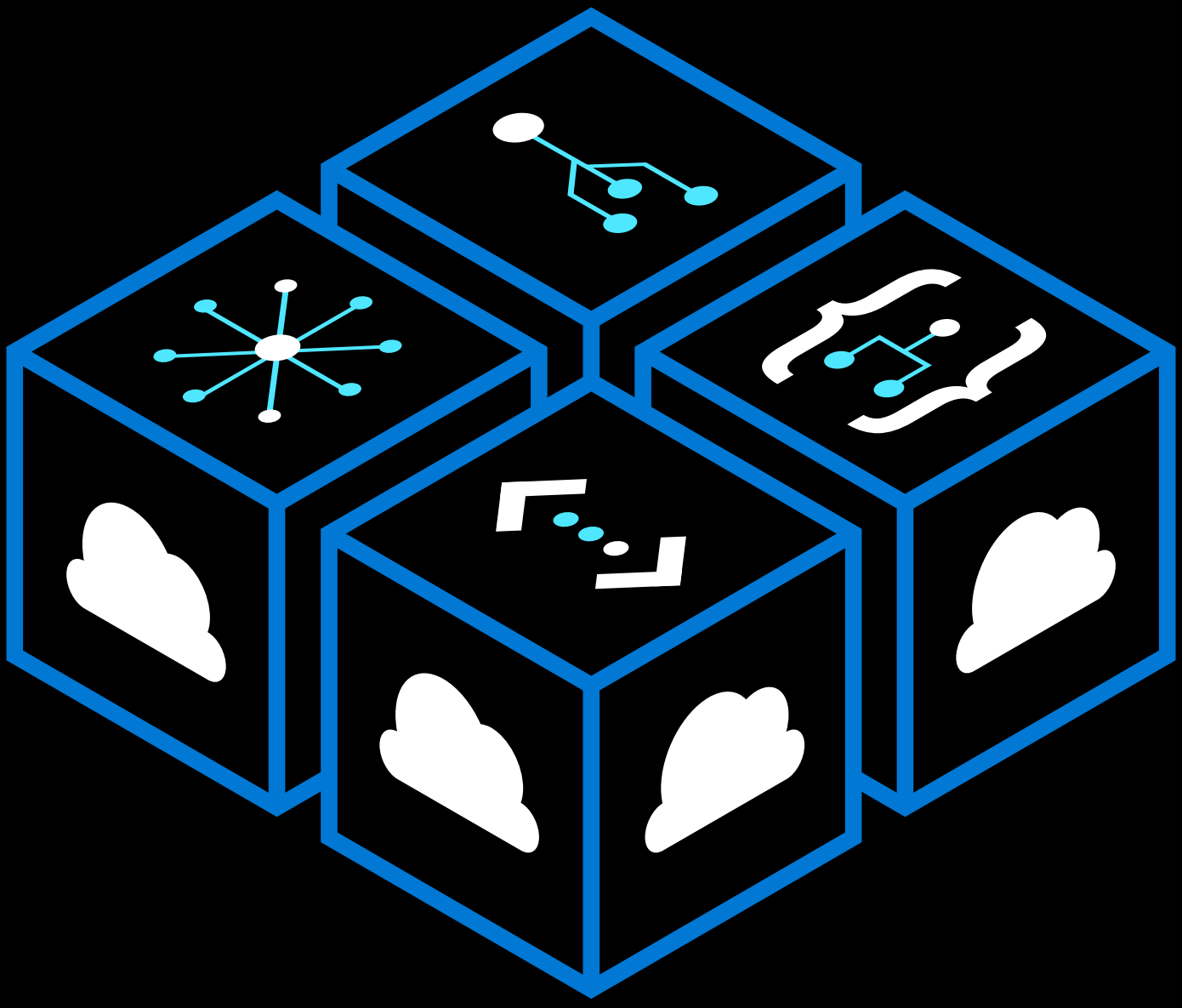
E-Book series

Microsoft Azure

Technical Guide to Building SaaS Apps on Azure



Technical Guide to Building SaaS Apps on Azure

3 /
Introduction

5 /
SaaS as a business model

7 /
Identity is everything

11 /
Build SaaS

16 /
Azure: Delivering building blocks as a service

22 /
Microsoft commercial marketplace: Sell your SaaS

27 /
Summary

28 /
Get started today

29 /
Resources

Introduction

Software-as-a-Service (SaaS) is increasingly becoming the driving force behind most organisations' business or operations.

This guide will assist you in making better decisions, understanding how SaaS technology is having a powerful impact on how organisations work, and implementing concepts such as the API economy, customer enrichment and onboarding.

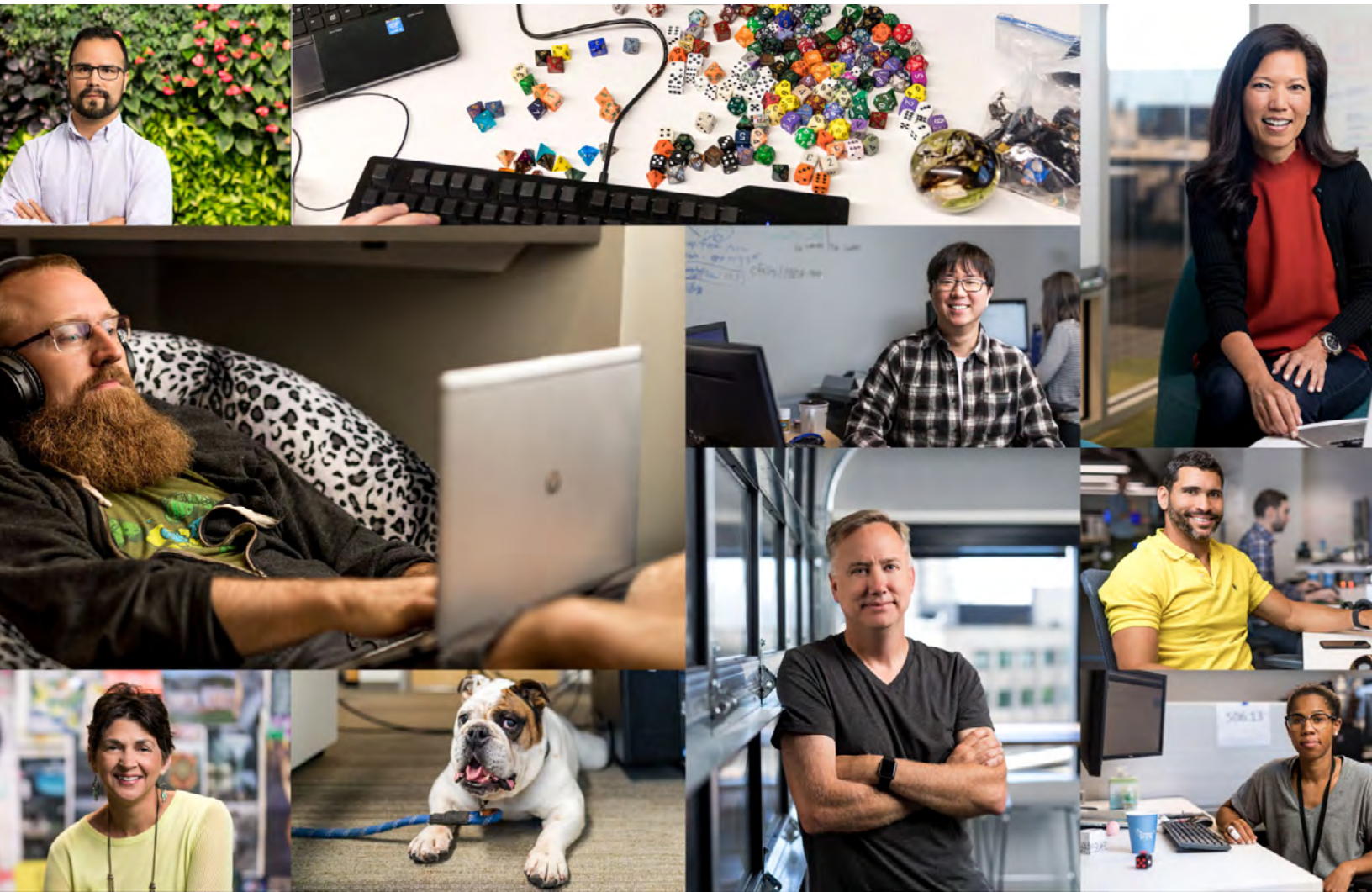
You'll discover that SaaS is more than just a software; it's a platform that promotes innovation and productivity throughout the business.

Microsoft Azure offers capabilities and tools to bring your SaaS application to customers around the world. The building blocks of Microsoft Azure give you the ability to integrate, deploy, monitor and expand easily. Together with the technical documentation, expert support, knowledge sharing and frameworks provided, Azure is one of the best platforms to build your SaaS applications on.

The primary audience for this guide includes architects, technical decision makers, technically oriented business decision makers and of course those who are building SaaS specifically as a business, namely, **Independent Software Vendors** (**ISVs**). These resources and insights will help ISVs to grow and optimise business models.

This eBook will help you with the following aspects of SaaS building through:

- Discussing SaaS as a business model and the impact of technology choices on your architecture.
- Clarifying the importance of identity and the principle of Zero Trust.
- Setting the scene for building cloud-native software and benefits of a cloud-native set-up.
- Getting around the Azure services you have available to help you build true SaaS platforms.
- Enabling you to sell your platform on the Microsoft commercial marketplace.



SaaS is a business model that appeals to many end users and consumers, as it's an easy way to get started quickly and with a controllable cost.

SaaS as a business model

Digital transformation is a given and noticeable everywhere, in every vertical, in every organisation, in every industry. Today, organisations can no longer focus solely on commodities and operations, but must also consider innovation and improving capabilities.

SaaS is one of the largest movements you see in the rise of new ways of offering software to users, whether it is by means of using low-code/no-code alternatives (citizen dev is a hot topic) or by simply moving into a subscriber model, such as GitHub or certain media streaming companies.

As a bonus, typically these types of subscriptions provide easy onboarding, which drives better adoption of your model and will not go unnoticed by the world of consumers.

Now, of course, this doesn't only have to apply to external users: making software usable in your own enterprise will, in the same way, make it more adopted within your own ranks. Driving this through features like Single Sign-On (SSO) or easy integrations with other tools, such as Power Apps/Power Platform, or in a collaboration mode from within Microsoft Teams, for example, will make it more liked by users as they will find it easy to get started.

Adopting such a delivery method will have some impact on the way you operate and charge for your software. Don't forget that with SaaS, you are running the infrastructure, applications and endpoints. This will have an impact on the cost of the choices you make in the technology stack.

As an ISV, you want to address all markets, global or local. Driving usage of your platform becomes your main focus next to the fact that you are a software company. Adoption of your platform is the most beautiful thing you can experience, as it will prove that you are providing a solution to a problem that many are facing.

This eBook will guide you in those technology choices and try to make it easier to understand the impact and the benefits. The next phase is to begin creating your platform after you've established your business strategy and goals for what you want to achieve as an organisation or with your platform.

Identity is everything

So where do you start and what does it take to get started? Well, first you need to make sure your platform can be widely adopted securely and safely. Next to that, you want to make sure that you don't frustrate users with onboarding. Combine these two elements and the solution is simple: in essence, identity becomes a key feature.

Identity is a foundational part of the Zero Trust principles. A basic Zero Trust process would look like this:

1. Authentication: Identify who a user is.
2. Authorisation: Determine what the user can do.
3. Using the principle of least privilege, act on the application securely and in a controlled way.

If you want to know more about Zero Trust, check out the Zero Trust model – modern security architecture.

But fair warning: you don't want to manage identity yourself. The user or the user's organisation is the owner of the identity. You want to avoid the ownership of identity at all costs. For this, you need to integrate with users' own identities. Typically, these will either be organisational identities or federated identities with some Identity Providers (IdPs) such as Microsoft accounts, Facebook and LinkedIn. And it doesn't stop there as you typically will have a multitenant-supported environment.

Azure Active Directory (Azure AD) can help you in both situations:

- Azure AD for SSO and support for multitenancy with work or school accounts. OR,
- Azure AD B2C for implementing business-to-consumer models with multiple choices of IdP where the user can choose their own typical identity to onboard. An example of a B2C implementation with a Microsoft account is shown in *Figure 1*. The left pane shows an example application with a login button. The right pane depicts the screen that the user will see when consent for the necessary permissions is requested.

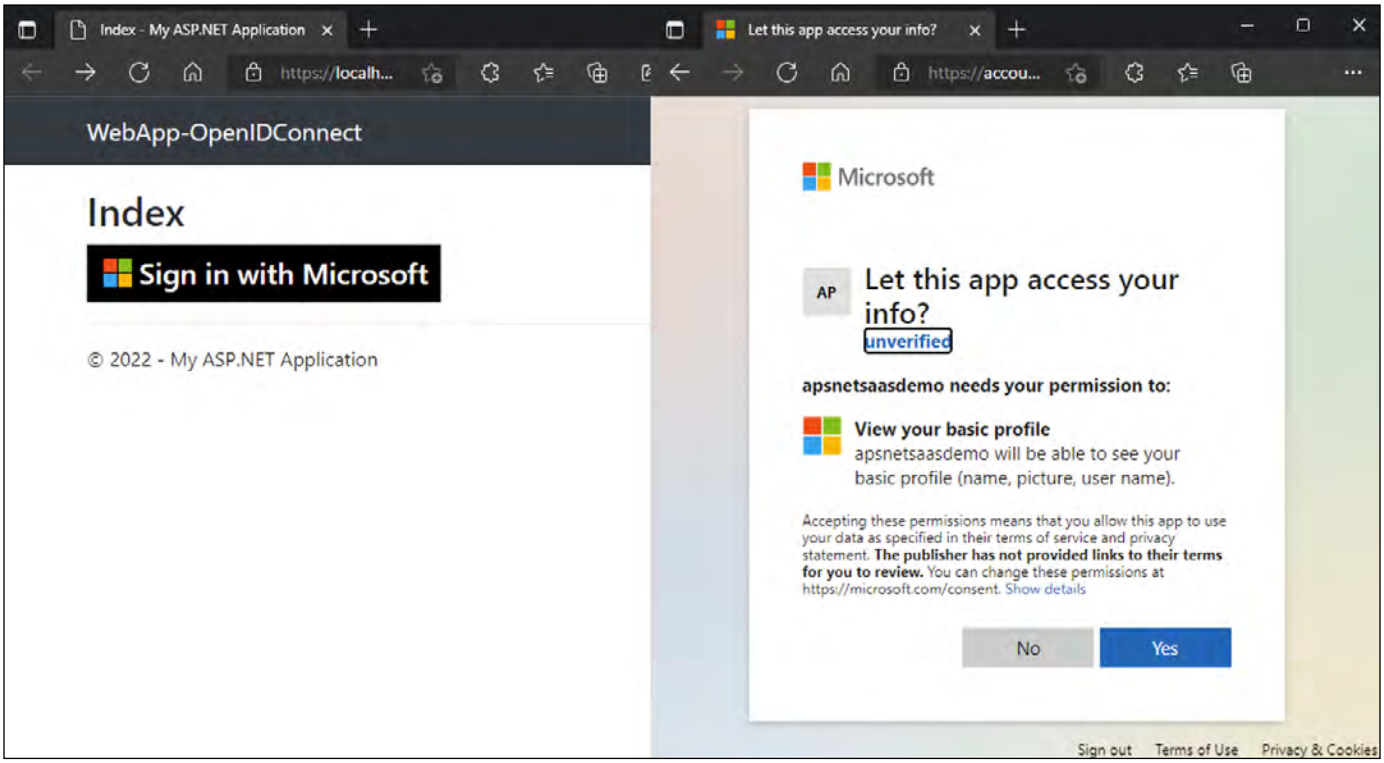


Figure 1: An example of Azure AD B2C sign-in implementation

Azure AD or Azure AD B2C brings you the following advantages:

- As a SaaS provider/organisation, you don't own an identity and thus don't need to own and store credentials. The main advantage here is that the possibility and severity of an attack will be reduced. There are no databases with users and passwords involved and thus you'll be less vulnerable to security issues related to identity theft.
- It's based on the core of Zero Trust principles:
  - There's always explicit authentication and authorisation based on the user identity and, as a bonus, these systems can provide additional benefits like Multifactor Authentication (MFA) out of the box.
  - It can be based on the principle of least privilege.
  - Breach assumption becomes easier as the auditing of users can be followed by the owning organisations.
- Bonus: If you implement Azure AD, you get the principles of SSO and enterprise discoverability out of the box by means of Azure AD app gallery, enabling your end users to get these applications immediately on their landing page.

This is a code-level implementation, although it can also be easily achieved by using services such as Azure App Service, as you can do a configuration on that level to enable easy authorisation on the application, making it smoother to get started.

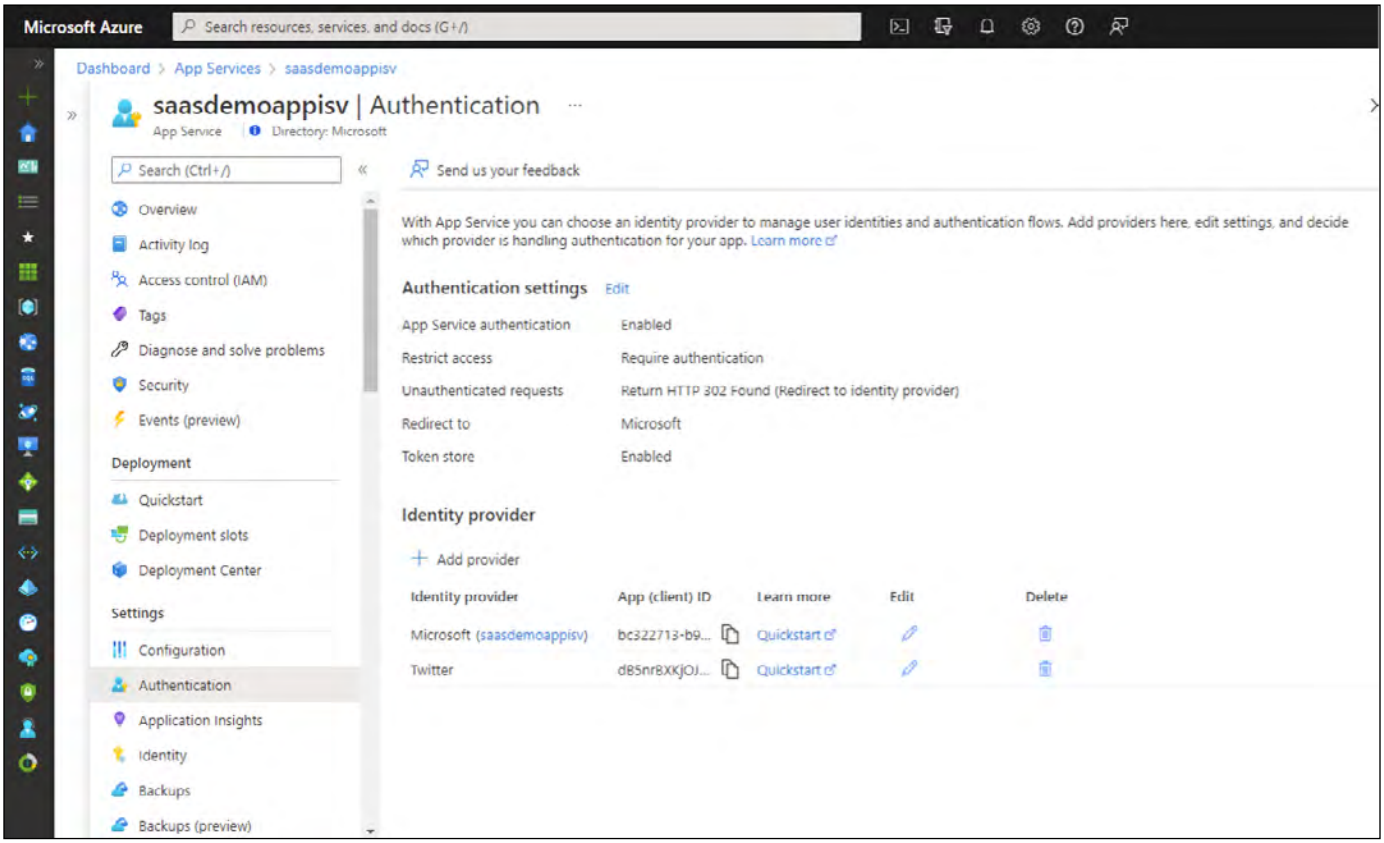


Figure 2: Example of easy authentication enablement on an Azure App Service (no code)

Typically, when you are building SaaS as an ISV, your systems will need to be customer multitenant, thus having the ability to have multiple customers running on a single platform. The systems described in *Figure 2*, like Azure AD, can help you with multitenant identities and enable easier onboarding.

An example of a multitenancy system is Office 365. There is one single platform, but many customers use it, and they can only see and administer their own data.

A tenant in this context is a unique customer. Multiple tenants or multitenancy means having more than one customer running on your systems. Designing for multitenancy requires thinking about how to segregate and segment data, making sure that customers only see the data they are allowed to see. An example of a multitenancy system is Office 365. There is one single platform, but many customers use it, and they can browse information from the tenant that are in their own profile information.

An excellent resource for architecting multitenant solutions can be found by following the video, Architecting multitenant solutions on Azure.

Of course, the user's identity is not the only thing you'll require. You need to make sure to use the correct building blocks and the advantages of the cloud to make your platform one of the finest ones available for customers (because you want everyone to use your platform, right?).

Build SaaS

When building your application or SaaS platform (more on why we call it a platform later), you need to make sure that it also acts like a cloud application. If you are aiming for a globally available application, you need to make sure that you follow the rules of thumb for cloud-enabled applications: they need to be resilient, elastic and highly available.

Today, you can build on technology stacks that have proven to have these capabilities. Sure, you can host or build your own infrastructure hosted in a private environment, like, let's say, your own data centre, but this will increase the complexity and limit the scaling. Running your SaaS environment on Microsoft Azure, for instance, helps you use such capabilities as they are built into the platform. But hosting is only one single element of this cloud-capable and cloud-native building. Your application also needs to be savvy around this.

When building in the 'old world' of on-premises, we only had the concept of n-tier applications with the common paradigm of monolithic building. Monoliths were the answer to quicker deployment and development, keeping everything in the same dev teams because these people were also typically seen as domain experts and thus also the product team. For their ease of working, they typically also made projects fully monolith-based, as in, all projects in one solution to make sure nothing broke during the build. When applications are designed for the cloud, monoliths can be difficult to scale. The entire application should be scaled up or out. This could be inefficient.

If you only need to scale the front end of an application, it's a waste of money to scale the entire application. In this context, microservices are one of the architectural patterns that could solve this problem.

These microservice architectures typically support scaling and high availability from within the application itself. When looking at these application architecture styles, cloud-native technologies also come into play, and this is what this section is about. You'll require an additional mechanisms and technologies in place that are designed and invented specifically for building cloud-based applications/solutions.

Think about technologies like Distributed Application Runtime (Dapr). It's an open-source project that aims to make it easier for developers to create microservices and integrate them into their applications. It helps in making communication easy between your services without having to worry about the communication part, but also in enabling you to have easier discoverability and observability for your platform.

Figure 3 shows a sample microservice architecture. Each hexagonal image depicts a service, with a Dapr sidecar attached. The developers creating such a service can talk to the Dapr sidecar without having to worry about the actual outside implementation of, for example, pub/sub mechanisms or input bindings. It simplifies development, extracting the heavy lifting to the sidecar. It also eliminates home-grown solutions for these components. For example, depending on Dapr for retrieving secrets also mitigates any leakage of secrets and makes the code inside the service less prone to security bugs:

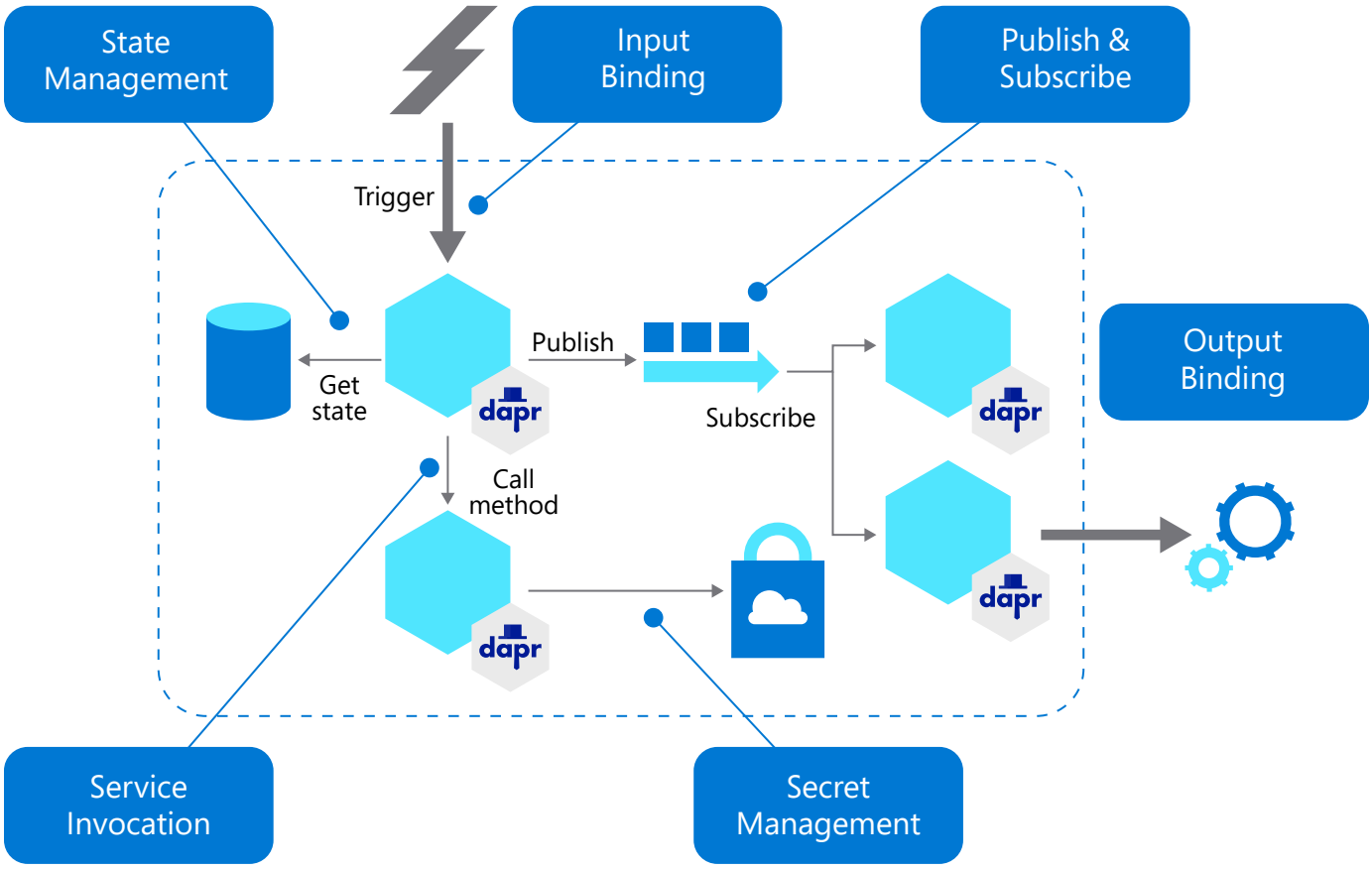


Figure 3: Key features of Dapr and the benefits it can bring in developing cloud-native applications

The features of Dapr are listed below:

- **State management:** Stores and queries data as key/value pairs, setting the choices on concurrency control and data consistency. Performs bulk update operations, CRUD, including multiple transactional operations
- **Input and output binding:** Removes the binding complexity and keeps the code SDK-/library-free, so no need to break compatibility. Low maintenance connections with a business logic focus instead of connection capability focus
- **Auto pub/sub capabilities:** Integrated capability for message queues and topics, making loosely coupling possible
- **Secret management:** Makes the setting up of secret stores and manageability and connectivity to those easily implementable
- **Service invocation:** Delivers autodiscovery of services, encrypted interservice communication and retry logic/transient failure mechanism

Pub/Sub Messaging

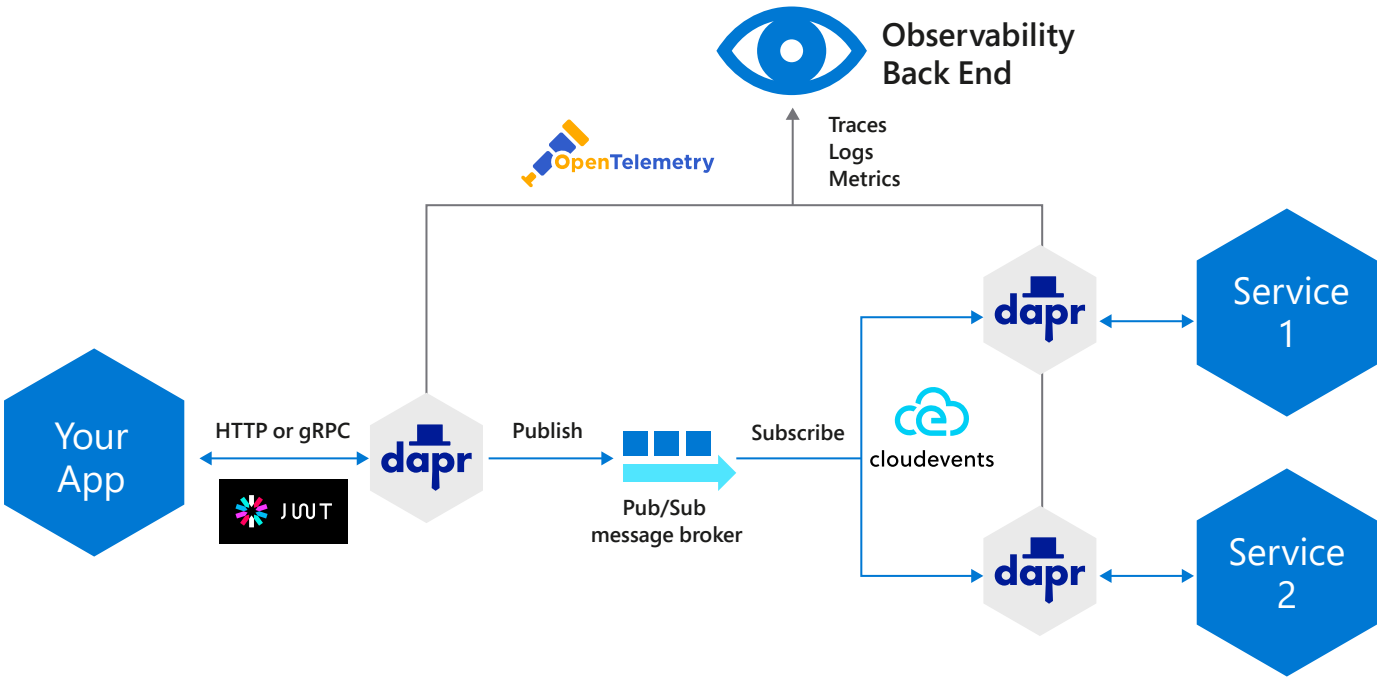


Figure 4: Example of a Dapr solution, and how Dapr can be used for sidecar implementations, observability and more

An interesting look on how to use multitenancy in AKS can be found here: Use Application Gateway Ingress Controller (AGIC) with a multi-tenant Azure Kubernetes Service.

Think about elements such as the event-driven architecture style **(eventing)**, with the usage of events or messaging, and, on top of that, making your applications scale inside of your infrastructure based upon the number of events that have been initiated or passed through. This is something that can be easily achieved with the usage of Kubernetes (a technology great for running microservices), preferably Azure Kubernetes Service, and something called Kubernetes Event-driven Autoscaling (KEDA).

KEDA allows you to easily use the scaling capabilities of Kubernetes to scale in directions you want based upon metrics available in your messaging layers, for instance. Look at some reference architectures with KEDA in the Azure Architecture Centre: Azure Kubernetes in event stream processing.

Most of these technologies are being built on top of open standards, allowing you to run this in your favourite cloud, on-premises, hybrid and so much more. The main goal is that it helps you achieve cloud-native building, thus preparing your platform for the future.

There are more technologies that can help you here, for instance, the usage of service mesh technologies to better create focus on your services instead of pure endpoints.

It helps you set the networking capabilities more easily. One of those available is Open Service Mesh, another initiative from the Cloud Native Computing Foundation (CNCF).

If you want to automate things easily, sometimes you also want to build on standards, looking at CloudEvents (an open standard for events), for example. CloudEvents is a specification for describing event data in a common way for all cloud platforms. This allows you to then automate based upon the events. One of the main services in Azure you can use is Azure Event Grid, which allows you to create triggers and actions-based automation, enabling you to react to almost anything inside your SaaS platform.

In addition to cloud-native features, Platform services that can help you construct scalable solution will be required. And, Microsoft Azure is the best cloud service provider to construct your solution.

Azure: Delivering building blocks as a service

All the factors we discussed in the earlier sections will aid your application's lifecycle and overall health. However, a scalable infrastructure that can handle these technologies and help you build a better SaaS platform is essential. As mentioned earlier: consider it more a platform than an application! But why is that?

Today there's an additional entry point for your environment: the API. It's not only on traditional web applications any more! APIs are at the heart of our capabilities in the new digitally altered world of the API economy. And this is how we drive business models forward. The consumption and subscriptions of these APIs make the business grow. Because of this, it is a platform and not just a 'simple application'. Everything to build a solid solution and solution platform will be highlighted in the next sections.

Azure Integration Services

The glue between your components is key for success. Azure provides a rich set of services for exactly this purpose:

- **Logic Apps:** Create workflows and orchestrate business processes to connect hundreds of services in the cloud and on-premises.
- **Service Bus:** Connect on-premises and cloud-based applications and services to implement highly secure messaging workflows.
- **API Management:** Publish your APIs securely for internal and external developers to use when connecting to back-end systems hosted anywhere.
- **Event Grid:** Connect supported Azure and third-party services using a fully managed event-routing service with a publish-subscribe model that simplifies event-based app development.
- **Azure Functions:** Simplify complex orchestration problems with an event-driven serverless compute platform.
- **Azure Data Factory:** Visually integrate data sources to construct **Extract, transform and load** (**ETL**) and **Extract, load and transform** (**ELT**) processes and accelerate data transformation, using 90+ prebuilt connectors to manage data pipelines and support enterprise workflows.

These components make it easy to integrate with your customers' applications and provide entry points and extensibility.

Power Platform – Consume APIs

APIs enable not only integrations and consumption metrics, but also easy extensibility with low-code/no-code environments like Power Automate or Power Apps. As a result, your end users will be able to create their own interfaces on top of your platform. Creating custom connectors or even registered connectors will help citizen developers and fusion developers create custom applications on top of any of your solution endpoints. That way, you can help them to achieve more! Power BI can even make use of these to provide reporting and dashboarding capabilities.

Azure App Service and Azure Static Websites – API and web hosting

Next, you want your own front-end application to scale, preferably with easy integration with DevOps with GitHub to make sure your developer teams can apply best practices and better availability during application deployments or upgrades. Think about services such as Azure Static Web Apps, which enable these features out of the box and get you up and running in a DevOps flow in no time. Other web development with more servers, depending on technologies like .NET or Java, can also benefit from Azure App Service.

Azure Front Door – API shielding, defence and routing

Defending your endpoints and making your application, available globally is unavoidable when you are running a SaaS platform. For that you need some strong defences and a manageable platform to help you. Azure Front Door delivers you that platform. It allows you to:

- Accelerate and optimise latency by means of the integrated CDN capabilities and Layer 7 routing features.
- Defend and protect your endpoints by using its built-in web application firewall, managed through policies from a single pane of glass, making it easier to control and monitor.
- Help you against DDoS attacks and provide bot protection, to deliver higher availability and resilience of your SaaS platform.
- As it's a consumable service, scale so you don't need to deploy additional devices to be resilient and highly available.
- Fully enable configuration through Infrastructure as Code (IaC), as it can be fully integrated into your developer stream or workflow.

Azure Data Platform: Azure Database, Cosmos DB, Azure Storage – API Persistent Storage Back End

Now, all these layers are nothing without a layer of persistence. Azure can provide both traditional ways, like Relational Database Management System (RDMS) approaches, and schemaless/no-SQL solutions. Choosing between them can be cumbersome, but if you require a global approach that can also provide multiple levels of consistency, then Azure Cosmos DB is the way forward. It gives you a high Service-level agreements (SLAs) and multimaster replication, enabling your application to be better globally available and enabling your end users to approach their data wherever they are. New features, such as near-real-time analytics with the help of other services like Azure Synapse Link for Azure Cosmos DB, enable analytics in platforms such as Azure Synapse without ETL and thus without additional data movement or duplication and even without degrading the performance of your data estate.

It doesn't have to stop there; Azure provides more than just those. Think about highly scalable and performant storage capabilities like Azure Blob Storage or Azure Data Lake, both providing you with scalable data storage up to petabytes and beyond of data.

Patterns and practices – Meet Azure Architecture Centre

All these services make it easier for you to build scalable and resilient platforms, but choosing the right ones for your needs can sound cumbersome. For this, Microsoft has a wealth of information written down in the Microsoft Azure Architecture Centre. One of the most useful features here is the decision trees to help you select the best of breed for your platform based upon requirements you've written down (please refer to *Figure 5*). These guides or trees make it easier to choose from the technologies needed.

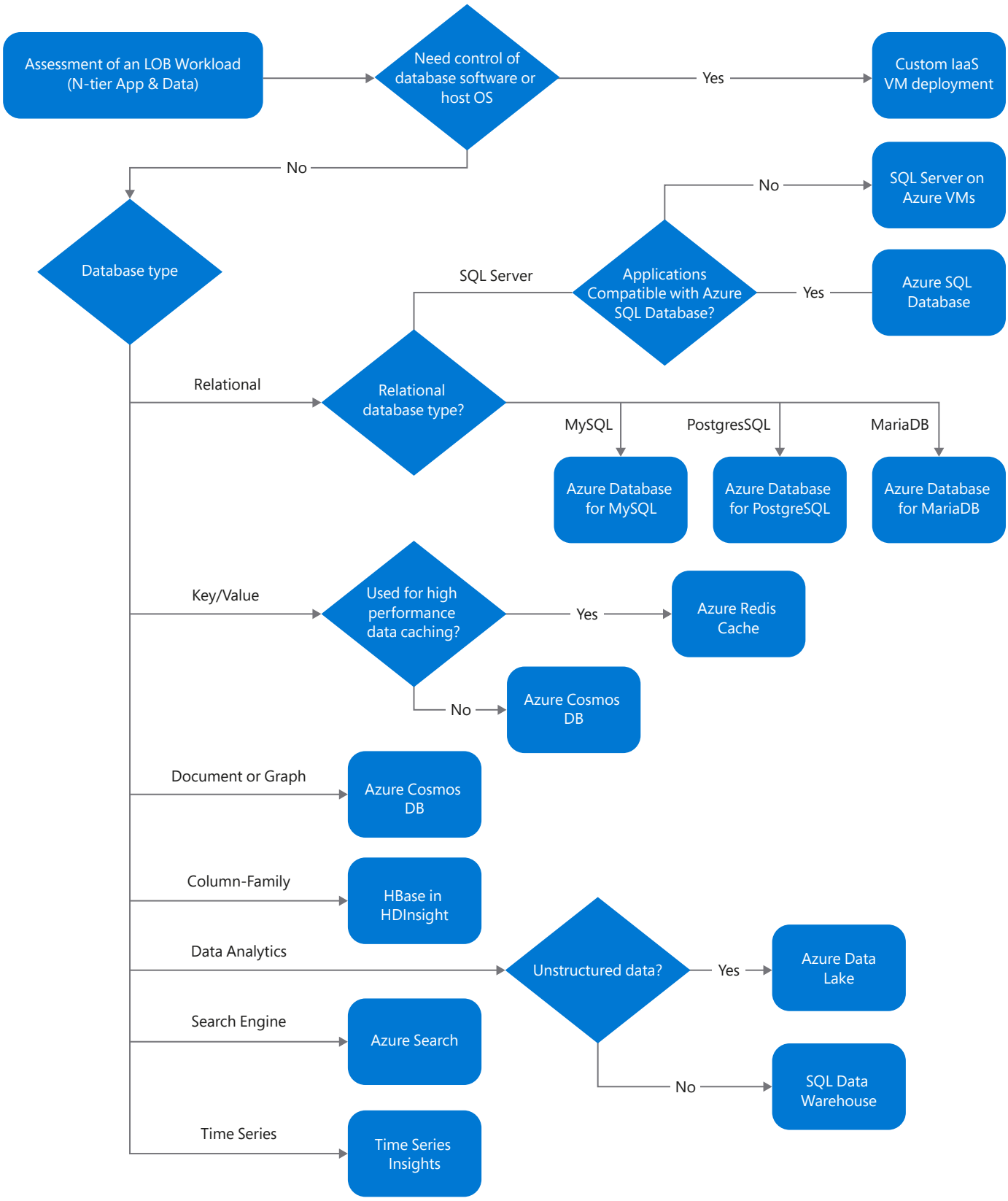


Figure 5: One of the Azure Architecture Centre decision trees, which can help you choose the best-of-breed Azure services

Now, choosing right technologies isn't the only difficult element of designing SaaS platforms. It's also about deciding what business model to focus on.

These helpers already exist today for compute, data stores, analytics, load balancing, IoT and so on.

It was mentioned that technology choices and architecture choices would be impacted by your business model decisions and vice versa. One of the topics that has a significant influence, specifically for SaaS, is the impact of multitenant architecture.

This impacts your compliance, security, cost and operations depending on the choices you make. Typically, you can see three types of architectures for this business model, ranging between fully isolated, shared compute/ separated storage and fully shared.

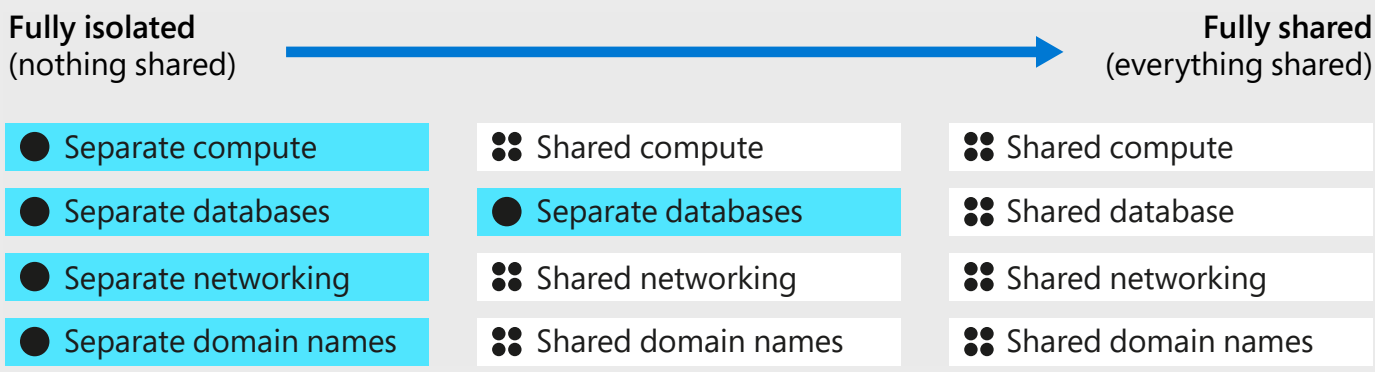


Figure 6: An isolation-level design of multitenant architecture

As shown in *Figure 6*, all these choices can have an impact on cost, operations, governance and compliance. Microsoft has a good guide to help you architect your multitenant solution (just as with the service decision trees) and the choices needed for your specific use case. This guide can be found here: [Architecting multitenant solutions on Azure](#).

Those are all the things needed to help you build better SaaS platforms. But why stop there? As you want to make sure you get traction on your platform, you want to make sure you can (re)sell it in a scalable way. Read on to the next section, on Azure Marketplace, where you'll find tips and tricks to help you reach (enterprise) customers using the features Microsoft provides.

Microsoft commercial marketplace: Sell your SaaS

Once your SaaS platforms is ready, you need to drive usage! Microsoft can help you there, too. ISVs can onboard themselves in the Partner community, by registering themselves as members of the Microsoft Partner Network (MPN). This brings advantages! One of these is the ability to sell your SaaS through Microsoft's Azure Marketplace, using any of the existing capabilities, such as a Teams application, Microsoft 365 plugin, Azure Marketplace or Microsoft AppSource (next to the earlier-mentioned Azure AD Gallery app capability, of course).

If you want to achieve this selling through Azure Marketplace, and thus enable your solution to be bought by customers on Azure, you don't have to jump through hoops or bend in impossible positions.

Once onboarded in the MPN, you'll get access to the Microsoft Partner Centre. This enables you to create offers on Microsoft's Azure Marketplace, (Please refer *Figure 7*), with the possibility to have multiple offers in different architectures. This can range from a simple 'Contact Me' form that allows you to immediately follow up on those leads to a more consumption-based offer via Azure Marketplace. Moreover, MPN provides transparent invoicing capabilities, which ultimately make transactions easy for end users. All these subscriptions you provide will be processed on the end user's Microsoft cloud invoice, thus making it easier to purchase and use the services you are providing.

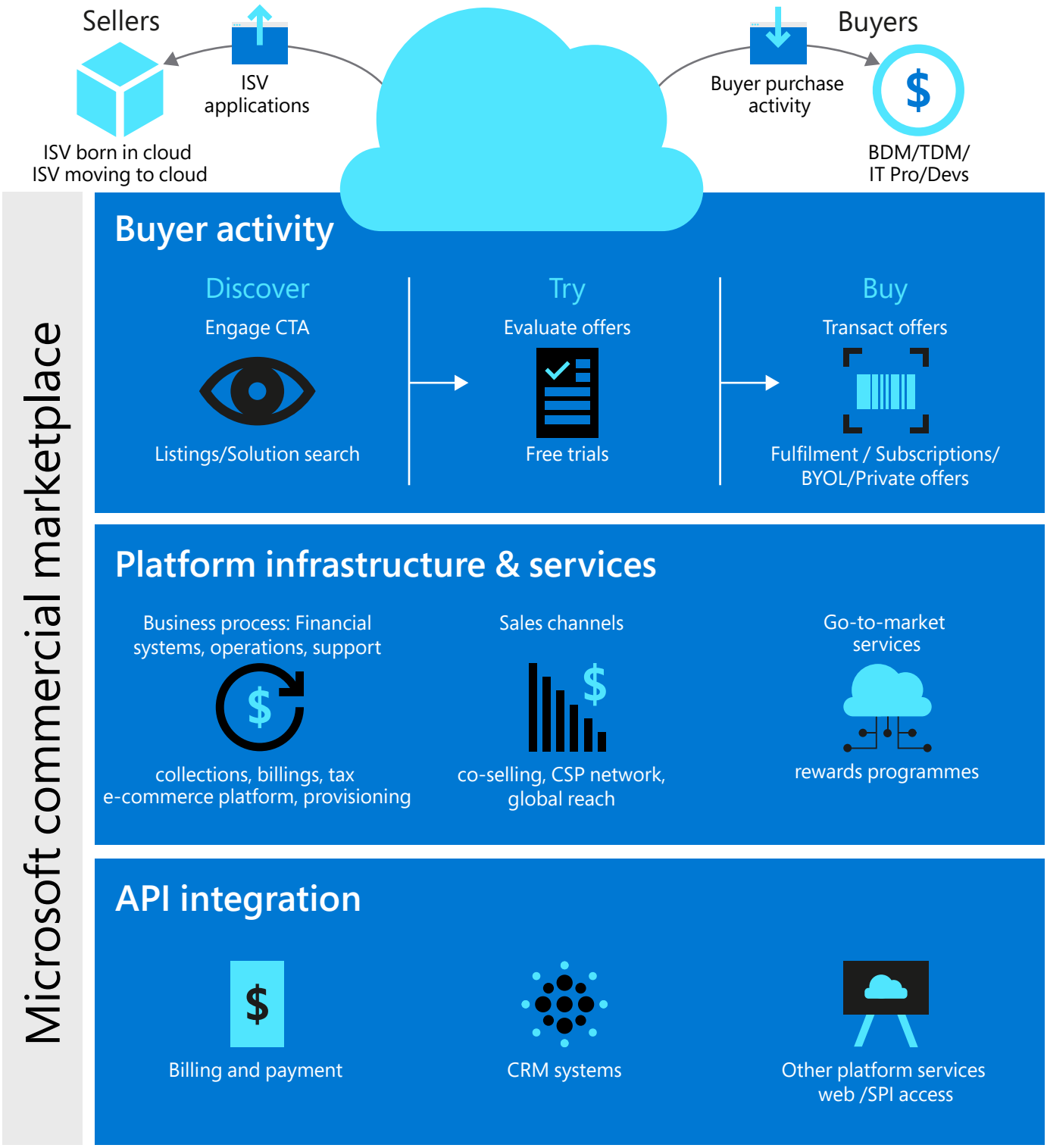


Figure 7: Microsoft commercial marketplace

To make your offer transactable, Microsoft provides the following help and guidance. There is a GitHub repository with examples, code samples and a client library for you to get acquainted with the tools; the latter wraps around the most important parts, namely, Marketplace fulfilment API and Marketplace metering service API.

The workings of these API services are simple and consist of only a few components:

- Start off by creating a SaaS landing page to do the onboarding with the end user's Azure AD account.
- Integrate the SaaS Fulfilment API for onboarding.
- Add Marketplace metered billing APIs for specific billing capabilities.

Figure 8 depicts the buying of a SaaS application through Azure Marketplace or Azure Portal:

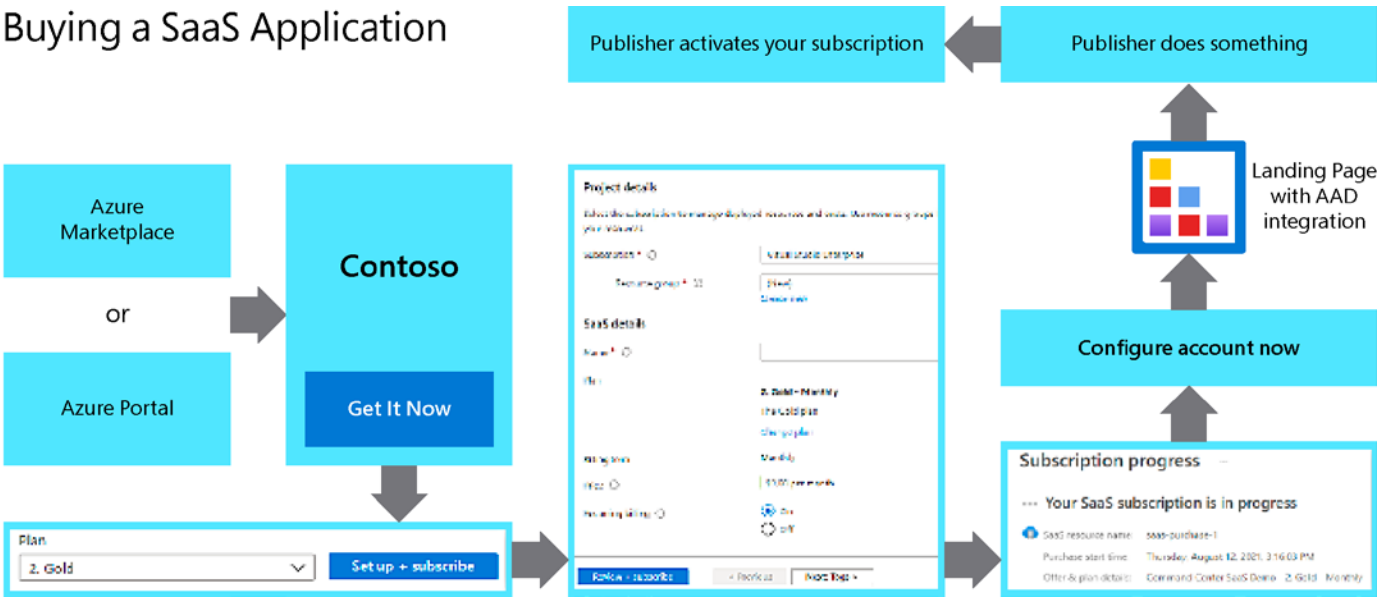


Figure 8: Buying a SaaS application

Now let's look at how you can handle billing and metering efficiently with the help of Azure Marketplace's metering service!

Billing and metering

The SaaS Fulfilment API will manage all the subscription logic, while the Metering API will allow you to charge the end user with the correct usage if you are charging your service on a metric such as hourly usage. This provides the flexibility to create different subscription types, for example, charging based on volume per month (1,000 emails sent per month, for instance).

The following is an example of such a plan:

- **Base plan**
  - Analyse 100 GB and generate 100 reports for USD 0/month.
  - Beyond the 100 GB, pay USD 10 for every 1 GB.
  - Beyond the 100 reports, pay USD 1 for every report.
- **Premium plan**
  - Analyse 1,000 GB and generate 1,000 reports for USD 350/month.
  - Beyond the 1,000 GB, pay USD 100 for every 1 TB.
  - Beyond the 1,000 reports, pay USD 0.5 for every report.

Having a metered billing plan will allow you to upsell your product and let your customers take advantage of volume discounts. All the advertising is done through the Marketplace fronts and can even be endorsed by reseller partners like a Cloud Solution Provider (CSP).

This mechanism provides easy onboarding and an easy way to buy your product. It can be invoiced through the end customer's Azure billing, which helps to avoid long purchase offer rounds and requests proposals as the purchase can be consumed under the same already budgeted Azure consumption agreements.

Now, you have fully SaaS-ified your business and you are ready to take on the world.

Microsoft SaaS Academy

Microsoft brings together customer stories, best practices and guidance in the Microsoft SaaS Academy. It's a great reference for business tracks, technical tracks and customers, SaaS stories.

Check out the interview with Visma, a very large European player in the field of ERP/HRM. Next to that, listen in on the Salutare SaaS Journey, a healthcare software start-up, with their fully born-in-the-cloud SaaS software. And finally, ToolsGroup helping with supply chain management in many domains, including automotive, pharma and retail.

It's very inspiring to see both software that was born-in-the-cloud and companies that have been around for many years, transitioning towards a SaaS model by leveraging the capabilities of the cloud.

Summary

As you can see, it's not only about all the technology choices, but also deciding what kind of SaaS you are building and what kind of business model you are using to build in terms of how certain choices have an impact on both tech and business. Throughout this journey, we:

- Discussed how to make our application more secure by following the Zero Trust principle and aiming for an identity-driven system.
- Understood how certain new standards, such as implementing cloud-native technologies can help you make use of and maximise your SaaS capabilities and elasticity.
- Learned about Azure benefits, specifically ones related to building better solutions, so you can focus more on your software instead of handling the underlying infrastructure.
- Discovered architectural guides in the Azure Architecture Centre, in a multitenant way.
- Finally, mentioned how selling and distribution can be handled easily, by using the Microsoft Marketplace capabilities as a partner, allowing you the opportunity to display, sell, distribute and transact your SaaS subscriptions.

A SaaS platform needs to be bound by the SLAs and availability that you want to provide to your end users. The Azure platform can help you do that with many of its services. It doesn't stop there as you can enrich your platform by extending it with API endpoints so you or your end users can extend the experience even further by adding no-code/low-code extension with the Power Platform (or even integrations through that).

Microsoft is more than a software company; it's a SaaS enabler on any level! We hope you continue to consult this guide to become better acquainted with building SaaS apps on Azure and determine how it best fits your business requirements.

Get started today

To start building your SaaS platform today, first let's look at the Azure Architecture Centre, with at its core is a Well-Architected Framework. The Azure Well-Architected Framework consists of five pillars, which we've been discussing all along:

- Reliability
- Security
- Operational excellence
- Performance efficiency
- Cost optimisation

Keeping in mind these five pillars and trying to build your architecture around that will get you started on the right foot for resilience, high availability and elasticity. Read all about Well-Architected Framework here: Microsoft Azure Well-Architected Framework. Here is the link to a video on Well-Architected Framework for SaaS: Microsoft SaaS Academy – Well Architected Framework (linkedin.com).

There is a tremendous learning zone available at the Microsoft SaaS Academy, with many good videos on how to drive SaaS to the fullest, not only from a technical, but also from a business perspective. This information can be found on the Microsoft SaaS Academy landing page.

Resources

The following are quick reference links for what we discussed earlier:

- Azure Architecture Centre
- Microsoft SaaS Academy
- Azure reference architectures for SaaS
- Microservice architecture style – Azure application architecture guide
- Architecting multitenant solutions on Azure
- SaaS digital business journey on Azure
- Getting started with Identity
- Azure API Management
- Azure Service Bus
- Azure Event Hubs
- Azure Cosmos DB
- Azure Kubernetes Service
- Azure Front Door
- Azure Functions
- Azure Event Grid
- Microsoft commercial marketplace
- Dapr
- KEDA
- Open Service Mesh
- Accelerate and De-Risk Your Journey to SaaS | OD140
- ISV considerations for Azure landing zones
- Azure SaaS Guide
- Azure webinar series: Build Your SaaS Application on Azure

© 2022 Microsoft Corporation. All rights reserved.
This document is provided 'as is'. Information and views expressed in this document, including URL and other internet website references, may change without notice. You bear the risk of using it. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.