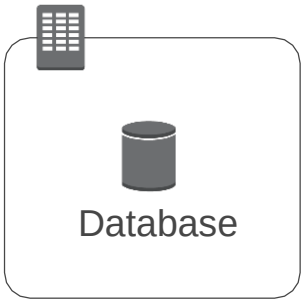
Getting Started with Amazon Aurora

What to expect from the session

- Introduction to Amazon RDS
- Why AWS built Aurora
- Customer adoption
- Aurora features

Options for hosting databases

Self-managed

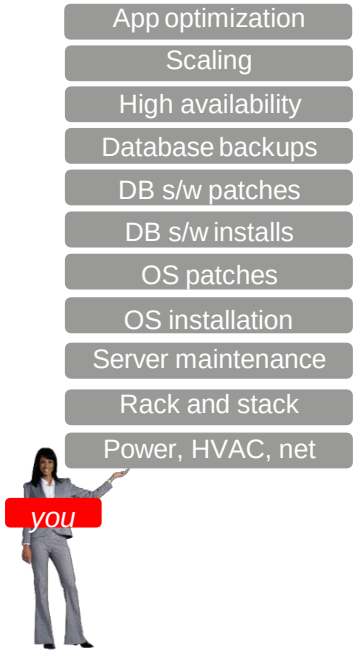


Corporate data center

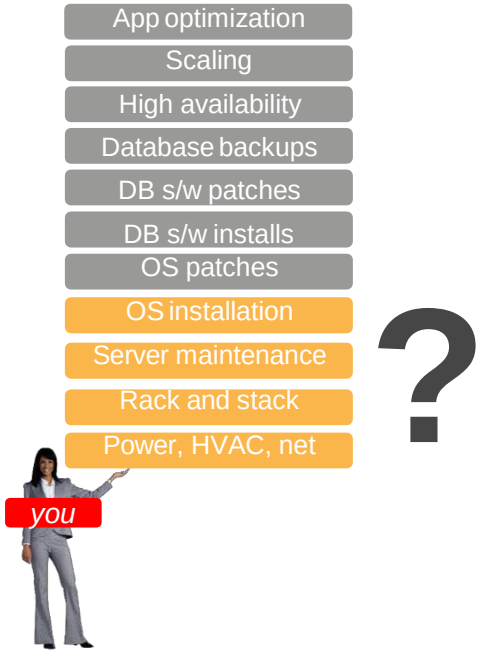
Amazon EC2 instances

Fully managed

If you host your databases on premises



If you host your databases on premises



If you host your databases in EC2

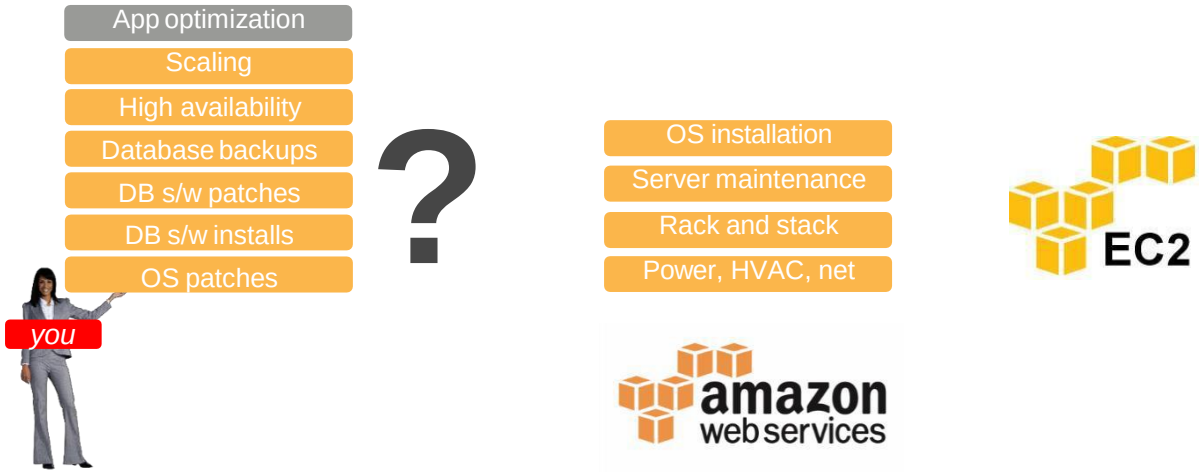
App optimization
Scaling
High availability
Database backups
DB s/w patches
DB s/w installs
OS patches

OS installation
Server maintenance
Rack and stack
Power, HVAC, net

amazon web services

EC2

If you host your databases in EC2



If you choose a managed database service

Scaling
High availability
Database backups
DB s/w patches
DB s/w installs
OS patches
OS installation
Server maintenance
Rack and stack
Power, HVAC, net

amazon web services

Amazon RDS

EC2

Amazon RDS

Relational databases

Fully managed and secure

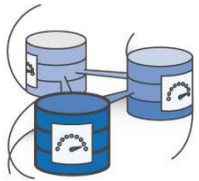
Fast, predictable performance

Simple and fast to scale

Low cost, pay for what you use

MySQL

PostgreSQL



Amazon Aurora

MariaDB

Microsoft SQL Server

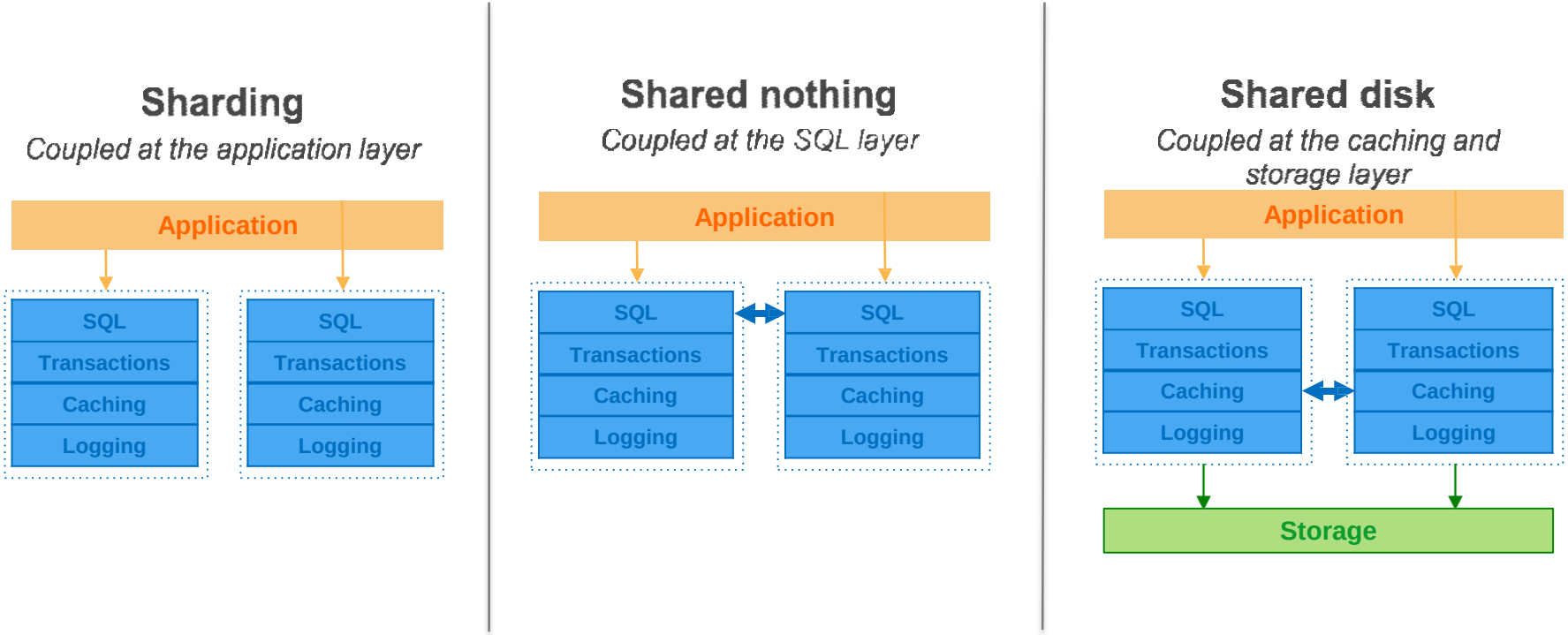
ORACLE

Why AWS built Amazon Aurora

- ☑ **Speed** and **availability** of high-end commercial databases
- ☑ **Simplicity** and **cost-effectiveness** of open source databases
- ☑ Drop-in **compatibility** with MySQL
- ☑ Simple **pay as you go** pricing

Delivered as a **managed** service

Database architectures in last 30 years



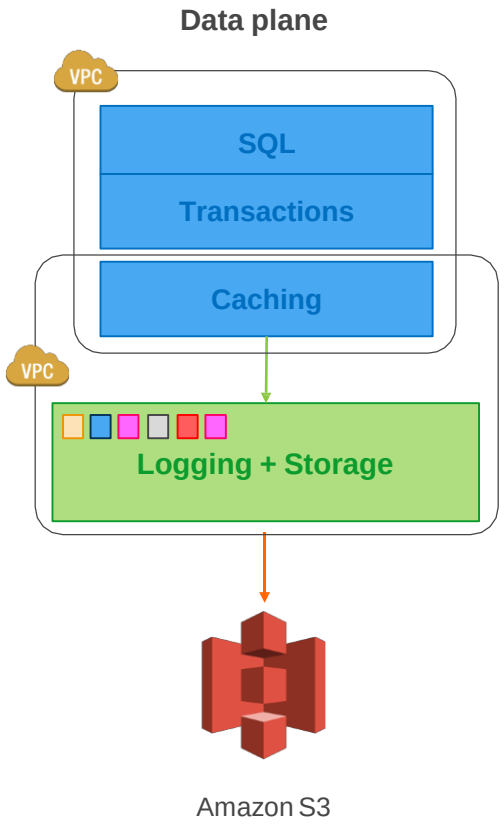
Even when you scale it out, you're still replicating the same stack

A service-oriented architecture applied to the database

1. Moved the logging and storage layer into a multitenant, scaled-out database-optimized storage service

2. Integrated with other AWS services like Amazon EC2, Amazon VPC, Amazon DynamoDB, Amazon SWF, and Amazon Route 53 for control plane operations

3. Integrated with Amazon S3 for continuous backup with 99.999999999% durability



Rapid adoption of Amazon Aurora

Aurora customer adoption

Fastest growing service

in AWS history

Expedia

Pearson

Alfresco

ISCS

zynga

zendesk

GE Oil & Gas

Expedia: On-line travel marketplace

World's leading online travel company with a portfolio that includes 150+ travel sites in 70 countries.

- Real-time business intelligence and analytics on a growing corpus of online travel marketplace data.
- Current Microsoft SQL Server–based architecture is too expensive. Performance degrades as data volume grows.
- Cassandra with Solr index requires large memory footprint and hundreds of nodes, adding cost.

Aurora benefits:

- Aurora meets scale and performance requirements with much lower cost.
- 25,000 inserts/sec with peak up to 70,000. 30 ms average response time for write and 17 ms for read, with 1 month of data.

ISCS: Insurance claims processing

Provides policy management, claim, and billing solutions for casualty and property insurance organizations.

- Have been using Oracle and SQL Server for operational and warehouse data.
- Cost and maintenance of traditional commercial database has become the biggest expenditure and maintenance headache.

Aurora benefits:

- The cost of a “more capable” deployment on Aurora has proven to be about 70% less than ISCS’s SQL Server deployments.
- Eliminated backup window with Aurora’s continuous backup; exploiting linear scaling with number of connections; continuous upload to Amazon Redshift using Aurora Replicas.

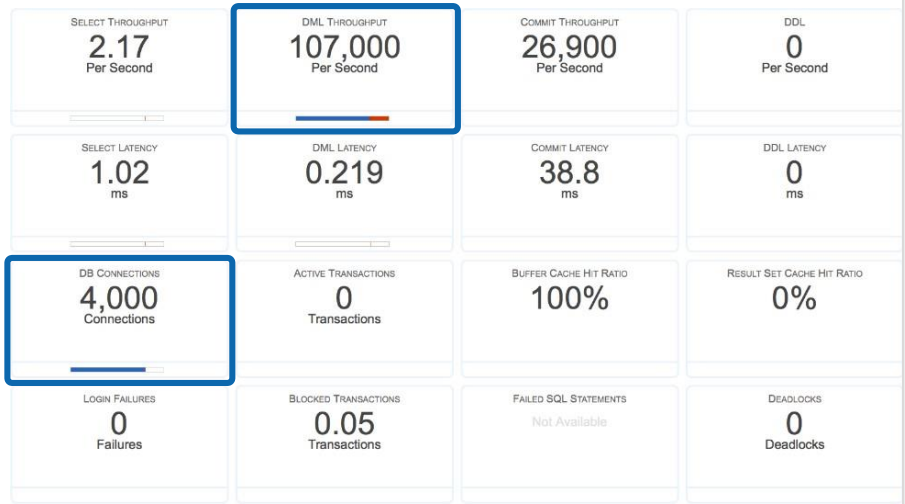
Amazon Aurora is fast

“When we ran Alfresco’s workload on Aurora, **we were blown away to find that Aurora was 10 times faster than our MySQL environment**,” said John Newton, founder and CTO of Alfresco. “Speed matters in our business, and Aurora has been faster, cheaper, and considerably easier to use than MySQL.”

SQL benchmark results

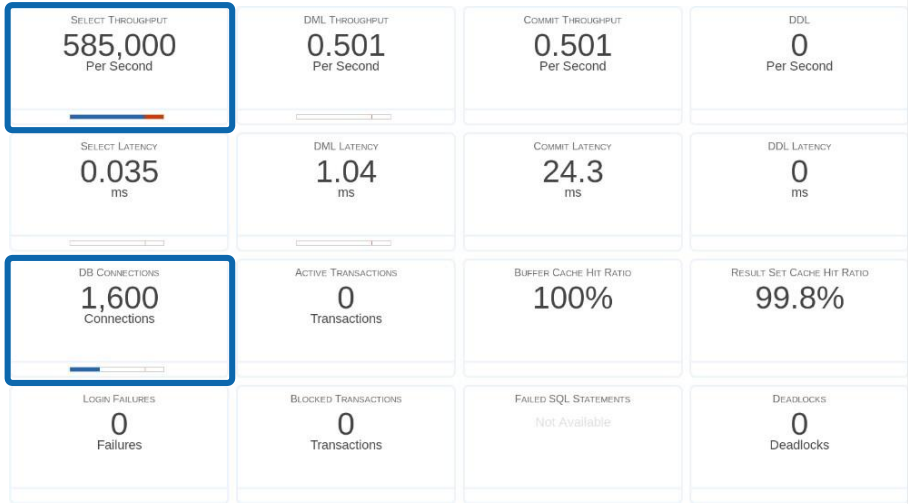
- MySQL SysBench
- R3.8XL with 32 cores and 244 GB RAM

WRITE PERFORMANCE



- Four client machines with 1,000 threads each

READ PERFORMANCE



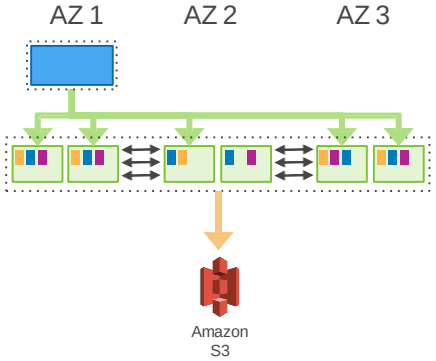
- Single client with 1,600 threads

Aurora performance

- Writes scale with connection count
- Consistent performance as table count increases
- Consistent performance with growing datasets
- Minimal replica lag with high update frequency on master (~5.38 ms with 10k updates per second)

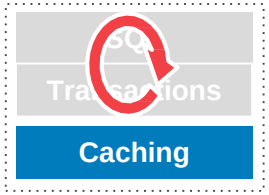
Amazon Aurora is highly available

Amazon Aurora is highly available



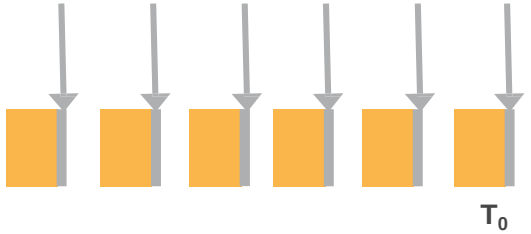
Highly available storage

- Six copies of data across three AZs
- Latency tolerant quorum system for read/write
- Up to 15 replicas with low replication lag



Survivable caches

- Cache remains warm in the event of a database restart
- Lets you resume fully loaded operations much faster



Instant crash recovery

- Underlying storage replays redo records on demand as part of a disk read
- Parallel, distributed, asynchronous

Choose cross-region read replicas for faster disaster recovery and enhanced data locality

Promote a read replica to a master for faster recovery in the event of disaster

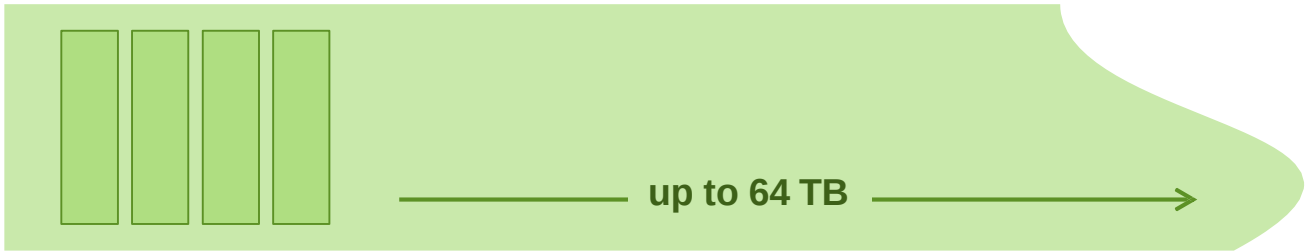
Bring data close to your customer's applications in different regions

Promote to a master for easy migration

Amazon Aurora is easy to use

“Amazon Aurora’s new user-friendly monitoring interface made it easy to diagnose and address issues. Its performance, reliability, and monitoring really shows Amazon Aurora is an enterprise-grade AWS database.” —Mohamad Reza, information systems officer at United Nations

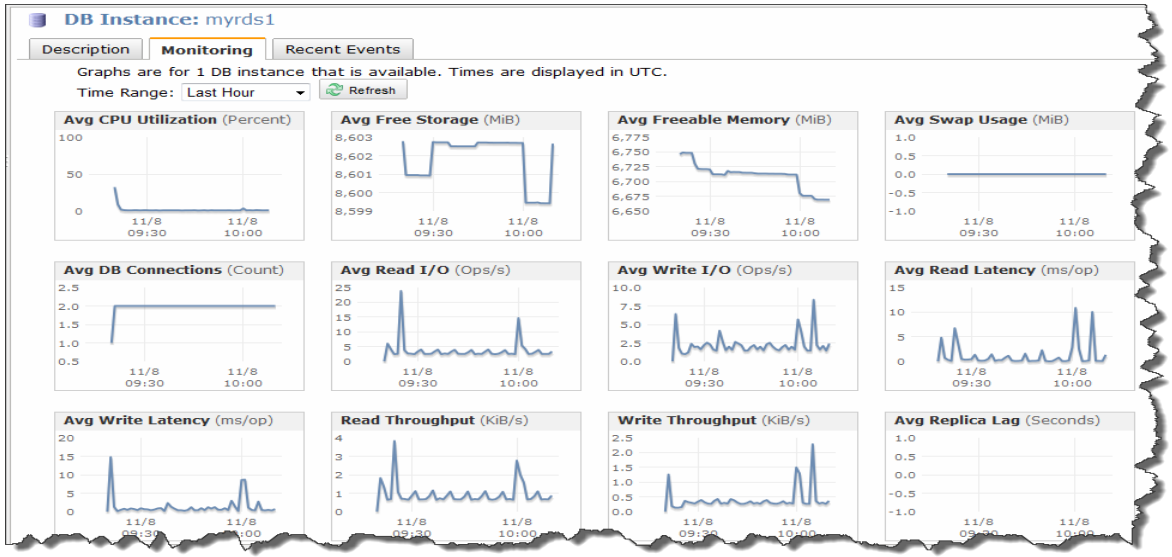
Simplify storage management



Up to 64 TB of storage—autoincremented in 10 GB units

- Automatic storage scaling up to 64 TB—no performance impact
- Continuous, incremental backups to Amazon S3
- Instantly create user snapshots—no performance impact
- Automatic restriping, mirror repair, hot spot management, encryption

Simplify monitoring with AWS Management Console



Amazon CloudWatch metrics for RDS

- CPU utilization
- Storage
- Memory
- 50+ system/OS metrics
- 1–60 second granularity
- DB connections
- Selects per second
- Latency (read and write)
- Cache hit ratio
- Replica lag

CloudWatch alarms

- Similar to on-premises custom monitoring tools

Simplify data security

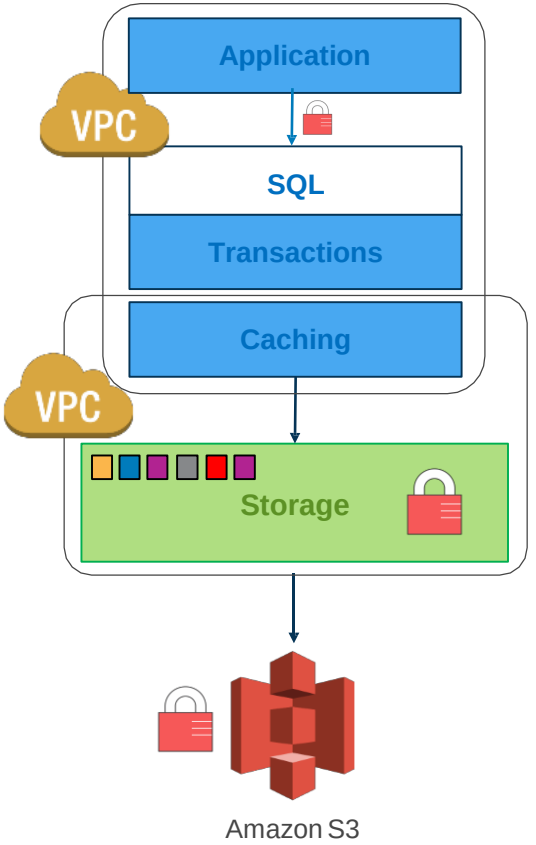
☑ Encryption to secure data at rest
- AES-256; hardware accelerated
- All blocks on disk and in Amazon S3 are encrypted
- Key management by using AWS KMS

☑ SSL to secure data in transit

☑ Network isolation by using Amazon VPC by default

☑ No direct access to nodes

☑ Supports industry standard security and data protection certifications



Migration to Aurora is easy

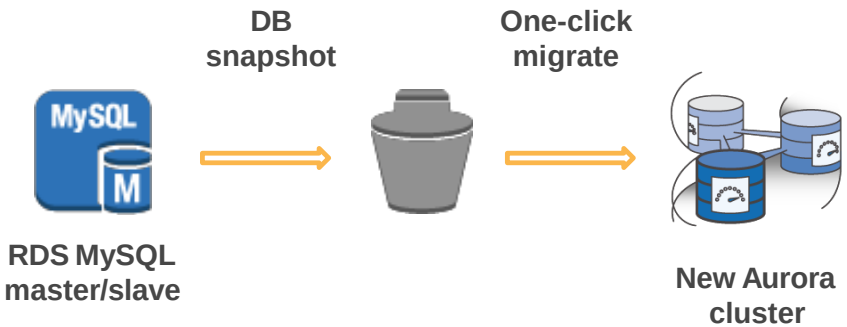
Migration from MySQL

Source database on RDS

- Snapshot migration: One-click migration from RDS MySQL 5.6 to Aurora

Source database external or on EC2

- Use native MySQL migration tools
- Back up to S3 using Percona XtraBackup, restore from S3



RDS snapshot migration

AWS Database Migration Service

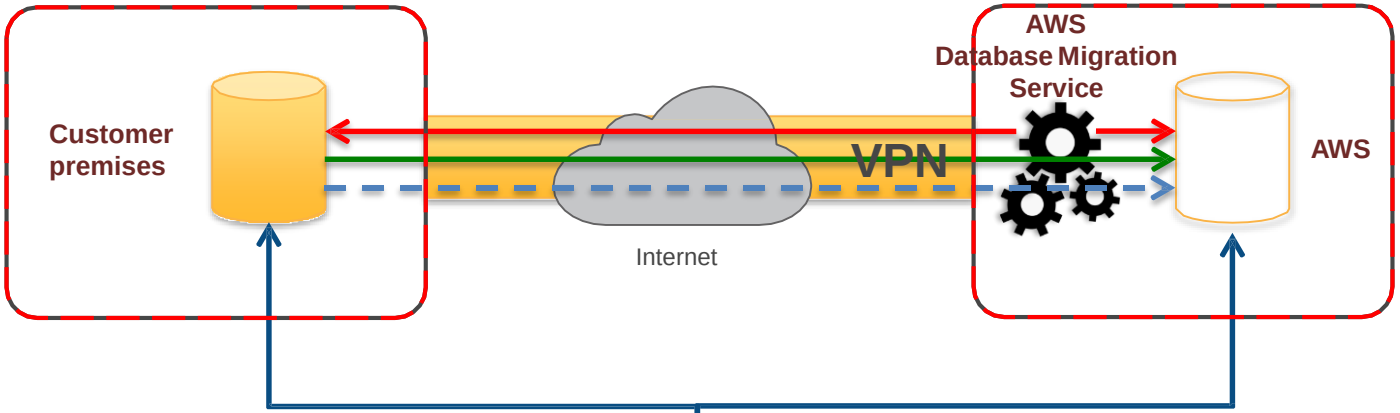
Start your first migration in 10 minutes or less

Keep your apps running during the migration

Replicate within, to, or from Amazon EC2 or RDS

Move data to the same or different database engine

Keep your apps running during the migration



Start a replication instance

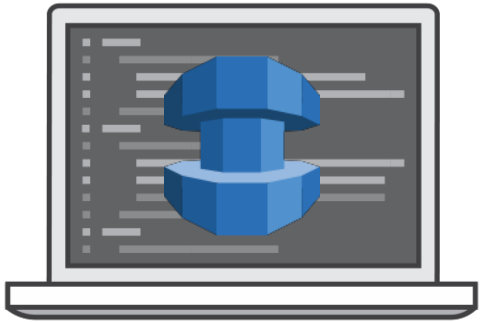
Connect to source and target databases

Select tables, schemas, or databases

Application users

Let AWS Database Migration Service create tables, load data, and keep them in sync

Switch applications over to the target at your convenience



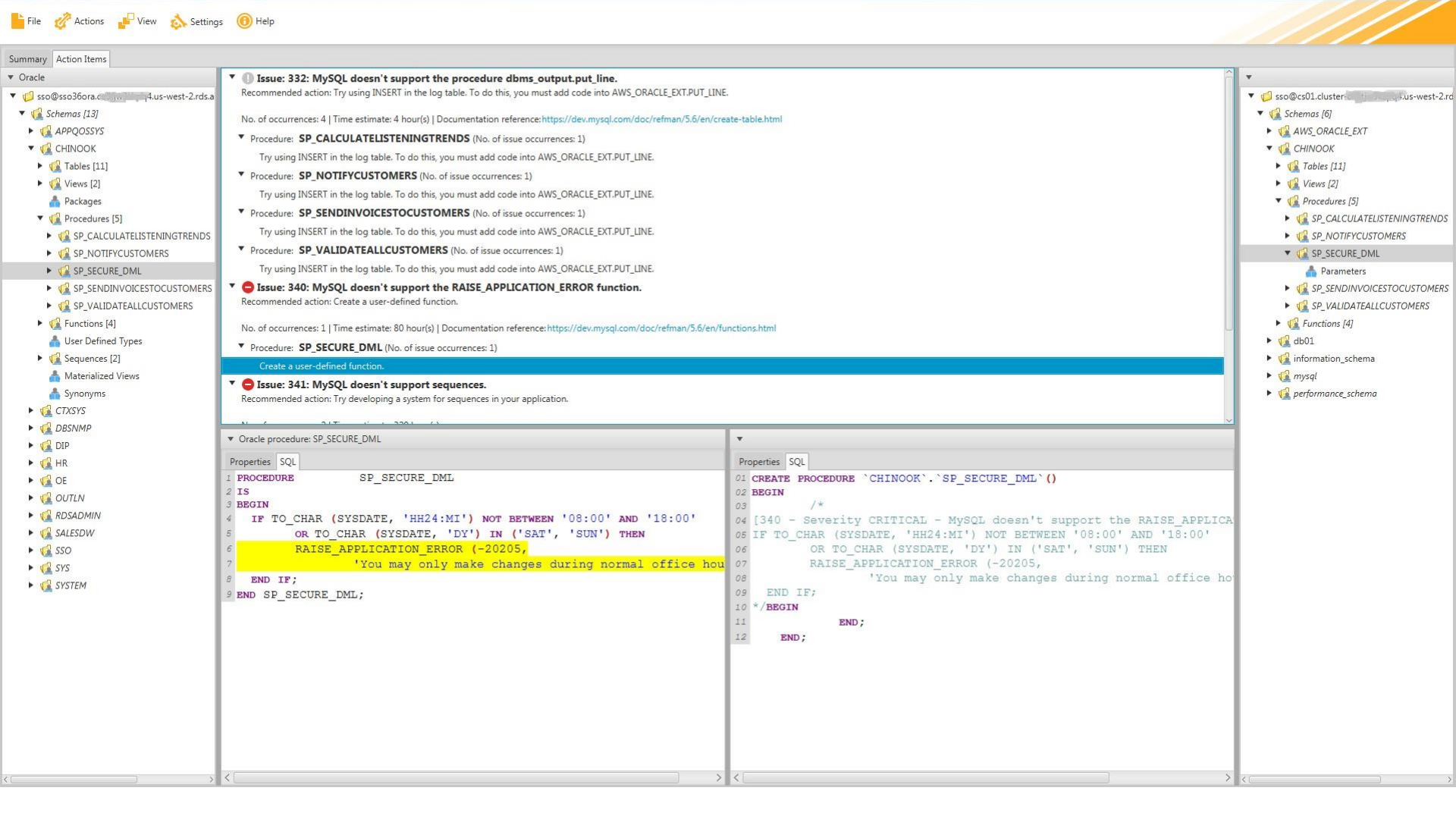
AWS Schema Conversion Tool

Migrate from Oracle and SQL Server

Move your tables, views, stored procedures, and data manipulation language (DML) to MySQL, MariaDB, and Amazon Aurora

Know exactly where manual edits are needed

Download at aws.amazon.com/dms


File
Actions
View
Settings
Help
Summary
Action Items
Oracle
sso@sso36ora.c[redacted]4.us-west-2.rds.a
Schemas [13]
APPQOSSYS
CHINOOK
Tables [11]
Views [2]
Packages
Procedures [5]
SP_CALCULATELISTENINGTRENDS
SP_NOTIFYCUSTOMERS
SP_SECURE_DML
SP_SENDINVOICESTOCUSTOMERS
SP_VALIDATEALLCUSTOMERS
Functions [4]
User Defined Types
Sequences [2]
Materialized Views
Synonyms
CTXSYS
DBSNMP
DIP
HR
OE
OUTLN
RDSADMIN
SALESDW
SSO
SYS
SYSTEM
Issue: 332: MySQL doesn't support the procedure dbms_output.put_line.
Recommended action: Try using INSERT in the log table. To do this, you must add code into AWS_ORACLE_EXT.PUT_LINE.
No. of occurrences: 4 | Time estimate: 4 hour(s) | Documentation reference:https://dev.mysql.com/doc/refman/5.6/en/create-table.html
Procedure: SP_CALCULATELISTENINGTRENDS (No. of issue occurrences: 1)
Try using INSERT in the log table. To do this, you must add code into AWS_ORACLE_EXT.PUT_LINE.
Procedure: SP_NOTIFYCUSTOMERS (No. of issue occurrences: 1)
Try using INSERT in the log table. To do this, you must add code into AWS_ORACLE_EXT.PUT_LINE.
Procedure: SP_SENDINVOICESTOCUSTOMERS (No. of issue occurrences: 1)
Try using INSERT in the log table. To do this, you must add code into AWS_ORACLE_EXT.PUT_LINE.
Procedure: SP_VALIDATEALLCUSTOMERS (No. of issue occurrences: 1)
Try using INSERT in the log table. To do this, you must add code into AWS_ORACLE_EXT.PUT_LINE.
Issue: 340: MySQL doesn't support the RAISE_APPLICATION_ERROR function.
Recommended action: Create a user-defined function.
No. of occurrences: 1 | Time estimate: 80 hour(s) | Documentation reference:https://dev.mysql.com/doc/refman/5.6/en/functions.html
Procedure: SP_SECURE_DML (No. of issue occurrences: 1)
Create a user-defined function.
Issue: 341: MySQL doesn't support sequences.
Recommended action: Try developing a system for sequences in your application.
Oracle procedure: SP_SECURE_DML
Properties
SQL
1 PROCEDURE SP_SECURE_DML
2 IS
3 BEGIN
4 IF TO_CHAR (SYSDATE, 'HH24:MI') NOT BETWEEN '08:00' AND '18:00'
5 OR TO_CHAR (SYSDATE, 'DY') IN ('SAT', 'SUN') THEN
6 RAISE_APPLICATION_ERROR (-20205,
7 'You may only make changes during normal office hou
8 END IF;
9 END SP_SECURE_DML;
Properties
SQL
01 CREATE PROCEDURE `CHINOOK`.`SP_SECURE_DML`()
02 BEGIN
03 /*
04 [340 - Severity CRITICAL - MySQL doesn't support the RAISE_APPLICA
05 IF TO_CHAR (SYSDATE, 'HH24:MI') NOT BETWEEN '08:00' AND '18:00'
06 OR TO_CHAR (SYSDATE, 'DY') IN ('SAT', 'SUN') THEN
07 RAISE_APPLICATION_ERROR (-20205,
08 'You may only make changes during normal office ho
09 END IF;
10 */BEGIN
11 END;
12 END;
sso@cs01.cluster[redacted].us-west-2.rd
Schemas [6]
AWS_ORACLE_EXT
CHINOOK
Tables [11]
Views [2]
Procedures [5]
SP_CALCULATELISTENINGTRENDS
SP_NOTIFYCUSTOMERS
SP_SECURE_DML
Parameters
SP_SENDINVOICESTOCUSTOMERS
SP_VALIDATEALLCUSTOMERS
Functions [4]
db01
information_schema
mysql
performance_schema

Aurora Partner Program

Amazon Aurora saves you money

Enterprise grade, open source pricing

	vCPU	Mem	Hourly Price
db.r3.large	2	15.25	$0.29
db.r3.xlarge	4	30.5	$0.58
db.r3.2xlarge	8	61	$1.16
db.r3.4xlarge	16	122	$2.32
db.r3.8xlarge	32	244	$4.64

- Storage consumed, up to 64 TB, is $0.10/GB-month
- IOs consumed are billed at $0.20 per million I/O
- Prices are for US East (N. Virginia) region

Simple pricing

No licenses

No lock-in

Pay only for what you use

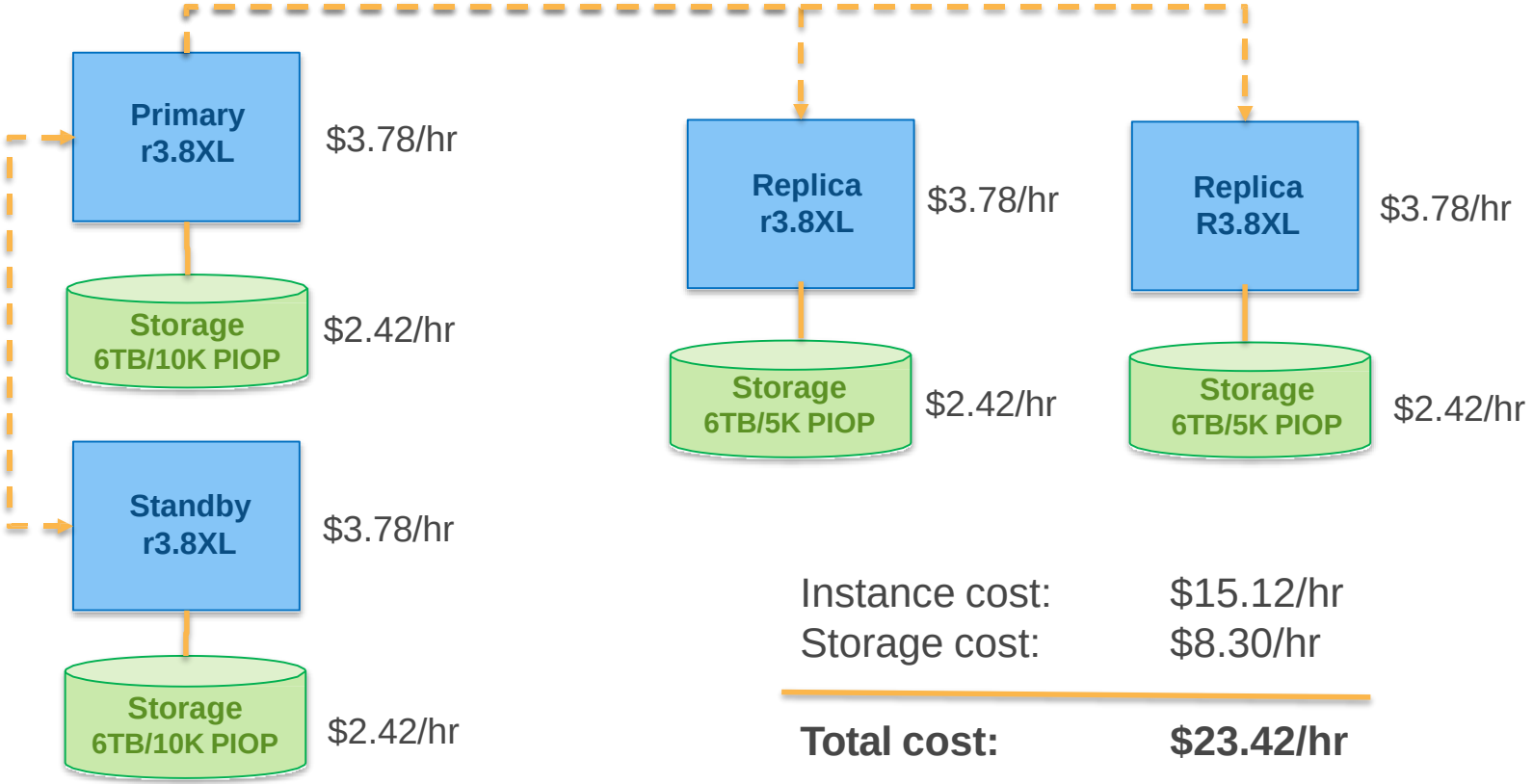
Discounts

44% with a 1-year Reserved Instance

63% with a 3-year Reserved Instance

Cost of ownership: Aurora vs. MySQL

MySQL configuration hourly cost

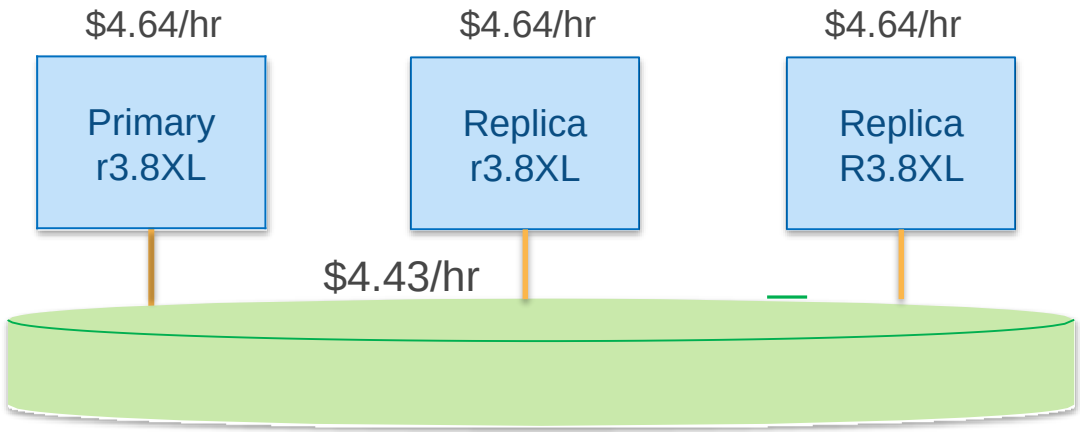


Instance cost:	$15.12/hr
Storage cost:	$8.30/hr
Total cost:	**$23.42/hr**

Cost of ownership: Aurora vs. MySQL

Aurora configuration hourly cost

- No idle standby instance
- Single shared storage volume
- No PIOPS—pay for use IO
- Reduction in overall IOP



Instance cost:	$13.92/hr
Storage cost:	$4.43/hr
Total cost:	**$18.35/hr**

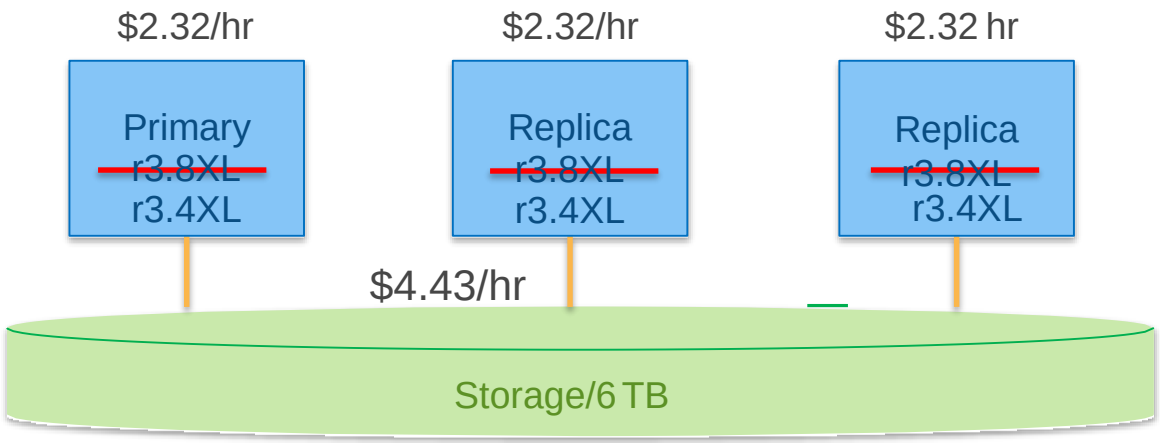
21.6% Savings

*At a macro level, Aurora saves over 50% in storage cost compared to RDS MySQL.

Cost of ownership: Aurora vs. MySQL

Further opportunity for saving

- Use smaller instance size
- Pay-as-you-go storage



Instance cost:	$6.96/hr
Storage cost:	$4.43/hr
Total cost:	**$11.39/hr**

51.3% Savings

Storage IOPS assumptions:

1. Average IOPS is 50% of maximum IOPS
2. 50% savings from shipping logs vs. full pages

Steve Loyd

Vice President, Engineering Operations

Kalyan Wunnava

Database Engineering Manager

zendesk

Zendesk is software for better customer service.

Growing global business

Sales, marketing, and customer success teams

Global customers

81,000

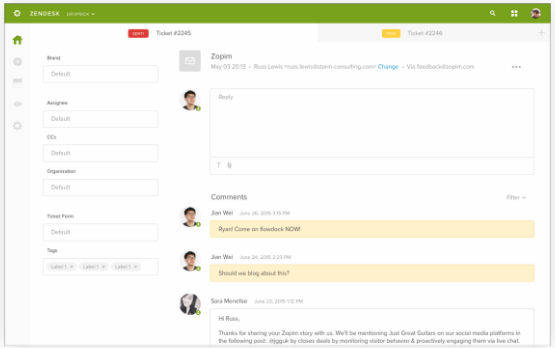
Year over year revenue growth

54%

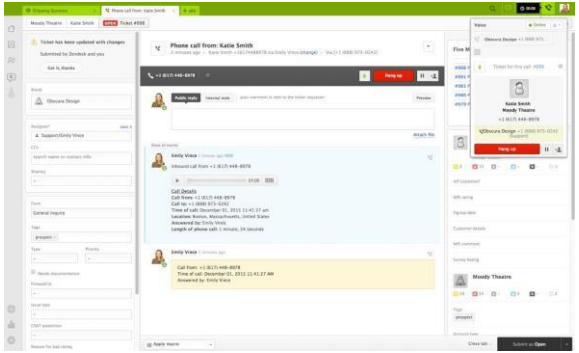
LATAM zipcar GRUBHUB DECKERS BRANDS california PIZZA KITCHEN

Multiproduct platform

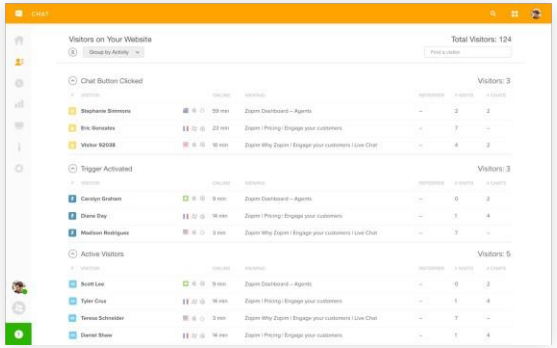
SUPPORT



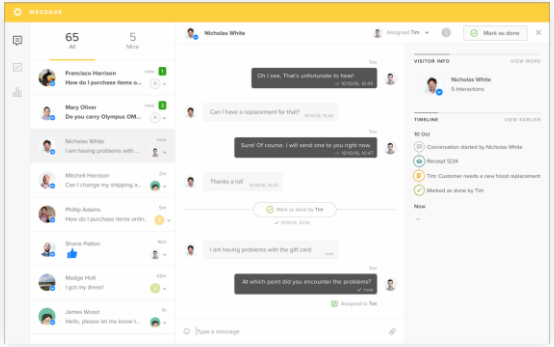
VOICE



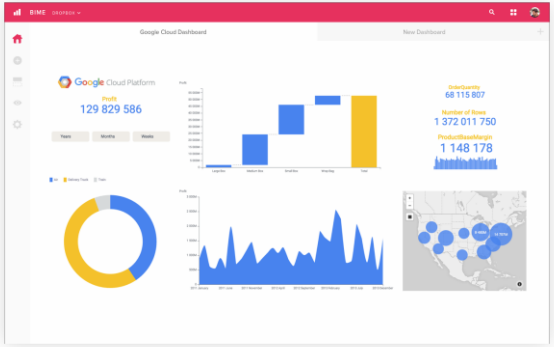
CHAT



MESSAGE



ANALYTICS



Security, compliance, and customer success for today's enterprise needs*

Security

Data encryption in transit/rest

Two-factor authentication

Automatic credit card redaction

SSO SAML support

Redundancy and disaster reco

Compliance

SOC II Type 2

ISO 27001

ISO 27018

HIPAA

PCI (Q3 2016)

Success and support

Professional services

Priority support

Service-level agreements

Training and certification

Customer success strategists

Global data centers (colocation and AWS)

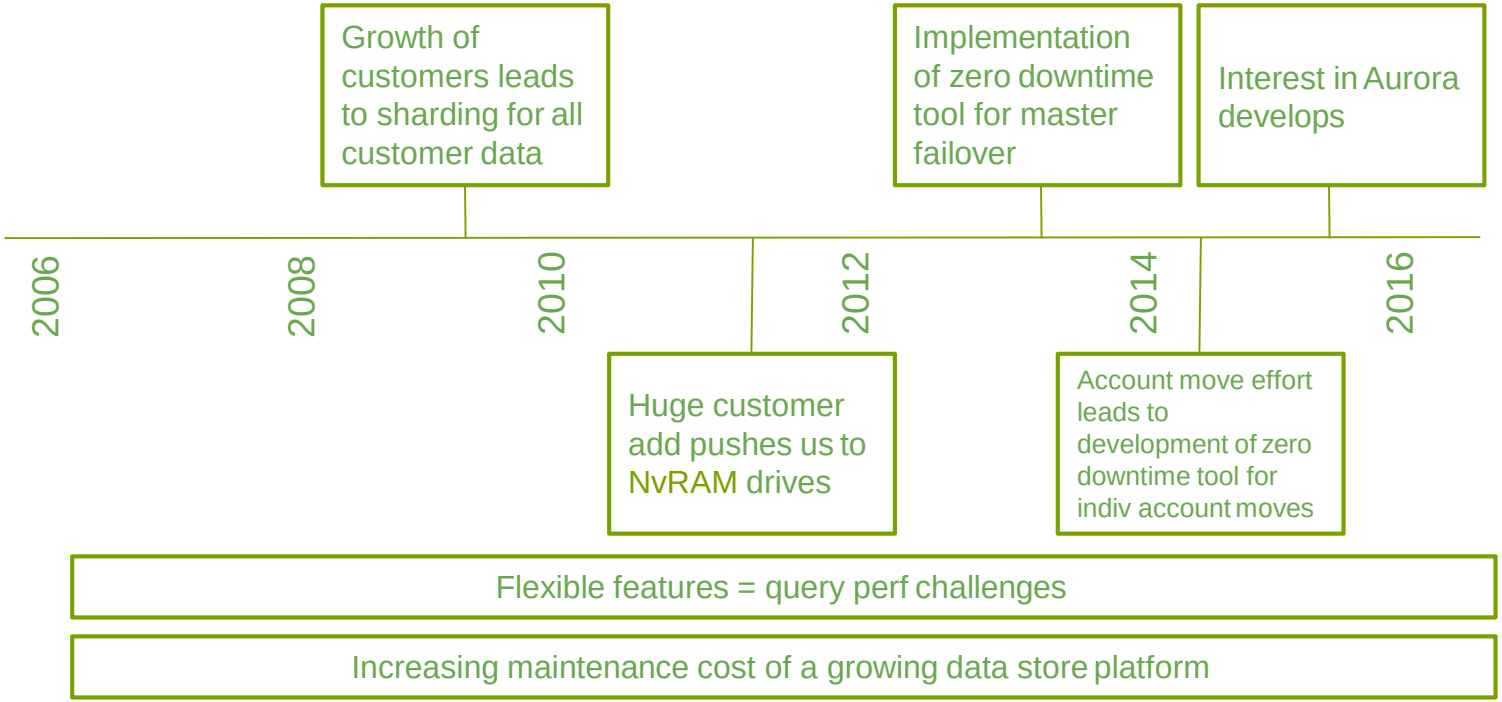
*Not all features listed are currently available on all products

MySQL data store importance

- Primary data store for key information since the beginning
- Three core uses: Account, alloc, sharded (customer)
- Sharded data store has seen the most growth year over year
- Over time we've added many other primary and secondary stores, but this relational data store is still key for much of our data

Growth challenges

- Since 2006 we've faced a number of challenges in scaling our MySQL datastore



2015 – Zendesk's first full AWS deployment

- Zendesk has been a user of AWS for past five years, but in very specific uses.

- 2015
  - Made our way to Vegas and learned all we could
  - Kicked off 120-day project to build full Zendesk stack in AWS
  - Succeeded and learned along the way
  - While our stack now includes more than 40 host groups, Amazon Aurora was a big part of our success

Image: https://jumpcloud.com/blog/wp-content/uploads/2014/11/AWS-in-vegas.jpg

Zendesk database environment – An overview

- 7 regions: 4 on premises and 3 on AWS
- Percona 5.6 and Amazon Aurora
- MySQL DBs: Shards (for customer data), account, and ID generation for shards
- MySQL storage on Fusion-io and NVMe for on-premises

Shards (DB cluster)

- Master and 2 slaves
- Single MySQL instance on a server
- Multiple shards on each server with identical schema
- Multiple customers on each shard
- Customer data on one shard only

Why Zendesk chose AWS Aurora

- Rapid provisioning compared to on-premises
- MySQL compatibility
- Managed environment with solid backups
- Elasticity – scale up/down with a button push
- Built-in support for encryption at rest

Aurora over RDS MySQL

- Cost effectiveness
- Database growth up to 64 TB
- Scalability to 15 replicas
- Better performance tests results

Limitations (in RDS) / workarounds

Limitation (RDS)	Workaround
MyISAM engine not supported (Aurora)	Use RDS MySQL which supports MyISAM
No DB server access (RDS)	Rewrite scripts to run from client side
No physical access to binary logs (RDS)	Use MySQL commands to read binlog events (e.g. Zendesk's accounts migration tool)
SUPER MySQL privilege not available (RDS)	Redefine grants (skip SUPER)

Limitations (in RDS) / workarounds

Limitation (RDS)	Workaround
Can’t run SET GLOBAL commands for dynamic config changes	Adjust config params in cluster/instance parameter group
Can’t run STOP/START SLAVE or CHANGE MASTER	Use RDS supplied system stored procs
Can’t turn OFF binary logging (global/session)	Not available!
COMPRESSED row format not supported	Aurora assumes COMPACT in the back end (Schema definition still conforms to original; no schema drift)

Example of COMPRESSED row format table

```
mysql> create table test1
    -> (
    ->   id INT NOT NULL PRIMARY KEY AUTO_INCREMENT
    ->  ,column1 TEXT NOT NULL
    ->  )
    ->  ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 ROW_FORMAT=COMPRESSED
    ->  ;
Query OK, 0 rows affected, 2 warnings (0.05 sec)

mysql> show warnings;
+---------+------+-----------------------------------------------------+
| Level   | Code | Message                                             |
+---------+------+-----------------------------------------------------+
| Warning | 1478 | ROW_FORMAT=COMPRESSED is not currently supported |
| Warning | 1478 | InnoDB: assuming ROW_FORMAT=COMPACT.                |
+---------+------+-----------------------------------------------------+

mysql> show create table test1\G
Create Table: CREATE TABLE `test1` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `column1` text NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 ROW_FORMAT=COMPRESSED
```