Overview

- What is Aurora?
  - Features & Capabilities
- Why consider Aurora?
- The various Aurora HA Setups
- Upsizing / Failover Example
- Aurora specific internals for MySQL architects & admins
- Other Aurora Features and Functionality
About Myself

- 20+ years MySQL experience in architecture and operations
- 15 years conference speaking
- Published author of 4 MySQL books

- Lead Data Architect/Engineer at Lifion by ADP

http://ronaldbradford.com
What is AWS RDS Aurora?

- Amazon Web Services (AWS)
- Relational Database Service (RDS)
  - MySQL/MariaDB/Postgresql/Oracle/SQL Server
- Aurora
  - MySQL and Postgres wire-compatible database built specifically for the AWS cloud

https://aws.amazon.com/rds/aurora
Aurora Features & Capabilities

- AWS managed RDBMS option
- Distributed cloud native architecture
- MySQL/Postgresql wire compatible
- A different transactional storage engine
- A different replication approach (read-free replicas)
- HA/Clustering/failover built-in by default
Aurora Features & Capabilities (2)

- Single writer/multiple readers
  - can support multi-master
- Decoupled compute/storage infrastructure
- Highly durable/redundant storage via quorum
- Log based architecture
- Improved recovery capabilities
- Fast DDL
Aurora Improved Availability, Backup & Recovery

- Fast recovery capabilities (log append design)
- Database cloning
- Snapshot restore
- Backtrack
- Zero Downtime Patching (ZDP)


Aurora Cluster Architecture Features

A cluster has:

- Data in 3 Availability Zones (AZ)
- 2 copies per AZ
- 4 of 6 need for Quorum
- Route 53 Cluster & Instance Endpoints
  - Writer, Reader, Custom (Cluster), Instance
- Automatic Instance failover
- Replica Autoscaling
Aurora Cluster - Single Instance

- Cluster
  - Storage in 3 AZs
  - Writer endpoint
  - Reader endpoint

- Single instance
  - In 1 AZ
  - Endpoint
  - Easily add additional instances

... (Diagram)
Cluster with Single Instance

AWS Region

VPC

Availability Zone 1

Availability Zone 2

Availability Zone 3

Primary

Cluster Volume

Writes

Reads

Amazon Aurora

Cluster with Single Instance
Aurora Cluster - Multiple Instances

- Cluster
- Writer endpoint
  - Primary
- Reader endpoint
  - Load balanced across non primary instance(s)
- Multiple instance(s)
  - AZs of choice
- Promotion Tiers
  ... (Diagram)
Aurora Cluster - Multi-Master

- DB Instances are read & write
  - --engine-mode multimaster

Limitations

- Snapshots / ZDP / Load Balancing / Backtrack / Performance Insights
- Binary Logging
- Certain Datatypes
- Foreign Key CASCADE
- no fast DDL

Multiple Aurora Clusters (1)

- Same region option
- Uses MySQL binary log replication
  - Needs to be enabled
  - GTID not support > 5.7
- Blue/Green deployments
- Shorter downtime upgrades

https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Replication.MySQL.html
Cluster with Single Instance
Two separate clusters
Two separate clusters with binlog replication
Multiple Aurora Clusters Considerations

Source

```sql
mysql> CALL mysql.rds_show_configuration;
mysql> CALL mysql.rds_set_configuration('binlog retention hours', 144);
mysql> CREATE USER 'repl_user'@'<domain_name>' IDENTIFIED BY '<password>';
mysql> GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'<domain_name>';
mysql> GRANT USAGE ON *.* TO 'repl_user'@'<domain_name>' REQUIRE SSL;
```

Target

```sql
# Get position from snapshot restore
$ aws rds describe-events

mysql> CALL mysql.rds_set_external_master (host_name, host_port, replication_user_name, replication_user_password,
            mysql_binary_log_file_name, mysql_binary_log_file_location,
            ssl_encryption);
mysql> CALL mysql.rds_start_replication;
mysql> SHOW SLAVE STATUS;
```
# Get position from snapshot restore
$ aws rds describe-events

```json
{
  "Events": [
    {
      "EventCategories": [],
      "SourceType": "db-instance",
      "Date": "2016-10-28T19:43:46.862Z",
      "Message": "Binlog position from crash recovery is mysql-bin-changelog.000003 4278",
      "SourceIdentifier": "sample-restored-instance"
    }
  ]
}
```
Multiple Aurora Clusters (2)

- Cross-region read replica
  - Support local read latency
- Improved DR
  - Failover not failback
- Region migration path
- Requires binary log replication
- Incurs cross-region transfer costs $$$

https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Replication.CrossRegion.html
Aurora Global Cluster

- One primary region
  - Up to 5 read-only secondary regions
- Uses Aurora storage for replication
  - Lag < 1 second
- RPO = 0
- Blocks writes before failover
- Read-only cluster supports write-forwarding capabilities
Maintenance Situations
Aurora Upgrades

- **In-place upgrades (e.g. 2.09.1 to 2.09.2)**
  - Whole process 5-10 minutes
  - DNS loss 10-20 seconds
  - ZDP (yet to see this work)

- **Minor version (e.g. 2.07.3 to 2.09.2)**
  - Very similar to in-place

- **Major version (e.g. 2.09.2 to ???)**
  - Yet to attempt

https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Updates.MajorVersionUpgrade.html
Aurora Upsizing / Failover

- Instances can be different instance types
  - Read Endpoint moves to Writer during upsize
- Controlled failover
  - Writer endpoint moves to new promoted instance
    - What was writer becomes a reader
- DNS connectivity loss 10-20 seconds
Aurora Upsizing / Failover Commands

```
CLUSTER_ID="demo"
INSTANCE_ID=${CLUSTER_ID}-0
aws rds describe-db-instances --db-instance-identifier ${INSTANCE_ID} | jq -r '.DBInstances[] | [.DBInstanceIdentifier, .DBInstanceClass, .DBInstanceStatus]'
[  "demo-0",  "db.r5.large",  "available" ]
aws rds modify-db-instance --db-instance-identifier ${INSTANCE_ID} --db-instance-class db.r5.4xlarge --apply-immediately
aws rds describe-db-instances --db-instance-identifier ${INSTANCE_ID} | jq -r '.DBInstances[] | [.DBInstanceIdentifier, .DBInstanceClass, .DBInstanceStatus]'
[  "demo-0",  "db.r5.large",  "modifying" ]
aws rds wait db-instance-available --db-instance-identifier ${INSTANCE_ID}
aws rds describe-db-instances --db-instance-identifier ${INSTANCE_ID} | jq -r '.DBInstances[] | [.DBInstanceIdentifier, .DBInstanceClass, .DBInstanceStatus]'
[  "demo-0",  "db.r5.4xlarge",  "available" ]

# Failover
aws rds describe-db-clusters --db-cluster-identifier ${CLUSTER_ID} | jq '.DBClusters[].DBClusterMembers'
aws rds failover-db-cluster --db-cluster-identifier ${CLUSTER_ID}
aws rds describe-db-clusters --db-cluster-identifier ${CLUSTER_ID} | jq '.DBClusters[].DBClusterMembers'
```
Aurora Upsizing / Failover Monitoring

# Endpoints
CLUSTER_ID="demo"
INSTANCE_ID="${CLUSTER_ID}-0"
aws rds describe-db-clusters --db-cluster-identifier ${CLUSTER_ID} | jq '.DBClusters[].DBClusterMembers'

# Cluster Status
while : ; do date ; aws rds describe-db-instances --db-instance-identifier ${INSTANCE_ID} | jq -r '.DBInstances[] | [.DBInstanceIdentifier, .DBInstanceClass, .DBInstanceStatus]'; sleep 5; done

# Instance endpoint availability (goes down during upsize)
MYSQL_HOST=$(aws rds describe-db-instances --db-instance-identifier ${INSTANCE_ID} | jq -r '.DBInstances[0].Endpoint.Address');
echo $MYSQL_HOST
while : ; do [ -n "${MYSQL_PASSWD}" ] && date; time mysql -h ${MYSQL_HOST} -u${MYSQL_USER} -p${MYSQL_PASSWD} -An --connect-timeout=1 -e "SELECT NOW(),@@aurora_server_id, variable_value from information_schema.global_status where variable_name='uptime'"; sleep 1; done

# Cluster reader endpoint (fails over for new connections)
MYSQL_HOST=$(aws rds describe-db-clusters --db-cluster-identifier ${CLUSTER_ID} | jq -r '.DBClusters[0].ReaderEndpoint'); echo $MYSQL_HOST
while : ; do [ -n "${MYSQL_PASSWD}" ] && date; time mysql -h ${MYSQL_HOST} -u${MYSQL_USER} -p${MYSQL_PASSWD} -An --connect-timeout=1 -e "SELECT NOW(),@@aurora_server_id, variable_value from information_schema.global_status where variable_name='uptime'"; sleep 1; done
## Aurora Upsizing / Failover Timing Example

<table>
<thead>
<tr>
<th>Status</th>
<th>Start Time</th>
<th>End Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>status=available</strong></td>
<td>17:30:01 EDT 2021</td>
<td>18:05:12 EDT 2021</td>
</tr>
<tr>
<td><strong>status=modifying</strong></td>
<td>17:30:02 EDT 2021</td>
<td>18:05:19 EDT 2021</td>
</tr>
<tr>
<td>Reads flip to writer endpoint</td>
<td>17:32:48 UTC 2021</td>
<td>18:07:10 EDT 2021</td>
</tr>
<tr>
<td>Lose reader access</td>
<td>17:33:13 EDT 2021</td>
<td>18:07:42 EDT 2021</td>
</tr>
<tr>
<td>Accessible reader instance</td>
<td>17:37:33 EDT 2021 Uptime 19s</td>
<td>18:12:42 EDT 2021 Uptime 18s</td>
</tr>
<tr>
<td><strong>status=modifying</strong></td>
<td>17:40:35 EDT 2021</td>
<td>18:14:46 EDT 2021</td>
</tr>
<tr>
<td><strong>status=storage-optimization</strong></td>
<td>17:41:40 EDT 2021</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>status=available</strong></td>
<td>17:53:53 EDT 2021</td>
<td>18:16:15 EDT 2021</td>
</tr>
</tbody>
</table>
Aurora Upsizing / Failover Graphs (CPU example)

First upsize

Second upsize
Other Topics (for another time)
Additional RDS/Aurora Capabilities

- IAM Authentication for users
- Aurora Query Cache
- Aurora Parallel Query
- Aurora Monitoring
- DMS source & target
  - Replicate to/from RDS to RDS/Redshift/Kinesis etc
- Database Activity Streams
  - CDC to Kinesis
- Aurora specific tuning (binlog)
- RDS Proxy
- Autoscaling (ASG) read replicas
- ...

[https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/MonitoringAurora.html](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/MonitoringAurora.html)
[https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/DBActivityStreams.html](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/DBActivityStreams.html)
[https://aws.amazon.com/rds/proxy/](https://aws.amazon.com/rds/proxy/)
Aurora Serverless

- For development & integration non 24x7 environments
- Cost versus performance benefits
- V1
- V2 (preview)

Chaos Aurora

SHOW VOLUME STATUS;

ALTER SYSTEM CRASH [ INSTANCE | DISPATCHER | NODE ];

ALTER SYSTEM SIMULATE percentage_of_failure PERCENT READ REPLICA FAILURE

[ TO ALL | TO "replica name" ]

FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE | SECOND };

ALTER SYSTEM SIMULATE percentage_of_failure PERCENT DISK FAILURE

[ IN DISK index | NODE index ]

FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE | SECOND };

ALTER SYSTEM SIMULATE percentage_of_failure PERCENT DISK CONGESTION

BETWEEN minimum AND maximum MILLISECONDS

[ IN DISK index | NODE index ]

FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE | SECOND };}
Aurora under the hood

Quorums


Conclusion
Conclusion

- Managed services helps less resourced teams
- Monitoring cost is important
- Review performance between native/ec2/rds/aurora MySQL installations
- With managed services, some existing actions are limited/restricted
- HA infrastructure/ failover / upgrades are built-in capabilities

Slides:
http://ronaldbradford.com/blog/understanding-aws-rds-aurora-capabilities-2021-05-13/