**Amazon Relational Database Service (Amazon RDS)**

## Overview of Amazon RDS

Why do you want a managed relational database service? Because Amazon RDS takes over many of the difficult or tedious management tasks of a relational database:

* When you buy a server, you get CPU, memory, storage, and IOPS, all bundled together. With Amazon RDS, these are split apart so that you can scale them independently. If you need more CPU, less IOPS, or more storage, you can easily allocate them.
* Amazon RDS manages backups, software patching, automatic failure detection, and recovery.
* To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and it restricts access to certain system procedures and tables that require advanced privileges.
* You can have automated backups performed when you need them, or manually create your own backup snapshot. You can use these backups to restore a database. The Amazon RDS restore process works reliably and efficiently.
* You can get high availability with a primary instance and a synchronous secondary instance that you can fail over to when problems occur. You can also use MySQL, MariaDB, or PostgreSQL Read Replicas to increase read scaling.
* You can use the database products you are already familiar with: MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server.
* In addition to the security in your database package, you can help control who can access your RDS databases by using AWS Identity and Access Management (IAM) to define users and permissions. You can also help protect your databases by putting them in a virtual private cloud.

# Amazon RDS DB Instances

A DB instance is an isolated database environment running in the cloud. It is the basic building block of Amazon RDS. A DB instance can contain multiple user-created databases, and can be accessed using the same client tools and applications you might use to access a standalone database instance.

**Note**

Amazon RDS supports access to databases using any standard SQL client application. Amazon RDS does not allow direct host access.

Each DB instance has a DB instance identifier. This customer-supplied name uniquely identifies the DB instance when interacting with the Amazon RDS API and AWS CLI commands. The DB instance identifier must be unique for that customer in an AWS Region.

Each DB instance supports a database engine. Amazon RDS currently supports MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server, and Amazon Aurora database engines.

When creating a DB instance, some database engines require that a database name be specified. A DB instance can host multiple databases, or a single Oracle database with multiple schemas. The database name value depends on the database engine:

* For the MySQL and MariaDB database engines, the database name is the name of a database hosted in your DB instance. Databases hosted by the same DB instance must have a unique name within that instance.
* For the Oracle database engine, database name is used to set the value of ORACLE\_SID, which must be supplied when connecting to the Oracle RDS instance.
* For the Microsoft SQL Server database engine, database name is not a supported parameter.
* For the PostgreSQL database engine, the database name is the name of a database hosted in your DB instance. A database name is not required when creating a DB instance. Databases hosted by the same DB instance must have a unique name within that instance.

Amazon RDS creates a master user account for your DB instance as part of the creation process. This master user has permissions to create databases and to perform create, delete, select, update, and insert operations on tables the master user creates. You must set the master user password when you create a DB instance, but you can change it at any time later

# Choosing the DB Instance Class

The DB instance class determines the computation and memory capacity of an Amazon RDS DB instance. The DB instance class you need depends on your processing power and memory requirements.

## DB Instance Class Types

Amazon RDS supports three types of instance classes:

* Standard,
* Memory Optimized,
* Burstable Performance.

# Amazon RDS DB Instance Storage

DB instances for Amazon RDS for MySQL, MariaDB, PostgreSQL, Oracle, and Microsoft SQL Server use Amazon Elastic Block Store (Amazon EBS) volumes for database and log storage.

## Amazon RDS Storage Types

Amazon RDS provides three storage types: General Purpose SSD (also known as gp2), Provisioned IOPS SSD (also known as io1), and magnetic. They differ in performance characteristics and price, which means that you can tailor your storage performance and cost to the needs of your database workload. You can create MySQL, MariaDB, and PostgreSQL RDS DB instances with up to 64 TiB of storage. You can create Oracle RDS DB instances with up to 64 TiB of storage. You can create SQL Server RDS DB instances with up to 16 TiB of storage. For this amount of storage, use the Provisioned IOPS SSD and General Purpose SSD storage types.

The following list briefly describes the three storage types:

* **General Purpose SSD** – General Purpose SSD, also called gp2, volumes offer cost-effective storage that is ideal for a broad range of workloads. These volumes deliver single-digit millisecond latencies and the ability to burst to 3,000 IOPS for extended periods of time.
* **Provisioned IOPS** – Provisioned IOPS storage is designed to meet the needs of I/O-intensive workloads, particularly database workloads, that require low I/O latency and consistent I/O throughput.
* **Magnetic** – Amazon RDS also supports magnetic storage for backward compatibility. We recommend that you use General Purpose SSD or Provisioned IOPS for any new storage needs.

The following are some limitations for magnetic storage:

* Doesn't allow you to scale storage when using the SQL Server database engine.
* Doesn't support elastic volumes.
* Limited to a maximum size of 3 TiB.
* Limited to a maximum of 1,000 IOPS.

### Combining Provisioned IOPS Storage with Multi-AZ deployments, or Read Replicas

For production OLTP use cases, we recommend that you use Multi-AZ deployments for enhanced fault tolerance with Provisioned IOPS storage for fast and predictable performance.

You can also use Provisioned IOPS SSD storage with Read Replicas for MySQL, MariaDB or PostgreSQL. The type of storage for a Read Replica is independent of that on the master DB instance.

## Monitoring storage performance

Amazon RDS provides several metrics that you can use to determine how your DB instance is performing. You can view the metrics on the summary page for your instance in Amazon RDS Management Console. You can also use Amazon CloudWatch to monitor these metrics.

The following metrics are useful for monitoring storage for your DB instance:

* **IOPS** – The number of I/O operations completed each second. This metric is reported as the average IOPS for a given time interval. Amazon RDS reports read and write IOPS separately on 1-minute intervals. Total IOPS is the sum of the read and write IOPS. Typical values for IOPS range from zero to tens of thousands per second.
* **Latency** – The elapsed time between the submission of an I/O request and its completion. This metric is reported as the average latency for a given time interval. Amazon RDS reports read and write latency separately on 1-minute intervals in units of seconds. Typical values for latency are in the millisecond (ms). For example, Amazon RDS reports 2 ms as 0.002 seconds.
* **Throughput** – The number of bytes each second that are transferred to or from disk. This metric is reported as the average throughput for a given time interval. Amazon RDS reports read and write throughput separately on 1-minute intervals using units of megabytes per second (MB/s). Typical values for throughput range from zero to the I/O channel’s maximum bandwidth.
* **Queue Depth** – The number of I/O requests in the queue waiting to be serviced. These are I/O requests that have been submitted by the application but have not been sent to the device because the device is busy servicing other I/O requests. Time spent waiting in the queue is a component of latency and service time (not available as a metric). This metric is reported as the average queue depth for a given time interval. Amazon RDS reports queue depth in 1-minute intervals. Typical values for queue depth range from zero to several hundred.

Measured IOPS values are independent of the size of the individual I/O operation. This means that when you measure I/O performance, you should look at the throughput of the instance, not simply the number of I/O operations.

## Factors That Affect Storage Performance

Both system activities and database workload can affect storage performance.

**System activities**

The following system-related activities consume I/O capacity and might reduce database instance performance while in progress:

* Multi-AZ standby creation
* Read replica creation
* Changing storage types

**Database workload**

**DB instance class**

# High Availability (Multi-AZ) for Amazon RDS

Amazon RDS provides high availability and failover support for DB instances using Multi-AZ deployments. Amazon RDS uses several different technologies to provide failover support.

* Multi-AZ deployments for Oracle, PostgreSQL, MySQL, and MariaDB DB instances use Amazon's failover technology.
* SQL Server DB instances use SQL Server Database Mirroring (DBM).

In a Multi-AZ deployment, Amazon RDS automatically provisions and maintains a synchronous standby replica in a different Availability Zone. The primary DB instance is synchronously replicated across Availability Zones to a standby replica to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.

**Note**

The high-availability feature is not a scaling solution for read-only scenarios; you cannot use a standby replica to serve read traffic. To service read-only traffic, you should use a Read Replica. For more information, see [Working with Read Replicas](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ReadRepl.html).


   High Availability Scenario 
  

DB instances using Multi-AZ deployments may have increased write and commit latency compared to a Single-AZ deployment, due to the synchronous data replication that occurs. You may have a change in latency if your deployment fails over to the standby replica.

## Modifying a DB Instance to Be a Multi-AZ Deployment

If you have a DB instance in a Single-AZ deployment and you modify it to be a Multi-AZ deployment (for engines other than SQL Server or Amazon Aurora), Amazon RDS takes several steps. First, Amazon RDS takes a snapshot of the primary DB instance from your deployment and then restores the snapshot into another Availability Zone. Amazon RDS then sets up synchronous replication between your primary DB instance and the new instance. This action avoids downtime when you convert from Single-AZ to Multi-AZ, but you can experience a significant performance impact when first converting to Multi-AZ. This impact is more noticeable for large and write-intensive DB instances.

Once the modification is complete, Amazon RDS triggers an event (RDS-EVENT-0025) that indicates the process is complete. You can monitor Amazon RDS events; for more information about events, see [Using Amazon RDS Event Notification](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_Events.html).

## Failover Process for Amazon RDS

In the event of a planned or unplanned outage of your DB instance, Amazon RDS automatically switches to a standby replica in another Availability Zone if you have enabled Multi-AZ.

The failover mechanism automatically changes the DNS record of the DB instance to point to the standby DB instance. As a result, you need to re-establish any existing connections to your DB instance. Due to how the Java DNS caching mechanism works, you may need to reconfigure your JVM environment. For more information on how to manage a Java application that caches DNS values in the case of a failover, see the [AWS SDK for Java](https://docs.aws.amazon.com/AWSSdkDocsJava/latest/DeveloperGuide/java-dg-jvm-ttl.html).

**Note**

You can force a failover manually when you reboot a DB instance. For more information, see [Rebooting a DB Instance](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_RebootInstance.html).

The primary DB instance switches over automatically to the standby replica if any of the following conditions occur:

* An Availability Zone outage
* The primary DB instance fails
* The DB instance's server type is changed
* The operating system of the DB instance is undergoing software patching
* A manual failover of the DB instance was initiated using **Reboot with failover**

There are several ways to determine if your Multi-AZ DB instance has failed over:

* DB event subscriptions can be setup to notify you via email or SMS that a failover has been initiated. For more information about events, see [Using Amazon RDS Event Notification](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_Events.html)
* You can view your DB events by using the Amazon RDS console or API operations.
* You can view the current state of your Multi-AZ deployment by using the Amazon RDS console and API operations.

# Amazon RDS DB Instance Lifecycle

The lifecycle of an Amazon RDS DB instance includes

* Creating
* Modifying
* maintaining and upgrading
* performing backups and restores
* rebooting, and deleting the instance.

To delete a DB instance, you must do the following:

* Provide the name of the instance
* Enable or disable the option to take a final DB snapshot of the instance
* Enable or disable the option to retain automated backups

# Working with Read Replicas

Amazon RDS uses the MariaDB, MySQL, Oracle, and PostgreSQL DB engines' built-in replication functionality to create a special type of DB instance called a Read Replica from a source DB instance. Updates made to the source DB instance are asynchronously copied to the Read Replica. You can reduce the load on your source DB instance by routing read queries from your applications to the Read Replica. Using Read Replicas, you can elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.

**Note**

The information following applies to creating Amazon RDS Read Replicas either in the same AWS Region as the source DB instance, or in a separate AWS Region.

In some cases, a Read Replica resides in a different AWS Region than its source DB instance. In these cases, Amazon RDS sets up a secure communications channel between the source DB instance and the Read Replica. Amazon RDS establishes any AWS security configurations needed to enable the secure channel, such as adding security group entries.

By default, a Read Replica is created with the same storage type as the source DB instance. However, you can create a Read Replica that has a different storage type from the source DB instance based on the options listed in the following table.

Amazon RDS doesn't support circular replication. You can't configure a DB instance to serve as a replication source for an existing DB instance. You can only create a new Read Replica from an existing DB instance. For example, if MyDBInstance replicates to ReadReplica1, you can't configure ReadReplica1 to replicate back to MyDBInstance. For MariaDB, MySQL, and PostgreSQL, you can create a Read Replica from an existing Read Replica. For example, fromReadReplica1, you can create a new Read Replica, such as ReadReplica2. For Oracle, you can't create a Read Replica from an existing Read Replica.

When you create a Read Replica, Amazon RDS takes a DB snapshot of your source DB instance and begins replication. As a result, you experience a brief I/O suspension on your source DB instance while the DB snapshot occurs. The I/O suspension typically lasts about one minute. You can avoid the I/O suspension if the source DB instance is a Multi-AZ deployment, because in that case the snapshot is taken from the secondary DB instance. An active, long-running transaction can slow the process of creating the Read Replica. We recommend that you wait for long-running transactions to complete before creating a Read Replica. If you create multiple Read Replicas in parallel from the same source DB instance, Amazon RDS takes only one snapshot at the start of the first create action.

When creating a Read Replica, there are a few things to consider. First, you must enable automatic backups on the source DB instance by setting the backup retention period to a value other than 0. This requirement also applies to a Read Replica that is the source DB instance for another Read Replica.

## Promoting a Read Replica to Be a Standalone DB Instance

You can promote a MySQL, MariaDB, Oracle, or PostgreSQL Read Replica into a standalone DB instance. When you promote a Read Replica, the DB instance is rebooted before it becomes available.

There are several reasons you might want to promote a Read Replica to a standalone DB instance:

* **Performing DDL operations (MySQL and MariaDB only)**
* **Sharding**
* **Implementing failure recovery**

The following steps show the general process for promoting a Read Replica to a DB instance:

1. Stop any transactions from being written to the Read Replica source DB instance, and then wait for all updates to be made to the Read Replica. Use the [Replica Lag](http://aws.amazon.com/rds/faqs/#105) metric to determine when all updates have been made to the Read Replica.
2. For MySQL and MariaDB only: If you need to make changes to the MySQL or MariaDB Read Replica, you must set the read\_only parameter to 0 in the DB parameter group for the Read Replica.
3. Promote the Read Replica by using the **Promote Read Replica** option on the Amazon RDS console, the AWS CLI command [promote-read-replica](https://docs.aws.amazon.com/cli/latest/reference/rds/promote-read-replica.html), or the [PromoteReadReplica](https://docs.aws.amazon.com/AmazonRDS/latest/APIReference/API_PromoteReadReplica.html)Amazon RDS API operation.

## Creating a Read Replica in a Different AWS Region

With Amazon RDS, you can create a MariaDB, MySQL, or PostgreSQL Read Replica in a different AWS Region than the source DB instance. You create a Read Replica to do the following:

* Improve your disaster recovery capabilities.
* Scale read operations into an AWS Region closer to your users.
* Make it easier to migrate from a data center in one AWS Region to a data center in another AWS Region.

### Cross-Region Replication Considerations

All of the considerations for performing replication within an AWS Region apply to cross-region replication. The following extra considerations apply when replicating between regions:

* You can only replicate between regions when using Amazon RDS DB instances of MariaDB, PostgreSQL (versions 9.4.7 and 9.5.2 and later), or MySQL 5.6 and later.
* A source DB instance can have cross-region Read Replicas in multiple regions.
* You can only create a cross-region Amazon RDS Read Replica from a source Amazon RDS DB instance that is not a Read Replica of another Amazon RDS DB instance.
* Within an AWS Region, all cross-region Read Replicas created from the same source DB instance must either be in the same Amazon VPC or be outside of a VPC.
* You can create a cross-region Read Replica in a VPC from a source DB instance that is in a VPC in another region. You can also create a cross-region Read Replica in a VPC from a source DB instance that is not in a VPC. You can also create a cross-region Read Replica that is not in a VPC from a source DB instance that is in a VPC.
* Due to the limit on the number of access control list (ACL) entries for a VPC, we can't guarantee more than five cross-region Read Replica instances.

### Cross-Region Replication Costs

For MySQL and MariaDB instances, you can reduce your data transfer costs by reducing the number of cross-region Read Replicas that you create. For example, suppose that you have a source DB instance in one AWS Region and want to have three Read Replicas in another AWS Region. In this case, you create only one of the Read Replicas from the source DB instance. You create the other two replicas from the first Read Replica instead of the source DB instance.

# Working with Storage for Amazon RDS DB Instances

To specify how you want your data stored in Amazon RDS, choose a storage type and provide a storage size when you create or modify a DB instance. Later, you can increase the amount or change the type of storage by modifying the DB instance.

In most cases, scaling storage doesn't require any outage and doesn't degrade performance of the server.

# DB Instance Billing for Amazon RDS

Amazon RDS instances are billed based on the following components:

* DB instance hours (per hour)
* Storage (per GiB per month)
* I/O requests (per 1 million requests per month)
* Provisioned IOPS (per IOPS 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** – Pay by the hour for the DB instance hours that you use.
* **Reserved Instances** – Reserve a DB instance for a one-year or three-year term and get a significant discount compared to the on-demand DB instance pricing.

### 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**

This option provides access to a reserved DB instance without requiring an upfront payment. Your No Upfront reserved DB instance bills a discounted hourly rate for every hour within the term, regardless of usage, and no upfront payment is required. This option is only available as a one-year reservation.

**Partial Upfront**

This option requires a part of the reserved DB instance to be paid upfront. The remaining hours in the term are billed at a discounted hourly rate, regardless of usage. This option is the replacement for the previous Heavy Utilization option.

**All Upfront**

Full payment is made at the start of the term, with no other costs incurred for the remainder of the term regardless of the number of hours used.

Size-flexible reserved DB instances can only scale in their instance class type.

# Backing Up and Restoring Amazon RDS DB Instances

# Working With Backups

Amazon RDS creates and saves automated backups of your DB instance. Amazon RDS creates a storage volume snapshot of your DB instance, backing up the entire DB instance and not just individual databases.

Amazon RDS creates automated backups of your DB instance during the backup window of your DB instance. Amazon RDS saves the automated backups of your DB instance according to the backup retention period that you specify. If necessary, you can recover your database to any point in time during the backup retention period.

Automated backups follow these rules:

* Your DB instance must be in the ACTIVE state for automated backups to occur. Automated backups don't occur while your DB instance is in a state other than ACTIVE, for example STORAGE\_FULL.
* Automated backups and automated snapshots don't occur while a copy is executing in the same region for the same DB instance.

You can also back up your DB instance manually, by manually creating a DB snapshot.

The first snapshot of a DB instance contains the data for the full DB instance. Subsequent snapshots of the same DB instance are incremental, which means that only the data that has changed after your most recent snapshot is saved.

You can copy both automatic and manual DB snapshots, and share manual DB snapshots.

## Backup Storage

Your Amazon RDS backup storage for each region is composed of the automated backups and manual DB snapshots for that region. Your backup storage is equivalent to the sum of the database storage for all instances in that region. Moving a DB snapshot to another region increases the backup storage in the destination region.

If you chose to retain automated backups when you delete a DB instance, the automated backups are saved for the full retention period. If you don't choose **Retain automated backups** when you delete a DB instance, all automated backups are deleted with the DB instance. After they are deleted, the automated backups can't be recovered. If you choose to have Amazon RDS create a final DB snapshot before it deletes your DB instance, you can use that to recover your DB instance. Or you can use a previously created manual snapshot. Manual snapshots are not deleted. You can have up to 100 manual snapshots per region.

## Backup Window

Automated backups occur daily during the preferred backup window.

During the automatic backup window, storage I/O might be suspended briefly while the backup process initializes (typically under a few seconds). You might experience elevated latencies for a few minutes during backups for Multi-AZ deployments. For MariaDB, MySQL, Oracle, and PostgreSQL, I/O activity is not suspended on your primary during backup for Multi-AZ deployments, because the backup is taken from the standby. For SQL Server, I/O activity is suspended briefly during backup for Multi-AZ deployments.

If you don't specify a preferred backup window when you create the DB instance, Amazon RDS assigns a default 30-minute backup window.

## Backup Retention Period

the default backup retention period is one day if you create the DB instance using the Amazon RDS API or the AWS CLI. The default backup retention period is seven days if you create the DB instance using the console. You can set the backup retention period to between 0 and 35 days. Setting the backup retention period to 0 disables automated backups. Manual snapshot limits (100 per region) do not apply to automated backups.

**Important**

An outage occurs if you change the backup retention period from 0 to a non-zero value or from a non-zero value to 0.

## Disabling Automated Backups

You might want to temporarily disable automated backups in certain situations; for example, while loading large amounts of data.

**Important**

We highly discourage disabling automated backups because it disables point-in-time recovery. Disabling automatic backups for a DB instance deletes all existing automated backups for the instance. If you disable and then re-enable automated backups, you are only able to restore starting from the time you re-enabled automated backups.

## Enabling Automated Backups

When automated backups are enabled, your RDS instance and database is taken offline and a backup is immediately created.

## Automated Backups with Unsupported MySQL Storage Engines

For the MySQL DB engine, automated backups are only supported for the InnoDB storage engine. Use of these features with other MySQL storage engines, including MyISAM, can lead to unreliable behavior while restoring from backups. Specifically, since storage engines like MyISAM don't support reliable crash recovery, your tables can be corrupted in the event of a crash. For this reason, we encourage you to use the InnoDB storage engine.

## Automated Backups with Unsupported MariaDB Storage Engines

For the MariaDB DB engine, automated backups are only supported with the InnoDB storage engine (version 10.2 and later) and XtraDB storage engine (versions 10.0 and 10.1). Use of these features with other MariaDB storage engines, including Aria, might lead to unreliable behavior while restoring from backups. Even though Aria is a crash-resistant alternative to MyISAM, your tables can still be corrupted in the event of a crash. For this reason, we encourage you to use the XtraDB storage engine.

# Monitoring Amazon RDS

Advice about specific types of metrics follows:

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

## Monitoring Tools

AWS provides various tools that you can use to monitor Amazon RDS. You can configure some of these tools to do the monitoring for you, while some of the tools require manual intervention. We recommend that you automate monitoring tasks as much as possible.

### 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

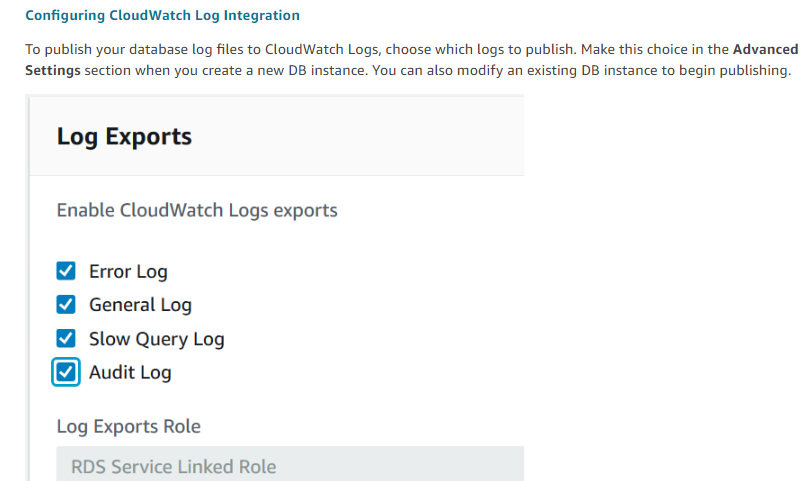
* From the Amazon RDS console, you can monitor the following items for your resources:
  + The number of connections to a DB instance
  + The amount of read and write operations to a DB instance
  + The amount of storage that a DB instance is currently utilizing
  + The amount of memory and CPU being 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

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



# Security in Amazon RDS

Too Lengthy so do it more tutorials dojo

# Best Practices for Amazon RDS

Too Lengthy so do it more tutorials dojo