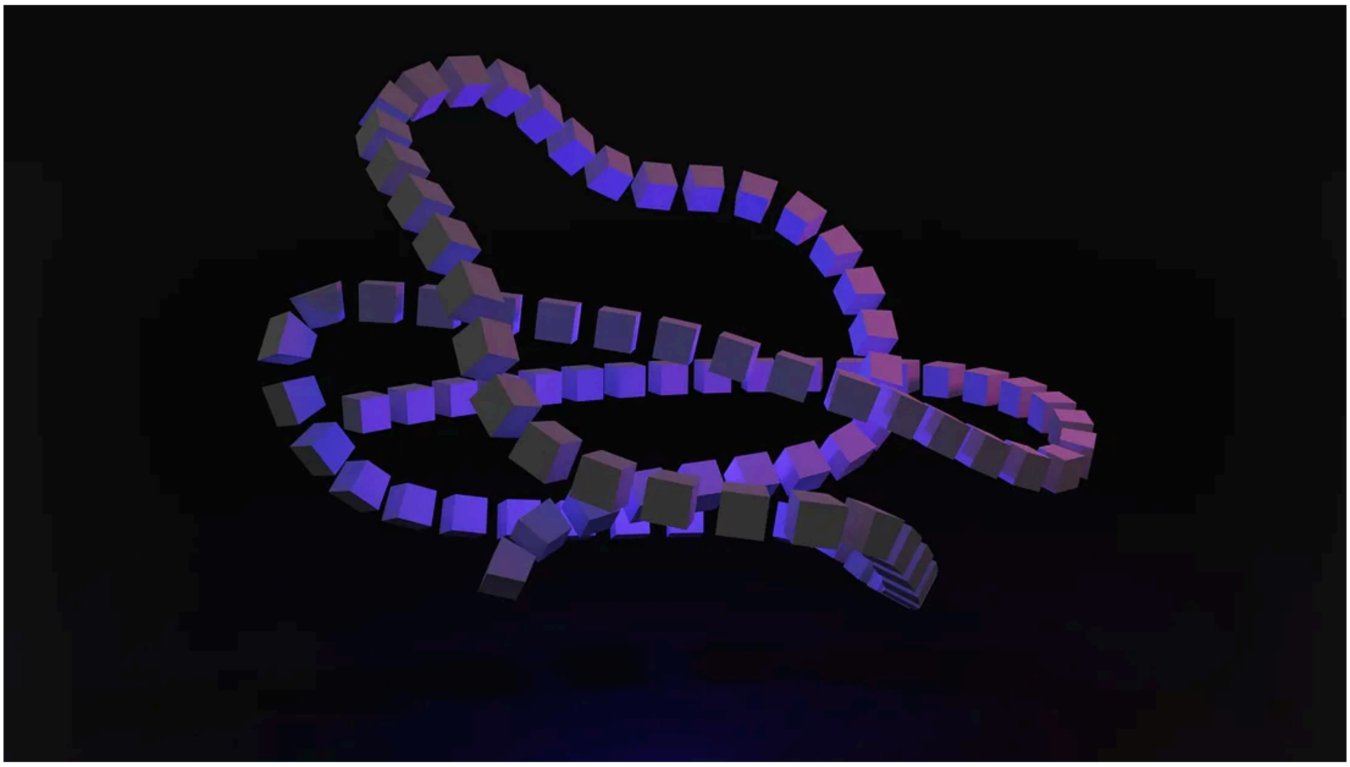
Open in app ↗

Search

Get unlimited access to the best of Medium for less than $1/week. Become a member



Deploying a Redis Stack Cluster using Docker images along with RedisInsight

Nakul Joshi · Follow
5 min read · Aug 23, 2023

Listen Share More

Redis is widely used today as a fast in-memory store and has extensive community support.

There can still be some issues one can encounter while dealing with a redis cluster setup, especially while using it inside a docker environment.

Here we will setup a redis cluster of 3 masters and 3 replicas for the *redis-stack-server* or the *redis-stack* docker image.

These images come pre-loaded with modules like RediSearch, ReJson, etc.

Things to keep in mind:

- `redis/redis-stack` contains both Redis Stack server and RedisInsight. This container is best for local development because you can use RedisInsight to visualize your data.
- `redis/redis-stack-server` provides Redis Stack but excludes RedisInsight. This container is best for production deployment.

Pre-Requisite:

- Basic knowledge of Redis
- A docker runtime environment
- Idea of docker-compose

For the current scenario, we will use the **redis-stack-server** and deploy a Redisinsight container alongside and make them work together.

We will create a base docker-compose.yml file and then make it more complex, step-by-step.

Base Docker-compose file:

```
services:
        redis:
                image: ‘redis/redis-stack-server:latest’
                ports:
                        - ‘127.0.0.1:6379:6379/tcp’
                volumes:
                        - ‘/Users/<dir>/Documents/redis/<some_dir>:/data:rw’
                environment:
                        - REDIS_ARGS = "--save 10 1 --appendonly yes"

        redisinsight:
                image: ‘redislabs/redisinsight:latest’
                ports:
                        - '127.0.0.1:8001:8001'
```

The above Docker compose file, will start a RedisInsight container on port 8001, and a single redis instance on default port *6379*, that has persistence enabled on disk, at a directory of your choice.

After running *docker-compose up* you can go to *127.0.0.1:8001* on your browser and add the newly created redis instance to start monitoring and running queries via the UI.

But we need a whole Redis cluster, not just one image!

To create a 6 node: 3Master+3Replica cluster, we would have to define 6 redis instances, with almost the same configuration, except different ports for them to run on.

In the current example, i am running them on my local MacOS machine, so 6 different ports on the same host would be assigned to each of the instances.

To get started, create 6 *redis.conf* files, where we will keep the port and other configurations for each of the instances.

Here's a sample *redis.conf* for one of the instances:

```
port 7000

cluster-enabled yes

cluster-config-file nodes.conf

cluster-node-timeout 5000

appendonly yes

protected-mode no
```

Similarly we can create 5 other conf files from port 7001 to 7005.

Tuning the redis service inside the docker-compose file, here's what 1 of the redis nodes config would look like (final docker-compose file at the end)

```
redis-node-1:
        image: ‘redis/redis-stack-server:latest’
        container_name: redis-node-1
        command: [ "redis-server", "/usr/local/etc/redis/redis.conf" ]
        ports:
                - ‘7000:7000/tcp’
        volumes:
                - ‘/Users/<dir>/Documents/redis/cluster/dump/7000:/data:rw’
                -/Users/<dir>/Documents/redis/cluster/7000/redis.conf:/usr/local/e
        environment:
                - REDIS_ARGS = "--save 10 1 --appendonly yes"
```

We are assigning this particular container a name *redis-node-1* for easier identification in our cli logs.

Added the config file, we had created in the previous step, that will be loaded onto the container while being created and referring to that in the start command.

Now, we can go ahead and add the rest 5 redis images into the docker-compose file.

But that would still create 5 independent redis instances, not working together as a cluster.

Creating a Cluster

A redis-cluster is created by running a command like:

```
redis-cli  —cluster create <ip>:7000 <ip>:7001 <ip>:7002 <ip>:7003 <ip>:7004 <ip>:7
```

IMPORTANT: The redis-cli command for cluster creation only takes in IPs of the redis instances and not domain names where they are located.

So if you mention something like *redis-cli –cluster create redis-node-1:7000* thinking that the hostname for redis-node-1 would be picked up, that would not be the case.

So what we need to do next is to configure a **network bridge** between the docker images so that the individual docker images can talk to each other while in a cluster mode.

This can be achieved by adding a network config and a subnet in the docker-compose file that can be used to assign an IP address, internal to docker, to each of these redis instances.

```
networks:
        redis_cluster_net:
                driver: bridge
                ipam:
                        driver: default
                        config:
                                - subnet: 173.18.0.0/16
```

Here *redis_cluster_net* is a customer bridge name we are assigning.

Now, we can attach this network bridge details along with the IPs to each of the redis nodes. And use those IPs in the redis-cluster create command.

Putting all the pieces together

The final docker-compose.yml should look something like this:

```
networks:
        redis_cluster_net:
                driver: bridge
                ipam:
                        driver: default
                        config:
                                - subnet: 173.18.0.0/16

services:
        redisinsight:
                image: ‘redislabs/redisinsight:latest’
                ports:
                        - ‘127.0.0.1:8001:8001'
                networks:
                        redis_cluster_net:
                                ipv4_address: 173.18.0.12

        redis-node-1:
                image: ‘redis/redis-stack-server:latest’
                container_name: redis-node-1
                command: [ "redis-server", "/usr/local/etc/redis/redis.conf" ]
```

```
          ports:
                  - ‘7000:7000/tcp’
          volumes:
                  - ‘/Users/<dir>/Documents/redis/cluster/dump/7000:/data:rw
                  - /Users/<dir>/Documents/redis/cluster/7000/redis.conf:/us
          environment:
                  - REDIS_ARGS = "--save 10 1 --appendonly yes"
          networks:
                  redis_cluster_net:
                          ipv4_address: 173.18.0.5

  redis-node-2:
          image: ‘redis/redis-stack-server:latest’
          container_name: redis-node-2
          command: [ "redis-server", "/usr/local/etc/redis/redis.conf" ]
          ports:
                  - ‘7001:7001/tcp’
          volumes:
                  - ‘/Users/<dir>/Documents/redis/cluster/dump/7001:/data:rw
                  - /Users/<dir>/Documents/redis/cluster/7001/redis.conf:/us
          environment:
                  - REDIS_ARGS = "--save 10 1 --appendonly yes"
          networks:
                  redis_cluster_net:
                          ipv4_address: 173.18.0.6

  redis-node-3:
          image: ‘redis/redis-stack-server:latest’
          container_name: redis-node-3
          command: [ "redis-server", "/usr/local/etc/redis/redis.conf" ]
          ports:
                  - ‘127.0.0.1:7002:7002/tcp’
          volumes:
                  - ‘/Users/<dir>/Documents/redis/cluster/dump/7002:/data:rw
                  - /Users/<dir>/Documents/redis/cluster/7002/redis.conf:/us
          environment:
          - REDIS_ARGS = "--save 10 1 --appendonly yes"
          networks:
                  redis_cluster_net:
                          ipv4_address: 173.18.0.7

  redis-node-4:
          image: ‘redis/redis-stack-server:latest’
          container_name: redis-node-4
          command: [ "redis-server", "/usr/local/etc/redis/redis.conf" ]
          ports:
                  - ‘127.0.0.1:7003:7003/tcp’
          volumes:
                  - ‘/Users/<dir>/Documents/redis/cluster/dump/7003:/data:rw
                  - /Users/<dir>/Documents/redis/cluster/7003/redis.conf:/us
          environment:
                  - REDIS_ARGS = "--save 10 1 --appendonly yes"
          networks:
                  redis_cluster_net:
                          ipv4_address: 173.18.0.8

  redis-node-5:
          image: ‘redis/redis-stack-server:latest’
          container_name: redis-node-5
          command: [ "redis-server", "/usr/local/etc/redis/redis.conf" ]
          ports:
                  - ‘127.0.0.1:7004:7004/tcp’
          volumes:
                  - ‘/Users/<dir>/Documents/redis/cluster/dump/7004:/data:rw
                  - /Users/<dir>/Documents/redis/cluster/7004/redis.conf:/us
          environment:
                  - REDIS_ARGS = "--save 10 1 --appendonly yes"
          networks:
                  redis_cluster_net:
                          ipv4_address: 173.18.0.9

  redis-node-6:
          image: ‘redis/redis-stack-server:latest’
          container_name: redis-node-6
          command: [ "redis-server", "/usr/local/etc/redis/redis.conf" ]
          ports:
                  - ‘127.0.0.1:7005:7005/tcp’
          volumes:
                  - ‘/Users/<dir>/Documents/redis/cluster/dump/7005:/data:rw
                  - /Users/<dir>/Documents/redis/cluster/7005/redis.conf:/us
          environment:
                  - REDIS_ARGS = "--save 10 1 --appendonly yes"
          networks:
                  redis_cluster_net:
                          ipv4_address: 173.18.0.10

  cluster_initiator:
          image: ‘redis/redis-stack-server:latest’
          ports:
          - ‘127.0.0.1:8000:8000'
          command: ‘redis-cli --cluster create 173.18.0.5:7000 173.18.0.6:70
          depends_on:
                  - redis-node-1
                  - redis-node-2
                  - redis-node-3
                  - redis-node-4
                  - redis-node-5
                  - redis-node-6
          networks:
                  redis_cluster_net:
                          ipv4_address: 173.18.0.11
```

And that’s it!

Your cluster should be good to go and you can validate it by going to the RedisInsight on the browser. Mentioning just of the master nodes should be enough, as it will automatically pull the details of the cluster and present it you.

Hopefully this provides you some help to get started with a cluster.

If you get stuck or need more details, do let me know, would be happy to help :)

Redis Docker Docker Compose Redis Cluster

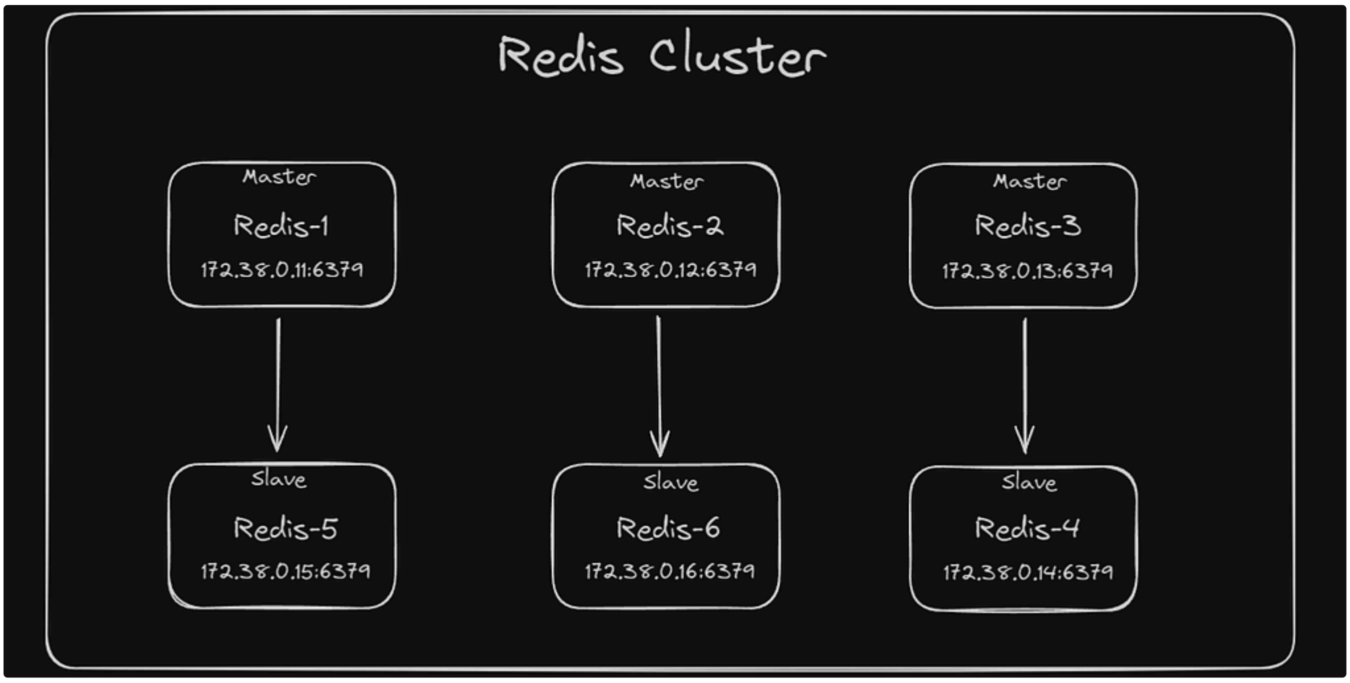
Follow

Written by Nakul Joshi

19 Followers

Data Engineer at Verizon

Recommended from Medium



Lim Yee Jie

Basic Docker Compose and Build a Redis Cluster with Docker Compose

Docker Compose

8 min read · Jan 23, 2024

14

Viacheslav Starikov

Correct way to run multiple Redis instances on Linux

Recently I was googling about how to start multiple Redis instances and was a bit confused of different approaches I found. It's not a...

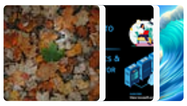
5 min read · Oct 3, 2023

105

Lists

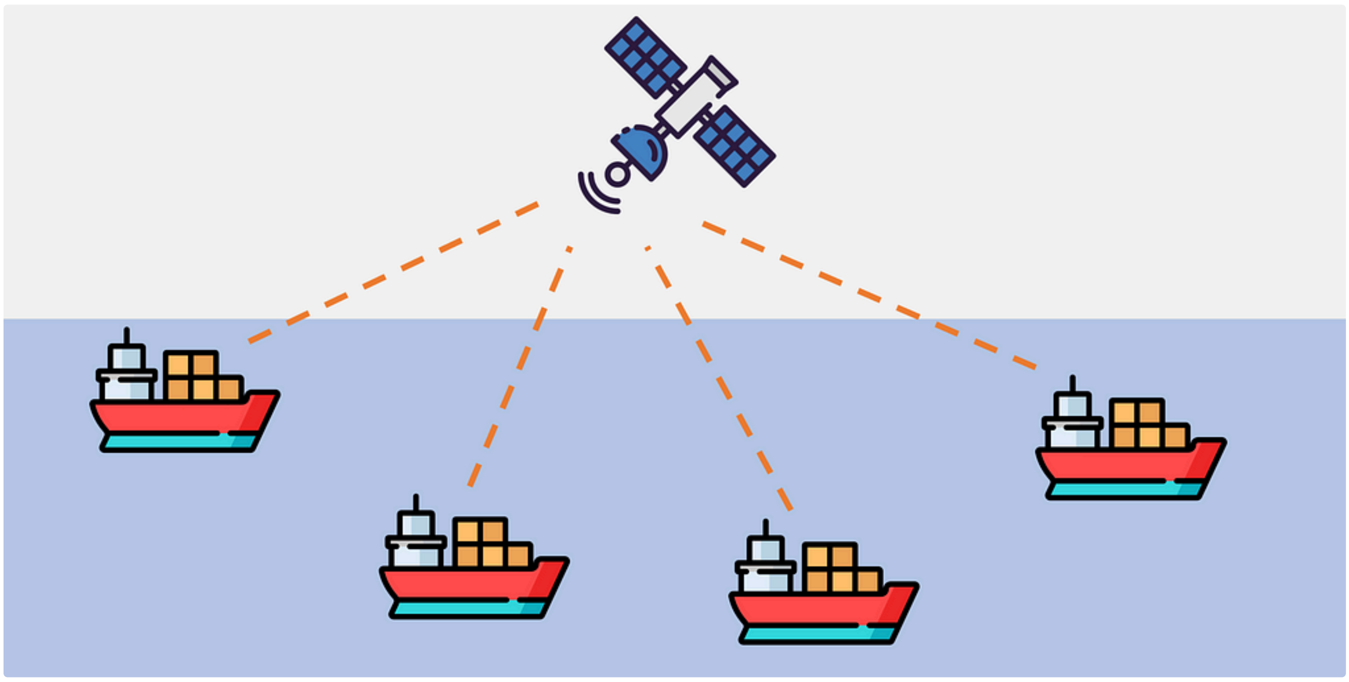
Coding & Development

11 stories · 494 saves



Natural Language Processing

1272 stories · 764 saves



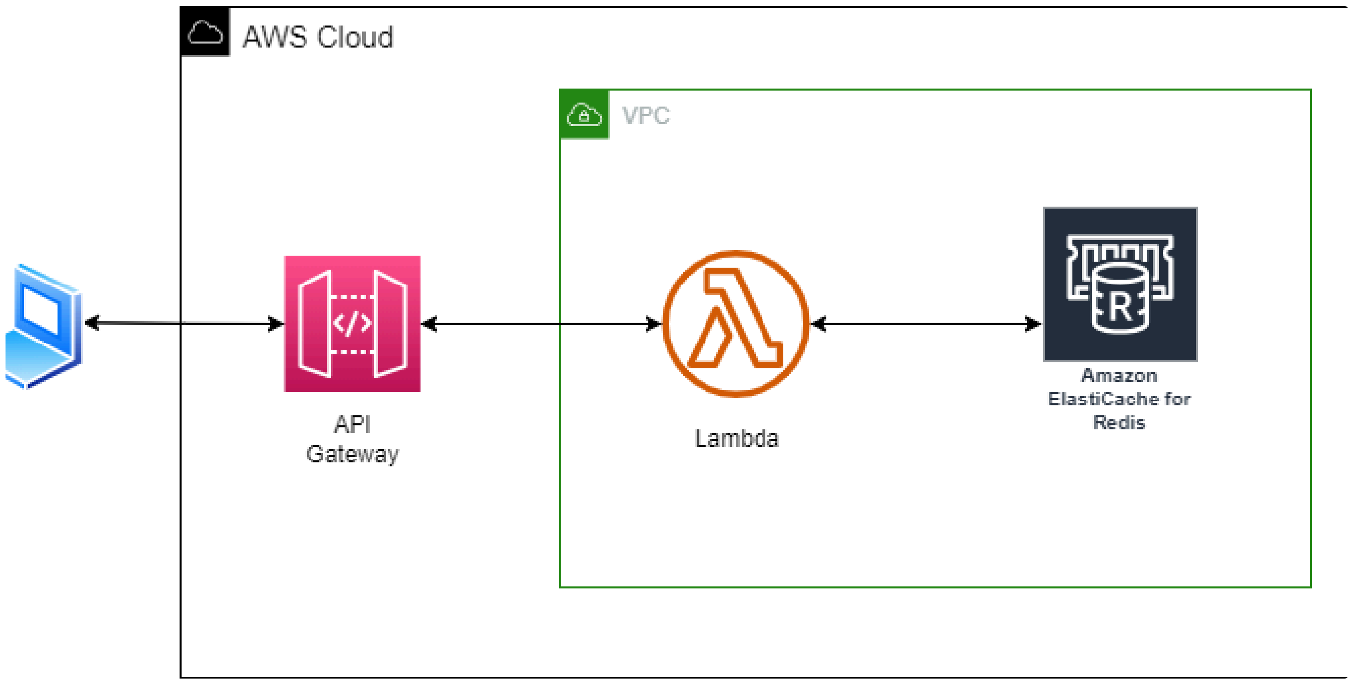
Ahmet Tuncer

Redis Cluster using Docker

We will talk about creating a redis cluster using docker, especially on Windows.

4 min read · Dec 11, 2023

4



Muhammad Shariq in AWS Tip

Integrating AWS Elasticache Redis with AWS Lambda using Serverless Framework and CloudFormation

Dive into the world of AWS cloud services with this in-depth tutorial. I'll walk you through the entire process of seamlessly integrating...

3 min read · Oct 22, 2023

28

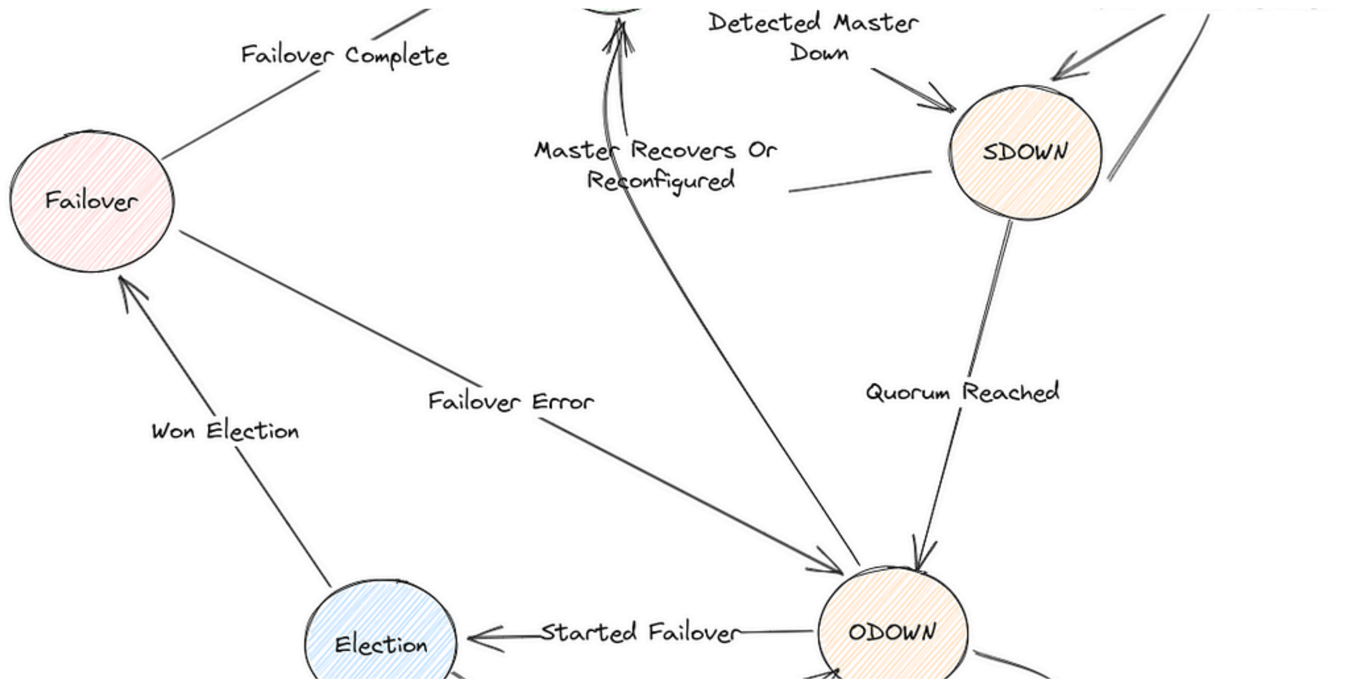
Yaren Boran

REDIS

Today, we will explore Redis' in-memory data storage system and usage examples. Let's get started!

5 min read · Jan 3, 2024

1



Andy Dunstall

Redis Sentinel

These are my notes on how Redis Sentinel works. This focuses on the design and implementation of Sentinel rather than configuration and...

10 min read · Oct 8, 2023

13

See more recommendations