Amazon Aurora High Availability and Disaster Recovery Features for Global Resilience

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers, or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

© 2024 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Table of Contents

Abstract and introduction

Abstract

Amazon Aurora is a fully managed relational database designed for unparalleled high performance and availability at global scale with full MySQL and PostgreSQL compatibility. Aurora provides managed high availability (HA) and disaster recovery (DR) capabilities in and across AWS Regions. In this whitepaper, explore Aurora HA and DR capabilities and discover design patterns that enable the development of globally resilient applications. Learn how to establish single Region and cross-Region HA and DR using Aurora features, including Multi-AZ deployments and Amazon Aurora Global Database.

Are you well-architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the AWS Well-Architected Tool, available at no charge in the AWS Management Console, you can review your workloads against these best practices by answering a set of questions for each pillar.

In Disaster Recovery of Workloads on AWS: Recovery in the Cloud, we describe a collection of customer-proven best practices for designing well-architected DR workloads.

For more expert guidance and best practices for your cloud architecture—reference architecture deployments, diagrams, and whitepapers—refer to the AWS Architecture Center.

Introduction

Aurora is a relational database management system (RDBMS) built for the cloud with full MySQL and PostgreSQL compatibility. Aurora gives you the performance and availability of commercial-grade databases at one-tenth the cost. Aurora is a fully managed database that automates aspects of database management, such as high availability (HA), disaster recovery (DR), replication, scaling, backup and restore, and monitoring. In this paper we will discuss the HA and DR capabilities of Aurora, and how you can take advantage of common architectural patterns to achieve single-Region and multi-Region HA and DR.

NOTE: All features, functionality, and architecture patterns in this whitepaper apply to both Aurora MySQL and Aurora PostgreSQL, unless otherwise specified.

Before we start exploring the HA and DR features of Aurora, let's understand what high availability and disaster recovery mean.

High availability

Availability is a commonly used metric to quantitatively measure resiliency. A workload's availability is measured as the percentage of time it's available for use. The percentage is calculated over a period of time such as a month or a year (available for use time / total time), for example, 99.99% (four nines).

A highly available database is able to ensure an agreed level of operational performance in the event of issues such as a hardware, software, or network failure, with little to no manual intervention. HA is traditionally achieved by creating a replica of the primary database on an isolated piece of hardware separate from the source database.[1] In an event of a disruption, the replica is promoted and assumes the role of a new primary database. Application connectivity can be managed by using methods such as a virtual IP (VIP), Domain Name System (DNS) redirection, or a proxy layer.[2] Detection of a disruption can be achieved by monitoring the health of the primary database using a combination of methodologies like a quorum voting system and a heartbeat.[3]

Disaster recovery

HA and DR are completely separate, yet equally important, facets of a highly resilient database architecture. DR is an organization's method of regaining access and functionality of its IT infrastructure after a natural or human disaster. DR policies can require manual intervention, such as running scripts, changing endpoints, and resizing infrastructure.

DR usually involves more than just the database layer. For example, after a large natural disaster, an entire data center might become unavailable. In such situations, DR procedures can be executed to restore the database and applications to continue operations, for example, in a different, unaffected AWS Region. DR procedures typically include a robust backup strategy. Backups allow databases to recover to a specific point in time, before the disaster occurred.

When designing DR procedures, two important factors to consider are Recovery Time Objective (RTO) and Recovery Point Objective (RPO). RTO and RPO are driven by business requirements for a specific application and its underlying database. Different applications and workloads within organizations, and even in the same department, can have differing RTO and RPOs.

RPO is the maximum acceptable amount of time since the last data recovery point. This determines what is considered an acceptable loss of data between the last recovery point and the interruption of the database. For example, if you define an RPO of 15 minutes, in the event of disaster you could lose up to 15 minutes of data, but not more.

RTO is the maximum acceptable delay between the interruption of the database and restoration of the service. This determines what is considered an acceptable time window when the database is unavailable. For example, if you determine that your application's RTO is 5 minutes, your DR strategy should allow your application (including the database and other application components) to return to service in no more than 5 minutes.

1 HA is a characteristic of a system that aims to ensure an agreed level of operational performance, usually uptime, for a higher than normal period.

2 A proxy, such as an Amazon RDS Proxy is an intermediary service that allows your applications to pool and share database connections to improve the applications' ability to scale. With a proxy service, you can handle unpredictable surges in database traffic and create connections at a fast rate, avoiding oversubscribing connections. It also reduces failover times in an HA configuration by removing dependency on DNS.

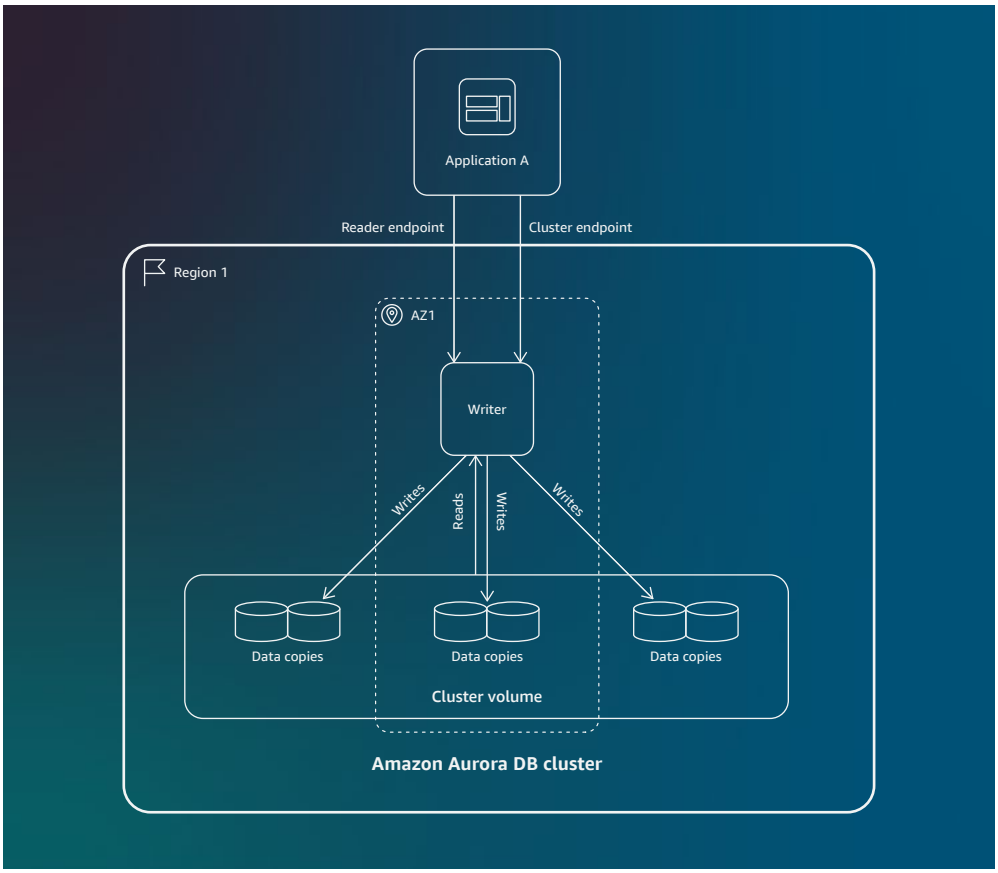
3 A quorum voting system is used by distributed systems to enforce consistent operations. A well-formed quorum obtains minimum votes, to determine whether a transaction can be allowed or not. A heartbeat is a signal generated by a system at predetermined intervals to let its partner know it's working normally. Heartbeat is a commonly used technique for synchronization in highly available systems.

Aurora architecture and HA and DR features

Distributed storage

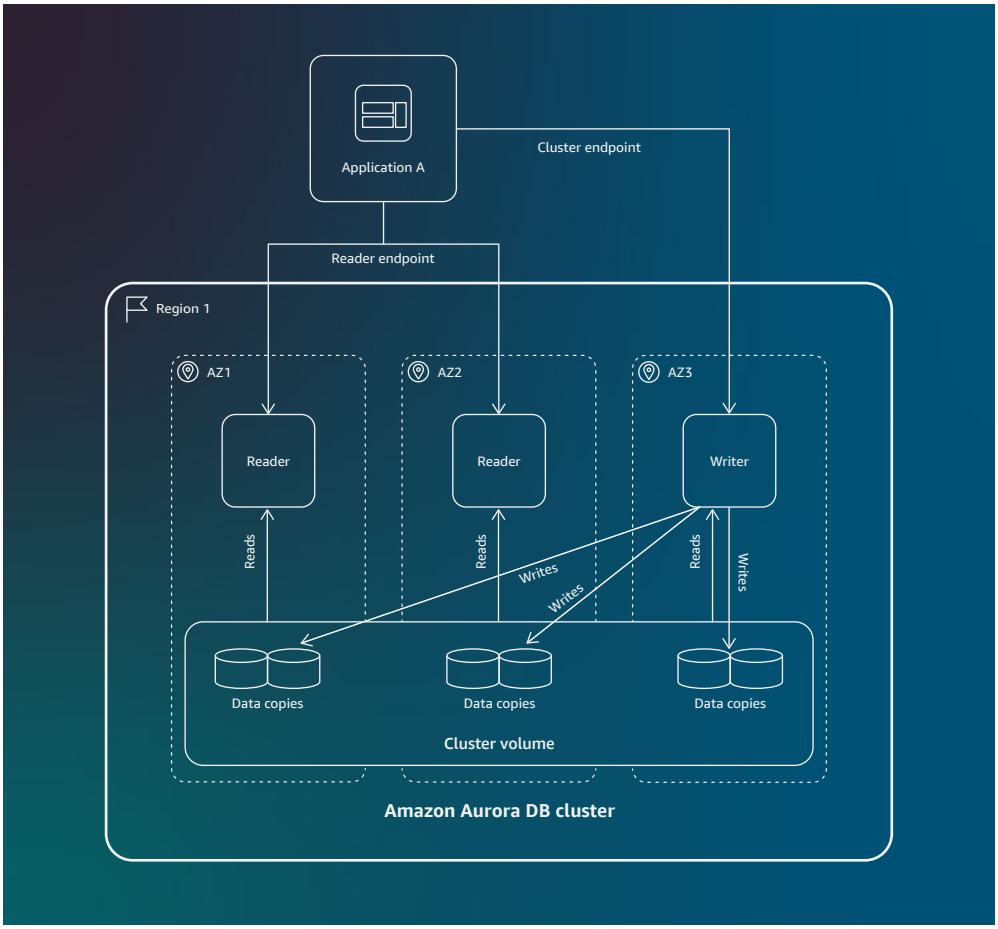
The architecture of Aurora is built from the ground up to be highly available and resilient to failures. The storage subsystem in Aurora is distributed and purpose built for use with Aurora. Aurora replicates new writes six ways across three Availability Zones. The distributed storage ensures your data can survive the rare occurrence of a full Availability Zone failure plus an additional concurrent storage node failure in a different Availability Zone (AZ+1 failure). This distributed storage architecture is also able to automatically scale and self-heal from issues such as node failures and fill in missing writes by using a peer-to-peer protocol between storage nodes.

It is worth noting that while the Aurora distributed storage subsystem provides enhanced durability, it does not make your database highly available by itself. HA options for an Aurora database (DB) cluster are discussed later in this paper.



Aurora Single-AZ architecture example

The architecture of Aurora decouples compute resources from its storage, which allows compute and storage subsystems to independently recover from failures. In a single Region, an Aurora DB cluster can be deployed as a Single-AZ (SAZ) configuration, or a Multi-AZ (MAZ) configuration. A SAZ Aurora DB cluster is made up of a single writer instance, which accepts both read and write requests.



Aurora Multi-AZ architecture example

A MAZ Aurora DB cluster is made up of a writer DB instance and at least one (and up to 15) reader DB instances. A reader DB instance acts as a low-lag read replica and can only accept read requests. A MAZ Aurora DB cluster configuration is a fully managed, single Region, HA option. A MAZ Aurora DB cluster configuration requires a writer DB instance and one or more reader DB instances in a different Availability Zone than the writer DB instance. When deployed as a MAZ configuration, Aurora offers a 99.99% (four nines) uptime SLA. An Aurora DB cluster provides a cluster (or writer) endpoint that always connects to the current writer DB instance and accepts both read and write requests. Aurora DB clusters also provide a reader endpoint that connects to the reader DB instance. If there is more than one reader DB instance, Aurora acts as a load balancer for all available reader DB instances.

When a MAZ architecture is used, Aurora automatically detects disruption in the writer DB instance and fails over to one of the designated failover targets. A failover target can be one of the reader DB instances in the DB cluster. If you have more than one reader DB instance, a priority order can be assigned to reader DB instances using a configurable parameter value (0 to 15). The reader DB instance with the highest priority (value 0) is chosen as the first failover target. After a successful failover, applications that try to reconnect using the writer endpoint are automatically redirected to the newly promoted writer DB instance. Hence, applications don't have to make any changes to reconnect to the database after a failover. The failover can take up to 60 seconds to complete, and any requests submitted by the application during and before failover will fail, so the application will have to retry those requests. The failover times can be further improved by using Amazon Relational Database Service (Amazon RDS) Proxy, which can automatically connect to the new DB instance while preserving application connections. When failovers occur, Amazon RDS Proxy directly routes requests to the new DB instance. This reduces failover times for Aurora databases by up to 66%.

Aurora also supports use of an advanced Java database connectivity (JDBC) wrapper in the AWS JDBC Driver. The wrapper is complementary to existing open source JDBC. It aims to extend the functionality of the driver to enable applications to take full advantage of the features of clustered databases such as Aurora. The AWS JDBC Driver does not directly connect to any database, but it enables support of AWS and Aurora functionality on top of an underlying JDBC driver of the user's choice, such as PostgreSQL JDBC Driver or MySQL JDBC Driver. The AWS JDBC Driver is designed to understand when a failover occurs and coordinate with the Aurora cluster to provide minimal downtime and allow connections to be very quickly restored in the event of a DB instance failure.

Aurora offers fully managed backups. You can configure Aurora to automatically back up your Aurora DB cluster by configuring a backup retention period between 1 day and 35 days. Once configured, Aurora automatically and continuously backs up your DB cluster. If you want to retain data beyond the backup retention period, you can take a snapshot of the data in your Aurora cluster volume. Note that Aurora DB cluster snapshots don't expire and you must take care to delete them if they are no longer needed. You can restore an Aurora database to any time during this configured backup retention period using a point-in-time restore (PITR). You can also use AWS Backup to manage backups of Aurora DB clusters.

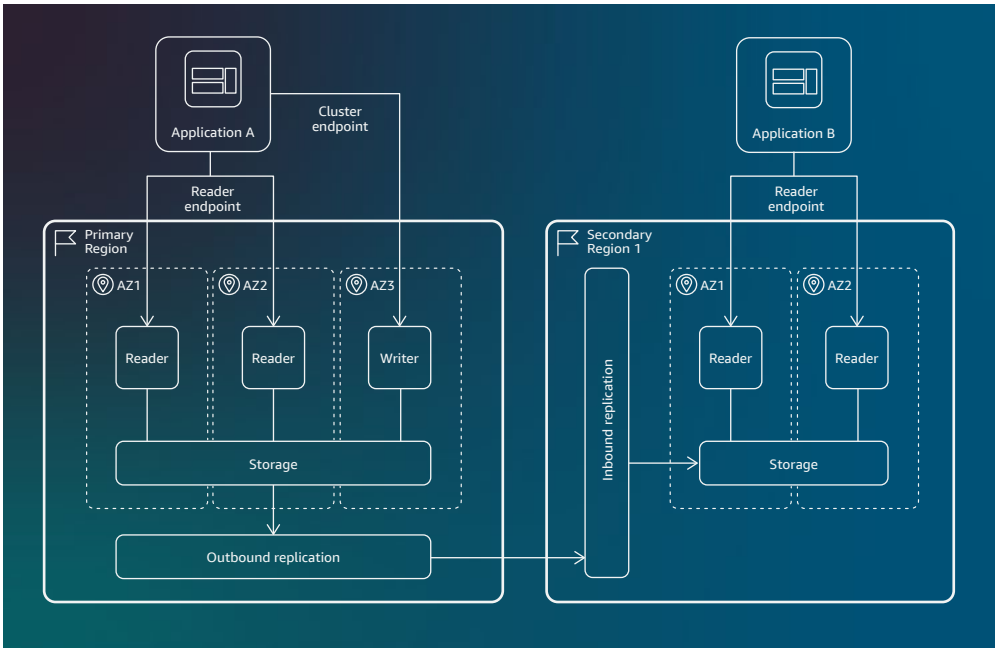
Aurora Global Database

Aurora also offers Amazon Aurora Global Database, which allows an Aurora DB cluster to span multiple Regions. Aurora Global Database asynchronously replicates your data, with a typical latency of less than 1 second, while leaving your database fully available to serve application workload. An Aurora Global Database can be set up across as many as five secondary Regions. Each replicated secondary Region can have up to 15 reader DB instances of their own. This architecture provides a massive read scale of up to five secondary Regions and up to 90 reader DB instances. Aurora Global Database enables fast local reads with low latency in each Region, and provides DR from Region-wide outages. If your primary Region suffers an outage, you can promote one of the secondary Regions to take read/write responsibilities. An Aurora DB cluster can typically recover in a minute, even in the event of a complete Regional outage. This typically provides your application with an effective RPO of 1 second and RTO of 1 minute, providing a strong foundation for global business continuity for your Aurora DB cluster.

An Aurora Global Database gives you the ability to quickly plan for and recover from a Regional outage. There are two different approaches to failover depending on the scenario: Aurora Global Database Switchover and Aurora Global Database Failover. A Global Database Switchover requires all participating DB clusters across Regions to be available. You can invoke a Global Database Switchover to switch primary and secondary cluster roles. Common use cases are cross-Region DR testing to meet compliance and carry out operational maintenance. Using the Global Database Switchover feature, you can switch over to one of the secondary Regions in a few quick steps using the `SwitchoverGlobalCluster` API or the `switchover-global-cluster` CLI operation. Note that this feature automatically reverses the replication flow after a switchover to a secondary Region. The Global Database Switchover feature also allows for a switchover to the original primary Region. Additionally, Global Database Switchover can be used for use cases such as regional rotations for a follow-the-sun operational model.

Global Database Failover is a cross-Region database failover process that can be initiated from one of the secondary Regions in the case of an outage, such as a Regional or a service outage. You can start Global Database Failover using the console, the `FailoverGlobalCluster` API, or the `failover-global-cluster` CLI operation and specifying the AllowDataLoss parameter. Global Database Failover promotes the chosen secondary Region DB cluster to a primary and reinitializes all available secondary Regions in the Aurora Global Database topology using a snapshot of the new primary Region DB cluster. When the old primary Region recovers from the outage, Aurora adds this Region back to the Aurora Global Database by restoring a snapshot of the current primary Region DB cluster. Additionally, Aurora takes a snapshot to preserve pre-failover data. Aurora Global Database replication is asynchronous, so a Global Database Failover has the potential of losing data that was not replicated to the secondary Region at the time of failover. For a detailed discussion on Global Database Failover and Global Database Switchover, please consult the Aurora user guide.

Aurora PostgreSQL Global Database offers a managed RPO mechanism that lets you plan and enforce your RPO for an Aurora Global Database configuration. Aurora Global Database also offers a write-forwarding feature, which can be used to forward writes from a secondary Region to the primary Region.



Aurora Global Database architecture example

Aurora also offers managed Blue/Green Deployments that can be used to reduce downtime for disruptive operations such as major and minor version database upgrades, testing new database and application features, and schema maintenance or changes. Aurora offers a zero-downtime patching (ZDP) feature that can greatly reduce application downtime during minor version patching. ZDP attempts, on a best-effort basis, to preserve client connections through Aurora minor version upgrades. If ZDP completes successfully, application sessions are preserved and the database engine restarts while the upgrade is in progress. The database engine restart can cause a drop in throughput, lasting for a few seconds to approximately 1 minute.

Achieving HA and DR in a single Region

Aurora provides fully managed automatic backups, enabling a single Region DR strategy aligned with business and compliance requirements. For long-term retention needs, options include using AWS Backup for manual snapshots with centralized policy management or exporting DB cluster snapshot data to Amazon S3 buckets: a background process that does not impact active cluster performance.

To architect Aurora for HA within a single Region, deploy a MAZ DB cluster configuration. MAZ DB cluster configuration comprises one writer DB instance and at least one reader DB instance provisioned across separate Availability Zones to provide failover redundancy, offering a 99.99% (four nines) uptime SLA. The MAZ configuration automatically detects and mitigates failures, such as those affecting the writer instance, by failing over to a designated reader and promoting it as the new primary. Applications can seamlessly connect without reconfiguration to the newly promoted instance using the cluster and reader endpoints. In case of a DB instance failure, the underlying instance is automatically replaced after failover in a MAZ setup, whereas a SAZ configuration can experience several minutes of downtime while a new instance becomes available.

Extending HA and DR across multiple Regions

A common cross-Region DR pattern with higher RTO and RPO tolerance involves configuring snapshot backups in secondary Regions. These backups, immune to primary Region disruptions, enable recovery strategies in the event of a primary Region failure by deploying the secondary Region backups.

Aurora Global Database offers a more robust business continuity and DR solution compared to the standard HA provisions of a single Region Aurora DB cluster deployment. The decoupled architecture design of Aurora allows a single DB cluster to span multiple Regions while facilitating low-latency local reads, providing resilience against Region-wide outages and making Aurora an ideal multi-Region solution for extending HA and DR strategies.

Aurora Global Database facilitates quick RTO in the order of minutes through its Global Database Failover capability, enabling failover to a secondary Region in case of primary Region failure. For scenarios like Regional rotations, follow-the-sun applications, or DR drills, the Global Database Switchover option can be used when both primary and secondary Regions are available and operational.

Finally, Aurora Global Database supports a headless configuration for secondary Regions, wherein the secondary cluster contains only the Aurora storage volume without any DB instances. In addition to serving as a cost-control measure, it allows you to secure backups against primary Region failures as part of your DR strategy. Prior to promotion, you can attach a DB instance to the secondary Region cluster. As an alternative, you have the option to provision an Aurora Serverless v2 instance in the secondary Region, which offers a cost-effective deployment solution. We recommend weighing trade-offs between RTO and cost savings if you are considering a headless configuration approach.

Monitoring your HA and DR environment

Aurora offers various observability tools, including Amazon CloudWatch Logs, enhanced monitoring, and Amazon RDS Performance Insights, to monitor the health, availability, and performance of DB clusters. Key CloudWatch metrics to monitor single Region Aurora DB clusters include:

`AuroraReplicaLag`
`CPUUtilization`
`DatabaseConnections`
`NetworkThroughput`
`NetworkTransmitThroughput`
`NetworkReceiveThroughput`
`StorageNetworkThroughput`
`StorageNetworkTransmitThroughput`
`StorageNetworkReceiveThroughput`

See Metrics Reference for Aurora and Monitoring Tools for additional metrics and tools for monitoring your Aurora DB clusters. Key CloudWatch metrics to monitor your Aurora Global Databases across Regions include:

`AuroraGlobalDBDataTransferBytes`
`AuroraGlobalDBProgressLag`
`AuroraGlobalDBReplicatedWriteIO`
`AuroraGlobalDBReplicationLag`
`AuroraGlobalDBRPOLag`

NOTE: The AuroraGlobalDBRPOLag is applicable only to user transactions. The AuroraGlobalDBProgressLag also includes health check transactions, so when applications are idle, you can still see some lag from health checks, which can be helpful to diagnose network issues if there is low or no user activity.

In addition, for Aurora PostgreSQL–based Global Databases, you can use two functions:

`aurora_global_db_status`
Shows the lag times of the Global Database secondary DB clusters.

`aurora_global_db_instance_status`
Lists all secondary DB instances for both the primary DB cluster and secondary DB clusters.

See Monitoring Aurora PostgreSQL–based Aurora Global Databases to learn more about how to use these functions.

Monitoring Aurora events

The generation of an Amazon RDS event indicates a change in the Aurora environment. For example, Aurora generates an event when a DB cluster is patched. Aurora delivers events to Amazon CloudWatch Events and Amazon EventBridge in near real time. Amazon RDS groups events into categories that you can subscribe to and be notified from when an event in that category occurs. See Working with Amazon RDS Event Notification in the Aurora user guide for more details.

Best practices

Define RTO and RPO

DR strategy is driven by well-defined business requirements. The initial step involves defining your RPO and RTO for each specific workload. You can further classify workloads into tiers, with stricter service levels such as lower RTO and RPO for mission-critical tier-one workloads and progressively relaxed constraints for lower tiers, taking into account the associated cost implications. It is crucial to align RTO and RPO targets with business priorities, as more stringent recovery objectives often entail trade-offs such as higher operational costs.

Define HA and DR strategy to align with RTO and RPO

HA strategy:

Create a MAZ Aurora DB cluster to serve as a highly available Aurora deployment in a single Region, backed by the Aurora uptime SLA of 99.99% (four nines). In addition, you can add Aurora readers to serve as failover targets, ready to take over in case the writer instance fails. This failover process is automatic and managed by Aurora.

DR strategy:

Once RTO and RPO are defined, you should establish retention periods for automatic backups that align with defined RTO and RPO. The retention period for automatic backups determines how far back in time you can restore your Aurora DB cluster. By default, Aurora retains automatic backups for 1 day, but you can configure the backup retention period for up to 35 days. The longer the retention period, the more historical data you have available for restoration, which directly impacts the RTO. You might also need longer-term retention of manual snapshots based on your DR strategy. Furthermore, you can achieve an additional layer of resiliency by maintaining backup copies in separate Regions and accounts. AWS Backup streamlines this process, enabling lifecycle management of manual snapshots and centralized backup plan configuration.

Document and test HA and DR procedures

Thoroughly document your HA and DR procedures. HA and DR processes like automated backups, backup window, maintenance window, and failover configuration are documented in the Aurora Database Administrator's Handbook. You can also test the fault tolerance of your Aurora DB clusters by using fault injection queries. However, it is important to create a runbook with all relevant details like location of scripts, what data points to gather, and which procedures to run in which order. These details need to be documented and clearly communicated in the event of a disaster. Once documented, test the procedure by conducting regular DR drills. Update runbooks as necessary.

Regularly test and review HA and DR procedures

Workload profiles change, and that change can impact the effectiveness of your current HA and DR procedures. Implement a process to regularly test your HA and DR procedures to validate their effectiveness and identify any areas of improvement. For example, the database might have grown in size, which makes the backup and recovery take longer than the initial design, and you would need to account for the additional time.

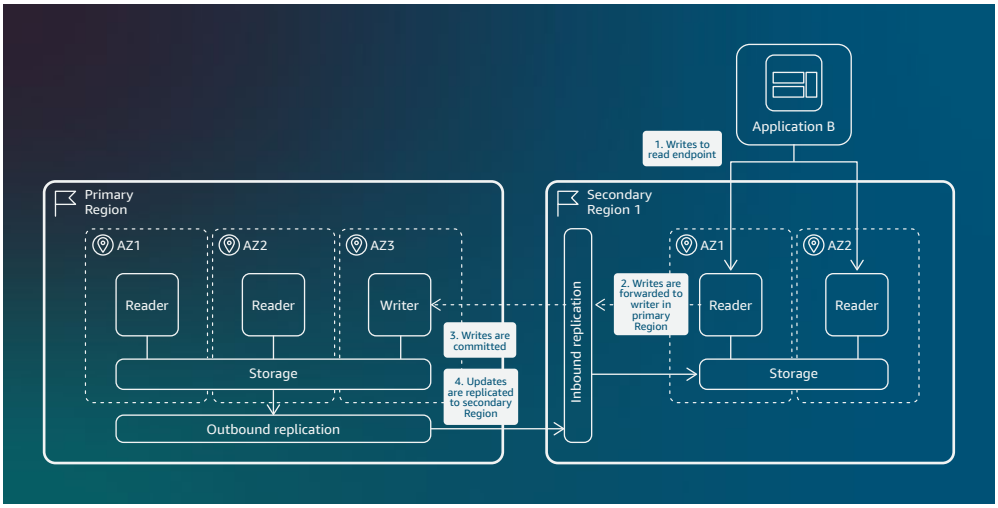
Common HA and DR use cases and design patterns

Use case: your multi-Region applications want read/write capabilities through the DR Region

In addition to serving low-latency reads closer to users across multiple Regions, your applications running in secondary Regions might require write capabilities to the database. As an example, write forwarding can reduce latency for globally distributed applications by allowing users in distant locations to write to a nearby secondary Region reader instance instead of directly writing to the primary Region.

Design pattern: write forwarding with global read replicas

- Use the Aurora Global Database DR reader DB instance for local reads, enhancing performance based on user proximity rather than solely for passive DR.
- Write forwarding enables the application to direct writes to the local reader DB instance. This direct writing transparently handles session and transactional context and enforces consistency between writes and subsequent reads.
- The primary DB cluster serves as the authoritative source, where data changes are initially applied at the storage layer and subsequently replicated to secondary DB clusters within the Aurora Global Database.
- This architecture simplifies application development by allowing writes to be directed to any Aurora Global Database remote DB cluster.



Aurora Global Database write-forwarding example

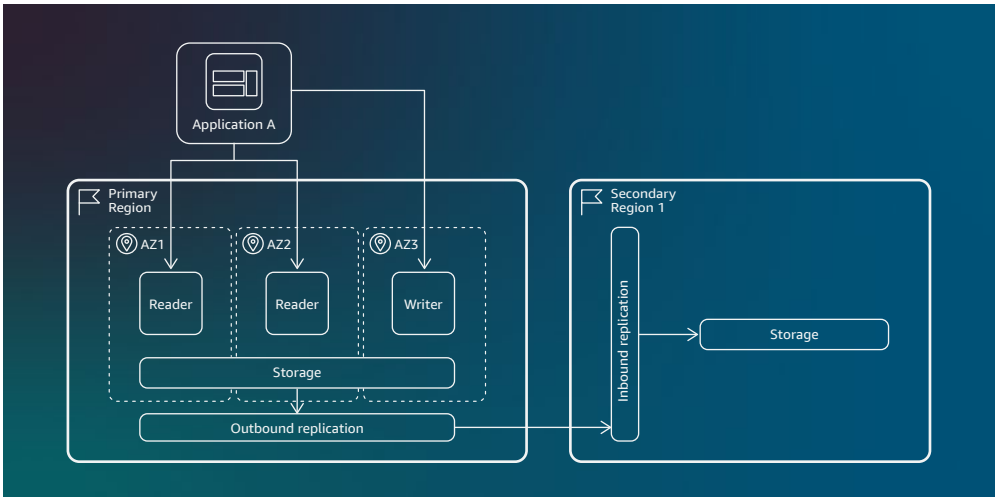
See Using Write Forwarding in an Aurora Global Database in the Aurora user guide for more details.

Use case: you want to save costs on DR

If you are looking for a cost-effective multi-Region resilience strategy with subsecond RPO lag, the Aurora Global Database headless clusters pattern allows for secondary Regions to only contain storage volumes without DB instances. This approach is suitable for DR scenarios with RTO exceeding the timeframe required to provision DB instances in secondary Regions, usually up to 10 minutes.

Design pattern: Aurora Global Database headless clusters

- A headless secondary Aurora Global Database is devoid of any DB instances, in contrast with the primary Region's cluster composition of one writer instance, one or more replicas, and a cluster volume representing the primary data.
- In this configuration, secondary Regions contain only the cluster volume representing the data in the secondary cluster, with Aurora replicating data across Regions using low-latency dedicated infrastructure over the AWS backbone.
- This headless cluster approach can reduce operational costs for an Aurora Global Database, as the decoupled storage and compute architecture removes compute charges for secondary Regions without provisioned DB instances.



Aurora Global Database headless cluster example

See Creating a Headless Aurora DB Cluster in a Secondary Region in the Aurora user guide for more details.

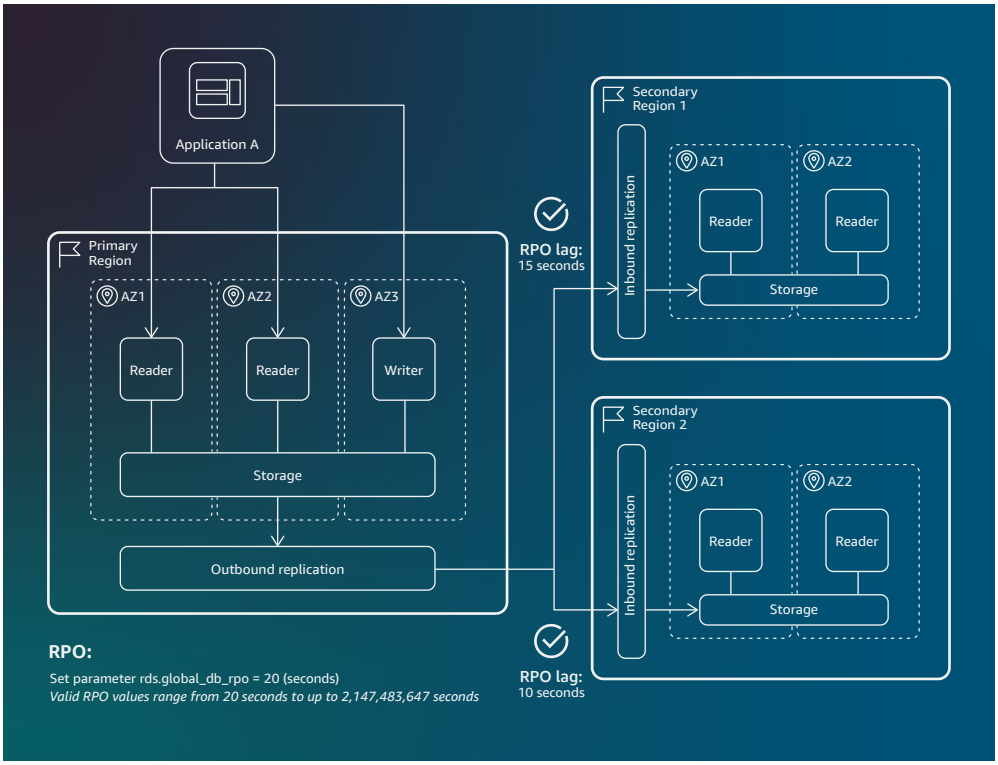
Use case: you want to cap maximum RPO loss

In certain scenarios, such as network- or workload-induced events, replication from primary to secondary clusters can experience delays, potentially leading to an increase in RPO lag. This design pattern addresses the need to mitigate RPO lag escalation in secondary clusters for applications requiring enhanced data protection.

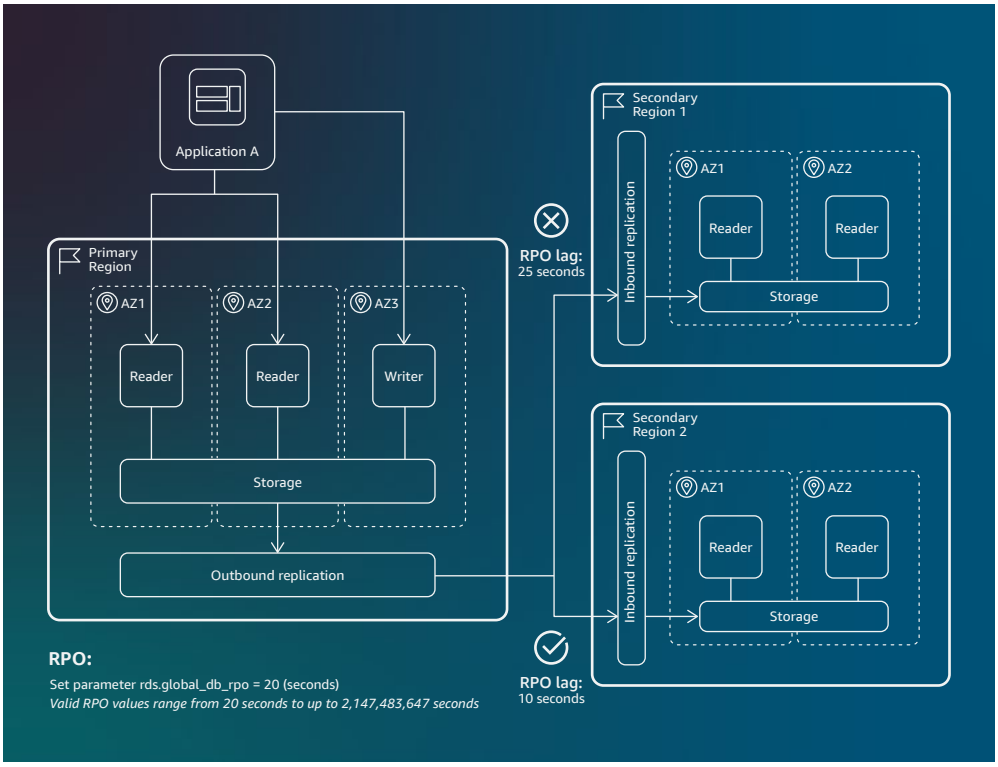
Design pattern: managed RPO

NOTE: This architecture pattern applies only to Aurora PostgreSQL Global Database.

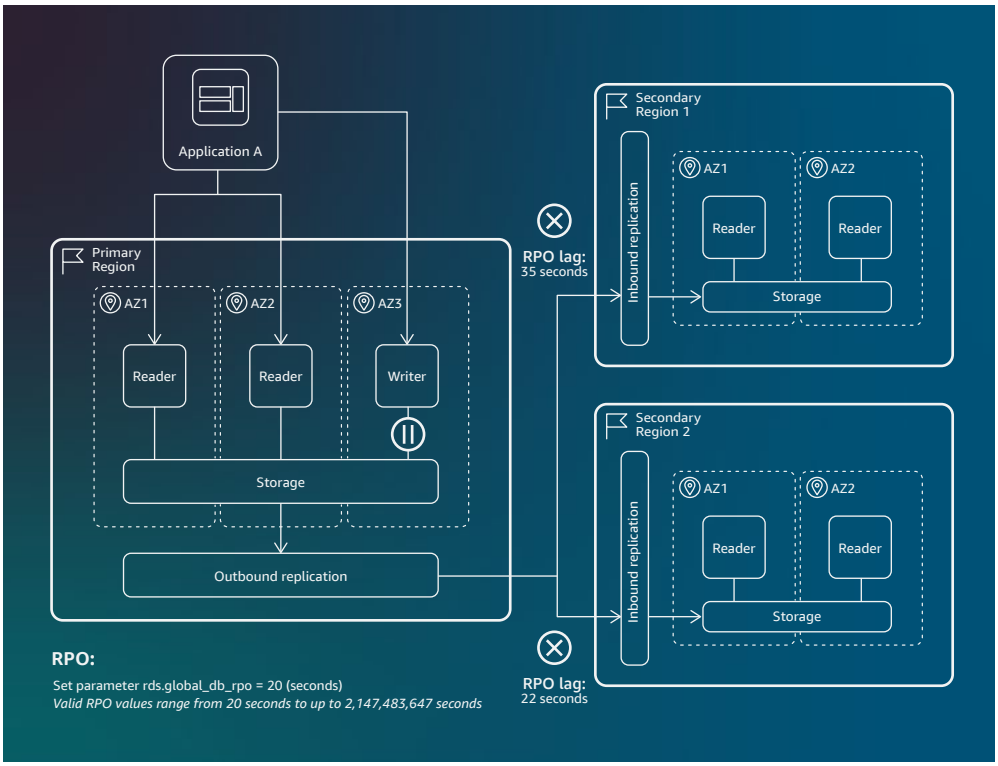
- Aurora PostgreSQL–based Global Database allows RPO management with the `rds.global_db_rpo parameter`.
- Aurora monitors the `AuroraGlobalDBRPOLag` metric, ensuring at least one cluster meets the target RPO window.
- Transactions on the primary cluster are committed if any secondary cluster's RPO lag time is within the target.
- If all secondary clusters exceed the RPO target, transactions are blocked on the primary cluster until one catches up, ensuring RPO compliance.



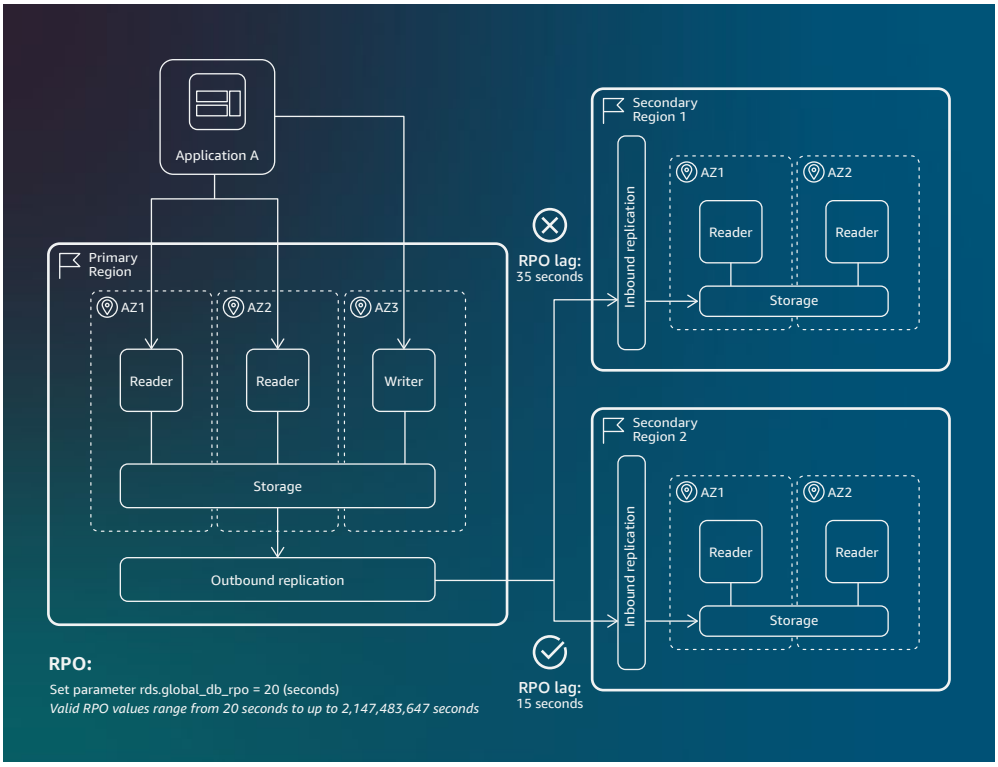
The RPO is set (rds.global_db_rpo= 20 sec). The RPO lag is okay in both secondary Regions.



The RPO lag is still okay in one of the secondary Regions, so write traffic continues



The lag exceeds the RPO in both secondary Regions, so write traffic is paused in the primary Region



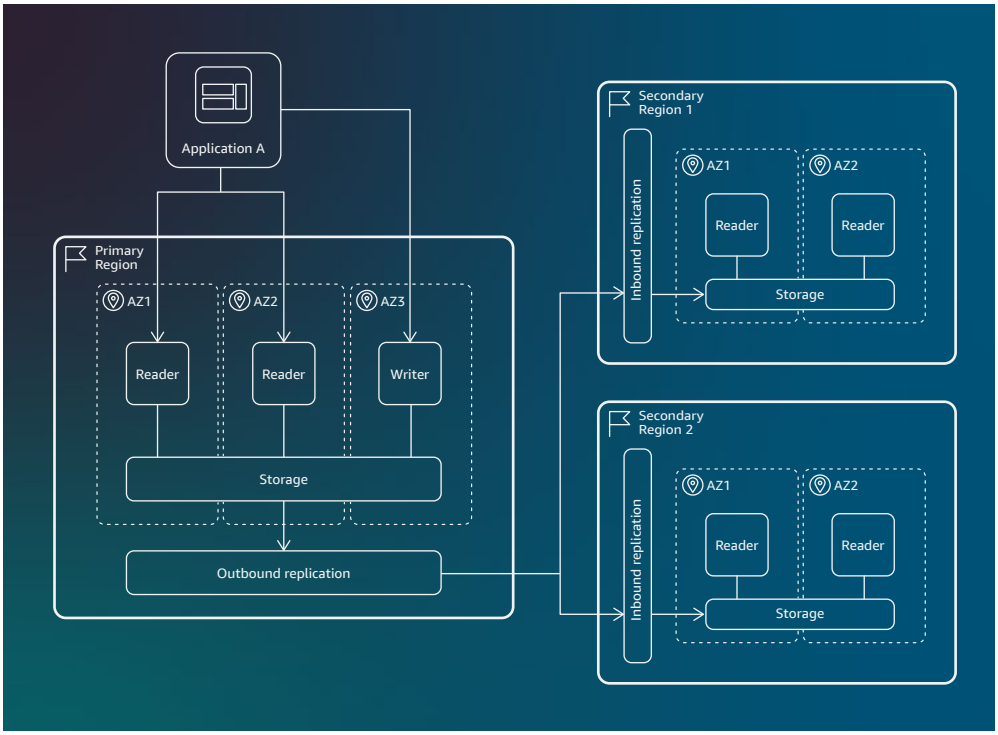
Lag in one of the secondary Regions is back within the limit, so writes are resumed

Use case: you need to meet regulatory compliance requirements for your DR tests

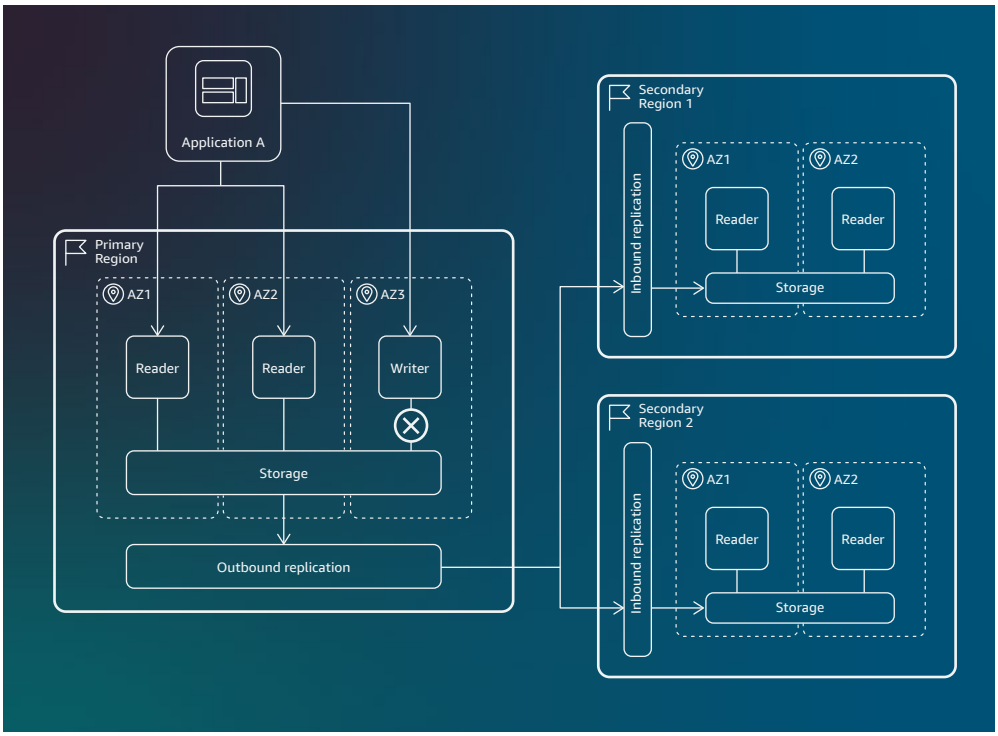
Enterprises often adopt a standard practice of regularly rotating primary systems across Regions. This not only guarantees procedural completeness and accuracy but also ensures staff readiness for DR scenarios. Global Database Switchover supports use cases with DR drills, primary database rotation, or reverting to a previous primary Region without cluster recreation.

Design pattern: Global Database Switchover

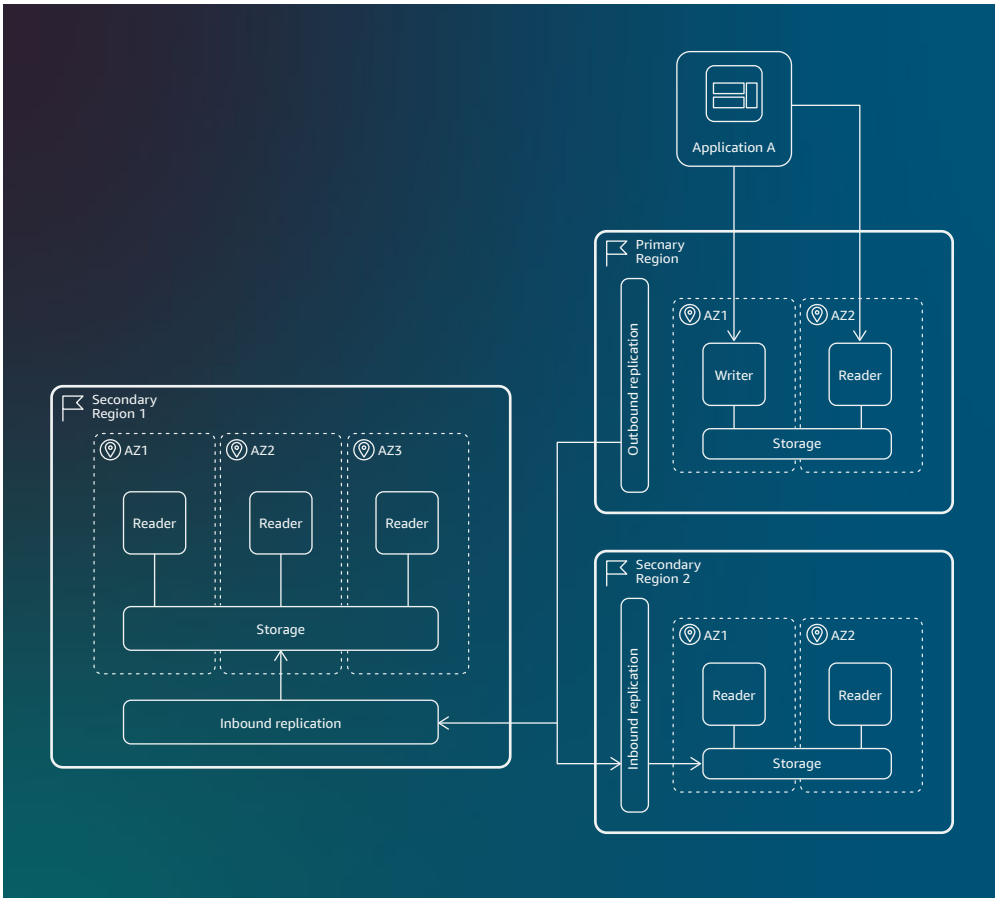
- Global Database Switchover facilitates routine relocation of the primary cluster of an Aurora Global Database to different Regions, suitable for controlled scenarios like operational maintenance and planned procedures.
- For instance, a financial institution with branch offices in various locations might employ this approach to rotate the primary cluster quarterly among designated secondary Regions.
- During the switchover, the primary cluster in the current primary Region transitions to a read-only state, while storage volumes synchronize across the secondary Region to ensure zero data loss (RPO = 0).
- The chosen secondary cluster is promoted to the primary role, the replication topology is maintained, and DB instances across all Regions restart, resulting in temporary unavailability not exceeding single-digit minutes in duration.



Aurora Global Database architecture example with three Regions



Upon switchover, secondary Region 1 is promoted as the new primary Region.
Writes are stopped in the old primary Region while the secondary Regions catch up (RPO=0).



Writes become available in the new primary Region and replication topology is maintained

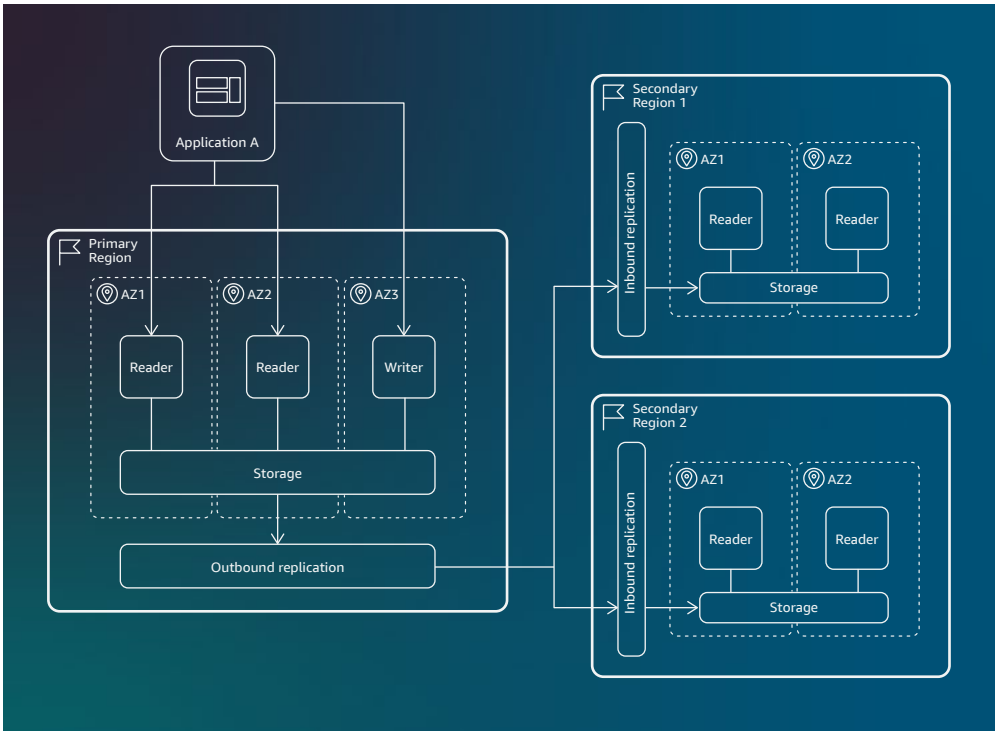
See Performing Global Database Switchover for Aurora Global Databases in the Aurora user guide for more details.

Use case: you need to rapidly recover from a Region failure

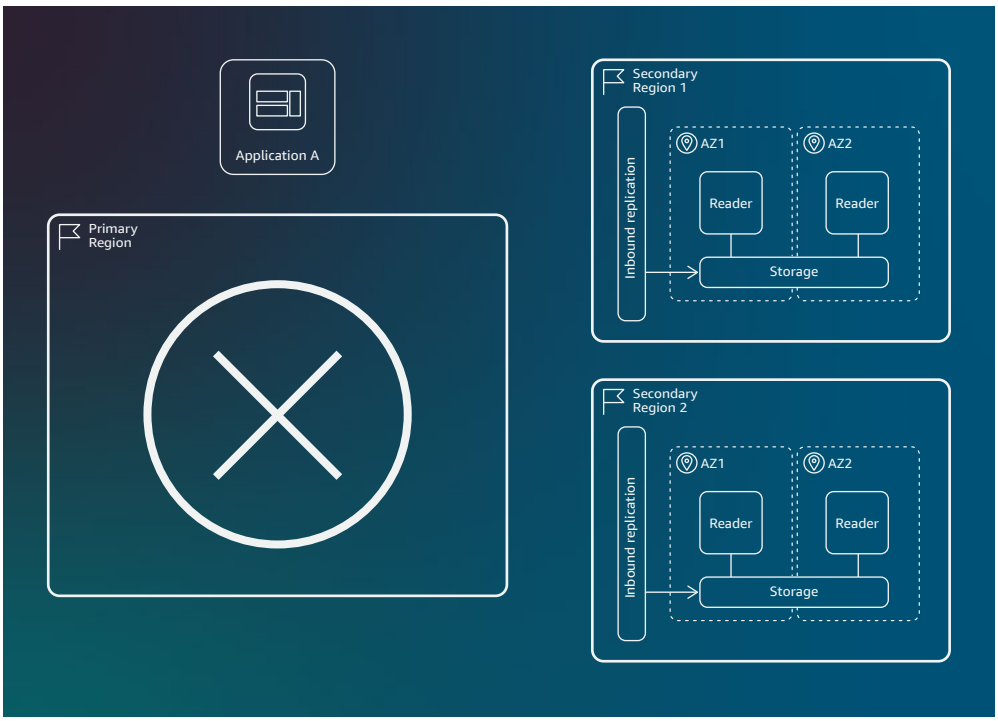
In rare cases, an Aurora Global Database might encounter an unforeseen outage in its primary Region, rendering the primary cluster and its writer DB instance unavailable while stopping replication. In such situations, the Global Database Failover design pattern minimizes downtime and data loss.

Design pattern: Global Database Failover ("Region Disaster" scenarios)

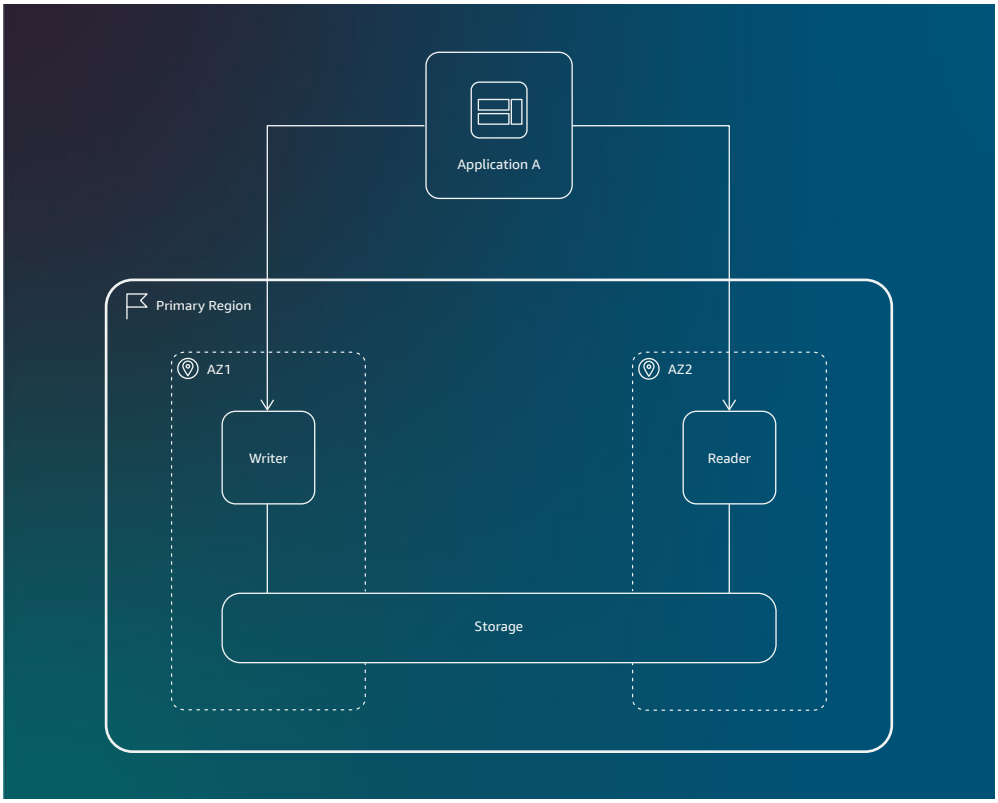
- Take applications offline to prevent writes from being sent to the primary cluster.
- Check lag times for all secondary Aurora DB clusters and choose the secondary Region with the least replication lag (`AuroraGlobalDBRPOLag`); the secondary Region will help minimize data loss with the current failed primary Region.
- Reconfigure the application to direct all write operations to the newly promoted Aurora DB cluster, updating endpoint references accordingly. Redirect write operations in Amazon RDS Proxy if applicable.
- Aurora automatically adds back the old primary Region to the Aurora Global Database as a secondary Region when it becomes available again. Thus, the original topology of your global cluster is maintained. See Performing Managed Failovers for Aurora Global Databases in the Aurora user guide for more details



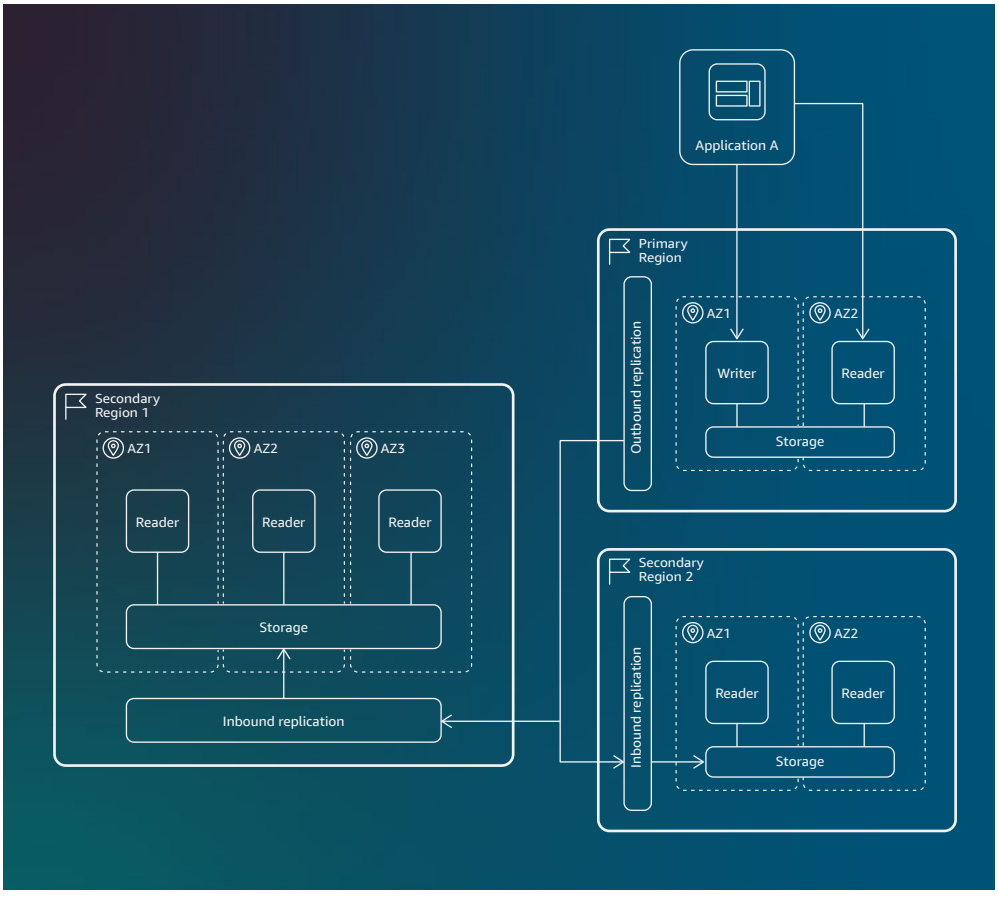
Aurora Global Database architecture example with three Regions



Upon failover, writes are stopped in the primary Region. A secondary cluster in one of the secondary Regions with the least replication lag is identified (secondary Region 1, in this example).



When failover completes, secondary Region 1 gets promoted as the new primary Region. Application A now points to the DB cluster endpoint in the new primary Region.



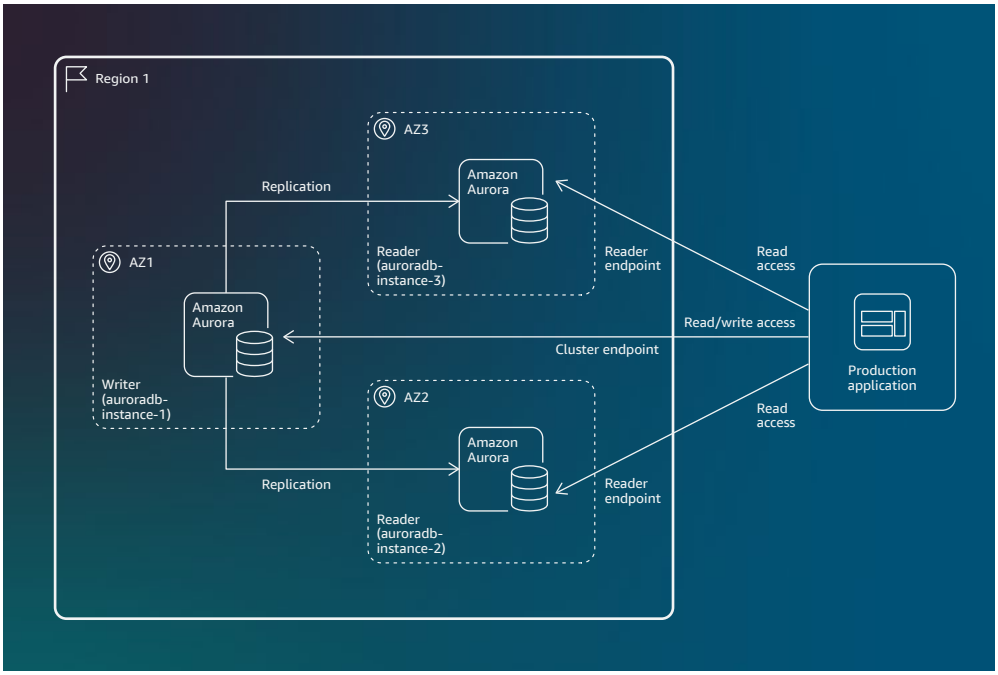
When the old primary Region becomes available again, Aurora automatically adds back the old primary to the Global Database topology as a seondary Region

Maintaining availability during patching or upgrades and disruptive schema changes

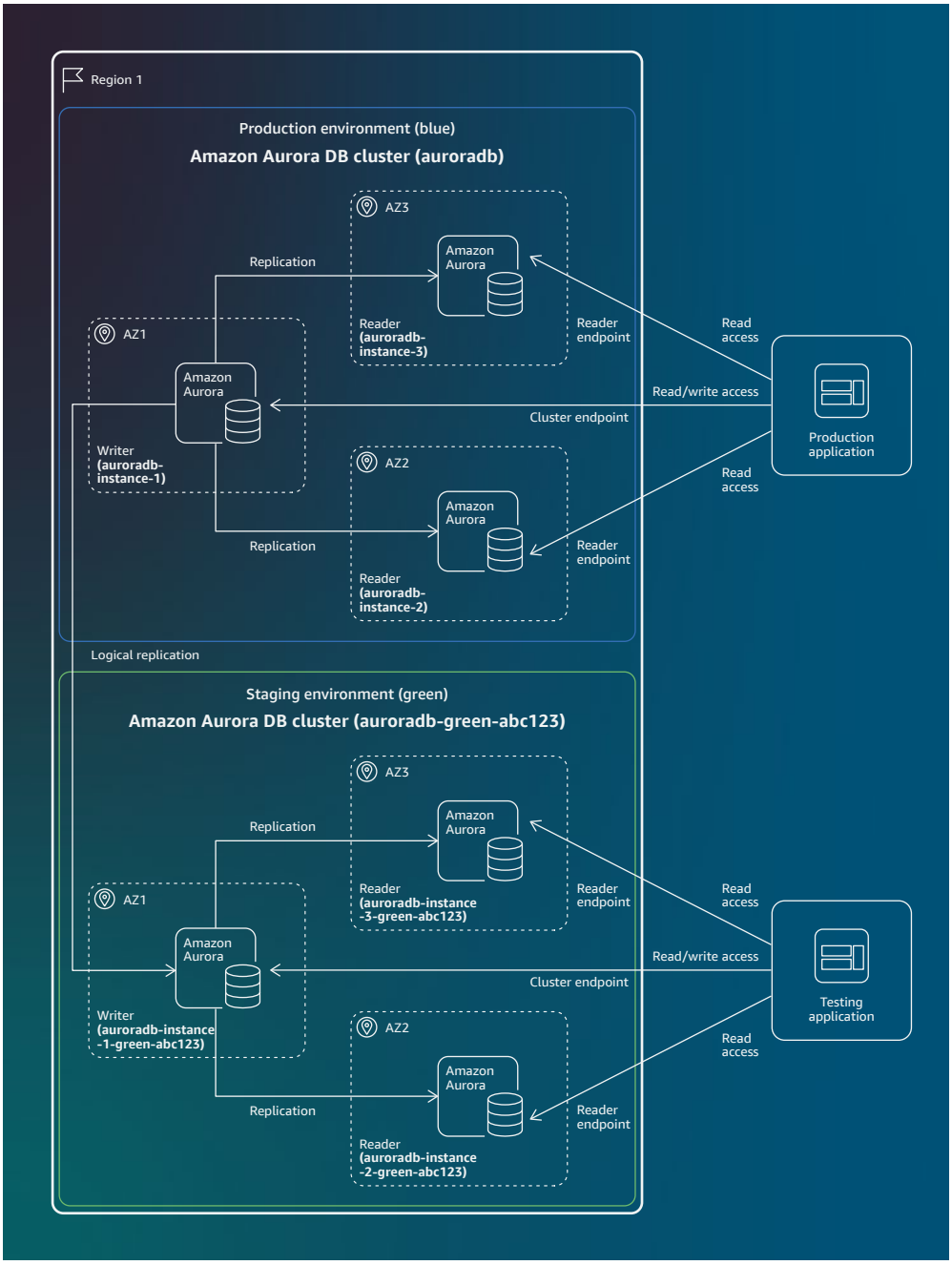
Planned downtime, often necessitated by maintenance tasks like version upgrades, patching, and schema changes, can vary in duration from minutes to days. Employing a database replica for these tasks, followed by redirecting production traffic to the promoted replica, mitigates downtime. However, replication setup, promotion, and switchover can be complex and error-prone, especially at scale. Aurora blue/green deployment offers a managed solution, significantly simplifying replication.

Blue/Green Deployments on Aurora

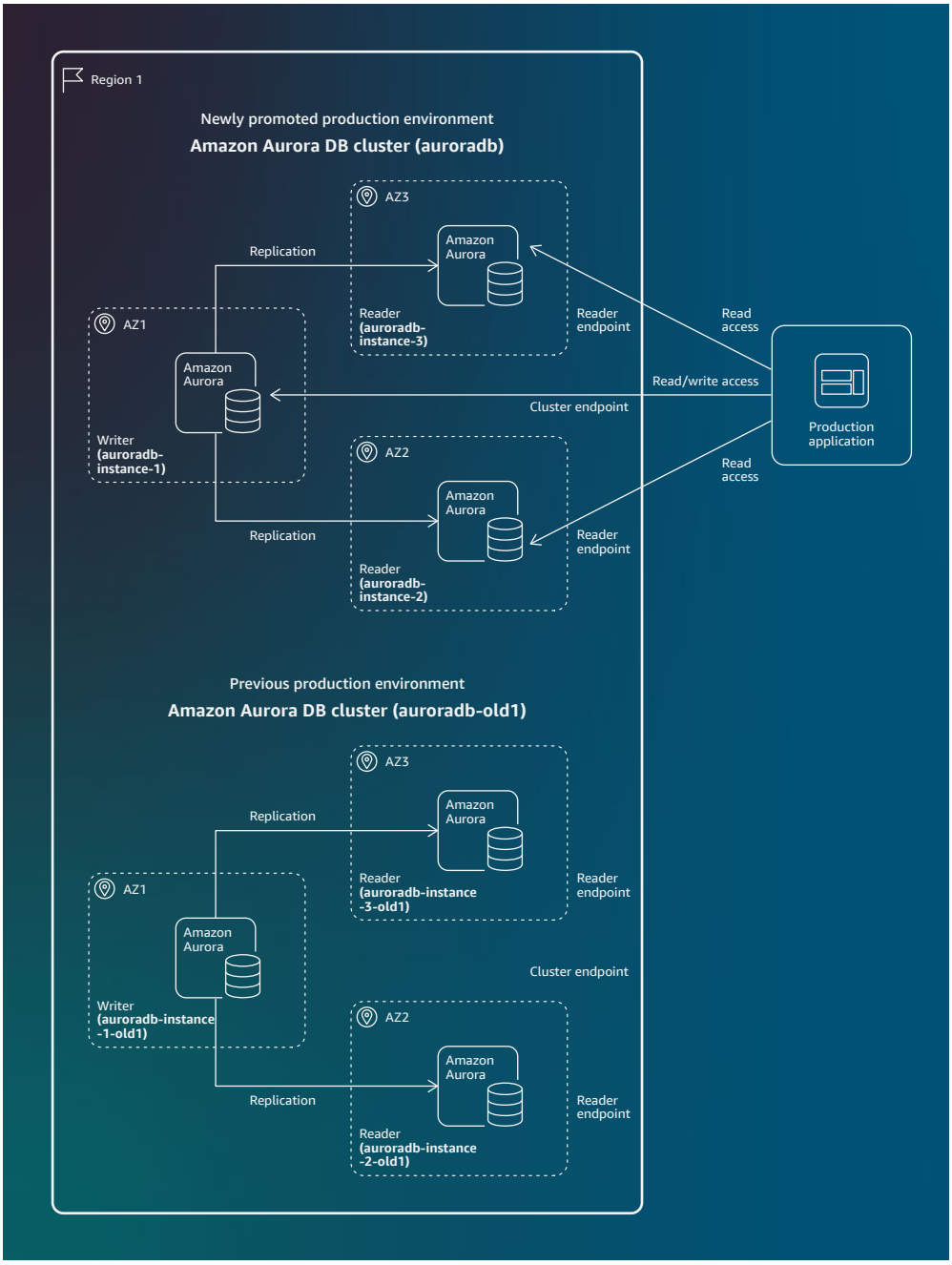
- Blue/Green Deployments on Aurora facilitate the creation of synchronized staging environments mirroring production environments.
- The production (blue) environment and the replicated (green) environment are kept in sync with logical replication using replication logs.
- The green environment can be quickly promoted to production with no data loss, blocking writes on both environments during switchover to ensure synchronization.
- Production traffic redirection to the newly promoted green environment results in a short downtime, usually under a minute, but downtime can be longer depending on your workload. Once switchover is complete, the names and endpoints in the blue environment are assigned to the newly promoted green environment, requiring no changes to your application.



Production environment



Production and staging environment



Newly promoted and previous production environment

See Using Amazon RDS Blue/Green Deployments for Database Updates in the Aurora user guide for more details.

Conclusion

Relational databases are crucial components of highly available applications. Aurora, a high-throughput RDBMS, maintains availability and durability in cloud-scale environments. Business-critical workloads can use MAZ Aurora clusters to enhance uptime and mitigate availability events. Aurora Global Database extends this capability and facilitates robust DR solutions by enabling globally distributed applications with minimal RTO, RPO, and low-latency reads across Regions.

To get started today, see the Aurora user guide and Working with Aurora Global Database.

Contributors

Contributors to this document include:

- **Aditya Samant**
  Principal RDS and Aurora Database Solutions Architect, AWS
- **Shayon Sanyal**
  Principal RDS and Aurora Database Solutions Architect, AWS

Further reading

For additional information, refer to:

- AWS Architecture Center
- Disaster Recovery of Workloads on AWS: Recovery in the Cloud
- Guidance for Disaster Recovery Using Amazon Aurora

© 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.