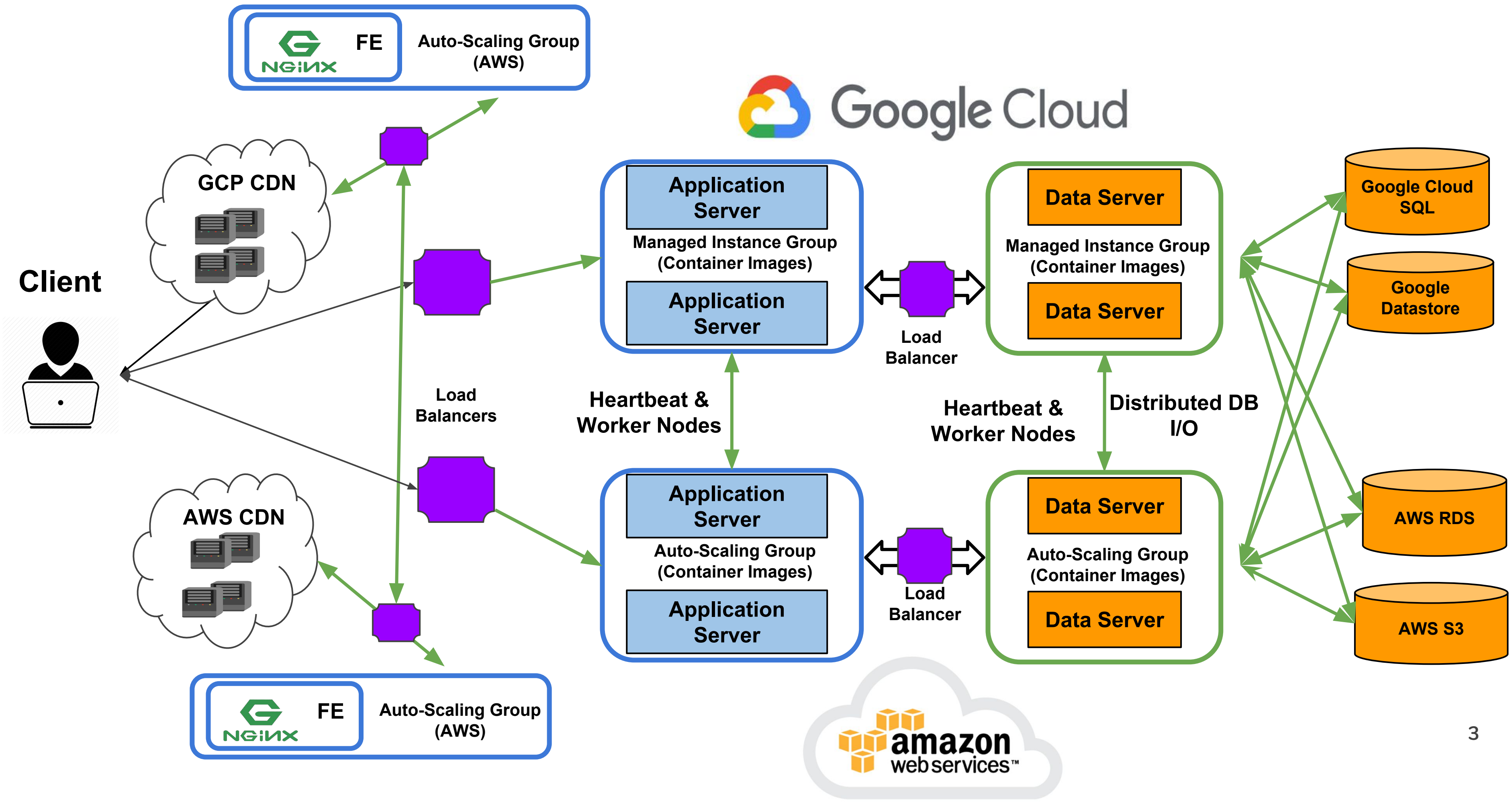
Cloud-Hydra:
A Cloud Native Multi-Cloud Defensive Load Balancing Framework

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

SPRINT 1

Goals of Cloud-Hydra

- To provide a framework in a cloud native package that will protect applications from cloud provider faults and DDoS attacks
- DNS Failover
- Request Load Balancing / cross cloud federation
- Distributed database writes with consensus for ordering
- Read from any available database


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

Sprint 1

Goal: Create Garage Reservation Application

Epic: Build and Deploy application on GCP and AWS

- Create Auto-Scaling Front-End Server with CDN for FE code
- Create Auto-Scaling Application Server
- Create SQL databases for storing user data
- Connect all these services within in each cloud
- Test each cloud application
- SPIKE for distributed consensus

Garage Reservation App

- **CRUD Application**
  - Users must be able to create, read, update, and delete information about their cars and reservations at different garages
  - Easy to use front-end that makes REST API calls to our backend server (HTML and JavaScript)
  - Back-end handles each CRUD operation for each object (cars, reservations, garages), written in **Golang**

Technology Stack -> Entirely on

(except the DB:)

AWS

- Compute - EC2 - minimal Linux 2 AMI with Docker image AMI
- FE - Nginx server in Docker Container serving FE code in auto scaling group with load balancer and
- CDN - AWS Cloudfront
- Application server - Go server Docker Container in ASG with Load Balancer
- DAO server - Go server Docker Container in ASG with Load Balancer
- Relational DB - PostgreSQL RDS
- Key-Value Storage: Dynamo

GCP

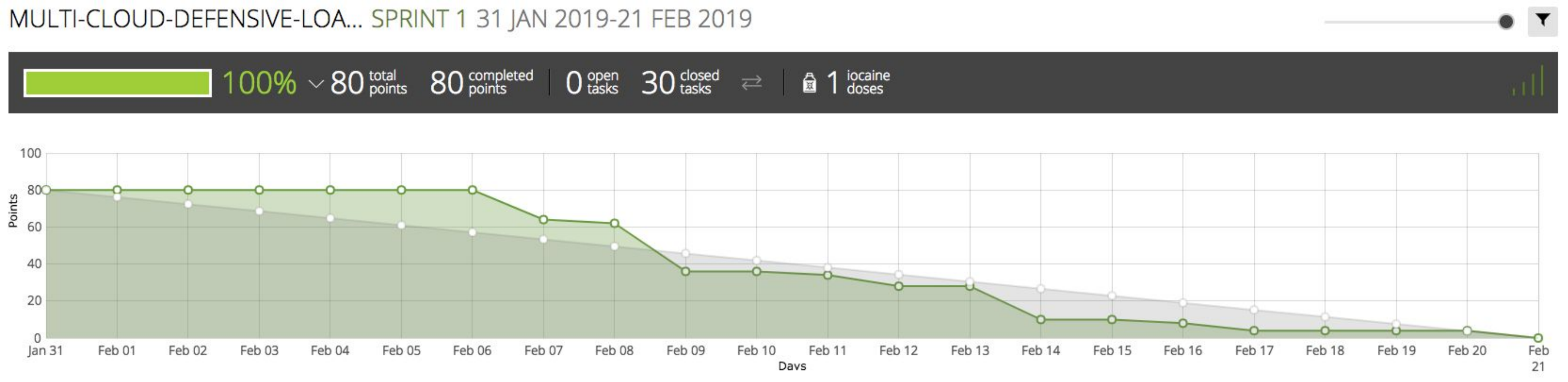
- Compute - GCP Compute Engine, via Docker Image based Managed Instance Groups
- FE - Nginx server in Docker Container serving FE code in MIG with load balancer and
- Google Cloud CDN
- Application server- Go server Docker Container in MIG with LB
- DAO server - Go server Docker Container in MIG with LB
- Relational DB - Google Cloud SQL
- Key-Value Storage: Google Cloud Datastore

Continuous Deployment

- Build Docker images and push to Docker Hub repo
- Create new instance template with new versions
- Managed Instance Groups / Auto-Scaling Groups Rolling updates with 0 downtime

STOP: DEMO TIME!

Sprint 1 Burndown



MULTI-CLOUD-DEFENSIVE-LOA... SPRINT 1 31 JAN 2019-21 FEB 2019

100% 80 total points 80 completed points 0 open tasks 30 closed tasks 1 iocaine doses

https://tree.taiga.io/project/bowenislandsong-multi-cloud-defensive-load-balancing/taskboard/sprint-1-13881

Next Steps

- DNS failover for request load balancing
- Load balance between clouds with health checks
- Travis CI for CI/CD
- Database distributed I/O between clouds (Raft vs Paxos)
- Impediments
  - Identifying technologies to use vs. in-house
  - Cross cloud communication
  - Capturing solutions into a generic framework

Thank you!
Questions?