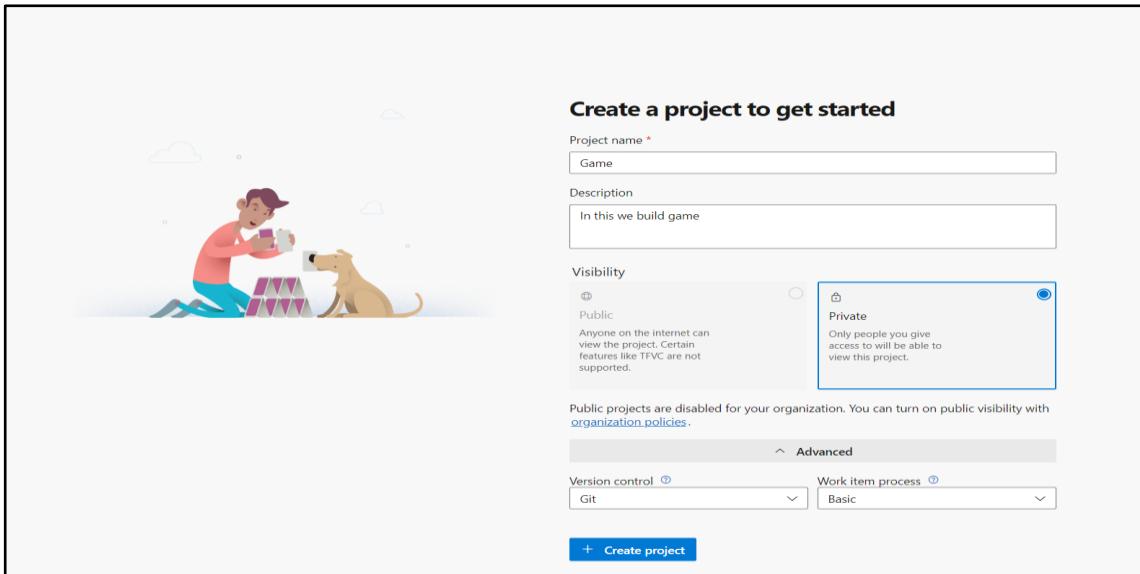


Figure 13.5: Advanced Option in Creation of Project

- 6) In version control, select **Git** and in the work item process, select **Basic** as shown in Figure 13.5. Then, click **Create project** and the project will be created. Refer to Figure 13.6.

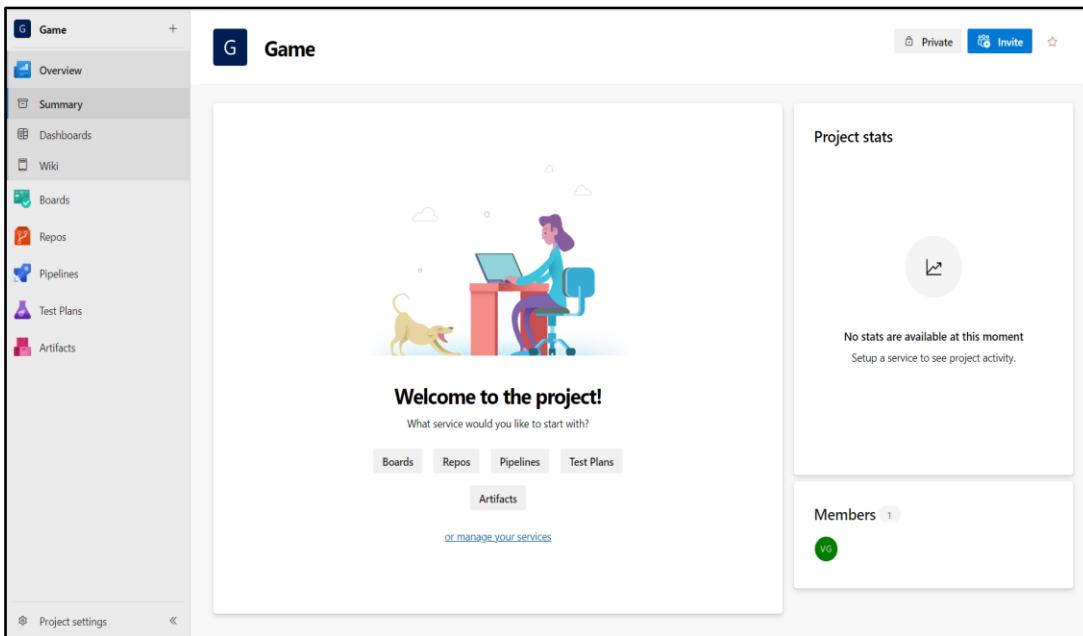


Figure 13.6: Welcome Page

- 7) Create or upload some files to the project.

If the developer does not have an existing project, then the developer can run a template that creates a demo project in Azure DevOps.

Developers can perform following steps to create a demo project:

- 1) Go to Azure DevOps Demo Generator (azuredevopsdemogenerator.azurewebsites.net) and click **Sign In** as shown in Figure 13.7.

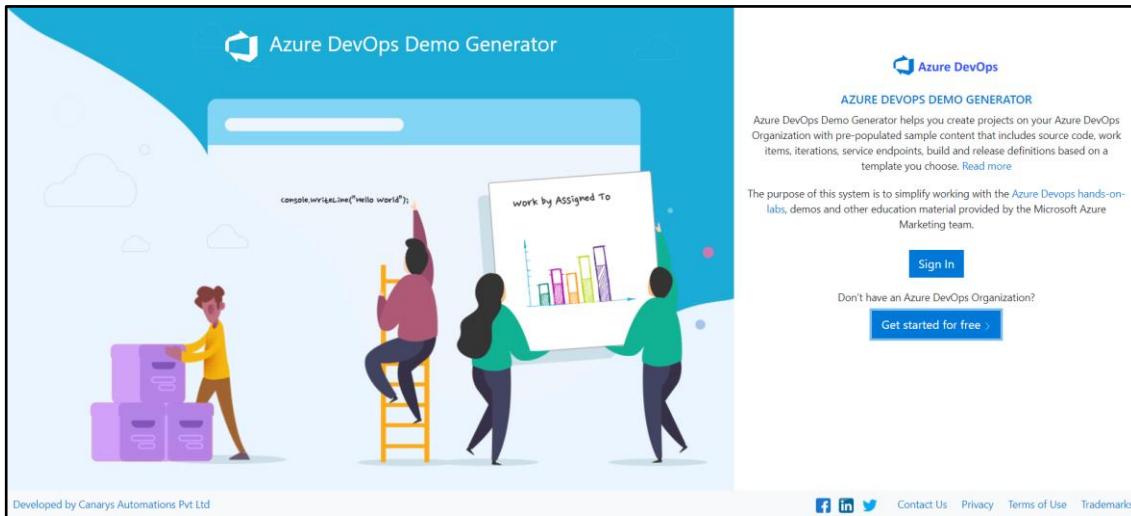


Figure 13.7: Azure Devops Demo Generator

- 2) Sign in by using a Microsoft account. If one does not have a Microsoft account, select **Create One!** and finish the steps. Then, to create a new project click **Choose template** and select a suitable template. Give a new project name and select the Organization. Finally, click **Create Project**. Refer to Figure 13.8. Here **eShopOnWeb** template has been chosen to create the demo project.

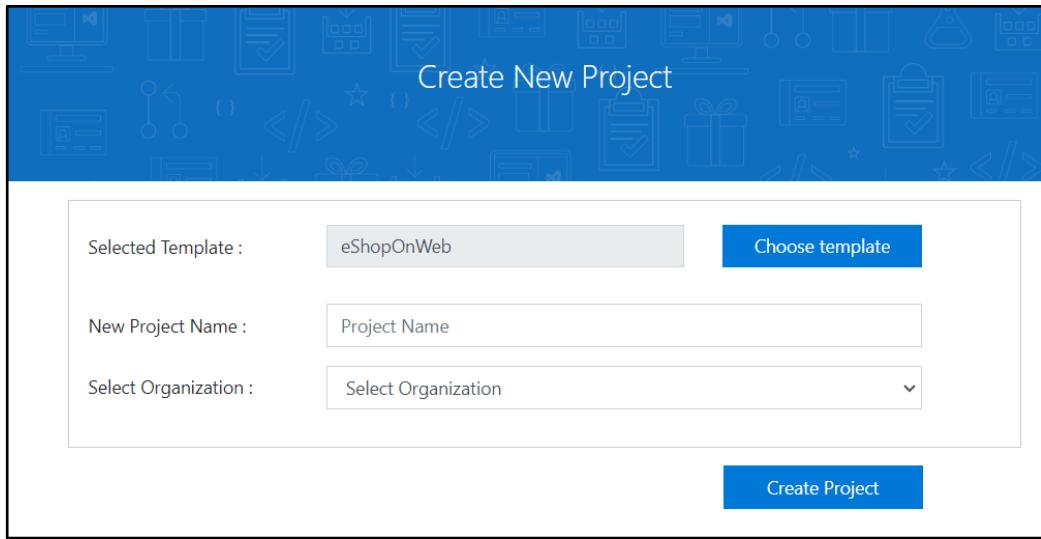


Figure 13.8: Create New Project

- 3) Click **Navigate to project** and the developer will be redirected to the project in Azure DevOps.

13.1.5 Configuring Git and Cloning a Project

Now, to configure Git and clone the project with Azure DevOps so that they can work on files from the cloud locally, developers must perform following steps:

- 1) Start Visual Studio (VS) Code as shown in Figure 13.9. If developers do not have it installed on their systems, they must install it before proceeding with further steps. Also, they must ensure that Git has been downloaded and installed on the local system. The link to download Git is here: <https://git-scm.com/download/win>.

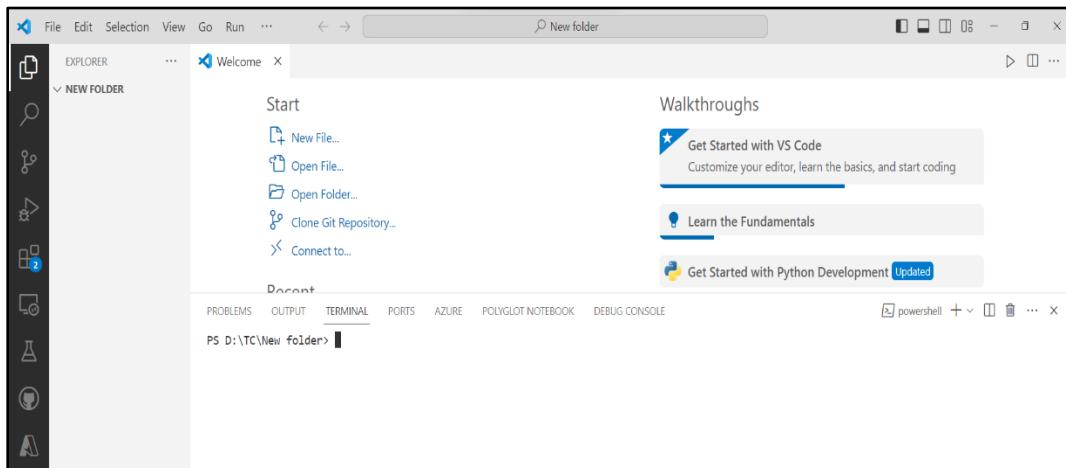


Figure 13.9: Visual Studio Code

- 2) Click the drop-down beside **add** icon in the Terminal of VS Code and select the **Git Bash** option as shown in Figure 13.10.

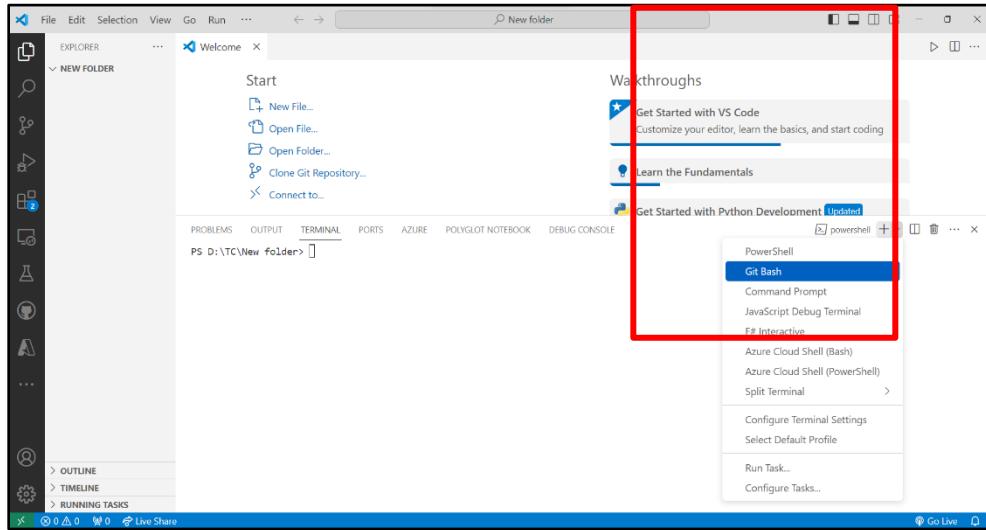


Figure 13.10: Select Git Bash

- 3) To configure Git:

- Set a username:

```
git config -- global user.name "YOUR_USERNAME"
```

- Set email address:

```
git config -- global user.email "YOUR_EMAIL"
```

- 4) Next, go to Azure DevOps Services (dev.azure.com) and sign in with the Account credentials. Select the project and the Project's Welcome Page appears. Then, click **Repos** and select files from the menu as shown in Figure 13.11.

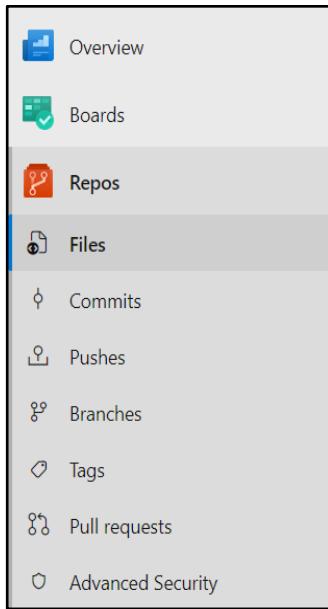


Figure 13.11: Repos Menu

- 5) In the **Files** view, choose **Clone** on the right to launch the **Clone Repository** pop-up window. Click **Generate Git Credentials** and copy the password. Then, click **Clone in VS Code** as shown in Figure 13.12.

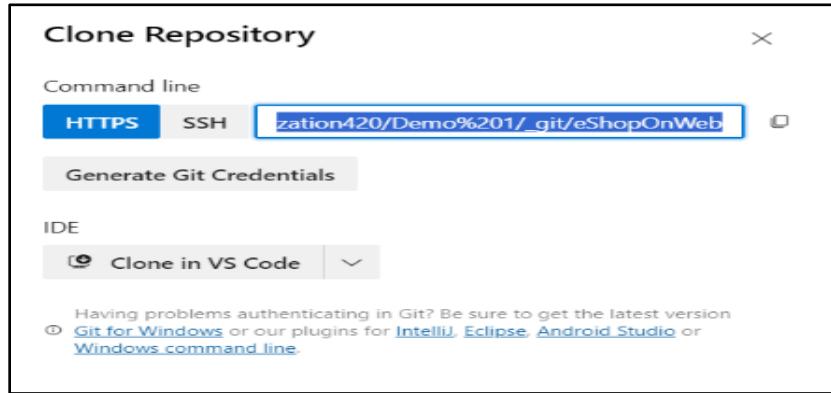


Figure 13.12: Clone Repository

A message box will appear as shown in Figure 13.13.

- 6) Click **Open Visual Studio Code**.

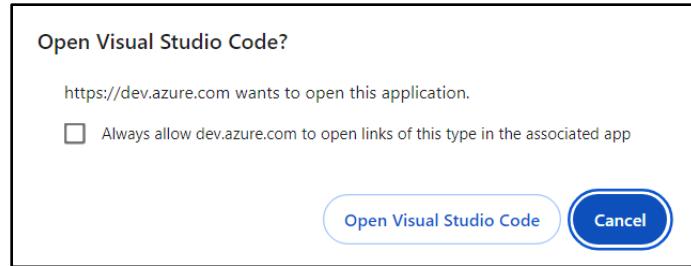


Figure 13.13: Message Box

- 7) Select the destination location where to clone and specify the password that was copied earlier so the process can be authenticated. The project cloning will be started. Once the cloning is completed, developers will be able to see the project files in their VS Code. They can then, work locally on those files and push them to Azure Devops later through commits.

13.1.6 Creating a Pipeline

To create a pipeline, perform following steps:

- 1) If the developer does not have an existing project, an existing template can be used that creates a demo project in Azure DevOps (Refer to section 13.1.4). Then, go to the Azure DevOps portal and click **Pipelines** from the menu. Pipelines section will appear as shown in Figure 13.14.

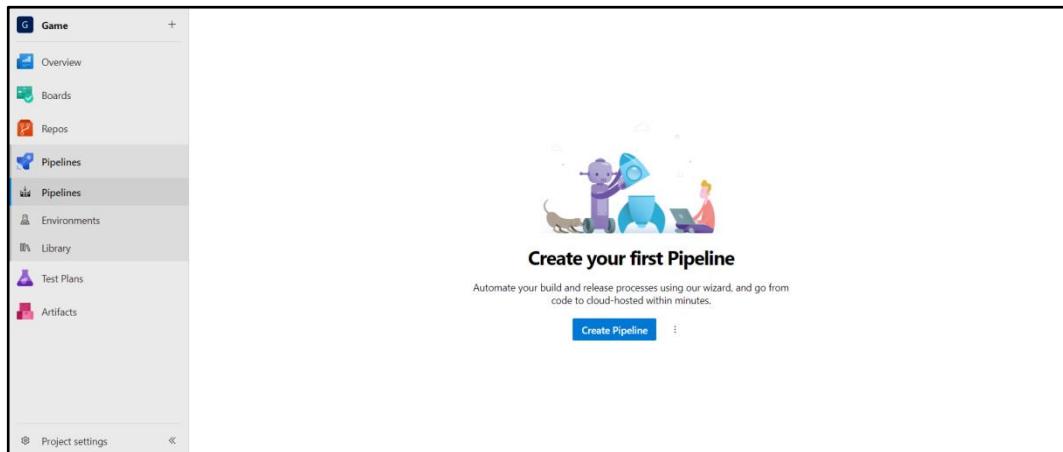


Figure 13.14: Pipelines Section

- 2) Click **Create Pipeline**, and select where to store the code (example, Azure Repo, Bitbucket Cloud, GitHub, and so on). Then, select **Azure Repos Git**. Refer to Figure 13.15.

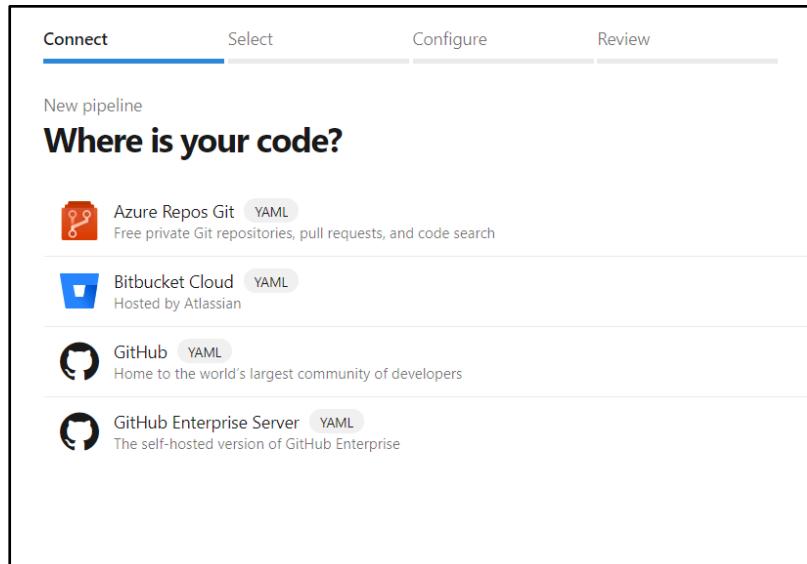


Figure 13.15: Connect to a New Pipeline

Developers will be asked to sign in to their GitHub account to authorize its use for Azure Pipelines.

- 3) Sign in to the GitHub account and authorize Pipelines. Developers will be shown a list of their existing repositories on GitHub.
- 4) Select the desired repository from the list. Refer to Figure 13.16.

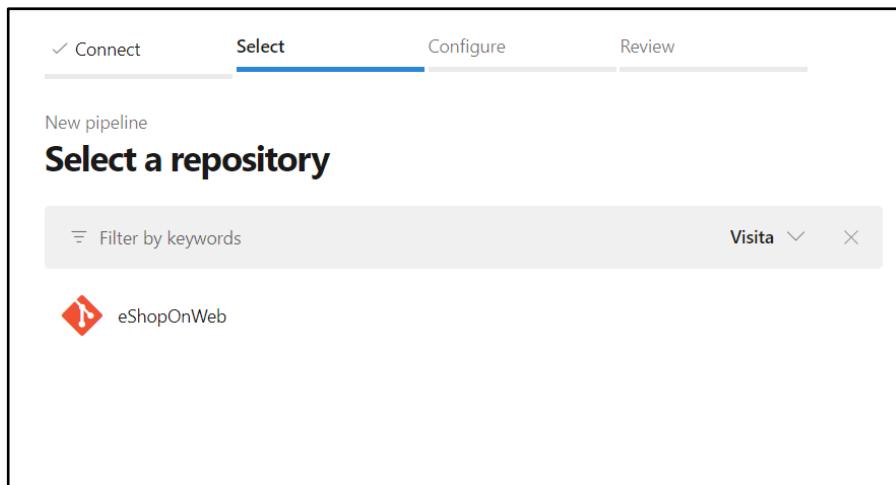


Figure 13.16: Select a Repository

- 5) In the **Configure your pipeline** section, choose a suitable configuration according to the project. Here, we select **ASP.NET Core (.NET Framework)**.

Refer to Figure 13.17.

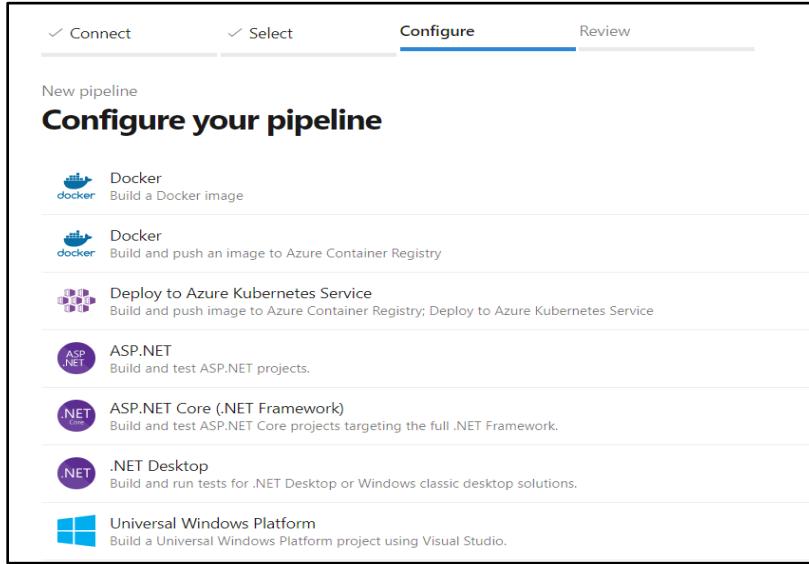


Figure 13.17: Configure Your Pipeline

- 6) In the Review tab, go to **Review your pipeline YAML** and click **Save and run**. Refer to Figure 13.18. Developers can optionally specify a commit message while clicking **Save and run**.

The screenshot shows the 'Review' tab of the pipeline configuration interface. The title is 'Review your pipeline YAML'. The code editor contains the following YAML pipeline definition:

```
trigger:
- main

pool:
  vmImage: 'windows-latest'

variables:
  solution: '**/*.sln'
  buildPlatform: 'Any CPU'
  buildConfiguration: 'Release'

steps:
  - task: NuGetToolInstaller@1
  - task: NuGetCommand@2
    inputs:
      restoreSolution: '$(solution)'

  - task: VSToolBuild@1
    inputs:
      msbuildArgs: '/p:DeployOnBuild=true /p:WebPublishMethod=Package /p:PackageAsSingleFile=true /p:SkipInvalidConfigurations=true /p:DesktopBuildPackageLocation="$(build.artifactStagingDirectory)\WebApp'
      platform: '$(buildPlatform)'
      configuration: '$(buildConfiguration)'

  - task: VSTest@2
    inputs:
      platform: '$(buildPlatform)'
      configuration: '$(buildConfiguration)'
```

At the top right, there are 'Variables' and 'Save and run' buttons. The 'Save and run' button is highlighted.

Figure 13.18: Review Your Pipeline

After running the pipeline, we get the final list of jobs as is shown in Figure 13.19. The pipeline is now created.

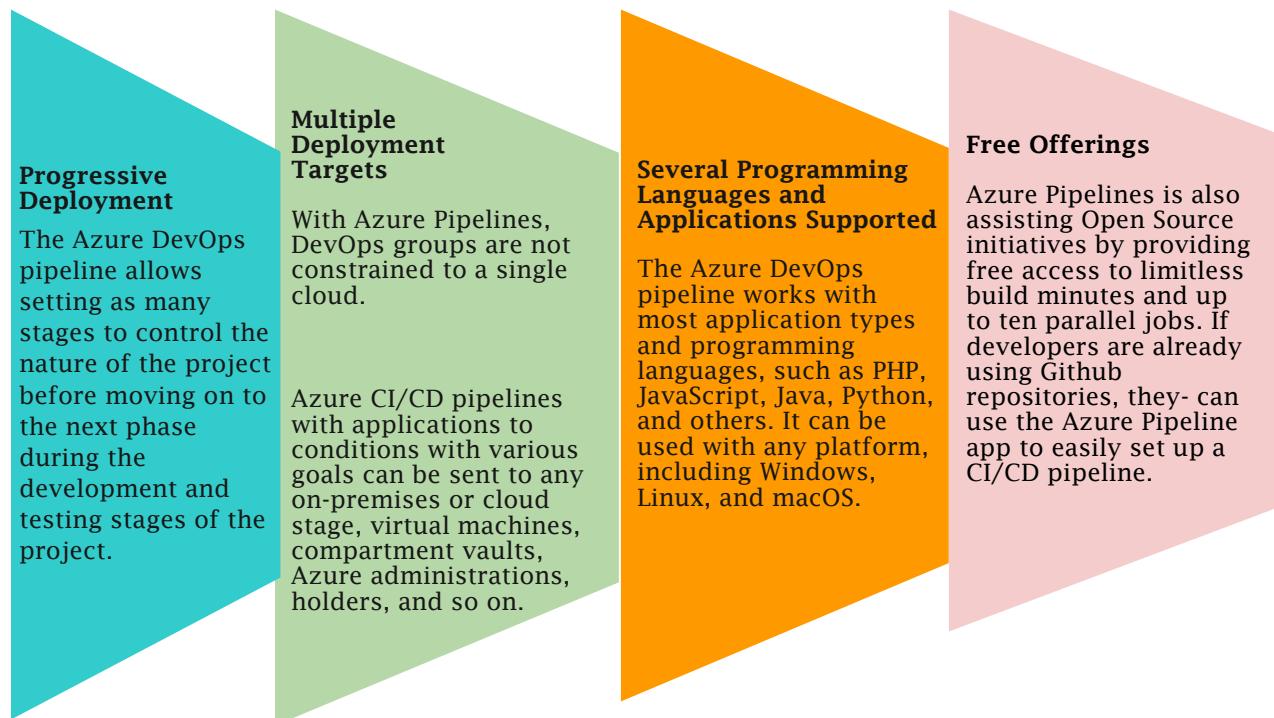
Job	
Pool: Azure Pipelines · Agent: Hosted Agent	
✓	Prepare job · succeeded
✓	Initialize job · succeeded
✓	Checkout · succeeded
✓	dotnet build Release · succeeded
✓	Post-job: Checkout · succeeded
✓	Finalize Job · succeeded

Figure 13.19: Job List

13.2 Advantages of Azure Pipelines

Azure Pipelines is the highest quality solution for DevOps that offers CI/CD solution regardless of any language, application, or platform.

Here are some key features of Azure pipelines:



13.3 Releases

In Azure Pipelines, release pipelines help teams continuously provide software to clients at a faster rate and with less risk. One can fully automate the testing and delivery of software in various stages all the way to production. Alternatively, one can just set up semi-automated processes with approvals and on-demand deployments.

The pipelines, stages, tasks, releases, and deployment data are saved via release pipelines in Azure Pipelines.

Step-by-step process to create a release pipeline is described as follows:

- 1) In the Azure DevOps, go to the project then, click **Pipelines** from the menu. Then, click **Release**. One will be navigated to the pipelines section as shown in Figure 13.20.

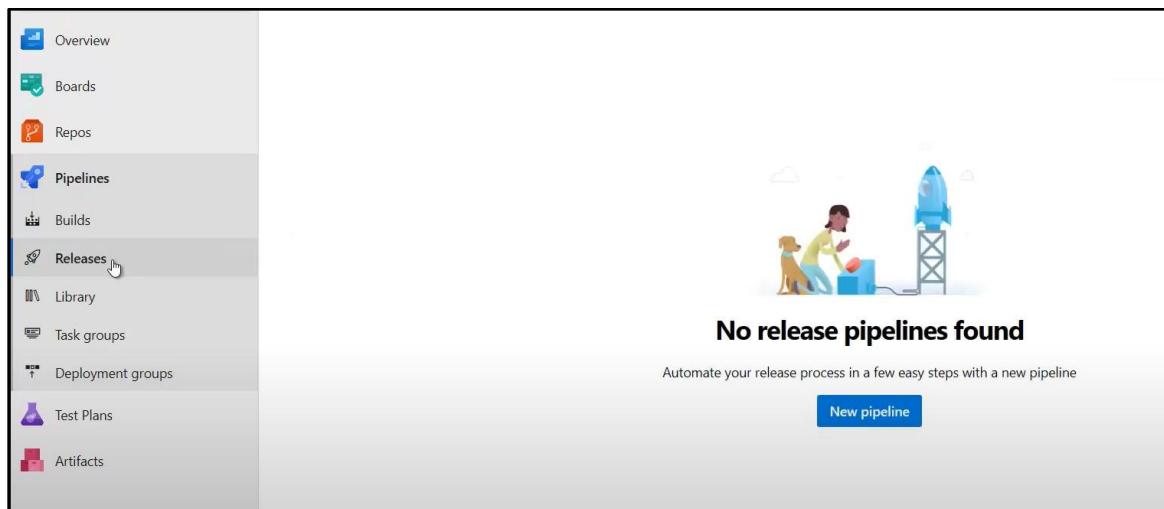


Figure 13.20: Release Pipelines Section

- 2) Click **New Pipeline** and **Select a Template** dialog box will appear as shown in Figure 13.21.

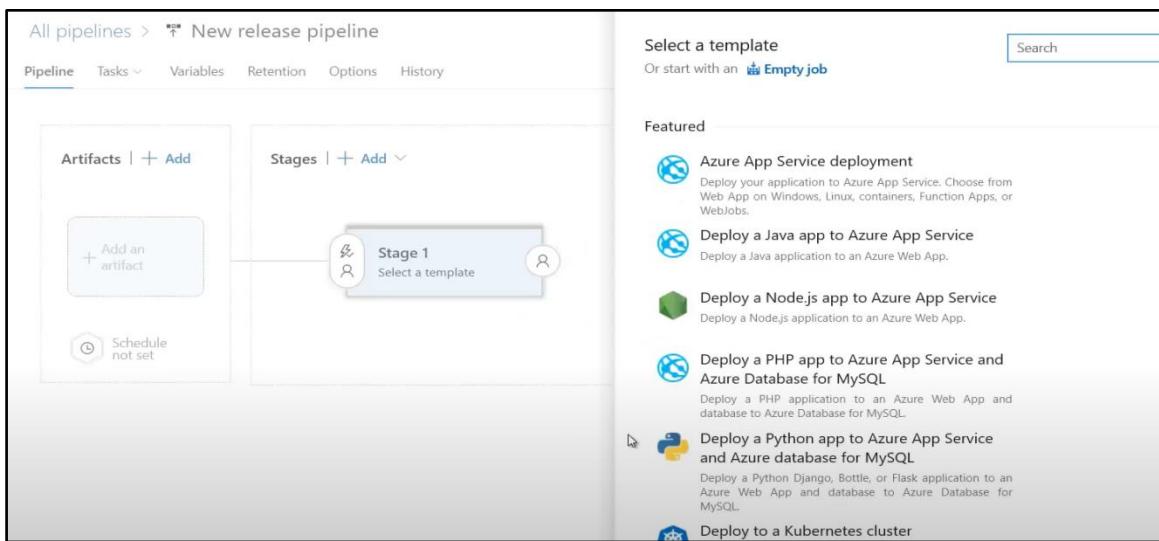


Figure 13.21: Create a New Pipeline

- 3) Click **Artifact**. A pop-up window **Add an artifact** will appear. Select the Source type according to the project. Then, select Source (build pipeline) and click **Add**. Refer to Figure 13.22.

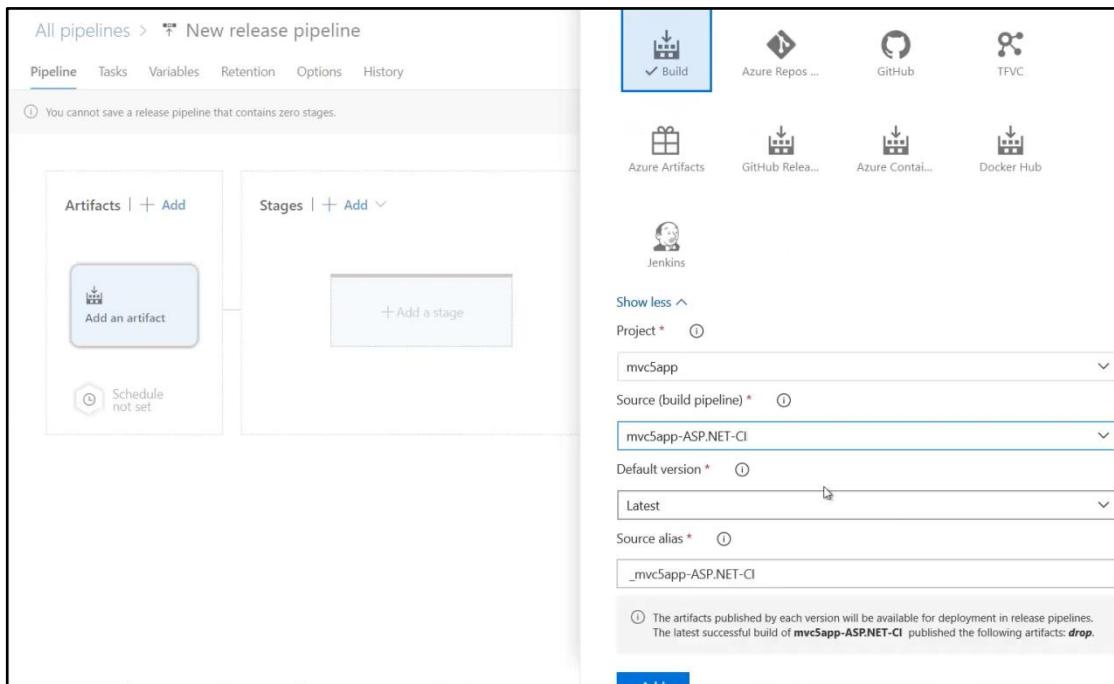


Figure 13.22: Add an Artifact

- 4) Click **Add a Stage** and a pop-up window **Select a template** will appear. Select a template according to the project application and click **Apply** as shown in Figure 13.23.

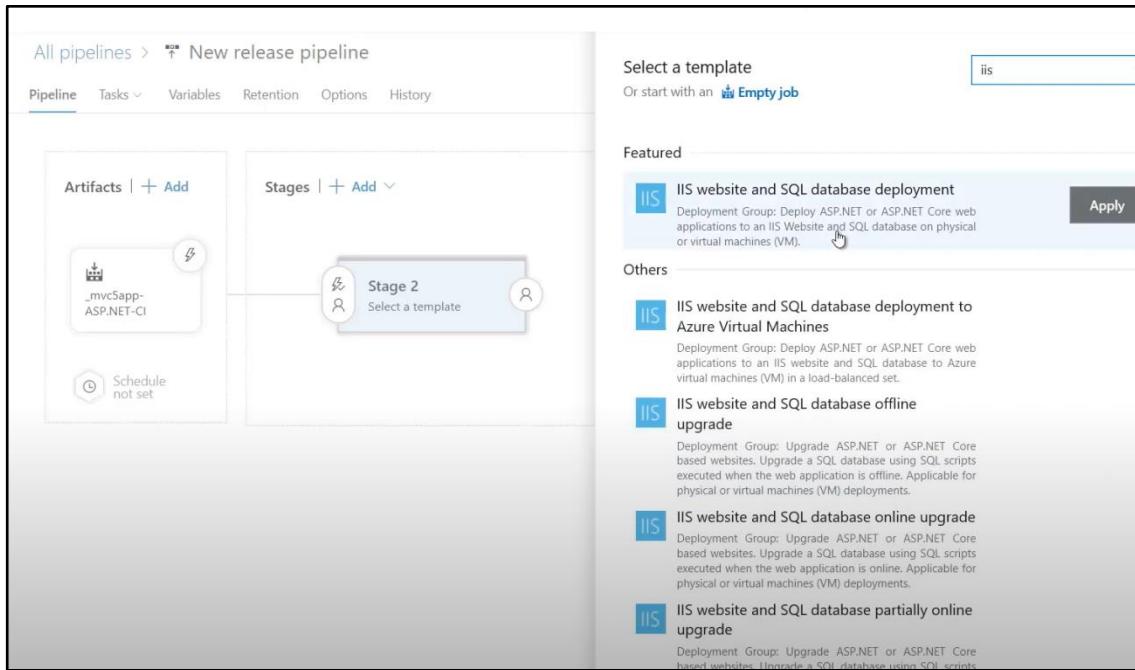


Figure 13.23: Select a Template

- 5) Click **Dev** in **Stages**. Refer to Figure 13.24.

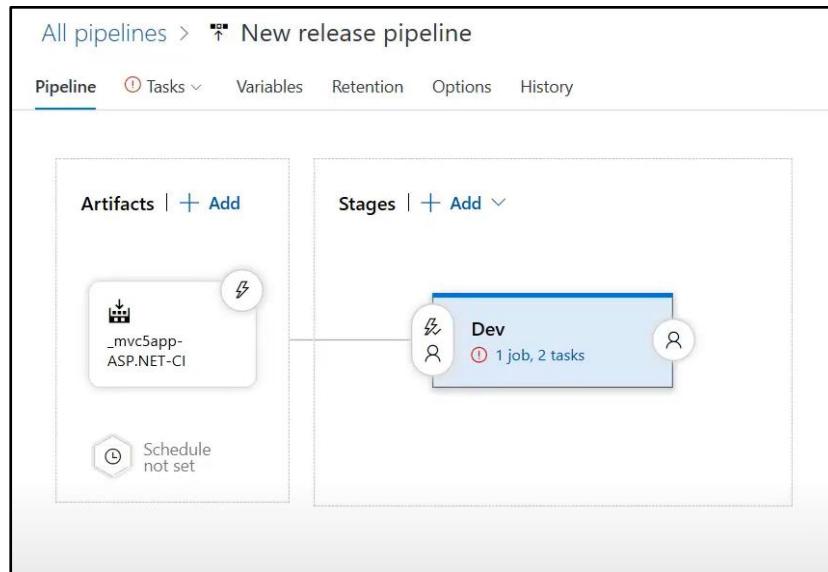


Figure 13.24: Stage

- 6) In the stage, select a name, set parameters, set action, give a Website Name, and add binding. Refer to Figure 13.25.

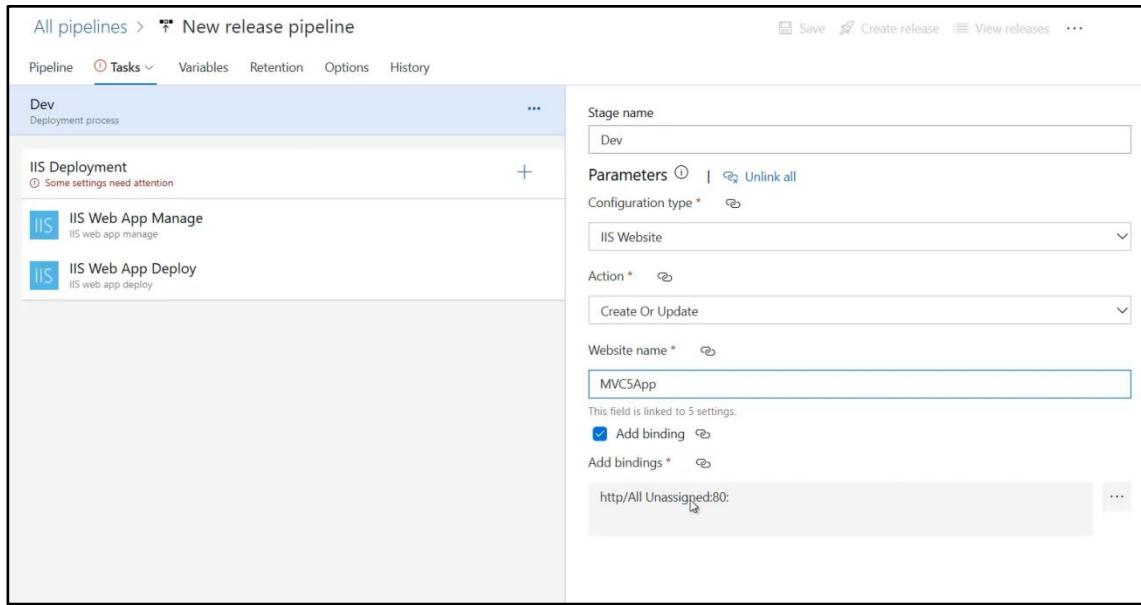


Figure 13.25: Stage Configuration

- 7) Click **IIS Deployment**. A **Deployment group job** pop-up window will appear. In this section, select a Display name, and Deployment group. Set the Timeout and job cancel timeout and then, click **Artifact download latest version** as shown in Figures 13.26A and 13.26B.

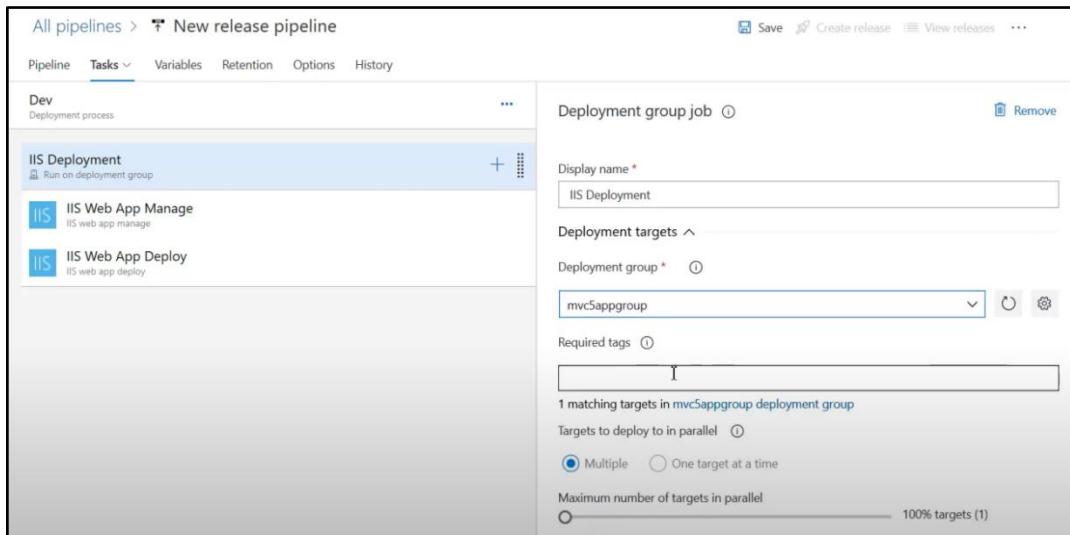


Figure 13.26A: Deployment Group Job 1

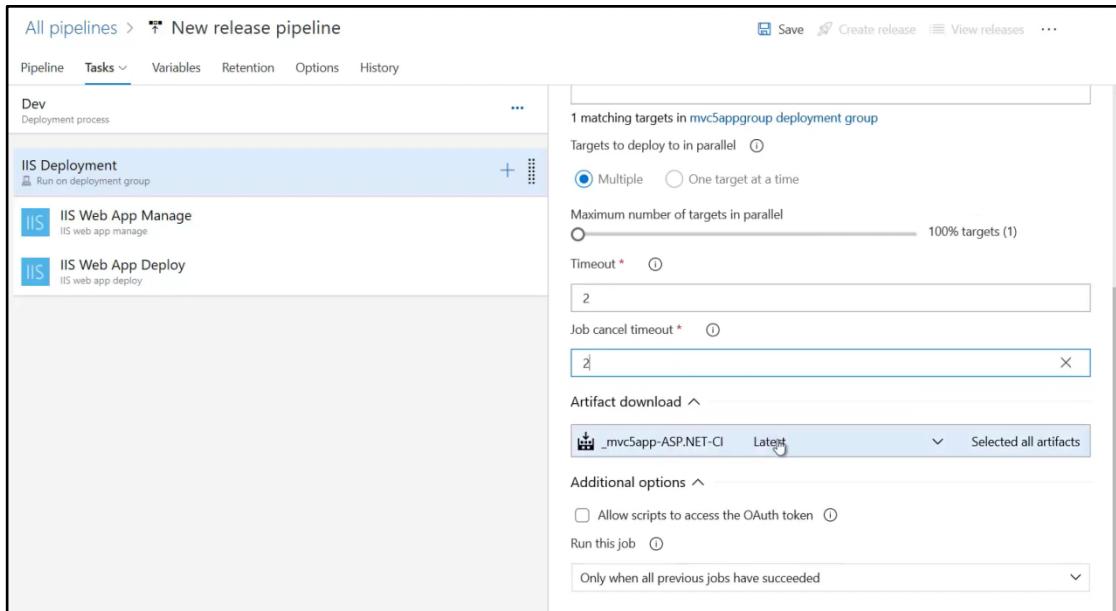


Figure 13.26B: Setting TimeOut

- 8) Click **+** in the IIS Deployment for adding new tasks to the release pipeline based on the requirement as shown in Figure 13.27.

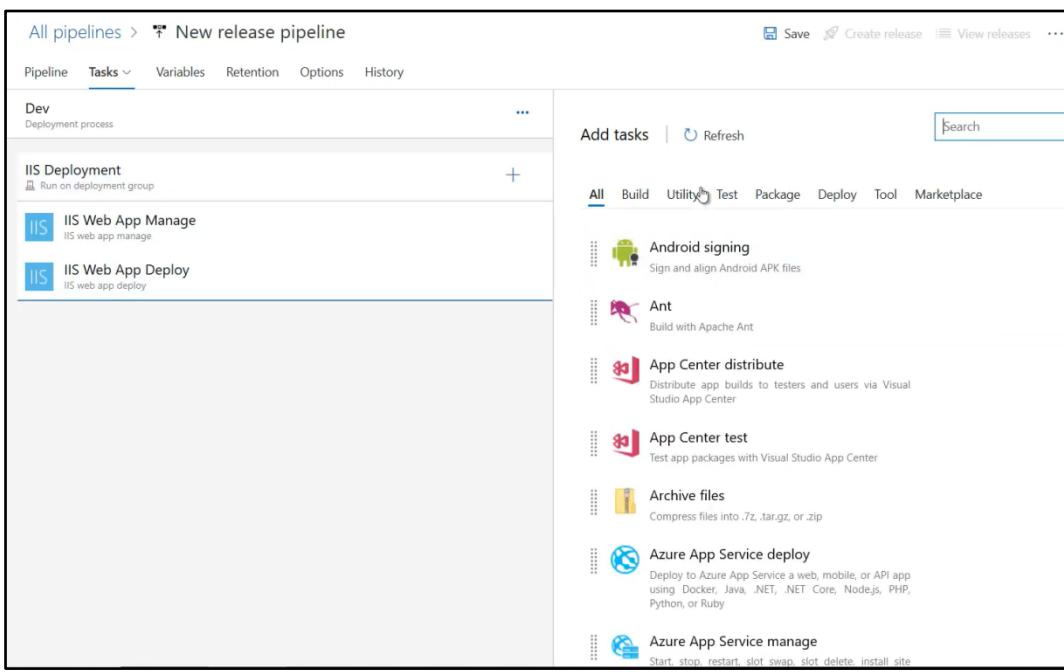


Figure 13.27: Add Tasks

- 9) Click **Variables** and a variable section will appear. Here, one can create a variable according to their requirement. Refer to Figure 13.28.

Figure 13.28: Variable Section

- 10) Click **Retention**, and a retention section will appear. Here, one can set days to retain a release, and minimum release. Refer to Figure 13.29.

Figure 13.29: Retention Section

- 11) Click **Save** and select the folder. Write a comment regarding the pipeline as shown in Figure 13.30.

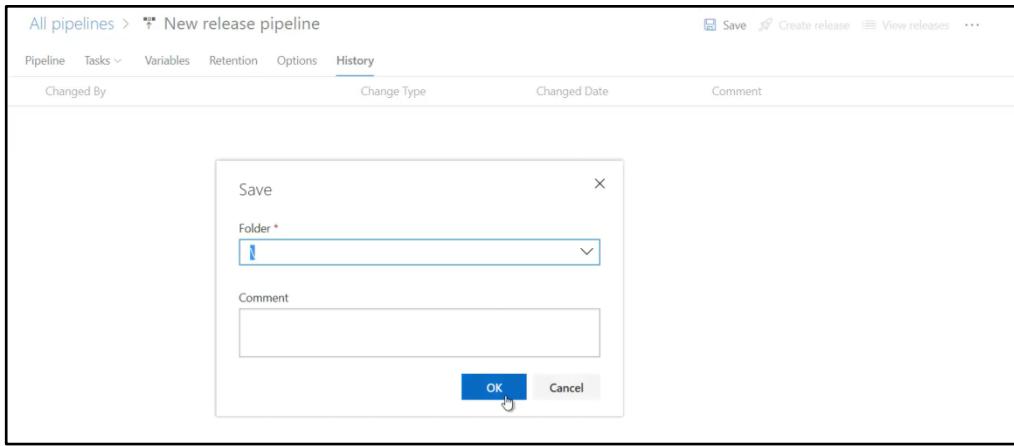


Figure 13.30: Save Your Pipeline

- 12) Click **Create release** and a pop-up window to create a new release will appear. Refer to Figure 13.31.

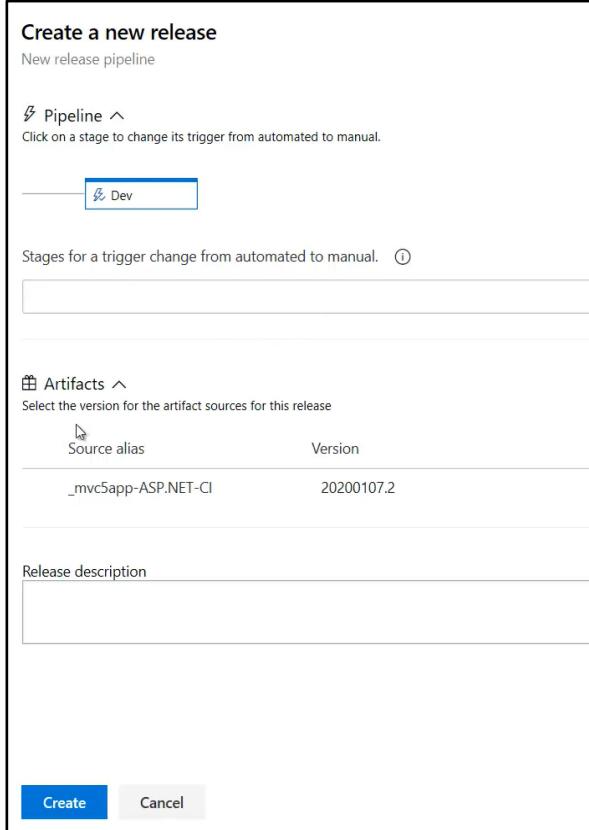


Figure 13.31: Create a New Release

- 13) Click **Create** and the pipeline will be released as shown in Figure 13.32.

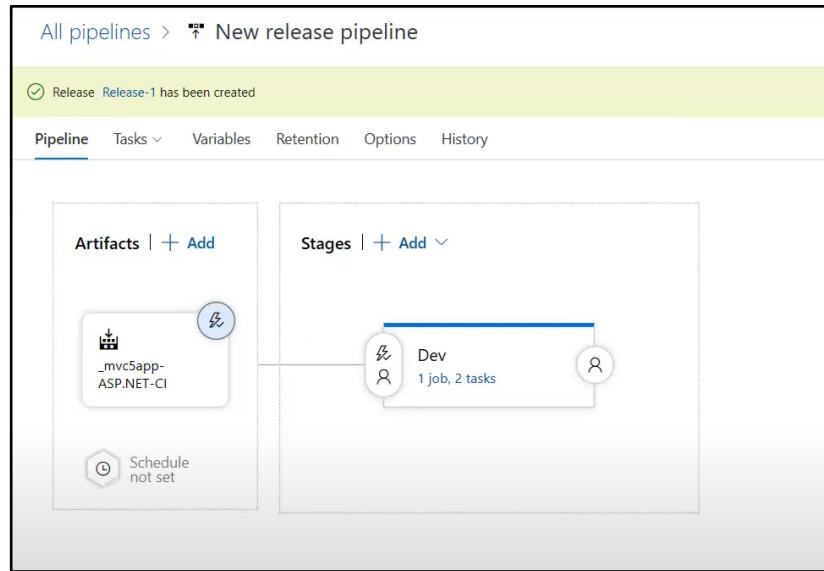


Figure 13.32: Pipeline Release

13.4 Integrating Azure DevOps with LambdaTest

The integration of LambdaTest with Azure DevOps allows developers to create a work item directly from the LambdaTest platform. It allows adding a bug, epic, task, or story to the project at any time, even in the middle of a test session. The fields developed by professionals when marking as a bug with LambdaTest appear as information on the work item in the Azure DevOps project for that testing instance.

Developers can follow these steps to integrate Azure DevOps into the LambdaTest account:

Step 1: Login to the LambdaTest account. Developer must have admin or user-level access to see and install integrations.

Step 2: From the left navigation menu bar, select **Settings** and click **Integrations**. This will show a screen with a list of third-party applications that developers can integrate with their LambdaTest account. Then, click **Connect**, under the block that says **Azure DevOps** as shown in Figure 13.33.

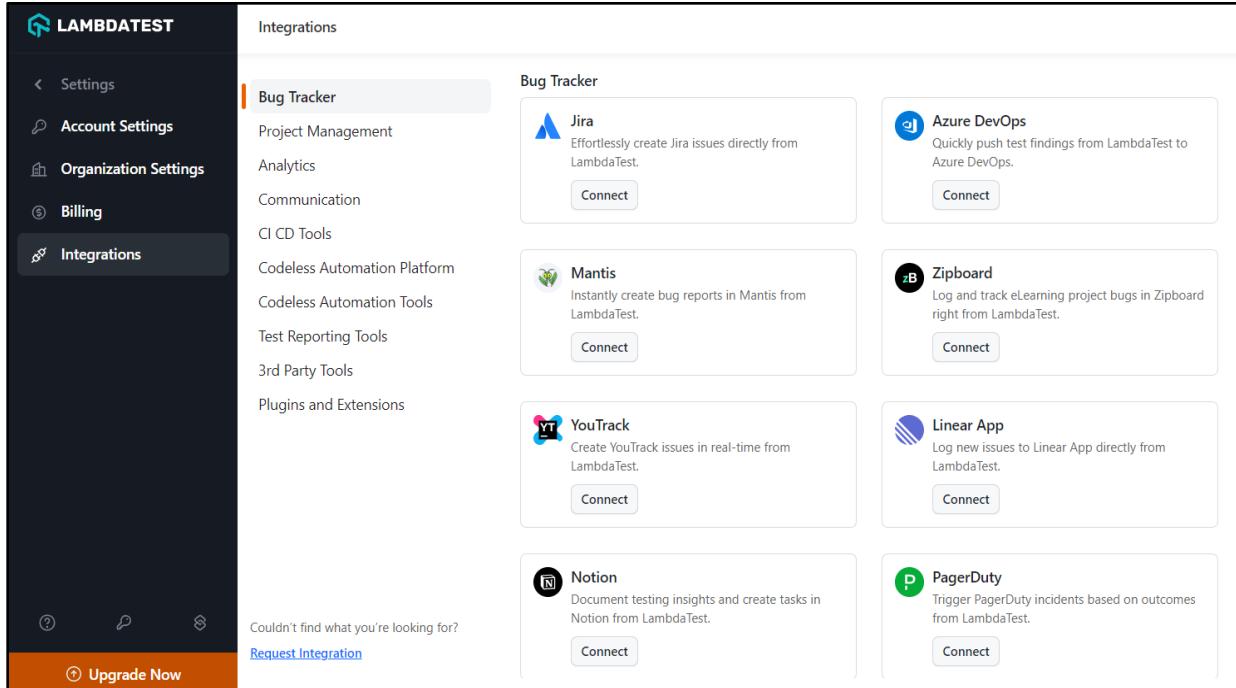


Figure 13.33: Integration of Azure DevOps

Step 3: When the developer clicks **INSTALL**, a pop-up window is shown where they must enter the Azure DevOps URL, and Azure DevOps Access Token as shown in Figure 13.34. The developer will be granting LambdaTest access to their user-owned resources on Microsoft's Visual Studio Team Services account by doing so.

This is required for authorization between APIs from two different applications.

Azure DevOps URL will be the domain name. Once the developers log into Azure DevOps account, they will get the URL under their organization.

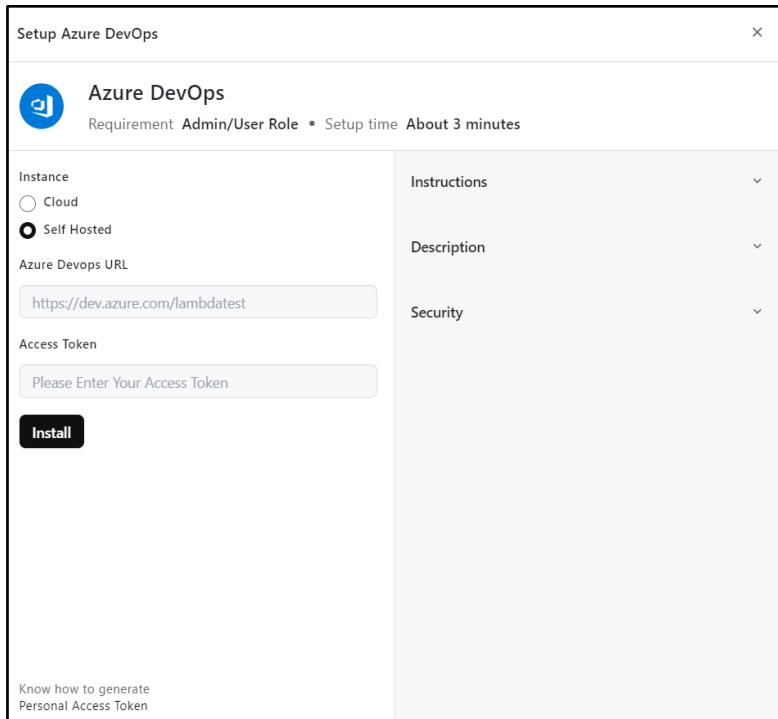


Figure 13.34: Set up Azure DevOps

Step 5: Developers can see it in the URL if they are currently in a project in their Azure DevOps account.

Azure DevOps Access Token

Access tokens are keyed strings required to access an API. They are issued to the client server and are generally opaque. Access to secured, user-specific resources is requested using access tokens. Access tokens are crucial from a security point of view and can be created in different formats, depending upon security requirements specified on the resource server.

Developers can generate their Azure DevOps access token under 'Personal access tokens'. This can be done by clicking the **settings icon** next to the top right corner where the user avatar is displayed.

Step 6: After clicking **+ New Token**, provide the access token a name. Set the parameters for access authorization as shown in Figure 13.35. By doing this, developer will grant any third- party API the degree of access required to grant it. Developer must choose the option button for Full access and click **Create** to integrate LambdaTest with Azure DevOps.

Create a new personal access token

Name
lamdatest

Organization
All accessible organizations

Expiration (UTC)
30 days 3/31/2024

Scopes
Authorize the scope of access associated with this token
Scopes Full access
 Custom defined

Create **Cancel**

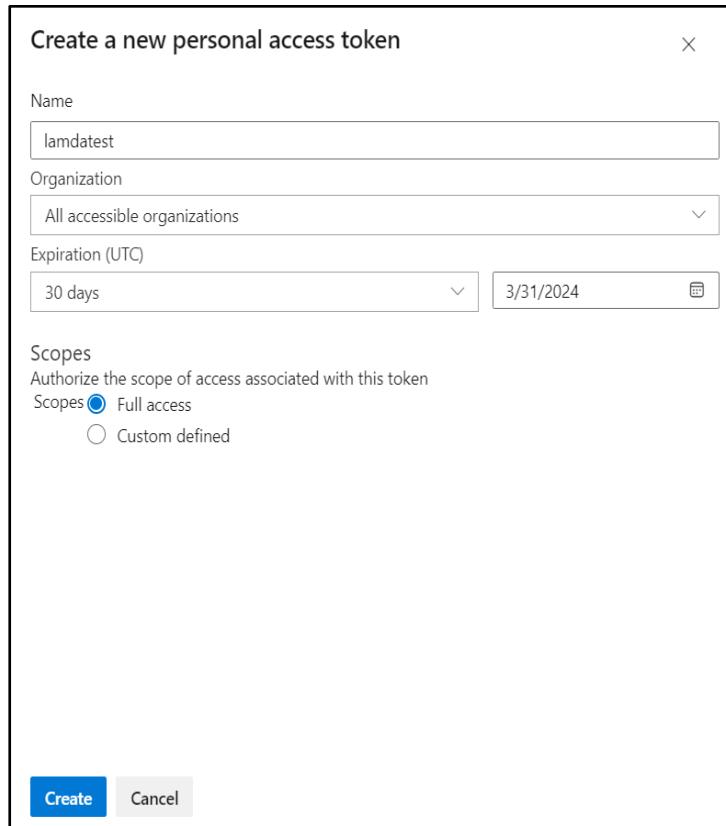


Figure 13.35: Create New Personal Access Token

When a token is created, copy it to clipboard and keep it within a secure location.

Access tokens must not be lost or fall into the wrong hands because they are just as crucial as passwords. One can always cancel the token and produce a new one if one loses it in the future or if it is shared with someone not trustworthy. Click any personal access token and then, select **Revoke** to cancel the token.

Step 7: In the space provided, enter Azure DevOps URL, and access token. Then, click **Install**.

13.5 Azure Artifacts

Azure Artifacts, a service within Azure DevOps, simplifies package management for developers. It generates, hosts, and distributes packages within a team. One can seamlessly integrate artifacts into CI/CD pipelines with just one click.

13.5.1 Connect to Feed in Azure Artifacts

The step-by-step process to connect to feed is described as follows:

- 1) In Azure DevOps, go to the project and click **Artifacts**. Then, an artifacts section project will appear as shown in Figure 13.36.

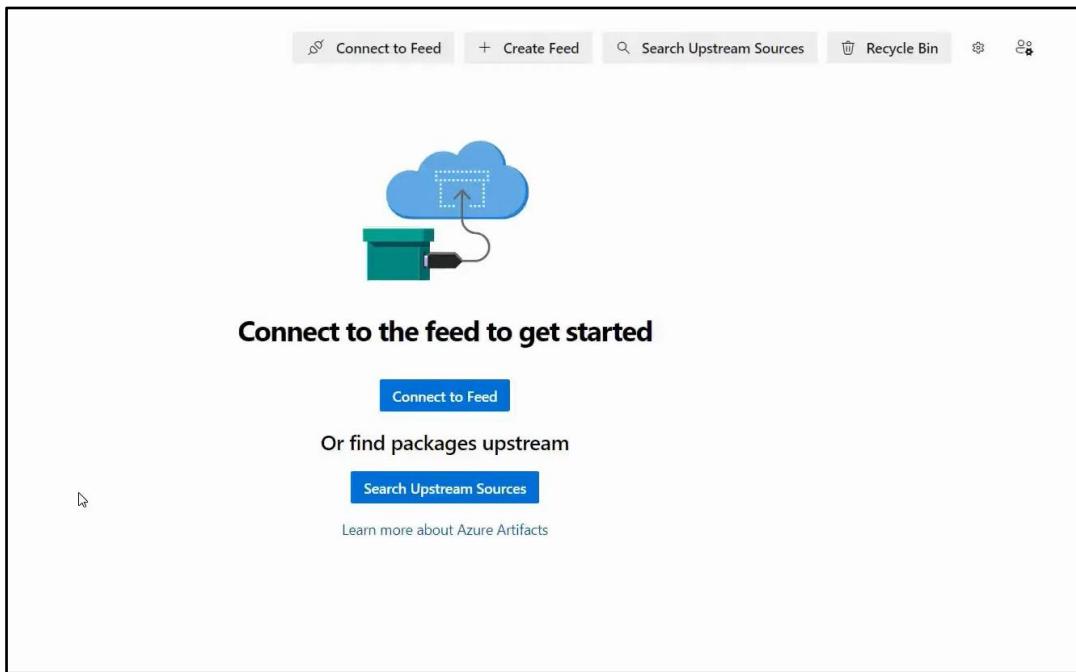


Figure 13.36: Artifacts Section

- 2) Click **Create feed**. Enter the name of the feed, select the visibility, and then, click **Create**. Refer to Figure 13.37.

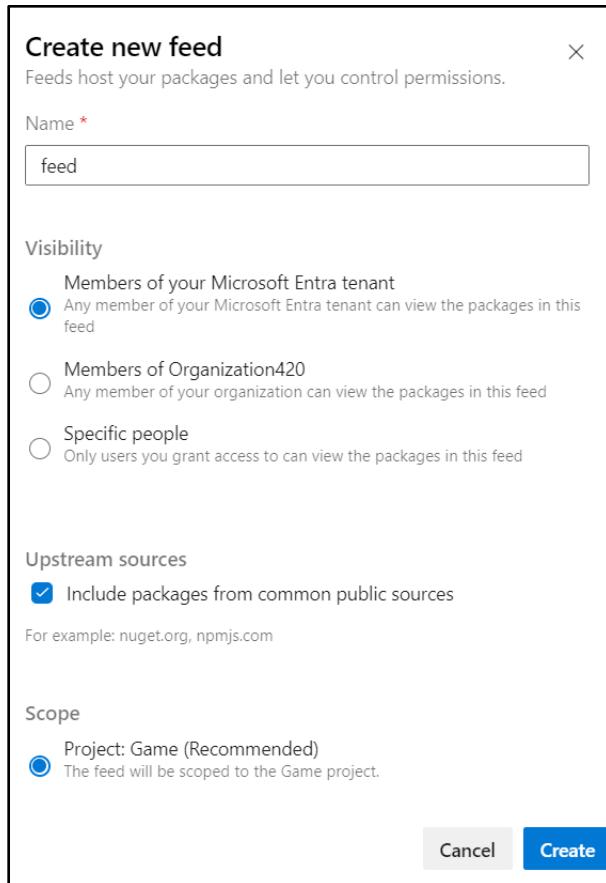


Figure 13.37: Create New Feed

- 3) Click **Connect to Feed** and select a feed according to the project. Then, include the script in the project code. Refer to Figure 13.38.

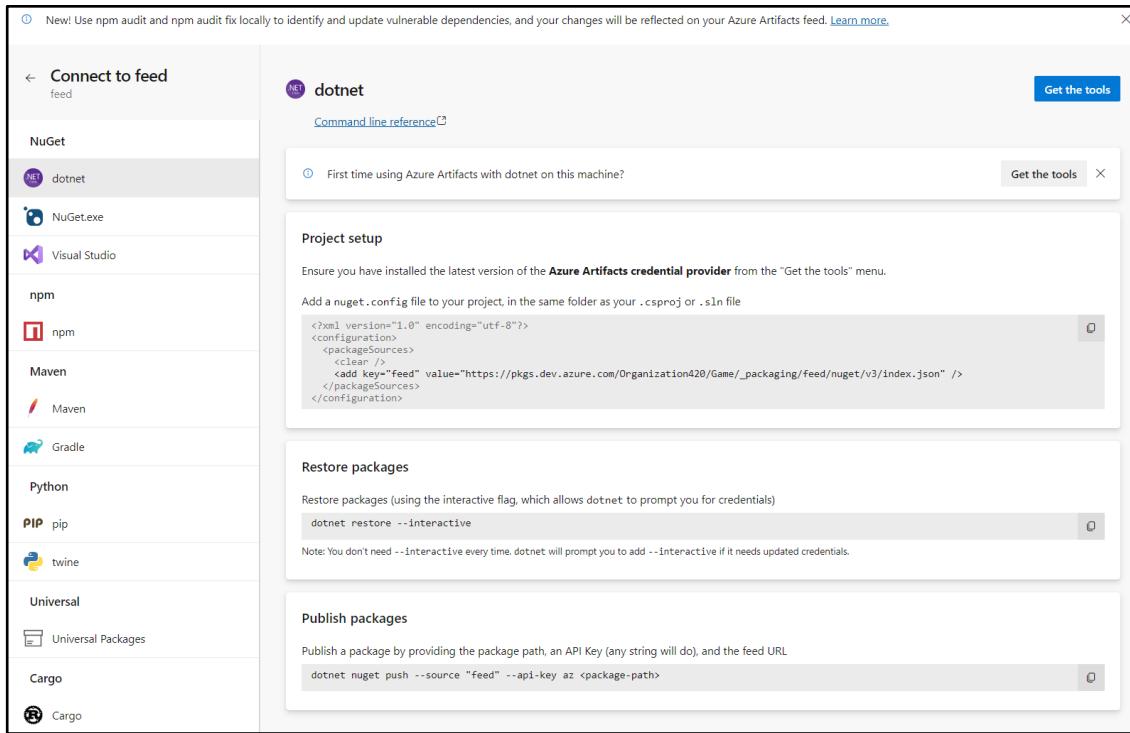


Figure 13.38: Connect to Feed

13.5.2 Roles and Importance of Azure Artifacts

Roles and importance of Azure artifacts are as follows:

1. Dependency Management: Azure Artifacts simplifies the management of dependencies by providing a centralized location for developers to handle all their libraries, packages, or other components. This consolidation helps teams prevent duplication, ensure uniformity, and streamline their development processes.

2. Package Hosting and Distribution:

- Developers can easily publish packages to designated feeds within Azure Artifacts, serving as repositories for various types of packages.
- These packages can be shared within teams, across different organizations, or even publicly on the Internet.
- Supported package types encompass NuGet, npm, Python, Maven, Cargo, and Universal Packages.

3. Package Consumption:

- Azure Artifacts empowers developers to retrieve packages from diverse feeds and public registries.

- This adaptability enables projects to capitalize on existing packages and avoid unnecessary duplication of efforts.

4. Consistency and Efficiency:

- Artifacts ensures the consistency of components by maintaining their immutability once published, ensuring uniformity across builds and deployments.
- The platform also prioritizes performance, ensuring efficient retrieval and resolution of packages.

13.6 Infrastructure as Code (IaC) Concepts

Infrastructure as Code (IaC) involves configuring infrastructure through code rather than manually. Manual processes require operators and system administrators to manually configure any infrastructure changes.

With IaC, Developers teams can store both the infrastructure configuration code and application code in a centralized repository. This approach ensures consistent and more secure deployments. By avoiding error-prone manual configuration and deployment, maintaining security standards and policies becomes easier. Additionally, Developers can enhance scalability and productivity through quicker deployments.

When implementing IaC, developers have a variety of tools for Microsoft Azure such as Azure Resource Manager (ARM) templates, HashiCorp Terraform and any other declarative scripting languages.

13.6.1 Approach of implementing IaC

There are two approaches to consider when implementing IaC:

1. Imperative Infrastructure as Code:

- This approach involves writing scripts in languages such as Bash or PowerShell. Developers explicitly specify commands to be executed to achieve a desired outcome.
- With imperative deployments, developers are responsible for managing the sequence of dependencies, error control, and updates to resources.

2. Declarative Infrastructure as Code:

- With this approach, developers define how they want their environment to appear. They specify the desired outcome rather than the specific steps to achieve it.

- The tooling used determines how to achieve the desired outcome by examining the current state, comparing it to the desired state, and then, applying the necessary changes.

13.6.2 ARM Template

For implementing IaC in their Azure solutions, developers can utilize Azure Resource Manager templates (ARM templates). These templates are JavaScript Object Notation (JSON) files that outline the infrastructure and configuration for a project. Using declarative syntax, ARM templates allow developers to specify what they want to deploy without requiring to write the sequence of programming commands to create it. Within the template, developers define the resources to deploy and their respective properties.

A blank ARM template is shown in Code Snippet 1.

Code Snippet 1:

```
{
  "$schema":
"https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
{
  "$schema": "https://schema.management.azure.com/schemas/2019-
04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
  },
  "variables": {
  },
  "functions": [],
  "resources": [
  ],
  "outputs": {
  }
}
```

The blank template of ARM template is structured into following sections:

- **Parameters:** These allow developers to provide values during deployment, enabling the template to be used across different environments.
- **Variables:** These define values that are reused within templates and can be constructed from parameter values.
- **User-defined functions:** These enable developers to create custom functions that simplify the template.
- **Resources:** This section specifies the resources to be deployed.
- **Outputs:** This section returns values from the deployed resources.

13.6.3 Terraform

Terraform is a paid templating tool capable of provisioning cloud-native applications on leading cloud platforms such as Azure, Google Cloud Platform, AWS, and AliCloud. Unlike JSON, it employs HashiCorp Configuration Language (HCL), which is more concise for defining templates.

A terraform sample template is shown in Code Snippet 2.

Code Snippet 2:

```
resource "azurerm_resource_group" "example" {
  name = "example-resources"
  location = "West Europe"
}
resource "azurerm_template_deployment" "example" {
  name = "acctesttemplate-01"
  resource_group_name = azurerm_resource_group.example.name
  template_body = <<DEPLOY
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountType": {
      "type": "string",
      "defaultValue": "Standard_LRS",
      "allowedValues": [
        "Standard_LRS",
        "Standard_GRS",
        "Standard_ZRS"
      ]
    }
  }
}
```

```

        ],
      "metadata": {
        "description": "Storage Account type"
      }
    }
  },
  "variables": {
    "location": "[resourceGroup().location]",
    "storageAccountName": "[concat(uniquestring(resourceGroup().id), 'storage')]",
    "publicIPAddressName": "[concat('myPublicIp',
uniquestring(resourceGroup().id))]",
    "publicIPAddressType": "Dynamic",
    "apiVersion": "2015-06-15",
    "dnsLabelPrefix": "terraform-acctest"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "name": "[variables('storageAccountName')]",
      "apiVersion": "[variables('apiVersion')]",
      "location": "[variables('location')]",
      "properties": {
        "accountType": "[parameters('storageAccountType')]"
      }
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "[variables('apiVersion')]",
      "name": "[variables('publicIPAddressName')]",
      "location": "[variables('location')]",
      "properties": {
        "publicIPAllocationMethod": "[variables('publicIPAllocationMethod')]"
      }
    }
  ],
  "outputs": {

```

```
"storageAccountName": {
    "type": "string",
    "value": "[variables('storageAccountName')]"
}
}

DEPLOY
# these key-value pairs are passed into the ARM Template's
`parameters` block
parameters = {
    "storageAccountType" = "Standard_GRS"
}

deployment_mode = "Incremental"
}
output "storageAccountName" {
    value=
azurerm_template_deployment.example.outputs["storageAccountName"]
}
```

13.7 Summary

- ✓ Continuous Integration/Continuous Delivery (CI/CD) is a way to regularly supply apps to clients by introducing automation into the tiers of app improvement.
- ✓ A CI/CD pipeline introduces ongoing automation and non-stop tracking during the lifecycle of apps, from integration and testing phases to delivery and deployment.
- ✓ Azure DevOps Starter produces a CI/CD pipeline within Azure Pipelines. Azure DevOps Starter configures a CI/CD pipeline in Azure Pipelines on its own.
- ✓ In Azure Pipelines, release pipelines help teams continuously provide software to clients at a faster rate and with less risk.
- ✓ The integration of LambdaTest with Azure DevOps allows developers to create a work item directly from the LambdaTest platform.
- ✓ Azure Artifacts is used to generate, host, and distribute packages within your team.
- ✓ Infrastructure as Code (IaC) is a key DevOps practice that involves the management of infrastructure, such as networks, compute services, databases, storages, and connection topology, in a descriptive model.

13.8 Test Your Knowledge

1. Which of the following key stages are not applicable in CI/CD pipeline?
 - A. Plan
 - B. Build
 - C. Deploy
 - D. Deliver
2. Which one of the following tools supports building, deploying, and automating software development projects?
 - A. Selenium
 - B. Jenkin
 - C. TestRail
 - D. LambdaTest
3. The Azure DevOps pipeline works with which of these?
 - A. Only PHP
 - B. Only JavaScript
 - C. Only Linux
 - D. All of these
4. Which of these statements about Azure Pipelines release are not true?
 - A. Help teams continuously provide software to clients at a faster rate and with less risk.
 - B. Automate software delivery in various stages
 - C. Automate testing process
 - D. Increase software delivery risks in various stages
5. Access tokens are _____.
 - A. CI/CD strings
 - B. Coded security system
 - C. Keyed strings
 - D. Code pass strings
6. What does IaC stand for in DevOps?
 - A. Infrastructure as Computer
 - B. Integration and Configuration
 - C. Infrastructure as Code
 - D. Input and Compute

13.8.1 Answers to Test Your Knowledge

1. A
2. B
3. D
4. D
5. C
6. C

Try It Yourself

1. Create an Azure DevOps organization.
2. Create a DevOps project.
3. Create a Pipeline.
4. Create a release pipeline.
5. Integrate Azure DevOps into LambdaTest account.
6. Perform Real time testing of your project.
7. Explore the other testing which is present in LambdaTest platform.



SESSION 14

IMPLEMENTING TRAFFIC MANAGEMENT AND MONITORING STRATEGIES FOR WEB SERVICES

Overview

This session provides an introduction to Azure load balancer and its components and features. It elaborates on the working of Azure Application Gateway and Traffic Manager. It then explains about Azure Application Insights, Log Analytics, Azure Event Hubs, and Stream Analytics. The session also discusses Azure Front Door. It concludes with the setup and configuration process of Azure Monitor and Application Insights.

Learning Objectives

In this session, students will learn to:

- Explain Azure load balancer and its components
- Illustrate how to work with Azure Application Gateway and Traffic Manager
- Explain Azure Application Insights and Log Analytics
- Illustrate how to configure Application Insights
- Explain Azure Event Hubs and Stream Analytics
- Explain Azure Front Door
- Illustrate how to set up and configure the Azure Monitor

14.1 Introduction to Azure Load Balancer

Azure load balancer can be defined as a tool that helps developers to scale their applications, thus, providing high availability for their services.

Some of its features are as follows:

- Suitable for both inbound and outbound scenarios
- Offers low latency
- Offers high throughput
- Scales a lot of flows for all TCP and UDP applications

Load balancer accepts inbound flows at its frontend and transmits them to backend pool instances depending on certain predefined rules. Furthermore, developers can use a public load balancer to create outbound networks for VMs within their virtual network. To achieve this, they just have to convert their existing private IP addresses to public ones.

Load balancer helps developers to perform the following:

- ❖ Load balancing of inbound Internet traffic to their VMs, which is called as a public load balancer
- ❖ Load balancing of traffic across VMs within a virtual network. Developers can go to the frontend of a load balancer from a given network in instances, such as a hybrid scenario
- ❖ Forwarding port traffic to the destined port on specific VMs with inbound rules of Network Address Translation (NAT)
- ❖ Giving outgoing connectivity for VMs within the virtual network by employing a public load balancer

Figure 14.1 shows how the load balancer can distribute traffic by implementing a hash-based algorithm.

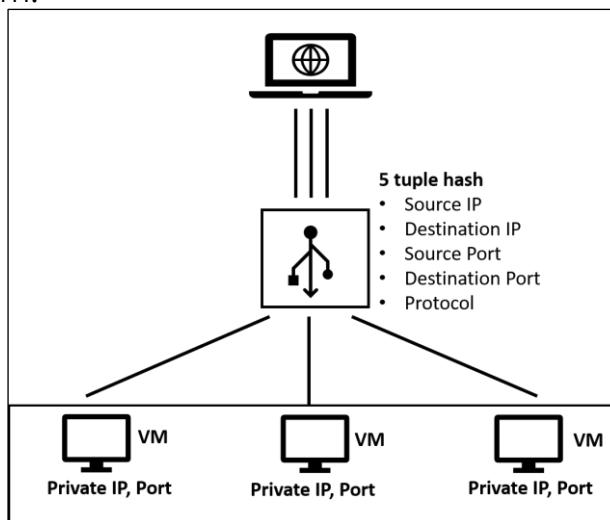


Figure 14.1: Hash-based Traffic Distribution by Azure Load Balancer

A developer can define a load-balancing rule to disseminate traffic at the frontend to instances of backend pools. While doing so, it implements a hash-based algorithm, as shown in Figure 14.1. A server obtains the new hash traffic when the output of a query implies a reliable backend endpoint. By default, a 5-tuple hash featuring a blend of source and destination IP addresses, source and destination ports, and IP protocol number reaches the available servers.

All packets belonging to the same traffic flow reach the same backend instance existing at the rear of the load-balanced front end. If a new flow is triggered from the same source IP, it results in a modified source due to which the 5-tuple hash algorithm may push the traffic to some other backend endpoint.

Following are the capabilities of the load balancer for applications implementing Transmission Control Protocol (TCP) and User Datagram Protocol (UDP):

Load balancing	Developers can develop custom rules to accept inbound traffic and allocate them to backend pool instances. To perform this, load balancer employs an in-built hash-based algorithm and the headers as per requirements.
Automatic reconfiguration	The load balancer can help to automatically upscale or downscale.
Port forwarding	Developers can generate an inbound NAT rule for a virtual network to port advancing traffic from a particular port of a certain frontend IP address to a required port of a certain backend example. Thereafter, this is followed by hash-based allocation.
Application agnostic and transparent	Though a load balancer can work with any TCP or UDP application instance, it does not communicate with them directly. It will not even cease flows or perform any task with respect to application layer gateway. The communication always takes place without an intermediary between client involved and backend pool.
Health probes	The load balancer utilizes health probes to ascertain the health of backend pool instances. In case there is no response from a probe, it ceases to transmit any new requests to that instance. It is important to note that this does not impact the prevailing connections.

14.2 Working with Azure Application Gateway and Traffic Manager

The Azure application gateway can be defined as a load balancer suitable for handling incoming Web traffic and targeting it towards the Web applications. Generally, load balancers function atop the transport layer, which is OSI layer 4, involving TCP and UDP. They distribute the incoming traffic depending on the source IP address and the port. However, by using this gateway, developers have the privilege to perform more specific routing. It has the ability to perform routing based on the URL and even more. Figure 14.2 shows how the application gateway handles resources arriving from various locations.

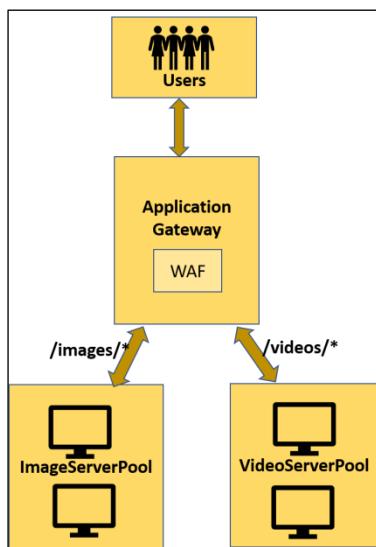


Figure 14.2: Role of an Application Gateway

Users are unaware of source of resources. To make these resources available from several locations is the responsibility of load balancer. This is handled at the layer of Azure application gateway, as shown in Figure 14.2.

Following are the features of Azure application gateway:

URL-based routing

It uses the URL paths specified in the request to route the incoming traffic to the backend pools. For example, a developer can use this routing to route incoming traffic requesting for various information types to different pools.

Multiple-site hosting

This allows developers to use multiple Websites on one application gateway instance. This creates a coherent topology as it allows developers to insert up to 20 Websites, which can be targeted to their respective pool.

Session affinity

This comes in handy when a developer prefers to have a session of a particular client on the same server. To achieve this, the application gateway can use the gateway-managed cookies. This approach is essential when the session for a particular client is stored locally on the server.

Redirection

It is a crucial task for Web applications to accommodate the conversion of HTTP to HTTPS so as to confirm that all the interactions that occur between an application and its clients are encrypted.

14.2.1 Azure Traffic Manager

Azure Traffic Manager allows developers to allocate traffic globally. It is dependent on DNS and offers good accessibility and responds quickly. With the help of DNS, it accepts the request and assigns it to a suitable service. To do this, it checks for healthy endpoints and uses a routing algorithm to route the traffic. An endpoint can be defined as a service, accessible by the Internet, and set up in Azure.

The traffic manager employs various methods to route traffic and to perform endpoint monitoring. It chooses an appropriate method depending on the requirements of the application and failover techniques. It is capable of recovering quickly from difficult conditions, for example breakdown of one complete Azure region.

Following are the key features of a traffic manager:

Enhance application accessibility: The traffic manager offers high-level accessibility for vital applications by keeping track of endpoints and offering automatic failover in cases of endpoint failure.

Perform maintenance without any deferment: It allows developers to perform scheduled maintenance tasks with no downtime. This is because, in such cases, it allocates traffic to other active endpoints.

Distribute traffic in case of huge setups: Using nested traffic manager profiles help developers to merge traffic-routing methods. This allows to develop cutting-edge and adaptable rules, which help in scaling up in case of huge and complex deployments.

Enhance the performance of applications: The traffic manager provides increased responsiveness of the applications by directing user traffic to the closest or most suitable endpoint, improving overall user experience.

Merge hybrid applications: With the help of the traffic manager, developers can deploy endpoints, which are external and non-Azure by nature. This allows it to be utilized with hybrid cloud and the existing deployments.

14.3 Introduction to Azure Application Insights and Log Analytics

Application Insights is a service that helps developers in Application Performance Management (APM) when working on numerous environments. It assists developers to track their live Web applications and identify performance glitches. It has various analytics tools, which they can use to detect problems and to analyze what people are really doing on the application. Using this service, they can focus on how to enhance the performance and ease the use of the application.

This service is suitable for applications deployed on various environments, such as .NET, Node.js, and J2EE. It works on applications installed on the system or in the cloud. It supports DevOps process and provides connection points to various development tools. In case of mobile applications, developers can use it to track and examine telemetry in combination with Visual Studio App Center.

To do this, they just have to install a package on their application and then, go to Microsoft Azure portal and configure an Application Insights resource. Once this is successful, the instrumentation package begins to track the applications and forwards telemetry statistics to the portal.

These statistics can also be obtained from the host platforms, for example performance counters, Azure diagnostics, or Docker logs. Another option is to configure Web tests that periodically forward requests to the Web service on a regular basis. Figure 14.3 shows the Application Insight for Azure Web application.

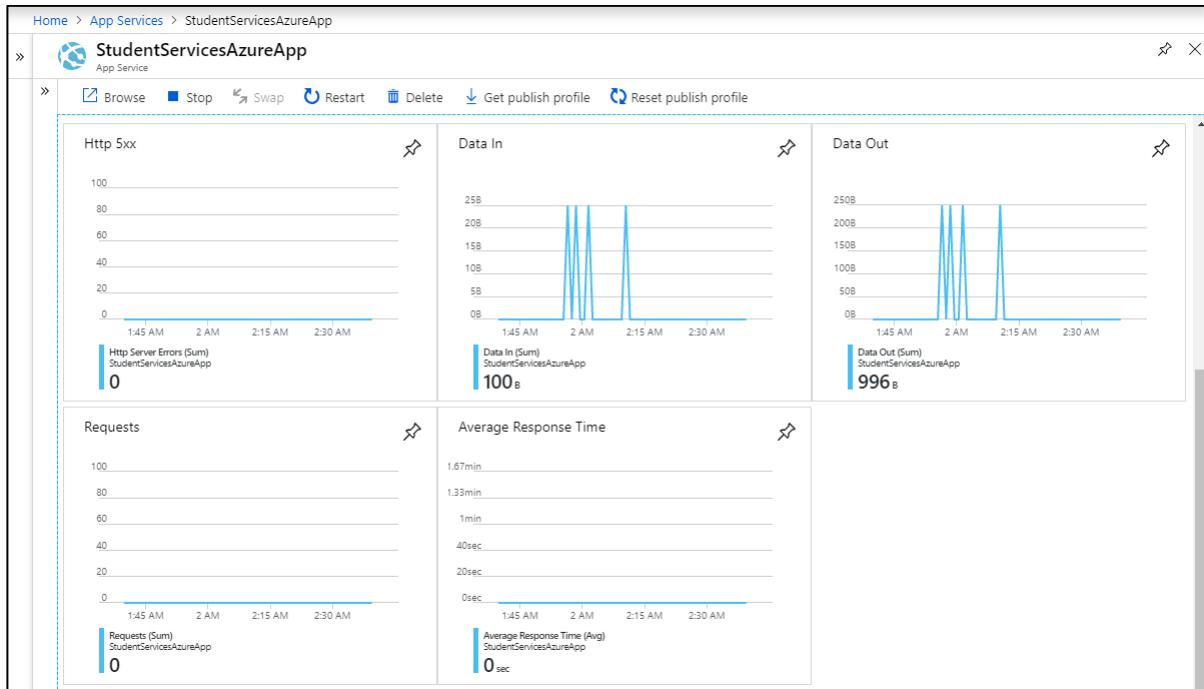


Figure 14.3: Application Insight for Azure Web App

14.3.1 Azure Log Analytics

The Azure Monitor gathers the log data and saves it in the Log Analytics. The Log Analytics is responsible for gathering telemetry and statistics from various resources. It provides developers with a query language that helps in advanced analytics.

They can use log query to obtain information from Log Analytics. This query forms the basis for performing any tasks, such as studying information in the portal, setting a rule to alert the developer for a certain issue and for obtaining information through the Log Analytics API. Figure 14.4 shows the Analytics for the usage of the Azure VM disk.

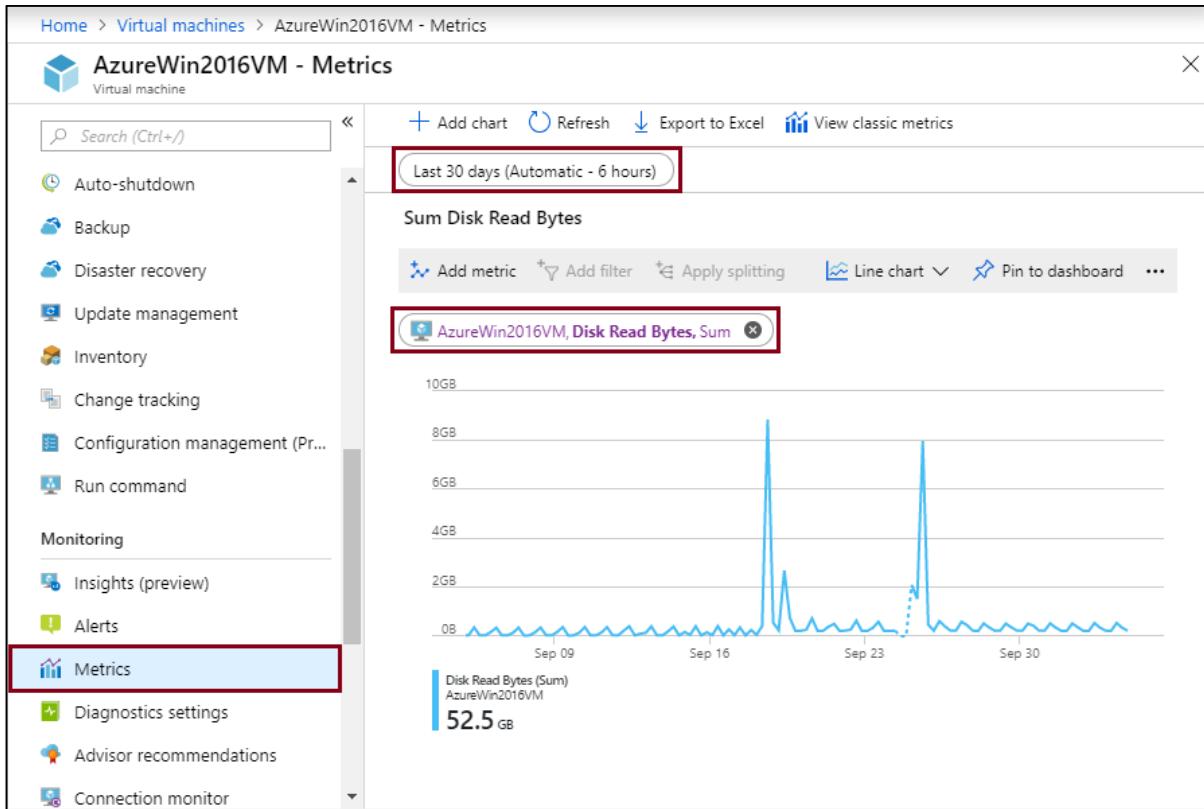


Figure 14.4: Azure Virtual Machine Metrics

Following are offered by Log Analytics page:

- **Tabs:** Developers must create individual tabs for every query they prefer to work with.
- **Visualizations:** These are various charting possibilities.
- **Syntax highlighting:** This helps to enhance the developers' readability of queries.
- **Query explorer:** Developers can use this to view already created and stored queries and functions.
- **Possibilities:** Developers can use this to view how their information is structured, which helps them to query in a better way.
- **Share:** Developers can generate links to queries and also pin them to Azure dashboard that is shared by other developers.
- **Smart Analytics:** Developers can detect any spikes and analyze cause for the same.
- **Column selection:** Developers can use this on the query results columns to group and categorize them.

14.4 Configuring Application Insights

Follow these steps to configure Application Insight for the hosted Web application.

Step 1

Log in to Azure Portal with the login id and password.

Step 2

From the main menu on the left, click **App Services**, which loads the list of the running Web Application. Figure 14.5 shows the Web Application List page.

The screenshot shows the 'App Services' section of the Azure portal. At the top, there are buttons for 'Add', 'Edit columns', 'Refresh', 'Assign tags', 'Start', 'Restart', 'Stop', and 'Delete'. Below this, a search bar says 'Filter by name...' and dropdown menus for 'All resource groups', 'All locations', 'All tags', and 'No grouping'. A message 'Subscriptions: Pay-As-You-Go' is displayed. The table below lists two items:

<input type="checkbox"/>	NAME	STATUS	APP TYPE	APP SERVIC...	LOCATION	SUBSCRIPTI...
<input type="checkbox"/>	func-getfullname	Running	Function App	CentralUSPlan	Central US	Pay-As-You-Go ...
<input type="checkbox"/>	StudentServicesAzureApp	Running	Web app	StudentServic...	Central US	Pay-As-You-Go ...

Figure 14.5: Web Application List Page

Step 3

Select and click the Web App created and hosted earlier. This loads the Web App properties page. Click **Application Insight** from the menu on the left. As there is no Application Insight created yet, this loads an empty screen as shown in Figure 14.6.

The screenshot shows the 'Application Insights' blade for a web app. On the left is a navigation menu with links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), and Deployment. A search bar at the top left contains the placeholder '\$Search'. In the center, a message says 'Enable Application Insights without redeploying your code' with a 'Turn on Application Insights' button. The 'Turn on Application Insights' button is highlighted with a red box.

Figure 14.6: Web App Application Insight

Step 4

In the Application Insight page, click **Turn on Application Insights** to start a new Application Insight. Figure 14.7 shows the creation of a new Application Insight.

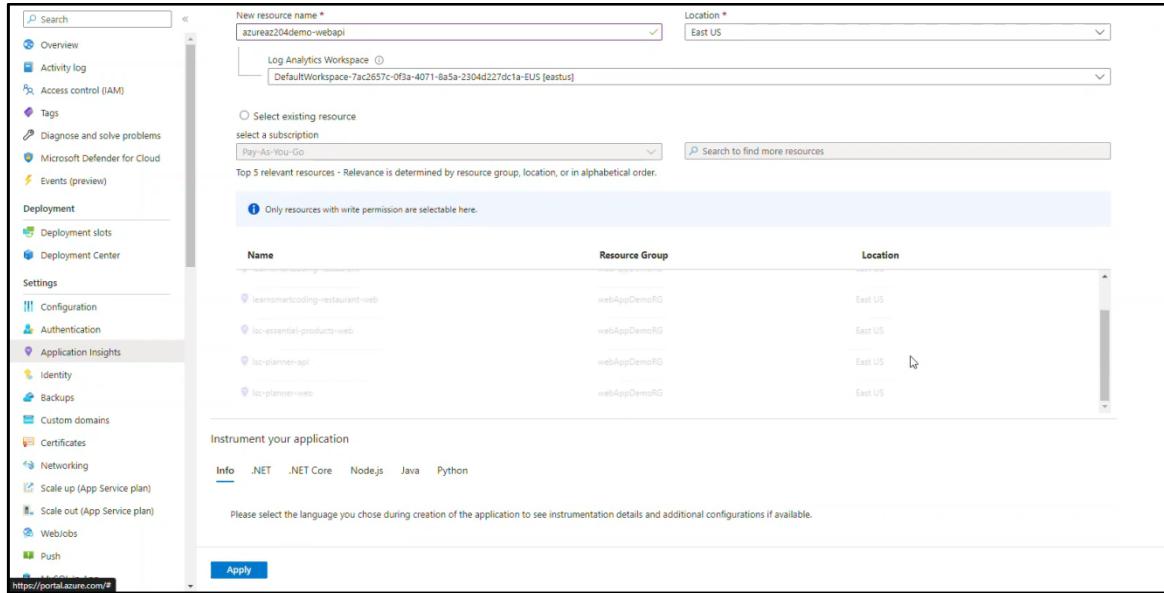


Figure 14.7: Creation of New Application Insight

Step 5

Provide the required information, such as New resource name, Location, and so on and click **Apply**. A confirmation window pops up. Click **Yes** to continue the creation process.

Now, the Application Insight is ready and the developer can load it from the **Manage** page. Figure 14.8 shows the Application Insight running.

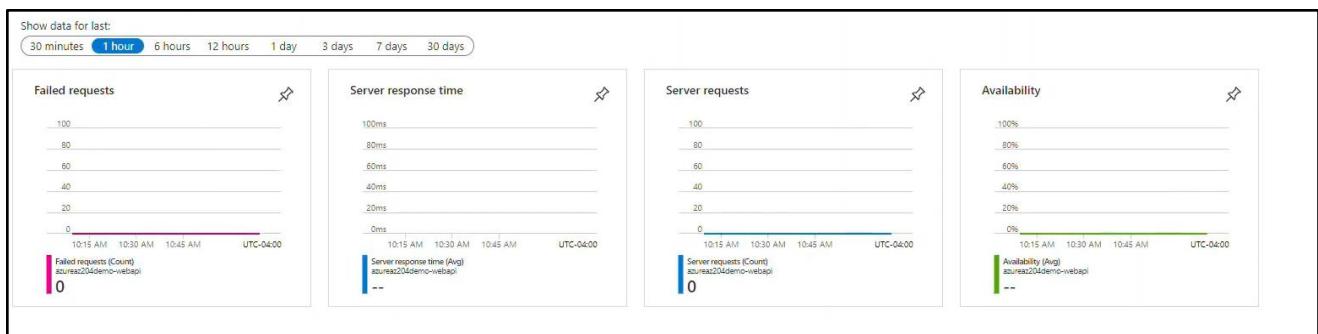


Figure 14.8: Application Insight Graph

14.5 Introduction to Azure Event Hubs and Stream Analytics

Azure Event Hubs provides an environment that allows to stream Big Data. It also offers a service called the event ingestion service, which has the ability to accept and execute close to millions of events per second. Various software and devices generate events, information, and telemetry, which can be executed and saved using the event hubs. Real-time analytics provider or batching/storage adapters can then, be used to convert and save such information.

Following are the instances in which developers can employ the event hubs:

- ❖ To identity any anomalies, such as fraud or outliers
- ❖ To implement logging in applications
- ❖ To perform analytics pipelines, for example, clickstreams
- ❖ To perform real-time dashboarding
- ❖ To archive information
- ❖ To process transactions
- ❖ To process user and device telemetry

So, why is it important for developers to employ event hubs? All the information that is obtained cannot be considered usable. Data is considered usable only if it can be easily processed and when one can obtain good insights. Event hubs offers a distributed execution environment that has various analytics services in and also outside Azure to create a full functioning Big Data pipeline.

Event hubs can be visualized as the entrance for an event pipeline, which is commonly known as an event ingestor. An event ingestor is defined an element that is positioned between the event producers and event users to separate the process of event stream generation from the process of event utilization.

Azure stream analytics is defined as an engine that executes events and allows developers to study large sections of information arriving from various devices. Information can be from sources, such as social media feeds, devices, sensors, Websites, and applications. Information can also be obtained from data streams and by detecting patterns and relationships. Such patterns can then, be utilized to activate other actions, such as alerts, feed data to a reporting tool, or save the data for future usage purposes.

Following are the reasons for employing stream analytics:

- ❖ To perform Internet of Things (IoT) Sensor fusion and clickstream analytics
- ❖ To perform Web logs
- ❖ To implement geospatial analytics for fleet management and in cases where vehicles do not have drivers

- ❖ To monitor remotely and maintain high-value assets
- ❖ To perform real-time analysis on Point of Sale (PoS) data for regulating inventory and detecting anomalies
- ❖ To perform live device telemetry

14.6 Azure Front Door

Azure Front Door is a service that enables developers to enhance availability, reduce latency, and scale effectively. It provides secure experiences for one's users, whether one is delivering content, files, or building global apps and APIs. Azure Front Door is a modern cloud Content Delivery Network (CDN) by Microsoft. It ensures speedy, dependable, and secure access for users worldwide.

Azure Front Door is a powerful service that combines global load balancing, content delivery, and security features

14.6.1 Creating a Front Door Profile

Follow these steps to create a Front Door profile:

Step 1

Log in to Azure Portal with login credentials.

Step 2

Navigate to the home page or the Azure menu and click **+ Create a resource**. Then, search **Front Door and CDN profiles** in the search box as shown in Figure 14.9.

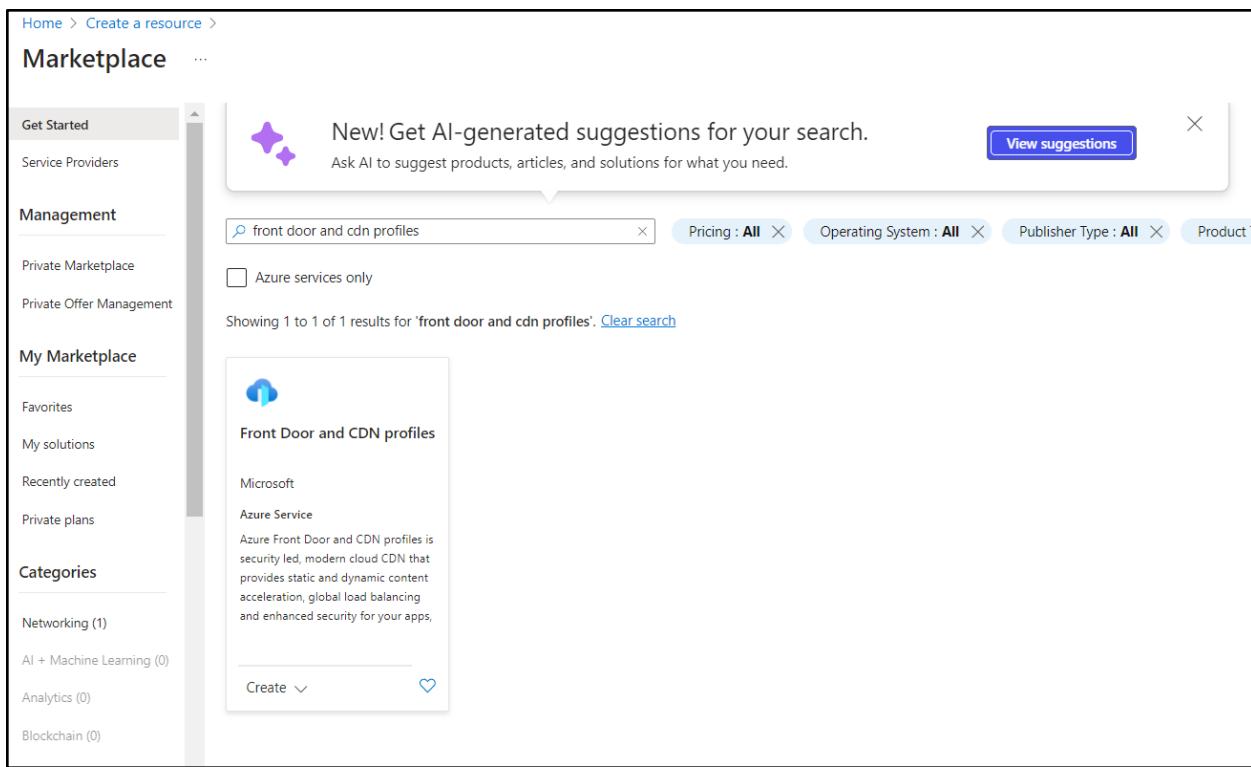


Figure 14.9: Front Door and CDN Profile

Step 3

Click **Create** then, select **Front Door and CDN profiles**. Then, one will be navigated to the compare offerings page. In this page, select **Azure Front Door** and **Quick Create**. Finally, click **Continue to create to Front Door**.

Refer to Figure 14.10.

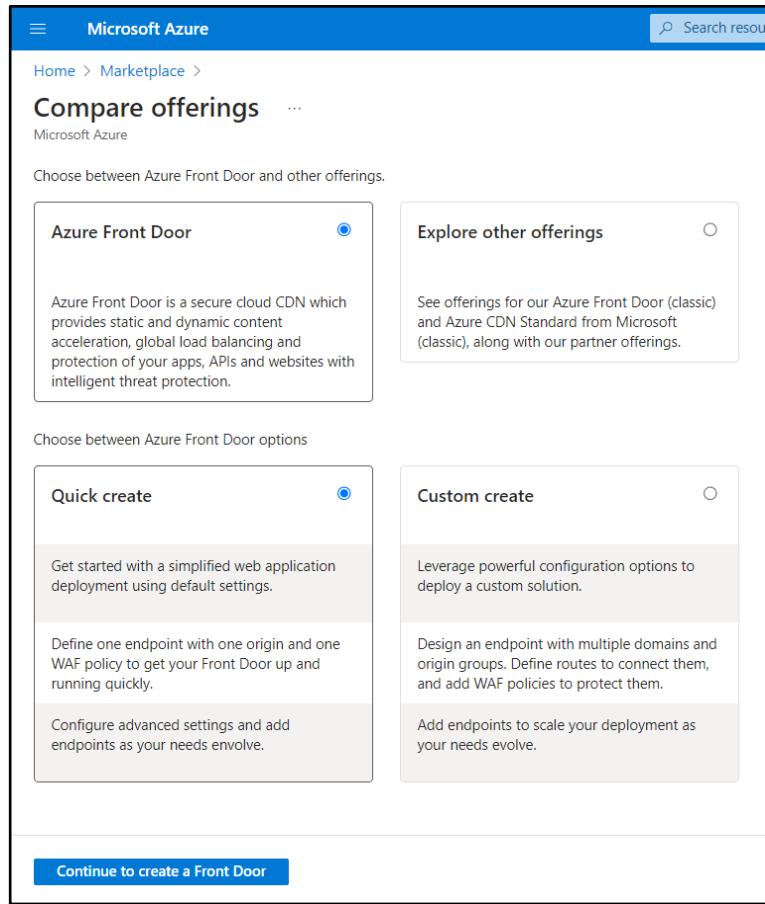


Figure 14.10: Compare Offerings Page

Step 4

In the **Create a Front Door Profile** page, provide information for required settings as shown in Figure 14.11.

Create a Front Door profile ...

Microsoft Azure

*** Basics** Tags Review + create

Azure Front Door is a modern application delivery network platform providing a secure, scalable CDN, dynamic site acceleration, and global HTTP(s) load balancing for your global web applications. [Learn more](#)

Project details
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Resource group *
[Create new](#)

Resource group location

Profile details

Name *

Tier * Standard
Content delivery optimized
 Premium
Security optimized

Endpoint settings

Endpoint name *

Endpoint hostname

Origin type *

Origin host name *

Caching Enable caching

WAF policy

[Review + create](#) [< Previous](#) [Next: Tags >](#) [Automation options](#)

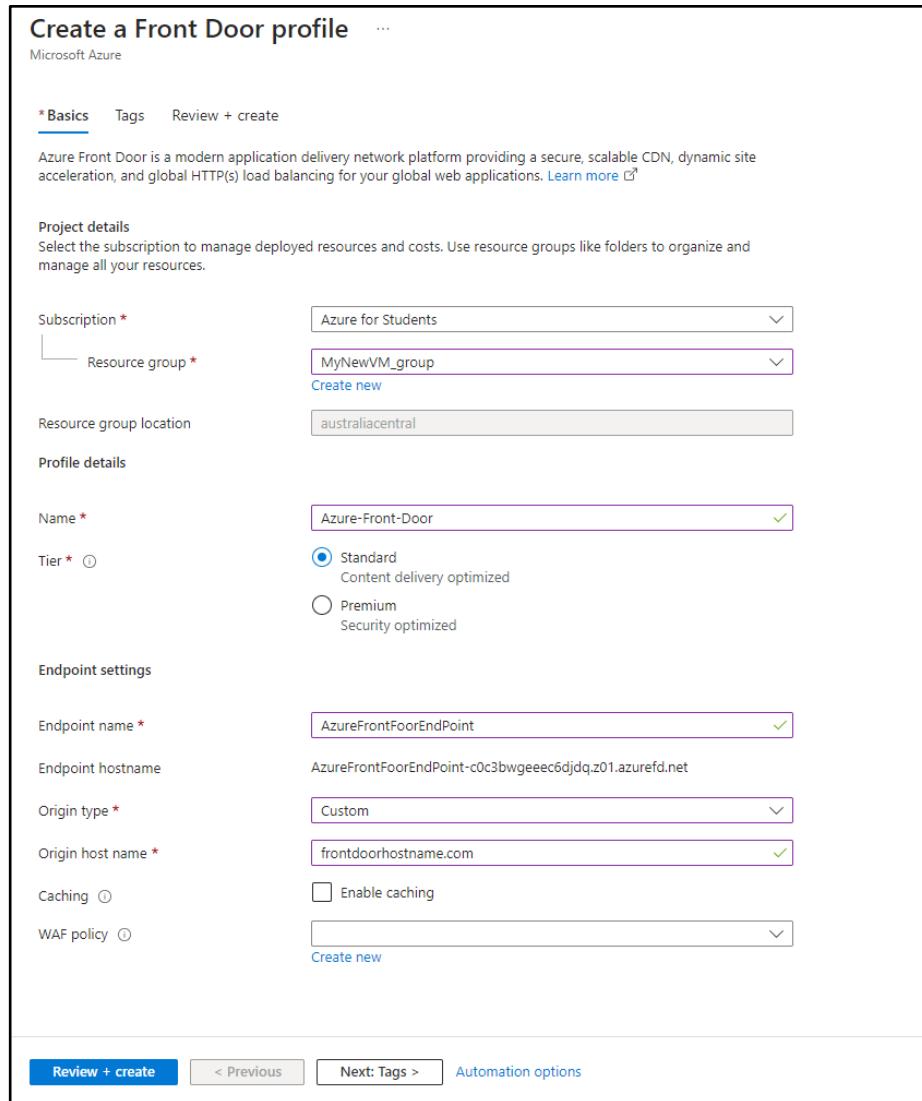


Figure 14.11: Create a Front Door Profile

Step 5

Click **Review + create** and check the validation pass. Then, click **Create** as shown in Figure 14.12.

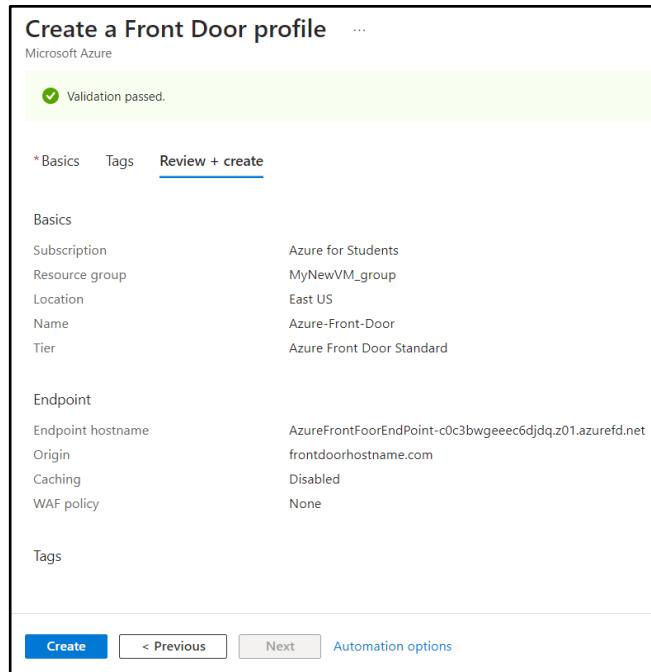


Figure 14.12: Validation Checking

14.6.2 Configuring Routing and Caching Options for Web Services Using Front Door

Azure Front Door is equipped with dynamic site acceleration and load balancing features. When caching is enabled on the route, the edge site receiving each request first checks its cache for a valid response. This caching mechanism aids in minimizing the volume of traffic routed to the origin server. In cases where no cached response is found, the request is then forwarded to the origin.

Follow these steps to configure route and caching options for Web services using the Front Door:

Step 1

Log in to Azure Portal with the login credentials. Then, navigate to Azure Front Door profile and click **Front Door manager**. Now, add a new route by clicking **Add a route** or Select existing routes. Refer to Figure 14.13.

The screenshot shows the Azure Front Door manager interface. On the left, there's a navigation sidebar with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (selected), Domains, Origin groups, Rule sets, Optimizations, Configuration, Properties, Locks, Security (Security policies, Identity), and Help. The main area has tabs for Endpoint name or host ..., Endpoint enabled status: All, and Endpoint provisioning state: All. Below these are search and filter fields. A message says: "You can create multiple endpoints for logical groups of domains. Add routes to connect domains with origin groups. You can add new or use existing domains and origin groups for your routes. Add security rules to complete your Front Door configuration. Learn more". The main content area shows a table for routes:

Routes	Domains	Origin group	Status	Provisioning state	...
default-route	1 selected	default-origin-group	Enabled	Succeeded	[More Options]
	AzureFrontDoorEnd...				

To the right, there's a section for Security policy with a table:

Name	Domain state
No results.	

Figure 14.13: Front Door Manager

Step 2

Select **Enable caching** and specify the query string caching behavior.

Optionally, click **Enable compression** for Front Door compressed responses to the client. Then, click **Update**. Refer to Figure 14.14.

Figure 14.14: Configure Route and Caching

14.7 Setting Up Monitoring and Telemetry Solutions Using Azure Monitor and Application Insights

Follow these steps to set up monitoring and telemetry solutions:

Step 1

Log in to Azure Portal with the login credentials. Then, navigate to Application Insights. Select the recently created Application Insights service. Now, copy the **Instrumentation Key**. Refer to Figure 14.15.

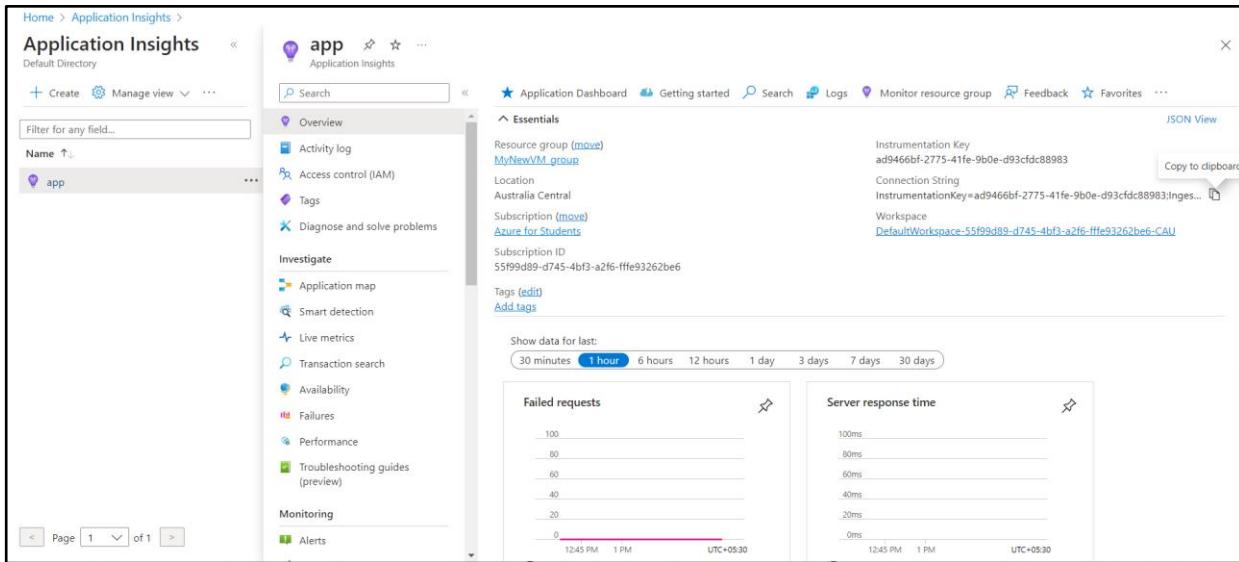


Figure 14.15: Copy Instrumentation Key

Step 2

Open Visual Studio and open the project. Under the project, locate the **appsettings.json** file as shown in Figure 14.16.

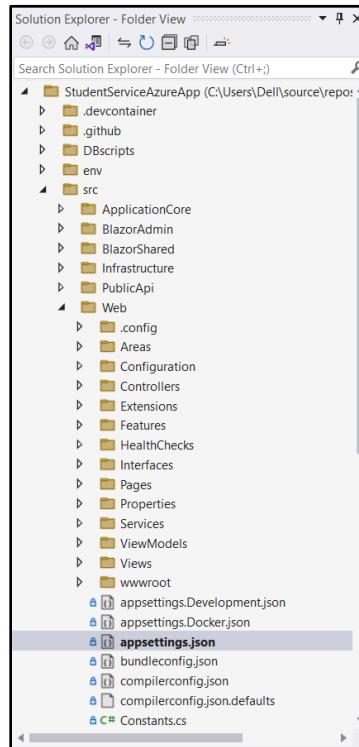


Figure 14.16: Locate Appsetting File

Step 3

Open the **appsettings.json** file. Add the code in Code Snippet 1 by substituting the text from '**'YOUR_INSTRUMENTATION_KEY'**' with the key of the existing Application Insights service. Refer to Figure 14.17.

Code Snippet 1:

```
"ApplicationInsights": {  
    "InstrumentationKey": "YOUR_INSTRUMENTATION_KEY"  
}
```

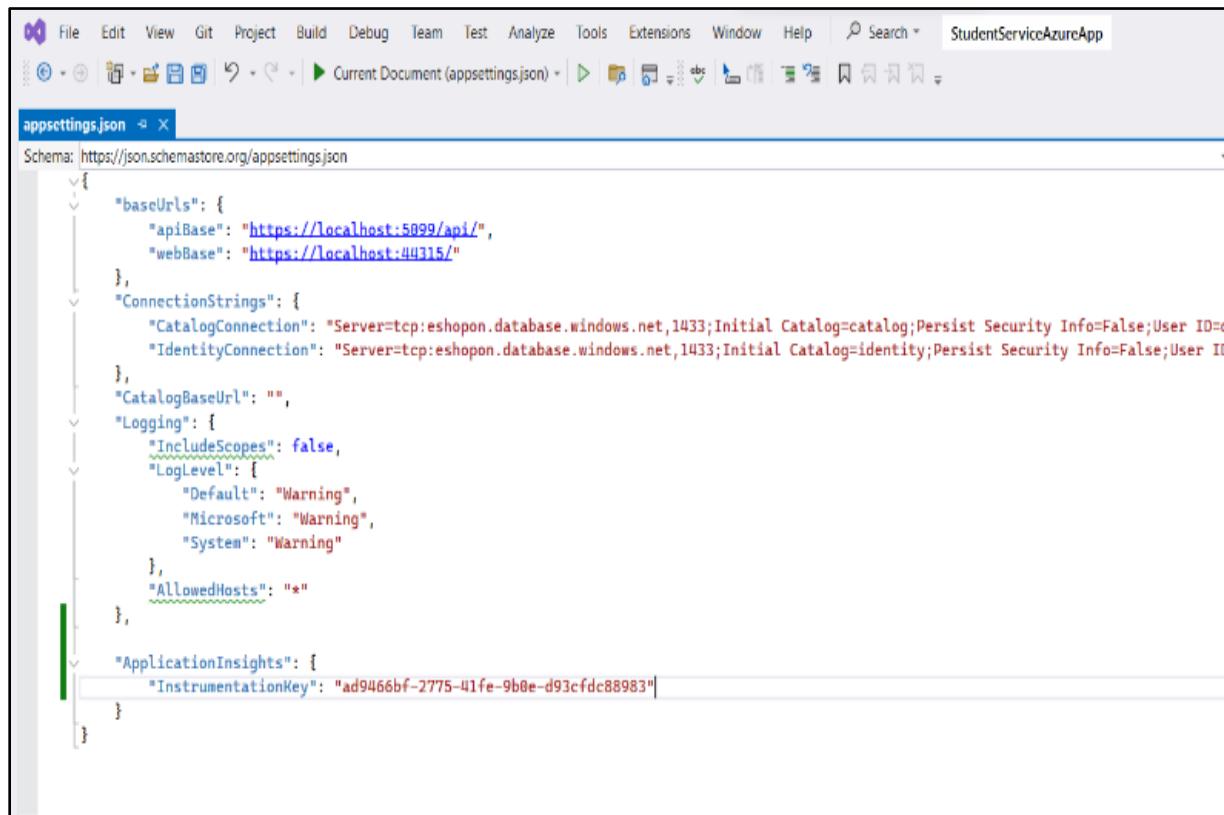


Figure 14.17: Instrumentation Key

Step 4

Right-click the project in the Solution Explorer and select **Application Insights**. Then, select **Search Debug Session Telemetry**. Refer to Figure 14.18.

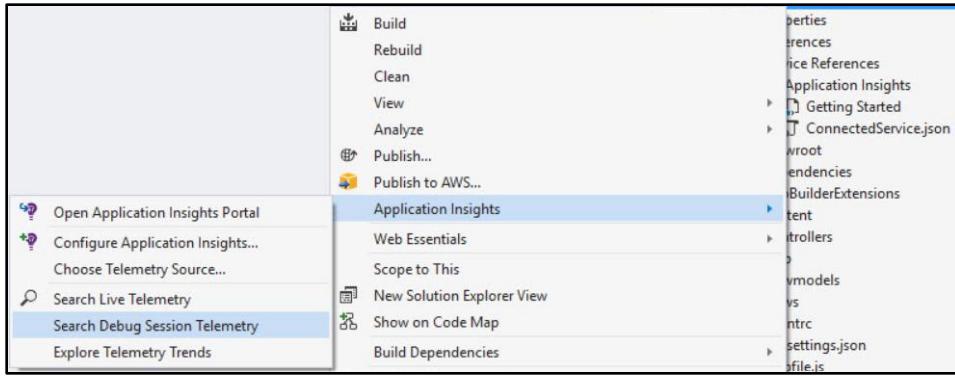


Figure 14.18: Search Debug Session Telemetry

Step 5

This view shows telemetry generated in the server side of the app. Experiment with the filters, and click any event to view more details. Refer to Figure 14.19.

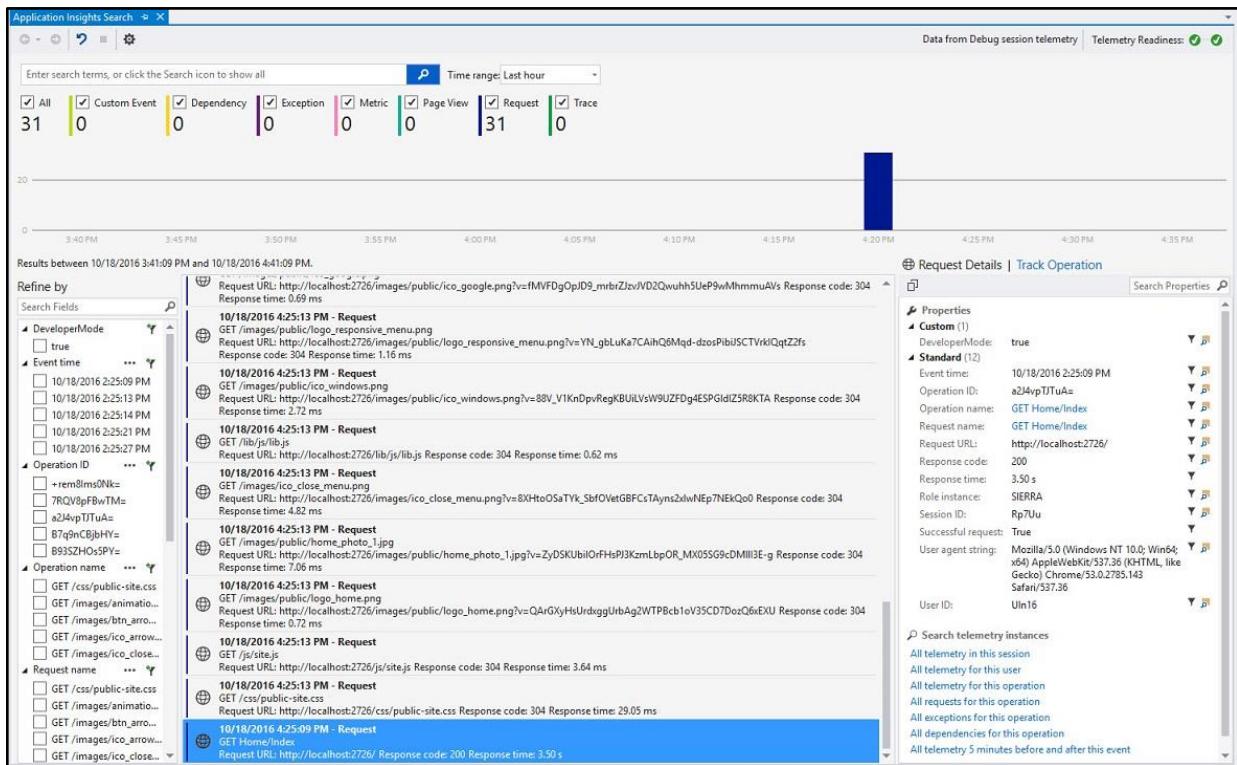


Figure 14.19: Telemetry Generated in Server Side

Step 6

Right-click again on the project in the Solution Explorer and select **Application Insights**. Then, select **Open Application Insights Portal**. Refer to Figure 14.20.

The portal opens on a view of the telemetry from the app as shown in Figure 14.21.

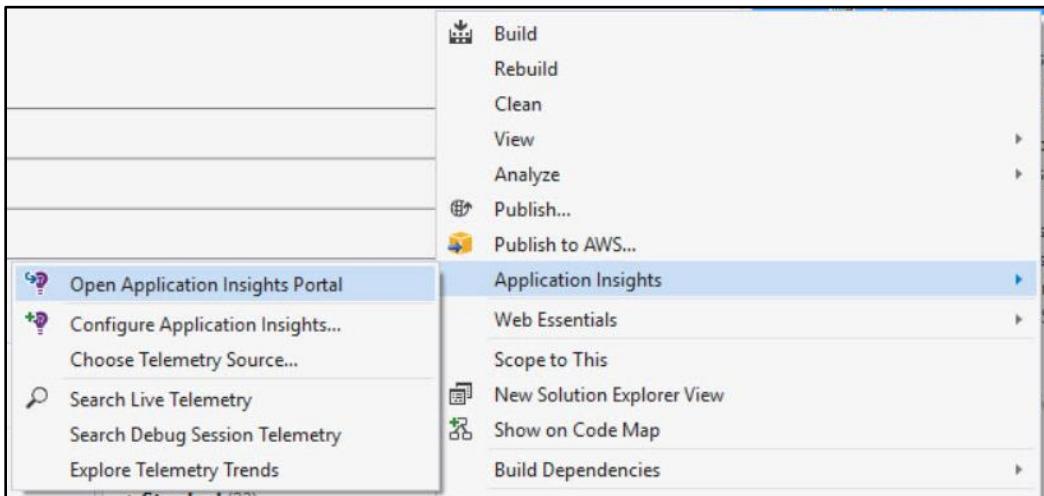


Figure 14.20: Open Application Insights Portal

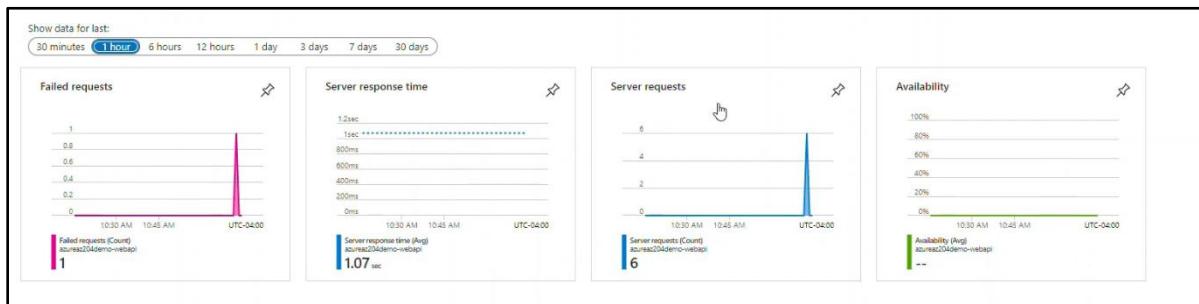


Figure 14.21: View of Telemetry on Portal

14.8 Summary

- ✓ Azure Load Balancer is a tool that helps developers to scale their applications, thus, providing high availability for their services.
- ✓ Azure Application Gateway is a load balancer suitable for handling incoming Web traffic and targeting it towards the Web applications.
- ✓ Azure Traffic Manager allows developers to allocate traffic globally. It is dependent on DNS, offers good accessibility, and responds quickly.
- ✓ Application Insights is a service that helps developers in Application Performance Management (APM) when working on numerous environments.
- ✓ Azure Event Hubs provides an environment that allows streaming Big Data.
- ✓ Azure Stream Analytics is an engine that executes events and allows developers to study large sections of information arriving from various devices.
- ✓ Azure Front Door is a modern cloud Content Delivery Network (CDN) by Microsoft, ensuring speedy, dependable, and secure access for users worldwide to a developer's Web content, whether it is static or dynamic.
- ✓ Azure Front Door is also a powerful service that combines global load balancing, content delivery, and security features.
- ✓ When caching is enabled on a route, the edge site receiving each request first checks its cache for a valid response. This caching mechanism aids in minimizing the volume of traffic routed to the origin server. In cases where no cached response is found, the request is then forwarded to the origin.

14.9 Test Your Knowledge

1. Which of the following are the functions of a load balancer?
 - A. Port forwarding and load balancing
 - B. Application agnostic and transparency
 - C. Automatic reconfiguration and health probes
 - D. All of these
2. Which of the following regarding Application Insight is false?
 - A. It helps to improve performance and usability
 - B. It can be integrated with SQL Azure
 - C. It helps to monitor the live Web application
 - D. None of these
3. Identify the correct features of Traffic Manager.
 - A. Increase application availability
 - B. Improve application performance
 - C. Service and maintenance without downtime
 - D. All of these
4. Which of the following are the functionalities of Event Hubs?
 - A. Anomaly detection
 - B. Application logging
 - C. Analytics pipelines and transaction processing
 - D. All of these
5. Which of the following statements about Stream Analytics are incorrect?
 - A. It can be used to implement geospatial analytics for fleet management and in cases where vehicles do not have drivers
 - B. It can be used to perform Internet of Things (IoT) Sensor fusion and click stream analytics
 - C. Information for analyzing through stream analytics can be from various sources except social media feeds
 - D. It is an engine that executes events and allows developers to study large sections of information arriving from various devices

6. Which of the following statements about Azure Front Door caching is true?
- A. Caching is only applicable to static assets such as images and CSS files
 - B. All request methods (GET, POST, and so on.) are cacheable
 - C. Caching is disabled by default for all routes
 - D. Front Door caches the entire file when delivering large files
7. What is the recommended approach for configuring Azure Application Insights for an ASP.NET Web app?
- A. Use instrumentation keys for telemetry collection
 - B. Manually add Application Insights SDK to the project
 - C. Update the ApplicationInsights.config file with connection strings
 - D. Enable caching for all telemetry data

14.9.1 Answers to Test Your Knowledge

1. D
2. B
3. D
4. D
5. C
6. A
7. C

Try It Yourself

1. Configure Application Insights.
2. Create a Front Door profile.
3. Configuring routing and caching options for Web services using Front Door.
4. Set up monitoring and telemetry solutions using Azure Monitor and Application Insight.
5. Analyze the Application Insights.



SESSION 15

DEPLOYING WEB APPLICATIONS AND AZURE APP SERVICE APPS

Overview

This session introduces Azure App Service Web Apps. It explains how to work with Azure App Services and explores pricing models. It also describes how to deploy Web applications and WCF Data Services.

Learning Objectives

In this session, students will learn to:

- Explain Azure App Services
- Define how to work with Azure App Services
- Explain deployment of Azure Web Applications
- Outline the process of deploying WCF data services
- Explain key provisions of GDPR, HIPAA, and similar data protection regulations
- Explain the concept of an Azure App Service Environment (ASE) in Azure

15.1 Introduction

Azure App Service is an Azure platform that helps host Web applications, REST APIs, and mobile backends. A Web application that is hosted in Azure App Service is termed as Web App. Azure App Services provide a HTTP-based hosting service or platform through which developers can build mobile or Web apps.

Developers can develop these apps in the desired language, such as C#, Java, PHP, Ruby, or Python. In Windows-based or Linux-based environments, these applications execute and scale quickly.

15.1.1 Benefits of Azure App Services

Azure App Services offer three main benefits and they are as follows:

High-Productivity Development

- Azure App Services helps build powerful Web, mobile, and API apps quickly utilizing Java, .NET, Node.js, PHP, .NET Core, Ruby, Python, and Docker.
- Adding Azure App Service into present frameworks offers advanced capabilities for consistent productivity. These capabilities include debugging live sites, continuously integrating the service, and so on.
- Web Apps with Azure App Services can be built using industry-leading Microsoft Visual Studio IDE.
- In addition to this, it accesses the ecosystem of already built apps, connectors, and APIs from Azure Marketplace and efficiently uses the updates with incorporated capabilities with Visual Studio Team Services, Docker Hub, Github, and Bit-bucket.

Fully Managed

- Azure App Services enable developers to use a fully managed platform to carry out operations such as infrastructure maintenance, load balancing, and so on to execute and scale the applications without any overhead on the underlying OS (Windows or Linux).
- Adding SSL certificates, Single Sign-On (SSO), custom domains, and integrating identity services into the apps is easier.
- For faster troubleshooting, services help get detailed performance and application health insights.

Enterprise-Grade Apps

- Azure App Services provide confidence to developers by offering enterprise-grade global data center network, security, and compliance to build and host apps.
- The connection between Web or mobile apps and enterprise systems or SaaS happens in minutes.

- Integrating Azure Active Directory (AD) is helpful to secure apps. Further, one can deploy them to App Service in the environments such as private cloud, public cloud, on-premises, or virtual network.

15.2 Types of Apps

Azure App Service offers a managed platform for different types of Apps.

15.2.1 Web Apps

Azure App Service offers a fully-managed platform called Web apps. Developers can use these Web apps for building, deploying, and scaling enterprise-grade Web apps in a short time. Key features of Web apps are as follows:

- Supports .NET, Java, PHP, Node.js, and Python
- Supports load balancing and has a built-in autoscale
- Promises high availability with auto-patching and offers automated management
- Offers benefits of DevOps optimization. Developers can take advantage of its capabilities such as continuous deployment from Azure DevOps, Docker Hub, GitHub, and other sources, package management, custom domain, staging environments, and SSL certificates
- Supports continuous integration, has deployment slots, and provides testing in production and Web jobs
- Includes an extensive list of templates such as WordPress, Umbraco, Joomla, and Drupal

Web apps are similar to earlier Azure Website services. This is why Azure Websites utilized in Azure are now called Web Apps such as Azure Websites. These execute and also host Web jobs as before depending on the level chosen by the client with automatic backup frequency.

Payment for the Azure Compute resources is usually done with App Service. The App Service plan determines the added resources executed on Web Apps.

15.2.2 Mobile Apps

Mobile Apps under Azure App Service adds a custom backend capability for mobile platforms such as iOS, Windows, Android, and multi-platform environments such as Cordova and Xamarin.

Key features and improvements of Mobile Apps are as follows:

- Built-in autoscale support for Mobile Apps (Depending on the real time load Mobile Apps automatically scale up/down)
- Send push notifications with customer segmentation
- Provides native sync experience which enables app to work offline
- Supports social integration with Facebook, Twitter, and Google
- Supports traffic management (helps scaling the apps around the world)
- Supports continuous integration/deployment with Visual Studio Online, GitHub, and Bitbucket
- Offers Hybrid connections and virtual networking support to on-premises data centers to access data securely
- Supports staged deployment and production testing
- For long-running background tasks, apps offer WebJobs support

Mobile Apps provide a highly scalable platform to develop mobile applications, for system integrators and enterprise developers.

15.2.3 Logic Apps

Logic Apps by Azure App Service provide the technical or non-technical user with the ability to automate the execution of the process across popular commercial services, consumers, and custom APIs on-premises. Key features of the Logic Apps are as follows:

- Creating business processes and workflows visually
- Helps in offering integration capabilities in Web, Mobile, and API Apps
- Supports integrating with SaaS and enterprise applications
- Supports automating EAI/B2B and business process
- Helps connect to on-premises data

15.2.4 API Apps

API Apps under Azure App Service can be used for building, consuming, and distributing APIs in the cloud and on-premises network. API Apps provide a rich platform and ecosystem to develop APIs in any language.

Following are key features of API Apps:

- Supports integrating with SaaS and enterprise applications
- Helps automate versions and deploying of API Apps
- Secures APIs with Single Sign-On, Active Directory, and OAuth
- Supports internal sharing of APIs with organizational gallery

API Apps help enhance the experience of creating, employing, publishing, consuming, handling, and monetizing RESTful Web APIs. Present APIs can use the API Apps platform without modifications and connect to Azure Active Directory. Connecting to SaaS platforms is easy with API Apps.

15.2.5 Azure Functions

An event-driven bean that can host logic code and process without a server is known as server-less computing. Developers can use Azure Functions, built on top of the App Service infrastructure, to run scripts or snippets of code in response to various events, such as events raised by Azure services or third-party services.

Azure Function supports multiple languages such as C#, F#, PHP, Python, and so on. Developers can also secure Azure Functions using OAuth or other identity providers such as Azure AD.

15.3 Azure App Service Pricing

Azure App Services pricing varies on the selection of different criteria such as OS, Region, Environment Type, Workload, and so on. Categories for plans include Basic, Standard, Premium, and so on. For example, some standard pricing for Windows machines for pay as you go plans are shown in Figure 15.1.

	Free Try for free	Shared Environment for dev/test	Basic Dedicated environment for dev/test	Standard Run production workloads	Premium Enhanced performance and scale	Isolated High-Performance, Security and Isolation
Web, mobile or API apps	10	100	Unlimited	Unlimited	Unlimited	Unlimited
Disk space	1 GB	1 GB	10 GB	50 GB	250 GB	1 TB
Maximum instances	–	–	Up to 3	Up to 10	Up to 30*	Up to 100
Custom domain	–	Supported	Supported	Supported	Supported	Supported
Auto Scale	–	–	–	Supported	Supported	Supported
Hybrid Connectivity	–	–	Supported	Supported	Supported	Supported
Virtual Network Connectivity	–	–	Supported	Supported	Supported	Supported
Private Endpoints	–	–	Supported	Supported	Supported	Supported
Compute Type	Shared	Shared	Dedicated	Dedicated	Dedicated	Isolated
Pay as you go price	Free	\$0.013/hour	\$0.075/hour	\$0.10/hour	\$0.20/hour	\$0.40/hour

Figure 15.1: Creating Web App, API App, or Logic App

One can check the Azure Price calculator which is available on the Azure official Website at:

<https://azure.microsoft.com/en-in/pricing/details/app-service/windows/>

15.4 Options for Creating Web Apps

There are many options for creating Webapps.

15.4.1 Azure Marketplace

Azure Marketplace is an online store with thousands of software IT applications and services from leading technology companies.

15.4.2 Visual Studio Code

Visual Studio Code merges the simplicity of a source code editor with powerful developer tools such as code completion and IntelliSense debugging.

15.4.3 Visual Studio IDE

The Integrated Development Environment (IDE) is a program that supports various characteristics of software development. The Visual Studio IDE is a creative launchpad that developers can use to edit, debug, and build developer code and publish their applications. Additionally, to the standard editor and debugger provided by most IDEs, Visual Studio includes compilers, code completion tools, graphics designers, and many other features to improve the software development process.

15.5 Create Apps

15.5.1 Creating Web App, API App, or Logic Apps

Following are the steps to create a Web App, API App, and Logic App respectively:

Step 1: Log in to Azure Portal with your login credentials, which will display the Dashboard or Welcome page.

Step 2: In the left main menu, click +Create a resource, which will load the Resource creation page as shown in Figure 15.2. The Web menu lists three App Services in the sub-menu as marked in Figure 15.2.

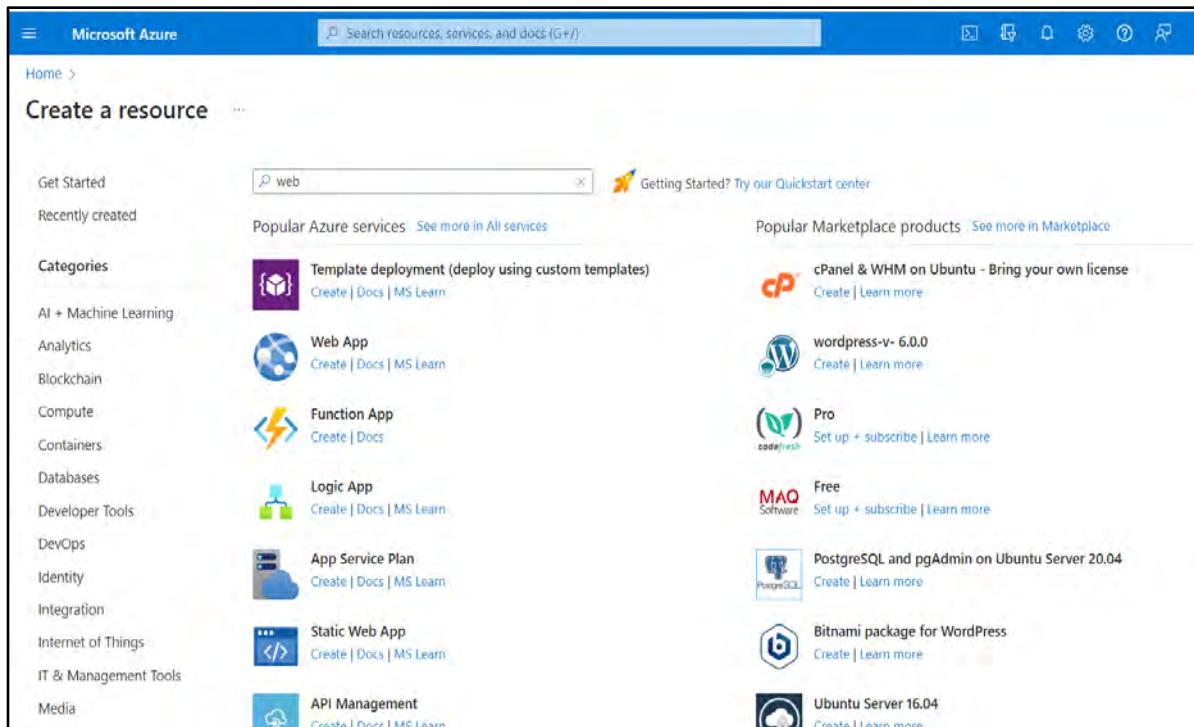


Figure 15.2: Creating Web App, API App, or Logic App

Step 3: Developers can click the Service of their choice after deciding whether to create a Web App, Logic App, or API App.

Step 4: Follow the step-by-step process in the Azure portal, which is self-descriptive. Figure 15.3 depicts creating Web App, API App, and Logic Apps.

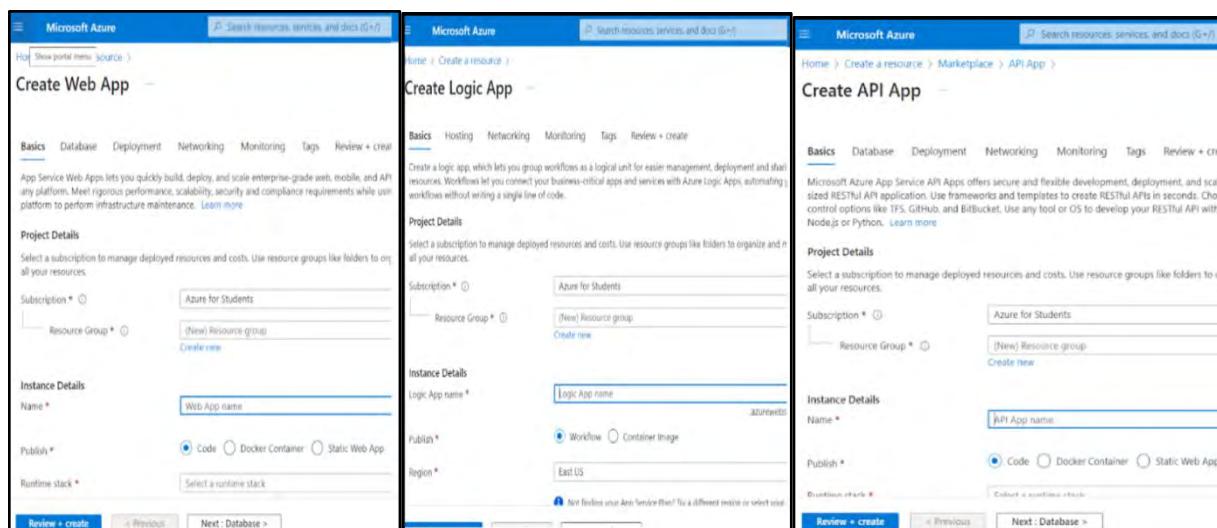


Figure 15.3: Steps to Create Web App, API App, and Logic App

15.5.2 Creating Mobile App

Follow these steps to create Mobile App Resources:

Step 1: After successfully logging into the Azure portal, on the left, click +Create a resource, which will load the Resource creation page as shown in Figure 15.4.

Under the Mobile menu, the Mobile App service is listed in the sub-menu, as shown in Figure 15.4.

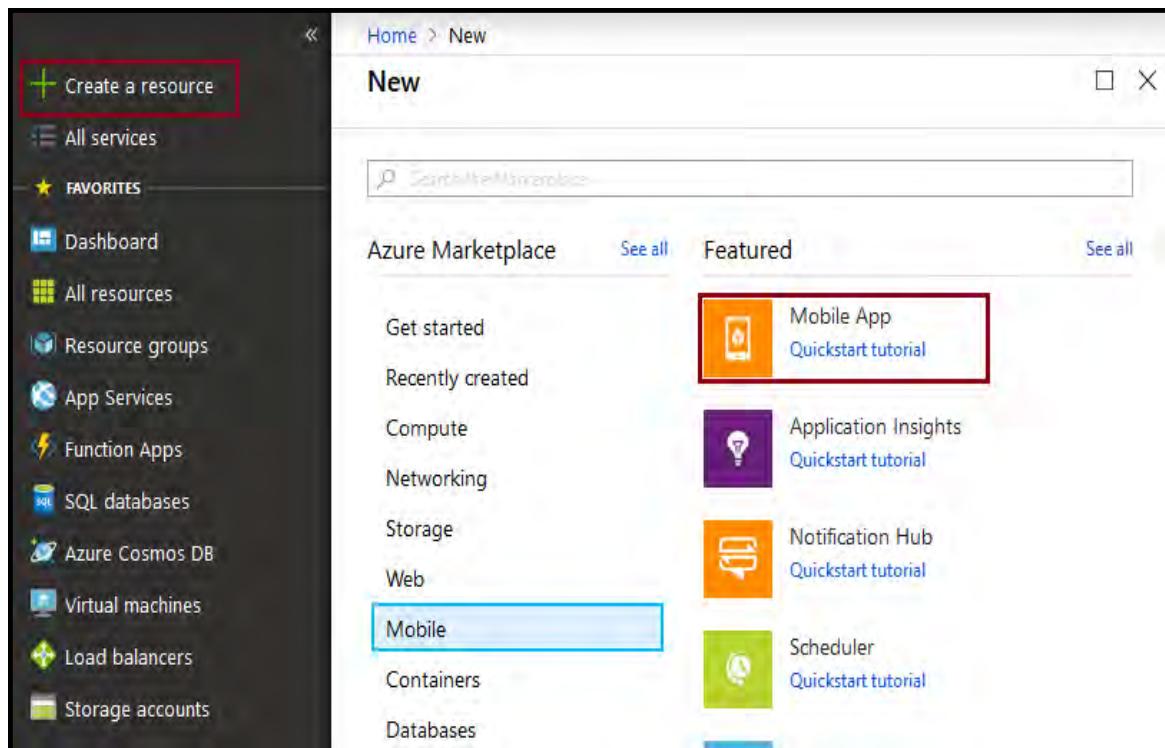


Figure 15.4: Creating Mobile App Services

Step 2: Follow the step-by-step process in the Azure portal, which is self-descriptive. Figure 15.5 depicts the creation of a Mobile App service.

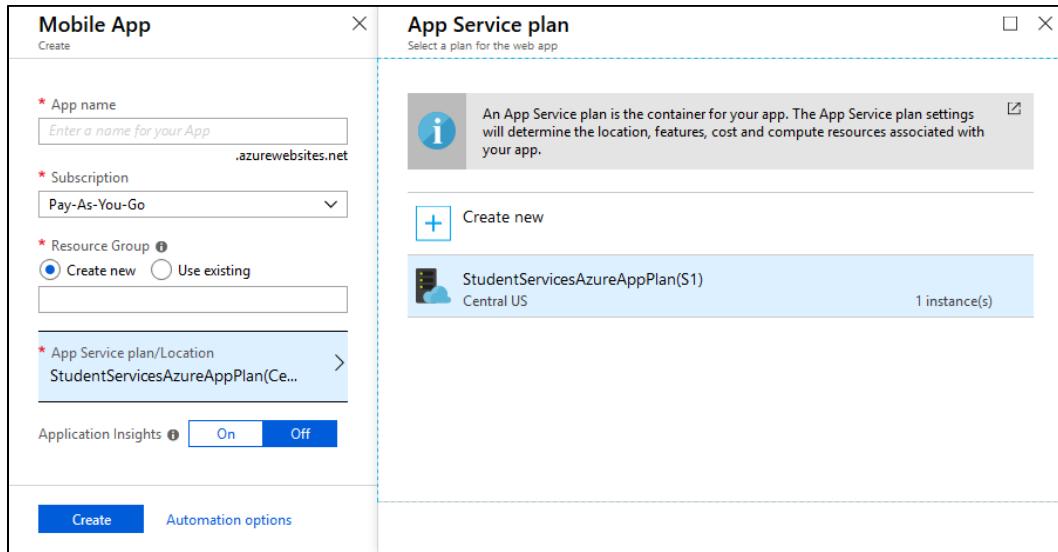


Figure 15.5: Create Mobile App Service Page

15.6 Logging

In Azure, there is a comprehensive list of security auditing and logging options that can be configured. These help in identifying lapses in security mechanisms and policies. Logs also offer data for keeping applications up and running. Thus, troubleshooting issues that occurred previously can be done and at the same time, new ones can be prevented. This helps in improving the performance and maintainability of an application. In addition, actions can be automated, so that manual intervention is not required.

Following are the types of Azure logs:

Control/Management Logs	Data Plane Logs	Processed Events
Give details on the Azure Resource Manager operations such as CREATE, DELETE, and UPDATE.	Give details on events presented as a part of Azure resource usage. For example, the Windows event system log, security logs, diagnostics logs configured through Azure Monitor, and application logs in a VM.	Give details on analyzed events or alerts processed on behalf of the user. For example, Azure Security Center alerts. In this, the Azure Security Center completes the processing and analysis of the subscription. It thus provides brief security alerts.

15.7 Deployment of Azure Web Applications

Once a Web application is built, it is required to deploy the application in a production environment where users can actually use it. Developers can prefer to deploy their application or service to a private environment with their own servers or they can select Azure. Besides selecting where to run the application, developers also encounter other challenges in complex applications. These challenges may involve configuring databases, assemblies, IIS settings, certificates, and other configuration details.

To surpass these challenges, it is recommended to use Microsoft Azure for deployment.

15.7.1 Deployment Sources (Github, Docker, Bitbucket, OneDrive)

Application code is stored in a deployment source. The deployment source for production apps is typically a repository hosted by version control software such as GitHub, Docker, BitBucket, or OneDrive. The deployment source for development and testing purposes can be stored on a local machine. Cloud folders such as OneDrive and Dropbox make it simpler to get started with App Service, but they are still not typically used for enterprise-level production applications.

15.7.2 Creating and Deploying an ASP.NET Core Web App

Azure Web Apps offers a highly scalable Web hosting service that patches/upgrades itself. Following steps shows how to deploy an ASP.NET Web app to Azure Web Apps. Once the developer is successful in the deploying process, he/she will have a resource group consisting of an App Service plan and an Azure Web app with a deployed Web application.

Step 1: Create an ASP.NET Core Web app using the usual procedure. Refer to Figure 15.6 to see an example.

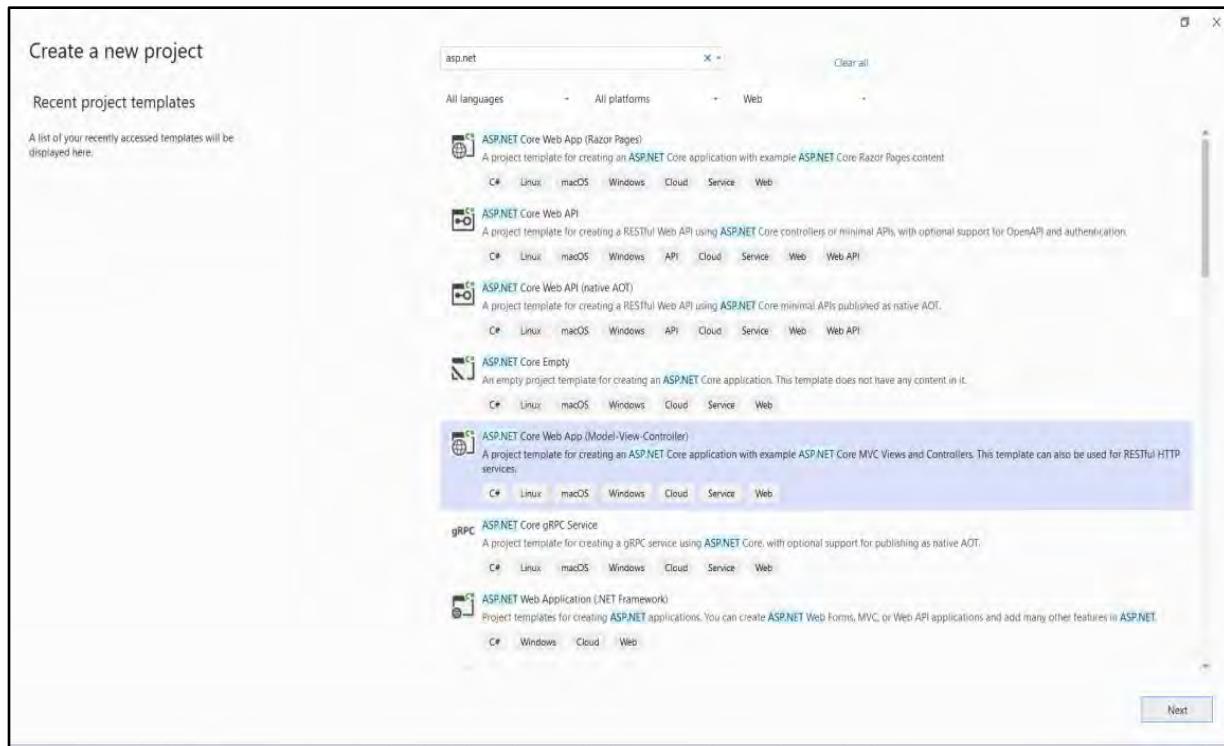


Figure 15.6: Create ASP.NET Core Web App

Step 2: While creating a Web Application, ensure that authentication is set to No Authentication and all other options are enabled, as shown in Figure 15.7 and click Create.

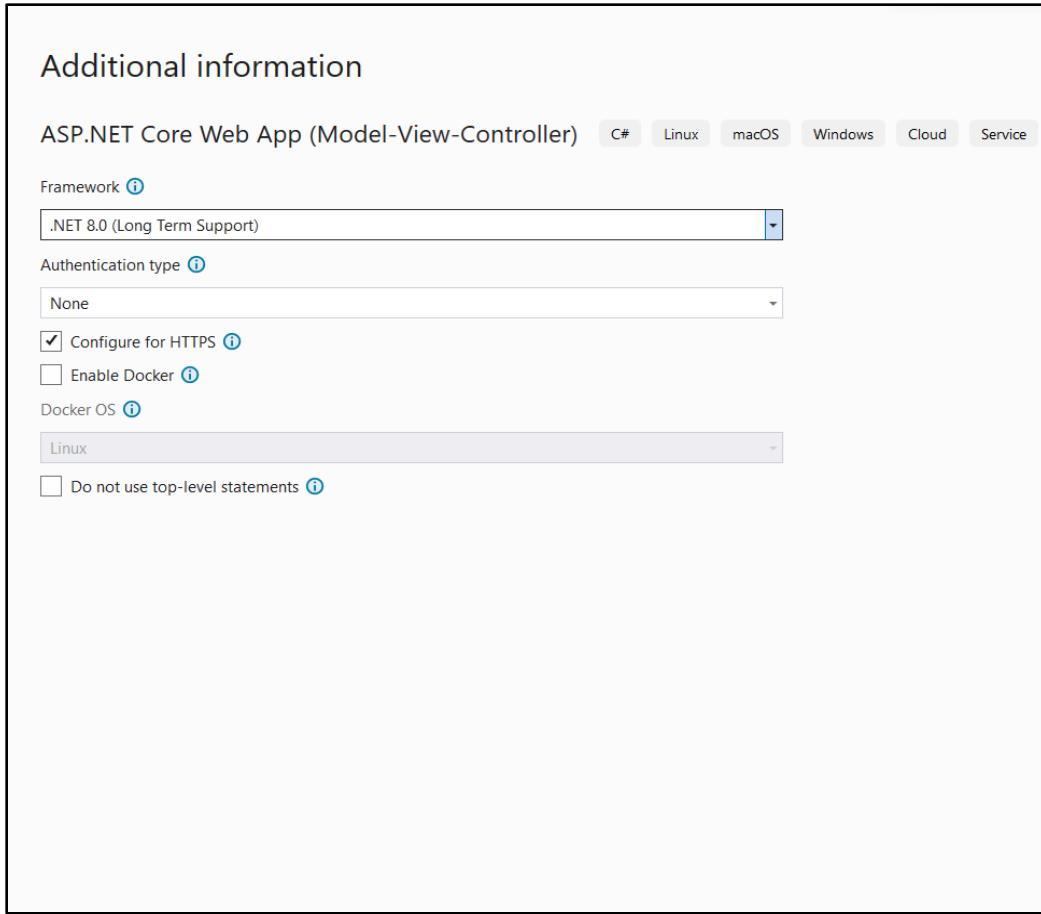


Figure 15.7: Configuring New ASP.NET Core Web Application

Step 3: Add desired code to the application. Build the application.

Step 4: Select Debug □ Start Without Debugging from the menu to run the Web app locally.

Step 5: Launch the publish wizard; in the Solution Explorer, right-click the project name and select Publish. The publish wizard launches automatically. Now, choose App Service → Publish, as shown in Figure 15.8. This will open the Create App Service dialog box, as shown in Figure 15.9.

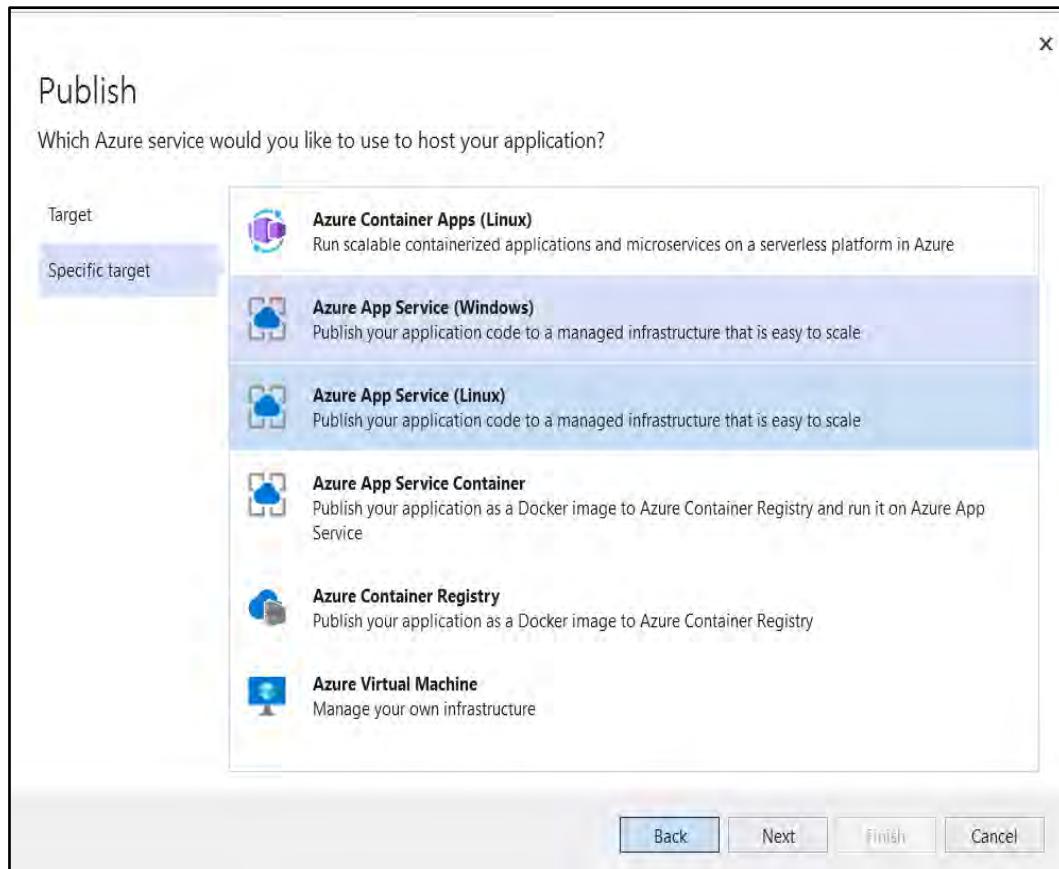


Figure 15.8: Publish Wizard

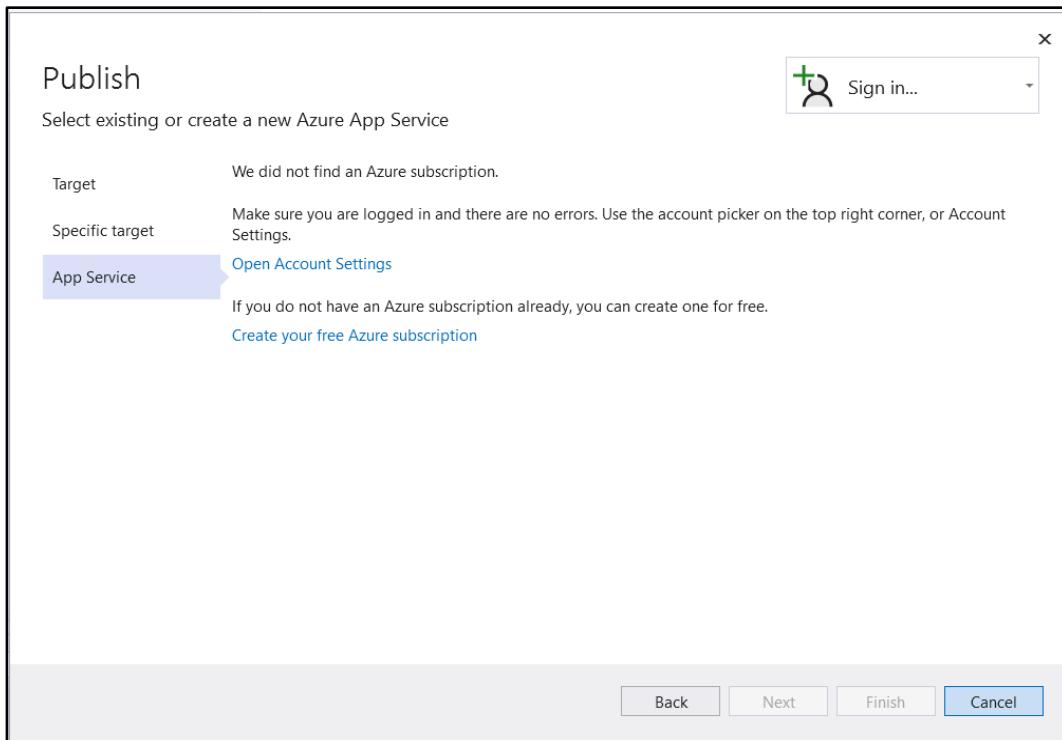


Figure 15.9: Create App Service Dialog Box

Step 6: Click Add an account and sign in using a valid Azure account.

Step 7: Now, create a resource group and an App Service hosting plan, as shown in Figure 15.10.

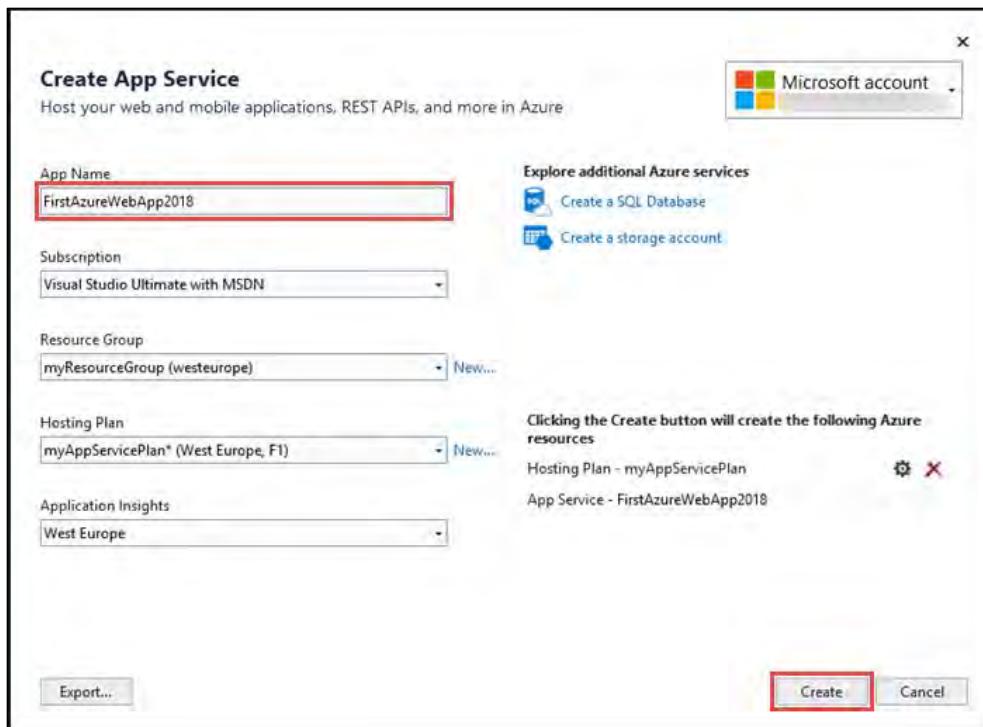


Figure 15.10: Hosting Options in Create App Service Dialog Box

Step 8: Click Create to publish the Web app.

15.8 Scale-Out

Scale-out is such as making multiple copies of the developer's Website and adding load balancers to distribute the load across their Websites. When scaling their Website on Azure Websites, the developer does not require to configure the load balancing separately provided by the platform.

Scale-Out and scale-up are the two main scaling work-flows.

Scale-Out

Essentially, this is adding multiple instances of the application running on the application, that is, depending on the price point, increasing the number of VM instances up to 30. However, a developer can scale up to 100 cases at the isolated level, depending on their requirements. The developer can also do the scaling calculations manually or set it to auto-scaling according to some rules.

Developers can perform scale-out from the Azure portal, as shown in Figure 15.11.

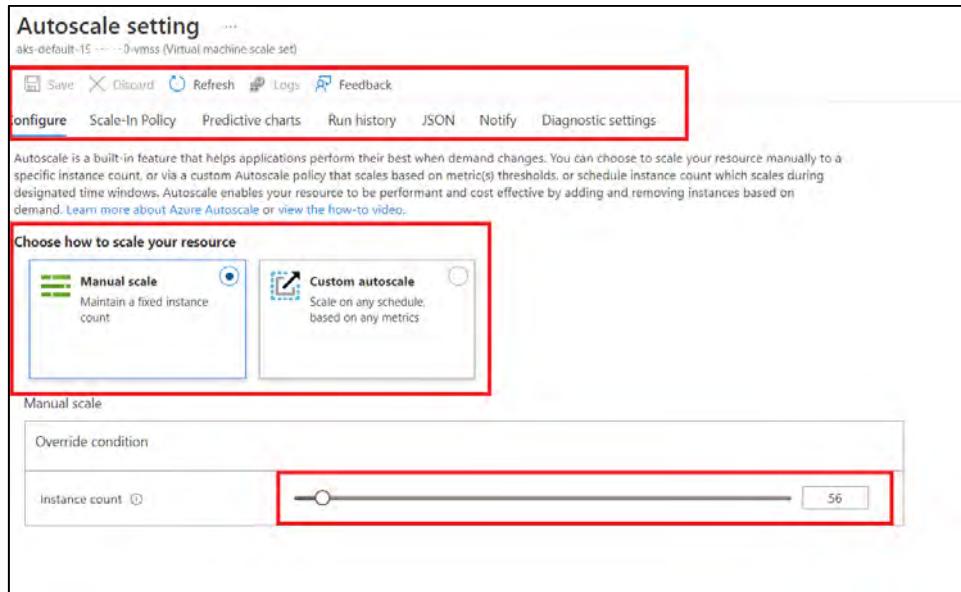


Figure 15.11: Autoscale Setting in Monitor

The developer can choose Scaling from the left navigation bar, then select Manual Scaling and set the instance number according to the requirements or one can choose Auto Scaling according to some rules. Refer to Figure 15.12.

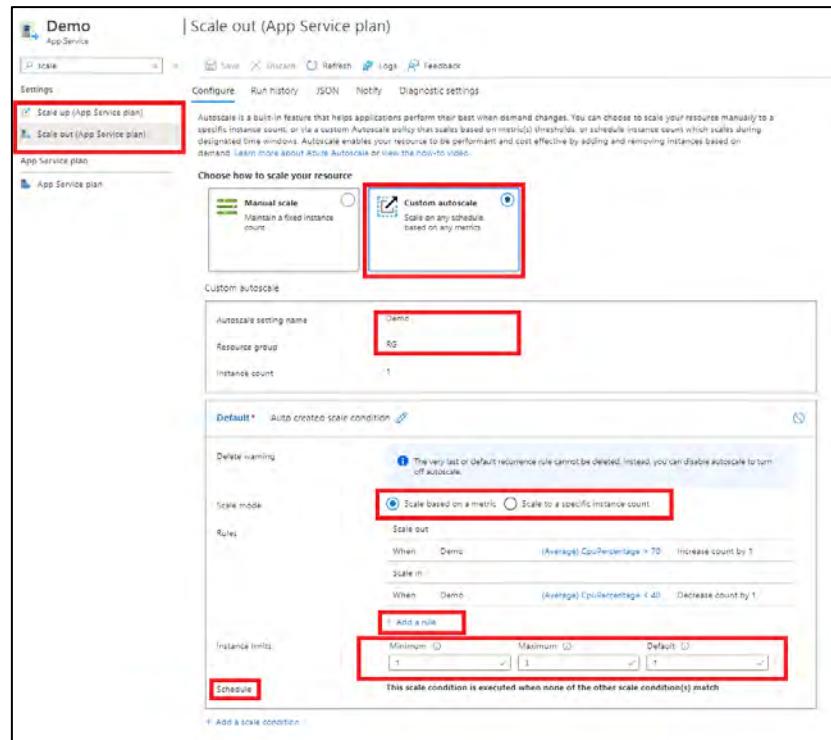


Figure 15.12: Scale out (App Service Plan)

Developers should automatically expand parameter names with resource group names. There are several more options, such as scaling modes, rules, instance limits, and schedules. Thus, the process of Scaling out (that is, scale-out in Azure App Services) can be achieved.

Scale-up

This scaling option allows the developer to change the performance of their instance in terms of CPU, memory, and disk space, directly affecting cost. This scaling is done by changing the App Service plan pricing tier. Also, depending on the pricing tier, several useful features are available in Azure App Service, such as custom domains and certificates, intermediate slots, auto-scaling, instances, daily backups, and more.

15.9 Limits and Quota

Azure uses limits and quotas to avoid fraudulent budget overruns and to comply with Azure capacity limits.

Some services have configurable limits:

- Table 15.1 uses the Limit header if the service does not have a custom limit. In this case, the default and maximum limits are the same
- If the limit is adjustable, the table includes the headings 'Default Limit' and 'Maximum Limit,' where the limit can be hiked beyond the default limit, but not more than the maximum limit

Resource	Limit
Resources per resource group	Resources are not limited by resource group. Instead, they are limited by resource type in a resource group.
Resources per resource group, per resource type	800 - Some resource types can exceed the 800 limit.
Deployments per resource group in the deployment history	800
Resources per deployment	800
Management locks per unique scope	20
Number of tags per resource or resource group	50
Tag key length	512
Tag value length	256

Table 15.1: Resource Group Limits

Quota defines limits on the resources a user subscription can provide or use.

Quotas exist for Azure Static Web Apps as shown in Figure 15.13 (subject to change depending on Microsoft).

Feature	Free plan	Standard plan
Included bandwidth	100 GB per month, per subscription	100 GB per month, per subscription
Overage bandwidth	Unavailable	\$0.20 per GB
Apps per Azure subscription	10	Unlimited
App size	250 MB 500 MB max app size for a single deployment, and 0.50 GB max for all staging and production environments	500 MB 500 MB max app size for a single deployment, and 2.00 GB max combined across all staging and production environments
Plan size		
Pre-production environments	3	10
Custom domains	2 per app	5 per app

Figure 15.13: Quotas for Azure Static Web Apps

15.10 Different Instances - DEV, UAT, and PROD

Testing ensures that the app delivered to the end user is of high quality. DEV, UAT, and PROD are different stages of testing.

DEV	UAT	PROD
A DEV test is performed by software developers who created the functions.	UAT test is performed by the client and a small group of registered testers. At this stage, a lot of feedback on features and user experience can be received.	During the development stage, the app is made available to the public. A group of users explore new features and list unexpected issues and bugs that are then fixed by professionals.

15.11 Deploying WCF Data Services

.NET Framework provides WCF data services, formerly known as ADO.NET data services that developers can use to create services using the Open Data (OData) protocol. OData is a REST-based protocol that allows developers to perform operations on resources, such as Create, Read, Update, and Delete (CRUD)

through Uniform Resource Locators (URLs). Using OData, developers can disclose data as resources that can be addressed by URLs. This makes the data accessible and changes data using REST syntax. OData uses the Entity Data Model of the Entity Framework to expose resources as related entities. Developers can perform CRUD operations on the entities using the standard HTTP methods, such as GET, PUT, POST, and DELETE. The result returned by the WCF data service can be in XML-based Atom and JSON formats.

Figure 15.14 shows the role of WCF data service in an enterprise application.

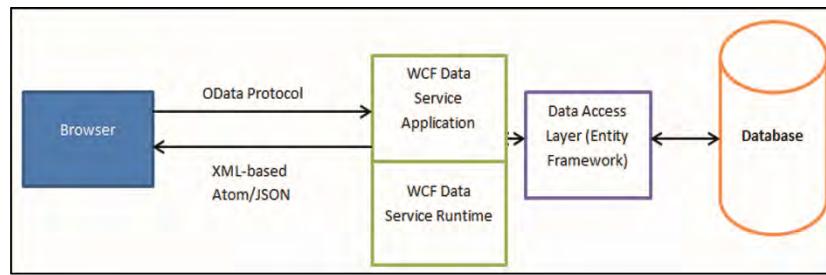


Figure 15.14: WCF Data Services

Following steps describe how to create and deploy a WCF data service:

Step 1: Go to File → New Project and select ASP.NET Web Application to create an ASP.NET Web app named WCFDataServiceDemo and click Create as shown in Figure 15.15.

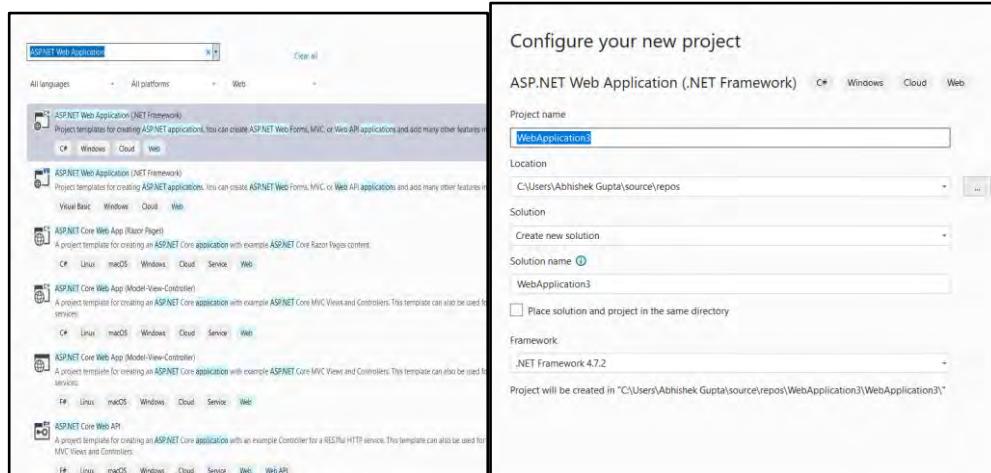


Figure 15.15: Create ASP.NET Web Application

Select Empty project from the dialog box as shown in Figure 15.16 and click Create.

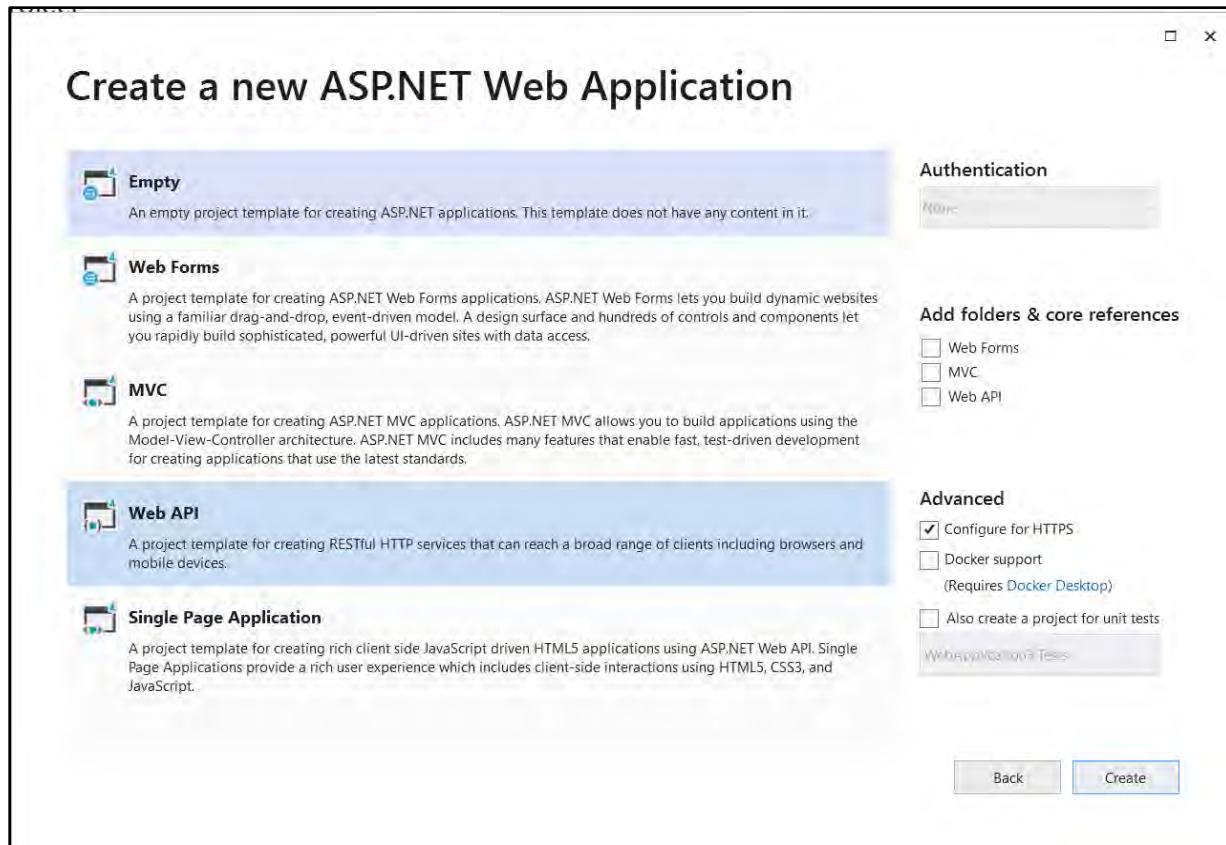


Figure 15.16: New Project Template Wizard

Step 3: In the Solution Explorer, right-click the ASP.NET project, and then, select Add → New Item.

In the Add New Item dialog box, select Web → WCF Data Service 5.6.4 and name it as ProductsWcfDataService.svc as shown in Figure 15.17.

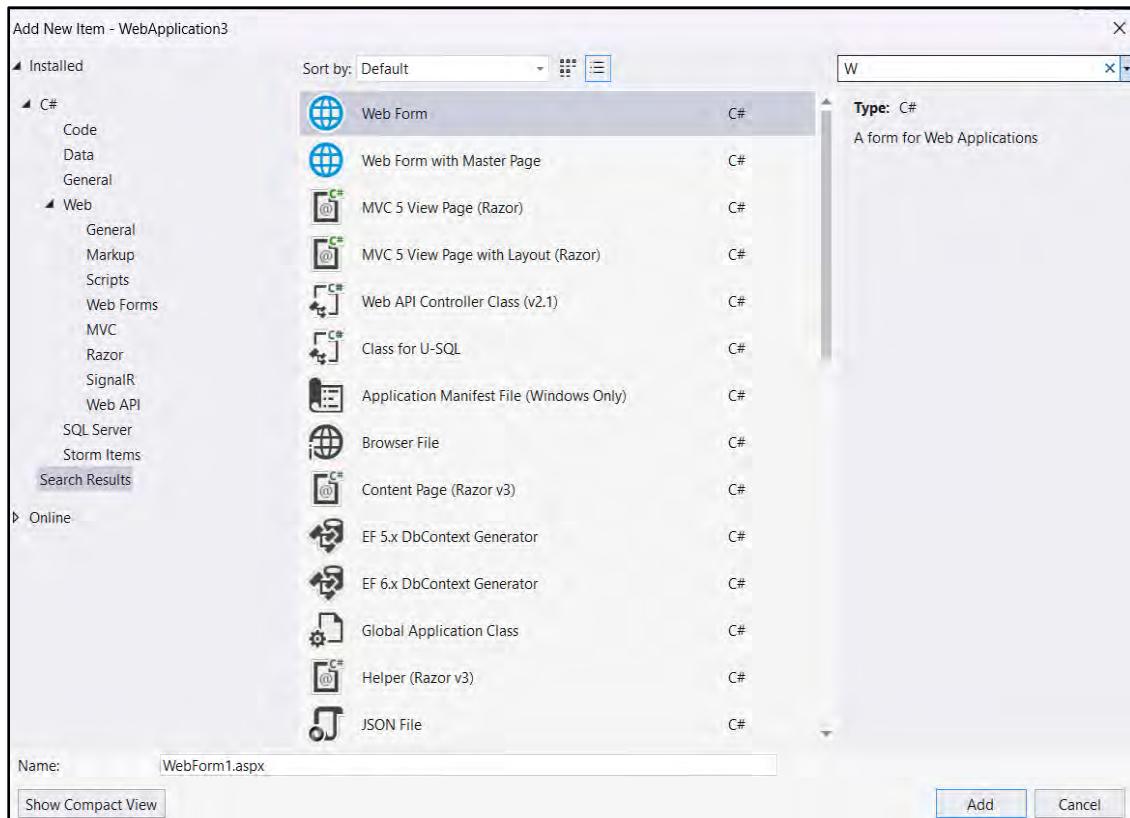


Figure 15.17: WCF Data Service Template in Add New Item Dialog

By default, the template for WCF Data Service will not be seen in Visual Studio 2022 Add New Item dialog box. Developers must add it by downloading and installing the extension.

Step 5: Right-click the project and select Add → New Item.

Step 6: In the Add New Item dialog box, select ADO.NET Entity Data Model and click Add.

Step 7: Select EF Designer from database and click Next.

Step 8: Click New Connection. This opens the connection properties dialog.

Step 9: Specify a valid server name and data source and click Test Connection, as shown in Figure 15.18. A message prompts up indicating that the test connection is successful. Click OK to fetch all details from the database server.

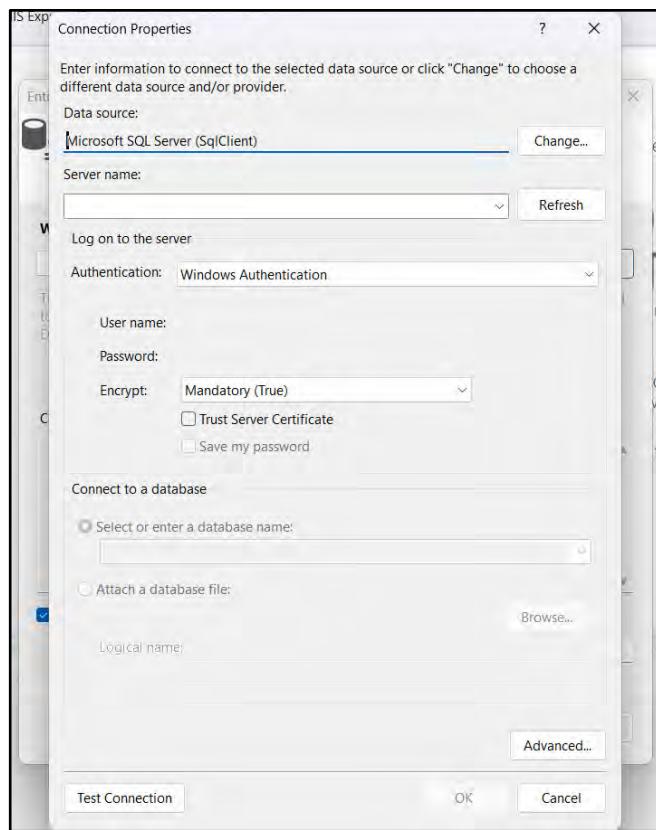


Figure 15.18: Testing the Connection

Step 10: Select the dbo and Tables check boxes and click Finish. Refer to Figure 15.19.

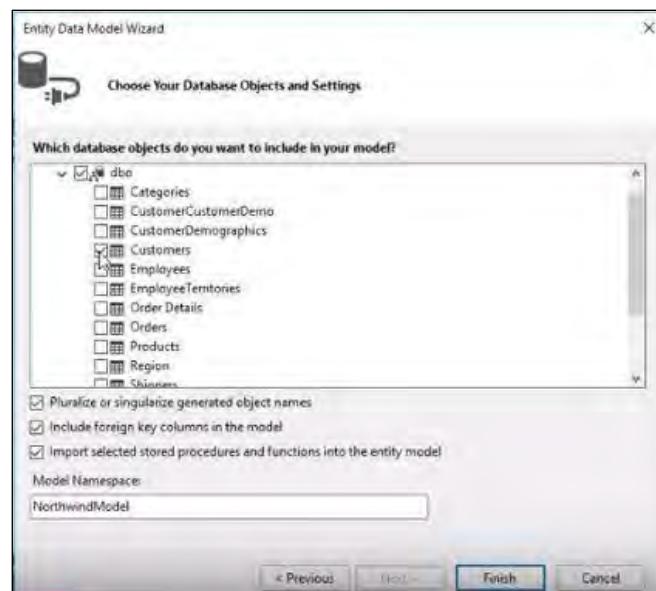
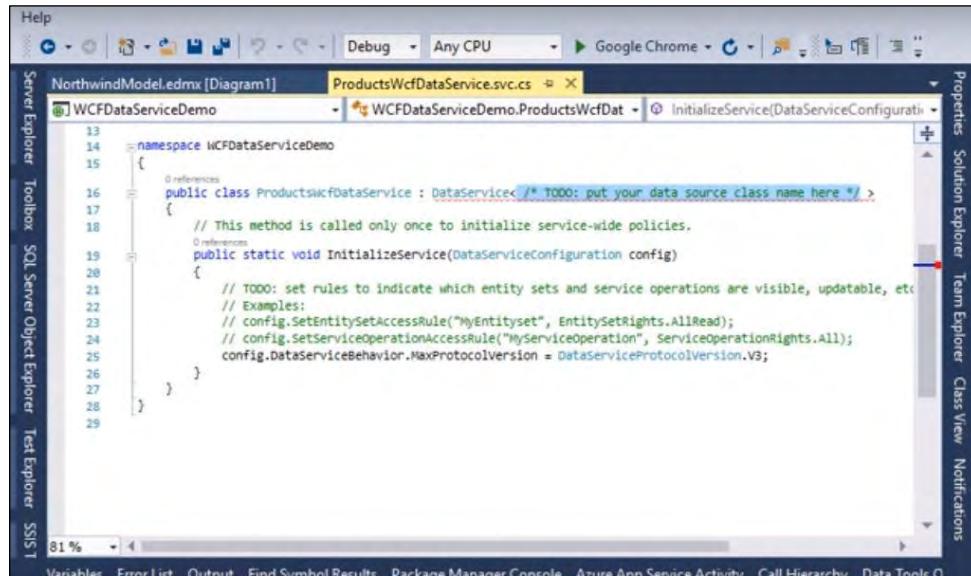


Figure 15.19: Choose Database Objects

Visual Studio creates XML markup and code files for the new service. By default, the code editor opens.

Step 11: Edit the ProductsWcfDataService.svc.cs file. Change the <TODO: put your data source class name here> to the name of entity model. Refer to Figure 15.20.



```
13
14  namespace WCFDataServiceDemo
15  {
16      // This method is called only once to initialize service-wide policies.
17      public static void InitializeService(DataServiceConfiguration config)
18      {
19          // TODO: set rules to indicate which entity sets and service operations are visible, updatable, etc.
20          // Examples:
21          // config.SetEntityTypeAccessRule("MyEntityType", EntitySetRights.AllRead);
22          // config.GetServiceOperationAccessRule("MyServiceOperation", ServiceOperationRights.All);
23          config.DataServiceBehavior.MaxProtocolVersion = DataServiceProtocolVersion.V3;
24      }
25  }
```

Figure 15.20: Change Service Model Name in Code

Step 12: Now, launch the XML format using the context menu of the application in the Solution Explorer. Refer to Figure 15.21.



Figure 15.21: View WCF Data in Browser

Step 13: In Solution Explorer, right-click the WCFDataServiceDemo Web app and select Add → Service Reference, as shown in Figure 15.22. The Add Service Reference dialog box is displayed.

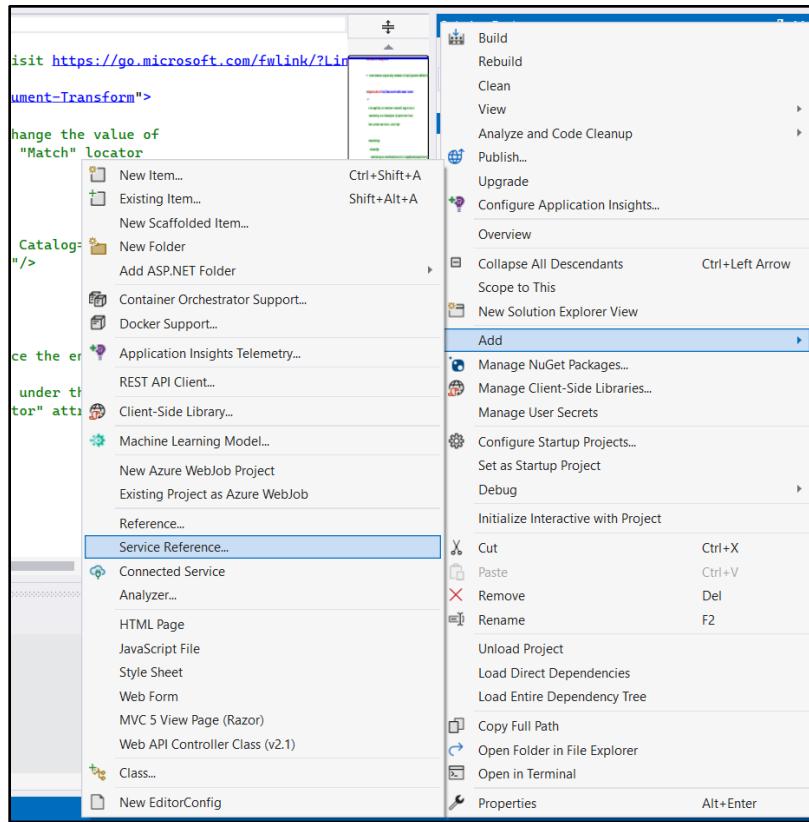


Figure 15.22: Add New Service Reference

Step 14: Enter the data service address in the Address field and click OK as shown in Figure 15.23. The address can be accessed from the browser as shown in Step 12.

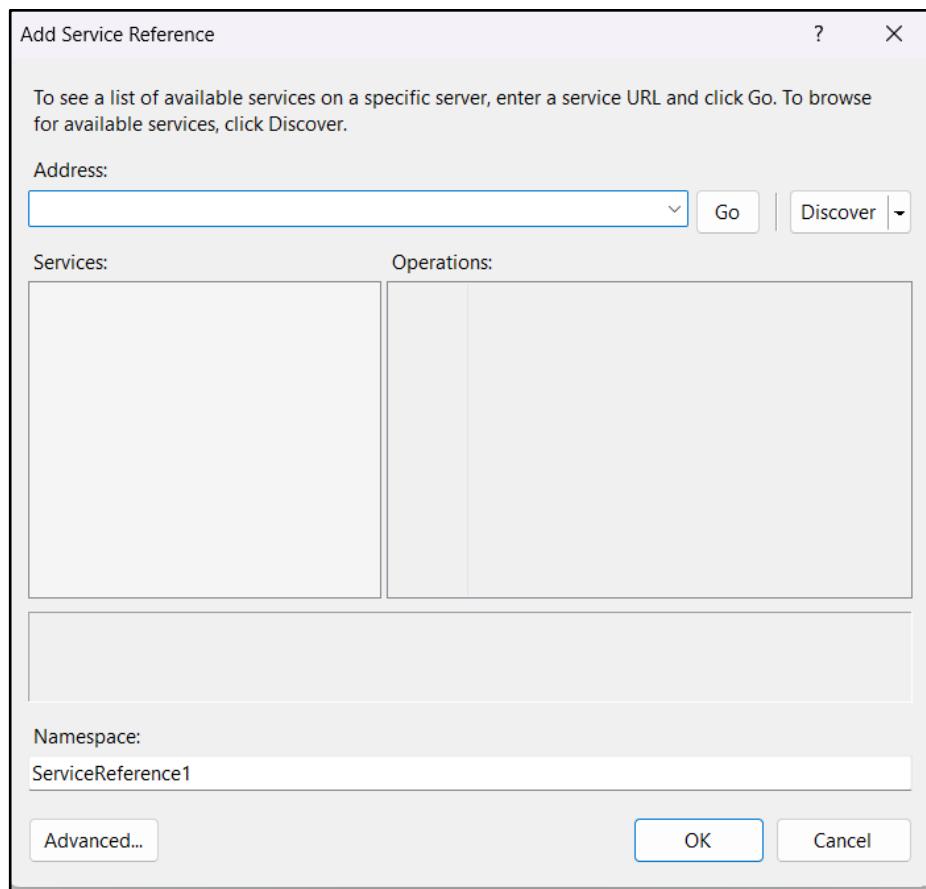


Figure 15.23: Add Service Reference Dialog

Step 15: In the Solution Explorer, right-click the Web app and select Set as StartUp Project, as shown in Figure 15.24.

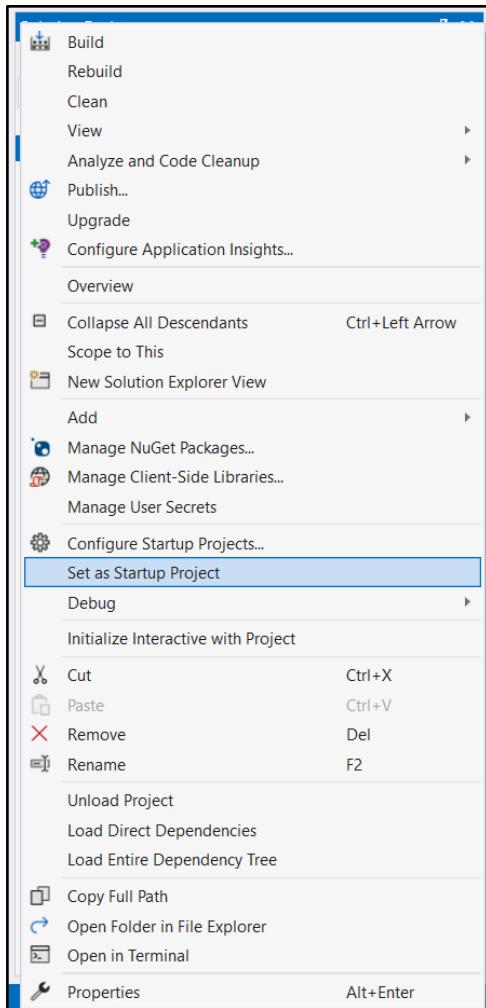


Figure 15.24: Setting Up As Startup Project

Step 16: Now, call the application using the WCF service name. The output is shown in Figure 15.25.

```
<?xml version="1.0" encoding="utf-8"?><feed xml:base="http://localhost:45529/ProductsWcfDataService.svc/" xmlns="http://www.w3.org/2005/Atom" xmlns:i="http://schemas.microsoft.com/ado/2007/08/dataservices" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"><id>http://localhost:45529/ProductsWcfDataService.svc/Products</id><title type="text">Products</title><updated>2017-08-21T06:57:19Z</updated><link rel="self" title="Products" href="Products" /><entry><id>http://localhost:45529/ProductsWcfDataService.svc/Products(1)</id><category term="Northwind\{Model.Product" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" /><link rel="edit" title="Product" href="Products(1)" /><title /><updated>2017-08-21T06:57:19Z</updated><author><name /></author><content type="application/xml"><m:properties><d:ProductID m:type="Edm.Int32">1</d:ProductID><d:ProductName>Test T</d:ProductName><d:SupplierID m:type="Edm.Int32">1</d:SupplierID><d:CategoryID m:type="Edm.Int32">1</d:CategoryID><d:QuantityPerUnit>10 boxes x 20 bags</d:QuantityPerUnit><d:UnitPrice m:type="Edm.Decimal">18.0000</d:UnitPrice><d:UnitsInStock m:type="Edm.Int16">34</d:UnitsInStock><d:UnitsOnOrder m:type="Edm.Int16">0</d:UnitsOnOrder><d:ReorderLevel m:type="Edm.Int16">10</d:ReorderLevel><d:Discontinued m:type="Edm.Boolean">false</d:Discontinued></m:properties></content></entry><entry><id>http://localhost:45529/ProductsWcfDataService.svc/Products(2)</id><category term="Northwind\{Model.Product" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" /><link rel="edit" title="Product" href="Products(2)" /><title /><updated>2017-08-21T06:57:19Z</updated><author><name /></author><content type="application/xml"><m:properties><d:ProductID m:type="Edm.Int32">2</d:ProductID><d:ProductName>Chang</d:ProductName><d:SupplierID m:type="Edm.Int32">1</d:SupplierID><d:CategoryID m:type="Edm.Int32">1</d:CategoryID><d:QuantityPerUnit>24 - 12 oz bottles</d:QuantityPerUnit><d:UnitPrice m:type="Edm.Decimal">19.0000</d:UnitPrice><d:UnitsInStock m:type="Edm.Int16">12</d:UnitsInStock><d:UnitsOnOrder m:type="Edm.Int16">48</d:UnitsOnOrder><d:ReorderLevel m:type="Edm.Int16">25</d:ReorderLevel><d:Discontinued m:type="Edm.Boolean">false</d:Discontinued></m:properties></content></entry><entry><id>http://localhost:45529/ProductsWcfDataService.svc/Products(3)</id><category term="Northwind\{Model.Product" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" /><link rel="edit" title="Product" href="Products(3)" /><title /><updated>2017-08-21T06:57:19Z</updated><author><name /></author><content
```

Figure 15.25: Output of WCF Data Service

Step 17: Now, right-click the WCFDataServiceDemo again in the Solution Explorer and select Publish. The Publish Azure Application wizard is displayed. Refer to Figure 15.26.

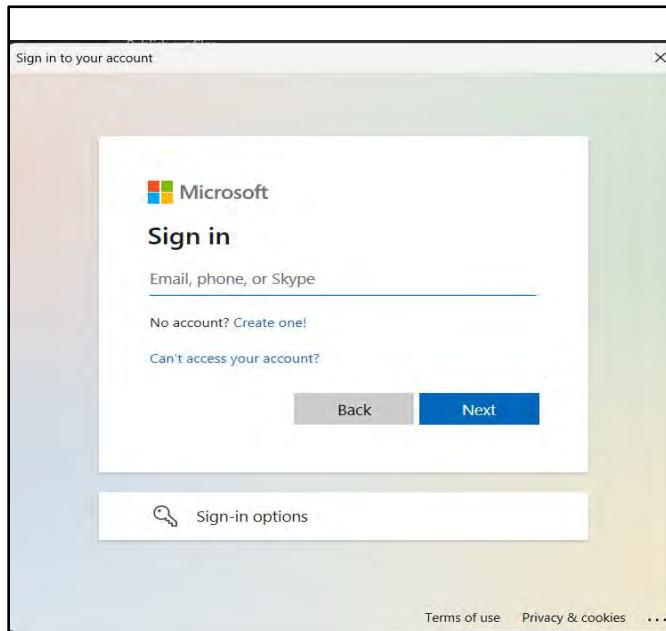


Figure 15.26: Azure Publish Sign in Wizard

Step 18: Click Next. This will open Create Cloud Service and Storage Account dialog box.

Step 19: Enter a unique hosting name in the Name field. Refer to Figure 15.27. If required, select a suitable region from the Region or Affinity Group drop-down field.



Figure 15.27: Setting Dialog Box

Step 20: Click the Create button. This opens the Remote Desktop Configuration dialog box to add credentials such as username and password to enable remote login for VM. Refer to Figure 15.28.

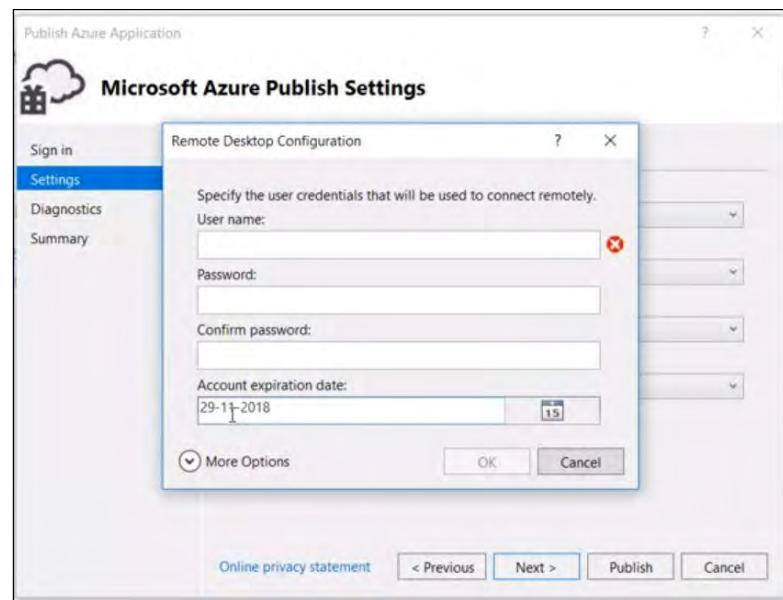


Figure 15.28: Remote Desktop Configuration

Step 21: Specify the credentials and click OK.

Step 22: Click Publish to deploy. The application will be deployed.

15.12 Azure Stack Edge, HCI, and Hub

Let us explore Azure Stack Edge, HCI, and Hub.

Azure Stack

Azure Stack Edge is the most effective way to transfer data to the cloud. It works as a network storage gateway, transferring data to Azure at high speeds. Developers can manage Edge directly from Azure portal. Models can be built and trained in Azure, run in Azure Stack Edge, and datasets can be returned to Azure. It is integrated with the Graphics Processing Unit (GPU) and Field Programmable Gate Array (FPGA) to accelerate model building and (re)training.

Azure Stack Hub

Azure Stack Hub is an Azure extension that allows you to run apps on-premises and deliver Azure services in your data center. It is built on industry-standard hardware and managed with the same tools that developers use to manage Azure subscriptions. The Azure Stack Hub architecture facilitates providing Azure services at the edge to remote locations even if one is not connected to the Internet.

Azure Stack HCI

Azure Stack HCI is a hyper converged infrastructure (HCI) cluster solution that can be used for hosting virtualized Windows and Linux workloads. It also allows storage in a hybrid environment which is a perfect combination of on-premises infrastructure with Azure cloud services. Azure hybrid services complement the cluster by providing VM backups, cloud-based monitoring, Site Recovery, and centralized management of all Azure Stack HCI deployments in the Azure portal. The cluster can be managed with existing tools, including Windows Admin Center and PowerShell.

15.13 Data Protection Regulations

Data protection regulations are laws or legal frameworks designed to safeguard the privacy, confidentiality, and integrity of personal data. These regulations aim to govern how companies acquire, process, store, and share personal

information. This ensures that individuals have control over their data and that it is handled appropriately.

Examples of prominent data protection regulations include the General Data Protection Regulation (GDPR) in the European Union, the Health Insurance Portability and Accountability Act (HIPAA) in the United States (specifically for healthcare data), and the California Consumer Privacy Act (CCPA) in California, USA.

15.13.1 General Data Protection Regulation (GDPR)

GDPR:

- Pertains to how personal data about individuals is processed within the EU.
- Requires express consent in order to process personal data.
- Gives people control over their data (portability, deletion, rectification, and access).
- Requires the reduction of data and its security via techniques such as encryption.
- Requires informing affected parties and supervisory authorities of data breaches.
- May mandate that certain organizations designate a Data Protection Officer.

15.13.2 Health Insurance Portability and Accountability Act (HIPAA)

HIPAA:

- Controls how protected health information (PHI) is used and disclosed.
- Establishes guidelines for PHI security, privacy, and breach notification.
- Mandates the use of security measures by covered entities to protect electronic PHI (ePHI).
- Requires informing the impacted parties and authorities of breaches involving unprotected PHI.
- Enforced by the Office for Civil Rights (OCR), and infractions are punishable by both civil and criminal law.

Similar other Data Protection Regulations (Example CCPA, LGPD, and so on):

- Govern the processing of personal data with specific territorial scopes.
- Grant individuals rights over their data (access, deletion, and opt-out).
- Require organizations to implement security measures to protect personal data.
- Mandate notification of data breaches to supervisory authorities and individuals.

- Enforce compliance through penalties such as fines, injunctions, or sanctions.

15.14 Azure App Service Environment (ASE)

A dedicated and isolated environment for hosting and administering Web applications is provided by Azure App Service Environment (ASE), a premium service inside Azure App Service. For businesses requiring better performance, scalability, and security for their apps, it offers a reliable answer.

The idea of ASE and its salient characteristics are explained as follows:

Isolated Environment

Web applications can be hosted in a completely dedicated and segregated environment from ASE. This is because it is powered by dedicated Azure Virtual Network (VNet) resources. Businesses can fully regulate network traffic flow and access.

Enhanced Security

- ASE provides improved security features, such as network isolation, private access, and integration with ExpressRoute and Azure Virtual Network (VNet) for safe connectivity. This is done by operating inside a dedicated Vnet.
- To further safeguard their ASE environment, organizations can install extra security mechanisms such as Azure Firewall, Application Gateway, and Network Security Groups (NSGs).

High Availability and Scalability

- ASE promotes high availability and scalability by enabling the deployment of numerous instances across several Availability Zones or regions.
- It includes auto-scaling capabilities that change resources based on application demand, assuring optimal performance and dependability.

Customization and Control

- ASE offers complete control and flexibility over the underlying infrastructure, including ability to select VM sizes, manage networking, and interface with other Azure services.
- Organizations can use ASE environment to install bespoke software, third-party components, and dependencies, giving them more flexibility and control over application deployment.

Dedicated Compute and Storage Resources

- ASE provides dedicated computation and storage resources to provide consistent performance and dependability for hosted applications.
- It integrates with Azure SQL Database, Azure Storage, and other Azure services for data storage and management.

Compliance and Regulatory Requirements

- ASE enables enterprises to meet compliance and regulatory standards by providing a dedicated and controlled environment for sensitive workloads and data.
- It supports certifications such as ISO, SOC, HIPAA, and PCI DSS, making it ideal for industries with stringent compliance standards.

15.15 Summary

- ✓ Azure App Services provides enterprise-grade development and deployment environments for both Windows-based environments and Linux-based environments.
- ✓ Azure App Services can be used to develop a variety of services such as Web App, Mobile App, Logic App, and API App.
- ✓ An App Service plan provides resources for a Web app to run in Azure App Service.
- ✓ Azure App can be subscribed in different compute plans such as Shared compute, dedicated compute, and Isolated.
- ✓ Azure App Service offers everything that is required for creating mobile back-ends, Websites, and Web APIs for any device or platform.
- ✓ The .NET Framework provides WCF data services that developers can use to create services using OData protocol.
- ✓ GDPR, HIPAA, and similar regulations oversee personal data processing, mandating consent, individual data control, security measures, breach notification, and enforcement.
- ✓ ASE provides a secure and dedicated environment for hosting Web apps with enhanced security, high availability, scalability, customization, and compliance features.

15.16 Test Your Knowledge

1. Which of the statements are true about benefits of App Services?
 - A. High-productivity Development
 - B. Enterprise-grade App Services
 - C. Fully-managed Platform
 - D. All of these

2. Which of the given plans is not an Azure App service Compute plan?
 - A. Free and Shared Compute
 - B. Virtual Compute
 - C. Dedicated Compute
 - D. Isolated Compute

3. Which step is not performed to publish ASP.NET Core app to a hosting environment?
 - A. By publishing in a folder on the hosting server
 - B. Using Process Manager
 - C. Restart IIS
 - D. Set up a reverse proxy to application

4. Which of these statements about WCF Data Services are not true?
 - A. WCF data services were formerly known as ADO.NET data services
 - B. Using OData, developers can disclose data as resources that can be addressed by URIs
 - C. OData is a REST-based protocol that allows you to perform CRUD operations on resources
 - D. The result returned by the WCF data service is only in CSV format

5. What is the name of the dialog box that appears when you attempt to publish a WCF Data service to the cloud?
 - A. Publish Azure Application
 - B. Cloud Application
 - C. Deploy Azure Application
 - D. Publish to Cloud

6. What does Scale Out in Azure do?

- A. Decreases instances
- B. Increases instances
- C. Increase Compute and Memory
- D. Decreases Compute Memory

7. Azure App Services can be used to develop which of these?

- A. Web App Services
- B. API Services
- C. Mobile App Services
- D. All of these

15.16.1 Answers to Test Your Knowledge

1. D
2. B
3. C
4. D
5. A
6. A
7. D

Try It Yourself

- Create and deploy a simple Azure Web App.
- Ensure compliance with GDPR, HIPAA, and other similar regulations.
- Develop a WCF data service and ensure it is configured correctly to connect to your data source.
- Deploy a WCF Data Service to Azure.

Appendix

Sr. No	Case Studies
1	<p>Project 1: Food Order Management System</p> <p>Scenario: A client, engaged in a food-related business under a long-term contract with a private company chain, aims to improve service efficiency. They seek a software solution to effectively manage the dietary requirements of employees within specific company premises. This initiative is intended to optimize their food catering service, ensuring greater effectiveness and efficiency.</p> <p>Additionally, the software has been designed to store all information in the cloud, enhancing data security and facilitating streamlined management.</p> <p>Functional Requirements:</p> <ol style="list-style-type: none"> 1. User Authentication: Users should be able to register for an account or log in with existing credentials. Registered users should have access to features such as viewing order history and updating account information. Authentication should be implemented securely to protect user data. 2. Record Management: Maintaining all the records of employees working in a specific company. 3. Food chart Management: Each employee had a custom food chart functionality that was divided into four parts – Morning diet details Afternoon diet details Evening diet details Night diet details. 4. Track System: There was a meal preparation tracking system used by the manager to keep track of the meal preparation progress for each of their staff members and the food delivery to the patient. Each step of the process was channelized and organized via this software system. 5. Review and Feedback: There was a daily progress checker and feedback mechanism through which the manager rated and reviewed each of the staff members for the quality of work achieved throughout their day. This system was implemented to maintain the quality of work.

	<p>Technical Requirements:</p> <ol style="list-style-type: none"> 1. Create a free Azure Subscription. 2. Create an Azure App Service after logging in to Azure account. 3. Develop and deploy a basic application. 4. Create an Azure Function to host the APIs which will interact with the database. 5. Create an Azure Cosmos DB to store data which can be retrieved using APIs. 6. Develop some basic APIs for CRUD operations and deploy them as HTTP triggers in Azure Functions. 7. Store the data in Azure Cosmos DB using SQL API. 8. Implement all the functional requirements. <p>Technologies to be used:</p> <ol style="list-style-type: none"> 1. Microsoft Azure Cloud services for backend infrastructure and services. 2. ASP.NET MVC 5.2.9 for building the Web application. 3. SQL Server Management Studio (SSMS), Azure SQL, and SQL Server 2022 for database management and administration. 4. C# 10 as the programming language for development. 5. Visual Studio 2022 Community Edition for development and debugging.
2	<p>Project 2: Weather Tracking Application</p> <p>Scenario:</p> <p>A regional emergency response organization requires a weather tracker application leveraging Azure Cloud services. The client envisions a Web application enabling users to track real-time weather updates for their selected locations.</p> <p>Additionally, the system should be capable of triggering Azure Functions to send alerts when specific weather thresholds, such as severe storms or extreme temperatures, are met.</p> <p>Functional Requirements:</p> <ol style="list-style-type: none"> 1. User Registration and Authentication: Users must be able to register and log in securely to access the application.

2. **Location Selection:** Users should be able to select their preferred location(s) for weather tracking.
3. **Real-time Weather Updates:** The application must provide real-time weather updates for the selected location(s).
4. **Weather Alerts:** The system should trigger alerts via Azure Functions when specific weather thresholds are met, based on user preferences.
5. **Accessibility:** The application should be accessible from different devices and browsers.

Technical Requirements:

1. Create an Azure App Service Web App.
2. Develop a basic Web application that uses weather APIs.
3. Containerize the application.
4. Publish the container image to Azure Container Registry.
5. Test the application using Azure Container Instance.
6. Implement an Azure Function to send alerts when a specified weather threshold is met.
7. Integrate Azure Function with the user's Web application.
8. Deploy the Web application to Azure Container Apps.
9. Set up a CI/CD pipeline for the user's application and Function.
10. Set up Application insights and Azure monitor.
11. Push to GitHub.
12. Implement all the functional requirements.

Technologies to be Used:

1. Microsoft Azure Cloud services for backend infrastructure and services.
2. ASP.NET MVC 5.2.9 for building the Web application.
3. SSMS, Azure SQL, and SQL Server 2022 for database management and administration.
4. C# 10 as the programming language for development.
5. Visual Studio 2022 Community Edition for development and debugging.

3

Project 3: E-commerce Platform

Scenario:

A global e-commerce platform seeks to modernize their infrastructure using Azure Cloud. The project entails designing and implementing a comprehensive solution, incorporating various Azure services. This includes migrating their on-premises database to Azure SQL Database for scalability and performance. They also plan to deploy a virtual machine-based application for legacy support, containerized microservices for scalability, and serverless functions for event-driven tasks. To ensure seamless integration, a messaging architecture will be employed.

Additionally, network performance and security will be optimized using Azure Virtual Network and Azure Security Center, respectively, to meet industry standards and ensure a robust and efficient infrastructure.

Functional Requirements:

1. **User Registration and Authentication:** Users must be able to register and log in securely to access the application.
2. **Product Management:** The platform should support the display of various products, categorized into different product categories or departments. Each product should have details such as name, description, price, images, and availability status. Admin users should have the ability to add, edit, or remove products from the catalog.
3. **Shopping Cart Functionality:** Users should be able to add products to their shopping cart, update quantities, and remove items as required. The shopping cart should display a summary of selected items, including prices and total order amount.
4. **Order Management:** Users should be able to proceed to checkout to complete their orders securely. The checkout process should include steps for providing shipping information, selecting payment methods, and reviewing order details before finalizing the purchase. Users should receive order confirmation emails with details of their purchases.
5. **Product Search and Filtering:** The platform should include search functionality to allow users to search for products by keywords, categories, or other criteria. Users should have the option to filter

search results based on attributes such as price range, brand, or product features.

Technical Requirements:

1. Create an Azure VM.
2. Develop and deploy a basic Web application.
3. Configure networking and security settings for the VM.
4. Create an Azure Kubernetes Service cluster.
5. Deploy a microservices application to the AKS cluster.
6. Create an Azure Functions app.
7. Write and deploy a few serverless functions.
8. Create an Azure Service Bus namespace.
9. Implement messaging between the VM-based application and the AKS microservices.
10. Create an Azure SQL Database.
11. Migrate an on-premises database to Azure using Azure Database Migration Service.
12. Create an Azure VPN Gateway to connect on-premises networks to Azure.
13. Optimize network performance using Azure Network Security Groups.
14. Implement load balancing using Azure Load Balancer.
15. Implement all the functional requirements.
16. Test the complete setup.

Technologies to be Used:

1. Microsoft Azure Cloud services for backend infrastructure and services.
2. ASP.NET MVC 5.2.9 for building the Web application.
3. SSMS, Azure SQL, and SQL Server 2022 for database management and administration.
4. C# 10 as the programming language for development.
5. Visual Studio 2022 Community Edition for development and debugging.

4 Project 4: Secure Document Management System

Scenario:

A multinational corporation seeks to improve its internal document management and collaboration capabilities. Recognizing the importance of data security and regulatory compliance, the corporation intends to leverage Microsoft Azure Cloud services to develop a Secure Document Management System (SDMS). This platform will centralize document storage, management, and collaboration while enforcing stringent security measures and compliance with data protection laws such as GDPR and HIPAA.

Functional Requirements:

1. **Document Storage and Retrieval:** Securely store and retrieve various types of documents (PDFs, Word, Excel, and so on), with support for version control.
2. **User Authentication and Access Control:** Implement robust user authentication and Role-Based Access Control (RBAC) to ensure that users can only access documents they are authorized to view or edit.
3. **Collaboration Features:** Support document sharing, commenting, and simultaneous editing to facilitate collaboration among teams.
4. **Audit Trails:** Maintain comprehensive audit trails of all actions performed on documents, including access, edits, and sharing activities.
5. **Data Encryption and Security:** Ensure end-to-end encryption of documents both in transit and at rest, alongside regular security assessments to protect against unauthorized access.
6. **Compliance and Data Protection:** Ensure the system complies with relevant data protection regulations such as GDPR and HIPAA, including features for data retention, deletion, and anonymization.

Technical Requirements:

1. **Azure Blob Storage:** For secure and scalable storage of document files.
2. **Azure SQL Database:** To store metadata, user profiles, audit logs, and access control lists.
3. **Azure Active Directory (AAD):** For user authentication and implementing RBAC.
4. **Azure Key Vault:** To manage encryption keys and secrets securely.
5. **Azure Functions:** To automate workflows, such as document processing, notifications, and compliance checks.
6. **Azure App Service:** To host the Web application front end.
7. **Azure DevOps:** For Continuous Integration and Continuous Deployment (CI/CD), ensuring smooth and secure updates to the platform.
8. **Azure Application Insights and Azure Monitor:** For monitoring the system's performance and security, ensuring quick identification and resolution of any issues.
9. **Containerization (Docker) and Azure Kubernetes Service (AKS) or Azure Container Apps:** For deploying and managing microservices architecture, ensuring scalability and resilience.
10. **Azure Information Protection:** To classify and protect documents based on sensitivity.
11. **GitHub:** For source code management and version control.

Technologies to be Used:

1. Microsoft Azure Cloud services for backend infrastructure.
2. ASP.NET MVC 5.2.9 for building the Web application.
3. Entity Framework Core for data access.
4. Angular or React for the frontend development.
5. C# 10 for backend logic.
6. Visual Studio 2022 Community Edition for development.

5 Project 5: Smart City Traffic Management System

Scenario:

A city council aims to develop a Smart City Traffic Management System to optimize traffic flow, reduce congestion, and enhance road safety. This initiative involves leveraging real-time data from traffic sensors, cameras, and IoT devices deployed across the city. The system will analyze traffic patterns, predict congestion points, and dynamically adjust traffic signal

timings. Additionally, it aims to provide citizens with real-time traffic updates and alternate route suggestions. The system will be built using Microsoft Azure Cloud services to ensure scalability, reliability, and real-time data processing capabilities.

Functional Requirements:

1. **Real-time Traffic Monitoring:** Collect and process real-time data from traffic sensors and cameras to monitor traffic conditions across the city.
2. **Traffic Prediction and Management:** Use predictive analytics to forecast traffic congestion and dynamically adjust traffic signals to optimize flow.
3. **User Interface for Citizens:** Provide a Web and mobile platform for citizens to view real-time traffic conditions, receive updates, and get alternate route suggestions.
4. **Emergency Vehicle Priority:** Implement a system to detect emergency vehicles and prioritize their movement through traffic lights.
5. **Data Security and Privacy:** Ensure the system adheres to data protection regulations, securing data transmission and storage.
6. **Integration with Public Transport Systems:** Integrate with existing public transport systems to offer comprehensive mobility solutions and encourage the use of public transport.

Technical Requirements:

1. **Azure IoT Hub:** To securely connect, manage, and ingest data from traffic sensors and IoT devices deployed throughout the city.
2. **Azure Stream Analytics:** For processing large streams of real-time data, enabling quick analysis and response to changing traffic conditions.
3. **Azure SQL Database:** To store and manage structured data, including traffic analytics, user preferences, and system configurations.
4. **Azure Functions:** To run serverless computing tasks, such as sending real-time traffic alerts to citizens and processing data from emergency vehicles.

5. **Azure App Service:** For hosting the Web and mobile applications that provide user interfaces for citizens and city traffic management staff.
6. **Azure Traffic Manager:** To ensure high availability and low latency for the applications, directing users to the nearest data center.
7. **Azure API Management:** To securely expose APIs for integration with other city systems, including public transport and emergency services.
8. **Azure Cognitive Services:** To analyze video feeds from traffic cameras for vehicle and pedestrian detection, enhancing traffic flow management.
9. **Azure DevOps:** For Continuous Integration and Continuous Deployment (CI/CD) of the system, ensuring rapid updates and maintenance.
10. **Azure Active Directory (AAD):** For secure authentication and authorization of users accessing the system, including city staff and citizens.
11. **Azure Monitor and Application Insights:** For monitoring the system's performance, tracking usage statistics, and identifying areas for improvement.

Technologies to be Used:

1. Microsoft Azure Cloud services for the infrastructure and data processing.
2. ASP.NET MVC 5.2.9 for developing Web applications and RESTful APIs.
3. Blazor or Angular for creating interactive and responsive user interfaces.
4. Entity Framework Core for data access and management.
5. Visual Studio 2022 for development, testing, and deployment activities.

6 Project 6: Healthcare Patient Management System

Scenario:

A healthcare provider wants to develop a Patient Management System to digitalize patient records, appointments, and treatments to enhance the efficiency of healthcare services and patient care. The system aims to provide a secure, accessible, and comprehensive platform for healthcare staff to manage patient data, track health records, schedule appointments, and facilitate telehealth consultations. Leveraging Azure Cloud services, the system will ensure high availability, data security, and compliance with health regulations such as HIPAA.

Functional Requirements:

1. **Digital Patient Records:** Digitize and manage patient records, including medical history, treatment plans, and test results with secure access controls.
2. **Appointment Scheduling:** Enable patients and healthcare providers to schedule, update, or cancel appointments through a Web interface.
3. **Telehealth Integration:** Support telehealth consultations, allowing doctors to conduct video consultations with patients remotely.
4. **Access Control and Security:** Implement strict access controls and authentication mechanisms to ensure patient data confidentiality and security.
5. **Compliance and Data Protection:** Ensure the system complies with healthcare regulations such as HIPAA for data protection and patient privacy.
6. **Reporting and Analytics:** Provide healthcare staff with tools for reporting and analytics on patient data for better healthcare outcomes.

Technical Requirements:

1. **Azure SQL Database:** To store patient records, appointment schedules, and other structured data securely and efficiently.
2. **Azure App Service:** For hosting the Web application that serves as the interface for patients and healthcare providers.

3. **Azure Active Directory (AAD)**: To manage user identities and access controls, ensuring that only authorized users can access sensitive patient data.
4. **Azure Kubernetes Service (AKS)**: To deploy and manage the application's microservices architecture, ensuring scalability and reliability.
5. **Azure Blob Storage**: For storing unstructured data such as medical images and documents securely.
6. **Azure Functions**: For serverless computing tasks, such as sending automated appointment reminders and processing telehealth video consultations.
7. **Azure API Management**: To securely expose APIs for integration with other healthcare systems, including electronic health records (EHR) and telehealth platforms.
8. **Azure DevOps**: For CI/CD pipelines, automating the build, test, and deployment processes for application updates.
9. **Azure Monitor and Application Insights**: For monitoring the application's performance, diagnosing issues, and ensuring a smooth user experience.
10. **Azure Key Vault**: To manage and safeguard encryption keys and secrets used in the application, enhancing data security.

Technologies to be Used:

1. Microsoft Azure Cloud services for backend infrastructure and services.
2. ASP.NET MVC 5.2.9 for building the Web application.
3. Entity Framework Core for ORM and data access.
4. Visual Studio 2022 Community Edition for development and debugging.
5. SSMS, Azure SQL, and SQL Server 2022 for database management and administration.