AWS RDS data model metrics

The goal of this project is to create an independent AWS Lambda driven stack that can be deployed like a plug-n-play stack to gather metrics around the health of data models within a RDS MySQL (or Aurora MySQL) database.

Amazon RDS is a managed database component that provides high availability and scalability out of the box. When an RDS instance is setup, there are a number of RDS related system metrics that AWS automatically generates and pipes into CloudWatch - for example, CPU Utilization, replica-lag, etc. However, Amazon stops its monitoring and metric instrumentation just before getting into the data model layer. That is justifiable, since data models (databases, tables, views, etc) are created by consumers and no metric can provide comprehensive details around this. Also, based on AWS’ shared security model, Amazon cannot (and should not) read into the data models without our permission.

Having said that, certain metrics are extremely useful for to avoid unexpected outages. Out of the box, the Lambda based system that you will build, when setup, would start piping a set of metrics to CloudWatch automatically, given access details around a set of RDS hosts.

One such interesting and useful metric is “auto-increment capacity”. This metric scans all tables within RDS and reports the maximum of the percentages of auto-increment number consumed. This metric will prevent the database from malfunctioning because it has run out of auto increment numbers. Another such interesting metric is “time to disk full”. This metric uses historical data to compute rate of consumption of database space and posts a metric outlining how long do we have till database runs out of space.

# Implementation

1. The Lambda function can be written in any language supported as a Lambda runtime.
2. The Lambda function should accept configurations in form of environment variables.
3. Assume that all RDS host instance that this Lambda requires access to, has the same username and password provided to it as configuration.
4. Feel free to include metrics that you believe are relevant, in addition to the one or more metrics described above.
5. Post configuration, if there are issues running the Lambda, the debugging of the same should be possible by seeing Lambda execution console logs.

## Things to watch out for

1. AWS Resource usage limits.
2. Ease of configuration.
3. Possible errors in any and all applicable API calls.
4. Issues with environment variable configurations provided to Lambda.
5. Extensibility of the Lambda code to add more useful metrics down the line without major code changes.

## Expected solution format

A [GitHub](https://github.com) repository link that contains the necessary AWS Lambda code, with relevant setup instructions.

The ideal setup scenario would be to follow a README on GitHub, which points to downloadable Lambda compatible archive. The archive, when uploaded and configured using environment variables as described in README should automatically start publishing metrics to CloudWatch with a namespace that lies close to the RDS instance being monitored.

It should be possible to debug Lambda execution errors via Lambda execution logs. And, the README should clearly state the purpose and location for each included metric.

### Reference Resources

1. Lambda: <https://aws.amazon.com/lambda/features/>
2. RDS: <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html>
3. CloudWatch: <https://aws.amazon.com/cloudwatch/>
4. Auto increment: <https://dev.mysql.com/doc/refman/8.0/en/example-auto-increment.html>