#FABCONSQLCON2026

FABCON
Microsoft Fabric
COMMUNITY CONFERENCE

SQLCON
Microsoft SQL
COMMUNITY CONFERENCE

ATLANTA MARCH 16 - 20, 2026

Driving Cost-Efficient Data Architectures for SQL Server, Azure SQL and Fabric using FinOps

Hamish Watson

Hamish Watson

He/him/WE

DevOps Consultant

Morph iT Limited

CEO & AI Consultant

Make Stuff Go Limited

- Many years mucking about with 1s & 0s
- Bringing DevOps to databases (and the masses) is a personal passion
- Understanding AI, data and cloud is a company driver
- Technologist who understands business value…
- #makeStuffGo

@theHybridDBA

https://www.makestuffgo.com

hamish@makestuffgo.com

redgate COMMUNITY AMBASSADOR 2024

Microsoft® MVP Most Valuable Professional

Microsoft® MVP Most Valuable Professional

makeStuffGo

The myth of DevOps

+

Why integrate FinOps with DevOps?

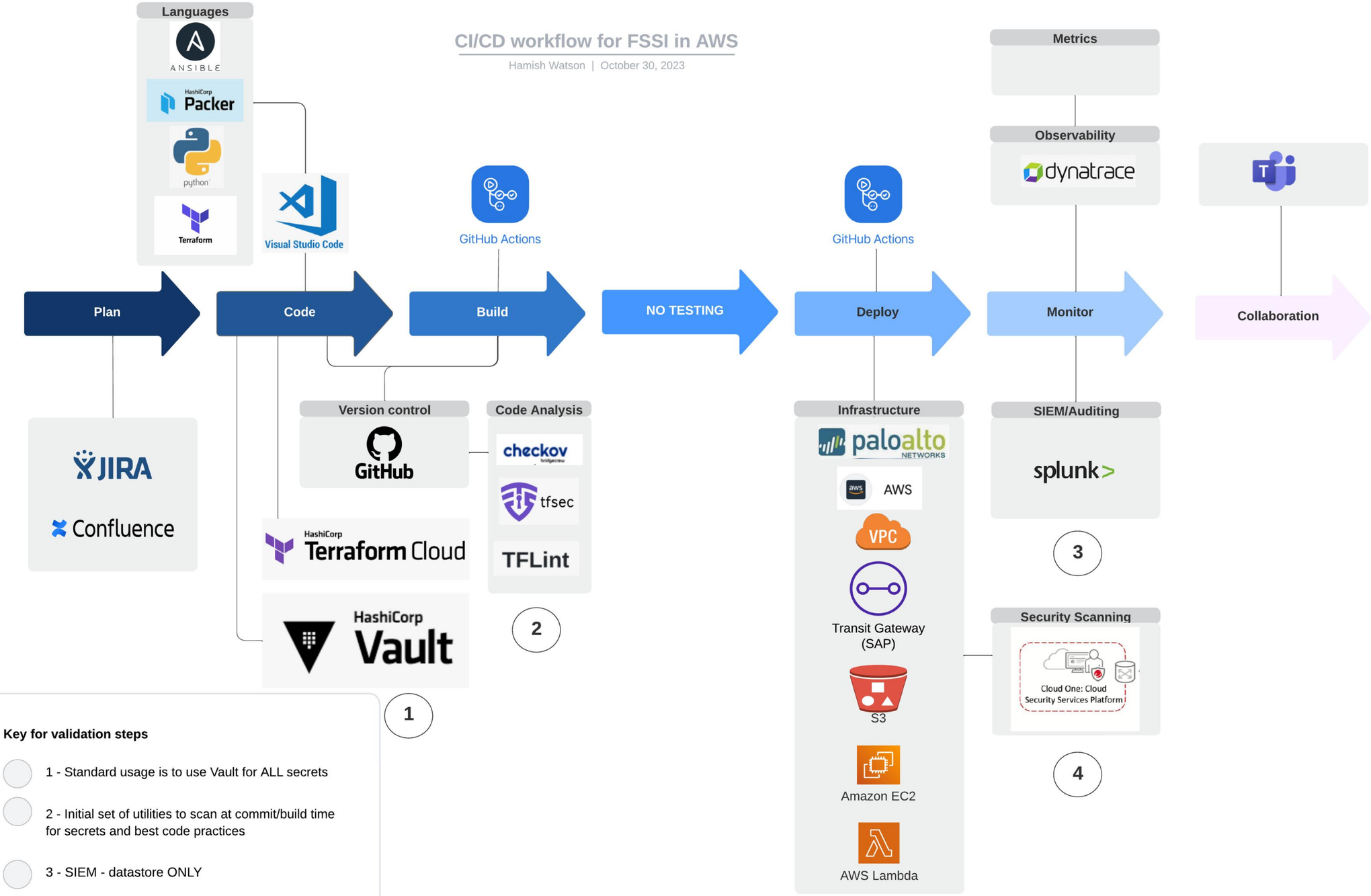
- Scalability and Dynamic Pricing
- Diverse Service Offerings
- Complex Architecture and Management

FinOps is the best mate of DevOps

- Enhanced Cost Efficiency
- Improved Financial Visibility
- Faster Innovation with Accountability

CI/CD workflow for FSSI in AWS

Hamish Watson | October 30, 2023

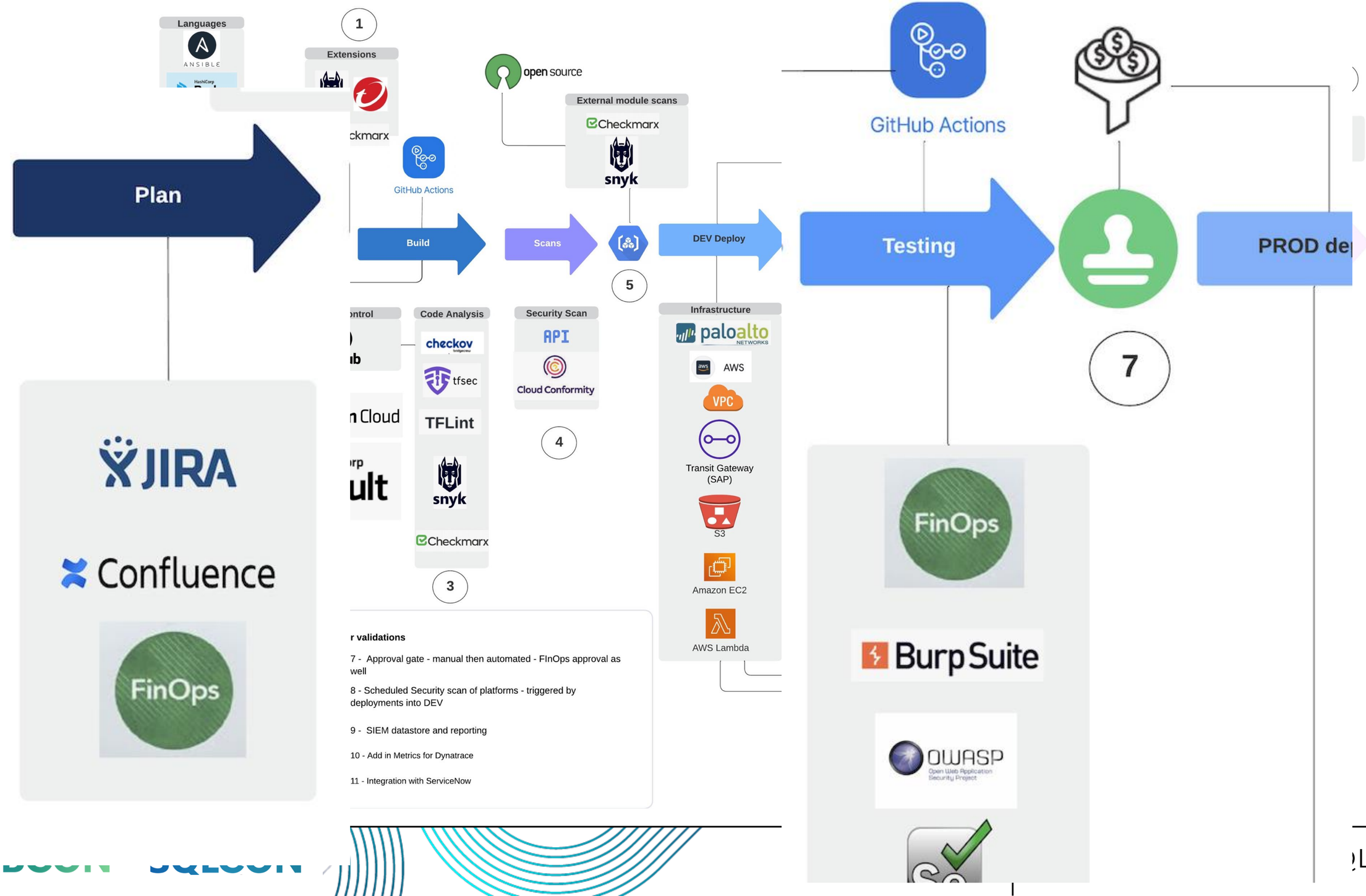


Key for validation steps

- 1 - Standard usage is to use Vault for ALL secrets
- 2 - Initial set of utilities to scan at commit/build time for secrets and best code practices
- 3 - SIEM - datastore ONLY
- 4 - Security scanning - manual at this stage

The true effect of FinOps and DevOps

- Automated Cost Monitoring
- Proactive Resource Optimisation
- Continuous Deployment with Cost Control


Languages
ANSIBLE
1
Extensions
ckmarx
open source
External module scans
Checkmarx
snyk
GitHub Actions
Plan
Build
Scans
5
DEV Deploy
GitHub Actions
Testing
PROD de
ntrol
Code Analysis
Security Scan
Infrastructure
checkov
API
paloalto NETWORKS
tfsec
AWS
Cloud Conformity
VPC
n Cloud
TFLint
4
Transit Gateway (SAP)
rp
ult
snyk
S3
Checkmarx
Amazon EC2
3
AWS Lambda
7
JIRA
Confluence
FinOps
FinOps
Burp Suite
OWASP
Open Web Application Security Project
r validations
7 - Approval gate - manual then automated - FInOps approval as well
8 - Scheduled Security scan of platforms - triggered by deployments into DEV
9 - SIEM datastore and reporting
10 - Add in Metrics for Dynatrace
11 - Integration with ServiceNow

Tooling to help you with cost management

CloudHealth by vmware

Cost Management

Infracost

AWS Cost Explorer

Driving Cost-Efficient Data Architectures

for SQL Server, Azure SQL & Microsoft Fabric

Applying FinOps + AI to modern data platforms

Core promise

Show where cost actually comes from, how FinOps makes it visible, and how AI turns cost data into architectural action.

Audience

Data architects
DBAs
Platform engineers
FinOps teams

Outcome

Leave with concrete patterns for right-sizing, workload scheduling, and hybrid cost governance.

Hamish • Conference Session

Opening story: the “why is our data bill so high?” moment

What happened

The analytics platform worked brilliantly. Adoption increased. More teams used the warehouse, notebooks, ETL and dashboards. Six months later the monthly bill had doubled.

Why nobody noticed earlier

- Compute was sized for peak
- Dev/test stayed online
- Storage kept accumulating
- Multiple teams shared the same capacity

The real lesson

Cost is not a finance-only problem. It is an architecture problem. If the platform shape is wrong, the bill will reflect it.

Why data platform costs explode

- Compute is provisioned for the worst day, but billed every day
- Storage rarely shrinks; historical data and duplication accumulate
- Shared analytics environments hide which team or workload caused spend
- Performance issues are solved by scaling up first and analysing later

Symptom

“We have good adoption and bad unit economics.”

Root cause

No continuous feedback loop between usage, architecture and cost.

FinOps answer

Make cost observable enough that engineers can act on it every week, not every quarter.

Driving Cost-Efficient Data Architectures

for SQL Server, Azure SQL & Microsoft Fabric

Applying FinOps + AI to modern data platforms

Core promise

Show where cost actually comes from, how FinOps makes it visible, and how AI turns cost data into architectural action.

Audience

Data architects
DBAs
Platform engineers
FinOps teams

Outcome

Leave with concrete patterns for right-sizing, workload scheduling, and hybrid cost governance.

Opening story: the “why is our data bill so high?” moment

What happened

The analytics platform worked brilliantly. Adoption increased. More teams used the warehouse, notebooks, ETL and dashboards. Six months later the monthly bill had doubled.

Why nobody noticed earlier

- Compute was sized for peak
- Dev/test stayed online
- Storage kept accumulating
- Multiple teams shared the same capacity

The real lesson

Cost is not a finance-only problem. It is an architecture problem. If the platform shape is wrong, the bill will reflect it.

Why data platform costs explode

- Compute is provisioned for the worst day, but billed every day
-
- Storage rarely shrinks; historical data and duplication accumulate
-
- Shared analytics environments hide which team or workload caused spend
-
- Performance issues are solved by scaling up first and analysing later

Symptom

“We have good adoption and bad unit economics.”

Root cause

No continuous feedback loop between usage, architecture and cost.

FinOps answer

Make cost observable enough that engineers can act on it every week, not every quarter.

FinOps for data teams: what it actually means

Visibility

Know who is spending, on what workload, at what time, and for what business outcome.

Optimisation

Right-size compute, schedule workloads, remove waste, and choose the right platform tier.

Governance

Create guardrails so the platform stays efficient as new teams, products and data volumes arrive.

Where cost shows up across the stack

SQL Server

- VM sizing
- Licensing
- HA/DR replicas
- Storage tiering
- Backup retention

Azure SQL

- vCore / DTU choice
- Serverless settings
- Elastic pool usage
- Long-running queries
- Idle non-prod databases

Microsoft Fabric

- Capacity size
- Shared workload contention
- Notebook / ETL scheduling
- Lakehouse sprawl
- Workspace governance

SQL Server: classic waste patterns

- Oversized VMs and always-on infrastructure for occasional peaks
-
- Replica and HA topologies sized for resilience but rarely reviewed for cost
-
- Licensing inefficiency when CPU allocation drifts upward over time
-
- Hot storage used for cold data because retention strategy is unclear

Low-effort win

Review HA/DR estate and environment tiers before buying more hardware or more SQL licences.

Architectural win

Move from “big iron everywhere” to workload-specific sizing and lifecycle tiers.

Azure SQL: flexibility creates new optimisation choices

- Right-size vCores against actual workload patterns instead of migration assumptions
-
- Use serverless carefully; it saves money only when pause/resume behaviour matches reality
-
- Separate production, dev and test service levels instead of cloning premium tiers
-
- Surface long-running queries before they silently become a compute tax

Good FinOps question

“What proportion of our Azure SQL estate is sized for migration comfort rather than current demand?”

Fabric: shared capacity makes governance essential

- One capacity can mask multiple competing workloads with different business value
-
- Interactive analytics, ETL, notebooks and warehousing fight for the same performance budget
-
- Without scheduling, overnight and background jobs consume premium capacity for low-value work
-
- Workspace sprawl makes ownership and optimisation difficult

Fabric mindset shift

Think in terms of workload management and capacity economics, not just feature enablement.

Bad architecture vs good architecture

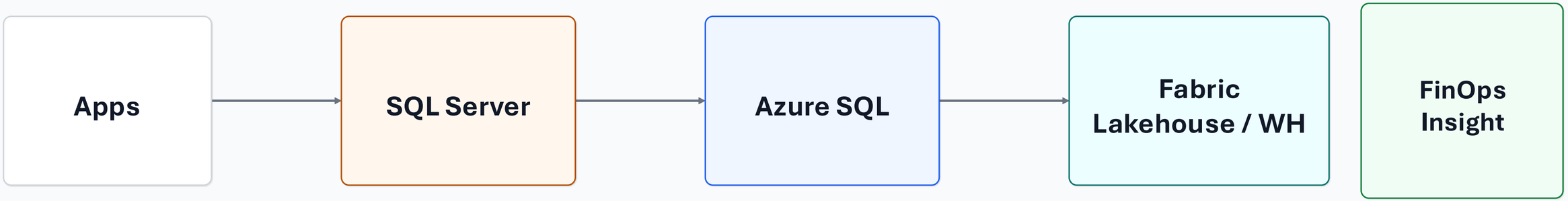
Bad: platform built for fear

- One large always-on environment
- Shared by every team
- No scheduling
- No ownership per workload
- Cost reviews happen after the bill arrives

Good: platform built for economics

- Workloads isolated by value and criticality
- Non-prod environments scaled differently
- Capacity scheduled around demand
- Owners can see what they consume
- Optimisation is part of weekly operations

Architecture diagram 1: hybrid data platform to FinOps insight



Operational data moves through the platform. Telemetry, utilisation and spend signals move with it.

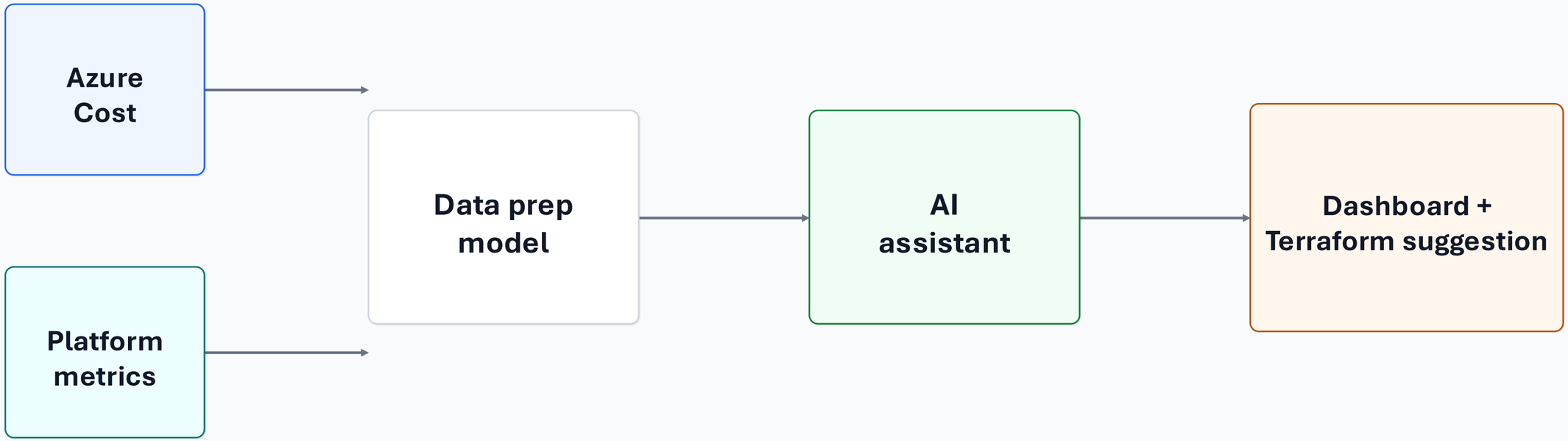
Architecture diagram 2: the FinOps control loop

Measure → attribute → analyse → change architecture → measure again

AI optimisation pipeline

Typical outputs: anomaly detection, spend forecasting, right-sizing suggestions, workload scheduling and environment shutdown windows.

Architecture diagram 3: demo architecture



The demo shows AI reading cost + usage data, highlighting waste, and proposing an operational response.

Demo walkthrough: what the audience should expect

1. Ingest

Bring together spend data, query/load telemetry and capacity history.

2. Analyse

Ask AI to find anomalies, idle assets, underused capacity and suspicious growth patterns.

3. Recommend

Generate right-sizing, scheduling and ownership recommendations for engineering teams.

4. Act

Turn the most valuable suggestions into a backlog item, policy or Terraform change.

FinOps maturity model

Allocate spend
Name owners
See waste

Right-size
Schedule
Tune workload mix

Guardrails
Policy
AI-assisted operations

Where AI agents can genuinely help

- Summarise what changed in spend this week and point to the most likely causes
-
- Suggest which environments can be scaled down, paused or rescheduled safely
-
- Draft Terraform or policy changes for human review instead of starting from scratch
-
- Turn raw telemetry into plain-language operational advice for platform teams

Important constraint

An agent should recommend and accelerate. It should not become an unsupervised cost optimiser with no architectural context.

A simple operating model for FinOps in data teams

Weekly

Review anomalies, idle assets, capacity hotspots and non-prod usage.

Monthly

Review architectural patterns, service tier choices and ownership alignment.

Quarterly

Revisit platform shape: where to consolidate, isolate, retire or automate.

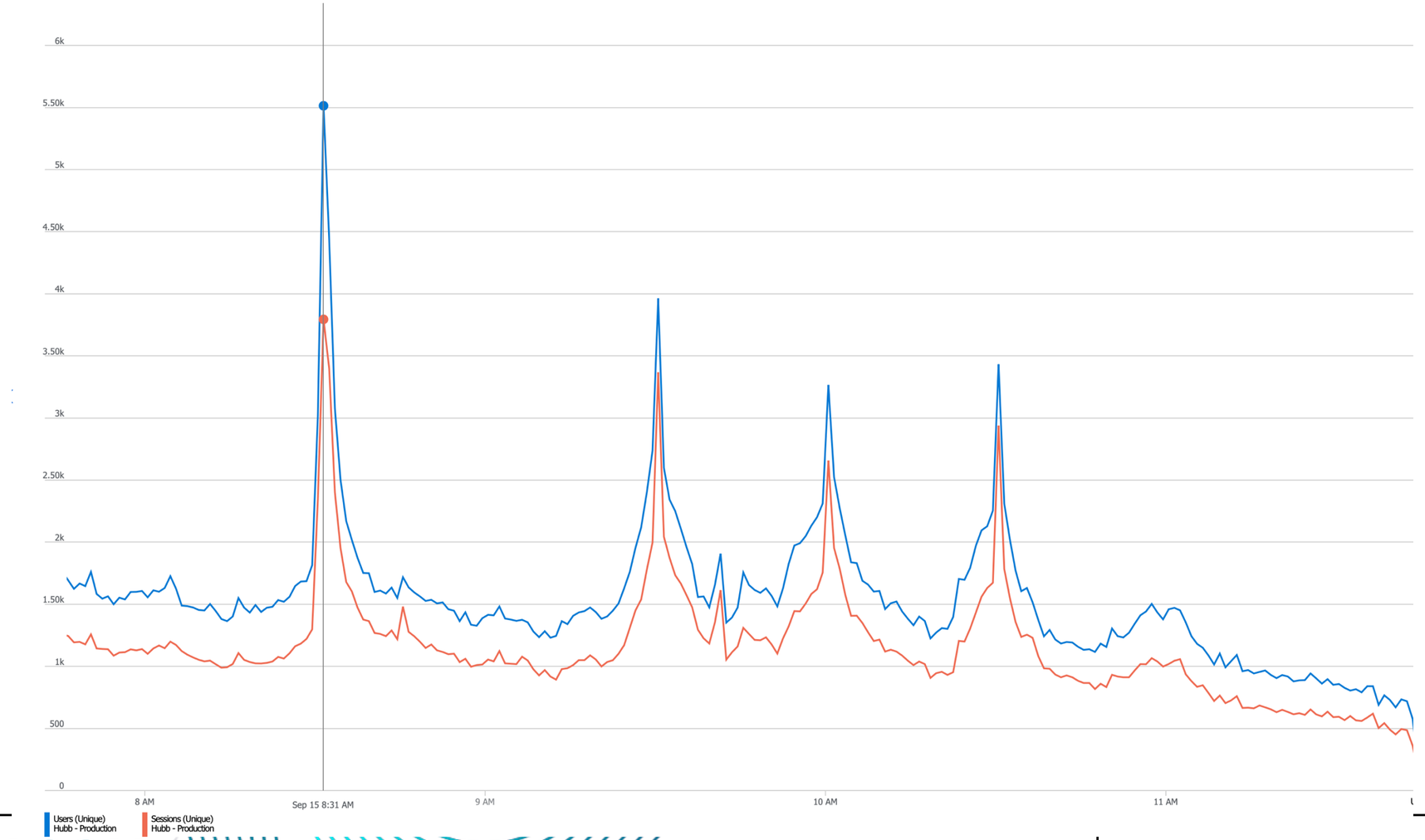
CASE STUDY

Covid wasn’t completely bad….

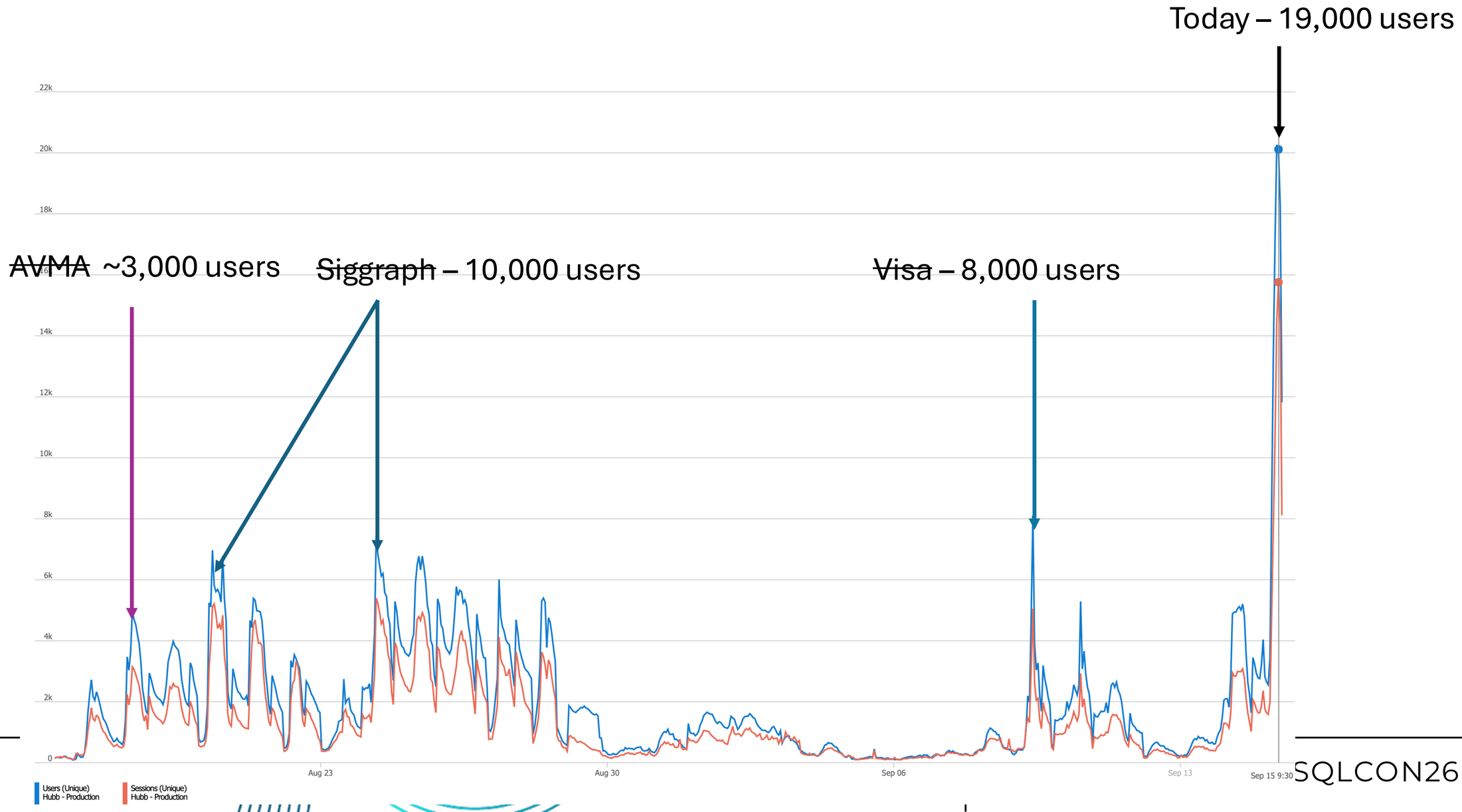
Discussion Points – each week then monthly

- What’s usage like?
- What’s the cost rate been like?
- What’s happened in the past month?
- What’s next?

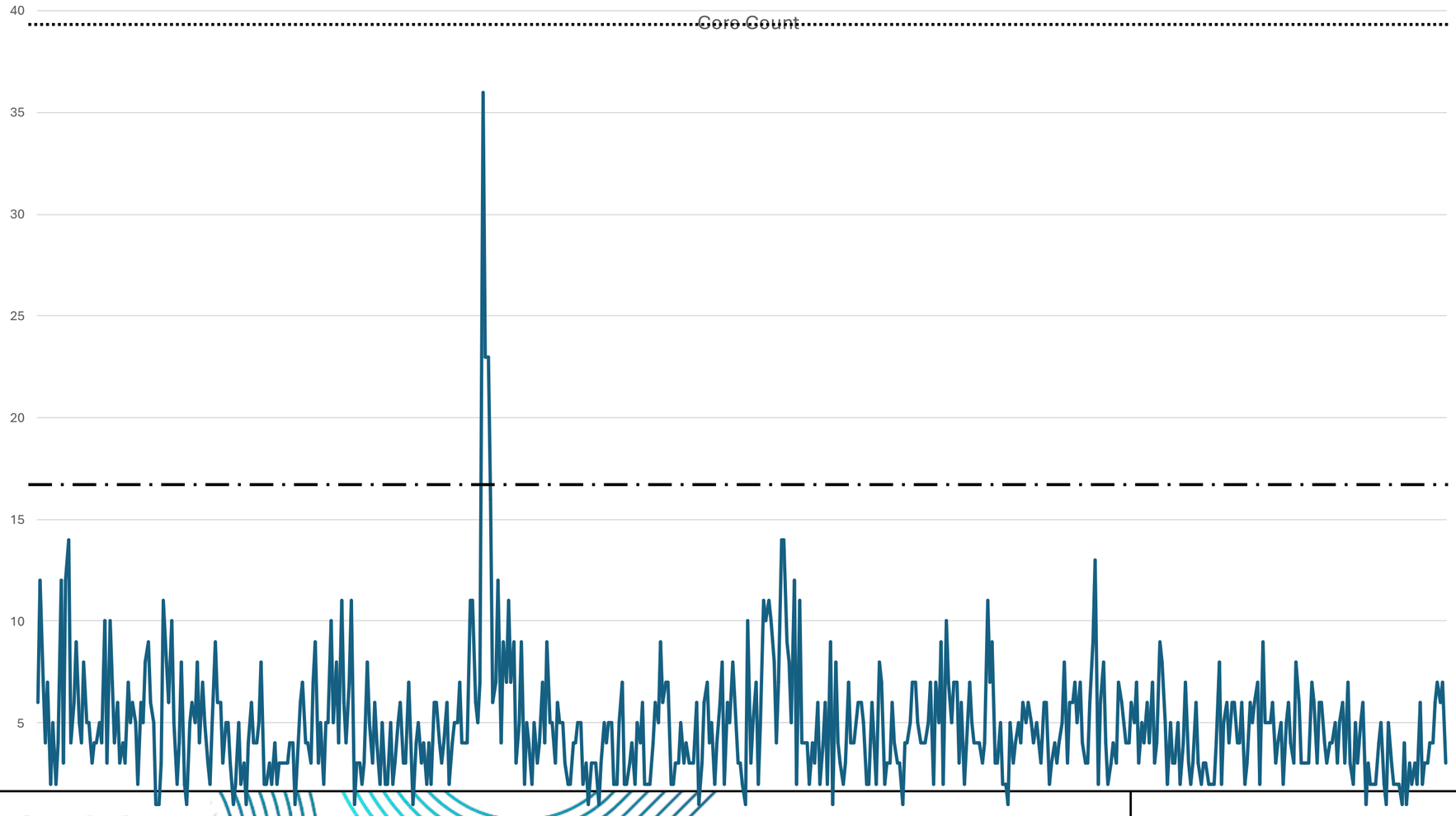
This week is going to be big...



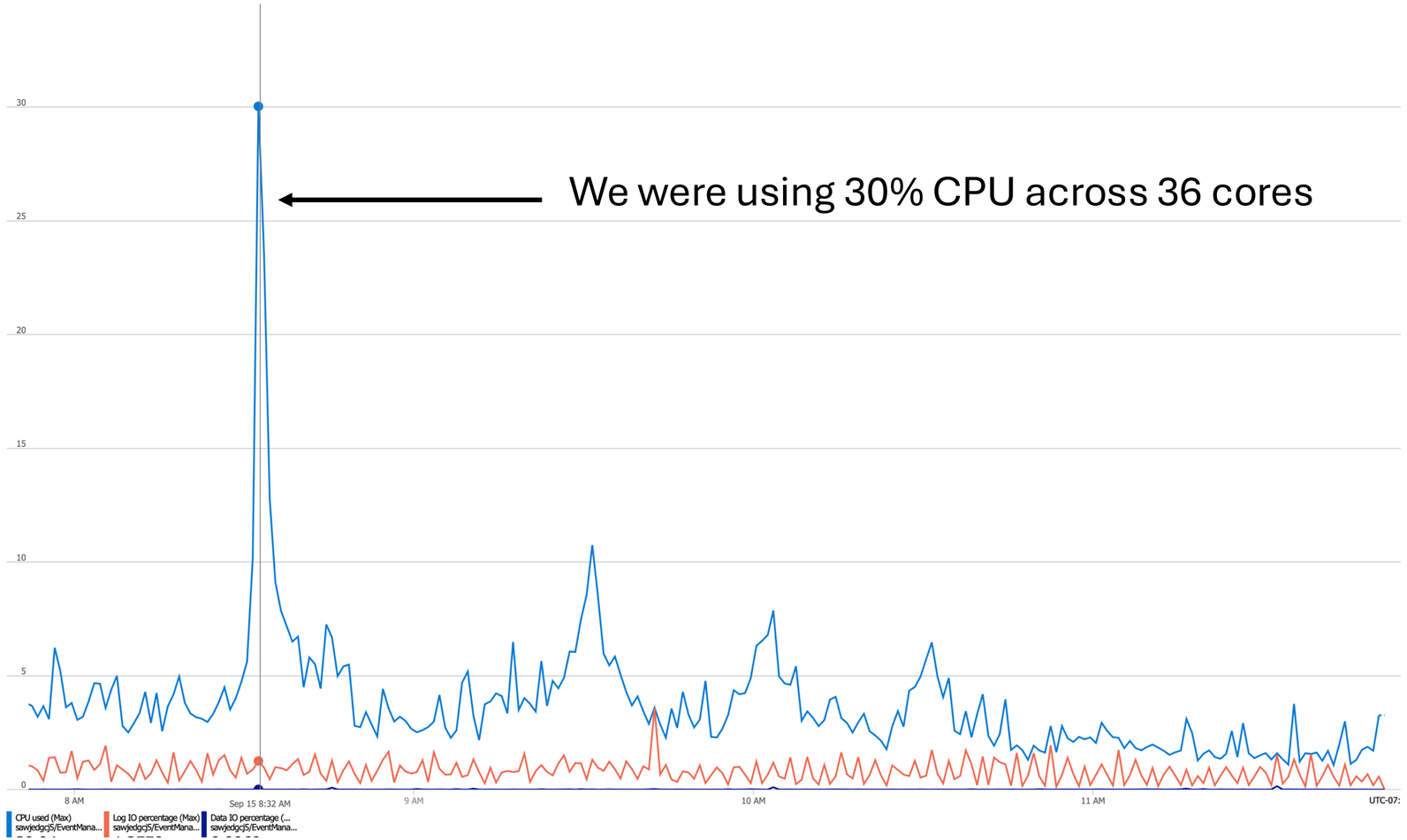
Let’s zoom out a bit…



The effect on the database - cores



How hot were we running?



Burn rate - $ (App Services)

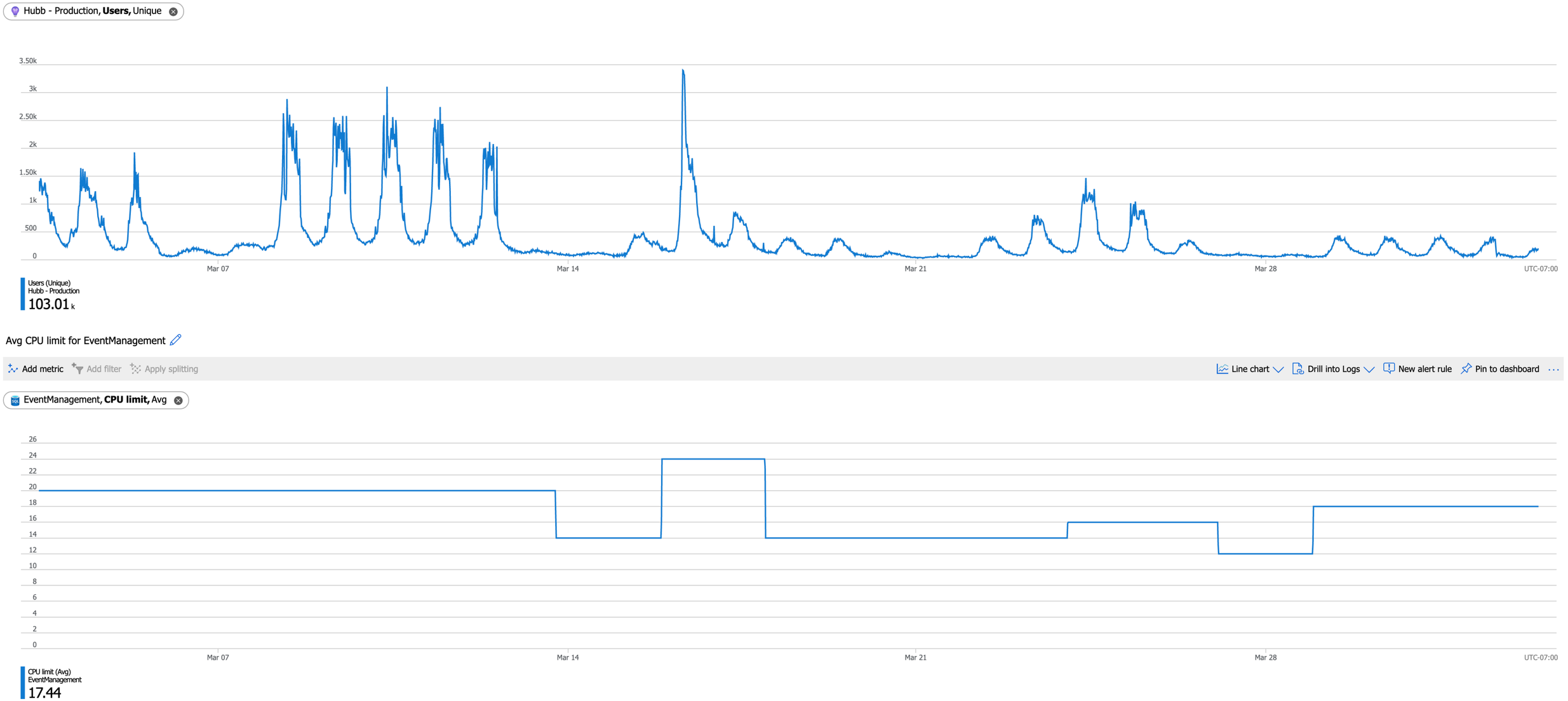
6th July	$3,520 / day	(ON FIRE)
14th July	$4,235 / day	(4 App Service Plans)
25th July	$3,310 / day	(first scale down weekend)
27th July	$5,534 / day	(all apps scaled out to max)
August w/e	~$845 /day	(automated scale/monitoring)
August weekday	~$2,500 / day	(scaled out)
September w/e	~$407 / day	(more scaling in)
September w/day	~$904 / day	(true scaling implemented)

Burn rate - $ (Database Cores)

More straightforward:

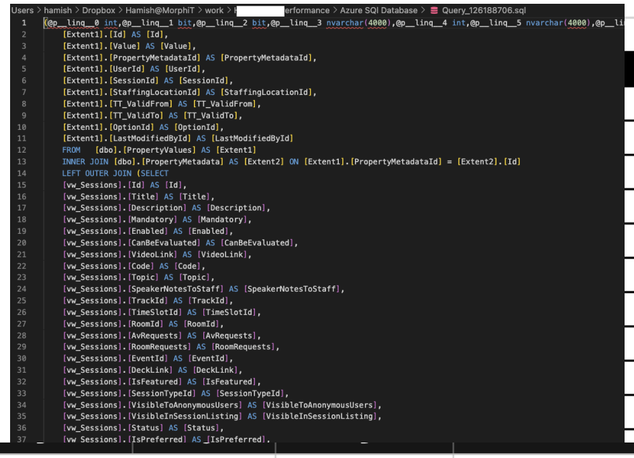
6th July	$2,530 / day
August w/e	$964 / day
August w/day	$1,530 / day
September w/e	~$260 / day
September w/day	~330 / day

Database scaling – eye on perf vs $



CONVERSATION

How did we make the database better...

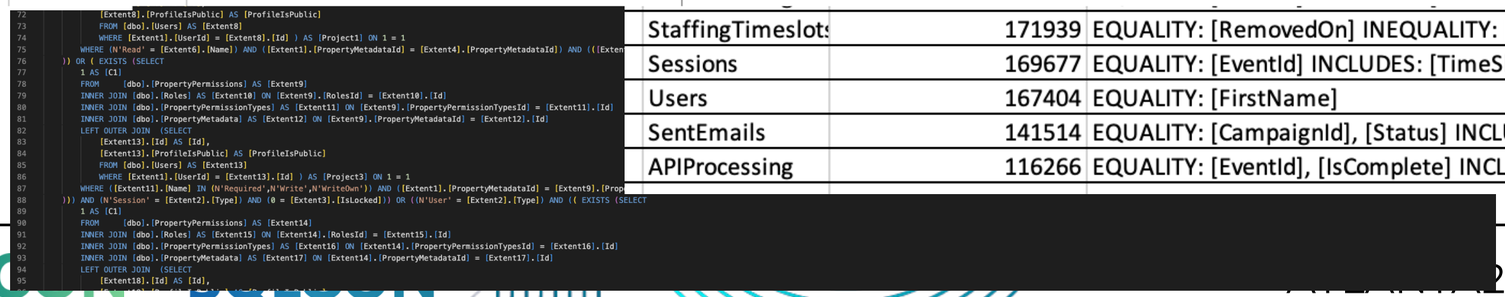


```
[@p__linq__0 int,@p__linq__1 bit,@p__linq__2 bit,@p__linq__3 nvarchar(4000),@p__linq__4 int,@p__linq__5 nvarchar(4000),@p__li
    [Extent1].[Id] AS [Id],
    [Extent1].[Value] AS [Value],
    [Extent1].[PropertyMetadataId] AS [PropertyMetadataId],
    [Extent1].[UserId] AS [UserId],
    [Extent1].[SessionId] AS [SessionId],
    [Extent1].[StaffingLocationId] AS [StaffingLocationId],
    [Extent1].[TT_ValidFrom] AS [TT_ValidFrom],
    [Extent1].[TT_ValidTo] AS [TT_ValidTo],
    [Extent1].[OptionId] AS [OptionId],
    [Extent1].[LastModifiedById] AS [LastModifiedById]
    FROM  [dbo].[PropertyValues] AS [Extent1]
    INNER JOIN [dbo].[PropertyMetadata] AS [Extent2] ON [Extent1].[PropertyMetadataId] = [Extent2].[Id]
    LEFT OUTER JOIN (SELECT
    [vw_Sessions].[Id] AS [Id],
    [vw_Sessions].[Title] AS [Title],
    [vw_Sessions].[Description] AS [Description],
    [vw_Sessions].[Mandatory] AS [Mandatory],
    [vw_Sessions].[Enabled] AS [Enabled],
    [vw_Sessions].[CanBeEvaluated] AS [CanBeEvaluated],
    [vw_Sessions].[VideoLink] AS [VideoLink],
    [vw_Sessions].[Code] AS [Code],
    [vw_Sessions].[Topic] AS [Topic],
    [vw_Sessions].[SpeakerNotesToStaff] AS [SpeakerNotesToStaff],
    [vw_Sessions].[TrackId] AS [TrackId],
    [vw_Sessions].[TimeSlotId] AS [TimeSlotId],
    [vw_Sessions].[RoomId] AS [RoomId],
    [vw_Sessions].[AvRequests] AS [AvRequests],
    [vw_Sessions].[RoomRequests] AS [RoomRequests],
    [vw_Sessions].[EventId] AS [EventId],
    [vw_Sessions].[DeckLink] AS [DeckLink],
    [vw_Sessions].[IsFeatured] AS [IsFeatured],
    [vw_Sessions].[SessionTypeId] AS [SessionTypeId],
    [vw_Sessions].[VisibleToAnonymousUsers] AS [VisibleToAnonymousUsers],
    [vw_Sessions].[VisibleInSessionListing] AS [VisibleInSessionListing],
    [vw_Sessions].[Status] AS [Status],
    [vw_Sessions].[IsPreferred] AS [IsPreferred],
```

Table	Magic Benefit	Missing Index Details	Avg Query Cost	Est Index Improvement	Seeks
FloorMapObjects	52617217	EQUALITY: [FloorMapObjectLayerID] INCLUDES: [Han	165.7	96.2	33
PropertyValues	47374051	EQUALITY: [LastModifiedById] INCLUDES: [TT_ValidF	83.6	91.4	62
Audit	19941429	EQUALITY: [RecordID], [Table], [OperationType] INCL	1,055.1	94.5	2
SentEmails	16180030	EQUALITY: [CampaignId] INCLUDES: [SentTo], [SentT	22.6	15.0	477
SentEmails	7324886	INEQUALITY: [Status] INCLUDES: [SentTo], [SentToCC	174.1	52.6	8
SentEmails	6920313	EQUALITY: [Status] INCLUDES: [UserId]	176.5	98.0	4
SentEmails	5674644	EQUALITY: [Status] INCLUDES: [SentTo], [SentToCC],	173.3	65.5	5
SentEmails	3201539	EQUALITY: [Status] INCLUDES: [SentTo], [SentToCC],	185.1	86.5	2
SentEmails	2260724	EQUALITY: [Status] INCLUDES: [SentTo], [SentToCC],	172.6	65.5	2
APIProcessing	2090575	EQUALITY: [LogicAppActionId], [IsComplete] INCLUDE	137.0	76.3	2

Queries by Total Duration

ost	QueryText	Query Type	Warnings	Executions
0	CREATE PROCEDUF	Procedure or Function: [dbo].[SP_DBMaint]	Forced Parameterization, Parameter Sniffing, Long Running Query, Function Join, Forced Serialization, Table Variables, Low Cost High CPU, >500mb Spills, MSTVFs	9
0	CREATE PROCEDURE	Procedure or Function: [dbo].[LogicAppProcesses]	Forced Parameterization, Plan Warnings, Forced Serialization, Table Variables	15098
5	CREATE PROCEDU	Procedure or Function: [dbo].[SP_GetNewEmails]	Forced Parameterization, Plan Warnings, Implicit Conversions, Forced Serialization, Table Variables	1741
0	select [Distinct1] . [Id	Statement	Forced Parameterization, Forced Serialization	149926
925	CREATE PROCEDURE	Procedure or Function: [dbo].[ProcessUserProfiles]	Forced Parameterization, Function Join, Trivial Plans, Forced Serialization, Calls 1 function(s), >= 5 Indexes Modified, Many Rows Table Spool, Many to Many Merge, non-SARGables	2635
3	CREATE TRIGGER [	Procedure or Function: [dbo].[TRG_Users_UpdatedData]	Forced Parameterization, Plan Warnings, Forced Serialization	161500
3	INSERT INTO dbo.Auc	Statement	Forced Parameterization, Unparameterized Query, Plan Warnings, Forced Serialization, Row Estimate Mismatch, non-SARGables	161500
0	CREATE PROCEDURE	Procedure or Function: [dbo].[User_Update]	Forced Parameterization, Forced Serialization, Calls 1 Function(s), Forced Indexes	55539
6	SELECT [Project9]	Statement	Compilation Timeout, Forced Parameterization, Forced Serialization, non-SARGables	2295
0.00658724	SELECT [Limit1].[I	Statement	Forced Parameterization, Forced Serialization	4133436



```
            [Extent8].[ProfileIsPublic] AS [ProfileIsPublic]
            FROM [dbo].[Users] AS [Extent8]
            WHERE [Extent1].[UserId] = [Extent8].[Id] ) AS [Project1] ON 1 = 1
        WHERE (N'Read' = [Extent6].[Name]) AND ([Extent1].[PropertyMetadataId] = [Extent4].[PropertyMetadataId]) AND (([Exten
    )) OR ( EXISTS (SELECT
        1 AS [C1]
        FROM     [dbo].[PropertyPermissions] AS [Extent9]
        INNER JOIN [dbo].[Roles] AS [Extent10] ON [Extent9].[RolesId] = [Extent10].[Id]
        INNER JOIN [dbo].[PropertyPermissionTypes] AS [Extent11] ON [Extent9].[PropertyPermissionTypesId] = [Extent11].[Id]
        INNER JOIN [dbo].[PropertyMetadata] AS [Extent12] ON [Extent9].[PropertyMetadataId] = [Extent12].[Id]
        LEFT OUTER JOIN  (SELECT
            [Extent13].[Id] AS [Id],
            [Extent13].[ProfileIsPublic] AS [ProfileIsPublic]
            FROM [dbo].[Users] AS [Extent13]
            WHERE [Extent1].[UserId] = [Extent13].[Id] ) AS [Project3] ON 1 = 1
        WHERE ([Extent11].[Name] IN (N'Required',N'Write',N'WriteOwn')) AND ([Extent1].[PropertyMetadataId] = [Extent9].[Prop
    ))) AND (N'Session' = [Extent2].[Type]) AND (0 = [Extent3].[IsLocked])) OR ((N'User' = [Extent2].[Type]) AND (( EXISTS (SELECT
        1 AS [C1]
        FROM     [dbo].[PropertyPermissions] AS [Extent14]
        INNER JOIN [dbo].[Roles] AS [Extent15] ON [Extent14].[RolesId] = [Extent15].[Id]
        INNER JOIN [dbo].[PropertyPermissionTypes] AS [Extent16] ON [Extent14].[PropertyPermissionTypesId] = [Extent16].[Id]
        INNER JOIN [dbo].[PropertyMetadata] AS [Extent17] ON [Extent14].[PropertyMetadataId] = [Extent17].[Id]
        LEFT OUTER JOIN  (SELECT
            [Extent18].[Id] AS [Id],
```

StaffingTimeslots	171939	EQUALITY: [RemovedOn] INEQUALITY: [StartTime]	3.1	35.2	16
Sessions	169677	EQUALITY: [EventId] INCLUDES: [TimeSlotId], [OnSch	4.8	58.4	6
Users	167404	EQUALITY: [FirstName]	16.8	99.8	1
SentEmails	141514	EQUALITY: [CampaignId], [Status] INCLUDES: [EmailE	1.6	96.1	9
APIProcessing	116266	EQUALITY: [EventId], [IsComplete] INCLUDES: [Recor	15.7	74.2	1

DEMO!!

Let’s see some of this in action

Key takeaways

- Cost efficiency is a platform design outcome, not a procurement exercise
-
- SQL Server, Azure SQL and Fabric each have distinct waste patterns; treat them explicitly
-
- The winning pattern is a control loop: observe, attribute, analyse, act
-
- AI becomes valuable when it turns cost data into faster architectural decisions

One action to start next week

Pick one shared analytics environment and map its top workloads, owners and active hours. That alone usually exposes the first optimisation win.

Questions?

Happy to discuss Fabric capacity strategy, Azure SQL right-sizing, SQL Server estate review, or AI-assisted FinOps operations.

Thank You

hamish@morphit.co.nz

https://www.makestuffgo.com

LinkedIn: /hamishwatson8

Sound off.
The mic is all yours.
Influence the product roadmap.

Join the Fabric User Panel

Share your feedback directly with our Fabric product group and researchers.

https://aka.ms/JoinFabricUserPanel

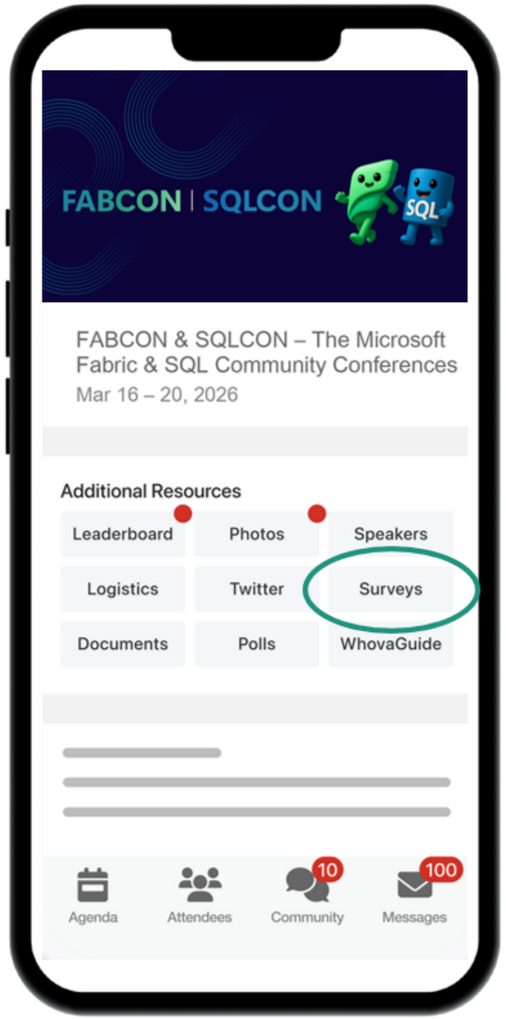
Join the SQL User Panel

Influence our SQL roadmap and ensure it meets your real-life needs

https://aka.ms/JoinSQLUserPanel

How was the session?

Complete Session Surveys in Whova for your chance to WIN PRIZES!



Get Two Fabric Certifications for FREE

Attendees of FABCON can take the Fabric Analytics Engineer or Fabric Data Engineer exam for free. Be part of the 2 fastest growing role-based certifications in Microsoft history.

Request your voucher by March 23, 2026.

https://aka.ms/fabcon/cert100