Testing for DR Failover Testing

Zehua Liu

Zendesk Singapore

SRECon Asia / Australia 2017

23 May 2017

About Me

- Zehua Liu
  - With Zendesk Singapore since 2015
  - Worked at startups at various stages (Atlassian, mig33, Circos Brand Karma)
  - Leads the tooling team at Zendesk SG

Disaster Recovery Failover Testing

Failing over from

the production data centre

to

the DR data centre

The Parent Problem

- A type of DiRT (Disaster Recovery Testing)
- Part of the BCDR project
  - Business Continuity and Disaster Recovery
- Our focus here
  - Testing lost of the data centre
  - Testing only customer facing features
    - Internal tools are excluded

Why conduct DR failover testing

- Compliance - SOC2 Testing twice a year
- Customer Agreements: Advanced Security Add-On
  - Recovery Time Objective - 8 hours
  - Recovery Point Objective - 0 hours
- Test and verify the procedures and documentation
- Identify gaps
- Improve the overall DR process
- Training for Responding Parties

chat

Past attempts of DR failover testing

- Two DR failover testing exercises
  - Four DR failover tests
- Encountered various issues
  - Infrastructure, e.g., database, network
  - Configuration
  - Application, couldn’t handle failure in infrastructure
- Examples of issues
  - Double billing customers
  - iOS app did not work
  - DB replication back to original production was too slow

The Problem

Can we increase
our confidence in
DR Failover Testing?

chat

The Answer

Test the DR environment before failing over

Testing the DR environment

- Ideal: automated testing while DR is still in standby mode
    - Run the exact same tests that we run for production
    - Automatically triggered after a change to DR
- Issues:
    - Most tests inevitably write data about the test accounts to the DBs in DR
    - Run just the read only tests?
- The big question:
    - Should we allow direct write into data stores in DR??

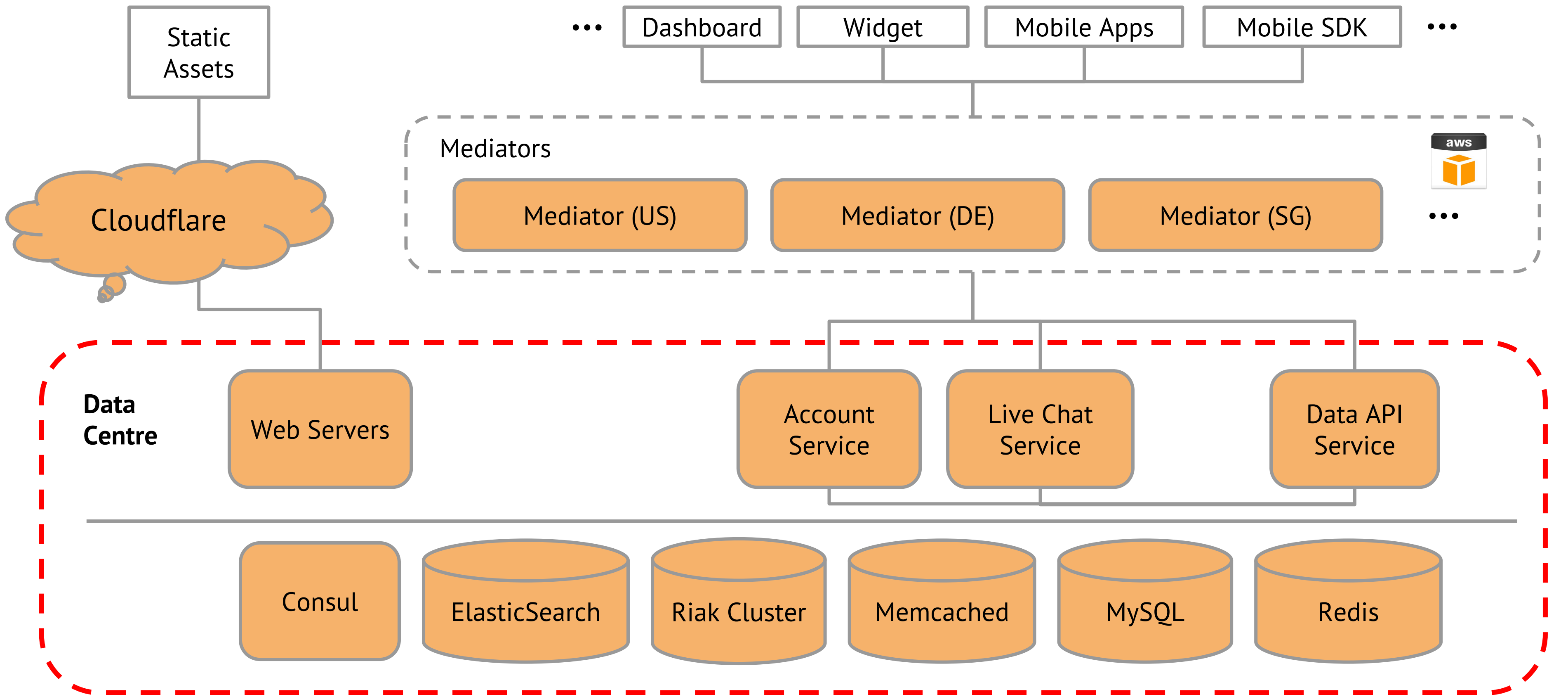
The Big Question

Should we allow direct write into data stores in DR?

Testing the DR environment

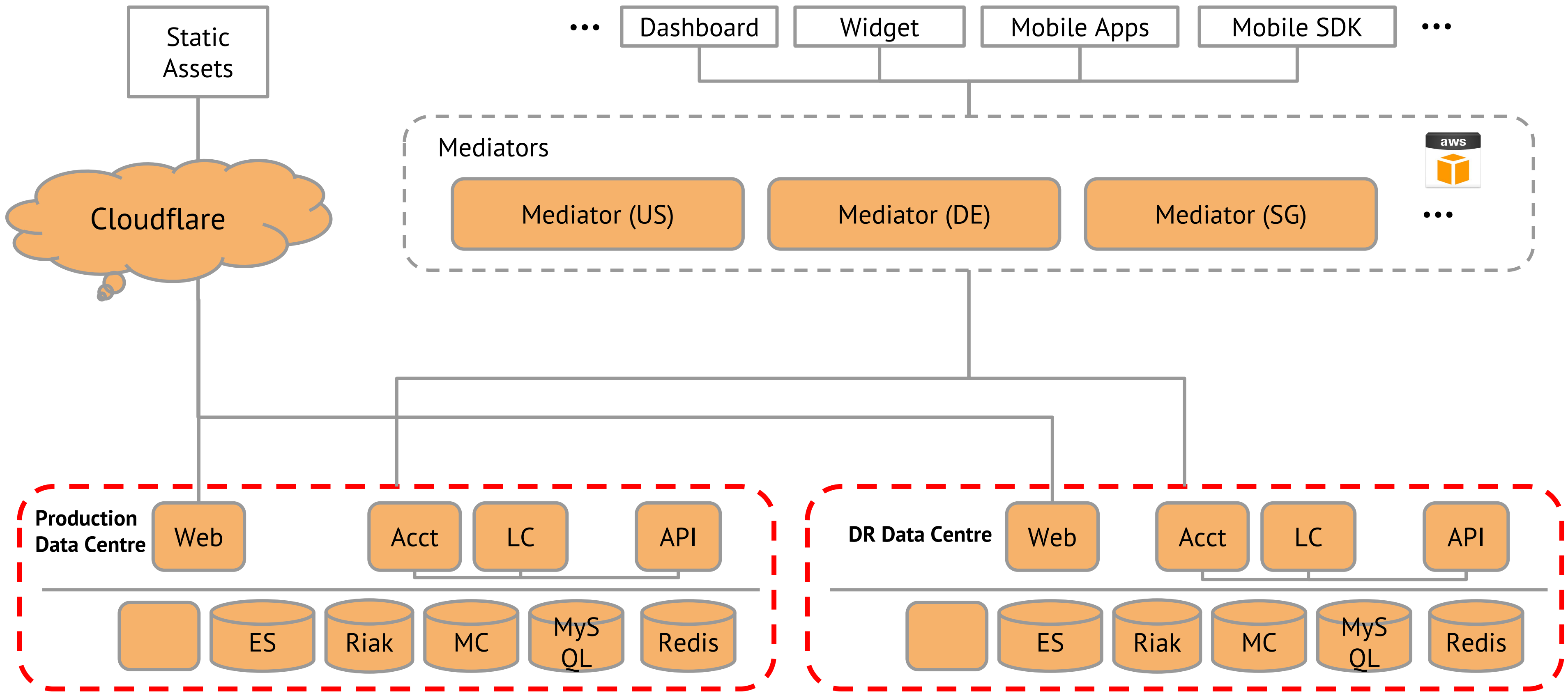
- The big question:
  - Should we allow direct write into data stores in DR??
- A ***trade-off*** between risk of production failure and risk of failed DR failover
  - writing to DR DB => risk of production failure
  - test coverage => risk of failed DR failover

Zendesk Chat Technical Architecture



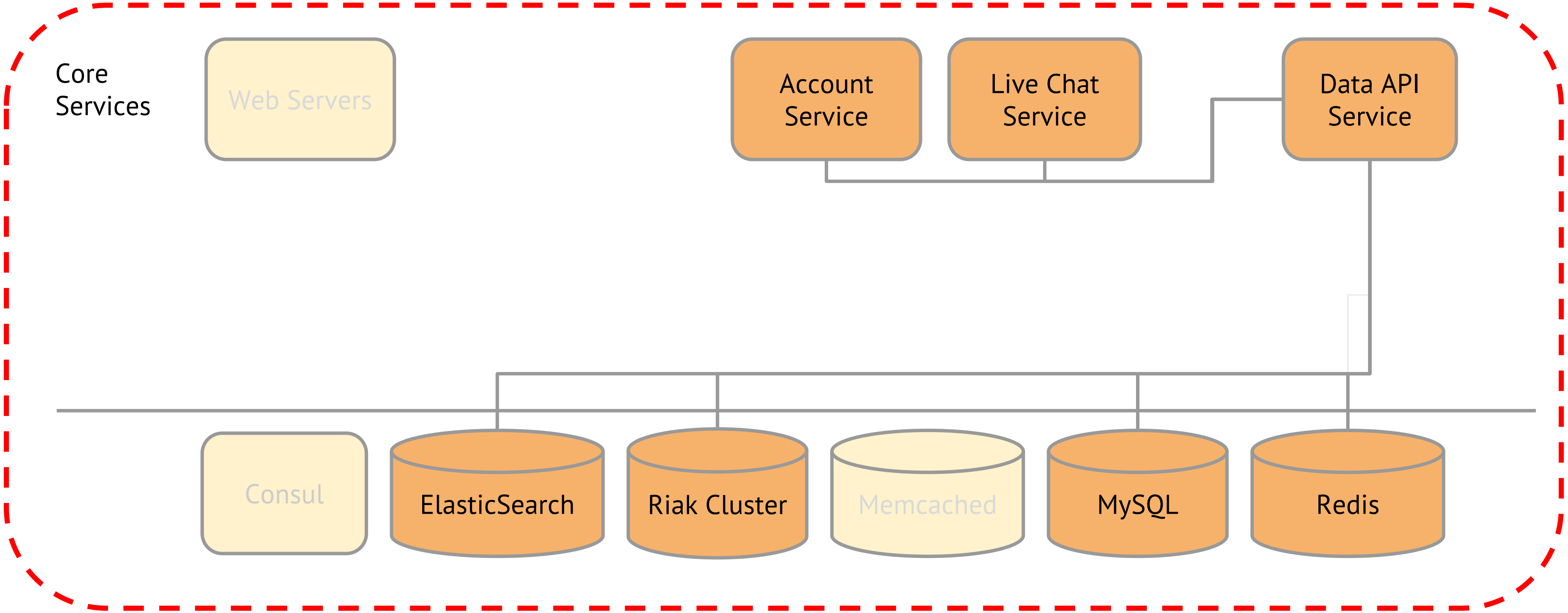
chat

DR Failover



chat

Zendesk Chat Technical Architecture



chat

Data Stores

- MySQL
  - master ⇒ slave replication (DR DB as read only slave)
  - Least confident, might cause data corruption, stop replication, etc
- Riak
  - Commercial license with multi-dc sync support
- ElasticSearch
  - Could be rebuilt from source of truth
- Redis: ephemeral data
- Memcached: cold start?

The Approach

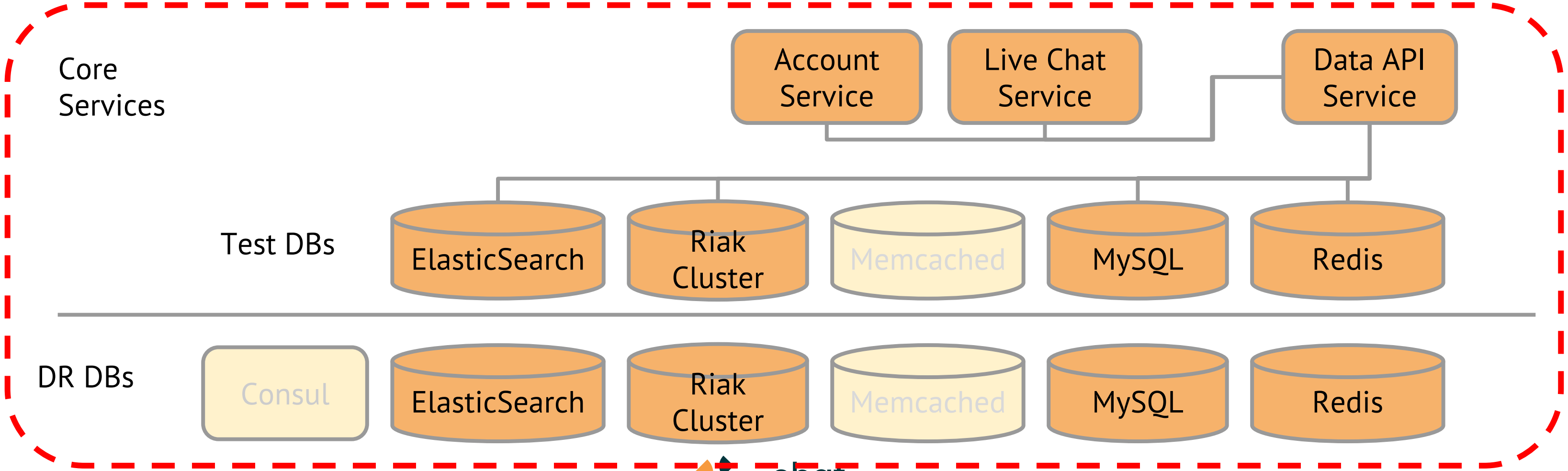
- Good news
  - The applications mostly partition data by accounts!
  - We could use a dedicated set of test accounts that would never get used on prod
    - In theory, these test data is isolated from other customer account data in data stores
    - Good to replicate back and forth between DR and production MySQL DBs

Alternatives

- Avoid writing to the real DR DBs?
- Allow writing to only less risky DBs?
- Allow writing to all DBs

Alternatives - Avoid writing to the real DR DBs?

- Setup a different set of test data store servers
  - Configure the apps to use them only during test
  - Switch back before the actual failover
  - Does not test the physical connection



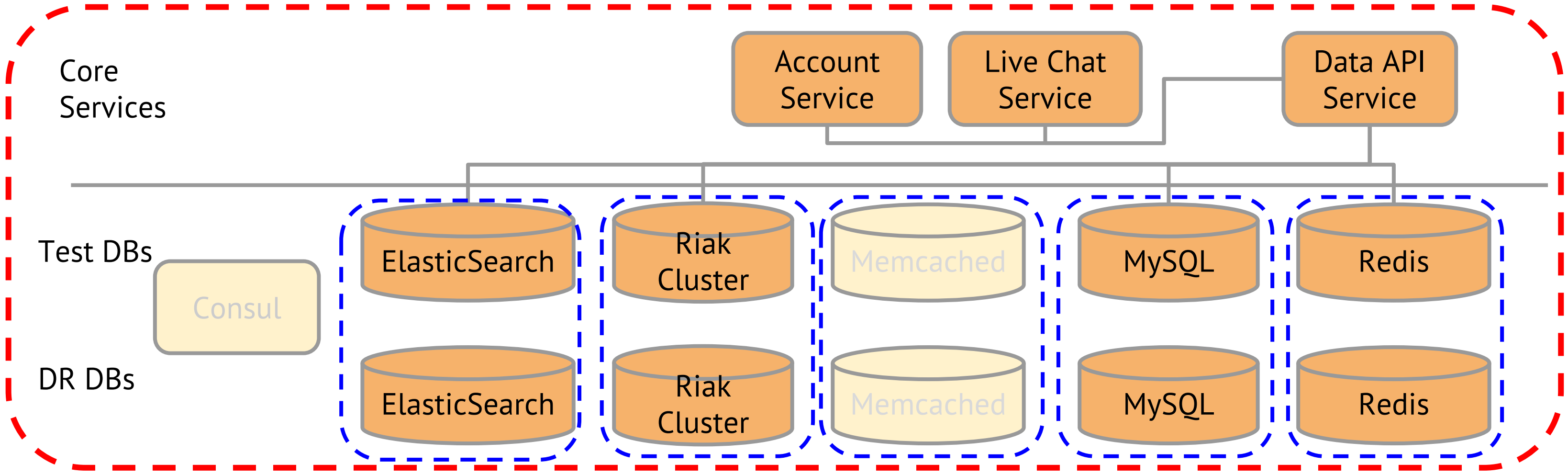
chat

Alternatives - Avoid writing to the real DR DBs?

- Setup the different set of DBs on the same physical servers as the real ones
  - Naming tricks:
    - test_account_db to mirror account database
    - test_chat_history for ES indices, etc
  - Covers the physical connection

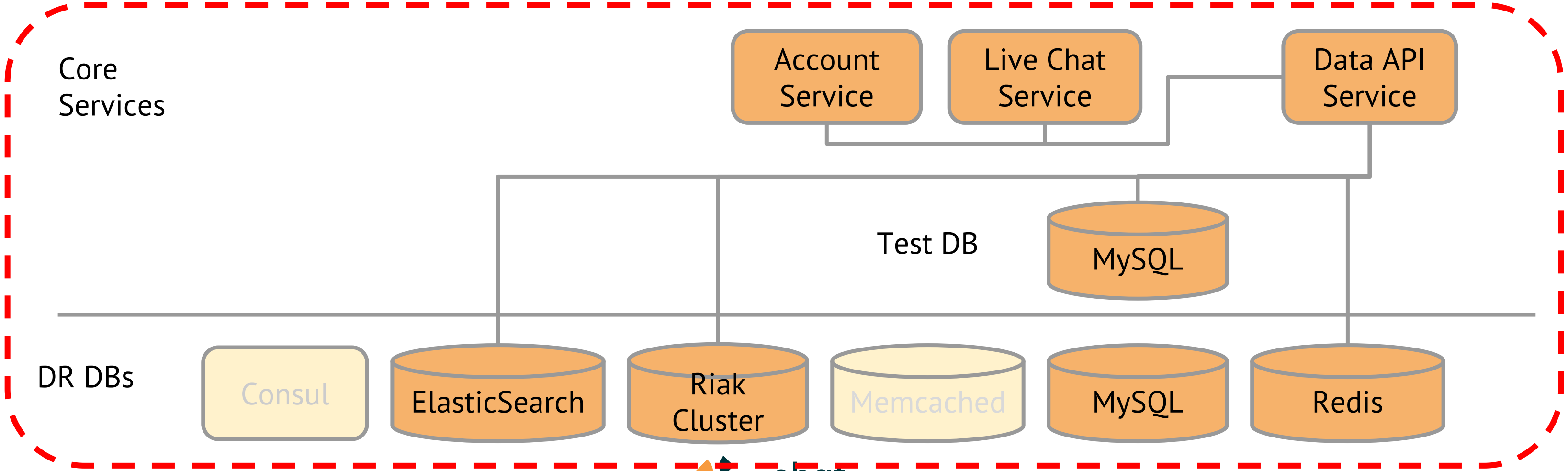
Alternatives - Avoid writing to the real DR DBs?

- Setup the different set of DBs on the same physical servers as the real ones



Alternatives - Allow partial writes

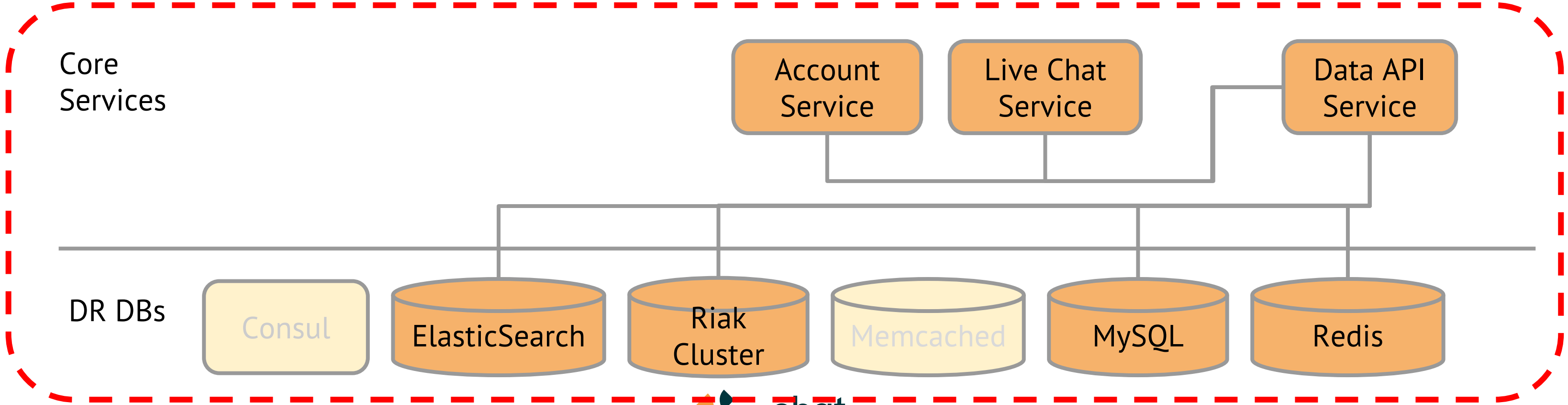
- Use the real ones for all DBs, except MySQL
  - Use a test DB for MySQL
    - MySQL is the most risky one to allow writes
  - Setup the test DB as a writable slave of the DR DB?



chat

Alternatives - Writing to real DR DBs!

- Use all real ones!
  - Data in DR DB will have to be eventually replicated back to production DB
  - Risks of test data in DR causing conflicts when replicated back to production DB



Testing the DR environment

- The big question:
  - Should we allow direct write into data stores in DR??
- A ***trade-off*** between risk of production failure and risk of failed DR failover
  - writing to DR DB => risk of production failure
    - Yes, let’s do it!
  - test strategy/coverage => risk of failed DR failover
    - ?

chat

The Final Proposal

- More issues:
  - Some tables use auto-increment column as primary key
  - Insertion into those tables in DR ⇒ replication conflicts
- Solutions:
  - Play with auto_increment_increment and offset
  - Avoid insertion into those tables
    - Identify those tables and avoid running tests that create new data in them
    - Luckily there are only a few non-critical ones

The Final Proposal

- More issues:
  - Someone might run the excluded tests and create new rows in the auto-increment tables in DR!
- Solution:
  - Use a different user with restricted permission
  - Switch back to a full access user before failover

The Final Proposal

- DR apps use real DR DBs
  - No test DBs in DR
  - Same configuration as production
- MySQL master-master replication between prod and DR
- Avoid doing insertion in tables with auto-increment pkey
  - Exclude integration tests that do such insertions
  - Setup a MySQL user with restricted access
- We could run end-to-end browser tests against DR while it’s in standby mode!

The Final Proposal

- The ***trade-off*** between risk of production failure and risk of failed DR failover
  - writing to DR DB => ***low*** risk of production failure
    - Replication might fail, but we would know it early
  - test strategy/coverage => ***low*** risk of failed DR failover
    - Application on DR might fail in the excluded test cases, but not critical

The Final Proposal - Caveats

- Does not cover all aspects of DR failover readiness
  - Only functional tests
  - A bit of network link testing via MySQL replication
- Adds to the complexity of DR failover
  - More steps to be performed during the failover

Conclusion

- It is possible to test the DR env in standby mode
- It is a trade-off between risk of production failure and risk of failed DR failover
- Avoid using auto-increment keys if multi-DC support is needed

Questions?