MySQL High Availability at GitHub

Shlomi Noach
GitHub

dataops.barcelona 2018

Agenda

MySQL @ GitHub

The HA story

orchestrator

Old vs. new design

Testing

Thoughts

About me

@github/database-infrastructure

Author of **orchestrator**, **gh-ost**, **freno**, **ccql** and others.

Blog at http://openark.org

github.com/shlomi-noach
@ShlomiNoach

GitHub

Built for Developers

Largest open source hosting

67M repositories, 24M users

Critical path in build flows

Best octocat T-Shirts and stickers

MySQL at GitHub

Stores all the metadata: users, repositories, commits, comments, issues, pull requests, . . .

Serves web, API and auth traffic

MySQL 5.7, semi-sync replication, RBR, cross DC

~15 TB of MySQL tables

~150 production servers, ~15 clusters

Availability is critical

MySQL High Availability

We wish to have:

Automation, reliable detection, DC tolerant failovers, DC isolation tolerance, reasonable failover time, reliable failover, lossless where possible.

MySQL High Availability

Write HA/read HA

MySQL High Availability

Detection

Recovery

Master discovery

orchestrator

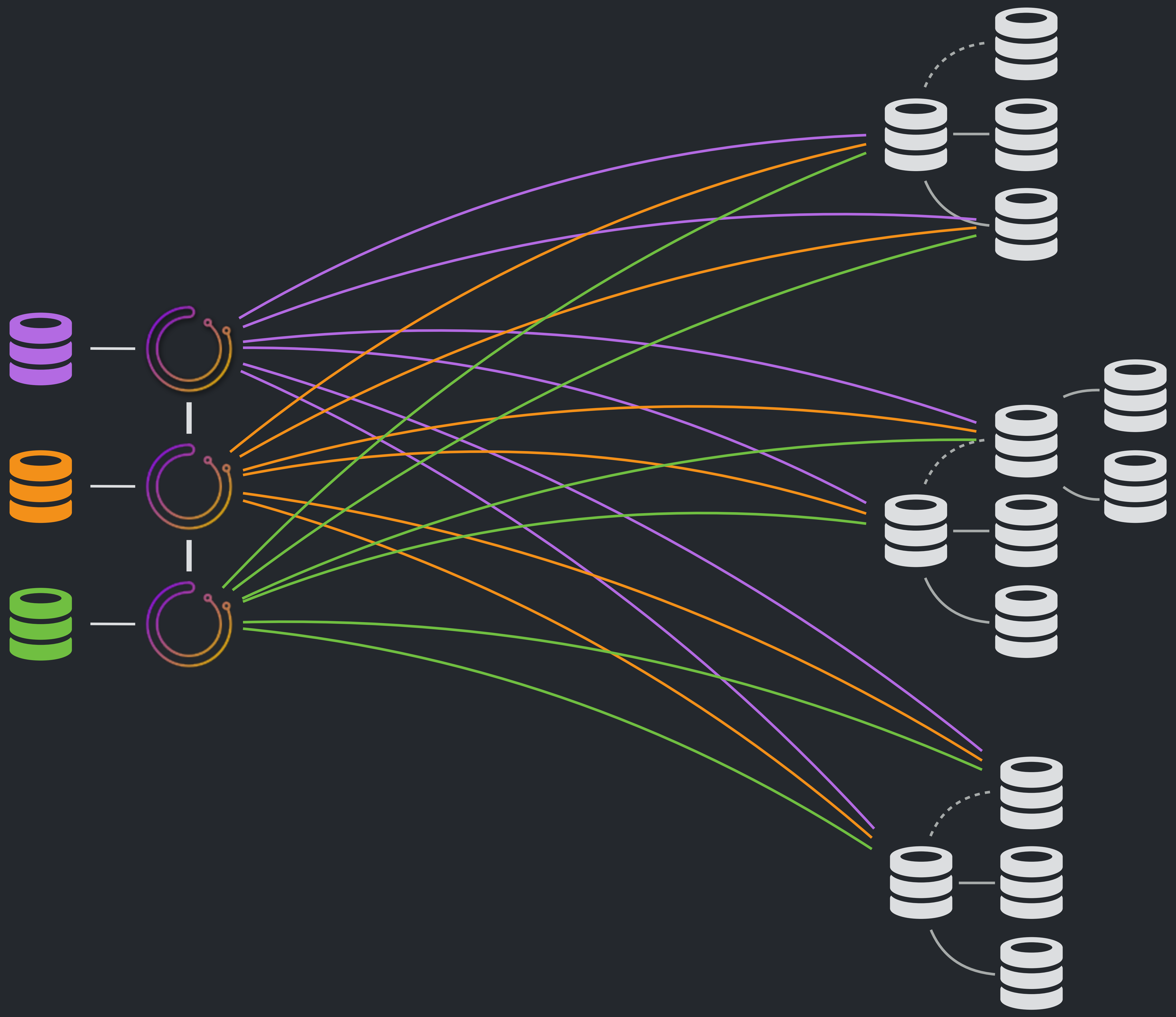
orchestrator, meta

Adopted, maintained & supported by GitHub,
github.com/github/orchestrator

Previously at Outbrain and Booking.com

Orchestrator is free and open source, released under the Apache 2.0 license
github.com/github/orchestrator/releases

Gaining wider adoption

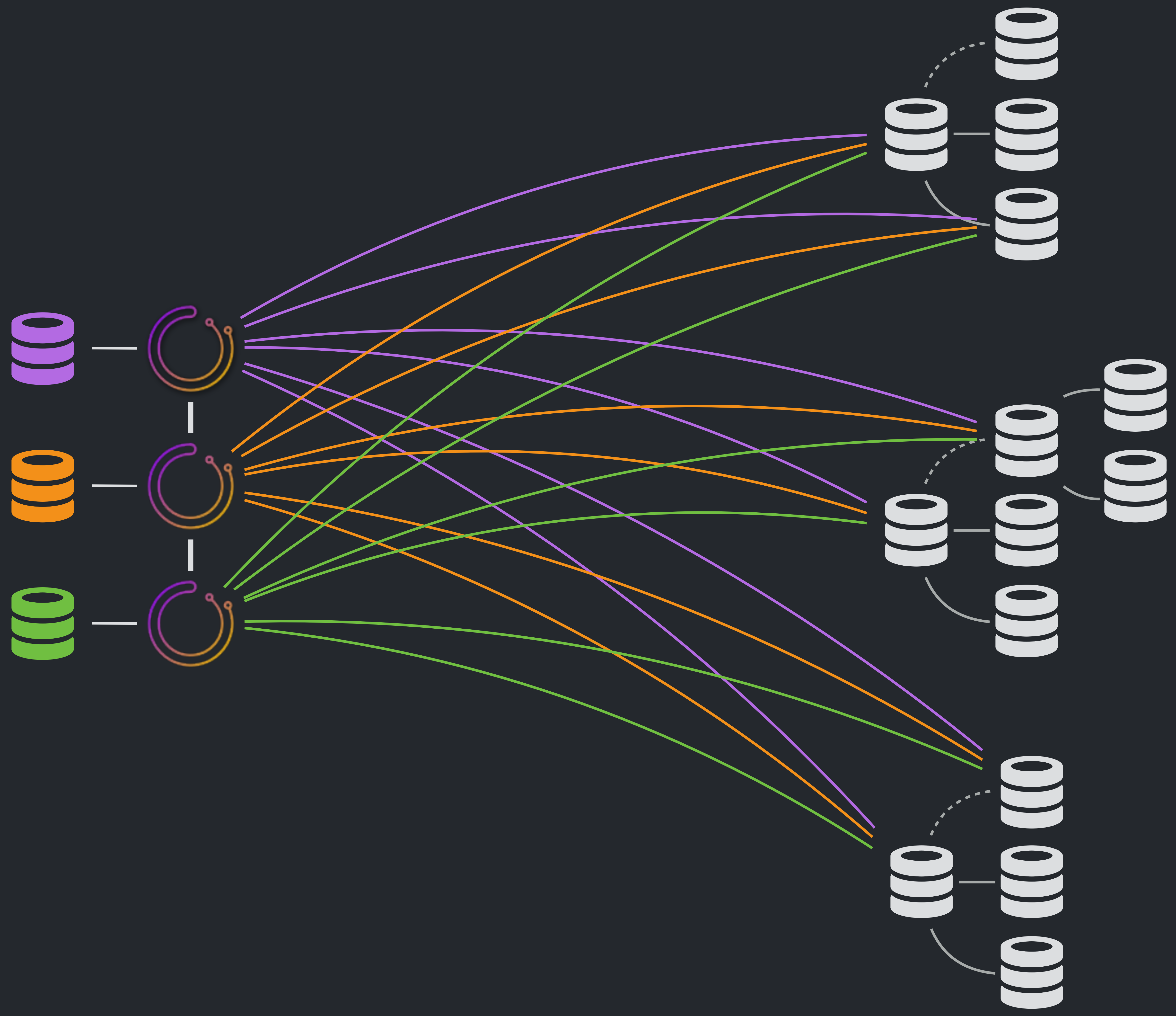


orchestrator

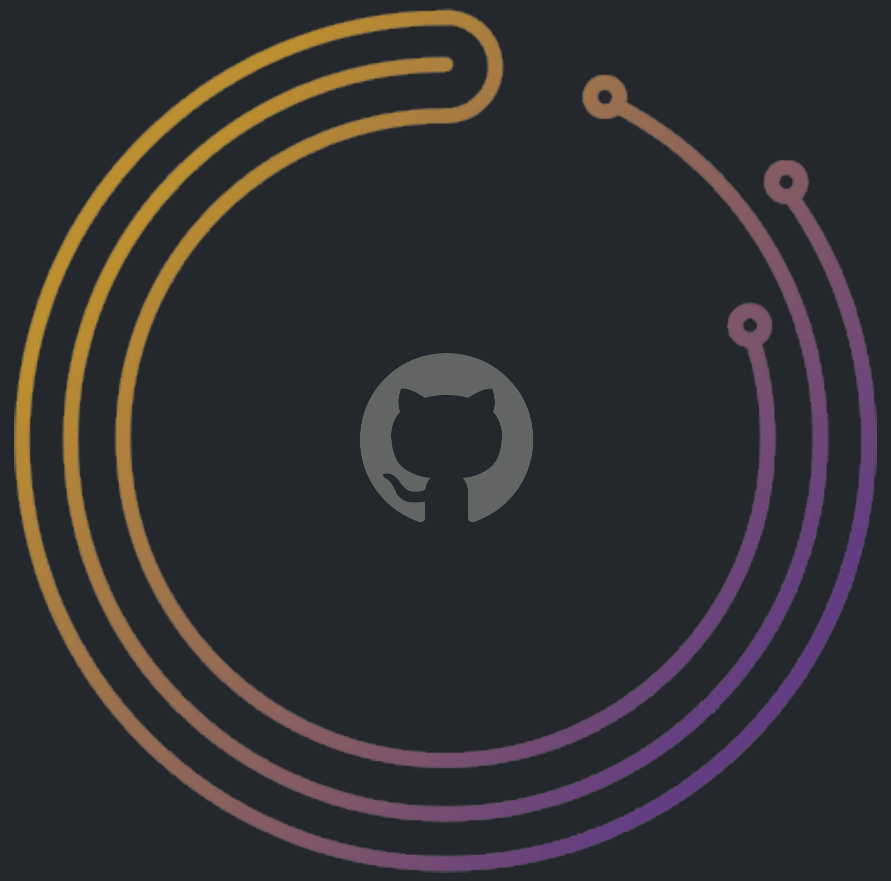
Discovery
Probe, read instances, build topology graph, attributes, queries

Refactoring
Relocate replicas, manipulate, detach, reorganize

Recovery
Analyze, detect crash scenarios, structure warnings, failovers, promotions, acknowledgements, flap control, downtime, hooks



orchestrator @ GitHub



orchestrator/raft deployed on 3 DCs

Automated failover for masters and intermediate masters

Chatops integration

orchestrator
Home
Clusters
Audit
Search
50s
Smart Mode

instance-69ac:3306
5.7 ROW/F 0s lag
Master

instance-fd21:3306
5.7 ROW/F 1s lag

instance-af50:3306
5.7 ROW/F 0s lag

instance-951e:3306
5.7 ROW/F 0s lag

instance-1c4e:3306
5.7 ROW/F 0s lag

instance-d783:3306
5.7 ROW/F

instance-e0aa:3306
5.7

instance-d54a:3306
5.7 ROW/F

instance-5807:3306
5.7 ROW/F

How is orchestrator different?

Holistic approach to failure detection

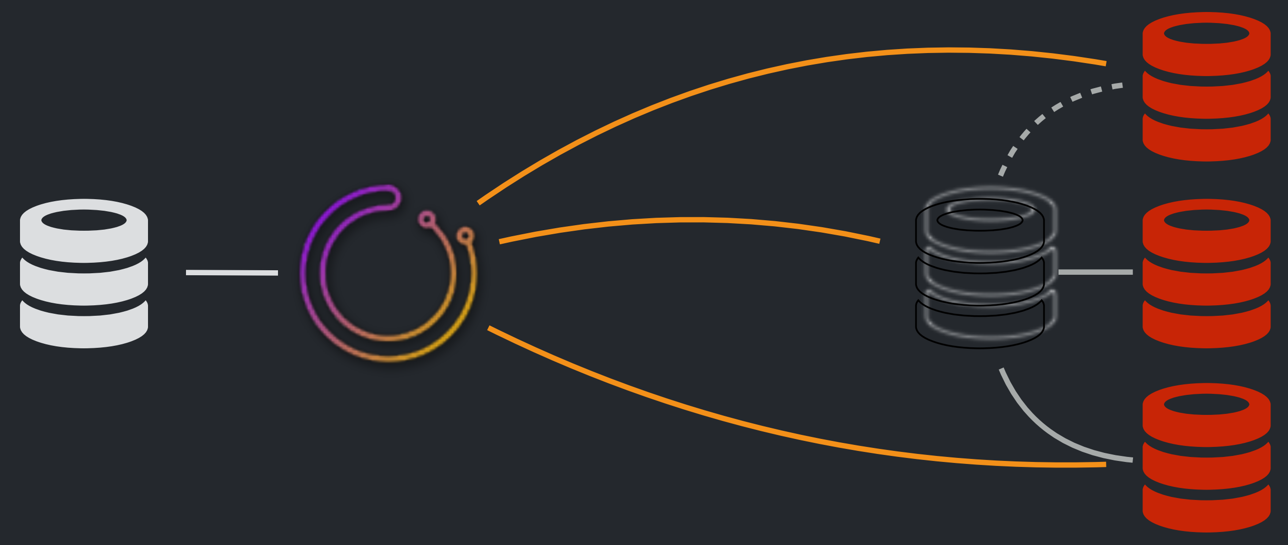
State based, elaborate recovery decision making

Detection: naive approach

Probe the master

Test failure? Try again n times, interval i

Detection: holistic approach



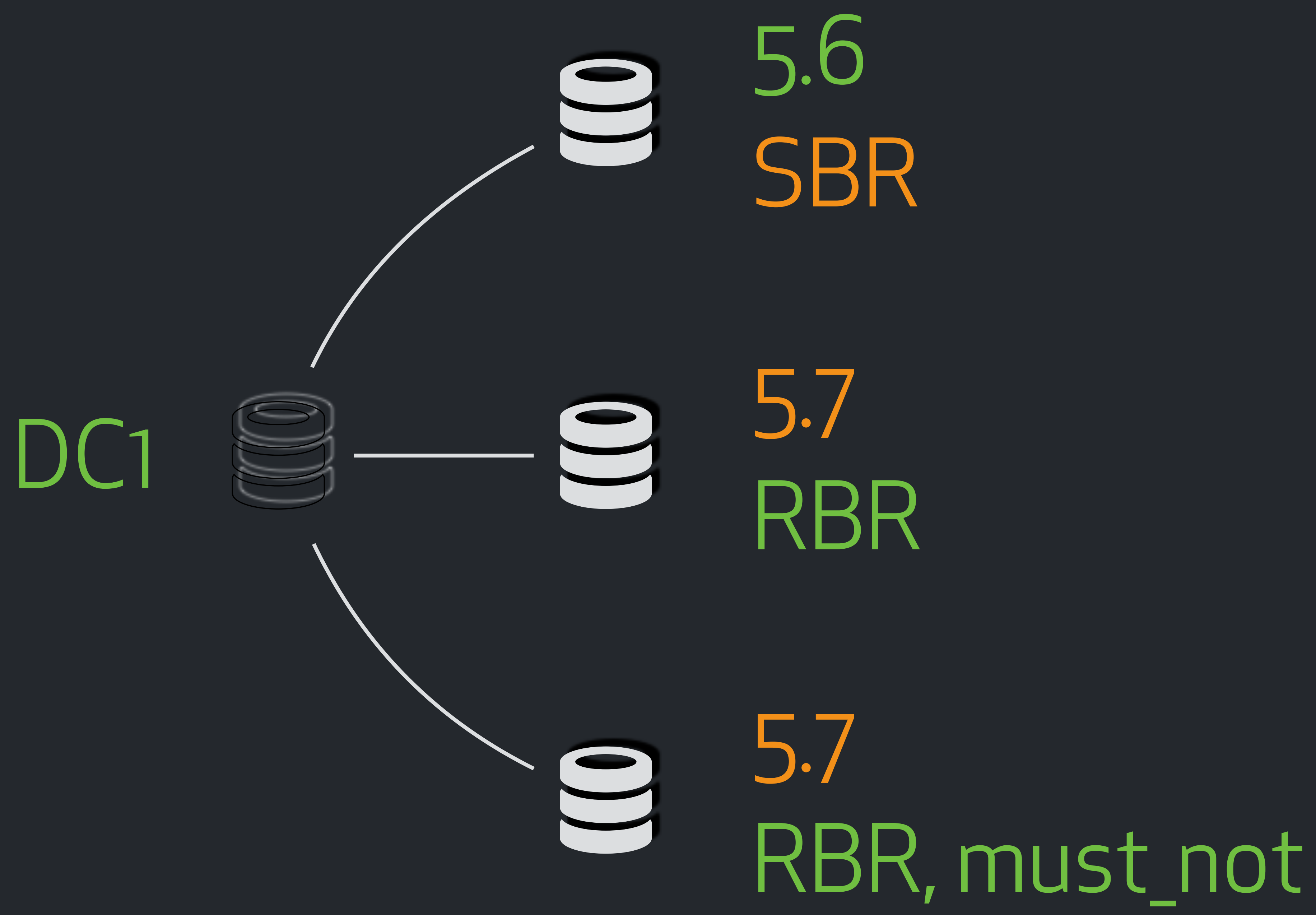
orchestrator:

Probe the master and its replicas

Expect agreement

Agreement achieved? The cluster is de-facto down.

Promotion: naive assumptions

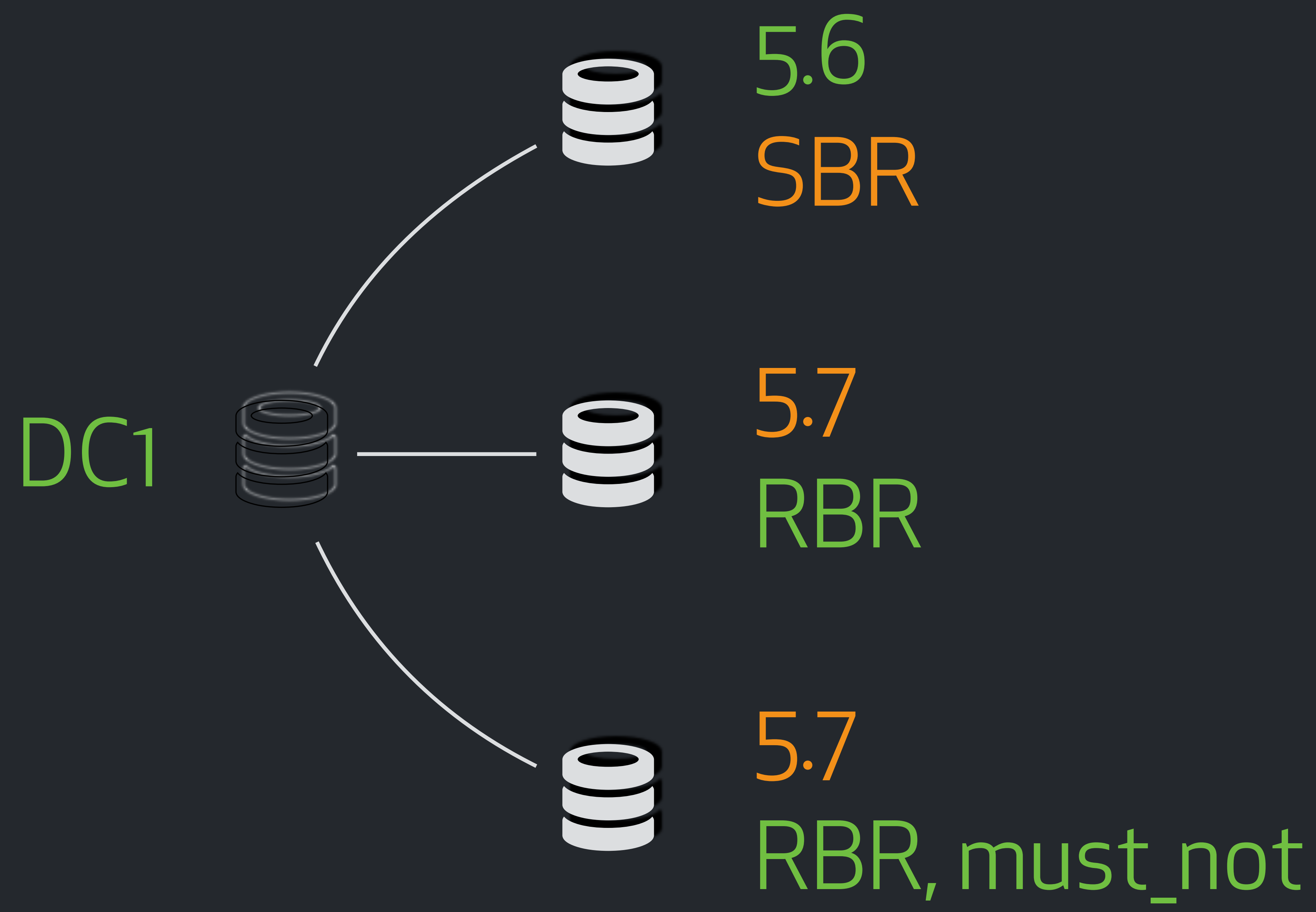


Configuration indicates which servers whitelisted or blacklisted.

Production operations must reflect in configuration changes.

Promote the most up-to-date replica.

Promotion constraints: real life



orchestrator:

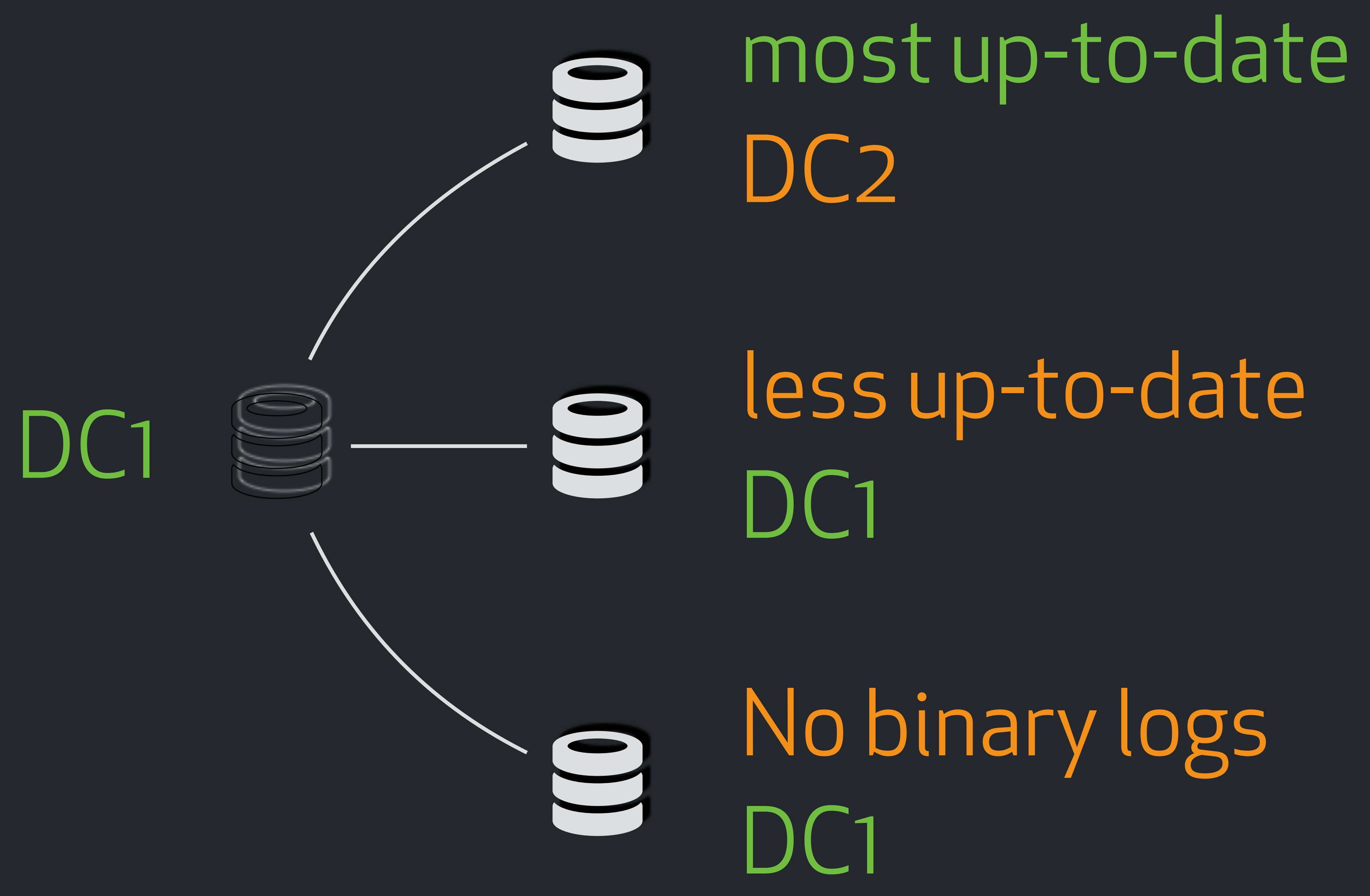
Recognizes environments are dynamic

Understands replication rules

Resolves version, DC, config, promotion rules

Acts based on state

Promotion constraints: real life



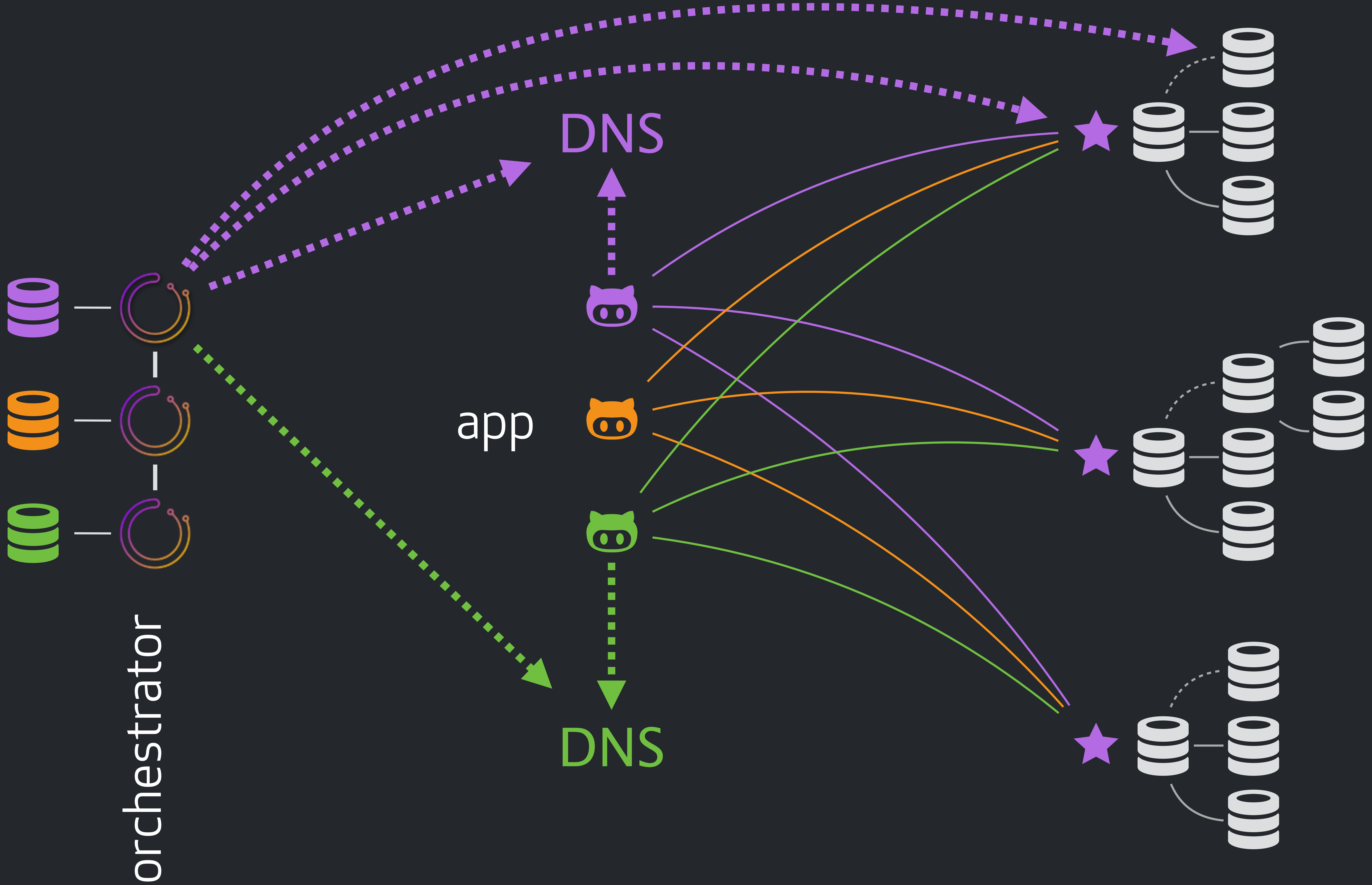
orchestrator can promote one, non-ideal replica, have the rest of the replicas converge,

and then *refactor again*, promoting an *ideal* server.

Earlier master discovery @ GitHub

VIP + DNS based

Master discovery via VIP+DNS



Earlier master discovery @ GitHub

Cooperative, long, not a good cross-DC story

A better story

GLB/HAProxy

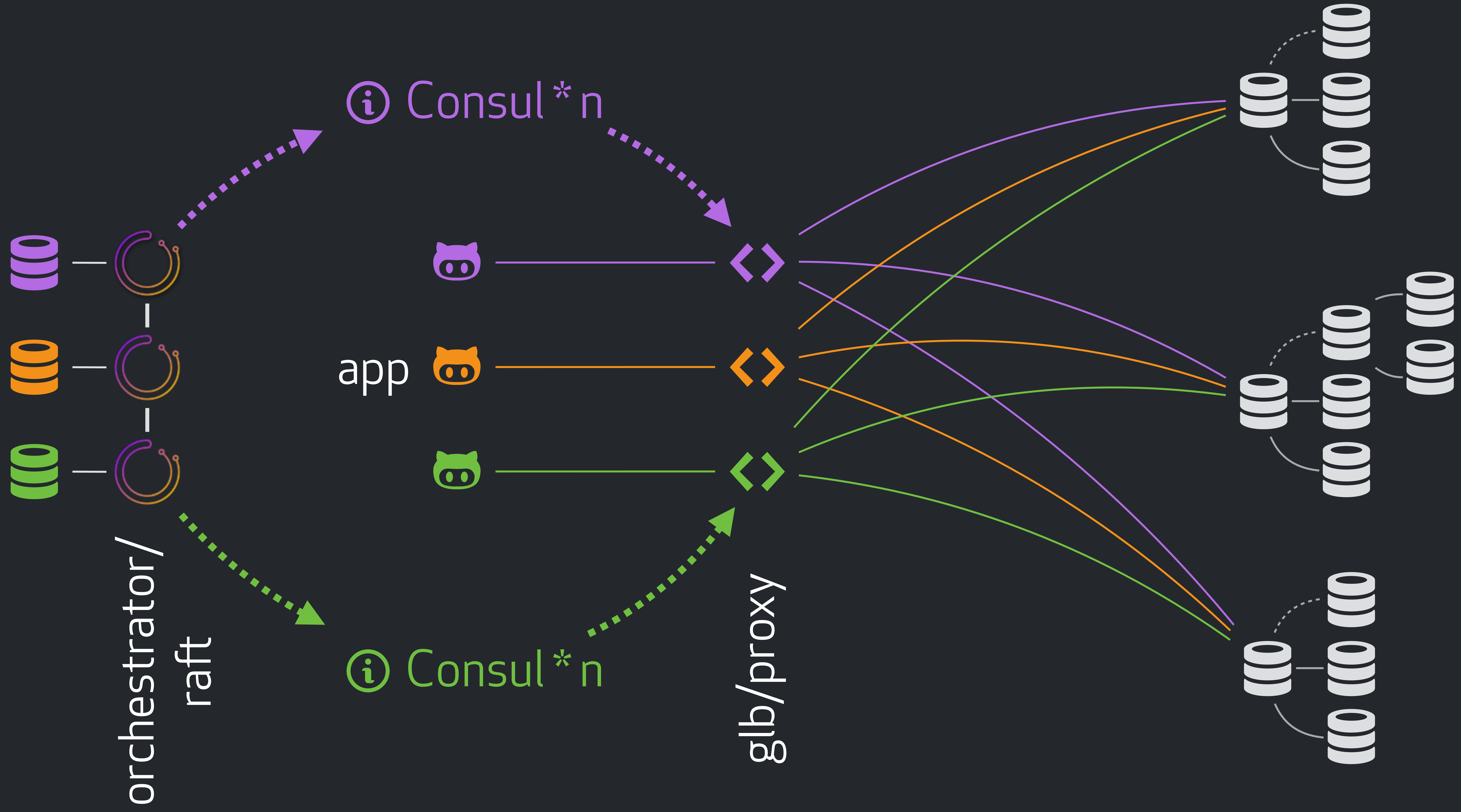
anycast

Consul

orchestrator

semi-synchronous replication

orchestrator/Consul/GLB(HAProxy) @ GitHub



A better story

More components, but less moving parts.

Better ownership

Decoupling

> with ♥ by GitHub

glb

GLB

High available, scalable proxy array

Lossless reloads, implicit SSL, consul integration

GLB director, load balancer array via HAProxy
https://githubengineering.com/introducing-glb/
https://githubengineering.com/glb-part-2-haproxy-zero-downtime-zero-delay-reloads-with-multibinder/

Consul

By HashiCorp
https://consul.io/

Mozilla Public License 2.0
https://github.com/hashicorp/consul

Consul

Service Discovery

Health checks, DNS, KV storage

Highly available

consul-template

Simple template engine

Listens to Consul updates

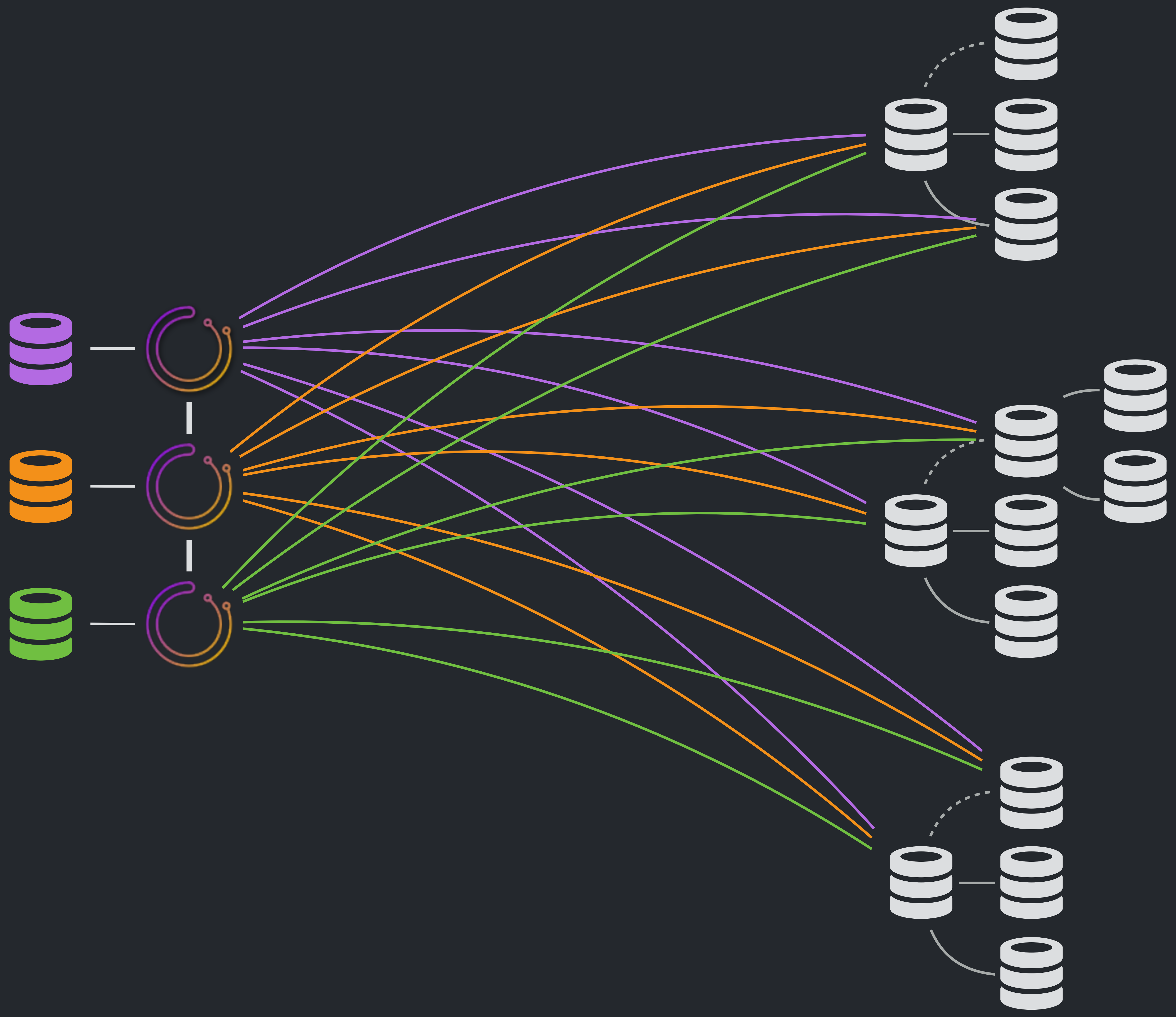
orchestrator/raft

A highly available **orchestrator** setup

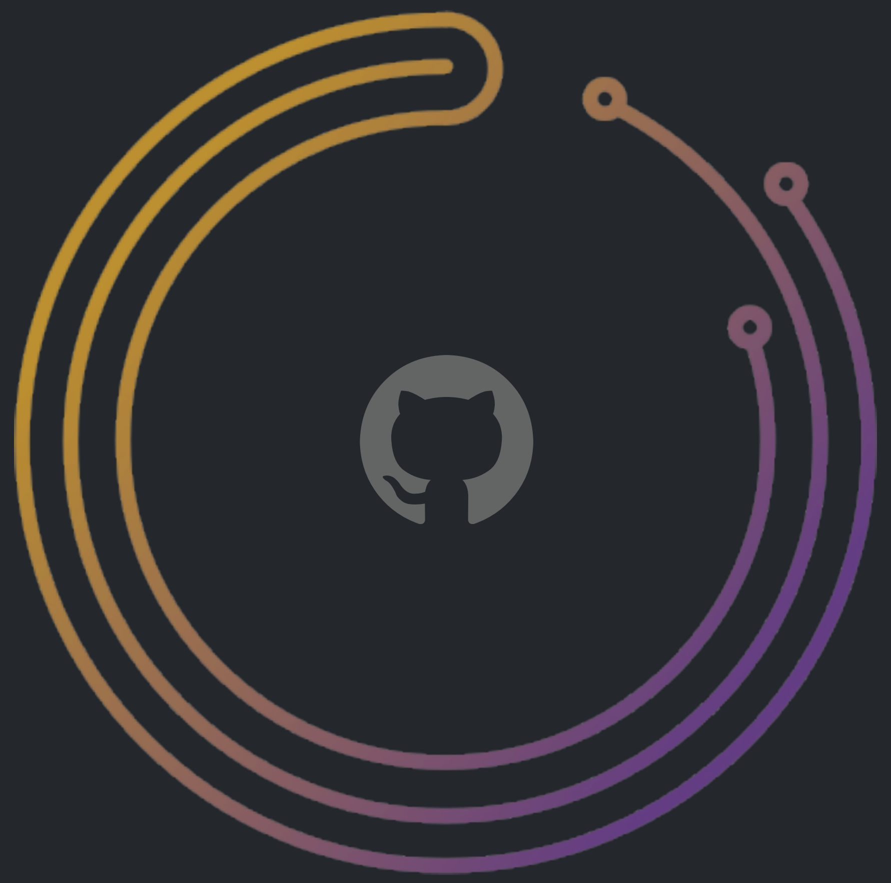
Self healing

Cross DC

Mitigates DC partitioning



orchestrator/raft



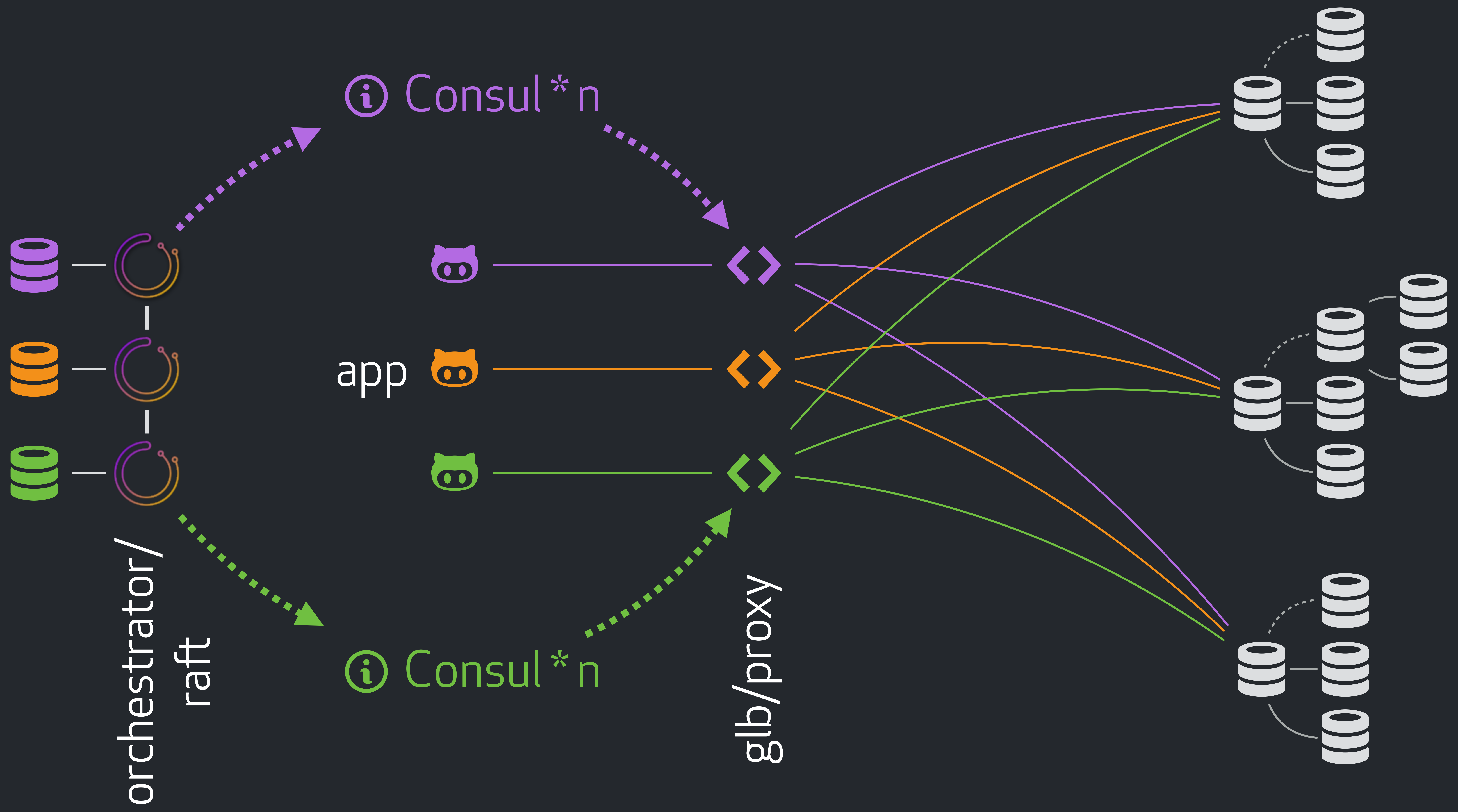
2 nodes per DC + mediator

more to come

Added raft features:

- Step down
- Yield
- SQLite log store

orchestrator/Consul/GLB(HAProxy) @ GitHub



orchestrator/Consul/GLB(HAProxy) @ GitHub

orchestrator owns recovery, updates Consul

consul-template runs on GLB servers, reconfigures & reloads GLB

GLB reroutes connections

  Hard-kill old connections

Apps connect via anycast, route through local GLB

Independent Consul deployments per DC are managed by orchestrator/raft

pt-heartbeat

Runs on all boxes

Longer poll intervals on **read_only**, supports going in and out of **read_only** mode. Contributed upstream.
https://github.com/percona/percona-toolkit/pull/302/files?w=1

No need to start/stop services remotely

Pseudo-GTID

Auto generated by **orchestrator**

Automatically injects on promoted master. No need to start/stop services remotely.

semi-synchronous replication

Lossless, best effort

500ms timeout

Effectively picks our ideal candidates

Results

Reliable detection

Recovery in:

10s - 13s (total outage time) , normal case

15s - 20s, difficult case

25s, rare

Cons

App identity unkown

Distributed system, calls for a variety of scenarios

STONITH, work in progress

Testing

Testing cluster in production environment

Continuously kill/block/reject

Thoughts

STONITH

Retries

orchestrator + Consul + proxy as appliance

Kubernetes

Thank you!

Questions?

github.com/shlomi-noach
@ShlomiNoach