Microsoft

# PostgreSQL For SQL Server Professionals

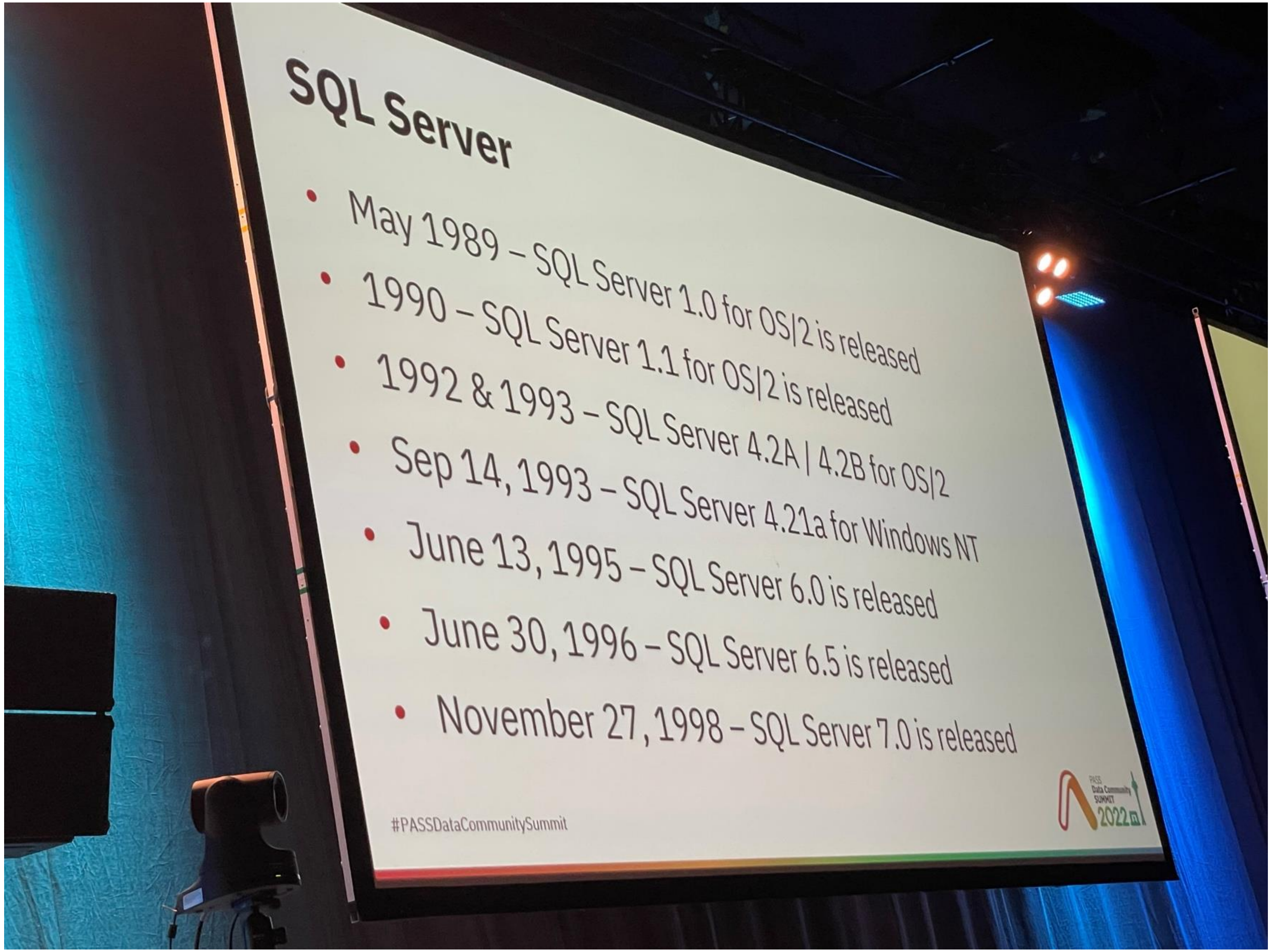
Silvano Coriani
Product @ Azure Postgres

Demos and slides published here:
https://github.com/scoriani/postgres-for-sqlserver-professionals

# Agenda

- Historical notes
- Architectures & Fundamentals
- Deployment options
- Data types & Indexes
- Management
- BCDR
- Programmability
- Security
- App Development
- Cloud Services: Azure Database for PostgreSQL

# SQL Server history



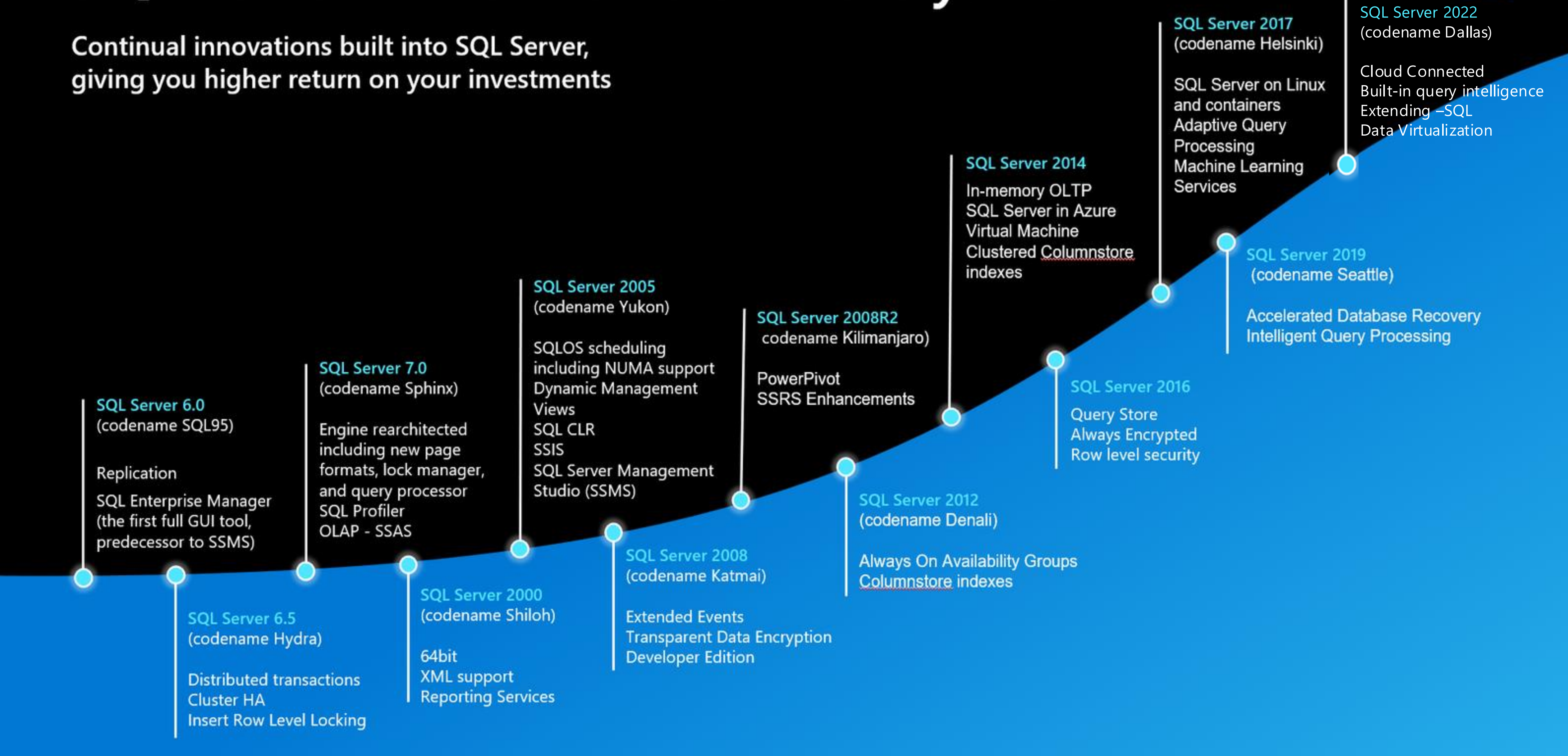
## SQL Server

- May 1989 – SQL Server 1.0 for OS/2 is released
- 1990 – SQL Server 1.1 for OS/2 is released
- 1992 & 1993 – SQL Server 4.2A / 4.2B for OS/2
- Sep 14, 1993 – SQL Server 4.21a for Windows NT
- June 13, 1995 – SQL Server 6.0 is released
- June 30, 1996 – SQL Server 6.5 is released
- November 27, 1998 – SQL Server 7.0 is released

#PASSDataCommunitySummit

# SQL Server investments over the years

Continual innovations built into SQL Server,
giving you higher return on your investments



**SQL Server 6.0**
(codename SQL95)

Replication

SQL Enterprise Manager
(the first full GUI tool,
predecessor to SSMS)

**SQL Server 6.5**
(codename Hydra)

Distributed transactions
Cluster HA
Insert Row Level Locking

**SQL Server 7.0**
(codename Sphinx)

Engine rearchitected
including new page
formats, lock manager,
and query processor
SQL Profiler
OLAP - SSAS

**SQL Server 2000**
(codename Shiloh)

64bit
XML support
Reporting Services

**SQL Server 2005**
(codename Yukon)

SQLOS scheduling
including NUMA support
Dynamic Management
Views
SQL CLR
SSIS
SQL Server Management
Studio (SSMS)

**SQL Server 2008**
(codename Katmai)

Extended Events
Transparent Data Encryption
Developer Edition

**SQL Server 2008R2**
codename Kilimanjaro)

PowerPivot
SSRS Enhancements

**SQL Server 2012**
(codename Denali)

Always On Availability Groups
Columnstore indexes

**SQL Server 2014**

In-memory OLTP
SQL Server in Azure
Virtual Machine
Clustered Columnstore
indexes

**SQL Server 2016**

Query Store
Always Encrypted
Row level security

**SQL Server 2017**
(codename Helsinki)

SQL Server on Linux
and containers
Adaptive Query
Processing
Machine Learning
Services

**SQL Server 2019**
(codename Seattle)

Accelerated Database Recovery
Intelligent Query Processing

SQL Server 2022
(codename Dallas)

Cloud Connected
Built-in query intelligence
Extending –SQL
Data Virtualization

# PostgreSQL History

### Inception

- PostgreSQL, initially known as Postgres, evolved from Ingres, a project at the University of California, Berkeley, in the mid-1980s led by Michael Stonebreaker.
- SQL Server (Sybase) and PostgreSQL actually share the same family tree with Ingress being their common ancestor.

### Release

- First released in 1989 as an open-source relational database management system (RDBMS).

### Postgres95

- In 1996, Postgres95 was released, marking a significant milestone with added SQL compliance and enhanced capabilities.

### Community Development

- Community-driven development began in the late 1990s, fostering rapid evolution and feature enhancements.

### Version 7.0

- Released in 2000, introduced several critical features like nested transactions and support for foreign keys.

### Official Name Change

- Renamed as PostgreSQL in 1996 to reflect its support for SQL standards more prominently.

### Ongoing Development

- Continuous development and updates have led to PostgreSQL becoming one of the most advanced and powerful open-source databases available today (v16.2).

# Core engines

# Key fundamentals (1/2)

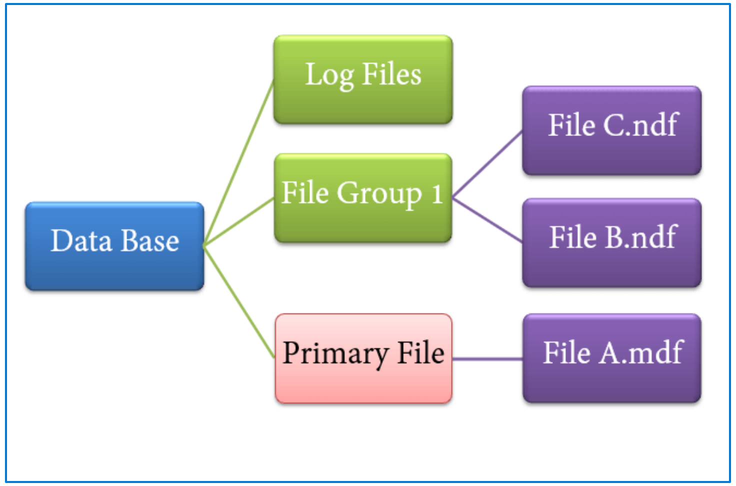
## SQL Server (proprietary)

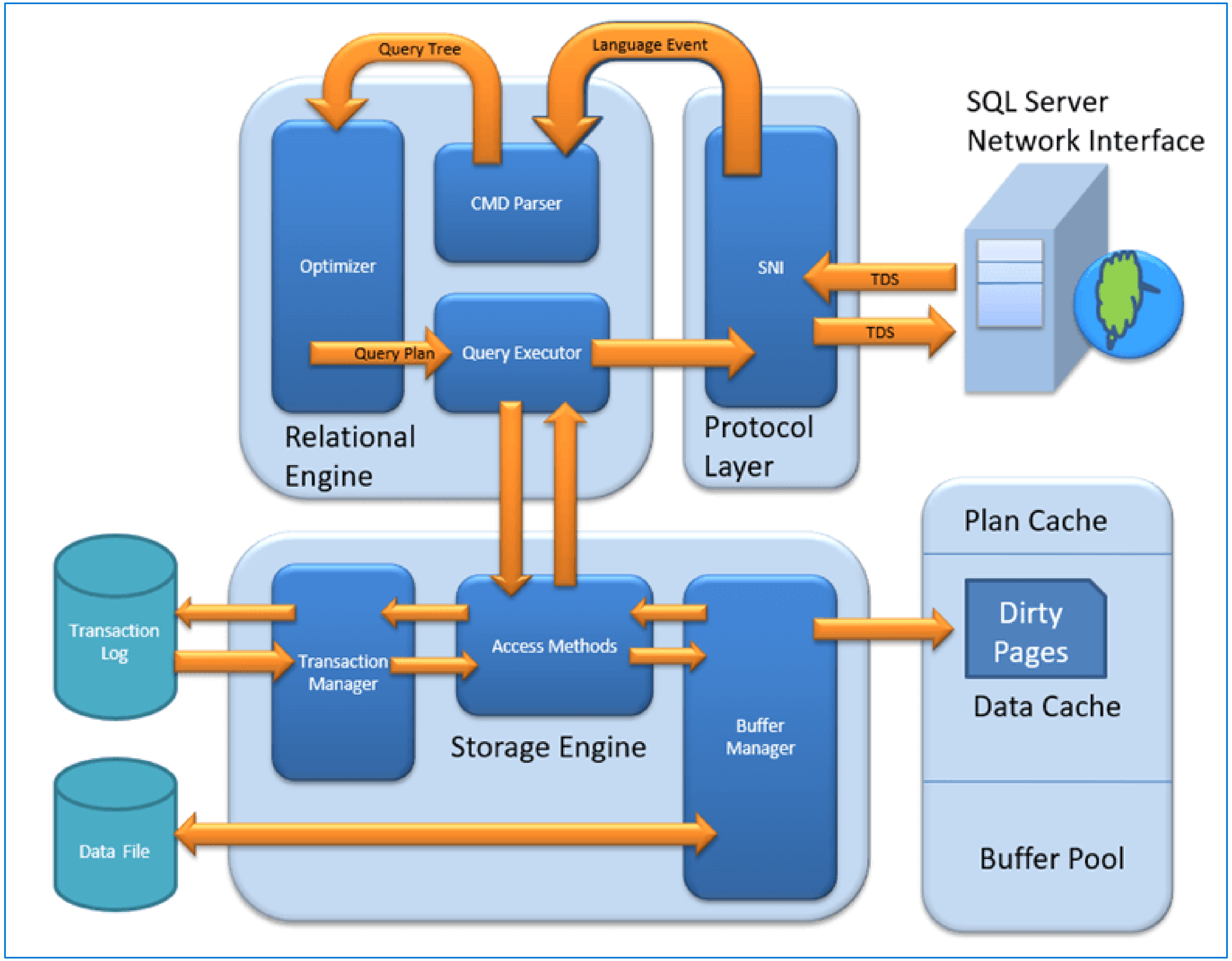
- Designed for Windows
- Ported on Linux in 2017 trhough a Platform Abstraction Layer (PAL)
- Supported on x64 proc architectures **for now**
- Multi-threaded
- Built on a concept called SQL Operating System (SQLOS) for
  - user mode thread scheduling
  - memory management
  - synchronization
  - diagnostics/DMVs

## PostgreSQL (open source)

- Community driven OSS , portable
  - Great Community Support and resources
  - Standard compliance
- Designed on Linux/Unix
  - “works” on Windows
  - Multi-process, shared memory
- Runs on all major proc architectures
- Extensibility by design from the core
  - Well-defined APIs to plug-in new implementations (types, operators, etc.)

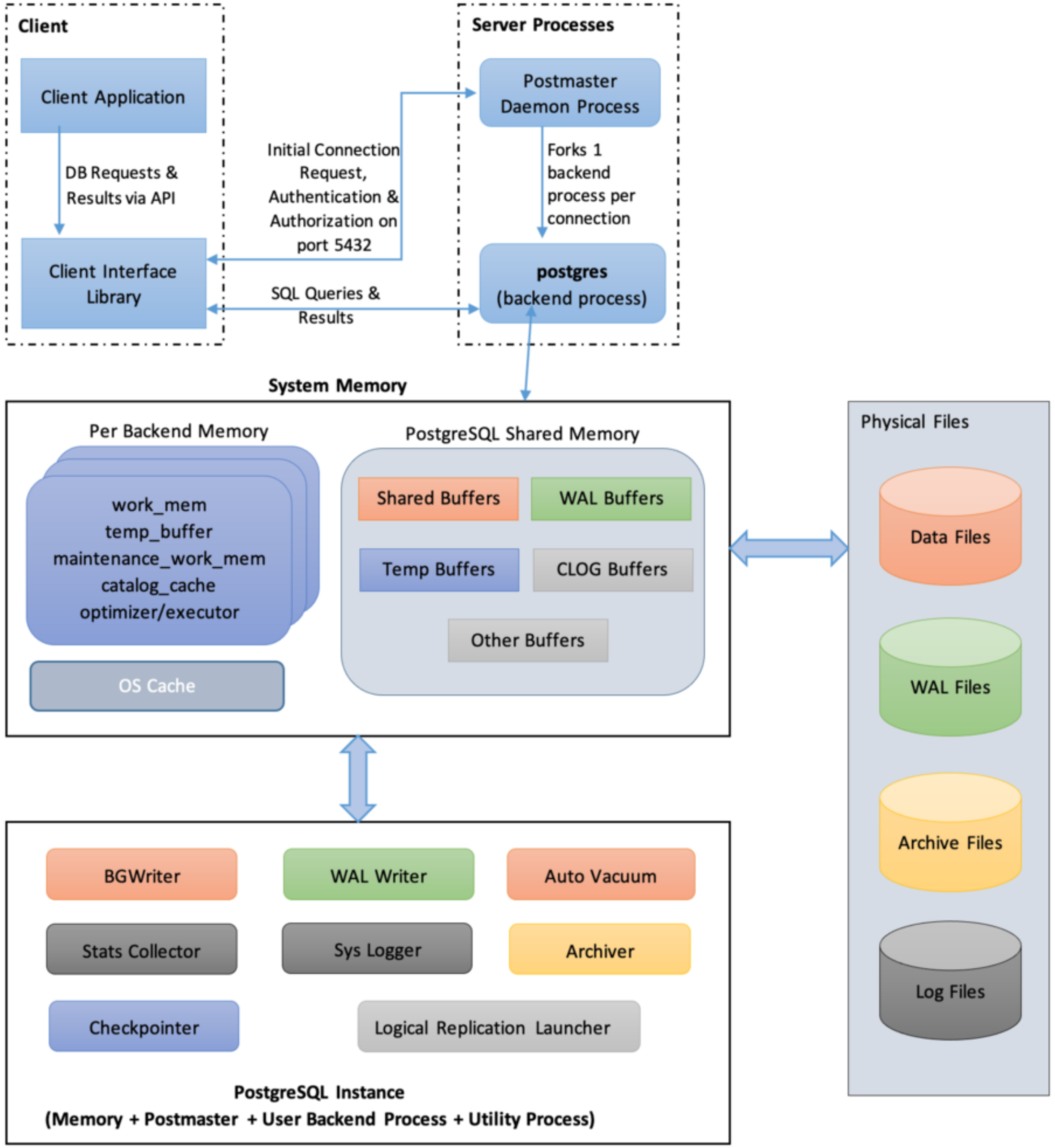
# SQL Server architecture



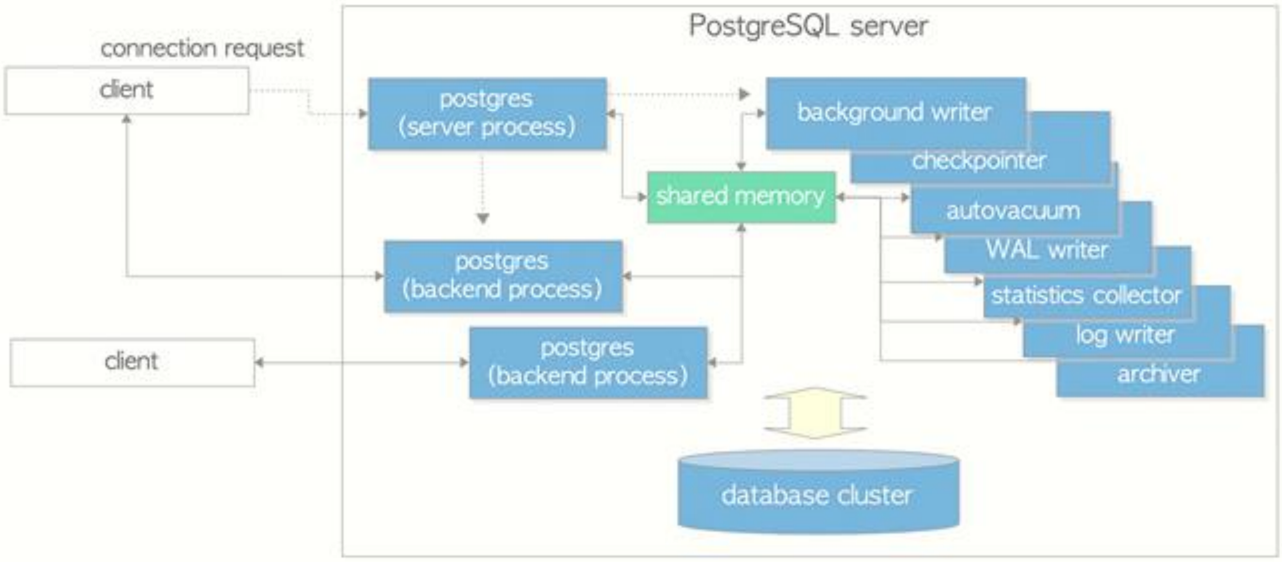


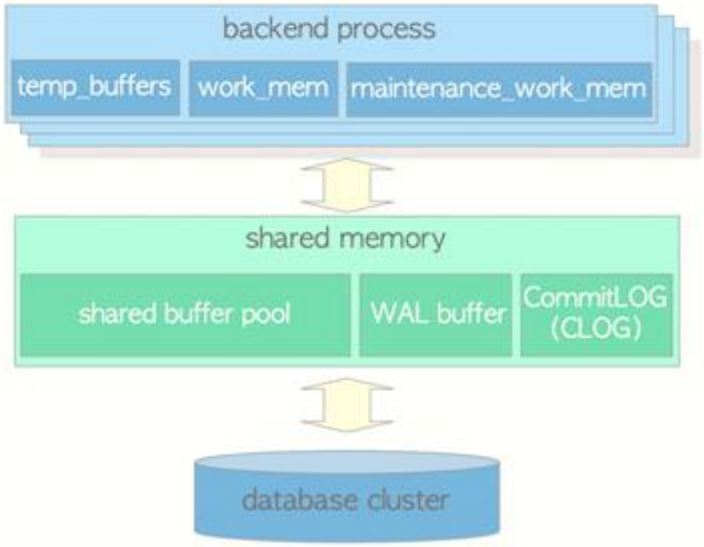
https://www.guru99.com/sql-server-architecture.html

# PostgreSQL architecture



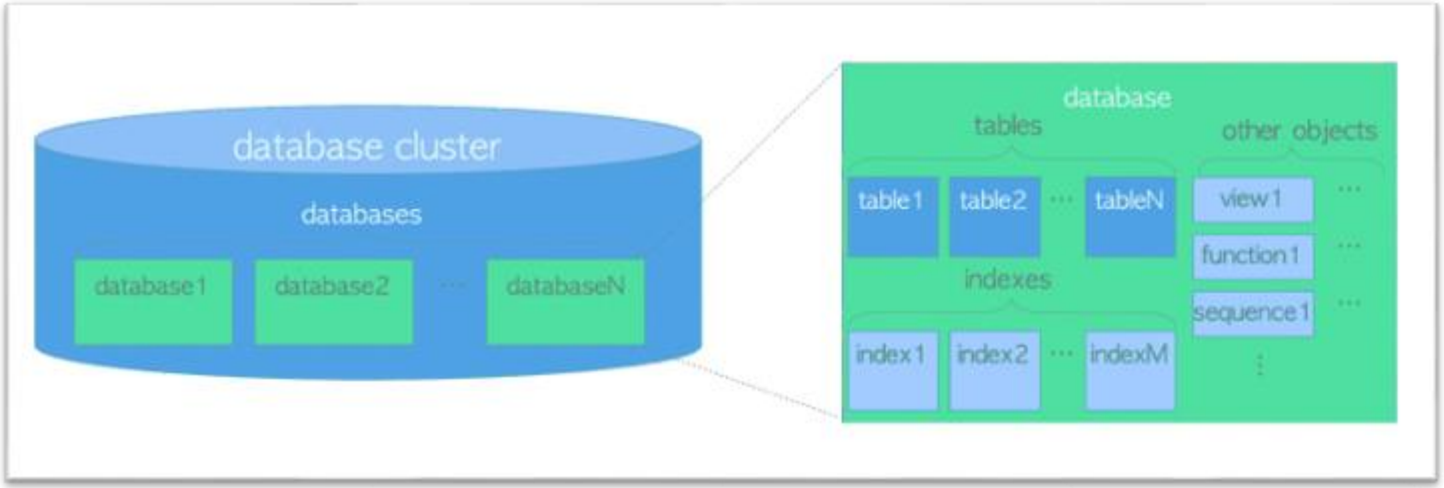
# PostgreSQL process and memory architecture

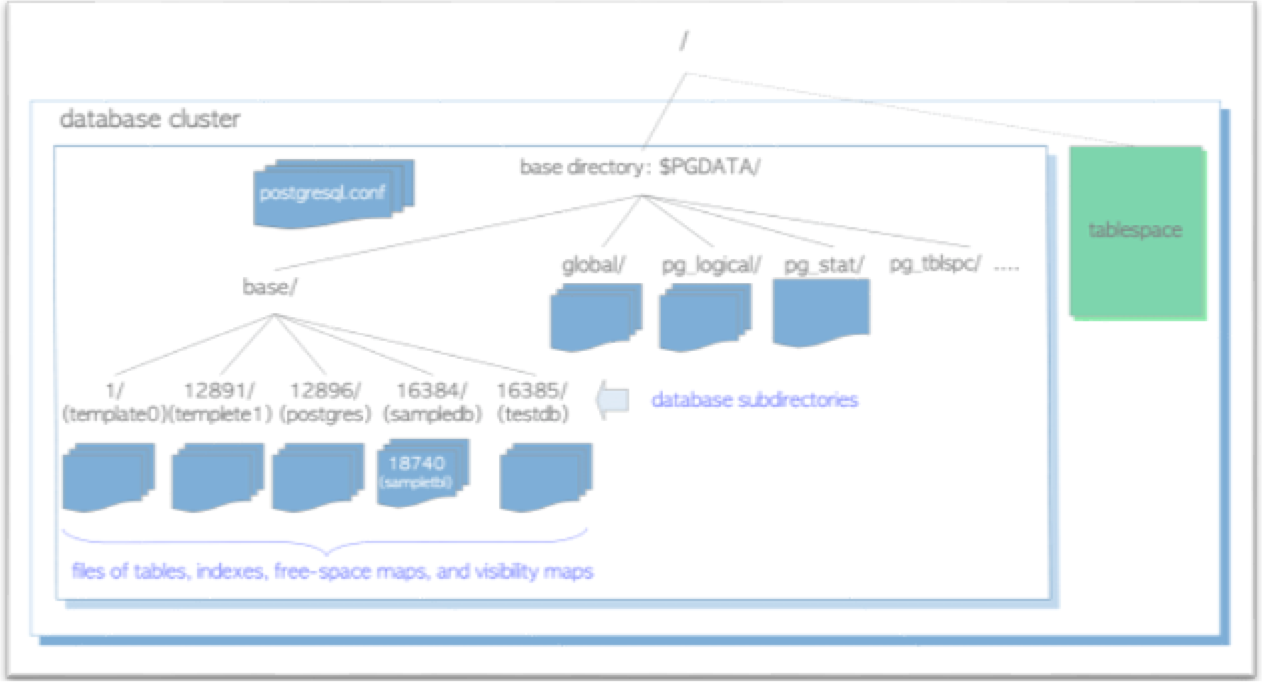


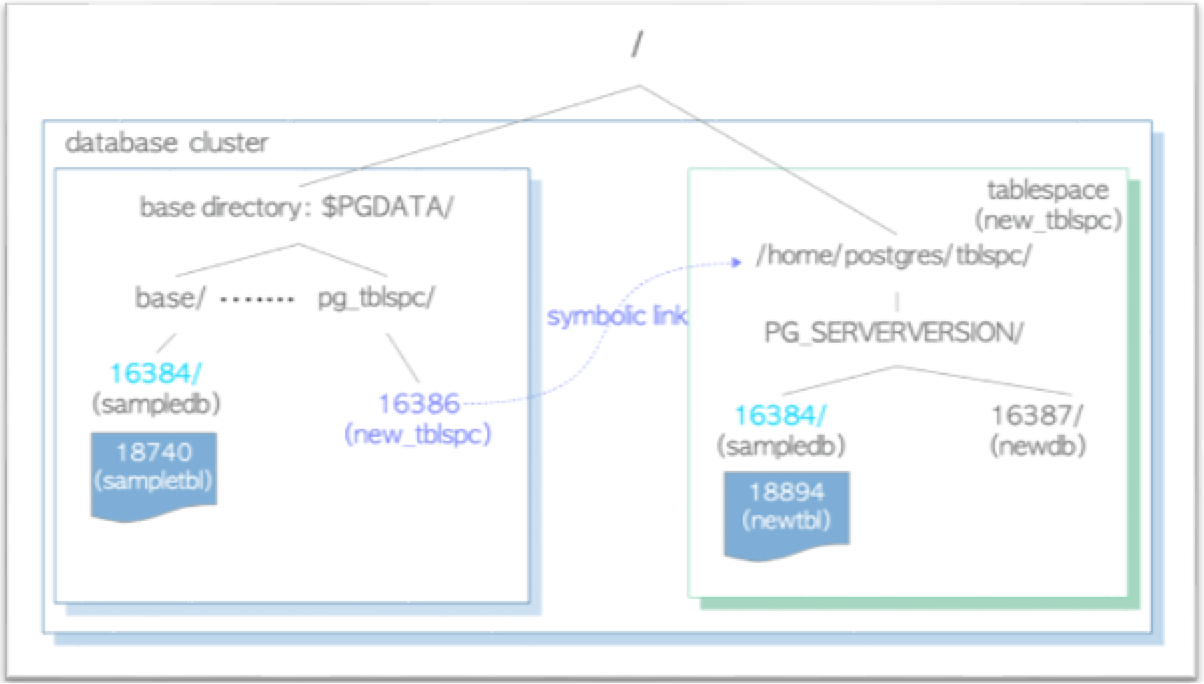


https://www.interdb.jp/pg/pgsql02/01.html

# PostgreSQL storage architecture







# Key fundamentals (1/2)

## SQL Server (proprietary)

- Designed for Windows
- Ported on Linux in 2017 trhough a Platform Abstraction Layer (PAL)
- Supported on x64 proc architectures only
- Multi-threaded
- Built on a concept called SQL Operating System (SQLOS) for
  - user mode thread scheduling
  - memory management
  - synchronization
  - diagnostics/DMVs

## PostgreSQL (open source)

- Community driven OSS , portable
  - Great Community Support and resources
  - Standard compliance
- Designed on Linux/Unix
  - “works” on Windows
  - Multi-process, shared memory
- Runs on all major proc architectures
- Extensibility by design from the core
  - Well-defined APIs to plug-in new implementations (types, operators, etc.)

# Key fundamentals (2/2)

## SQL Server (proprietary)

- Security / Encryption
- AlwaysON Availability Groups
- AlwaysON Failover Clustering
- Transactional, Merge, P2P replication
- Analysis Services
- Integration Services
- Reporting Services
- SQL Server Management Studio

## PostgreSQL (open source)

- Multi Version Concurrency Control (MVCC)
- Advanced data types (arrays, JSONB, geo, vectors etc.)
- Advanced indexing (GiST, GIN, BRIN, Bloom, etc.)
- Advanced programmability (PL/pgSQL, Perl, Python, Java, etc.)
- High availability with streaming replication
- Logical replication for data integration scenarios

# Deployment options (1/2)

## SQL Server

- Windows Server
  - Native service
  - Unlimited cores and up to 48TB RAM
- Windows 11 dev/test
  - Native service
  - Docker containers
- Linux
  - Supported on RH, SUSE and Ubuntu
  - Containers
  - Kubernetes

## PostgreSQL

- Supported natively on any operating system and proc architecture
  - Local Service or Daemon
  - Containers
  - Kubernetes
- Main distros' package managers
- PostgreSQL.org repositories
- Source code (GitHub or https://git.postgresql.org)
- Config files
  - General service
  - Authentication
  - Identities

# Deployment options (2/2)

## SQL Server

- Cloud Services
  - **Azure SQL**
  - Amazon RDS
  - GCP CloudSQL
  - AliCloud ApisaraDB
  - Others

## PostgreSQL

- Cloud Services
  - **Azure Database for PostgreSQL**
  - **Azure CosmosDB for PostgreSQL**
  - Amazon RDS
  - Amazon Aurora
  - GCP AlloyDB
  - CockroachDB
  - Neon
  - Others

# Demo:

## *Deployment options*

# Management

## SQL Server

- Known for ease maintenance efforts
- Maintenance plans
  - Index and stats
  - Backup/restore
  - Consistency checks
- Row versioning (mostly self managing)
  - Snapshot isolation
  - Read Committed Snapshot Isolation (RCSI)
  - Advanced Database Recovery
    - PVS
- SQL Agent

## PostgreSQL

- Routine maintenance tasks
  - Vacuuming
  - Reindexing
  - Backup/restore
  - Log management
- Multi Version Concurrency Control
  - The need for Vacuum process
- Vacuum
  - Concurrent vs full
  - After data load
  - Row changes (including non modification updates)
  - Impact on indexes
  - Autovacuum
  - Autovacuum budget
  - Change thresholds per table
  - HOT UPDATES
  - Inline vacuum
- pgCron

# Monitoring & troubleshooting

## SQL Server

- Activity monitor
- Performance dashboard
- Dynamic Management Views
- Query Store
- Perfmon
- XEvents/Tracing
- Database Tuning Advisor
- Query Tuning Assistant
- Tons of 3rd party tools

## PostgreSQL

- Standard Unix/Linux tools
- Cumulative Statistics System
  - Collection configuration parameters
  - Control functions
  - Predefined views
- Lock contention: pg_locks
- Progress reporting
- pg_stat_statement extension
- Troubleshooting
  - Log analyzer: pgBadger

# Demo:

## *Monitoring and troubleshooting*

# BCDR

## SQL Server

- Backup/Restore options
  - Full/Diff/Log
  - Files & Filegroups
  - Compression/Encryption
  - Backup to cloud
- Availability Group topologies
- Failover Clustering
- Log shipping
- Logical Replication
  - Transactional
  - Merge

## PostgreSQL

- Backup/Restore
  - **Logical and physical Backup/restore (pg_dump, pg_basebackup)**
  - File system backup
  - Continuous WAL archiving and Point-in-Time Recovery
- Physical replication for HA (sync/async)
  - Log replication to standby server
  - Failover process
- Logical replication
  - Works for data integration scenarios as well

# Programmability

## SQL Server

- Procedures, functions and triggers
- Spatial data types
- Graph
- .NET CLR
- Language extensibility
- Data virtualization / Linked servers
- Full-text search
- CDC/Change Tracking

## PostgreSQL

- Procedures, functions and triggers
- Schema and Programmability
  - Table inheritance
  - Declarative partitioning
  - Pattern matching ILIKE/SIMILAR TO Regex/POSIX Regex.
- JSON
  - Rich Set of JSON operators
  - Expansive JSON functions
  - Indexing to support pattern matching
- Spatial data types
  - Geometry, geography, raster
  - Indexes
    - Rtree, quadtree
  - Functions
    - ST_Distance, ST_Area, ST_GeometryType,ST_Intersection….
  - Related extensions
- Foreign Data Wrappers
  - SQL Server
  - Oracle
  - …

# Data types

## SQL Server

- Exact numerics
- (Unicode) character strings
- Approximate numerics
- Binary strings
- Date and time
- Spatial
- Hierarchyid
- Rowversion
- UUID

## PostgreSQL

- Numeric
- Monetary
- Character
- Binary Data
- Date/Time
- Boolean
- Enumerated
- Geometric
- Network Address
- Bit String
- Text Search
- UUID
- XML
- JSON
- Arrays
- Composite
- Range
- Domain
- Object Identifier
- pg_lsn
- Pseudo-Types

# Indexes

## SQL Server

- Btree
  - Clustered
  - Non-clustered
- ColumnStore
  - Clustered
  - Non-clustered
- Hash (in memory)
- Filtered
- Computed
- Included columns
- Spatial
- Full-text
- XML

## PostgreSQL

- Indexing & Advanced Indexing
  - Btree
  - Hash
  - GIN
  - GiST, SP-GiST
  - BRIN
  - Partial indexes
  - Index on expressions
  - Extensions can overload access methods to "extend" indexing to other data types (PostgreSQL: Documentation: 16: 38.16. Interfacing Extensions to Indexes)
- pg_trgm - fast searching for similar strings
  - gist_trgm_ops
  - gin_trgm_ops

# Demo:

*Data types and indexes*

# Security

## SQL Server

- Users and Principals
- Ownership and user-schema separation
- Server-level and database-level roles
- Permission hierarchy
- Windows and Azure Active Directory integration
- Encryption
  - Column
  - Storage
  - Always Encrypted
- Auditing
- Ledger

## PostgreSQL

- Robust access-control system
- Client Authentication: GSSAPI, SSPI, LDAP, SCRAM-SHA-256, Certificate, and more
- Database Roles
  - Users
  - Groups
  - Membership
- Column and row-level security
- Encryption
  - Columns
  - Partitions
  - Storage
- Multi-factor authentication with certificates and an additional method
- pgAudit

# Tools

## SQL Server

- Microsoft
  - “new” go-sqlcmd
  - SQL Server Management Studio
  - Azure Data Studio
  - VS Code Extensions
- 3rd party
  - Dev & test
  - Monitoring
  - Management
  - HammerDB
  - …

## PostgreSQL

- Client tools… some!
  - psql
  - pgAdmin
  - Azure Data Studio
  - DBeaver
- PostgreSQL client and server apps
  - reindexdb, vacuumdb, pg_dump, pg_waldump, etc
- Benchmarking
  - pgbench
  - HammerDB
- 3rd party
  - Dev & test
  - Monitoring
  - Management

# App development: drivers and frameworks

## SQL Server

- .NET SqlClient
- Entity Framework Core
- Java
  - JDBC Driver
  - Hibernate
  - Spring Data
- Python
  - pyODBC
  - SQLAlchemy
- Node
  - Tedious
  - Sequelize
- Go
  - go-mssqldb
  - GoORM
- Data API Builder

## PostgreSQL

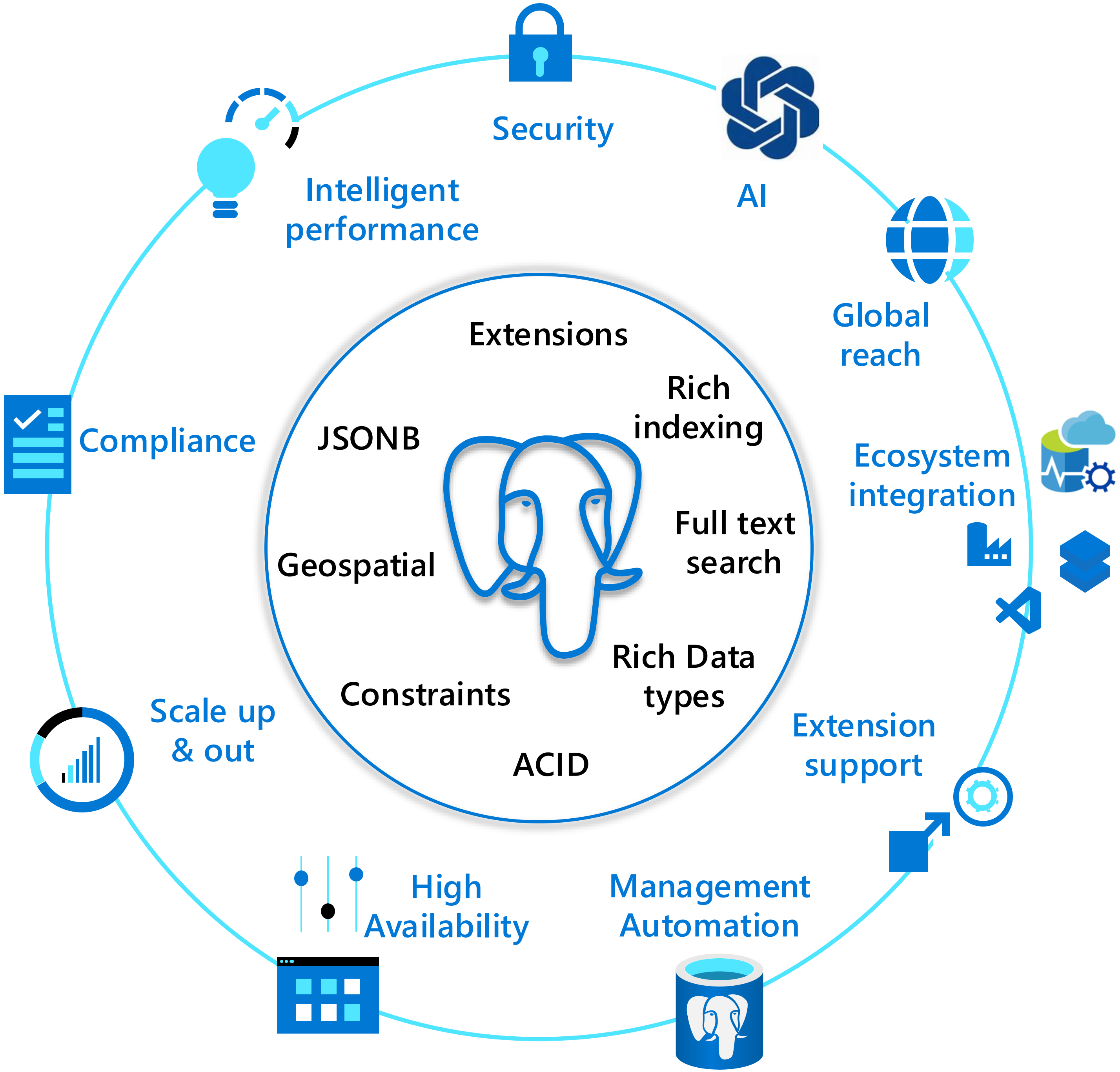
- libpq — C Library
- libpqxx – C++
- psycopg – Python, SQLAlchemy, Django
- psqlODBC - ODBC
- pjJDBC – Java, Hibernate and Spring
- R2DBC – Reactive Spring
- Npgsql - .NET and Entity Framework Core
- node-postgres – Node.js
- Sequalize and pg
- 3rd party
  - Prisma
  - Devart
  - Debezium
  - Many others
- **Data API Builder!**

# Demo:

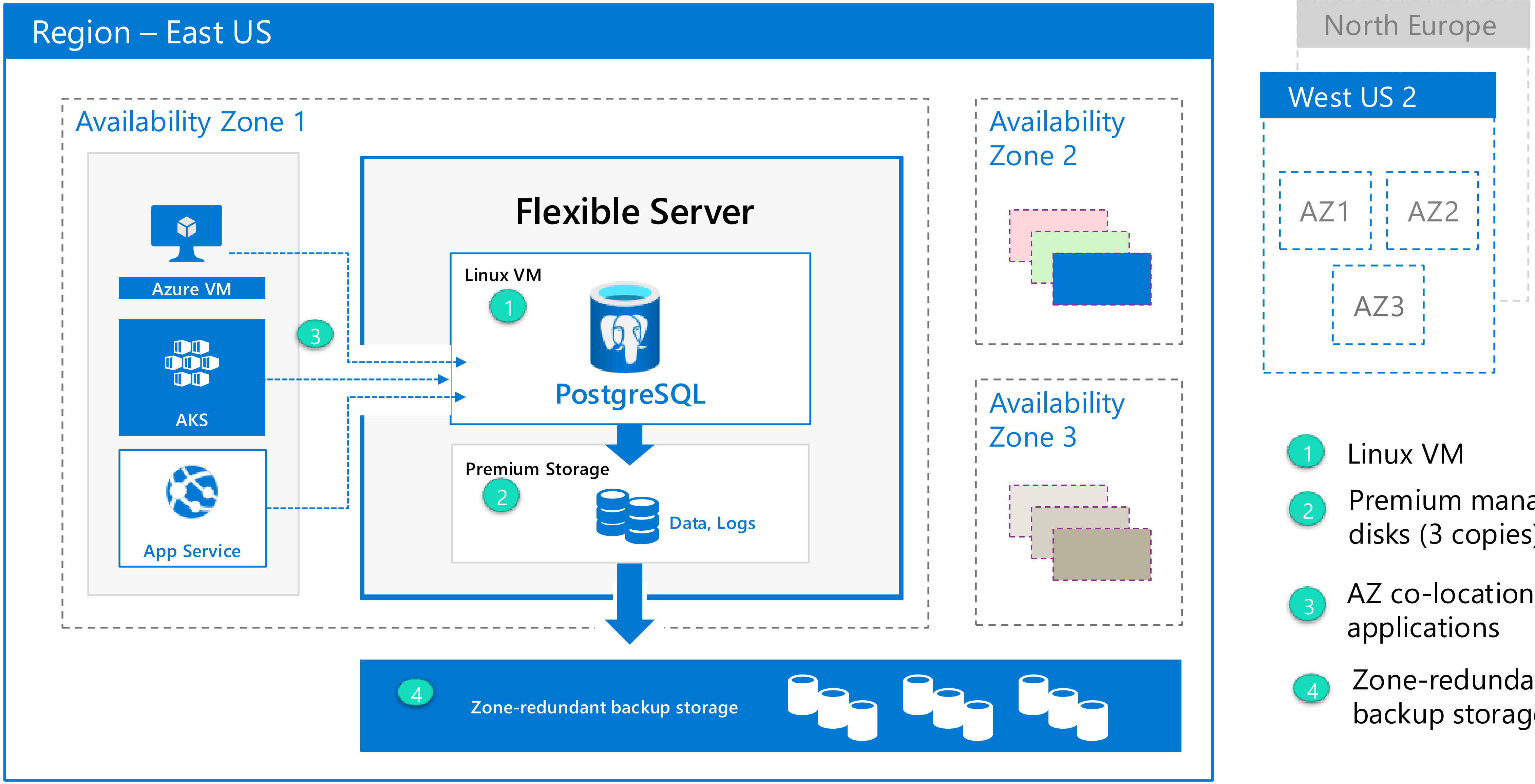
*Tools and App development*

# Cloud services

# Azure builds upon PostgreSQL



# Flexible Server Architecture



1 Linux VM

2 Premium managed disks (3 copies)

3 AZ co-location with applications

4 Zone-redundant backup storage

# Workload Optimized Compute SKU’s

Cost optimized for different workloads

Each switch between any SKU in minutes

Stop/Start during inactive periods

Reserved Capacity

## Memory Optimized

Up to 96 vCores with 1:8 CPU to Memory ratio optimized for best performance of IO intensive workloads

## General Purpose

Up to 96 vCores with 1:4 CPU to Memory ratio suitable for most database workloads

## Burstable

Highly cost effective, ideal for Development and Testing

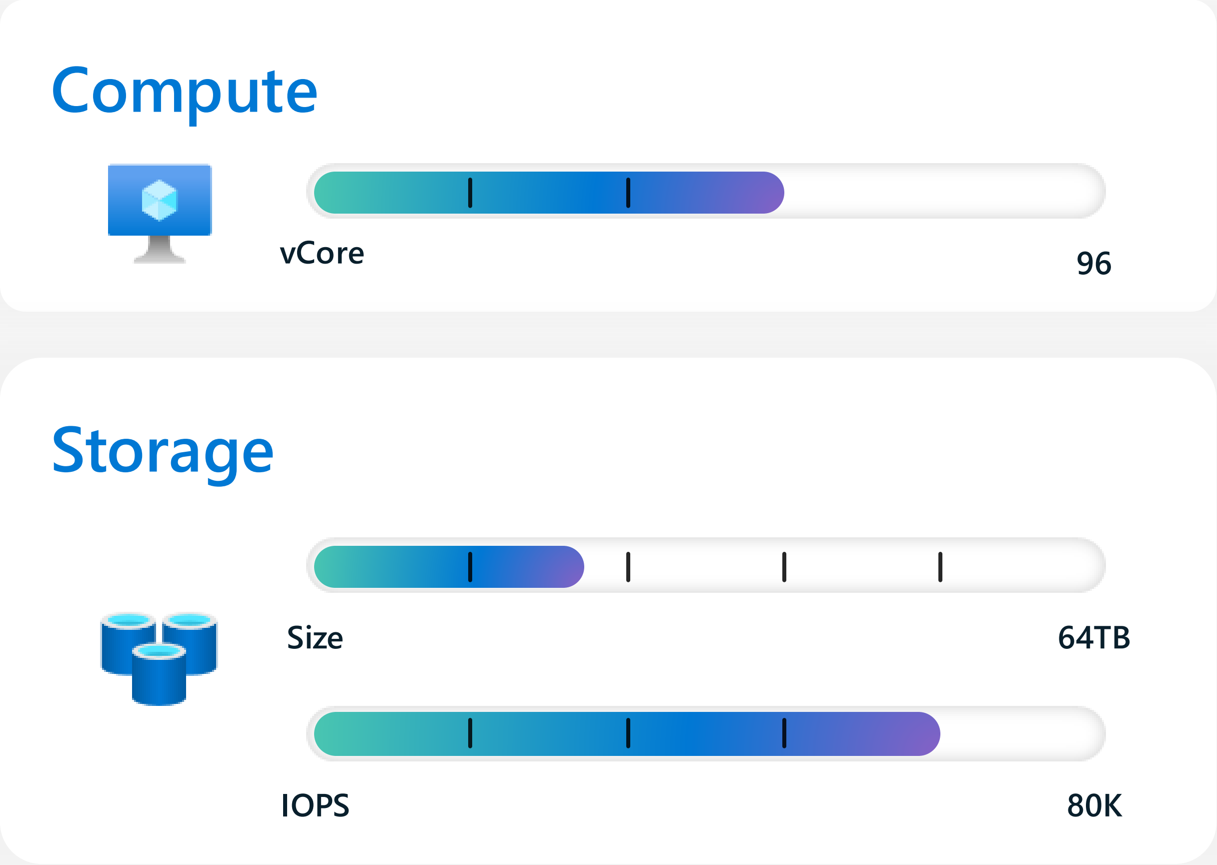
# Elastic Compute and Storage

Scale compute in **under 20s**

Scale storage **online**

Scale **storage size and IOPS** separately

**99.99% SLA** with Availability Zones



Compute

vCore

96

Storage

Size

64TB

IOPS

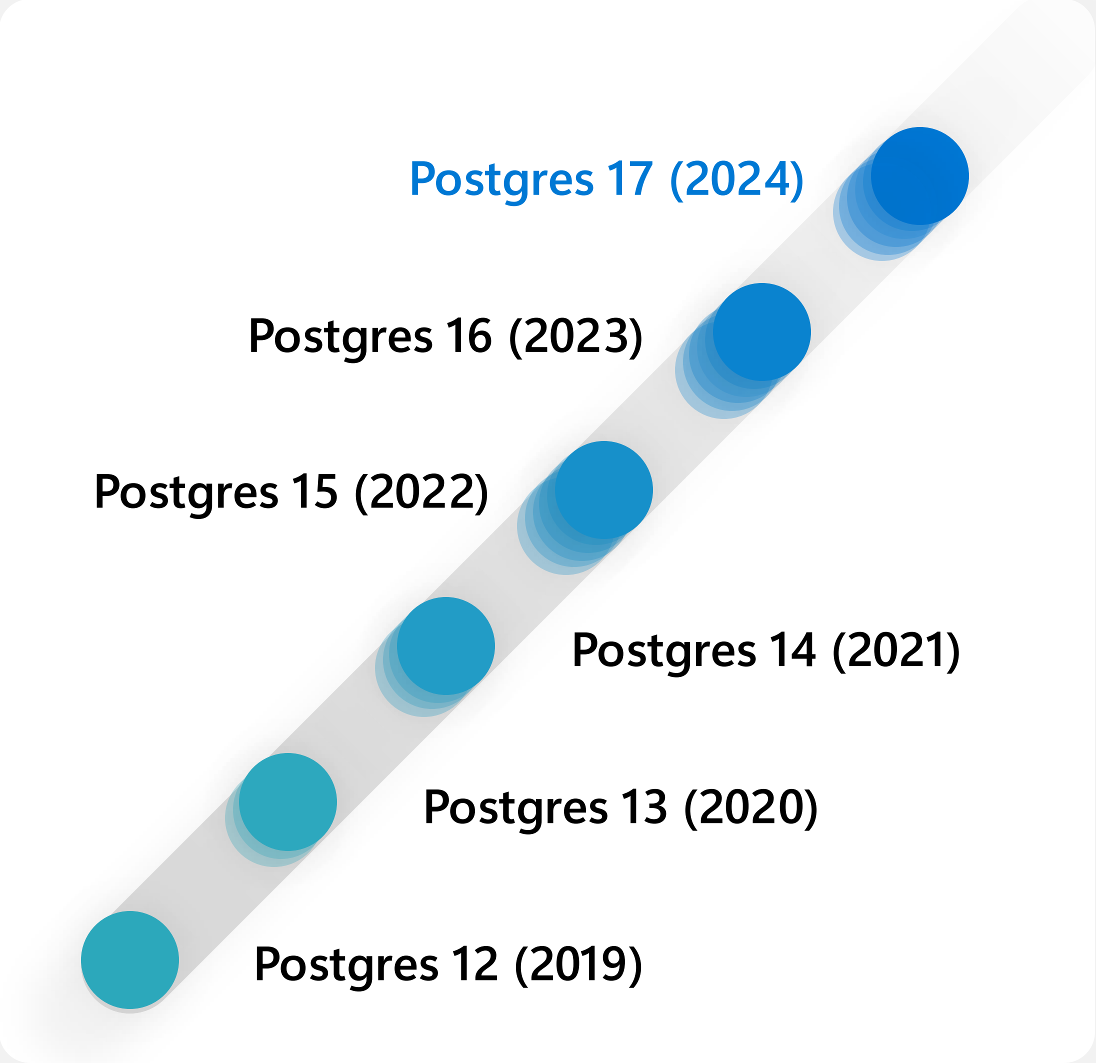
80K

# All Community Supported Postgres Versions

Major versions available **with weeks** of community GA

Minor versions **maintained automatically**

**Upgrade in-place** in minutes



# Broad support for common Postgres extensions

**60+ Postgres extensions** supported

Enables developers to **extend the functionality of Postgres** beyond core capabilities

Microsoft **automatically maintains** extensions versions

- address_standardizer
- address_standardizer_data_us
- amcheck
- azure_ai
- azure_local_ai (Preview)
- **azure_storage**
- bloom
- btree_gin
- btree_gist
- citext
- cube
- dblink
- dict_int
- dict_xsyn
- earthdistance
- fuzzystrmatch
- hstore
- hypopg
- intagg
- intarray
- isn
- lo
- login_hook
- ltree
- orafce
- pageinspect
- **pgaudit**
- pg_buffercache
- **pg_cron**
- pgcrypto
- pg_failover_slots (Preview)
- pg_freespacemap
- pg_hint_plan
- pglogical
- **pg_partman**
- pg_prewarm
- pg_repack
- pgrouting
- pgrowlocks
- pg_squeeze
- pg_stat_statements
- pgstattuple
- pg_trgm
- pg_visibility
- plpgsql
- plv8
- **postgis**
- postgis_raster
- postgis_sfcgal
- postgis_tiger_geocoder
- postgis_topology
- postgres_fdw
- semver
- session_variable
- sslinfo
- tablefunc
- tds_fdw
- timescaledb
- tsm_system_rows
- tsm_system_time
- unaccent
- uuid-ossp
- vector

# Extensive Monitoring

## Rich metrics and logs provide observability into the entire database workload

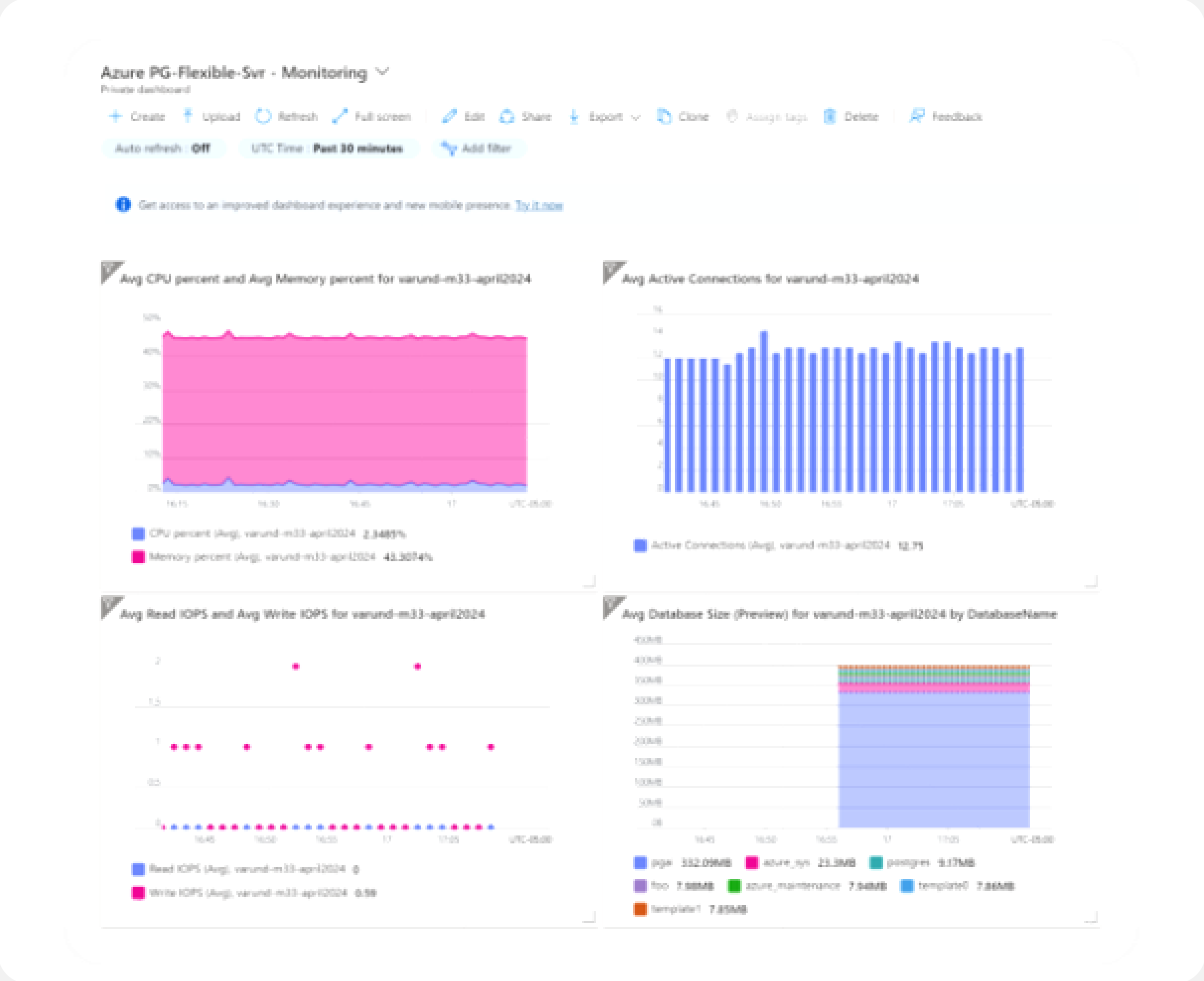
Access to detailed metrics and logs

Quickly **diagnose performance** issues

Make informed scaling decisions

**Set up alerts** and auto-scaling for quick responses

**Visualize data** using the Portal, Power BI, Grafana, or Log Analytics



Azure PG-Flexible-Svr - Monitoring

Private dashboard

Auto refresh : Off

UTC Time : Past 30 minutes

Get access to an improved dashboard experience and new mobile presence. Try it now

Avg CPU percent and Avg Memory percent for varund-m33-april2024

CPU percent (Avg), varund-m33-april2024 2.3485%

Memory percent (Avg), varund-m33-april2024 43.3074%

Avg Active Connections for varund-m33-april2024

Active Connections (Avg), varund-m33-april2024 12.75

Avg Read IOPS and Avg Write IOPS for varund-m33-april2024

Read IOPS (Avg), varund-m33-april2024 0

Write IOPS (Avg), varund-m33-april2024 0.59

Avg Database Size (Preview) for varund-m33-april2024 by DatabaseName

pgia 332.09MB

azure_sys 23.3MB

postgres 9.17MB

foo 7.98MB

azure_maintenance 7.94MB

template0 7.86MB

template1 7.85MB

# Enterprise Security

Azure Database for PostgreSQL is **the only Postgres offering** with support for Entra Id Authentication

Service or Customer Managed encryption keys to **protect data at rest**

**Secure network connectivity** via Private Endpoints

## Entra Id

Enterprise Identity built into Postgres

## Customer Managed Keys

Key Vault integration with optional HSM support

## Private Endpoints

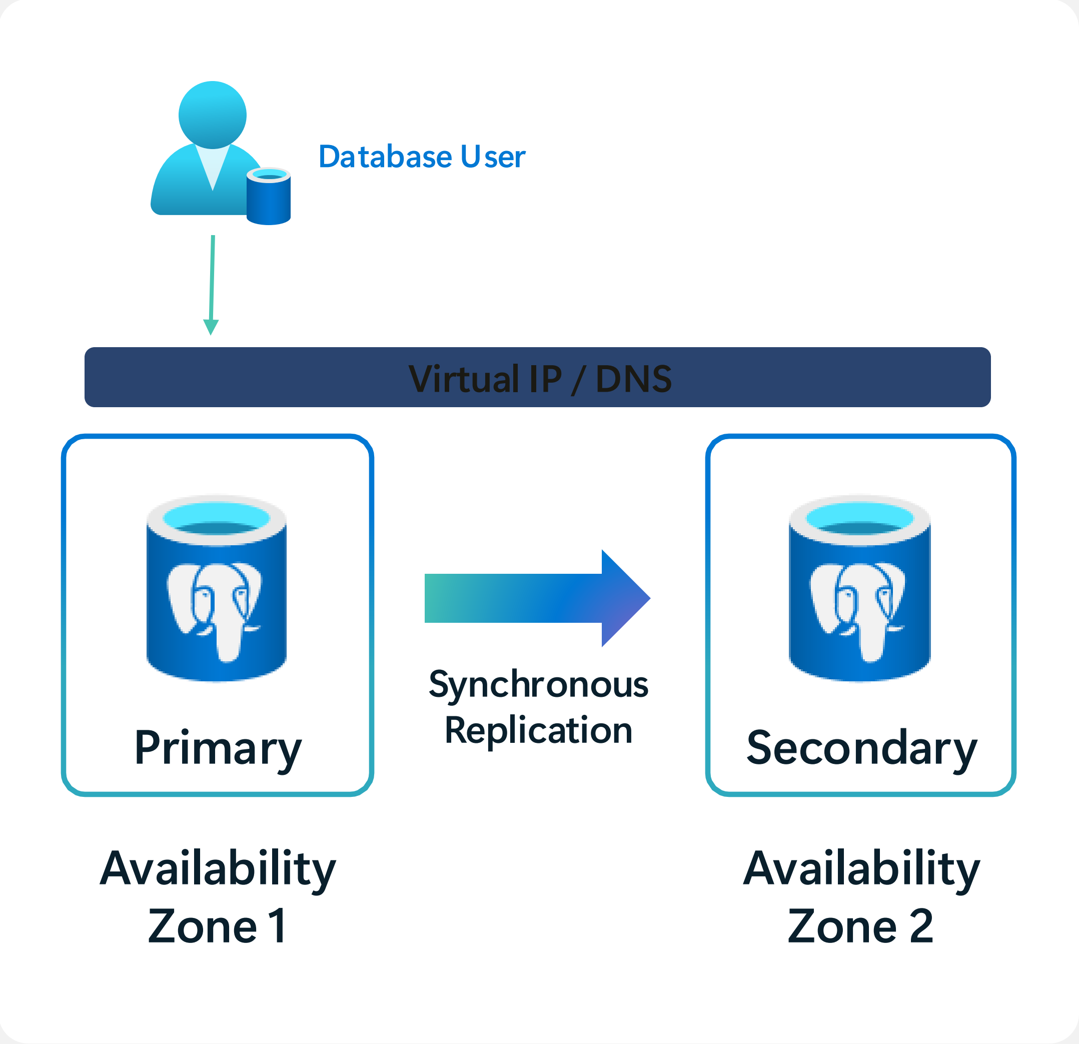
Network interface that uses a private IP address from your virtual network

# High Availability

Provides a replica of the production database across Availability Zones with a (no data loss) **Recovery Point Objective = 0**

**Automatic Failover** in scenarios where servers or zones fail

**Automatic rebuild** of the Secondary HA instance after failover occurs



HA-DR demo at Posette Conference 2024:
https://youtu.be/Yeb2Nv5mZtg?list=PLlrxD0HtieHg5ldHlEn3nMMpClX0RUntH&t=549

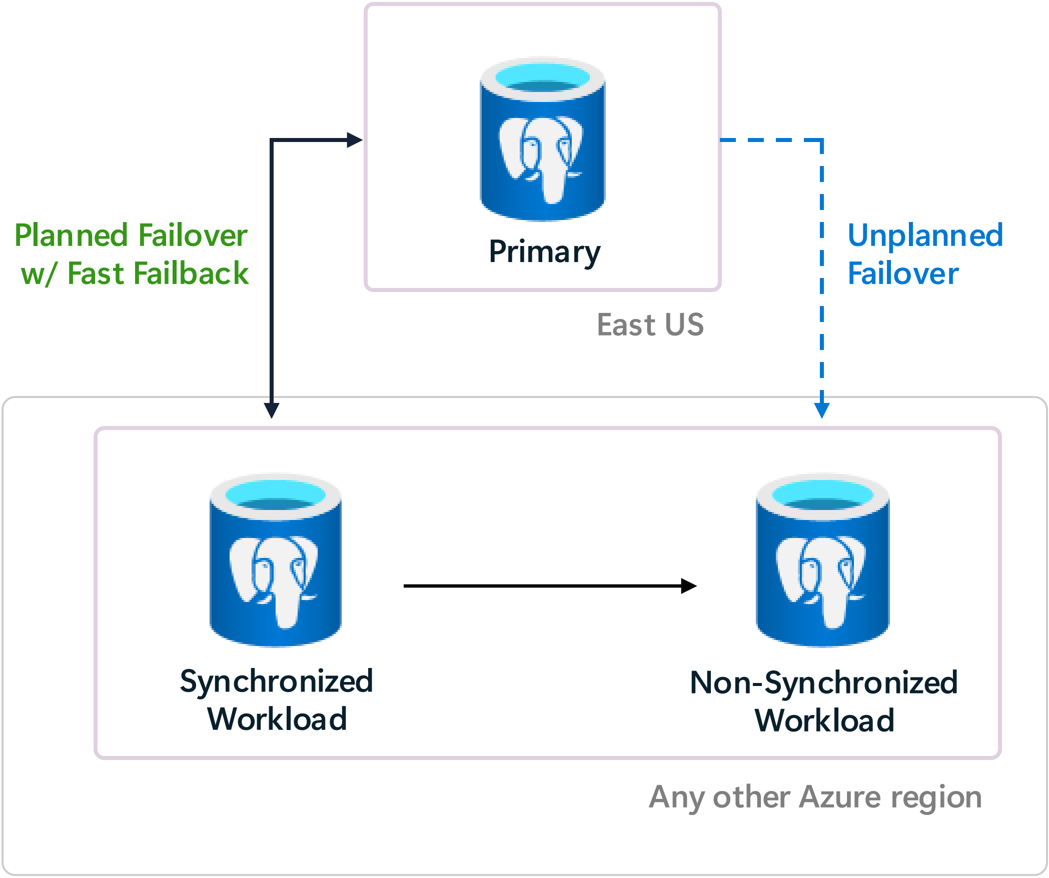
# Disaster Recovery

**Planned** or **Unplanned** failover to replicas running in any other Azure region

Execute a Planned Failover to **synchronize data between regions** prior to failover

Execute Unplanned Failover for **high impact emergency events** to move workload immediately without data synchronization

Both approached support Virtual IP with no application changes required



# Disaster Recovery

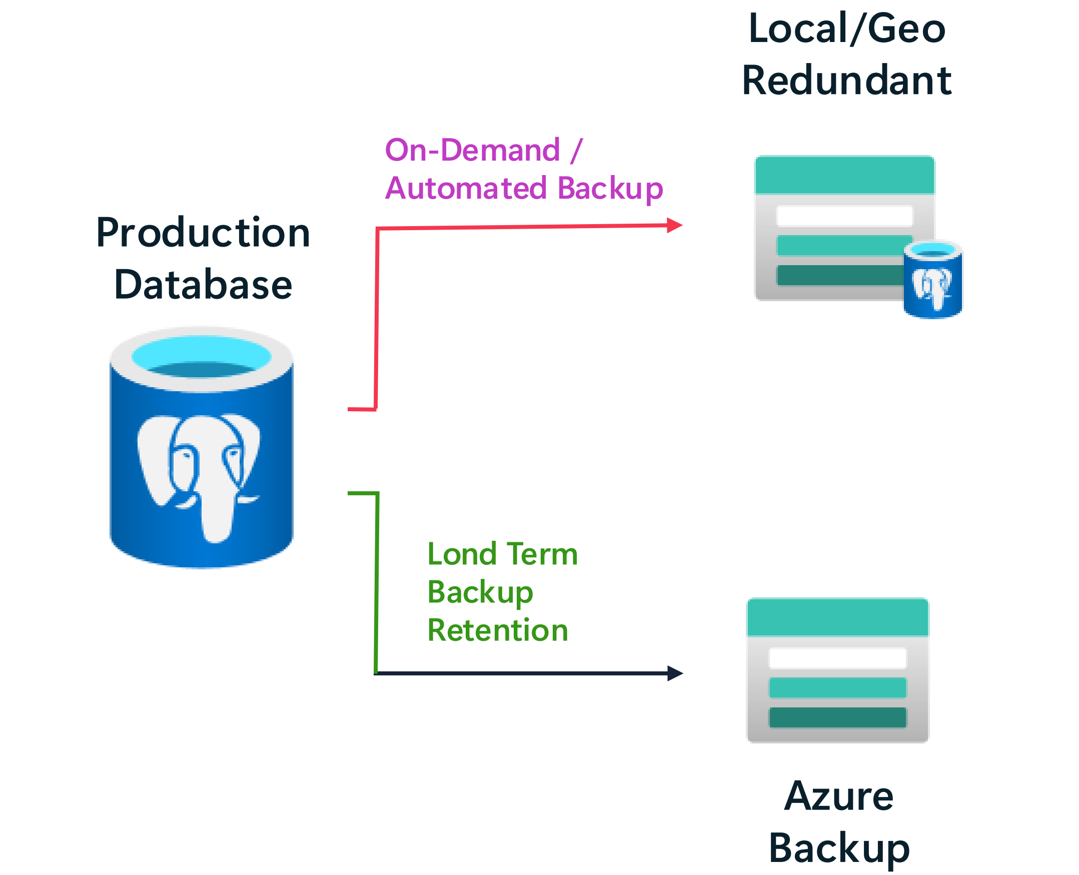
Backup protection for major disruptions

**On-Demand** or **Automated** backups protect data from loss with 30 Day retention

Configure backups for **geo-redundant storage**, and restore to different regions in event of failure

Long-term backup retention for **up to 10 years** supports compliance requirements for data retention

RPO of 5 minutes



# Comprehensive IaC

**Infrastructure-as-Code**

Use your **preferred IaC provider** to programmatically deploy and manage Flexible Server resources

Integrate with existing **infrastructure workflows**

**Version control** IaC for robust resource change management

## Terraform

Open-source IaC tool for configuring and deploying cloud infrastructure.

## Bicep

A domain-specific language (DSL) that uses declarative syntax to deploy Azure resources

## Ansible

Declaratively manage your Azure using a simple configuration language

## ARM Templates

Declaratively manage your Azure using a simple configuration language

.NET Aspire

Azure-Samples/eShopOnAzure

+ Rest API's, CLI, and PowerShell also

# Built-in Optimization

Unique to Azure

## Index Recommendations

Automatically determine optimal indexes based on user activity

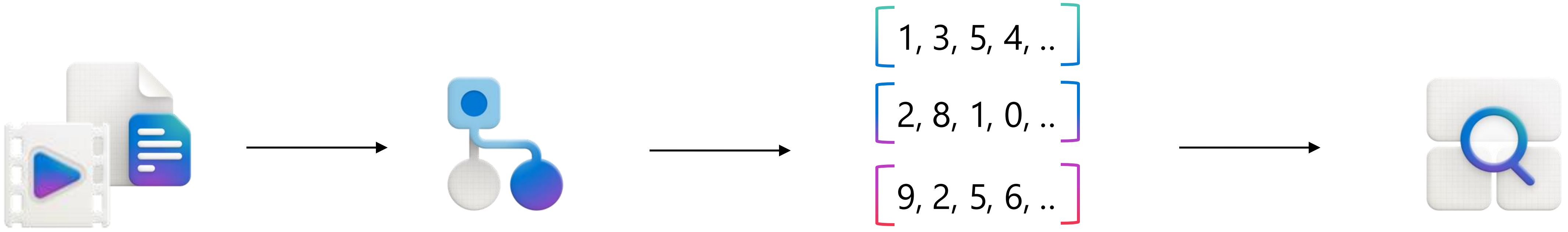
CREATE and DROP recommendations

Minimize Server Resources

Maximize Workload Performance

[Unleash the power of automated index tuning in Azure Database for PostgreSQL Flexible Server - Microsoft Community Hub]()

# How Vectors Work: Important in the AI Era



[1, 3, 5, 4, .. ]

[2, 8, 1, 0, .. ]

[9, 2, 5, 6, .. ]

Raw Data → Deep Learning Model → High-dimensional Vectors → Vector Similarity Search

Vector similarity search empowers Generative AI apps

Recommendation

Question Answering

RAG apps

...

# Azure Database for PostgreSQL: Native Vector Search

**Open-source Pgvector extension** provides support to store, index, and query vectors for similarity search scenarios

Supports multiple vector **distance functions**

Enables AI solutions to seamlessly integrate into **existing OLTP Postgres apps** without exporting data to specialize systems

Access control, encryption, high availability, disaster recovery all "**just work**"

## Generative AI apps

**RAG (Retrieval Augmented Generation) apps**

Retrieve private data to ground LLM model responses

**Recommendation/Semantic Search**

Retrieve similar documents by distance between vectors

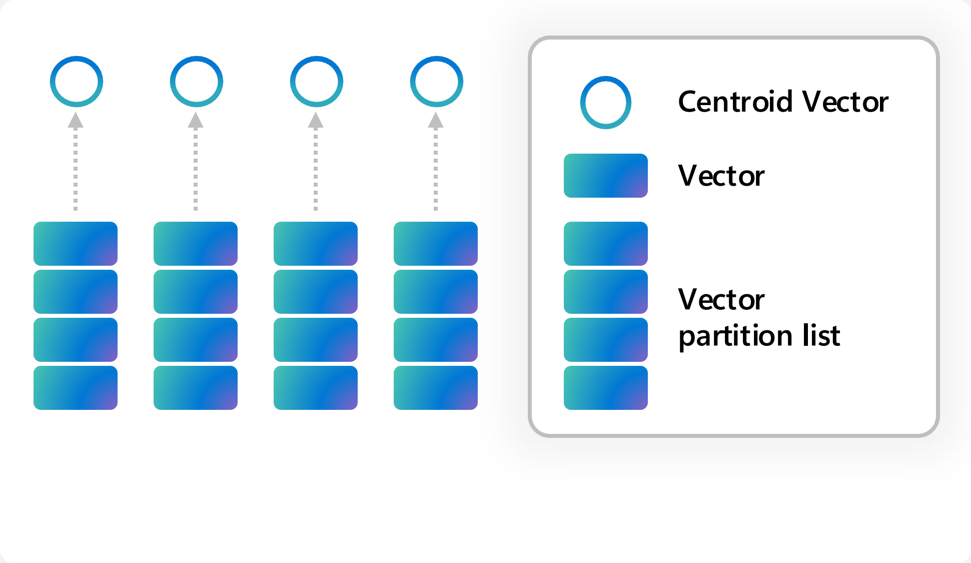
**Hybrid Search**

Combine vector search, row filtering, and full-text search

# Vector indexes supported today

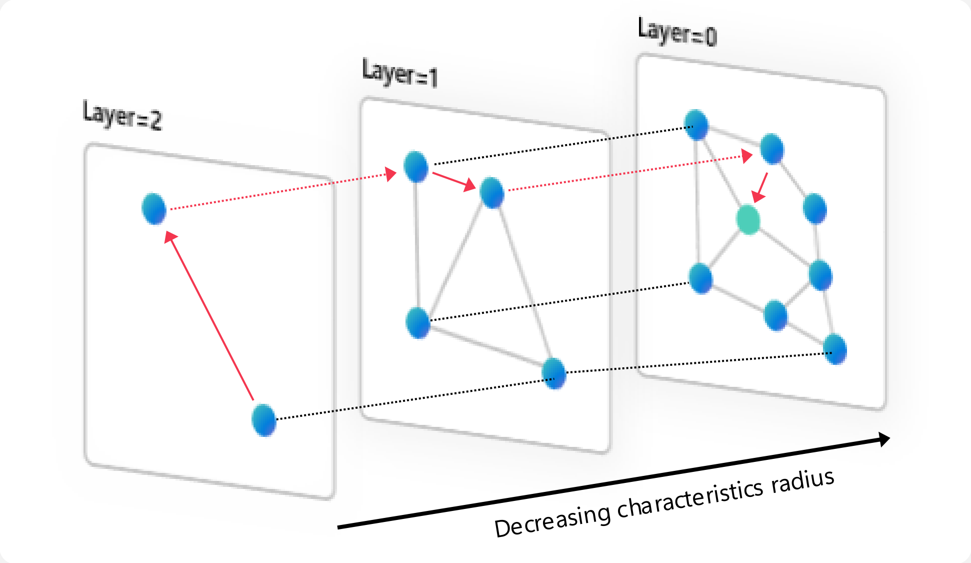
## IVFFlat

- Clusters vectors by applying k-means clustering.
- Memory efficient but requires index rebuilds.



## HNSW

- Builds a multi-layer graph with long and short connections between the vectors.
- The graph can be incrementally updated.



# AI Services integrated with Azure Postgres

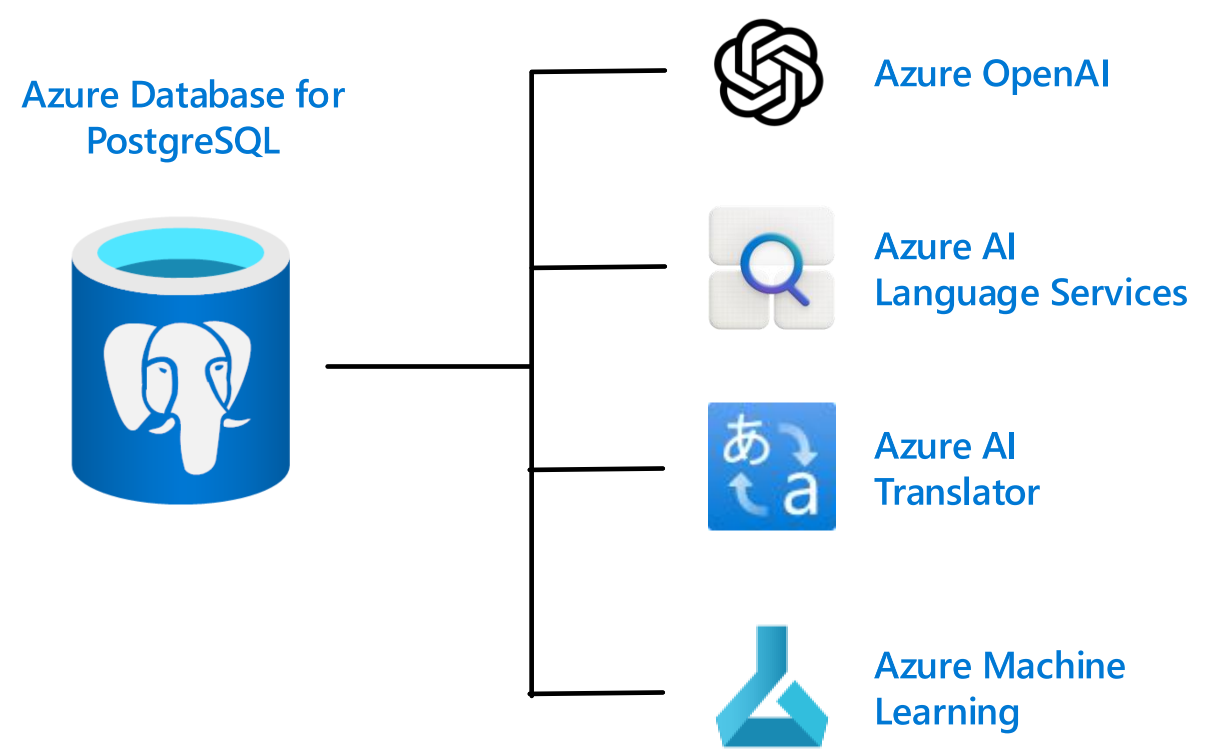
## Make remote calls directly from PostgreSQL

The **azure_ai extension** provides a SQL-based interface to integrate with AI services

Supports:

- Azure OpenAI
- Azure AI Language Services
- Azure AI Translator
- Azure Machine Learning

Enables developers to **rapidly integrate AI capabilities** into their app without complex re-architecture or refactoring



[Azure AI extension on Azure Database for PostgreSQL - Flexible Server]()

# In-Database Embedding Models

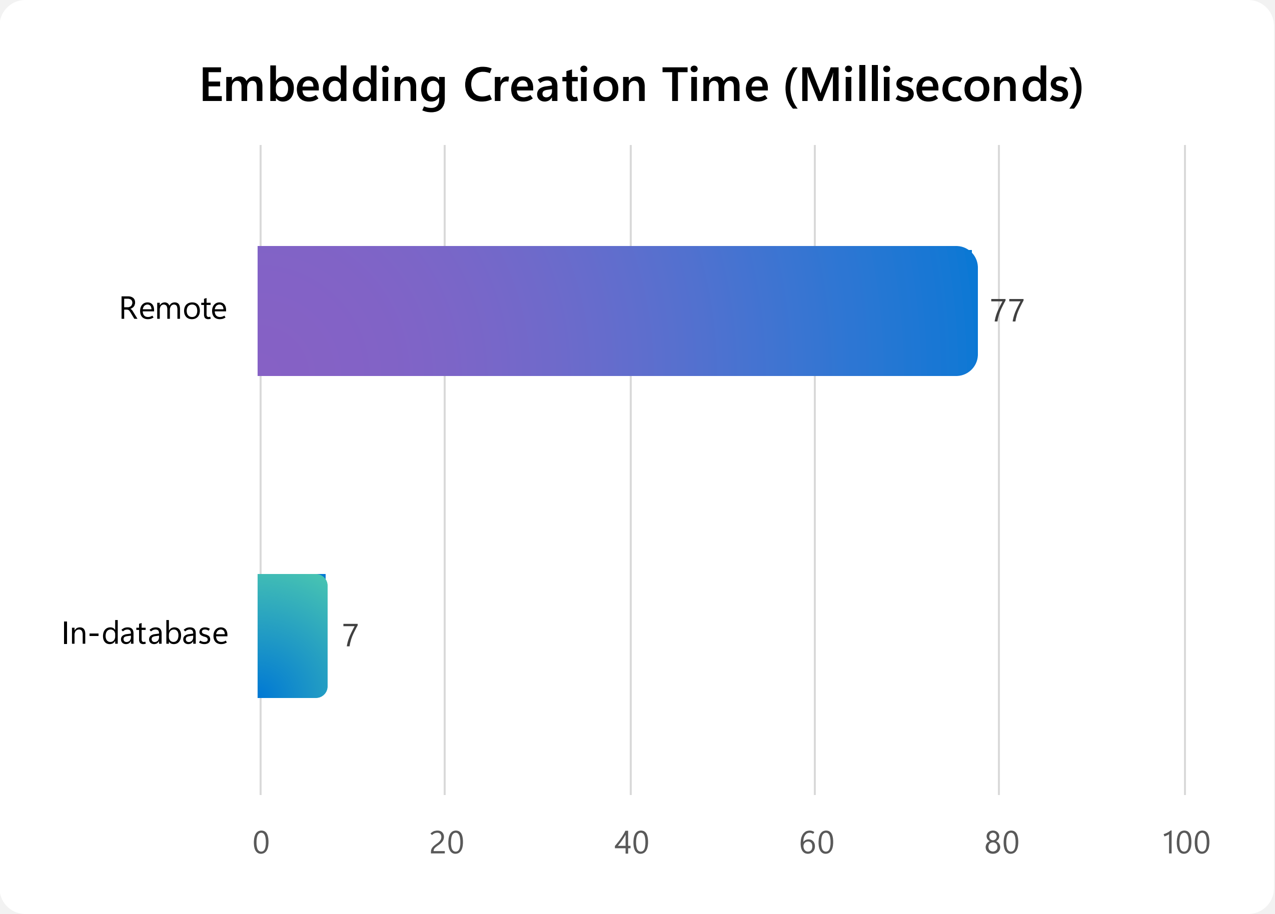
## Low-latency embedding creation for OLTP workloads

The **azure_local_ai extension** enables vector embeddings to be generated locally within the Postgres server

Based on the Microsoft **open-source E5** embedding model

Benefits:

- **~10x faster** creation time
- No external service setup, maintenance, or transaction costs
- All data remains **within Postgres**
- Perfect for workloads where the **underlying data changes frequently**



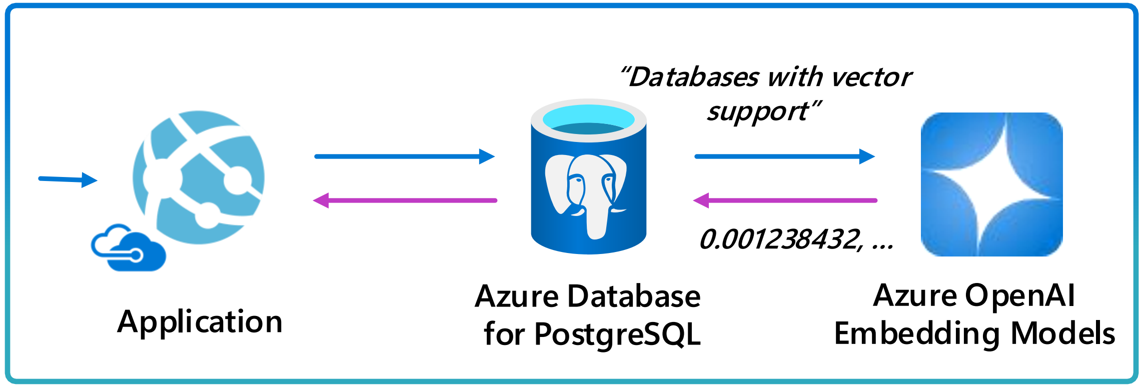
Create in-database embeddings with azure_local_ai extension - Azure Database for PostgreSQL - Flexible Server | Microsoft Learn

# Vector Generation

## Unique Remote + In-Database Embedding Models

### Remote Embedding Models

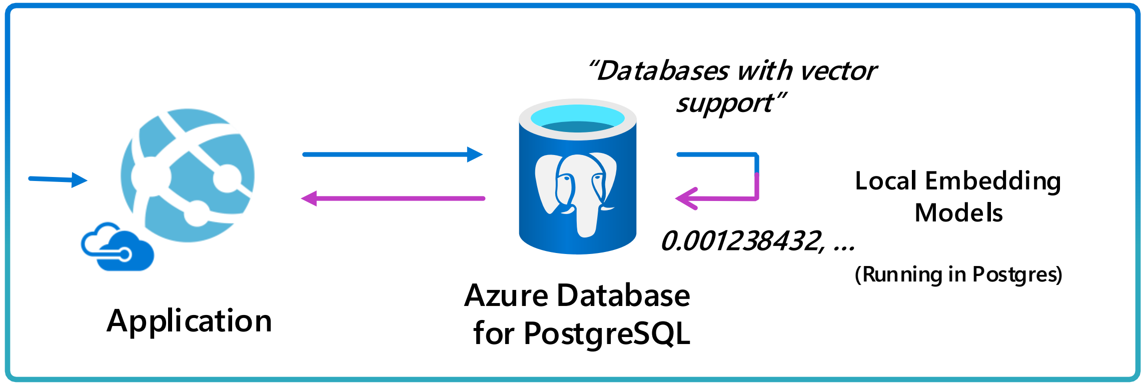
```
SELECT * FROM <table>
ORDER BY
database_description <->
azure_openai.create_embeddings(
‘text-embedding-ada-002’,
‘Databases with vector support’)
```



“Databases with vector support”
Application
Azure Database for PostgreSQL
0.001238432, ...
Azure OpenAI Embedding Models

### In-Database Embedding Models
(Preview)

```
SELECT * FROM <table>
ORDER BY
recipe_embedding <#>
azure_local_ai.create_embeddings(
‘multilingual-e5-small:v1’,
‘Databases with vector support’)
```



Announcing

# Mirroring Azure Database for PostgreSQL Flexible Server in Fabric!

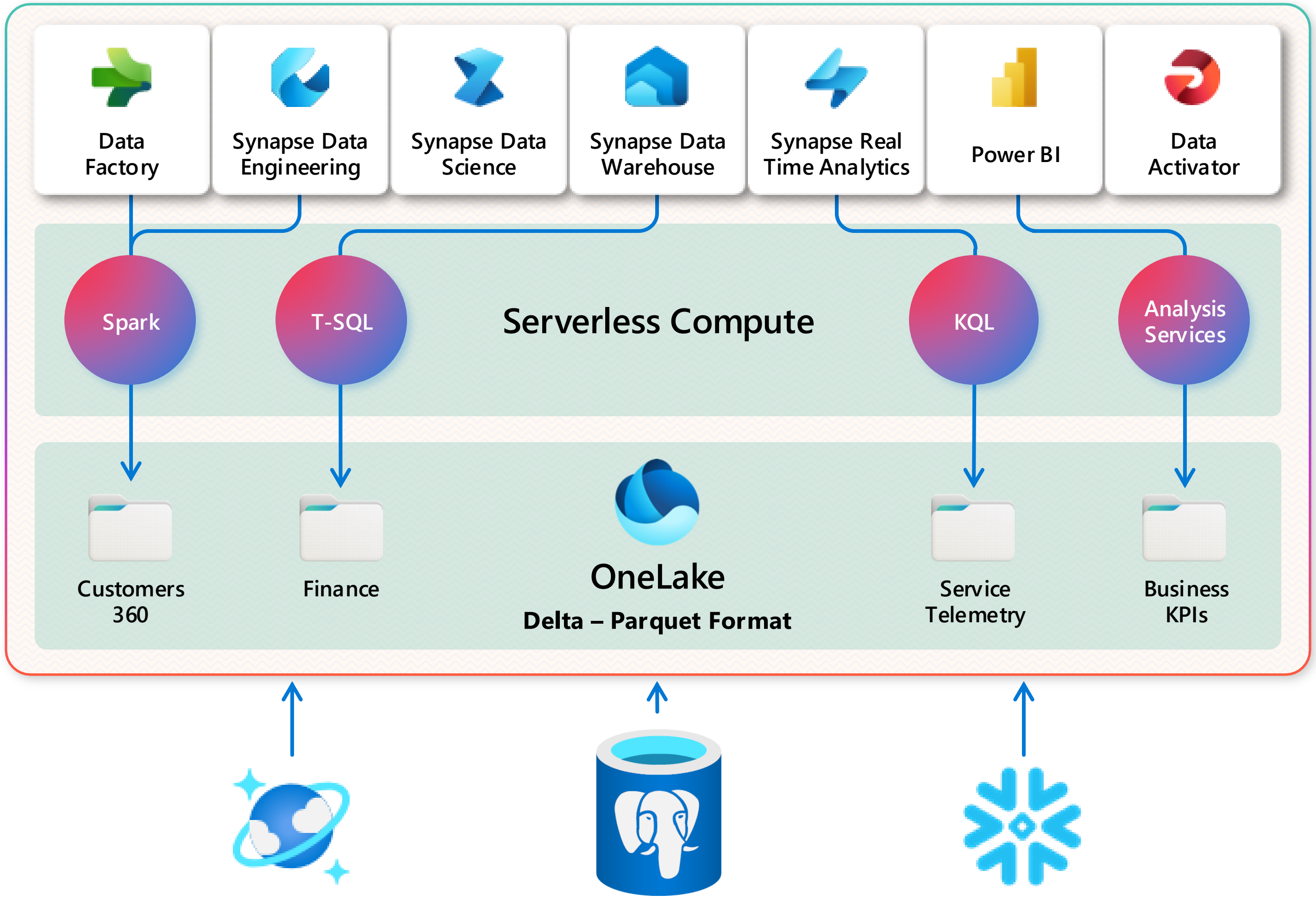
## Private Preview

Private preview form

https://forms.office.com/r/H0Mq1Yz8Pw

# Mirroring Azure Database for PostgreSQL Flexible Server

## Supercharge analytics time to value



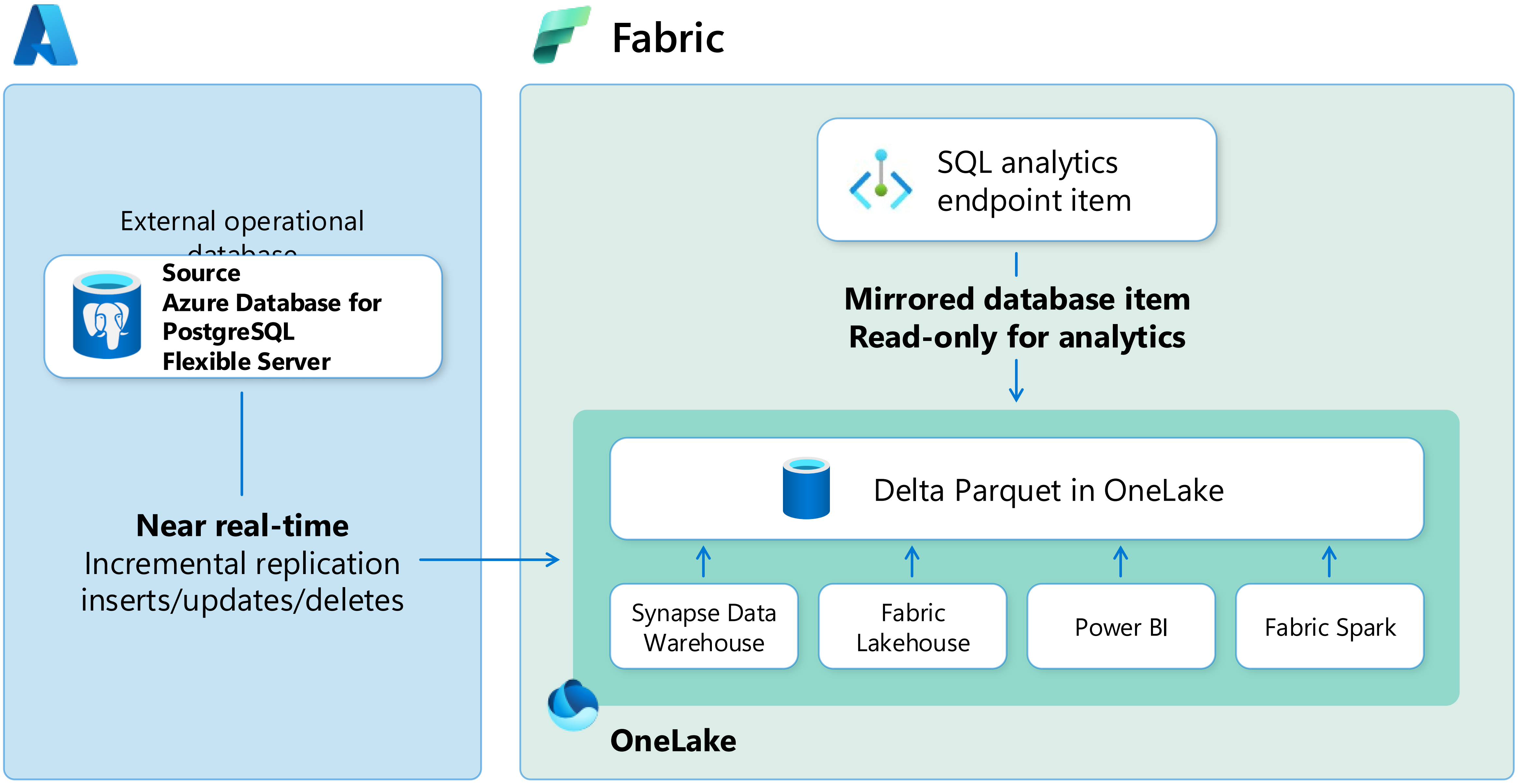
Mirroring replicates databases to Fabric with **zero ETL**

Compute replication is included with your Fabric capacity for **zero cost**.

**Free storage** costs based on Fabric capacity

Data is replicated into One Lake and **kept up-to-date in near real-time**

Mirroring **protects operational databases** from **analytical queries**



Fabric

External operational database

**Source**
**Azure Database for PostgreSQL Flexible Server**

**Near real-time**
Incremental replication
inserts/updates/deletes

SQL analytics endpoint item

**Mirrored database item**
**Read-only for analytics**

Delta Parquet in OneLake

Synapse Data Warehouse

Fabric Lakehouse

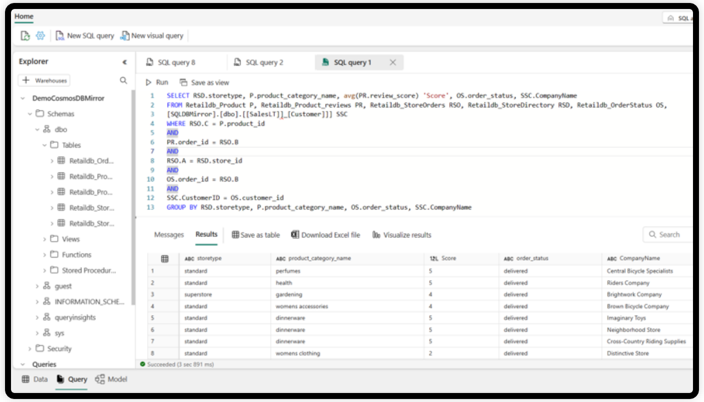
Power BI

Fabric Spark

**OneLake**

# Fabric Experiences Unlocked with Mirroring

## Cross-database querying with multi-cloud Shortcuts

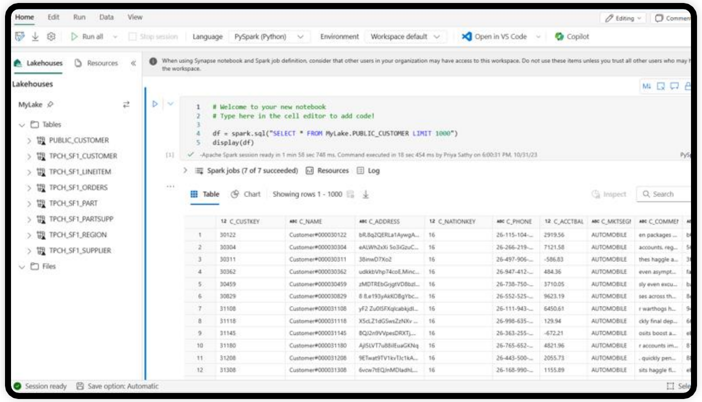


```sql
SELECT RSD.storetype, P.product_category_name, avg(PR.review_score) 'Score', OS.order_status, SSC.CompanyName
FROM Retaildb_Product P, Retaildb_Product_reviews PR, Retaildb_StoreOrders RSO, Retaildb_StoreDirectory RSD, Retaildb_OrderStatus OS,
[SQLDBMirror].[dbo].[[SalesLT]]_[Customer]]] SSC
WHERE RSO.C = P.product_id
AND
PR.order_id = RSO.B
AND
RSO.A = RSD.store_id
AND
OS.order_id = RSO.B
AND
SSC.CustomerID = OS.customer_id
GROUP BY RSD.storetype, P.product_category_name, OS.order_status, SSC.CompanyName
```

|  | storetype | product_category_name | Score | order_status | CompanyName |
|---|---|---|---|---|---|
| 1 | standard | perfumes | 5 | delivered | Central Bicycle Specialists |
| 2 | standard | health | 5 | delivered | Riders Company |
| 3 | superstore | gardening | 4 | delivered | Brightwork Company |
| 4 | standard | womens accessories | 4 | delivered | Brown Bicycle Company |
| 5 | standard | dinnerware | 5 | delivered | Imaginary Toys |
| 6 | standard | dinnerware | 5 | delivered | Neighborhood Store |
| 7 | standard | dinnerware | 5 | delivered | Cross-Country Riding Supplies |
| 8 | standard | womens clothing | 2 | delivered | Distinctive Store |

Succeeded (3 sec 891 ms)

## Data Science, ML & AI experiences



When using Synapse notebook and Spark job definition, consider that other users in your organization may have access to this workspace. Do not use these items unless you trust all other users who may the workspace.

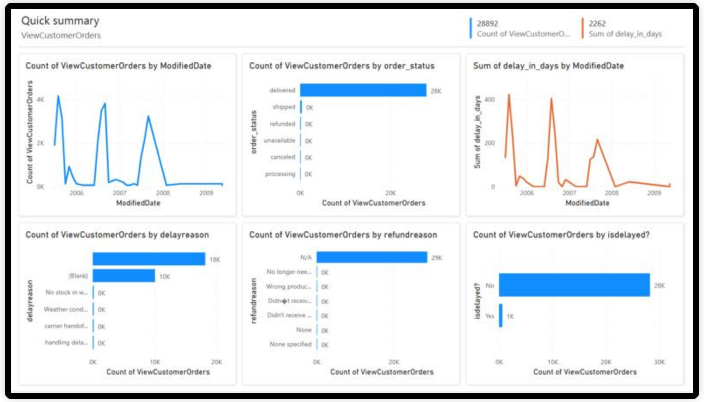
```python
# Welcome to your new notebook
# Type here in the cell editor to add code!

df = spark.sql("SELECT * FROM MyLake.PUBLIC_CUSTOMER LIMIT 1000")
display(df)
```

[1] -Apache Spark session ready in 1 min 58 sec 748 ms. Command executed in 18 sec 454 ms by Priya Sathy on 6:00:31 PM, 10/31/23

|  | C_CUSTKEY | C_NAME | C_ADDRESS | C_NATIONKEY | C_PHONE | C_ACCTBAL | C_MKTSEGI | C_COMMEI |
|---|---|---|---|---|---|---|---|---|
| 1 | 30122 | Customer#000030122 | bR.8q2QERLa1AywgA... | 16 | 26-115-104-... | 2919.56 | AUTOMOBILE | en packages ... |
| 2 | 30304 | Customer#000030304 | eALWh2xXi 5o3iGzuC... | 16 | 26-266-219-... | 7121.58 | AUTOMOBILE | accounts. reg... |
| 3 | 30311 | Customer#000030311 | 38inwD7Xo2 | 16 | 26-497-906-... | -586.83 | AUTOMOBILE | thes haggle a... |
| 4 | 30362 | Customer#000030362 | udkkbVhp74coE,Minc... | 16 | 26-947-412-... | 484.36 | AUTOMOBILE | even asympt... |
| 5 | 30459 | Customer#000030459 | zMDTREbGrjgtVD8bzl... | 16 | 26-738-750-... | 3710.05 | AUTOMOBILE | sly even excu... |
| 6 | 30829 | Customer#000030829 | 8 8.e193yAkKOBgYbc... | 16 | 26-552-525-... | 9623.19 | AUTOMOBILE | ses across th... |
| 7 | 31108 | Customer#000031108 | yF2 Zu0ISFXqlcabkjdl... | 16 | 26-111-943-... | 6450.61 | AUTOMOBILE | r warthogs h... |
| 8 | 31118 | Customer#000031118 | X5cLZ1dGSwsZzNXv ... | 16 | 26-998-635-... | 129.94 | AUTOMOBILE | ckly final dep... |
| 9 | 31145 | Customer#000031145 | BQJ2n9VVpesDRXTj... | 16 | 26-363-255-... | -672.21 | AUTOMOBILE | osits boost a... |
| 10 | 31180 | Customer#000031180 | AjI5LVT7u88ilEuaGKNq | 16 | 26-765-652-... | 4821.96 | AUTOMOBILE | r accounts im... |
| 11 | 31208 | Customer#000031208 | 9ETwat9TV1kvTJc1kA... | 16 | 26-443-500-... | 2055.73 | AUTOMOBILE | . quickly pen... |
| 12 | 31308 | Customer#000031308 | 6vcw7tEQJnMDladhL... | 16 | 26-168-990-... | 1155.89 | AUTOMOBILE | sits haggle fl... |

## Power BI Direct Lake mode

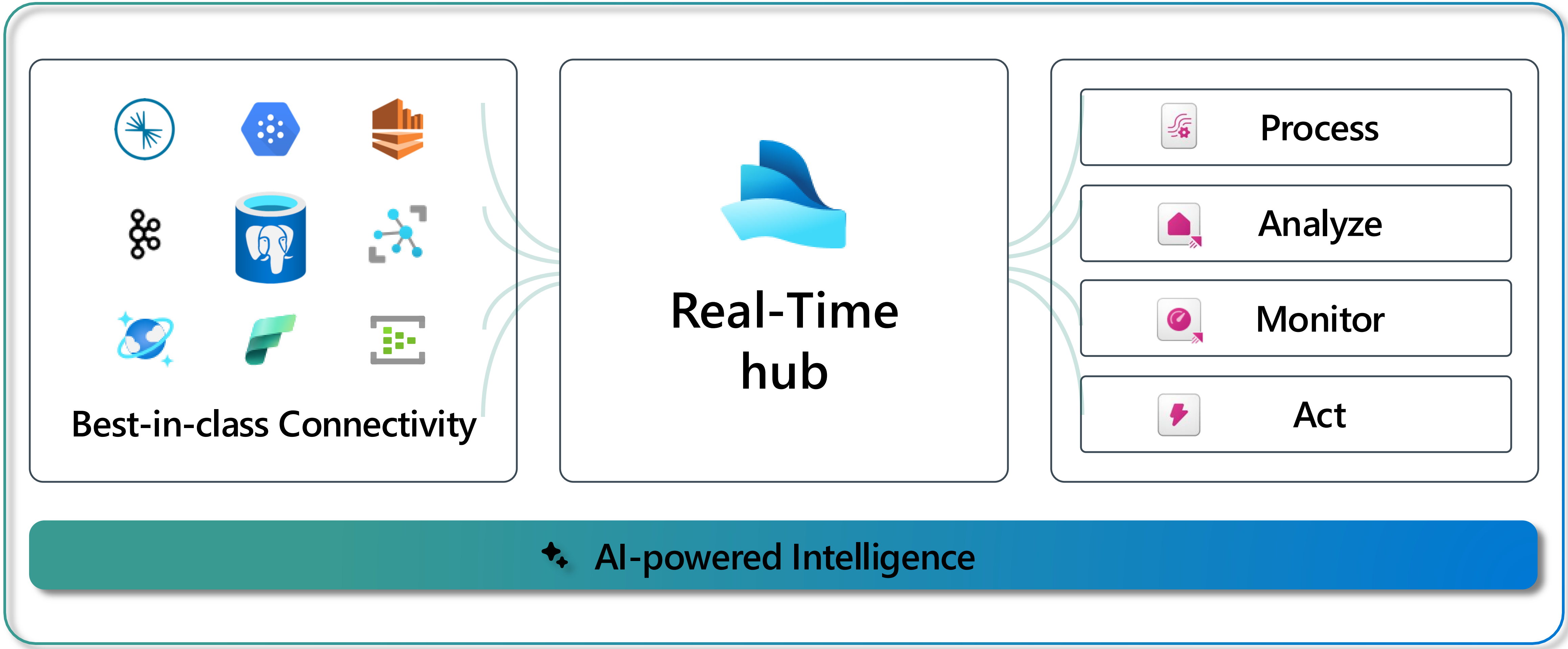


# PostgreSQL Database Change Data Capture (CDC) source connector for Microsoft Fabric

## Public Preview

Add PostgreSQL Database CDC source to an eventstream - Microsoft Fabric | Microsoft Learn

# Real-Time Intelligence in Microsoft Fabric



Best-in-class Connectivity

Real-Time hub

Process

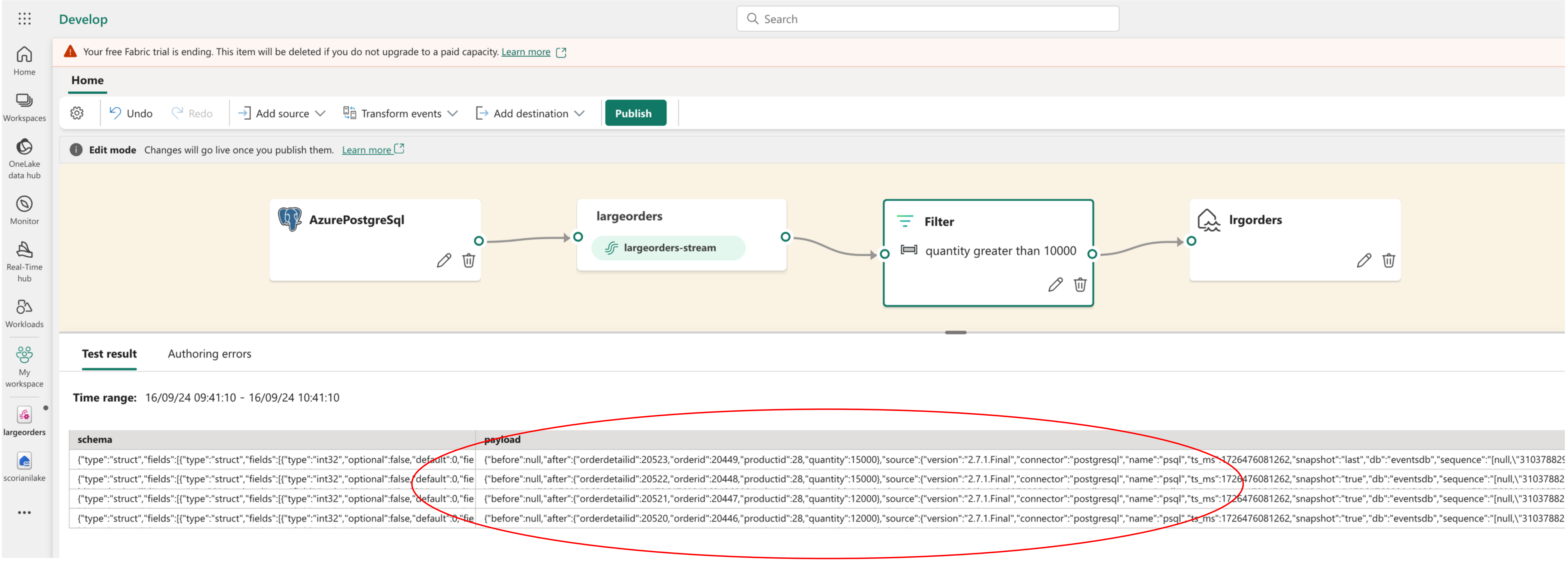
Analyze

Monitor

Act

AI-powered Intelligence

# Real-Time Intelligence with PostgreSQL data



Your free Fabric trial is ending. This item will be deleted if you do not upgrade to a paid capacity. Learn more

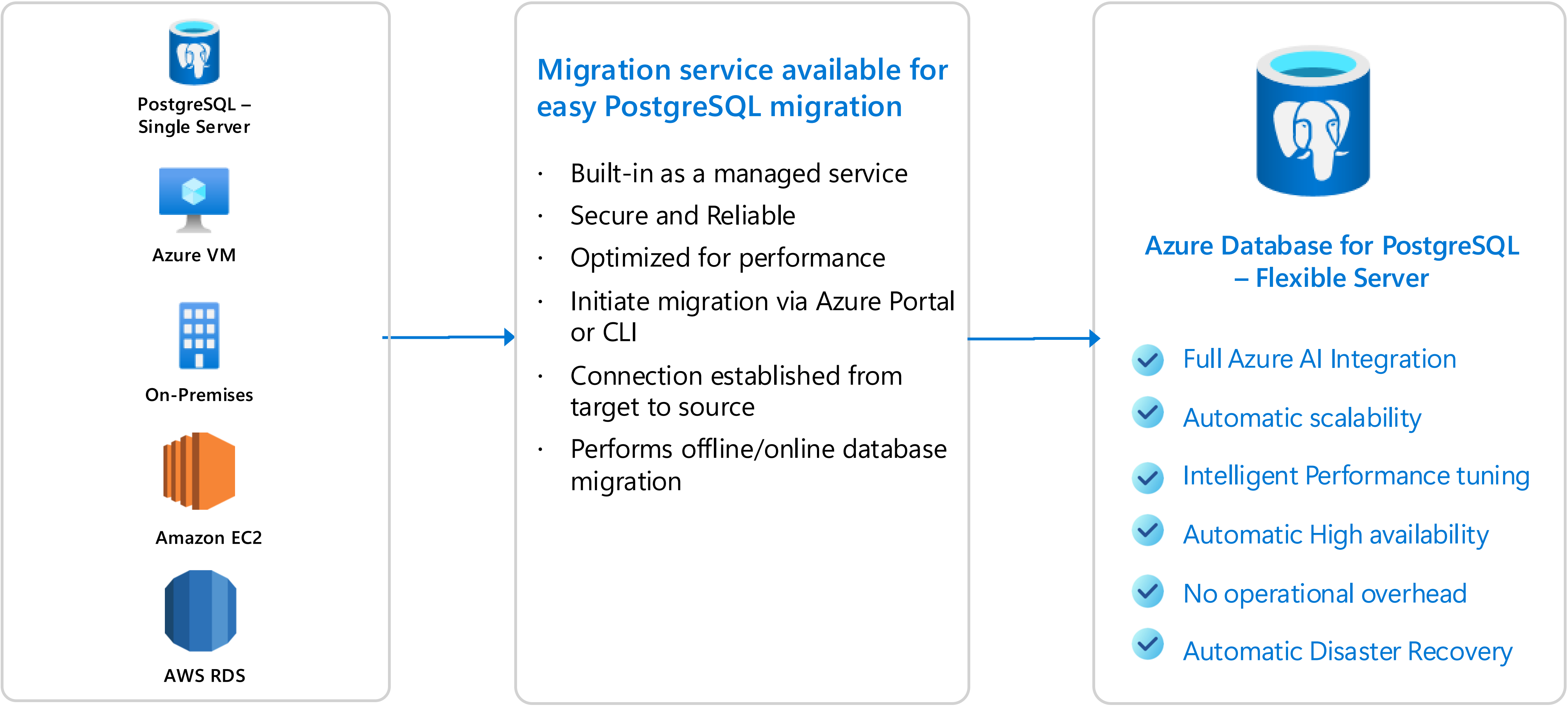
**Edit mode** Changes will go live once you publish them. Learn more

**Time range:** 16/09/24 09:41:10 - 16/09/24 10:41:10

| schema | payload |
|---|---|
| {"type":"struct","fields":[{"type":"struct","fields":[{"type":"int32","optional":false,"default":0,"fie | {"before":null,"after":{"orderdetailid":20523,"orderid":20449,"productid":28,"quantity":15000},"source":{"version":"2.7.1.Final","connector":"postgresql","name":"psql","ts_ms":1726476081262,"snapshot":"last","db":"eventsdb","sequence":"[null,\"310378829 |
| {"type":"struct","fields":[{"type":"struct","fields":[{"type":"int32","optional":false,"default":0,"fie | {"before":null,"after":{"orderdetailid":20522,"orderid":20448,"productid":28,"quantity":15000},"source":{"version":"2.7.1.Final","connector":"postgresql","name":"psql","ts_ms":1726476081262,"snapshot":"true","db":"eventsdb","sequence":"[null,\"31037882 |
| {"type":"struct","fields":[{"type":"struct","fields":[{"type":"int32","optional":false,"default":0,"fie | {"before":null,"after":{"orderdetailid":20521,"orderid":20447,"productid":28,"quantity":12000},"source":{"version":"2.7.1.Final","connector":"postgresql","name":"psql","ts_ms":1726476081262,"snapshot":"true","db":"eventsdb","sequence":"[null,\"31037882 |
| {"type":"struct","fields":[{"type":"struct","fields":[{"type":"int32","optional":false,"default":0,"fie | {"before":null,"after":{"orderdetailid":20520,"orderid":20446,"productid":28,"quantity":12000},"source":{"version":"2.7.1.Final","connector":"postgresql","name":"psql","ts_ms":1726476081262,"snapshot":"true","db":"eventsdb","sequence":"[null,\"31037882 |

# Built-in migration tools

## Migrate from Postgres anywhere to Flexible Server with ease



PostgreSQL – Single Server

Azure VM

On-Premises

Amazon EC2

AWS RDS

### Migration service available for easy PostgreSQL migration

- Built-in as a managed service
- Secure and Reliable
- Optimized for performance
- Initiate migration via Azure Portal or CLI
- Connection established from target to source
- Performs offline/online database migration

### Azure Database for PostgreSQL – Flexible Server

- ✓ Full Azure AI Integration
- ✓ Automatic scalability
- ✓ Intelligent Performance tuning
- ✓ Automatic High availability
- ✓ No operational overhead
- ✓ Automatic Disaster Recovery

*More sources coming soon (GCP & AWS Aurora coming in few weeks)

# Azure PostgreSQL Resources

**Azure Database for Postgres homepage**
http://aka.ms/postgres

**Azure Database for Postgres Docs**
http://aka.ms/postgresdocs

**Azure Postgres on X**
@AzureDBPostgres

**Azure Postgres Feedback Forum**
https://aka.ms/pgfeedback

**Get Started for Free with an Azure Free Account**
https://aka.ms/try-postgres-free

**Azure Database for Postgres Blog**
https://aka.ms/azurepostgresblog

**Azure Postgres on LinkedIn**
https://www.linkedin.com/company/azure-database-for-postgresql/

**AskAzurePostgreSQL**
AskAzurePostgreSQL@microsoft.com

Microsoft

# Thank you