

# Redis Use Cases

# Background

- Key-Value-Store
- Memory only (do NOT use VM)
- "Memcache on steroids
- lightweight

# Deployment

- 613kb tgz
- compiles in 12 seconds
- master-slave / master-master setup (memory only)
- sharding by e.g. type (up to you)

# Ruby API

```
require "rubygems"  
require "redis"
```

```
R = Redis.new  
R.set("hello", "world")  
=> OK  
R.get("hello")  
=> "world"
```

# Keys

- Strings, Namespaces by convention
- ops
  - keys <pattern>
  - del <key>
  - ttl, expire, expireat

# Data Types

- String
- Hash
- Set
- List
- Sorted Set

# String

- Status Messages
- Counters
- Flags
- Locks
- Simple Configs
- Cached HTML (with TTL)

# Hash

- "better" Strings (less memory)
- KV-Store inside KV-Store
- Resource Caching (e.g. everything for an Artist)
- Store attributes of "Resources"
- Same operations as strings

# Set

- no duplicates, no order
- friends/favorites for e.g. users
- tags
- ...

# List

- duplicates + order
- push
- pop
- => queues (e.g. Resque)

# Sorted Sets

- automatically sorted
- ops:
  - `zadd <set_key> <value> <key>`
  - `z(incr/decr)by <set_key> <incr> <key>`
  - `rem, rank, (rev)range, card, count`