Monitoring AWS Relational Database Services (RDS)

You’ve got your AWS Cloud data into Splunk Observability Cloud, and now you’re looking to answer some questions as it relates to Amazon’s Relational Database Services (RDS).

Some questions you might have about using Splunk Infrastructure Monitoring to monitor your RDS database instances include:

- How can you keep track of the instances deployed in AWS by class? This is important if you're responsible for system availability, performance, and cost-effectiveness.
- How can you keep control of provisioning database engines that do not adhere to company policy?
- How can you monitor the read performance of RDS database instances and notify the database support team? Degradations can impact your critical applications and end user experience.
- How can you monitor the write performance of RDS instances? Write latency and write throughput are key metrics you need to keep an eye on.
- How can you monitor system performance metrics for each RDS database instance? You need to monitor CPU% utilization, read-write latency, network throughput and disk IOPS to make sure the type of RDS instance has enough capacity provisioned, and is not over-provisioned.

Data required

Database data

About AWS RDS

RDS is an Amazon cloud hosted managed services for databases. This means that Amazon manages most of the management tasks for sustaining the database instances. The application developers can then focus on the application, and the users on providing value to the business.

Each RDS Database instance contains a DB engine that is specific to relational database software. Amazon RDS currently supports MySQL, MariaDB, PostgreSQL, Oracle, and Microsoft SQL Server.

Splunk Infrastructure Monitoring scans every RDS database instance for your AWS accounts and imports the properties of each instance plus any tags set on each instance, as shown in the table below. Engine and engine version properties are included in this table.

<table>
<thead>
<tr>
<th>RDS Name</th>
<th>Custom Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AvailabilityZone</td>
<td>aws_availability_zone</td>
<td>Name of the DB instance</td>
</tr>
<tr>
<td>RDS Name</td>
<td>Custom Property</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Availability Zone</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBClusterIdentifier</td>
<td>aws_db_cluster_identifier</td>
<td>If the DB instance is a member of a DB cluster, contains the name of the DB cluster</td>
</tr>
<tr>
<td>DBInstanceClass</td>
<td>aws_db_instance_class</td>
<td>Name of the compute and memory capacity class of the DB instance</td>
</tr>
<tr>
<td>DBInstanceStatus</td>
<td>aws_db_instance_status</td>
<td>Current state of the DB instance</td>
</tr>
<tr>
<td>Engine</td>
<td>aws_engine</td>
<td>Name of the database engine this DB instance uses</td>
</tr>
<tr>
<td>EngineVersion</td>
<td>aws_engine_version</td>
<td>Database engine version.</td>
</tr>
<tr>
<td>InstanceCreateTime</td>
<td>aws_instance_create_time</td>
<td>DB instance creation date and time</td>
</tr>
<tr>
<td>Iops</td>
<td>aws_iops</td>
<td>New Provisioned IOPS value for the DB instance. AWS might apply this value in the future, or might be applying it at the moment.</td>
</tr>
<tr>
<td>MultiAZ</td>
<td>aws_multi_az</td>
<td>Indicates if the DB instance is a Multi-AZ deployment</td>
</tr>
<tr>
<td>RDS Name</td>
<td>Custom Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PubliclyAccessible</td>
<td>aws_publicly_accessible</td>
<td>Accessibility options for the DB instance. “true” indicates an Internet-facing instance with a publicly resolvable DNS name that resolves to a public IP address. “false” indicates an internal instance with a DNS name that resolves to a private IP address.</td>
</tr>
<tr>
<td>ReadReplicaSourceDBInstanceIdentifier</td>
<td>aws_read_replica_source_db_instance_identifier</td>
<td>If the DB instance is a Read Replica, this value is the identifier of the source DB instance.</td>
</tr>
<tr>
<td>SecondaryAvailabilityZone</td>
<td>aws_second_availability_zone</td>
<td>If this property is present, and the DB instance has multi-AZ support, this value specifies the name of the secondary Availability Zone.</td>
</tr>
<tr>
<td>StorageType</td>
<td>aws_storage_type</td>
<td>Storage type associated with the DB instance</td>
</tr>
</tbody>
</table>

Each DB engine has its own supported features, and each version of a DB engine may include specific features. Additionally, each DB engine has a set of parameters in a DB parameter group that control the behavior of the databases that it manages.

The following image shows the UI when RDS is selected from the AWS Navigation in Splunk Infrastructure Monitoring. You can see a quick view of the instances with the provided heatmap or an instance list.

---

The information provided in Splunk Lantern is intended for informational and educational purposes only. All information is provided in good faith, however, Splunk disclaims any and all representations and warranties, express and implied, regarding the information provided, including without limitation any warranties and representations regarding the completeness, adequacy or accuracy of the information. You agree to take full responsibility for the results arising from the use of the information provided.
You can also group common RDS database instances by various options such as region, state, AWS tag name, and more. In the following image, you can see that the instances have been grouped by AWS region, so you can then see separate heat maps for each AWS region.

You can also hover over any instance in the region and click to drill down deeper and explore more metrics.

Monitoring RDS database services instances

Monitoring the instances you deployed in AWS is essential to ensure system availability, performance, and cost-effectiveness. You might want to keep track of the following:

Number of DB instances

It’s easy to add additional instances, but it’s expensive to leave them running when you don’t need them. For example, someone on your team might have added a read replica to do some analytics jobs, but then forgot to shut it down. Or you might have scripts in Elastic Beanstalk or other places that add new instances automatically or behave unexpectedly. It’s a good idea to set an alert to fire if the number of instances goes beyond a normal level.

Number of DBs by class

One of the easiest ways to scale your database is by increasing the size of your instance. But if you do this too many
times, this will cost you a lot of money for large instances. Keep an eye on these instance types to make sure you’re not overspending. It’s also a good idea to set an alert to fire if the number of large database instance classes exceed a normal level.

Check out this video to see how to create a detector in Splunk Infrastructure Monitoring to monitor RDS database instances by class.

---

**Engine names for compliance**

This property shows how many different engines your team is using. It’s nice to standardize your team on a single engine, but you’ll see it here if a team is using a different database like the newer Aurora.

You should set up an alert to monitor the use of database instances outside of company policy. Check out this video to see how to create a detector that does this using Splunk Infrastructure Monitoring.

---

**Monitoring RDS database performance**

Read performance is important for web applications because content is more often displayed than edited. Read latency can directly impact user experience by making pages faster to load. Because of this, optimizing reads can have a big impact on performance and cost-effectiveness.

The charts you’ll see in Splunk Infrastructure Monitoring which give you metrics for database performance are displayed with percentile distributions. Percentiles are often more useful than averages because outliers can misrepresent the typical performance.

Here are some examples of charts in Splunk Infrastructure Monitoring showing ReadIOPS, ReadLatency and ReadThroughput. The graphs plot the minimum, P10 (or tenth percentile), P50 (otherwise known as median), P90, and maximum. The most prominent color in the screenshot below is pink, which is the maximum.

---

**ReadIOPS**

ReadIOPS shows how many disk read I/O operations per second your database has. They take longer than responses...
that are cached in memory, so you want this number to be as low as possible. If it’s too high, consider adding more RAM to your instances. You can add capacity by switching to SSDs with provisioned IOPS storage.

**ReadLatency**

Read latency is the amount of time it takes to respond to a read request such as a select statement. The lower this is, the faster that pages load and transactions execute. For the most simple reads, you want this to be in the tens of milliseconds. If it’s too high, take a look at your slow query log to determine which queries are taking the longest and then tune them for better performance.

**ReadThroughput**

ReadThroughput shows how many bytes per second your database is reading from disk. Read throughput will be high if your application reads large volumes of data per request, or your responses are heavily cached. It can be lower if you have smaller data sizes, complicated queries that generate high latency, or slow magnetic disks.

---

**Monitoring write performance**

Monitoring write performance is important to make sure that updates are immediately available, or at least not far behind real-time. Also, if your database is heavily indexed, writes can be more resource-intensive. If you make heavy use of the query cache, lots of writes can lower your read latency.

Here are some examples of charts in Splunk Infrastructure Monitoring showing WriteIOps, WriteLatency and WriteThroughput.

---

**WriteIOps**

The average number of disk write I/O operations per second. Your database writes many I/O operations to disk per second. If you need more, consider switching to SSDs with provisioned IOPS.

**WriteLatency**

Write latency is the time it takes to complete a write operation in the database, such as an insert, update, or delete. Low latencies are important for real-time applications. If you use read replicas, also check your ReplicaLag to make sure they are not too far behind.

**WriteThroughput**

The number of bytes per second that you are writing into the database is write throughput. This can be lower if you use table indexes, table locking, foreign key constraints, or slow magnetic disks.
Check out this video to see how to create a detector in Splunk Infrastructure Monitoring to monitor read and write latency for the RDS database instances.

Monitoring RDS system metrics

You can also monitor the EC2 instance supporting the RDS instance to ensure its workload is healthy and sized correctly.

Check out this video to see how to create a detector in Splunk Infrastructure Monitoring to monitor RDS system metrics. You’ll also learn how to create a chart with an event overlay so when CPU utilization is elevated, you can easily see if any alerts such as read or write latency have occurred during the same time frame.

You might want to keep track of the following metrics:

**CPU utilization**

CPU utilization is the total number of CPU units used per RDS instance, expressed as a percentage of the total available. If this metric exceeds about 90 percent for more than a brief period, you could be having a negative impact on your read or write latency. You may want to consider upgrading to larger instances with more CPUs or additional read replicas or shards.

You can learn more about read replicas in [AWS documentation - Working with read replicas](#).

**DatabaseConnections**

This is the number of database connections. Check your instance type for the limit on the number of connections allowed. It’s a good idea to set an alert to fire before you hit the limit. You might want to check the connection pool size in your application servers or add additional capacity.
NetworkReceiveThroughput

It’s important to monitor the number of bytes per second you receive through the network interface. Each instance type has a set capacity for network throughput. Snapshots and replication can also use up your network capacity.

NetworkTransmitThroughput

Your database writes many I/O operations to disk per second. If you need more, consider switching to SSDs with provisioned IOPS.

Best practices on creating and managing detectors

Check out this video which explains some best practices for creating and managing detectors.

• Before developing detectors, spend some time writing down your requirements and expected outcomes.
• Apply good naming conventions to detector names. Configure detectors and alert messages in a standard way. Establish operational practices, policies, and governance around standards.
• Make sure each detector alert has a clear SOP (Standard Operating Procedures) documented. This drives operational excellence.
• Use critical severity sparingly and only under special conditions requiring the highest level of intervention. Consistent standards are also important so that severity levels are interpreted in a consistent way by all consumers.
• Detectors require validation and ongoing tuning to remain relevant. Establish best practices around managing the lifecycle of a detector, from initial creation and onboarding to archiving or decommissioning.
• Make sure detectors are validated and well-tuned before turning on the production operations notifications. Think of assigning operational recipients in Splunk Infrastructure Monitoring as a move to production.
• High quality alerts are key to driving operational excellence and value back to the business.

Next steps

Still having trouble? Splunk has many resources available to help get you back on track. We recommend the following:

• Splunk OnDemand Services: Credit-based services that allow direct access to Splunk technical consultants for a variety of technical services from a pre-defined catalog. Many Splunk customers already have OnDemand credits included as part of their software license. To request OnDemand Services, file a ticket through the Support Portal.
• Splunk Answers: Ask your question to the Splunk Community, which has provided over 50,000 user solutions to date.
• Splunk Customer Support: Contact Splunk to discuss your environment and receive customer support.

To fully unlock the power of Splunk, we strongly recommend our comprehensive Splunk training. If you haven't already,
we recommend you explore the Splunk Observability Training Courses.

Finally, now you're doing more with your AWS data, getting even more value through implementing use cases, or finding out how to get data in from additional data sources.