Creating a simple cloud application

zalnet

About me

Security Architect

Consultant

Microsoft Certified Trainer

AI & Cybersecurity Practitioner

Developer

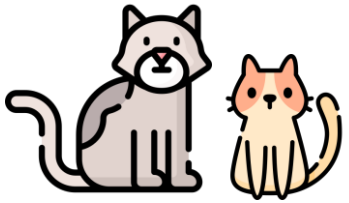
Freelancer

Azure @ ❤

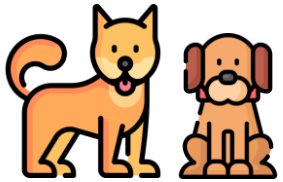
Google Cloud

1 Husband

1 Daughter



2 Cats



2 Dogs

Crime stories

Photography

zalnet https://www.linkedin.com/in/beatazalewa/ info@zalnet.pl BLOG https://zalnet.pl/

Agenda

- Explore Azure App Service
- Configure web app settings
- Scale apps in Azure App Service
- Explore Azure App Service deployment slots
- Explore Azure Functions
- Develop Azure Functions
- Explore Azure Cosmos DB
- Work with Azure Cosmos DB
- Manage Container Images in Azure Container Registry
- Run container images in Azure Container Instances
- Implement Azure Container Apps

Explore Azure App Service

Examine Azure App Service

Azure App Service is an HTTP-based service for hosting web applications, REST APIs, and mobile back ends. You can develop in your favorite programming language or framework. Applications run and scale with ease on both Windows and Linux-based environments.

Built-in scale support

- Save costs and meet demand through scaling
- Scale out/in manually or automated based on metrics
- Scale up/down by changing the app service plan

Continuous integration/ deployment support

- Azure DevOps/GitHub/Local Git repository/Bitbucket
- FTP
- Container registry
- And others

Deployment slots

- Target deployments to test or production environments
- Customize what settings are swapped between slots

Examine Azure App Service plans

App Service plans

- Define a set of compute resources
- Can run one or more apps in the same plan
- Can be scaled up or down at any time to meet compute or feature needs

Usage tiers

- **Shared**: shared compute resources targeting dev/test
- **Basic**: dedicated compute targeting dev/test
- **Standard**: run production workloads
- **Premium**: Enhanced performance and scale
- **Isolated**: high-performance, security and isolation

How does my app run and scale?

- **Shared**: apps receive CPU minutes on a shared VM instance and can't scale out
- Other tiers: apps run on all the VM instances configured in the App Service plan

Deploy to App Service

Every development team has unique requirements for their deployment pipeline. App Service supports both automated and manual deployment.

Automated deployment

- Azure DevOps
- GitHub
- Bitbucket

Manual deployment

- Git
- CLI
- Zipdeploy
- FTP/S

Sidecar containers

- Add up to nine sidecar containers for each side-car enabled container app.
- Add through the **Deployment Center** in the app's management page.

Deployment slots

- Target deployments to test or production environments
- Customize what settings are swapped between slots

Explore authentication and authorization in App Service

- Built-in authentication and authorization support
- Implement with low to no code changes in your web app

Identity providers available by default

- **Microsoft Entra** -> /.auth/login/aad
- **Facebook** -> /.auth/login/facebook
- **Google** -> /.auth/login/google
- **X** -> /.auth/login/x
- **Apple** (Preview)
- **OpenID Connect** -> /.auth/login/<providerName>
- **GitHub** -> /.auth/login/github

Discover App Service networking features

Multi-tenant deployments:

- **Inbound features**
  - App-assigned address
  - Access restrictions
  - Service endpoints
  - Private endpoints
- **Outbound features**
  - Hybrid Connections
  - Gateway-required VNet Integration
  - VNet Integration

Single-tenant networking

- Azure App Service Environment hosts Isolated SKU plans directly in your Azure virtual network.

- **External:** Exposes the hosted apps by using an IP address that is accessible on the internet.

- **Internal load balancer:** Exposes the hosted apps on an IP address inside your virtual network.

Configure web app settings

Configure application settings

- In **App Service**, app settings are passed as **environment variables** to the application code.
- For **Linux apps** and custom containers, App Service passes app settings to the container using the ***--env*** flag to set the environment variable in the container.
- In **either case**, they're injected into your app environment at **app startup**.
- When you add, remove, or edit app settings, App Service triggers an app restart.

Configure general settings

Stack settings

The software stack to run the app, including the language and SDK versions.

Platform settings

Configure settings for the hosting platform, including:

- WebSocket protocol
- Always On
- Managed pipeline version
- HTTP version
- ARR affinity

Debugging

Enable remote debugging for ASP.NET , ASP.NET Core, or Node.js apps.

Incoming client certificates

Require client certificates in mutual authentication. TLS mutual authentication is used to restrict access to your app by enabling different types of authentication for it.

Configure path mappings

Linux and containerized apps

- Add custom storage for your containerized app.
- Containerized apps include all Linux apps and Windows and Linux custom containers running on App Service.

Windows apps (uncontainerized)

- Customize the IIS handler mappings and virtual applications and directories.
- Handler mappings enable adding custom script processors to handle requests for specific file extensions.

Enable diagnostic logging

Type	Platform	Description
Application logging	Windows, Linux	Logs messages generated by your application code. The messages are generated by the web framework you choose.
Web server logging	Windows	Raw HTTP request data in the W3C extended log file format.
Detailed error messages	Windows	Copies of the *.html* error pages that would have been sent to the client browser.
Failed request tracing	Windows	Detailed tracing information on failed requests
Deployment logging	Windows, Linux	Deployment logging happens automatically and there are no configurable settings for deployment logging.

Configure security certificates

Options for adding certificates in App Service

- Free App Service managed certificate
- Purchase an App Service certificate
- Import a certificate from Key Vault
- Upload a private certificate
- Upload a public certificate

Scale apps in Azure App Service

Examine autoscale factors

- Autoscaling adjusts available resources based on the current demand.
- Autoscaling performs scaling in and out, as opposed to scaling up and down.
- Monitors the resource metrics of a web app as it runs and detects when additional resources are required based on the set conditions.
- When should you consider autoscaling?
  - Autoscaling provides elasticity for your services.
  - Autoscaling improves availability and fault tolerance.
  - Autoscaling isn't the best approach to handling long-term growth.

Identify autoscale factors

Autoscale conditions

- Scale based on a metric, such as CPU usage or the length of a disk queue.
- Scale based on a schedule, such as a time of day or day of the week.
- Can create multiple conditions to handle complex needs.

Autoscale metrics

- Metrics are based on all instances of an app
- CPU percentage
- Memory percentage
- Disk queue length – outstanding I/O requests
- HTTP queue length – number of client requests
- Data in – number of bytes received
- Data Out – number of bytes sent

Enable autoscale in App Service

- Enable autoscaling
  - Not all pricing tiers support autoscaling
  - Some are only single instance or limited to manual scaling
- Add scale conditions
  - A default scale condition is created
  - Edit the default or create additional conditions
- Create scale rules
  - Conditions can contain one or more scale rules
  - Rules can be set based on calendar, metric, or both
- Monitor autoscaling behavior
  - View autoscale changes on the **Run history** chart
  - Tracks number of instances and which condition triggered the change

Explore autoscale best practices

- Ensure the maximum and minimum values are different and have an adequate margin between them
- Choose the appropriate statistic for your diagnostics metric
- Choose the thresholds carefully for all metric types
- Check for conflicts when multiple rules are configured in a condition
- Always select a safe default instance count
- Configure autoscale notifications

Explore Azure App Service deployment slots

Explore staging environments

- Deployment slots enable you to preview, manage, test, and deploy different development environments.
- Deployment slots are live apps with their own host names.
- App content and configuration elements can be swapped between two slots.
- You can use a separate deployment slot instead of the default production slot when you're running in the **Standard**, **Premium**, or **Isolated** App Service plan tier.
- Deploying to a non-production slot has the following benefits:
  - You can validate app changes in a staging deployment slot before swapping it with the production slot.
  - Deploying an app to a slot first and swapping it into production makes sure that all instances of the slot are warmed up before being swapped into production.
  - After a swap, the slot with previously staged app now has the previous production app.

Examine slot swapping

When swapping slots, App Service does the following

1

- App Service authentication settings, if enabled.

Applies the following settings from the target slot to all instances of the source slot:

- Slot-specific app settings and connection strings, if applicable.
- Continuous deployment settings, if enabled.

2

Wait for every instance in the source slot to complete its restart.

3

If local cache is enabled, trigger local cache initialization by making an HTTP request to the application root ("/") on each instance of the source slot.

4

If auto swap is enabled with custom warm-up, trigger Application Initiation by making an HTTP request to the application root ("/") on each instance of the source slot.

5

If all instances on the source slot are warmed up successfully, swap the two slots by switching the routing rules for the two slots.

6

Now that the source slot has the pre-swap app previously in the target slot, perform the same operation by applying all settings and restarting the instances.

Swap deployment slots

- Swap deployment slots on your app's Deployment slots page and the Overview page.
- Configure auto swap
- Swap with preview
- Specify a custom warm-up
- Roll back and monitor a swap

Route traffic in App Service

Route production traffic automatically

- Go to your app's resource page and select **Deployment slots**.
- In the **Traffic %** column of the slot you want to route to, specify a percentage (between 0 and 100) to represent the amount of total traffic you want to route.

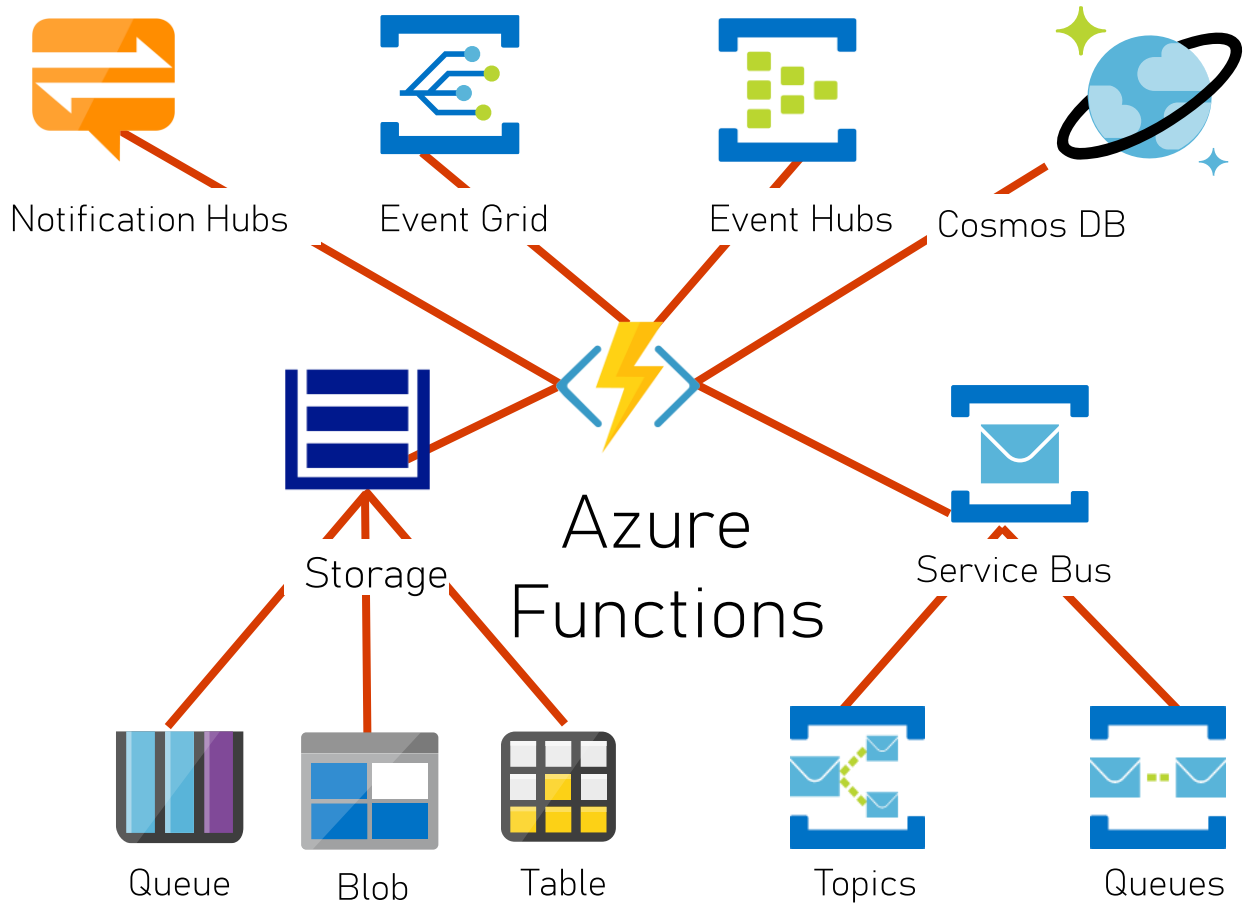
Route production traffic manually

- In addition to automatic traffic routing, App Service can route requests to a specific slot.
- This is useful when you want your users to be able to opt in to or opt out of your beta app.
- To route production traffic manually, you use the **x-ms-routing-name** query parameter.

Explore Azure Functions

Discover Azure Functions

- Consider **Functions** for tasks like image or order processing, file maintenance, or for any tasks that you want to run on a schedule.
- Azure Functions supports *triggers*, which start execution of your code, and *bindings*, which simplify coding for input and output data.



Compare Azure Functions hosting options

Hosting plans:

- Consumption plan
- Flex Consumption plan
- Premium plan
- Dedicated plan (App Service)
- Container Apps

Hosting plans dictate:

- How your function app is scaled.
- The resources available to each function app instance.
- Support for advanced functionality, such as Azure Virtual Network connectivity.

Compare Azure Functions hosting options

Hosting plan	Description
Consumption	Default plan. Pay for compute resources only when your functions are running with automatic scale. Instances of the Functions host are dynamically scaled based on the number of incoming events.
Flex Consumption	High scalability with compute choices, virtual networking, and pay-as-you-go billing. Automatic scaling, can specify pre-provisioned instances.
Premium	Automatically scales based on demand using prewarmed workers, which run applications with no delay after being idle, runs on more powerful instances, and connects to virtual networks.
Dedicated	Run your functions within an App Service plan, Best for long-running scenarios where Durable Functions can't be used.
Container Apps	Supports packaging custom libraries with your function code, provides high-end processing power provided by dedicated GPU compute.

Scale Azure Functions

- The unit of scale for Azure Functions is the function app.
- A *scale controller* monitors the rate of events to determine whether to scale out or scale in.
- Instances may be "scaled in" to zero when no functions are running within a function app.
- The next request has the latency – a *cold start* – of scaling from zero to one

Explore Azure Cosmos DB

Introduction

- Azure Cosmos DB is a fully managed NoSQL database
- Designed to provide low latency, elastic scalability of throughput
- Well-defined semantics for data consistency, and high availability

Identify key benefits of Azure Cosmos DB

- Azure Cosmos DB is a fully managed NoSQL database
- Designed to provide low latency, elastic scalability of throughput
- Well-defined semantics for data consistency, and high availability
- **Global replication:** Automatic and synchronous multi-region replication, supports automatic and manual failover
- **Varied consistency levels:** Offers five consistency models. Provides control over performance-consistency tradeoffs, backed by comprehensive SLAs
- **Low latency:** Serve <10 ms read and <10 ms write requests at the 99th percentile
- **Elastic scale-out:** Elastically scale throughput from 10 to 100s of millions of requests/sec across multiple regions

Explore the resource hierarchy

Elements in an Azure Cosmos DB account

- The account is the fundamental unit of global distribution and high availability.
- Azure Cosmos DB account contains a unique DNS name

Azure Cosmos DB databases

- You can create one or multiple Azure Cosmos DB databases under your account.
- A database is analogous to a namespace.
- A database is the unit of management for a set of Azure Cosmos DB containers.

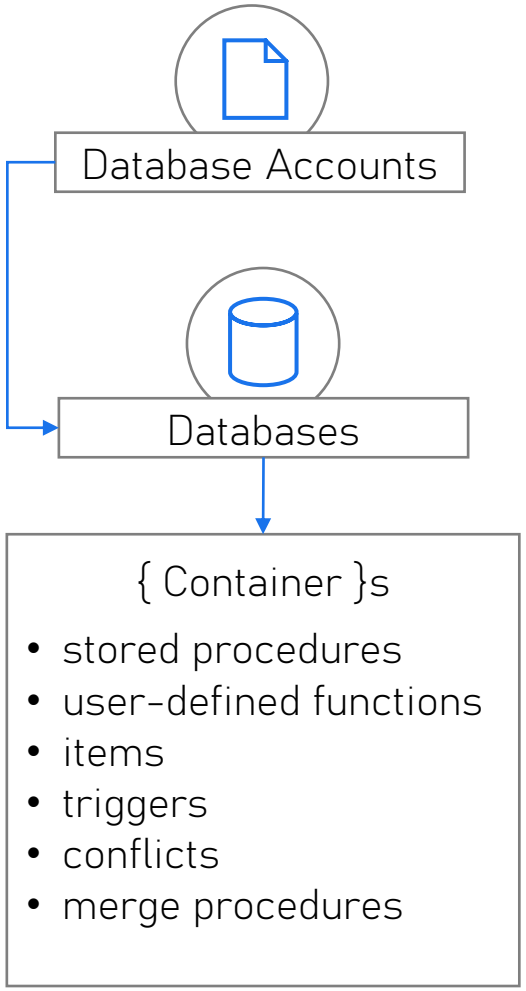
Azure Cosmos DB containers

- An Azure Cosmos DB container is the unit of scalability both for provisioned throughput and storage.
- A container is horizontally partitioned and then replicated across multiple regions.

Azure Cosmos DB items

- Depending on which API you use, an Azure Cosmos DB item can represent either a document in a collection, a row in a table, or a node or edge in a graph.

Explore the resource hierarchy



Depending on the Cosmos API, a container is realized as:

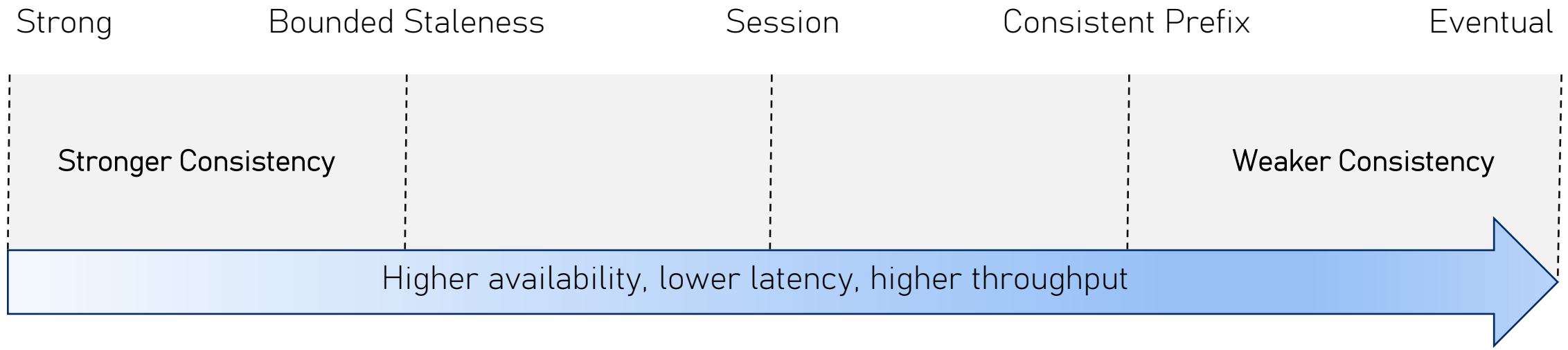
- collection
- table
- graph
- …

Depending on the Cosmos API, an item is realized as:

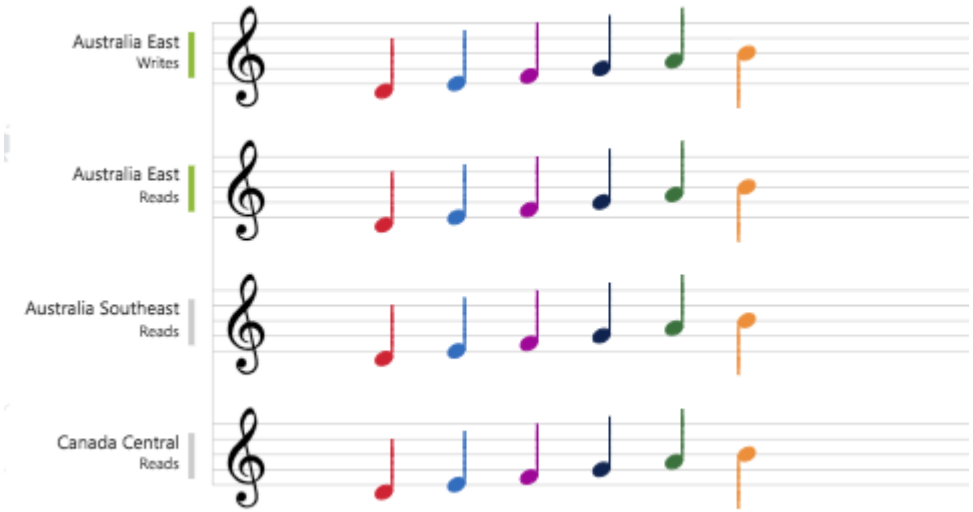
- document
- row
- node
- edge
- …

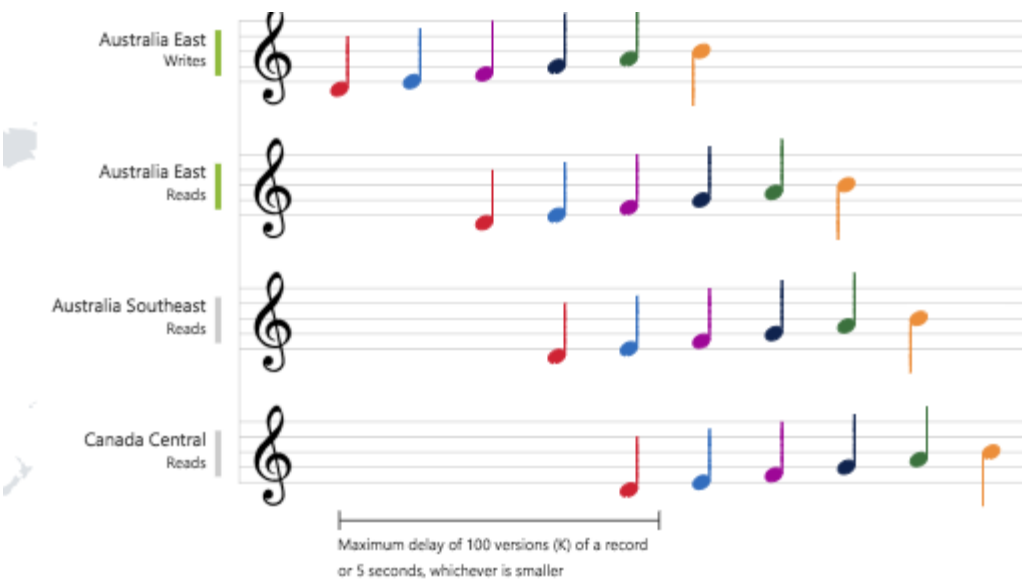
Explore consistency levels

Azure Cosmos DB approaches data consistency as a spectrum of choices instead of two extremes.



Explore consistency levels





Choose the right consistency level

Azure Cosmos DB for NoSQL Azure Cosmos DB for Table

For many real-world scenarios, session consistency is optimal and it's the recommended option.

If your application requires strong consistency, it is recommended that you use bounded staleness consistency level.

If you need the highest availability and the lowest latency, then use eventual consistency level.

Azure Cosmos DB for Cassandra Azure Cosmos DB for MongoDB Azure Cosmos DB for Apache Gremlin

Azure Cosmos DB provides native support for wire protocol-compatible APIs for popular databases.

These include API for MongoDB, API for Apache Cassandra, and API for Apache Gremlin.

When using API for Apache Gremlin the default consistency level configured on the Azure Cosmos account is used.

Consistency guarantees in practice

Consistency guarantees for a read operation correspond to the freshness and ordering of the database state that you request.

Read-consistency is tied to the ordering and propagation of the write/update operations.

The Probabilistically Bounded Staleness metric provides an insight into how often you can get a stronger consistency than the configured level.

Explore supported APIs

- **API for NoSQL:** This API stores data in document format. It offers the best end-to-end experience as we have full control over the interface, service, and the SDK client libraries.
- **API for MongoDB:** This API stores data in a document structure, via BSON format. It is compatible with MongoDB wire protocol.
- **API for Apache Cassandra:** This API stores data in column-oriented schema and is wire protocol compatible with Apache Cassandra.
- **API for Table:** This API stores data in key/value format.
- **API for Apache Gremlin:** This API allows users to make graph queries and stores data as edges and vertices.
- **API for PostgreSQL:** Stores data either on a single node, or distributed in a multi-node configuration

Discover request units

Account types and Request Units

The type of Azure Cosmos DB account created determines the way RUs are charged, there are three modes of account creation:

- **Provisioned throughput:** You manage the number of RUs for your app in 100 RUs per second increments.
- **Serverless:** Billed for the number of RUs consumed by your database operations.
- **Autoscale:** Automatically and instantly scale RUs based on usage.

Manage Container Images in Azure Container Registry

Discover the Azure Container Registry

Use the Azure Container Registry (ACR) service with your existing container development and deployment pipelines or use Azure Container Registry Tasks to build container images in Azure.

Use cases

Pull images from an Azure container registry to various deployment targets:

- *Scalable orchestration systems* that manage containerized applications across clusters of hosts.
- *Azure services* that support building and running applications at scale.

Azure Container Registry service tiers

Azure Container Registry is available in multiple service tiers.

- Basic
- Standard
- Premium

Discover the Azure Container Registry

Supported images and artifacts

- Grouped in a repository, each image is a read-only snapshot of a Docker-compatible container.
- Azure container registries can include both Windows and Linux images.
- Azure Container Registry also stores Helm charts and images built to the Open Container Initiative (OCI) Image Format Specification.

Azure Container Registry Tasks

- Use Azure Container Registry Tasks (ACR Tasks) to streamline building, testing, pushing, and deploying images in Azure.

Explore storage capabilities

Every Basic, Standard, and Premium Azure container registry benefits from advanced Azure storage features.

- **Encryption-at-rest**: All container images in your registry are encrypted at rest.
- **Geo-redundant storage**: Azure uses a geo-redundant storage scheme to guard against loss of your container images.
- **Geo-replication**: For scenarios requiring even more high-availability assurance, consider using the geo-replication feature of Premium registries.
- **Zone redundancy**: Premium service tier, uses Azure availability zones to replicate your registry to a minimum of three separate zones in each enabled region.
- **Scalable storage**: Create as many repositories, images, layers, or tags as you need, up to the registry limit.

Build and manage containers with tasks

ACR Tasks provides cloud-based container image building for platforms including Linux, Windows, and ARM. It can automate OS and framework patching for your Docker containers.

Task scenarios

ACR Tasks supports several scenarios to build and maintain container images and other artifacts:

- Quick task
- Automatically triggered tasks
- Multi-step task

Run container images in Azure Container Instances

Stale poszukuję nowych możliwości i ekscytujących wyzwań. Jeśli chcesz się ze mną skontaktować, proszę, skorzystaj z poniższych kanałów:

Email: info@zalnet.pl

LinkedIn: https://www.linkedin.com/in/beatazalewa/

Blog: https://zalnet.pl/pl/blog/

X: https://x.com/beatazalewa

GitHub - slajdy: https://github.com/beatazalewa/Conferences/

Github – linki:

https://github.com/beatazalewa/Sierpniowe-kolonie-na-chmurze-Azure-2025/