Microsoft

Building Distributed PostgreSQL Apps using Citus 11

Charles Feddersen

Group Product Manager – Citus & Cosmos DB for PostgreSQL

October 2022

Agenda

1. Quick intro (or refresher) on Citus
2. Dispelling the great distributed database myth
3. Building distributed databases
4. Distributed demos
5. Getting Started

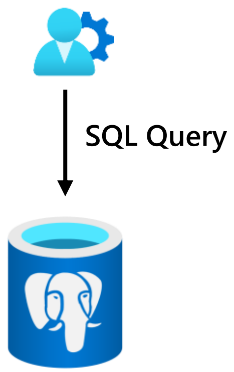
Quick intro (or refresher) on Citus

Citus powers Distributed SQL on open-source PostgreSQL

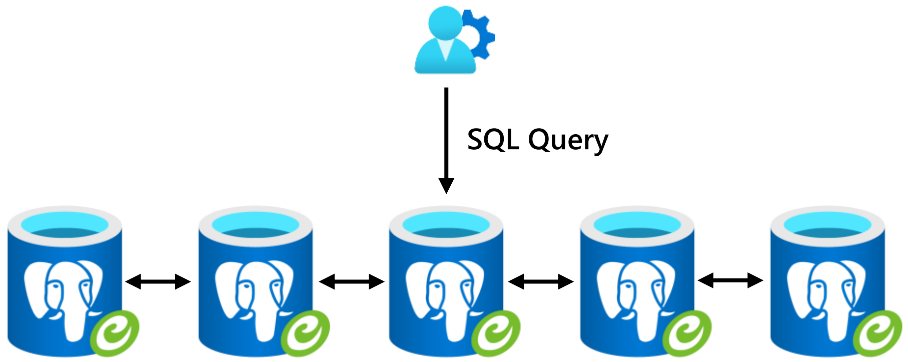
Provides simple scale-out of operational workloads to execute on a cluster on machines

Single connection – no code changes to the app

Scale locally across a cluster, or globally using replication



Traditional PostgreSQL
Single Node Scale



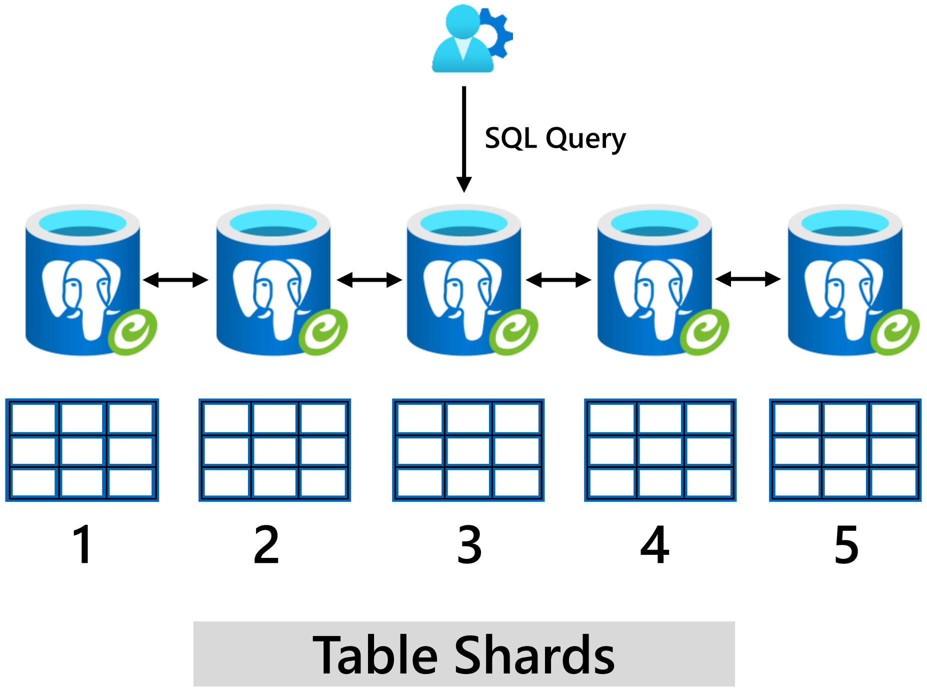
Distributed PostgreSQL
Cluster Scale

Citus powers Distributed SQL on open-source PostgreSQL

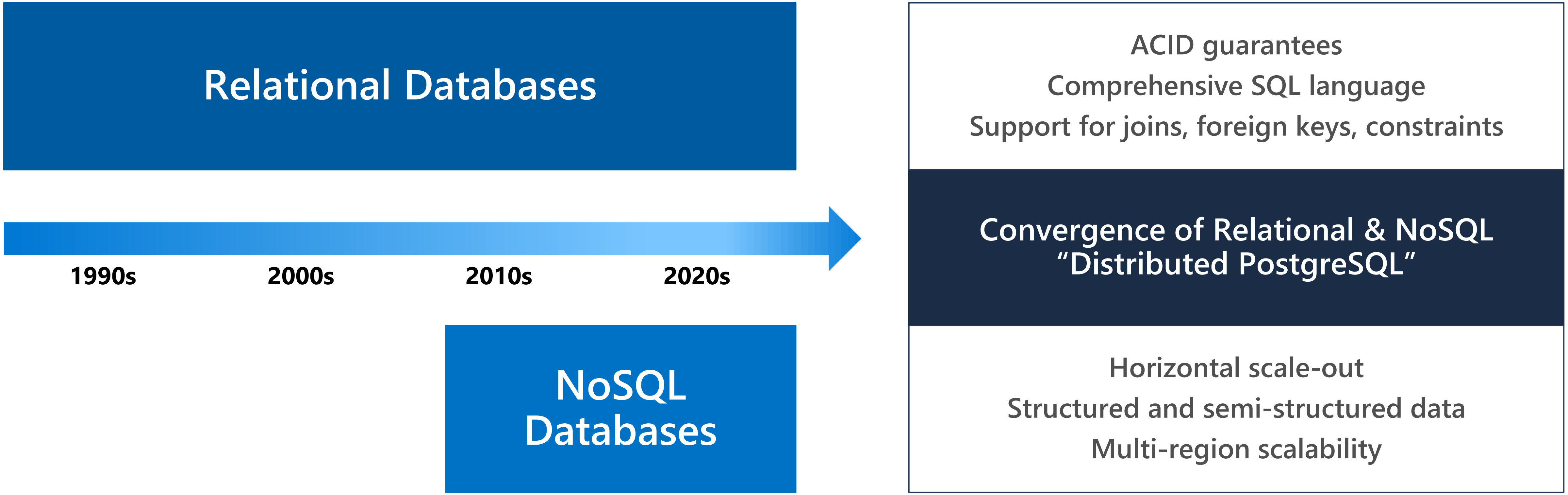
Provides simple scale-out of operational workloads to execute on a cluster on machines

Single connection – no code changes to the app

Scale locally across a cluster, or globally using replication



Evolution to Distributed SQL



Relational Databases

1990s

2000s

2010s

2020s

NoSQL Databases

ACID guarantees

Comprehensive SQL language

Support for joins, foreign keys, constraints

Convergence of Relational & NoSQL
“Distributed PostgreSQL”

Horizontal scale-out

Structured and semi-structured data

Multi-region scalability

Dispelling the great distributed database myth

Distributed Systems are not only for large data volumes

Easily the most common misconception about distributed databases is that they are only applicable to large data volumes

The definition of “large” varies depending who you talk to

However,

Distributed databases are incredibly powerful even for small data volumes

Distributed Systems are not “big” single node systems

There are several common database bottlenecks that single node systems encounter

Infrastructure

CPU – Increase total amount of read/write compute

Cache – Increase cache hit ratio

IO – Shared nothing architecture adds IO with new nodes

Database Operations

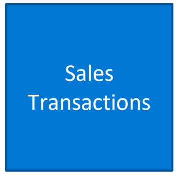
Queries – Reduce scans by isolating onto a shard

Data Modification – parallelize heavy operations

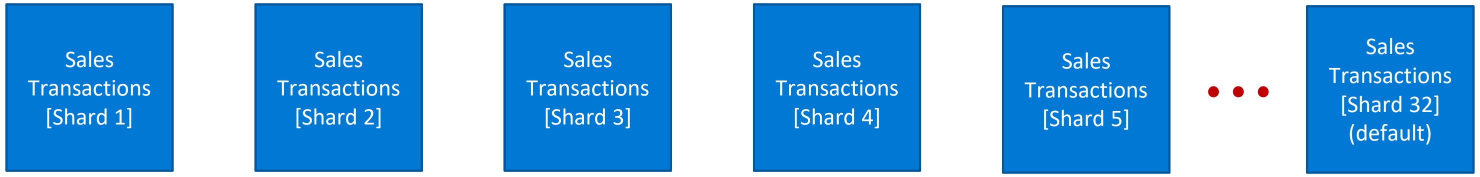
Maintenance – parallelize tasks such as backup

Creating distributed tables

```
CREATE TABLE
SalesTransactions
```

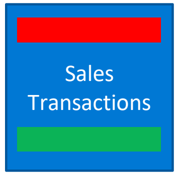


```
SELECT
create_distributed_table
```

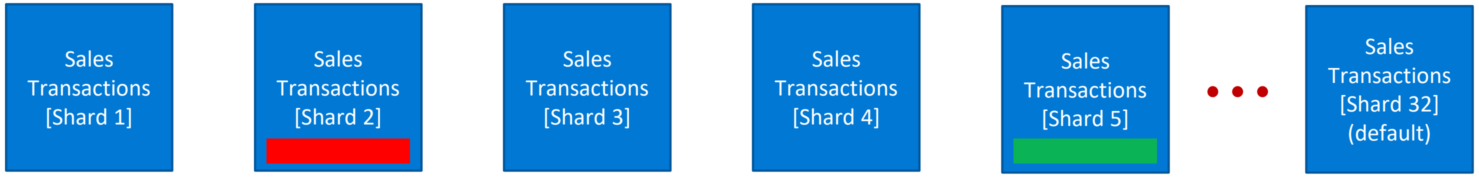


Creating distributed tables

```
CREATE TABLE
SalesTransactions
```



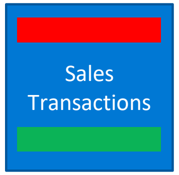
```
SELECT
create_distributed_table
```



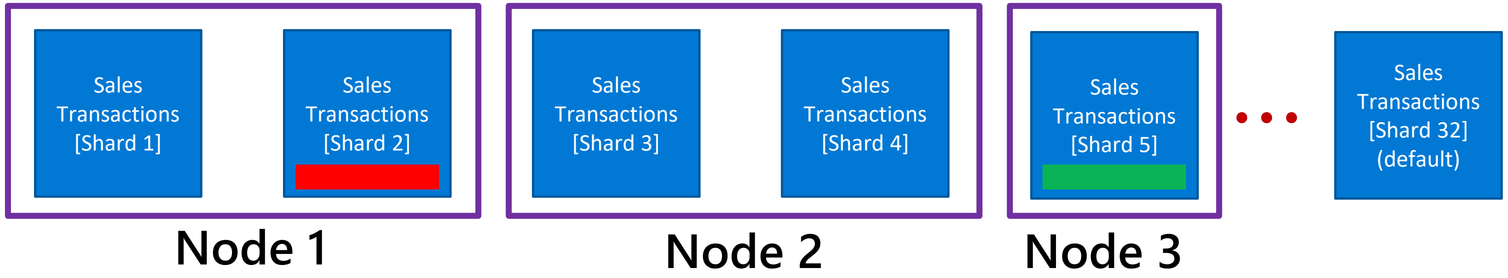
```
INSERT INTO SalesTransactions VALUES (1,1,10.23)
INSERT INTO SalesTransactions VALUES (2,3,17.94)
```

Creating distributed tables

```
CREATE TABLE
SalesTransactions
```



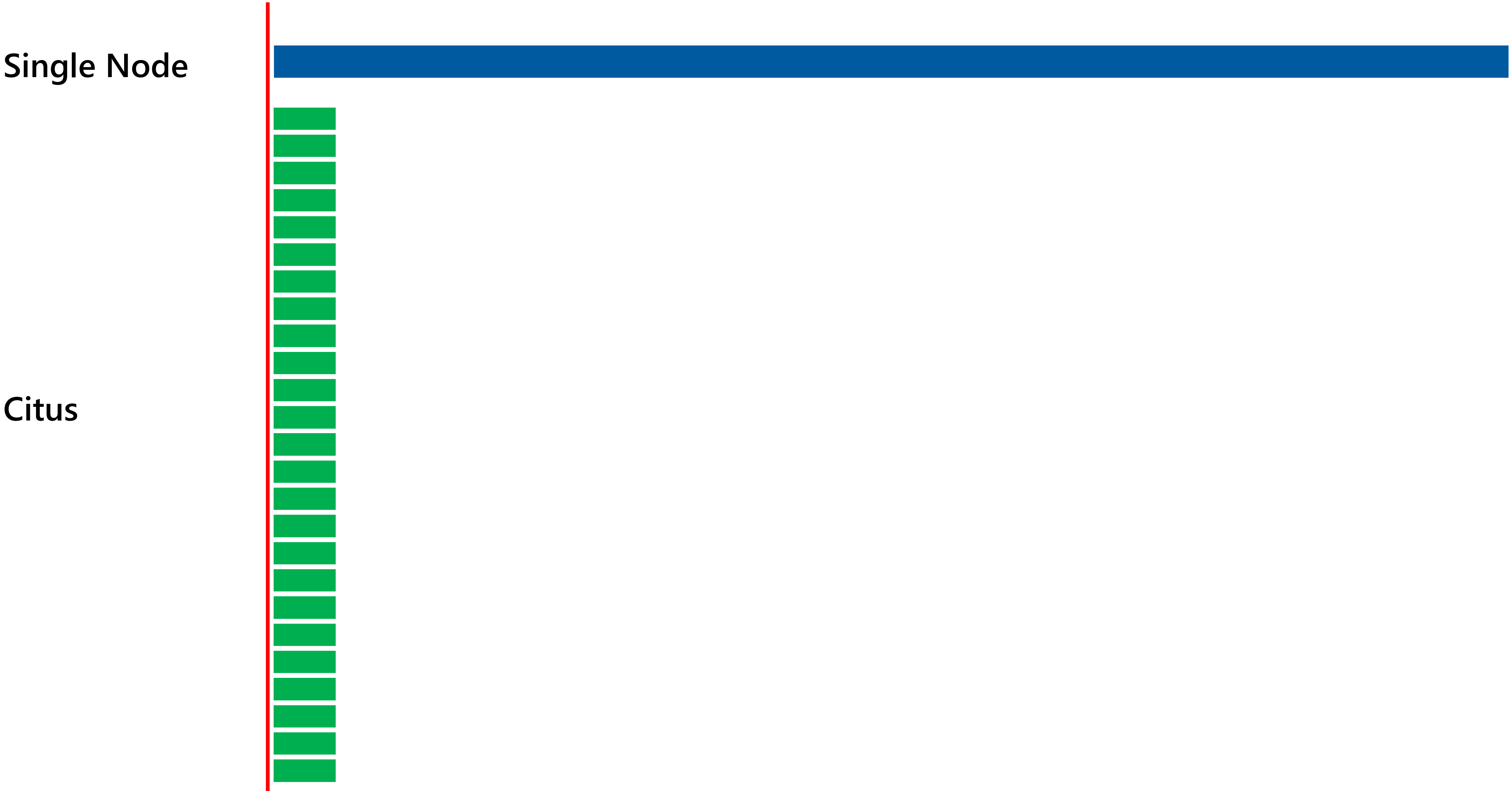
```
SELECT
create_distributed_table
```


Sales Transactions [Shard 1]
Sales Transactions [Shard 2]
Node 1
Sales Transactions [Shard 3]
Sales Transactions [Shard 4]
Node 2
Sales Transactions [Shard 5]
Node 3
• • •
Sales Transactions [Shard 32] (default)

```
INSERT INTO SalesTransactions VALUES (1,1,10.23)
INSERT INTO SalesTransactions VALUES (2,3,17.94)
```

DEMO: Single node vs. Parallel Update

Parallelism at work



Single Node

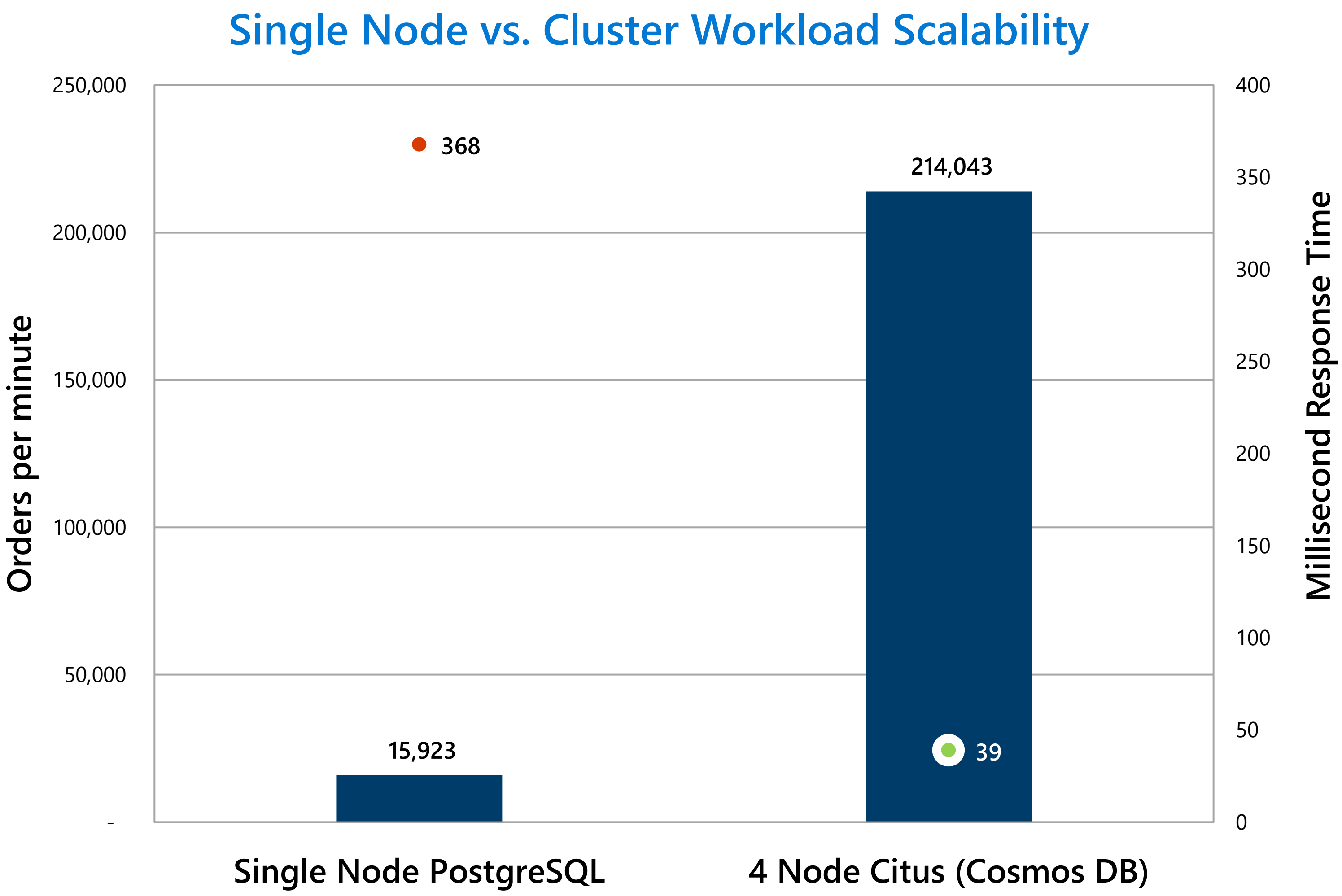
Citus

Parallelism to improve cache hit ratio

Relational database performance is highly sensitive to cache hit ratio

Performance can "fall off a cliff" once cache hit ration drops

Scale out can provide better than linear performance improvement where cache is under pressure



https://aka.ms/pg-hyperscale-perf

Building distributed databases

Building distributed databases

If built well, the scalability of distributed databases is magical

- Virtually linear scalability
- Well balanced across compute nodes
- Maximize local execution on the compute nodes (more on the next slide

…and if done wrong, they will slow crawl and waste an resource

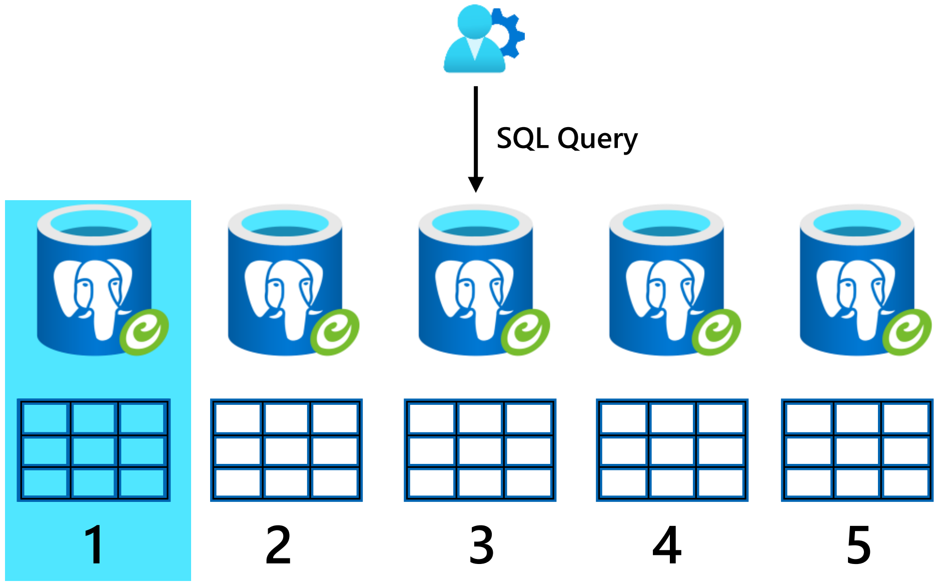
Maximizing local execution for performance and scale

1. Filter on shard column

```
CREATE TABLE Customer (
TenantId int …)

SELECT
create_distributed_table(
'customer', 'customerid’);

SELECT COUNT(*) FROM Customer
WHERE customerid = 1
```



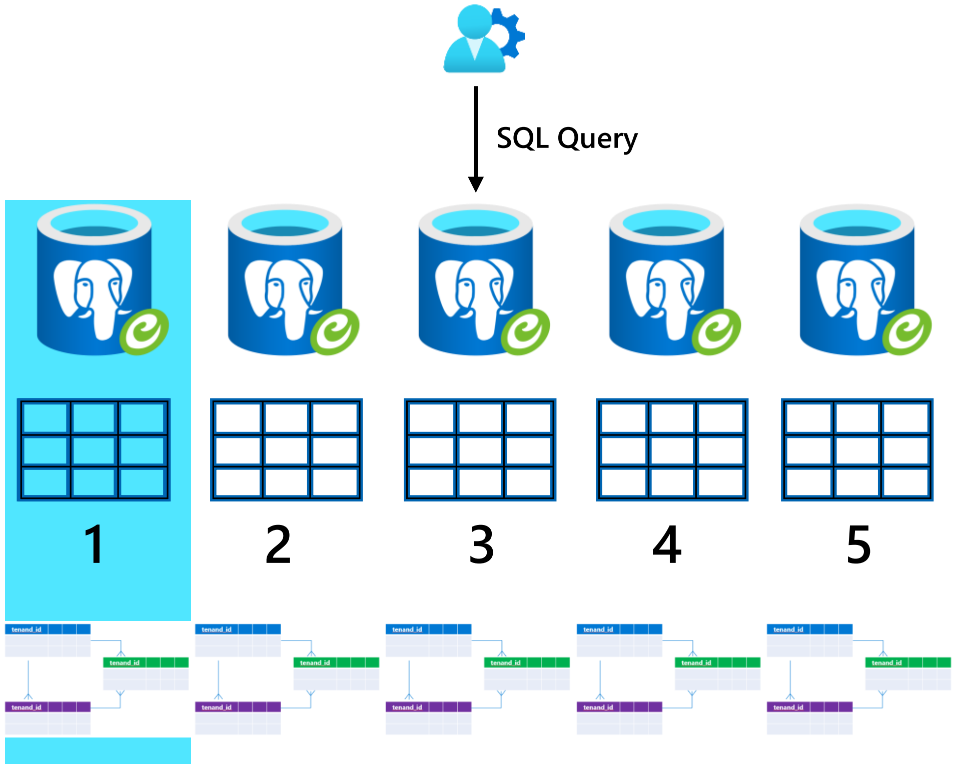
Maximizing local execution for performance and scale

2. Primary/foreign key checks within compute node

```
CREATE TABLE Customer (
TenantId int …)

CREATE TABLE SalesTransactions(
TenantId int …)

ALTER TABLE SalesTransactions
ADD CONSTRAINT fk_tid_cid
```



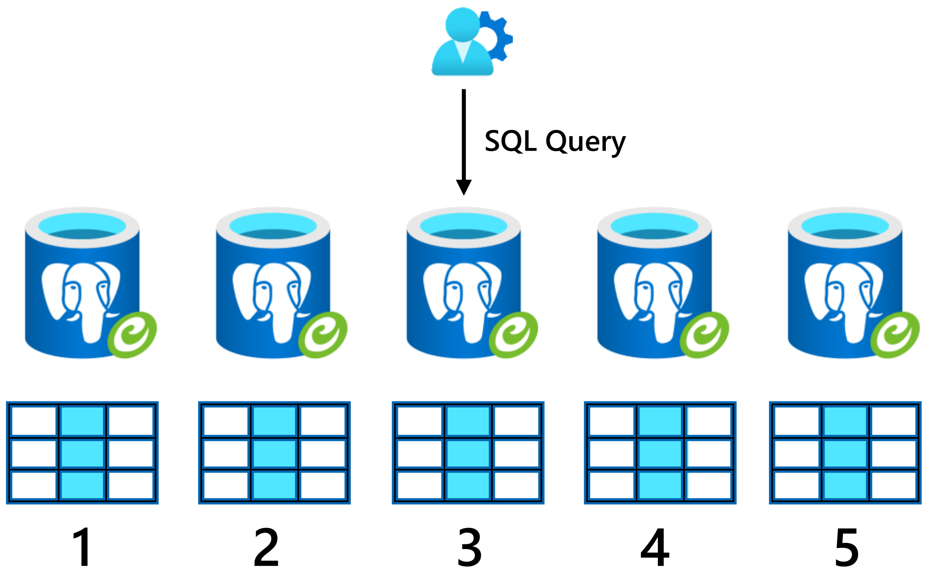
Maximizing local execution for performance and scale

3. Unique constraints within compute node

```
CREATE TABLE Customer (
TenantId int …)

CREATE TABLE SalesTransactions(
TenantId int …)

ALTER TABLE SalesTransactions
ADD CONSTRAINT fk_tid_cid
```



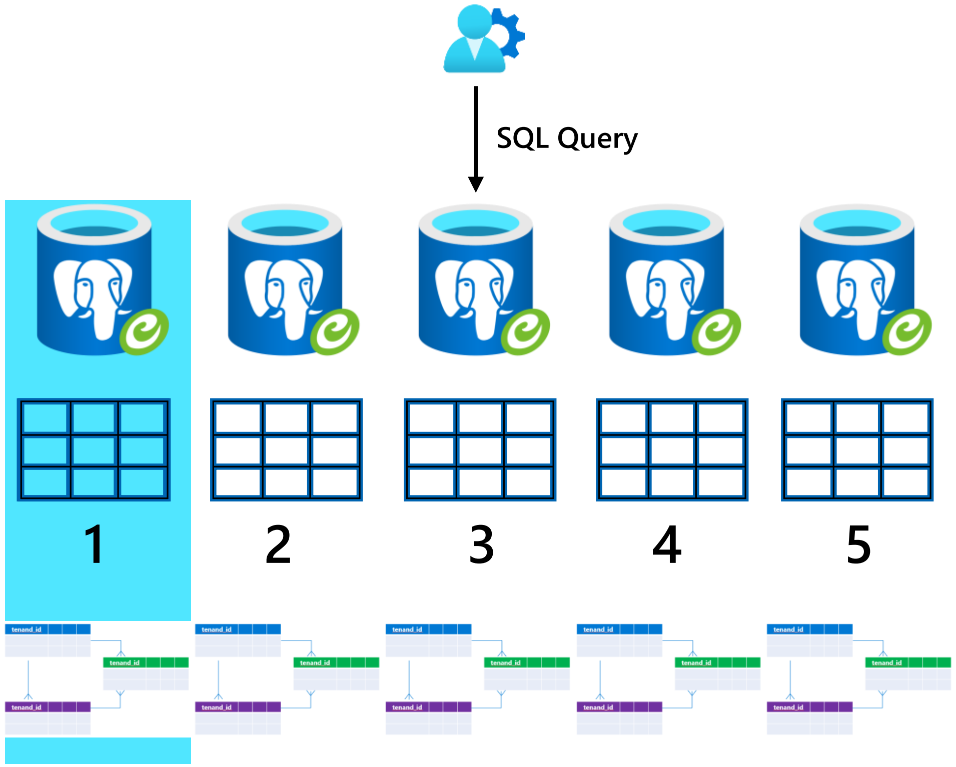
Maximizing local execution for performance and scale

4. Joins within compute node

```
CREATE TABLE Customer (
TenantId int …)

CREATE TABLE SalesTransactions(
TenantId int …)

FROM
    SalesTransactions sa
INNER JOIN Customer cu ON
    cu.TenantId = sa.TenantId AND
```

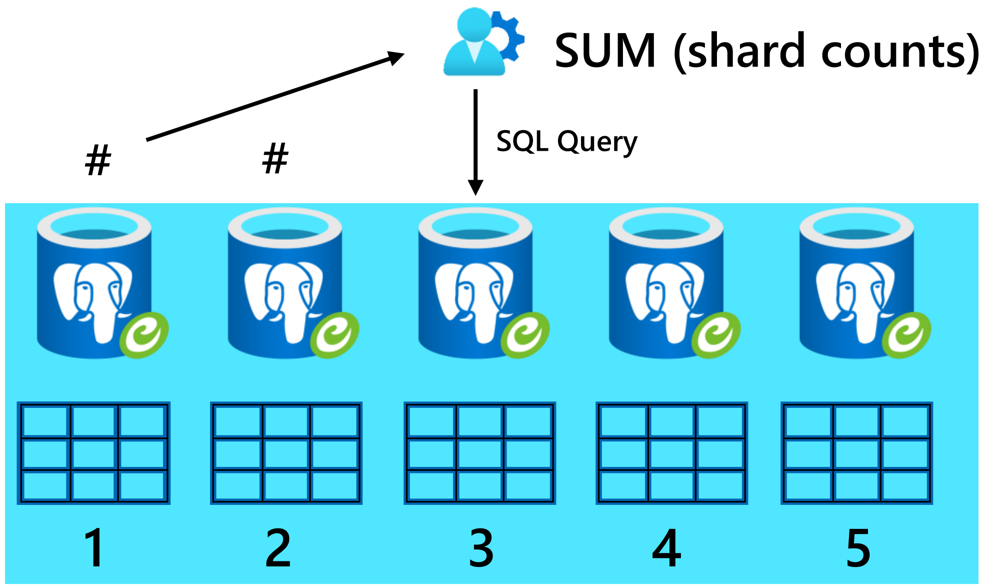


Maximizing local execution for performance and scale

2. Local/Global aggregates for HTAP

```
CREATE TABLE SalesTransactions(
TenantId int …)

SELECT COUNT(*)
FROM SalesTransactions
```

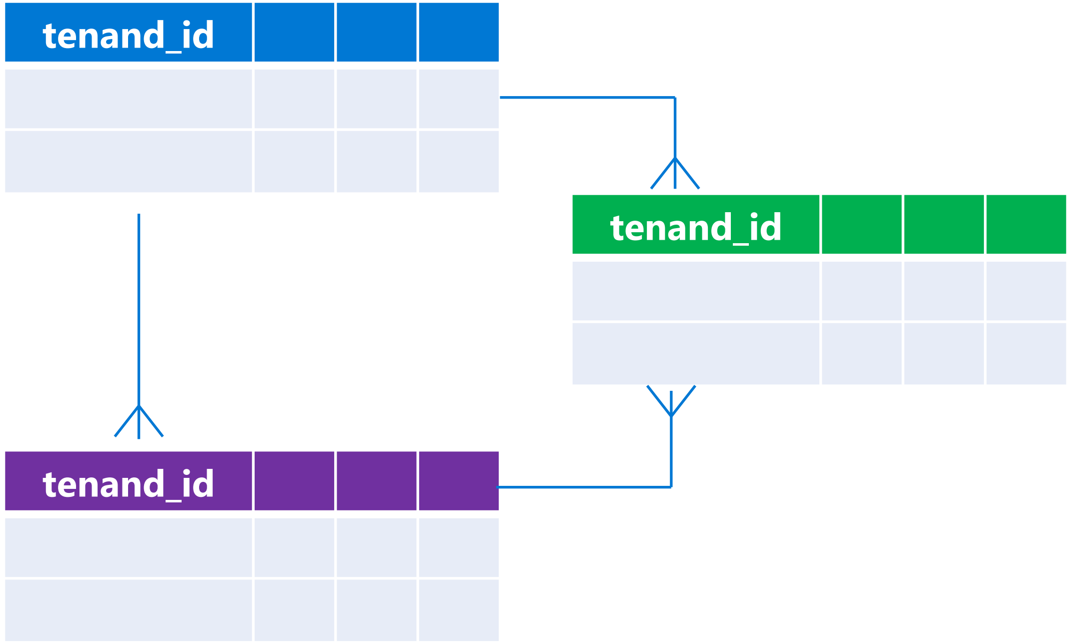


Distributed COUNT(*)

Applications that benefit from these features

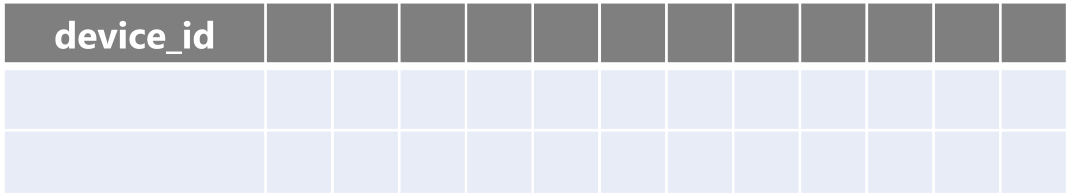
Multi-tenant SaaS

Complex models with common tenant_id shard key



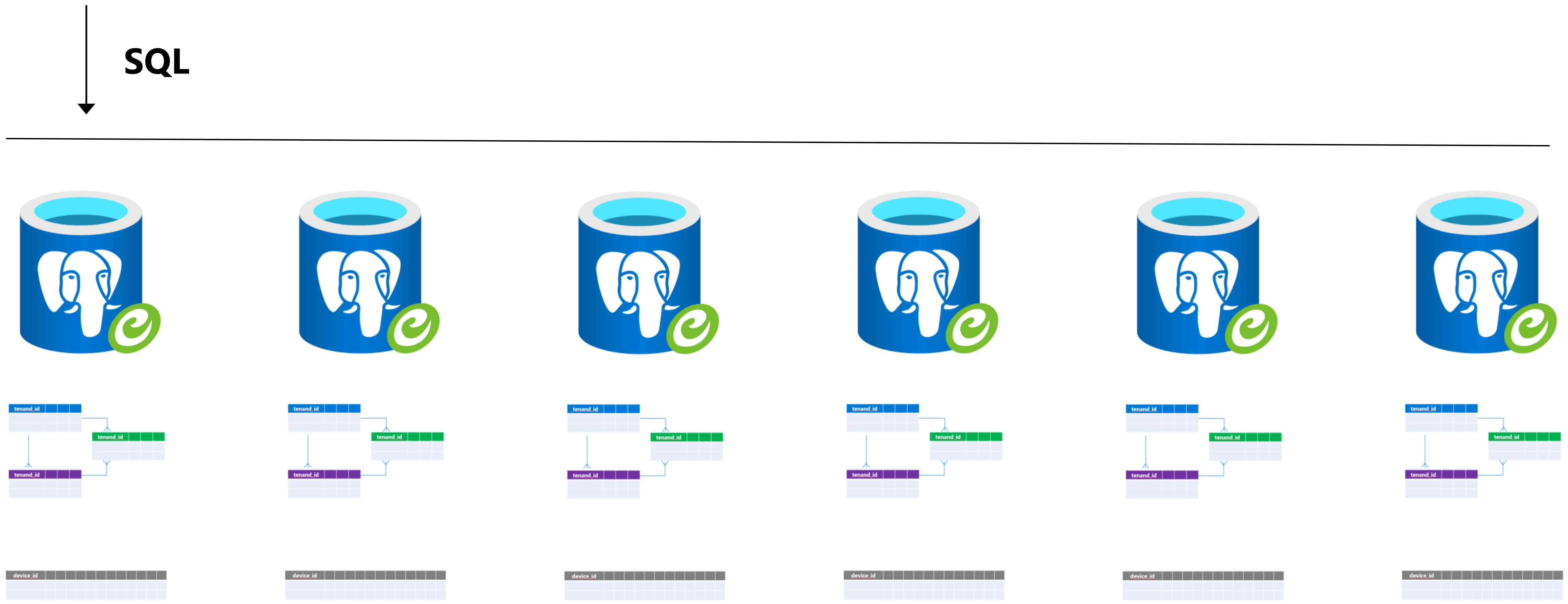
IoT

Simple model (often no relationasips) sharded on device_id



DEMO: Optimizing Distributed Data

But I can do this on a single node system…



SQL

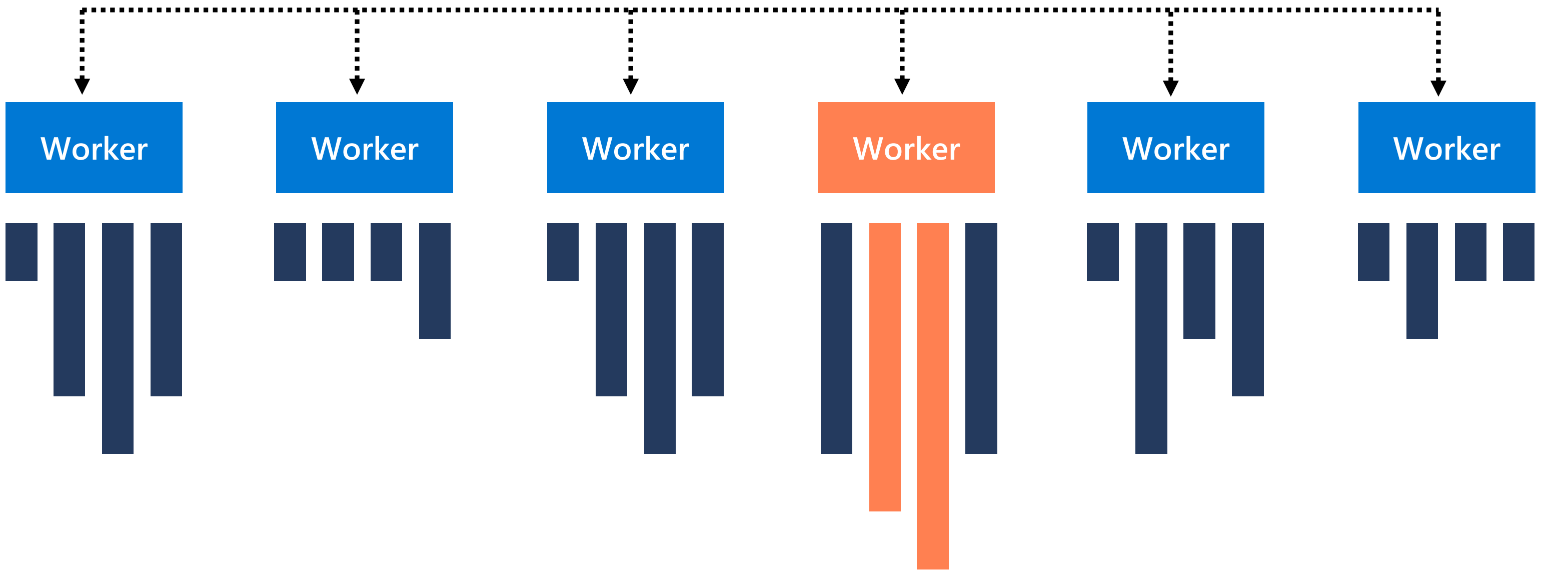
Some options to fix it when things slow down

Workload Rebalancing

Online rebalancing of shards to less utilized nodes

Shard move or shard split policies supported

Citus Cluster



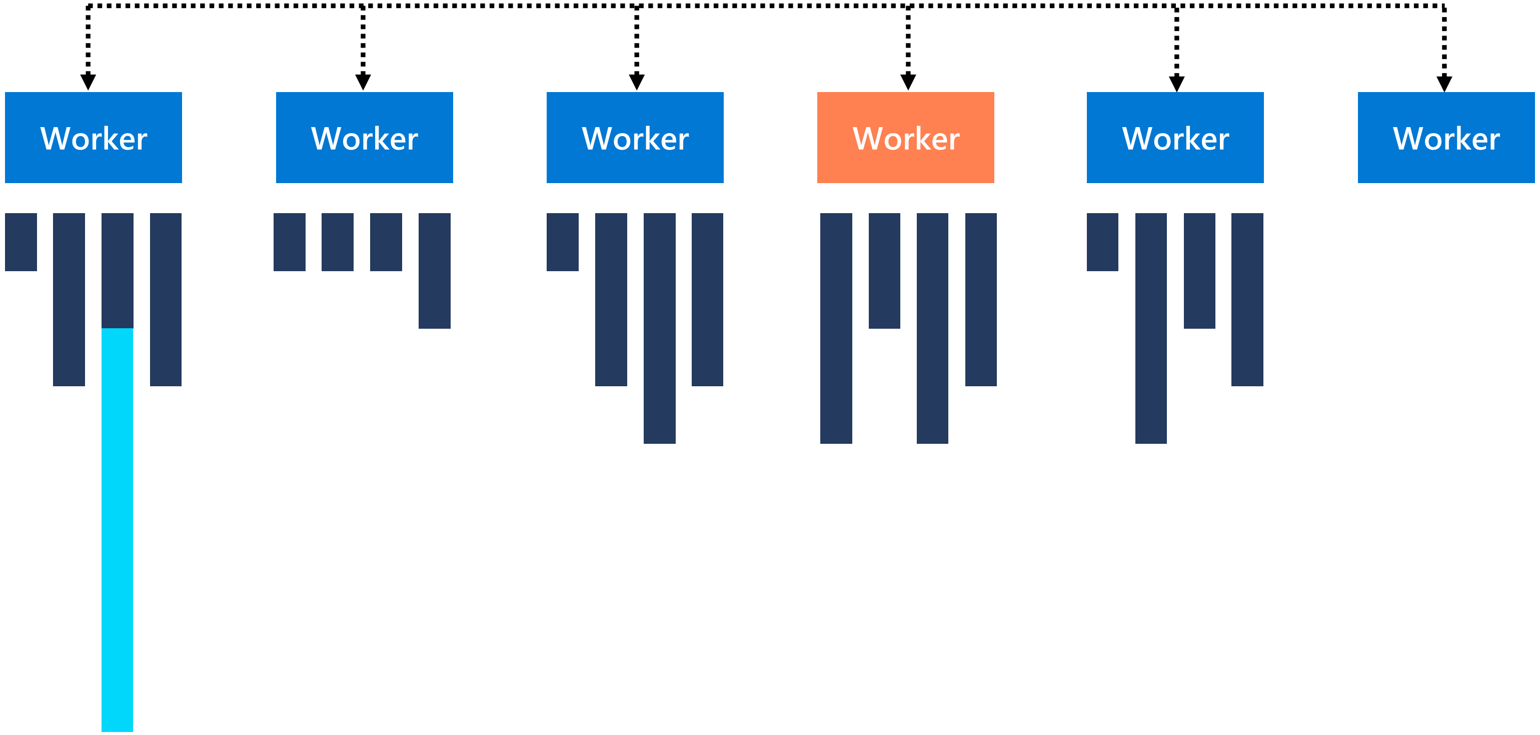
Online Tenant Isolation

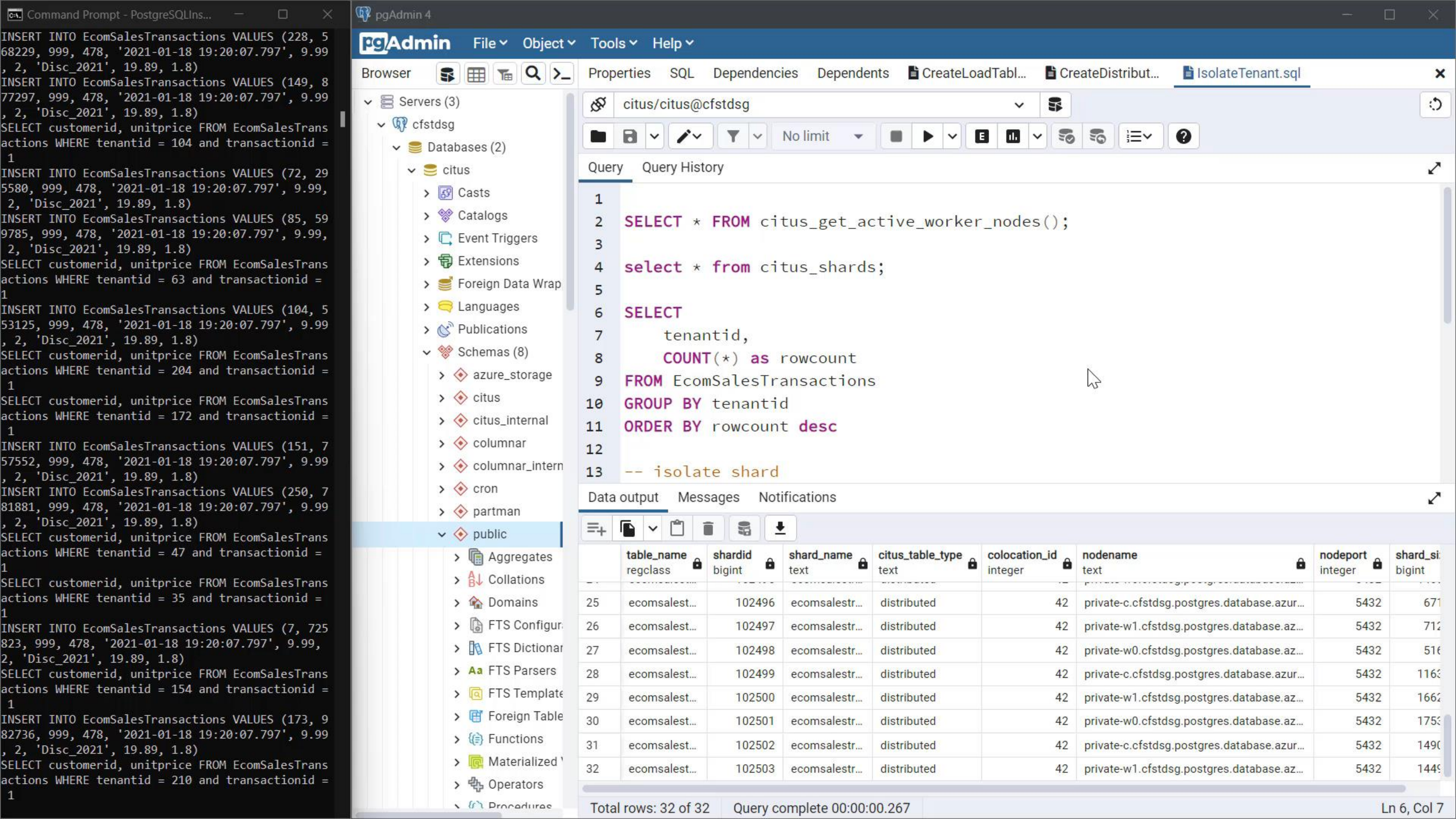
Enables online movement of tenants to new nodes

Large or busy tenants can be isolated to a dedicated node to maximize performance

No application code changes, or downtime required

Azure Cosmos DB for PostgreSQL Cluster





```
INSERT INTO EcomSalesTransactions VALUES (228, 5
68229, 999, 478, '2021-01-18 19:20:07.797', 9.99
, 2, 'Disc_2021', 19.89, 1.8)
INSERT INTO EcomSalesTransactions VALUES (149, 8
77297, 999, 478, '2021-01-18 19:20:07.797', 9.99
, 2, 'Disc_2021', 19.89, 1.8)
SELECT customerid, unitprice FROM EcomSalesTrans
actions WHERE tenantid = 104 and transactionid =
 1
INSERT INTO EcomSalesTransactions VALUES (72, 29
5580, 999, 478, '2021-01-18 19:20:07.797', 9.99,
 2, 'Disc_2021', 19.89, 1.8)
INSERT INTO EcomSalesTransactions VALUES (85, 59
9785, 999, 478, '2021-01-18 19:20:07.797', 9.99,
 2, 'Disc_2021', 19.89, 1.8)
SELECT customerid, unitprice FROM EcomSalesTrans
actions WHERE tenantid = 63 and transactionid =
1
INSERT INTO EcomSalesTransactions VALUES (104, 5
53125, 999, 478, '2021-01-18 19:20:07.797', 9.99
, 2, 'Disc_2021', 19.89, 1.8)
SELECT customerid, unitprice FROM EcomSalesTrans
actions WHERE tenantid = 204 and transactionid =
 1
SELECT customerid, unitprice FROM EcomSalesTrans
actions WHERE tenantid = 172 and transactionid =
 1
INSERT INTO EcomSalesTransactions VALUES (151, 7
57552, 999, 478, '2021-01-18 19:20:07.797', 9.99
, 2, 'Disc_2021', 19.89, 1.8)
INSERT INTO EcomSalesTransactions VALUES (250, 7
81881, 999, 478, '2021-01-18 19:20:07.797', 9.99
, 2, 'Disc_2021', 19.89, 1.8)
SELECT customerid, unitprice FROM EcomSalesTrans
actions WHERE tenantid = 47 and transactionid =
1
SELECT customerid, unitprice FROM EcomSalesTrans
actions WHERE tenantid = 35 and transactionid =
1
INSERT INTO EcomSalesTransactions VALUES (7, 725
823, 999, 478, '2021-01-18 19:20:07.797', 9.99,
2, 'Disc_2021', 19.89, 1.8)
SELECT customerid, unitprice FROM EcomSalesTrans
actions WHERE tenantid = 154 and transactionid =
 1
INSERT INTO EcomSalesTransactions VALUES (173, 9
82736, 999, 478, '2021-01-18 19:20:07.797', 9.99
, 2, 'Disc_2021', 19.89, 1.8)
SELECT customerid, unitprice FROM EcomSalesTrans
actions WHERE tenantid = 210 and transactionid =
 1
```

```sql

SELECT * FROM citus_get_active_worker_nodes();

select * from citus_shards;

SELECT
    tenantid,
    COUNT(*) as rowcount
FROM EcomSalesTransactions
GROUP BY tenantid
ORDER BY rowcount desc

-- isolate shard
```

	table_name regclass	shardid bigint	shard_name text	citus_table_type text	colocation_id integer	nodename text	nodeport integer	shard_si bigint
25	ecomsalest...	102496	ecomsalestr...	distributed	42	private-c.cfstdsg.postgres.database.azur...	5432	671
26	ecomsalest...	102497	ecomsalestr...	distributed	42	private-w1.cfstdsg.postgres.database.az...	5432	712
27	ecomsalest...	102498	ecomsalestr...	distributed	42	private-w0.cfstdsg.postgres.database.az...	5432	516
28	ecomsalest...	102499	ecomsalestr...	distributed	42	private-c.cfstdsg.postgres.database.azur...	5432	1163
29	ecomsalest...	102500	ecomsalestr...	distributed	42	private-w1.cfstdsg.postgres.database.az...	5432	1662
30	ecomsalest...	102501	ecomsalestr...	distributed	42	private-w0.cfstdsg.postgres.database.az...	5432	1753
31	ecomsalest...	102502	ecomsalestr...	distributed	42	private-c.cfstdsg.postgres.database.azur...	5432	1490
32	ecomsalest...	102503	ecomsalestr...	distributed	42	private-w1.cfstdsg.postgres.database.az...	5432	1449

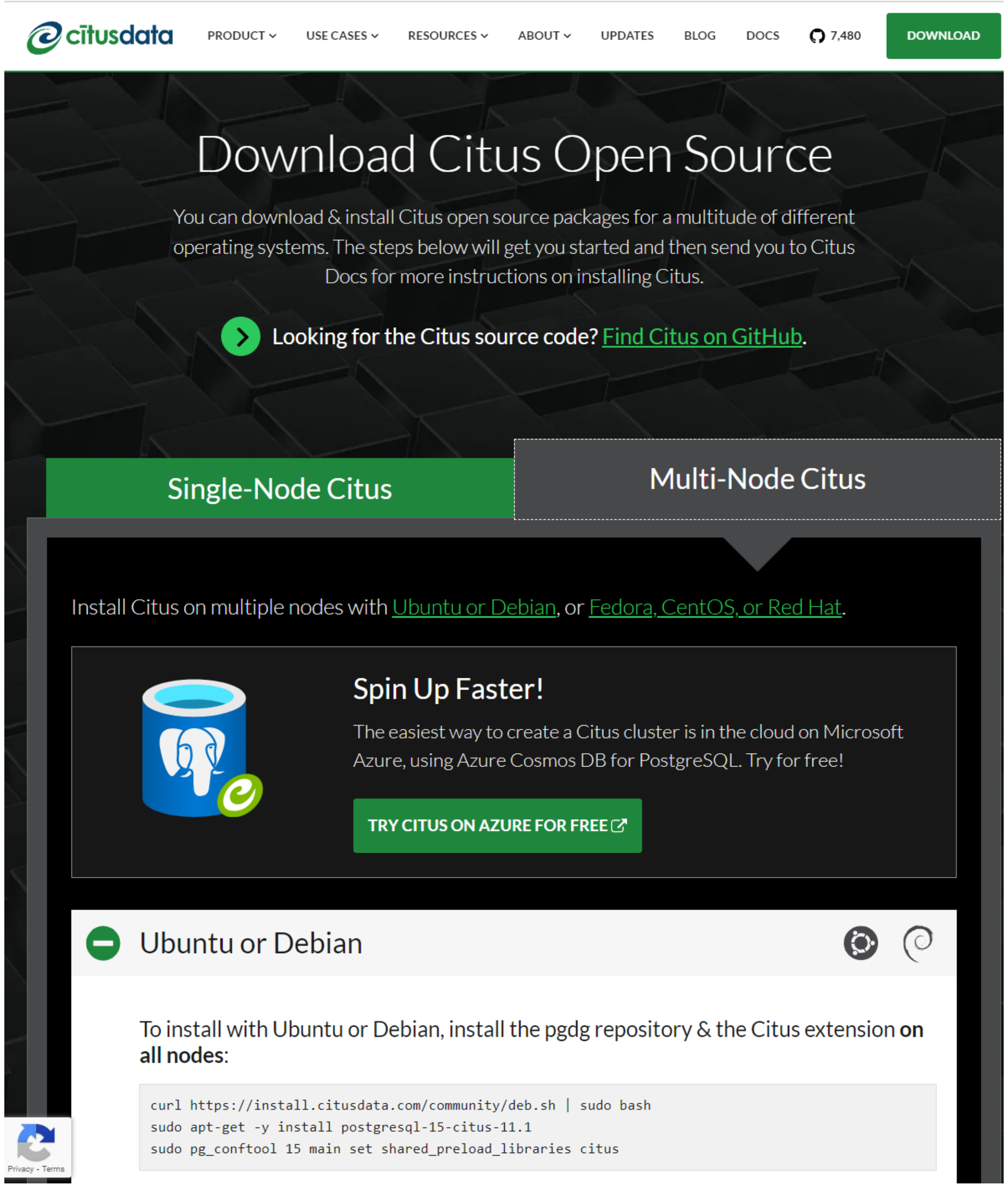
Total rows: 32 of 32 Query complete 00:00:00.267 Ln 6, Col 7

Getting started

Download Citus today

No Azure Subscription Required

No Credit Card Required



Download Citus Open Source

You can download & install Citus open source packages for a multitude of different operating systems. The steps below will get you started and then send you to Citus Docs for more instructions on installing Citus.

Looking for the Citus source code? Find Citus on GitHub.

Install Citus on multiple nodes with Ubuntu or Debian, or Fedora, CentOS, or Red Hat.

Spin Up Faster!

The easiest way to create a Citus cluster is in the cloud on Microsoft Azure, using Azure Cosmos DB for PostgreSQL. Try for free!

Ubuntu or Debian

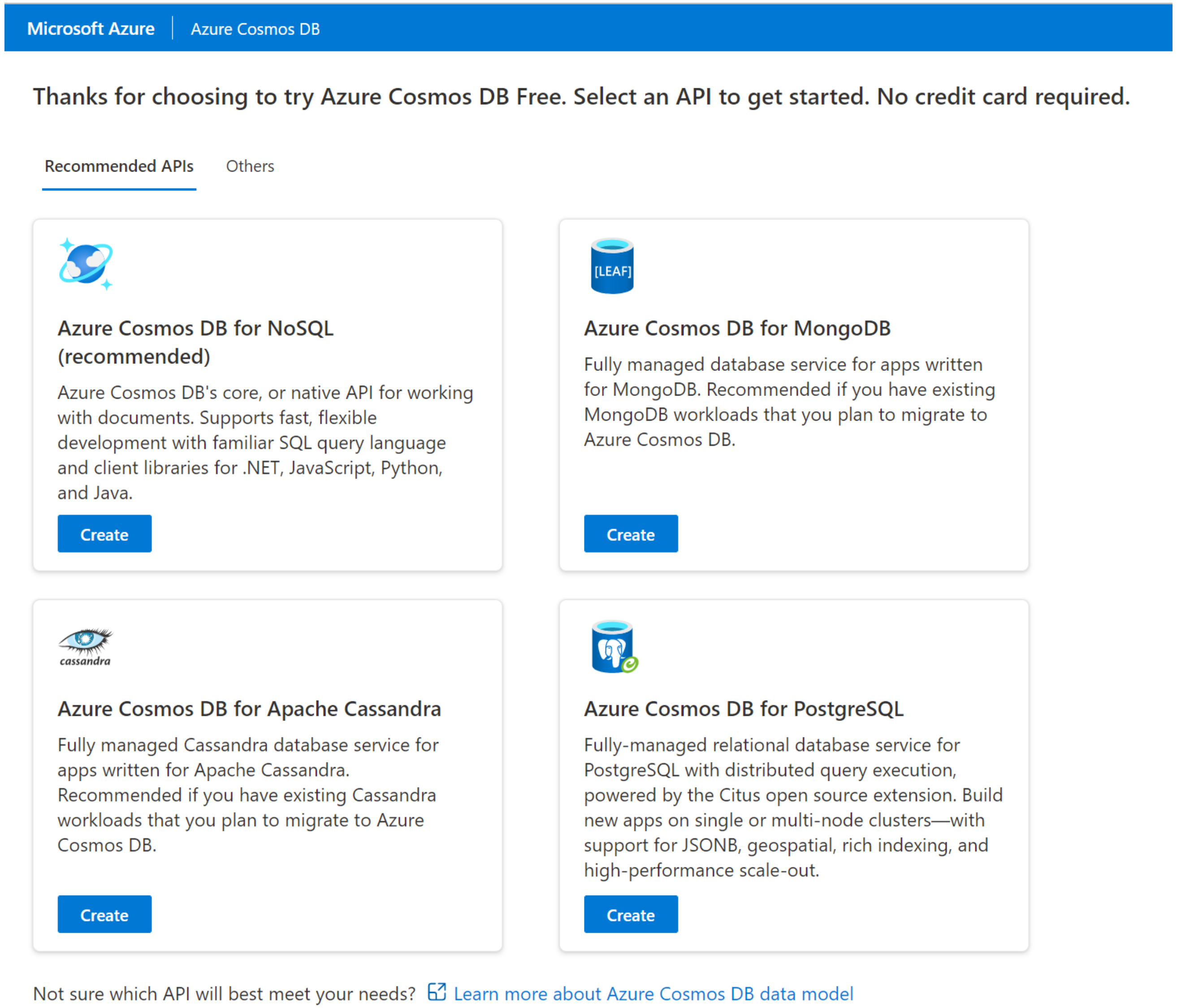
To install with Ubuntu or Debian, install the pgdg repository & the Citus extension **on all nodes**:

```
curl https://install.citusdata.com/community/deb.sh | sudo bash
sudo apt-get -y install postgresql-15-citus-11.1
sudo pg_conftool 15 main set shared_preload_libraries citus
```

Try Azure Cosmos DB for PostgreSQL free for 30 days

No Azure Subscription Required

No Credit Card Required



Thanks for choosing to try Azure Cosmos DB Free. Select an API to get started. No credit card required.

Azure Cosmos DB for NoSQL (recommended)

Azure Cosmos DB's core, or native API for working with documents. Supports fast, flexible development with familiar SQL query language and client libraries for .NET, JavaScript, Python, and Java.

Azure Cosmos DB for MongoDB

Fully managed database service for apps written for MongoDB. Recommended if you have existing MongoDB workloads that you plan to migrate to Azure Cosmos DB.

Azure Cosmos DB for Apache Cassandra

Fully managed Cassandra database service for apps written for Apache Cassandra. Recommended if you have existing Cassandra workloads that you plan to migrate to Azure Cosmos DB.

Azure Cosmos DB for PostgreSQL

Fully-managed relational database service for PostgreSQL with distributed query execution, powered by the Citus open source extension. Build new apps on single or multi-node clusters—with support for JSONB, geospatial, rich indexing, and high-performance scale-out.

Not sure which API will best meet your needs? Learn more about Azure Cosmos DB data model

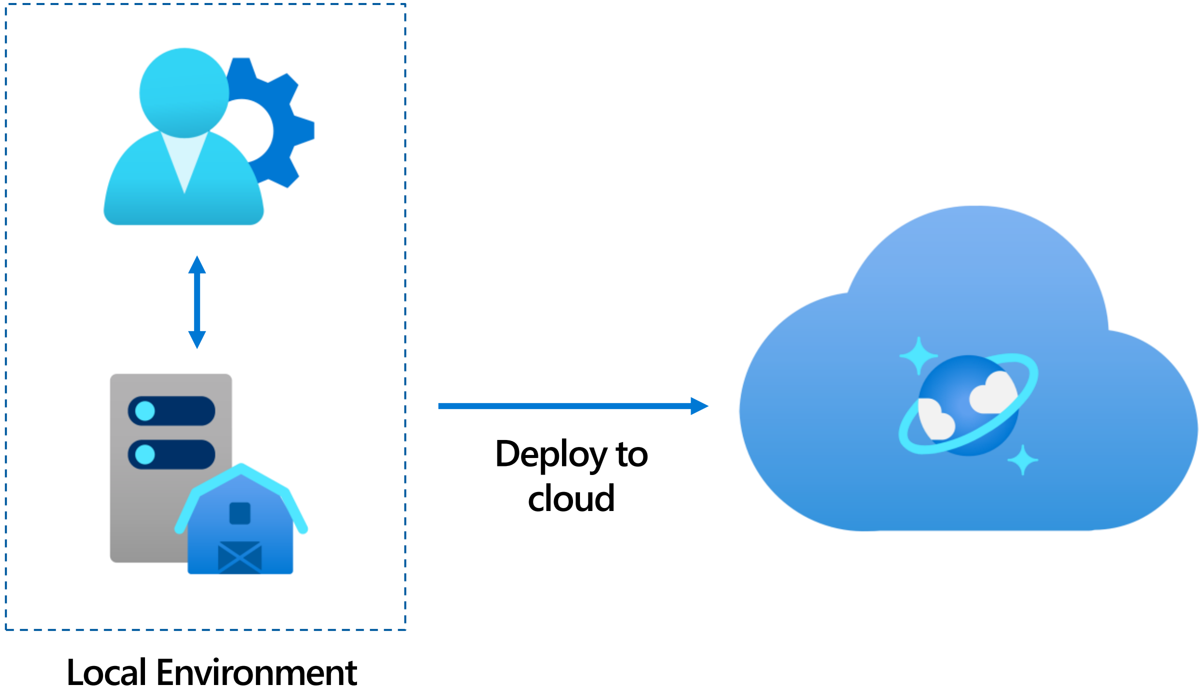
Managed Service

Develop locally – Deploy to cloud

Open-source PostgreSQL and Citus extension for distributed queries are free downloads

Empowers developers to develop and test locally, then deploy to the cloud

Zero code changes for deployment

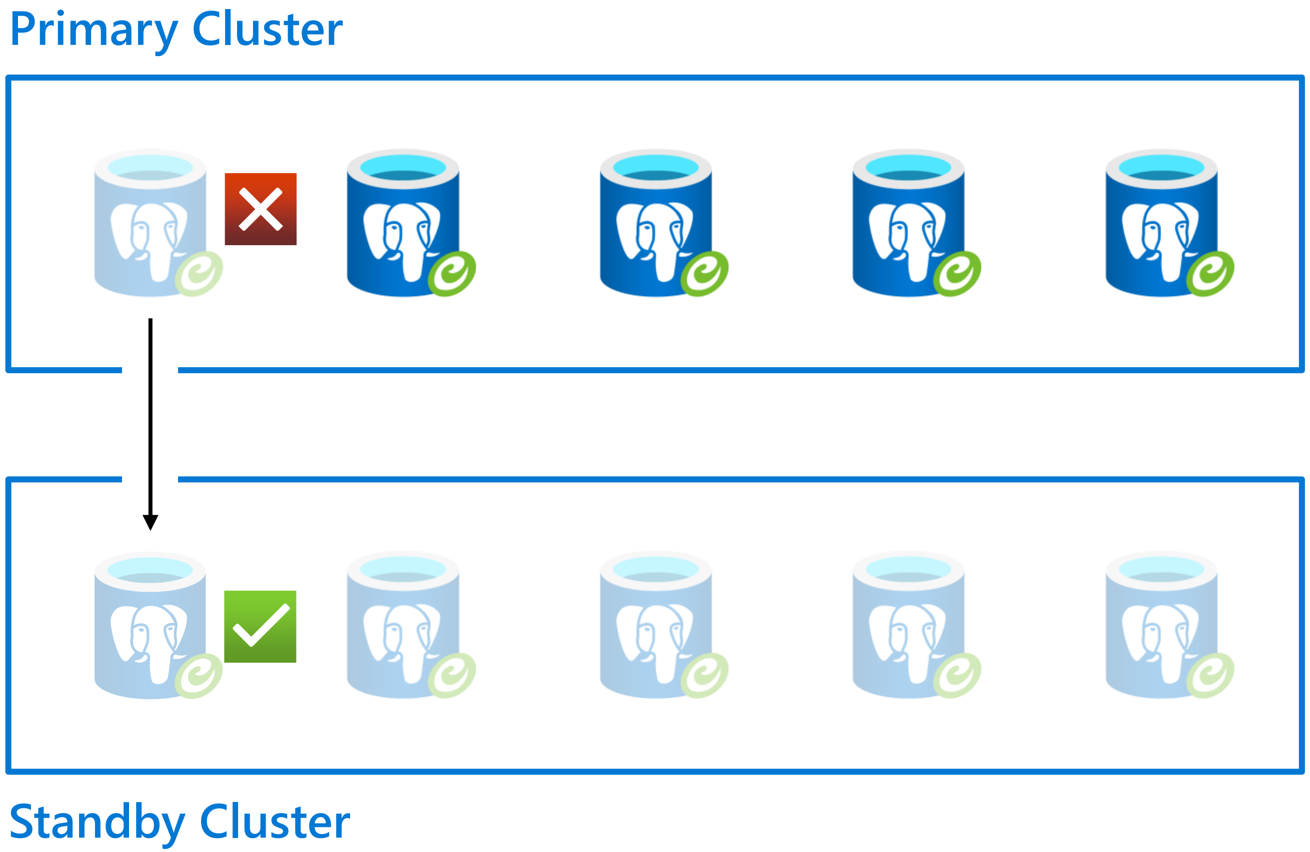


High Availability

Optional to enable

Synchronous replication provides zero data loss on failover

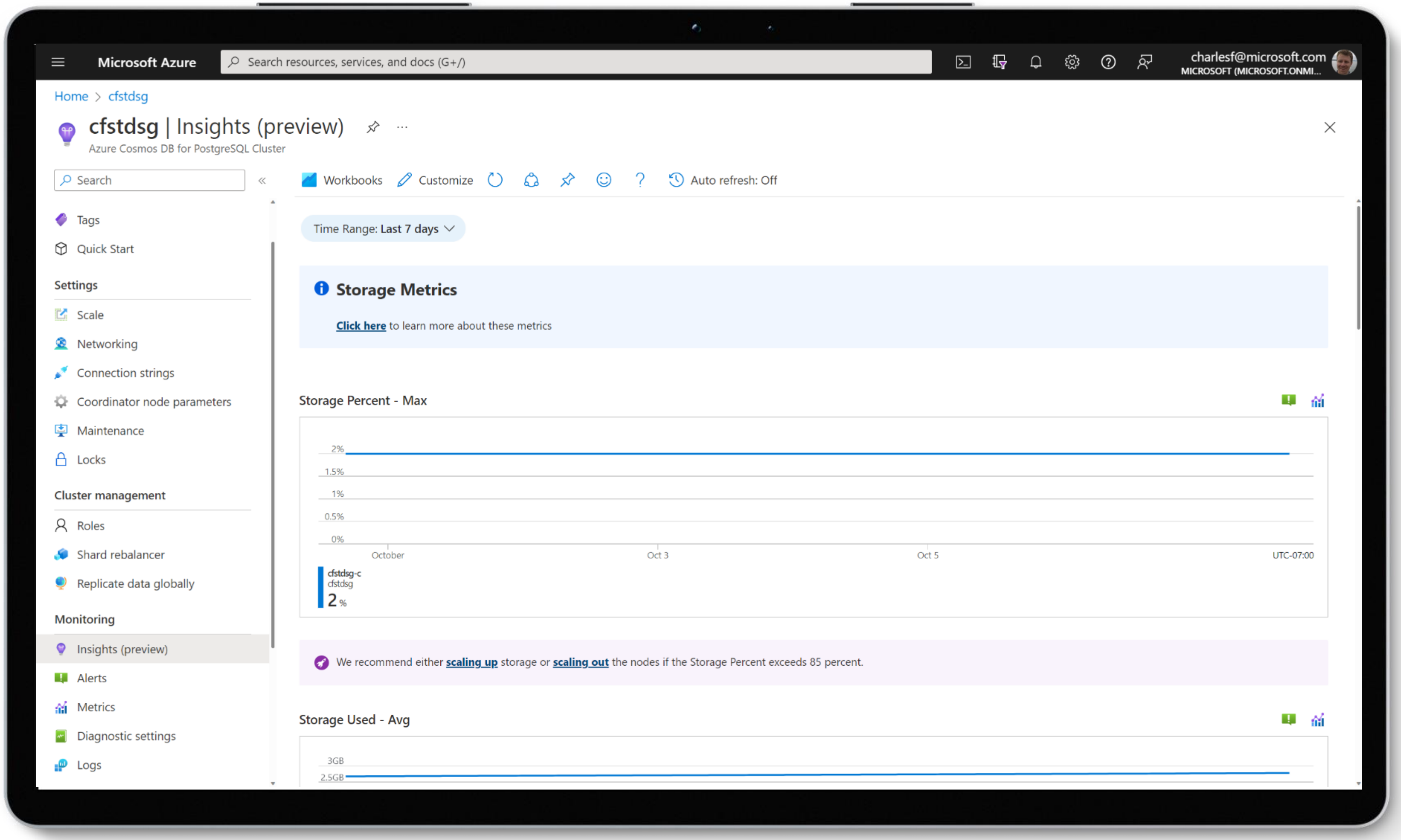
No application changes required



Built-in Monitoring

Pre-defined and configurable dashboards to monitor workload performance

Configure alerts based on business specific thresholds



Click here to learn more about these metrics

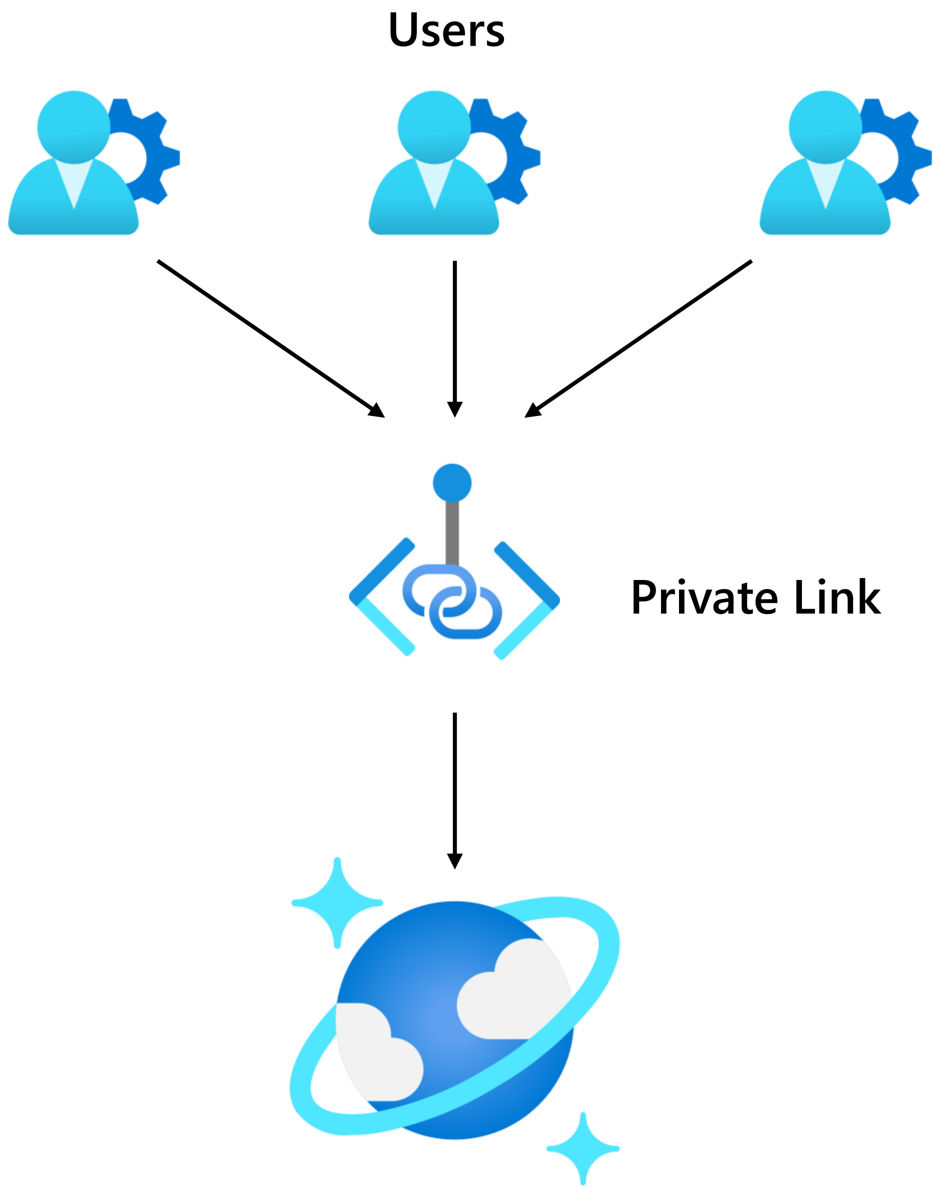
We recommend either scaling up storage or scaling out the nodes if the Storage Percent exceeds 85 percent.

Private Link

Provides private connectivity from a virtual network to Azure

Simplifies the network architecture

Secures connections between Azure endpoints by eliminating data exposure to the public internet



Azure Cosmos DB for PostgreSQL

General Availability – October 2022

Get started for free today as aka.ms/trycosmosdb

Microsoft