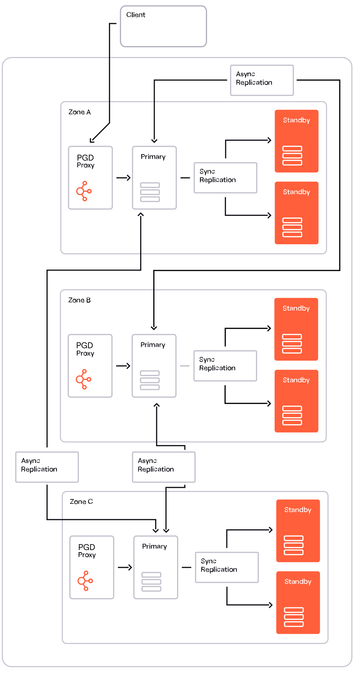
yugabyteDB

# 3 Ways to Scale PostgreSQL

## A Quick Peek at Distributed PostgreSQL

PostgreSQL is is a relational database designed for single-server deployments that lacks the capabilities of distributed databases. Typically, a 'distributed' PostgreSQL setup takes one of three forms.




*EDB PGD used for illustrative purposes

# 01

### Multi-Master with Asynchronous Replication

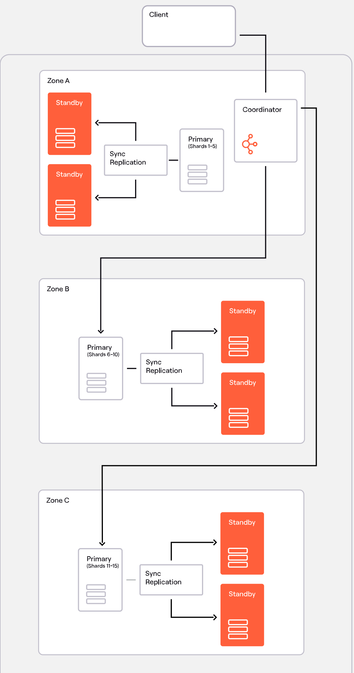
Provision multiple standalone PostgreSQL instances, with each storing a full data set and handling reads and writes. Instances replicate changes asynchronously.

#### Pros

+ Scale reads and writes.
+ Improve latency in selected locations

#### Cons

– Data conflicts can occur when using standard data types.
– Data volume is constrained by the capacity of a single database server.
– Upgrading to a larger server when data exceeds capacity can result in longer upgrade cycles, downtime, and reduced availability.




*CitusData used for illustrative purposes

# 02

### Multi-Master Sharded PostgreSQL with a Coordinator

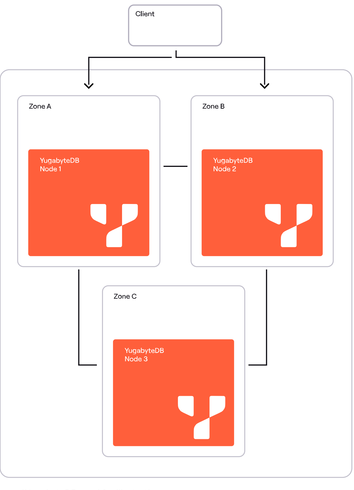
Data is sharded across multiple standalone PostgreSQL instances, with a coordinator node managing application connections and routing requests.

#### Pros

+ Scale data and read/write workloads horizontally.
+ Design configurations to meet specific availability SLAs, RPO, and RTO goals.
+ Good for multi-tenant apps and real-time analytics.

#### Cons

– Scaling process is not fully automated. Manual shard rebalancing may be necessary.
– Incomplete HA solution, requiring extra components for failover, failback, and load balancing.
– Limitations for OLTP workloads. Missing support for foreign/unique keys and eventually consistent cross-shard transactions.




*YugabyteDB used for illustrative purposes

# 03

### Multi-Master Shared-Nothing PostgreSQL

Utilizes a true distributed database, which is feature- and runtime-compatible with PostgreSQL.

#### Pros

+ Automatically scales data and read/write workloads (vertically and horizontally).
+ Inherently fault-tolerant.
+ Ensures RPO=0 and RTO between 3-15 seconds.
+ Eliminates many PostgreSQL maintenance tasks like vacuuming and managing transaction ID wraparounds.

#### Cons

– Certain application workloads and queries may need to be optimized to achieve greater performance within distributed database clusters.

Ready to learn more about distributed PostgreSQL?
Explore How to Scale a Single-Server Database: A Guide to Distributed PostgreSQL for a deeper understanding of these three architectures.

yugabyteDB

FOLLOW US

For additional information about our portfolio of services, contact your YugabyteDB expert at yugabyte.com/contact.