Hello!

I am Esmaeil Kazemi

I'm interested in learning how are you?

You can find me at @eskazemi

NOSQL

vs

SQL

Redis

Redis stands for Remote Dictionary Server

eskazemi

Map

Redis

1- Features

2- application

3- data type

4- Message Queue

5- Transactions

6- Pipelining

7- Lua Scripts

8- Persistence

9- Benchmarks

10- configuration

11- ACLs

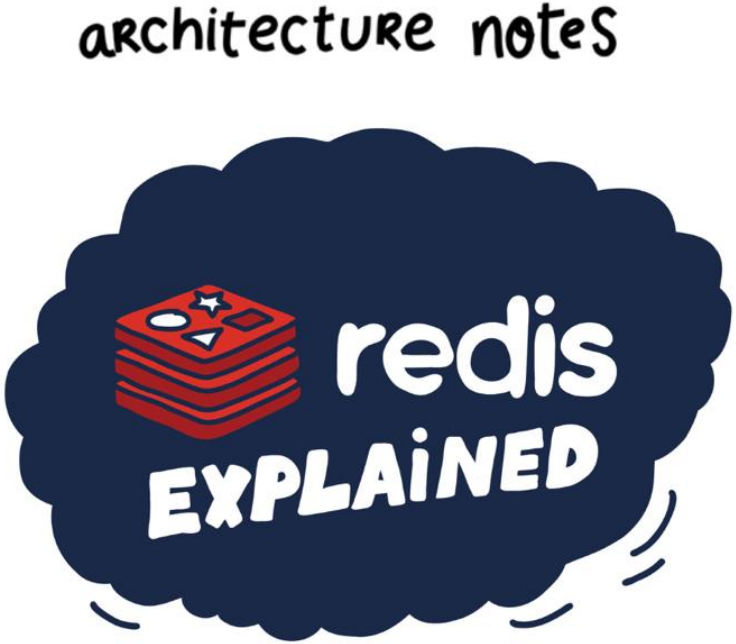
12- Redis Cluster

13- Redis vs Memcached

14- Redis vs Hazelcast

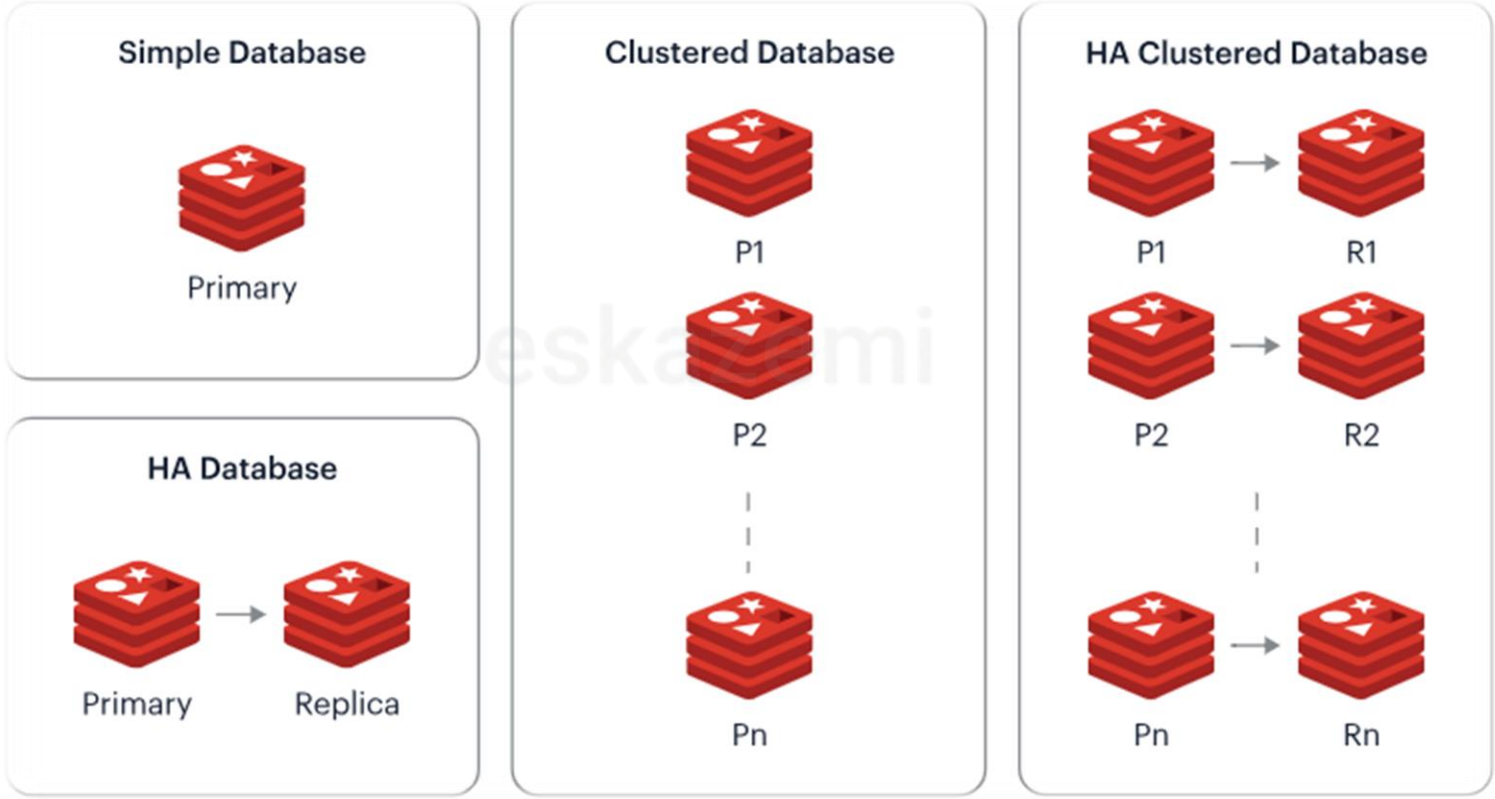
15- Redis vs RDBMS

eskazemi



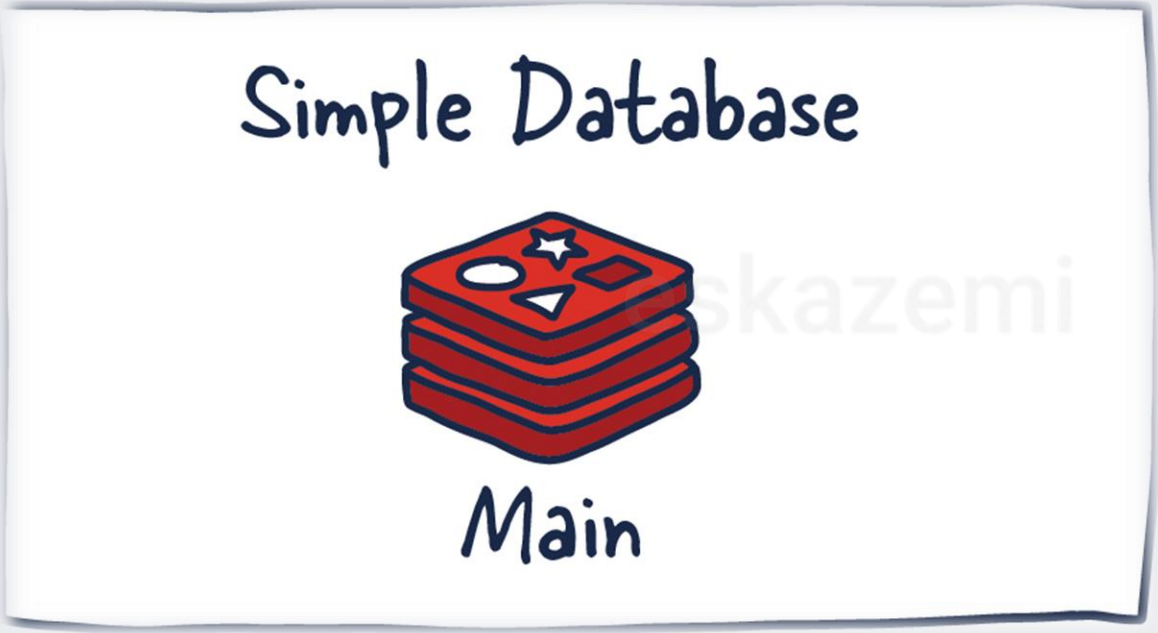
Redis Cluster: Architecture, Replication, Sharding and Failover

استقرار های مختلف Redis



بسته به مورد استفاده و مقیاس، می توانید تصمیم بگیرید از یکی از راه اندازی ها استفاده کنید.

Single Redis Instance

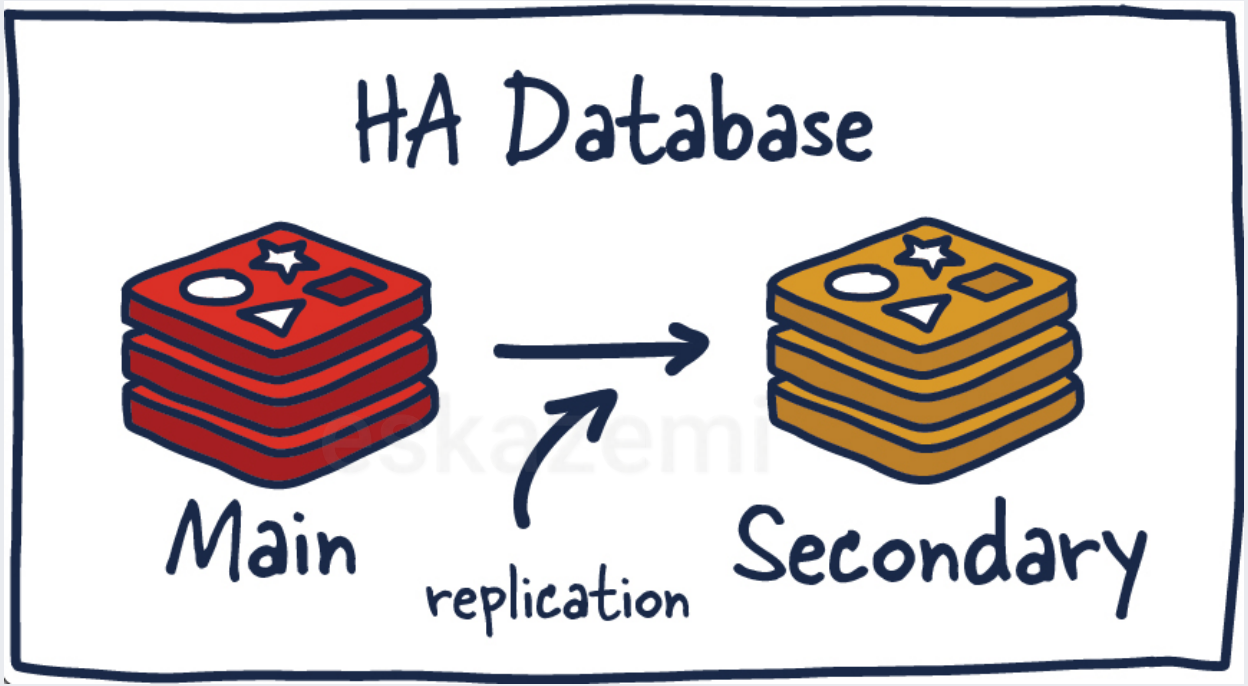


Singel Redis ساده ترین پیاده سازی Redis است. این سرویس به کاربران اجازه می دهد نمونه های کوچکی را راه اندازی و اجرا کنند که می تواند به رشد و سرعت بخشیدن به سرویس های آن ها کمک کند. با این حال، این استقرار بدون نقص نیست. به عنوان مثال، اگر این نمونه شکست بخورد یا در دسترس نباشد، تمام ارتباطات با Redis با شکست مواجه شده و در نتیجه عملکرد و سرعت کلی سیستم کاهش می یابد.

با توجه به حافظه کافی و منابع سرور، این نمونه می تواند قدرتمند باشد.

سناریویی که در درجه اول برای ذخیره سازی استفاده می شود، می تواند منجر به افزایش عملکرد قابل توجه با حداقل تنظیمات شود.

Redis HA



یکی دیگر از راه اندازی های محبوب با Redis ، استقرار اصلی با استقرار ثانویه است که با تکرار همگام می شود. همان طور که داده ها در نمونه اصلی نوشته می شوند ، کپی هایی از آن دستورات را برای نمونه های ثانویه به بافر خروجی replica client ارسال می کند که تکرار را تسهیل می کند. نمونه های ثانویه می توانند یک یا چند نمونه در استقرار شما باشند . این نمونه ها می توانند به scale read از Redis کمک کنند یا در صورت از بین رفتن نمونه اصلیfailover را فراهم می کند.

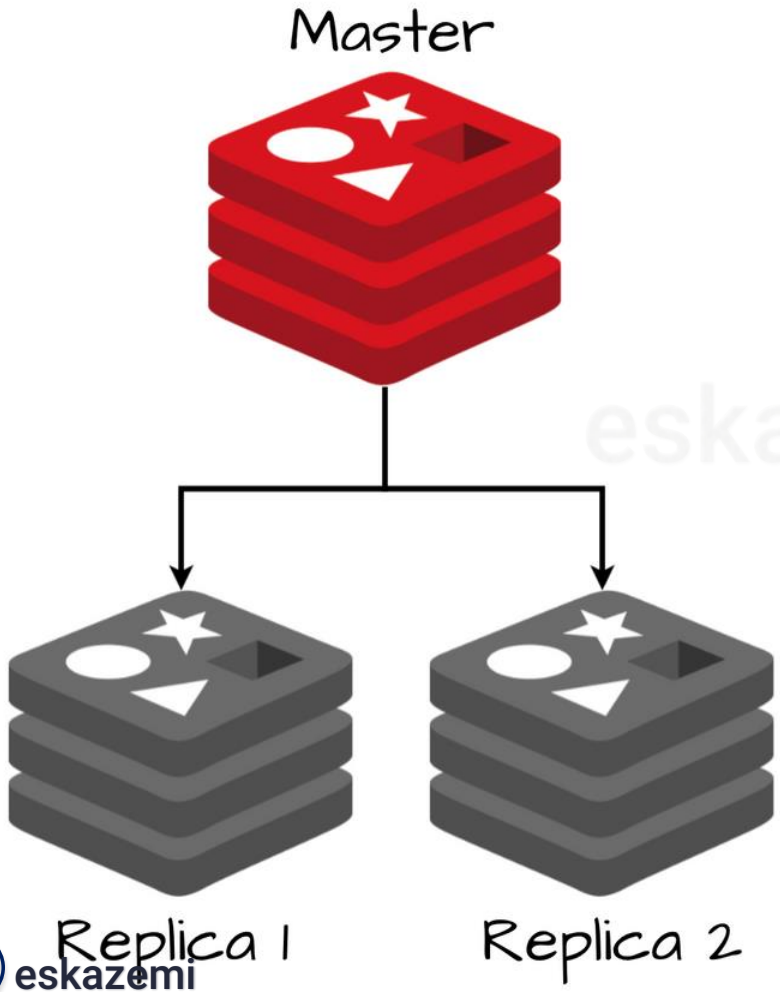
چندین چیز جدید برای در نظر گرفتن در این توپولوژی وجود
دارد زیرا ما اکنون وارد سیستم توزیع شده ای شده ایم که
اشتباهات زیادی دارد که باید در نظر بگیرید . چیز هایی که قبلا
ساده بودند اکنون پیچیده تر شده اند.

باتوجه به حافظه و منابع سرور کافی، این نمونه می تواند قدرتمند باشد. سناریویی primarily برای ذخیره
سازی استفاده می شود، می تواند منجر به افزایش قابل توجه عملکرد با حداقل تنظیمات شود.


REDIS
REPLICATION
eskazemi

Replication

- فرایندی است که به نمونه های Redis اجازه می دهد تا کپی دقیقی از نمونه اصلی باشند. به صورت پیش فرض یک فرایند ناهمزمان است .
- غیر مسدود کننده است این بدان معناست که وقتی یک یا چند کپی همگام سازی اولیه یا یک همگام سازی جزئی را انجام می دهند، نمونه اصلی (master) همچنان به رسیدگی به پرس و جوها ادامه خواهد داد. همچنین در سمت تکرار (replica) در حین انجام replication می تواند با استفاده از نسخه قدیمی مجموعه داده ، پرس و جو ها را مدیریت کند.
- خواندن replica توان عملیاتی خواندن را بهبود می بخشند و در برابر از دست رفتن داده ها در موارد شکست گره محافظت می کنند. Replication می تواند هم برای مقیاس پذیری، هم به منظور داشتن کپی های متعدد برای پرس و جوهای فقط خواندنی، یا به سادگی برای بهبود ایمنی داده ها و دسترسی بالا استفاده شود.
- جدا از اینکه به صورت پیش فرض فقط خواندنی است، یکی از تفاوت های مهم بین master و replica این است که replica کلیدهای منقضی شده / کپی را منقضی نمی کند، آن ها منتظر منقضی شدن کلیدها توسط master می مانند و زمانی که master منقضی میکند یا کلیدی را بیرون می کشد، کپی یک دستور DEL را تولید می کند که به تمام کپی ها منتقل می شود.



- A master can have multiple replicas.
- By default replication is an async process on both master and replicas.
- Master and replica synchronize each other with replicationID and offset.
- Replica instances can be promoted to master after a failover

Replication

Replication از سه مکانیسم استفاده می کند:

1. هنگامی که نمونه های اصلی (Master) و کپی (replica) به خوبی به هم متصل هستند، Master به روز نگه می دارد replica را با ارسال جریانی از دستورات به replica (کپی) ، تا اثرات روی مجموعه داده ای که در سمت master اتفاق می افتد را در replica داشته باشیم. مثلا کلید ها منقضی می شوند یا هر عمل دیگری که مجموعه داده (master) را تغییر می دهد باید این تغییرات در replica (کپی) اعمال شود.
2. هنگامی که ارتباط بین master و replica (کپی) قطع می شود، به دلیل مشکلات شبکه یا به دلیل اینکه یک وقفه زمانی در master یا replica (کپی) حس می شود، replica (کپی) که دوباره به Master متصل می شود و تلاش می کند تا با یک همگام سازی جزئی ادامه دهد: به این معنی است که سعی می کند فقط بخشی از جریان دستورهای از دست رفته در طول قطع ارتباط را به دست آورد.
3. هنگامی که همگام سازی جزئی امکان پذیر نباشد، replica خواستار همگام سازی کامل خواهد شد. این شامل یک فرآیند پیچیده تر خواهد بود که در آن master باید یک snapshot از تمام داده های خود ایجاد کند، آن را به replica (کپی) بفرستد، و سپس با تغییر مجموعه داده ها به ارسال جریان دستورها ادامه دهد.
4.

Configuration Replication

برای تنظیمات Replication فقط کافی است خط زیر را به فایل کانفیگ اضافه کنید

```
replicaof <redis master IP>
```

How replication happens

Master و Replica یکدیگر را با replicationID و offset همگام سازی می کنند می توان از دستور info replication برای بررسی offset های master و replica و replicationID

```
➢  info replication
# Replication
role:master
connected_slaves:1
slave0:ip=10.31.2.117,port=6379,state=online,offset=5491126744,lag=1
master_replid:ba06e48871ad41ba08bd33fd77a9b4cdf4c5c705
master_replid2:55b76e9a9968c9e802ae5aaab4071c556dcea9e0
master_repl_offset:5491126797
```

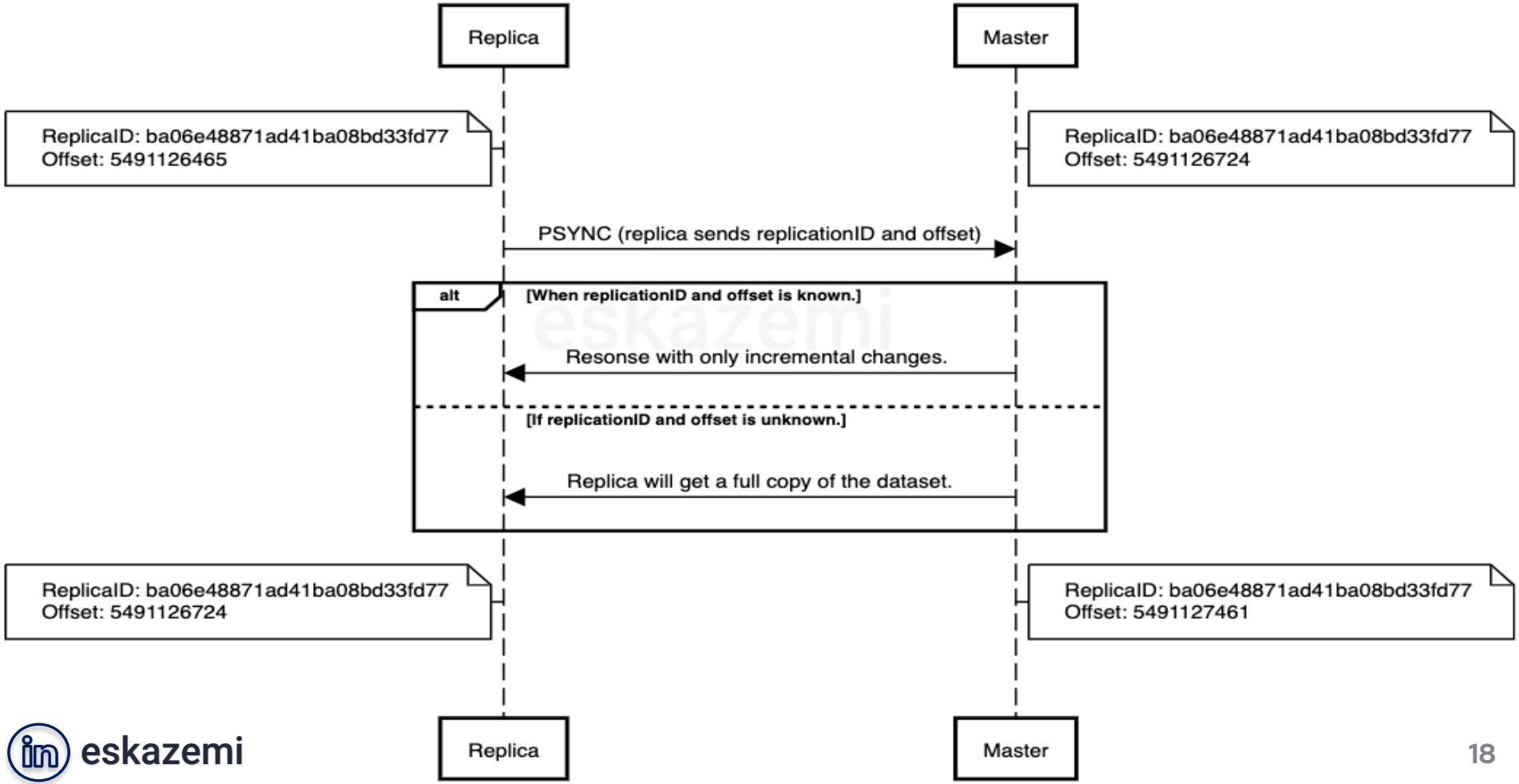
Replication Info on Master

```
➢  info replication
# Replication
role:slave
master_host:10.0.22.226
master_port:6379
master_link_status:up
master_last_io_seconds_ago:0
master_sync_in_progress:0
slave_repl_offset:5491126465
repl_sync_enabled:1
slave_read_reploff:5491126465
slave_priority:100
slave_read_only:1
connected_slaves:0
master_replid:ba06e48871ad41ba08bd33fd77a9b4cdf4c5c705
master_replid2:0000000000000000000000000000000000000000
master_repl_offset:5491126465
```

Replication info on Replica

هنگامی که کپی (replica) ها به master متصل می شوند، آن ها از دستور PSYNC استفاده می کنند تا master replicationID سابق خود و offset هایی که تاکنون پردازش کرده اند را ارسال کنند. به این ترتیب master می تواند فقط بخش افزایشی مورد نیاز را ارسال کند. با این حال، اگر بک لاگ کافی در بافرهای master وجود نداشته باشد، یا اگر کپی به یک تاریخچه (replicationID)اشاره کند که دیگر شناخته شده نیست، آنگاه یک همگام سازی مجدد کامل اتفاق می افتد: در این حالت، replica یک کپی کامل از مجموعه داده را از ابتدا دریافت خواهد کرد.

How redis replication happens

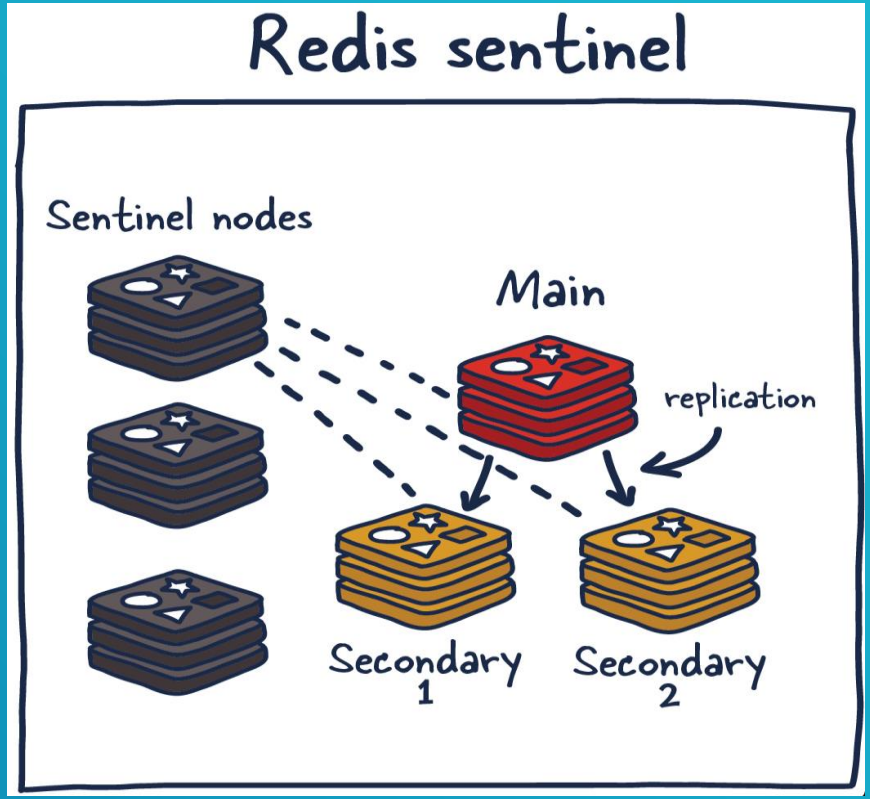


همگام سازی مجدد کامل چگونه اتفاق می افتد؟

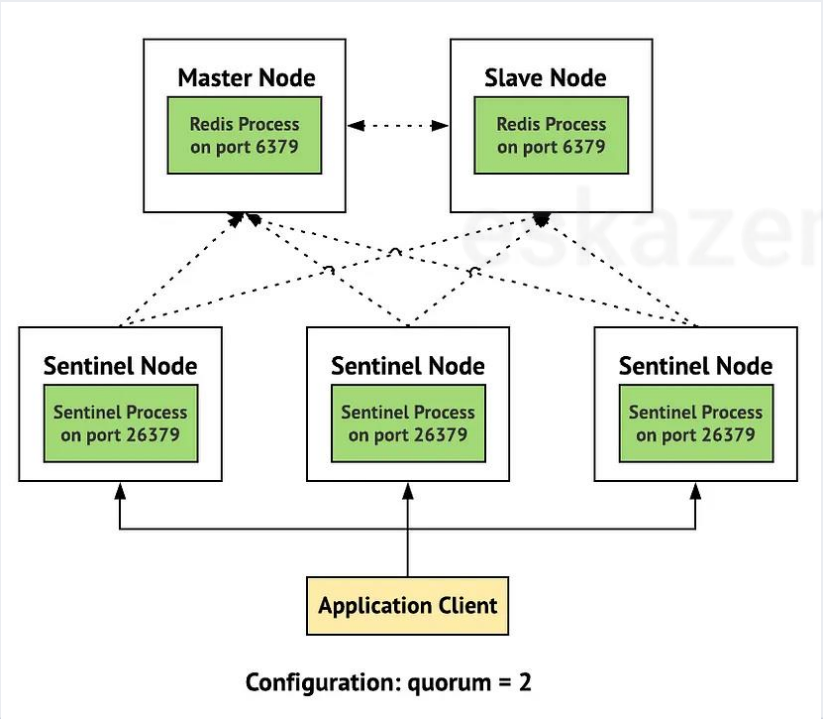
- ✓ Master یک فرایند ذخیره سازی که در پس زمینه برای تولید یه فایل RDB شروع می کند. همزمان شروع به بافر کردن تمام دستورات نوشتن جدید دریافت شده از سمت client می کند(Buffer یا بافر در حافظه RAM ناحیه ای است که برای ذخیره سازی اطلاعات موقت مورد استفاده قرار میگیرد.) هنگامی که فرایند ذخیره در پس زمینه تمام می شود Master فایل پایگاه داده (RDB File) را به Replica منتقل می کند ، که آن را روی دیسک ذخیره می کند و سپس آن را در Ram بارگذاری می کند . سپس Master تمام دستورات بافر شده را به replica ارسال می کند این در قالب یک جریان از دستورات انجام می شود و در همان فرمت خود پروتکل Redis است.
- ✓ از آنجایی که همگام سازی مجدد کامل یک فایل RDB روی دیسک ایجاد می کند این ممکن است یک گلوگاه عملکردی هم در Master و هم در صورت کندی I/O ایجاد کند. برای رفع این مشکل ، Redis از diskless replication در صورت همگام سازی مجدد کامل از نسخه 2.8.18 پشتیبانی می کند در این راه اندازی ، process child مستقیما RDB را از طریق سیم به replica (کپی) ها می فرستد، بدون اینکه از دیسک به عنوان ذخیره سازی میانی استفاده کند.

Redis Sentinel

Sentinel یک سیستم توزیع شده است. همانند تمام سیستم های توزیع شده، Sentinel دارای مزایا و معایب متعددی است.



How Redis offers High Availability and Automatic Failover ?



Redis sentinel راه حلی با دسترسی بالا است که توسط Redis پیشنهاد شده است که در صورت خرابی در Redis cluster ، Sentinel به طور خودکار نقطه شکست را تشخیص می دهد و خوشه را بدون دخالت انسانی به حالت پایدار باز می گرداند.

Redis Sentinel

چه اتفاقی در Redis Sentinel می افتد؟

Sentinelهمیشه نمونه‌های MASTER و SLAVE را در خوشه Redis بررسی می‌کند و بررسی می‌کند که آیا مطابق انتظار کار می‌کنند یا خیر. اگر Sentinel شکستی را در گره MASTER در یک خوشه مشخص تشخیص دهد، Sentinel یک فرآیند failover را شروع می کند. در نتیجه، Sentinel یک نمونه SLAVE را انتخاب می کند و آن را به MASTER ارتقا می دهد. در نهایت، سایر نمونه های SLAVE باقیمانده به طور خودکار برای استفاده از نمونه جدید MASTER پیکربندی مجدد خواهند شد.

اپلیکیشن ها متصل می شوند به Sentinels و Sentinels متصل می کنند به Master آماده خدمت.

Redis Sentinel

علاوه بر این، Sentinel یک سیستم توزیع شده قوی است، که در آن چندین Sentinel باید در مورد این واقعیت که یک Master معین دیگر در دسترس نیست، توافق کنند. سپس تنها فرآیند failover شروع به انتخاب یک گره MASTERجدید می کند. این توافق sentinel ها براساس مقدار quorum (حد نصاب) انجام می شود.

What is quorum ?

مقدار quorum (حد نصاب) تعداد نگهبانانی است که باید در مورد این واقعیت که master قابل دسترسی نیست به توافق برسند با این حال حد نصاب فقط برای تشخیص شکست استفاده می شود برای اینکه واقعا یک Failover انجام شود ، یکی از Sentinel ها باید به عنوان رهبر برای failover انتخاب شود و مجاز به ادامه کار باشد این تنها با رای اکثریت sentinel processes اتفاق می افتد.

Redis Sentinel

Number of Servers	Quorum	Number Of Tolerated Failures
1	1	0
2	2	0
3	2	1
4	3	1
5	3	2
6	4	2
7	4	3

جدول تعداد سرورها و حد نصاب با تعداد خرابی های تحمل شده.

این موضوع از سیستمی به سیستم دیگر متفاوت خواهد بود اما ایده کلی این است.

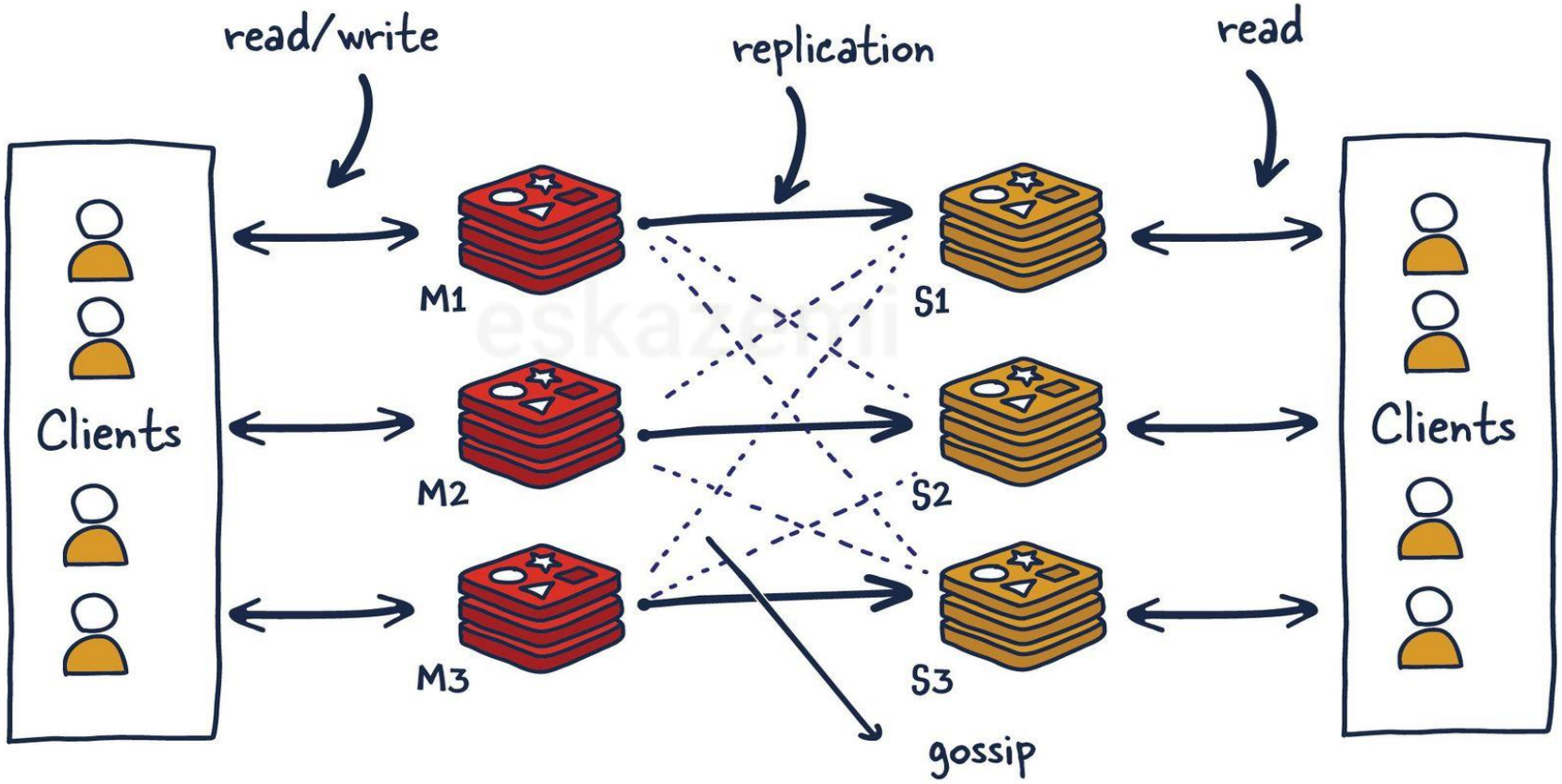
بنابراین Sentinel ها به طور مداوم در حال نظارت بر در دسترس بودن و ارسال این اطلاعات به clinet ها هستند تا اگر نقطه شکست بوجود آمد بتوانند به آن واکنش نشان دهند.

Redis Sentinel

این مسئولیت ها عبارتند از:

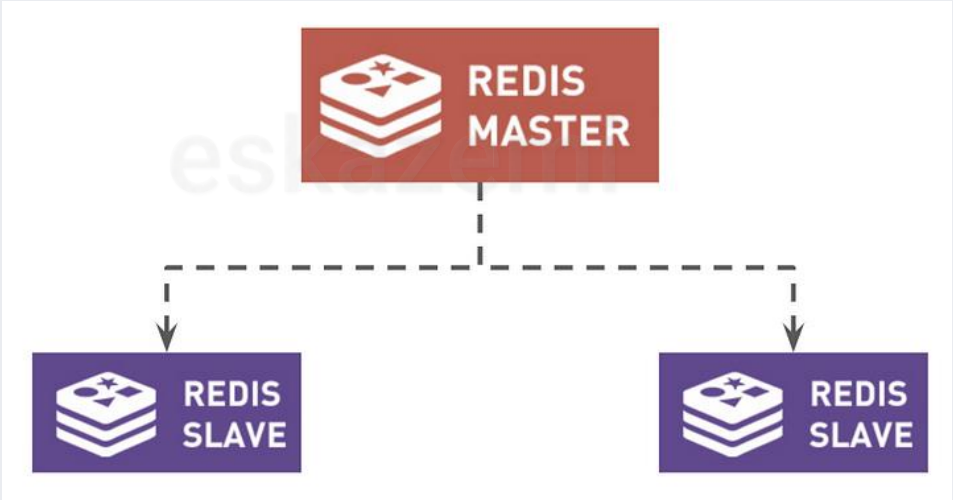
- ✓ نظارت - اطمینان از اینکه موارد اصلی (main) و ثانویه (secondary) مطابق انتظار کار می کنند.
- ✓ اطلاع رسانی - اطلاع رسانی به ادمین های سیستم در مورد اتفاقات رخ داده در نمونه های Redis.
- ✓ مدیریت شکست - گره های Sentinel می توانند یک فرآیند شکست را آغاز کنند اگر main instance در دسترس نباشد. استفاده از Redis Sentinel به این روش امکان تشخیص خرابی را فراهم می کند .این امر باعث افزایش استحکام و حفاظت در برابر سو رفتار یک ماشین و عدم دسترسی به گره اصلی Redis می شود.
- ✓ مدیریت پیکربندی - گره های Sentinel نیز به عنوان نقطه ای برای کشف نمونه اصلی (main) فعلی Redis عمل می کنند. در واقع هنگامی که یک client جدید تلاش می کند چیزی برای Redis بنویسد، Sentinelبه مشتری می گوید که نمونه اصلی(main) فعلی چیست.

Redis Cluster



Redis Cluster

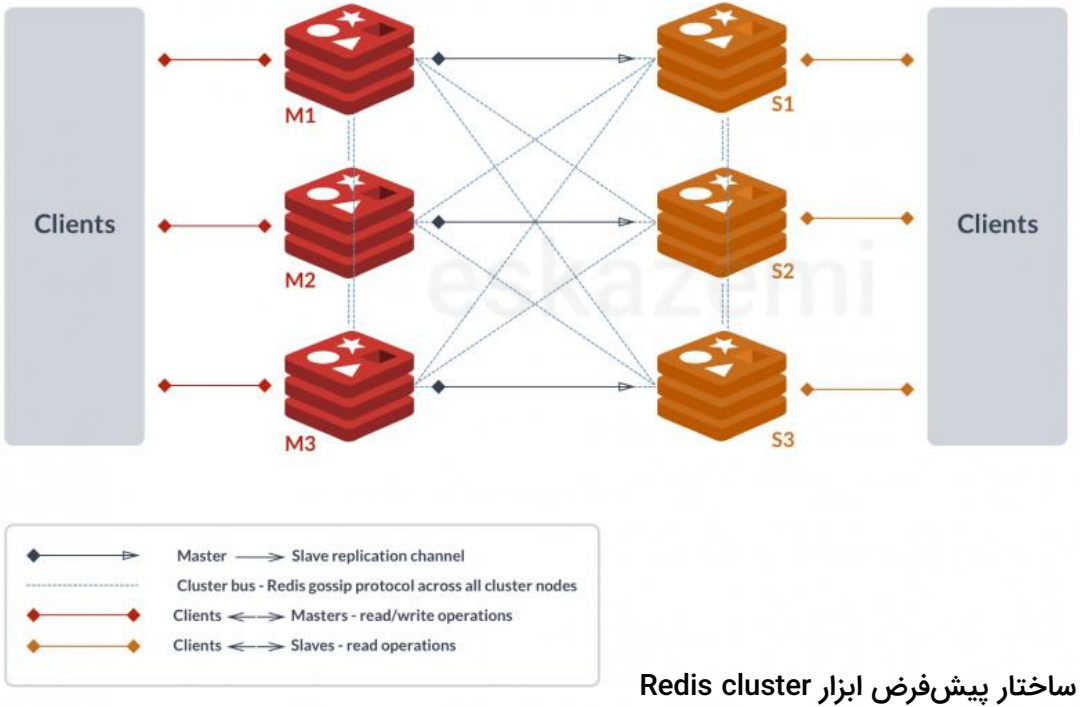
در مرحله اولیه، ما ردیس را در حالت Master - Slave قرار دادیم



نسخه‌های کپی Redis

Redis می‌تواند داده‌ها در چند پایگاه داده ثانویه تکرار کند.(با استفاده از Redis Replication) این نسخه‌های ثانویه، دارای کپی دقیقی از پایگاه داده اصلی خواهند بود. چنین چیزی واقعاً در بهینه‌سازی عملکرد مفید خواهد بود.

ابزار Redis cluster چیست؟



ساختار پیش‌فرض ابزار Redis cluster

ابزار Redis cluster در یک تعریف ساده، یک استراتژی شاردینگ یا طبقه‌بندی داده‌هاست. این ابزار به صورت اتوماتیک، داده‌ها را در سیستم‌های چندگانه Redis تقسیم‌بندی می‌کند. در واقع Redis cluster یک نسخه پیشرفته از Redis محسوب می‌شود که به ذخیره‌های گسترده دسترسی پیدا می‌کند و موجب جلوگیری از توقف عملکرد به علت مشکل در یک نقطه خاص شبکه می‌شود.

Redis Cluster

مفهموم

Redis Cluster یک پیاده سازی خوشه ای فعال - غیرفعال است که از گره های Master و Slave تشکیل شده است. این خوشه از پارتیشن بندی Hash برای تقسیم فضای کلید به ۱۶۳۸۴ اسلات کلید استفاده می کند که هر Master مسئول زیرمجموعه ای از آن اسلات ها است.

هر slave یک Master خاص را تکرار می کند و می تواند مجدادا به یک Master دیگر اختصاص یابد یا در صورت نیاز به عنوان یک گره اصلی انتخاب شود.

Ports Communication

هر گره در یک خوشه به دو پورت TCP نیاز دارد.

یک پورت برای اتصالات و ارتباطات client استفاده می شود. این پورتی است که شما در برنامه های کلاینت یا ابزارهای خط فرمان پیکربندی می کنید.

درگاه مورد نیاز دوم برای ارتباط گره به گره که در یک پروتکل باینری رخ می دهد و به گره ها اجازه بحث در مورد پیکربندی و در دسترس بودن گره را می دهد.

Failover

هنگامی که یک Master شکست می خورد یا مشخص می شود که توسط زمانی که یک یا چند replica شکست را تشخیص دهند، می توانند انتخابات را آغاز کنند.، همان طور که ارتباط گره ها از طریق پورت gossip تعیین می شود ، Master های باقی مانده با رای گیری یکی از slave های Master شکست خورده را به جای آن انتخاب می کنند.

↓

پیوستن دوباره به cluster

↓

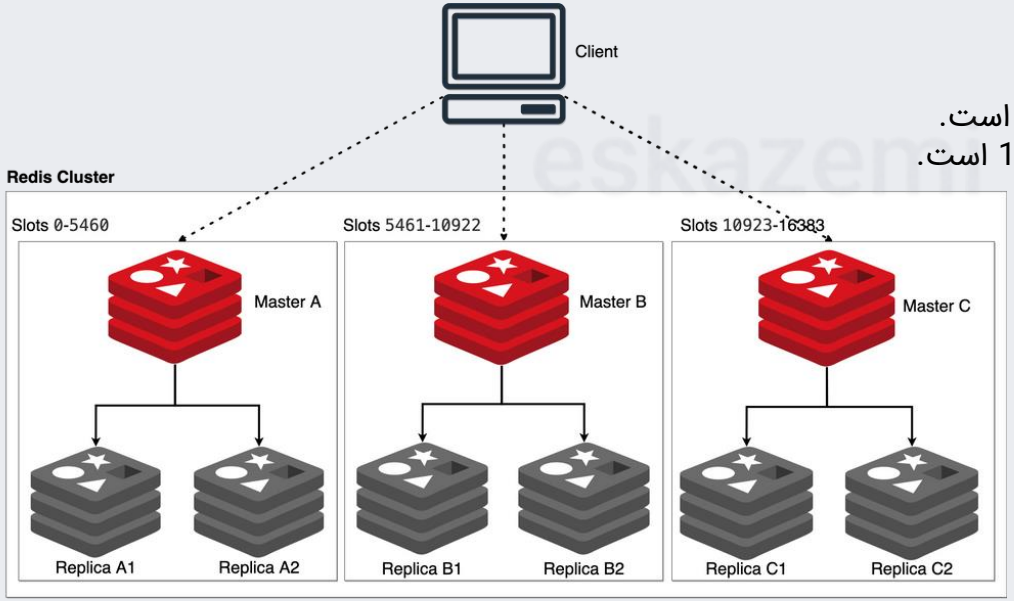
هنگامی که یک Master شکست خورده در نهایت دوباره به خوشه ملحق می شود ، به عنوان یک slave می پیوندد و شروع به کپی دیتا از Master می کند.

Sharding

Redis داده ها را به صورت خودکار در سرورها به اشتراک می گذارند.
ردیس مفهومی از hash slot به منظور تقسیم داده ها دارد. تمام داده ها به اسلات ها تقسیم می شوند.
۱۶۳۸۴ سهمیه وجود دارد. این اسلات ها براساس تعداد سرورها تقسیم می شوند.

اگر ۳ سرور وجود داشته باشد؛ A، Bو Cآنگاه:

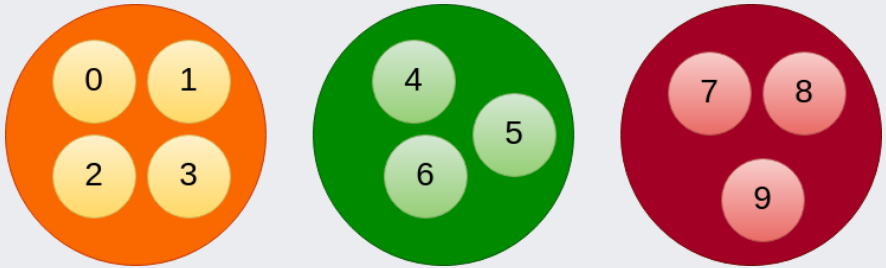
سرور ۱ شامل اسلات های هش از 0 تا 5460 است.
سرور ۲ شامل اسلات های هش از 5461 تا 10922 است.
سرور ۳ شامل اسلات های هش از 10923 تا 16283 است.



✓ هر کلیدی که در Redis cluster ذخیره می‌کنید با یک hash slot همراه خواهد بود.
✓ هر کدام از نقطه‌های اصلی یا Master کنترل زیرمجموعه‌ای شامل ۱۶۳۸۴ عدد hash slot را برعهده دارد.

الگوریتم توزیعی که Redis Cluster برای انتقال کلیدها به hash slot ها استفاده می‌کند، به صورت زیر است:
به عنوان مثال، فرض کنید که فضای کلید به ۱۰ اسلات (صفر تا ۹) تقسیم شده باشد. هر کدام از نقاط یک زیرمجموعه از hash slot ها را در بر خواهد داشت.

HASH_SLOT = CRC16(key) mod HASH_SLOTS_NUMBER



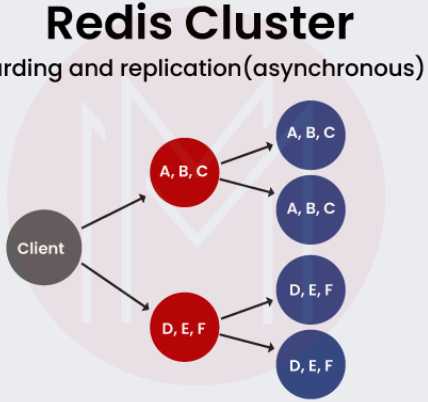
کلید " "name در اسلات به صورت روبرور خواهد بود.

slot = CRC16("name") % 16384

تفاوت بین sharding و replication :

Sharding: تقسیم بندی (پارتیشن بندی) نیز نامیده می شود که به تقسیم داده ها توسط کلید اشاره دارد. ما از اشتراک گذاری برای افزایش عملکرد و کاهش hit به دیتابیس و همچنین بار حافظه بر روی یک منبع واحد استفاده می کنیم.

Replication: این روش که mirroring نیز نامیده می شود، به کپی کردن تمام داده ها اشاره دارد. این کار کمک می کند تا دسترسی بالایی به خواندن داشته باشید. اگر در حال خواندن چندین کپی باشید، نزخ hit به تمام منابع کاهش خواهد یافت. اما نیاز به حافظه برای همه منابع یکسان است.



Redis Cluster

به طور خلاصه ابزار Redis cluster دارای ویژگی‌های زیر است:

- ✓ مقیاس‌پذیر از نظر افقی: می‌توانیم بنابر نیاز به ظرفیت بیشتر در شبکه، نقطه اضافه کنیم.
- ✓ شاردینگ داده اتوماتیک: می‌تواند به صورت اتوماتیک داده‌ها را بین نقاط مختلف پارتیشن‌بندی و جداسازی کند.
- ✓ تاب‌آوری: حتی اگر یک نقطه از شبکه را از دست بدهیم، همچنان می‌توانید بدون اینکه به داده‌ها آسیبی وارد شود، به عملکرد خود ادامه بدهیم.
- ✓ مدیریت غیرمتمرکز: هیچ‌کدام از نقاط شبکه، عنصر وابسته به کل کلاستر نیست. هر کدام از نقاط در تنظیمات کلاستر مشارکت می‌کند با استفاده از پروتکل (gossip)

25. What are the advantages of using Redis Cluster?
Answer: Redis Cluster offers high availability, scalability, and fault tolerance.

Redis Cluster

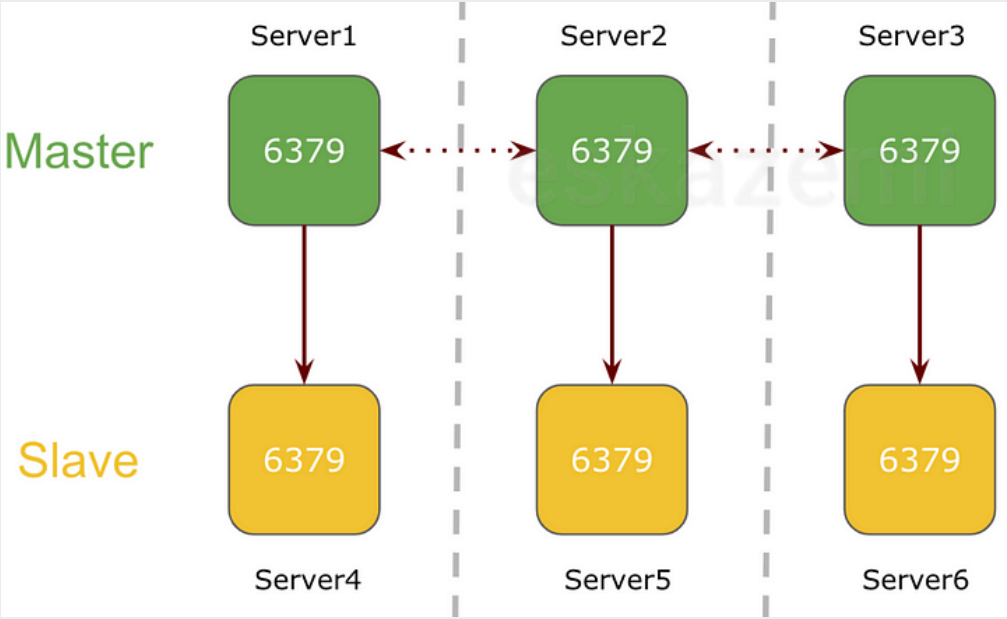
توپولوژی کلاستر

در ابتدا به تعداد و ترکیب Worker ها می پردازیم. چیزی که Redis پیشنهاد کرده اینطوره :

۳ سرور مجزا

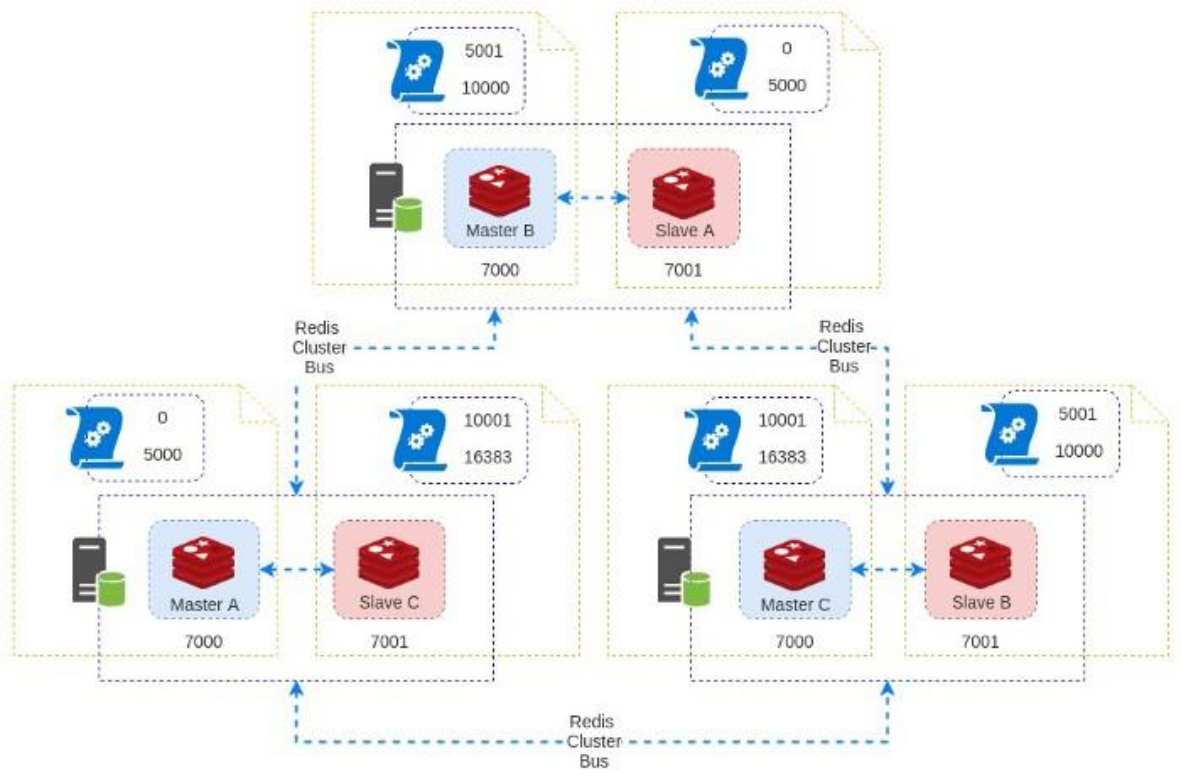
- کانفیگ ۳ Master Node روی هر دستگاه
- کانفیگ ۳ Slave Node به ازای هر Master

6 Node M/S Cluster



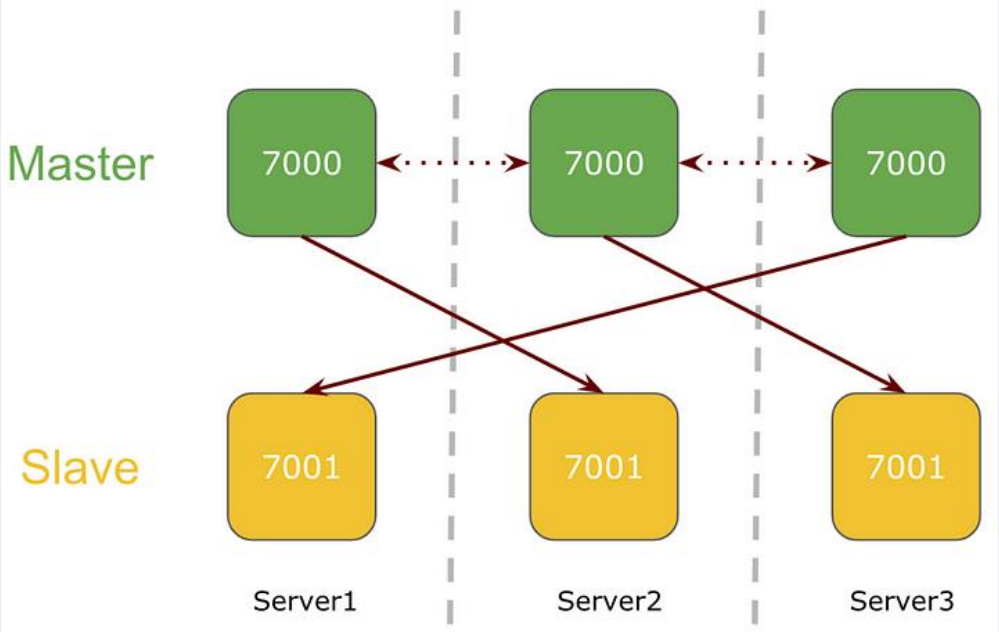
در اینجا ، سرویس Redis بر بروی port 6379 در تمام سرور های این کلاستر اجرا می شود هر سرور اصلی در حال تکثیر کلید ها به گره slave مربوط به خود است که در طول فرآیند ایجاد خوشه اختصاص داده شده است.

نمایی دیگر



۳ سرور فیزیکی با نود های Master و Slave مجزا و بازه ذخیره سازی اسلات برای هر کدام

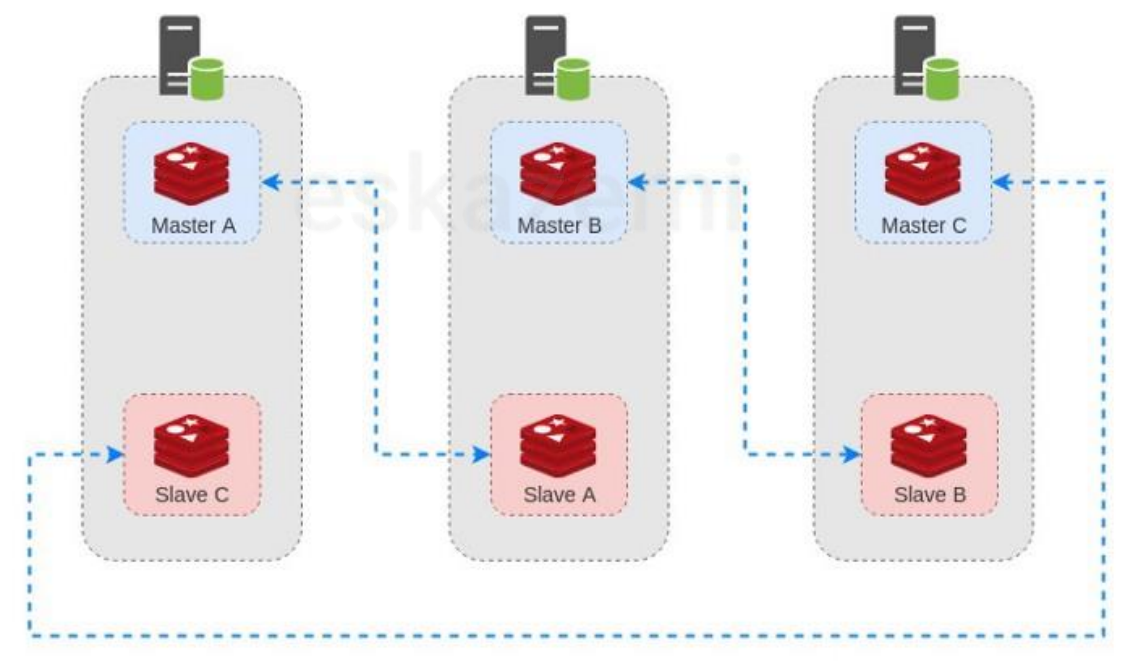
3 Node M/S Cluster



- در حالت خوشه ای ۳ گرهی، ۲ سرویس روی هر سرور در پورت های مختلف اجرا خواهد شد. هر ۳ گره به عنوان یک Master عمل خواهند کرد و گره های متقاطع به عنوان slave عمل خواهند کرد.
- در اینجا، دو سرویس Redis بر روی هر سرور بر روی دو پورت مختلف اجرا خواهد شد و هر سرور در حال تکرار کلیدهای مربوط به Redis Slave خود است که بر روی گره های دیگر اجرا می شود.

Fail-Over

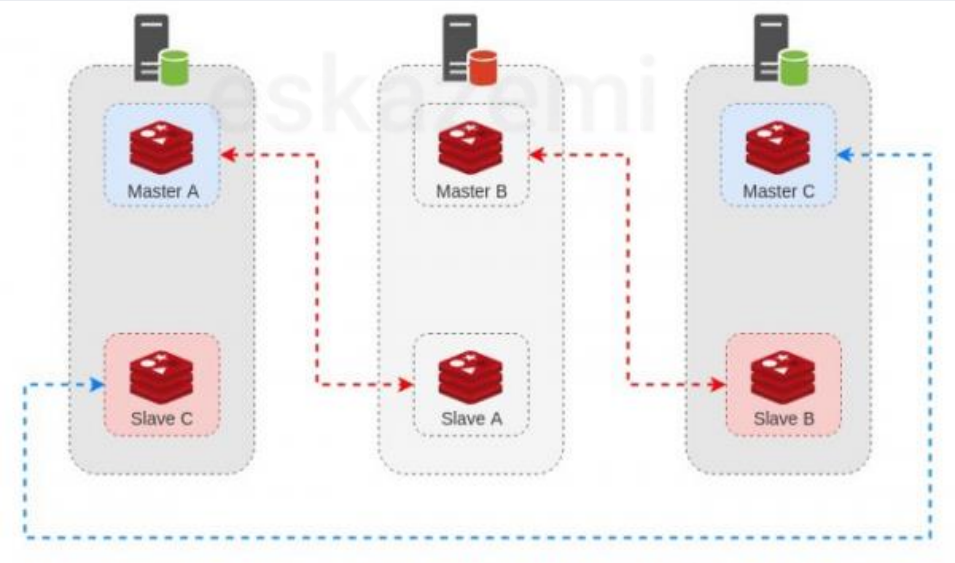
توپولوژی 6 گره



Fail-Over

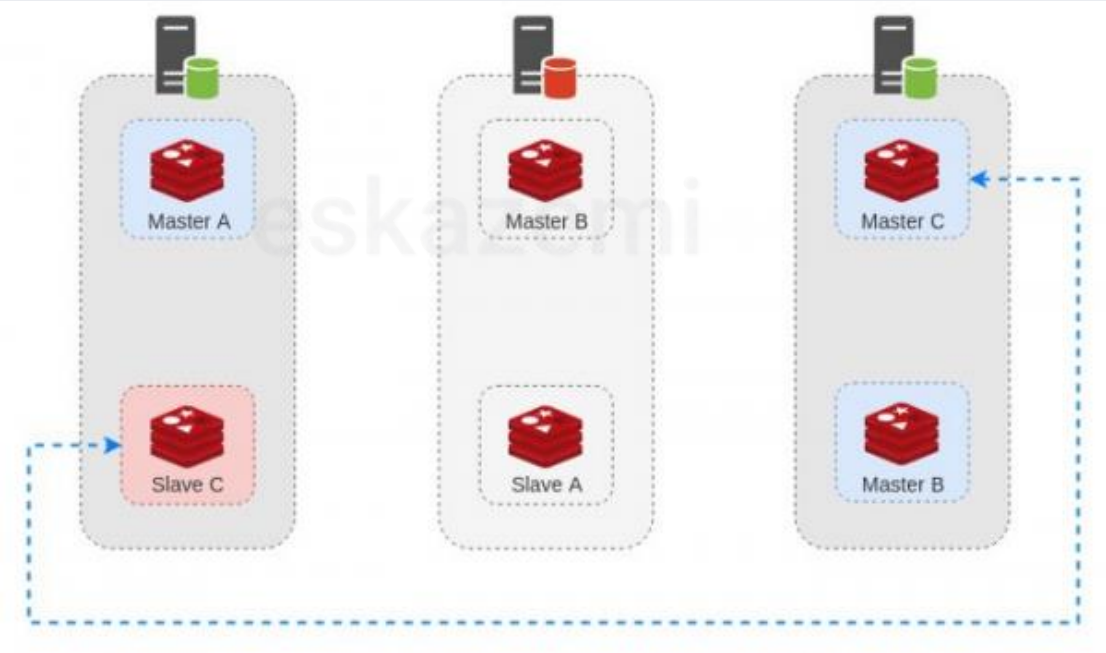
هرگاه یکی از سرور ها دارای مشکل شده و از شبکه خارج بشه Slave مربوطه آن به عنوان Master ارتقا می‌یابد.

فرض کنید در همین دیاگرام , سرور دوم از دسترس خارج بشه :



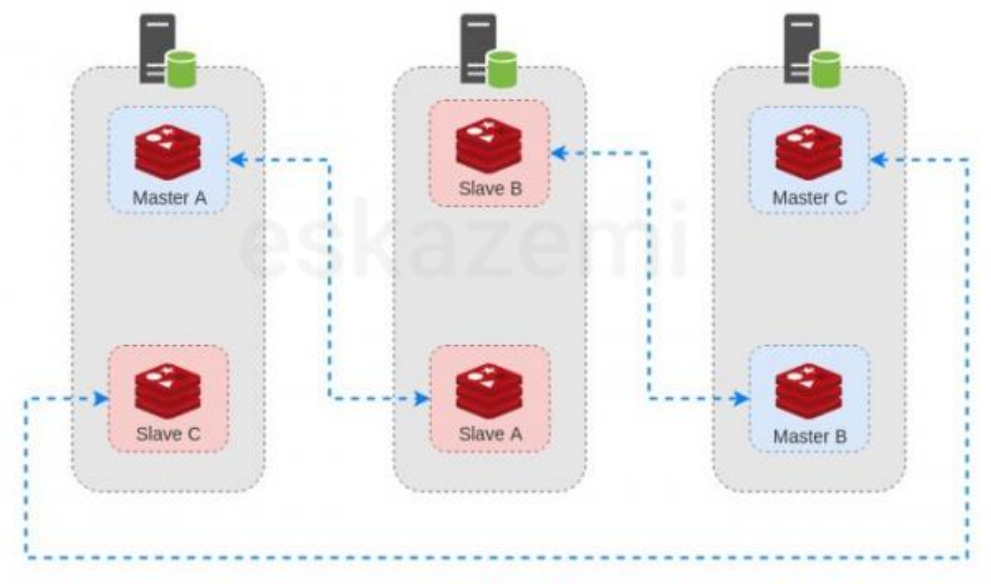
Fail-Over

وقتی کلاستر تشخیص بده که Master B در دسترس نیست , روند FailOver انجام میشه و Slave B رو به عنوان Masterجدید قرار میده :



Fail-Over

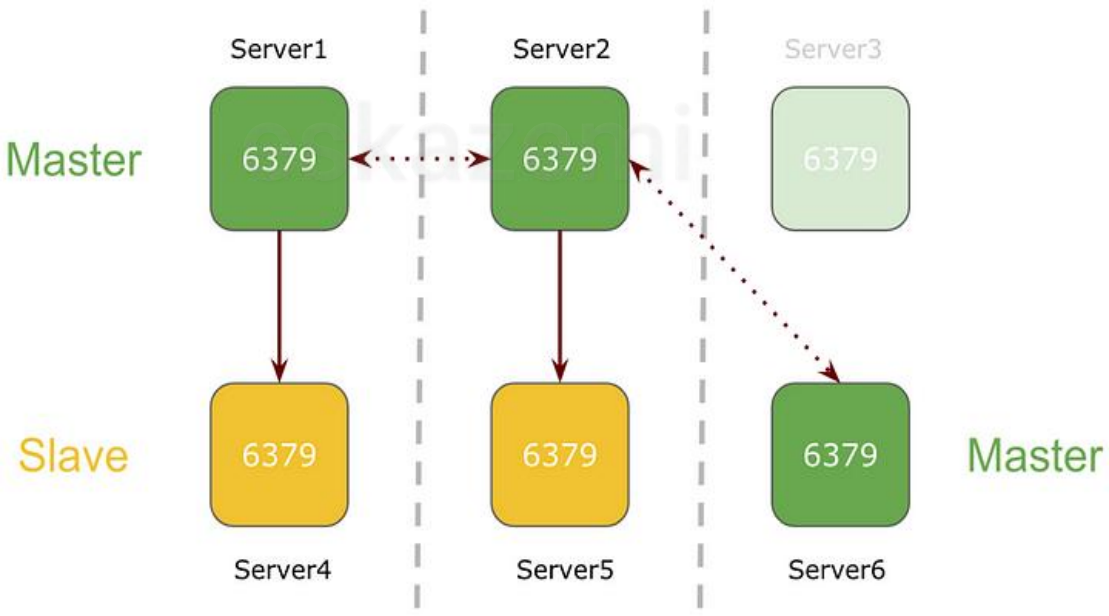
وقتی که مشکل برطرف بشه و سرور دوم مجددا به روند قبلی خودش برگرده , کلاستر مجددا به روزرسانی شده و سرور به مجموعه بر میگرده



توجه کنید که در این حالت , یک شرایط ویژه و خطرناک رخ میده. چرا که اگر به دیاگرام بالا دقت کنید , با اینکه همه ی اجزا وجود دارن ولی دیگه سر جای خودشون نیستن. حالا ما ۲ نود Masterدر یک سرور داریم که اگر اون سرور (سرور اول) از کار بیوفته با مشکلات بیشتری مواجه میشیم.

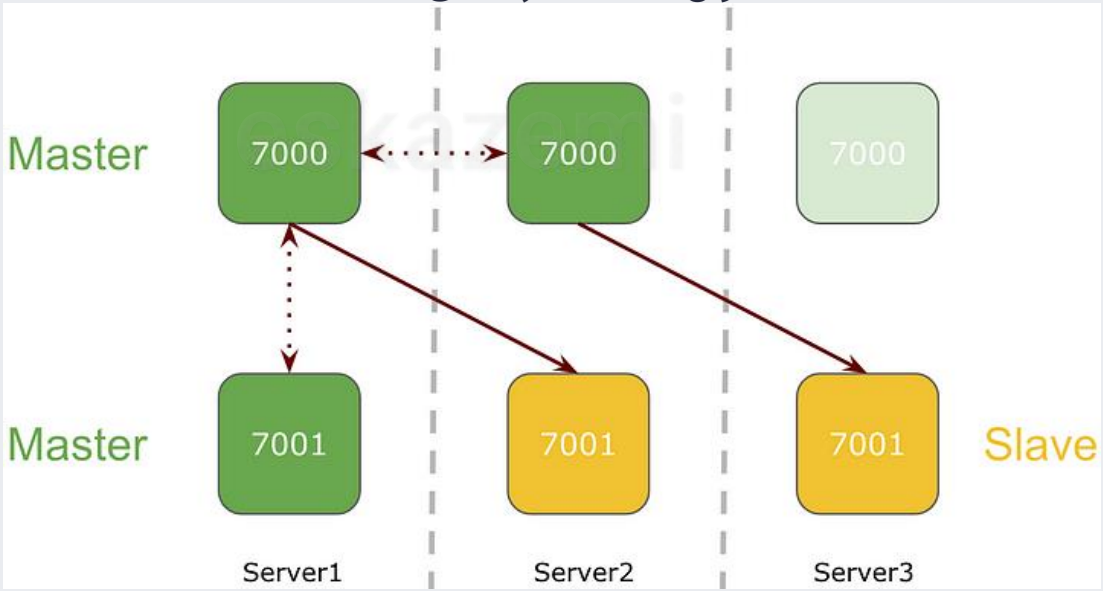
Fail-Over

در این مثال، Master Server3 پایین می آید و slave مربوط به آن به عنوان Master ارتقا می یابد



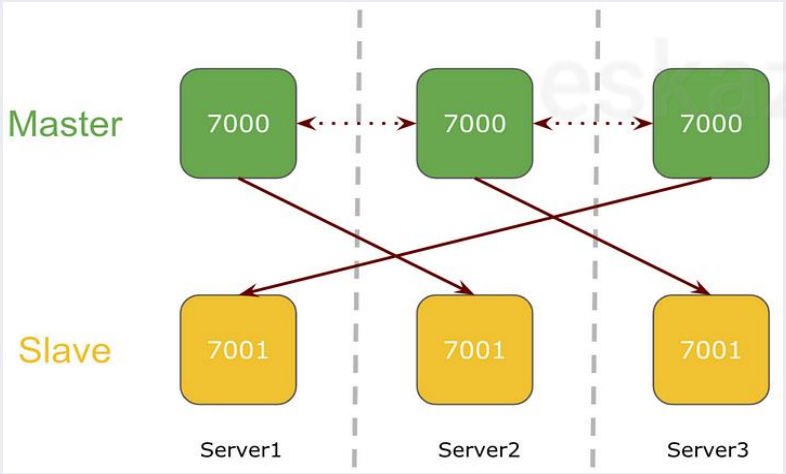
Fail-Over

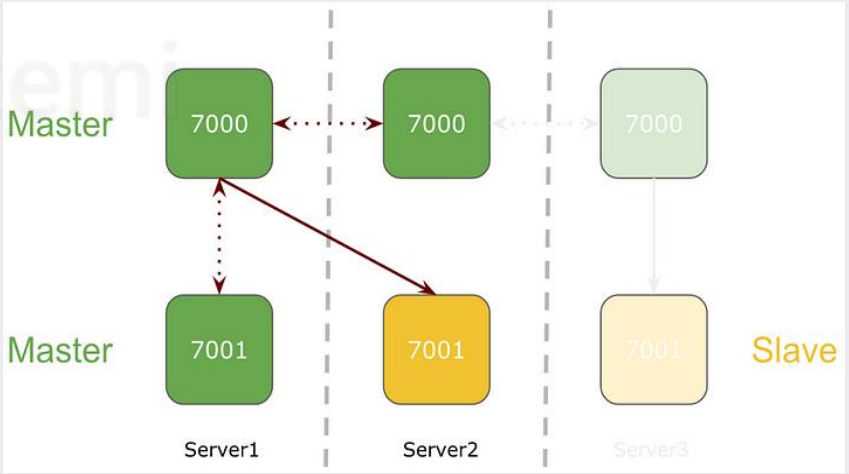
اگر سرویس Redis روی یکی از گره ها در راه اندازی خوشه 3 گره Redis از کار بیفتد ، Slave مربوط به آن به عنوان Master ارتقا می یابد .



Fail-Over

اگر یکی از گره ها در راه اندازی خوشه 3 گره Redis پایین بیاید ، Slave مربوطه آن که روی گره جداگانه اجرا می شود به Master ارتقا می یابد . در مثال زیر ، سرور 3 پایین می آیید و slave در حال اجرا در Server1 به Master ارتقا می یابد .





در نتیجه

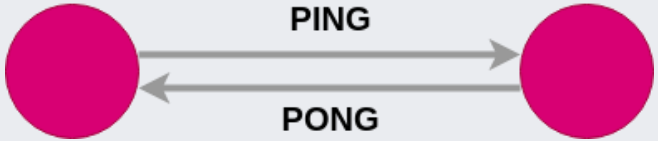
Redis به منظور افزایش دسترسی به داده‌ها، مفهوم اصلی-فرعی یا Master-Slave را معرفی کرده است. در نتیجه، وقفه در یک نقطه، باعث وقفه کل شبکه نمی‌شود. هر کدام از نقاط اصلی در یک Redis cluster حداقل یک نقطه فرعی دارد. وقتی عملکرد این نقطه اصلی متوقف شود یا دسترسی به آن غیرممکن گردد، کلاستر به صورت اتوماتیک، نقطه فرعی آن را برمی‌گزینند. در این حالت، این نقطه، نقطه مستر جدید خواهد بود. بنابراین، مشکل در یک نقطه، باعث توقف عملکرد کل سیستم نمی‌شود.

نحوه تشخیص مشکل و وقفه

- هر کدام از نقاط دارای یک شناسه منحصر به فرد در کلاستر هستند. این شناسه برای تشخیص هر کدام از نقاط در سراسر کلاستر با استفاده از پروتکل gossip به کار می‌رود.

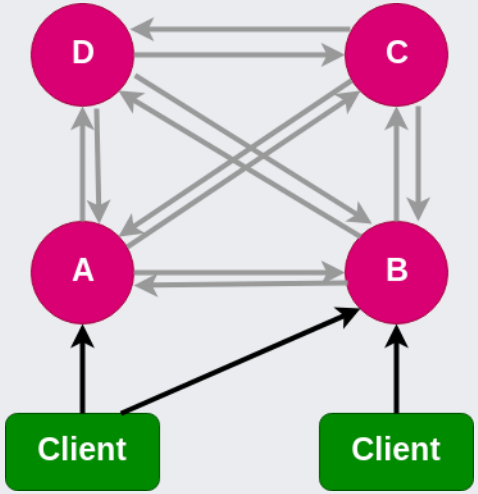
- بنابراین، یک نقاط حاوی اطلاعات زیر خواهد بود.
  - شناسه نقطه، آدرس IP و پورت
  - یک سری علائم (flag)
  - در صورتی که نقطه به صورت فرعی علامت‌گذاری شده باشد، نقطه اصلی آن مشخص است.
  - آخرین باری که نقطه ping شده، چه زمانی بوده است.
  - آخرین باری که pong دریافت شده، چه زمانی بوده است.

- وقتی دستور ping را برای یک نقطه Redis اجرا می‌کنید، اگر عملکرد درستی داشته باشد، جواب را با یک pong می‌دهد.



نقاط در یک کلاستر همیشه با هم ارتباط gossip دارند. در نتیجه، می‌توانند به صورت اتوماتیک وضعیت نقاط دیگر را داشته باشند.

به عنوان مثال، اگر A با B ارتباط داشته باشد (Bرا بشناسد) و به همین ترتیب، B نیز با Cدر ارتباط باشد، معمولاً B پیام gossip در مورد C به نقطه A ارسال می‌کند. سپس A نقطه C را به عنوان بخشی از شبکه درنظر می‌گیرد و سعی می‌کند که با C ارتباط برقرار کند.



دو نوع شناسه یا flag برای تشخیص وقفه در سیستم استفاده می‌شوند؛ PFAIL و FAIL.

PFAIL(مشکل احتمالی): یک مشکل از نوع ناشناخته.

FAIL: این نوع شناسه به شما می‌گوید که یک نقطه در حالت توقف عملکرد است و این موضوع توسط اکثریت نقاط مستر در یک مدت‌زمان مشخص تأیید شده است.

ارتباط نقطه به نقطه از پروتکل باینری (Cluster Bus) پیروی می‌کند که از نظر سرعت و پهنای باند بهینه‌سازی شده است. ولی برای ارتباط نقطه با کلاینت از پروتکل ASCII استفاده می‌شود.

داشتن نسخه کپی بهتر است یا کلاسترینگ؟

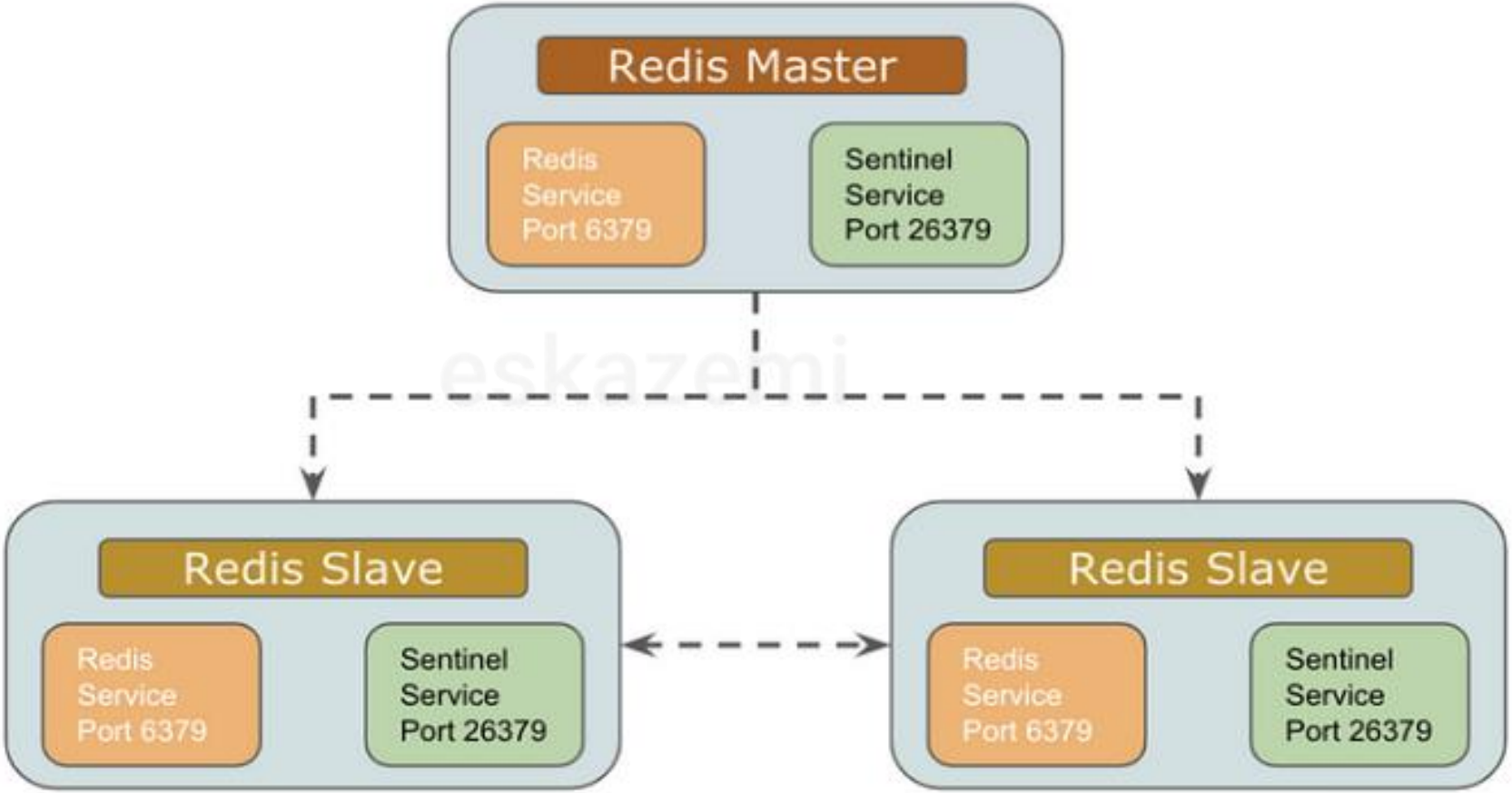
- اگر داده‌هایتان از فضای RAM موجود در یک سیستم بیشتر بود، بهتر است از ابزار Redis Claster برای تقسیم‌بندی داده‌ها در سراسر پایگاه‌های داده چندگانه استفاده کنید.
- در صورتی که مقدار داده‌هایتان از RAM سیستم کمتر بود، می توانید نسخه اصلی و فرعی تکراری برای جلوگیری از شکست سیستم ایجاد کنید و یک سیستم دیده‌بان نیز برای آنها درنظر بگیرید. این دیده‌بان سلامت نسخه‌های اصلی و فرعی را بررسی می‌کند و در صورتی که نسخه اصلی در دسترس نباشد، نسخه فرعی را معرفی می‌کند. شما باید حداقل ۳ دیده‌بان برای تأیید دسترسی به نقاط داشته باشید.
- **آیا به حداقل ۳ نقطه اصلی در ابزار Redis cluster نیاز داریم؟**
- در فرآیند تشخیص مشکل شبکه، حداکثر تعداد نقاط مستر باید به توافق برسند. اگر تنها ۲ نقطه اصلی یا مستر داشته باشید. مثلاً نقاط A و B در صورتی که B با مشکل روبرو شود، نقطه A نمی‌تواند بر اساس پروتکل، به یک تصمیم برسد و به یک نقطه سوم C نیاز دارد که عدم‌دسترسی به نقطه B را تأیید کند.

به هنگام از معماری Master-Slave موارد زیر ما را مجبور می کند از Redis Sentinel استفاده کنیم:

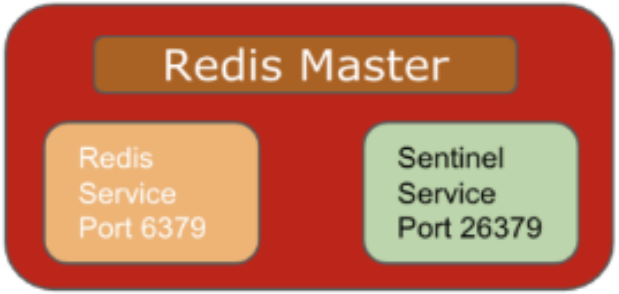
- تنها یک Master با چندین Slave برای تکرار وجود خواهد داشت.
- همه نوشتن به Master می رود که بار بیشتری روی گره اصلی ایجاد می کند.
- اگر Master از دسترس خارج شود(Down) ، کل معماری در معرض SPOF(نقطه شکست واحد)قرار می گیرد.
- معماری Master - Slave زمانی که پایگاه کاربر شما رشد می کند، کمکی به مقیاس دهی (scaling) نمی کند.

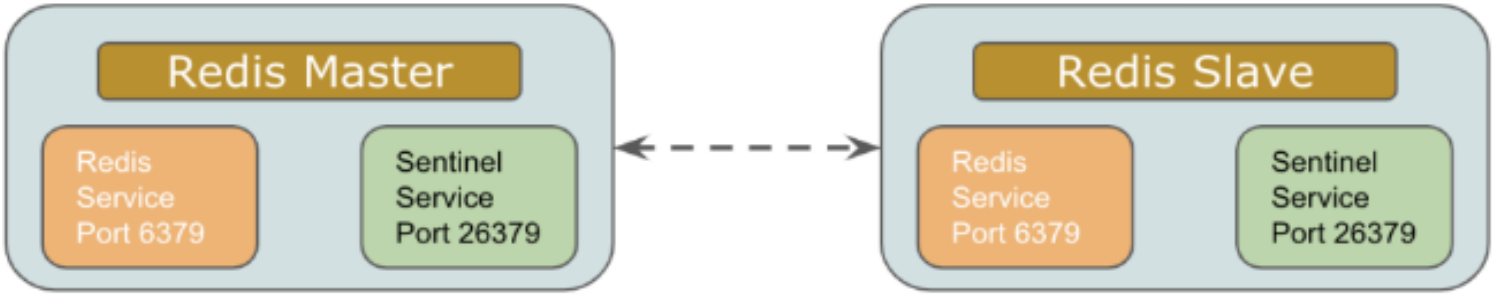
بنابراین ما به یک فرآیند برای نظارت بر Master در صورت خرابی یا خاموش شدن، یعنی Sentinel نیاز داریم.

Initial Setup



Failover Handling





Redis Sentinel vs Clustering

Features	Redis Sentinel	Redis Cluster	Comments
Multiple Logical Databases	✓	✗	Redis standard with Sentinel can offer 16 databases
Strong Consistency	✗	✗	Both have a short interval where writes can be lost.
Horizontal Scaling	✗	✓	Up to 1000 nodes , it scales linearly
Replication	✓	✓	Both async
Sharding Capabilities	✗	✓	Using hash slots
Failover	✓	✓	Depending on the replicas setup
Notification API	✓	✗	Sentinel offers an API to system admins
Multi-key operations	✓	✗	For Redis Cluster, only if all keys belong to same node.
Use replica to scale reads	✓	✓	
Publish/Subscribe	✓	✓	Redis cluster has same extra capabilities

به طور کلی در مورد sentinel می توان گفت:

- ✓ می توانیم نتیجه بگیریم که Sentinel نظارت، تحمل خطا و اعلان ها را ارائه می دهد. همچنین یک ارائه دهنده پیکربندی است. علاوه بر این، می‌تواند قابلیت‌های احراز هویت و کشف خدمات مشتری را فراهم کند. این می تواند 16 پایگاه داده منطقی، تکرار غیر همگام و در دسترس بودن بالا را برای خوشه شما فراهم کند.

- ✓ با این وجود، ماهیت تک رشته ای Redis و موانع مقیاس بندی عمودی، عوامل محدود کننده قابلیت های مقیاس بندی Sentinel هستند. با این حال، برای پروژه های کوچک و متوسط، ممکن است ایده آل باشد.

- ✓ همچنین درست است که استقرار با Sentinel به گره های کمتری نیاز دارد . بنابراین مقرون به صرفه تر استبه گره های کمتری نیاز دارد .بنابراین، مقرون به صرفه تر است.

به طور کلی در مورد Redis cluster می توان گفت :

- ✓ Redis Cluster یک پیاده سازی کاملاً توزیع شده با قابلیت تقسیم خودکار (قابلیت مقیاس بندی افقی) است که برای عملکرد بالا و مقیاس بندی خطی تا 1000 گره طراحی شده است. علاوه بر این، تا زمانی که تمام داده‌های استادها از اکثریت (توسط کابل اصلی یا کپی برای انجام خرابی) قابل دسترسی باشد، درجه معقولی از ایمنی نوشتن و ابزاری برای زنده ماندن در بلایا فراهم می‌کند.

- ✓ نسخه Clusterهمچنین توانایی پیکربندی مجدد نگاشت بین Mastersو Replicasرا برای تعادل مجدد در صورت بروز چندین خرابی مستقل از یک گره ارائه می دهد. با این حال، بار دیگر، گره‌های بیشتری برای خوشه‌ای قوی‌تر و در نتیجه هزینه بیشتر ضروری هستند. علاوه بر این، مدیریت گره‌های بیشتر و متعادل‌سازی خرده‌ها پیچیدگی بیشتری است که نمی‌توان نادیده گرفت.

- ✓ نکته دیگر این است که در طول یک پارتیشن شبکه، بخش اقلیت خوشه از پذیرش درخواست ها خودداری می کند. از سوی دیگر، یک استقرار Sentinelبسته به تنظ ممکن است تا حدی به کار خود ادامه دهد. همانطور که قبلا ذکر شد، در دسترس بودن خوشه و تحمل خطا به پیکربندی و ترکیب خوشه بستگی دارد.

- ✓ به طور خلاصه، Redis Cluster وقتی صحبت از استقرار بزرگ با مجموعه داده‌های بزرگی می‌شود که به توان عملیاتی و مقیاس‌پذیری بالا نیاز دارند، می‌درخشد.

Redis – Server commands

Sr.No	دستور	توضیح
1	SELECT n	به صورت پیش فرض روی دیتایس 0 کار می کنیم اما می توان با این دستور دیتابیس رو تغییر داد 127.0.0.1:6379[3] داخل دیتابیس سوم هستیم
2	CONFIG GET databases	پاسخ این دستور عدد 16 را به ما بر می گرداند به این معناست به صورت پیش فرض می توانیم 16 دیتابیس را ایجاد کنیم و با آنها کار کنیم
3	SWAPDB n m	اطلاعاتی که داخل دیتابیس n(عدد) بفرست به دیتابیس m (عدد)
4	DBSIZE	تعداد کلید هایی که داخل دیتابیس هست به ما میدهد

فرمان های سرور Redis اساسا برای مدیریت سرور Redis استفاده می شوند.

Redis – Server commands

Sr.No	دستور	توضیح
5	info	اطلاعات بیشتری در مورد دیتابیس در اختیار ما قرار می دهد
6	CLIENT LIST	اطلاعات کسانی که وصل شدن به سرور به ما میدهد
7	CLIENT GETNAME	گرفتن نام متصل شونده به سرور
8	CLIENT SETNAME	ست کردن نام برای client
9	MONITOR	به تمام درخواست های دریافت شده توسط سرور در زمان واقعی گوش می دهد.

Redis – Server commands

Sr.No	دستور	توضیح
10	FLUSHALL	حذف می کند دیتا رو از تماس دیتابیس ها
11	FLUSHDB	حذف میکند دیتا رو از دیتابیس که یوزر در حال استفاده از آن می باشد
12	LASTSAVE	برمی گرداند Uinx time stamp آخرین ذخیره موفق بر روی دیسک
13	TIME	زمان حال حاضر سرور را بر می گرداند

Thanks!

Any questions?

You can find me at:

- @eskazemi
- m.esmaeilkazemi@gmail.com