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 Persistance

Different kind of persistance in Redis

Redis سرعت در درجه اول قرار داد و همه consistency guarantees در درجه دوم قرار داد این ممکن است یک موضوع بحث بر انگیز باشد ، اما حقیقت دارد

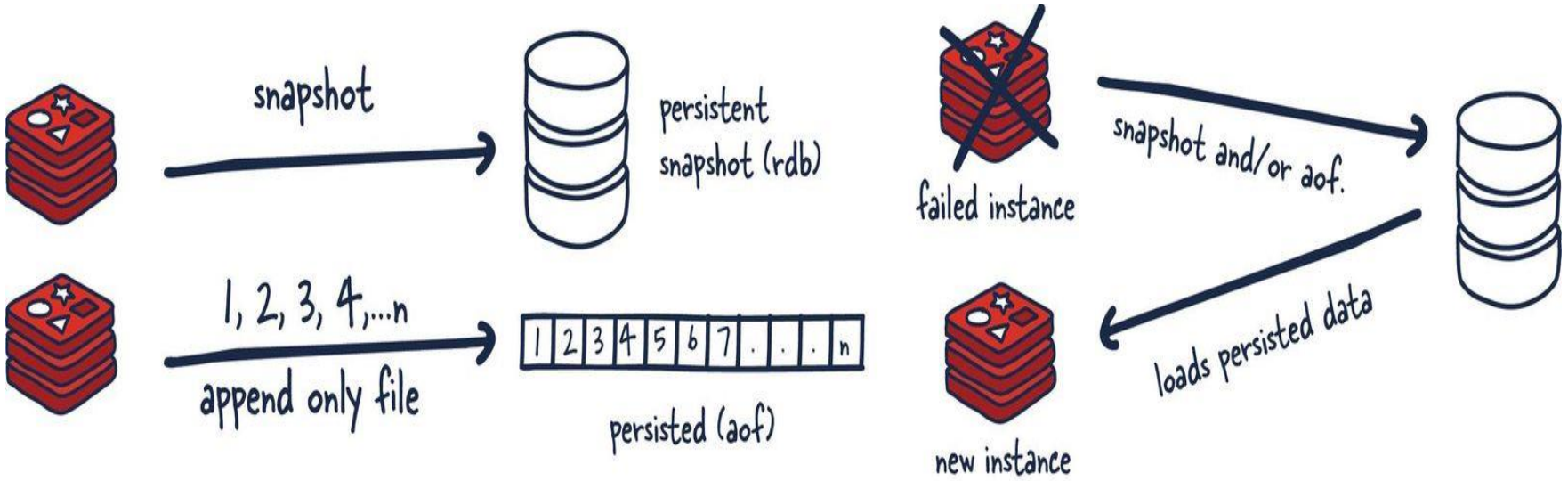
persistence

Redis همان طور که گفتیم یک پایگاه داده کاملا پایدار است و اطلاعات داخل Ram ذخیره می کنه برای اینکه سرعت خواندن و نوشتن اطلاعات بالا باشه حالا ویژگی مهم که Persistence گفته می شود اساسا به معنای نوشتن داده ها روی حافظه ذخیره سازی که با داوم مانند SSD یا HDD است Persistence در بازیابی داده ها خیلی مهمه زیرا اطمینان حاصل می کنه که داده ها برای استفاده ، در صورت راه اندازی مجدد سرور یا سوختن رم در دسترس هستند

Redis فراهم کرده 4 روش Persistence برای کاربرها هر کدوم از این چهار روش براساس نیازمندی های خودشون می توانند انتخاب کنند این 4 روش عبارتند از :

1. Redis persistence default(RDB)
2. Append only file(AOF)
3. No persistence
4. RDB + AOF

AOF and RDB


snapshot
persistent
snapshot (rdb)
failed instance
snapshot and/or aof.
1, 2, 3, 4,...n
append only file
1 2 3 4 5 6 7 . . . n
persisted (aof)
new instance
loads persisted data

RDB

RDB File (snapshot)

RDB (snapshot)

- یک snapshot یک کپی نقطه به نقطه از داده های Redis ذخیره شده در حافظه است. snapshot با استفاده از گزینه تداوم Redis DataBase (RDB) ایجاد می شود که اجازه می دهد وضعیت پایگاه داده Redis در فواصل زمانی مشخص روی دیسک ذخیره شود. هنگامی که یک RDB snapshot گرفته می شود، Redis یک child process برای انجام snapshot ایجاد می کند، که به main process اجازه می دهد تا به درخواست های سرویس دهی ادامه دهد. در نهایت یه فایل واحد ایجاد می شود که نمایش نقطه به نقطه داده هاست.

- عیب اصلی این مکانیزم این است که داده های بین snapshots از دست خواهند رفت. علاوه بر این، این مکانیزم ذخیره سازی نیز متکی بر forking فرآیند اصلی است و در یک مجموعه داده بزرگ تر، این امر ممکن است منجر به تاخیر لحظه ای در ارائه درخواست ها شود.

- اگر سرور down شود می توان از این snapshot ها برای بازیابی وضعیت پایگاه داده قبلی استفاده کرد. فاصله ای که در آن snapshot ها گرفته می شوند را می توان تنظیم کرد. به عنوان مثال، می توانید پایگاه داده خود را طوری تنظیم کنید که هر ۱ دقیقه یک snapshot بگیرد اگر ۱۰ تغییر در مجموعه داده ها اتفاق افتاده باشد یا هر ۵ دقیقه یک بار اگر ۱۰۰۰ تغییر در مجموعه داده ها اتفاق افتاده . از این snapshots ها برای بازگرداندن پایگاه داده به هر نقطه ای از زمان در صورت بروز فاجعه استفاده کنید.

- به طور پیش فرض، Redis این snapshot ها را در یک فایل باینری به نام dump.rdb ذخیره می کنند. و این فایل RDB هر زمان که یک snapshot جدید ایجاد می شود، جایگزین می شود.

Forking

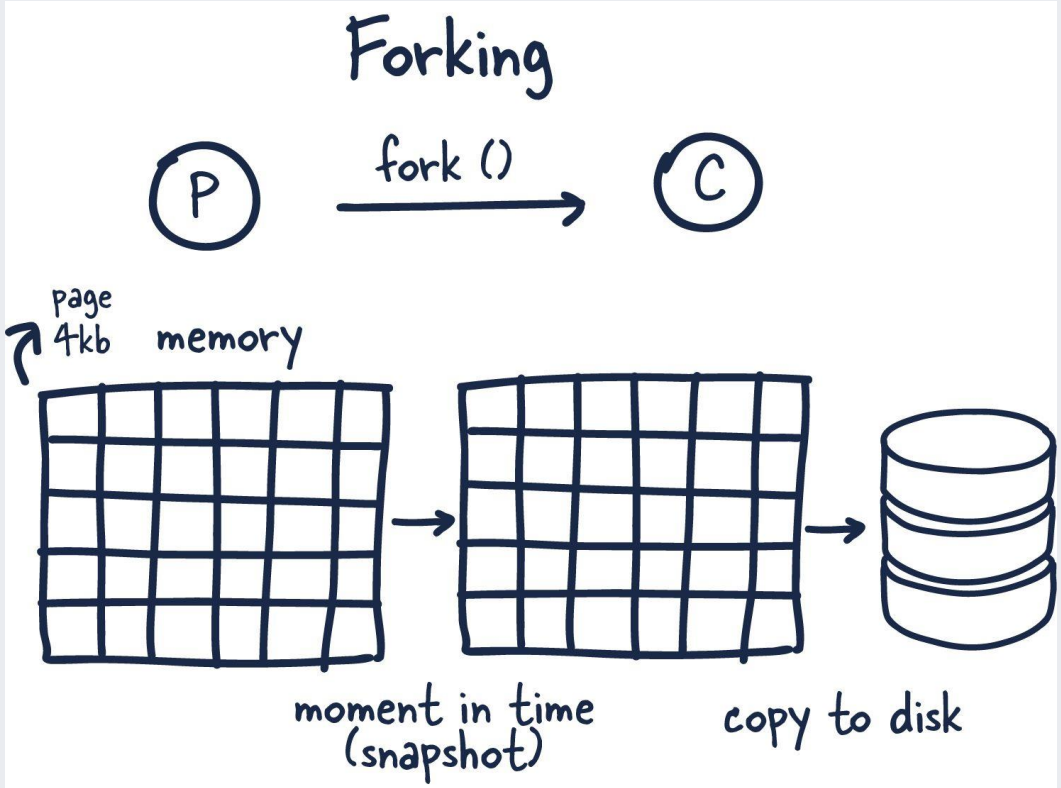
Redis takes snapshots by forking its process into a **parent** and **child** process. Then, the child process starts writing a new RDB file. And when it’s done writing the new RDB file, it replaces the old one.

parent

درخواست ها پردازش و پاسخ می دهد

child

عملیات backup گیری را انجام می دهد



Snapshots vs. backups

Snapshots و backup طراحی شده اند برای دو چیز متفاوت در حالی که snapshot حمایت میکند از دوام داده ها (به صورت اتوماتیک دیتا رو برمیگرداند زمانی که هیچ کپی از دیتا در حافظه نداریم.) اما backup پشتیبانی میکند از disaster recovery (زمانی که کل خوشه باید از ابتدا بازسازی شود)

مزیت های RDB

- ✓ همان طور که گفته شد، فایل های RDB بسیار سریع تر از AOF در حافظه بارگذاری می شوند.
- ✓ فایل های RDBبرای پشتیبان گیری بسیار خوب هستند.
- ✓ Restore کردن اطلاعات RDB سریعتر از AOF می باشد.
- ✓ Persistence پیش فرض Redis می باشد.

Commands

برای تغییر تنظیمات مربوط RDB (فعال سازی RDB) ابتدا باید فایل redis.conf را با Text editor باز کنید :

```
$ sudo nano /etc/redis/redis.conf -> Linux
$ sudo nano /opt/homebrew/etc/redis.conf -> macOS
```

به قسمت SNAPSHOTTING بروید و در آنجا تنظیمات مربوط به RDB را خواهید دید:

```
# save 60 10000
```

دستورات شبیه بالا به معنای آن است در 60 ثانیه قبلی اگر 10000 هزار تغییر داشته باشیم backup گیری انجام بده. Syntax آن را در پایین مشاهده می کنید و در دستور بالا همان طور که می بینید comment شده و غیر فعال است اگر از حالت کامنت در بیاریم فعال می شود.

```
#save <seconds> <changes>
```

Commands in redis-cli

این command save استفاده می شود برای ایجاد backup از دیتابیس که یوزر در حال استفاده از آن می باشد.

Syntax

```
127.0.0.1:6379> SAVE
```

Example

```
127.0.0.1:6379> SAVE
OK
```

این command ایجاد می کند فایل dump.rdb در دایرکتوری redis و نکته ای که باید در نظر گرفت از این دستور زمانی باید استفاده کرد که تعداد کلید ها کم می باشد.

زمانی که تعداد کلید خیلی زیاد می باشد از دستور BGSAVE استفاده می کنیم که backup گیری در background انجام دهد

Example

```
127.0.0.1:6379> BGSAVE
Background saving started
```

AOF

Append-only file (AOF) data persistence

AOF

AOF ⟶ Append - Only File

- ✓ یک مکانیزم ثبت است که هر دستوری که ما اجرا کنیم در پایگاه داده Redis را در یک فایل لاگ در دیسک می نویسد. (به عبارتی هر عملیات نوشتن به انتهای فایل لاگ ضمیه می کند تا در صورت نیاز دوباره اجرا شود)
- ✓ AOF برای حفظ دوام داده ها مفید است، زیرا فایل لاگ می تواند برای بازسازی پایگاه داده در صورت خرابی مورد استفاده قرار گیرد. هنگامی که Redisمجددا راه اندازی می شود، فایل لاگ را می خواند و عملیات نوشتن در فایل را مجددا اجرا می کند تا پایگاه داده را به حالت قبلی خود بازگرداند.

AOF

AOF دوام داده بهتری نسبت به گزینه RDB فراهم می کند با این حال، کندتر است و به فضای دیسک بیشتری نیاز دارد، زیرا باید هر عملیات نوشتن را در فایل لاگ بنویسد.

فایل AOF را می توان طوری پیکربندی کرد که وقتی بیش از حد بزرگ می شود، با استفاده از فرایندی به نام AOF fsync در پس زمینه بازنویسی شود. نکته منفی این است که این فرمت فشرده نیست و از دیسک بیشتری نسبت به فایل های RDBاستفاده می کند.

هر قطعه از پایگاه داده اضافه می کند خط جدیدی را به یک از روش های زیر به فایل persistence خودش:

every second (fast but less safe)
every write (safer but slower)

Commands

برای تغییر تنظیمات مربوط AOF (فعال سازیAOF) ابتدا باید فایل redis.conf را با Text editor باز کنید :

```
$ sudo nano /etc/redis/redis.conf -> Linux
$ sudo nano /opt/homebrew/etc/redis.conf -> macOS
```

این خط در فایل کانفیگ پیدا کنید :

```
# appendonly no
```

فعال سازیAOF : برای فعال سازی از کامنت در بیارید و به صورت پیش فرض AOF غیر فعال می باشد که اگر به جای no ، yes قرار دهید AOF فعال می شود. و مطمئن شوید که ویژگی های زیر فعال هستند:

```
appendfsync everysec
no-appendfsync-on-rewrite no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
```

شما همچنین می توانید نام فایل AOF را تغییر دهید با چنین دستوری:

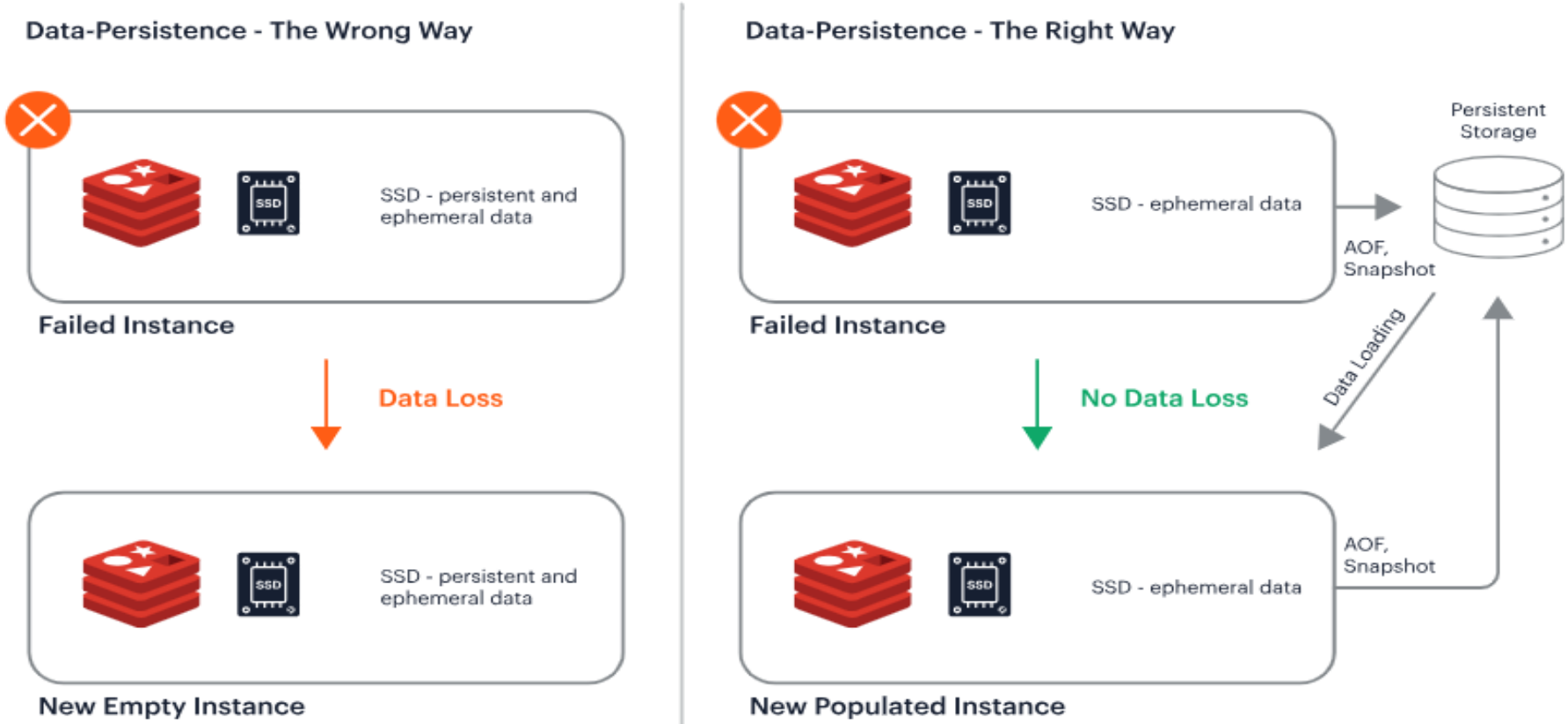
```
appendonlyfilename “appendonly.aof”
```

تغییرات را ذخیره کنید سپس سرور Redis را دوباره راه اندازی کنید:

```
$ sudo service redis-server restart
```

ترکیب AOF و RDB

امکان ترکیب AOF و RDB در Redis وجود دارد که باید بین سرعت و دوام یک tradeoff داشته باشید در این صورت مایل به انجام آن (فعال سازی هر دو) هستید .من فکر می کنم این یک راه قابل قبول برای راه اندازی Redis است. در صورت راه اندازی مجدد، به یاد داشته باشید که اگر هر دو فعال باشند، Redis از AOF برای بازسازی داده ها استفاده خواهند کرد زیرا کامل ترین است.



eskazemi

هیچ اصراری وجود ندارد بر فعال سازی persistence

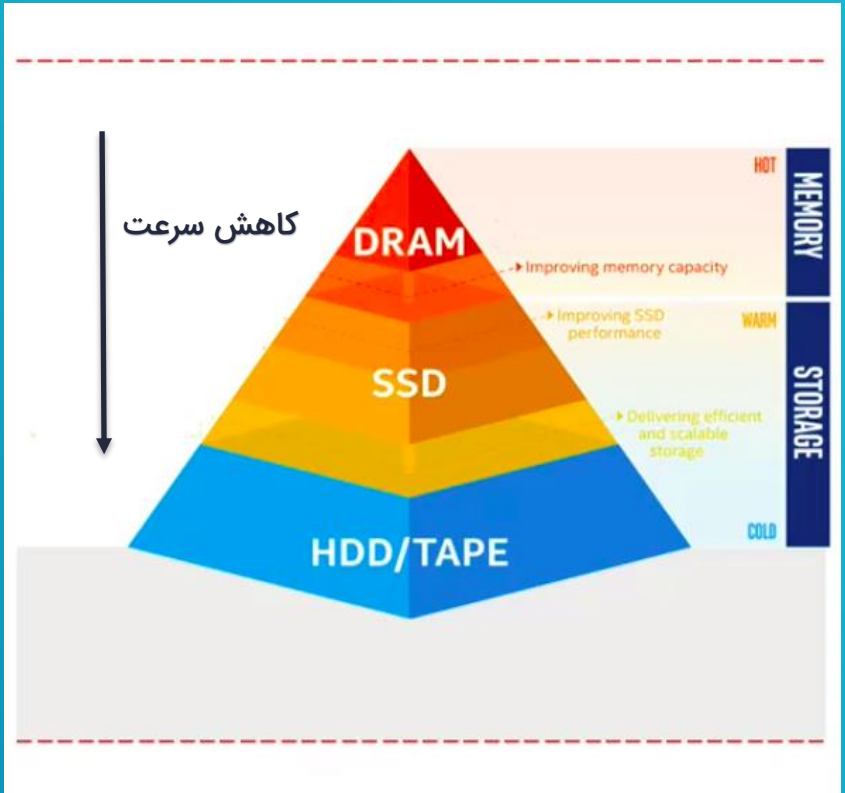
در صورت تمایل می توانید persistence را به طور کلی غیرفعال کنید. این سریع ترین راه برای اجرای ردیس است و هیچ تضمین دوامی ندارد

آیا از آنجایی که Redis داده ها را روی RAM نگه می دارد، بعد از خاموش و روشن شدن و یا هر اتفاق غیر قابل پیش بینی که بیافتد و RAM سیستم خالی شود داده های ما پاک می شوند؟

خیر، Redis برای نگه داری دائمی داده ها آنها را با توجه به تنظیماتی که ما برای آن مشخص می کنیم به دیسک اصلی سیستم منتقل می کند و بعد از پاک شدن RAM دوباره می تواند آنها را منتقل کند و کار را از سر بگیرد.

این ویژگی باعث شده اصطلاحا به آن on-disk persistence بگویند و این کار را می تواند در سطوح مختلفی انجام دهد. این سطوح شامل موارد زیر می شوند:

- نکته مهم : در سرور ، محدودیت رم داریم بنابراین ردیس هوشمندانه ذخیره اطلاعات رو کنترل می کنه و اطلاعات که کاربرد بیشتری دارند داخل رم نگه می داره و مابقی رو داخل هارد ذخیره می کنه .
- اما این موضوع هم در نظر داشته باشید که بیشترین استفاده از ردیس به عنوان حافظه نهان که اطلاعات در رم ذخیره می شود برای افزایش سرعت خواندن و نوشتن.



Redis - Benchmarks

Redis benchmark ابزاری برای بررسی عملکرد Redis با اجرای همزمان n دستور است.این ابزار زمانی مفید هست اش که شما تغییری در ردیس یا برنامه تون ایجاد کردن و می خواین ببینین که چه تاثیری روی سرعت گذاشته

basic syntax of Redis benchmark.

```
redis-benchmark [option] [option value]
```

به طور مثال چک کردن عملکرد redis با 10000 دستور

```
redis-benchmark -n 100000

PING_INLINE: 141043.72 requests per second
PING_BULK: 142857.14 requests per second
SET: 141442.72 requests per second
GET: 145348.83 requests per second
INCR: 137362.64 requests per second
LPUSH: 145348.83 requests per second
LPOP: 146198.83 requests per second
SADD: 146198.83 requests per second
SPOP: 149253.73 requests per second
LPUSH (needed to benchmark LRANGE): 148588.42 requests per second
LRANGE_100 (first 100 elements): 58411.21 requests per second
LRANGE_300 (first 300 elements): 21195.42 requests per second
LRANGE_500 (first 450 elements): 14539.11 requests per second
LRANGE_600 (first 600 elements): 10504.20 requests per second
MSET (10 keys): 93283.58 requests per second
```

Redis - Configuration

در Redis وجود دارد یه فایل کانفیگ به نام redis.conf
که در دسترس در دایرکتوری redis همچنین شما می توانی ببینید و
و تنظیم کنید کانفیگ های مربوط به Redis با استفاده از command -> CONFIG

Syntax

```
redis 127.0.0.1:6379> CONFIG GET CONFIG_SETTING_NAME
```

Example

```
redis 127.0.0.1:6379> CONFIG GET loglevel
1) "loglevel"
2) "notice"
```

Access Control Lists (ACLs)

از Redis 6 به بعد ویژگی لیست های کنترل دسترسی (ACL) برای اهداف امنیتی و انطباق بهتر در دسترس است.(دسترسی مربوط به کاربر ها را تنظیم می کند) پیش از ورژن 6، مفهوم کاربران وجود نداشت و تنها یک استراتژی احراز هویت مبتنی بر رمز عبور برای اهداف احراز هویت استفاده می شد.

Access Control Lists (ACLs)

دستور Redis AUTH که در ورژن 6 گسترش یافت و به ما اجازه داد تا یک نام کاربری را به عنوان پارامتری برای دستور AUTH مشخص کنیم.(که در ورژن های قبلی فقط پارامتر پسورد می پذیرفت)

```
AUTH <username> <password>
```

به منظور حفظ سازگاری با نسخه های قبلی ، Redis در آخرین نسخه ها (6 به بالا) کاربر پیش فرض را به کاربران ACL معرفی کرد.
اگر تنها یک پارامتر را با دستور AUTH مشخص کنیم، نشان می دهیم که نام کاربری برای احراز هویت برابر کاربر "پیش فرض" است. این رویکرد طراحی با مکانیزم احراز هویت قبلی Redis سازگار است، بنابراین سازگاری با ورژن های قبلی را هم فراهم می کند.

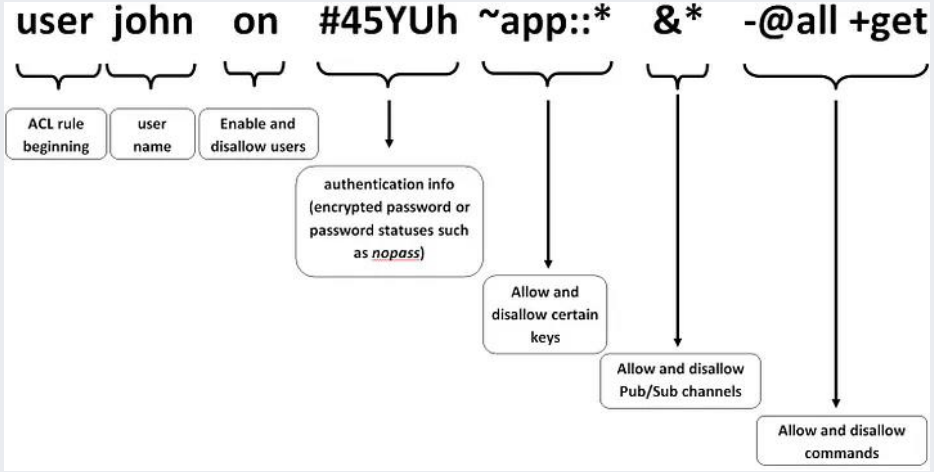
Configuring ACLs using the ACL command

برای درک ویژگی ACL، بیایید ابتدا با استفاده از دستور ACL LIST، ACL های فعال فعلی را در یک سیستم جدید که هیچ کاربر ACL را پیکربندی نکرده ایم، بررسی کنیم. ACL چگونگی احراز هویت یک کاربر معین و همچنین منابع و مرزهای اجرایی که برای هر کاربر اعمال می شود را توضیح می دهد.

```
127.0.0.1:6379> ACL LIST
1) "user default on nopass ~*  &*  +@all"
```

دو کلمه اول در قانون ACL "کاربر" و پس از آن نام کاربری ("default") هستند. کلمات بعدی قوانین ACL هستند که جنبه های احراز هویت و مجوز را توصیف می کنند. کلمه بعدی نشان می دهد که آیا کاربر مشخص شده فعال (on) است یا غیرفعال (off) در این مثال، کاربر پیش فرض به گونه ای پیکربندی شده است که فعال باشد، که نشان دهنده وضعیت روشن است. علاوه بر این، کاربر پیش فرض طوری پیکربندی شده است که نیازی به رمز عبور (nopass) نداشته باشد، دسترسی به هر کلید ممکن (~*) به تمام دستورات دسترسی دارد (+@all)

ساختار قانون ACL



دستورها و دسته ها را با پیشوند "+" برای دستورها یا پیشوند "@+" برای دسته های دستور وارد کنید.

دستورات و دسته را با پیشوند «-» برای دستورات یا پیشوند «@-» برای دسته های فرمان حذف کنید.

اجازه دسترسی به pub/sub channels با پیشوند "&" (پشتیبانی شده برای پایگاه های داده با نسخه Redis 6.2 و بالاتر).

کلیدها یا الگوهای کلید را با پیشوند "~" اضافه کنید.

Create an ACL lucy User:

```
127.0.0.1:6379> ACL SETUSER lucy
OK
127.0.0.1:6379> ACL LIST
1) "user default on #43f8a2ad1882637f749ce419d33a02f74debb66991d7d65aade4eea9ded2a120 ~* &* +@all"
2) "user lucy off resetchannels -@all"
```

وضعیت off نشان می دهد که احراز هویت از کار افتاده است. یعنی اجرای دستور AUTH روی این کاربر کار نخواهد کرد. همچنین برای این کاربر رمز عبور تعیین نشده است.
resetchannels مقداری مربوط به acl-pubsub که به صورت پیش فرض به هنگام ایجاد کاربر برای کاربر تنظیم می شود دسترسی به کانال کاربر را با شستشوی الگوهای مجاز کانال و قطع ارتباط مشتریان Pub/Subکاربر محدود می کند که مکانیزم امنیتی بهتری را فراهم می کند. این رفتار پیش‌فرض محدودکننده pubsub از Redis 7.0 به بعد معرفی شد. در نسخه‌های قبلی Redis وقتی صحبت از رفتار پیش‌فرض pubsub می‌شود که به همه کانال‌های (&*)Pub/Sub دسترسی پیدا می‌کند، رویکرد ساده‌تری داشتند.
all@- کاربر به هنگامی که ایجاد می شود توانایی اجرای هیچ دستوری ندارد.
علاوه بر این، هیچ الگوی کلیدی وجود ندارد که کاربر بتواند به آن دسترسی داشته باشد زیرا الگوی تعیین کننده نحو (~) در قانون وجود ندارد.

created user and add a password and key pattern and commands allowed to perform

```
127.0.0.1:6379> ACL SETUSER lucy on >strongpassword ~cached:* +get
OK
127.0.0.1:6379> ACL LIST
1) "user default on #43f8a2ad1882637f749ce419d33a02f74debb66991d7d65aade4eea9ded2a120 ~* &* +@all"
2) "user lucy on #05926fd3e6ec8c13c5da5205b546037bdcf861528e0bdb22e9cece29e567a1bc ~cached:* resetchannels -@all +get"
```

وضعیت on به این معنی است که احراز هویت فعال است. این بدان معناست که اجرای فرمان AUTH روی این کاربر کار خواهد کرد.
کاربر پسورد تعیین کرد زیرا هش رمز وجود دارد.
الگوی کلیدی که کاربر مجاز است با آن تعامل داشته باشد، "Cached:" است. به این معنی است که هر کلیدی که با الگوی "Cached:" شروع می شود، می تواند تعامل داشته باشد.
کاربر مجوز دارد دستور get اجرا کند به این معنی است که کاربر فقط می تواند کلیدهای الگوی مشخص شده را دریافت کند. انجام سایر اقدامات بر روی الگوی کلید مشخص شده مجاز نیست.

Command Categories:

```
127.0.0.1:6379> ACL CAT
1) "keyspace"
2) "read"
3) "write"
4) "set"
5) "sortedset"
6) "list"
7) "hash"
8) "string"
9) "bitmap"
10) "hyperloglog"
11) "geo"
12) "stream"
13) "pubsub"
14) "admin"
15) "fast"
16) "slow"
17) "blocking"
18) "dangerous"
19) "connection"
20) "transaction"
21) "scripting"
```

اگر بخواهیم قانون ACL را با مشخص کردن همه دستورها یکی پس از دیگری به روز کنیم، کار بسیار وقت گیری خواهد بود ، برای جلوگیری از این موضوع، مفهوم دسته های دستور را در قوانین Redis ACL داریم.
برای نمایش تمام دسته های دستور موجود می توانیم از دستور ACL CAT در داخل Redis استفاده کنیم.

Command Categories:

برای شناسایی دستورات اختصاصی داده شده به دسته دستورات خاص، می توانیم از این دستور استفاده کنیم ACL CAT
command_category استفاده کنیم با اجرای این دستور مجموعه ای از دستورات مرتبط با مقدار دسته بندی بر می گردد.

```
127.0.0.1:6379> ACL CAT dangerous
1) "keys"
2) "config|rewrite"
3) "config|resetstat"
4) "config|get"
5) "config|set"
6) "pfdebug"
7) "monitor"
8) "info"
9) "bgrewriteaof"
10) "flushall"
```

همان طور که گفته شد، می توانیم با گنجاندن دسته بندی های ACL، قانون ACL را بیش ازپیش بهبود دهیم.

```
127.0.0.1:6379> ACL SETUSER lucy -set -get +@all -@dangerous
OK
127.0.0.1:6379> ACL LIST
1) "user default on #43f8a2ad1882637f749ce419d33a02f74debb66991d7d65aade4eea9ded2a120 ~* &* +@all"
2) "user lucy on #05926fd3e6ec8c13c5da5205b546037bdcf861528e0bdb22e9cece29e567a1bc ~cached:* ~users:*
~logistics:* resetchannels +@all -@dangerous"
```

ابتدا دستورات "get" و "set" را حذف می کنیم سپس با افزودن "@all" به همه دستورات دسترسی دادیم و سپس -"@dangerous" همه دستورات موجود در دسته "dangerous" را حذف کرده ایم .

ACL - Commands

Sr.No	دستور	توضیح
1	ACL LIST	تمام کاربر ها را برای شما بر می گرداند
2	ACL CAT	تمام دسته بندی ها را بر می گرداند
3	ACL CAT category_name	تمام دستورات مربوط به دسته بندی خاص را بر می گرداند
4	ACL SETUSER username	کاربر جدید ایجاد می کند
5	ACL SETUSER username on	
6	ACL SETUSER username password	ست کردن پسورد

ACL - Commands

Sr.No	دستور	توضیح
7	ACL GENPASS	یک پسورد قوی برای ما ایجاد می کند
8	ACL WHOAMI	اعلام می کند چه کاربر به سرور متصل است
9	AUTH username password	ورود کاربر
10	ACL GETUSER username	اطلاعات مربوط به یک کاربر را برمی گرداند

Thanks!

Any questions?

You can find me at:

- @eskazemi
- m.esmaeilkazemi@gmail.com