
Beyond the Cache
with
Redis + Node.js
redis

# Guy Royse

Developer Advocate

redis

@guyroyse

github.com/guyroyse

guy.dev


# Assumptions

I am assuming that…

…you know modern JavaScript.

…you have a basic understanding of Express.

…you know absolutely nothing about Redis.

# What Will We Cover?

We will cover…

…installing Redis and RedisInsight.

…using Redis commands directly.

…using Redis from Node.js.

…using RedisJSON & RediSearch.

redis


# What Are We Doing Here?

We're building a Bigfoot Tracker API

**This is a workshop, not a talkshop.**

express

node JS

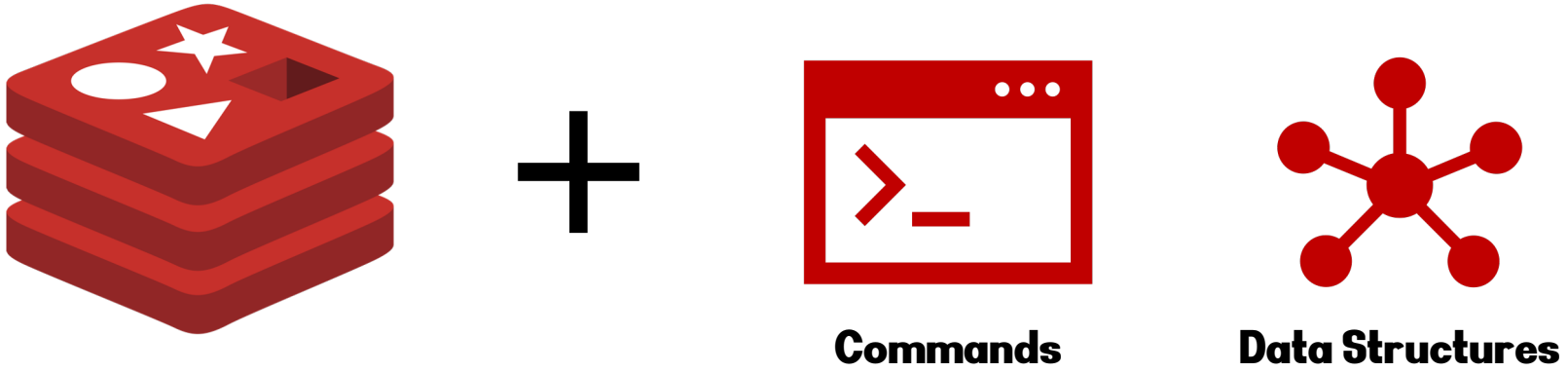
redis

# Which Redis?


Has the data structures we know and love.
Makes an excellent cache, database, or session store.

Not so good at JSON.
Searching can be difficult.
Clustering requires a fair bit of work.

# OSS Redis is Extensible

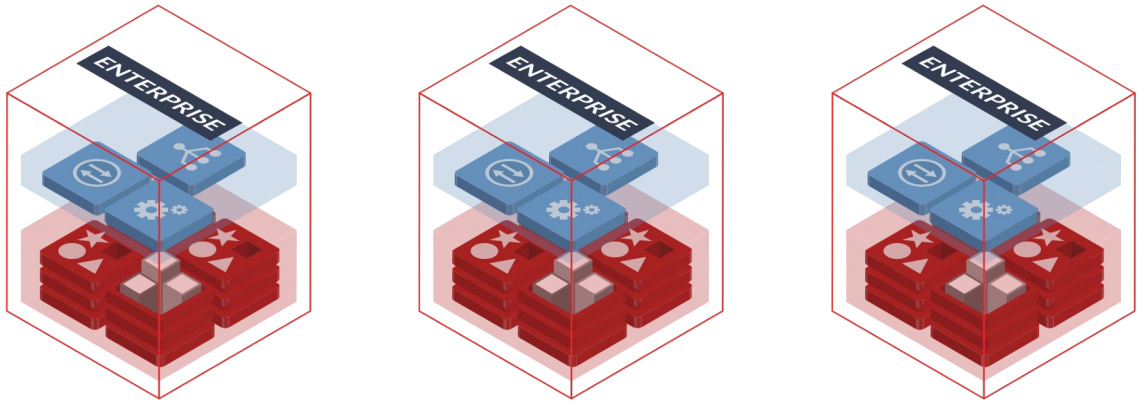


# We've Implemented a Few

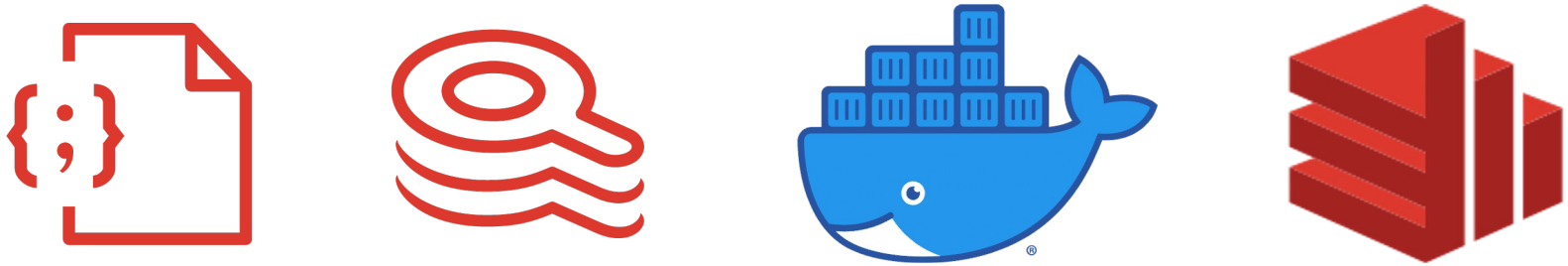

redis stack

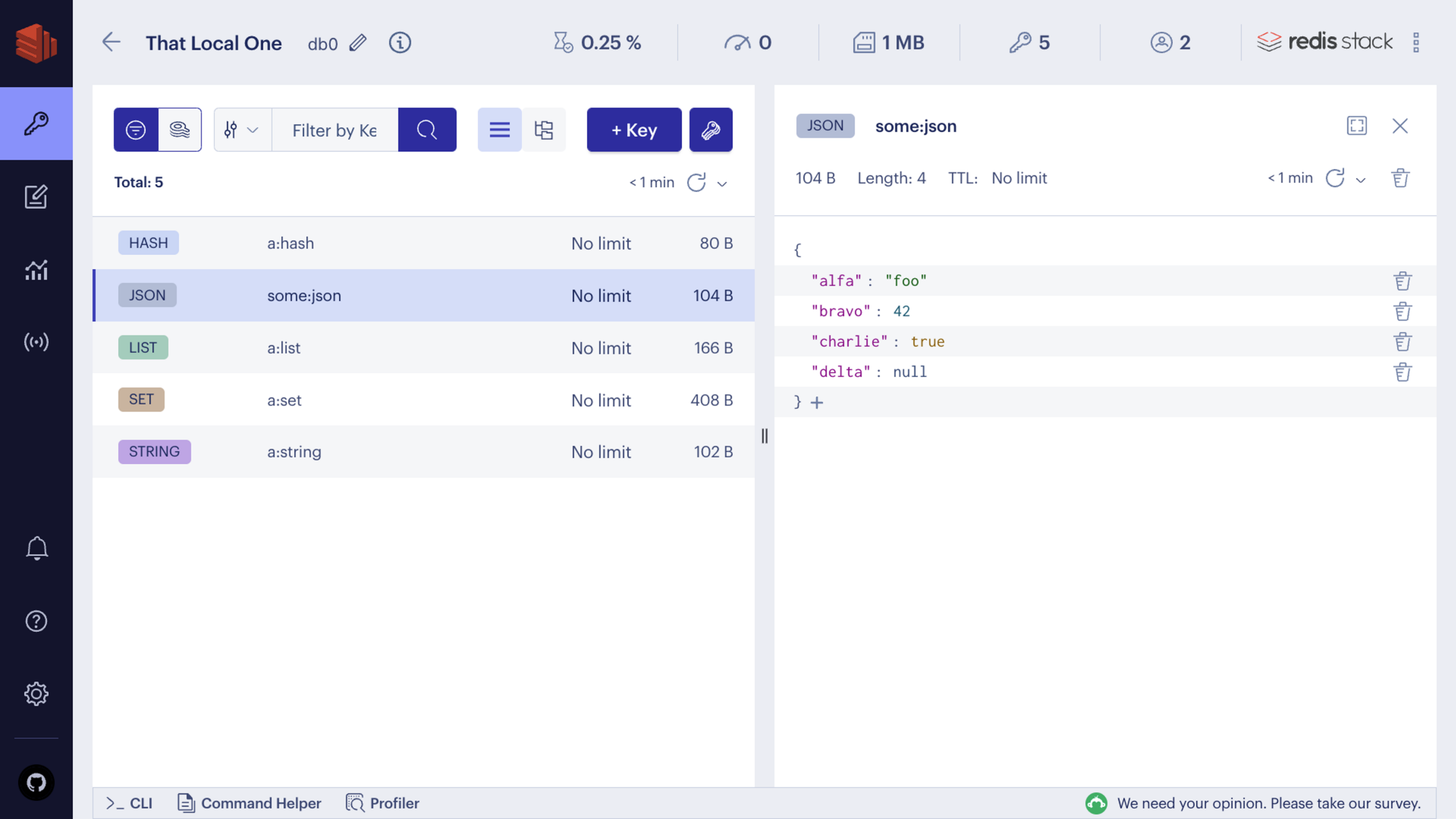
# Which Redis?

Includes all the modules.
Takes care of the clustering and high availability for you.
Easy to use proxy hides the clustering details from us developers.
On Prem or in the Cloud.

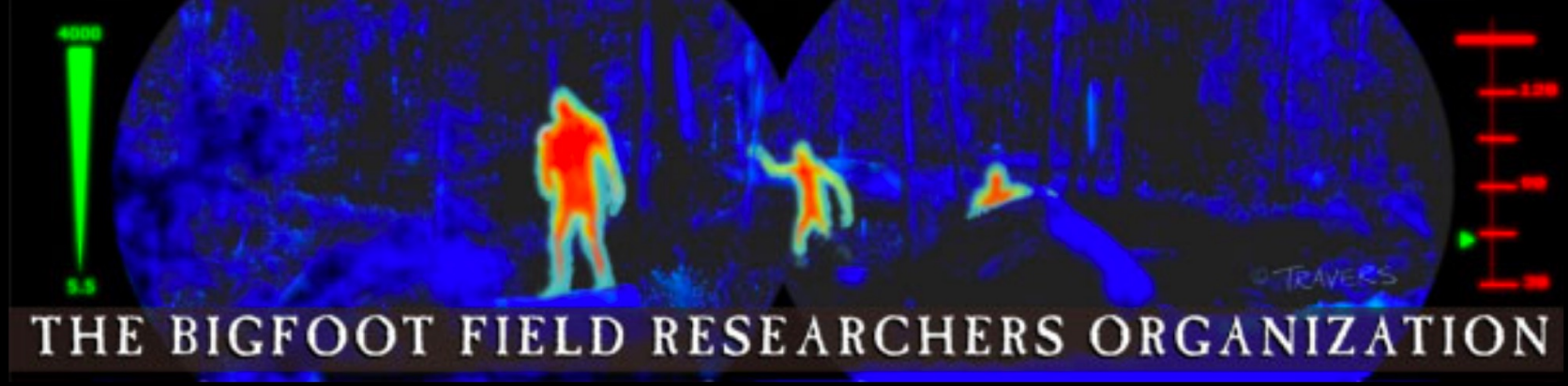


# Which Redis?

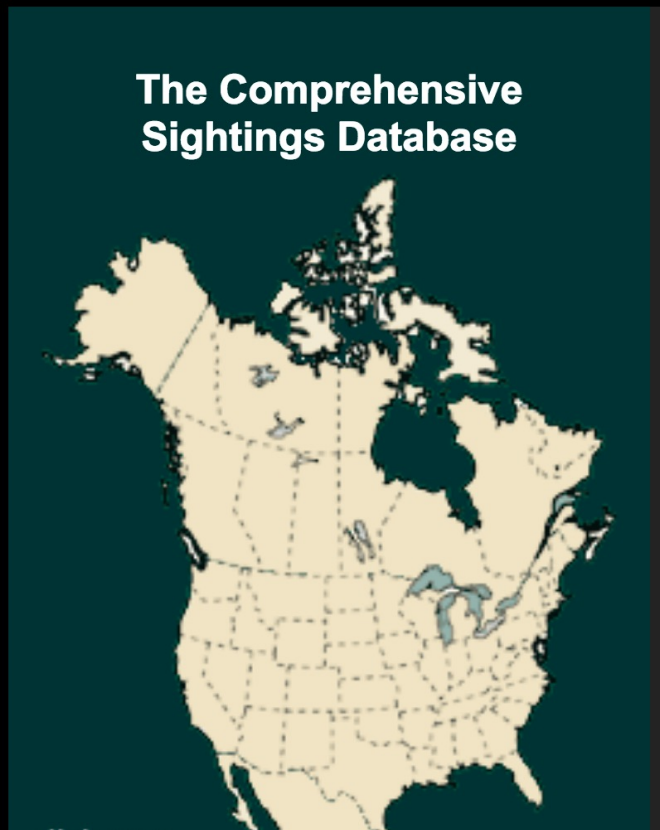

That Local One
db0
0.25 %
0
1 MB
5
2
redis stack
Filter by Ke
+ Key
Total: 5
< 1 min
HASH a:hash No limit 80 B
JSON some:json No limit 104 B
LIST a:list No limit 166 B
SET a:set No limit 408 B
STRING a:string No limit 102 B
JSON some:json
104 B Length: 4 TTL: No limit
< 1 min
{
"alfa" : "foo"
"bravo" : 42
"charlie" : true
"delta" : null
} +
CLI
Command Helper
Profiler
We need your opinion. Please take our survey.

# Where Are We Gonna Get Bigfoot Data?




**Founded in 1995 -- The only scientific research organization exploring the bigfoot/sasquatch mystery.**

**Contact us at ContactUs@BFRO.NET or Phone (408) 634-BFRO [408-634-2376]**





## Bigfoot Town Hall

### Meetings in USA & Canada

**Tim Renner**
@timothyrenner

Following

| FOLLOWERS | FOLLOWING | LIBRARY |
|---|---|---|
| 443 | 8 | 5 |

I do geo data science at HomeAway. Autistic. Obsessed with weird data.

HomeAway

http://timothyrenner.github.io

1–5 of 5

Any ▾ Sort ▾

## Haunted Places

OPEN

Haunted places in the United States pulled from the Shadowlands Haunted Places Index.

Dataset • Updated Jan 28 • Public Domain License

paranormal, geography

4 Comment

## Bigfoot Sightings

OPEN

Full text and geocoded sighting reports from the Bigfoot Field Researchers Organization (BFRO).

Dataset • Updated Dec 2, 2017 • Public Domain License

geography, clustering, sasquatch

59 Comment

## UFO Sightings

OPEN

Full text and geocoded UFO sighting reports from the National UFO Research Center (NUFORC).

Dataset • Updated Oct 25, 2017 • Public Domain License

ufo, paranormal, geography

13 Comment

RECENT COMMENTS

Regarding the UFO sightings - MUFON is probably a good place to start - http://www.mufon.com/ It's possible they have a direct download available.
in Understanding Bigfoot Sightings/Understanding Bigfoot Sightings

Is the timestamp field in the CSV file incomplete? I was messing with this a while back and it seemed okay to me, but I didn't dig in too deep.
in Understanding Bigfoot Sightings/Analysis

Also, there's code for mapping sightings out in this notebook: https://github.com/timothyrenner/bfro_sightings_data/tree/master/notebooks I did a clustering analysis for my blog too: https://timothyrenner.github.io/datascience/2017/06/30/finding-bigfoot.html
in Understanding Bigfoot Sightings/Understanding Bigfoot Sightings

@ninja The JSON file has state and county fields, and it joins to the geolocated reports on the report number. That might be easier than a spatial join against a county/state shapefile.
in Understanding Bigfoot Sightings/Understanding Bigfoot Sightings

I'm glad you like it - I'm happy to join the conversation!
in Bigfoot Sightings/BFRO Sightings data

# Bigfoot Field Researchers Organization Data

| | |
|---|---|
| **ID** | 8086 |
| **Date** | 15-Jan-1958 |
| **Title** | A series of large, human-like footprints are found on a farm near Wayne National Forest |
| **Observed** | While during some yard chores, we noticed a series of tracks going into the hollow on our property. Upon examination… |
| **Classification** | Class B |
| **County** | Noble |
| **State** | Ohio |
| **Latitude** | 39.63382 |
| **Longitude** | -81.40079 |
| **Location Details** | Closest town was Harriettsville. Closest main road is State Route 145. Right on the border of Noble and Washington Counties. |

# Dark Sky Weather Data

| | |
|---|---|
| **High/Mid/Low Temperature** | 34.0 / 30.0 / 26.0 °F |
| **Humidity & Dew Point** | 93% / 31.2°F |
| **Cloud Cover** | 100% |
| **Moon Phase** | 86% |
| **Precipitation Intensity** | 6.7 mm/hr |
| **Precipitation Probability** | 100% |
| **Precipitation Type** | snow |
| **Air Pressure** | 1011.09 millibars |
| **Summary** | Light snow (< 1 in.) starting in the afternoon. |
| **UV Index** | 1 |
| **Visibility** | 2.62 miles |
| **Wind Bearing/Speed** | 344° @ 10.12 mph |

# Getting Started

## Full Instructions on GitHub

github.com/guyroyse/beyond-the-cache*

* This link will never give you up nor let you down.

# Let's Take a Look Around

# Installing Redis Stack

## Redis Stack

```
docker run -d -p 6379:6379 -p 8001:8001
           --name redis-stack redis/redis-stack:latest
```

## Redis Insight

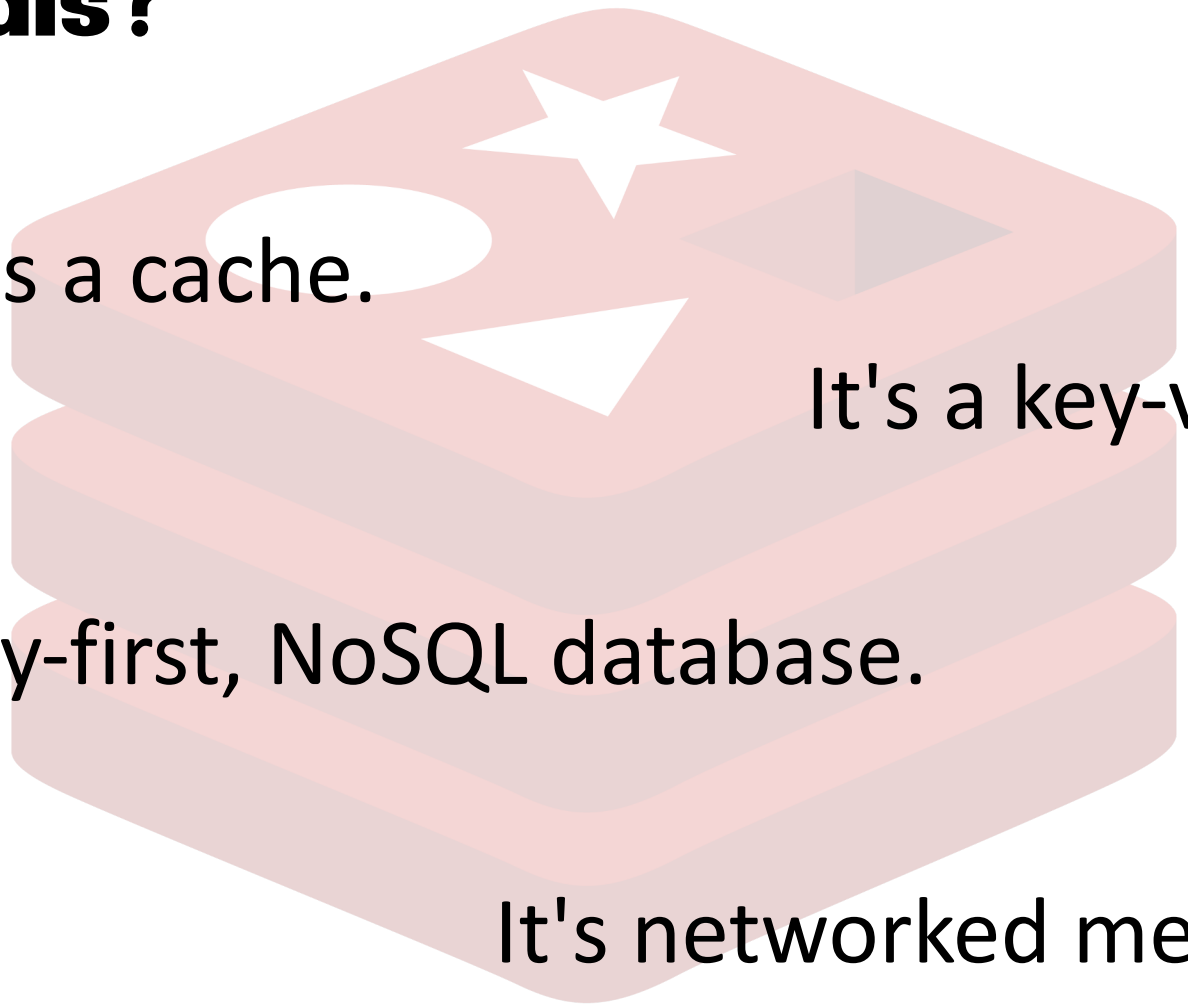
Point your browser at localhost:8001

# Start Your Redis!

01-INSTALLATION.md

# An Introduction to Redis

# What is Redis?



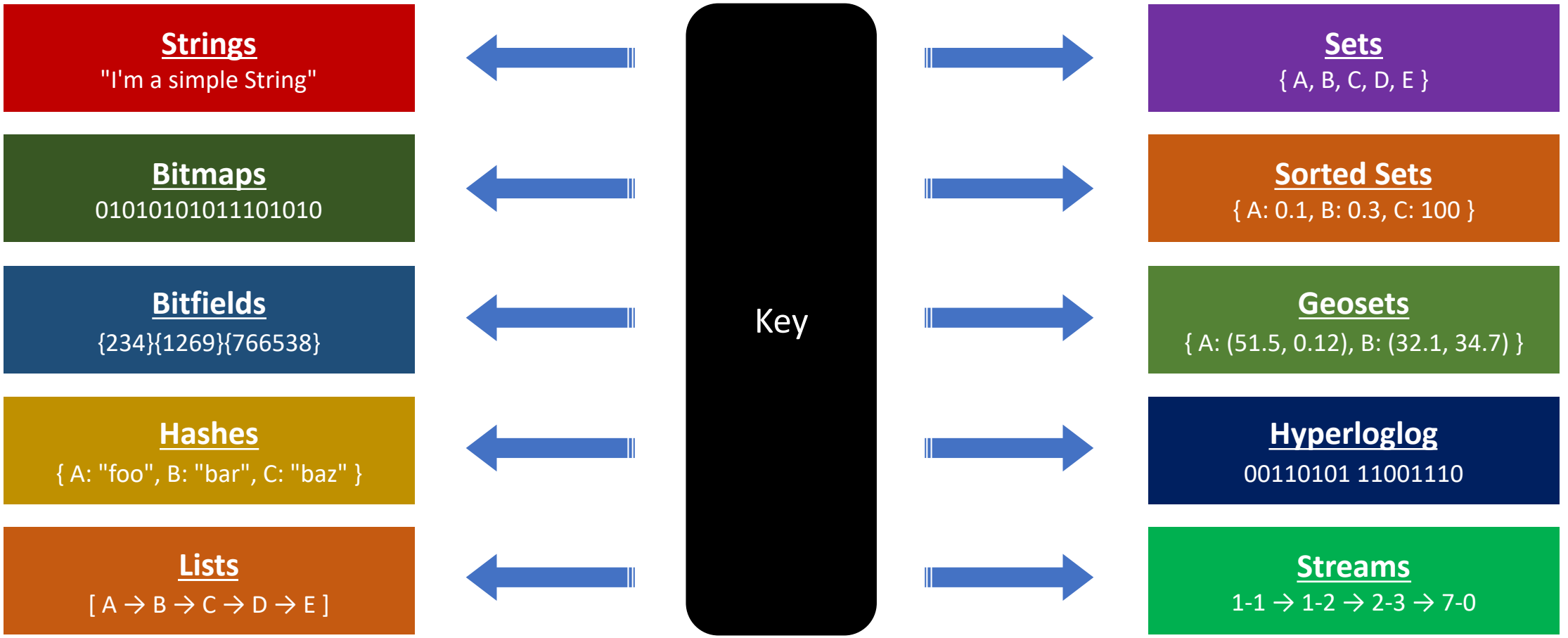
It's a cache.

It's a key-value store.

It's a memory-first, NoSQL database.

It's networked memory.

# A Giant Hash Table



# Redis Commands

```
PING                    PONG
SET foo bar               OK
GET foo                  "bar"
SADD foo bar          (integer) 1
SMEMBERS foo             1)"bar"
HSET foo bar baz      (integer) 1
HGET foo bar             "baz"
LPUSH foo bar         (integer) 1
RPOP foo                 "bar"
DEL foo               (integer) 1
QUIT                      OK
```

redis.io/commands

# Let's Try A Few Commands!

# RESP: The Wire Protocol

SET motd Greetings!

```
*3\r\n
$3\r\n
SET\r\n
$4\r\n
motd\r\n
$10\r\n
Greetings!\r\n
```

OK

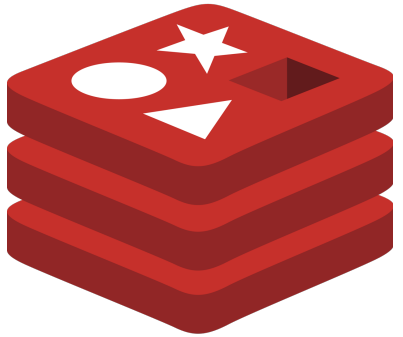
```
+OK\r\n
```

GET motd

```
*2\r\n
$3\r\n
GET\r\n
$4\r\n
motd\r\n
```

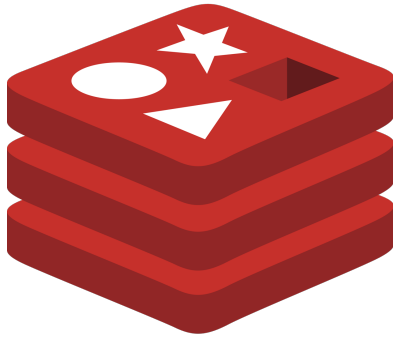
"Greetings!"

```
$10\r\n
Greetings!\r\n
```



# Sending RESP By Hand

```
echo -e \
  '*3\r\n$3\r\nSET\r\n$4\r\nmotd\r\n$10\r\nGreetings!\r\n' \
  | nc localhost 6379
```



```
+OK\r\n
```

```
nc -cv localhost 6379
*2\r\n
$3\r\n
GET\r\n
$4\r\n
motd\r\n
```

```
$10\r\n
Greetings!\r\n
```

# Let's Try RESP By Hand!

# Removing Keys From Redis

## DEL

```
> SET foo 1
OK
> DEL foo
Memory freed
(integer) 1
> GET foo
(nil)
```

## UNLINK

```
> SET foo 1
OK
> UNLINK foo
(integer) 1
> GET foo
(nil)
…
…
…

Memory freed
```

## EXPIRE

```
> SET foo 1
OK
> EXPIRE foo 10
(integer) 1
> GET foo
"1"
About 10 sec
later…
> GET foo
(nil)
```

## Eviction

```
> SET foo 1
OK
> SET bar 2
OK
> GET foo
(nil)




*LRU
```

# Eviction Options

## Eviction

```
> SET foo 1
OK
> SET bar 2
OK
> GET foo
(nil)




*LRU
```

**volatile-lru**: Evict using approximated LRU, only keys with an expire set.
**allkeys-lru**: Evict any key using approximated LRU.
**volatile-lfu**: Evict using approximated LFU, only keys with an expire set.
**allkeys-lfu**: Evict any key using approximated LFU.
**volatile-random**: Remove a random key having an expire set.
**allkeys-random**: Remove a random key, any key.
**volatile-ttl**: Remove the key with the nearest expire time (minor TTL)
**noeviction**: Don't evict anything, just return an error on write operations.


# Persistence

## RDB

**redis.conf**

```
save 3600 1
save 300 100
save 60 10000
```

```
127.0.0.1> BGSAVE
```

## AOF

**redis.conf**

```
appendonly yes
appendfsync always | everysec | no
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 64mb
```

```
127.0.0.1> BGREWRITEAOF
```

# Tell Redis What To Do

02-REDIS-BASICS.md
through
07-REDIS-SETS.md

# Building the Bigfoot Tracker API

# How Are We Building This?

express

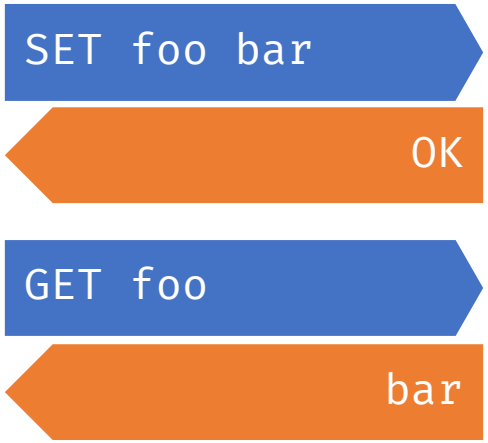
node JS

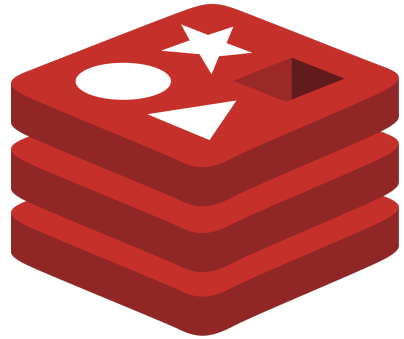
redis stack

# A Proper REST API

| Method | URL | Description |
|---|---|---|
| GET | /sightings | Gets all the things. A collection. |
| POST | /sightings | Add a thing to the collection. **Not** idempotent. |
| GET | /sightings/12345 | Get one of the things in the collection. |
| PUT | /sightings/12345 | Adds or replaces a thing in the collection. Idempotent. |
| DELETE | /sightings/12345 | Removes a thing. |

# Talking to Redis from JavaScript

```
npm install redis
```

github.com/redis/node-redis

# Using Node Redis

```
import { createClient } from 'redis'

const url = 'redis://alice:foobared@awesome.redis.server:6380'
const redis = createClient({ url })
```

# Redis URLs

```
redis://alice:foobared@awesome.redis.server:6380
```

USER PASSWORD HOSTNAME PORT

```
redis://:foobared@awesome.redis.server:6380
redis://awesome.redis.server:6380
redis://awesome.redis.server
```

# Connecting All the Ways

```
const redis = createClient()

const redis = createClient({
  url: 'redis://alice:foobared@awesome.redis.server:6380'
})

const redis = createClient({
  socket: {
    host: 'awesome.redis.server',
    port: 6380
  },
  username: 'alice',
  password: 'foobared'
})
```

| Property | Description |
| --- | --- |
| url | redis[s]://[[username][:password]@][host][:port][/db-number] (see redis and rediss IANA registration for more details) |
| socket | Socket connection properties. Unlisted net.connect properties (and tls.connect) are also supported |
| socket.port | Redis server port |
| socket.host | Redis server hostname |
| socket.family | IP Stack version (one of 4 \| 6 \| 0) |
| socket.path | Path to the UNIX Socket |
| socket.connectTimeout | Connection Timeout (in milliseconds) |
| socket.noDelay | Toggle Nagle's algorithm |
| socket.keepAlive | Toggle keep-alive functionality |
| socket.tls | See explanation and examples below |
| socket.reconnectStrategy | A function containing the Reconnect Strategy logic |
| username | ACL username (see ACL guide) |
| password | ACL password or the old "--requirepass" password |
| name | Connection name (see CLIENT SETNAME) |
| database | Redis database number (see SELECT command) |
| modules | Included Redis Modules |
| scripts | Script definitions (see Lua Scripts) |
| functions | Function definitions (see Functions) |
| commandsQueueMaxLength | Maximum length of the client's internal command queue |
| disableOfflineQueue | Disables offline queuing, see FAQ |
| readonly | Connect in READONLY mode |
| legacyMode | Maintain some backwards compatibility (see the Migration Guide) |
| isolationPoolOptions | See the Isolated Execution Guide |

# Using Node Redis

```
import { createClient } from 'redis'

const url = 'redis://alice:foobared@awesome.redis.server:6380'
const redis = createClient({ url })

await redis.connect()

await redis.set('foo', 'bar')

const foo = await redis.get('foo')
```

# Start Building Your API

08-API-SETUP.md
through
12-NODE-REDIS-HASHES.md

# Pipelining & Transactions

# Pipelining

```
GET foo
        foo
GET bar
        bar
GET baz
        baz
GET qux
        qux
```

```
GET foo
GET bar
GET baz
GET qux
        foo
        bar
        baz
        qux
```

# Pipelining with Node Redis

```
redis.set('foo', 'foo')
redis.set('bar', 'bar')
redis.set('baz', 'baz')
redis.set('qux', 'qux')

redis.set('foo', 'foo')
redis.set('bar', 'bar')
redis.set('baz', 'baz')
await redis.set('qux', 'qux')

const results = await Promise.all([
  redis.get('foo'),
  redis.get('bar'),
  redis.get('baz'),
  redis.get('qux')
])
```

# Transactions

Transactions…

…must happen on the same connection.

…queue a series of commands.

…execute atomically.

# Transaction Commands

**MULTI**: Starts a transaction.

**EXEC**: Executes the queued commands.

**DISCARD**: Discards the queued commands.

**WATCH**: Starts an optimistic lock on a key.

**UNWATCH**: Removed a lock on a key.

# A Simple Example

```
> MULTI
OK

(TX)> SADD bigfoot:sightings:ohio 8086
QUEUED

(TX)> INCR bigfoot:sightings:count
QUEUED

(TX)> HSET bigfoot:sighting:8086 id 8086 state Ohio
QUEUED

(TX)> EXEC
1) (integer) 1
2) (integer) 1
3) (integer) 2
```

# Discarding

```
> MULTI
OK

(TX)> SADD bigfoot:sightings:ohio 8086
QUEUED

(TX)> INCR bigfoot:sightings:count
QUEUED

(TX)> HSET bigfoot:sighting:8086 id 8086 state Ohio
QUEUED

(TX)> DISCARD
OK
```

# Syntactic Errors Fail

```
> MULTI
OK

(TX)> SADD bigfoot:sightings:ohio 8086
QUEUED

(TX)> INCR bigfoot:sightings:count
QUEUED

(TX)> HSTE bigfoot:sighting:8086 id 8086 state Ohio
(error) ERR unknown command `HSTE`, with args beginning
    with: `bigfoot:sighting:8086`, `id`, `8086`,
    `state`, `Ohio`,

(TX)> EXEC
(error) EXECABORT Transaction discarded because of
    previous errors.
```

# Semantic Errors Succeed

```
> MULTI
OK

(TX)> SADD bigfoot:sightings:ohio 8086
QUEUED

(TX)> INCR bigfoot:sightings:ohio
QUEUED

(TX)> EXEC
1) (integer) 1
2) (error) WRONGTYPE Operation against a key holding
    the wrong kind of value
```

# Watching for Changes

```
> WATCH bigfoot:sightings:ohio
OK

> MULTI
OK

(TX)> SADD bigfoot:sightings:ohio 8086
QUEUED

(TX)> HSET bigfoot:sighting:8086 id 8086 state Ohio
QUEUED

(TX)> EXEC
1) (integer) 1
2) (integer) 2
```

# Changes Happen!

```
> WATCH bigfoot:sightings:ohio
OK

> MULTI
OK
```

```
> SADD bigfoot:sightings:ohio 8087
(integer) 1
```

```
(TX)> SADD bigfoot:sightings:ohio 8086
QUEUED

(TX)> INCR bigfoot:sightings:count
QUEUED

(TX)> EXEC
(nil)
```

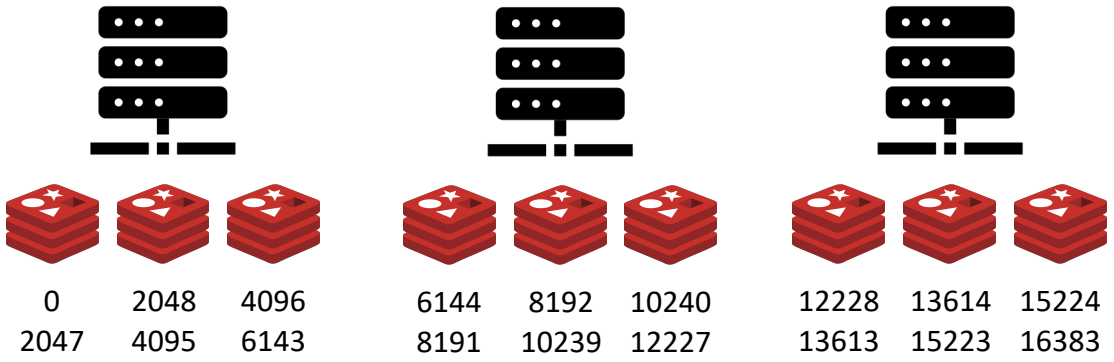
# Watching for Changes

```
> WATCH bigfoot:sightings:ohio
OK

> MULTI
OK

(TX)> SADD bigfoot:sightings:ohio 8086
QUEUED

(TX)> HSET bigfoot:sighting:8086 id 8086 state Ohio
QUEUED

(TX)> EXEC
(nil)
```

# Transaction Demo

# Redis Clusters

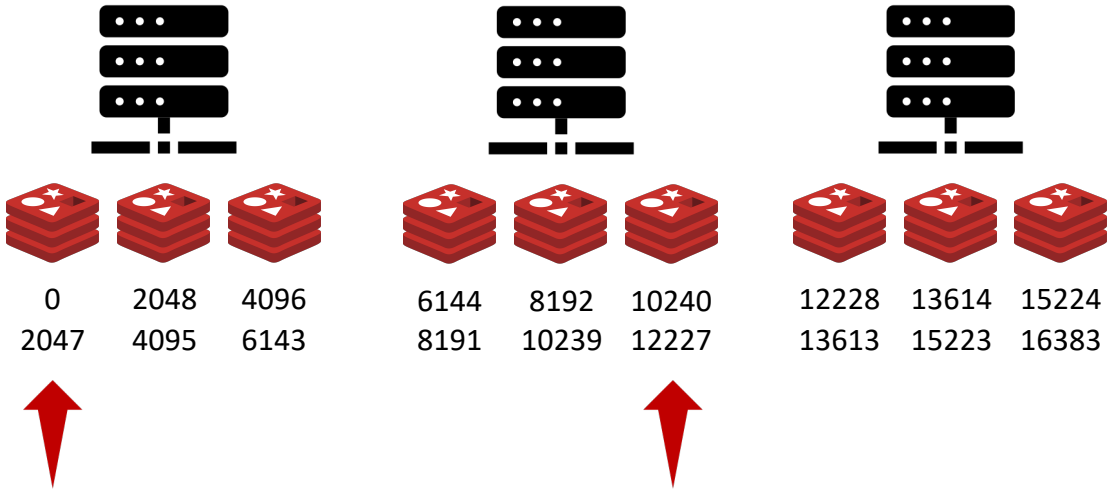
A cluster contains many **nodes.**
A nodes contain many **shards.**
A shard contains many **hash slots.**
There are always **16,384 hash** slots in a cluster.




Transactions cannot span hash slots.

# Computing Hash Slots

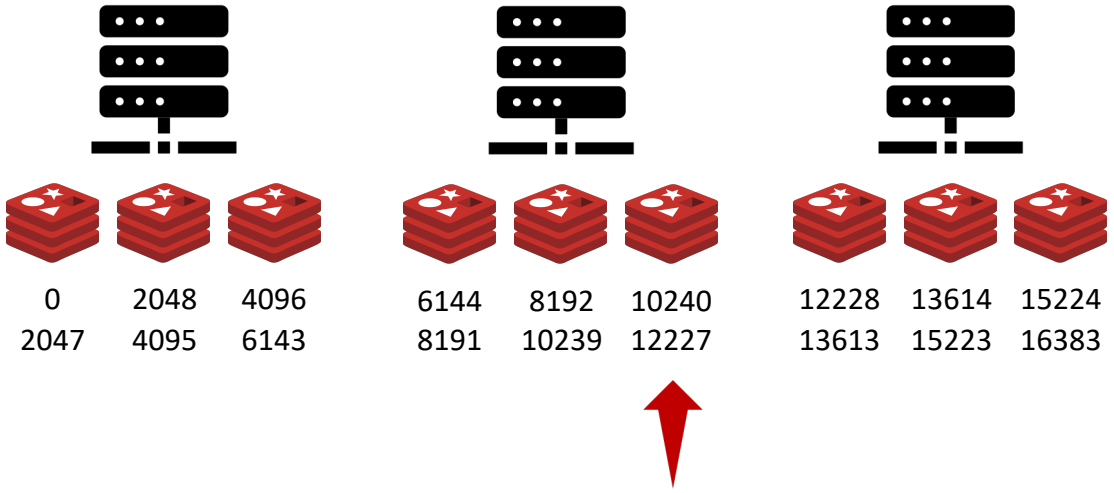
```
bigfoot:sightings:ohio
bigfoot:sightings:count

CRC16('bigfoot:sightings:ohio') mod 16384 = 12194
CRC16('bigfoot:sightings:count') mod 16384 = 1827
```



# Keeping Keys Together

```
{bigfoot:sightings}:ohio
{bigfoot:sightings}:count
```

```
CRC16('bigfoot:sightings') mod 16384 = 11876
```

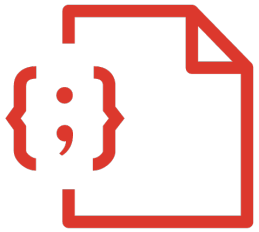


# Add Transactions

13-TRANSACTIONS.md

# RedisJSON & RediSearch

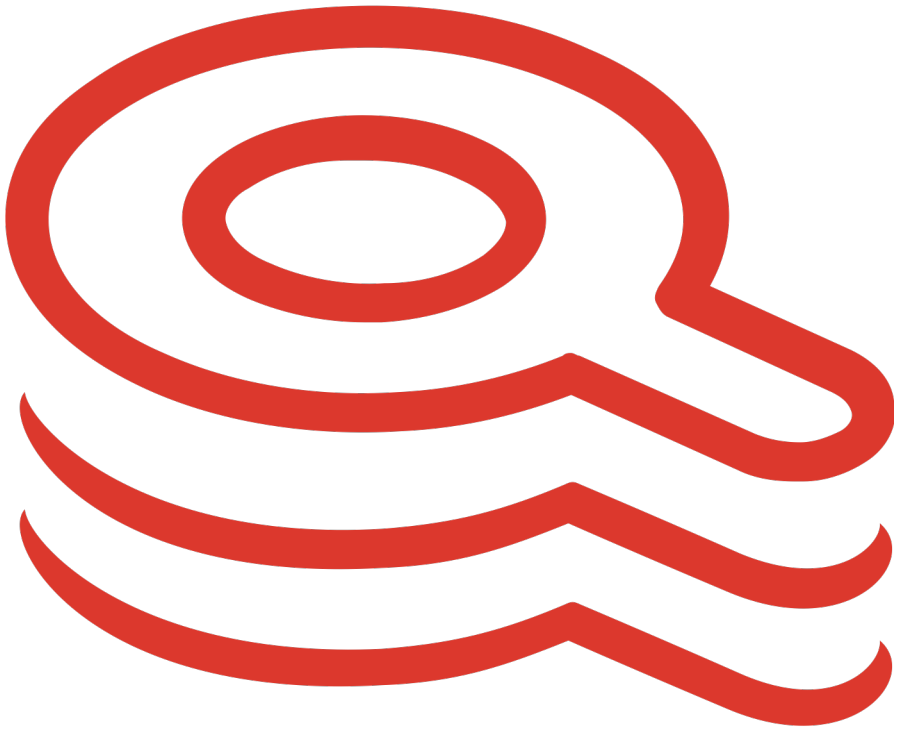
# Using RedisJSON




```
> JSON.SET foo $ '{ "bar":"bar", "baz":42, "qux":true }'
OK

> JSON.GET foo
"{\"bar\":\"bar\",\"baz\":42,\"qux\":true}"

> JSON.GET foo $
"[{\"bar\":\"bar\",\"baz\":42,\"qux\":true}]"

> JSON.GET foo $.bar
"[\"bar\"]"

> JSON.GET foo $.bar $.baz $.qux
"{\"$.bar\":[\"bar\"],\"$.baz\":[42],\"$.qux\":[true]}"
```

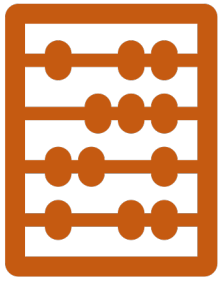
# RedisJSON Demo

# RediSearch



Indexing

Full-text search



Aggregation

# Creating Indices with RediSearch

```
FT.CREATE bigfoot:sighting:index
  ON JSON
  PREFIX 1 bigfoot:sighting:
  SCHEMA
    $.observed AS observed TEXT
    $.state    AS state    TAG
    $.humidity AS humidity NUMERIC
    $.location AS location GEO
```

# Find All the Bigfoots

```
FT.SEARCH bigfoot:sighting:index "*"
  LIMIT 0 3
  RETURN 2 id state
```

index

query

starting index

number of results

fields to return

```
1) (integer) 4586
2) "bigfoot:sighting:27167"
3) 1) "id"
   2) "27167"
   3) "state"
   4) "Mississippi"
4) "bigfoot:sighting:20002"
5) 1) "id"
   2) "20002"
   3) "state"
   4) "Michigan"
6) "bigfoot:sighting:28711"
7) 1) "id"
   2) "28711"
   3) "state"
   4) "Minnesota"
```

redis

# Four Ways to Find Bigfoot

Text

Tag


Numeric

Geo

# Texting Bigfoot

```
> FT.CREATE bigfoot:sighting:index
  ON hash PREFIX 1 bigfoot:sighting:
  SCHEMA
    title    TEXT
    observed TEXT
```

```
> FT.SEARCH bigfoot:sighting:index
      "creek"
      "creek river"
      "creek ~river"
      "creek | river"
      "(creek | river) stream"
      "(creek | river) -stream"
      "@title:creek @observed:river"
```


# Tagging Bigfoot

```
> FT.CREATE bigfoot:sighting:index
  ON hash PREFIX 1 bigfoot:sighting:
  SCHEMA
    state          TAG
    classification TAG
```

```
> FT.SEARCH bigfoot:sighting:index
      "@state:{ Ohio }"
      "@state:{ Ohio | Kentucky }"
      "@state:{ West Virginia }"
      "@classification:{ Class\\ A }"
      "@classification:{ Class\\ A }
       @classification:{ Class\\ B }"
```

# Numbering Bigfoot

```
> FT.CREATE bigfoot:sighting:index
  ON hash PREFIX 1 bigfoot:sighting:
  SCHEMA
    temperature_high NUMERIC
    temperature_low  NUMERIC
```

```
> FT.SEARCH bigfoot:sighting:index
      "@temperature_high:[ 60 75 ]"
      "@temperature_high:[ 60 +inf ]"
      "@temperature_high:[ -inf 75 ]"
      "@temperature_high:[ -inf +inf ]"
      "@temperature_high:[ (60 (75 ]"
```

# Locating Bigfoot

```
> FT.CREATE bigfoot:sighting:index
  ON hash PREFIX 1 bigfoot:sighting:
  SCHEMA
    location GEO
```

longitude

```
-84.5120,39.1031
```

latitude

# Locating Bigfoot

```
> FT.CREATE bigfoot:sighting:index
  ON hash PREFIX 1 bigfoot:sighting:
  SCHEMA
    location GEO
```

```
> FT.SEARCH bigfoot:sighting:index
     "@location:[ -84.5120 39.1031 50 mi ]"
     "@location:[ -84.5120 39.1031 100 ft ]"
     "@location:[ -84.5120 39.1031 25 m ]"
     "@location:[ -84.5120 39.1031 10 km ]"
```

redis

# RediSearch Demo

# Some SQL Equivalents

| SQL | RediSearch |
|---|---|
| `WHERE x='foo' AND y='bar'` | `@x:foo @y:bar` |
| `WHERE x='foo' AND y!='bar'` | `@x:foo -@y:bar` |
| `WHERE x='foo' OR y='bar'` | `(@x:foo)\|(@y:bar)` |
| `WHERE x IN ('foo', 'bar','hello world')` | `@x:(foo\|bar\|"hello world")` |
| `WHERE y='foo' AND x NOT IN ('foo','bar')` | `@y:foo (-@x:foo) (-@x:bar)` |
| `WHERE x NOT IN ('foo','bar')` | `-@x:(foo\|bar)` |
| `WHERE num BETWEEN 10 AND 20` | `@num:[10 20]` |
| `WHERE num >= 10` | `@num:[10 +inf]` |
| `WHERE num > 10` | `@num:[(10 +inf]` |
| `WHERE num < 10` | `@num:[-inf (10]` |
| `WHERE num <= 10` | `@num:[-inf 10]` |
| `WHERE num < 10 OR num > 20` | `@num:[-inf (10] \| @num:[(20 +inf]` |
| `WHERE name LIKE 'john%'` | `@name:john*` |

# Try JSON & Search

14-REDISJSON.md
through
19-NODE-REDIS-SEARCH.md

# Introducing Redis OM

# What Is Redis OM?

A layer of abstraction over RediSearch & RedisJSON

Built on top of Node Redis

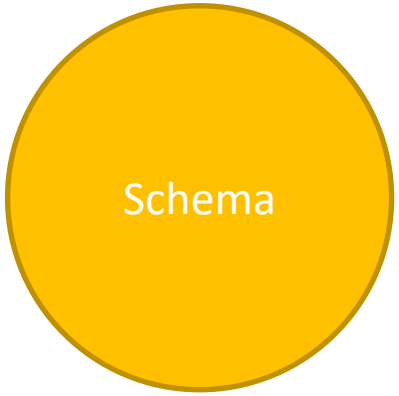
Supports index creation using `Schema`

Enables basic CRUD Operations using `Repository`

Provides a fluent search interface using `Search`

redis ōM

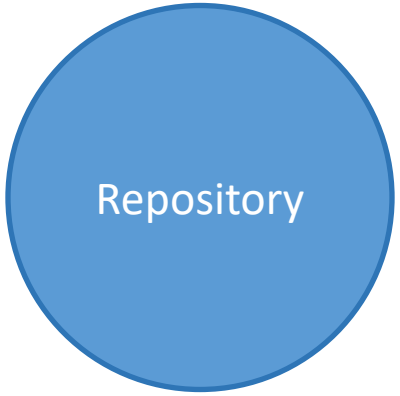
# Schemas & Repositories




Defines the structure of your data

Used to build indices for RediSearch

Used to convert JavaScript types to and from Redis types




Provides CRUD operations for your data

Creates indices for search

Initiates search using RediSearch

```
npm install redis-om
```

redisOM

# Creating a Schema

```
const schema = new Schema('sighting', {

  reportId:    { type: 'string' },
  tags:        { type: 'string[]' },
  verified:    { type: 'boolean' },

  description: { type: 'text' },

  witnesses:   { type: 'number' },
  date:        { type: 'date' },

  location:    { type: 'point' }

})
```

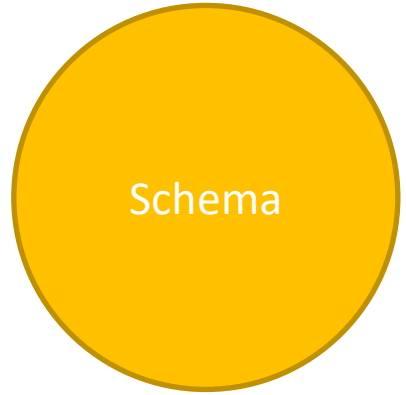
prefixes keys in Redis

maps to TAG

maps to TEXT

maps to NUMERIC

maps to GEO




redisOM

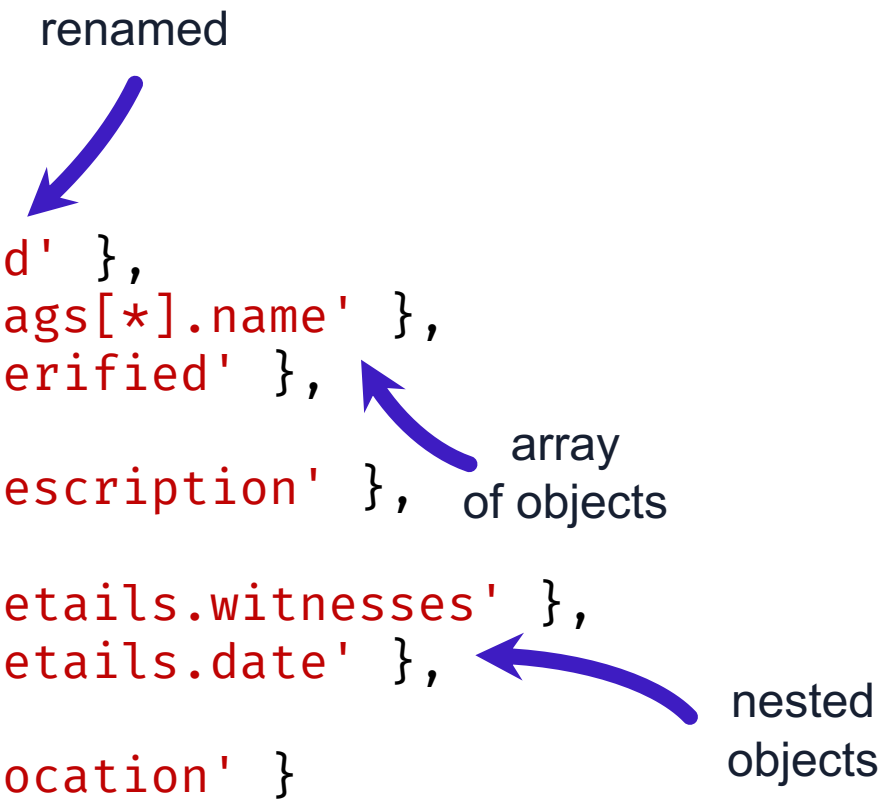
# JSONPath is Implied

```
const schema = new Schema('sighting', {

  reportId:    { type: 'string',   path: '$.reportId' },
  tags:        { type: 'string[]', path: '$.tags[*]' },
  verified:    { type: 'boolean',  path: '$.verified' },

  description: { type: 'text',     path: '$.description' },

  witnesses:   { type: 'number',   path: '$.witnesses' },
  date:        { type: 'date',     path: '$.date' },

  location:    { type: 'point',    path: '$.location' }

})
```


redisOM

# Can Be Renamed and Nested

```
const schema = new Schema('sighting', {

  reportId:    { type: 'string',   path: '$.id' },
  tags:        { type: 'string[]', path: '$.tags[*].name' },
  verified:    { type: 'boolean',  path: '$.verified' },

  description: { type: 'text',     path: '$.description' },

  witnesses:   { type: 'number',   path: '$.details.witnesses' },
  date:        { type: 'date',     path: '$.details.date' },

  location:    { type: 'point',    path: '$.location' }

})
```




redisOM

# Missing Fields

Missing fields are saved but not indexed and converted to the extent possible

Strings are Strings

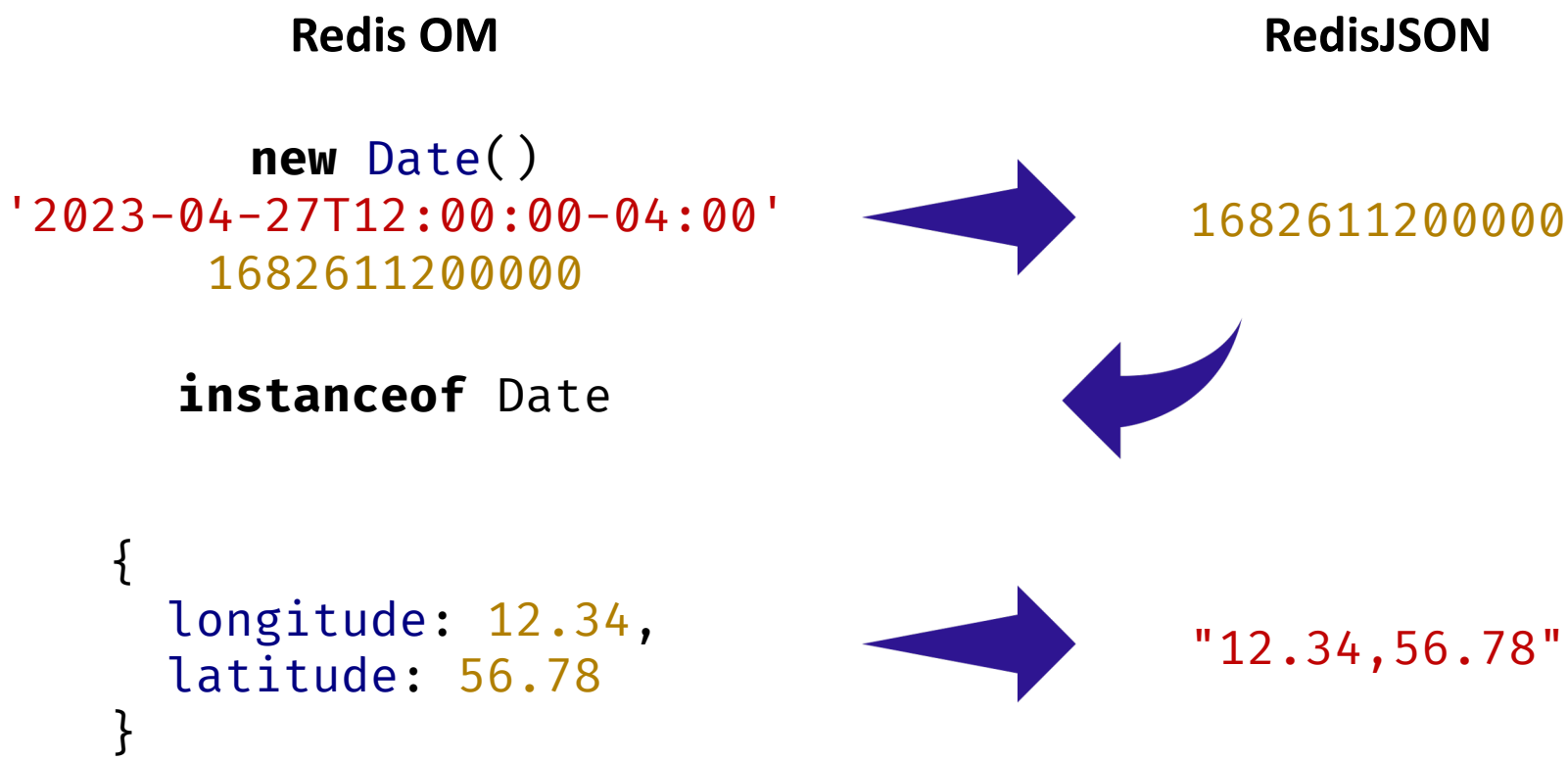
Numbers are Numbers

Booleans are Booleans

Dates & Points are special

redisōM

# If A Date or Point is Defined




redisOM

# If A Point or Date is Missing

**Redis OM**

```
new Date()
'2023-04-27T12:00:00-04:00'
1682611200000

{
  longitude: 12.34,
  latitude: 56.78
}
```

**RedisJSON**

```
1682611200000
'2023-04-27T12:00:00-04:00'
1682611200000

{
  longitude: 12.34,
  latitude: 56.78
}
```

redisōM

# Creating Repositories

```
const repository = new Repository(schema, redis)

let sighting = {
  reportId: "12345",
  tags: [ "creek", "woodlands" ],
  verified: false,
  description: "I saw Bigfoot in the woods",
  witnesses : 2,
  date: '2023-04-27',
  location: {
    longitude: 12.34,
    latitude: 56.78
  }
}

sighting = await repository.save(sighting)
```

redisōM

# What is an Entity?

```
sighting.reportId      // "12345"
sighting.tags          // [ "creek", "woodlands" ]
sighting.verified      // false
sighting.description   // "I saw Bigfoot in the woods"
sighting.witnesses     // 2
sighting.location      // { longitude: 12.34, latitude: 56.78 }

sighting.date        // instanceof Date

sighting[EntityId]   // "01FJYWEYRHYFT8YTEGQBABJ43J"
```

## Can I Use My Own EntityID?

```
sighting = await repository.save("12345", sighting)
```

redisOM

# Calling Save Again

```
sighting[EntityId]  // "12345"

sighting.tags.push('spring')
sighting.description = "I saw Bigfoot at Walmart buy size 17s"
sighting.verified = true

sighting = await repository.save(sighting)
```

Saving an Entity with an EntityID does an upsert

redisOM

# Fetching & Removing

```
const sighting = await repository.fetch("12345")

await repository.remove("12345")
```

redisOM

# Indexing & Finding All The Things

```
await repository.createIndex()
```

creates (or recreates) the index

```
redis.local:6389> FT.CREATE bigfoot:sighting:index
                     ON JSON
                     PREFIX 1 bigfoot:sighting:
                     SCHEMA
                       ...
```

creates a Search instance

```
const sightings = await repository.search().return.all()
```

```
redis.local:6379> FT.SEARCH bigfoot:sighting:index "*"
```

triggers the search

redisOM

# Finding Some of the Things

```
const sightinfs = await repository.search()
  .where('reportId').equals('12345')          // string
    .and('witnesses').is.greaterThan(1)       // number
    .and('verified').is.false()               // boolean
    .and('date').is.after('2023-04-27')       // date
    .and('tags').contains('woodlands')        // string[]
    .and('description').matches('creek')      // full-text search
    .and('location').inRadius(                // point
       circle => circle.origin(12.34, 56.78).radius(10).miles)
         .return.all()
```

Full documentation at github.com/redis/redis-om-node/.

redisOM

# Try Redis OM

20-REDIS-OM-BASICS.md
through
21-REDIS-OM-SEARCHING.md

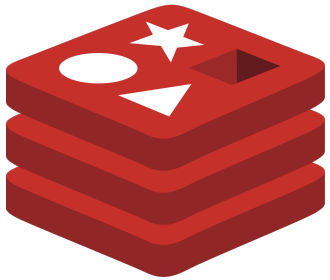
# Check Out Our Stuff

## Redis Discord Server

https://discord.gg/redis

## Redis University

https://university.redis.com/



## Redis Cloud

https://redis.com/try-free/

# Guy Royse

Developer Advocate

redis

@guyroyse

github.com/guyroyse

guy.dev

# Thanks for Attending!


redis