



Flavors of Postgres

An examination of the sociotechnical and community impacts of Postgres-like systems

Celeste Horgan, Sr. OSS Advocate @ Snowflake
October 2025, PGCon EU

This is v3 of this talk.

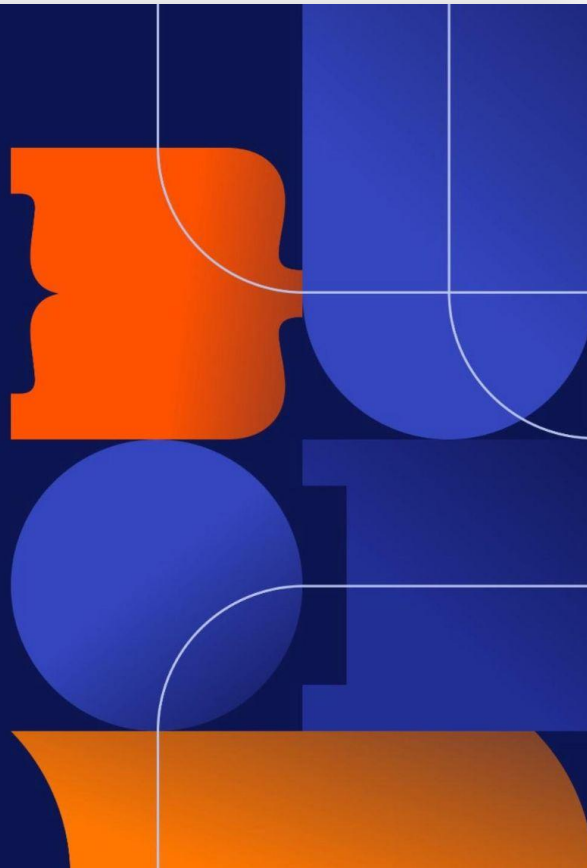
And it matters

v1: for the customer, by the company



Deep Dive: Flavors of PostgreSQL®

Celeste Horgan, Dev Educator @ Aiven



v2: The loose cannon

Celeste Horgan – Flavors of PostgreSQL® and you: how to choose a Postgres #bbuzz

Plain Schwarz 3.57K subscribers

32 views 4 months ago

More: <https://2025.berlinbuzzwords.de/session/>

Speaker: Celeste Horgan ...more

I no longer have a company-backed horse in this race

Things I wouldn't say if I still had a job will be in green

Celeste Horgan – Flavors of PostgreSQL

>BLN
BZZ
WORDS

From the series

From Plain Schwarz

Ved Prakash – Siphon : Modern Data Stack with SF-CH & ...
Plain Schwarz
128 views · 4 months ago

Isaac Chung – Reproducibility in Embedding Benchmarks #bbuzz
Plain Schwarz
146 views · 4 months ago

From SaaS to Data Company: Building Moats, Expansion, an...
SaaSiest
8 views · 13 days ago

How to Setup Windows PROPERLY
Linux Tech Tips
1.9M views · 4 months ago

event.15 MT :: Marilia Moita Humanizing Tech through ...
matosinhos tech
18 views · 12 days ago

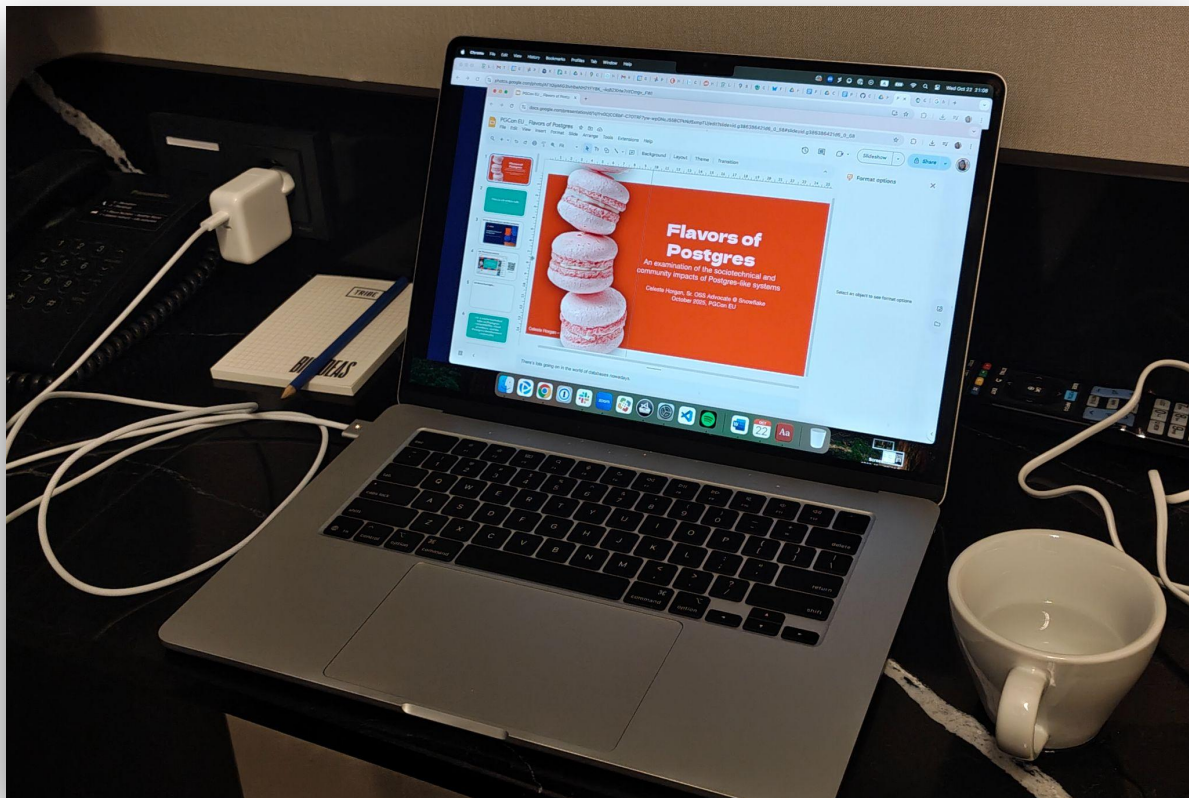
How to Identify Quality in Clothing (A Rant)
Bernadette Banner
1.7M views · 8 months ago

God-Tier Developer Roadmap
Fireship
7.8M views · 3 years ago



Watch it here

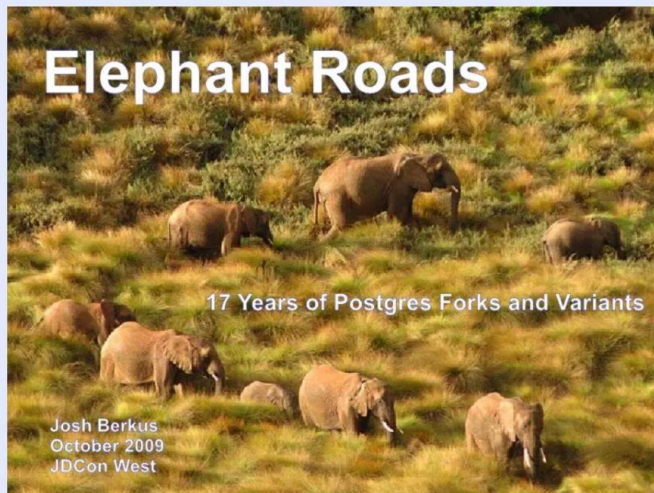
v3: About last night...



But is it really a v3?

A shoutout to 2009

<https://www.slideshare.net/slideshow/elephant-roads-a-tour-of-postgres-forks/5376286#19>



Flavors of Postgres v3: **a socio-technical take on Postgres** **compatibility, corporate open source,** **and what that means for the Postgres** **project community**

Wish me luck, lol

v3 agenda

v3 of this talk – 

A survey of closed-source database services that claim
Postgres compatibility: but are they Postgres-y? (the
technical bit)

The elephant-shaped clouds in the room (the socio bit)

Community compatibility conversations (the community bit)

Your host

Sr. OSS Advocate & OSPO @ Snowflake

Open source true believer

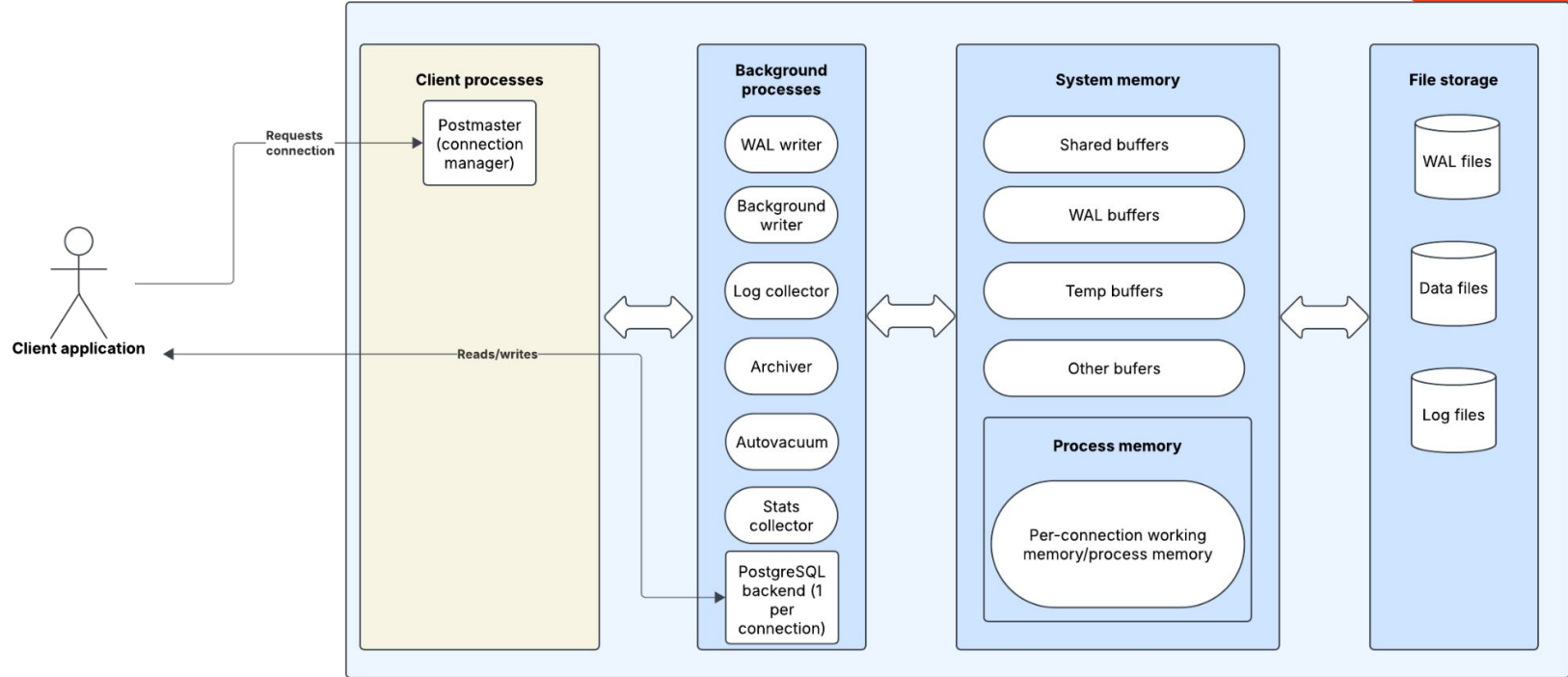
Previous experience includes Aiven, Stripe, The Linux Foundation, commercetools & more



What makes Postgres Postgres-y?

- **ACID adherent and transactional:** Database transactions are **A**tomic, **C**onsistent, **I**solated and **D**urable
- **Typed:** Data in the database must have a type. You can define your own data type if needed.
- **SQL standardization:** Postgres adheres closely to the SQL standard
- **Relational:** Data in the database must have and describe its relationship to other data in the database.
- **Lightweight:** Due to extension framework, the core code is relatively lightweight
- **Extensible and forkable:** The database has built-in support for user-created packages (extensions), and the entire project can be forked for any purpose.

How Postgres works, generally



What do we mean by ‘flavors’?

- **Forked from PostgreSQL (Redistributions)**
 - Either under open source or closed/BSL licensing
- **Extensions**
 - i.e. PostGIS, Timescale and pgvector
- **Patches or add-ons**
 - Additional code or patched functionality that is compatible with PostgreSQL project code
- **Architecture compatibility projects**
 - I.e. Patroni or CloudNativePG

Flavors of the week



AlloyDB Omni



Amazon Aurora



Neon

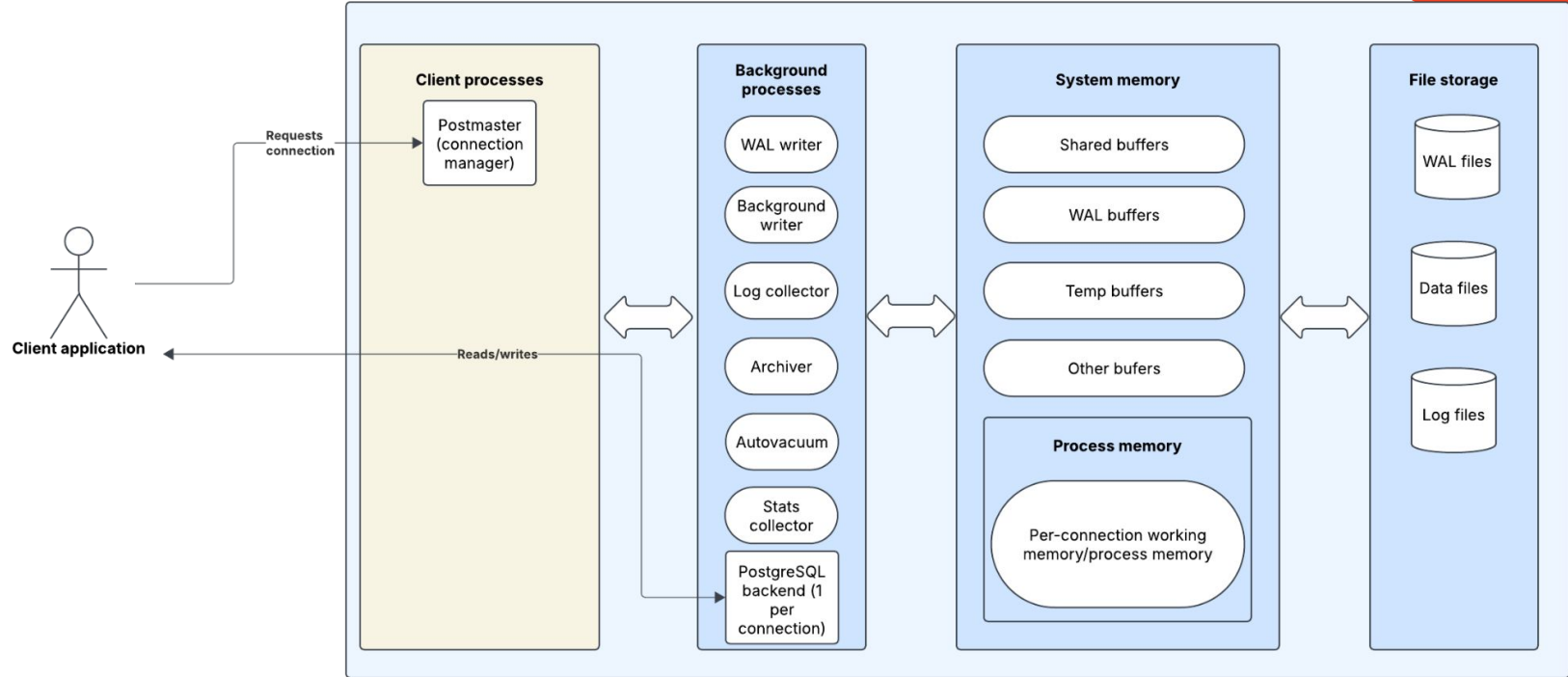
AlloyDB Omni



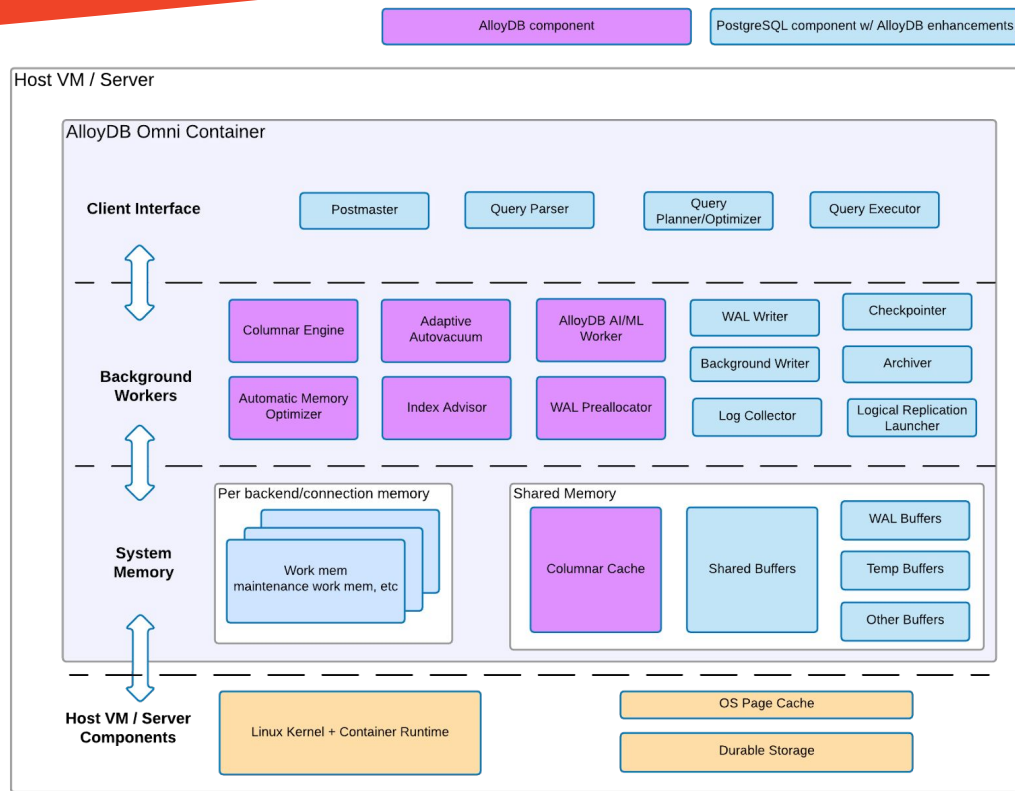
AlloyDB Omni in a nutshell

- **What it is:** “AlloyDB is a fully-managed, PostgreSQL-compatible database for demanding transactional workloads. It provides enterprise-grade performance and availability while maintaining 100% compatibility with open-source PostgreSQL.”
- **PostgreSQL compatibility:** A PostgreSQL fork with additional functionality in the shared processes and buffering layers, and a rewrite of some existing PostgreSQL processes
- **Licence:** Proprietary
- **AlloyDB Omni availability:**
 - **Direct from cloud providers:** AlloyDB on Google Cloud only
 - **Multicloud via service providers:** Like my former employer, Aiven!
 - **Via Docker:** as a container image.

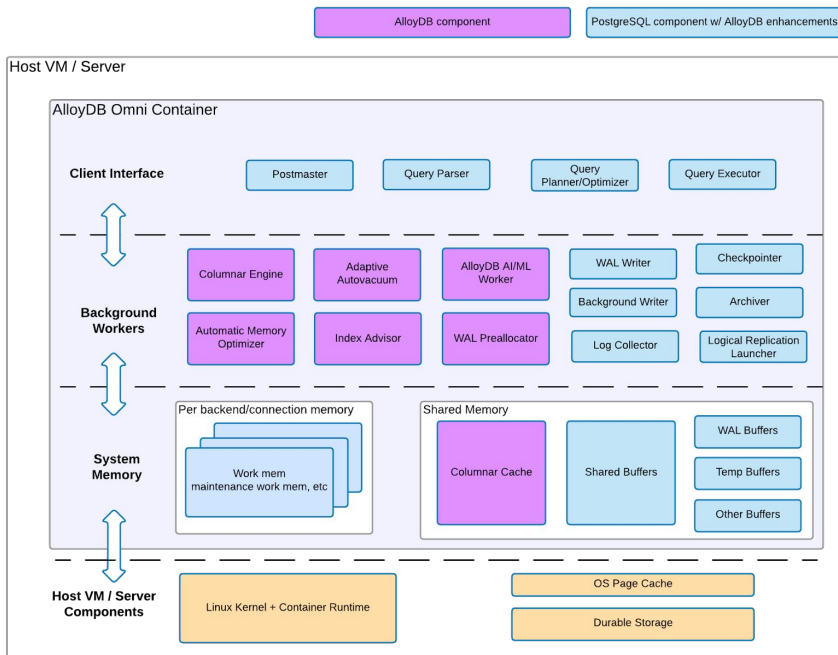
Revisiting this...



AlloyDB Omni architecture



AlloyDB Omni architecture*



is this all that's actually going on though? This code is completely un-introspectable, and the documentation is very light on the ground, and big purple boxes don't tell you much about how these things actually work?

Vector support

- Main support is through **vector**, a proprietary version of **pgvector**
- [Query remote models](#) and use to Vertex AI and the [Vertex AI model garden](#)
 - AI/ML models are remotely hosted by Google and you can query them using an API -> Provides access to many specialized models without the overhead of maintaining the infra yourself
- One of AlloyDB Omni's greatest strengths is its ability to work with generative AI/ML workloads, and to leverage Google's expertise in this space
 - You can (of course) use other AI/ML models provided by HuggingFace etc. however, as under the hood it's mostly PostgreSQL
- Do all of this **from SQL** - no need to do these operations in a runtime lang

Columnar engine

- Enables a second key use case for AlloyDB Omni: **analytical queries and complex SQL actions**

Operation	Query patterns
Table scan	Selective filters, such as WHERE clauses. A small number of columns from a larger table or materialized view. Expressions such as LIKE , SUBSTR , or TRIM .
Aggregation functions	Only expressions such as SUM , MIN , MAX , AVG , and COUNT . At the beginning of the query of a columnar scan. Ungrouped or grouped by columns.
ORDER-BY	Only if the operator is at the beginning of the query of a columnar scan.
SORT	Only if the operator is at the beginning of the query of a columnar scan and sorts only on the base columns of the table or the materialized view.
LIMIT	Only if the operator is at the beginning of the query of a columnar scan and is before any SORT or GROUP BY operators.
INNER HASH JOIN	Only if the keys used are columns and no join qualifiers are used.
Selective joins	Only if the joins are at the beginning of the query of a columnar scan.

- But** AlloyDB isn't a dedicated columnar database - it is fundamentally transactional, being based on PostgreSQL!

AlloyDB Omni: is it Postgres-y?

- ACID adherent and transactional ✓
 - Typed: ✓
 - SQL standardization: ✓
 - Relational: ✓
 - Lightweight: ?
 - Extensible: ✓
 - Forkable: ✗
-
- Conclusion: More Postgres-shaped than most

Why you'd want to use AlloyDB Omni

- **You're already on Google Cloud:** most of the benefits you'll see, particularly the AI related ones, need a Google Cloud account
- **You're noncommittal about AI/ML:** and don't want to invest in a dedicated vector store
- **You're columnar curious:** The columnar engine is a great way to explore whether this data storage is right for you!
- **You value the continued portability of your data:** AlloyDB Omni is more Postgres-shaped than many others

.. & why not

- **The documentation sucks:** and reads more like marketing copy than documentation
- **The Docker container is a black box:** And the docs only give us a very high level architecture diagram
- **No edge compute story:** If you need high performance reads globally, probably not the service for you

AWS Aurora & Databricks Neon

AWS Aurora in a nutshell

- **What it is:** Aurora is Amazon's distributed relational database service. Offers Aurora for PostgreSQL as one of many SQL offerings.
- **PostgreSQL compatibility:** Amazon promises [end result compatibility](#) with PostgreSQL: the same query should return the same result in both Amazon Aurora and an equivalent PostgreSQL database. Amazon has no language around Postgres version compatibility.
- **Licence:** Proprietary
- **Availability:** only AWS at this time

Differentiating Amazon RDS PostgreSQL, Amazon Aurora PostgreSQL and Amazon Aurora PostgreSQL DSQ, very briefly because it's very complicated and confusing

And an article that does a far better job of explaining it than I can in ~2 slides:
<https://dev.to/aws-heroes/amazon-aurora-dsdl-which-postgresql-service-should-i-use-on-aws--1598>

Amazon RDS PostgreSQL

“A Postgres in the Cloud”

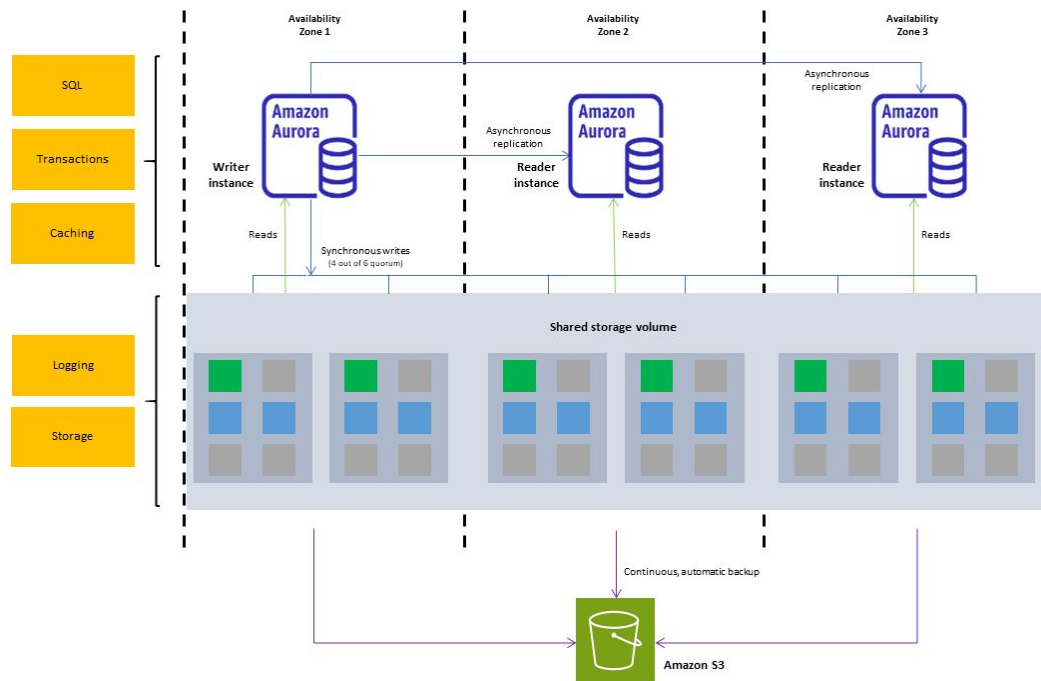
Amazon Aurora PostgreSQL

“An Aurora (serverless) database in the Cloud that can act as a drop-in replacement to Postgres”

Amazon Aurora PostgreSQL DSQ

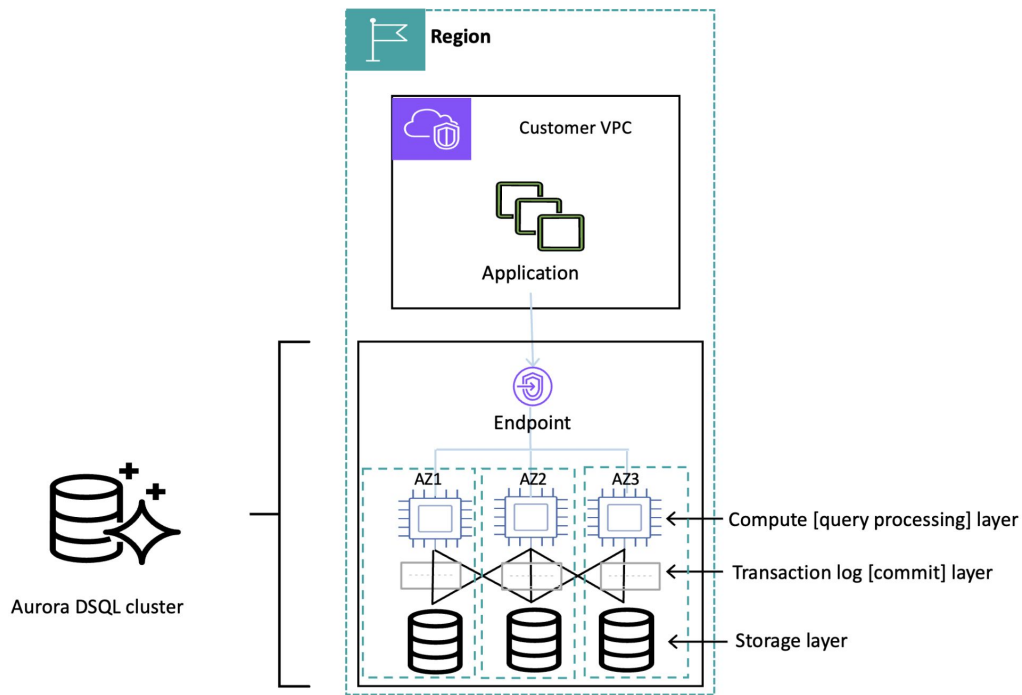
“An Aurora (serverless) distributed in the Cloud with ‘Postgres compatibility’”

AWS Aurora* architecturally



as usual there's a caveat to this, which is that this is a general architecture for an AWS Aurora *cluster* rather than a specific implementation for Postgres, which, as previously mentioned... there are at least three. However, we can glean from this *cluster* that things look Very Different under the hood from a Postgres perspective, because how else would you accomplish this?

AWS Aurora DSQL architecturally



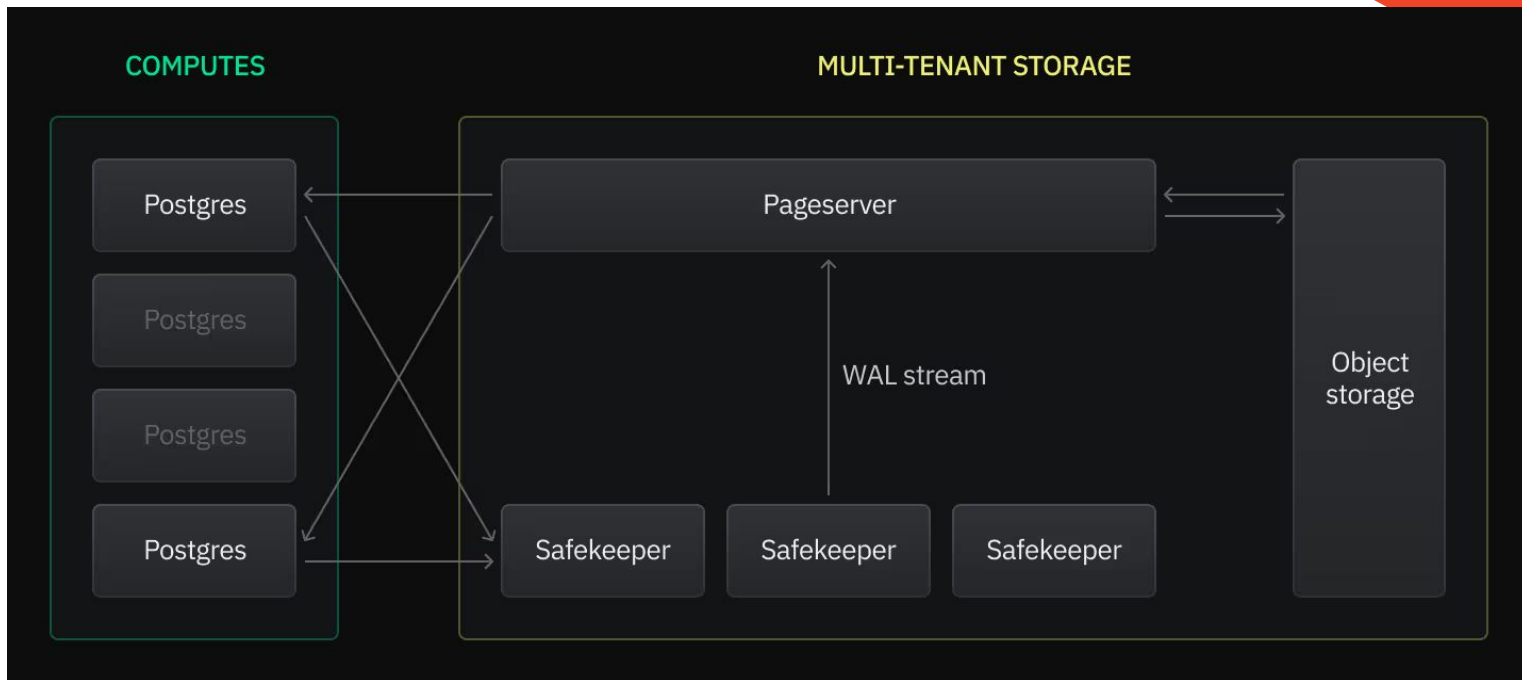
Neon in a nutshell

- **What it is:** Serverless, distributed PostgreSQL
 - Users interact with a Neon PostgreSQL server using psql and familiar tooling, but the back end is radically different
 - Architecturally, Neon is heavily based on the [Amazon Aurora whitepaper](#)
- **PostgreSQL compatibility:** Neon supports the five latest PostgreSQL versions, per the PostgreSQL projet maintenance policy
- **Licence:** Apache 2.0
- **Availability:** Only direct through Neon at this time

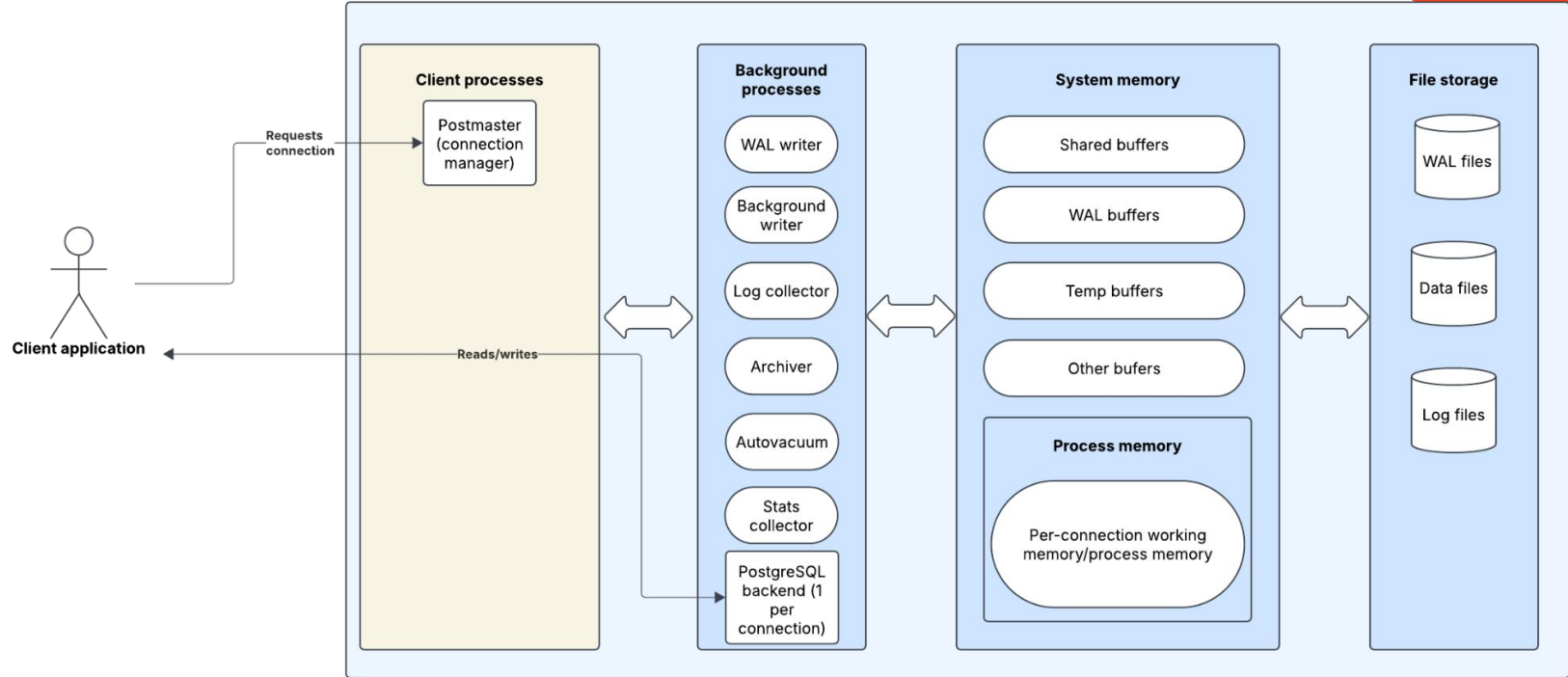


with Neon's very recent (May 2025) Acquisition by Databricks, both the licensing and availability might change in the near future.

Neon architecturally (look similar?)



Revisiting this again..



... But are they Postgres-y?

- ACID adherent and transactional ✓
 - Typed: ✓
 - SQL standardization: ~ with caveats
 - Relational: ✓
 - Lightweight: ✗
 - Extensible: ~ limited
 - Forkable: ✗
-
- Conclusion: not really Postgres-y

The big idea: decoupling storage from compute

- The key issue most traditional databases have in cloud computing environments is the **tight coupling of storage and compute** (or processes)
- Cloud environments are based on the idea that **at some point, some part of a system will fail**. Therefore, in separating the system into as many individual nodes as possible, the failure can be isolated, and the system can create replicants of those nodes for failover purposes
- Per the whitepaper: “**Instance lifetime does not correlate well with storage lifetime**. Instances fail. Customers shut them down. They resize them up and down based on load. **For these reasons, it helps to decouple the storage tier from the compute tier.**”
- Both Amazon Aurora and Neon might be PostgreSQL-compatible on some level, but to satisfy the above requirements they are not PostgreSQL from an architectural perspective.

The elephant in the room

A joke I'm sure has never been made in this
community before

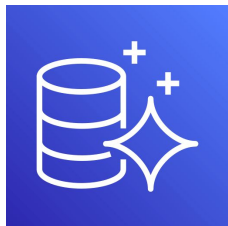




Created by Google



Acquired by Snowflake



Created by Amazon



Acquired by Microsoft



Acquired by Databricks

**Someone's first experience with Postgres
might not *be* Postgres**

A silver fork is positioned diagonally from the bottom left towards the center of the frame. The background is a solid, vibrant red. The text "Postgres is forkable by design" is centered in the upper half of the image in a white, bold, sans-serif font.

**Postgres is forkable
by design**

Core Team

For more information on the purpose of the core team, please see the [FAQ](#).

Contributor
Peter Eisentraut (peter.eisentraut at enterprisedb.com) EDB Dresden, Germany
Andres Freund (andres at anarazel.de) Microsoft San Francisco, USA
Magnus Hagander (magnus at hagander.net) Redpill Linpro Stockholm, Sweden
Jonathan Katz (jonathan.katz at excoventures.com) Amazon Web Services New York, NY
Tom Lane (tgl at sss.pgh.pa.us) Snowflake Inc Pittsburgh, Pennsylvania, USA
Bruce Momjian (bruce at momjian.us) EDB Newtown Square, Pennsylvania, USA
Dave Page (dpage at pgadmin.org) pgEdge Oxfordshire, United Kingdom



CloudNativePG

Ecosystem expansion

 **microsoft / documentdb** Public

DocumentDB is the open-source engine powering vCore-based Azure Cosmos DB for MongoDB. It offers a native implementation of document-oriented NoSQL database, enabling seamless CRUD operations on BSON data types within a PostgreSQL framework.

Forking keeps the community strong

Service & support

Redpill Linpro



Innovation

Amazon Aurora: Design Considerations for High Availability and Scalability Throughput Cloud-Native Relational Database

Alexandre Verbitski, Anurag Gupta, Debanjan Saha, Murali B. Raman Mittal, Sailesh Krishnamurthy, Sandor Maurice, Teng...

Amazon Web Services

ABSTRACT

Amazon Aurora is a relational database service for OLTP workloads offered as part of Amazon Web Services (AWS). In this paper, we describe the architecture of Aurora and the design considerations leading to that architecture. We believe the central constraint in high throughput data processing has moved from compute and storage to the network. Aurora brings a novel architecture to the relational database to address this constraint, most notably by pushing redo processing to a multi-tenant scale-out storage service, purpose-built for Aurora. We describe how doing so not only reduces network traffic, but also allows for fast crash recovery, failovers to replicas without loss of data, and fault-tolerant, self-healing storage. We then describe how Aurora achieves consensus on durable state across numerous storage nodes using an efficient asynchronous scheme, avoiding

The IO bottleneck in this environment, and many disks in nodes are no longer network between the tier that performs packets per second traffic since a per node, disk or network. Although most operations, there are self-healing operations. These are a disk re A read-ahead can



**... but companies are trading on
“compatibility” at the project’s expense**



Community initiatives on compatibility



Schedule - Establishing the PostgreSQL standard: What's Postgres compatible?

postgres.eu/events/pgconfeu2025/schedule/session/7152-establishing-the-postgresql-standard-what-s-postgres-compatible/

Establishing the PostgreSQL standard: What's Postgres compatible?

Tuesday, October 21 at 09:00–12:30

Room: Lambda

Level: Intermediate

What does it mean to be "PostgreSQL compatible"? As PostgreSQL becomes "the new Linux" for the enterprise, this question becomes increasingly important. This half-day session invites Postgres developers, contributors, and community members to define a practical framework of criteria and tests for PostgreSQL compatibility.

About

As PostgreSQL's success grows, so do the challenges of brand dilution and feature fragmentation from "Postgres-compatible" derivatives. This session will focus on designing a test harness using existing regression tests and propose new tests covering core features, build behaviour, Application Binary Interface (ABI) stability, and backup/replication. The goal is to outline a concrete path towards meaningful compatibility certification.

This will be a focused, technical working session.

Agenda

- Introduction & Motivation: A brief overview of the current landscape and the goals of the session.
- Group Discussions: Participants will split into small groups, with discussion leads for each topic, to identify compatibility concerns and propose test approaches.
- Presentations & Roadmap: Each group will present their findings, followed by a collaborative session to outline a preliminary roadmap.

Interest areas

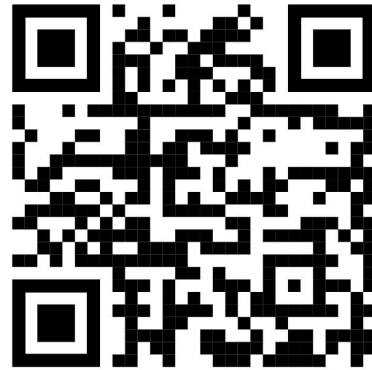
- SQL Features & Core Behaviour: What defines the essential PostgreSQL user experience?
- Test Coverage: Build tests, behavioural tests, and ABI stability.
- Backup & Replication: Ensuring compatibility with standard tools and protocols (WAL, PITR, streaming, logical)
- Identity: If a product is compatible but not the C (vanilla) implementation, is it still PostgreSQL?

We aim to close the session with a preliminary outline of criteria for meaningful compatibility testing and a roadmap for continued collaboration.

Community initiatives on compatibility



Read the notes



Join the telegram chat

Thank you!

<https://bsky.app/profile/celestehorgan.bsky.social>

<https://www.linkedin.com/in/celeste-horgan/>



Feedback?



Watch v2?



Why you'd want to use AWS Aurora or Neon

- **You're serving data globally and need the performance boost of edge compute:** *Really* evaluate your global needs before committing, though.
- **You're comfortable with the relative data lock-in:** due to the architecture choices driven by the edge compute model
- **You need high reliability but cannot sacrifice read speed**
- **You have a dedicated DevOps/SRE team who can self-serve**
Using AWS' excellent documentation

.. & why not

- **You want more hand-held support:** AWS is a big company, and you are a small fish in that sea. Neon might be a better bet for you in this regard
- **You have other means of storing data for portability purposes:** and the Aurora/Neon layer is just a serving layer

A silver fork is positioned diagonally from the bottom left towards the center of the frame. The background is a solid, vibrant red. The text is centered in the upper half of the image.

**To err is human, to fork
is Postgres**

The big idea: decoupling storage from compute

- The key issue most traditional databases have in cloud computing environments is the **tight coupling of storage and compute** (or processes)
- Cloud environments are based on the idea that **at some point, some part of a system will fail**. Therefore, in separating the system into as many individual nodes as possible, the failure can be isolated, and the system can create replicants of those nodes for failover purposes
- Per the whitepaper: “**Instance lifetime does not correlate well with storage lifetime**. Instances fail. Customers shut them down. They resize them up and down based on load. **For these reasons, it helps to decouple the storage tier from the compute tier.**”
- Both Amazon Aurora and Neon might be PostgreSQL-compatible on some level, but to satisfy the above requirements they are not PostgreSQL from an architectural perspective.

Postgres 17

Sorry folks, I got asked to give this talk yesterday at ~noon and I didn't have time to update this section

PG18 looks cool tho



New in PG17: Incremental backups

- **What versions:** PG 17
- **What it is:**
 - Introduces the WAL summarizer background process (turn on using the `summerizer_wal` config option) which lets you take incremental backups in addition to full backups.
 - First you take a `pg_basebackup`, then you can use `pg_basebackup -i=<Previous Backup>` to backup changes made since then as an incremental backup.
- **Why you should care:** This is a more efficient way to generate database backups if PITR isn't a hard requirement, and provides a bit more flexibility to the backup story in Postgres.

New in PG17: JSON tables

- **What versions:** PG 17
- **What it is:** Postgres implements the SQL/JSON data model, allowing for JSON transformations to Postgres tables and back again.
- **Why you should care:** JSON is the de-facto standard for sending and receiving data from most APIs and web applications: if data is being moved, at some point you can transform it safely into JSON. Native JSON support.
 - Native JSON support to the level that Postgres now provides means that you can directly ingest data from an API or other data source without having to write looping application code to do so. Once the data is in Postgres, you can then bulk apply transformations to (for example) reformat timestamps into Postgres' timestamp format.

The world beyond Postgres core



Patroni

- **What it is:** A (confusingly named) extension. Time series data storage and processing on PostgreSQL
 - Specifically implements **hypertables** which partition data by time. Querying a hypertable addresses only the partition which contains the timestamp, speeding up queries to time series data
 - Also provides improved compression and performance for operations on large amounts of data
- **PostgreSQL compatibility:** “Apache edition” is an open source extension usable with Aiven for PostgreSQL or any other PostgreSQL.
- **License:**
 - Apache edition: Apache 2.0
 - Community edition: business source licensed
- **Availability:** any PostgreSQL instance with permission to use the **CREATE EXTENSION** command

CloudNativePG

- **What it is:** A Kubernetes operator for a highly available Postgres cluster.
 - The killer feature of CloudNativePG is the ability to manage a Postgres cluster in a cloud native way *for anyone*
 - Some cloud providers and managed service providers have k8s operator functionality, but this gives you full control over the deployment.
 - Best suited for people with a dedicated SRE team heavily invested in k8s infrastructure and companies experienced with managing the full lifecycle of a Postgres database independently
- **PostgreSQL compatibility:** It's Postgres, baby!
- **License:** Apache 2.0
- **Availability:** Any Kubernetes cluster where you have full permissions

Timescale (Apache edition)

- **What it is:** A (confusingly named) extension. Time series data storage and processing on PostgreSQL
 - Specifically implements **hypertables** which partition data by time. Querying a hypertable addresses only the partition which contains the timestamp, speeding up queries to time series data
 - Also provides improved compression and performance for operations on large amounts of data
- **PostgreSQL compatibility:** “Apache edition” is an open source extension usable with Aiven for PostgreSQL or any other PostgreSQL.
- **License:**
 - Apache edition: Apache 2.0
 - Community edition: business source licensed
- **Availability:** any PostgreSQL instance with permission to use the **CREATE EXTENSION** command

Documentdb

- **What it is:** A Postgres extension that allows for document database models.
 - Often called a MongoDB clone, this is a bit incorrect: it provides document database models *on top of* Postgres, which is fundamentally different than MongoDB
 - Useful for those who might have a specific use case for document data stores, but who otherwise want or need a relational database
- **PostgreSQL compatibility:** An extension
- **License:** Apache 2.0
- **Availability:** any PostgreSQL instance with permission to use the `CREATE EXTENSION` command
- Special shoutout to the **FerretDB** team, who are implementing a more fully featured document data store based on documentdb.

Why ‘vanilla’ Postgres or an extension?

- **Maximum forward compatibility:** It's Postgres, no bells and whistles, and you know your data will be safe, portable and compatible with almost anything as a result
- **The power of the extensions ecosystem:** With far better compatibility guarantees than some of the closed source solutions
- **Full control:** Do what you want!
- **Community support:** service providers, architects & more, built up over years of community work

.. & why not

- **You don't have headcount/interest in managing a database:** And not everyone does!
- **You're trying to hit committed cloud spend with a major provider:** fair enough I guess
- **You're heavily invested into AI, high performance analytics, or edge computing use cases**



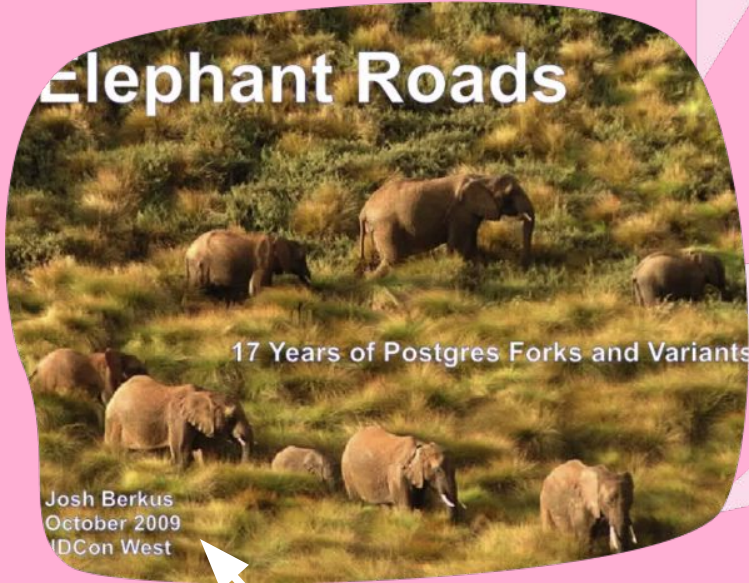
Closing time



We've been here before

A shoutout to Josh Berkus

<https://www.slideshare.net/slideshow/elephant-roads-a-tour-of-postgres-forks/5376286#19>



Core Team

For more information on the purpose of the core team, please see the [FAQ](#).

Contributor
Peter Eisentraut (peter.eisentraut at enterprisedb.com) EDB Dresden, Germany
Andres Freund (andres at anarazel.de) Microsoft San Francisco, USA
Magnus Hagander (magnus at hagander.net) Redpill Linpro Stockholm, Sweden
Jonathan Katz (jonathan.katz at excoventures.com) Amazon Web Services New York, NY
Tom Lane (tgl at sss.pgh.pa.us) Snowflake Inc Pittsburgh, Pennsylvania, USA
Bruce Momjian (bruce at momjian.us) EDB Newtown Square, Pennsylvania, USA
Dave Page (dpage at pgadmin.org) pgEdge Oxfordshire, United Kingdom



CloudNativePG

Ecosystem expansion

microsoft / documentdb Public

DocumentDB is the open-source engine powering vCore-based Azure Cosmos DB for MongoDB. It offers a native implementation of document-oriented NoSQL database, enabling seamless CRUD operations on BSON data types within a PostgreSQL framework.

Forking keeps the community strong

Service & support

Redpill Linpro



Innovation

Amazon Aurora: Design Considerations for High Availability and Scalability Throughput Cloud-Native Relational Database

Alexandre Verbitski, Anurag Gupta, Debanjan Saha, Murali B. Raman Mittal, Sailesh Krishnamurthy, Sandor Maurice, Tengren Wang

Amazon Web Services

ABSTRACT

Amazon Aurora is a relational database service for OLTP workloads offered as part of Amazon Web Services (AWS). In this paper, we describe the architecture of Aurora and the design considerations leading to that architecture. We believe the central constraint in high throughput data processing has moved from compute and storage to the network. Aurora brings a novel architecture to the relational database to address this constraint, most notably by pushing redo processing to a multi-tenant scale-out storage service, purpose-built for Aurora. We describe how doing so not only reduces network traffic, but also allows for fast crash recovery, failovers to replicas without loss of data, and fault-tolerant, self-healing storage. We then describe how Aurora achieves consensus on durable state across numerous storage nodes using an efficient asynchronous scheme, avoiding

The IO bottleneck in this environment, and many disks in nodes are no longer network between the tier that performs packets per second traffic since a per node, disk or network. Although most operations, there are self-healing operations. These are a disk repair. A repair thread can





My biggest personal opinion (the tea)



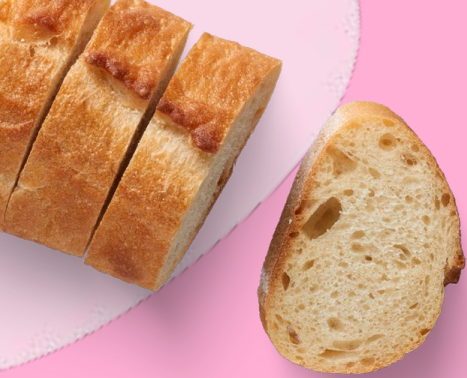
Thank you!

<https://bsky.app/profile/celestehorgan.bsky.social>



LinkedIn (let's chat
about freelance or
full time technical
writing/DevRel!)





Contents

Introduction

History of Baking

Bread & Pastry

Baking Techniques & Ingredients

Baking Tools

Conclusion



Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam porta sapien in lacinia blandit. Nam cursus lorem dolor, nec euismod elit bibendum eget. Phasellus ut odio convallis, suscipit turpis eu, semper ex.

Fusce massa nibh, cursus a elit feugiat, eleifend molestie ante. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam porta sapien in lacinia blandit. Nam cursus lorem dolor, nec euismod elit



History of Baking

You may add
historical details
about your topic.



You may add
historical details
about your topic.



1960

1980

2000

2020



You may add
historical details
about your topic.



You may add
historical details
about your topic.

Bread

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam porta sapien in lacinia blandit. Nam cursus lorem dolor, nec euismod elit bibendum eget. Phasellus ut odio convallis, suscipit turpis eu, semper ex. Fusce massa nibh, cursus a elit feugiat, eleifend molestie ante. Duis aute irure



Pastry

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam porta sapien in lacinia blandit. Nam cursus lorem dolor, nec euismod elit bibendum eget. Phasellus ut odio convallis, suscipit turpis eu, semper ex. Fusce massa nibh, cursus a elit feugiat, eleifend molestie ante. Duis aute irure

Types of Bread & Pastries

Name of Bread
or Pastries

Name of Bread
or Pastries

Name of Bread
or Pastries

Name of Bread
or Pastries

Name of Bread
or Pastries

Name of Bread
or Pastries

Name of Bread
or Pastries

Name of Bread
or Pastries

Name of Bread
or Pastries

Name of Bread
or Pastries

Name of Bread
or Pastries

Name of Bread
or Pastries

Baking
Technique

Baking
Technique

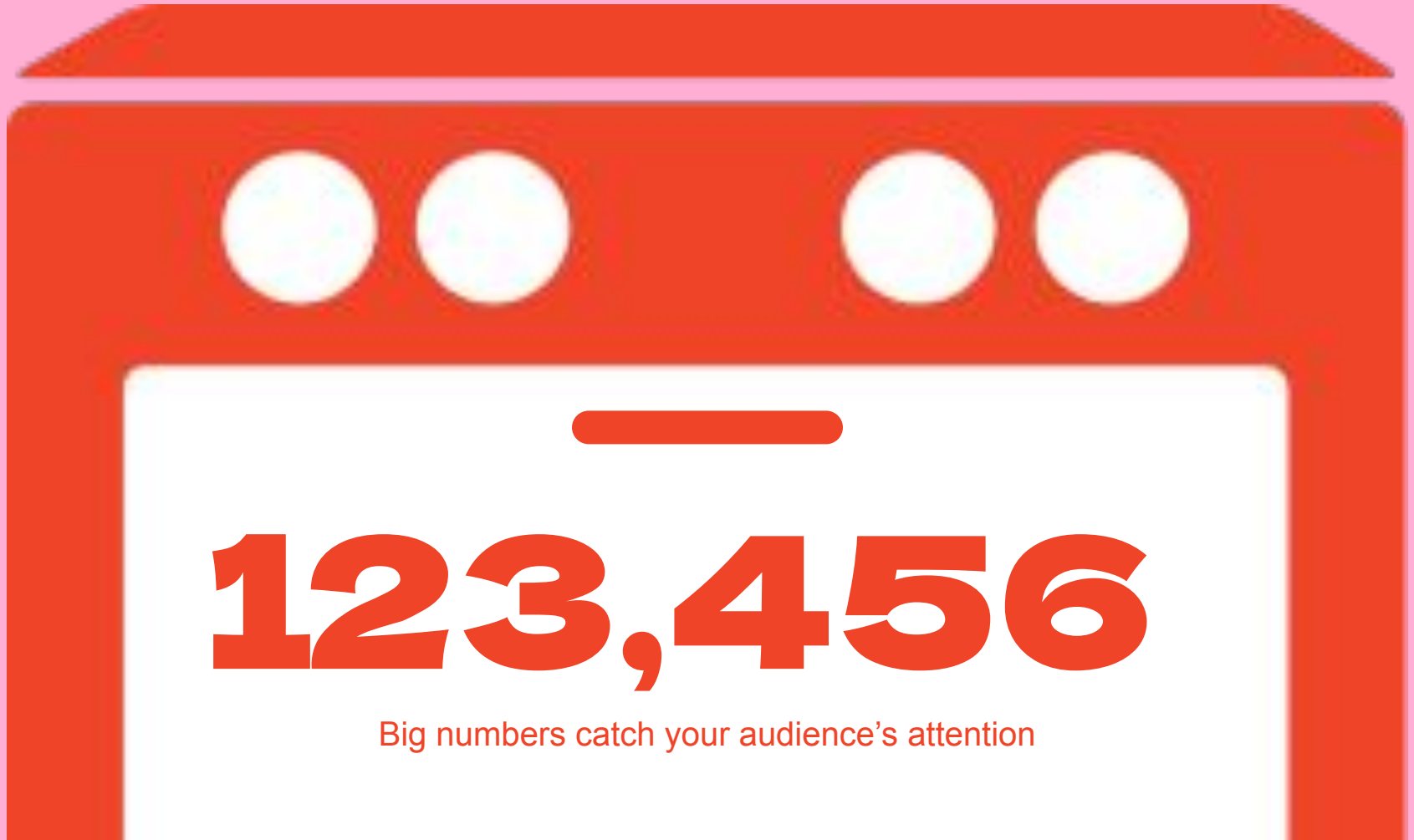
Baking
Technique

Baking
Technique

**“THIS IS A QUOTE WORDS
FULL OF
WISDOM.”**



Someone
Famous



123,456

Big numbers catch your audience's attention

Baking Techniques

BAKING TECHNIQUE

Lorem ipsum dolor sit amet.

BAKING TECHNIQUE

Lorem ipsum dolor sit amet.

BAKING TECHNIQUE

Lorem ipsum dolor sit amet.

BAKING TECHNIQUE

Lorem ipsum dolor sit amet.

BAKING TECHNIQUE

Lorem ipsum dolor sit amet.

BAKING TECHNIQUE

Lorem ipsum dolor sit amet.

BAKING TECHNIQUE

Lorem ipsum dolor sit amet.

BAKING TECHNIQUE

Lorem ipsum dolor sit amet.

BAKING TECHNIQUE

Lorem ipsum dolor sit amet.



Baking Ingredients





Baking Tools





Conclusion

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam porta sapien in lacinia blandit. Nam cursus lorem dolor, nec euismod elit bibendum eget. Phasellus ut odio convallis, suscipit turpis eu, semper ex. Fusce massa nibh, cursus a elit feugiat, eleifend molestie ante. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam porta sapien in lacinia blandit. Nam cursus lorem dolor, nec euismod elit bibendum eget. Phasellus ut.

Thank You!



RESOURCE PAGE

Use these design resources in your Canva Presentation.

Fonts

This presentation template
uses the following free fonts:

TITLES:
DELA GOTHIC ONE

BODY COPY:
BRICE
SEMIEXPANDED
Nunito

You can find these fonts online too.

Colors



#EF4428 #FFAFD4

Design Elements



CREDITS

This presentation template is free for everyone to use thanks to the following:



for the presentation template

Pexels, Pixabay

for the photos

Happy designing!