Cloud-Hydra: A Cloud Native Multi-Cloud Defensive Load Balancing Framework

Josh Stern, Rachid Tak Tak, Julian Trinh, Filip Vukelic

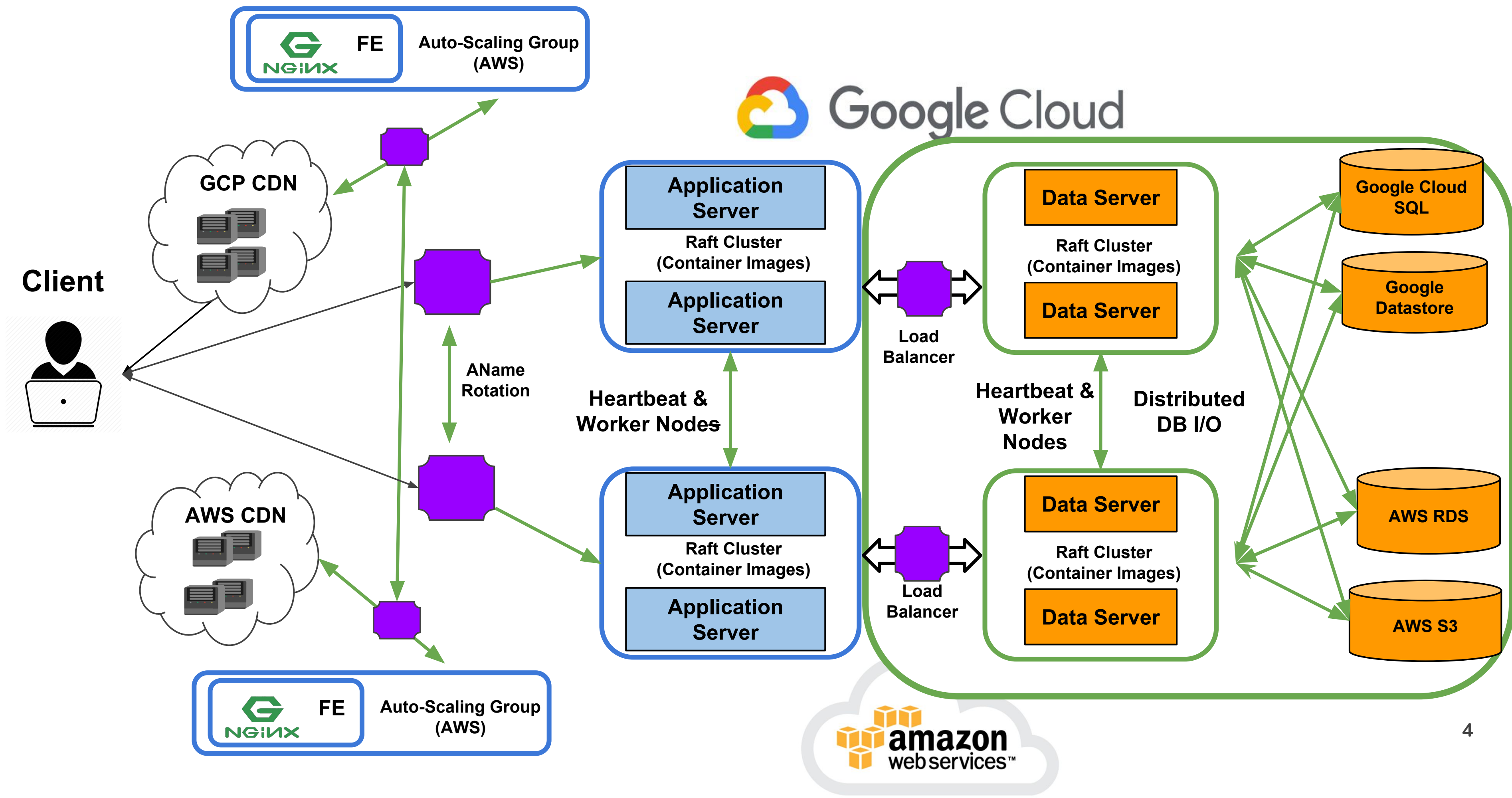
SPRINT 4

Sprint History

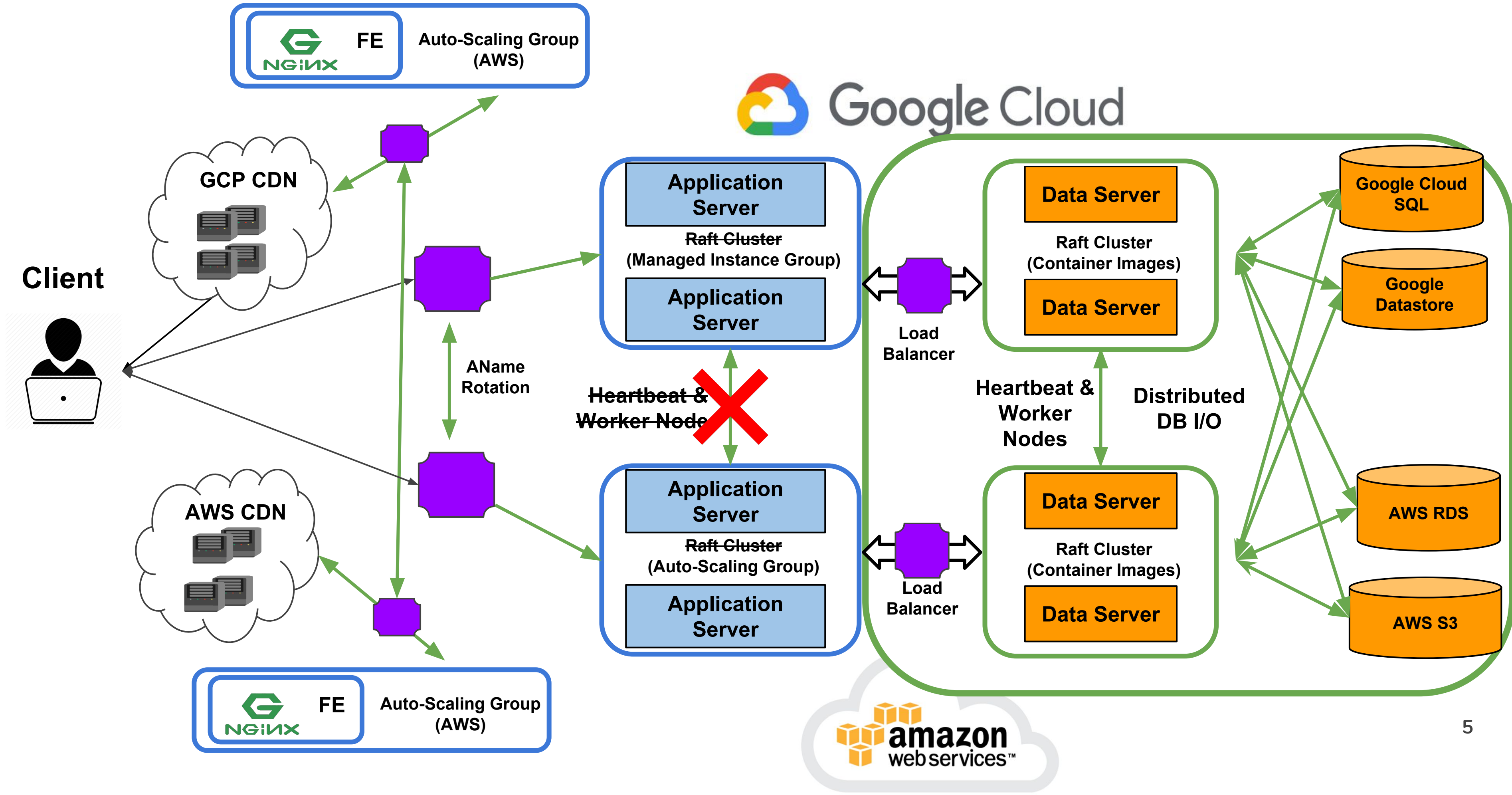
- **Sprint 1- Built fullstack web app in GCP and AWS**
  - Compute Engine
  - Managed Instance Groups (with Docker Images)
  - Nginx front end server + CDNs
- **Sprint 2 - Unified Data Layer**
  - Raft consensus
  - New leader election
  - Distributed Raft log
  - Cross-cloud consensus, single cloud load balancing
- **Sprint 3 - Cross cloud forwarding and recovery**
  - Leader Forwarding
  - Data layer Load Balancing
  - Database Recovery

Goals of Sprint 4

- Testing Raft Cluster I/O with node failures
- Testing Multi-Cloud DBRecovery
- Synchronizing AWS and GCP databases



FE
Auto-Scaling Group (AWS)
Google Cloud
GCP CDN
Client
Application Server
Raft Cluster (Container Images)
Application Server
Data Server
Raft Cluster (Container Images)
Data Server
Google Cloud SQL
Google Datastore
Load Balancer
AName Rotation
Heartbeat & Worker Nodes
Heartbeat & Worker Nodes
Distributed DB I/O
AWS CDN
Application Server
Raft Cluster (Container Images)
Application Server
Load Balancer
Data Server
Raft Cluster (Container Images)
Data Server
AWS RDS
AWS S3
FE
Auto-Scaling Group (AWS)
amazon web services

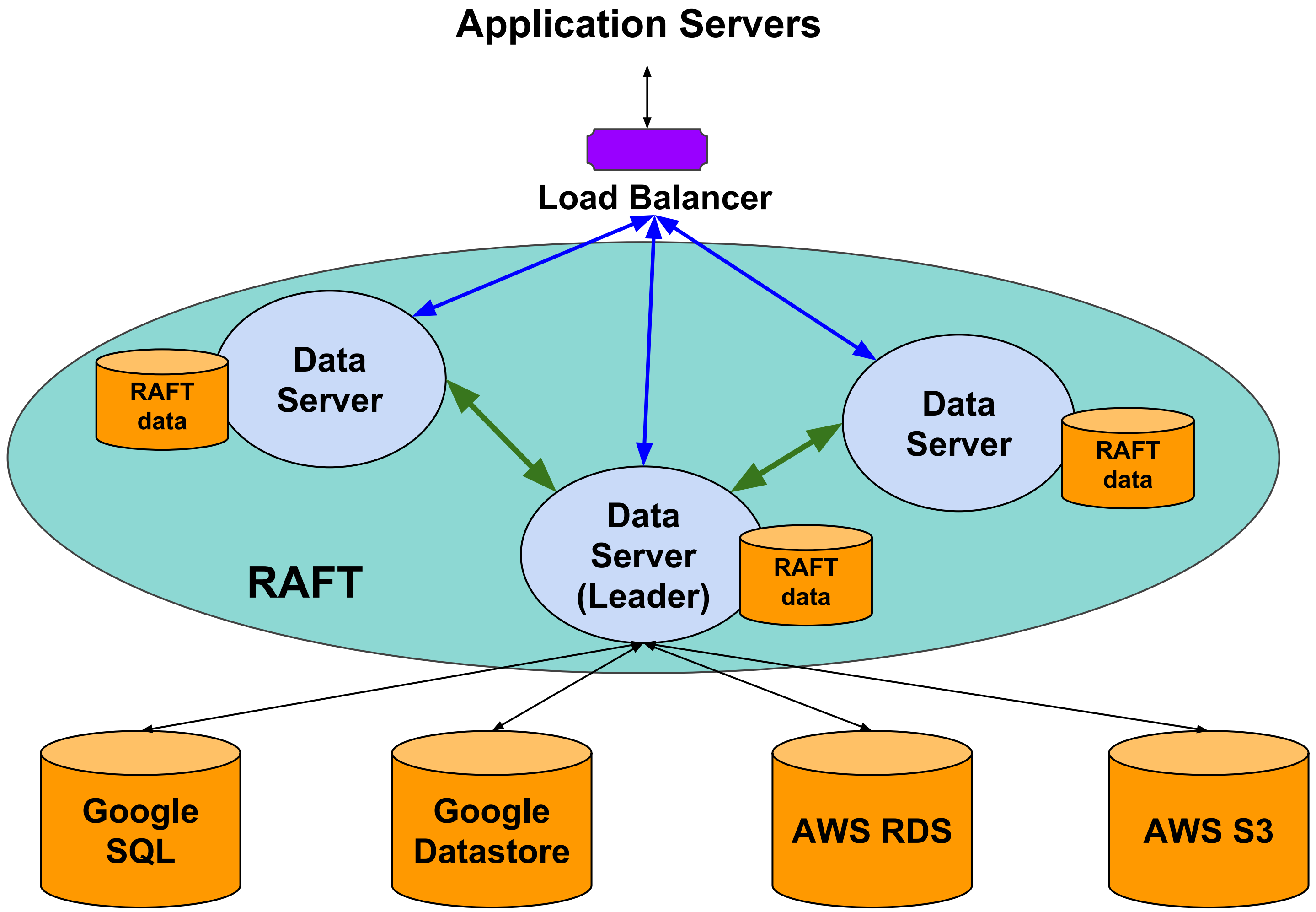

FE
Auto-Scaling Group (AWS)
Google Cloud
GCP CDN
Client
Application Server
~~Raft Cluster~~
(Managed Instance Group)
Application Server
Data Server
Raft Cluster
(Container Images)
Data Server
Google Cloud SQL
Google Datastore
Load Balancer
AName Rotation
~~Heartbeat & Worker Node~~
Heartbeat & Worker Nodes
Distributed DB I/O
AWS CDN
Application Server
~~Raft Cluster~~
(Auto-Scaling Group)
Application Server
Data Server
Raft Cluster
(Container Images)
Data Server
AWS RDS
AWS S3
Load Balancer
FE
Auto-Scaling Group (AWS)
amazon web services

Application Layer is Stateless

- Originally planned to configure the application layer into a Raft cluster
- The application (webserver) doesn’t care about state
- Can be treated like a microservice to scale horizontally without much overhead

Configuring the Data Layer Load Balancer

- Using Nginx as a proxy
- Requests are distributed to the cluster via Round Robin
- Nginx will periodically heartbeat with cluster nodes to ensure availability before scheduling a request


Application Servers
Load Balancer
Data Server
RAFT data
Data Server
RAFT data
Data Server (Leader)
RAFT data
RAFT
Google SQL
Google Datastore
AWS RDS
AWS S3

DEMO TIME ！

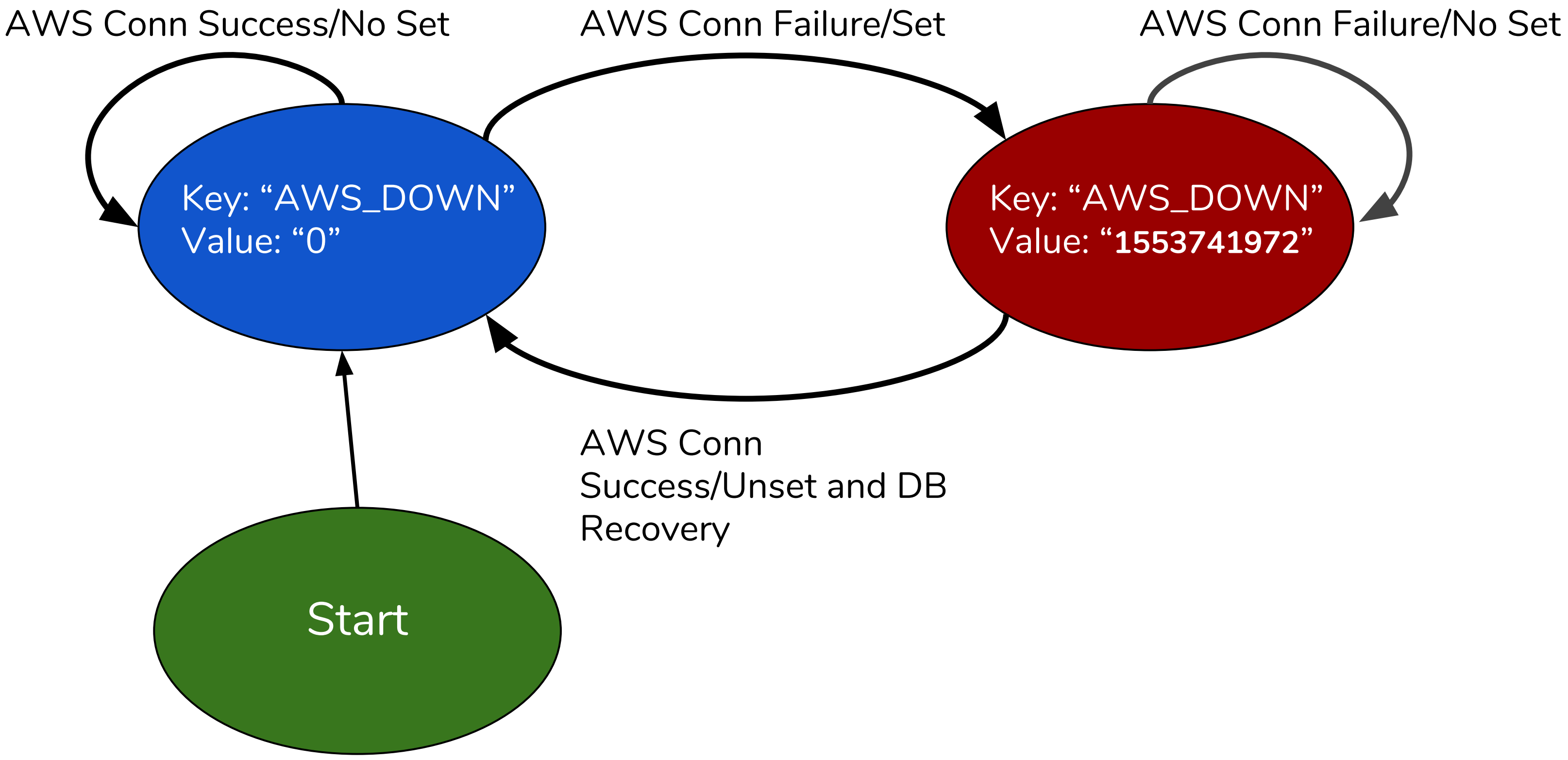
Multi-Cloud Database I/O

- The database servers will read from a pseudo random database that is healthy
- If read encounters unhealthy DB, it will notify leader and read from a healthy DB
- On writes, the dao servers will write to all the DBs
- If one or all the dbs are down, beside the instructions, the db down state will be written to the raft log as well

Database Forwarding

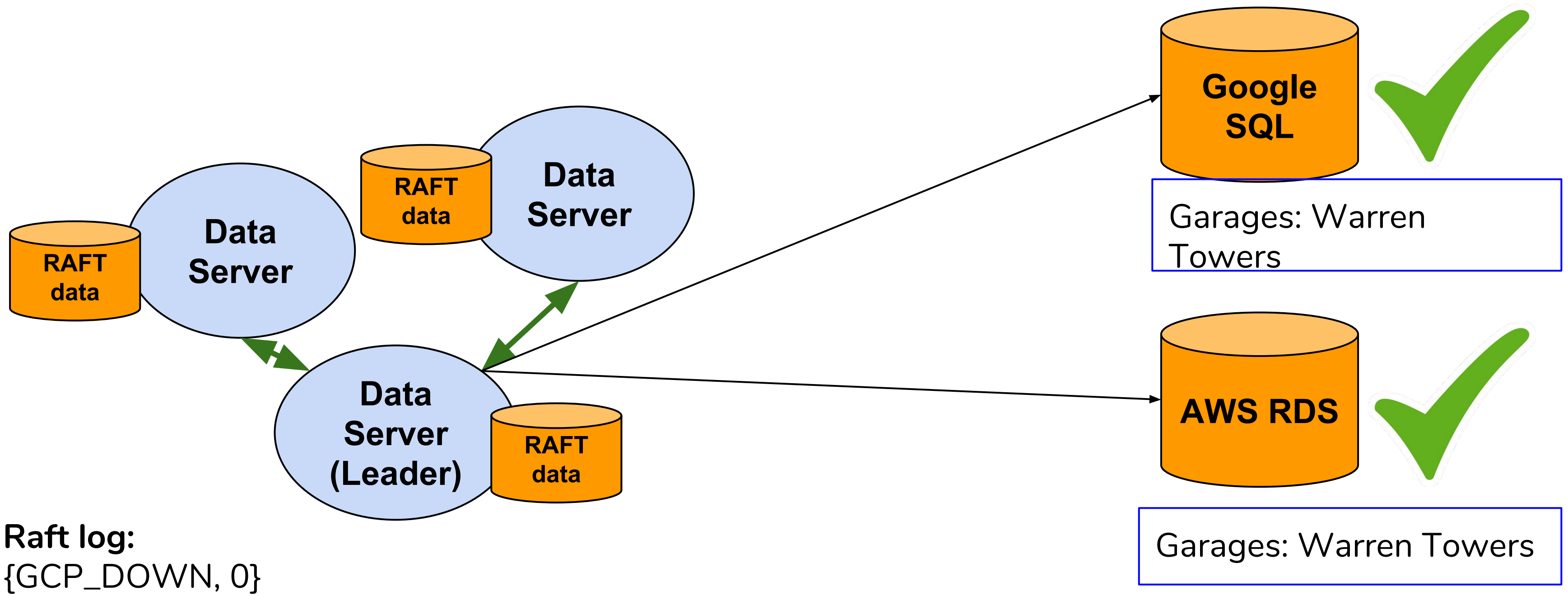
- Detecting down databases
- Detecting databases that have come back from the dead
  - Use raft log for recovery
- If a non leader node detects a database state change, notify the leader
  - Only the leader can propose changes to the raft log / update db state
- Internal Timestamps -- need to switch to external source

Database Recovery



Database Recovery Example

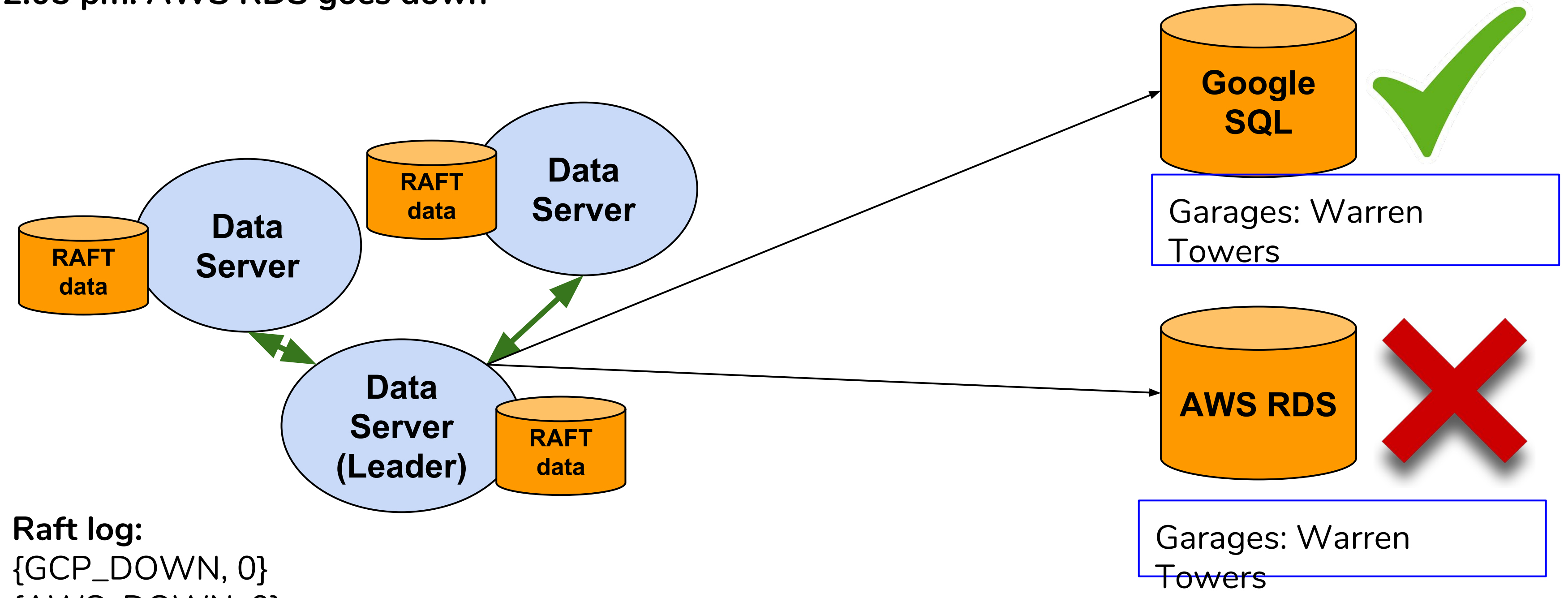
2:00 pm: Client issues write (Insert Warren Towers into Garages)



Raft log:
{GCP_DOWN, 0}
{AWS_DOWN, 0}
{Insert Warren Towers into Garages, timestamp: 2:00pm}

Database Recovery Example

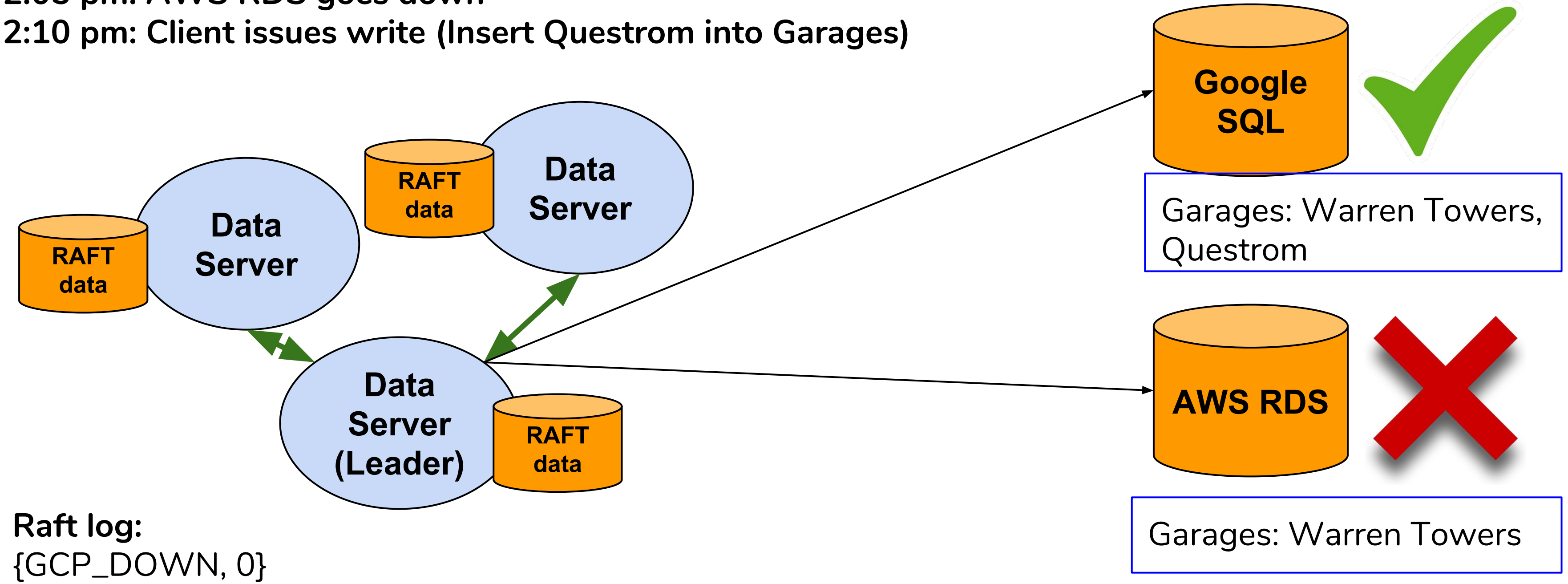
2:00 pm: Client issues write (Insert Warren Towers into Garages)
2:05 pm: AWS RDS goes down



Raft log:
{GCP_DOWN, 0}
{AWS_DOWN, 0}
{Insert Warren Towers into Garages, timestamp: 2:00pm}

Database Recovery Example

2:00 pm: Client issues write (Insert Warren Towers into Garages)
2:05 pm: AWS RDS goes down
2:10 pm: Client issues write (Insert Questrom into Garages)



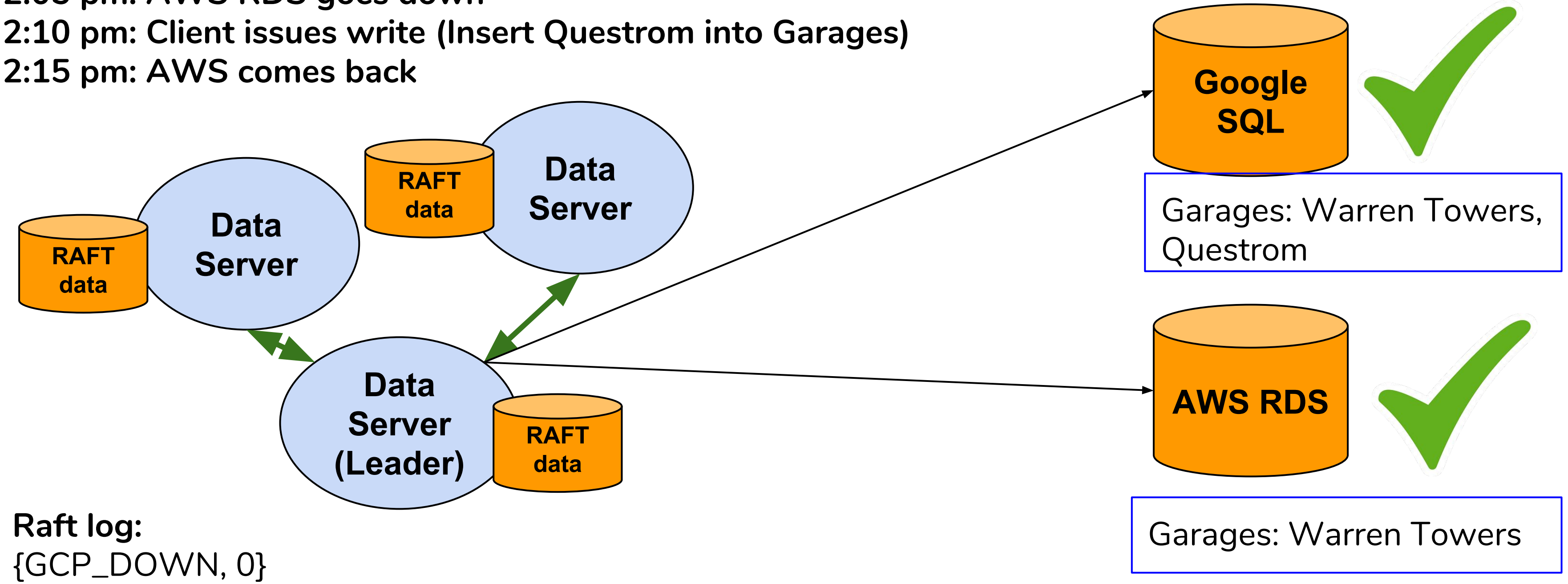
Garages: Warren Towers, Questrom

Garages: Warren Towers

Raft log:
{GCP_DOWN, 0}
{AWS_DOWN, 2:10pm}
{Insert Warren Towers into Garages, timestamp: 2:00pm}
{Insert Questrom into Garages, timestamp: 2:10pm}

Database Recovery Example

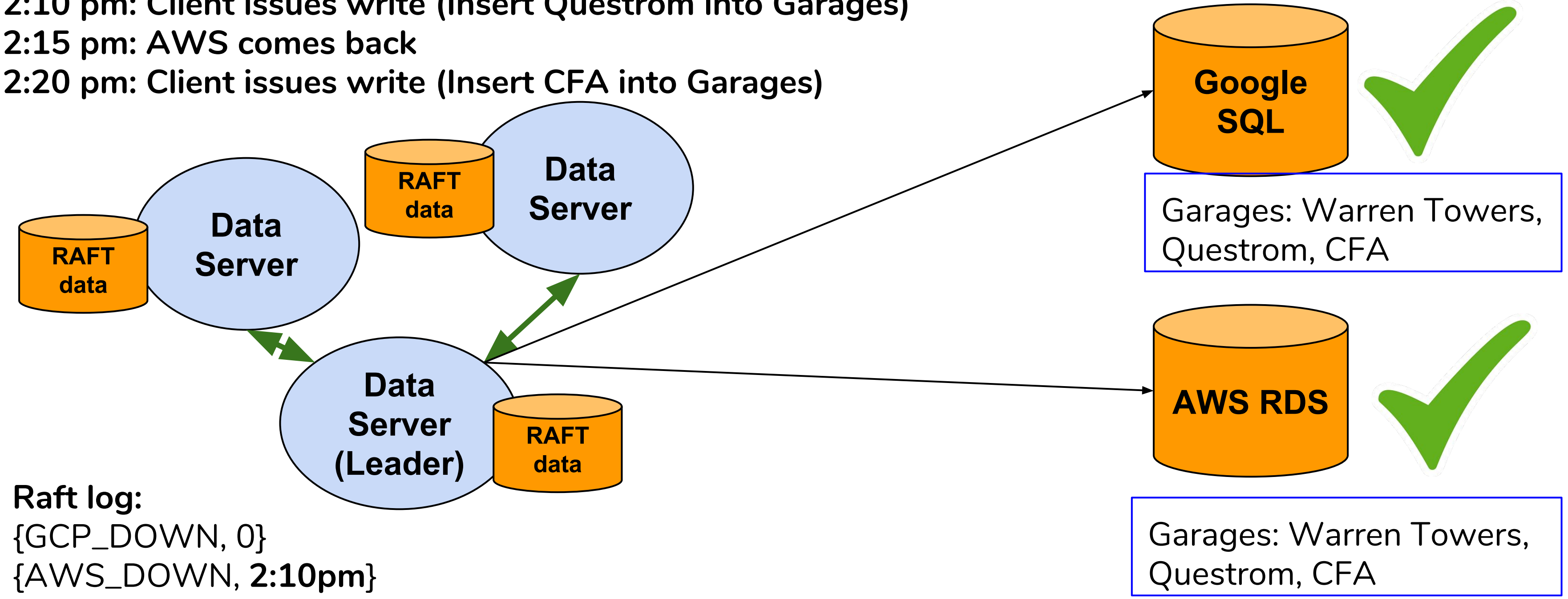
2:00 pm: Client issues write (Insert Warren Towers into Garages)
2:05 pm: AWS RDS goes down
2:10 pm: Client issues write (Insert Questrom into Garages)
2:15 pm: AWS comes back



Raft log:
{GCP_DOWN, 0}
{AWS_DOWN, 2:10pm}
{Insert Warren Towers into Garages, timestamp: 2:00pm}
{Insert Questrom into Garages, timestamp: 2:10pm}

Database Recovery Example

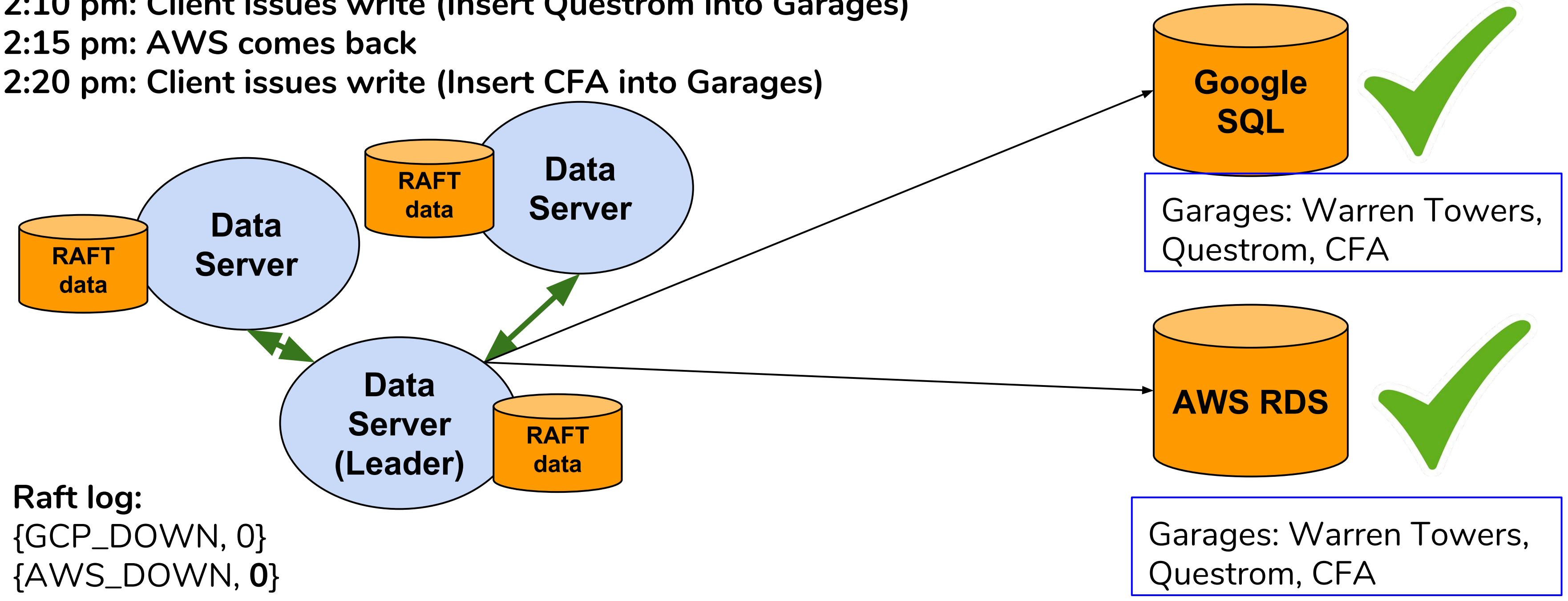
2:00 pm: Client issues write (Insert Warren Towers into Garages)
2:05 pm: AWS RDS goes down
2:10 pm: Client issues write (Insert Questrom into Garages)
2:15 pm: AWS comes back
2:20 pm: Client issues write (Insert CFA into Garages)



Raft log:
{GCP_DOWN, 0}
{AWS_DOWN, **2:10pm**}
{Insert Warren Towers into Garages, timestamp: 2:00pm}
{Insert Questrom into Garages, **timestamp: 2:10pm**}
{Insert CFA into Garages, **timestamp: 2:20**}

Database Recovery Example

2:00 pm: Client issues write (Insert Warren Towers into Garages)
2:05 pm: AWS RDS goes down
2:10 pm: Client issues write (Insert Questrom into Garages)
2:15 pm: AWS comes back
2:20 pm: Client issues write (Insert CFA into Garages)



Garages: Warren Towers, Questrom, CFA

Garages: Warren Towers, Questrom, CFA

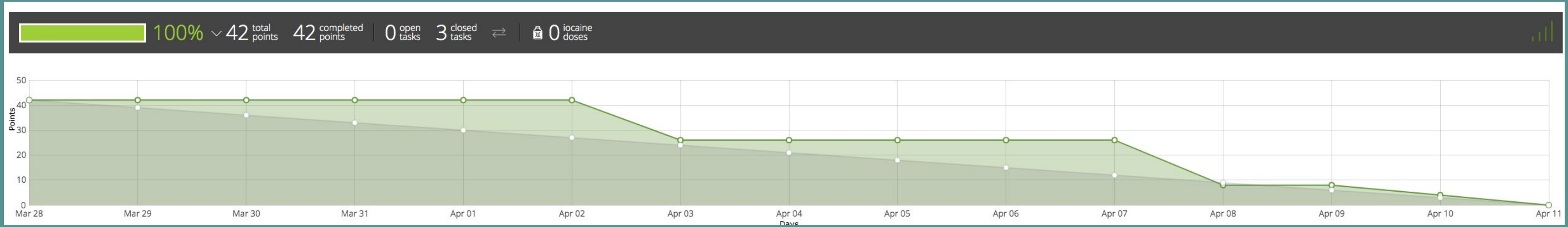
Raft log:
{GCP_DOWN, 0}
{AWS_DOWN, **0**}
{Insert Warren Towers into Garages, timestamp: 2:00pm}
{Insert Questrom into Garages, timestamp: 2:10pm}
{Insert CFA into Garages, timestamp: 2:20}

DNS A-NAME rotation

- What is DNS A-NAME rotation?
- DNS maps a domain name to an IP address
- We implement ANAME rotation by mapping single DNS name to multiple IP addresses
- The DNS service heartbeats with the hosts to ensure they are up before choosing the IP, so it knows if any of the machines is down
- Whenever api.cloud-hydra.com is requested, it uses round robin to either send it to AWS or GCP load balancers

DEMO TIME!

Sprint4 Burndown


100% 42 total points 42 completed points 0 open tasks 3 closed tasks 0 iocaine doses

Next Steps

- Create a test suite to test Hydra
  - Unit tests
  - System and E2E tests
  - Metrics and timing characteristics
  - Validate consistency claims via test results
- DNS failover for application servers
- Identify behaviors if Hydra fails
- Enhance UI/UX of the frontend

Thank you!
Questions?