Implementing High Availability with PostgreSQL and other open source tools

Sponsors

CONSORZIO
UNIVERSITARIO
DI PORDENONE
MOLTIPLICATORE DI VALORE

altitudo

beanTech
IT moves your business

[stesi]
Powered by Innovation

DATA
SATURDAYS

InnOva
community

About me

- Danilo Dominici
- Freelance Senior Consultant
  - SQL Server
  - Postgres
  - Redis
  - … data architectures
- Microsoft Certified Trainer dal 2000
- 6x Data Platform MVP
- SQL Start! founder

DATA SATURDAYS

inn0va community

DATA SATURDAYS

High Availability

Concepts

DATA SATURDAYS

What is «High Availability»

«The ability of a system to operate continuosly by eliminating a single point of failure»

How to measure High Availability

- Sooner or later, every system experiences a failure or an outage
- Power line can be interrupted, Disks can be corrupted, application instances can crash due to a memory overflow
- Measuring the uptime percentage, tipically referred to as percentage, over your 24X7 service should be part of your SLA

Uptime	Downtime per year
99.9%	8.76 hours
99.99%	56.2 minutes
99.999%	5.25 minutes
99.9999%	31.56 seconds

Factors affecting HA: RTO

- Availability SLA is affected also by the Recovery Point Objective (RPO) and Recovery Time Objective (RTO)
- Recovery time objective (RTO) is the amount of time within which the service must be restored and returned to normal operations.

Uptime	Downtime per year
99.9%	8.76 hours
99.99%	56.2 minutes
99.999%	5.25 minutes
99.9999%	31.56 seconds

Factors affecting HA: RPO

- Recovery point objective (RPO) is the maximum acceptable amount of data loss after an incident/failure/outage
- Usually, data loss is measured in bytes/megabytes/etc., by the number of lost transactions, or by the last seconds/minutes/etc. of changes.

Uptime	Downtime per year
99.9%	8.76 hours
99.99%	56.2 minutes
99.999%	5.25 minutes
99.9999%	31.56 seconds

1TB

Backup vs High Availability vs Disaster Recovery

Backup

When data is lost, corrupted or deleted, you can restore it

> RTO

> (potential) RPO

$

High availability

When applications have a catastrophic failure, run a second instance

< RTO

< RPO

$$

Disaster recovery

When applications have a catastrophic failure, run them in a different site

> (potential) RTO

> RPO

$$$

DATA SATURDAYS

HA scenarios with PostgreSQL

Single cluster: no HA at all

- With a single database cluster is virtually impossible to avoid the single point of failure
- We can use some techniques to reduce the impact of an outage/failure:
  - Frequent backups
  - Storage mirroring + virtualization/containerization
- But probably we are not able to strictly assure the SLAs

Use a standby server

- Using a standby server, that receive the changes occurred to the primary server, and a failover/failback solution, we can reduce the downtime to an acceptable amount of time
- PostgreSQL includes several data replication techiques:
  - Write-ahead Log (WAL) Shipping
    - File-based log shipping (WAL continuous archiving)
    - Streaming replication
  - Logical replication

https://www.postgresql.org/docs/current/different-replication-solutions.html

PostgreSQL features for HA

- WAL shipping based replication
- Streaming replication
- Logical replication

Streaming replication

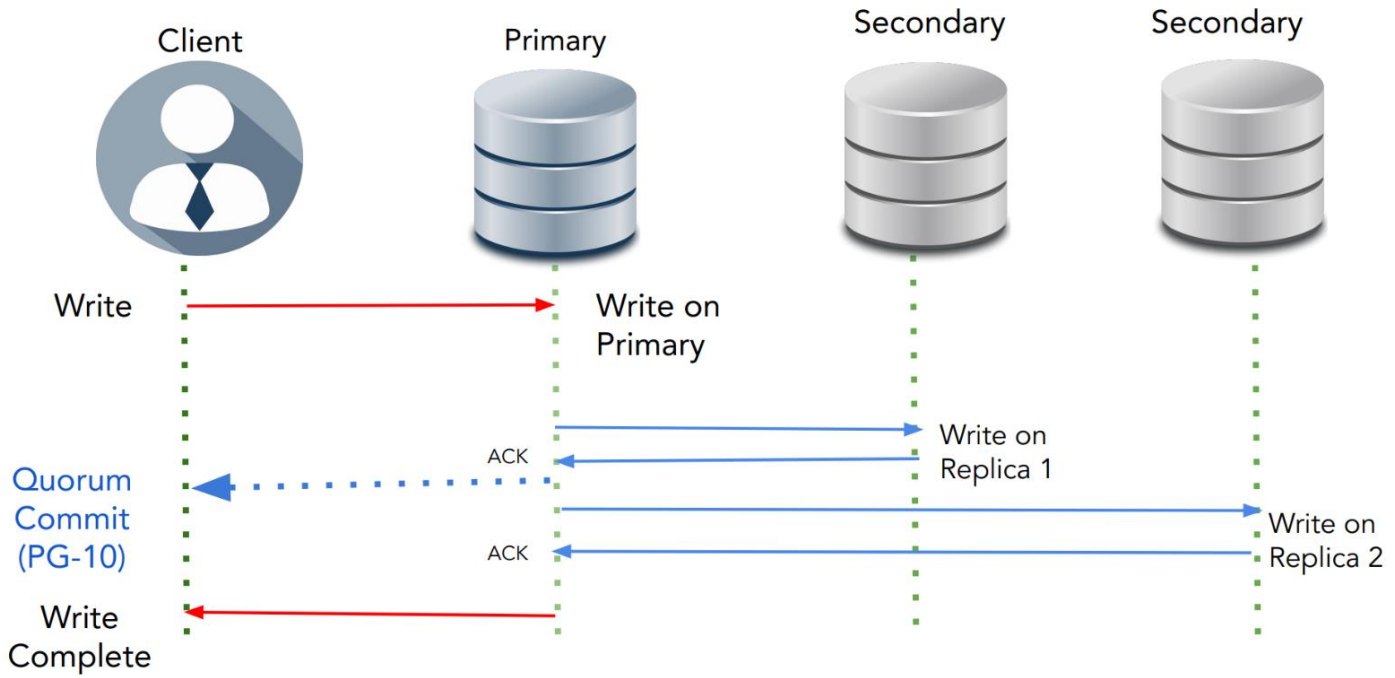
With streaming replication, the entire database cluster is replicated from one server, known as the primary, to one or more standby servers.

The primary server continuously streams the write-ahead logs (WAL) to the standby servers, which apply the changes to their own database copies. Streaming replication comes in two modes:

- **Synchronous streaming replication:** the primary server waits for confirmation from at least one standby server before committing a transaction. That confirmation ensures the data is replicated to a certain level of redundancy with minimal data loss.
- **Asynchronous streaming replication:** This is the **default** way of replicating data in PostgreSQL setups. In this mode, the primary server does not wait for confirmation before committing to transactions. The upside is enhanced performance, but there can be data loss in the event of a failure.

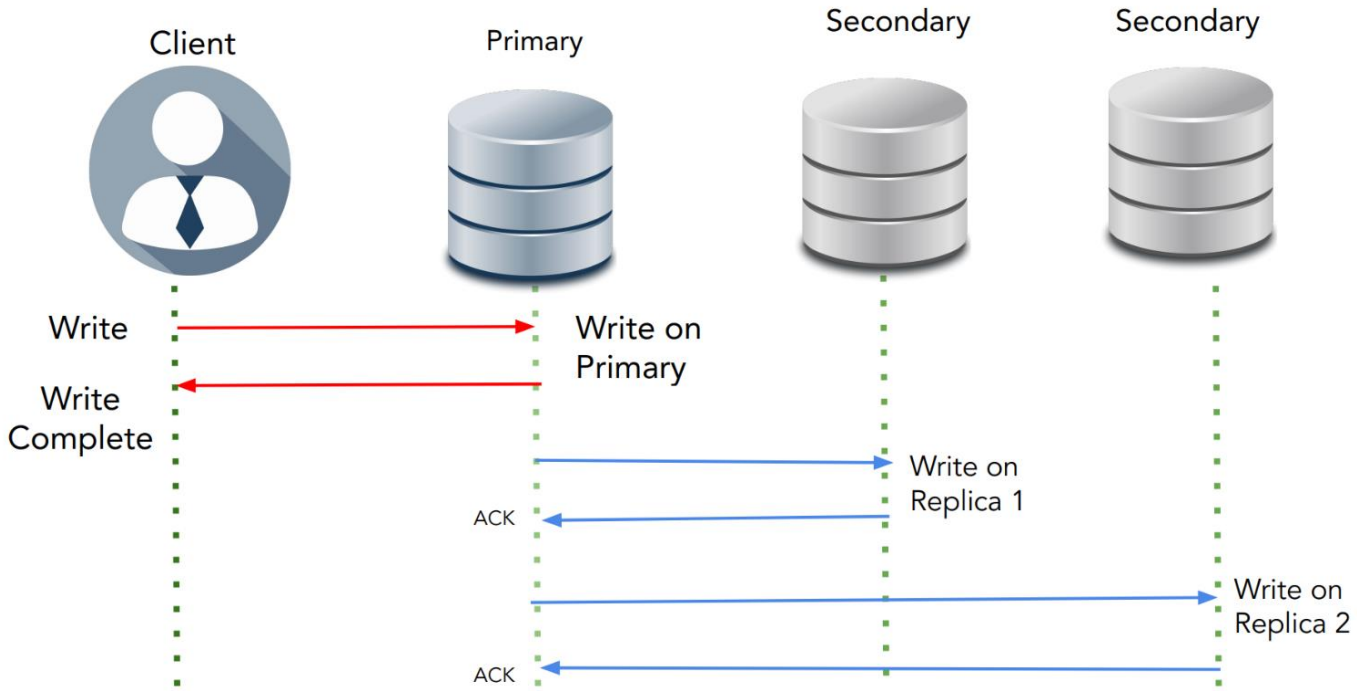
Streaming replication

Synchronous Replication



Streaming replication

Asynchronous Replication



Logical replication

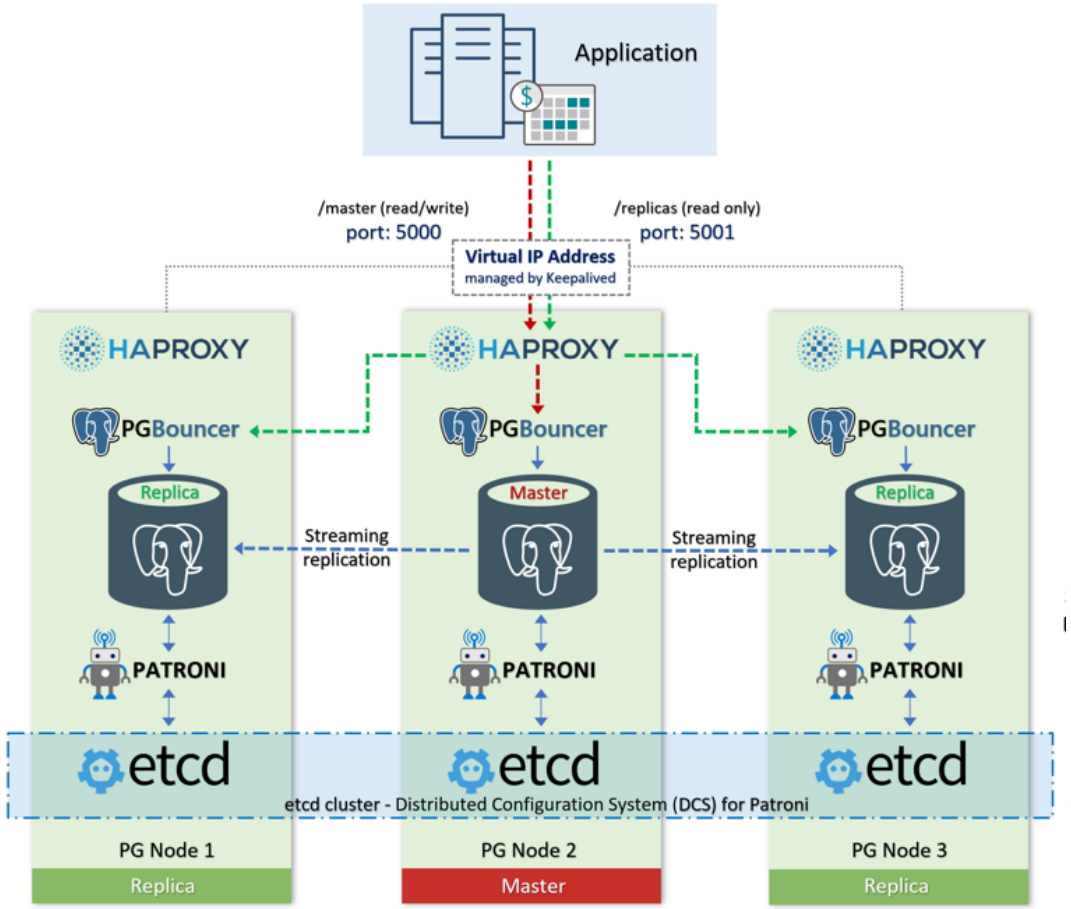
- Occurs at the **logical** level of the database objects, such as tables, rather than at the physical level of behind-the-scenes storage. Unlike streaming replication, which replicates the entire cluster, logical replication allows for more control over what data is replicated or not replicated.
- The two main components of logical replication are publishers and subscribers. A publisher shares data and a subscriber receives and applies the replicated data. This approach provides more flexibility for replication configuration.

Logical replication

- Setup
  - Change postgresql.conf to enable logical replication
- Create a publication
  - Ex. CREATE PUBLICATION my_publication FOR TABLE table1, table2
- Create a subscription
  - Ex. CREATE SUBSCRIPTION my_subscription CONNECTION 'host=source_host port=5432 dbname=source_db user=replicator password=secret' PUBLICATION my_publication

The «Patroni» scenario

The big picture



DATA SATURDAYS

HAProxy

HAProxy is a free, very fast and reliable reverse-proxy offering high availability, load balancing, and proxying for TCP and HTTP-based applications

HAProxy is built with sophisticated and customizable health checks methods, allowing a number of services to be load balanced in a single running instance.

A front-end application that relies on a database backend can easily over-saturate the database with too many concurrent running connections. HAProxy provides queuing and throttling of connections towards one or more PostgreSQL Servers and prevents a single server from becoming overloaded with too many requests.

All clients connect to the HAProxy instance, and the reverse proxy forwards the connection to one of the available PostgreSQL Servers based on the load-balancing algorithm used.

https://severalnines.com/resources/whitepapers/postgresql-load-balancing-haproxy/

pgBouncer

pgBouncer is a lightweight connection pooler for PostgreSQL

Advantages:

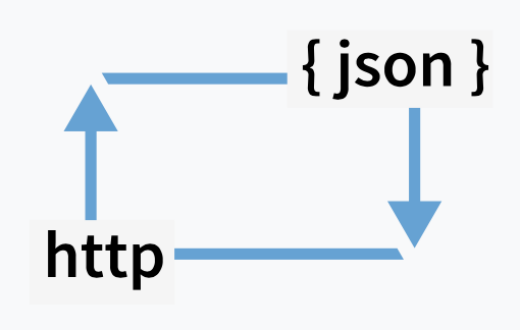
- Connection Pooling: PgBouncer maintains a pool of connections to the PostgreSQL database. Instead of connecting directly to the database, clients connect to PgBouncer, which then manages the connections to the database server.
- Connection Pool Management: It efficiently manages connections, reducing the overhead of connection setup and teardown. Connections are reused rather than being opened and closed for each client request, which can greatly improve performance, especially in high-traffic environments.
- Query Caching: It can cache frequently used queries, which can further reduce the load on the PostgreSQL server and improve overall performance.
- Authentication: PgBouncer can handle client authentication, reducing the need for the database server to authenticate each connection individually.

DATA SATURDAYS

ETCD

etcd is a strongly consistent, distributed key-value store that provides a reliable way to store data that needs to be accessed by a distributed system or cluster of machines. It gracefully handles leader elections during network partitions and can tolerate machine failure, even in the leader node.

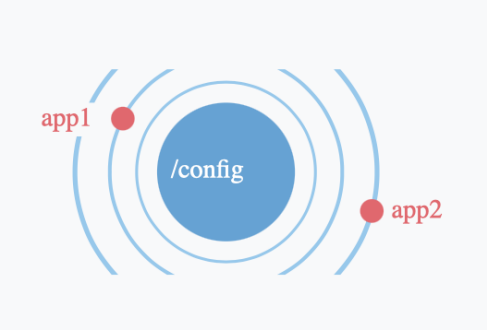
Read and write values using standard HTTP tools, like curl



Store data in hierarchically organized directories, as in a normal filesystem

```
/config
  └─/database
/feature-flags
  ├─/verbose-logging
  └─/redesign
```

Watch specific keys or directory changes, and react to changes in values



DATA SATURDAYS

Patroni

Patroni is an open-source python package that manages postgres configuration.

Patroni is a template for you to create your own customized, high-availability solution developed using Python.

It is also capable of handling Database replication, backup and restoration configurations.

Patroni's REST API uses port 8008 to provide the health checks and cluster messaging between the participating nodes

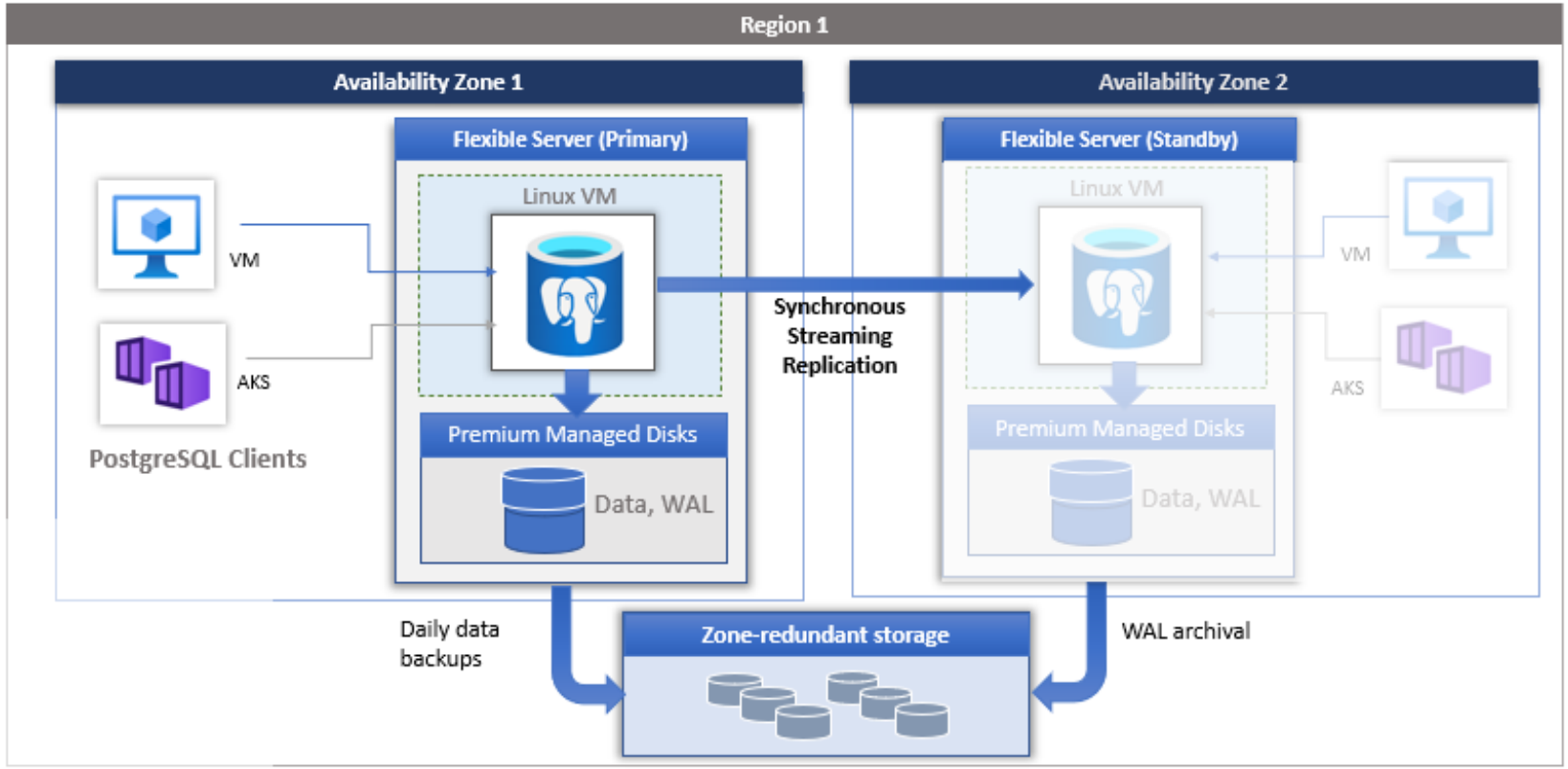
DATA SATURDAYS

DATA SATURDAYS

Azure Database for PostgreSQL

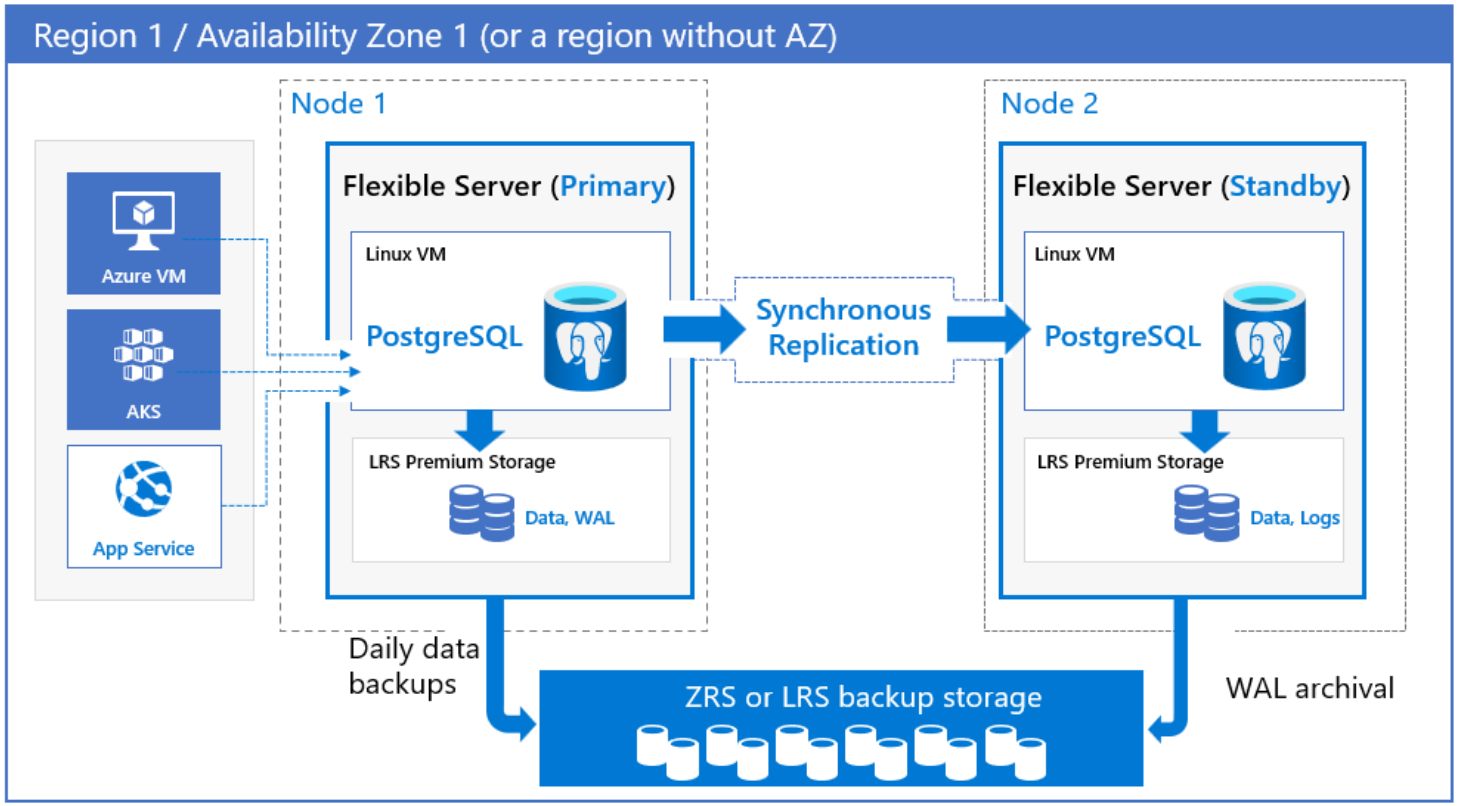
Azure Database for PostgreSQL – Flexible server

- Zone redundant



Azure Database for PostgreSQL – Flexible server

- Zonal deployment option



- A demo of the high availability for Azure Database for PostgreSQL here: https://www.youtube.com/watch?v=Yeb2Nv5mZtg

Recap

- PostgreSQL has High Availability capabilities from itself
  - …but if you want to automate the failover process needs some help
- Using PostgreSQL + etcd + HAProxy + Patroni you can achieve a real HA
- Dockerizing your environment is a key to a fast implementation

- A good example to start using VMs: https://docs.percona.com/postgresql/17/solutions/ha-setup-apt.html#install-the-software

- And using Docker: https://github.com/patroni/patroni/blob/master/docker/README.md