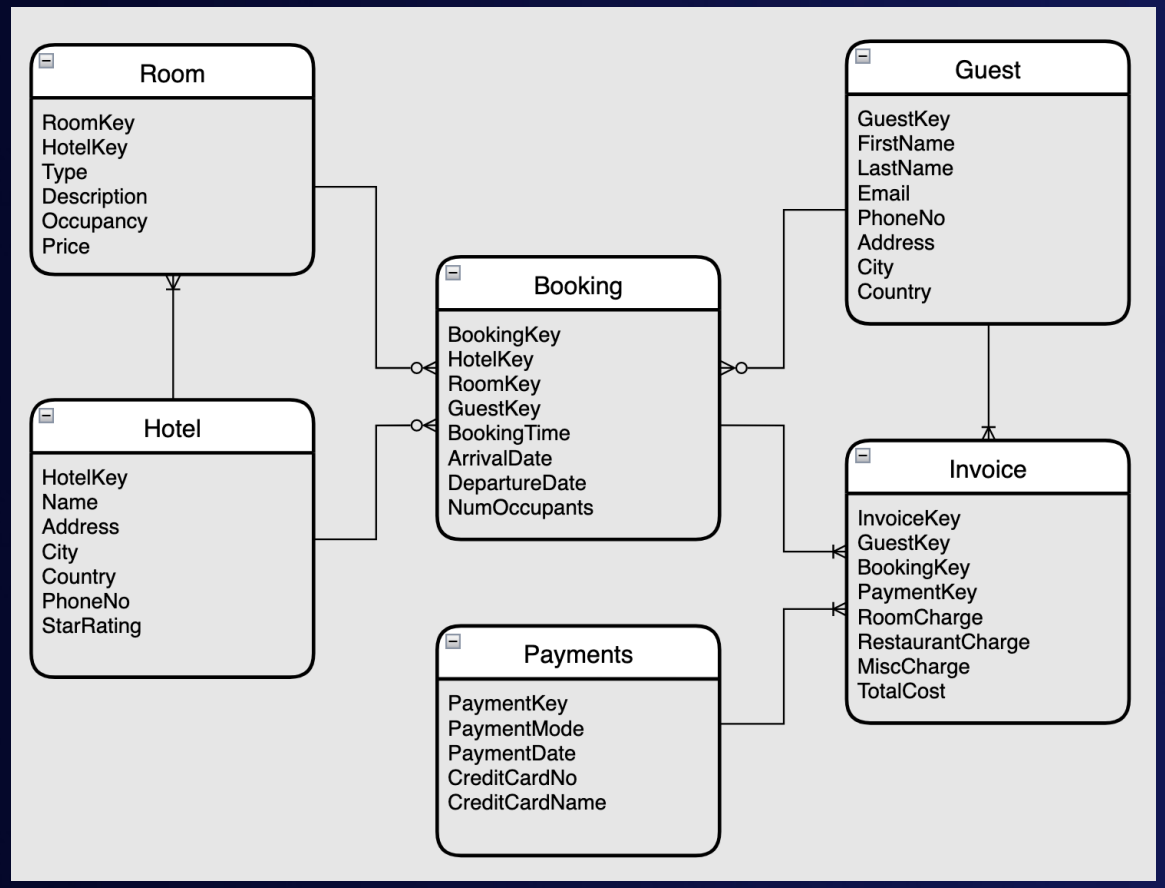
Databases
Part 2: Relational DBs

Aleksandr Bernadskii
Solutions Architect
Amazon Web Services

Relational DBs or NoSQL?




```
{
    "object": {
        "type": "booking",
        "bookingKey": "asdbc-odpkg-otjg7",
        "bookingTime": "2022-05-09 10:00:00 CET",
        "arrivalDate": "2022-05-10 10:00:00 CET",
        "departureDate": "2022-05-12 10:00:00 CET",
        "rooms": [{
            "type": "DoubleRoom",
            "Description": " Double Room with see view",
            "price": 145,
            "hotel": {
                "name": "AnyHotel",
                "address": "AnyAddress",
                "city": "AnyCity",
                "country": "AnyCountry"
            }
        }]
    }
}
```

Amazon Relational Database Service (RDS)

Managed relational database service with a choice of popular database engines

Amazon Aurora | MySQL | PostgreSQL | MariaDB | Microsoft SQL Server | ORACLE

Easy to administer

Easily deploy and maintain hardware, OS and DB software; built-in monitoring

Performant & scalable

Scale compute and storage with a few clicks; minimal downtime for your application

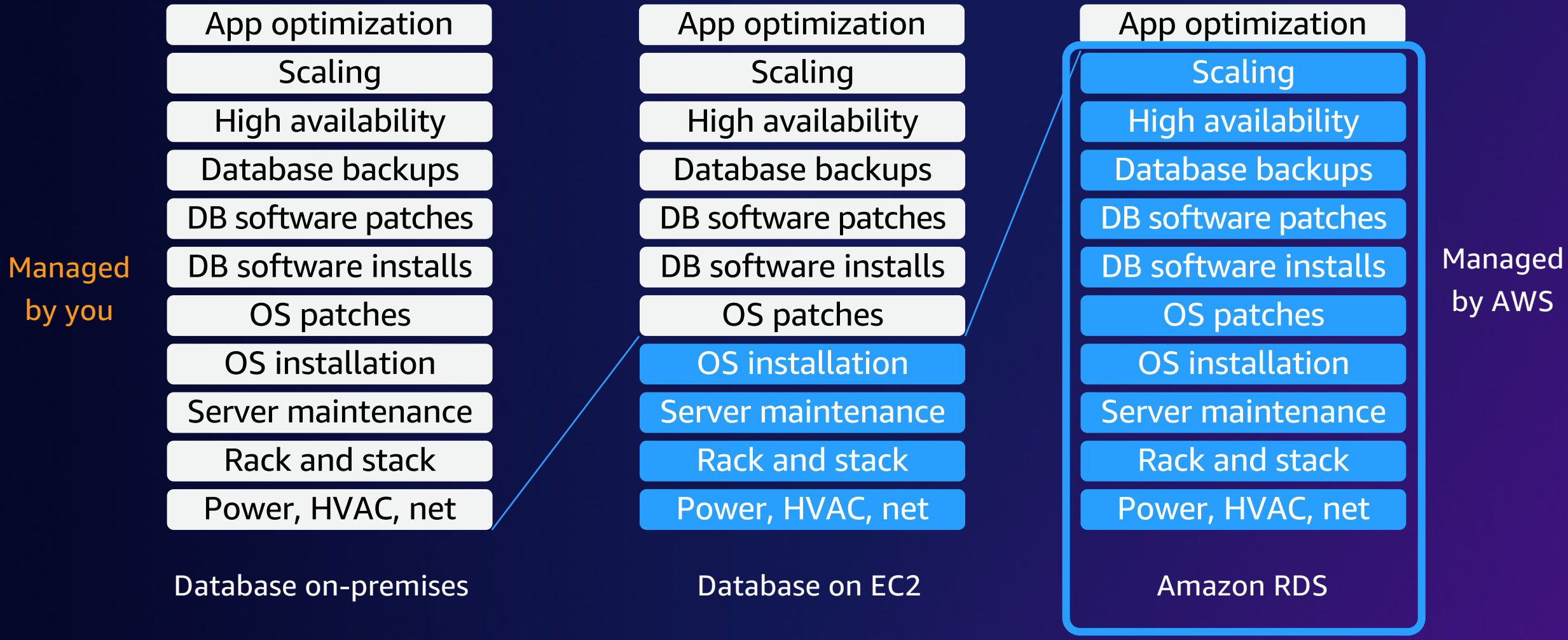
Available & durable

Automatic Multi-AZ data replication; automated backup, snapshots, and failover

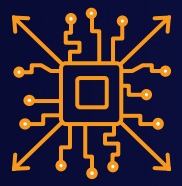
Secure & compliant

Data encryption at rest and in transit; industry compliance and assurance programs

Everything You Get From Managed Database…



Scale Compute and Storage With Ease



Scale compute to handle increased load

- Up to 64 vCPU and 488 GiB of RAM

Scale storage for larger data sets

- Quickly scale EBS storage up to 16TB
- No downtime for storage scaling

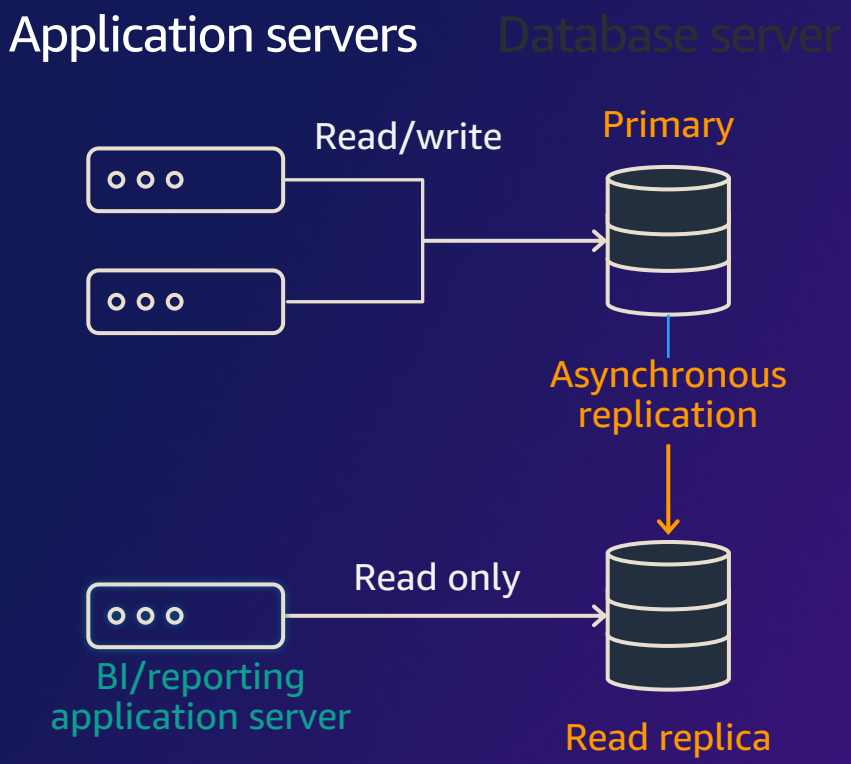
Scale down to control costs

- As little as 1vCPU / 1 GiB of RAM

Read Replicas

Read Scaling and Disaster Recovery

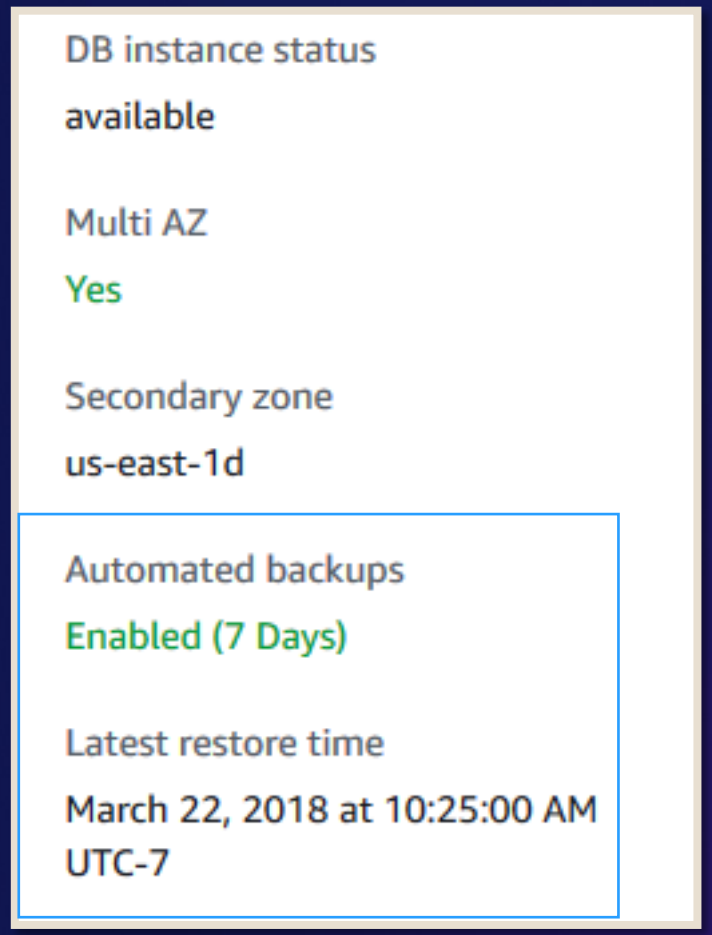
- Relieve pressure on your primary node with additional read capacity
- Bring data close to your applications in different regions
- Promote a read replica to a primary for faster recovery in the event of disaster



Automated Backups

Point-in-time Recovery for Your DB Instance

- Scheduled daily volume backup of entire instance
- Archive database change logs
- 35–day maximum retention
- Minimal impact on database performance
- Taken from standby when running Multi-AZ

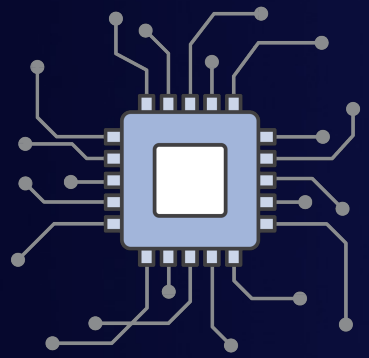


Every day during your backup window, RDS creates a storage volume snapshot of your instance

Every five minutes, RDS backs up the transaction logs of your database

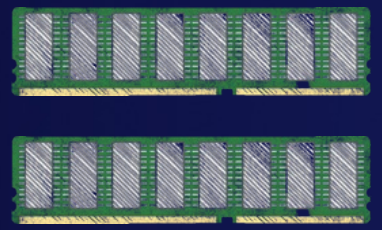
RDS Performance Factors

RDS DB Instance Class



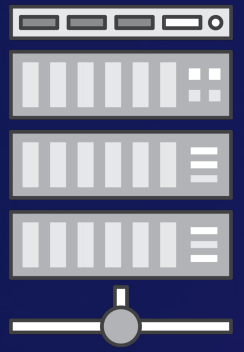
Compute Capabilities

vCPUs



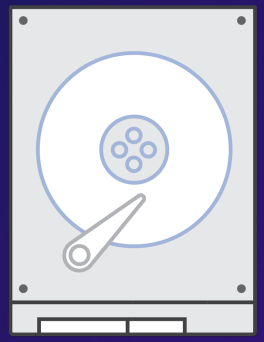
Memory Capabilities

GB of RAM



Network Performance

MB/s (Throughput)



Storage Performance

I/O Throughput

RDS Storage Type

Choose the right storage

Type	Size	IOPs	Burst	Pricing
GP2 (General Purpose SSD)	20 GiB - 64 TiB	3 IOPs/GiB, up to 16,000 IOPs	Yes, up to 3000 IOPS per volume, subject to credits (< 1 TiB in size)	Allocated storage
GP3 (General Purpose SSD)	20 GiB - 64 TiB	Baseline - 3000 IOPS	Provision how many your application needs	Allocated storage IOPS throughput
IO1 (Provisioned IOPS SSD)	Up to 64 TiB	1,000 - 80,000 IOPs*	No, fixed allocation	Allocated storage; Provisioned IOPS

* Nitro-based instance types, ½ for other instance types.

Scalability consideration: database schema

Primary key

- Define primary for each table
- Avoid running out of integer value, use bigint instead of integer

Foreign key

- Remove/reduce foreign key constraints on database
- Enforce referential integrity at application layer

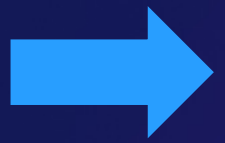
Choice of data type

- Choose smallest data type to represent desired range of values (e.g. use Boolean instead of “Yes”/”No” for binary flag)
- Offload large fields to separate tables/Amazon S3

Scalability consideration: connection management

Connection is expensive: OS process + associated backends

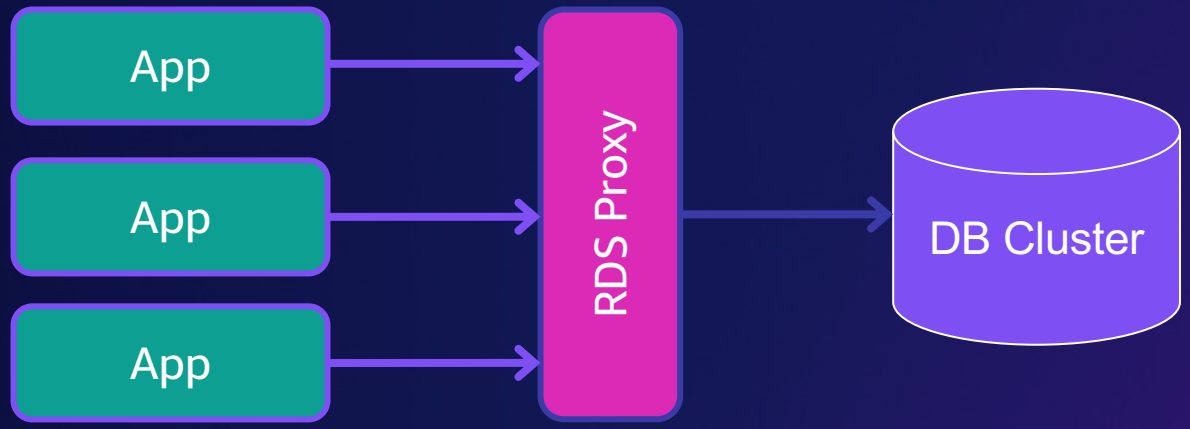
- Limit # of database connection with connection pool
- Avoid frequent connection/reconnect
- Avoid connection storms
- Recycle connection periodically to release resources
- Align timeout settings for all application layers



Implement connection proxy solution (RDS Proxy, pgBouncer, and etc.)

Connection Pooling with RDS Proxy

- Stabilizes and reduces connections against backend DB
- Multiplexes client requests at transaction boundaries
- Detects session state alterations and pins clients to backend connections
- Limit/avoid pinning by setting consistent variables on all sessions



Scalability consideration: workload characteristics

Optimize write heavy workload

- Remove unused indexes
- Ensure old row versions are efficiently purged
- Trade durability for immediate performance
- Checkpoint tuning

Optimize read heavy workload

- Limit columns returned in SELECT statements
- Index tuning (covering index, push predicate down)
- Avoid spilling sort/hash operation to disk
- Breakdown large complex query into smaller queries, merge results on the application layer
- Offload read-only workload to read replica
- Move OLAP workloads to a data warehousing solution, like Redshift

Scalability consideration: manage data growth

1. **Splitting reads and writes**
   Use endpoints, offload read-only traffics to reader, fast clones

2. **Sharding**
   Horizontal scaling (more clusters, or readers): customer growth, fleet re-balancing, changes in active/inactive mix)

3. **Partitioning**
   Subset of rows in separate table space, allows for archiving, pruning/selectivity in queries

4. **Optimize large object (LOB) storage**
   Offload large fields to separate tables/Amazon S3

Performance consideration: index

Common indexing problems and antipatterns

1. **Redundant secondary indexes**
   Same columns in the same left to right order are covered by other indexes
2. **Indexing on columns individually instead of composite**
   Only one index is selected for each query processing stage
3. **Low cardinality indexes**
   The engine still has to process a large number of index row matches, and pull row data from the clustered index
4. **Full length (or large prefix length) indexes on string columns**
   Storage, I/O and memory inefficiencies
5. **Index on NULLs**
   Index on nullable column occupy space, use partial index to save space

Troubleshooting database performance

Poor database performance symptoms:

- High CPU utilization
- High DB connections
- High memory utilization
- High read/write latency
- Lower throughput
- High active sessions
- High I/O usage
- High query execution times
- High History List Length

Potential reasons for slow queries:

- Lock, deadlock
- Insufficient hardware capacity (CPU,RAM)
- Parallel jobs
- Suboptimal queries

Monitoring performance

RDS

- Performance Insights – Counters and Wait types
- CloudWatch Metrics – DB Connections, CPU Utilization, Freeable Memory, Read/Write Latency, Queue Length, Replica Lag
- Enhanced Monitoring - memory.free cpuUtilization.*, loadAverageMinute.*, processList, read/write latency, read/write IOs PS

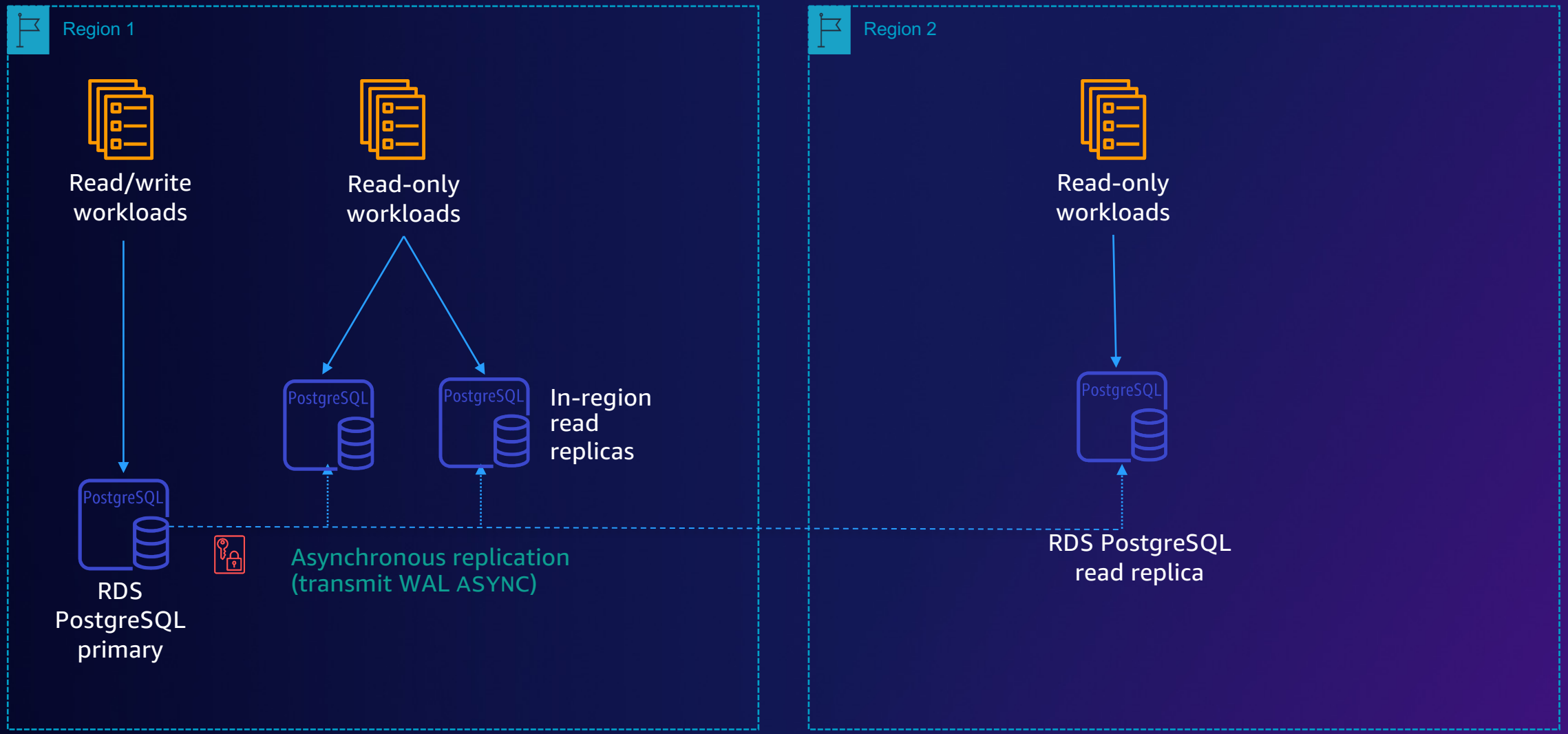
PostgreSQL Engine

- Postgres log – log_* (log_statement, log_min_duration_statement, log_connections/log_disconnections, log_autovacuum_min_duration)
- Postgres statistics collector
- Schedule to capture of statistics views
- Contrib modules – pg_stat_statements, auto_explain, pg_buffercache, pgrowlocks, pgfreespacemap, pgstattuple

Query Analysis

- Explain Analyze
- Pg_stat_statements
- Auto_explain

HA & DR

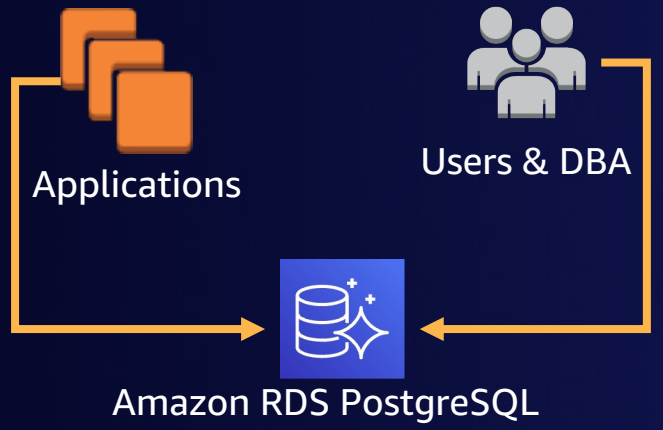


Choose the right HA/DR options

Feature	RPO (approximate)	RTO (approximate)
Multi-AZ for high availability	0	1 to 2 minutes
Automated Snapshot restore	Hours	<1 hour
Manual Snapshot restore	Depends on the time of snapshot	< 1 hour
Point-in-time restore	5 minutes	<1 to several hours
RDS Read replicas (in-region)	Depends on the replication lag	<5 minutes
RDS Read replicas (cross-region)	Depends on the replication lag	<5 minutes
External replicas	Depends on the replication lag	Minutes to hours

Access control at a glance

Access control at DB level

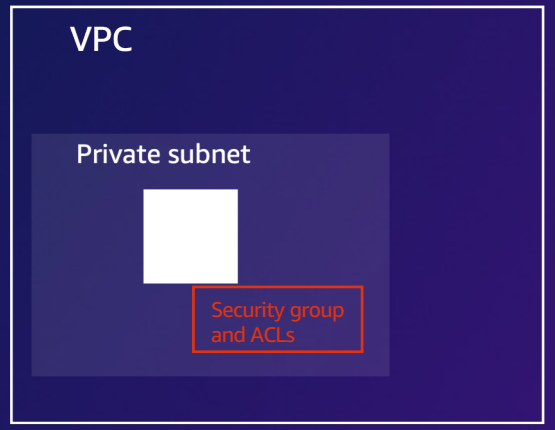


Schema/object level privileges
Role based access controls in Postgres

Controlled with IAM

Governed access: use IAM to control who can perform actions with Amazon RDS

Network Security



Secure network access controlled through Amazon VPC security groups

Encryption

Encryption of data at rest

- AES256-based storage encryption, incl. backups, snapshots and metadata (Performance Insights, CloudWatch Logs publishing, Enhanced Monitoring)
- Key management using AWS KMS
- No performance impact on workloads
- Other options:
  - Client-side encryption
  - pgcrypto extension

Encryption of data in transit (TLS)

Server

- A region-specific SSL certificate is available on Amazon RDS instances
  - Used to encrypt network traffic
  - Also used to verify the endpoint to guard against spoofing attacks
- Set rds.force_ssl to 1 to force SSL

Client

- Request the type of SSL connection in sslmode connection string, sslmode can be “require”, “verify-ca”, “verify-full”

```
psql -h $RDSHOST -p 5432 "dbname=postgres user=jim
sslrootcert=rds-bundle.pem sslmode=verify-ca"
```

Amazon Aurora

MySQL and PostgreSQL-compatible relational database built for the cloud

Performance and availability of commercial-grade databases at 1/10th the cost

---

Performance & Scalability

5x throughput of standard MySQL and 3x of standard PostgreSQL; scale-out up to 15 read replicas

Availability & Durability

Fault-tolerant, self-healing storage. Six copies of data across three AZs. Continuous backup to Amazon S3

Highly Secure

Network isolation, encryption at rest and in transit

Fully Managed

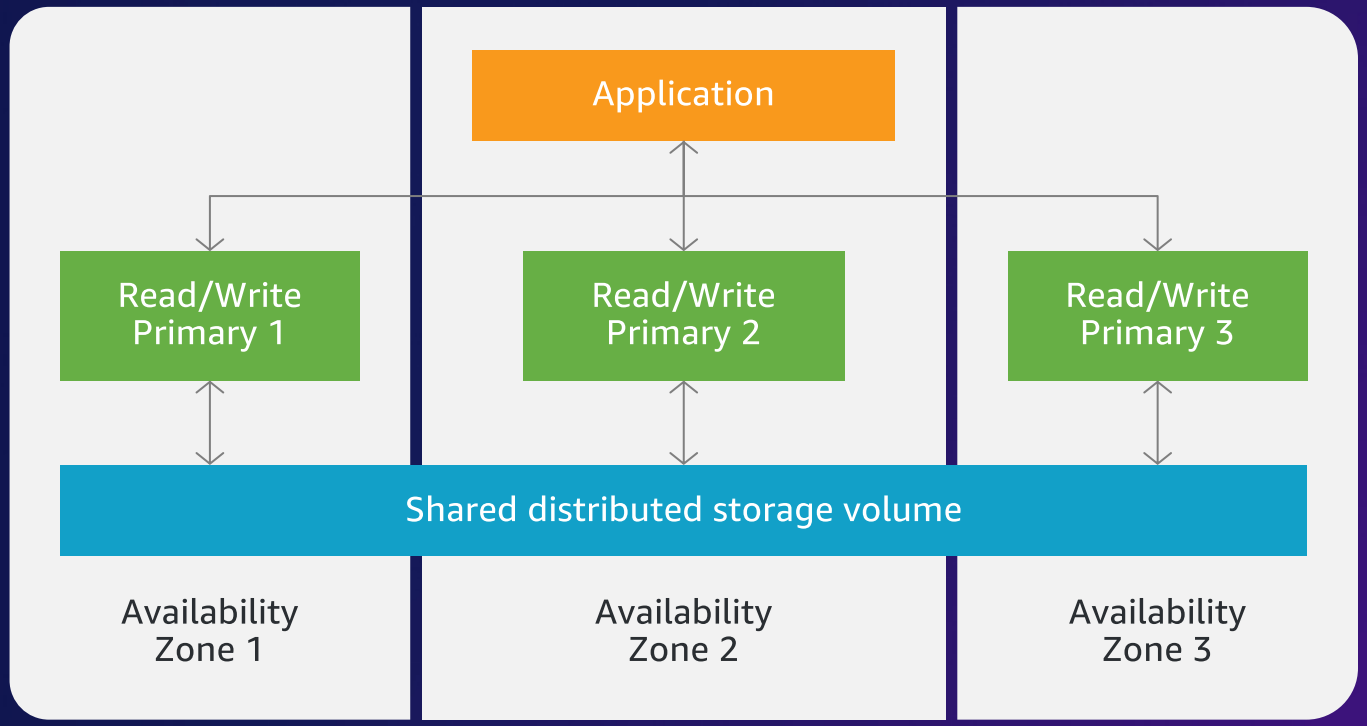
Managed by Amazon RDS: no hardware provisioning, software patching, setup, configuration, or backups

Amazon Aurora MultiPrimary

First MySQL-Compatible DB Service with Scale-Out Across Multiple Data Centers

- Zero downtime from ANY instance failure
- Zero downtime from ANY AZ failure
- Faster write performance and higher scale
- Aurora Global Database

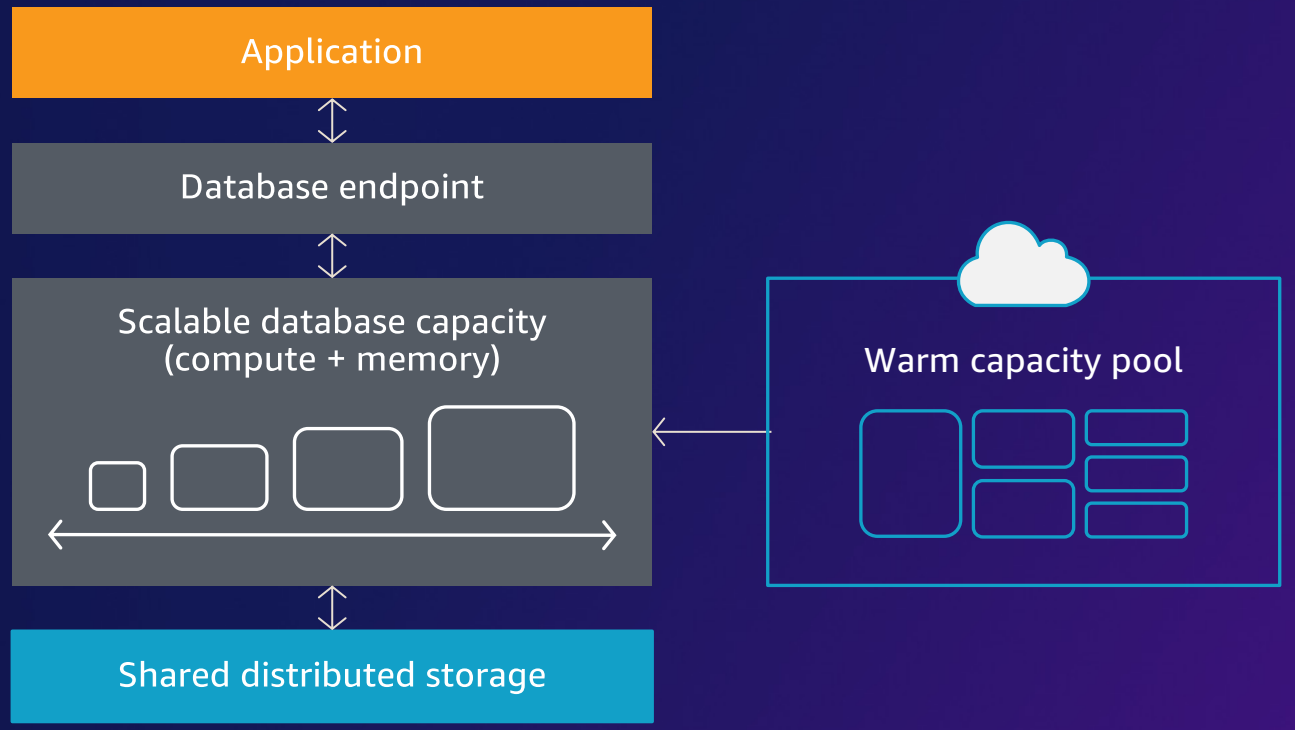
Scale-out both reads **and writes**



Amazon Aurora Serverless

On-Demand, Auto-Scaling Database for Applications with Variable Workloads

- Starts up on demand, shuts down when not in use
- Automatically scales, with no instances to manage
- Pay-per-second for the database capacity you use



Amazon Redshift – Data Warehousing

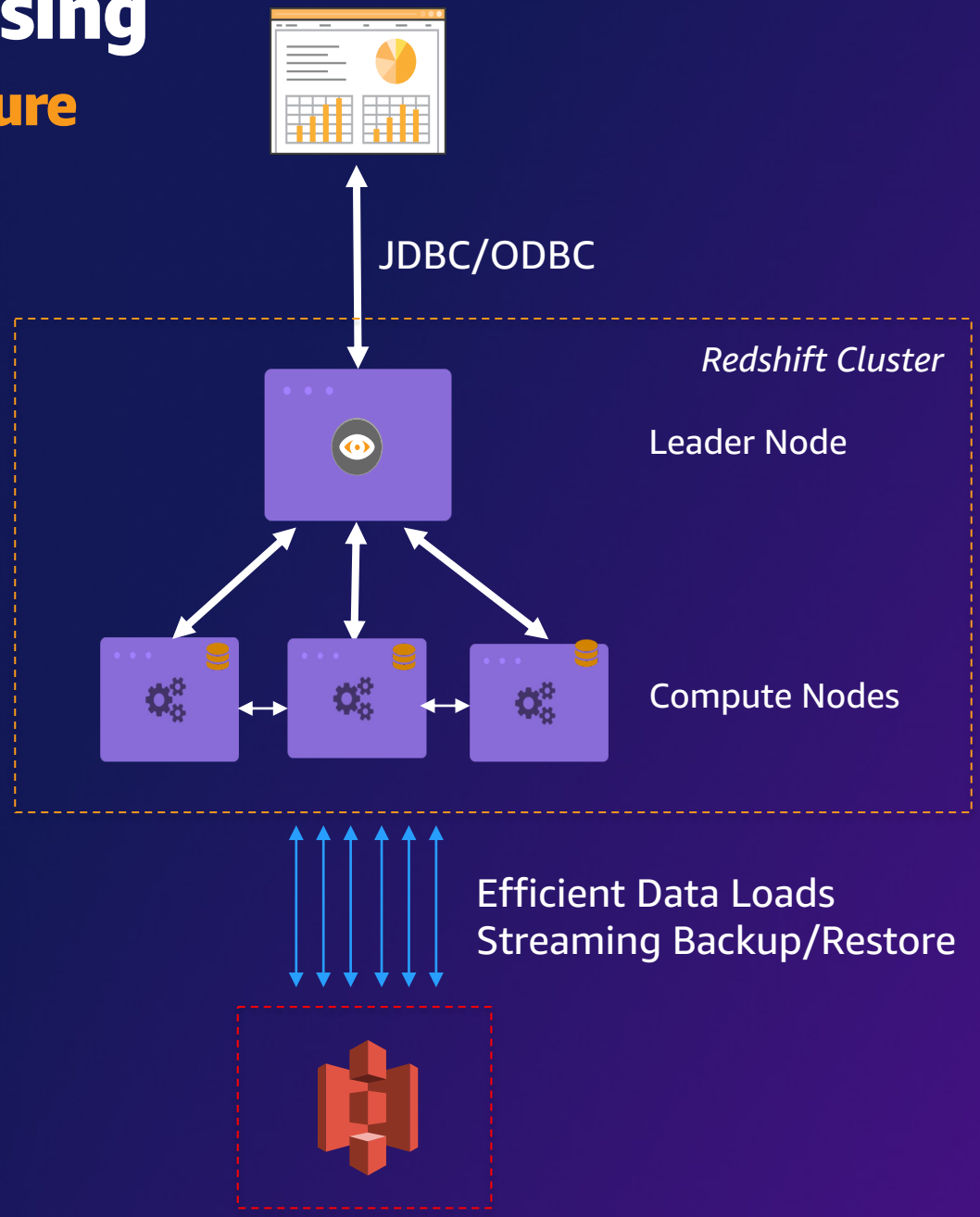
Massively Parallel, Shared-Nothing Architecture
Streaming Backup/Restore from S3

Leader node

- SQL endpoint
- Stores metadata
- Coordinates parallel SQL processing

Compute nodes

- Local, columnar storage
- Executes queries in parallel
- Load, backup, restore
- 2, 16, or 32 slices



Amazon Athena

Query across a modern data architecture with SQL

Serverless

Quickly query S3 data without managing infrastructure, and pay only for the queries you run

Fast interactive performance

Parallel execution to deliver most results within seconds, with no cluster management required

Open and standard

Use ANSI SQL for querying with support for Parquet, CSV, JSON, Avro and other standard data formats

Cost effective

Pay only for queries run and save 30–90% by compressing, partitioning, and converting your data into columnar formats

Links

Docs:

- https://docs.aws.amazon.com/rds/index.html
- https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/CHAP_AuroraOverview.html
- https://aws.amazon.com/blogs/database/

Workshops:

- https://catalog.us-east-1.prod.workshops.aws/workshops/2a5fc82d-2b5f-4105-83c2-91a1b4d7abfe/en-US
- https://catalog.us-east-1.prod.workshops.aws/workshops/31babd91-aa9a-4415-8ebf-ce0a6556a216/en-US/prep/env/own-account
- https://catalog.us-east-1.prod.workshops.aws/workshops/098605dc-8eee-4e84-85e9-c5c6c9e43de2/en-US

Thank you!