Database of the future is here – Azure SQL Hyperscale deep dive

Aditya Badramraju, **Arvind** Shyamsundar

Principal Product Managers, Microsoft Azure SQL DB

sqlbits 2024

About us

Arvind Shyamsundar

Principal Product Manager, Microsoft

@arvisam (Bluesky, Mastodon and X)

Aditya Badramraju

Principal Product Manager, Microsoft

@aditya_feb22 (X)

Agenda

- Hyperscale **architecture**: motivations, **approaches** and **implementation.**
- **Working** with Hyperscale databases.
- **Performance** monitoring and tuning.
- **Demos** to bring things to life!

Success criteria for us: leave you convinced that Hyperscale is the best cloud native database for you!

https://sqlb.it/?12734

Related sessions

- More for less: Cost optimizing your Azure SQL databases – Friday, 1:40PM
- Experts Lounge: “Meet The PG: All things Azure SQL DB” – Friday, 12PM
- Microsoft booth in the Expo floor: Wednesday, 6-8PM; Thursday, 10:30AM-1PM; Saturday, 9-11AM
- “Perfecting business continuity for Azure SQL DB” – Saturday, 10:10AM
- “SQL DB: a developer's catalyst” – happening now ☺
- “A Deep Dive into DevOps Practices with Azure SQL” – Thursday, 10:10AM

Are you...

A developer?

Currently using Azure SQL DB?

Long time SQL Server user, wanting to know more about Hyperscale?

Using other cloud DBs?

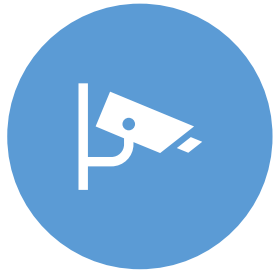
Using other DBs but mostly on-prem?

Challenges with databases

SCALE

PERFORMANCE

AVAILABILITY



SECURITY

What does “Hyperscale” mean?

Hyperscale computing

2 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

(Redirected from Hyperscale)

In computing, **hyperscale** is the ability of an architecture to scale appropriately as increased demand is added to the system.

This typically involves the ability to seamlessly provide and add compute, memory, networking, and storage resources to a given node or set of nodes that make up a larger computing, distributed computing, or grid computing environment. Hyperscale computing is necessary in order to build a robust and scalable cloud, big data, map reduce, or distributed storage system and is often associated with the infrastructure required to run large distributed sites such as Google, Facebook, Twitter, Amazon, Microsoft, IBM Cloud or Oracle. Companies like Ericsson, AMD, and Intel provide hyperscale

Hyperscale in Azure SQL

Hyperscale (https://aka.ms/hs) is the trusted and well-known SQL Server engine, rearchitected for the cloud

- Distributed architecture, with independent services for storage and compute
- Scalable from 10 GB to 100TB and from 1 core to 128 vCore
- Multi-tiered cache architecture for great performance, no matter the database size
- Scalable (out) up to 30 secondary read-only replicas

"The only cloud relational database you will need": use Hyperscale for all new applications on Azure SQL DB, no matter the size.

Microsoft

Architecting for scale-out

Shared Nothing

Log Shipping (Stand By Replica)

Always On availability groups

Shared Storage

Failover clusters (but then, that’s not really scale out…)

Architecture “S”

Virtually unlimited read scale without data duplication

Storage capacity not being bounded by any single machine

Socrates: The New SQL Server in the Cloud

Panagiotis Antonopoulos, Alex Budovski, Cristian Diaconu, Alejandro Hernandez Saenz, Jack Hu, Hanuma Kodavalla, Donald Kossmann, Sandeep Lingam, Umar Farooq Minhas, Naveen Prakash, Vijendra Purohit, Hugh Qu, Chaitanya Sreenivas Ravella, Krystyna Reisteter, Sheetal Shrotri, Dixin Tang, Vikram Wakade
Microsoft Azure & Microsoft Research

ABSTRACT

The database-as-a-service paradigm in the cloud (DBaaS) is becoming increasingly popular. Organizations adopt this paradigm because they expect higher security, higher availability, and lower and more flexible cost with high performance. It has become clear, however, that these expectations cannot be met in the cloud with the traditional, monolithic database architecture. This paper presents a novel DBaaS architecture, called Socrates. Socrates has been implemented in Microsoft SQL Server and is available in Azure as SQL DB Hyperscale. This paper describes the key ideas and features of Socrates, and it compares the performance of Socrates with the previous SQL DB offering in Azure.

CCS CONCEPTS

• **Information systems → DBMS engine architectures**;

KEYWORDS

Database as a Service, Cloud Database Architecture, High Availability

ACM Reference Format:
Panagiotis Antonopoulos, Alex Budovski, Cristian Diaconu, Alejandro Hernandez Saenz, Jack Hu, Hanuma Kodavalla, Donald Kossmann, Sandeep Lingam,, Umar Farooq Minhas, Naveen Prakash, Vijendra Purohit, Hugh Qu, Chaitanya Sreenivas Ravella, Krystyna Reisteter, Sheetal Shrotri, Dixin Tang, Vikram Wakade. 2019. Socrates: The New SQL Server in the Cloud. In *2019 International Conference on Management of Data (SIGMOD '19), June 30-July 5, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3299869.3314047

SIGMOD '19, June 30-July 5, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5643-5/19/06…$15.00
https://doi.org/10.1145/3299869.3314047

1 INTRODUCTION

The cloud is here to stay. Most start-ups are cloud-native. Furthermore, many large enterprises are moving their data and workloads into the cloud. The main reasons to move into the cloud are security, time-to-market, and a more flexible "pay-as-you-go" cost model which avoids overpaying for under-utilized machines. While all these reasons are compelling, the expectation is that a database runs in the cloud at least as well as (if not better) than on premise. Specifically, customers expect a "database-as-a-service" to be highly available (e.g., 99.999% availability), support large databases (e.g., a 100TB OLTP database), and be highly performant. Furthermore, the service must be elastic and grow and shrink with the workload so that customers can take advantage of the pay-as-you-go model.

It turns out that meeting all these requirements is not possible in the cloud using the traditional monolithic database architecture. One issue is cost elasticity which never seemed to have been a consideration for on-premise database deployments: It can be prohibitively expensive to move a large database from one machine to another machine to support a higher or lower throughput and make the best use of the computing resources in a cluster. Another, more subtle issue is that there is a conflict between the goal to support large transactional databases and high availability: High availability requires a small mean-time-to-recovery which traditionally could only be achieved with a small database. This issue does not arise in on-premise database deployments because these deployments typically make use of special, expensive hardware for high availability (such as storage area networks or SANs); hardware which is not available in the cloud. Furthermore, on-premise deployments control the software update cycles and carefully plan downtimes; this planning is typically not possible in the cloud.

To address these challenges, there has been research on new OLTP database system architectures for the cloud over the last ten years; e.g., [5, 8, 16, 17]. One idea is to decompose the functionality of a database management system and deploy the compute services (e.g., transaction processing) and storage services (e.g., checkpointing and recovery) independently. The first commercial system that adopted this idea is Amazon Aurora [20].

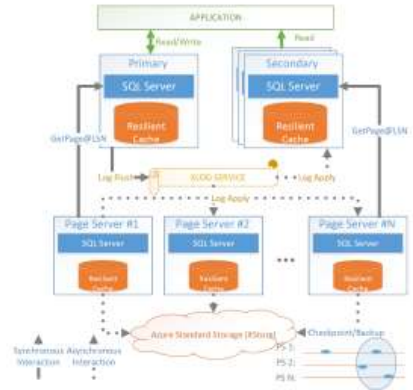


Figure 2: Socrates Architecture

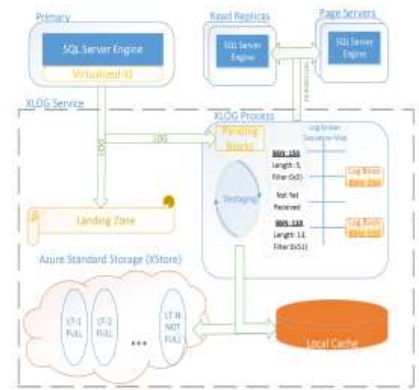


Figure 3: XLOG Service

nodes cache data pages in main memory and on SSD in a resilient buffer pool extension (Section 3.3).

The second tier of the Socrates architecture is the XLOG service. This tier implements the "separation of log" principle, motivated in Section 4.1.4. While this separation has been proposed in the literature before [6, 8], this separation of log differentiates Socrates from other cloud database systems such Amazon Aurora [20]. The XLOG service achieves low commit latencies and good scalability at the storage tier (scale-out). Since the Primary processes all updates (including DML operations), only the Primary writes to the log. This single writer approach guarantees low latency and high throughput when writing to the log. All other nodes (e.g., Secondaries) consume the log in an asynchronous way to keep their copies of data up to date.

The third tier is the storage tier. It is implemented by Page Servers. Each Page Server has a copy of a partition of the database, thereby deploying a scale-out storage architecture which, as we will see, helps to bound all operations as postulated in Section 4.1.2. Page Servers play two important roles: First, they serve pages to Compute nodes. Every Compute node can request pages from a Page Server, following a shared-disk architecture (Section 4.1.3). We are currently working on implementing bulk operations such as bulk loading, index creation, DB reorgs, deep page repair, and table scans in Page Servers to further offload Compute nodes as described in Section 4.1.5. In their second role, Page Servers checkpoint data pages and create backups in XStore (the fourth tier). Like Compute nodes, Page Servers keep all their data in main memory or locally attached SSDs for fast access.

The fourth tier is the Azure Storage Service (called XStore), the existing storage service provided by Azure independently of Socrates and SQL DB. XStore is a highly scalable, durable, and cheap storage service based on hard disks. Data access is remote and there are throughput and latency limits imposed by storage at that scale and price point. Separating Page Servers with locally attached, fast storage from durable, scalable, cheap storage implements the design principles outlined in Section 4.1.1.

Compute nodes and Page Servers are stateless. They can fail at any time without data loss. The "truth" of the database is stored in XStore and XLOG. XStore is highly reliable and has been used by virtually all Azure customers for many years without data loss. Socrates leverages this robustness. XLOG is a new service that we built specifically for Socrates. It has high performance requirements, must be scalable, affordable, and must never lose any data. We describe our implementation of XLOG, Compute nodes, and Page Servers in more detail next.

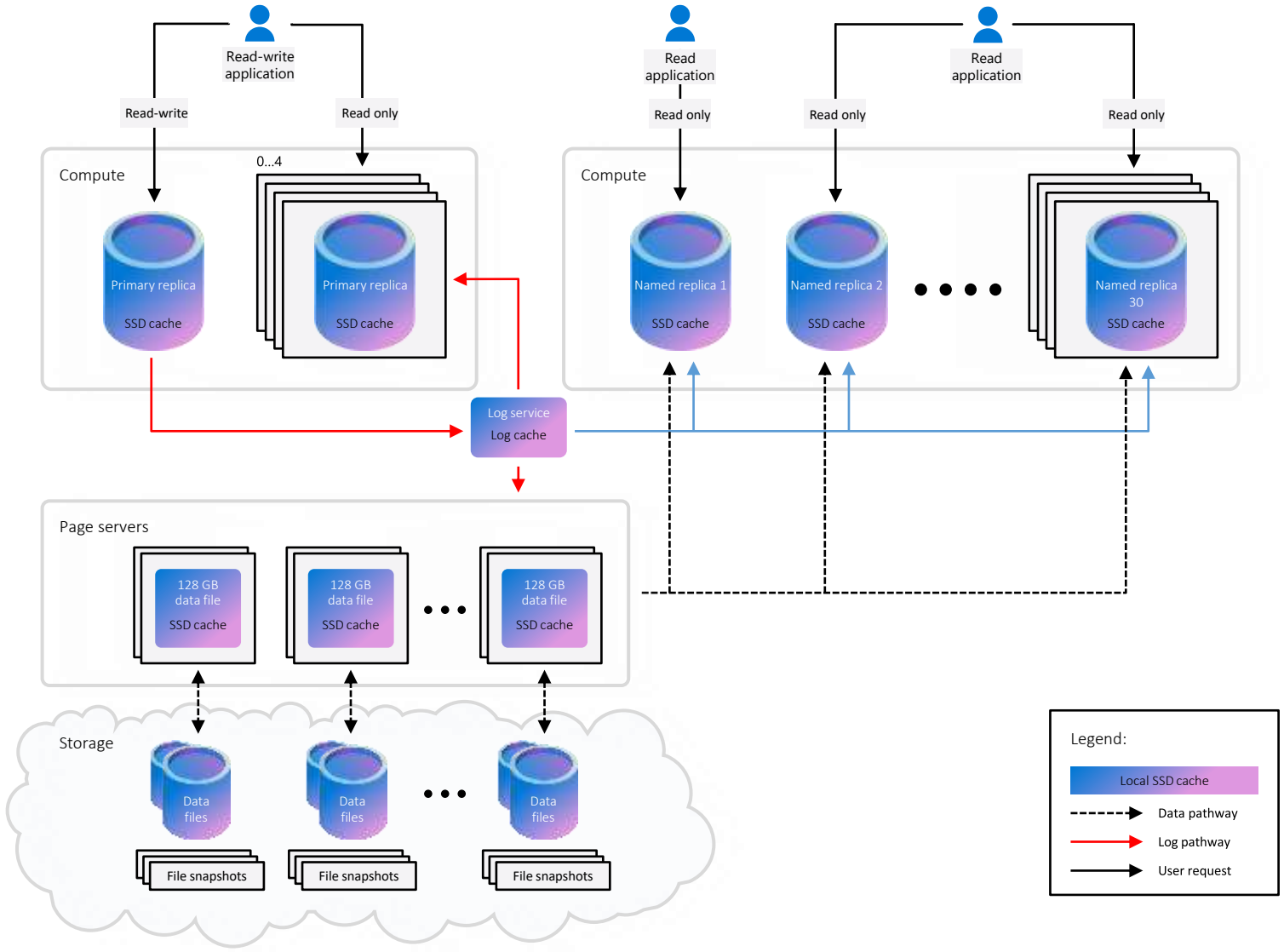
4.3 XLOG Service

Figure 3 shows the internals of the XLOG Service. Starting in the upper left corner of Figure 3, the Primary Compute node writes log blocks directly to a landing zone (LZ) which is a fast and durable storage service that provides strong guarantees on data integrity, resilience and consistency; in other words, a storage service that has SAN-like capabilities.

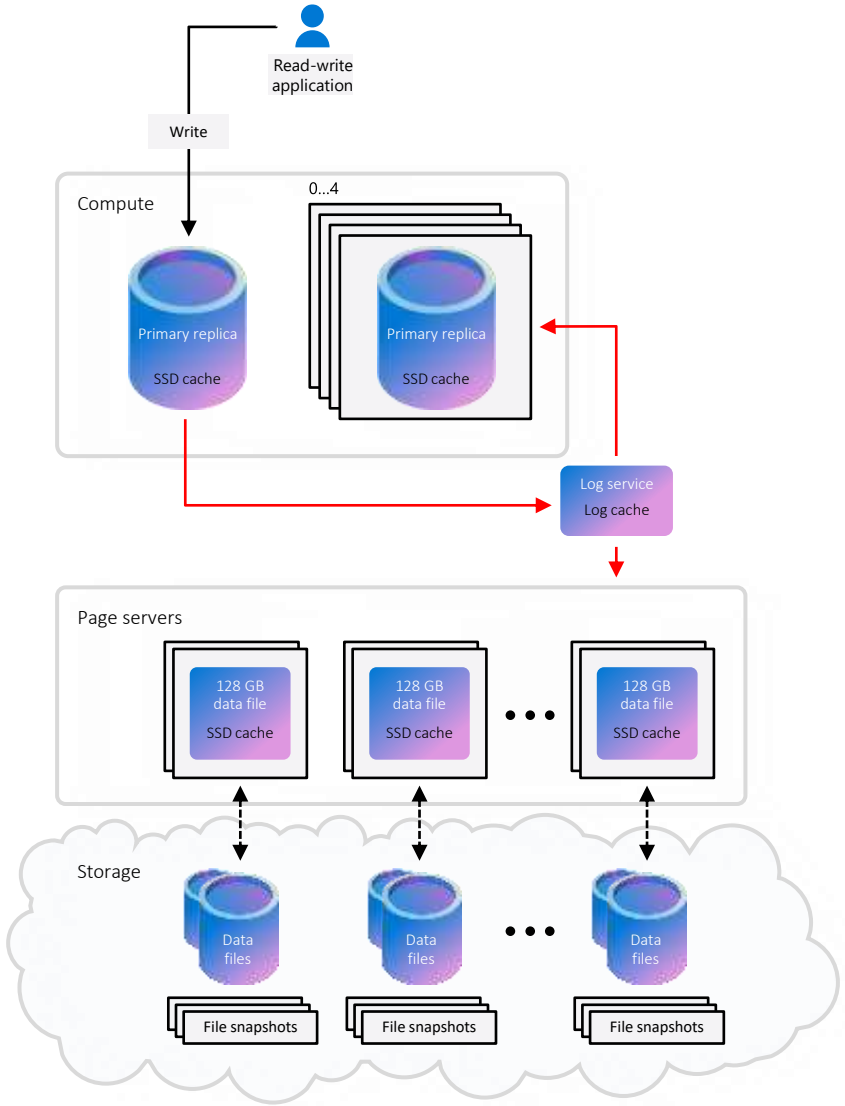
https://www.microsoft.com/en-us/research/uploads/prod/2019/05/socrates.pdf

Built-for-the cloud architecture enables near-limitless growth potential

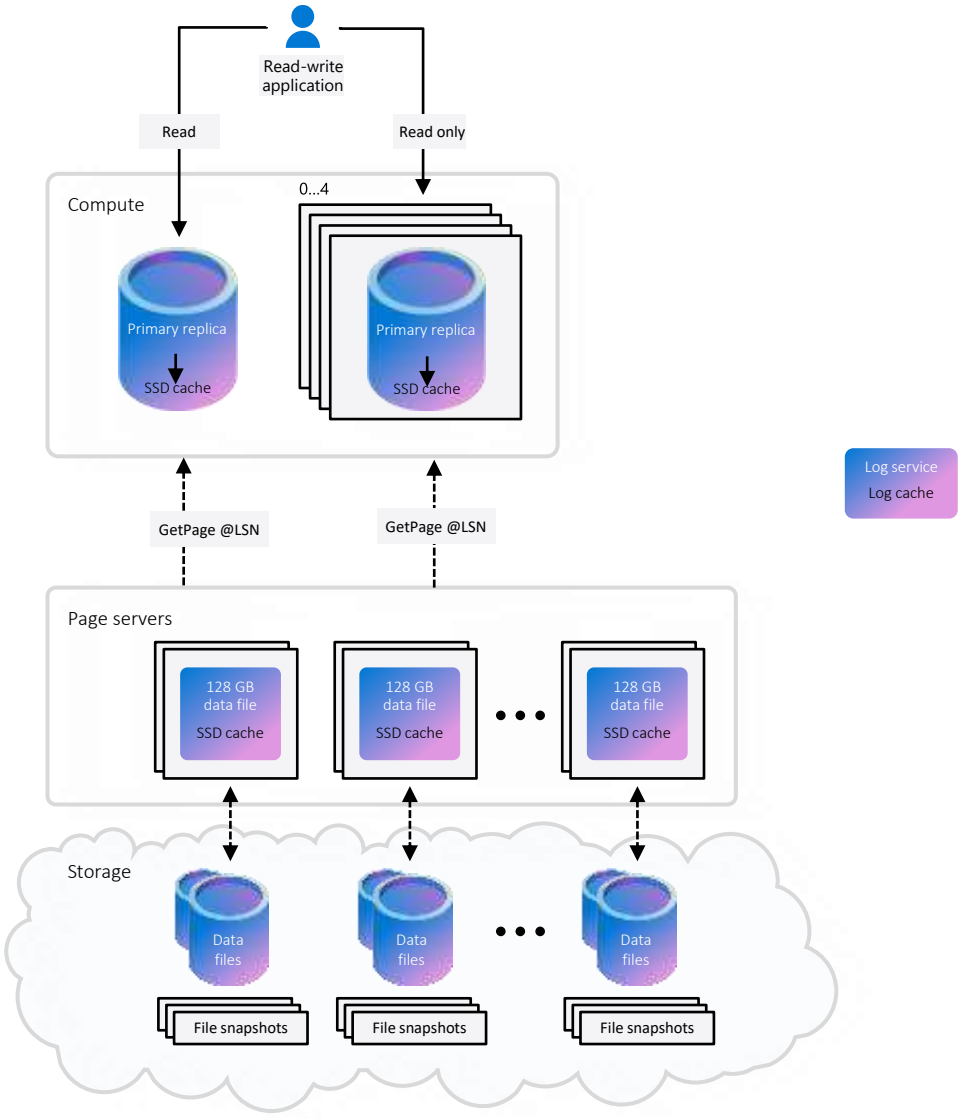
Distributed, scale-out and resilient



“Write path”


Read-write application
Write
Compute
0...4
Primary replica
SSD cache
Primary replica
SSD cache
Log service
Log cache
Page servers
128 GB data file
SSD cache
128 GB data file
SSD cache
128 GB data file
SSD cache
Storage
Data files
Data files
Data files
File snapshots
File snapshots
File snapshots

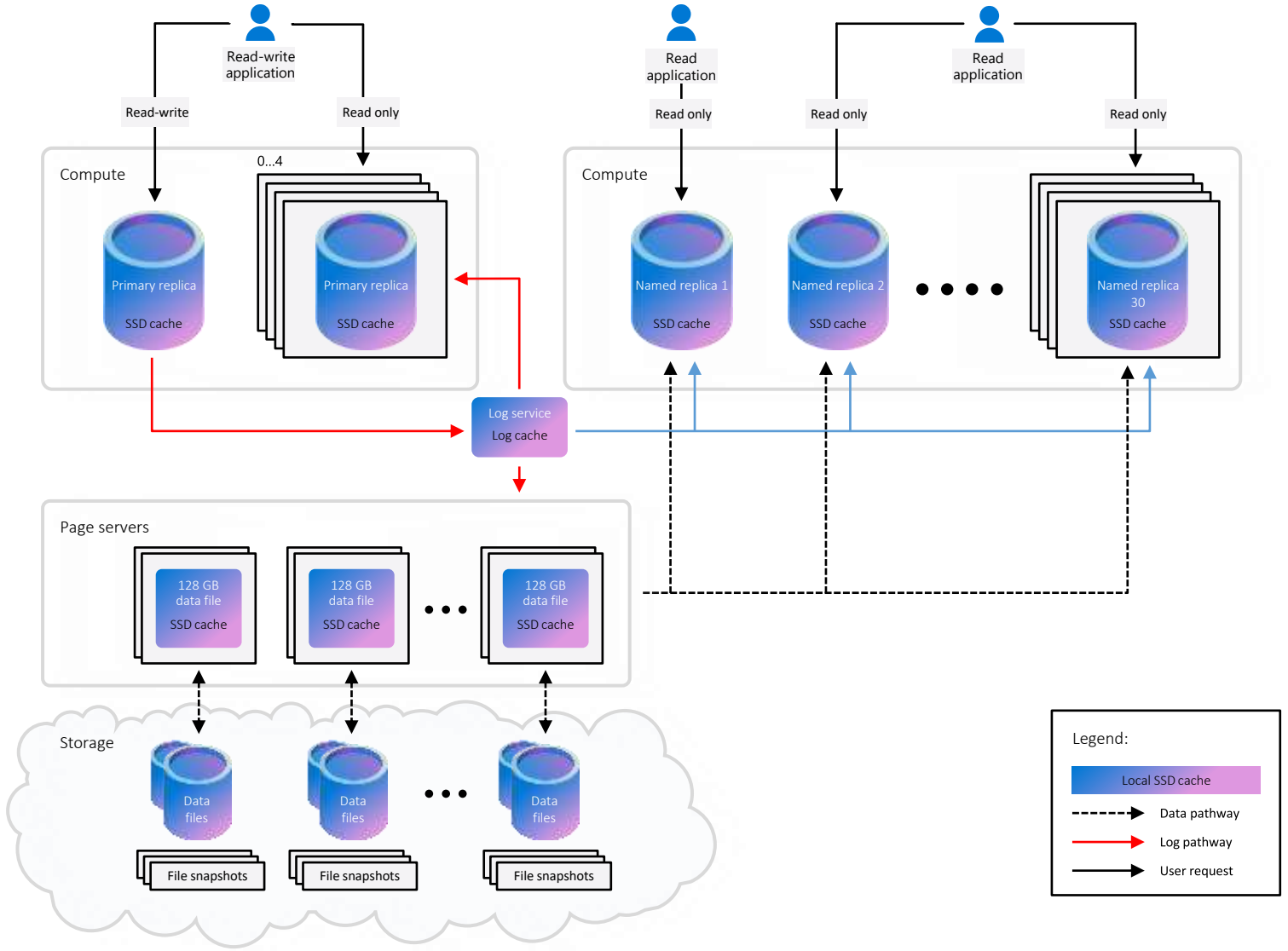
“Read path”


Read-write application
Read
Read only
Compute
0...4
Primary replica
SSD cache
Primary replica
SSD cache
GetPage @LSN
GetPage @LSN
Log service
Log cache
Page servers
128 GB data file
SSD cache
128 GB data file
SSD cache
128 GB data file
SSD cache
Storage
Data files
Data files
Data files
File snapshots
File snapshots
File snapshots

Built-for-the cloud architecture enables near-limitless growth potential

Distributed, scale-out and resilient

- Separation of compute & storage
- Smart SSD cache for compute
- Externalized log service
- Paired page servers, fully covering SSD cache
- Redundant data and log in Azure Storage
- 0 to 4 secondary HA replicas
- 0-30 named replicas for read scale
- Fast Backup/Restore via storage snapshots



Quiz time!

What is the maximum number of DB replicas (in a single server) which can be associated with the same underlying Hyperscale storage?

And the Answer is

155

https://aka.ms/hsignite2023

Announcing new lower price tier for cloud native applications on Azure SQL

Asad Khan

November 14th, 2023 | 0 | 0

Today we are announcing a new pricing tier for Azure SQL Database targeted at developers building cloud native applications. It is based on Azure SQL Database Hyperscale tier, which offers awesome cloud scalability with one of the lowest TCO in the industry. Azure SQL Database Hyperscale delivers outstanding performance and scalability through adaptive, cloud native resource management, optimized for the workload patterns of each database application. Hyperscale storage scales automatically, providing consistent performance at all scales. Azure SQL Database Hyperscale provides the best database capabilities to build cloud native applications, including abilities to create APIs for your database using Data API builder, calling external endpoints, integrating with Azure functions, and more.

Market Leading Database Capabilities

Azure SQL Database Hyperscale brings three decades of SQL innovation to developers, along with the cloud scale and the latest AI capabilities. Some salient features that differentiate SQL Hyperscale database layer from the competition include:

- **Storage snapshot** capability in Hyperscale provides fast backup and restore, that meets the aggressive requirements of cloud native workloads. **Support for 30 Named Replicas allowing different compute sizes than the primary for each replica** is not offered in any other cloud database service. This is a huge advantage for highly transactional and read-scalable workloads both in terms of performance as well as cost.

https://aka.ms/hsprice2023

Frequently Asked Questions

When does the change take effect, and what does it impact?

The pricing changes take effect on December 15th, 2023 at 00:00 hours UTC. The changes will apply to the following resources created on or after Dec 15th, 2023.

1. Any newly created Hyperscale single provisioned compute databases.
2. Any (existing or new) Hyperscale single serverless databases (currently in preview).
3. Any (existing or new) Hyperscale elastic pools (currently in preview).
4. Any newly created Hyperscale elastic pooled databases (currently in preview).

What happens to my existing Hyperscale resources?

To start with, nothing changes till December 15th, 2023. Here's what will happen starting December 15th, 2023:

1. To provide a seamless experience without any impact on existing workloads, all **existing Hyperscale single databases with provisioned compute** created before December 15th, 2023, will continue to be billed at the existing rates, for a period of up to 3 years (ending December 14th, 2026). Customers will receive further notice of the pricing change to their Hyperscale databases in advance of December 14th, 2026.
2. All **existing Hyperscale single databases with serverless compute** (currently in preview) will automatically switch to the new pricing starting December 15th, 2023.
3. All **existing Hyperscale elastic pools** (currently in preview) will automatically switch to the new pricing starting December 15th, 2023.

What if I update / change a database to a different deployment option?

Hyperscale allows seamless, rapid scaling of the database compute. You can also scale a Hyperscale database to move from provisioned compute to serverless compute (or the other way around). You can add an existing Hyperscale database to an elastic pool or move an elastic pooled database out of the elastic pool to a single database. Here's how your costs are impacted if you perform any of these changes on or after December 15th, 2023:

Change	Impact
Hyperscale (serverless single, or elastic pooled) database is changed to a Hyperscale single database with provisioned compute.	The final cost of the database will be based on when the database was created. If the database was created prior to December 15th, 2023, it will be billed as per the existing pricing. If the database was created on or after December 15th, 2023, it will be billed as per the new pricing.

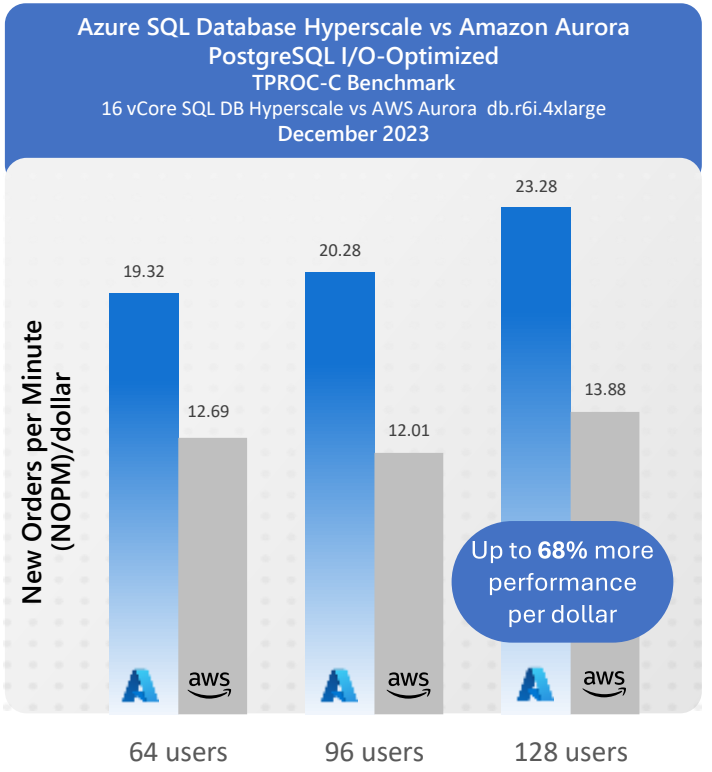
Azure SQL Database Hyperscale outperforms

Build new, highly scalable cloud applications with the performance and security of Azure SQL *for the price of commercial open-source databases*

Principled Technologies published a study where they tested throughput performance between Azure SQL Database Hyperscale and Amazon Aurora PostgreSQL.

SQL Database Hyperscale emerged as the price-performance leader for mission-critical workloads, delivering up to 68 percent more performance per dollar than Amazon Aurora PostgreSQL.[1]

Blog: https://devblogs.microsoft.com/azure-sql/build-highly-scalable-ai-ready-applications-on-azure-sql-database-hyperscale/



Database performance per dollar comparison.
Higher is better.

Price-performance claims based on data from a study commissioned by Microsoft and conducted by Principled Technologies in December 2023. The study compared performance and price performance between a 16 vCore and 32 vCore Azure SQL Database using premium-series hardware on the Hyperscale service tier and the db.r6i.4xlarge and db.r6i.8xlarge offerings for Amazon Web Services Aurora PostgreSQL I/O-Optimized (AWS Aurora). Benchmark data is taken from a Principled Technologies report which used the HammerDB TPROC-C benchmark. The TPROC-C workload is derived from the TPC-C Benchmark and results were obtained with the HammerDB TPROC-C workload . The HammerDB TPROC-C workload is derived from the TPC-C benchmark and is not comparable to published TPC-C Benchmark results, as this implementation does not comply with all requirements of the TPC Benchmark. Price-performance is calculated by Principled Technologies as the cost of running the cloud platform continuously divided by new orders per minute throughput, based upon the standard. Prices are based on publicly available US pricing in East US 1 for Azure SQL Database and US East for AWS Aurora as of December 2023. Performance and price-performance results are based upon the configurations detailed in the Principled Technologies report. Actual results and prices may vary based on configuration and region.

Demo

Getting started with Hyperscale

Home > myResourceGroup >

mylogicalserver

SQL server

Search

Overview
Activity log
Access control (IAM)
Tags
Quick start
Diagnose and solve problems

Settings

Microsoft Entra ID
SQL databases
SQL elastic pools
DTU quota
Properties
Locks

Create database | New elastic pool | New dedicated SQL pool (formerly SQL DW) | Import database | Reset password | ...

Essentials JSON View

Notifications (0) **Features (6)**

All | Security (4) | Performance (1) | Recovery (1)

Microsoft Entra admin

Allows you to centrally manage identity and access to your Azure SQL databases.

CONFIGURED

Microsoft Defender for SQL

Vulnerability Assessment and Advanced Threat Protection.

CONFIGURED

Automatic tuning

Monitors and tunes your database automatically to optimize performance.

CONFIGURED

Auditing

Track database events and writes them to an audit log in Azure storage.

NOT CONFIGURED

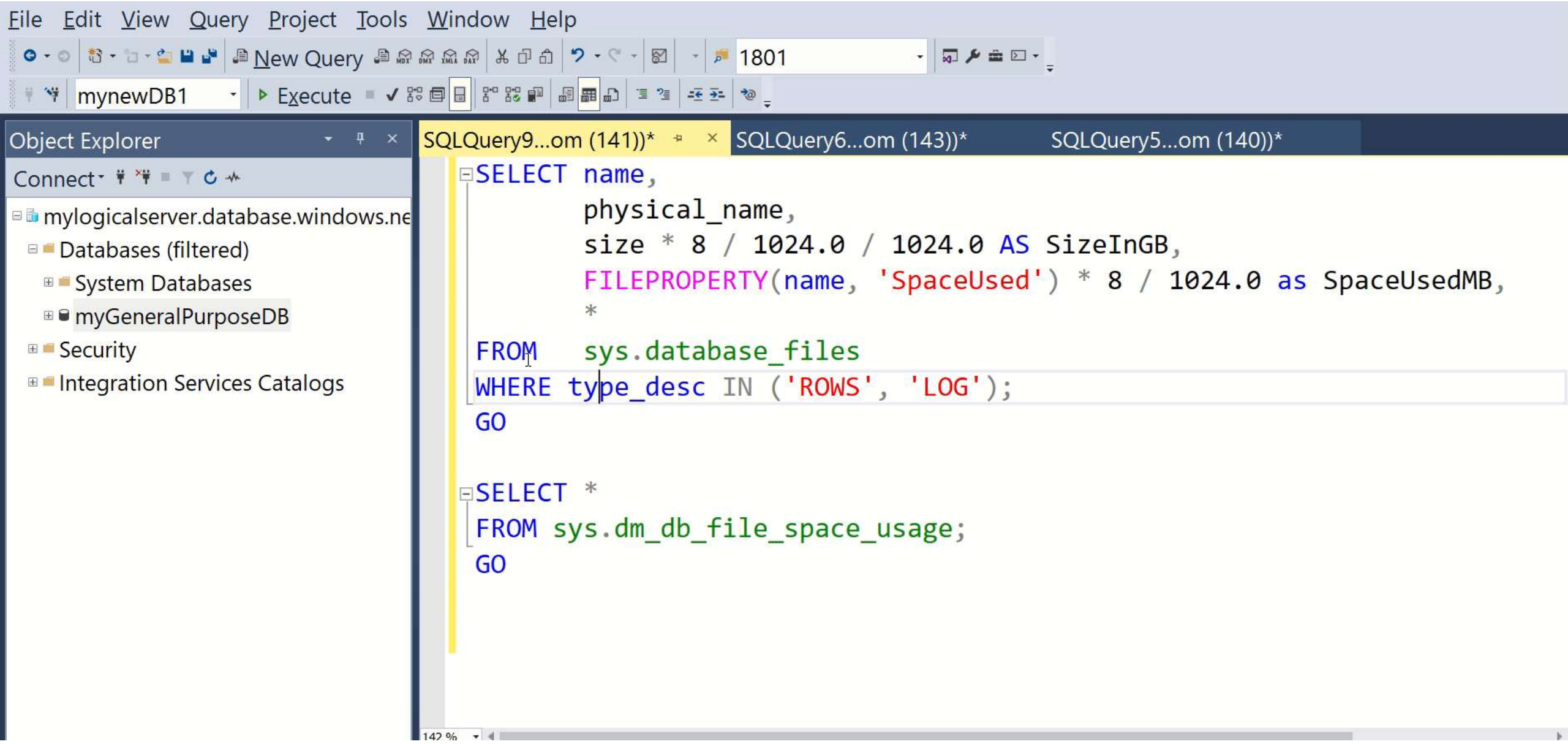
Available resources

Filter by name

All types

Demo

What’s inside a Hyperscale DB?


File Edit View Query Project Tools Window Help
New Query
1801
mynewDB1
Execute
Object Explorer
Connect
mylogicalserver.database.windows.ne
Databases (filtered)
System Databases
myGeneralPurposeDB
Security
Integration Services Catalogs
SQLQuery9...om (141))*
SQLQuery6...om (143))*
SQLQuery5...om (140))*
SELECT name,
physical_name,
size * 8 / 1024.0 / 1024.0 AS SizeInGB,
FILEPROPERTY(name, 'SpaceUsed') * 8 / 1024.0 as SpaceUsedMB,
*
FROM sys.database_files
WHERE type_desc IN ('ROWS', 'LOG');
GO
SELECT *
FROM sys.dm_db_file_space_usage;
GO
142 %

Home >

myNewDB1 (mylogicalserver/myNewDB1)

SQL database

Search

- Overview
- Activity log
- Tags
- Diagnose and solve problems
- Query editor (preview)

Settings

- Compute + storage
- Connection strings
- Maintenance
- Properties
- Locks

Data management

- Replicas

Copy | Restore | Export | Set server firewall | Delete | Connect with... | Feedback

Essentials

JSON View

Getting started | Monitoring | Properties | Features | Notifications (0) | Integrations | Tutorials

Start working with your database

Connect to your database and start working with data with a few simple steps. Learn more

Configure access

Configure network access to your SQL server. Learn more

Configure

Connect to application

Use connection strings to connect to your SQL database from your applications and favorite tools.

See connection strings

Demo

Migrating existing DBs to Hyperscale

myGeneralPurposeDB (mylogicalserver/myGeneralPurposeDB) | Compute + storage

SQL database

Search

Feedback

Overview
Activity log
Tags
Diagnose and solve problems
Query editor (preview)

Settings

Compute + storage
Connection strings
Maintenance
Properties
Locks

Data management

Replicas
Sync to other databases

Integrations

Azure Synapse Link
Stream analytics (preview)
Add Azure AI Search

Service tier: General Purpose (Most budget friendly)

Compare service tiers

Compute tier

- **Provisioned** - Compute resources are pre-allocated. Billed per hour based on vCores configured.
- **Serverless** - Compute resources are auto-scaled. Billed per second based on vCores used.

Compute Hardware

Select the hardware configuration based on your workload requirements. Availability of compute optimized, memory optimized, and confidential computing hardware depends on the region, service tier, and compute tier.

Hardware Configuration

Standard-series (Gen5)
up to 128 vCores, up to 625 GB memory
Change configuration

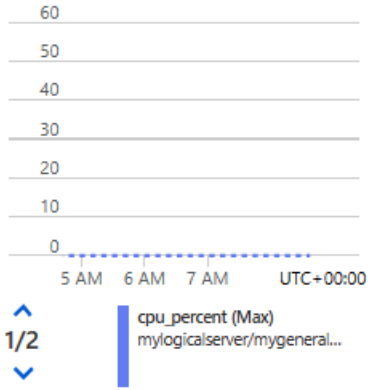
Save money

Already have a SQL Server License? Save with a license you already own with Azure Hybrid Benefit. Actual savings may vary based on region and performance tier. Learn more

Yes / No

vCores Compare vCore options — 24

Data max size (GB) — 4096



Cost summary

General Purpose (GP_Gen5_24)

Cost per **vCore** (in USD)	184.09
vCores selected	x 24
Cost per **GB** (in USD)	0.12
Max storage selected (in GB)	x 5120
ESTIMATED COST / MONTH	5007.05 USD

myGeneralPurposeDB (mylogicalserver/myGeneralPurposeDB) | Compute + storage

SQL database

Search

Overview
Activity log
Tags
Diagnose and solve problems
Query editor (preview)
Resource visualizer

Settings
Compute + storage
Connection strings
Maintenance
Properties
Locks

Data management
Replicas
Sync to other databases

Integrations
Azure Synapse Link
Stream analytics (preview)

Feedback

Service and compute tier

Select from the available tiers based on the needs of your workload. The vCore model provides a wide range of configuration controls and offers Hyperscale and Serverless to automatically scale your database based on your workload needs. Alternately, the DTU model provides set price/performance packages to choose from for easy configuration. Learn more

SQL Database Hyperscale: Low price, high scalability, and best feature set. Learn more

Service tier: General Purpose (Most budget friendly)

Compare service tiers

Compute tier

- (●) **Provisioned** - Compute resources are pre-allocated. Billed per hour based on vCores configured.
- ( ) **Serverless** - Compute resources are auto-scaled. Billed per second based on vCores used.

Compute Hardware

Select the hardware configuration based on your workload requirements. Availability of compute optimized, memory optimized, and confidential computing hardware depends on the region, service tier, and compute tier.

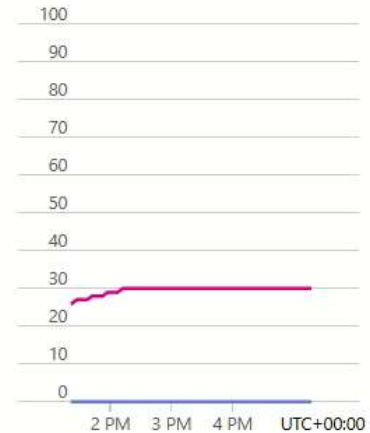
Hardware Configuration

emory

Save money

Apply

Database utilization



1/2

cpu_percent (Max)
mylogicalserver/mygeneral...

Cost summary

General Purpose (GP_Gen5_24)

Cost per **vCore** (in USD)	184.09
vCores selected	x 24
Cost per **GB** (in USD)	0.12
Max storage selected (in GB)	x 5120

Demo

Predictably quick scaling with Hyperscale

Home >

myNewDB1 (mylogicalserver/myNewDB1)

SQL database

Search

- Overview
- Activity log
- Tags
- Diagnose and solve problems
- Query editor (preview)

Settings

- Compute + storage
- Connection strings
- Maintenance
- Properties
- Locks

Data management

- Replicas

Copy | Restore | Export | Set server firewall | Delete | Connect with... | Feedback

Essentials

JSON View

Getting started | Monitoring | Properties | Features | Notifications (0) | Integrations | Tutorials

Start working with your database

Connect to your database and start working with data with a few simple steps. Learn more

Configure access

Configure network access to your SQL server. Learn more

Configure

Connect to application

Use connection strings to connect to your SQL database from your applications and favorite tools.

See connection strings

Home >

myNewDB1 (mylogicalserver/myNewDB1)

SQL database

Search

- Overview
- Activity log
- Tags
- Diagnose and solve problems
- Query editor (preview)

Settings

- Compute + storage
- Connection strings
- Maintenance
- Properties
- Locks

Data management

- Replicas

Copy | Restore | Export | Set server firewall | Delete | Connect with... | Feedback

Scaling from HS_Gen5_10 to HS_PRMS_8 database level See more

Essentials JSON View

Getting started | Monitoring | Properties | Features | Notifications (1) | Integrations | Tutorials

Start working with your database

Connect to your database and start working with data with a few simple steps. Learn more

Configure access

Configure network access to your SQL server. Learn more

Configure

Connect to application

Use connection strings to connect to your SQL database from your applications and favorite tools.

See connection strings

Home >

myNewDB1 (mylogicalserver/myNewDB1)

SQL database

Search

- Overview
- Activity log
- Tags
- Diagnose and solve problems
- Query editor (preview)

Settings

- Compute + storage
- Connection strings
- Maintenance
- Properties
- Locks

Data management

- Replicas

Copy | Restore | Export | Set server firewall | Delete | Connect with... | Feedback

Scaling operation completed

Essentials JSON View

Getting started | Monitoring | Properties | Features | Notifications (0) | Integrations | Tutorials

Start working with your database

Connect to your database and start working with data with a few simple steps. Learn more

Configure access

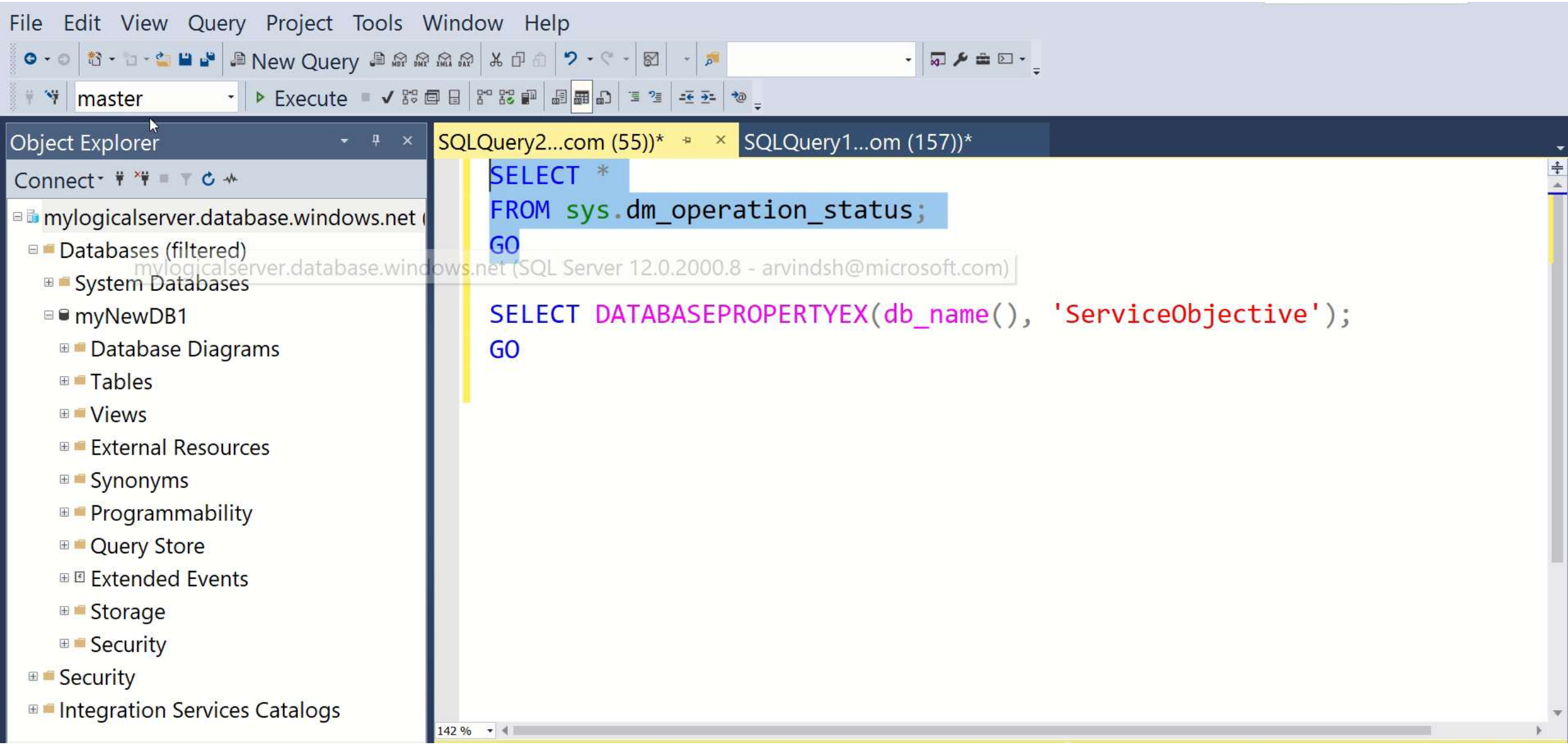
Configure network access to your SQL server. Learn more

Configure

Connect to application

Use connection strings to connect to your SQL database from your applications and favorite tools.

See connection strings


File Edit View Query Project Tools Window Help
New Query
master
Execute
Object Explorer
Connect
mylogicalserver.database.windows.net
Databases (filtered)
System Databases
myNewDB1
Database Diagrams
Tables
Views
External Resources
Synonyms
Programmability
Query Store
Extended Events
Storage
Security
Security
Integration Services Catalogs
SQLQuery2...com (55))*
SQLQuery1...om (157))*
SELECT *
FROM sys.dm_operation_status;
GO
mylogicalserver.database.windows.net (SQL Server 12.0.2000.8 - arvindsh@microsoft.com)
SELECT DATABASEPROPERTYEX(db_name(), 'ServiceObjective');
GO
142 %

Demo

Published characteristics (“limits”) of a Hyperscale DB

Hyperscale - provisioned compute - premium-series

Hyperscale premium-series (part 1 of 3)

Compute sizes (service level objectives, or SLOs) for Hyperscale premium-series databases follow the naming convention `HS_PRMS_` followed by the number of vCores.

The following table covers these SLOs: `HS_PRMS_2`, `HS_PRMS_4`, `HS_PRMS_6`, `HS_PRMS_8`, and `HS_PRMS_10`:

Expand table

vCores	2	4	6	8	10
Hardware	Premium-series	Premium-series	Premium-series	Premium-series	Premium-series
Memory (GB)	10.4	20.8	31.1	41.5	51.9
Columnstore support	Yes	Yes	Yes	Yes	Yes
In-memory OLTP storage (GB)	N/A	N/A	N/A	N/A	N/A
Max data size (TB)	100	100	100	100	100
Max log size (TB)	Unlimited	Unlimited	Unlimited	Unlimited	Unlimited
Tempdb max data size (GB)	64	128	192	256	320
Max local SSD IOPS [1]	8,000	16,000	24,000	32,000	40,000
Max log rate (MBps)	100	100	100	100	100
Local read IO latency[2]	1-2 ms	1-2 ms	1-2 ms	1-2 ms	1-2 ms
Remote read IO latency[2]	1-4 ms	1-4 ms	1-4 ms	1-4 ms	1-4 ms
Write IO latency[2]	1-4 ms	1-4 ms	1-4 ms	1-4 ms	1-4 ms
Storage type	Multi-tiered[3]	Multi-tiered[3]	Multi-tiered[3]	Multi-tiered[3]	Multi-tiered[3]

https://aka.ms/hslimits

Resource management in Azure SQL Database

Article • 12/28/2023 • 16 contributors

Feedback

In this article

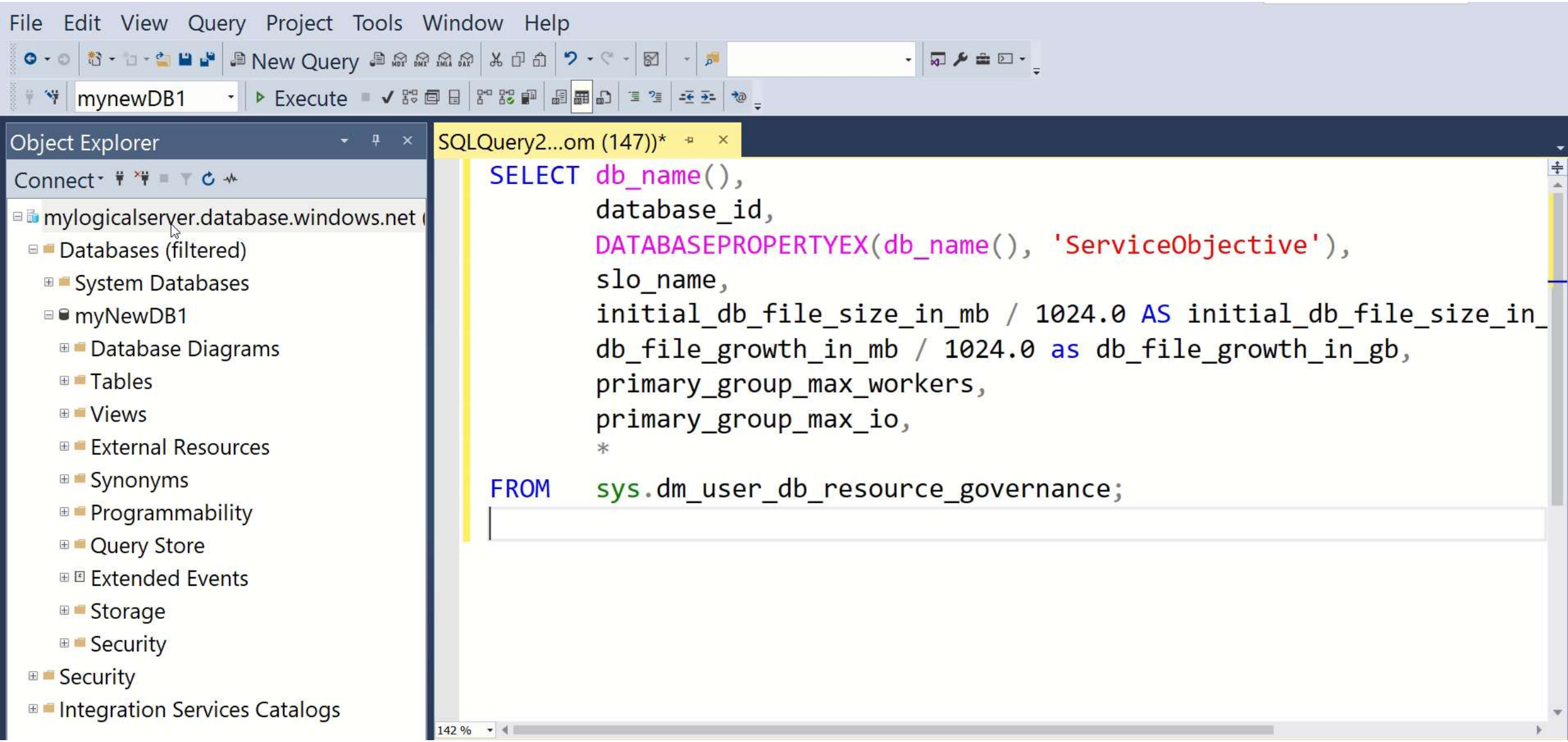
Applies to: Azure SQL Database

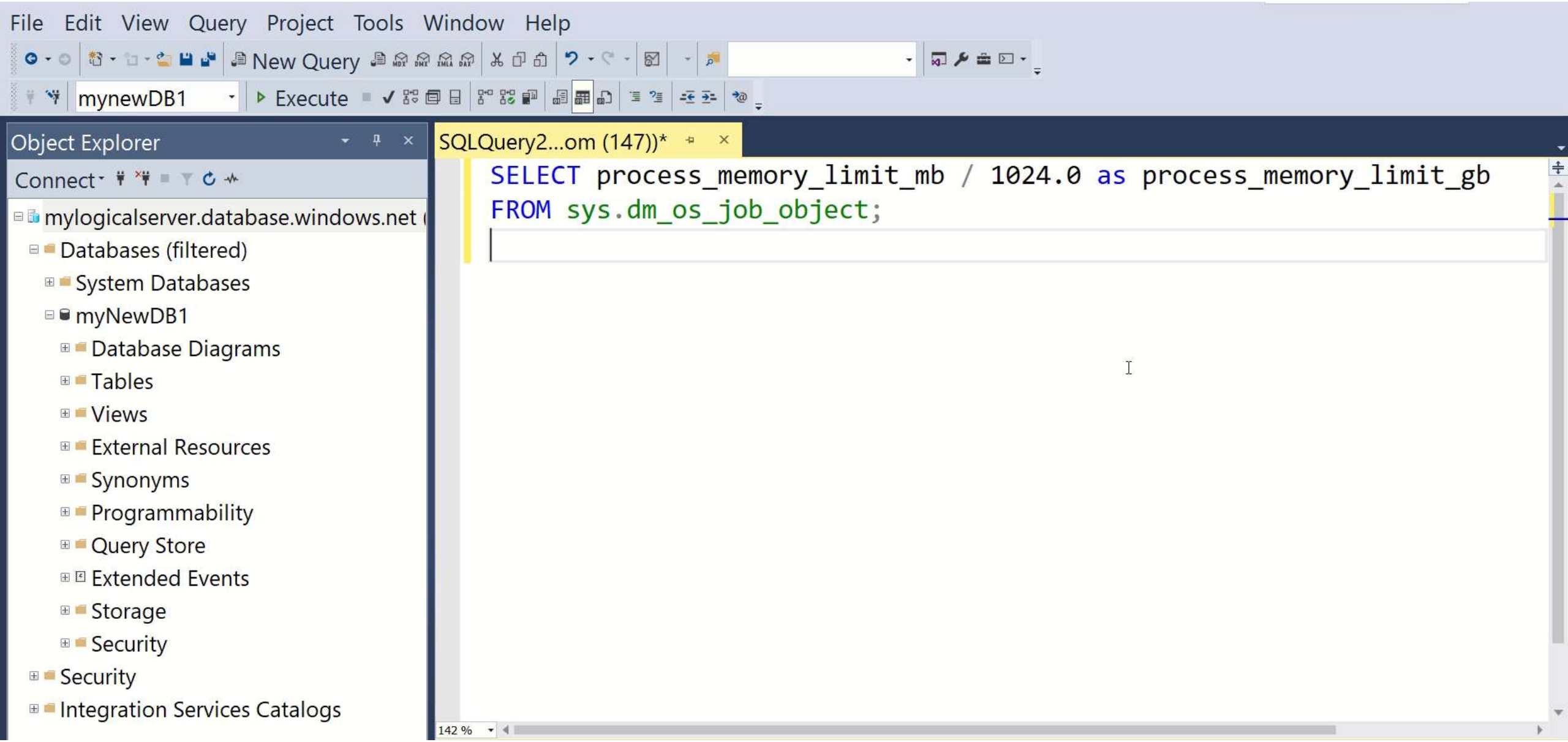
Azure SQL Database logical server

This article provides an overview of resource management in Azure SQL Database. It provides information on what happens when resource limits are reached, and describes resource governance mechanisms that are used to enforce these limits.

For specific resource limits per pricing tier (also known as service objective) for single databases, refer to either DTU-based single database resource limits or vCore-based single database resource limits. For elastic pool resource limits, refer to either DTU-based elastic pool resource limits or vCore-based elastic pool resource limits. For Azure Synapse Analytics dedicated SQL pool limits, see capacity limits and memory and concurrency limits.

https://aka.ms/sqldbrg


File Edit View Query Project Tools Window Help
New Query
mynewDB1
Execute
Object Explorer
Connect
mylogicalserver.database.windows.net
Databases (filtered)
System Databases
myNewDB1
Database Diagrams
Tables
Views
External Resources
Synonyms
Programmability
Query Store
Extended Events
Storage
Security
Security
Integration Services Catalogs
SQLQuery2...om (147))*
SELECT db_name(),
database_id,
DATABASEPROPERTYEX(db_name(), 'ServiceObjective'),
slo_name,
initial_db_file_size_in_mb / 1024.0 AS initial_db_file_size_in_
db_file_growth_in_mb / 1024.0 as db_file_growth_in_gb,
primary_group_max_workers,
primary_group_max_io,
*
FROM sys.dm_user_db_resource_governance;
142 %


File Edit View Query Project Tools Window Help
New Query
mynewDB1
Execute
Object Explorer
Connect
mylogicalserver.database.windows.net
Databases (filtered)
System Databases
myNewDB1
Database Diagrams
Tables
Views
External Resources
Synonyms
Programmability
Query Store
Extended Events
Storage
Security
Security
Integration Services Catalogs
SQLQuery2...om (147))*
SELECT process_memory_limit_mb / 1024.0 as process_memory_limit_gb
FROM sys.dm_os_job_object;
142 %

Hyperscale FAQs

Why decouple compute and storage?

Why this complex architecture?

Why is the log generation rate for Hyperscale limited to 100 MB / sec?

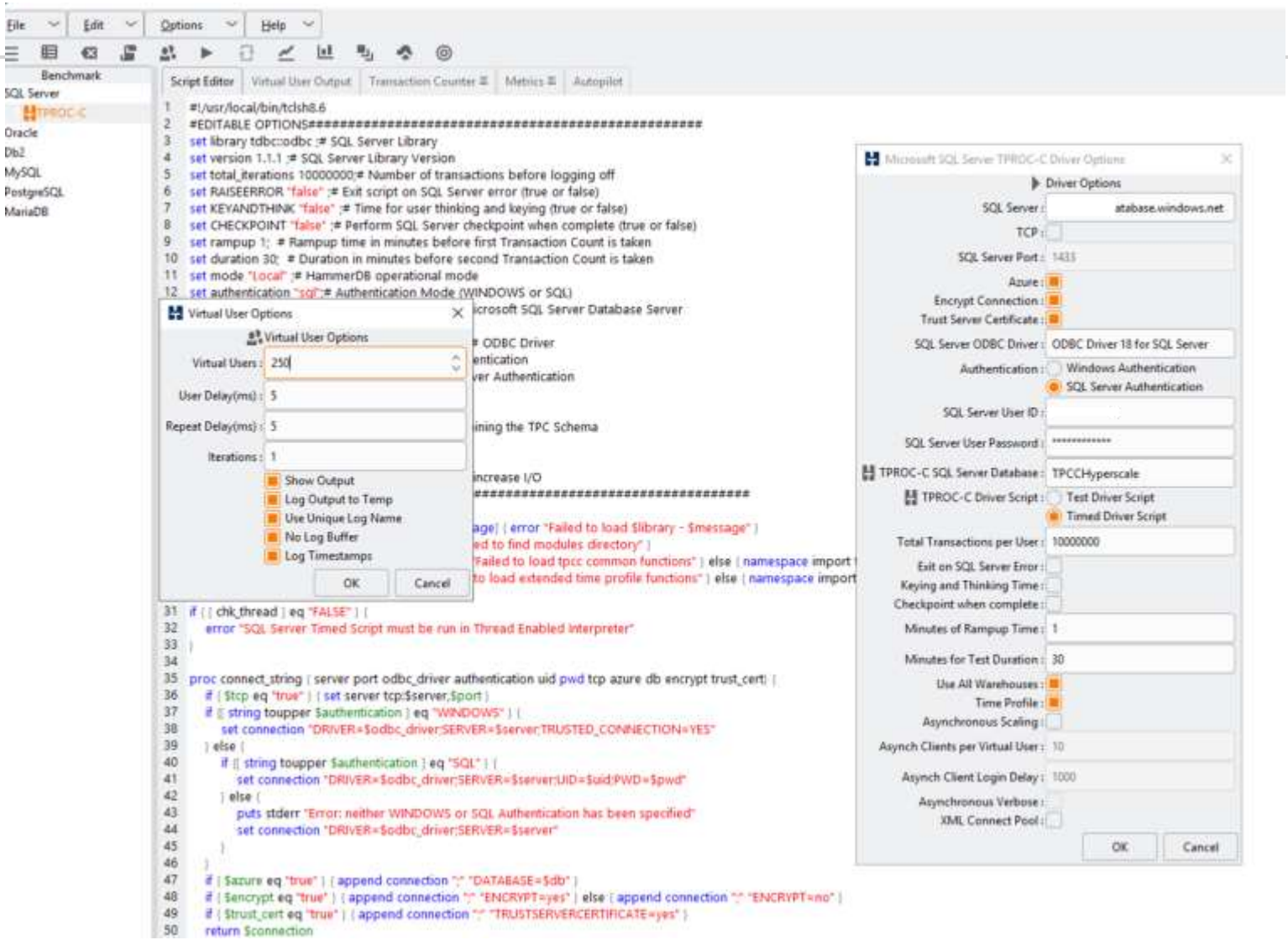
Why can’t I restore a backup of a Hyperscale DB elsewhere?

What does a PAGEIOLATCH_ * wait mean in a Hyperscale database?

Does adding more vCores in Hyperscale always lead to better performance?

Let’s talk performance tuning

Specifics of workload



Specifics
SQL DB HS - vCore -16
Client vCore- 16
DB size- 3.3 TB

Demo

sys.dm_db_resource stats

SQLQuery5...ram (243))* | SQLQuery4...ram (810))* | SQLQuery3...ram (240))* | SQLQuery2...ram (417))* | SQLQuery1...ram (177))*

```
DECLARE @db_max_log_rate AS FLOAT = (SELECT primary_max_log_rate
                                     FROM   sys.dm_user_db_resource_governance
                                     WHERE  database_id = db_id());

SELECT   TOP 1 used_storage_mb,
               used_storage_mb / 1024.0 / 1024.0 AS UsedSpaceTiB,
               allocated_storage_mb,
               allocated_storage_mb / 1024.0 / 1024.0 AS AllocatedSpaceTiB,
                avg_data_io_percent,
               avg_log_write_percent,
               avg_log_write_percent * @db_max_log_rate / (100.0 * 1024.0 * 1024.0) AS LogWriteMiBPerSec,
               avg_cpu_percent AS avg_cpu_percent
FROM     sys.dm_db_resource_stats
ORDER BY end_time DESC;
```

150 %

Query executed successfully.

adbadram.database.windows.n... | adbadram (243) | TpccHyperscale | 00:00:00 | 1 rows

PAGEIOLATCH_*

PAGEIOLATCH_EX	Occurs when a task is waiting on a latch for a buffer that is in an I/O request. The latch request is in Exclusive mode - a mode used when the buffer is being written to disk. Long waits might indicate problems with the disk subsystem. For more information, see Slow I/O - SQL Server and disk I/O performance.

SQLQuery3...ram (240))* | SQLQuery2...ram (417))* | SQLQuery1...ram (177))*

```
SELECT    wait_type,
          wait_duration_ms,
          resource_description
FROM      sys.dm_os_waiting_tasks AS WT
          INNER JOIN
          sys.dm_exec_sessions AS S
          ON WT.session_id = S.session_id
WHERE     S.is_user_process = 1


go

select * from sys.databases where database_id=
go
sp_helpfile
```

150 %

Query executed successfully.

adbadram.database.windows.n... | adbadram (240) | TpccHyperscale | 00:00:00 | 59 rows

Microsoft

PAGEIOLATCH_* in context of Hyperscale

- Same definition as in SQL Server.
- Indicates remote IO to the Page Servers
- Occurs because of reasons such as:
  - BPool + RBPEX smaller than what the workload ideally needs.
  - Higher transaction rate which is slowing down Redo in Pageservers.

WRITELOG

`WRITELOG`	Occurs while waiting for a log flush to complete. Common operations that cause log flushes are transaction commits and checkpoints. Common reasons for long waits on `WRITELOG` are: disk latency (where transaction log files reside), the inability for I/O to keep up with transactions, or, a large number of transaction log operations and flushes (commits, rollback)

SQLQuery3...ram (240))* | SQLQuery2...ram (417))* | SQLQuery1...ram (177))*

```
SELECT    wait_type,
          wait_duration_ms,
          resource_description
FROM      sys.dm_os_waiting_tasks AS WT
          INNER JOIN
          sys.dm_exec_sessions AS S
          ON WT.session_id = S.session_id
WHERE     S.is_user_process = 1
and wait_type like '%write%'
go
```

150 %

Query executed successfully.

adbadram.database.windows.n... | adbadram (240) | TpccHyperscale | 00:00:00 | 14 rows

Microsoft

WRITELOG in context of Hyperscale

- Same definition as in SQL Server.
- Here the transaction log that is written to is the “XLOG service”.
- This occurs in Hyperscale only if
  - Higher commits on the application
  - There are any bottlenecks with the Log Service:
    - This should be only for exceptional scenarios.
    - Usually, we’ll know well before you know if there is such a bottleneck with the Log Service.
    - And if that’s the case, we (Microsoft) will internally resolve it.

Quiz time!

Would you see a LOG_RATE_GOVERNOR wait type in the on-prem world?

SQLQuery4....ecuting...* | SQLQuery3...ram (240))* | SQLQuery2...ram (417))* | SQLQuery1...ram (177))*

```
alter index PK_Stock on stock rebuild
```

150 %

Results | Messages

Executing query...

adbadram.database.windows.n... | adbadram (810) | TpccHyperscale | 00:01:43 | 0 rows

Home > TpccHyperscale (adbadram/TpccHyperscale) >

Metrics

Azure Monitoring

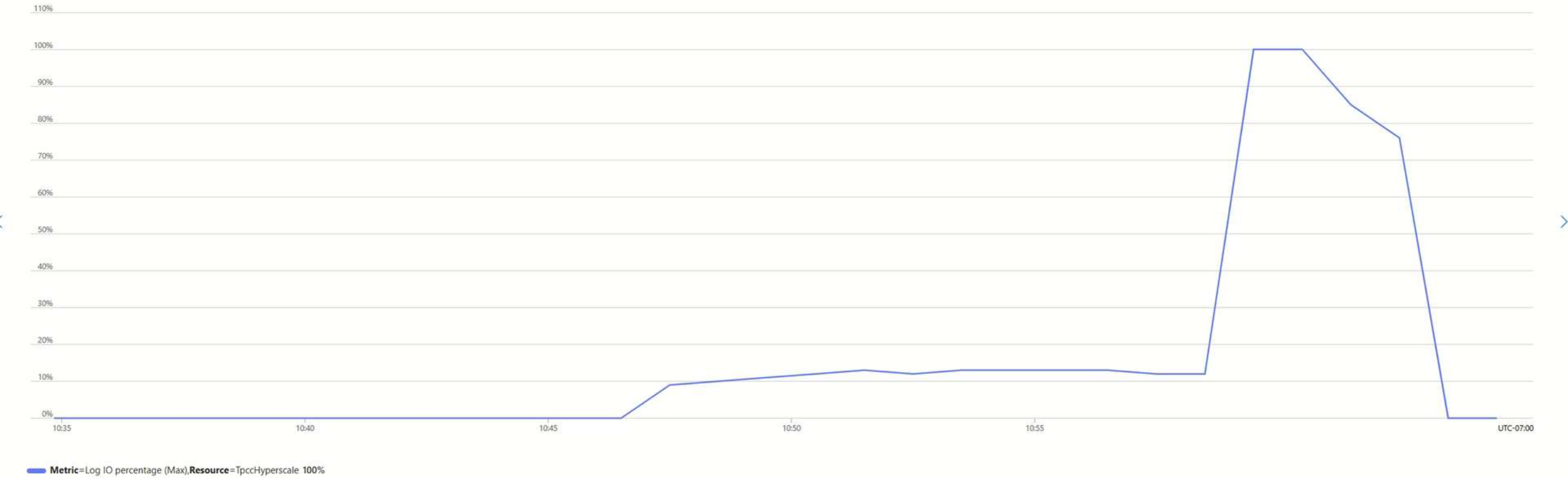
New chart | Refresh | Share | Feedback

Local Time: Last 30 minutes (1 minute)

Compute utilization

Add metric | Add filter | Apply splitting | Line chart | Drill into Logs | New alert rule | Save to dashboard

adbadram/TpccHyperscale, **Log IO percentage,** Max



Metric=Log IO percentage (Max),**Resource**=TpccHyperscale **100%**

Azure SQL DB Hyperscale serverless

GA !!

Automatic scaling of compute and memory; log throughput independent of compute

Auto-scaling independence of the primary replica, high availability replicas, and named replicas

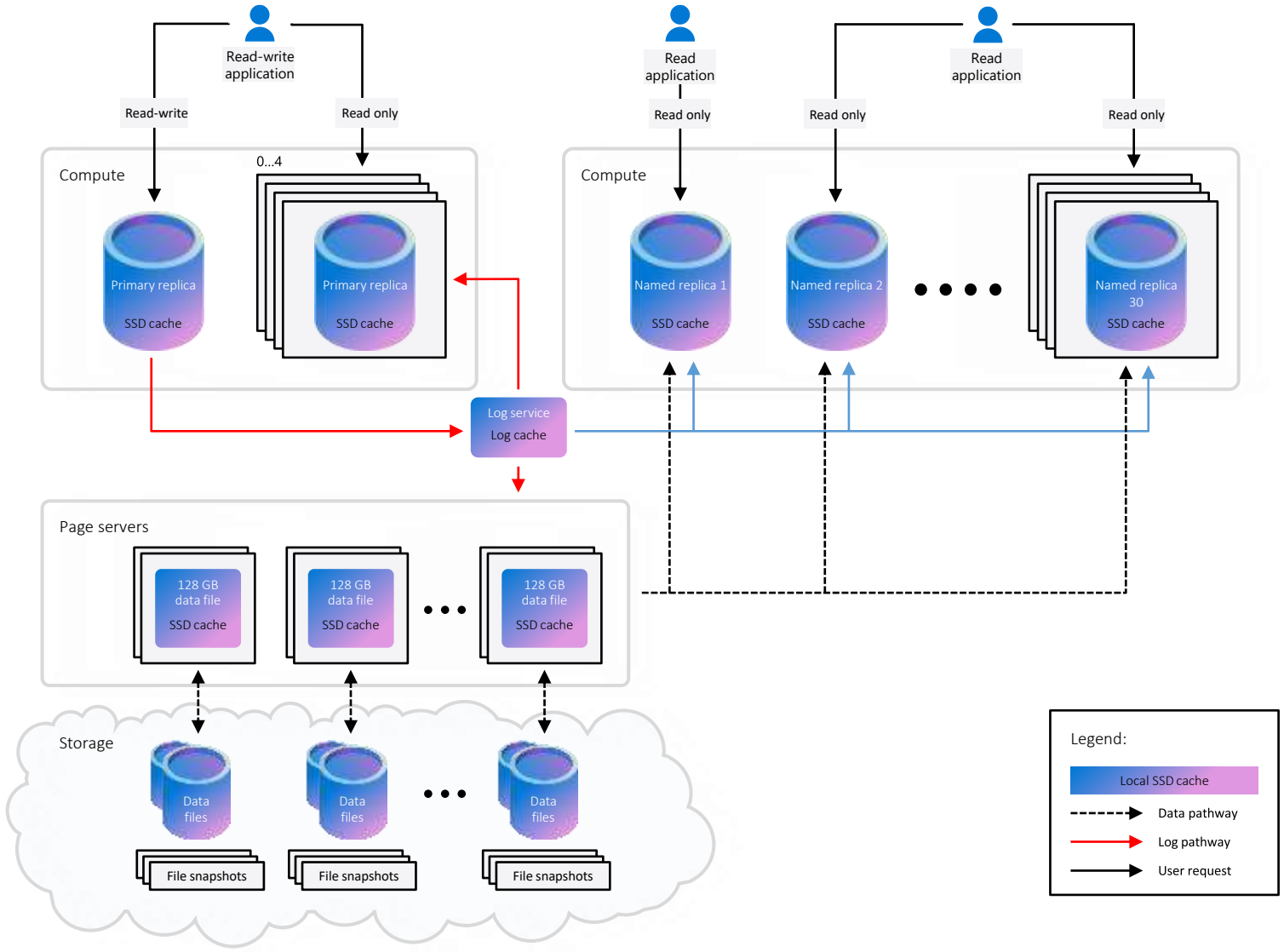
Automatic scaling of database storage up to 100 TB

Auto-scaling independence of CPU and memory to match workload demand

Per-Second Billing of used resources

Built-for-the cloud architecture enables near-limitless growth potential

Distributed, scale-out and resilient



Demo

Virtually infinite read scale with Hyperscale + serverless

Home >

myNewDB1 (mylogicalserver/myNewDB1)

SQL database

Search

- Overview
- Activity log
- Tags
- Diagnose and solve problems
- Query editor (preview)

Settings

- Compute + storage
- Connection strings
- Maintenance
- Properties
- Locks

Data management

- Replicas

Copy | Restore | Export | Set server firewall | Delete | Connect with... | Feedback

Essentials JSON View

Getting started | Monitoring | Properties | Features | Notifications (0) | Integrations | Tutorials

Start working with your database

Connect to your database and start working with data with a few simple steps. Learn more

Configure access

Configure network access to your SQL server. Learn more

Configure

Connect to application

Use connection strings to connect to your SQL database from your applications and favorite tools.

See connection strings

myNewDB1 (mylogicalserver/myNewDB1) | Replicas

SQL database

Search

Overview
Activity log
Tags
Diagnose and solve problems
Query editor (preview)

Settings
Compute + storage
Connection strings
Maintenance
Properties
Locks

Data management
Replicas

Integrations
Azure Synapse Link
Stream analytics (preview)
Add Azure AI Search

Power Platform

Create replica | Refresh | Feedback

Want to enable disaster recovery at lower price? Create standby replica for General Purpose or Business Critical service tier, with savings of up to 40%. Learn more

Geo and named replicas for your database are listed below. Named replicas reside in the same region as the primary and enable load-balancing of read-only workloads and other scenarios. Geo replicas reside on a different logical server from the primary and protect against regional failures or prolonged data center outage. Learn more

Name	Server	Region	HA replica count	Failover poli...	Pricing tier	Replica state
Primary						
myNewDB1	mylogicalserver	East US	1	None	Hyperscale: Premium-series, 8 vCores	Online
Named replicas						
myNewDB1_NR1	mylogicalserver	East US	4		Hyperscale - Serverless: Standard-series (Gen5), ...	Readable
myNewDB1_NR10	mylogicalserver	East US	4		Hyperscale - Serverless: Standard-series (Gen5), ...	Readable
myNewDB1_NR11	mylogicalserver	East US	4		Hyperscale - Serverless: Standard-series (Gen5), ...	Readable
myNewDB1_NR12	mylogicalserver	East US	4		Hyperscale - Serverless: Standard-series (Gen5), ...	Readable
myNewDB1_NR13	mylogicalserver	East US	4		Hyperscale - Serverless: Standard-series (Gen5), ...	Readable
myNewDB1_NR14	mylogicalserver	East US	4		Hyperscale - Serverless: Standard-series (Gen5), ...	Readable
myNewDB1_NR15	mylogicalserver	East US	4		Hyperscale - Serverless: Standard-series (Gen5), ...	Readable
myNewDB1_NR16	mylogicalserver	East US	4		Hyperscale - Serverless: Standard-series (Gen5), ...	Readable
myNewDB1_NR17	mylogicalserver	East US	4		Hyperscale - Serverless: Standard-series (Gen5), ...	Readable
myNewDB1_NR18	mylogicalserver	East US	4		Hyperscale - Serverless: Standard-series (Gen5), ...	Readable

Behind the scenes…

```
(1..30) | % { $jobs += Start-Job -ScriptBlock
{ New-AzSqlDatabaseSecondary -ResourceGroupName
"myResourceGroup" -ServerName "mylogicalserver" -
DatabaseName "myNewDB1" -PartnerResourceGroupName
"myResourceGroup" -PartnerServerName
"mylogicalserver"  -PartnerDatabaseName
("myNewDB1_NR" + $args[0]) -SecondaryType Named -
SecondaryServiceObjectiveName HS_S_Gen5_2 -
MaxVcore 2 -MinVcore 0.5 -
HighAvailabilityReplicaCount 4
} -ArgumentList ($_)  }
```

myNewDB1_NR20 (mylogicalserver/myNewDB1_NR20)

SQL database

Search

Overview
Activity log
Tags
Diagnose and solve problems
Query editor (preview)

Settings
Compute + storage
Connection strings
Maintenance
Properties
Locks

Data management
Replicas

Integrations
Azure Synapse Link
Stream analytics (preview)
Add Azure AI Search

Copy | Restore | Export | Set server firewall | Delete | Connect with... | Feedback

Essentials

JSON View

Getting started | Monitoring | Properties | Features | Notifications (0) | Integrations | Tutorials

Start working with your database

Connect to your database and start working with data with a few simple steps. Learn more

Configure access

Configure network access to your SQL server. Learn more

Configure

Connect to application

Use connection strings to connect to your SQL database from your applications and favorite tools.

See connection strings

Start developing

Work in your database by using tools to add, modify and query data. Compare tools

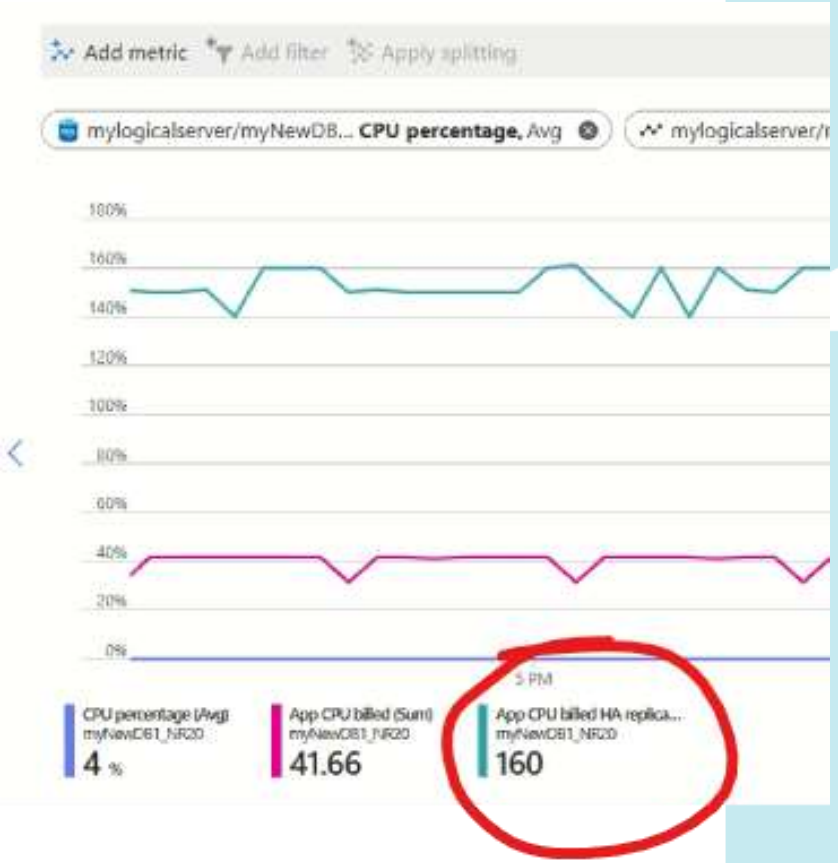
Open Azure Data Studio

Open in Visual Studio

Open in Visual Studio Code

Quiz time!

Why is the *App CPU billed (HA replicas)* so much higher than *App CPU billed*?



myNewDB1_NR20 (mylogicalserver/myNewDB1_NR20) | Metrics

SQL database

Search

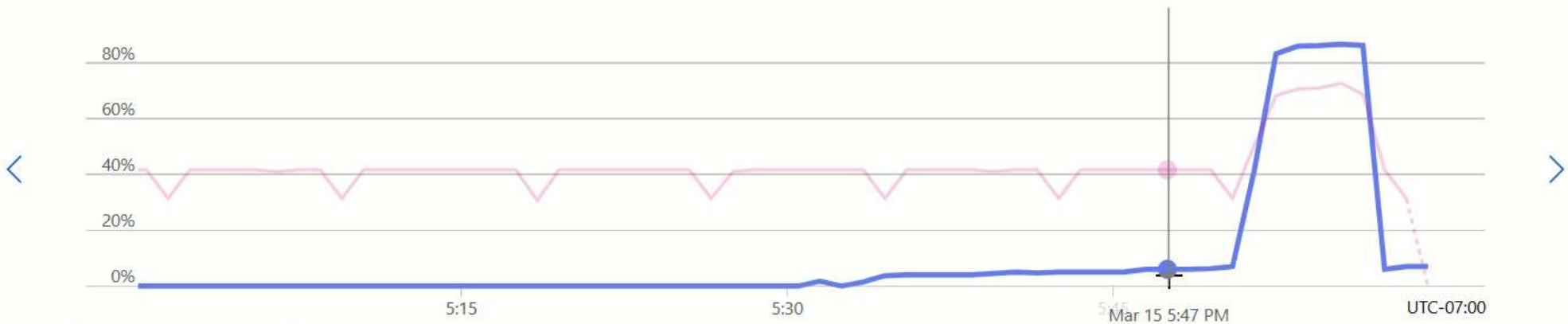
New chart | Refresh | Share | Feedback | Local Time: Last hour (Automatic - 1 minute)

Avg CPU percentage and Sum App CPU billed for myNewDB1_NR20

Add metric | Add filter | Apply splitting | Line chart | Drill into Logs | New alert rule | Save to dashboard

mylogicalserver/myNewDB... **CPU percentage,** Avg

mylogicalserver/myNewDB... **App CPU billed,** Sum



CPU percentage (Avg)
myNewDB1_NR20
6 %

App CPU billed (Sum)
myNewDB1_NR20
41.66

Writer @ 2024-03-16 01:19:10.963 Log rate 9.13 MB/sec; writer refresh 300 ms (press +/- to adjust)

Rdr # 0 OK @ 2024-03-16 01:19:09.635 Avg. latency: 59 ms	Rdr # 5 OK @ 2024-03-16 01:19:09.291 Avg. latency: 61 ms	Rdr # 10 OK @ 2024-03-16 01:19:09.635 Avg. latency: 54 ms	Rdr # 15 OK @ 2024-03-16 01:19:09.635 Avg. latency: 60 ms	Rdr # 20 OK @ 2024-03-16 01:19:09.291 Avg. latency: 55 ms	Rdr # 25 OK @ 2024-03-16 01:19:09.635 Avg. latency: 52 ms
Rdr # 1 OK @ 2024-03-16 01:19:09.635 Avg. latency: 56 ms	Rdr # 6 OK @ 2024-03-16 01:19:09.291 Avg. latency: 59 ms	Rdr # 11 OK @ 2024-03-16 01:19:09.635 Avg. latency: 52 ms	Rdr # 16 OK @ 2024-03-16 01:19:09.635 Avg. latency: 50 ms	Rdr # 21 OK @ 2024-03-16 01:19:09.635 Avg. latency: 51 ms	Rdr # 26 OK @ 2024-03-16 01:19:09.635 Avg. latency: 55 ms
Rdr # 2 OK @ 2024-03-16 01:19:09.635 Avg. latency: 57 ms	Rdr # 7 OK @ 2024-03-16 01:19:09.291 Avg. latency: 56 ms	Rdr # 12 OK @ 2024-03-16 01:19:09.635 Avg. latency: 53 ms	Rdr # 17 OK @ 2024-03-16 01:19:09.291 Avg. latency: 61 ms	Rdr # 22 OK @ 2024-03-16 01:19:09.635 Avg. latency: 50 ms	Rdr # 27 OK @ 2024-03-16 01:19:09.635 Avg. latency: 52 ms
Rdr # 3 OK @ 2024-03-16 01:19:09.635 Avg. latency: 51 ms	Rdr # 8 OK @ 2024-03-16 01:19:09.635 Avg. latency: 52 ms	Rdr # 13 OK @ 2024-03-16 01:19:09.635 Avg. latency: 51 ms	Rdr # 18 OK @ 2024-03-16 01:19:09.635 Avg. latency: 50 ms	Rdr # 23 OK @ 2024-03-16 01:19:09.291 Avg. latency: 57 ms	Rdr # 28 OK @ 2024-03-16 01:19:09.635 Avg. latency: 62 ms
Rdr # 4 OK @ 2024-03-16 01:19:09.635 Avg. latency: 51 ms	Rdr # 9 OK @ 2024-03-16 01:19:09.635 Avg. latency: 56 ms	Rdr # 14 OK @ 2024-03-16 01:19:09.635 Avg. latency: 53 ms	Rdr # 19 OK @ 2024-03-16 01:19:09.635 Avg. latency: 50 ms	Rdr # 24 OK @ 2024-03-16 01:19:09.635 Avg. latency: 60 ms	Rdr # 29 OK @ 2024-03-16 01:19:09.291 Avg. latency: 58 ms

Hyperscale Elastic Pools

Public preview

Increased storage limits

Storage up to 100 TB per elastic pool

Higher log generation rates

Log throughput independent of compute

Predictable scaling

Scale the compute for the elastic pool up or down in minutes, regardless of how big the pooled DBs are!

Pool-level read scale

Optional pool replicas to host read-only workload

https://aka.ms/hsepvideo (4 minute video); https://aka.ms/hsep (docs)
https://aka.ms/hsep-public-preview-tech-blog (public preview announcement).

Energy

Life Is On Schneider Electric

TRACTO

e·on

BKW

Payment Solutions

GlobalPay Solutions

PAYSPACE

CLEARENT INTELLIGENT PROCESSING

Health Care

RevenueWell

Metro South Health | Queensland Government

Entertainment

HI-REZ STUDIOS

BMI

Others

udaan

Deutsche Post DHL Group

Takeaways

Hyperscale is THE cloud native database for all types of workloads.

Hyperscale grows as your workloads grow – offering great scalability.

Hyperscale offers leading price-performance, especially with options like serverless and elastic pools.

Please do provide feedback!

https://sqlb.it/?12734

sqlbits 2024

References

- More about Hyperscale: https://aka.ms/hs
- Hyperscale simplified and lower pricing: https://aka.ms/hsignite2023 and https://aka.ms/hsprice2023
- Hyperscale serverless: https://aka.ms/sqldbserverless
- Hyperscale elastic pools: https://aka.ms/hsep-prms and https://aka.ms/hsep

Thank you!

@aditya_feb22 (X)

@arvisam (Bluesky, Mastodon and X)

sqlbits 2024