Globalizing Player Accounts with MySQL at Riot Games

Tyler Turk
Riot Games, Inc.
About Me

Senior Infrastructure Engineer

Player Platform at Riot Games

Father of three

Similar talk at re:Invent last year
Accounts Team

Responsible for account data

Provides account management

Ensures players can login

Aims to mitigate account compromises
Overview

The old and the new
League’s growth and shard deployment

Launched in 2009

Experienced rapid growth

Deployed multiple game shards

Each shard used their own MySQL DBs
Some context

Hundreds of millions of players worldwide

Localized primary / secondary replication

Data federated with each shard

Account transfers were difficult
Why MySQL?

- Widely used & adopted at Riot
- Used extensively by Tencent
- Ensures ACID compliance
Catalysts for globalization

General Data Protection Regulation

Decoupling from game platform

Single source of truth for accounts
Globalization of Player Accounts

Migrating from 10 isolated databases to a single globally replicated database
Data deployment considerations

- Globally replicated, multi-master
- Globally replicated, single master
- Federated or sharded data
- To cache or not to cache
Global database expectations

Highly available

Geographically distributed

< 1 sec latency replication

< 20ms read latency

Enables a better player experience
Continuent Tungsten

Third-party vendor

Provides cluster orchestration

Manages data replication

MySQL connector proxy
Why Continuent Tungsten?

Prior issues with Aurora

RDS was not multi-region

Preferred asynchronous replication

Automated cluster management
Explanation & tolerating failure
Deployment

Terraform & Ansible (docker initially)

4 AWS regions

r4.8xlarge (10Gbps network)

5TB GP2 EBS for data

15TB for logs / backups
Migrating the data

Multi-step migration of data

Consolidated data into 1 DB

Multiple rows for a single account
Load testing
Chaos testing
Monitoring
Performing backups

- Leverage standalone replicator
- Backup with xtrabackup
- Compress and upload to S3
- Optional delay on replicator
Performing maintenance

Cluster policies

Offline and shun nodes

Perform cluster switch
Performing schema changes

Schema MUST be backwards compatible

Order of operations for schema change:
1. Replicas in non-primary region
2. Cluster switch on relay
3. Perform change on former relay
4. Repeat steps 1-3 on all non-primary regions
5. Replicas in primary region
6. Cluster switch on write primary
7. Perform change on former write

The Process

- Offline node
- Wait for connections to drain
- Stop replicator
- Perform schema change
- Start replicator
- Wait for replication
- Online node
De-dockering

Fully automated the process

One server at a time

Performed live

Near zero downtime
Current state

Database deployed on host

No docker for database / sidecars

Accounts are distilled to a single row

Servicing all game shards
Lessons Learned

Avoiding the same mistakes we made
Databases in docker

Partially immutable infrastructure

Configuration divergence possible

Upgrades required container restarts

Pain in automating deploys
Large data imports

Consider removing indexes

Perform daily delta syncs

Migrate in chunks if possible
Think about data needs

Synchronous vs asynchronous

Read heavy vs write heavy
Impacts of replication latency

Replication can take >1 second

Impacts strongly consistent expectations

Immediate read-backs can fail

Think about “eventual” consistency
WAN replication is fragile

Not completely infallible

Think through your needs

Architect and design accordingly

Even with RiotDirect, it’s not perfect
Backup with caution (aka backups v1)
Demo Time!
Thank You!

Tyler Turk
tturk@riotgames.com
Rate My Session

Introducing gh-ost: triggerless, painless, trusted online schema migrations

Schedule
Timezone: Europe/Berlin +02:00

MON 3
TUE 4
WED 5

11:20

Introducing gh-ost: triggerless, painless, trusted online schema migrations
11:20 - 12:10, Matterhorn 2

Details

Rate & Review

Tap to rate

Feedback (optional)

SUBMIT