Microsoft

SaaS Lab

Welcome to the

SaaS Lab Program

Session 5

Application Storage Modernization

This event will be recorded. Your name or other information may end up in the recording. If you do not wish to be recorded, please drop out of this session.

The event will start shortly

Hello, meet your session presenters

Daniel Scott-Raynsford

Partner Technology Strategist

About: Daniel is a technology professional with 23 years' experience designing, building, delivering and running applications and systems utilizing a range of software development technologies. Daniel now has a strong focus on cloud architecture, SaaS development and DevOps practices.

daniel.scottraynsford@microsoft.com
https://www.linkedin.com/in/dscottraynsford/

Microsoft

If you'd like guidance on your Azure modernization journey, please e-mail the SaaS Lab team

saaslab@microsoft.com

We would love to help!

Microsoft

Joining AI & ML Session?

Please fill in the survey to help us tailor the session!

https://forms.office.com/r/hFcUr8XjcY

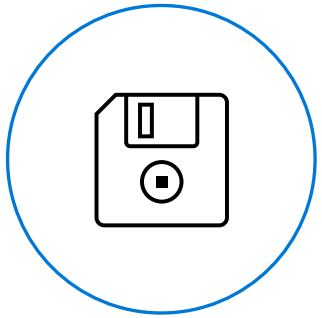
About this session

Agenda

- Modernizing beyond managed services
- Introduction to storage multi-tenancy
- Break for questions & comments
- A typical storage architecture modernization journey
- Quiz time
- Q&A

How do applications commonly store data in Azure?

Azure SaaS apps store application data in different services. Here are the most common* ones in Azure.



File Based Storage

- IaaS Storage Servers (unmanaged)
- Azure Blob Storage
- Azure Files

Relational Databases

- IaaS Databases (unmanaged)
- Azure SQL
- Azure SQL MI
- Azure Database for PostgresSQL
- Azure Database for MySQL

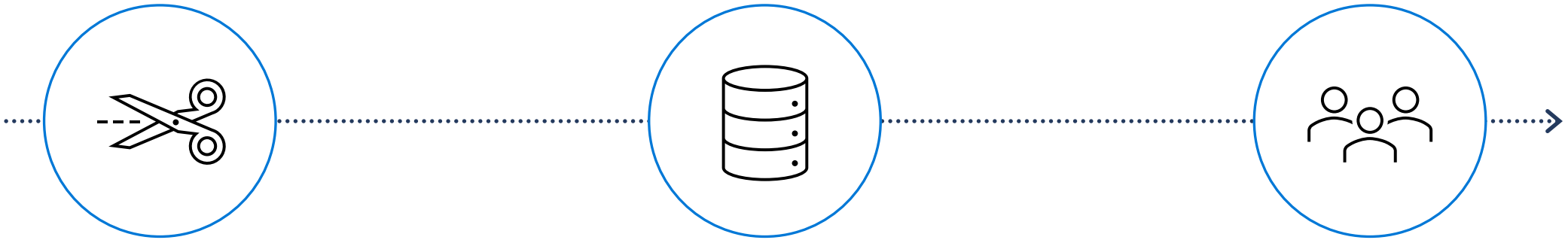
NoSQL Databases

- IaaS NoSQL (e.g., MongoDB)
- Azure Table Storage
- Cosmos DB, which includes:
  - Document DB (MongoDB API, Cassandra API, SQL API)
  - Graph DB (Gremlin API)

** There are other data store types, such like etcd, but those are out of scope for this session.*

Storage modernization journey

The common steps a SaaS builder takes to modernize their application persistence layer.



Decouple Persistence Layer

Separate the application persistence layer (storage) from the application layer.

- **Horizontal scaling of application**
- **Enable high-availability**
- **Modern architecture patterns**

Decoupling is often combined with moving to managed services.

Use Managed Services

Move from unmanaged storage services running in IaaS virtual machines to managed storage services.

- **Reduce operational cost**
- **Increased elasticity**
- **Faster tenant onboarding**

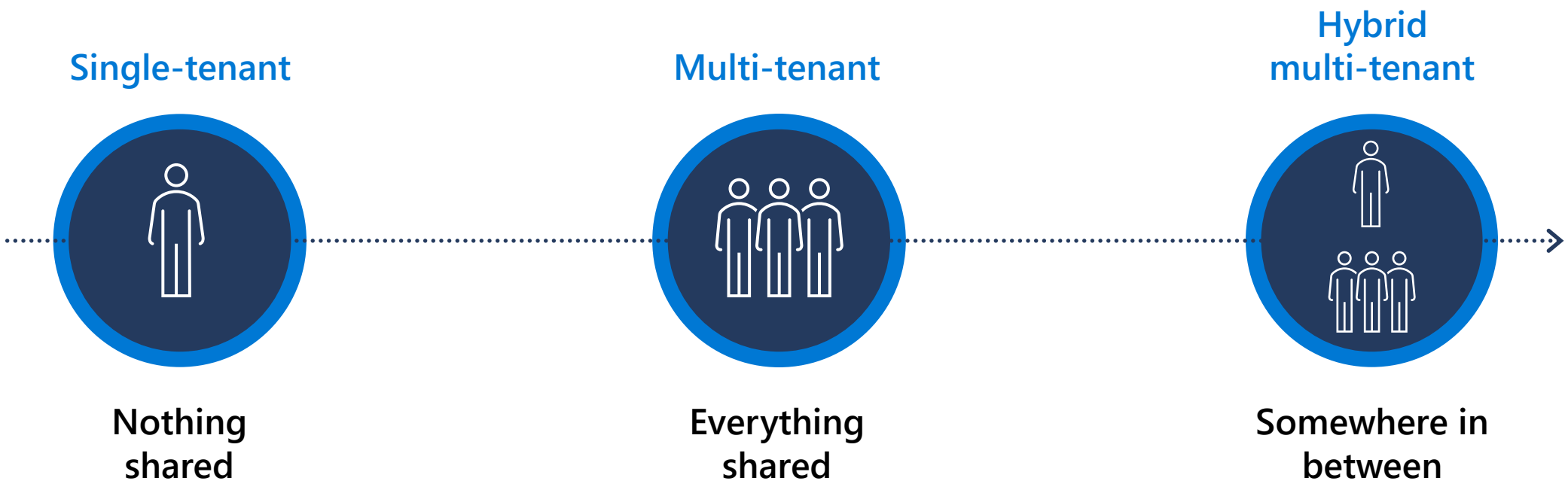
Multi-tenancy

Enable multi-tenant patterns on storage services.

- **Increase resource utilization**
- **Reduce cost per tenant**
- **Increase maximum tenant limits**

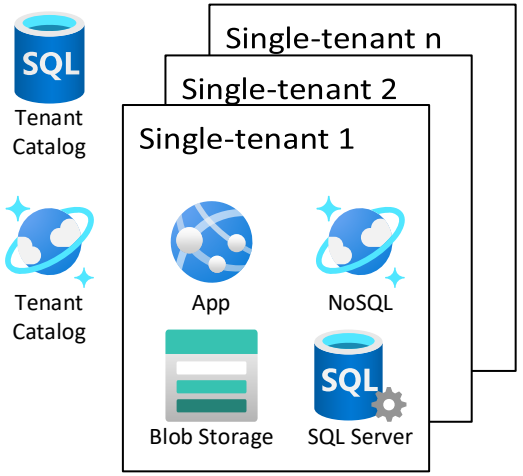
Multi-tenanting of storage can be done with un-managed services, but with reduced effectiveness.

SaaS multi-tenancy



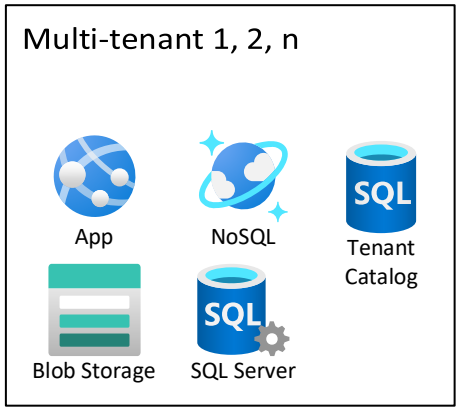
Single-tenant: shared nothing

- Single tenant per deployment
  - Storage account per tenant
  - Database per tenant
  - Document/graph store per tenant
- Customization can be done in each deployment
  - ! This does not scale and will eventually become unmaintainable
- Higher operational costs & more waste
- ☆ High level of tenant isolation
- ☆ Easy to understand & measure per-tenant costs



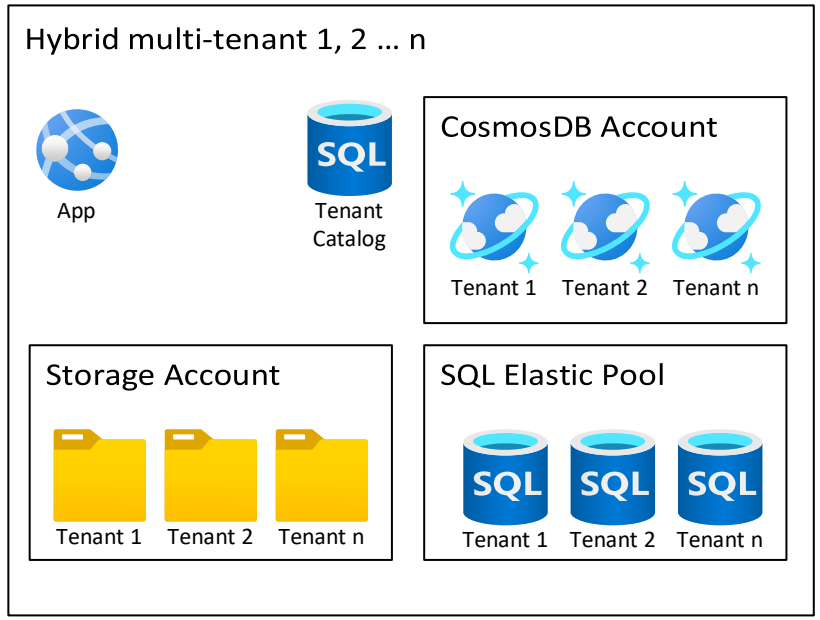
Multi-tenant: shared everything

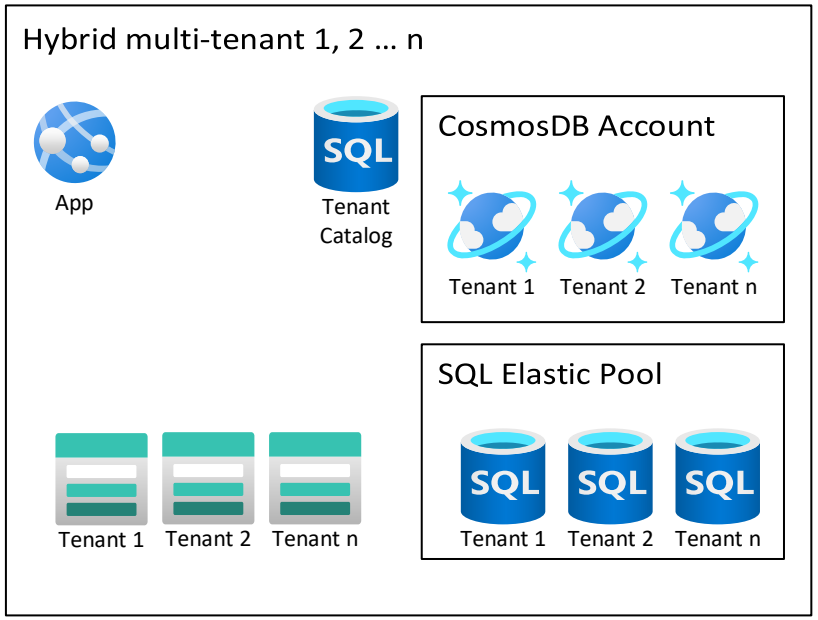
- Multiple tenants in a single deployment
  - Single storage account
  - Single database
  - Single document/graph store
- Tenant customization via data
- ! Lower level of isolation
  - Potential noisy-neighbor
  - Commercial sales challenges
- ! Difficult to measure per-tenant costs



Hybrid multi-tenant: somewhere in between

- Many different hybrid models:





  - And anywhere in between
- Good balance between single & multi-tenant
- Enable different commercial models

Deployment stamps architecture

Enable multi-region deployments

- Reduce latency to regional customers
- Comply with data residency requirements
- Access Azure services only available in specific regions

Enable horizontal scaling within region

- Enable deployment scale out
- Overcome Azure resource limits/subscription quotas
- Segment tenants by requirements

https://docs.microsoft.com/en-us/azure/architecture/patterns/deployment-stamp

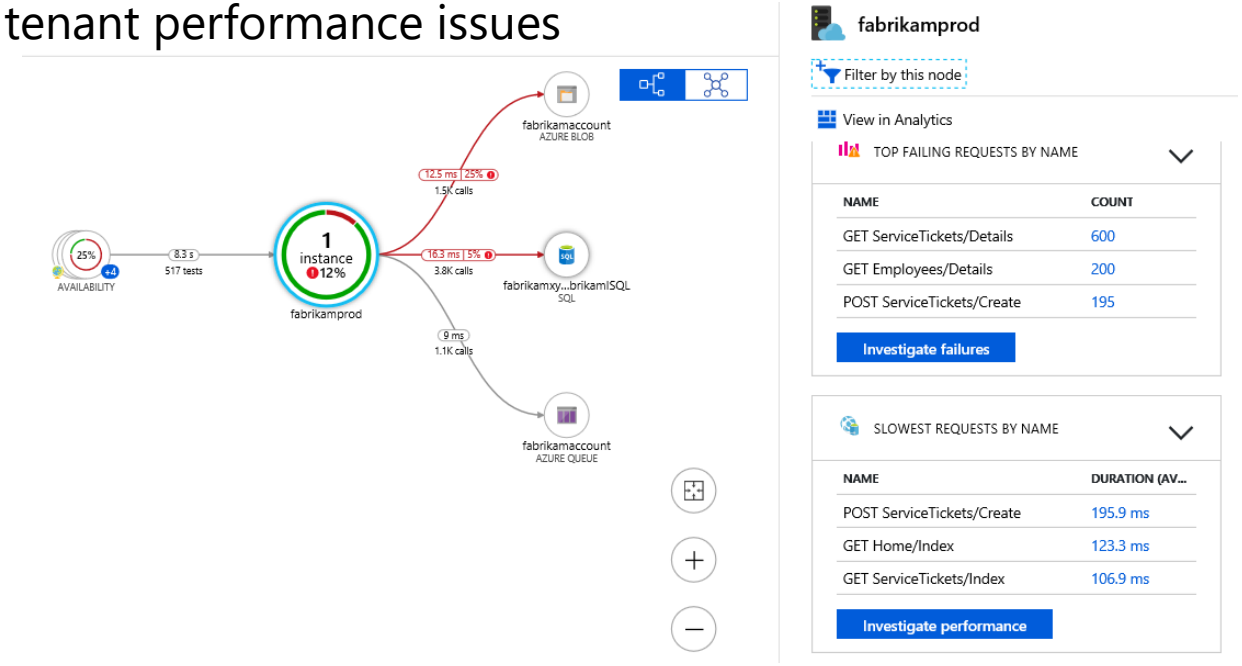
slido

Where are you on your SaaS application storage architecture journey - choose the closest model?

ⓘ Start presenting to display the poll results on this slide.

Monitoring multi-tenant storage

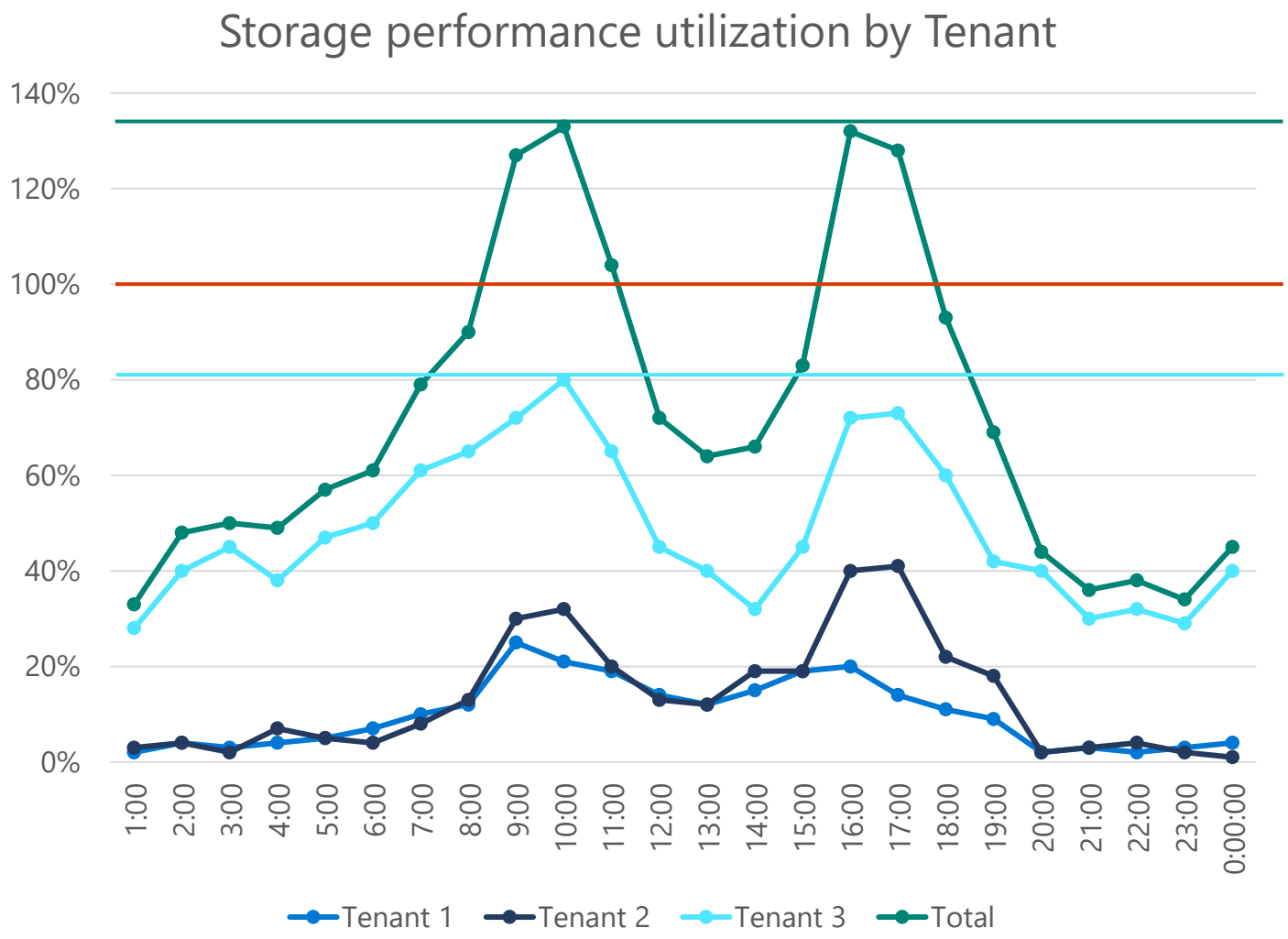
- Enable Application Insights on your application
- Identify **key tenant storage operations** and **continuously monitor baseline performance**
- Enable Application Insights Smart Detection - Watch out for performance anomalies
- Send custom telemetry w/ **tenant information** to Application Insights
- Use ITelemetryInitializer to enrich metrics with TenantID.
- Set up alerts to notify of tenant or system performance degradation
- Trigger automation to mitigate performance conditions (e.g., migrate tenant)
- Use Application Map to trace bottlenecks and tenant performance issues



Application Insights

A quick look at Application Insights
Adding tenant information with ITelemetryInitializer
Monitoring with Application Map

Deployment & tenant storage utilization



Peak load

Overloaded: scale deployment

Noisy tenant: must mitigate

Mitigating noisy tenants

- Enable auto-scale on Azure services
- Optimise/modernize storage services
- Migrate to deployment stamp with lower load
- Migrate to new deployment stamp
- Cordon the tenant with request/usage throttling
- Consider requiring purchase of a higher service tier

Let’s have a break for a moment…

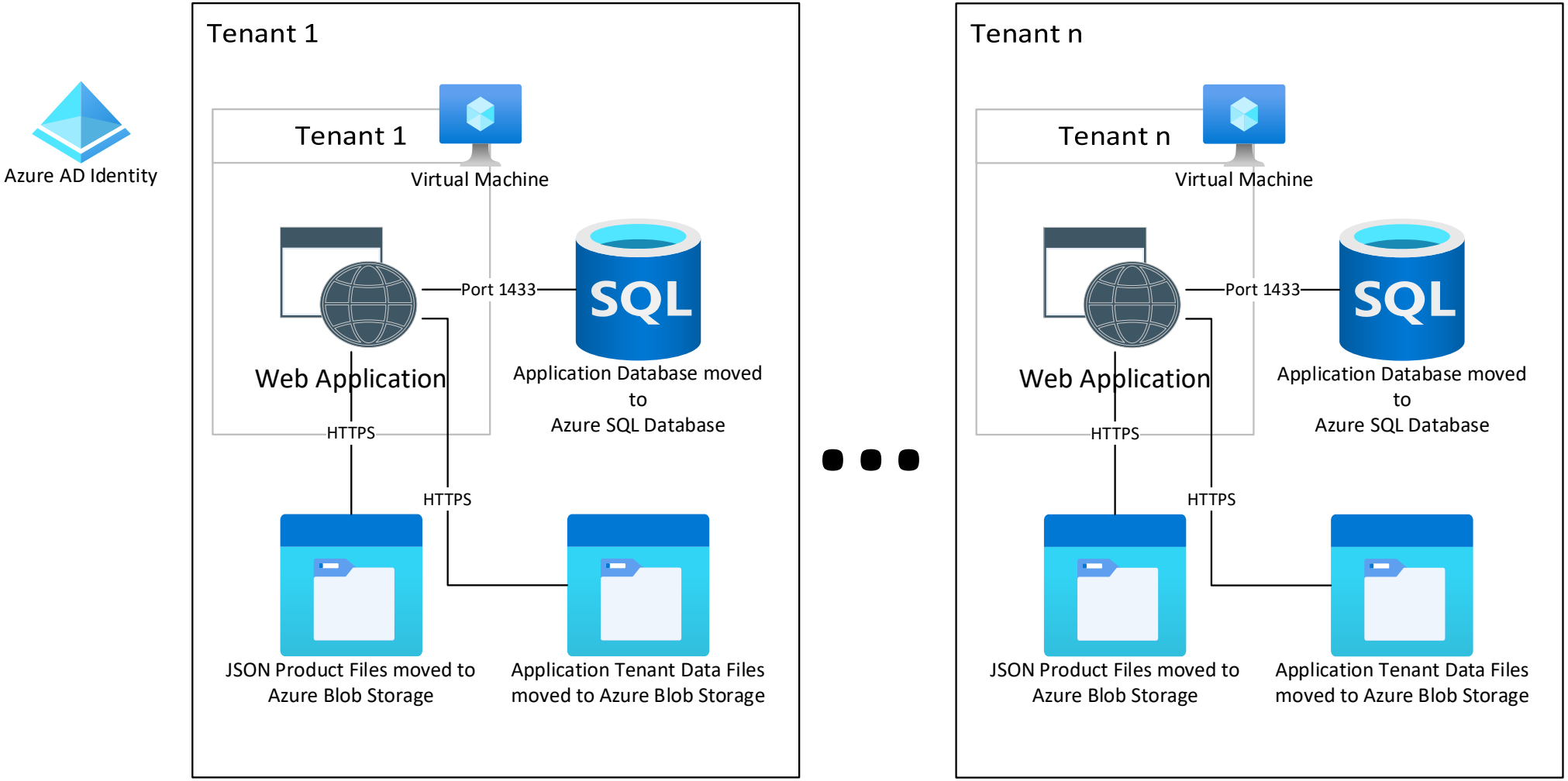
…do you have any questions or comments?

A typical storage architecture modernization journey

TAILWIND TRADERS

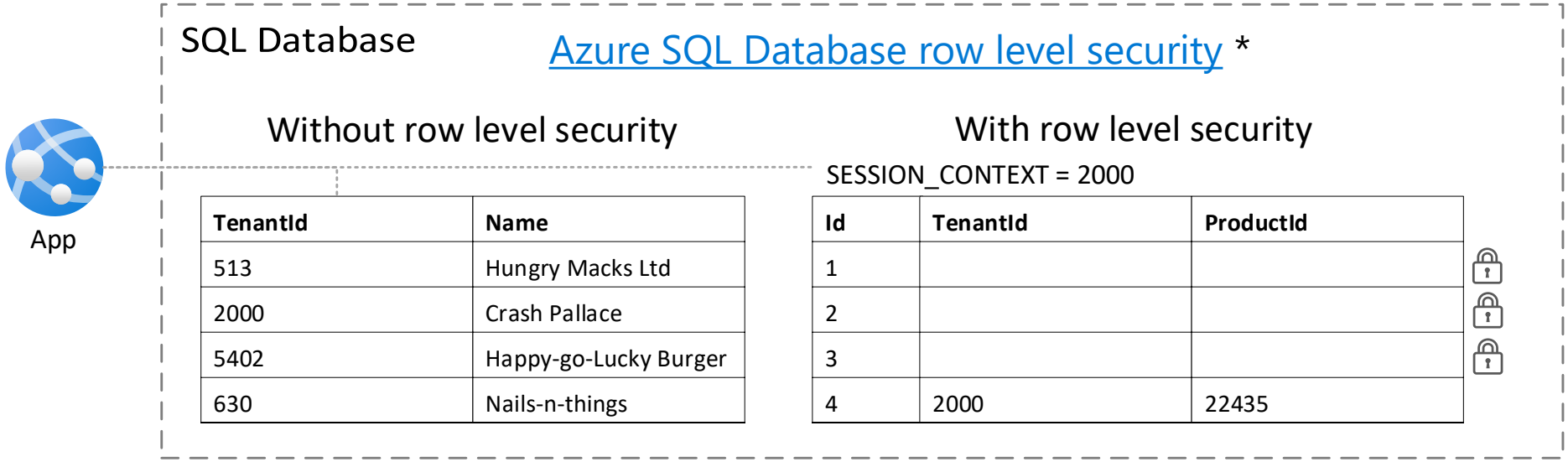
- Recently migrated an on-premises version of their retail services solution as a **SaaS application in Azure**.
- Are expecting to onboard at least **5000 tenants** in 3 years.
- Plan to enable new commercial models like **trial** or **monthly offers**, **paid add-ons** and a **premium tier**.
- Need to **reduce operational cost per tenant** and **onboard time**.
- Delight customers with **high-quality**, **reliable service** and continuous delivery of **innovative new features**.
- A managed identity platform, Azure AD, is used.

Initial single-tenant architecture



Azure SQL multi-tenant patterns

Table multi-tenancy



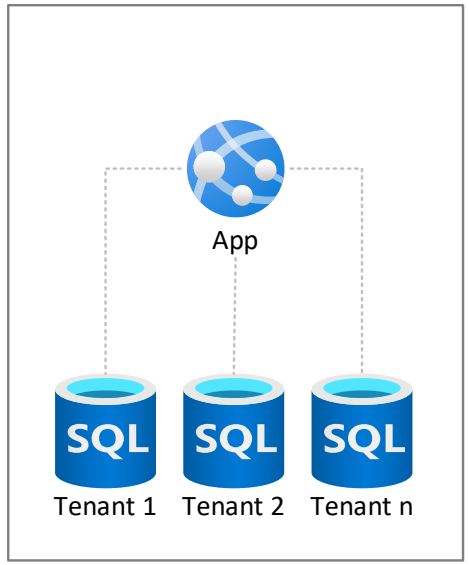
TenantId	Name
513	Hungry Macks Ltd
2000	Crash Pallace
5402	Happy-go-Lucky Burger
630	Nails-n-things

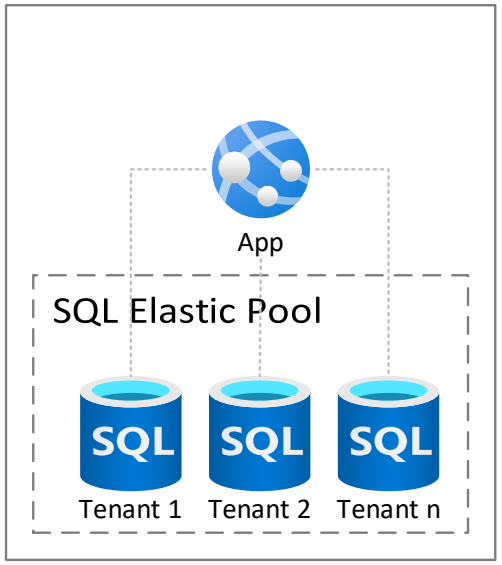
Id	TenantId	ProductId
1		
2		
3		
4	2000	22435

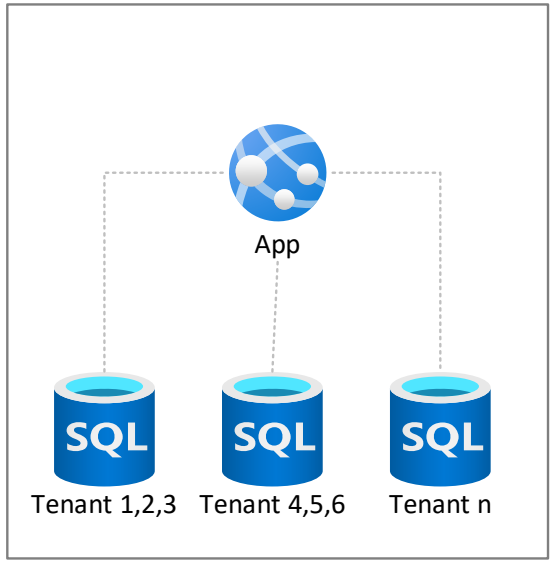
	Without row level security	With row level security
Scale	Low, 1-1,000s	Low, 1-1,000s
Tenant isolation	**Low**	**Medium**
Database cost per tenant	Lowest	Lowest
Development complexity	Medium	Medium
Operational complexity	Low	Low

** Row-level security is supported in other RDBMS in Azure but is not covered here.*

Azure SQL database patterns





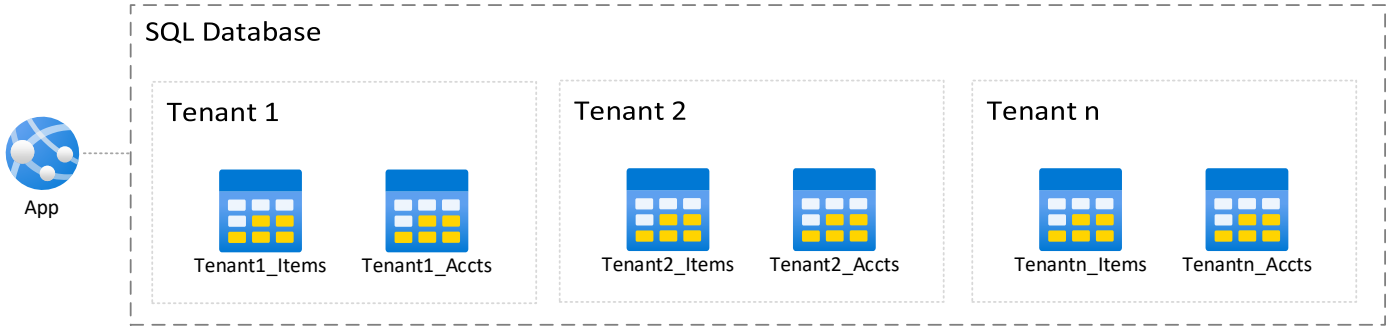


	Database per-tenant	Elastic pool database-per-tenant	Sharded multi-tenant databases
Scale	Very High, 1-100,000s	Very High, 1-100,000s	Unlimited, 1-1,000,000s
Tenant isolation	High	High	Low except for singleton tenant in a database
Database cost per tenant	Low	Low	Lowest
Development complexity	Low	Low	Medium due to sharding
Operational complexity	Low-Medium	Low-Medium	Low-High

https://docs.microsoft.com/en-us/azure/azure-sql/database/saas-tenancy-app-design-patternsharded multi-tenant database

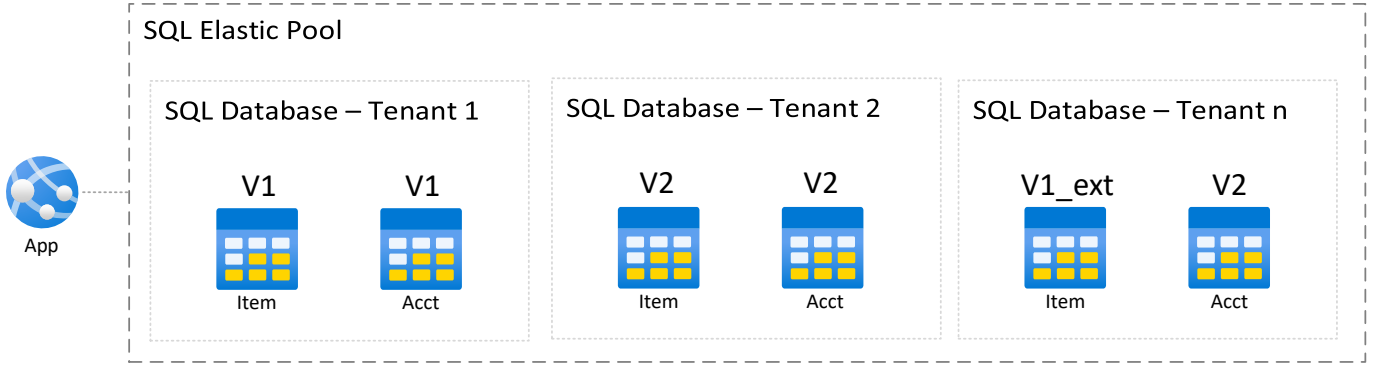
SQL multi-tenant anti-patterns

❌ Single database with multi-tenant table set



Scale	Very Low, 1-100s
Tenant isolation	Low
Database cost per tenant	Low
Development complexity	High; increasing with scale
Operational complexity	High; increasing with scale

❌ Per-tenant schema customizations



Scale	Very High, 1-100,000s
Tenant isolation	High
Database cost per tenant	Low
Development complexity	High; increasing with scale
Operational complexity	High; increasing with scale

Consider adopting NoSQL if per tenant schema customization is required.

Consider Azure limits

Azure SQL Server limits

Resource	Limit
Databases per server	5000
Default number of servers per subscription in any region	20
Max number of servers per subscription in any region	200
DTU / eDTU quota per server	54,000
vCore quota per server per instance	540
Max pools per server	Limited by number of DTUs or vCores. For example, if each pool is 1000 DTUs, then a server can support 54 pools.

Azure SQL Elastic Pool limit

Resource	Limit
Pools per server	500 (default max)

As your application scales to 100s, 10,000s or even 1,000,000s will consider how cost per tenant and performance will be impacted.

Q: How will large numbers of connections affect your front end?

Q: Will you have issues with connection exhaustion?

Q: Can one tenant disrupt service for another?

Consider all pillars of the well-architected framework.

Resource limits for Azure SQL Database

SQL Database resource limits for single databases

SQL Database resource limits for elastic pools and pooled databases

SQL Database resource limits for SQL Managed Instance

Limitations in Azure Database for MySQL

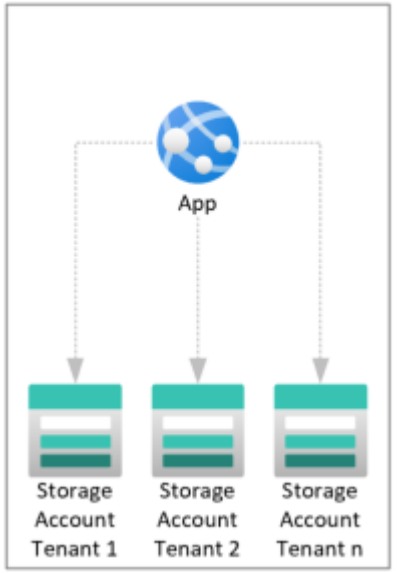
Limitations in Azure Database for PostgreSQL

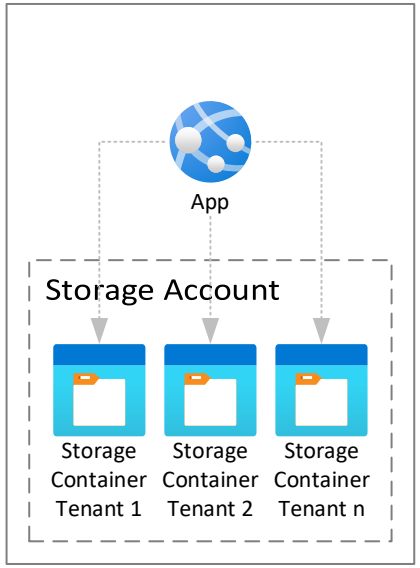
Azure SQL multi-tenant

A brief look at a multi-tenanted Azure SQL Databases

Azure Blob Storage multi-tenant patterns

Azure Blob Storage multi-tenant patterns

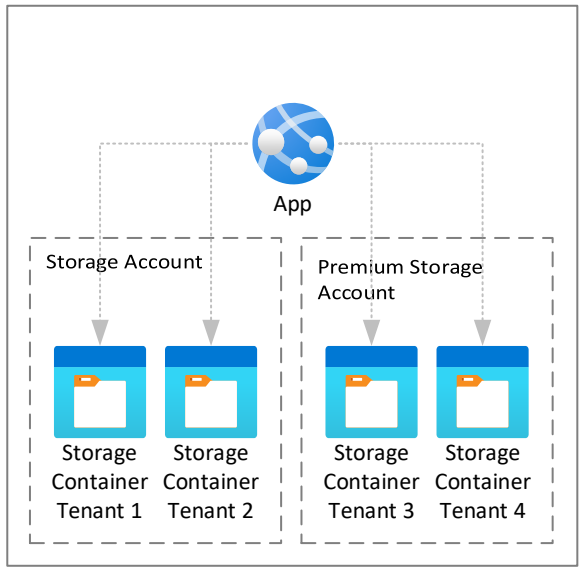


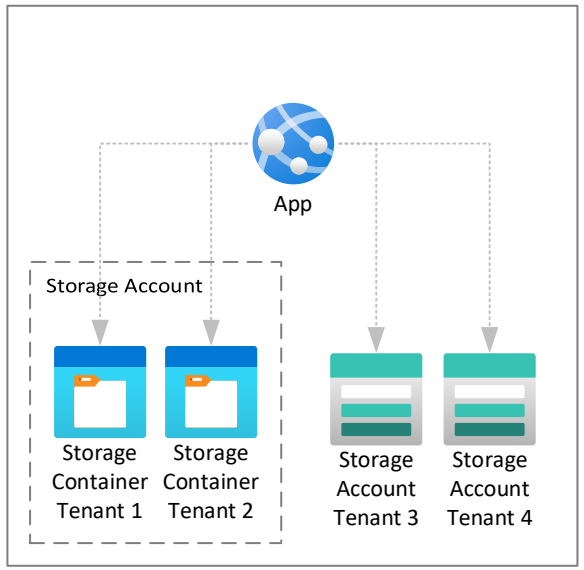


	Account per-tenant	Shared account with container per-tenant
Scale	Very Low, 1-250	Unlimited, 1-1,000,000s
Tenant isolation	High	Low
Performance monitoring and management	Per-tenant	Per-account, per-tenant difficult
Development complexity	Low	Low
Operational complexity	Low	Low

https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/azure-subscription-service-limits#storage-limits

Azure Blob Storage hybrid multi-tenant patterns





	Shared account with container per-tenant with optional tenant service tiers	Shared account with container per-tenant with account per-tenant option
Scale	Unlimited, 1-1,000,000s	Unlimited, 1-1,000,000s
Tenant isolation	High	Low/High
Performance monitoring and management	Per-account, per-tenant difficult	Per-tenant + Per-account, per-tenant difficult
Development complexity	Low-Medium	Low-Medium
Operational complexity	Low-Medium	Low-Medium

https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/azure-subscription-service-limits#storage-limits

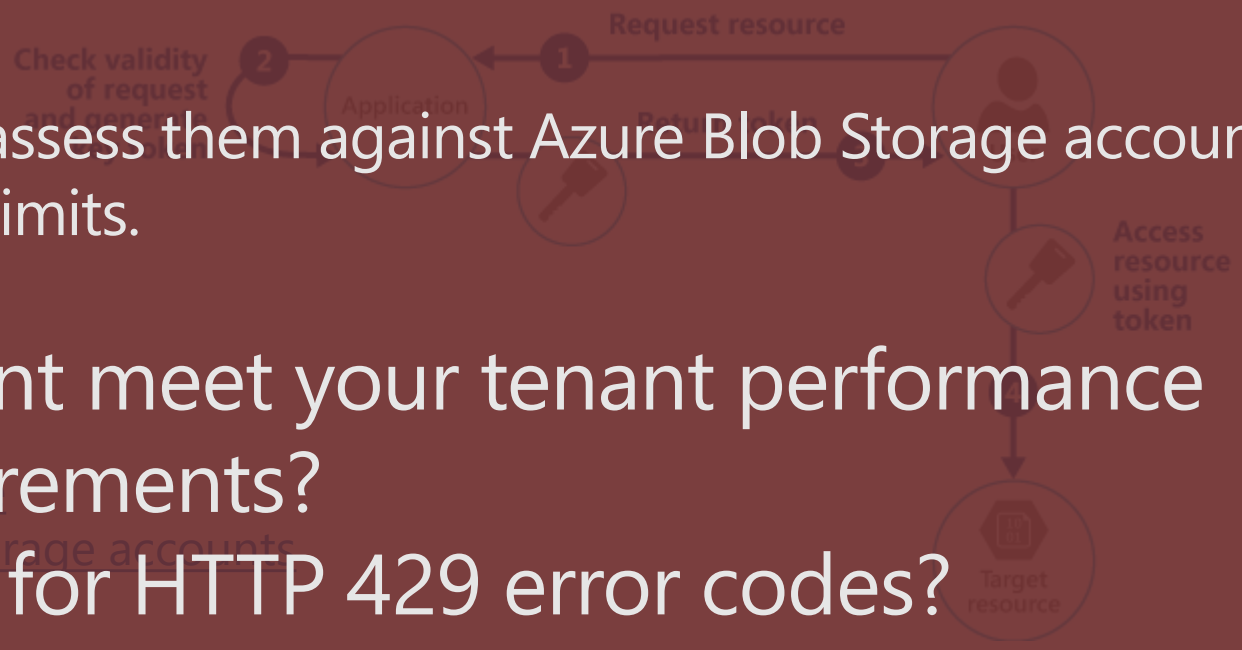
Considerations

Azure Storage Account limits

Resource	Limit
Storage Accounts per region per subscription	250
Egress/Ingress per account	Unlimited
Maximum storage account capacity	5 PiB
Maximum request rate per storage account	20,000 requests per second

Resource limits for Azure Blob Storage
Scalability and performance targets for Blob Storage
Scalability and performance targets for standard storage accounts

Consider using valet key pattern



Consider blob tenant access patterns and assess them against Azure Blob Storage account limits.

Q: Will a single storage account meet your tenant performance requirements?

Q: Are you monitoring for HTTP 429 error codes?

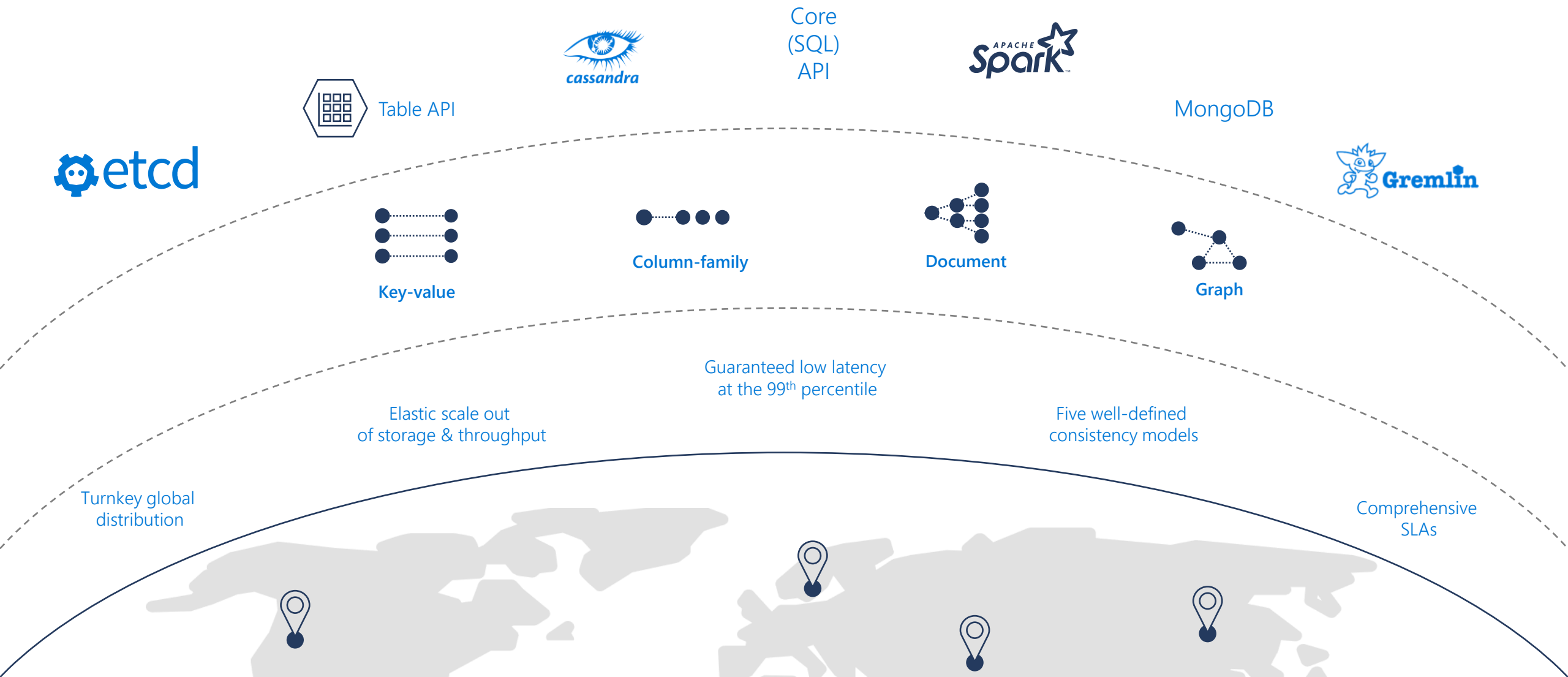
- Model blob tenant access patterns and assess against limits:
  - Ingress/Egress
  - Requests per second
- Clients will receive 429 errors if throttled.
  - SDKs will usually automatically retry, but performance will degrade.

Azure Blob Storage multi-tenant

A brief look at a multi-tenanted Azure Blob Storage containers

Azure Cosmos DB

A globally distributed, massively scalable, multi-model database service



Cosmos DB advantages for multi-tenant

- Different tenancy models
- Modern cloud design patterns
- Dedicated and shared throughput on databases & containers
- Supports autoscale throughput
- Scale across regions enables Geode architecture pattern *
- Easy to measure tenant consumption

Cosmos DB multi-tenant patterns

	Database Account (per tenant)	Container w/ Dedicated Throughput (per tenant)	Container w/ Shared Throughput (per tenant)	Partition Key (per tenant)
Isolation Knobs	Independent geo-replication knobs Multiple throughput knobs (dedicated throughput – eliminating noisy neighbors)	Independent throughput knobs (dedicated throughput – eliminating noisy neighbors) Group tenants within database account(s) based on regional needs	Share throughput across tenants grouped by database (great for lowering cost on "spiky" tenants) Easy management of tenants (drop container when tenant leaves) Mitigate noisy-neighbor blast radius (group tenants by database)	Share throughput across tenants grouped by container (great for lowering cost on "spiky" tenants) Enables easy queries across tenants (containers act as boundary for queries) Mitigate noisy-neighbor blast radius (group tenants by container)
Throughput requirements	>400 RUs per Tenant (> $24 per tenant)	>400 RUs per Tenant (> $24 per tenant)	>100 RUs per Tenant (> $6 per tenant)	>0 RUs per Tenant (> $0 per tenant)
T-Shirt Size	Large Example: Premium offer for B2B apps	Large Example: Premium offer for B2B apps	Medium Example: Standard offer for B2B apps	Small Example: B2C apps

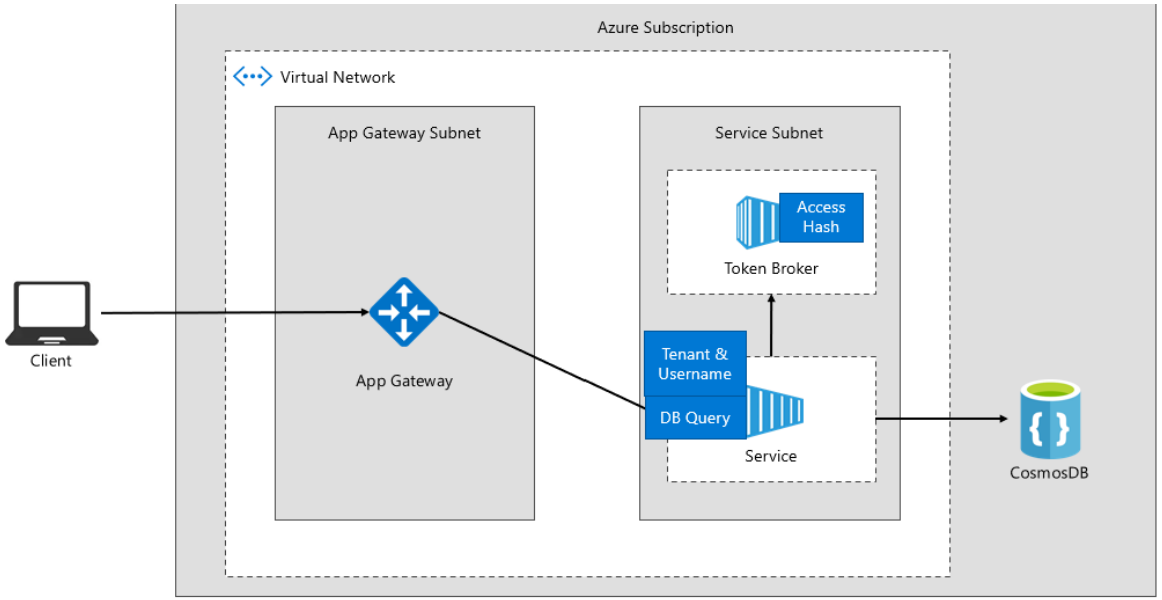
Considerations

Provisioned throughput limits

Resource	Limit
Maximum number of databases	Unlimited
Maximum number of containers per database with shared throughput	25
Maximum number of containers per database or account with dedicated throughput	Unlimited

Azure Cosmos DB service quotas

Use Token Broker pattern for tenant security

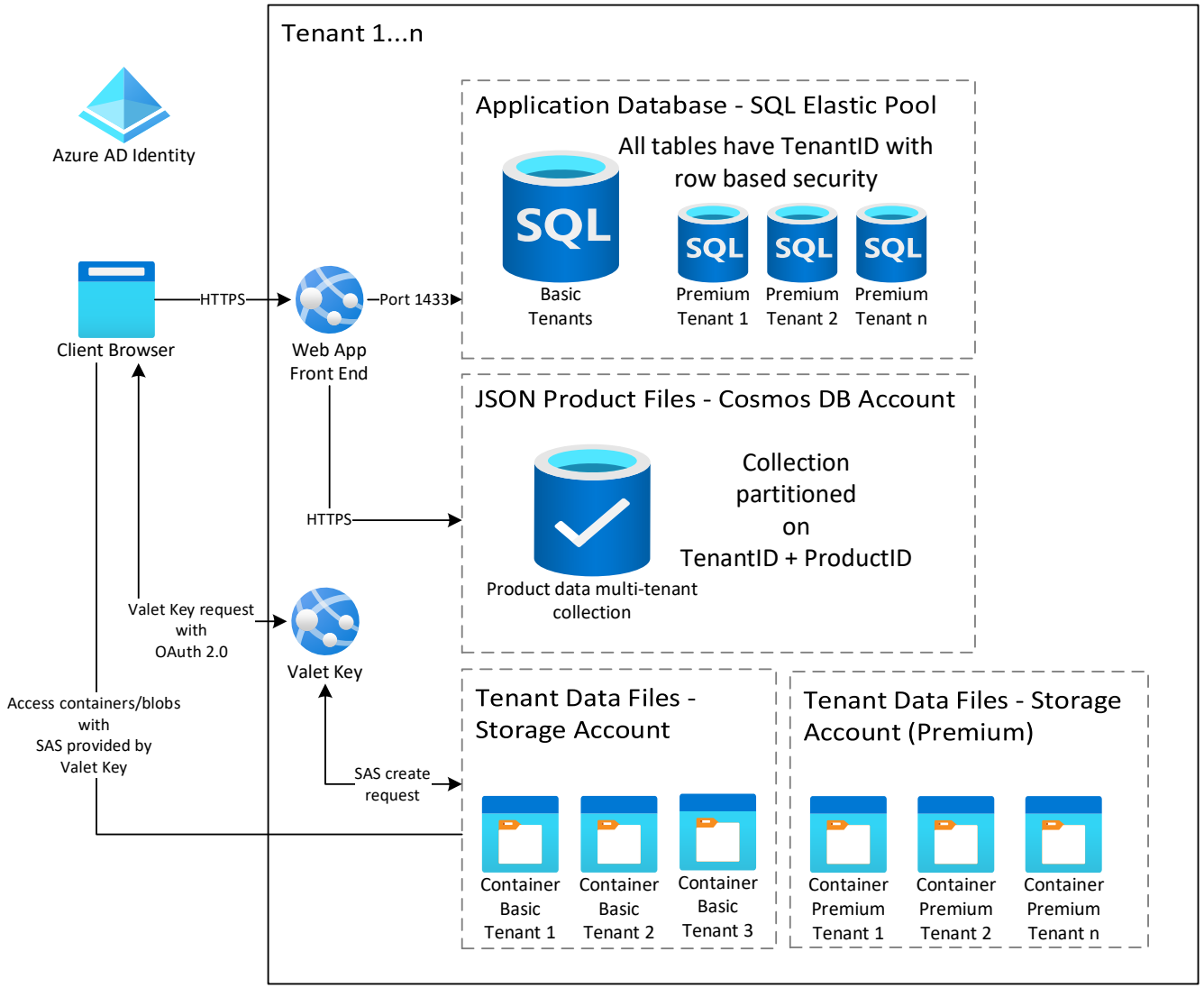


- Partition key modelling is critical
- Clients will receive 429 errors if throttled.
  - SDKs will usually automatically retry, but performance will degrade.

Azure Cosmos DB multi-tenant

A brief look at a multi-tenanted Azure Cosmos DB

Multi-tenant with architecture



What else should we consider in more detail?

- Reliability
- Security
- Limitations of front-end services

Knowledge check time!

http://www.kahoot.it or use the Kahoot! mobile app

<Game Pin>

Session takeaways

Continuously Improve

Storage modernization is a journey. Continuously evaluate your customers needs.

Evaluate and review new storage technologies and architectural methodologies for benefits.

Don't be afraid to **rearchitect**.

Adopt a strategic Multi-tenant approach

Plan your multi-tenant or hybrid multi-tenant architecture carefully, considering:

- Customer needs
- Commercial models
- Azure service limits
- Development and Operational complexity

Monitor & Protect

Multi-tenant carefully by monitoring and protecting your customers. Use Application Insights and custom telemetry.

Monitor key tenant baseline metrics such as page load time for important pages.

Plan for Future Scale

Architect your solution to account for future customer growth, selling into different regions and offering different commercial models.

Avoid storage anti-patterns

There are patterns that will inhibit your future growth and result in an unmanageable, unscalable or wasteful solution. Avoid these.

Use features of managed services

Managed storage services in Azure provide many features that can help you reduce costs and provide new features for your customers.

Consider additional value you could add just be using the services provided.

Q & A

Your feedback is important

Please help us improve this program by completing this short feedback form.

https://aka.ms/saaslabfeedback5

Microsoft

If you'd like more help on your Azure modernization journey, please e-mail the SaaS Lab team

saaslab@microsoft.com

Thank you for being part of the SaaS Lab Program