3 December
CONTAINERS TODAY Stockholm
Powered by
Coro3 LEF marketing & events
REDIS
RUN THE MOST LOVED DATABASE IN YOUR CONTAINERS INFRASTRUCTURE

Tug Grall
Redis Labs
@tgrall
About me

• **Tugdual “Tug” Grall**
  • Redis Labs (TAM)
  • Red Hat (PM Director Dev Experience)
  • MapR (DevRel & PM)
  • MongoDB (DevRel)
  • Couchbase (DevRel)
  • eXo Platform (CTO)
  • Oracle (PM & Soft Engineer)

• **Pet Projects**
  • [https://promoglisse-speed-challenge.com](https://promoglisse-speed-challenge.com)
  • Promoglisse Mobile App (iOS/Android)
How popular is Redis?

**MOST LAUNCHED**

Launches/day: Docker Hub, Nov. 2019

**MOST USED**

Sumo Logic, Sept. 2019

**MOST LOVED**

Stack Overflow, 2017, 2018, 2019
Why?
Why Redis?

Humans

Apps

Machines

App servers

Database

50ms

50ms

100ms
Why Redis?

Humans → Cloud → App servers → Machines

- 100ms
- 50ms
- 50ms
- 1ms

App servers

Humans

Apps

Machines
### Use cases?

<table>
<thead>
<tr>
<th><strong>OPERATIONAL</strong></th>
<th><strong>ANALYTICS</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>User Session Store</td>
<td>Real Time Analytics</td>
</tr>
<tr>
<td>Content Caching</td>
<td>Complex Statistical Analysis</td>
</tr>
<tr>
<td>High Speed Transactions</td>
<td>Leaderboards</td>
</tr>
<tr>
<td>Job &amp; Queue Management</td>
<td>Time Series Data</td>
</tr>
<tr>
<td>Real Time Data Ingest</td>
<td>Location-based Processing</td>
</tr>
<tr>
<td>Time Series Data</td>
<td>Recommendation Engine</td>
</tr>
<tr>
<td>Notifications</td>
<td>Real Time Data Ingest</td>
</tr>
<tr>
<td>Distributed Lock</td>
<td>Page Ranking</td>
</tr>
<tr>
<td>Search</td>
<td>Counting</td>
</tr>
<tr>
<td>Location-based Processing</td>
<td>Search &amp; Secondary Index</td>
</tr>
<tr>
<td>Streaming Processing</td>
<td>ML/DL Serving</td>
</tr>
</tbody>
</table>
Multi-Model Database
https://oss.redislabs.com/

Redis Streams: (Redis Core)
XADD mystream * sensor-id 1234 temperature 19.8
XRANGE mystream - + COUNT 2

RediSearch:
FT.SEARCH myIdx "hello world" LIMIT 0 10
FT.SUGGET autocomplete "he"

Redis TimeSeries:
TS.ADD temperature:3:11 1548149181 30
TS.RANGE temperature:3:11 1548149180 1548149210 AGGREGATION avg 5
Multi-Model Database
https://oss.redislabs.com/

**RedisGraph:** *(Cipher query)*
GRAPH.QUERY MotoGP "MATCH (r:Rider)-[:rides]->(t:Team) WHERE t.name = 'Yamaha' RETURN r.name, t.name"

**RedisJSON:**
JSON.SET foo . "bar"
JSON.NUMINCRBY num . 1.5
JSON.ARRPOP arr

**RedisBloom:**
BF.ADD newFilter foo
BF.EXISTS newFilter foo
BF.MEXISTS myFilter foo nonexistent bar
Redis core
- Strings
- Bitmaps
- Bit Field
- Hashes
- Lists
- Sets
- Sorted Sets
- Geospatial
- Hyperloglog
- Streams

Redis Enterprise

Redis modules
- JSON
- Search
- Graph
- Gears
- BloomFilter
- AI
- TimeSeries

Redis Enterprise core

Features
- Linear scalability
- HA
- Durability
- Backup & restore
- Geo-Distribution
- Tiered-memory access
- Multi-tenant
- Security
RedisInsight : A standard UI for Redis

- Developer-friendly UI
- Adds insight to existing and new data structures: Streams, Search, JSON, Graph, Time-series
- Enhanced memory-analysis and recommendation capabilities
- Admin monitoring, commands tracer, CLI interface, bulk operations

redislabs.com/redisinsight
How?
Multiple Delivery Models

**DATABASE-AS-A-SERVICE**

- Cloud Essentials
  - QA & Dev, Small Prod Workloads
- Cloud Pro, Ultimate
  - Production, Hosted or in customer’s VPC

**SOFTWARE**

- Downloadable
  - Native Kubernetes Essentials
- PCF over BOSH or PKS Essentials
- OpenShift over Kubernetes
Develop with Redis Container
Run Redis locally easily

• Images available on Docker Hub:
  – Redis Enterprise + Modules: `docker pull redislabs/redis`
  – Redis OSS Core: `docker pull redis`
  – RedisSearch: `docker pull redislabs/redisearch`
  – Redis TimeSeries: `docker pull redislabs/redistimeseries`
  – Redis Graph: `docker pull redislabs/redisgraph`
  – Redis Bloom: `docker pull redislabs/rebloom`
  – Redis JSON: `docker pull redislabs/rejson`
  – Redis Insight: `docker pull redislabs/redisinsight`

• Or From RedisLabs download center:
  – Docker, Kubernetes, PCF: [https://redislabs.com/download-center/](https://redislabs.com/download-center/)
Redis & Docker

<table>
<thead>
<tr>
<th>NAMES</th>
<th>IMAGE</th>
<th>CONTAINER ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>competent_pike</td>
<td>redis</td>
<td>56181b0a2fc9</td>
</tr>
<tr>
<td>priceless_germain</td>
<td>redislabs/redisearch</td>
<td>972f21d0fc62</td>
</tr>
<tr>
<td>hardcore_agnesi</td>
<td>redislabs/redisgraph</td>
<td>b7ba09dfff7f1</td>
</tr>
<tr>
<td>friendly_merkle</td>
<td>redislabs/rebloom</td>
<td>8bd7a3e7f2ce</td>
</tr>
<tr>
<td>happy_meninsky</td>
<td>redislabs/rejson</td>
<td>d9d9603e60d4</td>
</tr>
<tr>
<td>sweet_ishizaka</td>
<td>redislabs/redistimeseries</td>
<td>189314ba7475</td>
</tr>
<tr>
<td>redis-node3</td>
<td>redislabs/redis</td>
<td>4702dc1220ea</td>
</tr>
<tr>
<td>redis-node2</td>
<td>redislabs/redis</td>
<td>62857eceb312</td>
</tr>
<tr>
<td>redis-node1</td>
<td>redislabs/redis</td>
<td>febab4e226d0</td>
</tr>
</tbody>
</table>

```
- docker ps --format "table {{.Names}}\t{{.Image}}\t{{.ID}}"
```
Run a Redis Enterprise Cluster with Docker

http://tgrall.github.io/blog/2019/09/05/multi-nodes-redis-cluster-with-docker/
Run a Redis Enterprise Cluster with Docker

```
redislabs@6ca09094c1fc:/opt$ radmin status

CLUSTER NODES:

<table>
<thead>
<tr>
<th>NODE ID</th>
<th>ROLE</th>
<th>ADDRESS</th>
<th>EXTERNAL ADDRESS</th>
<th>HOSTNAME</th>
<th>SHARDS</th>
<th>CORES</th>
<th>RAM</th>
<th>AVAILABLE_RAM</th>
<th>VERSION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>node:1</td>
<td>master</td>
<td>172.17.0.2</td>
<td>6ca09094c1fc</td>
<td>3/100</td>
<td>5</td>
<td>4.4GB/9.74GB</td>
<td>2.6GB/7.99GB</td>
<td>5.4.6-18</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>node:2</td>
<td>slave</td>
<td>172.17.0.3</td>
<td>99a6a9fb2054</td>
<td>4/100</td>
<td>5</td>
<td>4.4GB/9.74GB</td>
<td>2.35GB/7.99GB</td>
<td>5.4.6-18</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>node:3</td>
<td>slave</td>
<td>172.17.0.4</td>
<td>3afeaf33db0</td>
<td>1/100</td>
<td>5</td>
<td>4.4GB/9.74GB</td>
<td>2.44GB/7.99GB</td>
<td>5.4.6-18</td>
<td>OK</td>
<td></td>
</tr>
</tbody>
</table>

DATABASES:

<table>
<thead>
<tr>
<th>DB ID</th>
<th>NAME</th>
<th>TYPE</th>
<th>STATUS</th>
<th>SHARDS</th>
<th>PLACEMENT</th>
<th>REPLIICATION</th>
<th>PERSISTENCE</th>
<th>ENDPOINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>db:1</td>
<td>db-001</td>
<td>active</td>
<td>enabled</td>
<td>dense</td>
<td>enabled</td>
<td>disabled</td>
<td>redis-12000.redislabs.cluster.demo.com:12000</td>
<td></td>
</tr>
<tr>
<td>db:2</td>
<td>db-002</td>
<td>active</td>
<td>enabled</td>
<td>dense</td>
<td>enabled</td>
<td>disabled</td>
<td>redis-14874.redislabs.cluster.demo.com:14874</td>
<td></td>
</tr>
</tbody>
</table>

ENDPOINTS:

<table>
<thead>
<tr>
<th>DB ID</th>
<th>NAME</th>
<th>ID</th>
<th>NODE</th>
<th>ROLE</th>
<th>SSL</th>
</tr>
</thead>
<tbody>
<tr>
<td>db:1</td>
<td>db-001</td>
<td>endpoint:1:1</td>
<td>node:1</td>
<td>single</td>
<td>No</td>
</tr>
<tr>
<td>db:2</td>
<td>db-002</td>
<td>endpoint:2:1</td>
<td>node:3</td>
<td>single</td>
<td>No</td>
</tr>
</tbody>
</table>

SHARDS:

<table>
<thead>
<tr>
<th>DB ID</th>
<th>NAME</th>
<th>ID</th>
<th>NODE</th>
<th>ROLE</th>
<th>SLOTS</th>
<th>USED MEMORY</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>db:1</td>
<td>db-001</td>
<td>redis:1</td>
<td>node:1</td>
<td>slave</td>
<td>0-5460</td>
<td>2.9MB</td>
<td>OK</td>
</tr>
<tr>
<td>db:1</td>
<td>db-001</td>
<td>redis:2</td>
<td>node:2</td>
<td>master</td>
<td>0-5460</td>
<td>2.9MB</td>
<td>OK</td>
</tr>
<tr>
<td>db:1</td>
<td>db-001</td>
<td>redis:3</td>
<td>node:1</td>
<td>slave</td>
<td>5461-10922</td>
<td>2.9MB</td>
<td>OK</td>
</tr>
<tr>
<td>db:1</td>
<td>db-001</td>
<td>redis:4</td>
<td>node:2</td>
<td>master</td>
<td>5461-10922</td>
<td>2.9MB</td>
<td>OK</td>
</tr>
<tr>
<td>db:1</td>
<td>db-001</td>
<td>redis:5</td>
<td>node:1</td>
<td>slave</td>
<td>10923-16383</td>
<td>2.9MB</td>
<td>OK</td>
</tr>
<tr>
<td>db:1</td>
<td>db-001</td>
<td>redis:6</td>
<td>node:2</td>
<td>master</td>
<td>10923-16383</td>
<td>2.98MB</td>
<td>OK</td>
</tr>
<tr>
<td>db:2</td>
<td>db-002</td>
<td>redis:7</td>
<td>node:3</td>
<td>master</td>
<td>0-16383</td>
<td>2.9MB</td>
<td>OK</td>
</tr>
<tr>
<td>db:2</td>
<td>db-002</td>
<td>redis:8</td>
<td>node:2</td>
<td>slave</td>
<td>0-16383</td>
<td>2.9MB</td>
<td>OK</td>
</tr>
</tbody>
</table>
```

http://tgrall.github.io/blog/2019/09/05/multi-nodes-redis-cluster-with-docker/
Redis & Kubernetes
Redis Enterprise Scalable Architecture on Kubernetes

Redis Enterprise Node

Zero Latency Proxy

Redis Shards

Secured UI/REST API/CLI

Cluster Manager

Redis Enterprise Cluster

Fully compatible with open source commands & data structures
Redis Enterprise & Kubernetes: Architecture

Redis Enterprise

1 Pod, multiple services

- Resources efficiency: high
- Performance: high
- Isolation: high

Traditional

Multiple Pods, 1 service

- Resources efficiency: low
- Performance: medium
- Isolation: high
Kubernetes Operator

> kubectl apply -f redis-enterprise-operator.yml
> kubectl apply -f redis-enterprise-crd.yml
> kubectl apply -f redis-enterprise-cr.yml
8 principles for managing production Redis in a Kubernetes environment
8 principles for managing production **Redis** in a k8s environment

1. Dual orchestrator

- Node auto-healing
- Node scaling
- Shards failover & scaling
- Configuration & monitoring
- Service discovery
- Upgrade
- Automated recovery
8 principles for managing production Redis in a k8s environment

1. Dual orchestrator
2. Uneven number of pods
8 principles for managing production Redis in a k8s environment

1. Dual orchestrator
2. Uneven number of Pods
3. Stateful set
8 principles for managing production Redis in a k8s environment

1. Dual orchestrator
2. Uneven number of Pods
3. Stateful set
4. Anti-affinity
8 principles for managing production Redis in a k8s environment

1. Dual orchestrator
2. Uneven number of Pods
3. Stateful set
4. Anti-affinity
5. Multi-AZ (failure domains)
8 principles for managing production Redis in a k8s environment

1. Dual orchestrator
2. Uneven number of Pods
3. Stateful set
4. Anti-affinity
5. Multi-AZ (failure domains)
6. Rolling upgrade
8 principles for managing production **Redis** in a k8s environment

1. Dual orchestrator
2. Uneven number of Pods
3. Stateful set
4. Anti-affinity
5. Multi-AZ (failure domains)
6. Rolling upgrade
7. Automated cluster recovery
Active-Active Redis (CRDTs-based)
Active-Active Redis (CRDTs-based)
Active-Active Redis (CRDTs-based)

Session management

Disaster recovery

Globally distributed applications

Local latency <1ms
Multi-cloud, on-premises, and hybrid deployments
Redis & Kubernetes & Your Microservices
Microservices running in Kubernetes

- Authentication
- Users
- Catalog
- Search
- Fraud detection

BloomFilter
Graph
JSON
Search
TimeSeries
AI
Microservices running in Kubernetes
Conclusion

- Redis your Cloud Native Database
  - Run anywhere: Bare Metal, Docker, Kubernetes, OpenShift, Pivotal, AWS, GCP, Azure...
  - Multi-model database for your micro services

- [http://redislabs.com](http://redislabs.com)
- [https://redislabs.com/tech-blog/](https://redislabs.com/tech-blog/)
- [https://redis.io](https://redis.io)
- [http://university.redislabs.com](http://university.redislabs.com)

- Kubernetes Blog posts:
  - [Redis Enterprise Operator for Kubernetes](https://redis.io/blog/redis-enterprise-operator-for-kubernetes)
  - [Redis Enterprise Release Using Helm Charts](https://redis.io/blog/redis-enterprise-release-using-helm-charts)
  - [Running Redis Enterprise Kubernetes Service](https://redis.io/blog/running-redis-enterprise-kubernetes-service)
Thank You

Tug Grall
Redis Labs
@tgrall