Neon - Cloud-native PostgreSQL

Heikki Linnakangas

Neon Co-founder, PostgreSQL hacker

What is Neon?

- Startup
  - founded March 2021
- Cloud service
  - https://neon.tech/
- New storage system for PostgreSQL
  - Open source

What is Neon?

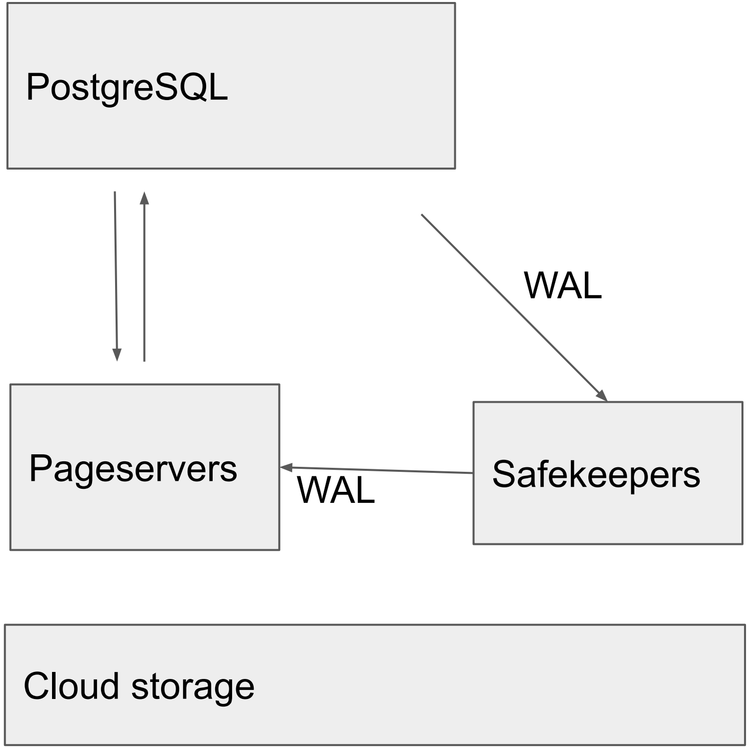
- Storage and compute are separated
- Single writer node, multiple readers
- Multi-tenant storage
- Single-tenant compute
  - runs in Kubernetes containers / VMs
- Cheap copy-on-write branching and timetravel query
- Open source

Neon architecture

Separation of Storage and Compute

Compute = PostgreSQL

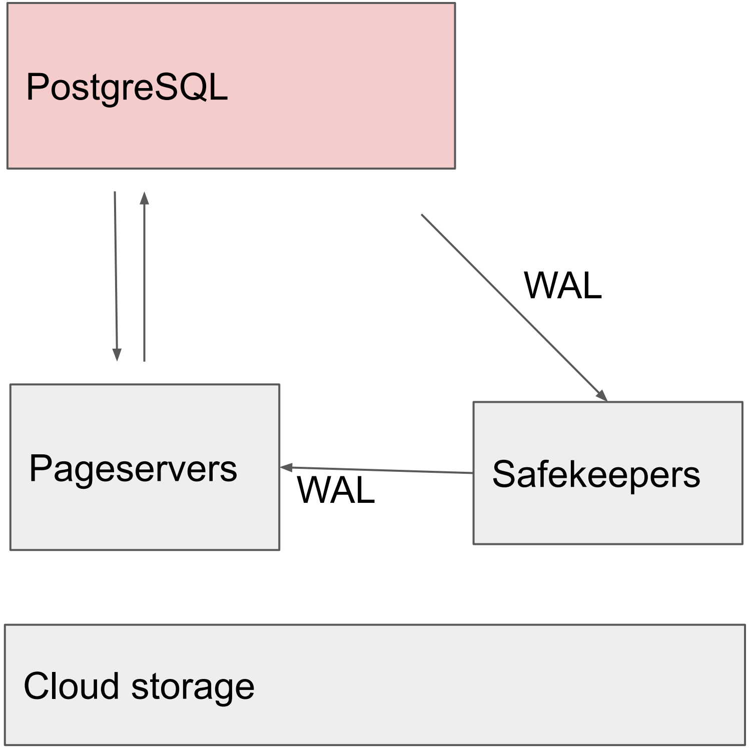
Storage = Neon storage system



Separation of Storage and Compute

PostgreSQL:

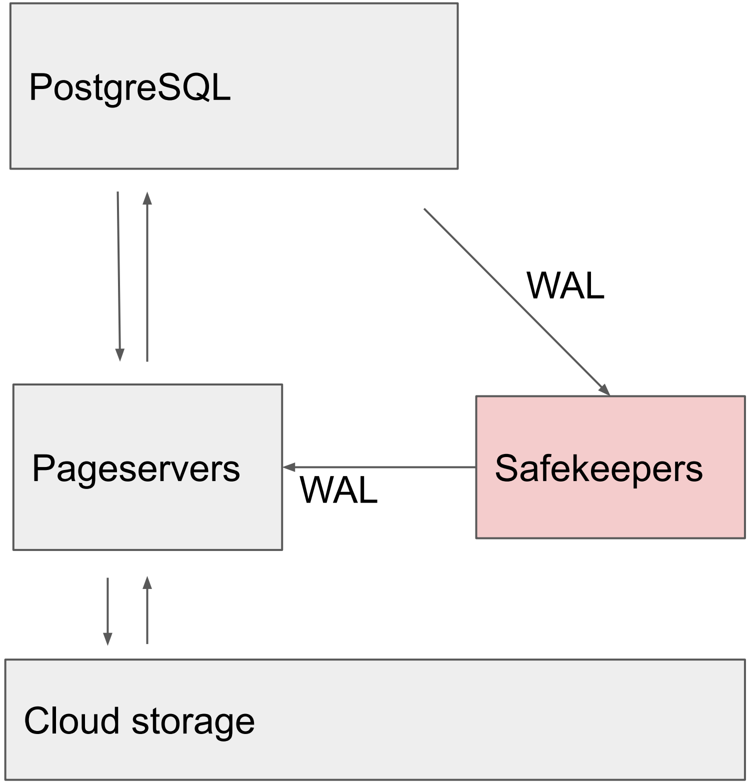
- streams WAL to the safekeepers
- reads pages from pageservers
- write() is a no-op
- local disk only for temporary files, sorting etc



Write path

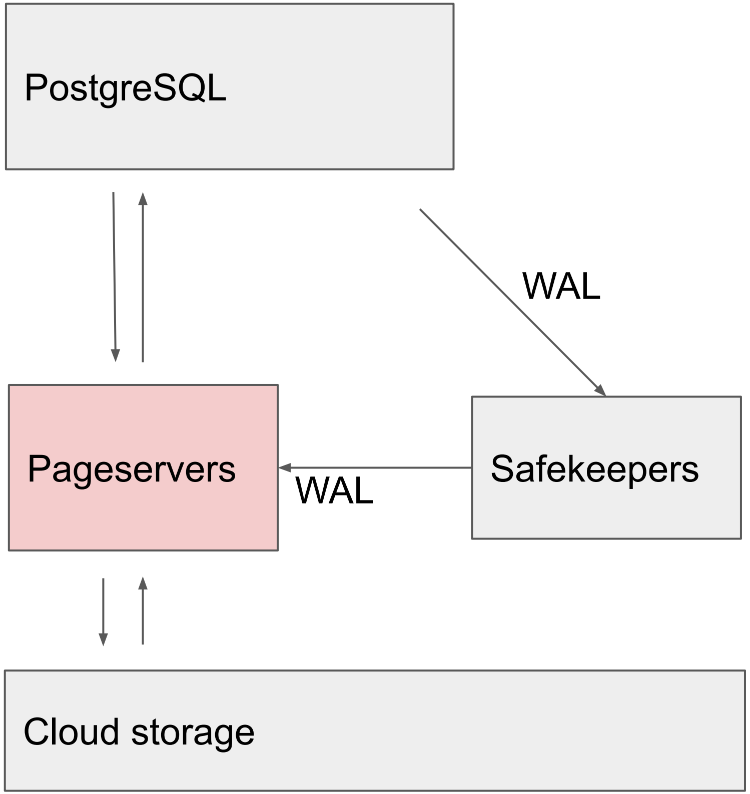
PostgreSQL streams the WAL to the safekeepers

- Three running safekeepers
- Consensus algorithm based on Paxos
- Ensures durability of recent transactions
- WAL is stored on local SSDs



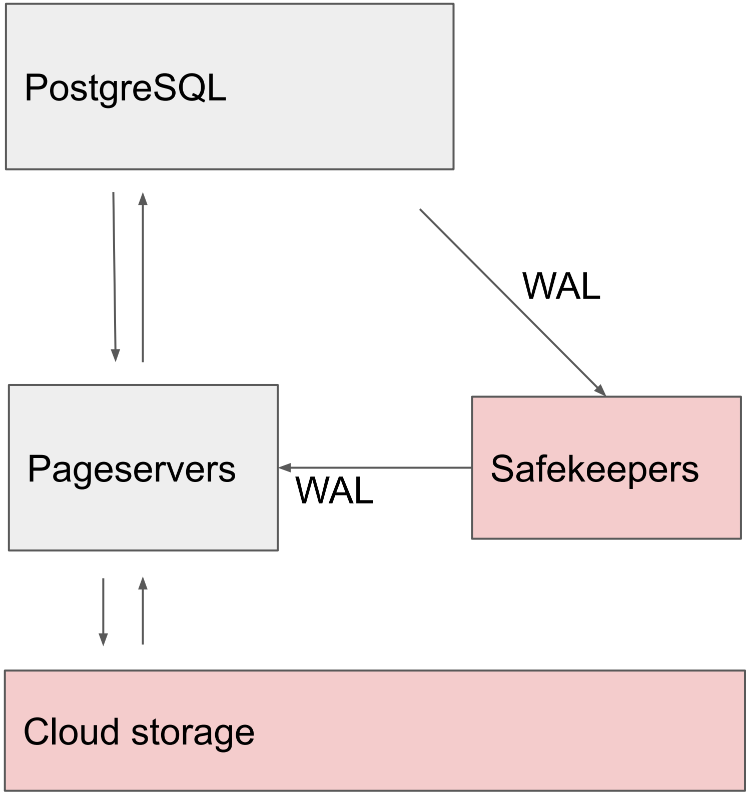
Write path: Pageservers

- Digests the PostgreSQL WAL
- Re-orders and processes it into immutable files
- Uploads files to cloud storage
- Local SSDs for caching



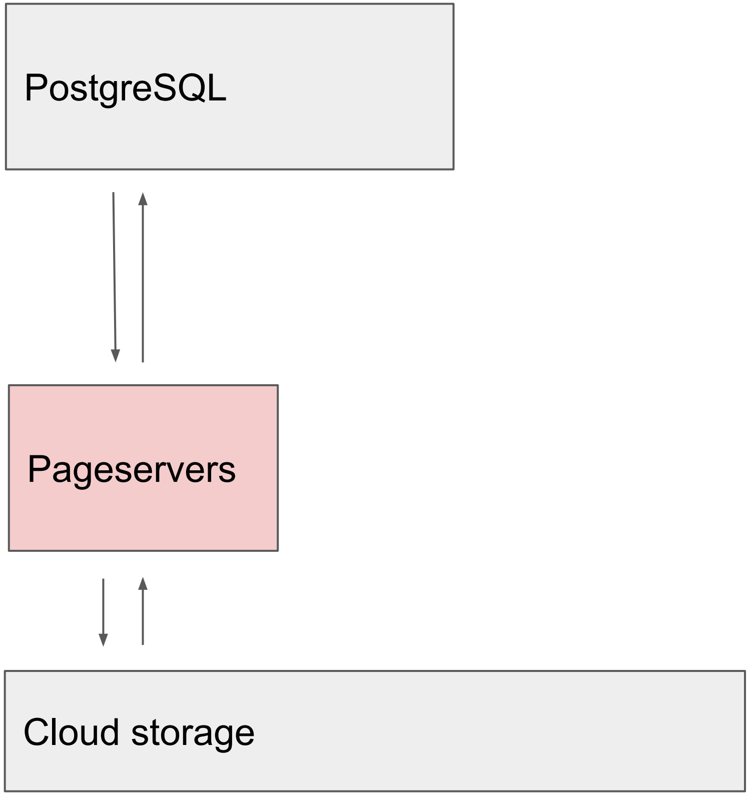
Durability

- Recent transactions (= recent WAL) are made durable in safekeepers
- Older WAL is uploaded to cloud storage, in processed format
- Pageservers are disposable



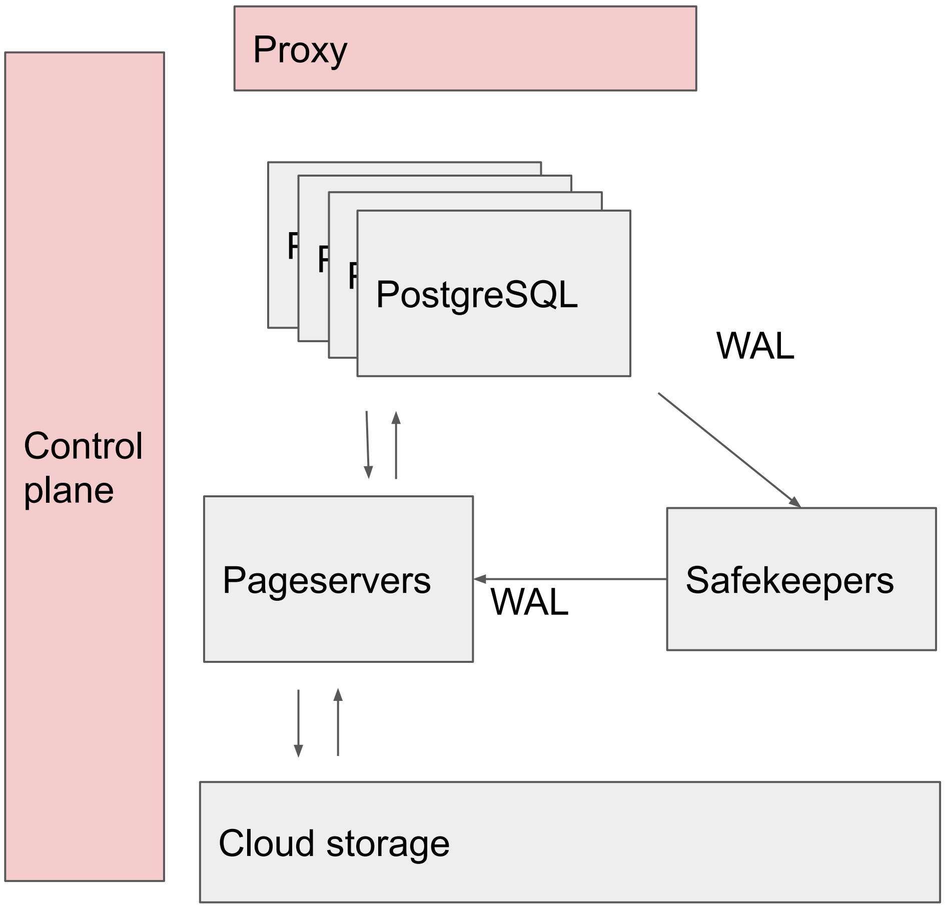
Read path: Pageservers

- Replays WAL to reconstruct pages, *on demand*
- Can reconstruct any page *at any point in time*
    - Keeps all history, up to some retention period



Control plane and proxy

- Control plane starts and stops compute nodes
- Provide web user interface and user-facing API for creating databases, branches etc.
- Proxy to accept and authenticate user connections



Ok, but why?

It’s serverless

- We launch Postgres on-demand, when you connect to it
- Shut down automatically if it’s idle for 5 mins
- Autoscaling up & down depending on load

Read-only replicas

- Scale out reads by adding read-only nodes
- Attached to the same storage, no need to duplicate the data

Bottomless Storage

- No need to provision storage ahead of time
- Unused data is swapped out to cloud storage (S3), and swapped back when needed

Branching

- The storage system allows cheap Copy-on-Write branches
- Want to try out a schema change?
    - Create a branch and test it there
- Integrate into CI workflow
    - Create branch of the database for every PR

Backups & archiving

- Neon storage keeps all history, and *makes it all accessible*
- Replaces traditional backups and WAL archive
- You can create a branch ***from any point in time***

Current status

PostgreSQL modifications

We try to minimize changes to PostgreSQL:

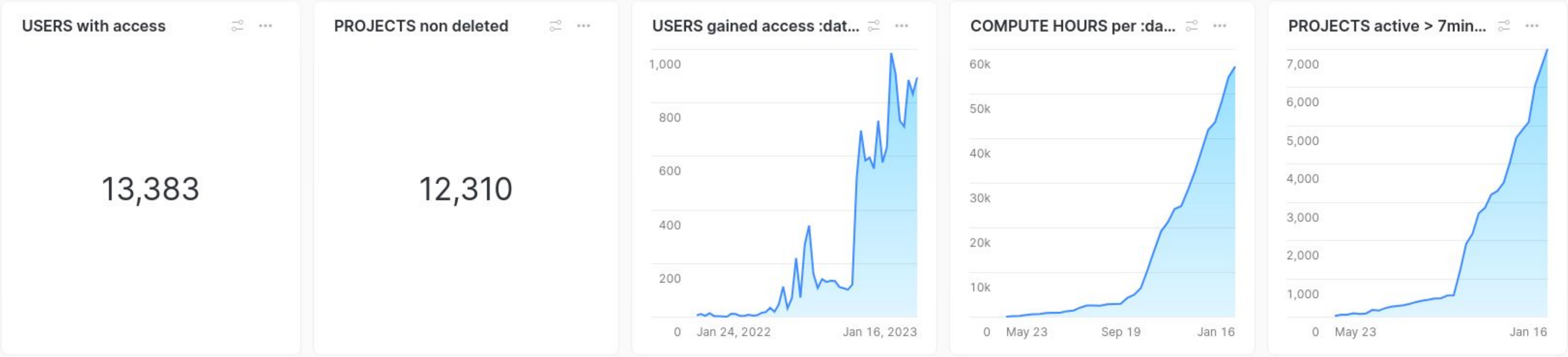
- No changes to planner or executor
- Support all extensions, tools
- Support all PostgreSQL index types
- PostgreSQL handles MVCC

Replace low-level storage, close to where read() & write() happens

- Goal is to get all changes into upstream PostgreSQL
- ~1000 lines diff at the moment

Technical preview

- It works!
- We are learning how to run a cloud service
- Usage is growing



USERS with access

13,383

PROJECTS non deleted

12,310

USERS gained access :dat...

COMPUTE HOURS per :da...

PROJECTS active > 7min...

Roadmap

- Billing
- UI for backup & restore
- UI for read-only nodes
- Multi-region
- Autoscaling
- Support larger databases, storage sharding
- Performance
- Extensions
- SOC 2 compliance
- *And remain easy to use!*

Thank you! Q & A!

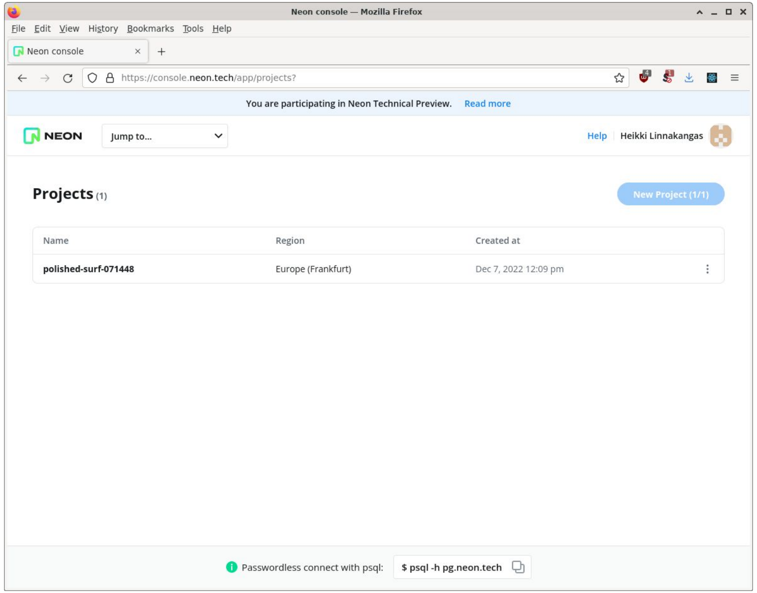
Try it: neon.tech

or: `psql -h pg.neon.tech`

Code: https://github.com/neondatabase/neon/

Feedback: heikki@neon.tech

We are hiring!



Neon console — Mozilla Firefox
File Edit View History Bookmarks Tools Help
Neon console
https://console.neon.tech/app/projects?
NEON
Jump to...
Help
Heikki Linnakangas
New Project (1/1)

You are participating in Neon Technical Preview. Read more

Projects (1)

Name	Region	Created at
polished-surf-071448	Europe (Frankfurt)	Dec 7, 2022 12:09 pm

Passwordless connect with psql: `$ psql -h pg.neon.tech`