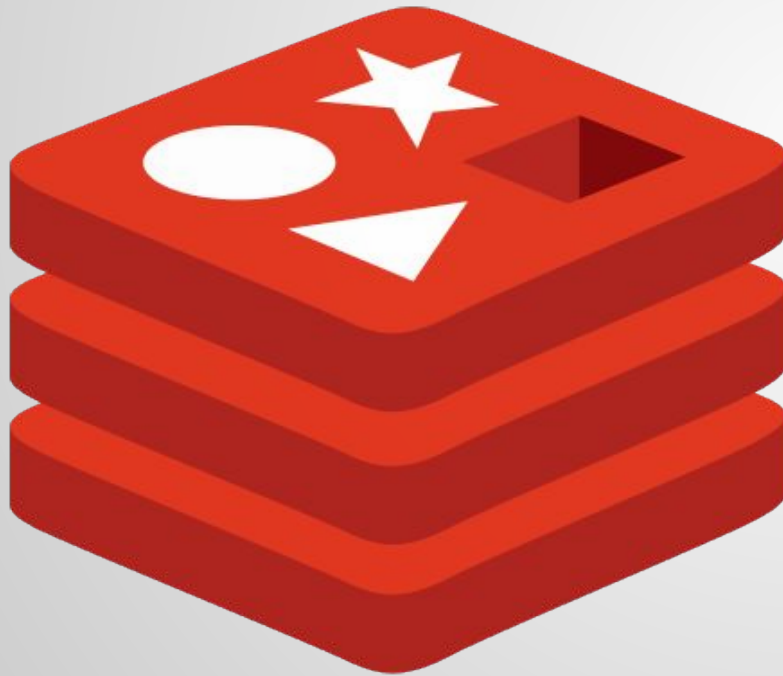


UPenn & Redis Workshop

Beats by Redis



Savannah Norem
&
Justin Castilla



Savannah Norem

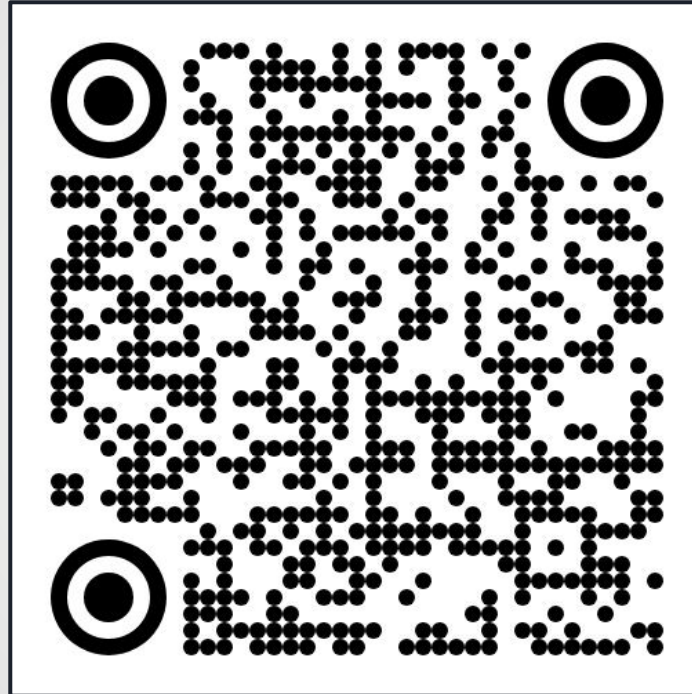
Developer Advocate, Redis



Justin Castilla

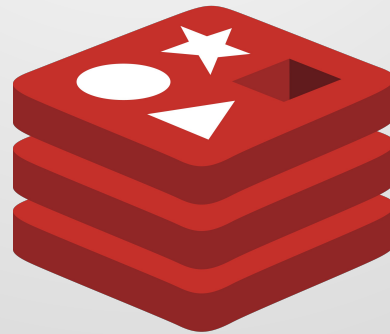
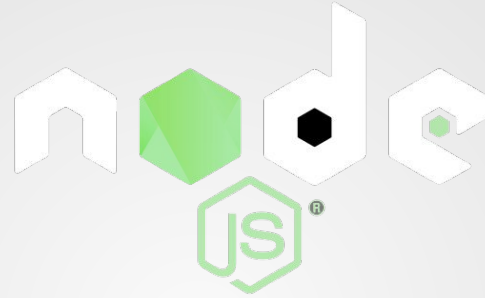
Developer Advocate, Redis

Workshop Repository



github.com/redislabs-training/beats-by-redis-edu

What Are We Doing Here?



Redis Capabilities We're Using



Streams



JSON



Search



Sorted Sets



Time Series

redis^{OM}

Which Redis To Use?

An Overview of Redis Installation Options

Which Redis?

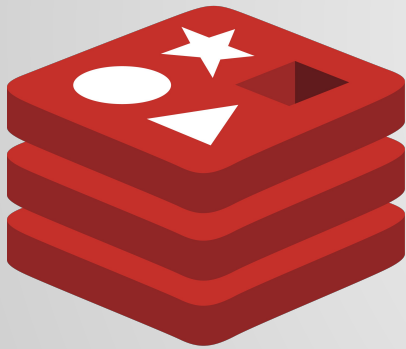
OSS Redis

- 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.



Which Redis?

OSS Redis is Extensible

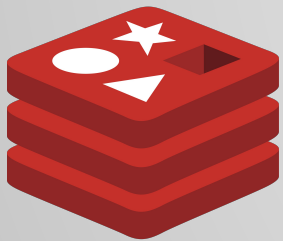


+



Which Redis?

We've Implemented a Few



+



JSON



Search



TimeSeries



Graph

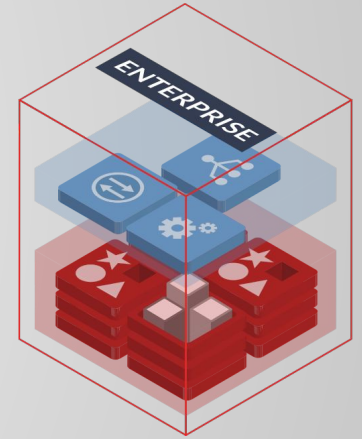
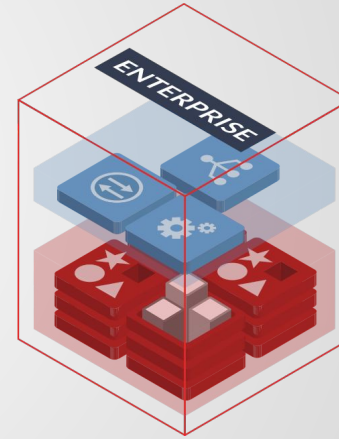
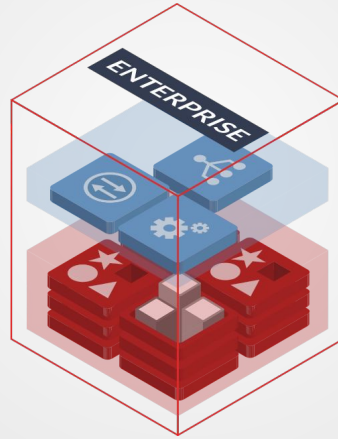


Bloom

Which Redis?

Redis Enterprise

- Includes all the Redis Stack capabilities.
- Takes care of clustering and high availability.
- Easy to use proxy hides the clustering details from us developers.
- On Prem (on your computers) or in the Cloud (on our computers.)



Which Redis?

Docker





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

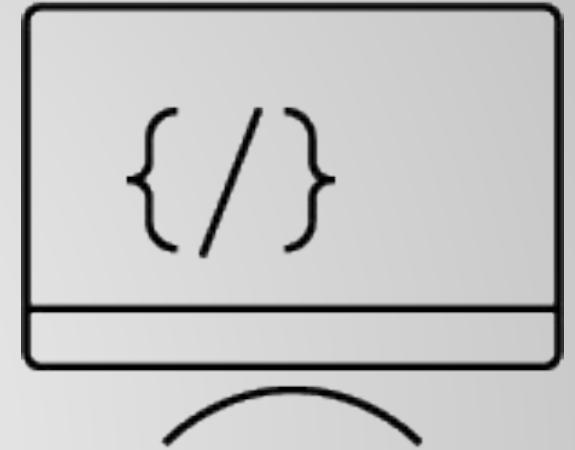
< 1 min

```
{
  "alfa" : "foo"
  "bravo" : 42
  "charlie" : true
  "delta" : null
} +
```



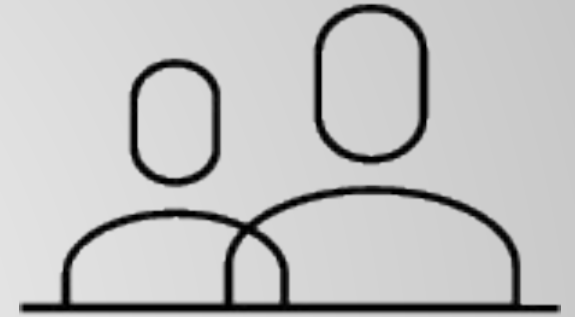
Demo

RedisInsight



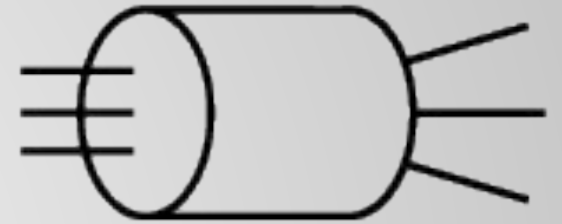
Try Out Redis Insight

Follow the instructions in
01-INSTALLATION.md



Capturing & Querying Events

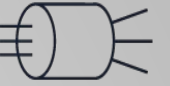
An Overview of Redis Streams



Event Streams

Introduction

- Series of events containing data and indexed by an ID.
- Chronological by nature.
- Used to capture, monitor, and replay a series of events.
- Makes a decent queue.
- Makes a great microservice bus.



1655410429872-0



1655410429872-0



1655410429867-0



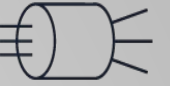
1655410429863-1



1655410429863-0

Event Streams

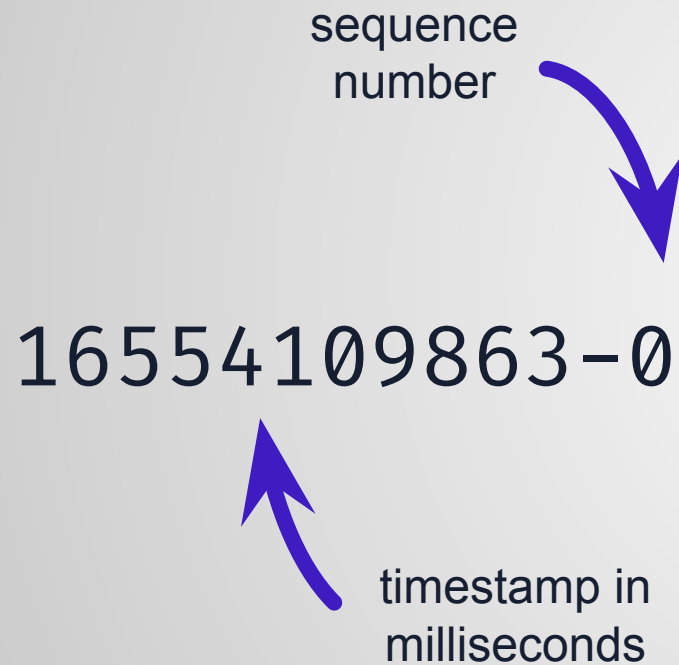
IDs & Data with Redis Streams



sequence
number

16554109863-0

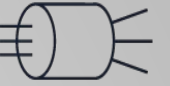
timestamp in
milliseconds



Key	Value
toAccount	5558675309
toAccountName	Always Watching H.O.A.
fromAccount	1580783161
fromAccountDescription	Justin Castilla, Esq.
amount	\$167.34
transactionDate	2023-04-27
transactionType	Authorized

Event Streams

Adding Events



stream key

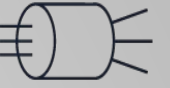
event id

```
redis.local:6379> XADD transactions * amount 167.34  
fromAccount 5558675309 toAccount 1580783161  
transactionDate 2023-04-27 transactionType Authorized
```

event data

Event Streams

Reading Events



stream
key

starting
event id

```
redis.local:6379> XREAD STREAMS transactions 16554109863-0
```

```
redis.local:6379> XREAD COUNT 5 STREAMS transactions 16554109863-0
```

limit the
return

Event Streams

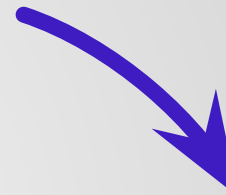
Awaiting Events



wait
1000ms



starting
event id



```
redis.local:6379> XREAD BLOCK 1000 STREAMS transactions 16554109863-0
```

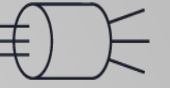
```
redis.local:6379> XREAD BLOCK 1000 STREAMS transactions $
```



new stuff
only

Event Streams

Other Commands



```
redis.local:6379> XRANGE transactions 0-0 16554109863-0
```

```
redis.local:6379> XDEL transactions 16554109863-0 16554109121-0 16554109121-1
```

```
redis.local:6379> XLEN transactions
```

```
redis.local:6379> XTRIM transactions MAXLEN 1000
```

```
redis.local:6379> XTRIM transactions MINID 16554109863-0
```

```
redis.local:6379> XTRIM transactions MAXLEN ~ 1000
```

```
redis.local:6379> XTRIM transactions MINID ~ 16554109863-0
```

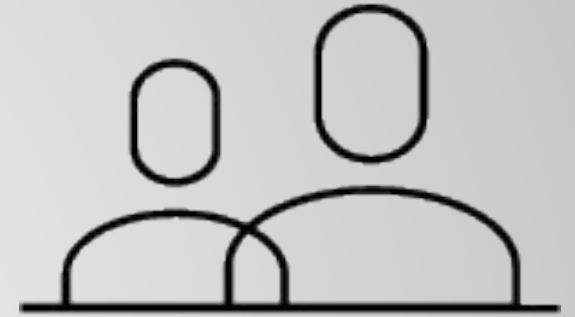
Demo

Redis Streams



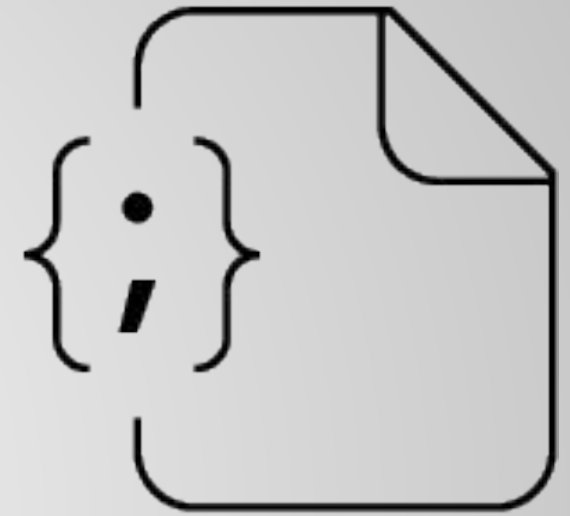
Try Out Streams

Follow the instructions in
02-STREAMS.md



Storing JSON the Good Way

An Overview of RedisJSON



RedisJSON

Introduction

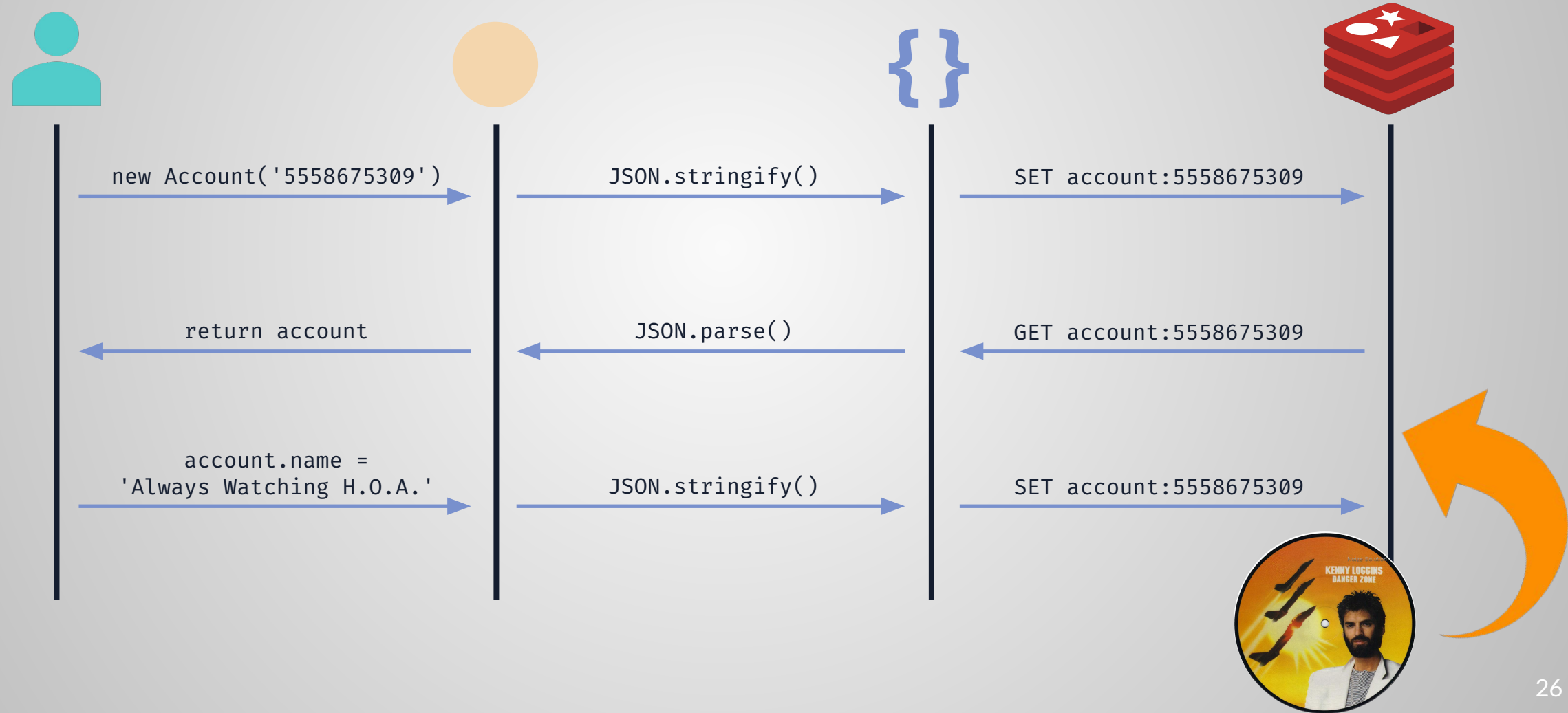
- Store JSON in Redis *as* JSON.
- Query all or part of a document using JSONPath.
- Manipulate all or part of a document.
- All operations are atomic.

```
{
  "type": "Authorized",
  "date": "2023-04-27",
  "amount": 167.34,
  "to": {
    "account": "5558675309",
    "name": "Always Watching H.O.A."
  },
  "from": {
    "account": "1580783161",
    "name": "Justin Castilla, Esq."
  }
}
```



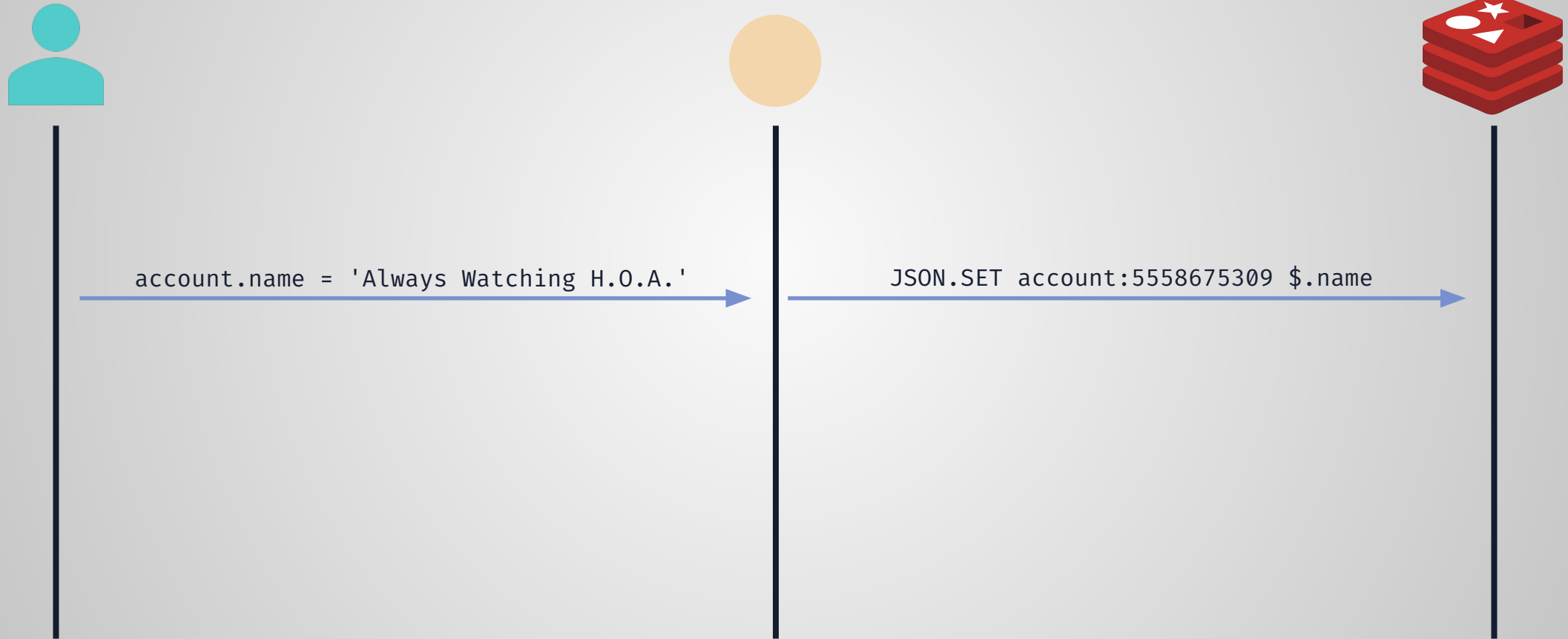
RedisJSON

A Common Scenario



RedisJSON

A Common Scenario



RedisJSON

Writing JSON



JSON
key

JSONPath

JSON
string

```
redis.local:6379> JSON.SET account:5558675309 $ '{"account":"5558675309"}'  
redis.local:6379> JSON.SET account:5558675309 $.balance 167.34  
redis.local:6379> JSON.SET account:5558675309 $.name '"Always Watching H.O.A.'"
```



RedisJSON

Reading JSON



```
redis.local:6379> JSON.GET account:5558675309
```

```
{  
  "account": "5558675309",  
  "balance": 167.34,  
  "name": "Always Watching H.O.A."  
}
```

```
redis.local:6379> JSON.GET account:5558675309 $
```

```
[  
  {  
    "account": "5558675309",  
    "balance": 167.34,  
    "name": "Always Watching H.O.A."  
  }  
]
```

RedisJSON

Reading Using JSONPath



```
redis.local:6379> JSON.GET account:5558675309 $.name
```

```
[  
  "Always Watching H.O.A."  
]
```

```
redis.local:6379> JSON.GET account:5558675309 $.*
```

```
[  
  "5558675309",  
  167.34,  
  "Always Watching H.O.A."  
]
```



RedisJSON

Reading Using Multiple JSONPaths

```
redis.local:6379> JSON.GET account:5558675309 $ $.name $.account
```

```
{
  "$": [
    {
      "account": "5558675309",
      "balance": 167.34,
      "name": "Always Watching H.O.A."
    }
  ],
  "$.name": [ "Always Watching H.O.A." ],
  "$.account": [ "5558675309" ]
}
```

RedisJSON

Removing JSON



removes
document



```
redis.local:6379> JSON.DEL account:5558675309
```

```
redis.local:6379> JSON.DEL account:5558675309 $
```

removes
root node



```
redis.local:6379> JSON.DEL account:5558675309 $.name
```

removes
node



RedisJSON

Other Commands



```
redis.local:6379> JSON.TYPE account:5558675309 $.balance
redis.local:6379> JSON.TOGGLE account:5558675309 $.overdrawn
redis.local:6379> JSON.ARRAPPEND account:5558675309 $.approvals JC GR TH
redis.local:6379> JSON.NUMINCRBY account:5558675309 $.balance 100
redis.local:6379> JSON.STRLEN account:5558675309 $.name
redis.local:6379> JSON.OBJKEYS account:5558675309 $
redis.local:6379> JSON.MGET account:5558675309 account:6365553226 $.balance
```

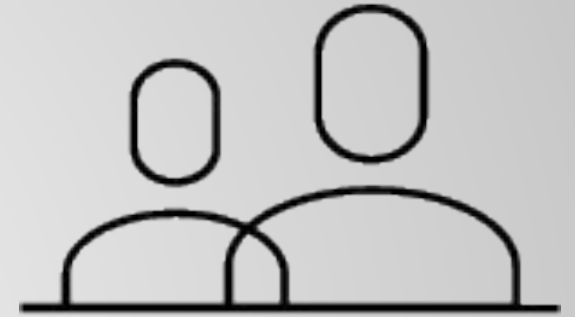
Demo

RedisJSON



Try Out RedisJSON

Follow the instructions in
03-REDIS-JSON.md



Searching & Querying

An Overview of RediSearch



Searching the Bad Way

Brute Force



```
redis.local:6379> JSON.GET account:5558675309
```

```
redis.local:6379> KEYS account:*
```

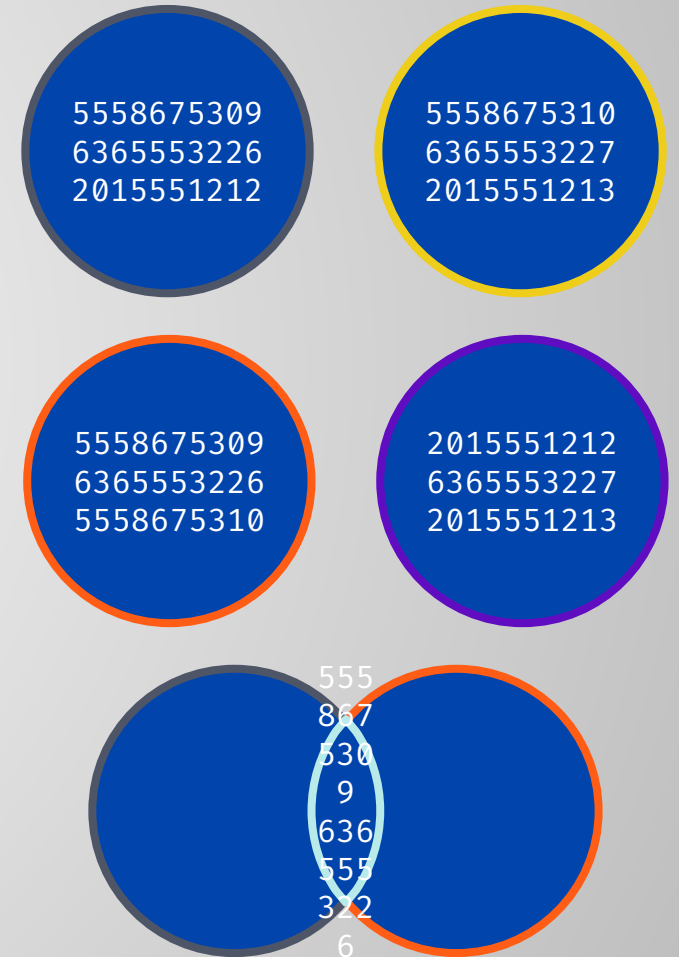
```
redis.local:6379> SCAN * MATCH account:*
```

Searching the Hard Way

Maintaining Your Own Indices



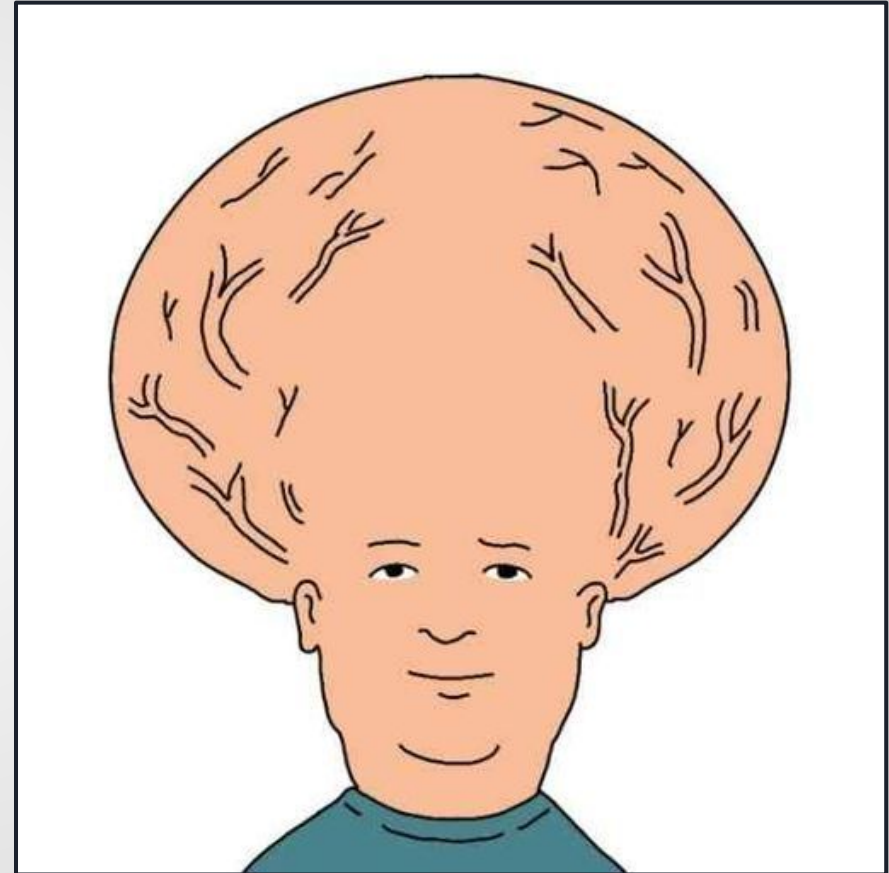
```
redis.local:6379> SADD accounts:type:checking  
5558675309 6365553226 2015551212  
  
redis.local:6379> SADD accounts:type:savings  
5558675310 6365553227 2015551213  
  
redis.local:6379> SADD accounts:overdrawn:true  
5558675309 6365553226 5558675310  
  
redis.local:6379> SADD accounts:overdrawn:false  
2015551212 6365553227 2015551213  
  
redis.local:6379> SINTER accounts:overdrawn:true  
accounts:type:checking
```



Searching the Smart Way

Introducing RediSearch

- Indexes JSON documents and Hashes.
- Provides full-text search and aggregation.
- Indices are updated automatically when data is updated.



Redisearch

Types of Fields



TEXT



TAG



NUMERIC



GEO



RediSearch

Creating Indices

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

returns single item

\$.description AS description TEXT

\$.type AS type TAG

\$.approvals[*] AS approvals TAG

\$.balance AS balance NUMERIC

\$.location AS location GEO

returns multiple items

longitude first
in format
12.34,56.78



RediSearch

Searching

```
redis.local:6379> FT.SEARCH account:index "*"
redis.local:6379> FT.SEARCH account:index "@description:watch"
redis.local:6379> FT.SEARCH account:index "@approvals:{ JC | GR }"
redis.local:6379> FT.SEARCH account:index "@balance:[ -inf 100 ]"
redis.local:6379> FT.SEARCH account:index "@location:[ 12.34 56.78 10 mi ]"
redis.local:6379> FT.SEARCH account:index "
    @description:watching
    @approvals:{ JC | GR }
    @balance:[ -inf 100 ]
    @location:[ 12.34 56.78 10 mi ]"
```



Redisearch

Other Useful Commands

```
redis.local:6379> FT._LIST
```

```
redis.local:6379> FT.INFO account:index
```

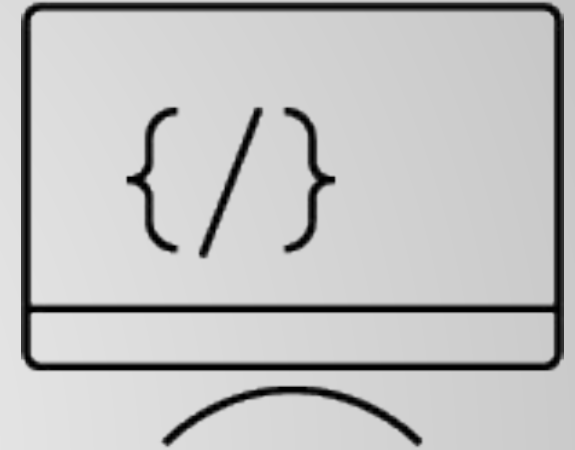
```
redis.local:6379> FT.DROPINDEX account:index
```

```
redis.local:6379> FT.AGGREGATE account:index "*"
                    GROUPBY 1 @type REDUCE COUNT 0 AS typeCount
```

```
redis.local:6379> FT.EXPLAIN account:index "
                    @description:watching
                    @approvals:{ JC | GR }
                    @balance:[ -inf 100 ]
                    @location:[ 12.34 56.78 10 mi ]"
```

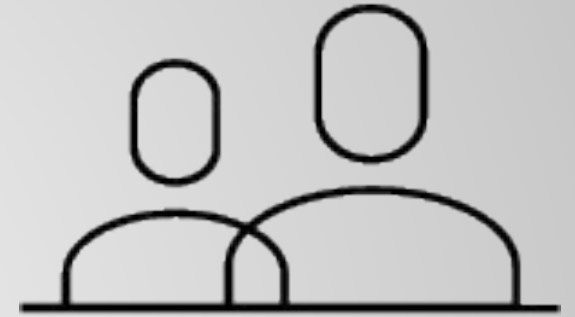
Demo

Redisearch



Try Out Redisearch

Follow the instructions in
04-REDISearch.md



Searching & JSON Made Easy

An Introduction to **redis^{OM}**

Redis OM



What's Redis OM

- A layer of abstraction over RedisSearch & RedisJSON
- Available for Java, .NET, Node.js, and Python
- We'll cover Redis OM for [Node.js](#)
- Built on top of Node Redis
- Supports index creation using **Schema**
- Enables basic CRUD Operations using **Repository**
- Provides a fluent search interface using **Search**

Node Redis

What's Node Redis

- Official and supported Redis client for Node.js
- Supports RedisJSON, RediSearch and all the other Redis modules
- 1-to-1 mapping of Redis commands to JavaScript code



```
npm install redis
```


Node Redis

Creating a Client



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

Node Redis

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`

Node Redis

Connecting to Redis

```
import { createClient } from 'redis'

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

redis.on('error', error ⇒ console.log('Redis Client Error', error))

await redis.connect()
```

Node Redis

Sending Commands to Redis

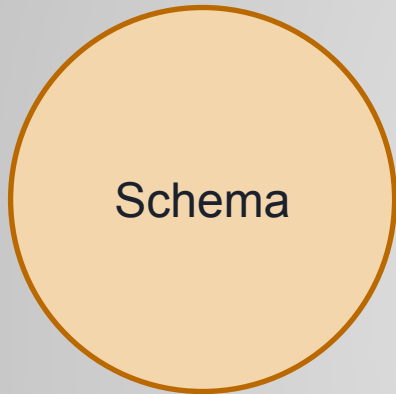
```
await redis.sAdd('accounts:type:checking',  
  [ 5558675309, 6365553226, 2015551212 ])
```

```
await redis.json.set('account:5558675309', '$', {  
  account: '5558675309',  
  name: 'Always Watching H.O.A.'  
})
```

```
const names = await redis.json.get('foo', '$.name')
```

Redis OM

Schema and Repository



- 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@beta
```

Schemas

Creating a Schema

```
const schema = new Schema('transaction', {  
  account: { type: 'string' },  
  approvals: { type: 'string[]' },  
  overdrawn: { type: 'boolean' },  
  
  memo: { type: 'text' },  
  
  amount: { type: 'number' },  
  date: { type: 'date' },  
  
  location: { type: 'point' }  
})
```

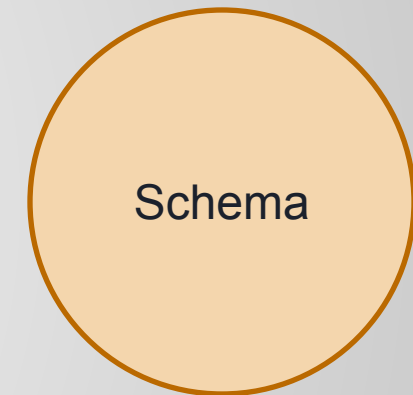
prefixes keys
in Redis

maps to
TAG

maps to
TEXT

maps to
NUMERIC

maps to
GEO



Schemas

JSONPath is Implied

```
const schema = new Schema('transaction', {  
  account: { type: 'string', path: '$.account' },  
  approvals: { type: 'string[]', path: '$.approvals[*]' },  
  overdrawn: { type: 'boolean', path: '$.overdrawn' },  
  
  memo: { type: 'text', path: '$.memo' },  
  
  amount: { type: 'number', path: '$.amount' },  
  date: { type: 'date', path: '$.date' },  
  
  location: { type: 'point', path: '$.location' }  
})
```

← note the array

Schemas

Can Be Mutated and Nested

```
const schema = new Schema('transaction', {  
  account: { type: 'string', path: '$.accountNumber' },  
  approvals: { type: 'string[]', path: '$.approvals[*].initials' },  
  overdrawn: { type: 'boolean' },  
  
  memo: { type: 'text', path: '$.memo' },  
  
  amount: { type: 'number', path: '$.details.amount' },  
  date: { type: 'date', path: '$.details.date' },  
  
  location: { type: 'point', path: '$.location' }  
})
```

renamed

array of objects

nested objects

Schemas

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

Schemas

If a Date or Point is Defined

Redis OM

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

```
instanceof Date
```

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

RedisJSON

1682611200000

"12.34,56.78"



Schemas

If a Date or Point 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  
}
```

Repositories

Creating a Repository and Saving an Entity

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

```
let transaction = {  
  account: "5158675309",  
  approvals: [ "JC", "GR" ],  
  overdrawn: false,  
  memo: "Payment for watching",  
  amount: 125.00,  
  date: '2023-04-27',  
  location: {  
    longitude: 12.34,  
    latitude: 56.78  
  }  
}
```

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

Repositories

What's in an Entity?

```
transaction.account      // "5158675309"  
transaction.approvals    // [ "JC", "GR" ]  
transaction.overdrawn    // false  
transaction.memo         // "Payment for watching"  
transaction.amount       // 125.00  
transaction.location     // { longitude: 12.34, latitude: 56.78 }  
  
transaction.date         // instanceof Date  
  
transaction[EntityId]    // "01FJYWEYRHYFT8YTEGQBABJ43J"
```

Can I Use My Own EntityID?

```
transaction = await repository.save("5558675309-90210", transaction)
```

Calling Save Again

```
transaction[EntityId] // "5558675309-90210"  
  
transaction.approvals.push('PF')  
transaction.memo = "Payment for watching, always watching"  
transaction.amount = 250.0  
  
transaction = await repository.save(transaction)
```

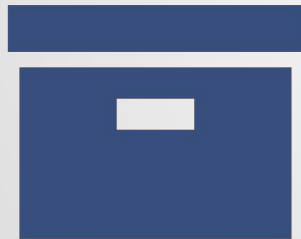
Saving an Entity with an EntityID does an upsert

Repositories

Fetching and Removing

```
const transaction = await repository.fetch("5558675309-90210")
```

```
await repository.remove("5558675309-90210")
```



Searching

Finding All The Things

redis^oM



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

creates (or recreates)
the index

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

creates a **Search**
instance

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

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

triggers
the search

Searching

Finding Specific Things

redis **OM**

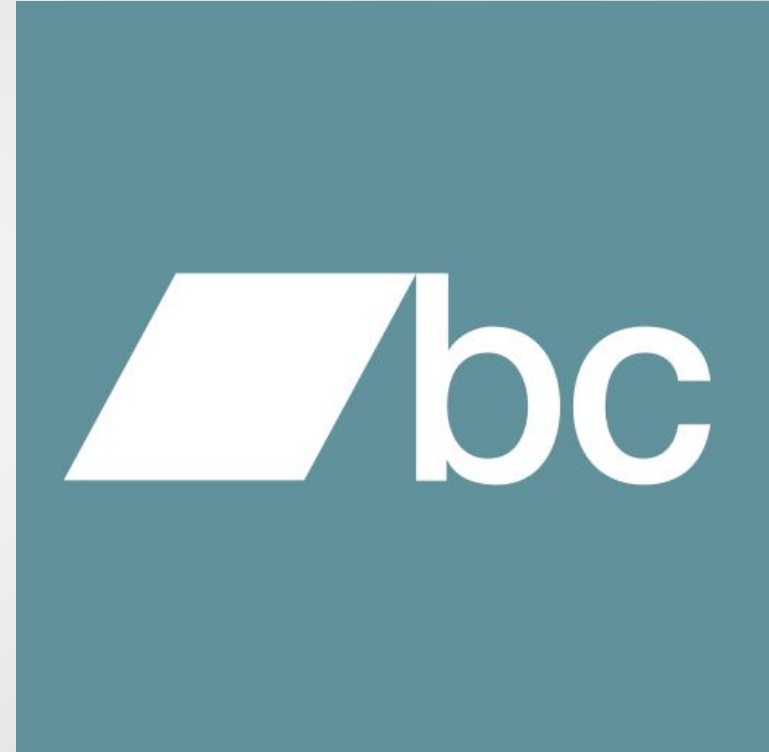


```
const transactions = await repository.search()
  .where('account').equals('5558675309')    // string
  .and('amount').is.greaterThan(20.00)      // number
  .and('overdrawn').is.false()              // boolean
  .and('date').is.after('2023-04-27')       // date
  .and('approvals').contains('JC')          // string[]
  .and('memo').matches('watch')             // full-text search
  .and('location').inRadius(               // point
    circle => circle.origin(12.34, 56.78).radius(10).miles)
  .return.all()
```

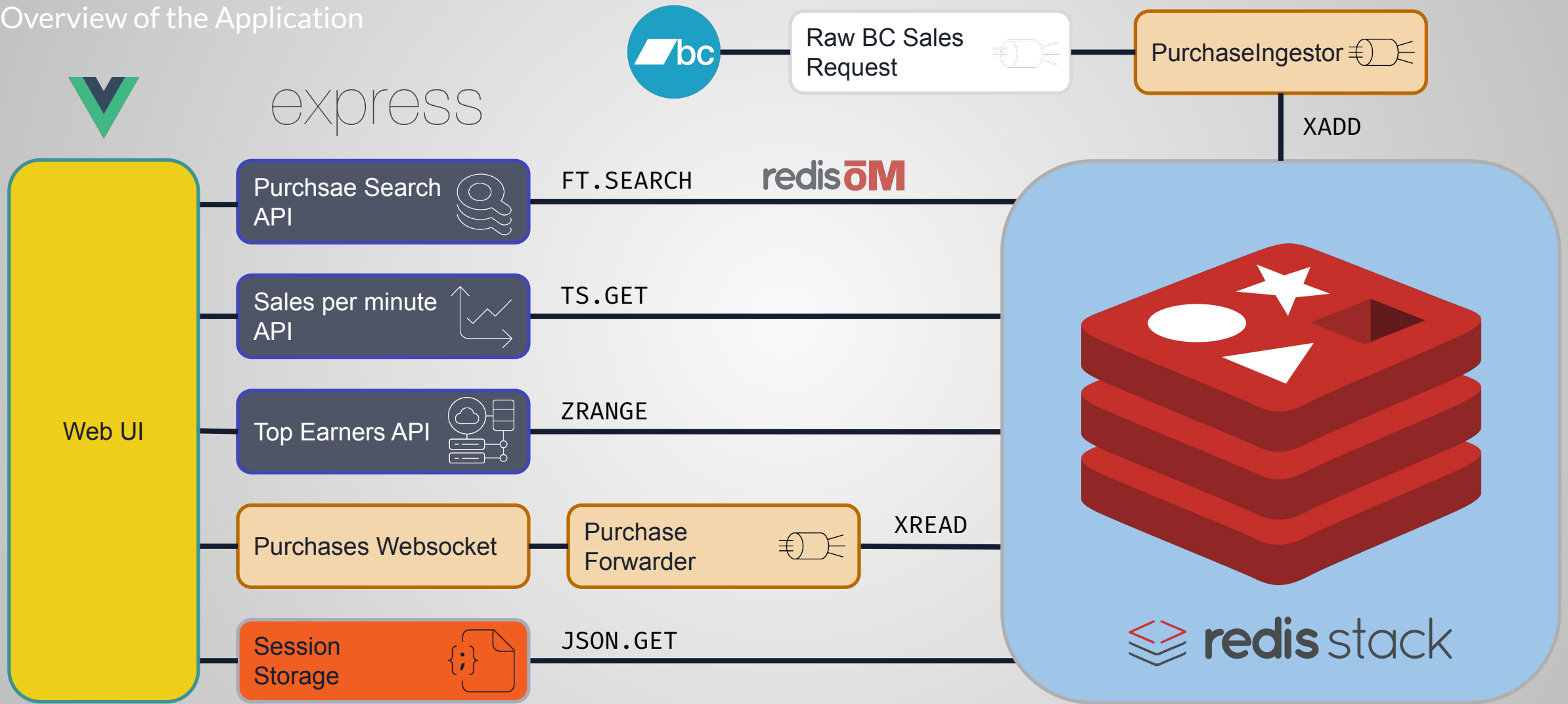
Full documentation at github.com/redis/redis-om-node/.

Beats By Redis

A Simple Application

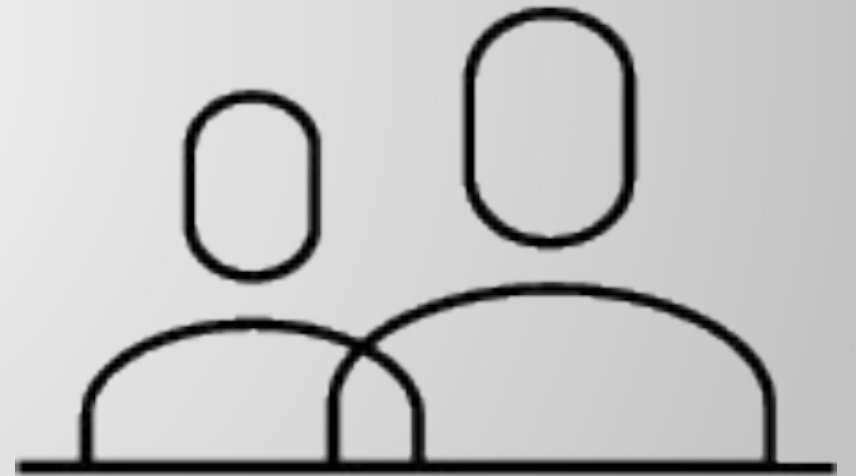


Overview of the Application



Complete the Beats-By-Redis API

Follow the instructions in **05-BEATS-BY-REDIS.md**



Connect, Learn, Explore



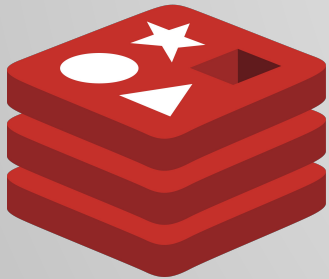
Redis Discord

discord.gg/redis



Redis University

university.redis.com



Redis Cloud

redis.com/try-free

Thank You!

savanna.norem@redis.com

justin.castilla@redis.com

