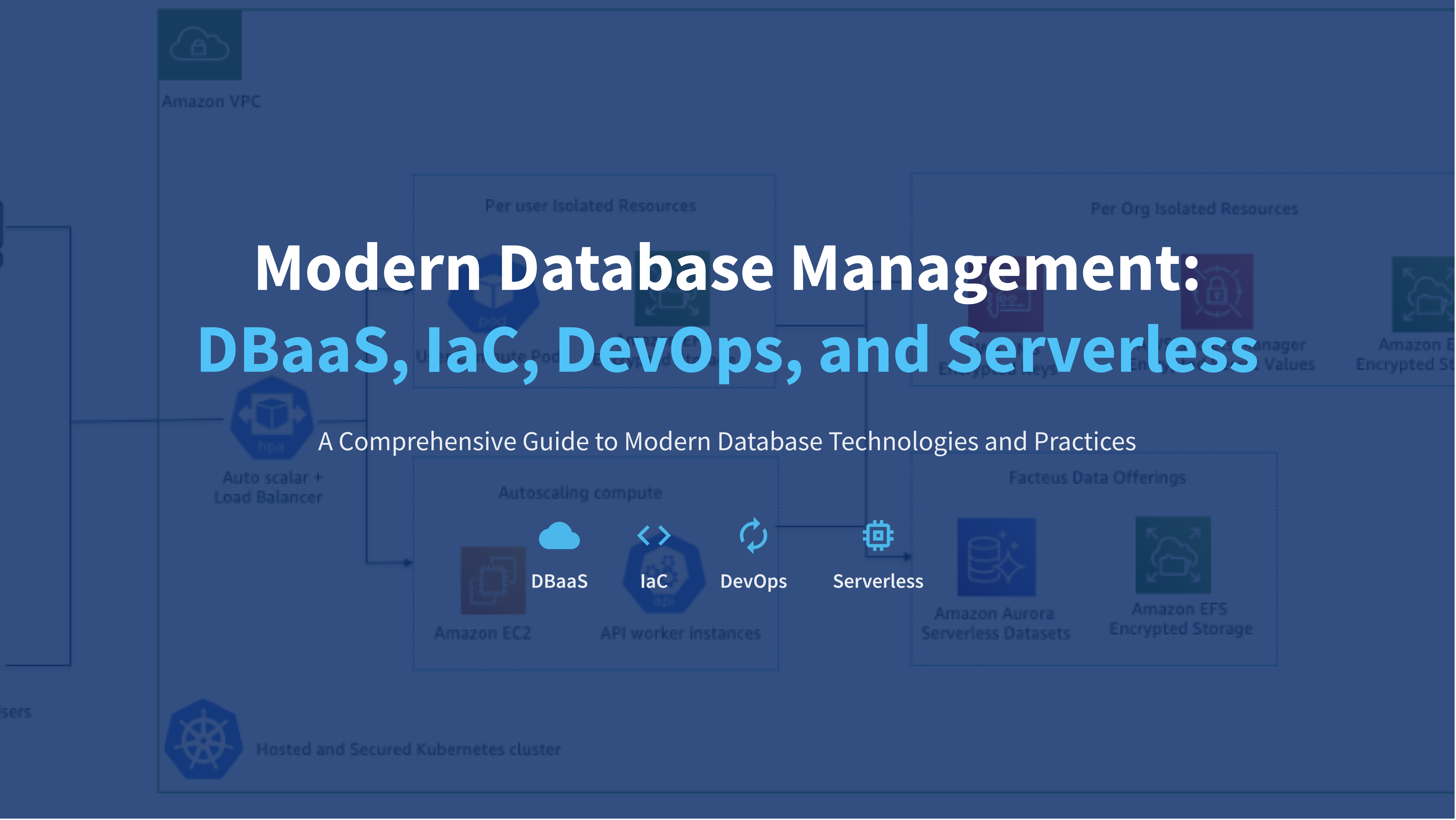


Modern Database Management: DBaaS, IaC, DevOps, and Serverless

A Comprehensive Guide to Modern Database Technologies and Practices

DBaaS IaC DevOps Serverless

Overview

01

Database-as-a-Service (DBaaS)

AWS RDS, Azure SQL, Google Cloud SQL

02

Infrastructure-as-Code (IaC)

Terraform, Ansible for database management

03

DevOps for Databases

CI/CD pipelines for database changes

04

Serverless Databases

Amazon Aurora Serverless and beyond

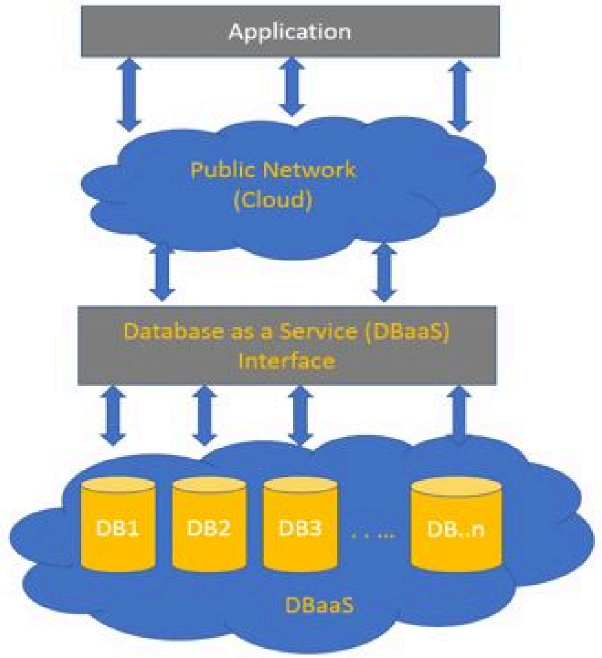
Database-as-a-Service (DBaaS) - Introduction

What is DBaaS?

A cloud computing service that provides database functionality without requiring physical hardware installation or software configuration

Key Benefits

- ✓ Reduced operational overhead
- ✓ Built-in high availability
- ✓ Pay-as-you-go pricing
- ✓ Automatic scaling
- ✓ Automated backups
- ✓ Faster deployment



Cloudbase Database Architecture (DBaaS)

DBaaS Providers: AWS RDS, Azure SQL, Google Cloud SQL

AWS RDS

Key Features

- Automated backups & patching
- Multi-AZ deployments
- Read replicas for scaling
- VPC integration

Supported Engines

MySQL · PostgreSQL · MariaDB · Oracle · SQL Server · Aurora

Unique Advantage

Deep integration with AWS ecosystem and Aurora for high performance

Azure SQL

Key Features

- Built-in intelligence
- Advanced threat protection
- Automatic tuning
- Hyperscale tier

Supported Engines

SQL Server · MySQL · PostgreSQL · MariaDB

Unique Advantage

Seamless integration with Microsoft ecosystem and advanced security features

Google Cloud SQL

Key Features

- Automated failover
- Point-in-time recovery
- Easy scaling
- Data encryption

Supported Engines

MySQL · PostgreSQL · SQL Server

Unique Advantage

Integration with Google's data analytics and machine learning tools

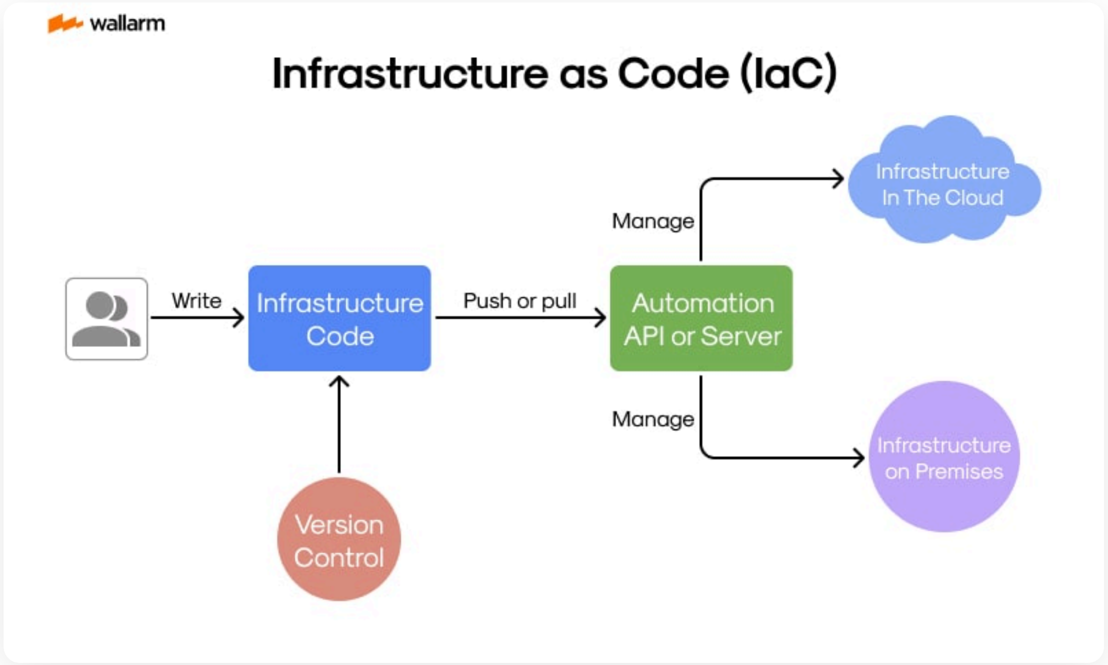
Infrastructure-as-Code (IaC) for Databases - Introduction

What is IaC for Databases?

Managing database infrastructure through machine-readable definition files rather than manual configuration

Key Benefits

- ✓ Consistency across environments
- ✓ Version control for database changes
- ✓ Automated provisioning
- ✓ Reduced configuration drift
- ✓ Reproducible environments
- ✓ Faster deployment cycles



IaC Tools: Terraform and Ansible for Databases

Terraform

Declarative infrastructure provisioning tool that manages database resources through configuration files

Key Strengths

- Declarative syntax (HCL)
- State management
- Multi-cloud support
- Immutable infrastructure

Database Use Case: Provisioning cloud database instances (RDS, Azure SQL, Cloud SQL)

Ansible

Configuration management tool that automates database setup and configuration tasks

Key Strengths

- Agentless architecture
- Idempotent operations
- YAML-based playbooks
- Day-2 operations focus

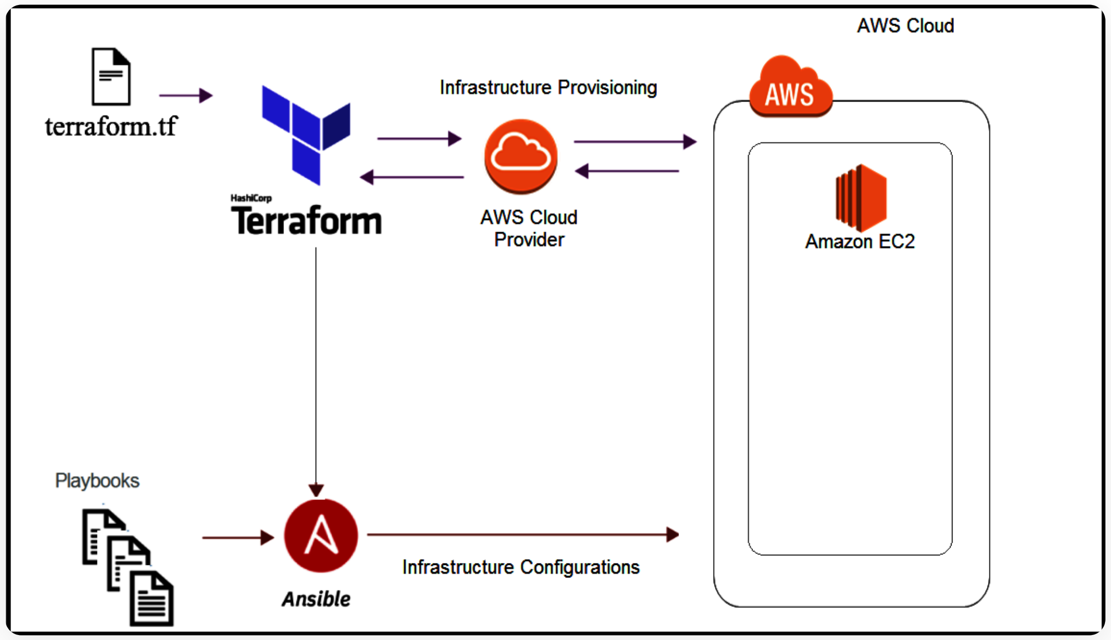
Database Use Case: Configuring database users, permissions, and schema management

Key Differences

Aspect	Terraform	Ansible
Primary Focus	Provisioning	Configuration
Approach	Declarative	Procedural
State Management	Built-in state file	No state tracking
Execution	Client-server model	Agentless (SSH/WinRM)
Learning Curve	Moderate	Easier

Combined Workflow

1. Use Terraform to provision database infrastructure
2. Use Ansible to configure databases post-provisioning
3. Integrate both tools in CI/CD pipelines



DevOps for Databases - Introduction

What is DevOps for Databases?

Applying DevOps principles and practices to database management, enabling faster, safer, and more reliable database changes

Key Benefits

- Faster deployments
- Better collaboration
- Automated testing
- Reduced errors
- Version control
- Consistent environments

Why Continuous Delivery for Databases Matters

Databases are often the bottleneck in application delivery. Implementing CD for databases ensures that both application and database code evolve together, eliminating delays and reducing risk

Challenges Addressed

- Manual database deployments causing delays
- Schema drift between environments
- Lack of version control for database changes
- Risk of data loss during deployments
- Slow feedback loops for database changes

DevOps Solutions

- Database schema versioning
- Automated testing of database changes
- CI/CD pipelines for database deployments
- Environment-specific configuration management
- Rollback mechanisms for database changes

CI/CD Pipelines for Databases

Key Components

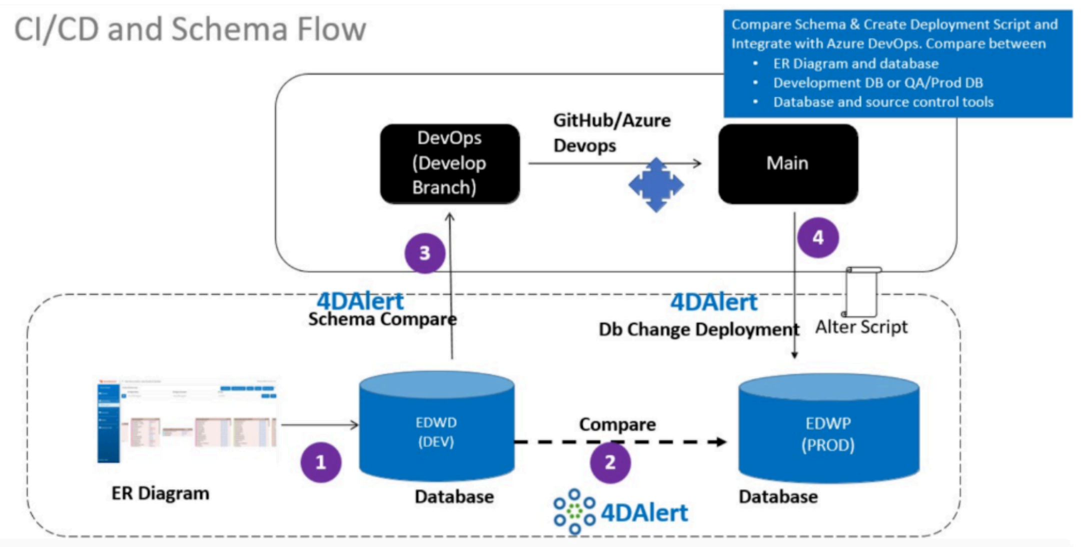
- Version Control
- Testing Framework
- Monitoring
- Build Automation
- Deployment Tools
- Rollback Mechanism

5 Reasons for CI/CD Database

- Align database and application change speed
- Enable digital transformation
- Enable self-service for development
- Enhance data security
- Maximize ROI of application release automation

Best Practices

- Store database changes in version control
- Use environment-specific configurations
- Implement automated testing for all changes
- Create idempotent deployment scripts
- Establish clear rollback procedures
- Monitor database performance post-deployment


CI/CD and Schema Flow
Compare Schema & Create Deployment Script and Integrate with Azure DevOps. Compare between
• ER Diagram and database
• Development DB or QA/Prod DB
• Database and source control tools
DevOps (Develop Branch)
GitHub/Azure Devops
Main
3
4
4DAlert Schema Compare
4DAlert Db Change Deployment
Alter Script
ER Diagram
1
EDWD (DEV)
Database
Compare
2
EDWP (PROD)
Database
4DAlert

Serverless Databases - Introduction

What are Serverless Databases?

Cloud databases that automatically scale compute capacity up and down based on application demand, eliminating the need for manual capacity management

Key Benefits

- ✓ Automatic scaling
- ✓ Pay-per-use pricing
- ✓ Reduced operational overhead
- ✓ High availability
- ✓ No capacity planning
- ✓ Faster deployment

Traditional vs. Serverless

Aspect	Traditional	Serverless
Capacity	Fixed provisioning	Auto-scaling
Pricing	Fixed hourly rate	Pay per ACU/second
Management	Manual scaling	Automated
Scaling Time	Minutes	Seconds

When to Use Serverless Databases

- Applications with unpredictable or spiky workloads
- Intermittent or periodic usage patterns
- Development and testing environments
- Cost-sensitive applications with variable traffic
- Rapid prototyping and new projects

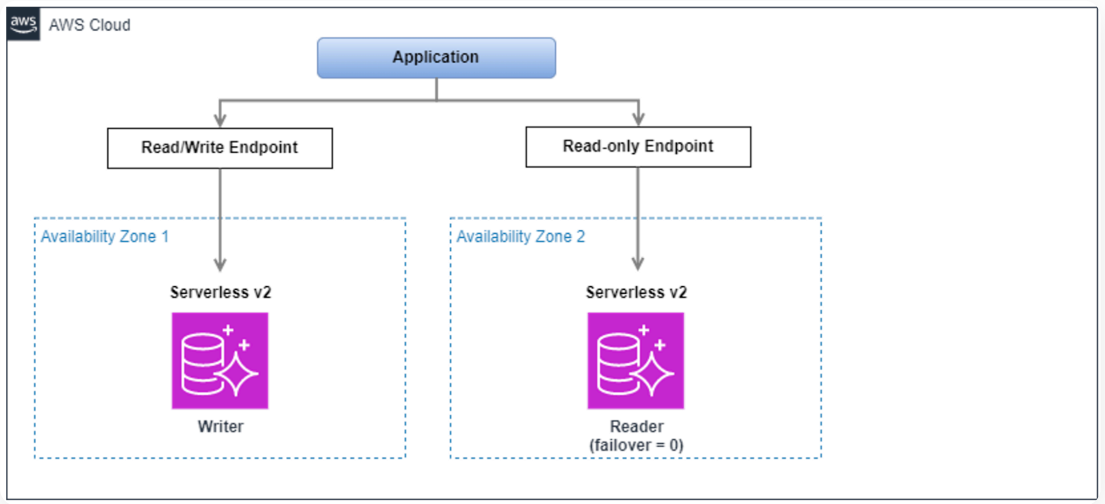
Amazon Aurora Serverless

Key Features of Aurora Serverless v2

- Simpler capacity management
- Faster scaling (seconds)
- Horizontal scaling
- Elastic instance scaling
- Reader DB instances
- Multi-AZ clusters
- Global databases
- Enhanced security features

Use Cases

- Applications with unpredictable traffic spikes
- Development and test environments
- Variable workloads with infrequent access
- New applications with uncertain requirements
- Workloads requiring rapid scaling



Cost Benefits

ACUs scale in 0.5 increments, reducing over-provisioning. Pay only for what you use with per-second billing.

Comparison and Use Cases

Approach	Complexity	Cost	Scalability	Management Overhead
DBaaS	★★☆☆☆	★★★☆☆	★★★★☆	★☆☆☆☆
IaC	★★★☆☆	★★☆☆☆	★★★☆☆	★★☆☆☆
DevOps	★★★★☆	★★☆☆☆	★★★★☆	★★★☆☆
Serverless	★★☆☆☆	★★★★☆	★★★★★	★☆☆☆☆

Ideal Use Cases

DBaaS

- ✓ Standard applications with predictable workloads
- ✓ Teams with limited database expertise
- ✓ Projects requiring quick deployment

IaC

- ✓ Multi-environment deployments
- ✓ Teams practicing GitOps
- ✓ Infrastructure requiring version control

DevOps

- ✓ Applications with frequent schema changes
- ✓ Teams with rapid release cycles
- ✓ Organizations prioritizing collaboration

Serverless

- ✓ Applications with unpredictable traffic
- ✓ Development and testing environments
- ✓ Cost-sensitive projects with variable usage

Conclusion and Q&A

Key Takeaways

DBaaS

- Reduced operational overhead
- Automatic scaling and backups
- Multiple provider options

IaC

- Consistent environments
- Version control for infrastructure
- Terraform for provisioning, Ansible for configuration

DevOps

- Faster, safer database changes
- CI/CD pipelines for databases
- Version control for database changes

Serverless

- Automatic scaling in seconds
- Pay-per-use pricing model
- Ideal for unpredictable workloads

Why Modern Database Management Matters

Modern database approaches enable organizations to reduce costs, improve agility, enhance reliability, and accelerate innovation in today's fast-paced digital landscape

Thank You

contact@example.com www.example.com/resources