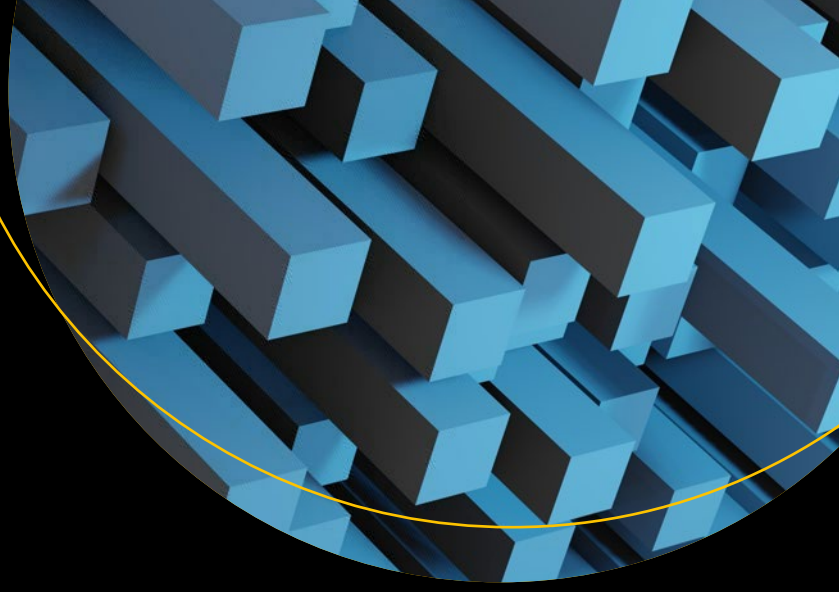

Developing Cloud-Native Solutions with Microsoft Azure and .NET

Build Highly Scalable Solutions for the Enterprise

—

Ashirwad Satapathi
Abhishek Mishra

Apress®

Developing Cloud-Native Solutions with Microsoft Azure and .NET

Build Highly Scalable Solutions for the Enterprise

Ashirwad Satapathi
Abhishek Mishra

Apress®

Developing Cloud-Native Solutions with Microsoft Azure and .NET: Build Highly Scalable Solutions for the Enterprise

Ashirwad Satapathi
Gajapati, Odisha, India

Abhishek Mishra
Navi MUmbai, India

ISBN-13 (pbk): 978-1-4842-9003-3
ISBN-13 (electronic): 978-1-4842-9004-0
https://doi.org/10.1007/978-1-4842-9004-0

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Smriti Srivastava
Development Editor: Laura Berendson
Coordinating Editor: Shrikant Vishwakarma
Copyeditor: William McManus

Cover designed by eStudioCalamar

Cover image by Yaroslav A on Unsplash (www.unsplash.com)

Distributed to the book trade worldwide by Apress Media, LLC, 1 New York Plaza, New York, NY 10004, U.S.A. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub (https://github.com/Apress). For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

The book is dedicated to my mother, Mrs. Sabita Panigrahi, for supporting me through every phase of my life.
—Ashirwad Satapathi

This book is dedicated to my parents, wife, and lovely daughter Aaria.
—Abhishek Mishra

Table of Contents

About the Authors

Ashirwad Satapathi is a software developer with a leading IT firm whose expertise is building scalable applications with .NET Core. He has a deep understanding of how to build full-stack applications using .NET and Azure PaaS and serverless offerings. He is an active blogger in the C# Corner developer community. He was awarded the C# Corner MVP (September 2020) for his remarkable contributions to the developer community.

Abhishek Mishra is a cloud architect at a leading organization and has more than 17 years of experience in building and architecting software solutions for large and complex enterprises across the globe. He has deep expertise in enabling digital transformation for his customers using the cloud and artificial intelligence. He speaks at conferences on Azure and has authored five books on Azure prior to writing this new book.

About the Technical Reviewer

As a Cloud Solutions Architect, **Viachaslau Matsukevich** has delivered 20+ DevOps projects for a number of Fortune 500 and Global 2000 enterprises.

Viachaslau has been certified by Microsoft, Google, and the Linux Foundation as a Solutions Architect Expert, Professional Cloud Architect, and Kubernetes Administrator.

He has written many technology articles about Cloud-Native technologies and Kubernetes on *Red Hat Enable Architect, SD Times, Hackernoon, Dzone,* and *Medium's* largest and most followed independent DevOps publication.

Viachaslau has participated as an Industry Expert and Judge for the Globe Awards, including the IT World Awards, Golden Bridge Awards, Disruptor Company Awards, and American Best in Business Awards. He also is the author of online courses about DevOps and Kubernetes tools and is a member of the Leader of Excellence from Harvard Square. You can find him at `www.linkedin.com/in/viachaslau-matsukevich/`.

Acknowledgments

We would like to thank the Apress team for giving us the opportunity to work on this book. Also, thanks to the technical reviewer and the editors for helping us deliver this manuscript.

Introduction

This book will help you learn how to build cloud-native solutions using Microsoft Azure and .NET. It provides step-by-step explanations of essential concepts and practical examples. We will begin by exploring multiple Azure PaaS and serverless offerings by building solutions to solve real-world problems by following the industry best practices.

The books start with Azure fundamentals, followed by scenario-driven chapters on building distributed solutions using Azure Web App and Azure Service Bus. The next set of chapters focuses on building containerized workloads using Azure container-based services. We will explore how to build intelligent applications using Azure AI and IoT services in subsequent chapters. Finally, we will explore ways to deploy our applications using GitHub Actions.

By the end of this book, you will be able to build various types of applications by leveraging Azure PaaS and serverless services and be able to deploy them using Azure GitHub Actions.

This book presents near-production scenarios and provides lab scenarios that deliver the right set of hands-on experience to the reader. The practical approach adopted in the book will help users gain deep proficiency in developing coud-native solutions.

This book is intended for all who wish to get a deep understanding of Azure to build highly scalable and reliable cloud-native applications. It provides a well-illustrated guide for the readers to kickstart their journey with Microsoft Azure.

Source Code

All source code used in this book can be downloaded from `https://github.com/apress/developing-cloud-native-solutions-azure-.net`.

CHAPTER 1

Introduction

Modern applications are being migrated to the cloud like never before. All new applications designed using new-age architectures like microservices, Onion Architecture, and many more are hosted on the cloud. Your applications hosted on the cloud are highly available, fault-tolerant, reliable, and scalable. Cloud computing helps you save infrastructure ownership costs and operational costs to a large extent.

There are many cloud vendors today from which you can rent cloud services. Microsoft Azure is popular among these vendors and is preferred across many industries, such as manufacturing, finance, healthcare, aviation, and others.

In this chapter, we will explore the following Azure-related topics:

- Introduction to cloud computing
- Cloud deployment models
- Cloud service types
- Serverless computing
- A quick tour of Azure services

After studying this chapter, you will understand the fundamentals of cloud computing and the cloud services offered by Azure. Let's start with a brief introduction to what exactly cloud computing is.

Introduction to Cloud Computing

A cloud vendor builds and manages a large number of datacenters. Using virtualization technologies, it spins up virtual machines. It runs many cloud services for hosting applications, storage, databases, artificial intelligence (AI), machine learning (ML), Internet of Things (IoT), and many more. All these services running on the cloud vendor's datacenter are available to the consumers by using a portal or a command-line

A. Satapathi and A. Mishra, *Developing Cloud-Native Solutions with Microsoft Azure and .NET*,
https://doi.org/10.1007/978-1-4842-9004-0_1

utility. As a consumer, you can use these services and pay as and when you use them. You just need to request the creation of these services on the cloud vendor datacenter using the portal or the command-line utility. Once the services get created, you can start using them. You get billed for the period you are using it. And when you do not use it, you can decommission these services. Creation and decommission of these services are instantaneous. In a few minutes, you can create and delete these services using the vendor portal or a command-line utility. As soon as you delete the services, your billing stops. Hosting an application on the cloud is more cost-effective than hosting the application on an on-premises server.

The cloud vendor manages the datacenter. The cloud vendor owns the infrastructure and maintains it, saving you the effort and cost of owning and maintaining it. You simply consume the services without worrying about the underlying hosting infrastructure. You save both capital expenditure (CapEx) by not needing to own the infrastructure and operational expenditure (OpEx) by not needing to maintain the cloud infrastructure.

There are industry-standard cloud architectures and practices that you can use to build cloud-based applications. You can execute a service level agreement (SLA) with the vendor for the cloud services you plan to consume. The vendor will guarantee that the applications running on the cloud are highly available, reliable, secured, scalable, and fault-tolerant.

Microsoft Azure, Amazon Web Services (AWS), and Google Cloud Platform are the most popular cloud vendors. However, there are a lot of other cloud vendors that you can rely on.

Cloud Deployment Models

You can deploy your application to the cloud using the following deployment models:

- Public cloud
- Private cloud
- Hybrid cloud

Let's explore each of these deployment models in detail.

Public Cloud

Large cloud vendors have built massive datacenters spread across the globe. They manage these datacenters and have complete control over the datacenter infrastructure. These cloud vendors have enabled a very high degree of virtualization on these datacenters and have developed cloud services running on the virtualized environment on top of the datacenter infrastructure. Anyone across the globe can purchase a subscription for these cloud services and start using them.

As a consumer of these cloud services, you need not worry about owning and managing the underlying infrastructure hosting the cloud services. You just need to pay for the cloud service you are using. The cloud vendor will ensure that the hardware is always up and running and the cloud service is available.

You can create a cloud service on the public cloud in a few minutes, and when you no longer need the service, you can decommission it. Your billing starts when the service gets provisioned, and the billing stops as soon as you decommission the service. You end up paying the cloud vendor only when using the service. This approach saves much cost for you.

Microsoft Azure, Amazon Web Service, and Google Cloud Platform are examples of public cloud providers.

Private Cloud

Large enterprises and organizations execute thousands of customer projects. To cater to the projects they are executing, they can build their datacenter, run the virtualized cloud environment on their datacenter, and host their customer projects. The enterprise owns the datacenter and manages it. It has complete control over the infrastructure and tighter control over the data, network, and infrastructure level security. The projects in the enterprise can host their applications on the private enterprise cloud.

Building and managing a private cloud is costly compared to running the applications on the public cloud. However, the enterprise gets greater control over the cloud infrastructure than it would have over the public cloud.

Hybrid Cloud

A hybrid cloud is a combination of a public cloud and a private cloud. Some application components are hosted on the public cloud and some are hosted on the private cloud.

The public and private cloud components communicate with each other using a dedicated and secured communication channel. For example, you may choose to keep the data on the private cloud and host the application on the public cloud.

Cloud Service Types

There are three types of cloud services:

- Infrastructure-as-a-Service (IaaS)
- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)

Let's discuss each of the service types in detail.

Infrastructure-as-a-Service

Virtual machines run on cloud-based datacenters. You can build your application and deploy it on a virtual machine. The hosting experience is precisely the same as that in the on-premises server. You get complete control over the virtual machine and its operating system. You can install any software, configure the hosting environment based on your needs, and then run your application without any restriction, just like an on-premises server.

You get greater control over the virtual machine and its environment. However, you will have to spend much operational expenses to keep the virtual machine running. You will have to patch the operating system with the latest security fixes and keep upgrading the installed software and the application-hosting environment.

Platform-as-a-Service

In the case of Platform-as-a-Service, you build your application and data and deploy it to the cloud service without worrying about the underlying hosting infrastructure. The cloud vendor manages all operational aspects of the virtual machine hosting your application and data. You do not have any control over the underlying infrastructure or virtual machine hosting your application. However, you can configure the hosting environment by setting the configuration variables for your application to some extent.

The same virtual machine can host two different applications running on a PaaS-based cloud service in isolation.

Platform-as-a-Service is cheaper than Infrastructure-as-a-Service.

Software-as-a-Service

In the case of Software-as-a-Service, you build your application and host it on the cloud. End users can use an instance of your application by paying a subscription fee for using the application features. You can bill the end users on the number of features and functionalities they choose to use. The end users can configure the data and security for their application instance.

Microsoft 365, Microsoft Dynamics 365, and Netflix are examples of Software-as-a-Service.

Serverless Computing

In the case of serverless computing, you get billed only when the service hosting your code executes. You do not get charged when the service is idle and not doing any job. You can build your application and host it on the serverless service without concern for the underlying hosting infrastructure. The cloud vendor manages the hosting infrastructure, and you have no control over your application's infrastructure. The serverless services scale on their own. You need not set any scale settings. The services will scale automatically, and new service instances will get added or deleted based on the need. You have no control over how the service scales.

Serverless computing is a cheaper hosting option than all the service types we have discussed, if you design it accurately. You get charged for the computing and resources consumed by each service instance. You must design your cloud solution to control the number of service instances that get provisioned when the service scales.

Note Although serverless computing appears to be the same as Platform-as-a-Service, they are different in some respects. In both cases, you do not have any control over the underlying hosting infrastructure. However, in the case of PaaS, you can control the scaling behavior of the application by setting the manual or automatic scale settings. In the case of serverless services, you do not have any

control over how the services scale. In the case of PaaS, your billing starts as soon as you have provisioned the service. You get charged even if the service is idle. However, in the case of serverless services, you get billed only when the service hosting your code is doing some work. You are not billed when the service is idle in the case of serverless computing.

A Quick Tour of Azure Services

Microsoft Azure is a popular cloud choice for many global and large enterprises. It provides a wide range of cloud services suiting your application needs. Using Azure, you can build solutions that are highly reliable, secured, highly available, and fault-tolerant. Let's take a look at some of the service offerings from Azure.

Compute Services

Let us discuss the popular computing services used frequently in cloud-based solutions.

Azure Virtual Machines

Azure Virtual Machines is an IaaS offering. You can host your application on a virtual machine just like you would host it in the on-premises environment. You get greater control over the underlying hardware and the hosting environment. Once your application is ready, you need to spin up the virtual machine with your desired operating system and compute size. You then need to install the hosting software, such as Tomcat or Microsoft Internet Information Services (IIS), and all the application dependencies the application needs to run. Once you are done with these prerequisite steps, you can host your application. You can choose a virtual machine based on the compute size that meets your application needs. If your application is memory intensive, you can choose an Azure D-series, Ds-series, E-series, or M-series virtual machine SKU. If your application is compute intensive, you can choose a high-performance SKU like the H-series, HB-series, or HC-series virtual machine.

Azure App Service

Azure App Service is a PaaS offering. You can build your application using a supported programming language like C# (.NET and .NET Core), Java, PHP, Python, or one of many others, and then host the application on the App Service without concern for the underlying infrastructure. You need to spin up the App Service based on your need and select a plan that meets your computing needs. App Service creates the necessary infrastructure on which you can run your application. Your application runs in a shared environment on a virtual machine. The virtual machine hosts multiple applications, but in isolation from each other. You do not have any control over the virtual machine hosting your application. The underlying Azure platform creates the hosting environment for you, and you need to deploy your application on it. You can set the scale settings that can be either manual or automated.

Azure Functions

Azure Functions is a serverless offering. You need to build your application and host it on the Azure Function. You do not have any access to the underlying hosting environment and the infrastructure running your Function code. You get billed only when the code executes and are not charged when the Function is idle. You do not have control over how the Function scales if you use the consumption or the premium plan. New Function instances will get added or removed based on the incoming traffic load. If you use the dedicated plan, then the Function behaves like an App Service, and you can control the scaling behavior just like in the case of an App Service.

Azure Logic Apps

Azure Logic Apps is a serverless offering. It helps you integrate with a wide range of Azure services, on-premises services, and other cloud services with ease using connectors. You can build graphical workflows without having to write any code. You can trigger these workflows using triggers. Each workflow step performs an action you can configure based on your need. Azure Logic Apps helps you build enterprise-grade integration solutions.

Azure Kubernetes Service

Azure Kubernetes Service (AKS) is managed Kubernetes on Azure that helps you spin up the Kubernetes control plane along with the worker nodes, known as virtual machine scale sets. You can manage the worker nodes, but you do not have any control over the control plane. The underlying Azure platform manages the control plane. The application containers run inside the pods in the worker nodes.

You can scale your application running on the Azure Kubernetes Cluster by adding the number of nodes or pods. Scaling the nodes can be challenging, as the virtual machines may take some time to get added to the virtual machine scale set. To mitigate this challenge, you can use Azure Container Instances as serverless nodes. Azure Container Instances is a container-based offering on Azure and can run a single container.

You can also run Azure Functions inside the Azure Kubernetes Service cluster and bring in serverless capability to the Azure Kubernetes Service by using Kubernetes Event-Driven Autoscaling (KEDA).

Data Services

Now let's discuss the Azure-based data services that are most frequently used.

Azure Storage Account

Azure Storage Account provides blobs, files, queues, and tables to store data. You can use the blob storage to keep blob files like images, text files, videos, and many more types. You can build distributed applications leveraging queues. The distributed application components can communicate using the queues. The sender application component can ingest messages in the queue that the receiver application component can receive and process. Tables store semi-structured data as key/value pairs. Azure Files helps you keep and organize folders and files in a share. You can work with the files using the standard SMB or NFS protocol or HTTP requests. Data stored in the Storage Account is secured using Access Keys.

Azure Data Lake

The first generation of Azure Data Lake was based on HDFS file systems running on Hadoop clusters. Hadoop clusters are expensive. The blob storage in the Azure Storage

Account hosts the second-generation Azure Data Lake. The blob storage is robust enough to support the storage of a huge amount of unstructured data, and you can process the data easily and quickly. You can create a hierarchical folder structure and keep your data.

Azure SQL

Azure SQL is PaaS-based Microsoft SQL Server offering on Azure. You need not own and manage a SQL server to host the SQL database. You can provision the Azure SQL service, and the underlying Azure platform will spin up a SQL Server and give you a database where you can host your data. Azure SQL does not support all features of Microsoft SQL Server, like SQL Server Integration Services (SSIS) and SQL Server Reporting Services (SSRS).

Azure Data Factory

Azure Data Factory helps you connect with different data stores on Azure, on-premises, or in other clouds and process the data. It provides a wide range of connectors with which you can connect to different data stores. You can build pipelines that can get the data from a data store, process it, and keep the processed data in another data store on Azure, on-premises, or in other clouds. You can schedule the pipelines to process the data or trigger the pipelines based on the need.

Azure Synapse

Azure Synapse is a Datawarehouse on Azure. You can perform data integration, data analytics and data visualization, and warehousing of your data. You can work with the Datawarehouse using either Scala, Python, SQL, .NET, or Java. Azure Synapse supports either a SQL or Spark analytics runtime that you can leverage to perform analytics on the big data. You can build pipelines and execute them to perform data processing on a vast amount of data and derive insights from complex datasets. You can also leverage Databricks to process and analyze data managed by Azure Synapse. Azure Synapse provides capabilities to build machine learning solutions based on big data.

Artificial Intelligence– and Machine Learning–Based Services

Now let's discuss the popular offerings on Azure that you can leverage to build complex artificial intelligence– and machine learning–based solutions.

Azure Cognitive Services

Azure Cognitive Services helps you build artificial intelligence solutions easily without needing to build any AI models. Microsoft has already developed these models and hosts them on Azure. You consume these services using REST APIs or client library SDKs. You can perform complex activities like extracting meaningful information from videos and images, analyzing text sentiments and languages, face detection, speech to text and vice versa, language translation, and many more complex AI use cases.

Azure Machine Learning

Azure Machine Learning helps you build and expose custom machine learning models that you can use across your applications. It provides rich capabilities and predefined ML models that you can use to author workflows for your ML model. You need not write much code to build the models. Using Azure Machine Learning Studio, you can use drag-and-drop features to build and train your models.

Azure Bot Service

You can build conversational applications or bots using Azure Bot Framework and host them on Azure Bot Service. You can leverage Azure Cognitive Services and Azure Machine Learning to build intelligent bots that can also process natural user languages.

Other Services

Let's discuss a few other Azure Services we can use in our Azure solutions.

Azure API Management

You can build and host APIs on Azure Functions, Azure App Service, Azure Kubernetes Service, or any other service on Azure. Using Azure API Management, you can manage

the hosted APIs. You can set Cross-Origin Resource Sharing (CORS) rules, perform redirection, tweak headers, check for valid access, and conduct many other such API management activities using these services. You can inspect and act on the requests and responses for the APIs.

Azure Active Directory

Azure Active Directory (Azure AD) is an identity and access management offering on Azure. You can perform authentication, authorization, and single sign-on (SSO) for your applications running on Azure, on-premises, or on other clouds. You can create users and roles in Azure AD and even join the on-premises domain controller to Azure AD to integrate on-premises users to Azure. You need not set up any additional infrastructure to handle authentication and authorization for your applications. You need to configure your applications to use Azure AD for authentication and authorization purpose.

Azure Monitor

Azure Monitor helps you collect logs and performance metrics for your application and infrastructure on Azure, on-premises, or on other clouds. You can analyze these logs and metrics for debugging purposes or gather additional insights. You can set performance alerts on the logs and metrics to notify the user if the application fails or performance counters fall below the expected level. Azure Monitor also integrates with IT Service Management (ITSM) tools like ServiceNow and automatically logs tickets in case of performance issues and failures.

Summary

In this chapter, you learned about the basic cloud concepts and the cloud deployment models. We explored the details of the public cloud, private cloud, and hybrid cloud deployment models. We also covered the available cloud service types, including Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS), and the concept of serverless computing. We wrapped up by surveying several popular Azure services at a very high level.

In the next chapter, you will learn how to build a Web API that can send messages to Azure Service Bus Queue.

CHAPTER 2

Build a Web API to Send Messages to Azure Service Bus

In the previous chapter, we briefly discussed key Azure services that are used to build resilient, scalable, and secure cloud native solutions. Many of these solutions leverage the power of message brokers to enable inter-application or service communication in an asynchronous manner. There are many such messaging systems available, including Apache Kafka, RabbitMQ, Azure Service Bus, ActiveMQ, and Amazon MQ. The focus of this chapter is to build decoupled solutions using Azure Service Bus, which is the enterprise-grade messaging solution offered by Microsoft Azure.

We will explore the key concepts and features of Azure Service Bus by building solutions using ASP.NET Core Web API for a fictional dental clinic to schedule appointment notifications for patients in a Service Bus Queue. The solution will be able to schedule a message for a future date as well as cancel it if the message has not been enqueued or has not been processed yet.

In this chapter, we will explore the following topics:

- Introduction to Azure Service Bus
- Features of Azure Service Bus
- Getting started with Azure Service Bus: Queue and Topic
- Provision an Azure Service Bus instance
- Build an ASP.NET 6 Web API to send messages to Azure Service Bus Queue

A. Satapathi and A. Mishra, *Developing Cloud-Native Solutions with Microsoft Azure and .NET*,
https://doi.org/10.1007/978-1-4842-9004-0_2

After studying this chapter, you should have a solid grasp of the fundamentals of Azure Service Bus, as well as cloud services on Azure.

Introduction to Azure Service Bus

Enterprise messaging software, also known as message-oriented middleware, has been in existence since the late 1980s. Message-oriented middleware is a concept that involves the passing of data between applications using a communication channel that delivers messages from the sender to the receiver. These messages are transferred in an asynchronous manner between the sender and receiver using a message broker. These kinds of messaging systems help in decoupling applications and keeping the sender and receiver anonymous from each other. Azure Service Bus is one such messaging solution offered by Microsoft Azure as part of its cloud suite.

Azure Service Bus is an enterprise-grade, multitenant cloud messaging service of Microsoft Azure. It is a Platform-as-a-Service (PaaS) offering of Azure and is fully managed by Microsoft. As developers, we don't need to worry about tasks such as handling of backups or hardware failures for the services because Azure takes care of it for us. Azure Service Bus supports point-to-point message delivery with Queues and message broadcasting using the publisher-subscriber model with Topics. In a later section of the chapter, we are going to briefly discuss Azure Service Bus Queues and Topics.

Azure Service Bus allows applications to interact or transfer data with each other using messages. It provides an excellent mechanism to build decoupled solutions and make them more scalable, reliable, and resilient. The producer should not be impacted due to the unavailability of the consumer. It supports data in different formats, such as JSON, XML, plain text, and so on, and provides client libraries in different languages to build solutions using it. Apart from that, it offers various advanced features such as message deferral, duplicate detection, and auto-forwarding, to name a few.

When to Use Azure Service Bus

There are various scenarios in which using Azure Service Bus would be appropriate and helpful for the overall function of our application as well as the architecture of the system. Some of them are listed here:

- Increasing scalability of the application
- Replacing RPCs
- Integrating heterogeneous applications
- Reducing the coupling between applications

There are many additional scenarios where Azure Service Bus can be handy to use. Let's discuss how Service Bus can help in the case of heterogeneous applications.

Suppose we have two applications, Application A and Application B. Application A is written in Java while Application B is written in Python. To fulfill some requirements, we want to establish communication between these applications.

Normally, this would be really difficult to do between two applications written in different languages unless you have a common medium of communication between them.

Service Bus helps us do so without a lot of trouble. Service Bus provides clients for multiple languages, including .Net, Python, Ruby, and many more, which helps in sending messages between heterogeneous applications via its message broker.

Features of Azure Service Bus

The following are some of key features of Azure Service Bus:

- Message filtering
- Message sessions
- Message auto-forwarding
- Duplicate detection
- Message expiration
- Message deferral
- Message batching

Let's explore each of these features in detail.

Message Filtering

Azure Service Bus supports message filtering with Topics. It is supported with queues. You can configure subscription rules for Azure Service Bus Topics to deliver messages categorically to different subscribers as per the business requirements. A subscription rule can consist of filter condition or actions. To learn more about the message filtering feature of Azure Service Bus, take a look at https://docs.microsoft.com/en-us/azure/service-bus-messaging/topic-filters.

Message Sessions

Azure Service Bus provides a guaranteed reliable delivery of related messages in a particular sequence to the receiver/subscriber application by using message sessions. To leverage this feature, you will have to enable sessions while creating queues/topics. While sending a message to the queue/topic as part of a session, the sender application will also have to set the session ID property. This will help the receiver application to distinguish if a particular message is associated with a particular session or not. To learn more about message sessions of Azure Service Bus, check out https://docs.microsoft.com/en-us/azure/service-bus-messaging/message-sessions.

Message Auto-forwarding

Azure Service Bus provides a way to send messages from a particular queue or subscription to another queue or topic within the namespace for the purpose of chaining them. Any messages coming to our source queue/subscription will be removed and moved to the destination queue/topic automatically.

Duplicate Detection

In certain scenarios, a message sent by the sender application may not get delivered within a certain time period, because of which the sender application might resend the message to the service bus queue to be picked up by the receiver application for further processing. Now our receiver application will be processing the same message twice because of the message being resent by the sender. To remove such duplicate messages, Azure Service Bus provides a duplicate detection feature that discards the duplicate message for us.