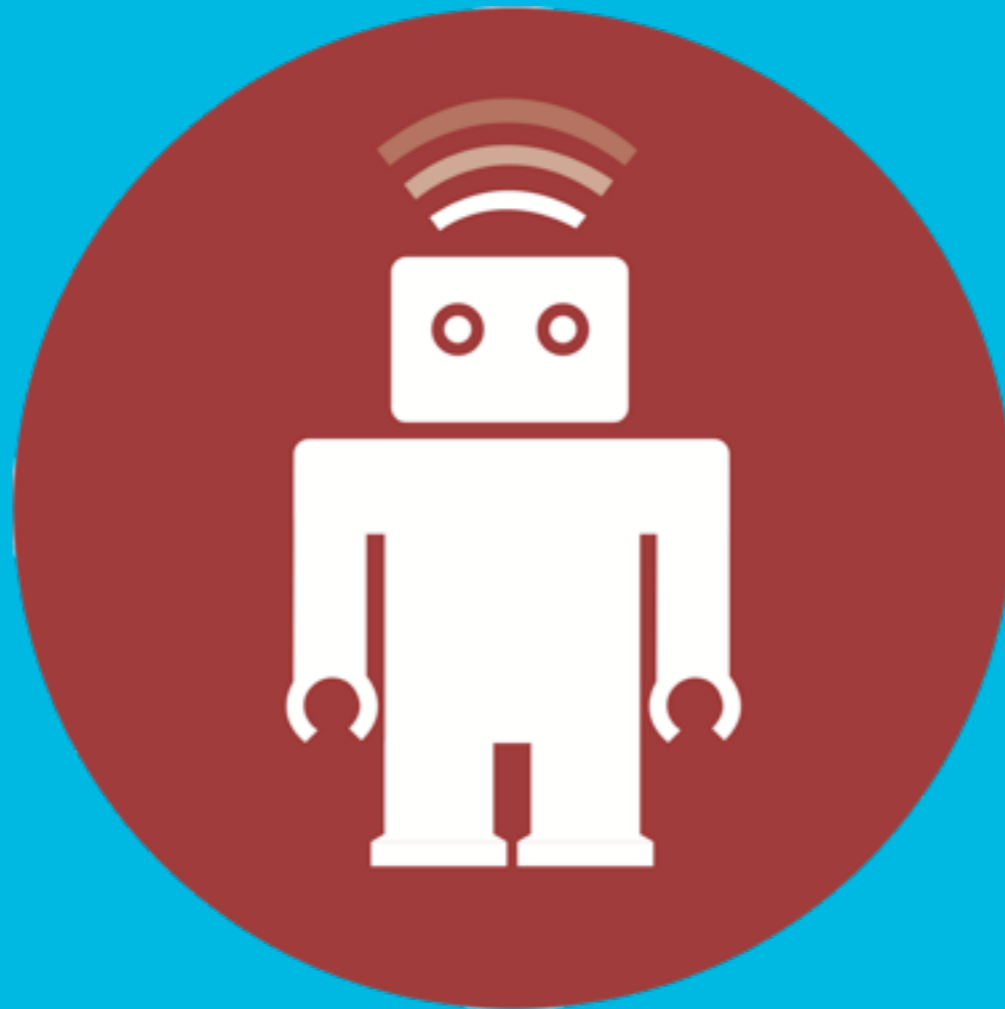


# Redis: Data Cheeseburgers

**Nick Quaranto**

**@grush / [nick@quaran.to](mailto:nick@quaran.to)**

**i work at**



**thoughtbot.com**

**we use**



redis

**on**



**hoptoadapp.com**

**and**



**rubygems.org**







**SQL**



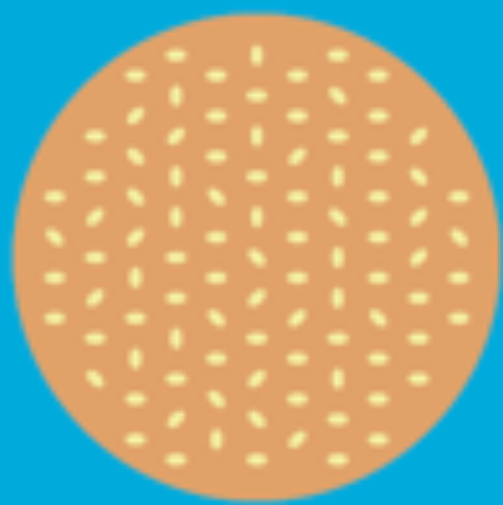


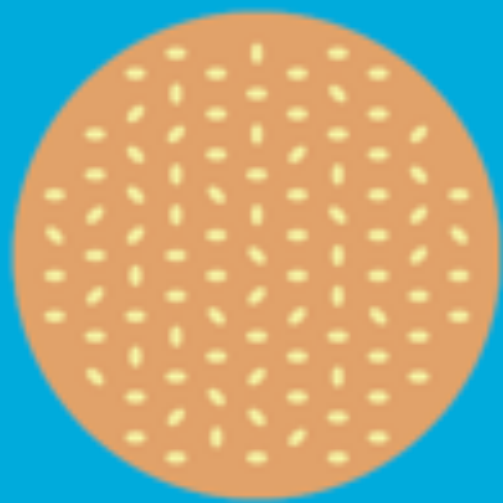


**EPIC**

**SQL**

**TIME**





# Redis

**redis**

**“an  
advanced  
key-value  
store”**

# no

- **tables**
- **relations**
- **documents**
- **graphs**

**yes**

**data structures**

# a big hash





**really it's just**

key => data structure

# run commands

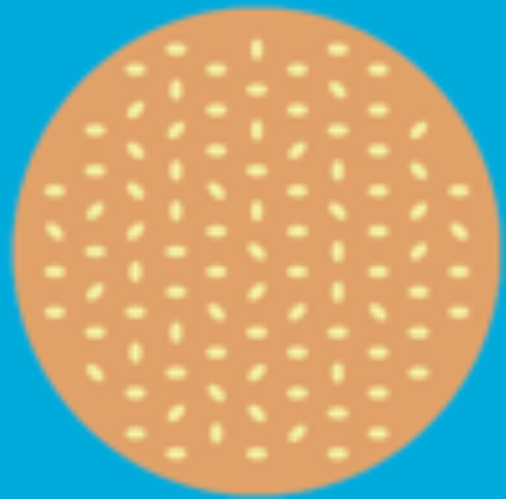
```
> GET foo
```

```
"bar"
```

# memory

- **all keys & values in RAM**
- **sync to disk when necessary**
- **various levels of paranoia**

- **data structures**
- **getting started**
- **use cases**



# **data structures**

# strings

- **like memcached**
- **chunk of data**
- **binary aware**

fries



> get fries  
(nil)

fries



> set fries over  
OK

fries



> get fries  
over

fries

over

```
> strlen fries  
4
```

fries

over

```
> getrange fries 0 1  
ov
```

fries

overdone

```
> append fries over  
done
```



chips

overdone

```
> rename fries chips
```

```
OK
```

chips

```
> del chips
```

```
1
```

chips

```
> exists chips
```

```
0
```

# counters

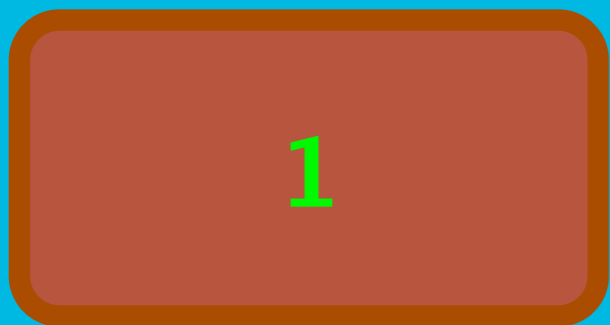
- **like strings, cast to an integer**
- **atomic increment/decrement**
- **very, very fast**

burgers



```
> get burgers  
nil
```

burgers



```
> incr burgers  
1
```

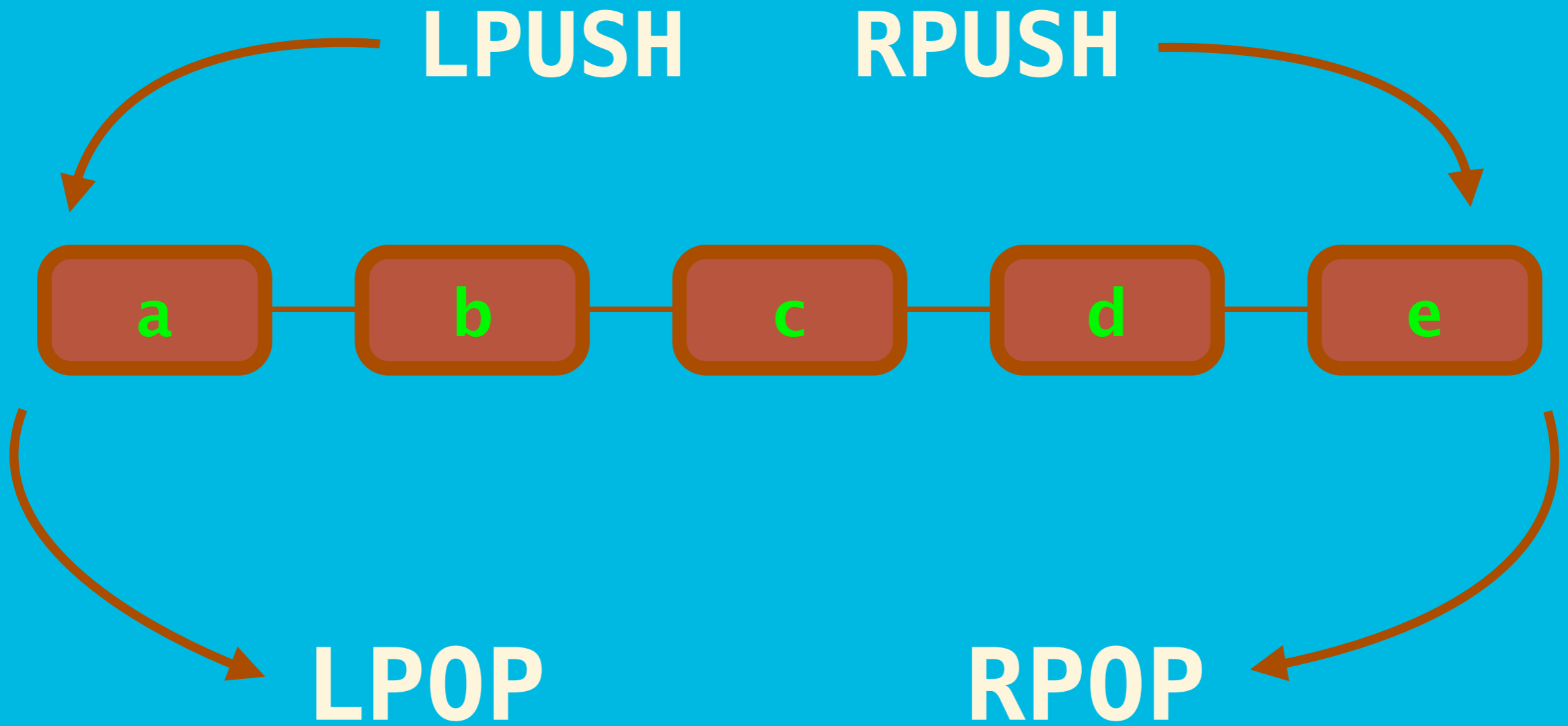
burgers



```
> incrby burgers 41  
42
```

# lists

- **push, pop**
- **random access**
- **blocking actions**



thanks to peter cooper (@peterc) for the graphs!

order



```
> rpush order burger
1
> rpush order hotdog
2
> rpush order fries
3
```

# order



```
> lrange order 0 1
```

```
1. burger
```

```
2. hotdog
```

```
> lindex order 2
```

```
fries
```

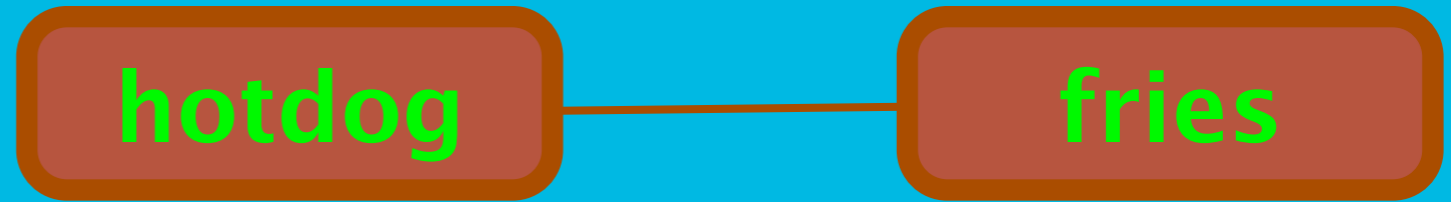
# order



```
> \range order 0 -1  
1. burger  
2. hotdog  
3. fries
```



order



```
> lpop order  
burger
```

# sets

- **unique elements**
- **intersect, union, difference**

✓ s add meat bacon

1

✓ s add meat turkey

1

✓ s add toppings bacon

1

✓ s add toppings bacon

0

**meat**

**burger turkey ham  
capicola jerky bacon**

**toppings**

**relish bacon ketchup  
mustard pickles**

**thanks to peter cooper (@peterc) for the graphs!**

> sinter meat toppings  
1. bacon

**meat**

**burger turkey ham  
capicola jerky**

**bacon**

**relish ketchup  
mustard pickles**

**toppings**

# sorted sets

- **high score list**
- **set algebra**
- **ranges by score  
or rank**

> zadd menu 4.99 burger

1

> zadd menu 2.99 shake

1

> zadd menu 1.99 fries

1

# menu

key

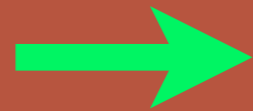
score

fries



1.99

shake



2.99

burger



4.99

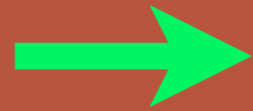


# menu

key

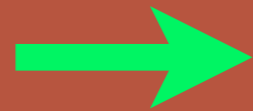
score

fries



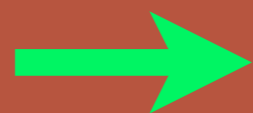
1.99

shake



2.99

burger



4.99

```
> zrange menu  
0 -1
```

```
1. fries  
2. shake  
3. burger
```

# menu

key

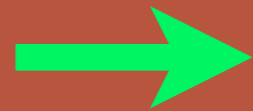
score

fries



1.99

shake



2.99

burger



4.99

```
> zrank menu  
fries
```

```
0
```

```
> zscore menu  
fries
```

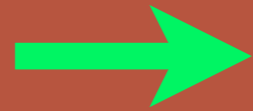
```
1.99
```

# menu

key

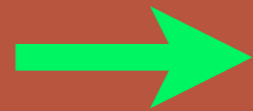
score

fries



1.99

shake



2.99

burger



4.99

```
> zrangebyscore  
menu 2 5
```

```
1. shake  
2. burger
```

# menu

key

score

fries



1.99

deleted!

shake



2.99

burger



4.99

```
> zremrangebyscore  
menu 1.50 4.50  
2
```

# hashes

- **easier to get all keys, values**
- **space saver**

**orders:1**

```
nick      burger
john      fries
mike      shake
joe       salad
created_at 1298686121
```

```
> hset orders:1 nick burger
```

```
1
```

```
> hset orders:1 john fries
```

```
1
```

orders:1

nick	burger
john	fries
mike	shake
joe	salad
created_at	1298686121

```
> hget orders:1 mike  
shake
```

```
> hlen orders:1
```

```
5
```

```
> hexists orders:1 brian
```

```
0
```

orders:1

nick	burger
john	fries
mike	shake
joe	salad
created_at	1298686121

> hvals orders

1. burger

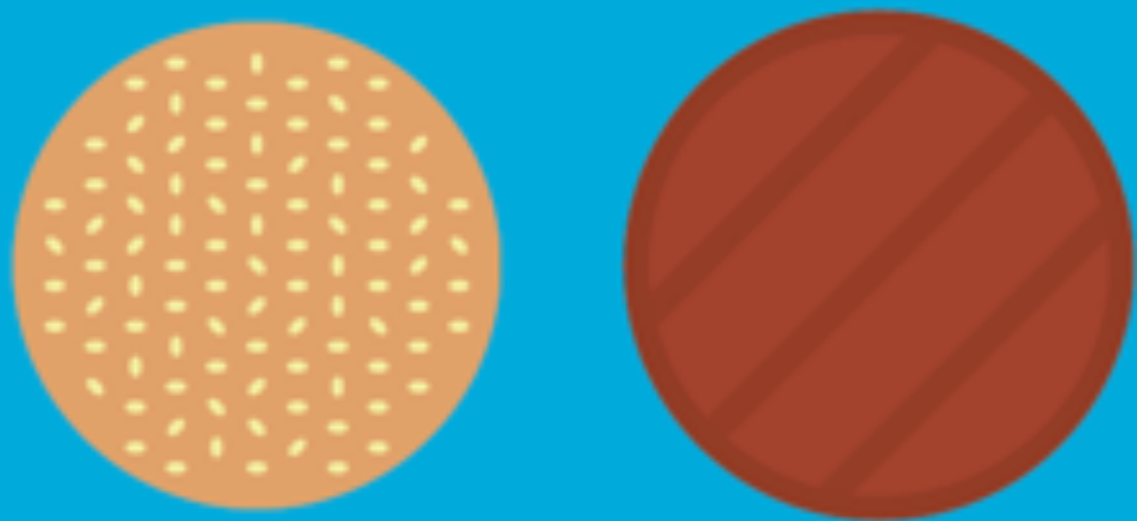
2. fries

3. shake

4. salad

5. 1298686121





**getting started**

# community

**BSD, on GitHub**

**Awesome maintainer (antirez)**

**IRC, wiki, mailing list**

**Hosting providers:  
Redis TO GO, OpenRedis**

# installing

```
git clone git://github.com/antirez/redis
```

```
make
```

```
redis-server
```

**or... \$YOUR\_DISTRO's package manager**

# sysadmin

**simple telnet interface**

**no authorization/authentication by default**

**AUTH password**

**use appendonly & daily cron BGREWRITEAOF**

# performance

**depends on system and configuration**

**ranges from 5,000 to 120,000 ops/sec**

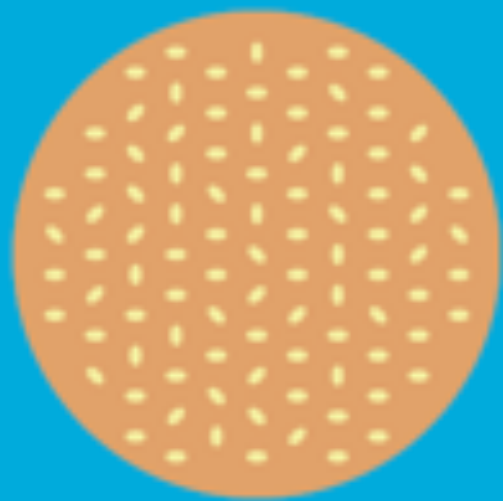
**commands have  $O(n)$  complexity listed**

# **misc**

**master/slave replication**

**diskstore soon**

**cluster support someday?**



**use cases**

# when?

**Whenever you want to store data fast that doesn't need to be 100% consistent.**

**-Mathias Meyer**

[http://www.paperplanes.de/2009/10/29/when\\_to\\_redis.html](http://www.paperplanes.de/2009/10/29/when_to_redis.html)



# hit counter

**ideal with speed and INCR/INCRBY**

**total counts = counters**

**daily per URL = sorted set (ZINCRBY)**

# on a URL hit...

**HTTP:**

```
GET /index.html
```

**Redis:**

```
INCR index.html
```

```
ZINCRBY hits 1 index.html
```

# basic stats

```
> get index.html
```

```
3910
```

```
> get pages/docs.html
```

```
2983
```

```
> get public/404.html
```

```
199
```

# advanced

```
> zrange hits 0 -1 withscores  
1) "public/404.html"  
2) "199"  
3) "pages/docs.html"  
4) "2983"  
5) "index.html"  
6) "3910"
```

# queue

**atomic pops = multiple workers**

**start a job:**

```
RPUSH key '{"some":"data"}'
```

**worker daemon(s):**

```
BLPOP key
```

<http://redis.io>



**Thanks!**