**Amazon Aurora**

## Using the Instance Endpoints

In advanced use cases, you might configure some DB instances differently than others. In this case, use the instance endpoint to connect directly to an instance that is smaller, larger, or otherwise has different characteristics than the others. Also, set up failover priority so that this special DB instance is the last choice to take over as the primary instance. We recommend that you use custom endpoints instead of the instance endpoint in such cases. Doing so simplifies connection management and high availability as you add more DB instances to your cluster.

Each DB instance in an Aurora cluster has its own built-in instance endpoint, whose name and other attributes are managed by Aurora. You can't create, delete, or modify this kind of endpoint.

## How Aurora Endpoints Work with High Availability

For clusters where high availability is important, where practical use the cluster endpoint for read-write connections and the reader endpoint for read-only connections. These kinds of connections manage DB instance failover better than instance endpoints do. The instance endpoints connect to a specific DB instance in a DB cluster, requiring logic in your application to choose a different endpoint if the DB instance becomes unavailable.

If the primary DB instance of a DB cluster fails, Aurora automatically fails over to a new primary DB instance. It does so by either promoting an existing Aurora Replica to a new primary DB instance or creating a new primary DB instance. If a failover occurs, you can use the cluster endpoint to reconnect to the newly promoted or created primary DB instance, or use the reader endpoint to reconnect to one of the Aurora Replicas in the DB cluster. During a failover, the reader endpoint might direct connections to the new primary DB instance of a DB cluster for a short time after an Aurora Replica is promoted to the new primary DB instance.

# Amazon Aurora DB Clusters

An Amazon Aurora DB cluster consists of one or more DB instances and a cluster volume that manages the data for those DB instances. An Aurora cluster volume is a virtual database storage volume that spans multiple Availability Zones, with each Availability Zone having a copy of the DB cluster data. Two types of DB instances make up an Aurora DB cluster:

* **Primary DB instance** – Supports read and write operations, and performs all of the data modifications to the cluster volume. Each Aurora DB cluster has one primary DB instance.
* **Aurora Replica** – Connects to the same storage volume as the primary DB instance and supports only read operations. Each Aurora DB cluster can have up to 15 Aurora Replicas in addition to the primary DB instance. Maintain high availability by locating Aurora Replicas in separate Availability Zones. Aurora automatically fails over to an Aurora Replica in case the primary DB instance becomes unavailable. You can specify the failover priority for Aurora Replicas. Aurora Replicas can also offload read workloads from the primary DB instance.
* For Aurora multi-master clusters, all DB instances have read-write capability. In this case, the distinction between primary instance and Aurora Replica doesn't apply. For discussing replication topology where the clusters can use either single-master or multi-master replication, we call these writer and reader DB instances.

The following diagram illustrates the relationship between the cluster volume, the primary DB instance, and Aurora Replicas in an Aurora DB cluster.


        Amazon Aurora Architecture
      

**Note**

The preceding information applies to all the Aurora clusters that use single-master replication. These include provisioned clusters, parallel query clusters, global database clusters, serverless clusters, and all MySQL 5.7-compatible and PostgreSQL-compatible clusters.

Aurora clusters that use multi-master replication have a different arrangement of read-write and read-only DB instances. All DB instances in a multi-master cluster can perform write operations. There isn't a single DB instance that performs all the write operations, and there aren't any read-only DB instances. Therefore, the terms primary instance and Aurora Replica don't apply to multi-master clusters. When we discuss clusters that might use multi-master replication, we refer to writer DB instances andreader DB instances.

The Aurora cluster illustrates the separation of compute capacity and storage. For example, an Aurora configuration with only a single DB instance is still a cluster, because the underlying storage volume involves multiple storage nodes distributed across multiple Availability Zones (AZs).

# Amazon Aurora Connection Management

Amazon Aurora typically involves a cluster of DB instances instead of a single instance. Each connection is handled by a specific DB instance. When you connect to an Aurora cluster, the host name and port that you specify point to an intermediate handler called an endpoint. Aurora uses the endpoint mechanism to abstract these connections. Thus, you don't have to hardcode all the hostnames or write your own logic for load-balancing and rerouting connections when some DB instances aren't available.

For certain Aurora tasks, different instances or groups of instances perform different roles. For example, the primary instance handles all data definition language (DDL) and data manipulation language (DML) statements. Up to 15 Aurora Replicas handle read-only query traffic.

Using endpoints, you can map each connection to the appropriate instance or group of instances based on your use case. For example, to perform DDL statements you can connect to whichever instance is the primary instance. To perform queries, you can connect to the reader endpoint, with Aurora automatically performing load-balancing among all the Aurora Replicas. For clusters with DB instances of different capacities or configurations, you can connect to custom endpoints associated with different subsets of DB instances. For diagnosis or tuning, you can connect to a specific instance endpoint to examine details about a specific DB instance.

## Types of Aurora Endpoints

An endpoint is represented as an Aurora-specific URL that contains a host address and a port. The following types of endpoints are available from an Aurora DB cluster.

**Cluster endpoint**

A cluster endpoint for an Aurora DB cluster that connects to the current primary DB instance for that DB cluster. This endpoint is the only one that can perform write operations such as DDL statements. Because of this, the cluster endpoint is the one that you connect to when you first set up a cluster or when your cluster only contains a single DB instance.

Each Aurora DB cluster has one cluster endpoint and one primary DB instance.

The cluster endpoint provides failover support for read/write connections to the DB cluster. If the current primary DB instance of a DB cluster fails, Aurora automatically fails over to a new primary DB instance. During a failover, the DB cluster continues to serve connection requests to the cluster endpoint from the new primary DB instance, with minimal interruption of service.

**Reader endpoint**

A *reader endpoint* for an Aurora DB cluster connects to one of the available Aurora Replicas for that DB cluster. Each Aurora DB cluster has one reader endpoint. If there is more than one Aurora Replica, the reader endpoint directs each connection request to one of the Aurora Replicas.

The reader endpoint provides load-balancing support for read-only connections to the DB cluster. Use the reader endpoint for read operations, such as queries. You can't use the reader endpoint for write operations.

The DB cluster distributes connection requests to the reader endpoint among the available Aurora Replicas. If the DB cluster contains only a primary DB instance, the reader endpoint serves connection requests from the primary DB instance. If one or more Aurora Replicas are created for that DB cluster, subsequent connections to the reader endpoint are load-balanced among the Replicas.

**Custom endpoint**

A *custom endpoint* for an Aurora cluster represents a set of DB instances that you choose. When you connect to the endpoint, Aurora performs load balancing and chooses one of the instances in the group to handle the connection. You define which instances this endpoint refers to, and you decide what purpose the endpoint serves.

An Aurora DB cluster has no custom endpoints until you create one. You can create up to five custom endpoints for each provisioned Aurora cluster. You can't use custom endpoints for Aurora Serverless clusters.

The custom endpoint provides load-balanced database connections based on criteria other than the read-only or read-write capability of the DB instances. For example, you might define a custom endpoint to connect to instances that use a particular AWS instance class or a particular DB parameter group. Then you might tell particular groups of users about this custom endpoint. For example, you might direct internal users to low-capacity instances for report generation or ad hoc (one-time) querying, and direct production traffic to high-capacity instances.

This feature is intended for advanced users with specialized kinds of workloads where it isn't practical to keep all the Aurora Replicas in the cluster identical. With custom endpoints, you can predict the capacity of the DB instance used for each connection. When you use custom endpoints, you typically don't use the reader endpoint for that cluster.

**Instance endpoint**

An *instance endpoint* connects to a specific DB instance within an Aurora cluster. Each DB instance in a DB cluster has its own unique instance endpoint. So there is one instance endpoint for the current primary DB instance of the DB cluster, and there is one instance endpoint for each of the Aurora Replicas in the DB cluster.

The instance endpoint provides direct control over connections to the DB cluster, for scenarios where using the cluster endpoint or reader endpoint might not be appropriate. For example, your client application might require more fine-grained load balancing based on workload type. In this case, you can configure multiple clients to connect to different Aurora Replicas in a DB cluster to distribute read workloads. For an example that uses instance endpoints to improve connection speed after a failover for Aurora PostgreSQL, see [Fast Failover with Amazon Aurora PostgreSQL](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraPostgreSQL.BestPractices.html#AuroraPostgreSQL.BestPractices.FastFailover). For an example that uses instance endpoints to improve connection speed after a failover for Aurora MySQL, see [MariaDB Connector/J failover support – case Amazon Aurora](https://mariadb.org/mariadb-connectorj-failover-support-case-amazon-aurora/" \t "_blank).

# Amazon Aurora Storage and Reliability

## Overview of Aurora Storage

Aurora data is stored in the cluster volume, which is a single, virtual volume that uses solid state drives (SSDs). A cluster volume consists of copies of the data across multiple Availability Zones in a single AWS Region. Because the data is automatically replicated across Availability Zones, your data is highly durable with less possibility of data loss. This replication also ensures that your database is more available during a failover. It does so because the data copies already exist in the other Availability Zones and continue to serve data requests to the DB instances in your DB cluster. The amount of replication is independent of the number of DB instances in your cluster.

## What the Cluster Volume Contains

The Aurora shared storage architecture makes your data independent from the DB instances in the cluster. For example, you can add a DB instance quickly because Aurora doesn't make a new copy of the table data. Instead, the DB instance connects to the shared volume that already contains all your data. You can remove a DB instance from a cluster without removing any of the underlying data from the cluster. Only when you delete the entire cluster does Aurora remove the data.

## How Aurora Storage Grows

Aurora cluster volumes automatically grow as the amount of data in your database increases. An Aurora cluster volume can grow to a maximum size of 64 tebibytes (TiB). Table size is limited to the size of the cluster volume. That is, the maximum table size for a table in an Aurora DB cluster is 64 TiB.

## Amazon Aurora Reliability

Aurora is designed to be reliable, durable, and fault tolerant. You can architect your Aurora DB cluster to improve availability by doing things such as adding Aurora Replicas and placing them in different Availability Zones, and also Aurora includes several automatic features that make it a reliable database solution.

### Storage Auto-Repair

Because Aurora maintains multiple copies of your data in three Availability Zones, the chance of losing data as a result of a disk failure is greatly minimized. Aurora automatically detects failures in the disk volumes that make up the cluster volume. When a segment of a disk volume fails, Aurora immediately repairs the segment. When Aurora repairs the disk segment, it uses the data in the other volumes that make up the cluster volume to ensure that the data in the repaired segment is current.

### Survivable Cache Warming

Aurora "warms" the buffer pool cache when a database starts up after it has been shut down or restarted after a failure. That is, Aurora preloads the buffer pool with the pages for known common queries that are stored in an in-memory page cache. This provides a performance gain by bypassing the need for the buffer pool to "warm up" from normal database use.

The Aurora page cache is managed in a separate process from the database, which allows the page cache to survive independently of the database. In the unlikely event of a database failure, the page cache remains in memory, which ensures that the buffer pool is warmed with the most current state when the database restarts.

### Crash Recovery

Aurora is designed to recover from a crash almost instantaneously and continue to serve your application data without the binary log. Aurora performs crash recovery asynchronously on parallel threads, so that your database is open and available immediately after a crash.

The following are considerations for binary logging and crash recovery on Aurora MySQL:

* Enabling binary logging on Aurora directly affects the recovery time after a crash, because it forces the DB instance to perform binary log recovery.
* The type of binary logging used affects the size and efficiency of logging. For the same amount of database activity, some formats log more information than others in the binary logs. The following settings for the binlog\_format parameter result in different amounts of log data:
  + ROW – The most log data
  + STATEMENT – The least log data
  + MIXED – A moderate amount of log data that usually provides the best combination of data integrity and performance

The amount of binary log data affects recovery time. If there is more data logged in the binary logs, the DB instance must process more data during recovery, which increases recovery time.

* Aurora does not need the binary logs to replicate data within a DB cluster or to perform point in time restore (PITR).
* If you don't need the binary log for external replication (or an external binary log stream), we recommend that you set the binlog\_format parameter to OFF to disable binary logging. Doing so reduces recovery time.

# Amazon Aurora Security

Security for Amazon Aurora is managed at three levels:

* To control who can perform Amazon RDS management actions on Aurora DB clusters and DB instances, you use AWS Identity and Access Management (IAM). When you connect to AWS using IAM credentials, your IAM account must have IAM policies that grant the permissions required to perform Amazon RDS management operations.
* Aurora DB clusters must be created in an Amazon Virtual Private Cloud (VPC). To control which devices and Amazon EC2 instances can open connections to the endpoint and port of the DB instance for Aurora DB clusters in a VPC, you use a VPC security group. In addition, firewall rules at your company can control whether devices running at your company can open connections to a DB instance.
* To authenticate logins and permissions for an Amazon Aurora DB cluster, you can take either of the following approaches, or a combination of them.
  + You can take the same approach as with a stand-alone DB instance of MySQL or PostgreSQL.
  + You can also use IAM database authentication for Aurora MySQL.

## Using SSL with Aurora DB Clusters

Amazon Aurora DB clusters support Secure Sockets Layer (SSL) connections from applications using the same process and public key as Amazon RDS DB instances.

# High Availability for Aurora

Aurora stores copies of the data in a DB cluster across multiple Availability Zones in a single AWS Region, regardless of whether the instances in the DB cluster span multiple Availability Zones.

In an Aurora cluster using single-master replication, when you create reader instances across Availability Zones, Amazon RDS automatically provisions and maintains them synchronously. The primary DB instance is synchronously replicated across Availability Zones to Aurora Replicas to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.

Using the RDS console, you can create a Multi-AZ deployment by simply specifying Multi-AZ when creating a DB cluster. If a DB cluster is in a single Availability Zone, you can make it a Multi-AZ DB cluster adding another DB instance in a different Availability Zone.

For a cluster using single-master replication, after you create the primary instance, you can create up to 15 Aurora Replicas. These read-only DB instances support SELECT queries for read-intensive applications. We recommend that you distribute the primary instance and Aurora Replicas in your DB cluster over multiple Availability Zones to improve the availability of your DB cluster.

# Amazon Aurora Global Database

## Overview of Aurora Global Database

An Aurora global database consists of one primary AWS Region where your data is mastered, and one read-only, secondary AWS Region. Aurora replicates data to the secondary AWS Region with typical latency of under a second. You issue write operations directly to the primary DB instance in the primary AWS Region. An Aurora global database uses dedicated infrastructure to replicate your data, leaving database resources available entirely to serve application workloads. Applications with a worldwide footprint can use reader instances in the secondary AWS Region for low latency reads. In the unlikely event your database becomes degraded or isolated in an AWS region, you can promote the secondary AWS Region to take full read-write workloads in under a minute.

The Aurora cluster in the primary AWS Region where your data is mastered performs both read and write operations. The cluster in the secondary region enables low-latency reads. You can scale up the secondary cluster independently by adding one or more DB instances (Aurora Replicas) to serve read-only workloads. For disaster recovery, you can remove and promote the secondary cluster to allow full read and write operations.

Only the primary cluster performs write operations. Clients that perform write operations connect to the DB cluster endpoint of the primary cluster.

### Advantages of Aurora Global Database

* The replication performed by an Aurora global database has limited to no performance impact on the primary DB cluster. The resources of the DB instances are fully devoted to serve read and write workloads.
* Changes are replicated between AWS Regions with minimal lag, typically less than 1 second.
* The secondary cluster enables fast failover for disaster recovery. You can typically promote a secondary cluster and make it available for writes in under a minute.
* Applications in remote AWS Regions experience lower query latency when they read from a secondary cluster.
* You can add up to 16 Aurora Replicas to the secondary cluster, allowing you to scale reads beyond the capacity of a single Aurora cluster.

### Current Limitations of Aurora Global Database

* You can't create a cross-region Read Replica from the primary cluster in same region as the secondary. See [Replicating Amazon Aurora MySQL DB Clusters Across AWS Regions](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Replication.CrossRegion.html) for information about cross-region Read Replicas.
* The following features aren't supported for Aurora Global Database:
  + Cloning.
  + Backtrack.
  + Parallel query.
  + Aurora Serverless.
  + Stopping and starting the DB clusters within the global database.

# Replication with Amazon Aurora

## Aurora Replicas

Aurora Replicas are independent endpoints in an Aurora DB cluster, best used for scaling read operations and increasing availability. Up to 15 Aurora Replicas can be distributed across the Availability Zones that a DB cluster spans within an AWS Region. The DB cluster volume is made up of multiple copies of the data for the DB cluster. However, the data in the cluster volume is represented as a single, logical volume to the primary instance and to Aurora Replicas in the DB cluster.

As a result, all Aurora Replicas return the same data for query results with minimal replica lag—usually much less than 100 milliseconds after the primary instance has written an update.

Aurora Replicas work well for read scaling because they are fully dedicated to read operations on your cluster volume. Write operations are managed by the primary instance. Because the cluster volume is shared among all DB instances in your DB cluster, minimal additional work is required to replicate a copy of the data for each Aurora Replica.

To increase availability, you can use Aurora Replicas as failover targets. That is, if the primary instance fails, an Aurora Replica is promoted to the primary instance. There is a brief interruption during which read and write requests made to the primary instance fail with an exception, and the Aurora Replicas are rebooted. If your Aurora DB cluster doesn't include any Aurora Replicas, then your DB cluster will be unavailable for the duration it takes your DB instance to recover from the failure event. However, promoting an Aurora Replica is much faster than recreating the primary instance. For high-availability scenarios, we recommend that you create one or more Aurora Replicas. These should be of the same DB instance class as the primary instance and in different Availability Zones for your Aurora DB cluster..

**Note**

You can't create an encrypted Aurora Replica for an unencrypted Aurora DB cluster.

You can't create an unencrypted Aurora Replica for an encrypted Aurora DB cluster.

**DB Instance Billing for Aurora**

Amazon RDS instances in an Aurora cluster are billed based on the following components:

* DB instance hours (per hour)
* I/O requests (per 1 million requests per month)
* Backup storage (per GiB per month)
* Data transfer (per GB)

Amazon RDS provides the following purchasing options to enable you to optimize your costs based on your needs:

**On-Demand Instances**

**Reserved Instances**

### Reserved Offering Types

Reserved DB instances are available in three varieties—No Upfront, Partial Upfront, and All Upfront—that let you optimize your Amazon RDS costs based on your expected usage.

**No Upfront**

**Partial Upfront**

**All Upfront**

# Migrating Data to an Amazon Aurora DB Cluster

## Migrating Data to an Amazon Aurora MySQL DB Cluster

You can migrate data from one of the following sources to an Amazon Aurora MySQL DB cluster.

* An Amazon RDS MySQL DB instance
* A MySQL database external to Amazon RDS
* A database that is not MySQL-compatible

## Migrating Data to an Amazon Aurora PostgreSQL DB Cluster

You can migrate data from one of the following sources to an Amazon Aurora PostgreSQL DB cluster.

* An Amazon RDS PostgreSQL DB instance
* A database that is not PostgreSQL-compatible

**Amazon Aurora Serverless**

Amazon Aurora Serverless is an on-demand, autoscaling configuration for Amazon Aurora. An *Aurora Serverless DB cluster* is a DB cluster that automatically starts up, shuts down, and scales up or down its compute capacity based on your application's needs. Aurora Serverless provides a relatively simple, cost-effective option for infrequent, intermittent, or unpredictable workloads. It can provide this because it automatically starts up, scales compute capacity to match your application's usage, and shuts down when it's not in use.

**Note**

A non-Serverless DB cluster for Aurora is called a *provisioned DB cluster*. Aurora Serverless clusters and provisioned clusters both have the same kind of high-capacity, distributed, and highly available storage volume.

## Advantages of Aurora Serverless

Aurora Serverless provides the following advantages:

**Simpler**

**Scalable**

**Cost-effective**

**Highly available storage**

## Use Cases for Aurora Serverless

**Infrequently used applications**

**New applications**

**Variable workloads**

**Unpredictable workloads**

**Development and test databases**

**Multi-tenant applications**

## Limitations of Aurora Serverless

The following limitations apply to Aurora Serverless:

* Aurora Serverless is only available for the following:
  + Aurora with MySQL version 5.6 compatibility
  + Aurora with PostgreSQL version 10.7 compatibility.
* The port number for connections must be:
  + 3306 forAurora MySQL
  + 5432 for Aurora PostgreSQL
* You can't give an Aurora Serverless DB cluster a public IP address. You can access an Aurora Serverless DB cluster only from within a virtual private cloud (VPC) based on the Amazon VPC service.
* Each Aurora Serverless DB cluster requires two AWS PrivateLink endpoints. If you reach the limit for PrivateLink endpoints within your VPC, you can't create any more Aurora Serverless clusters in that VPC.
* You can't access an Aurora Serverless DB cluster's endpoint through an AWS VPN connection or an inter-region VPC peering connection. There are limitations in accessing a cluster's endpoint through an intra-region VPC peering connection; However, you can access an Aurora Serverless cluster's endpoint through an AWS Direct Connect connection.
* A DB subnet group used by Aurora Serverless can’t have more than one subnet in the same Availability Zone.
* Changes to a subnet group used by an Aurora Serverless DB cluster are not applied to the cluster.
* Aurora Serverless doesn't support the following features:
  + [Loading data from an Amazon S3 bucket](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Integrating.LoadFromS3.html)
  + [Saving data to an Amazon S3 bucket](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Integrating.SaveIntoS3.html)
  + [Invoking an AWS Lambda function with an Aurora MySQL native function](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Integrating.Lambda.html#AuroraMySQL.Integrating.NativeLambda)
  + [Aurora Replicas](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Replication.html)
  + [Backtrack](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Managing.Backtrack.html)
  + [Multi-master clusters](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-multi-master.html)
  + [Database cloning](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.Managing.Clone.html)
  + [IAM database authentication](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/UsingWithRDS.IAMDBAuth.html)
  + [Restoring a snapshot from a MySQL DB instance](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Migrating.RDSMySQL.html)
  + [Migrating backup files from Amazon S3](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Migrating.ExtMySQL.html#AuroraMySQL.Migrating.ExtMySQL.S3)
  + [Amazon RDS Performance Insights](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/USER_PerfInsights.html)

**Note**

You can access an Aurora Serverless DB cluster from AWS Lambda.

You can connect to Aurora Serverless clusters using the Transport Layer Security/Secure Sockets Layer (TLS/SSL) protocol.

## Aurora Serverless Architecture

The following image provides an overview of the Aurora Serverless architecture.


                  Aurora Serverless Architecture
              

You can specify the minimum and maximum ACU. The minimum Aurora capacity unit is the lowest ACU to which the DB cluster can scale down. The maximum Aurora capacity unit is the highest ACU to which the DB cluster can scale up. Based on your settings, Aurora Serverless automatically creates scaling rules for thresholds for CPU utilization, connections, and available memory.

Aurora Serverless manages the warm pool of resources in an AWS Region to minimize scaling time. When Aurora Serverless adds new resources to the Aurora DB cluster, it uses the proxy fleet to switch active client connections to the new resources. At any specific time, you are only charged for the ACUs that are being actively used in your Aurora DB cluster.

## Autoscaling for Aurora Serverless

After scaling up, the cooldown period for scaling down is 15 minutes. After scaling down, the cooldown period for scaling down again is 310 seconds.

**Note**

There is no cooldown period for scaling up. Aurora Serverless can scale up whenever necessary, including immediately after scaling up or scaling down.

A scaling point is a point in time at which the database can safely initiate the scaling operation. Under the following conditions, Aurora Serverless might not be able to find a scaling point:

* Long-running queries or transactions are in progress
* Temporary tables or table locks are in use

In these cases, Aurora Serverless continues to try to find a scaling point so that it can initiate the scaling operation. It does this for as long as it determines that the DB cluster should be scaled.

## Automatic Pause and Resume for Aurora Serverless

You can choose to pause your Aurora Serverless DB cluster after a given amount of time with no activity. You specify the amount of time with no activity before the DB cluster is paused. The default is five minutes. You can also disable pausing the DB cluster.

When the DB cluster is paused, no compute or memory activity occurs, and you are charged only for storage. If database connections are requested when an Aurora Serverless DB cluster is paused, the DB cluster automatically resumes and services the connection requests.

**Note**

If a DB cluster is paused for more than seven days, the DB cluster might be backed up with a snapshot. In this case, the DB cluster is restored when there is a request to connect to it.

## Timeout Action for Capacity Changes

You can change the capacity of an Aurora Serverless DB cluster. When you change the capacity, Aurora Serverless tries to find a scaling point for the change. If Aurora Serverless can't find a scaling point, it times out. You can specify one of the following actions to take when a capacity change times out:

* **Force the capacity change** – Set the capacity to the specified value as soon as possible.
* **Roll back the capacity change** – Cancel the capacity change.

**Important**

If you force the capacity change, connections that prevent Aurora Serverless from finding a scaling point might be dropped.

## Aurora Serverless and Failover

The DB instance for an Aurora Serverless DB cluster currently is created in a single Availability Zone (AZ). If the DB instance or the AZ becomes unavailable, Aurora recreates the DB instance in a different AZ. We refer to this capability as automatic multi-AZ failover.

This failover mechanism takes longer than for an Aurora Provisioned cluster. The Aurora Serverless failover time is currently undefined because it depends on demand and capacity availability in other AZs within the given AWS Region.

Because Aurora separates computation capacity and storage, the storage volume for the cluster is spread across multiple AZs. Your data remains available even if outages affect the DB instance or the associated AZ.

## Aurora Serverless and Snapshots

The cluster volume for an Aurora Serverless cluster is always encrypted. You can choose the encryption key, but not turn off encryption. To copy or share a snapshot of an Aurora Serverless cluster, you encrypt the snapshot using your own KMS key.

**Security in Amazon Aurora**

* Run your DB cluster in a virtual private cloud (VPC)
* Use AWS Identity and Access Management (IAM) policies to assign permissions IAM to
* Use security groups Use Secure Socket Layer (SSL) connections with DB clusters
* Use Amazon Aurora encryption to secure your DB clusters and snapshots at rest.
* Use the security features of your DB engine to control who can log in to the databases on a DB cluster.
* IAM database authentication works with Aurora MySQL and Aurora PostgreSQL. With this authentication method, you don't need to use a password when you connect to a DB cluster. Instead, you use an authentication token.

# Data Protection in Amazon Aurora

For data protection, we recommend that you protect AWS account credentials and set up principals with AWS Identity and Access Management (IAM). Doing this means that each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

* Use multi-factor authentication (MFA) with each account.
* Use SSL/TLS to communicate with AWS resources.
* Set up API and user activity logging with AWS CloudTrail.
* Use AWS encryption solutions, along with all default security controls within AWS services.
* Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.
* For Aurora PostgreSQL, use database activity streams to monitor and audit database activity to provide safeguards for your database and meet compliance and regulatory requirements.

# Internetwork Traffic Privacy

Connections are protected both between Amazon Aurora and on-premises applications and between Amazon Aurora and other AWS resources within the same AWS Region.

# Logging and Monitoring in Amazon Aurora

**Amazon CloudWatch Alarms**

**AWS CloudTrail Logs**

**Enhanced Monitoring**

**Amazon RDS Performance Insights**

**Database Logs**

**Amazon Aurora Recommendations**

**Amazon Aurora Event Notification**

**AWS Trusted Advisor**

**Database Activity Streams**

# Resilience in Amazon Aurora

## Backup and Restore

Aurora backs up your cluster volume automatically and retains restore data for the length of the backup retention period. Aurora backups are continuous and incremental so you can quickly restore to any point within the backup retention period. You can specify a backup retention period, from 1 to 35 days, when you create or modify a DB cluster.

If you want to retain a backup beyond the backup retention period, you can also take a snapshot of the data in your cluster volume. Aurora retains incremental restore data for the entire backup retention period. Thus, you need to create a snapshot only for data that you want to retain beyond the backup retention period. You can create a new DB cluster from the snapshot.

You can recover your data by creating a new Aurora DB cluster from the backup data that Aurora retains, or from a DB cluster snapshot that you have saved. You can quickly create a new copy of a DB cluster from backup data to any point in time during your backup retention period. The continuous and incremental nature of Aurora backups during the backup retention period means you don't need to take frequent snapshots of your data to improve restore times.

## Replication

Aurora Replicas are independent endpoints in an Aurora DB cluster, best used for scaling read operations and increasing availability. Up to 15 Aurora Replicas can be distributed across the Availability Zones that a DB cluster spans within an AWS Region. The DB cluster volume is made up of multiple copies of the data for the DB cluster. However, the data in the cluster volume is represented as a single, logical volume to the primary DB instance and to Aurora Replicas in the DB cluster. If the primary DB instance fails, an Aurora Replica is promoted to be the primary DB instance.

Aurora also supports replication options that are specific to Aurora MySQL and Aurora PostgreSQL.

## Failover

Aurora stores copies of the data in a DB cluster across multiple Availability Zones in a single AWS Region. This storage occurs regardless of whether the DB instances in the DB cluster span multiple Availability Zones. When you create Aurora Replicas across Availability Zones, Aurora automatically provisions and maintains them synchronously. The primary DB instance is synchronously replicated across Availability Zones to Aurora Replicas to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups. Running a DB cluster with high availability can enhance availability during planned system maintenance, and help protect your databases against failure and Availability Zone disruption.

# Infrastructure Security in Amazon Aurora

## Security Groups

Security groups control the access that traffic has in and out of a DB instance. By default, network access is turned off to a DB instance. You can specify rules in a security group that allow access from an IP address range, port, or Amazon EC2 security group. After ingress rules are configured, the same rules apply to all DB instances that are associated with that security group.

## Public Accessibility

When you launch a DB instance inside a virtual private cloud (VPC) based on the Amazon VPC service, you can turn on or off public accessibility for that instance. To designate whether the DB instance that you create has a DNS name that resolves to a public IP address, you use the Public accessibility parameter. By using this parameter, you can designate whether there is public access to the DB instance. You can modify a DB instance to turn on or off public accessibility by modifying the Public accessibility parameter.

# Security Best Practices for Amazon Aurora

* Assign an individual IAM account to each person who manages Amazon Aurora resources. Don't use AWS root credentials to manage Amazon Aurora resources; you should create an IAM user for everyone, including yourself.
* Grant each user the minimum set of permissions required to perform his or her duties.
* Use IAM groups to effectively manage permissions for multiple users.
* Rotate your IAM credentials regularly.
* Configure AWS Secrets Manager to automatically rotate the secrets for Amazon Aurora.

**Monitoring an Amazon Aurora DB Cluster**

Advice about specific types of metrics follows:

* **High CPU or RAM consumption**
* **Disk space consumption**
* **Network traffic**
* **Database connections**
* **IOPS metrics**

## Monitoring Tools

### Automated Monitoring Tools

You can use the following automated monitoring tools to watch Amazon RDS and report when something is wrong:

* **Amazon RDS Events**
* **Database log files**
* **Amazon RDS Enhanced Monitoring**

In addition, Amazon RDS integrates with Amazon CloudWatch for additional monitoring capabilities:

* **Amazon CloudWatch Metrics**
* **Amazon CloudWatch Alarms**
* **Amazon CloudWatch Logs**

### Manual Monitoring Tools

* From the Amazon RDS console, you can monitor the following items for your resources:
  + The number of connections to a DB instance
  + The amount of read and write operations to a DB instance
  + The amount of storage that a DB instance is currently utilizing
  + The amount of memory and CPU being utilized for a DB instance
  + The amount of network traffic to and from a DB instance
* From the AWS Trusted Advisor dashboard, you can review the following cost optimization, security, fault tolerance, and performance improvement checks:
  + Amazon RDS Idle DB Instances
  + Amazon RDS Security Group Access Risk
  + Amazon RDS Backups
  + Amazon RDS Multi-AZ
  + Aurora DB Instance Accessibility

For more information on these checks, see [Trusted Advisor Best Practices (Checks)](https://aws.amazon.com/premiumsupport/trustedadvisor/best-practices/).

* CloudWatch home page shows:
  + Current alarms and status
  + Graphs of alarms and resources
  + Service health status

In addition, you can use CloudWatch to do the following:

* + Create [customized dashboards](https://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/CloudWatch_Dashboards.html) to monitor the services you care about
  + Graph metric data to troubleshoot issues and discover trends
  + Search and browse all your AWS resource metrics
  + Create and edit alarms to be notified of problems