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

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

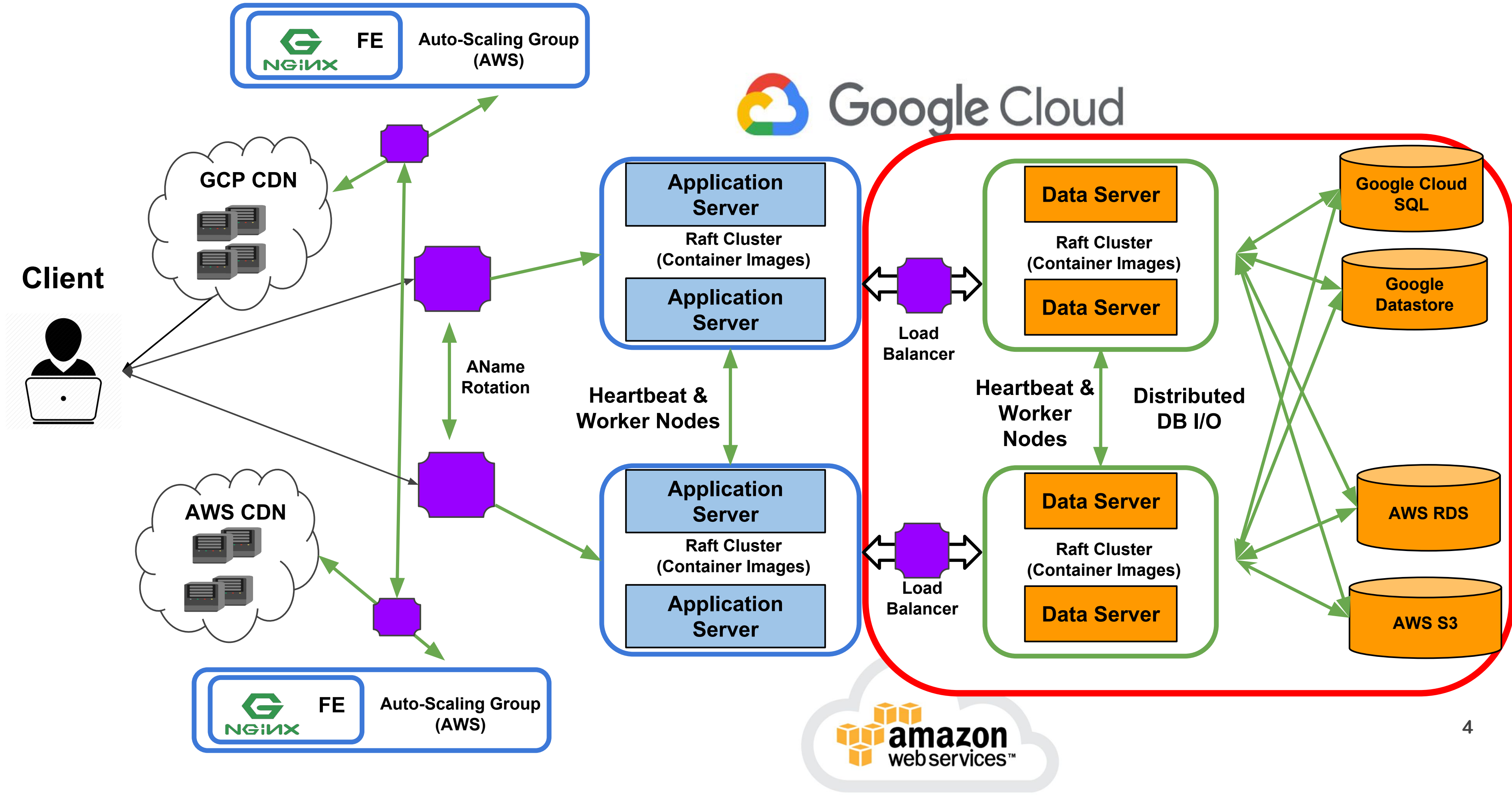
SPRINT 2

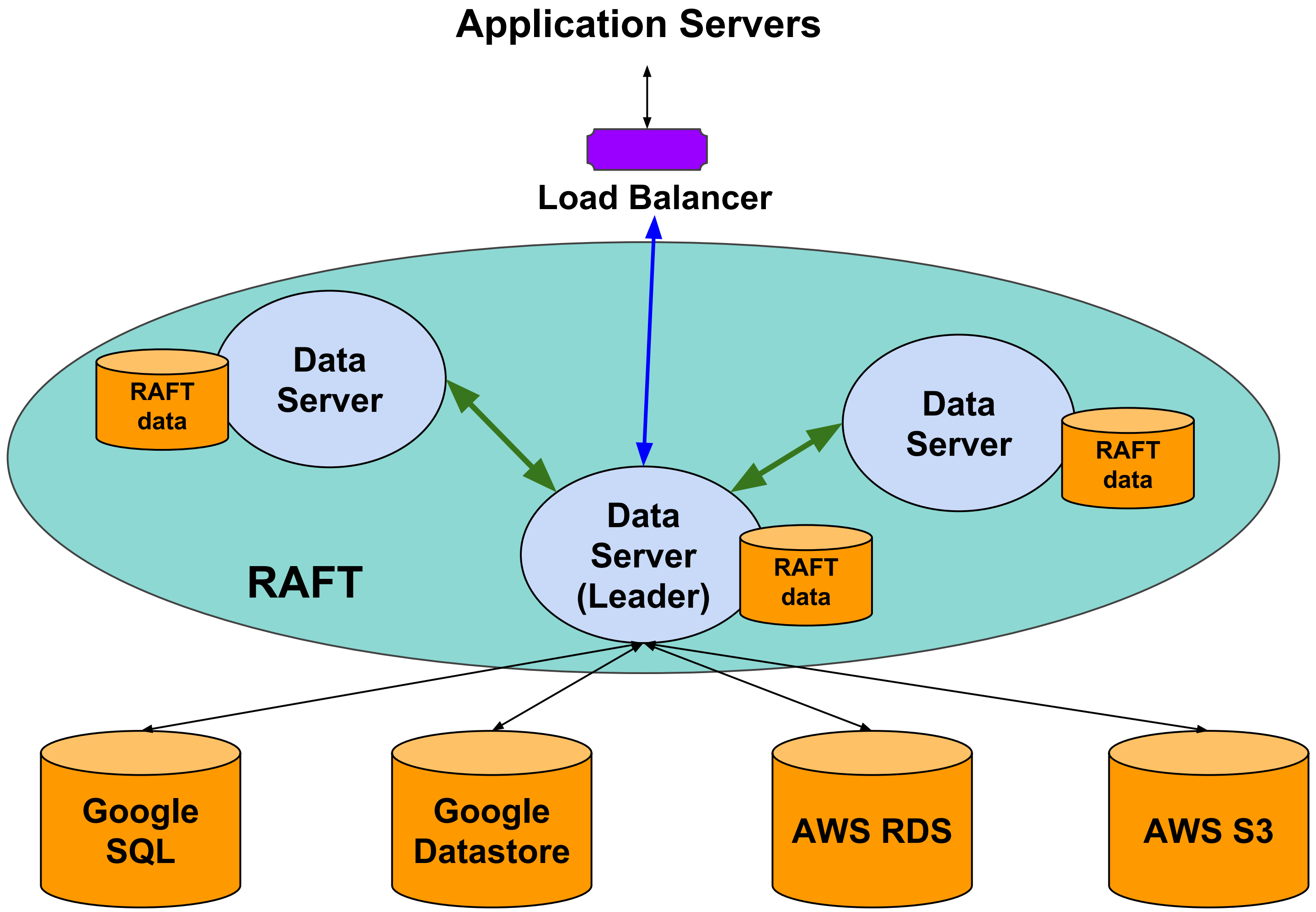
Goals of Sprint 2

- Mainly focused on the the data layer
- Load balance requests from the application servers
- Unified data layer that abstracts individual servers
- Data servers configured in a Raft cluster

Consensus (With Raft)

- Agreement on a unified cluster state by the cluster nodes
- Why? Prevent duplicate writes, defending against failover
- Using Raft -> simpler than Paxos, Good Hashicorp implementation in GO which was simple to modify
- Persistent store on each node with leader for state agreement (Term/Round, latest transaction counter and majority of nodes have it)
- How do we know which is accurate data? Leader or if no leader: term, latest transaction counter
- DBs can go down and need to be caught up


FE
Auto-Scaling Group (AWS)
Google Cloud
GCP CDN
Client
Application Server
Raft Cluster (Container Images)
Application Server
Load Balancer
Data Server
Raft Cluster (Container Images)
Data Server
Google Cloud SQL
Google Datastore
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


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

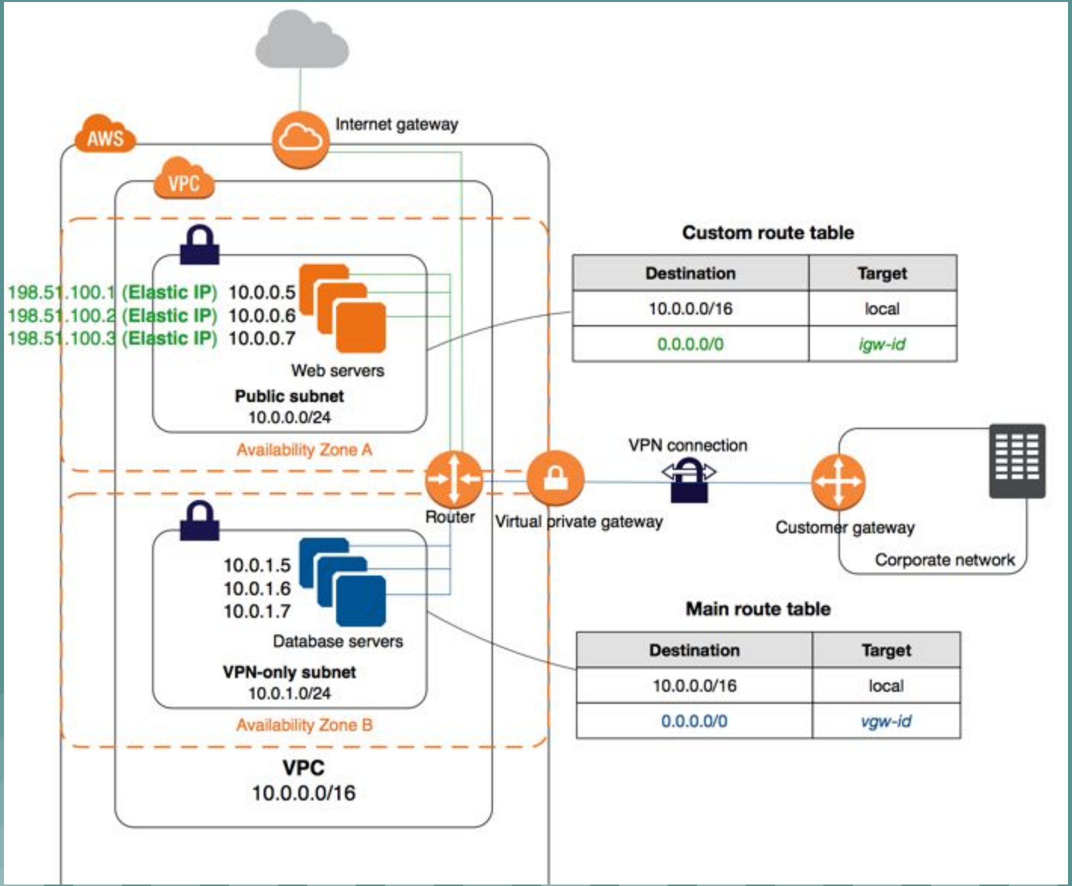
Data Layer Raft Cluster

- Assume fail-stop model
- To handle n failures, need 2n+1 nodes
- Leader of the cluster handles reads and writes
- Writes are committed to all nodes via consensus before an actual DB write happens
- GETs (theoretically) may be serviced by any node
- When a node comes back up, writes that have occured will be forwarded to it
- New leader will be elected if a leader dies

Cross Cloud Consensus

- Want to keep data servers within private VPC (Virtual Private Cloud)

- Need to be able to communicate to cross cloud VPCs using VPN (Virtual Private Network)



Custom route table

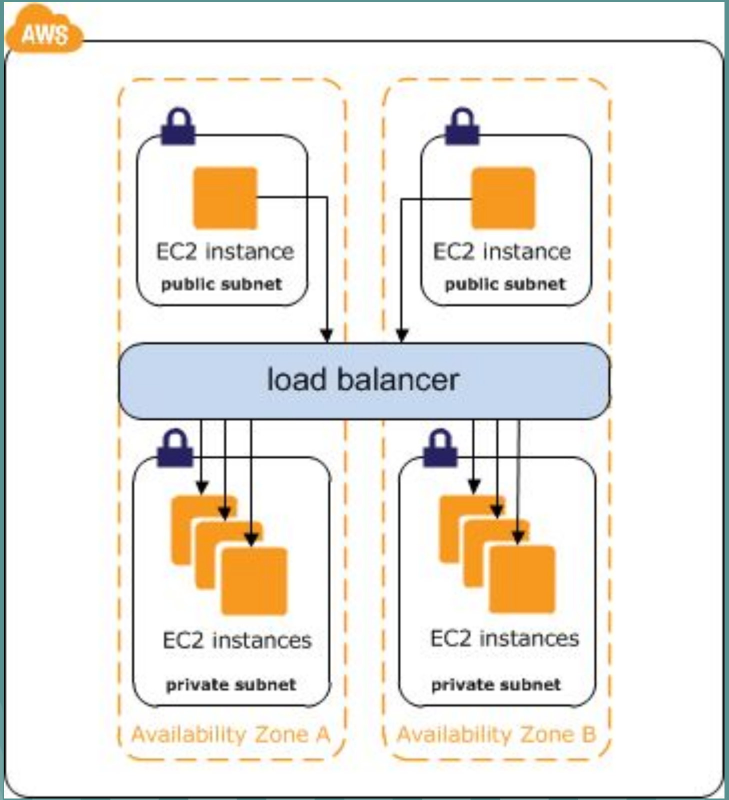
Destination	Target
10.0.0.0/16	local
0.0.0.0/0	*igw-id*

Main route table

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	*vgw-id*

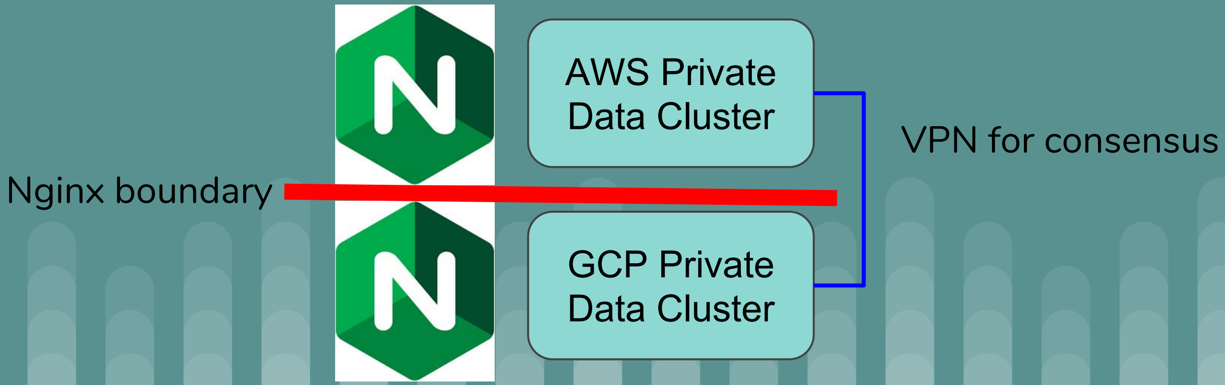
Cross Cloud Load Balancing

- Tried to do cross cloud load balancing
- AWS makes it very hard
- To load balance a private subnet, the load balancer must be private
- Heartbeat problems
- Easier solution: Nginx



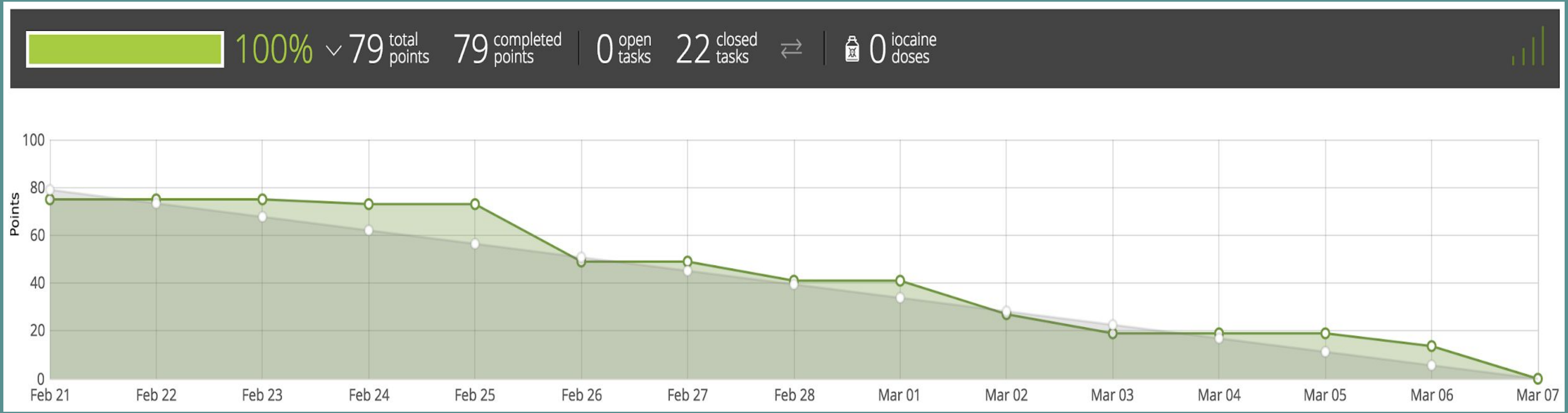
Single Cloud Load Balancing

- Load balance across each cloud separately using Nginx
- Use cross cloud VPN communication for (only) consensus on each layer
- Already implemented Nginx load balancing to dao servers which have consensus
    - Still needed: leader forwarding and replicating this on each layer.



DEMO TIME!

Sprint 2 Burndown


100% 79 total points 79 completed points 0 open tasks 22 closed tasks 0 iocaine doses

https://tree.taiga.io/project/bowenislandsong-multi-cloud-defensive-load-balancing/taskboard/sprint-2-7327

Next Steps

- Raft leader forwarding
- Detecting when databases are down
- Forwarding recovered databases to the current state
- Load balancing traffic from the application server to cross-cloud data layer

Thank you!
Questions?