COSC 416: Topics in Databases (DBaaS)

TOPIC 7: RDS EVENTS AND MONITORING

SCHEDULE

1. *RDS Events*
    1. *Using Event Subscriptions with SNS*
    2. *Using the CLI/SDK to access event data*
2. *Quiz #1 and Lab 1.2 discussion*

USING EVENTS IN RDS

MANAGING STATUS IN RDS

- In our lab, we've used the technique of repeatedly sending status requests using the SDK to determine when an instance is fully created and available, or fully deleted
- This works, but making repeated requests until status changes is less than ideal – isn't there a way we could get Amazon to just tell us when the instance is available?

AMAZON EVENTS

- It is for this reason that Amazon Events exist
- Most Amazon AWS services will generate events that indicate when resources have a significant change in status
- We can both query the events as a log based on time (i.e. get events that occurred in the last 12 hours), or we can set up subscriptions that will provide notification when an event occurs

RDS-SPECIFIC EVENTS

- There are many RDS-specific events, ranging from basic availability status changes (such as restarts) to events specifying that a DB instance has failed or that a recovery is taking place
- In addition, events can also be grouped by type – for example, events that are related to an instance versus events that are related to a snapshot

VIEWING A LOG OF RECENT EVENTS

- In the AWS GUI, while in the RDS dashboard, you can click into the "Events" tab to see what events have recently occurred. For example, this is the output after creating a new instance.

Events (5)

Filter events

Source	Type	Message
database-1	Instances	Finished DB Instance backup
rds:database-1-2020-01-29-21-15	Snapshots	Automated snapshot created
rds:database-1-2020-01-29-21-15	Snapshots	Creating automated snapshot
database-1	Instances	Backing up DB instance
database-1	Instances	DB instance created

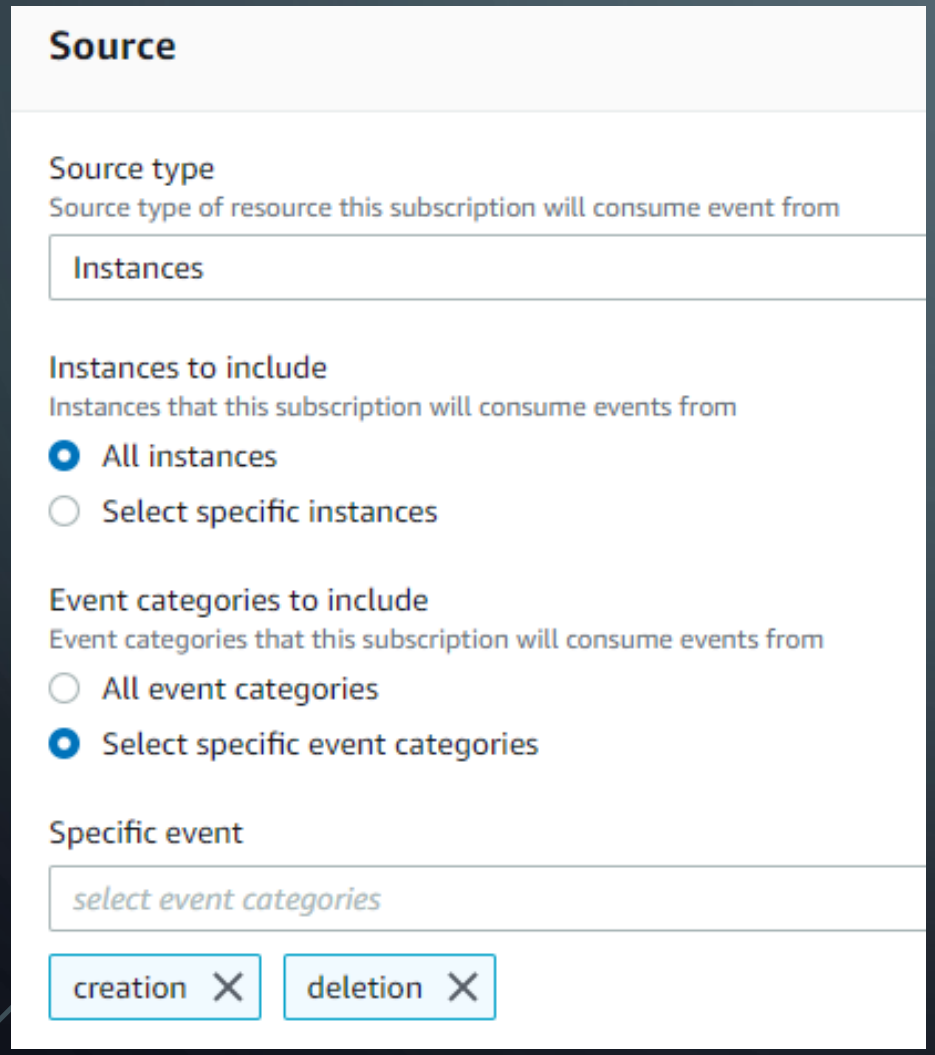
LISTENING IN TO EVENTS

- We can subscribe to events using a service known as the Amazon Simple Notification Service
- We can do this either through the “Event subscriptions” page in the RDS dashboard, or by going to the dedicated SNS (Simple Notification Service) dashboard
- Using the RDS dashboard is simpler

VIA THE RDS DASHBOARD

- Under the “Event subscriptions” page, you can create a new subscription
- You’ll be asked to provide a name, an ARN (Amazon resource, don’t worry about this for now) or email topic (for email notifications)
- You will also be asked for one or more source types (type of events this subscription should consume, i.e. Instance-type events, of one or more event categories like creation or availability)

AN EXAMPLE



- Here, we can see that our subscription will target **creation and deletion** events that occur on an **Instance** object in RDS
- You can also monitor events on a specific instance

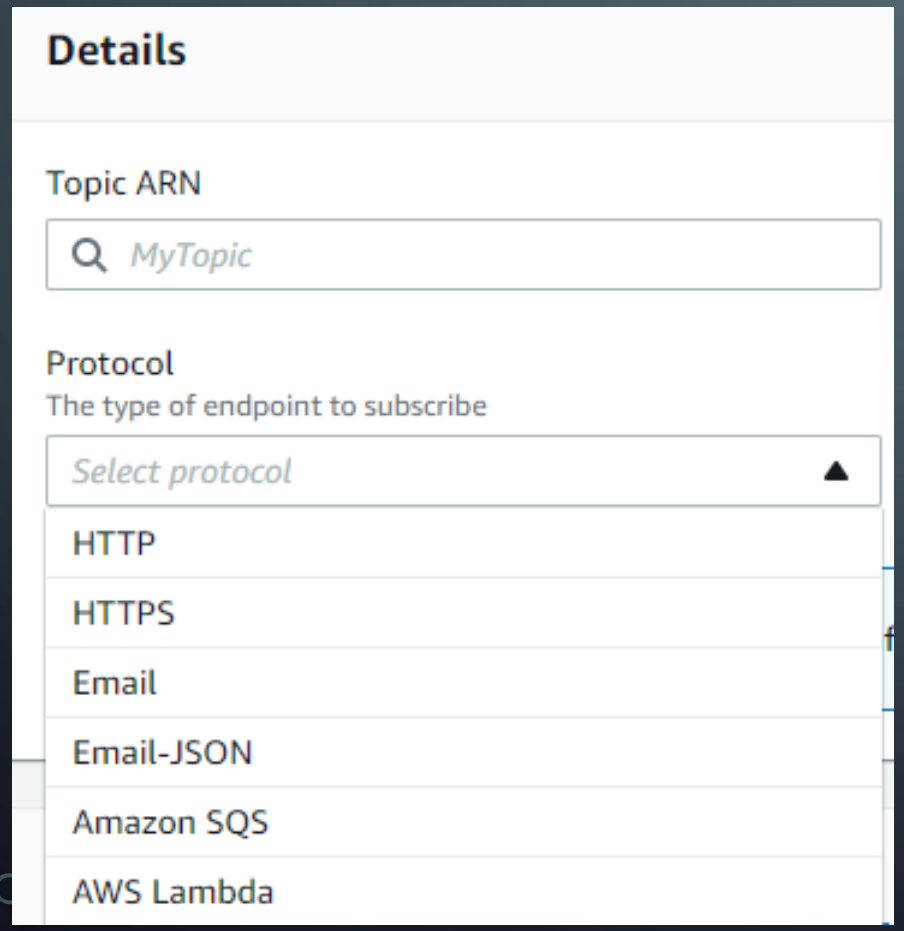
FINER GRAINED CONTROL

- By default, the RDS Event subscription creation will allow you to send email notifications, or redirect the subscribed events to another ARN (another Amazon instance)
- We can get finer control over how we manage this subscription directly through the Simple Notification Service (SNS) dashboard in AWS

TOPICS AND SUBSCRIPTIONS

- In the SNS dashboard, we'll see that our subscription is broken into two components: a topic, and a subscription
- The topic has an ARN associated with it, and is itself associated with an AWS service that it receives events from (such as an RDS instance)
- The subscription handles how events from this topic are communicated to us

CREATING AN SNS SUBSCRIPTION



- When creating a new SNS subscription, we'll be prompted for our topic ARN
- We can also select a protocol – we have access to a wider list of protocols here than in the RDS dashboard

SUBSCRIPTION PROTOCOL OPTIONS

- SNS supports a variety of notification methods for events. The major ones are:
  - Email, either in a plaintext format or as JSON-encoded data
  - HTTP/HTTPS request with data encoded in the body
  - In some regions, data can also be sent via SMS (text message)
- A variety of notification options that could be used to support many different architectures and scenarios

AN EXAMPLE SUBSCRIPTION EMAIL

RDS <no-reply@sns.amazonaws.com>

to me

Event Source : db-instance

Identifier Link: https://console.aws.amazon.com/rds/home?region=us-east-2#dbinstance:id=database-2

SourceId: database-2

Notification time : 2020-01-29 22:18:24.727

Message : DB instance created

Event ID : http://docs.amazonwebservices.com/AmazonRDS/latest/UserGuide/USER_Events.html#RDS-EVENT-0005

- An example create event, captured via email

POSSIBLE USAGE SCENARIOS

- Consider we are building a web application that relies on being able to create and destroy RDS instances
- Rather than tie up the server repeatedly querying to see if an instance has been created, we could just use a webhook that catches an HTTP/HTTPS request from SNS
- No more loops or continuous SDK requests – we can just wait for Amazon to notify us via the SNS service instead!

COST OF THE SNS SERVICE

- Unfortunately, there is one major downside to using the SNS service in Amazon to manage our events: namely, it costs money
- However, there is a free tier available, with up to:
  - 1 million mobile push notifications
  - 100 SMS messages
  - 1,000 email messages
  - 100,000 HTTP/HTTPS messages
- A sneaky note – Amazon considers one 64KB chunk to be 1 request, so larger packets will be considered multiple requests for the purposes of pricing!

HANDLING EVENTS WITH THE CLI/SDK

- An alternative to using the SNS service is monitoring the events ourselves using the Amazon CLI or SDK
- For example, you could fairly easily build your own monitoring application that logs all events and performs some action if certain events occur (such as a failure or a failover)
- This is entirely free, as it doesn't involve the SNS service

THE DESCRIBE_EVENTS FUNCTION

- In the AWS SDK, we can use the describe_events() function to get event information
- We can use parameters to fetch specific types/categories of events, as well as getting events within certain timeframes, or for certain durations
- It will, in turn, return a JSON-encoded output of events that have occurred

SOURCE TYPE PARAMETERS

- Describe_events() can take a source identifier and source type
- SourceType can be one of:
  - ‘db-instance’
  - ‘db-parameter-group’
  - ‘db-security-group’
  - ‘db-snapshot’
  - ‘db-cluster’
  - ‘db-cluster-snapshot’
- Allows you to define what type of events you are want to retrieve. If not provided, all types of events will be returned

SOURCE IDENTIFIER PARAMETERS

- You may also wish to specify that you only want to see events for a particular instance or snapshot. You can use the SourceIdentifier parameter to pass in the unique identifier of an instance, snapshot, or group
- I.e.:

  SourceIdentifier='my-first-instance'

- In this case, only events related to the my-first-instance RDS instance will be retrieved

MANAGING TIME

- We have three major parameters we can use for to specify a time range to retrieve events for:
  - StartTime and EndTime, which take an ISO8601 formatted date for a beginning and ending cutoff time i.e.

    StartTime = `2020-01-29T18:00Z`
  - Duration, which takes an integer number of minutes, and will retrieve events that occurred within that duration (i.e. simply setting a duration of 60 will retrieve events from the last hour)

FILTERING EVENTS BASED ON CATEGORY

- Finally, we can use the EventCategories parameter to define a list of one or more categories of events we would like to retrieve (default: retrieve all categories)
- A full reference to the categories available can be found at this link:

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_Events.html#USER_Events.Messages

TAKEAWAY ON EVENTS

- We have several options for managing events:
  - Built-in SNS subscription service, allowing us to send notifications via Email/HTTP/SMS when an event occurs
  - CLI/SDK describe_events() function, which allows us to retrieve a filtered subset of events that have occurred
- This can be useful for determining the status of an instance (such as when waiting for a created instance to come online), monitoring our instances (notifying us if something goes offline), or simply as a logging tool
- Overall, a more descriptive and useful interface than just requesting the DBInstanceStatus from describe_db_instances()

WRAPPING UP AMAZON RDS

- At this point, we're coming to a close with Amazon RDS
- What we've covered:
  - Creating, destroying, and connecting to RDS instances
  - Managing RDS instances via the CLI/SDK
  - Some of RDS's more advanced features, like read-replicas, encryption, and Aurora clusters
  - Using events and subscriptions to help log and notify us when changes occur to our instances

NEXT UP: MICROSOFT AZURE

- For our next topic, we'll be leaving Amazon behind and checking out Microsoft's Azure platform, and the relational Database-As-A-Service options that it offers
- We'll be coming back to Amazon later with DynamoDB, as an example of a non-relational DBaaS service
- For now, we'll see how the competitors manage a DBaaS service

QUIZ #1 AND LAB 1.2

QUIZ #1 – NEXT WEEK

- We have quiz #1 scheduled for next week, Feb 3-7th
- I'm thinking this will be on Wednesday, but I'm open to a class vote if people have midterms for other courses scheduled that day (alternate option is Monday)
- Topics: DBaaS architecture basics (what makes a DBaaS, a DBaas?), Amazon RDS knowledge
  - I will not ask you to write direct RDS SDK/CLI code or commands, but may ask about the functions that the SDK/CLI provide that we've covered, and what they do

LAB 1.2 POSTED

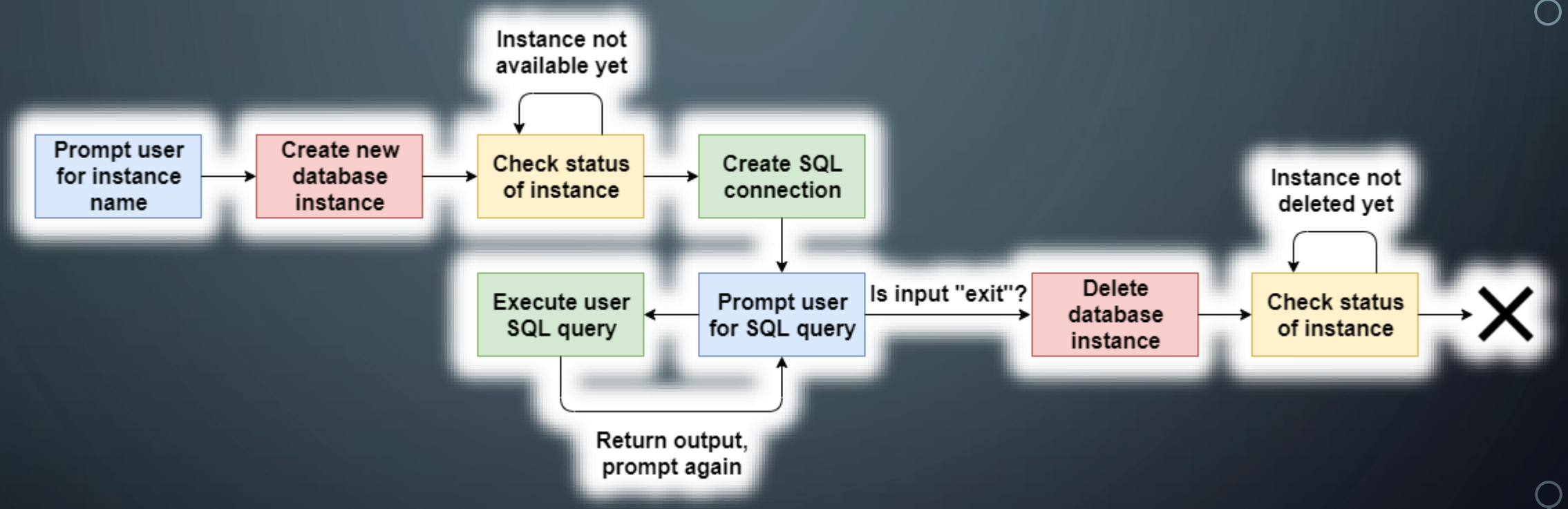
- I have posted up the document for Lab 1.2 on GitHub and Moodle
- Link can be found here:

https://github.com/MattFritter/COSC416-Database-as-a-Service/blob/master/Labs/Lab%201.2%20-%20Amazon%20RDS%20-%20SDK%20Application.md

LAB 1.2 REQUIREMENTS

- You will have two weeks to complete lab 1.2
- I've provided you with code snippets for most of the code required, but it is up to you to piece them together, and implement a final extra function from a list of functions in the lab
- Sample output is provided in the lab – try to make sure your application matches it closely if possible

THE BASIC STRUCTURE OF THE LAB APPLICATION



- Three major loops to work with: creating an instance, passing user SQL input, and deleting an instance

SCHEDULE

1. *RDS Events*
    1. *Using Event Subscriptions with SNS*
    2. *Using the CLI/SDK to access event data*
2. *Quiz #1 and Lab 1.2 discussion*

SO LONG, FOLKS!