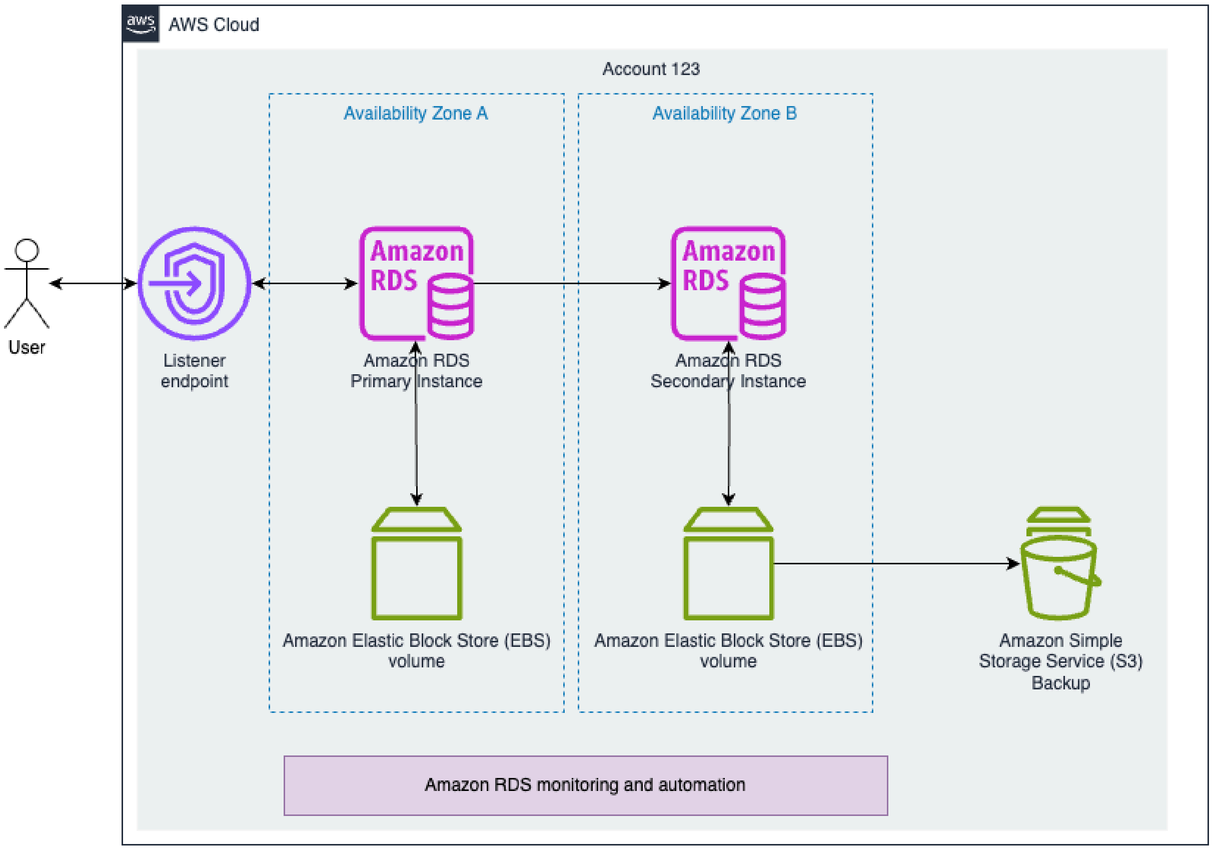
8. Backup & Disaster Recovery Strategy

This solution implements layered backup and disaster recovery mechanisms across both cloud-hosted and on-premise components.
The strategy focuses on **data durability, recoverability, and operational safety**, while avoiding unnecessary over-engineering for a proof-of-concept scope.


AWS Cloud
Account 123
Availability Zone A
Availability Zone B
User
Listener endpoint
Amazon RDS
Amazon RDS Primary Instance
Amazon RDS
Amazon RDS Secondary Instance
Amazon Elastic Block Store (EBS) volume
Amazon Elastic Block Store (EBS) volume
Amazon Simple Storage Service (S3) Backup
Amazon RDS monitoring and automation

8.1 Amazon RDS (MySQL) – Backup & Disaster Recovery (Implemented)

Amazon RDS provides native, managed backup and availability features that are leveraged to meet disaster recovery requirements with minimal operational overhead.

Automated Backups and Point-in-Time Recovery (PITR)

```
backup_retention_period = 7
```

Behavior

- AWS automatically takes daily snapshots of the database
- Transaction logs are continuously captured
- Enables point-in-time recovery within the last seven days

Design Value

- Database can be restored to any second within the retention window
- Protects against accidental deletes, faulty deployments, and data corruption
- No manual snapshot scheduling required

Multi-AZ Deployment (High Availability)

```
multi_az = true
```

Behavior

- A synchronous standby replica is created in a separate Availability Zone
- AWS automatically manages replication and failover

Design Value

- Automatic failover during AZ outages
- Minimal application downtime
- Provides availability and resilience (not read scaling)

> **Note:** Multi-AZ is an availability mechanism, not a scaling strategy.

Final Snapshot on Deletion (Data Safety Net)

```
skip_final_snapshot       = false
final_snapshot_identifier = "bookdb-final-snapshot"
```

Behavior

- A final snapshot is taken before the RDS instance is deleted
- Snapshot is retained independently of the database lifecycle

Design Value

- Prevents irreversible data loss
- Enables full database restoration after accidental deletion
- Acts as a last-resort recovery mechanism

Deletion Protection (Human Error Mitigation)

```
deletion_protection = true
```

Behavior

- Prevents accidental deletion via Terraform or AWS Console
- Deletion requires explicit disabling of protection

Design Value

- Safeguards production data
- Enforces intentional, reviewed destructive actions

Storage Encryption (Data at Rest)

```
storage_encrypted = true
```

Behavior

- Encrypts database storage
- Encrypts automated backups and snapshots

Design Value

- Aligns with security best practices and compliance expectations
- Protects data even if storage media is accessed improperly

Maintenance Window and Controlled Patching

```
maintenance_window         = "sun:02:00-sun:03:00"
auto_minor_version_upgrade = true
```

Behavior

- AWS applies minor version upgrades and patches only during the defined window

Design Value

- Predictable maintenance behavior
- Avoids unplanned downtime during business hours
- Reduces operational surprises

Snapshot Tagging (Operational Clarity)

```
copy_tags_to_snapshot = true
```

Behavior

- Automatically copies resource tags to snapshots

Design Value

- Simplifies snapshot identification
- Improves cost allocation and auditability
- Supports cleaner operations in multi-environment setups

Restore Capabilities (Disaster Scenarios)

With this configuration, the database can be:

- Restored to a new RDS instance
- Restored into a different Availability Zone
- Copied and restored into another region or VPC (manual cross-region copy)

This satisfies disaster recovery requirements without introducing unnecessary complexity.

8.2 On-Premise Backup Strategy (Conceptual, Realistic)

Unlike Amazon RDS, on-premise systems do not provide managed backups.
Backups are therefore explicitly automated and stored off-site in Amazon S3.

Backup Scope

The following components are backed up:

- MySQL database (`mysqldump`)
- Application artifacts
- Configuration files

Backup Flow (Conceptual)

```
On-Prem Server
   ↓
mysqldump / tar
   ↓
Upload to Amazon S3
   ↓
S3 Lifecycle Policies
```

This approach ensures off-site, durable backups with no dependency on local storage.

Example Backup Script (Conceptual)

```
#!/bin/bash
mysqldump -u root -p appdb > appdb.sql
aws s3 cp appdb.sql s3://onprem-backups/appdb-$(date +%F).sql
```

Design Value

- Enables regular, automated off-site backups
- Protects against on-prem hardware failure
- Allows straightforward restoration into Amazon RDS if required

---

S3 Lifecycle Policies (Retention and Cost Control)

Configured conceptually to:

- Retain daily backups for 30 days
- Transition older backups to lower-cost storage (e.g., Glacier)
- Automatically expire outdated backups

Why Amazon S3

- Extremely high durability (11 nines)
- Cost-effective compared to block storage
- Native lifecycle management simplifies retention enforcement

---

8.3 Design Intent Summary

- Cloud-hosted databases rely on managed AWS backup and availability features

- On-premise systems use scripted backups with off-site storage
- Data protection is layered, intentional, and operationally simple
- The strategy balances resilience with PoC-appropriate complexity