

How to build Highly Available and Scalable Cloud solutions on Azure, what to do and what to avoid!

David.Gristwood@microsoft.com

Chris.Marchal@microsoft.com

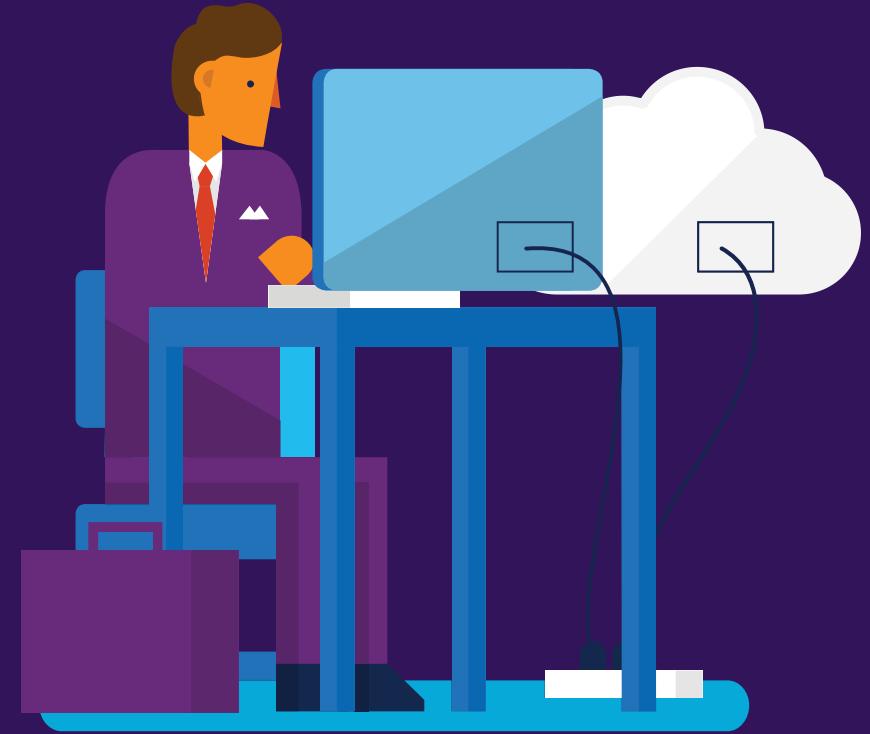
Ross.P.Smith@microsoft.com



<http://aka.ms/azureevent>

#Azure

#BuildWithAzure



vipazure@microsoft.com

Other technical activities from our team

Code with hacks

<https://aka.ms/ukocpisvhack0817>



Microsoft UK 'Code With' Azure hacks 2017/18

Microsoft UK's One Commercial Partner team plan to run a number of coding hacks over the 2017/18 period, focused on a number of key Azure related technologies. Please use this form to register your interest for any of these hacks along with the technologies or scenarios you are most interested in. As and when these hacks get scheduled, we will contact you to give you an opportunity to register for them. If you have any questions about these hacks, their suitability for you, or this application process, please email vipazure@microsoft.com

* Required

1. Company name *

Enter your answer

2. Primary contact with company email address (We will use this to contact you directly with a registration link, as and when a scheduled hack is announced) *

Enter your answer

Workshops

<https://aka.ms/AzureEventList>



Azure Serverless & Microservices Workshop – London October 10th 2017

What if you could spend all your time building and deploying great apps, and none of your time managing servers? Server less computing lets you do just that, because the infrastructure you need to run and scale your apps is managed for you. Alternatively, are you using a monolith architecture at work and keen to learn how you can decompose it into discreet microservices? Our technical experts will walk through Microsoft's offering with talks and demos highlighting what to look for and what to avoid.

[Register for London October 10th](#)



GDPR, Security and Privacy features for cloud applications – London, October 17th 2017

Join us for a day in which we explore Microsoft Azure and the services and features related to security, privacy, governance and GDPR. In this workshop we will explore how to secure your investments, infrastructure, data and applications whilst covering topics such as security patterns, access control, identity, networking, hybrid configurations, databases and threat detection.

[Register for London October 17th](#)

TWO DAY AZURE HACKATHON FOR ISVs

11TH & 12TH OCTOBER

MICROSOFT, THAMES VALLEY PARK, READING



It will be a practical "hands on" development "hack", in which attendees will move their own software to Azure.

Applications close at 5pm on Monday 2nd October. All applicants will be informed by 4th October.

Apply - <https://aka.ms/ukocpisvsohackoct17>

Membership

How It Works ▾

Program Updates

Incentives

Your Accounts and Reports ▾

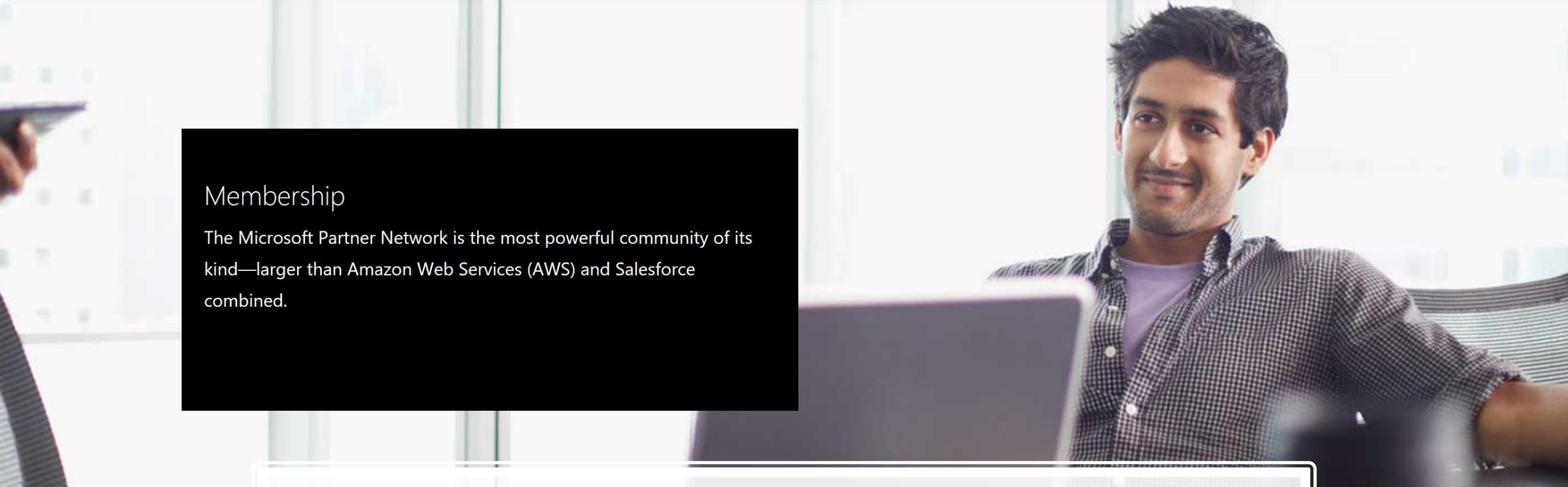
Enroll

Renew

Membership

The Microsoft Partner Network is the most powerful community of its kind—larger than Amazon Web Services (AWS) and Salesforce combined.

Join MPN as a Network member, as entry level into the program
<https://partner.microsoft.com/en-gb/membership>



You want to grow your business. We know how.

Partnership with Microsoft pays off.

Agenda

1. Main session:
 - What to do (Cloud Design Patterns and good practices)
 - What to avoid (Classic errors and mistakes)
2. Lunch
3. Afternoon
 - Demos, deeper dives, interactive group discussion
 - Finish around 3:30



The Question is:

*How do partners build very highly scalable and
highly available cloud solutions that can
perform with confidence on the Azure cloud,
even under extreme loads?*

2 subjects, 5 key areas

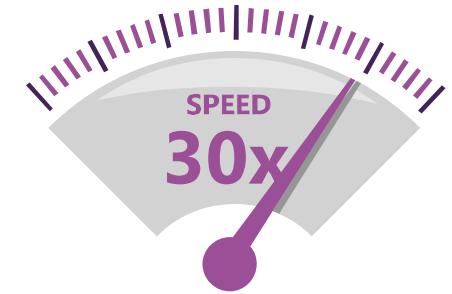
{"Scalability and Performance"}

David Gristwood



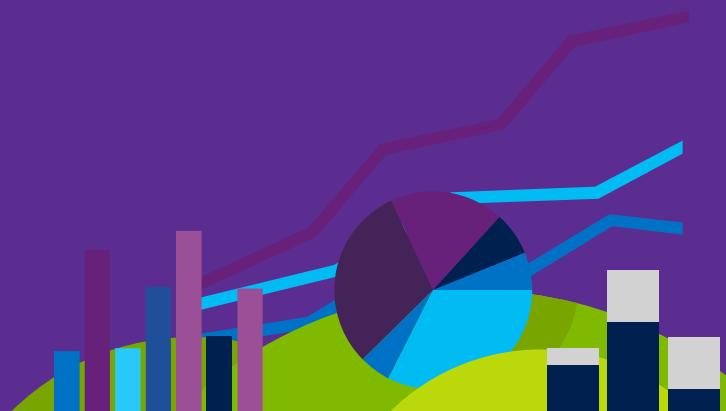
{"Resilience, Availability
and Disaster Recovery"}

Chris Marchal



{Scalability and Performance}

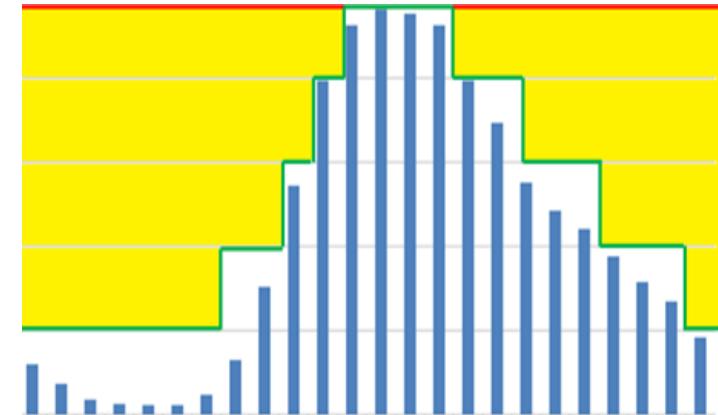
- *Good philosophy and strategies*
- *Some well tested patterns*
- *Azure services you can lean on*
- *Use the best tools at your disposal*
- *Some dos and don'ts*



Philosophy & strategies

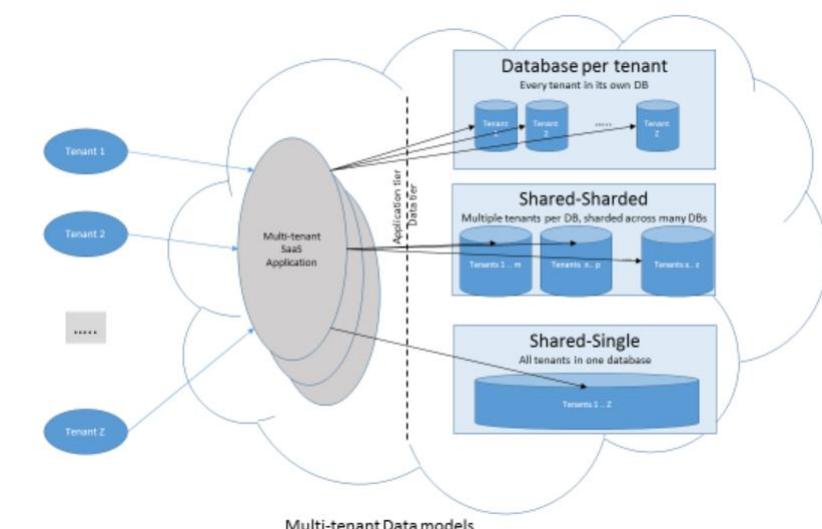
Scale out, not up

- Scale out is a “good place” to be
 - Jump from 1 to n can be major step change
 - No real upper limits
 - No reliance on bespoke high performance hardware
- Finer granularity = cheaper
- Use services to help scale workload
 - Use ones that spin up super-quickly, such as Azure App Services, Web jobs and Functions
- Manage state carefully
 - Stateless services are easier to scale out
 - Externalise state to a distributed cache (Redis) where necessary
- Good to think in terms of “Scale Units”
 - X amount of work can be done with a unit eg cars/hours on road
 - 1 x S1 IoT Hub handles 400,000 msg/day
 - 1 x Media Services streaming unit handles 200 Mbps



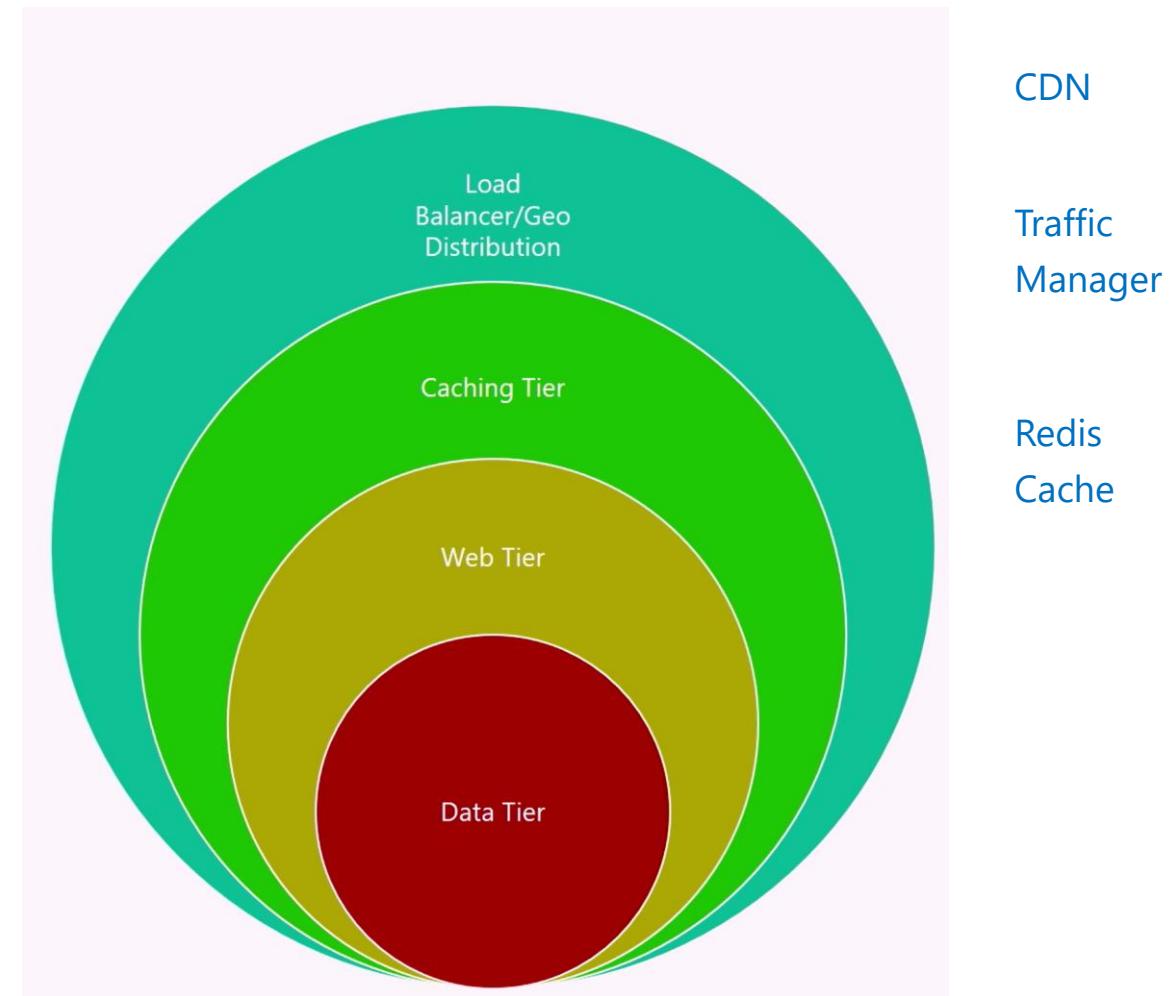
Partition your system

- Use decoupled architectural patterns
 - Microservice model
 - Queue Based Load Levelling, Competing Consumers, etc
- Shard / Partition data layer
 - Avoid one big database
 - Consider partition by tenant
 - e.g. SQL Database elastic pools
 - Use store with in-build partition options
 - e.g. Table Storage, Cosmos DB



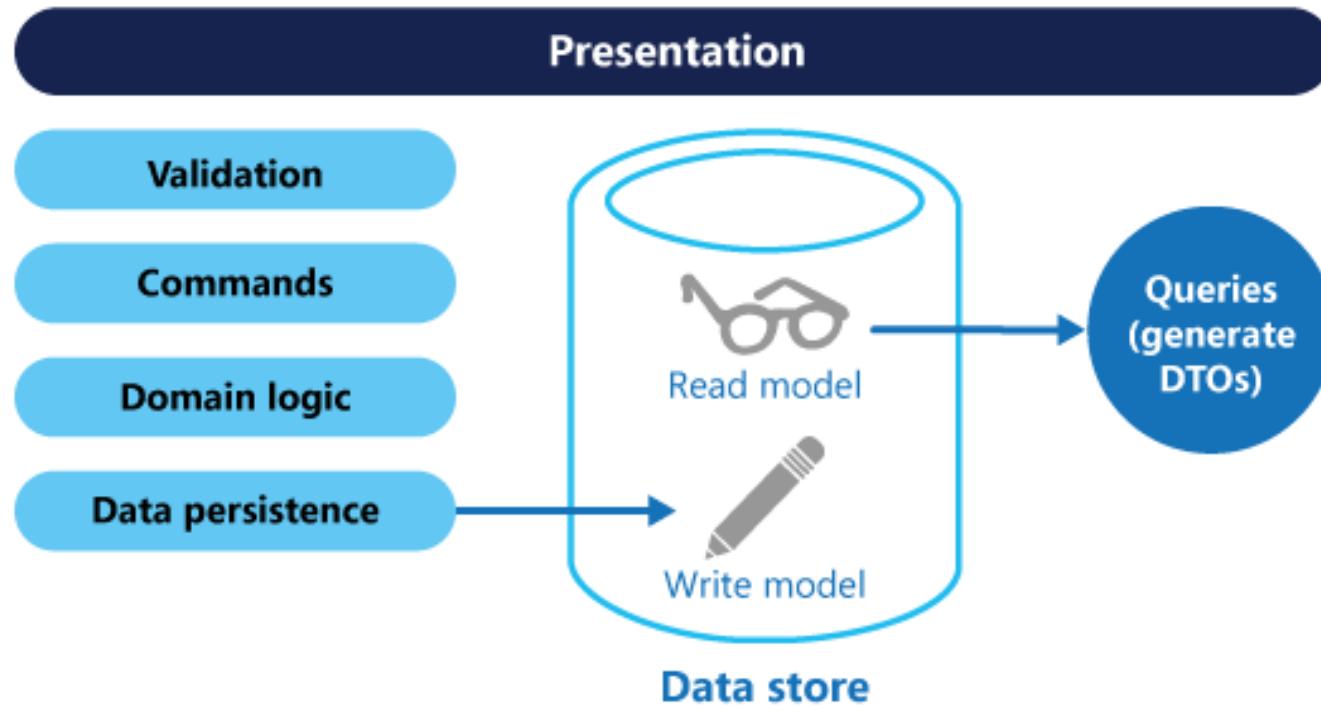
Scale in Tiers

- Each layer reduces call volume by an order of magnitude
- Typical model
 - Geo-distribution of system
 - CDN to place assets at network edge
 - Cache to reduce impact on web pipeline
- Sochi Olympics & Azure Media Services



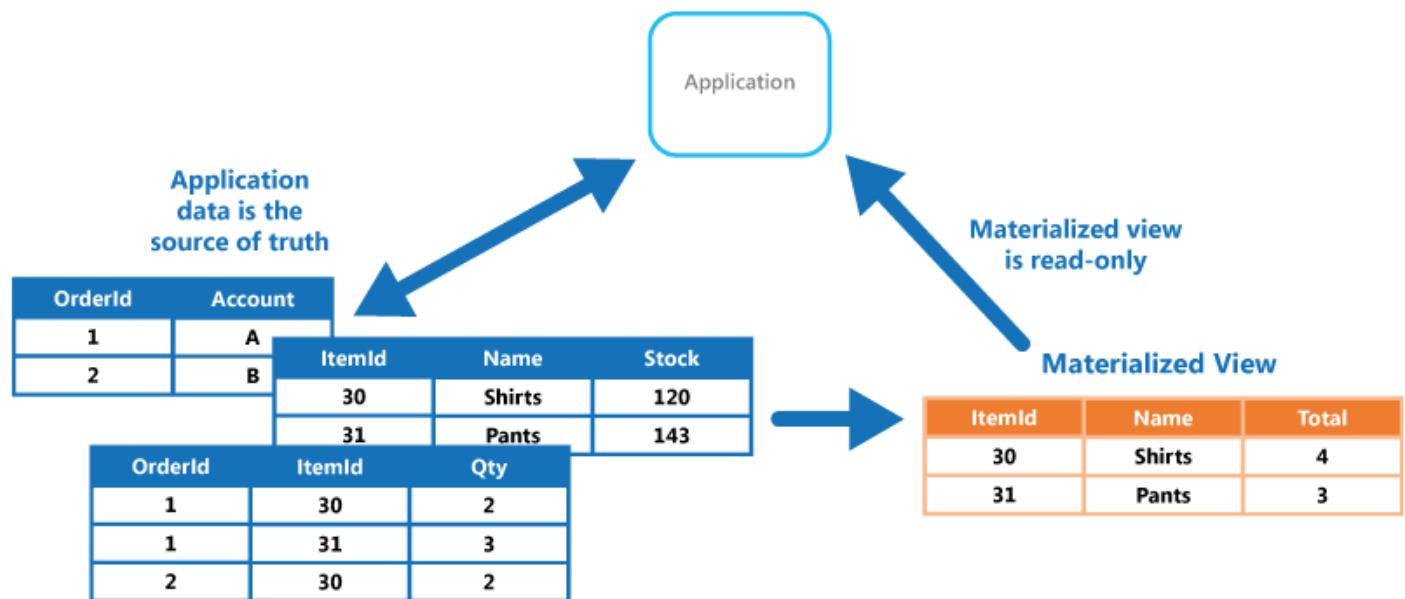
Patterns

Command and Query Responsibility Separation (CQRS) Pattern



- *"Separate reads from writes"*
- *Trade off of complexity for performance*
- *Avoid multi-master issues in geo-distributed systems*

Materialized View pattern



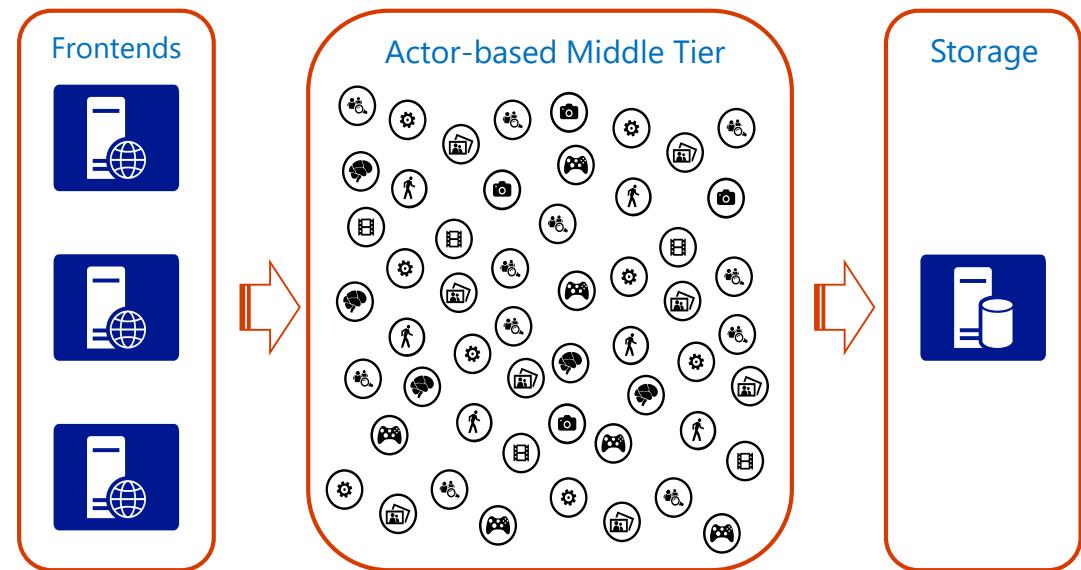
Data often optimised for storage rather than reads

Used when data isn't suitable format for querying, where generating suitable query is difficult, or query performance poor due to nature of the data or its store.

Often part of CQRS

Actor model

- Model for concurrent computation back in 1973
 - Erlang early implementation
- Reduce code complexity in complex systems
 - Actors always exist, virtually
 - Deadlock and re-entrant issues handled by system
 - Asynchronous communications between actors
- Improve performance by holding state in middle tier
 - Middle layer isn't just wrapper round database CRUD calls
- Project "Orleans"
 - Open source - <https://github.com/dotnet/orleans>
 - Developed by Microsoft Research, used in Xbox Halo 4 and 5
- Service Fabric
 - Implements Reliable Actors application framework



"Orleans: Distributed Virtual Actors for Programmability and Scalability"*

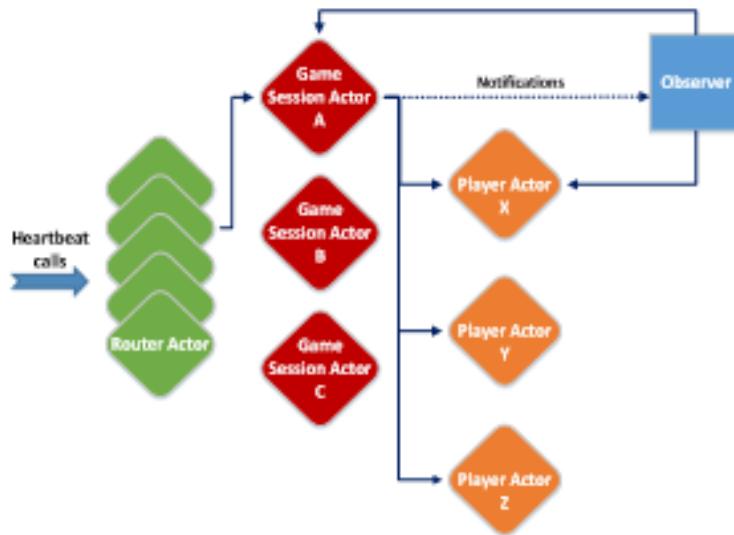


Figure 1: Halo 4 Presence Service

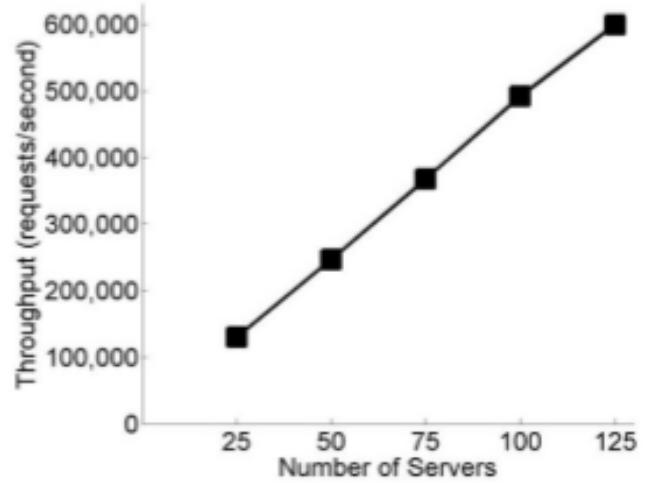


Figure 6: Throughput of Halo 4 Presence service.
Linear scalability as number of server increases.

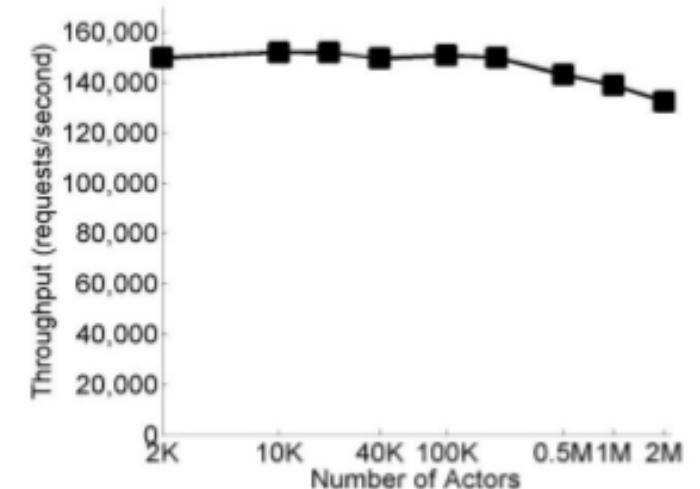
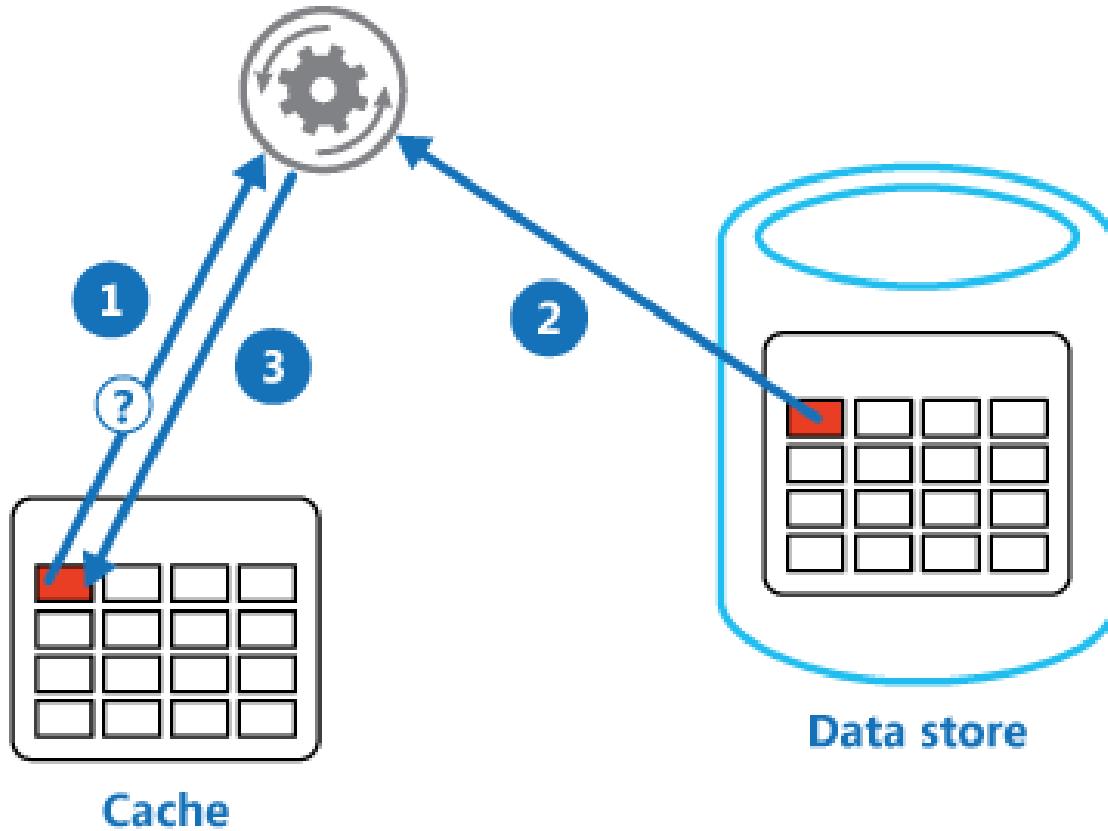


Figure 7: Throughput of Halo 4 Presence service.
Linear scalability as number of actors increases.

* <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/Orleans-MSR-TR-2014-41.pdf>

Cache-Aside Pattern



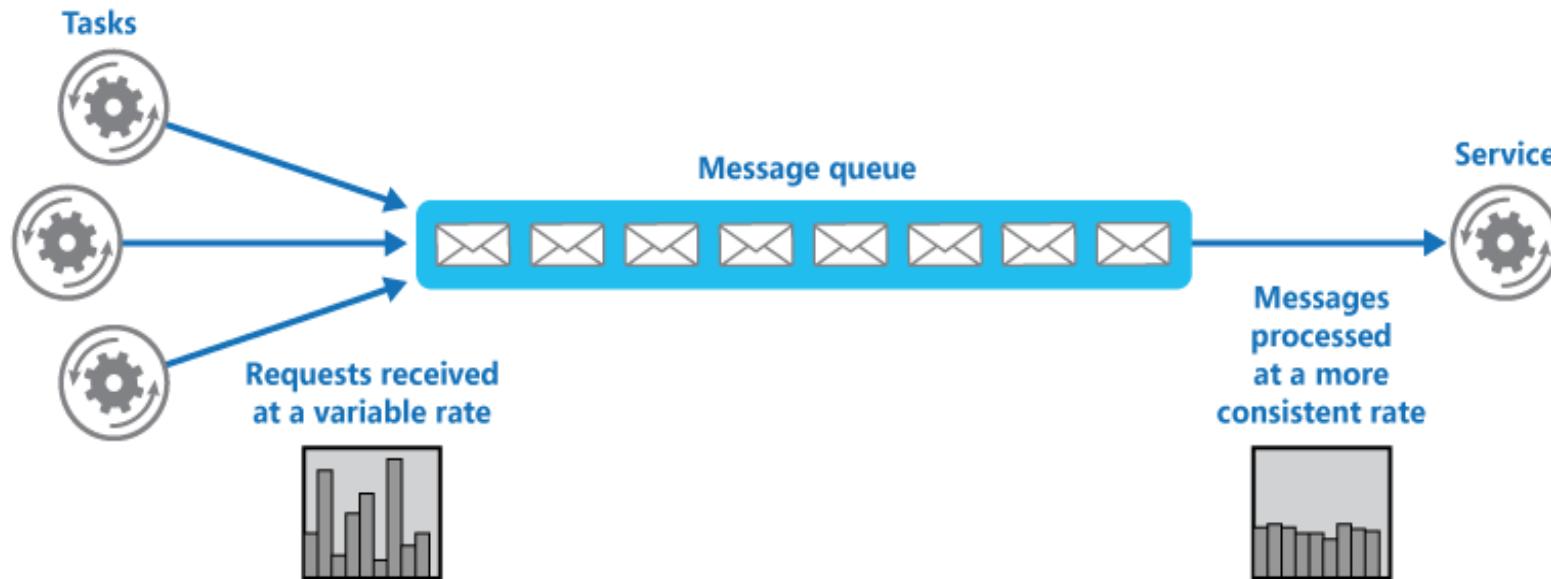
Load data on demand into a cache from a data store

- 1. Determine if item is in cache*
- 2. If not load it from the store*
- 3. Store item in cache*
- 4. Return item*

Issues and Considerations

- Lifetime of cached data*
- Evicting data e.g. LRU*
- Consistency*

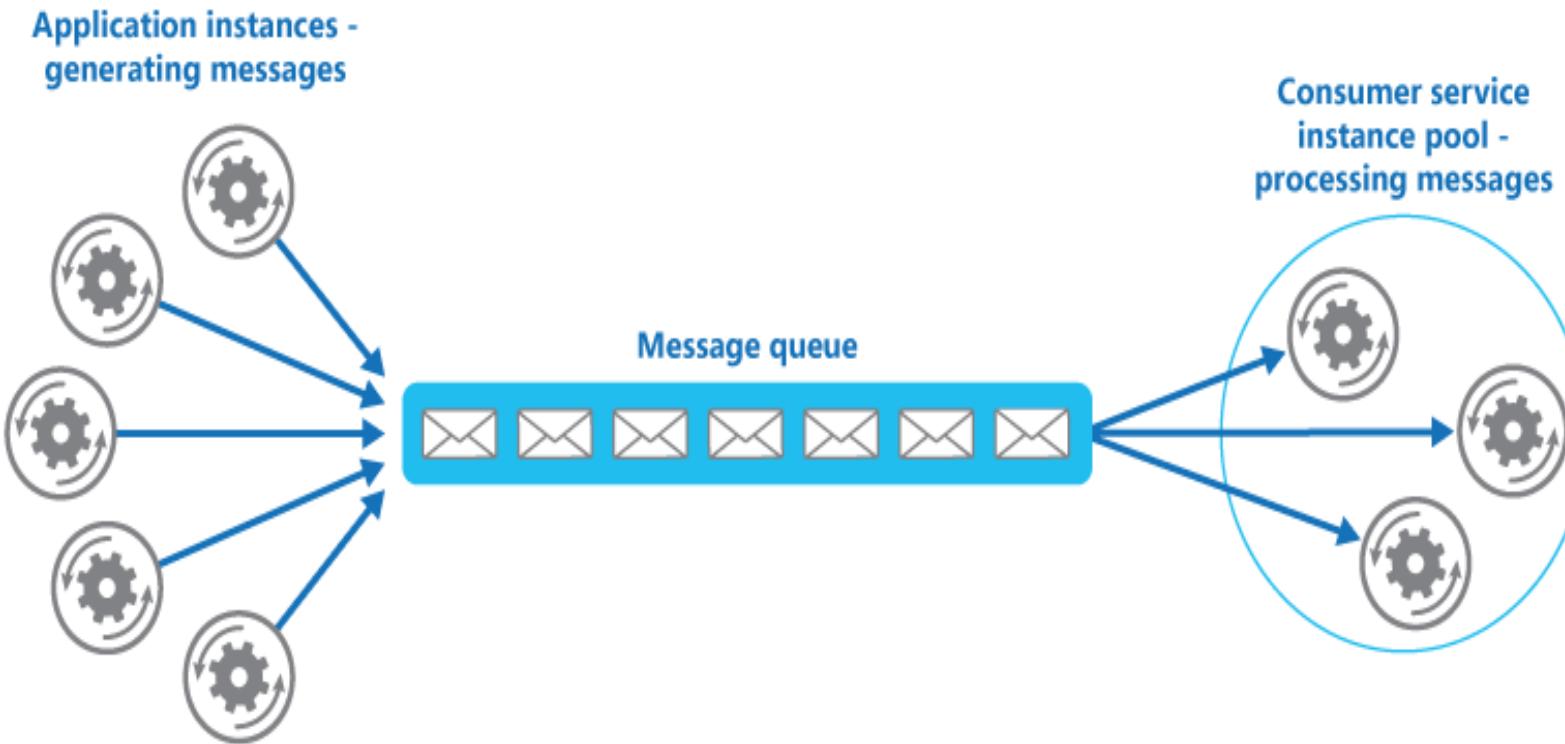
Queue Based Load Leveling Pattern



- *Queue that acts as a buffer between task and service*
- *Smooth intermittent heavy loads where flooding a service could cause failure or timeouts*

<https://aka.ms/queue-based-load-leveling-pattern>

Competing Consumers Pattern



Multiple concurrent consumers process messages from same messaging channel

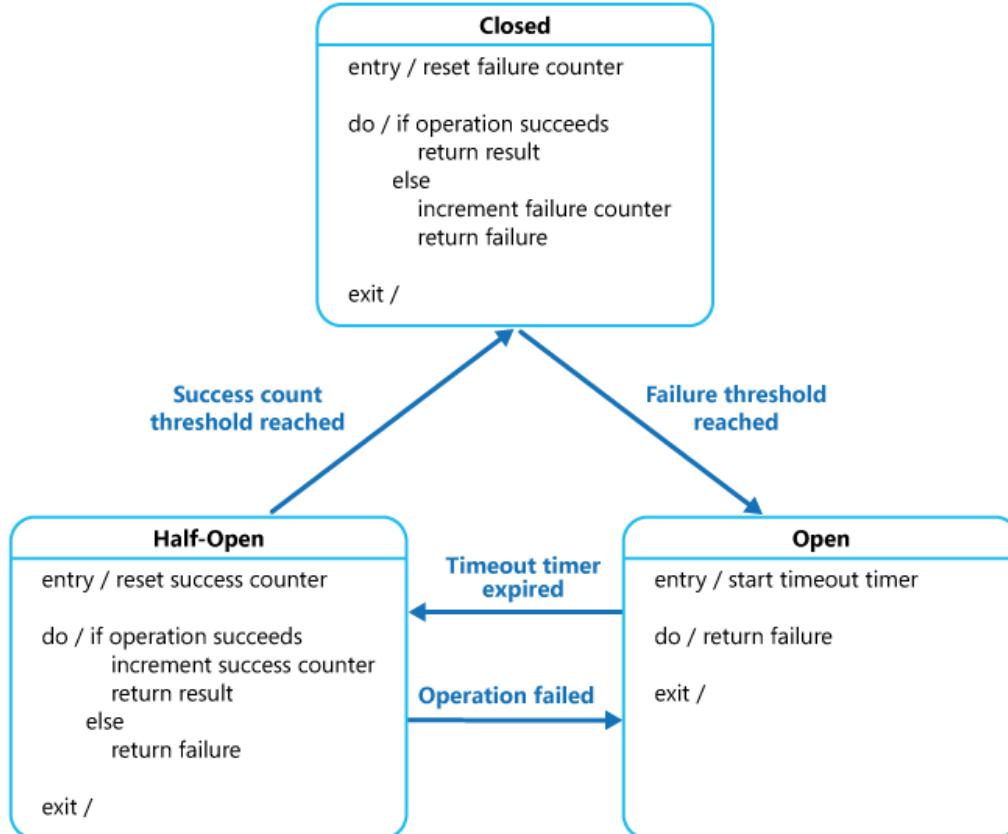
Optimize throughput, improve scalability

Issues and Considerations

- *Idempotency to handle messaging order*
- *Detecting poison messages*
- *Handling results back*



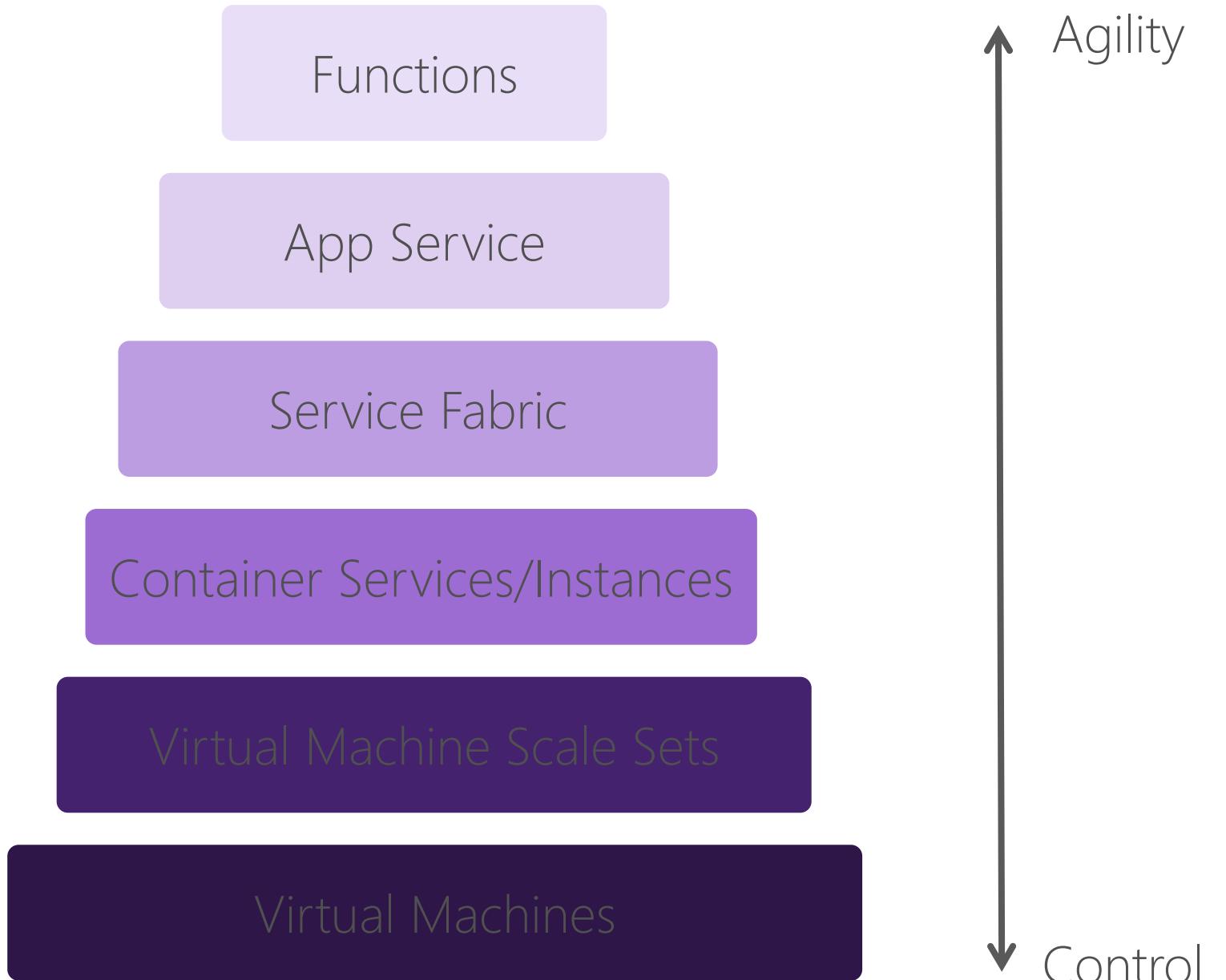
“Failover” and “Circuit Breaker”



- Use smart retry/back-off logic to mitigate the effect of Transient failures
 - Such as network issues, service throttling, etc
 - e.g. Transient Fault Handling Application Block
- When we know a system is down, behave accordingly and let it recover
 - Circuit Breaker: Closed/Half-Open/Open
- If we have a service that can be used instead for full functionality or degraded operations then use that instead until our primary service is back online.

Azure Services

Code



Choose the right VM sizes

General Purpose	Balanced CPU/memory: dev & test workloads, low / med traffic sites, small / med db	B (Preview), Dsv3, Dv3, DSv2, Dv2, DS, D, Av2, A0-7
Compute Optimised	High CPU/memory ratio : medium web, network, app servers	F, Fs
Memory optimised	High memory/core ratio: db servers, cache, in-memory analytics	Esv3, Ev3, M, GS, G, DSv2, DS, Dv2, D
Storage optimised	High disk throughput and IO: big data, sql, no-sql	Ls Series
GPU	Compute & graphics-intensive workloads	NV, NC
High Performance Compute	Fastest most powerful CPUs, RDMA	H Series, A8 - 11

VM Scale Sets

- Deploy and manage a set of identical VMs
- Control it like IaaS, scale it like PaaS
- Supports Managed Disks
 - 10,000 VM **disks** in a subscription
 - No 20,000 IOPS / account limit
 - Enhanced availability through availability sets
 - Easy migration to SSD premium storage
- Auto scale support
- Used by other Azure Services
 - Batch, Service Fabric, etc.

* Scale set virtual machine size >
1x Standard D1 v2

Disk Type
 Managed Unmanaged

Autoscale
 Enabled Disabled

Autoscale minimum number of VMs
1

Autoscale maximum number of VMs
10

Scale out CPU percentage threshold
75

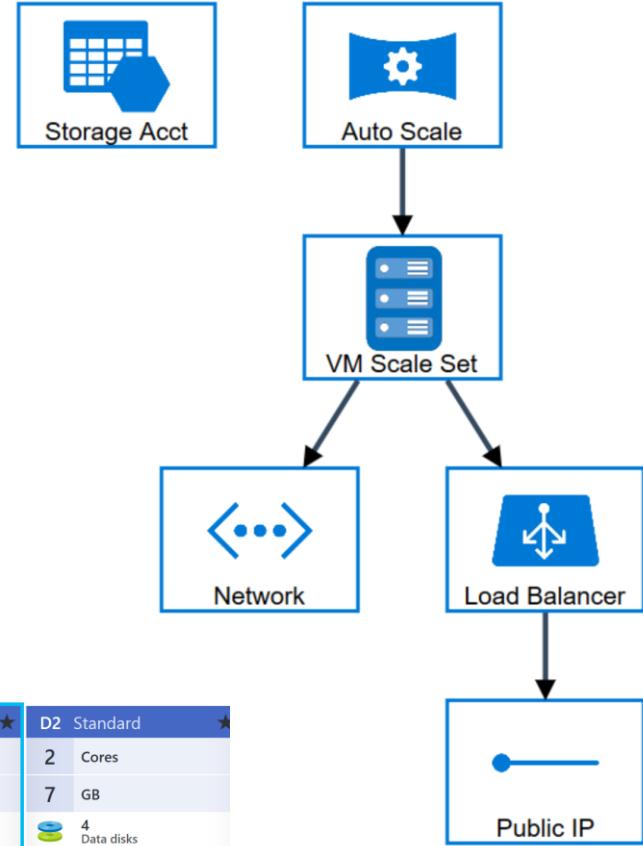
Number of VMs to increase by on scale out
1

Scale in CPU percentage threshold
25

Number of VMs to decrease by on scale in
1

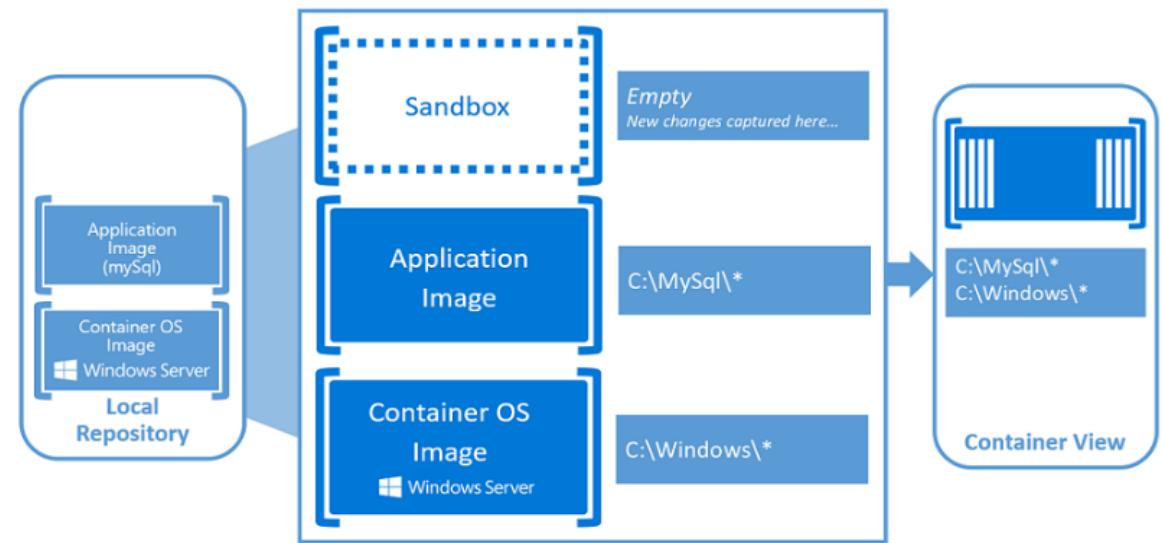
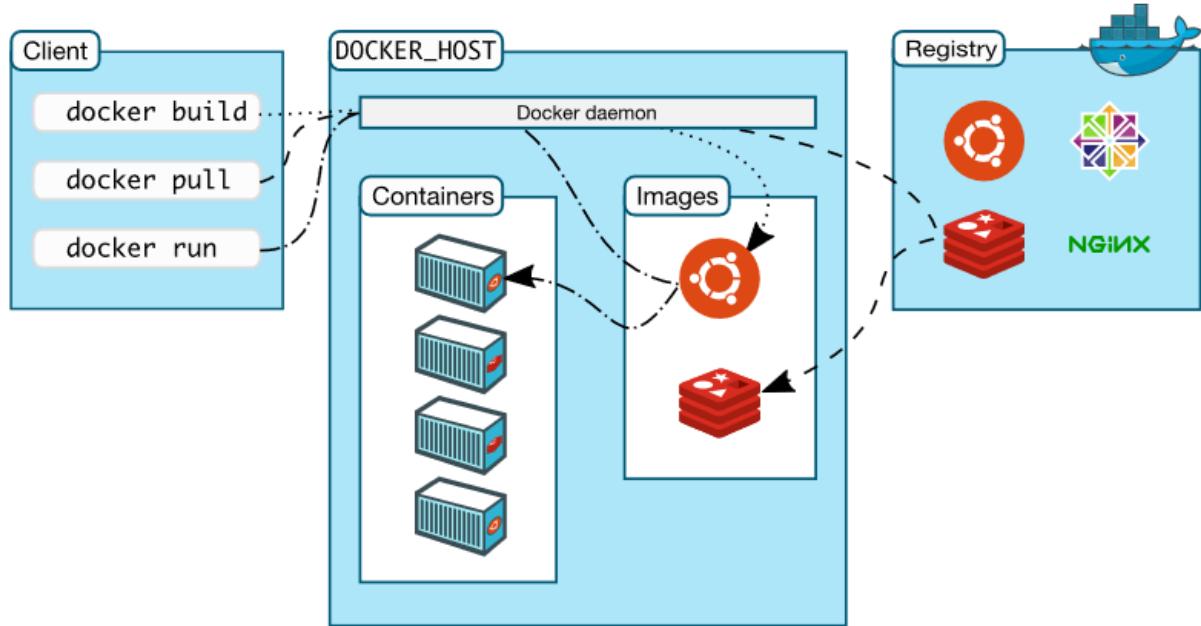
D1_V2 Standard	
1	Core
3.5	GB
	2 Data disks
	2x500 Max IOPS
	50 GB Local SSD
	Load balancing
67.65 GBP/MONTH (ESTIMATED)	

D2 Standard	
2	Cores
7	GB
	4 Data disks
	4x500 Max IOPS
	100 GB Local SSD
	Load balancing
143.06 GBP/MONTH (ESTIMATED)	



Containers

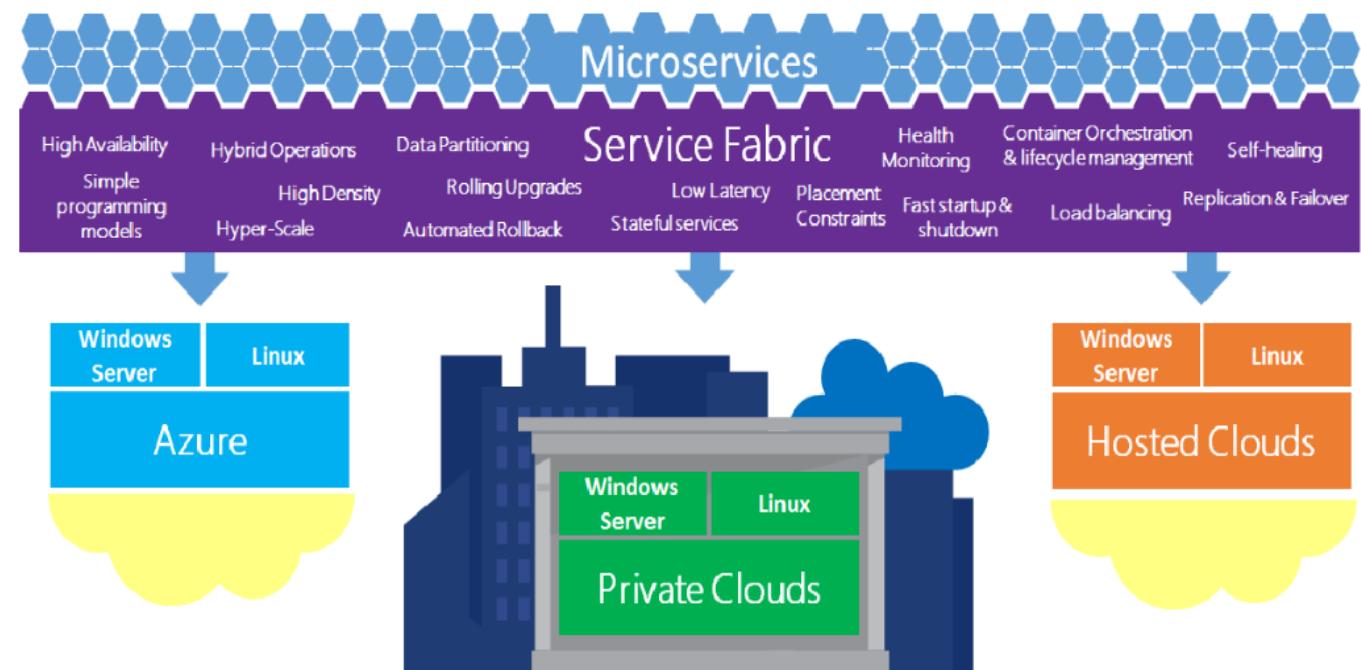
- New way of running applications in lightweight, isolated units
- Apps think they are running on a normal server, but it's sharing the operating system of the host
- Docker has defined the platform technology
- Great for underpinning digital transformations, such as modernizing traditional applications or new microservice architectures
- Originally Linux, now part of Windows Server 2016
- Azure Container Instances in preview





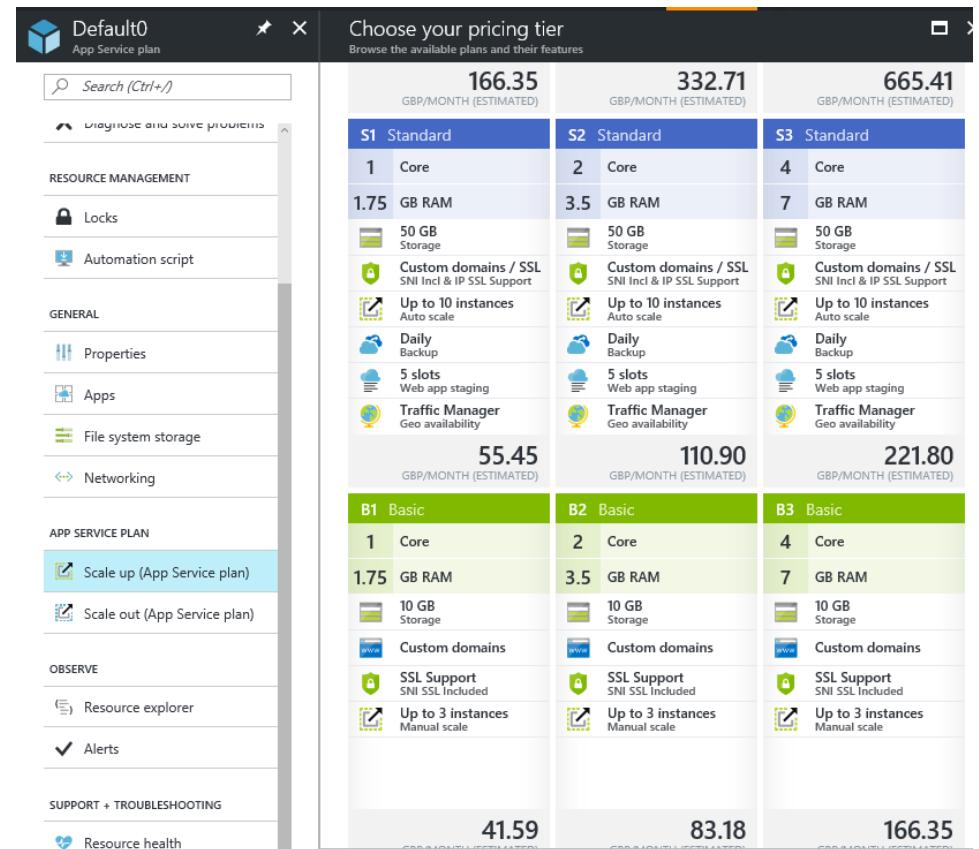
Azure Service Fabric

- Powers many Microsoft services
 - SQL Database, Azure DB, Cortana, Power BI, Intune, IoT Hub,
- Runs Windows and Linux, on-premise and cloud and dev
- Scale from a few to thousands of servers
- Stateless and stateful programming models
- Comprehensive runtime and lifecycle management capabilities
- Container deployment and orchestration



Azure App Services

- Run web apps and web API
- Supports global scale with high availability
- App Service Plans represent physical resources available to your apps
 - Instance count
 - Instance size (Small, Medium, Large)
 - Tier (Free, Shared, Basic, Standard*, Premium*, Isolated*) * support autoscale

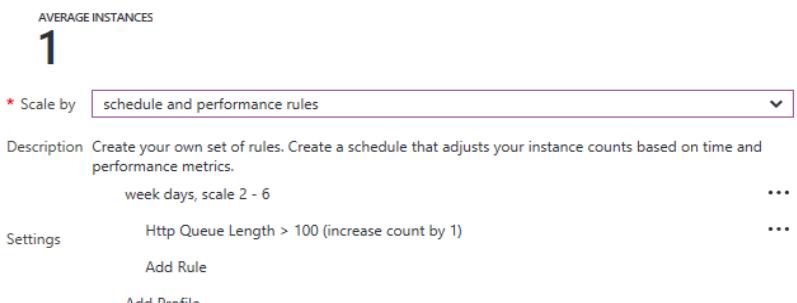
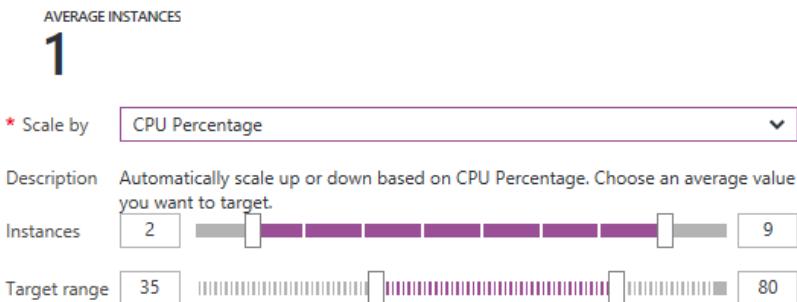
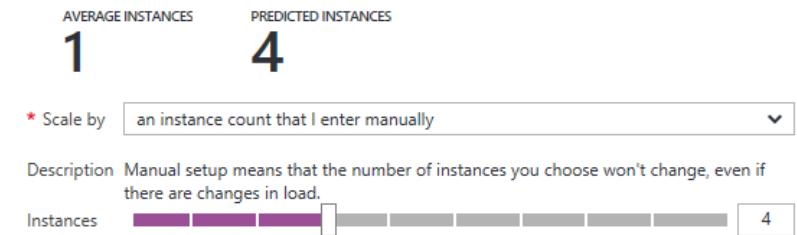
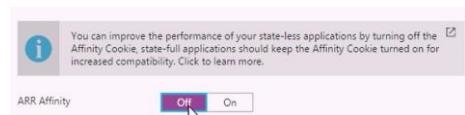


The screenshot shows the Azure portal interface for managing an App Service plan named 'Default0'. On the left, there's a sidebar with various management options like 'Diagnose and solve problems', 'Resource management', 'General', 'App Service plan', 'Observe', and 'Support + Troubleshooting'. The main area is titled 'Choose your pricing tier' and displays a grid of pricing tiers:

Choose your pricing tier			
Browse the available plans and their features			
	166.35 GBP/MONTH (ESTIMATED)	332.71 GBP/MONTH (ESTIMATED)	665.41 GBP/MONTH (ESTIMATED)
S1 Standard	S2 Standard	S3 Standard	
1 Core	2 Core	4 Core	
1.75 GB RAM	3.5 GB RAM	7 GB RAM	
50 GB Storage	50 GB Storage	50 GB Storage	
Custom domains / SSL SNI Incl & IP SSL Support	Custom domains / SSL SNI Incl & IP SSL Support	Custom domains / SSL SNI Incl & IP SSL Support	
Up to 10 instances Auto scale	Up to 10 instances Auto scale	Up to 10 instances Auto scale	
Daily Backup	Daily Backup	Daily Backup	
5 slots Web app staging	5 slots Web app staging	5 slots Web app staging	
Traffic Manager Geo availability	Traffic Manager Geo availability	Traffic Manager Geo availability	
55.45 GBP/MONTH (ESTIMATED)	110.90 GBP/MONTH (ESTIMATED)	221.80 GBP/MONTH (ESTIMATED)	
B1 Basic	B2 Basic	B3 Basic	
1 Core	2 Core	4 Core	
1.75 GB RAM	3.5 GB RAM	7 GB RAM	
10 GB Storage	10 GB Storage	10 GB Storage	
Custom domains	Custom domains	Custom domains	
SSL Support SNI SSL Included	SSL Support SNI SSL Included	SSL Support SNI SSL Included	
Up to 3 instances Manual scale	Up to 3 instances Manual scale	Up to 3 instances Manual scale	
41.59	83.18	166.35	

App Services Specific optimisations

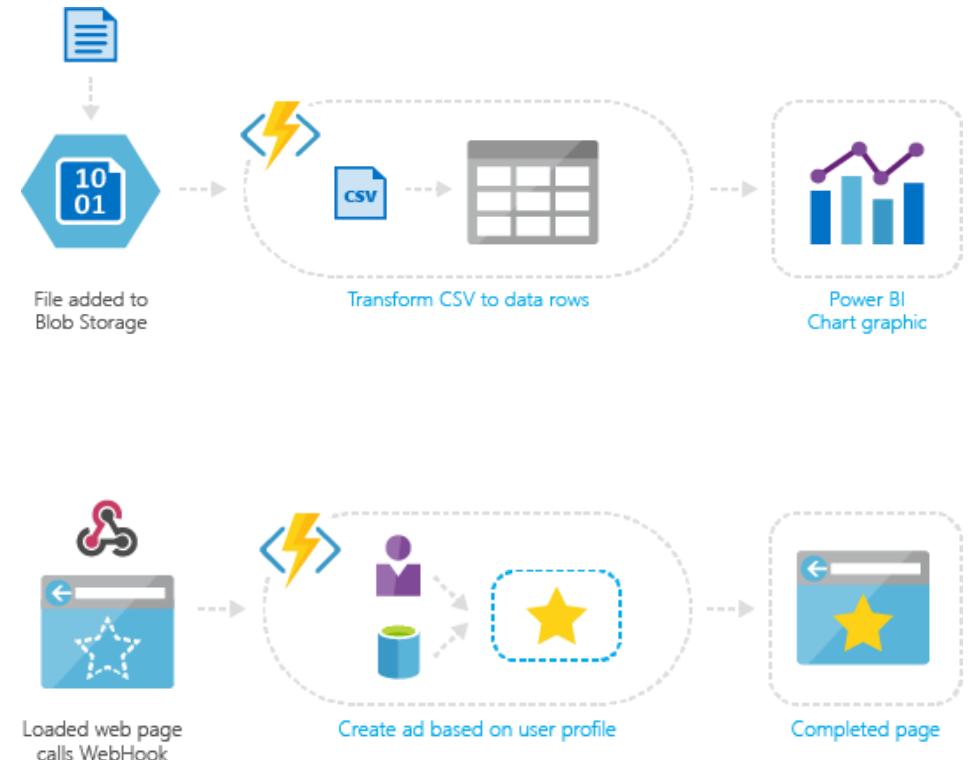
- Enable Auto-Scaling in App Service plan
 - Manually, CPU, schedule + rules
 - Prefer time based scale over auto-scale where possible
- Tips:
 - Enable AlwaysOn
 - Disable Session Affinity Cookie
 - [//azure.microsoft.com/en-us/blog/disabling-arrs-instance-affinity-in-windows-azure-web-sites/](http://azure.microsoft.com/en-us/blog/disabling-arrs-instance-affinity-in-windows-azure-web-sites/)



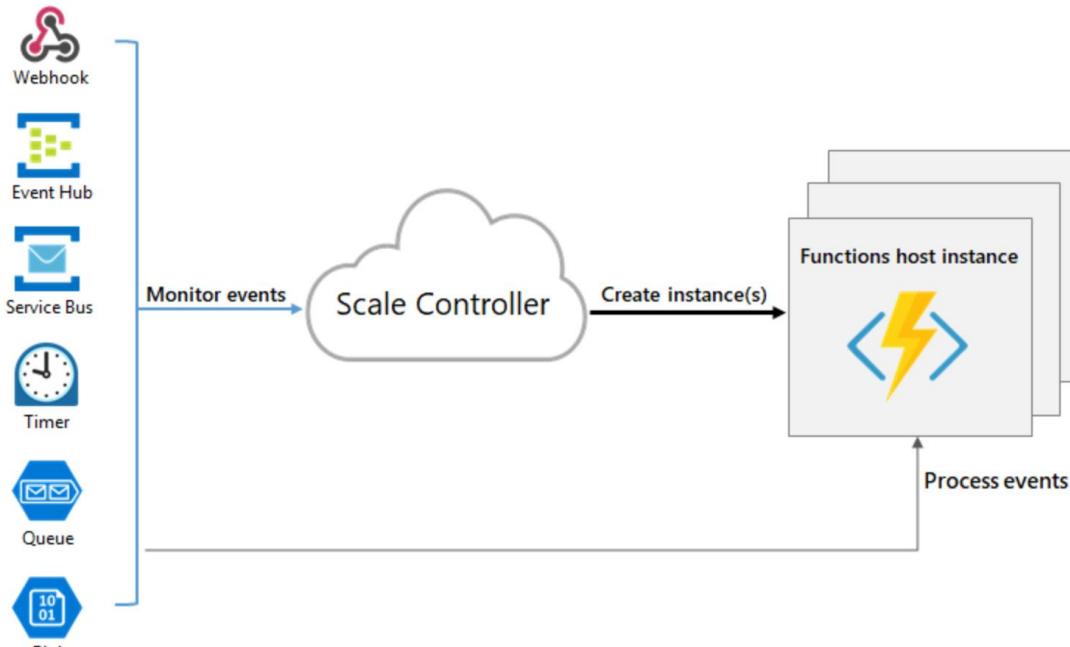
Azure Functions



- Serverless computing
 - With serverless consumption plan
- Built on App Service WebJobs
- Build “nano-services” as individual functions
- Event driven model
 - Timer, http hook, queues, storage, etc
 - Easy way to implement load levelling
 - No upper limit on compute
- Easy to test, tune and deploy



How Functions Work



run.csx Save ▶ Run </> Get function

```
1 using System.Net;
2
3 public static async Task<HttpResponseMessage> Run(HttpRequestMessage req, TraceWriter
4 {
5     // The sentiment category defaults to 'GREEN'.
6     string category = "GREEN";
7
8     // Get the sentiment score from the request body.
9     double score = await req.Content.ReadAsAsync<double>();
10    log.Info(string.Format("The sentiment score received is '{0}'.",
11                           score.ToString()));
12
13    // Set the category based on the sentiment score.
14    if (score < .3)
15    {
16        category = "RED";
17    }
18    else if (score < .6)
19    {
20        category = "YELLOW";
21    }
22    return req.CreateResponse(HttpStatusCode.OK, category);
```

Data

“human data”



Transactional integrity,
operational
information, etc.

“machine data”



Independent,
telemetry, insights, etc.

Polyglot Persistence

Different databases are designed to solve different problems. Using a single database engine for all of the requirements usually leads to non-optimal solutions

e.g.:

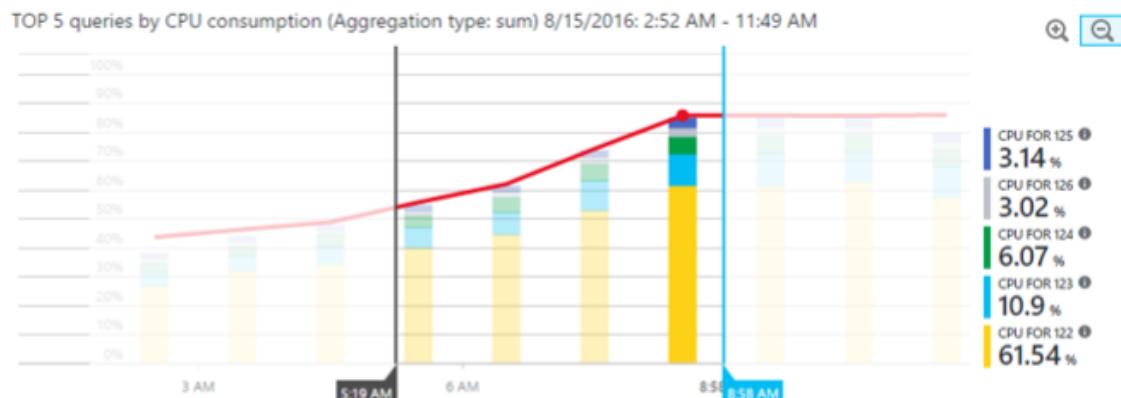
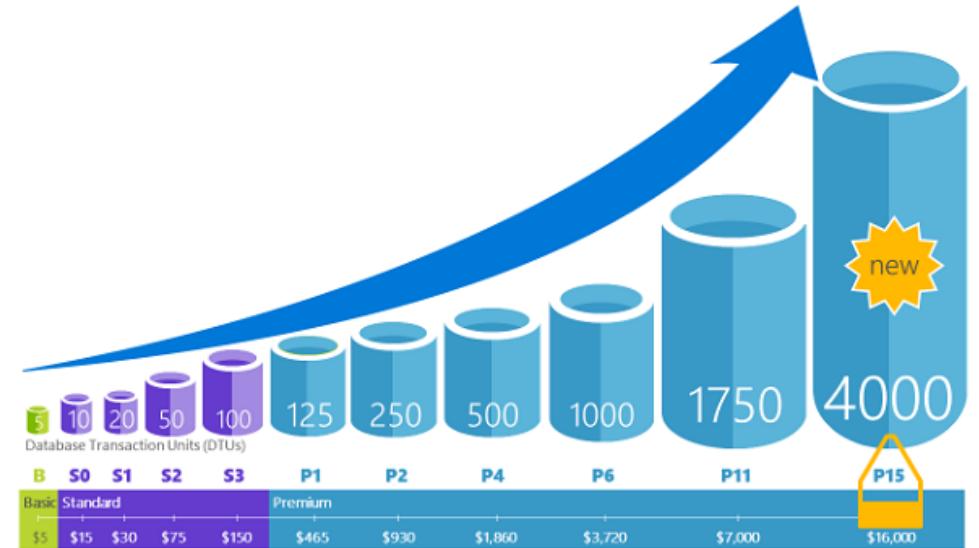
- User session
- Catalogue data
- Product search
- Shopping cart
- Orders database
- Analytics
- Reporting,





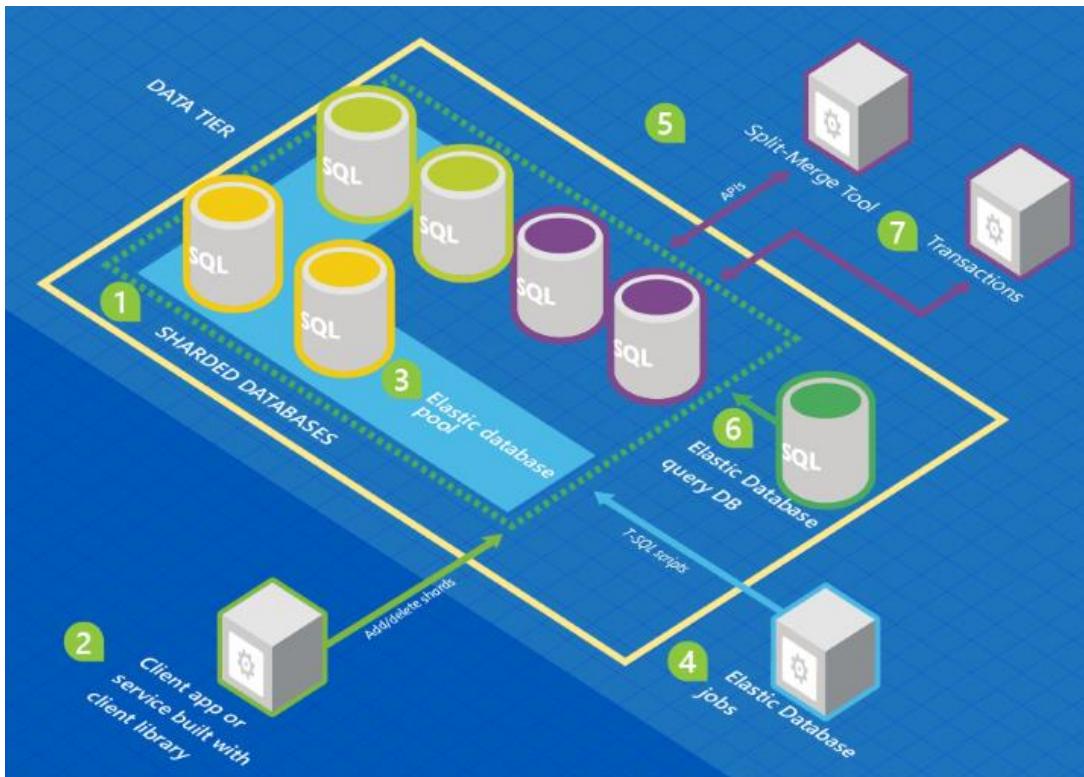
Azure SQL Database

- Scale performance on the fly
 - Within tier
 - Scale up for peak workloads, such as imports, heavy database access, etc
- Advisor
 - recommendations on indexes, schema issues,...
- Query Performance Insight
 - Deeper insight into DTU usage and top queries



Basic: 2 GB max size, 30 sessions
Standard: 250 GB max size, 200 sessions
Premium: 1 TB max size, 400 sessions

Scaling out SQL Database with Elastic Tools

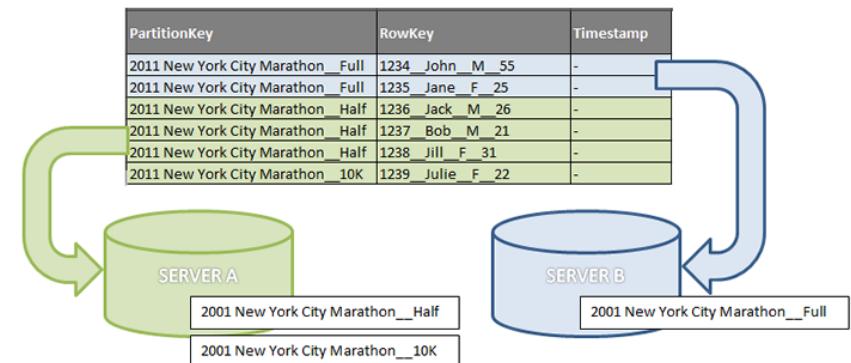


Databases with the same colour share the same schema

1. **Azure SQL databases** are hosted on Azure using sharding architecture.
2. The **Elastic Database client library** is used to manage a shard set.
3. A policy is applied to a subset of the databases through an **elastic pool** to manage variable demands cost effectively
4. An **Elastic Database job** runs scheduled or ad-hoc T-SQL scripts against all databases.
5. The **split-merge tool** is used to move data from one shard to another.
6. The **Elastic Database query** allows you to write a query that spans all databases in the shard set.
7. **Elastic transactions** allows you to run transactions that span several databases

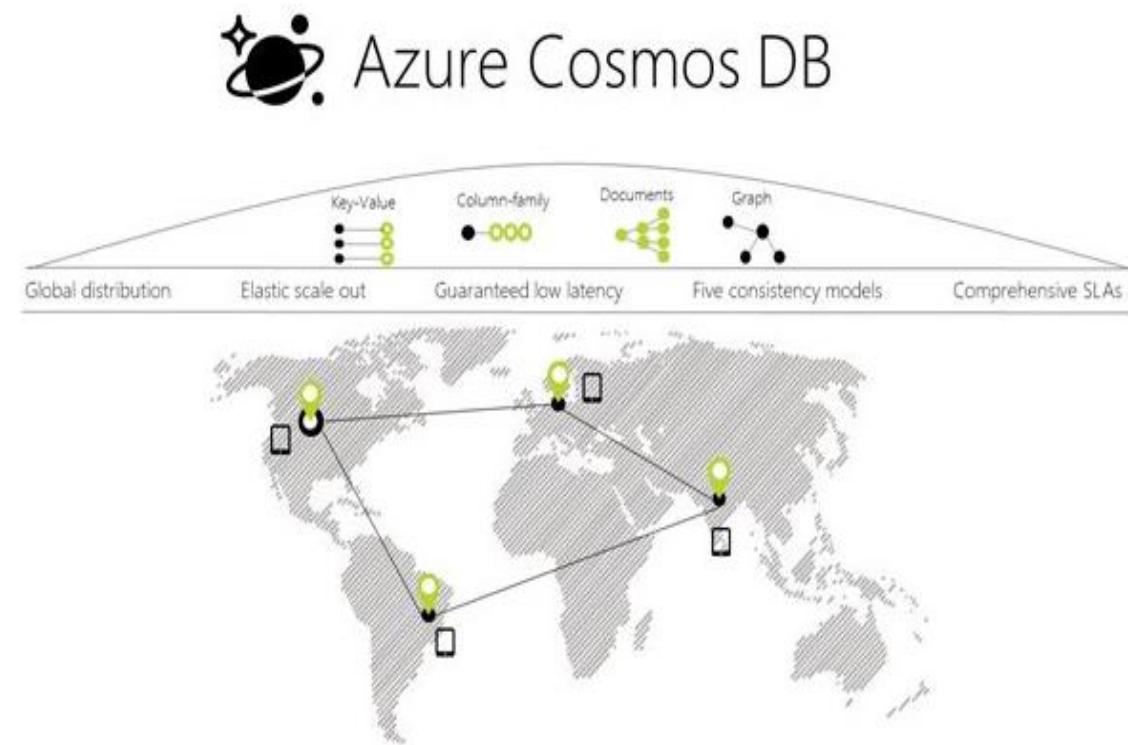
Azure Table Storage - Scalable Partitioning

- Table storage is fast and low cost NoSQL store
 - 1 MB max size of entity with a max 255 properties
- Consider separate storage accounts to overcome any service limits
 - 500 TB per account, 20,000 transactions per second per account
 - 200 accounts per subscription
- PartitionKey and RowKey combine to create single clustered index
 - Think about retrieval scenarios
- Partitions are always served from one server
 - Avoid scans that go across multiple partitions



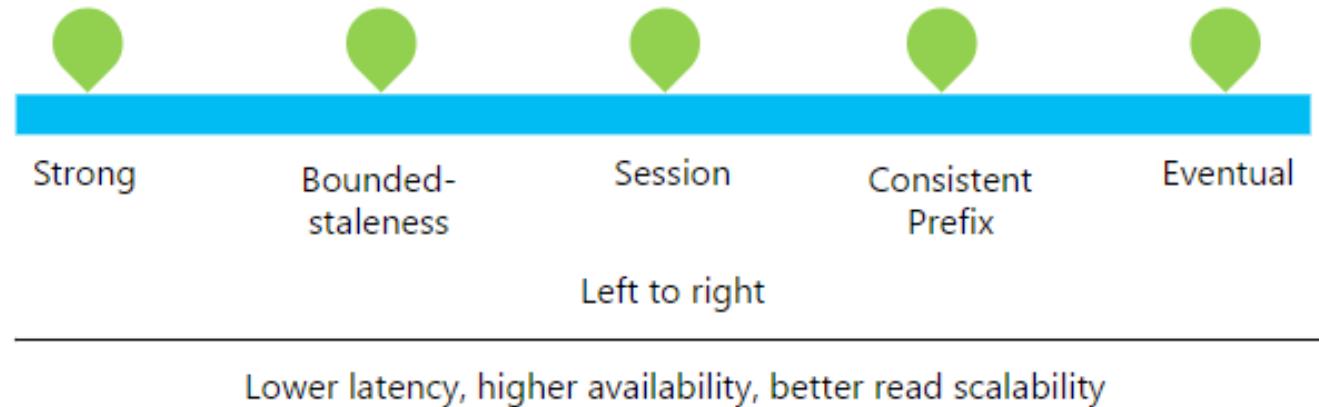
Azure Cosmos DB

- Replicate and scale data globally
 - Place data near users, seamlessly
- Multi-model + multi-API
 - Key-value, graph, and document data
 - SQL, JavaScript, MongoDB, Table storage, Gremlin Graph
- Guaranteed low latency at the 99th percentile
 - < 10 ms on reads, < 15 ms on writes
- Multiple, well-defined consistency choices
 - **strong**, bounded staleness, consistent-prefix, session, and **eventual**



Consistency and Performance

- Five consistency levels enable well-reasoned trade-offs between consistency, availability, and latency
 - Based on formal specification language TLA+ developed Turning award recipient by Leslie Lamport



- CAP theorem (aka Brewer's) states It is impossible for a *distributed* system to simultaneously provide more than two out of three of the following guarantees:
 - Consistency - Every read receives the most recent write or an error
 - Availability - Every request receives a response, without guarantee that it contains the most recent write
 - Partition tolerance - System continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes
- "strong" is fully CP and "Eventual" is fully AP

Events and Messages



Messages and Events

Messages

- Typically carry information needed for a step in a defined workflow
- May express inherent monetary value or commands to perform actions
 - Consider [Azure Service Bus](#) or [Azure Queues](#)

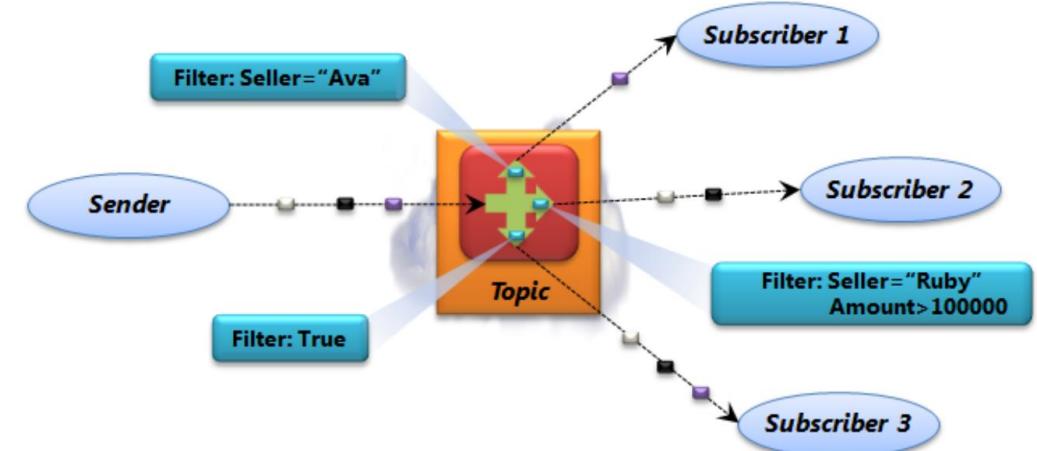
Events

- Don't generally convey publisher intent, other than to inform
 1. "Business logic activity" carried out by publishing application
 - Something has happened in system X that may be of interest elsewhere
 - Consider [Azure Event Grid](#)
 2. Informational data points from continuously published stream: IoT, etc
 - Logic often related to changes in pattern (such as sensor temperature rising) rather than individual data points
 - "Complex Event Processing" model
 - Consider [Azure Event Hubs / IoT Hubs](#)

Azure Service Bus

- Reliable information delivery service
 - Duplicate detection, time-based expiration, batching, etc
 - Separate Send and Listen access
- Brokered messaging support:
 - Queues – single consumer
 - Topics & subscriptions – multiple consumers
- Premium tier addresses common requests around scale, performance and availability
 - Dedicated resources in form of messaging unit

Premium	Standard
High throughput	Variable throughput
Predictable performance	Variable latency
Fixed pricing	Pay as you go variable pricing
Ability to scale workload up and down	N/A
Message size up to 1 MB	Message size up to 256 KB

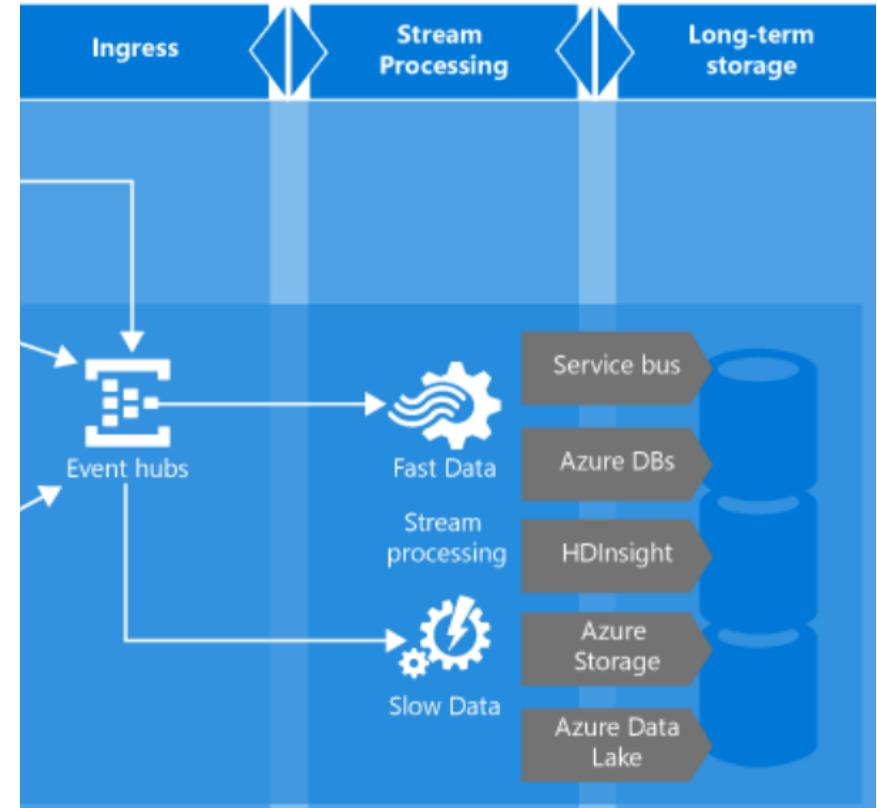


Best Practices for performance improvements:

- Use AMQP and SBMP over HTTP
- Use Asynchronous operations on queues
- Client-side batching
- Use partitioned queues or topics
- docs.microsoft.com/azure/service-bus-messaging/service-bus-performance-improvements

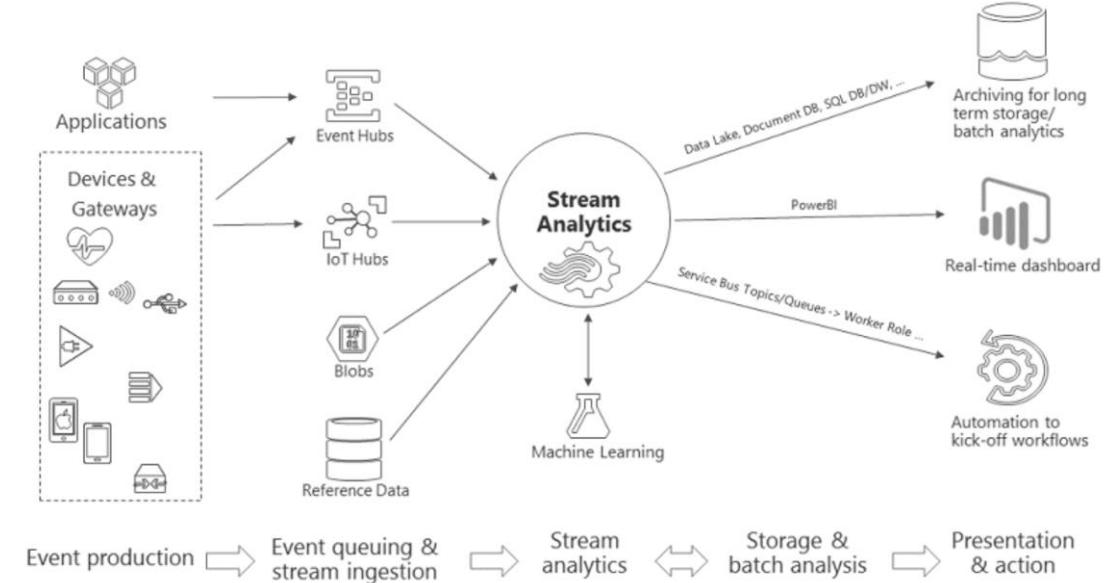
Azure Event Hubs

- “Event Ingestor”
 - Accepts and stores event data
 - Makes data available for fast “pull” retrieval
- Log millions of events per second in near real time
 - Low latency, elastic scale
 - Configurable time retention, read data using publish-subscribe semantics
 - Partition is used to store ordered sequence of events
 - Multiple subscribers via consumer group
- Scale through Throughput Units (TUs)
 - Single TU entitles you to 1MB/second or 1000 events/second ingress and 2MB/second or 2000 events/second egress
 - Auto-scale up

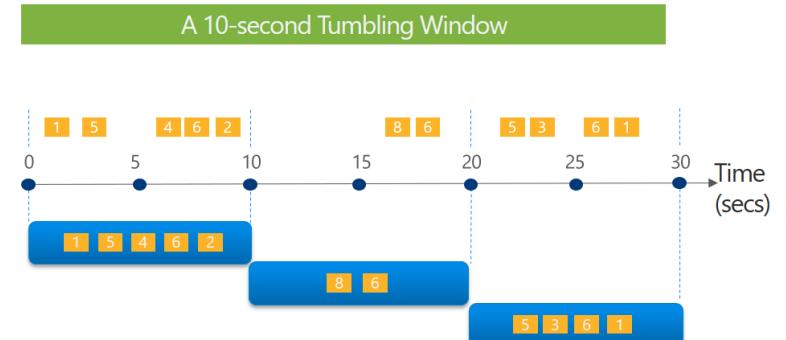


Stream Analytics

- Massively parallel Complex Event Processing pipeline
 - Integration into source and destination
 - Support for partition in event hubs, blob, etc
- Declarative SQL like Stream Analytics query language
- Streaming units (SUs) represent resources & computing to execute an Stream Analytics job
 - Blended measure of CPU, memory and read/write rates
 - Corresponds to around 1 MB/sec throughput



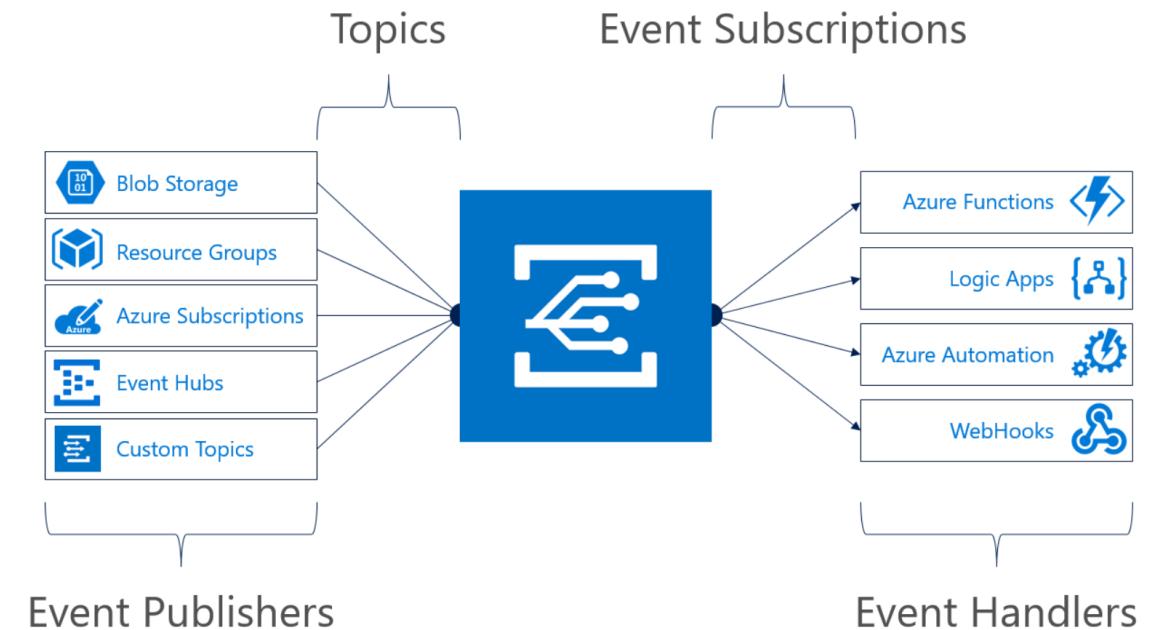
Tell me the count of tweets per time zone every 10 seconds



```
SELECT TimeZone, COUNT(*) AS Count  
FROM TwitterStream TIMESTAMP BY CreatedAt  
GROUP BY TimeZone, TumblingWindow(second,10)
```

Azure Event Grid (Preview)

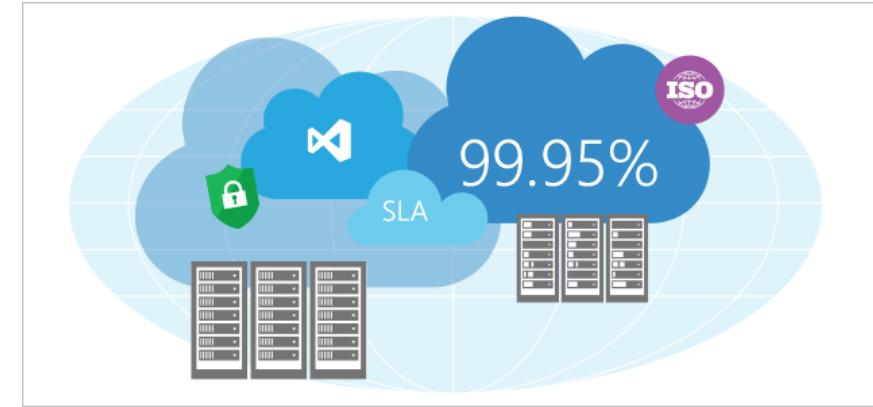
- Route events from any source to any destination
 - Scale dynamically, near-real-time event delivery using a publish-subscribe model
- Scenarios
 - Serverless architecture
 - application integration
 - Ops automation



Tools

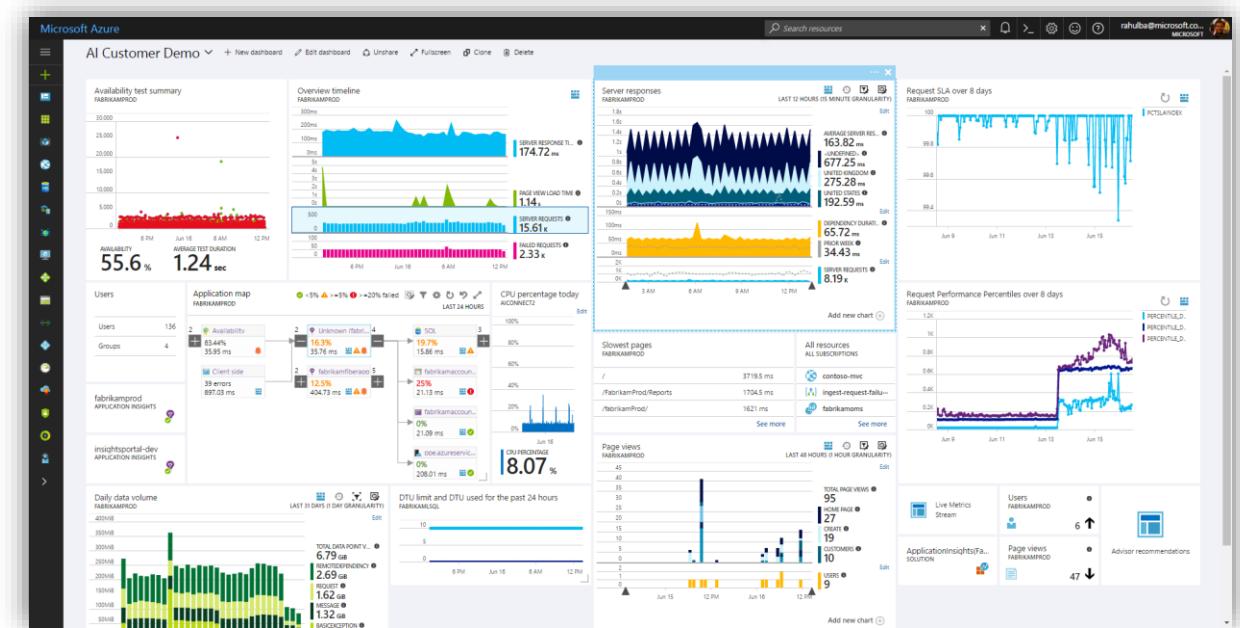
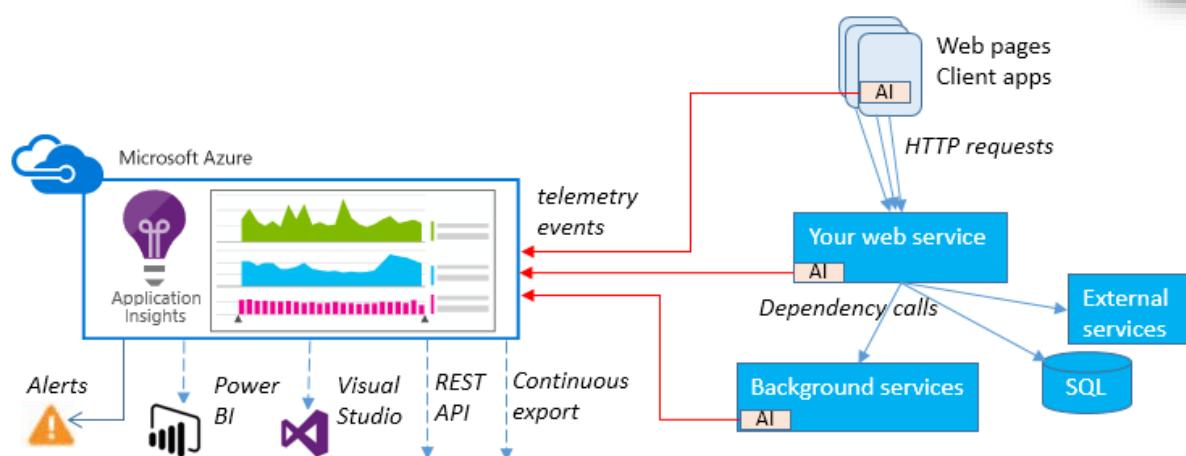
Load test your application before deployment

- At the absolute least, ensure that some kind of automated load testing is performed to detect a noticeable change in performance of specific, critical endpoints
- Ideally set actual goals
 - Expected req/sec, latency needed, concurrency
 - Explore real vs perceived latency
- Test the application at significant load to ensure that spikes in user activity and load will not take down the application at burst times.
- Test your application automatically **before** deployment using CI / CD Practices
 - The earlier you find these problems, cheaper they are to fix



Azure Application Insights

- Powerful analytics tool
- Instrumentation package in your application monitors your app and sends telemetry data to the portal
 - Impact on app's performance is very small



- Diagnostic search for instance data
- Metrics Explorer for aggregated data
- Dashboards
- Live Metrics Stream
- Analytics with query language
- Automatic and manual alerts
- Power BI integration

Real Time Health Check

Live Metrics Stream
fabrikamprod

Incoming Requests

Requests/Sec

Request Duration (ms)

Requests Failed/Sec

Outgoing Requests

Dependency Calls/Sec

Dependency Call Duration (ms)

Dependencies Failed/Sec

Overall Health

Committed Memory (MB)

CPU Total (%)

Exceptions/Sec

Servers

SERVER NAME	REQUESTS	REQUESTS FAILED	CPU TOTAL	COMMITTED MEMORY
AIConnect2	0/sec	0/sec	7%	1708 MB
RD00155DA9A9C0	0/sec	0/sec	0%	125 MB

Sample Telemetry

Time	Message	User ID
1:50:09 PM	Dependency 409 51 ms	@RD00155DA9A9C0
1:50:09 PM	PUT fabrikamaccount Container=fabrikamfiber	
1:50:06 PM	Trace New Request Received	@RD00155DA9A9C0
1:50:06 PM	Request POST Customers/Create	@AIConnect2
1:50:06 PM	Exception DbUpdateException	@AIConnect2
1:50:06 PM	An error occurred while updating the entries. The statement has been terminated.	
1:50:06 PM	Dependency 547 16 ms	@AIConnect2
1:50:06 PM	tcp:fabrikamxyz.database.windows.net:1433 FabrikamSQL ErrorMessage=The INSERT statement	
1:50:06 PM	Trace Starting to save New Customer	@AIConnect2
1:50:06 PM	New Request Received	@AIConnect2
1:50:02 PM	Request POST Customers/Create	@AIConnect2
1:50:02 PM	Exception DbUpdateException	@AIConnect2
1:50:02 PM	An error occurred while updating the entries. The statement has been terminated.	
1:50:02 PM	Request GET Customers/Details	@RD00155DA9A9C0
1:50:02 PM	Exception FormatException	@RD00155DA9A9C0
1:50:02 PM	Input string was not in a correct format.	
1:50:01 PM	Dependency 409 134 ms	@RD00155DA9A9C0
1:50:01 PM	POST fabrikamaccount/fabrikamfiber	
1:50:01 PM	Dependency 547 18 ms	@AIConnect2
1:50:01 PM	tcp:fabrikamxyz.database.windows.net:1433 FabrikamSQL ErrorMessage=The INSERT statement	
1:50:01 PM	Trace New Request Received	@AIConnect2
1:50:00 PM	Dependency 409 99 ms	@RD00155DA9A9C0
1:50:00 PM	POST fabrikamaccount/Tables	
1:50:00 PM	Trace New Request Received	@RD00155DA9A9C0
1:50:00 PM	Trace	@AIConnect2
Exception message	An error occurred while updating the entries. See the inner exception for details. <--- The INSERT statement conflicted with the CHECK constraint "chk_read_only". The conflict occurred in database "FabrikamSQL", table "dbo.Customers". The statement has been terminated.	
Instance name	AIConnect2	
System.Data.Entity.Infrastructure.DbUpdateException: An error occurred while updating the entries. The statement has been terminated. at System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action<SqlException> userAction, String handleProcessSqlException) at System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncEvent) at System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject stateObj, Boolean& dataIsAvailable, Boolean startWithFirstData, Boolean isCommandText) at System.Data.SqlClient.SqlDataReader.get_MetaData() at System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavior, Boolean calledFromExecuteReader) at System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, Boolean async, Int32 timeout, TaskCompletionSource< SqlDataReader> taskCompletionSource, Boolean& mustCloseConnection)		

Application Topology

fabrikamprod - Application map
Application Insights - Last 24 hours - fabrikamprod

Search (Ctrl+ /) Time range Filters Options Refresh Restore defaults Learn more

Warning thresholds: <5% >=5% >=20% failed

Overview Activity log Access control (IAM) Tags

INVESTIGATE Application map Smart Detection Live Metrics Stream Metrics Explorer Metrics (preview) Availability Failures Performance Servers Browser

USAGE (PREVIEW) Users Sessions Events Retention Groups

Availability
2 83.43% passing 35.96 ms

Client: fabrikamprod
38 errors 831.33 ms 48 page views

Unknown (fabrikam...)
9.8K requests 35.86 ms 16.4% failures

fabrikamfiberapp
5.9K requests 403.11 ms 12.5% failures

SQL
3.7K calls 15.83 ms 19.6% failures

tcp:fabrikamxyz.dat...
3.7K calls 15.82 ms 19.6% failures

tcp:fabrikamxyz.dat...
10 calls 19.4 ms 0% failures

tcp:fabrikamxyz.dat...
2.2K calls 21.02 ms 0% failures

tcp:fabrikamxyz.dat...
2 calls 16 ms 0% failures

ppe.azureservicepro...
96 calls 207.94 ms 0% failures

Unknown (fabrikamprod) SERVERAPPLICATION

Top Errors See all errors

- GET Home/Index 1 problem, 3030 instances Failed Dependency: 409
- GET Customers/Details 3 problems, 2876 instances Failed Request: 500 System.FormatException at Fabrikam... Failed Dependency: 409
- POST ServiceTickets/Create 3 problems, 2172 instances Failed Request: 500 System.Data.SqlClient.SqlException a... Failed Dependency: 547
- GET Employees/Create 2 problems, 1727 instances Failed Request: 404 System.Web.HttpException at System...
- POST Customers/Create 3 problems, 9 instances
- GET Reports/Employees 3 problems, 8 instances
- GET /Content/fonts/segoewp... 1 problem, 7 instances
- GET /Content/fonts/segoewp... 1 problem, 7 instances
- GET robots.txt/Index 2 problems, 4 instances

Ad-Hoc Data Analytics

Application Insights fabrikamprod > Analytics Report bug Rahul Bagaria

SCHEMA FILTER [Home Page](#) [fabrikam_inci...](#) [Jeremy Demo](#) [+](#) Export Last 24 hours GO

//Latest 100 Records
requests | take 100

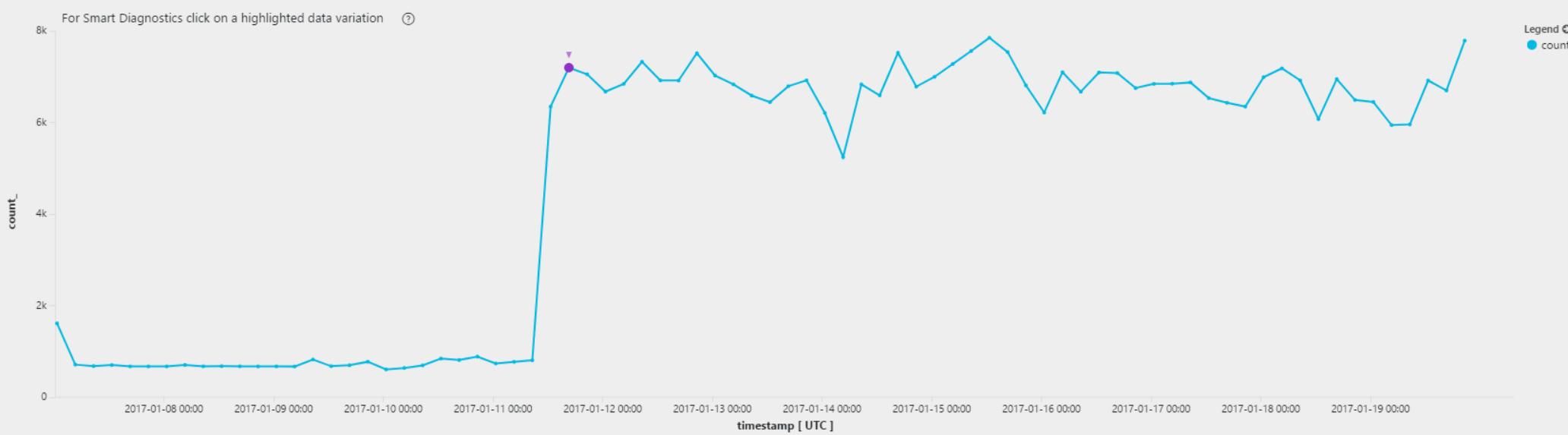
//Identifying too long requests with dynamic buckets
requests
| where timestamp >= ago(1d)
| extend responseBucket = iff(duration > 3000, "too long", "ok")
| project name, duration, responseBucket

//Am I meeting my SLA?
requests
| where timestamp >= ago(8d)
| summarize slaMet=count(duration < 3000), slaBreached=count(duration >= 3000), totalCount=count() by bin(timestamp, 1h)
| extend pctSLAIndex = slaMet*100.0 / totalCount
| project pctSLAIndex, timestamp
| render timechart

//Identifying spikes in request load
requests
| where timestamp between (datetime(01-07-2017)..datetime(01-20-2017))
| summarize count() by bin(timestamp, 4h)
| render timechart

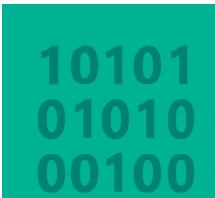
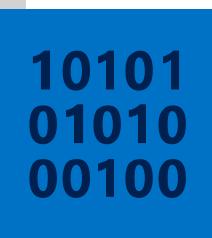
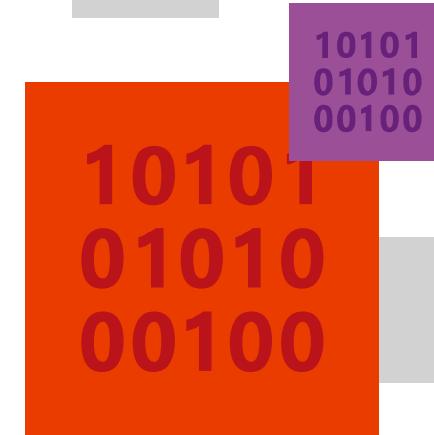
Completed 00:00:00.955 78 records loaded

TABLE CHART | Line Timestamp Count_ Sum
For Smart Diagnostics click on a highlighted data variation Legend count_



Timestamp [UTC]	Count_
2017-01-08 00:00:00	~1500
2017-01-09 00:00:00	~500
2017-01-10 00:00:00	~500
2017-01-11 00:00:00	~500
2017-01-12 00:00:00	~6500
2017-01-13 00:00:00	~7000
2017-01-14 00:00:00	~5500
2017-01-15 00:00:00	~7000
2017-01-16 00:00:00	~6800
2017-01-17 00:00:00	~6800
2017-01-18 00:00:00	~7000
2017-01-19 00:00:00	~6200
2017-01-20 00:00:00	~7500

Best Practices (Do's)



Do understand Azure Scalability & Performance Targets

- Storage Scalability and Performance Targets
 - azure.microsoft.com/documentation/articles/storage-scalability-targets
- Azure service limits, quotas, and constraints
 - docs.microsoft.com/azure/azure-subscription-service-limits
 -

Scalability targets for blobs, queues, tables, and files

Resource	Default Limit
Number of storage accounts per subscription	200 ¹
Max storage account capacity	500 TB ²
Max number of blob containers, blobs, file shares, tables, queues, entities, or messages per storage account	No limit

App Service limits

The following App Service limits include limits for Web Apps, Mobile Apps, API Apps, and Logic Apps.

Resource	Free	Shared (Preview)	Basic	Standard	Premium (Preview)
Web, mobile, or API apps per App Service plan ¹	10	100	Unlimited ²	Unlimited ²	Unlimited ²
Logic apps per App Service plan ¹	10	10	10	20 per core	20 per core
App Service plan	1 per region	10 per resource group	100 per resource group	100 per resource group	100 per resource group

Do use the appropriate type of Store

Storage Mechanism	Interaction	Starts at*	Min GB	Max TB	Use Case
'Cool' Block Blob storage	REST (Blob), SDKs	£0.01/GB	0	500	Archive, nearline unstructured https storage
'Hot' Block Blob storage	REST (Blob), SDKs	£0.02/GB	0	500	Online unstructured https storage
Virtual Machine Disk / Page Blob Storage	VM Only	£0.05/GB	0	500 (Not per VM)	High Performance storage, up to 500 IOPS per disk volume (up to max volumes per VM type).
Table Storage	REST (Blob), SDKs	£0.07/GB	0	500	Tabular, non-relational (NoSQL) Mass-scale dictionary https lookup service, partitioned by default, no secondary indexes allowed.
Premium Virtual Machine Disk/ Page Blob Storage	VM Only	£0.13/GB	128	35	GUARANTEED High Performance storage, up to 5000 IOPS per disk volume (up to max volumes per VM type > 80,000 IOPS & 2TB / Second).
SQL Database Basic	T-SQL (TDS)	£2.33/GB *	2	1	Tabular, Scalable, Classic Relational DBMS, Always On
SQL Data Warehouse	T-SQL (TDS)	£1.03/GB *	1024	~250 Compressed (~1PB)	Scalable, Tabular, Parallelized, Relational DBMS, can be paused.
DocumentDB	REST (Blob), SDKs	£4.62/GB *	1	0.01 (10GB)	JSON Indexed document storage, can be partitioned.
Data Lake Store	WebHDFS	£0.06/GB *	1024	1PB per file (!)	Hadoop based unstructured data storage layer. – Preview price includes 50% discount off US pricing, not available in Europe yet.

* These are indicative numbers, these numbers are variable on a number of factors. For example SQL DW starts billing at 1TB of capacity, storage has an access charge per million transactions.

Anti-Practices (Don'ts)



Scalability – Ignore at your peril



IGNORE YOUR SCALABILITY APPROACH

Some customers believed that the Azure platform automatically handled scaling across all Azure services, without any configuration. This is not true.

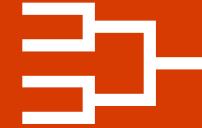
This is an important concept, which customers must be aware of. For example, Web Apps require scalability rules to be configured, as do VM Scale Sets.



IGNORE DOWNTIME FOR SCALE UP

A number of customers had Single Points of Failure, due to having single instances of certain resources. As such, if they wanted to scale up/out, there could potentially be some down time.

Customers should be aware of the requirements to meet Azure SLAs (particularly for VMs), and also how the platform scales services in use.



SHARE APP SERVICE STAGING SLOTS

Some customers have planned to host their separate environments on the same Web Application, but across different staging slots. All slots run on the same machine, so any strain on load would effect all environments.

Customers should ensure that the environments are appropriately separated, so that load in one environment, does not negatively impact another.

Performance – ignore at your peril



IGNORE CACHING / OR USE EXCESSIVE CACHING

Many customers caching approach is not appropriate for their solution. This is true across various levels (e.g. Caching of data via Redis cache, through to caching of assets via CDN).

Customers should consider their SLAs (in terms of service requirements), and understand how caching can improve performance compared to adding additional complexity and cache lifecycle headaches.



LACK OF DESIGN FOR LATENCY

Highly variable latency is normal for access to internet based services as are random timeouts and transient errors as services auto-scale and auto-recover.

When operating services in the cloud, latency constraints usually need to be relaxed and there needs to be an acceptance that some operations will timeout and need to be retried automatically.

