

redis

REremote **DI**ctionary **S**erver

Gabriel Borges
Guilherme Barth
Leonardo Carmona
Robson Chiarello

Introdução

- ▶ Banco de Dados NoSQL Redis, focado em key-value store;
- ▶ Disponível como serviço diretamente no AWS para uma solução rápida;
- ▶ Usado pela maioria das empresas atualmente (Twitter, GitHub, Pinterest, Snapchat, StackOverflow, Flickr);
- ▶ Indicado quando se precisa de velocidade, os dados cabem na memória e não são críticos.

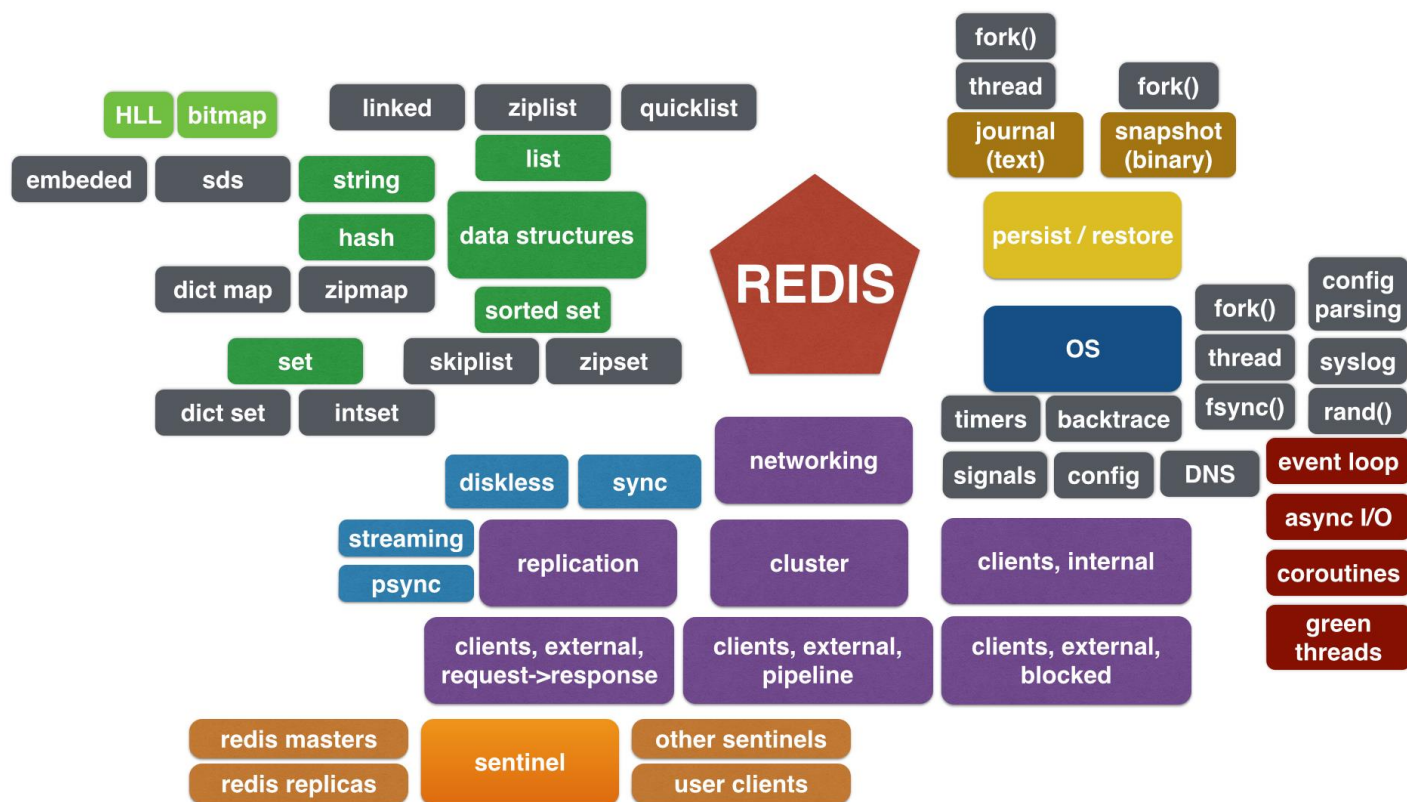
Exemplos de Uso do Redis

- ▶ Cache;
- ▶ Database auxiliar;
- ▶ Extensão in-memory da aplicação;
- ▶ Contagem de coisas;
- ▶ Carrinho de compras.

Características do Redis

- ▶ Memória RAM é a principal fonte de dados, disco é só um fallback;
- ▶ Puro Key-Value;
- ▶ Listas;
- ▶ Incremento e decremento de valores armazenados facilmente;
- ▶ Rapidez de escrita;
- ▶ Single Threaded;
- ▶ Deve ser utilizado juntamente com um banco SQL ou NoSQL Document Store.

Arquitetura do Redis



Comandos

- ▶ **Set**: seta o valor de uma key;
- ▶ **Get**: recupera o valor de uma key;
- ▶ **Incr**: incremento atômico do valor de uma key;
- ▶ **Del**: deleta uma key;
- ▶ **Expire**: seta o tempo de vida de uma key;
- ▶ **TTL**: recupera o tempo de vida restante de uma key.

Comandos Sobre Listas

- ▶ **RPUSH**: insere um valor ao final da lista;
- ▶ **LPUSH**: insere um valor no início da lista;
- ▶ **LRANGE**: retorna uma determinada parte da lista;
- ▶ **LLEN**: retorna o tamanho da lista;
- ▶ **LPOP**: remove o primeiro elemento da lista e o retorna;
- ▶ **RPOP**: remove o último elemento da lista e o retorna.

Comandos Sobre Sets

- ▶ **SADD**: adiciona um valor a um set;
- ▶ **SREM**: remove um valor de um set;
- ▶ **SISMEMBER**: verificar se um valor se encontra em um set;
- ▶ **SMEMBERS**: retorna uma lista dos membros de um set;
- ▶ **SUNION**: combina dois ou mais sets e retorna uma lista de seus elementos.

(Diferença entre sets e listas: sets não possuem valores repetidos)

(Sorted sets funcionam da mesma forma que sets mas possuem um atributo para ordenação)

Comandos Sobre Hashes

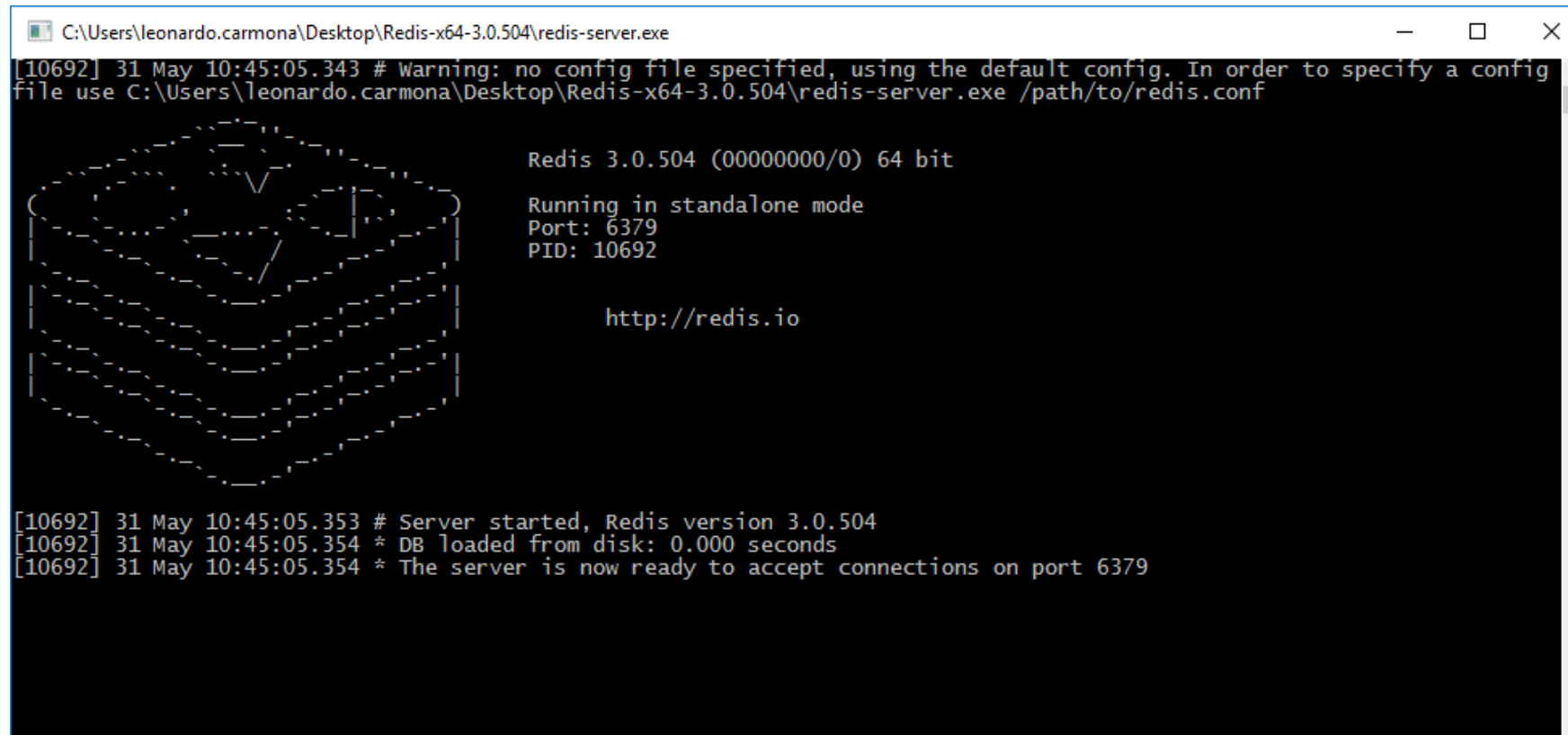
- ▶ **HSET**: seta um atributo de uma hash;
- ▶ **HGET**: retorna um atributo de uma hash;
- ▶ **HGETALL**: retorna uma hash;
- ▶ **HMSET**: permite setar diversos atributos de uma hash simultaneamente;
- ▶ **HINCRBY**: permite o incremento atômico de valores numéricos armazenados em uma hash;
- ▶ **HDEL**: deleta um atributo de uma hash.

Tipos de Dados do Redis

Tipo de Dado	Descrição
String	Pode conter objetos, imagens, etc. Tam max 512MB
List	Coleção de strings ordenada por inserção
Hash	Mapa entre valores e campos de strings
Set	Coleção não ordenada de strings não repetidas
Sorted Set	Coleção ordenada de strings não repetidas da pontuação menor para a maior
Bitmap e HyperLogLog	Baseados em Strings mas com uma semântica própria.

Redis em Ação

Servidor Redis



```
C:\Users\leonardo.carmona\Desktop\Redis-x64-3.0.504\redis-server.exe

[10692] 31 May 10:45:05.343 # Warning: no config file specified, using the default config. In order to specify a config
file use C:\Users\leonardo.carmona\Desktop\Redis-x64-3.0.504\redis-server.exe /path/to/redis.conf

Redis 3.0.504 (00000000/0) 64 bit

Running in standalone mode
Port: 6379
PID: 10692

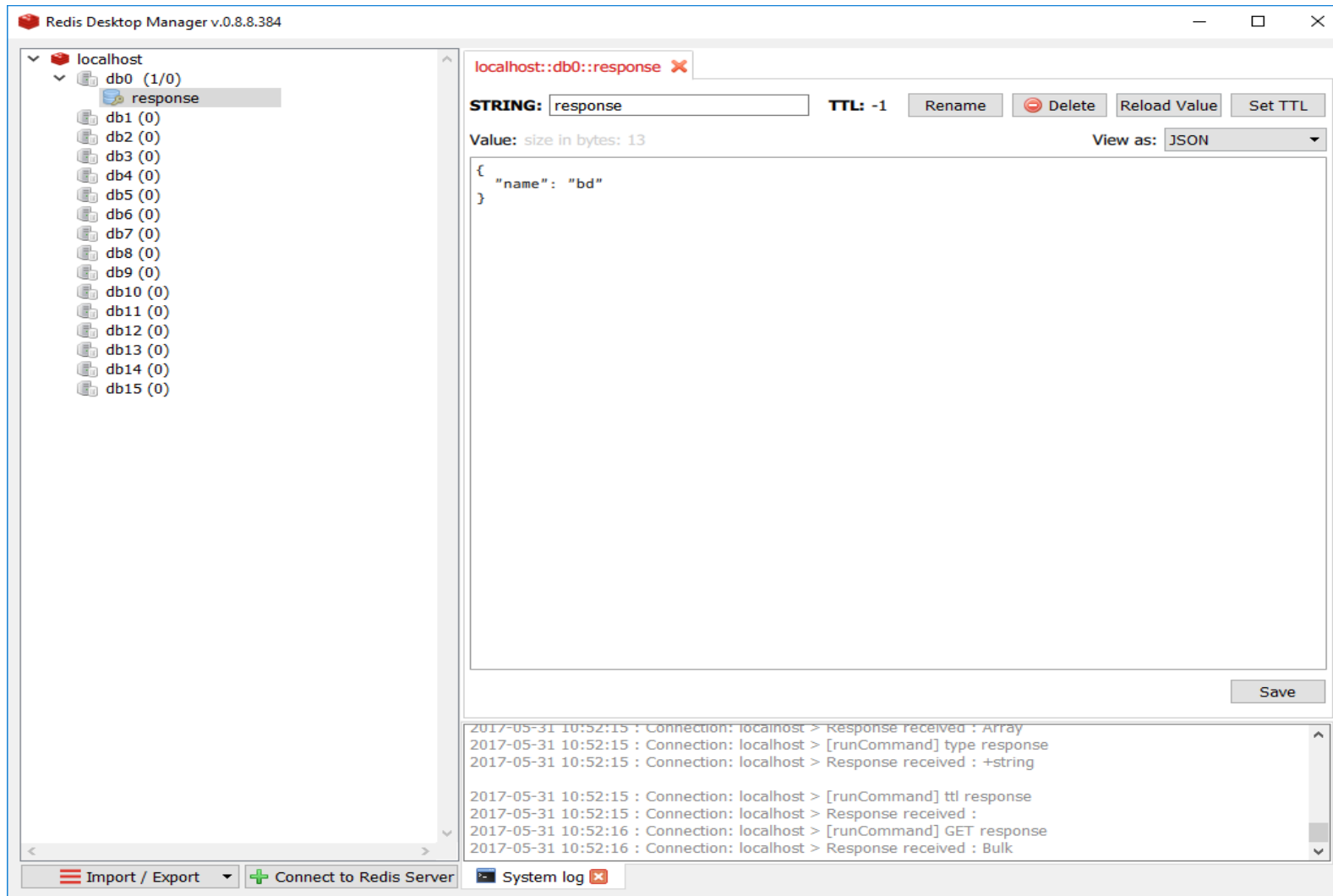
http://redis.io

[10692] 31 May 10:45:05.353 # Server started, Redis version 3.0.504
[10692] 31 May 10:45:05.354 * DB loaded from disk: 0.000 seconds
[10692] 31 May 10:45:05.354 * The server is now ready to accept connections on port 6379
```

Cliente Redis

```
C:\Users\leonardo.carmona\Desktop\Redis-x64-3.0.504\redis-cli.exe
127.0.0.1:6379> ping
PONG
127.0.0.1:6379> set response "{\"name\":\"bd\"}"
OK
127.0.0.1:6379> get response
"{\"name\":\"bd\"}"
127.0.0.1:6379> expire response 10
(integer) 1
127.0.0.1:6379> get response
"{\"name\":\"bd\"}"
127.0.0.1:6379> get response
(nil)
127.0.0.1:6379>
```

Redis Desktop Manager

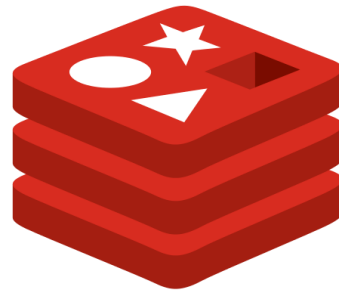
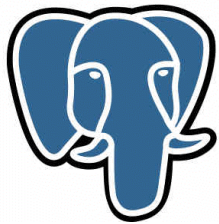


Protótipo com Redis

Tecnologias utilizadas



PostgreSQL

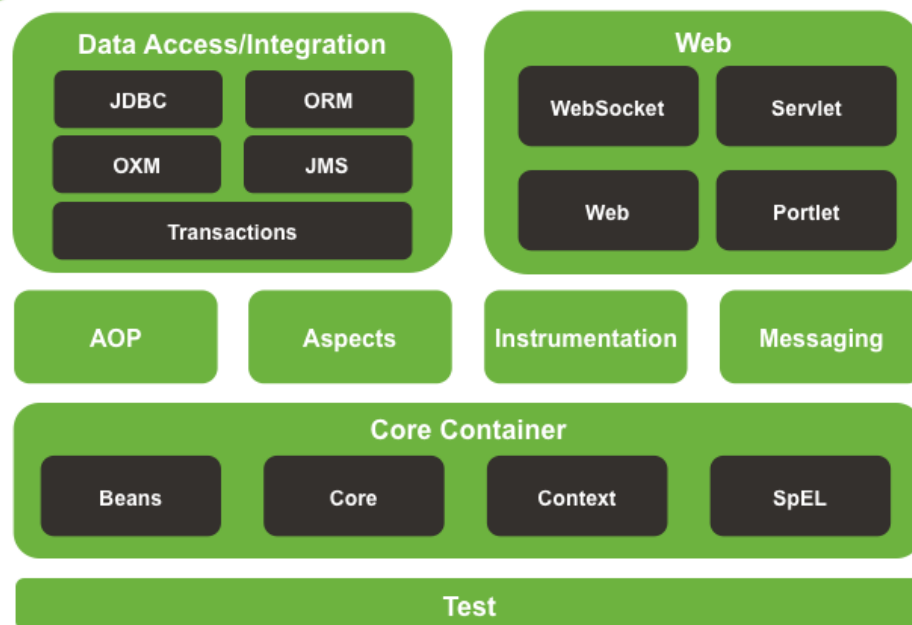


redis

Um pouco sobre o Spring



Spring Framework Runtime



Requisição sem cache

The screenshot displays a REST client interface for a project named "GPA - API Gateway". The request is a GET to `http://localhost:8080/api/person/1`. The "Headers" tab is active, showing a single header: `X-Disable-springcache: true`. The "Body" tab shows the response in JSON format: `{\"id\": 1, \"name\": \"leonardo\"}`. The status is `200 OK` and the response time is `827 ms`.

Key	Value
<input checked="" type="checkbox"/> X-Disable-springcache	true
New key	value

```
1 {
2   "id": 1,
3   "name": "leonardo"
4 }
```

Requisição com cache

The screenshot displays an API client interface for a request to the endpoint `http://localhost:8080/api/person/1`. The request method is `GET`. Under the `Headers` tab, the header `X-Disable-springcache` is set to `false`. The response status is `200 OK` with a response time of `28 ms`. The response body is shown in JSON format as `{id: 1, name: 'leonardo'}`.

Person

Person

GET `http://localhost:8080/api/person/1` Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Code

Key	Value	Bulk Edit	Presets
<input checked="" type="checkbox"/> X-Disable-springcache	false		
New key	value		

Body Cookies Headers (3) Tests Status: 200 OK Time: 28 ms

Pretty Raw Preview JSON Save Response

```
1 {
2   "id": 1,
3   "name": "leonardo"
4 }
```

Conclusão

- ▶ Simples de implementar em uma solução existente;
- ▶ Poderosa ferramenta de armazenamento de dados;
- ▶ Mais rápida do que bancos convencionais por ser usada em memória.