Let's Chat

Ruby and **Elixir**. An exploration of cross language communication

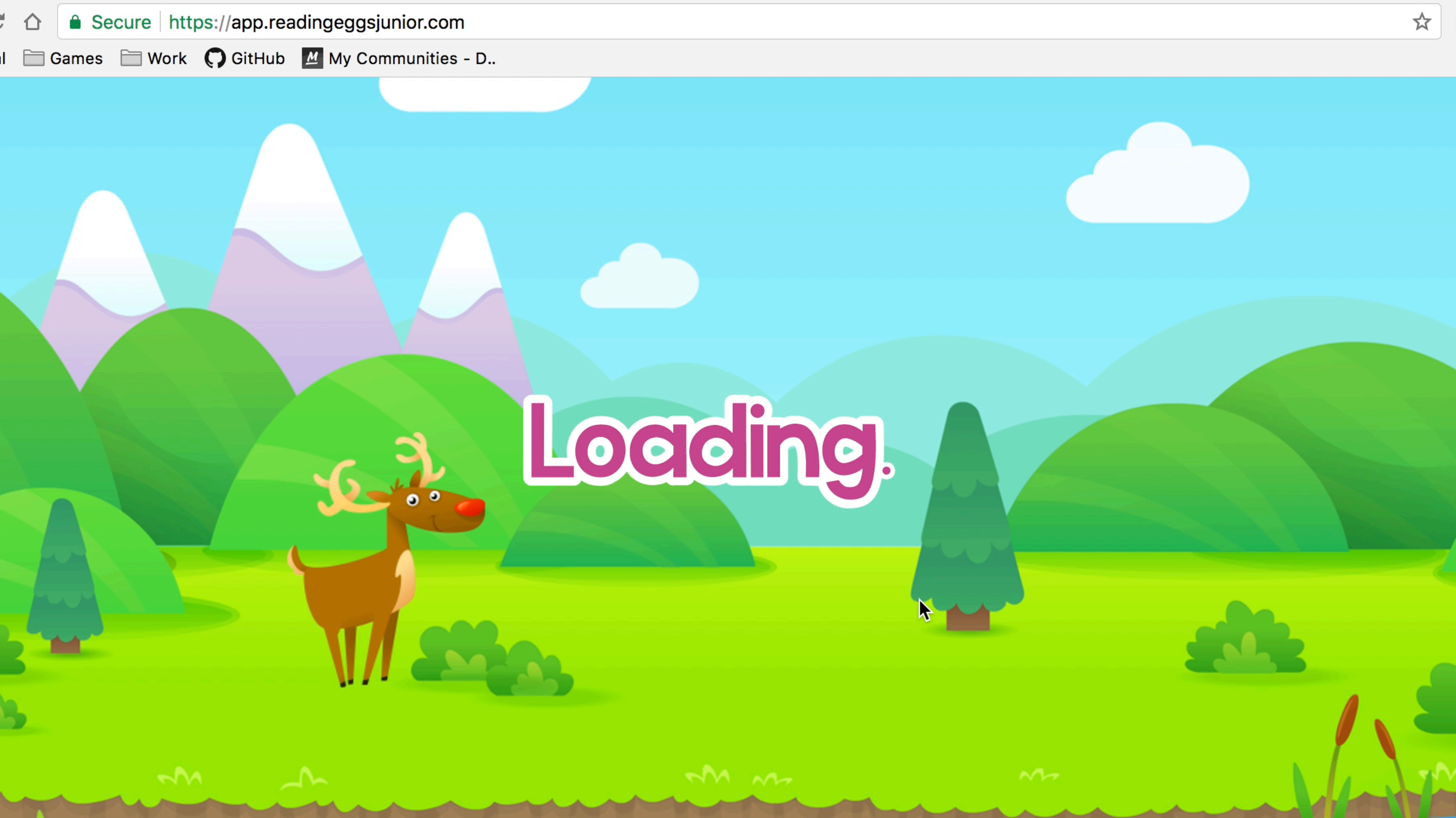

Hey,
I'm Garrett
4
Kwantime
28
0
8
8
1
1
ENEMY TURN
3
6
10
10
11
11
12
12
13
13
2
5
26
1
4
Swipe
4
Mire Keeper
4
Mire Keeper

I work at Netflix


NETFLIX
STRANGER THINGS
WELCOME TO HAWKINS

I ~~work~~ for Blake Education

We teach children


Secure
https://app.readingeggsjunior.com
Games
Work
GitHub
My Communities - D..
Loading.

We use

Ruby && Elixir

Why did we move?

— really fast
— immutable data
— parallel / concurrent
— easy to isolate complexity into applications (microservices w/o http)
— encourages better design
— shiny
— does your taxes
— #1 hacker news

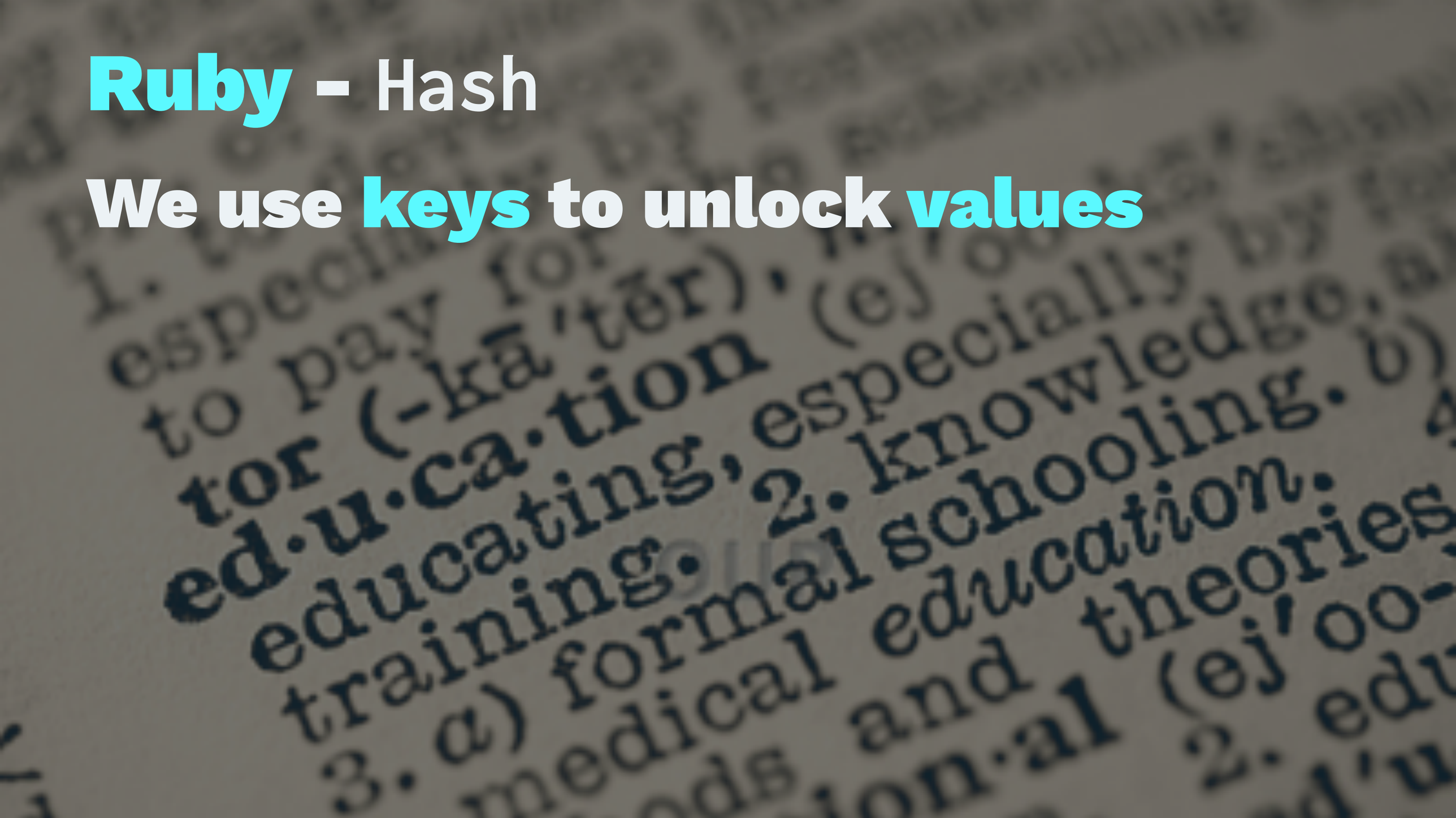
Agenda

— Refresher on **Ruby**
— Explain **Redis**
— Explore **Sidekiq**
— Introduce **Elixir**
— **Demo**
— Recap

Refresher on Ruby

Ruby – Hash

We use keys to unlock values



Ruby - Hash

Ruby - Hash

🏠 my_house

🚪 door_1 🔑 🚿 shower
🚪 door_2 🔑 🛋️ living_room
🚪 door_3 🔑 🛏️ bedroom

Ruby - Hash

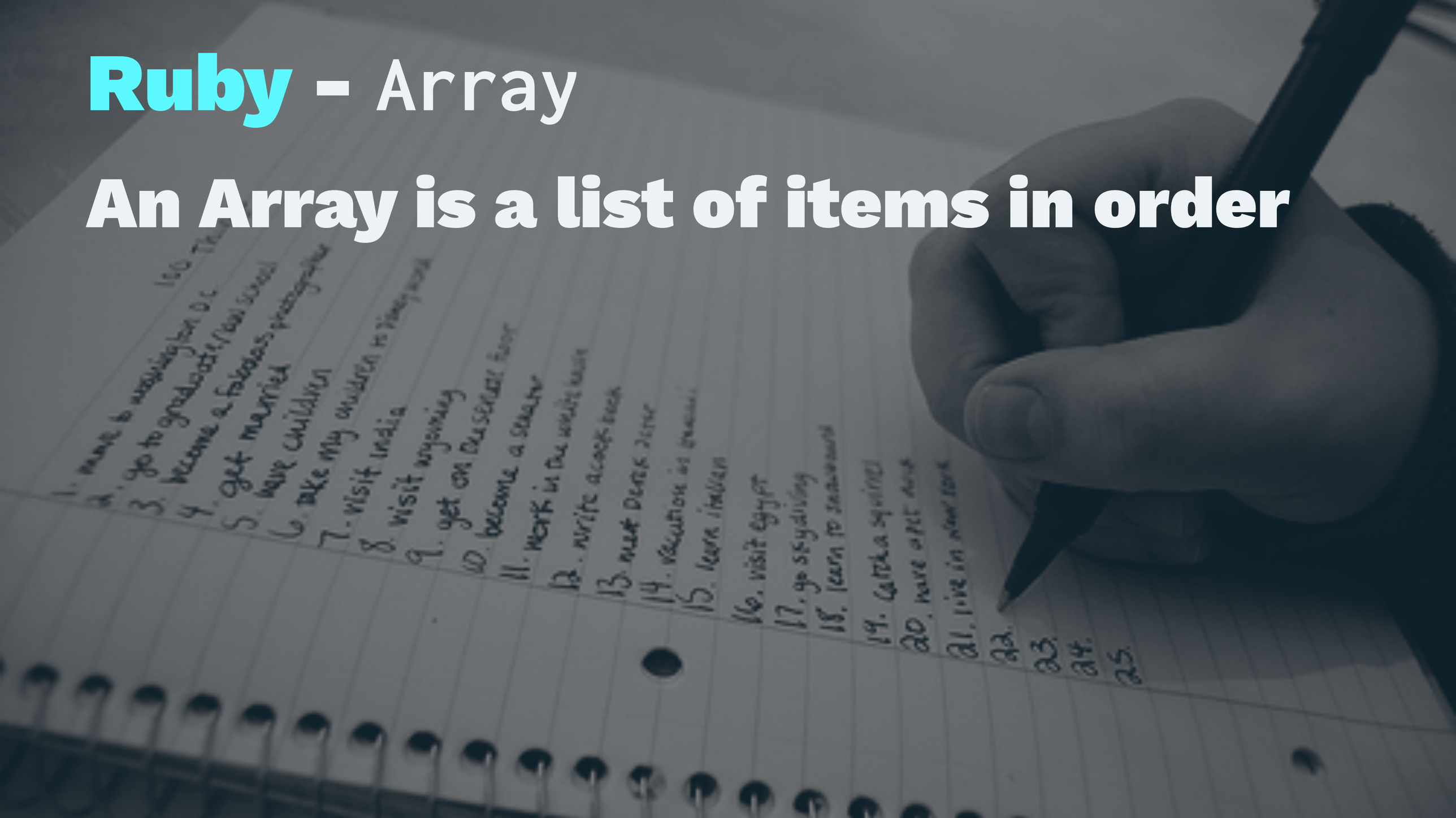
```
my_house = {
  door_1: "shower",
  door_2: "living_room",
  door_3: "bedroom",
}
```

Ruby - Hash

```ruby
person = {
  name: "Garrett",
  age: 27,
  hearthstone_rank: 3,
  fav_things: []
}
person[:hearthstone_rank]
# => 3
person[:fav_things] << "gummy bears"
# => ["gummy bears"]
```

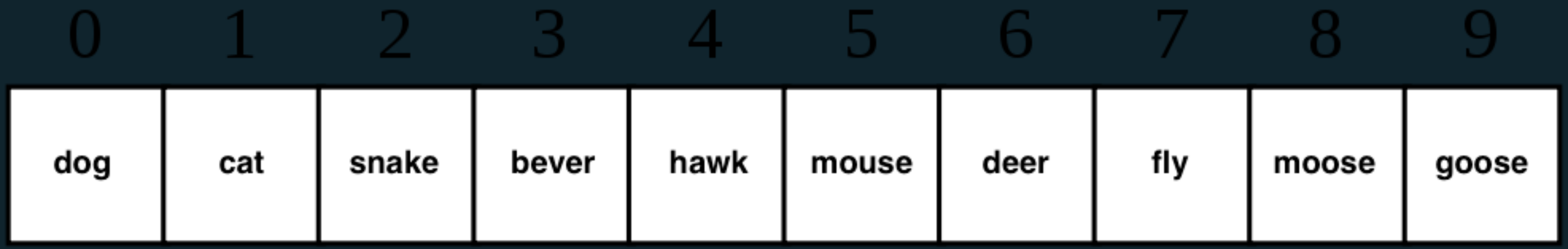
Ruby - Array

An Array is a list of items in order



Ruby - Array

Ruby - Array


0
1
2
3
4
5
6
7
8
9
dog
cat
snake
bever
hawk
mouse
deer
fly
moose
goose

Ruby - Array

```ruby
a = []              # => []
a << "abc"          # => ["abc"]
a                   # => ["abc"]
a.pop               # => "abc"
a                   # => []
```

Ruby - Array

```ruby
a = []              # => []
a.push("abc")       # => ["abc"]
a                   # => ["abc"]
a.pop               # => "abc"
a                   # => []
```

Ruby - Array

Appending to a list

```
x << y
x.push(y)
```

Ruby - Array

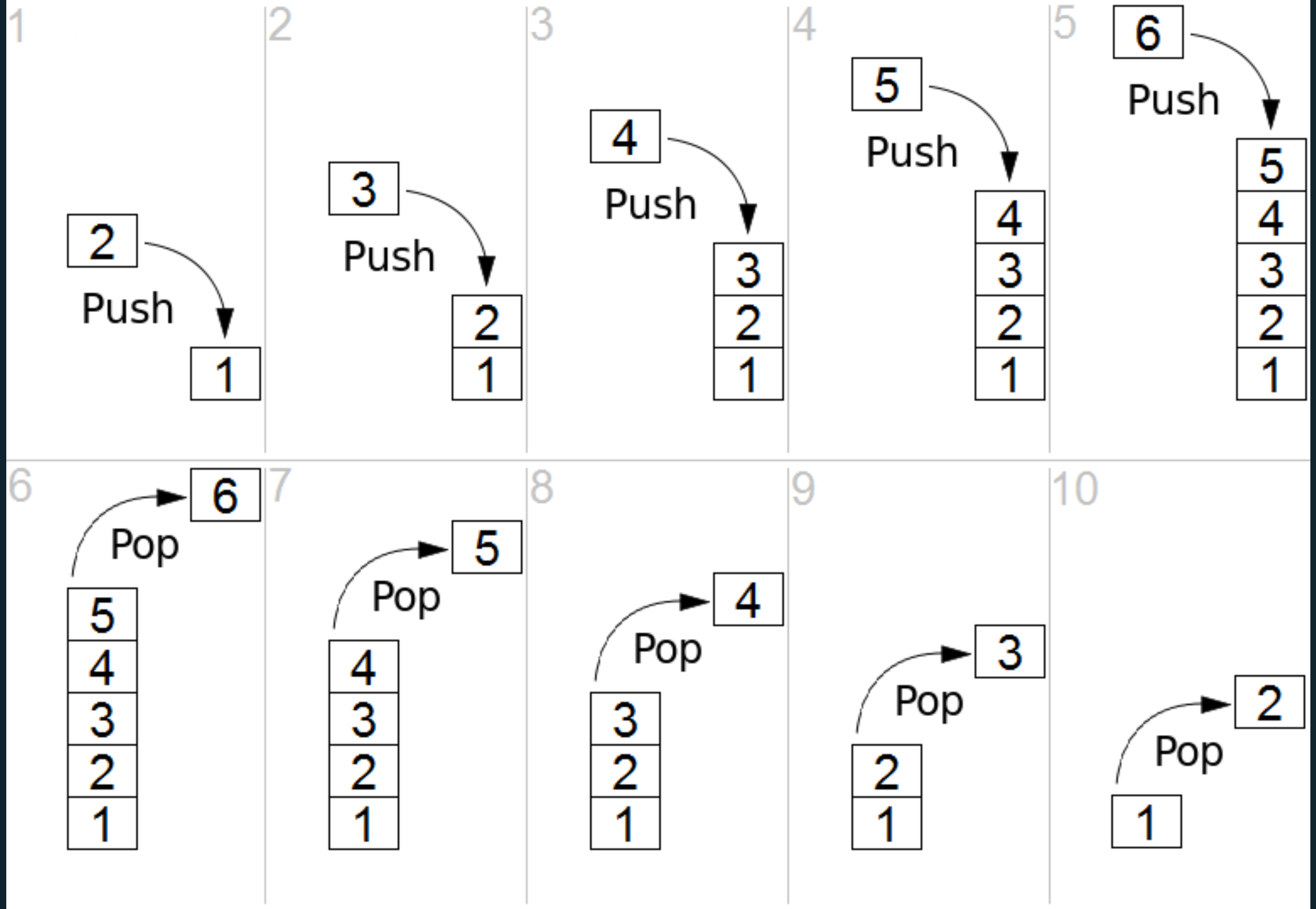
Pulling items off a list

```
x.pop(y)
```

Ruby - Array

Push && Pop

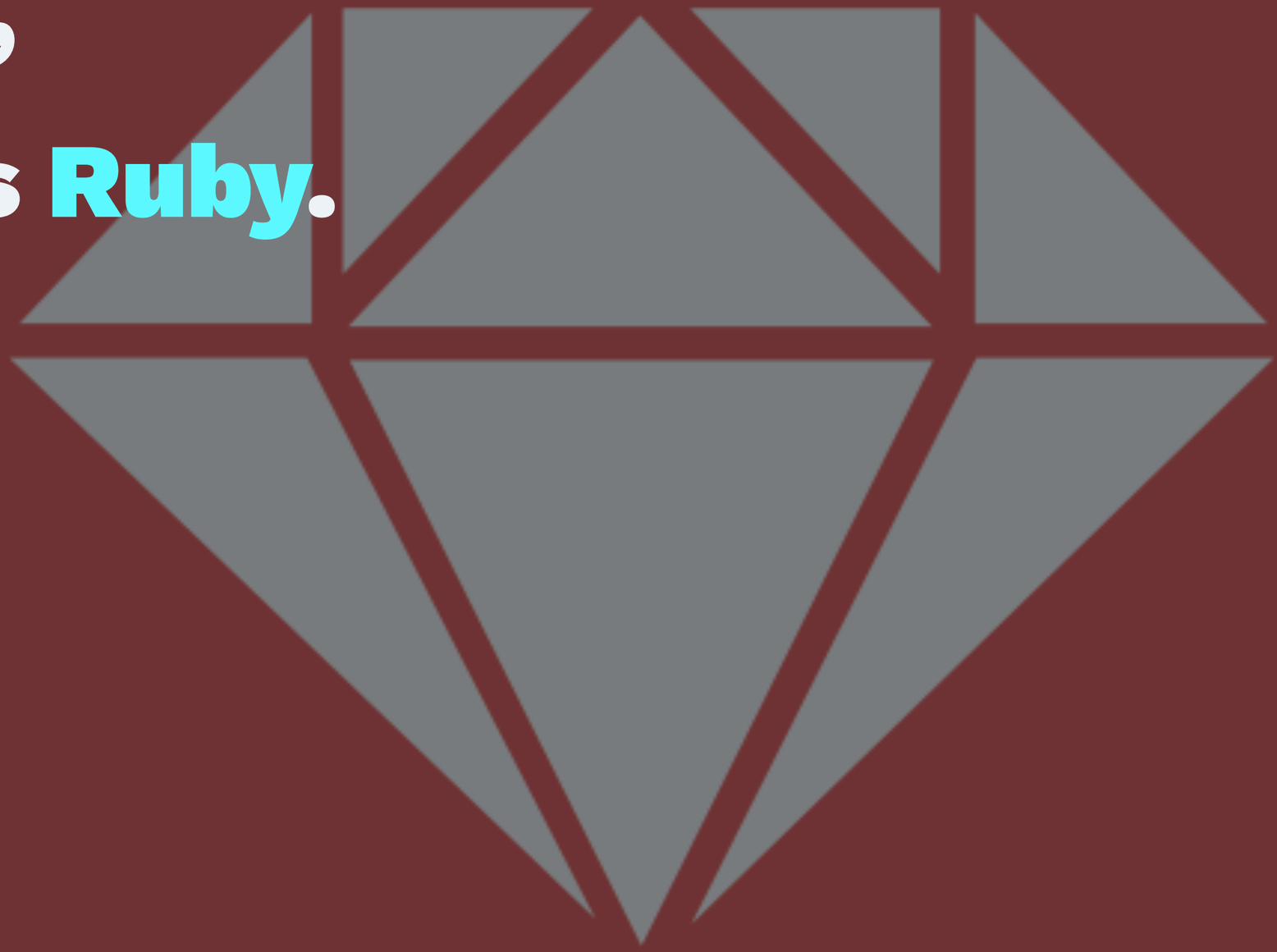
```
x.push(y)
x.pop(y)
```


1
2
Push
1
2
3
Push
2
1
3
4
Push
3
2
1
4
5
Push
4
3
2
1
5
6
Push
5
4
3
2
1
6
Pop
6
5
4
3
2
1
7
Pop
5
4
3
2
1
8
Pop
4
3
2
1
9
Pop
3
2
1
10
Pop
2
1


PushPop

Well,

that's Ruby.



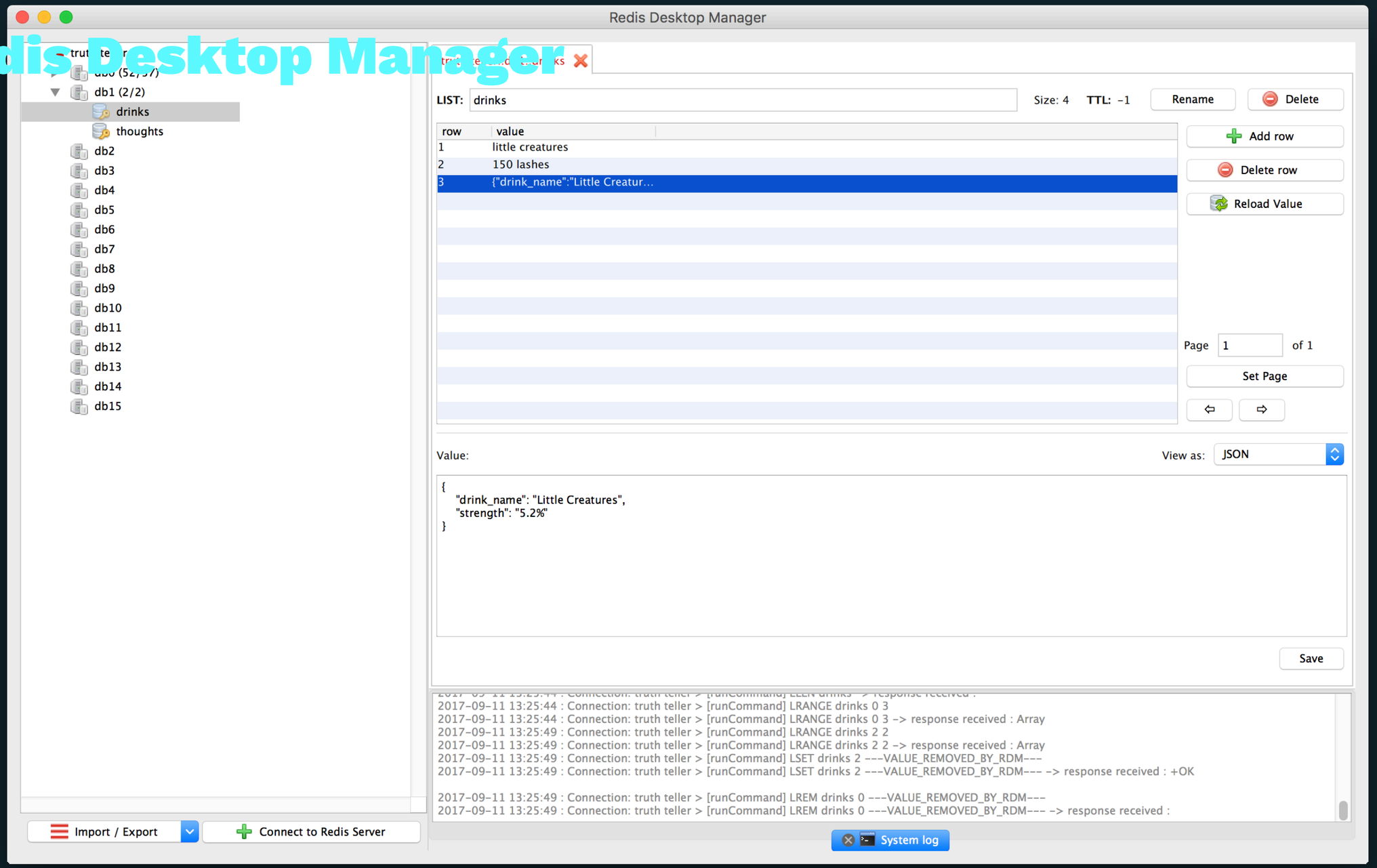
Explain Redis


Redis is a hash
in the cloud.

```
$redis = {}

# setter
$redis[:thought] = "Where's the Elixir already?"

# getter
$redis[:thought]
# => "Where's the Elixir already?"

# setter
$redis[:drinks] << "little creatures"
$redis[:drinks] << "150 lashes"

# getter
$redis[:drinks]
# => ["little creatures", "150 lashes"]
```

```
$ redis-cli

set thought "Where's the Elixir already?"

get thought
# => "Where's the Elixir already?"

rpush drinks "little creatures"
rpush drinks "150 lashes"

lrange drinks 0 -1
# =>  1) "little creatures"
# =>  2) "150 lashes"

lpop drinks
# =>  "little creatures"
```

Redis Desktop Manager

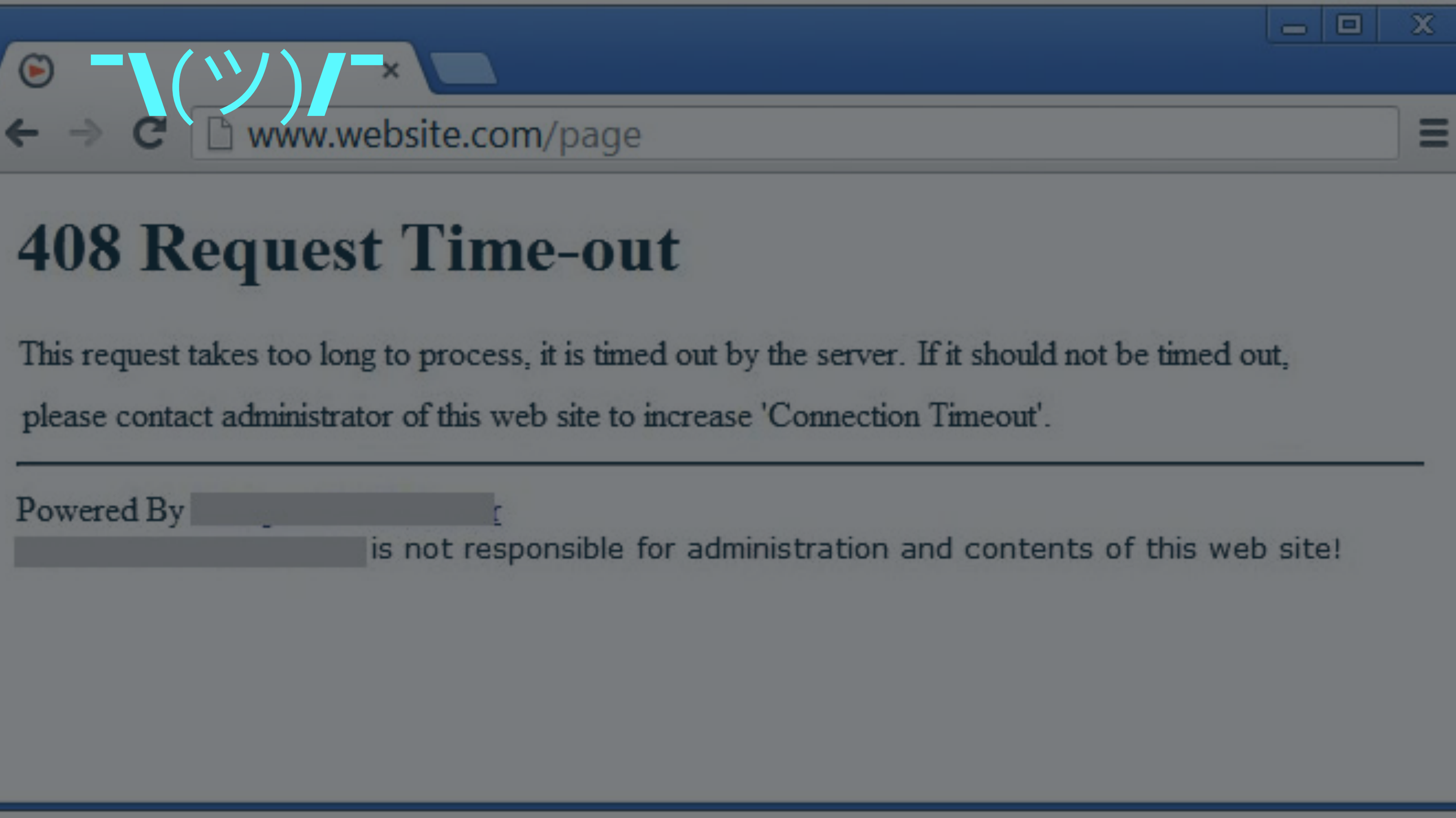


Explore Sidekiq

Sidekiq

framework for background job processing

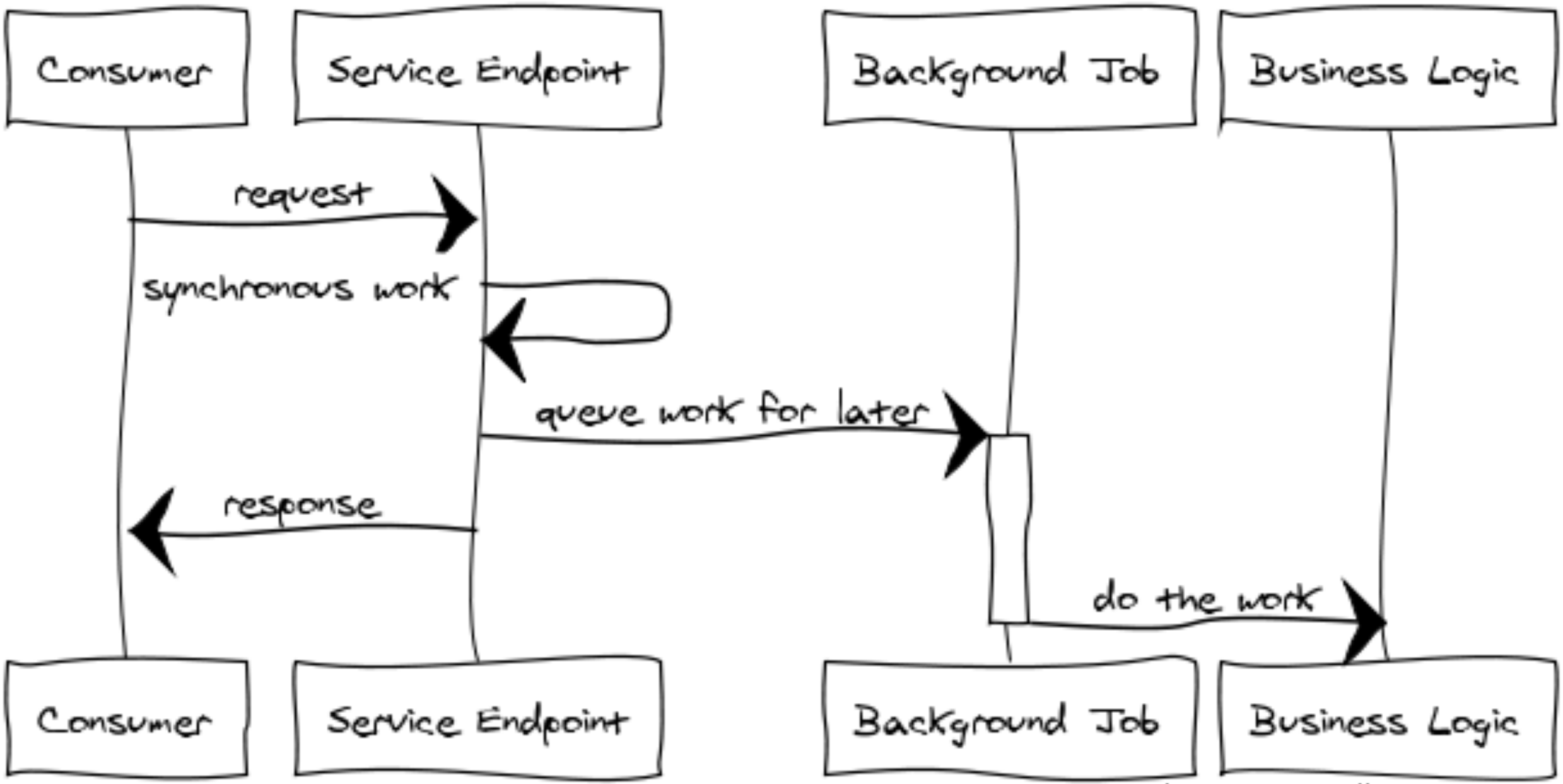
Why do I need a Sidekiq


¯\(ツ)/¯
www.website.com/page
408 Request Time-out
This request takes too long to process, it is timed out by the server. If it should not be timed out,
please contact administrator of this web site to increase 'Connection Timeout'.
Powered By
is not responsible for administration and contents of this web site!

Common Use cases

— Emails
— Expensive computation
— Imagine Processing
— Reports
— Etc..

Background Job



Sidekiq is built on 3 parts

- Client
  - push jobs into redis
- Redis
  - store the jobs
- Server
  - process the jobs

Client

```
MyWorker.perform_async({abc: "123", simple_as: "drm"})
```

Redis

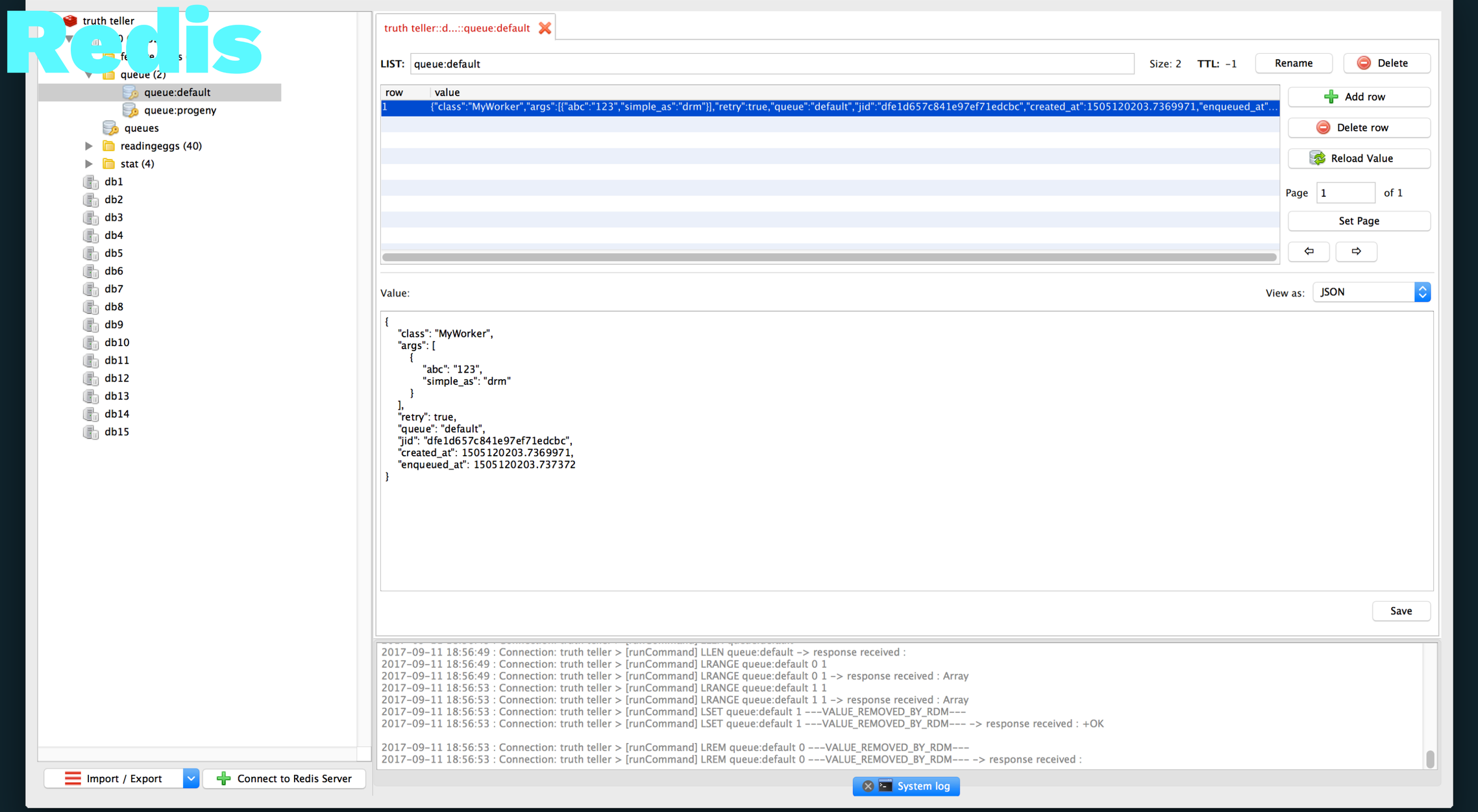
```
rpush queue:default
  "{\"class\":\"MyWorker\",\"args\":[{\"abc\":\"123\",\"simple_as\":\"drm\"],
  \"retry\":true,\"queue\":\"default\", \"jid\":\"f03cd72fef59bfbbf4098890\",
  \"created_at\":1503728661.8513222,\"enqueued_at\":1503728661.851379}"
```

Server

```
class MyWorker
  include Sidekiq::Worker

  def perform(hash)
    puts hash.inspect # => {abc: "123", simple_as: "drm"}
  end
end
```

Client

```
class SomethingSingularOrPluarl < BigBaseController
  def create
    MyWorker.perform_async(superfriend_params)
    head :created
  end

  private def superfriend_params
    params.permit! # {abc: "123", simple_as: "drm"}
  end
end
```

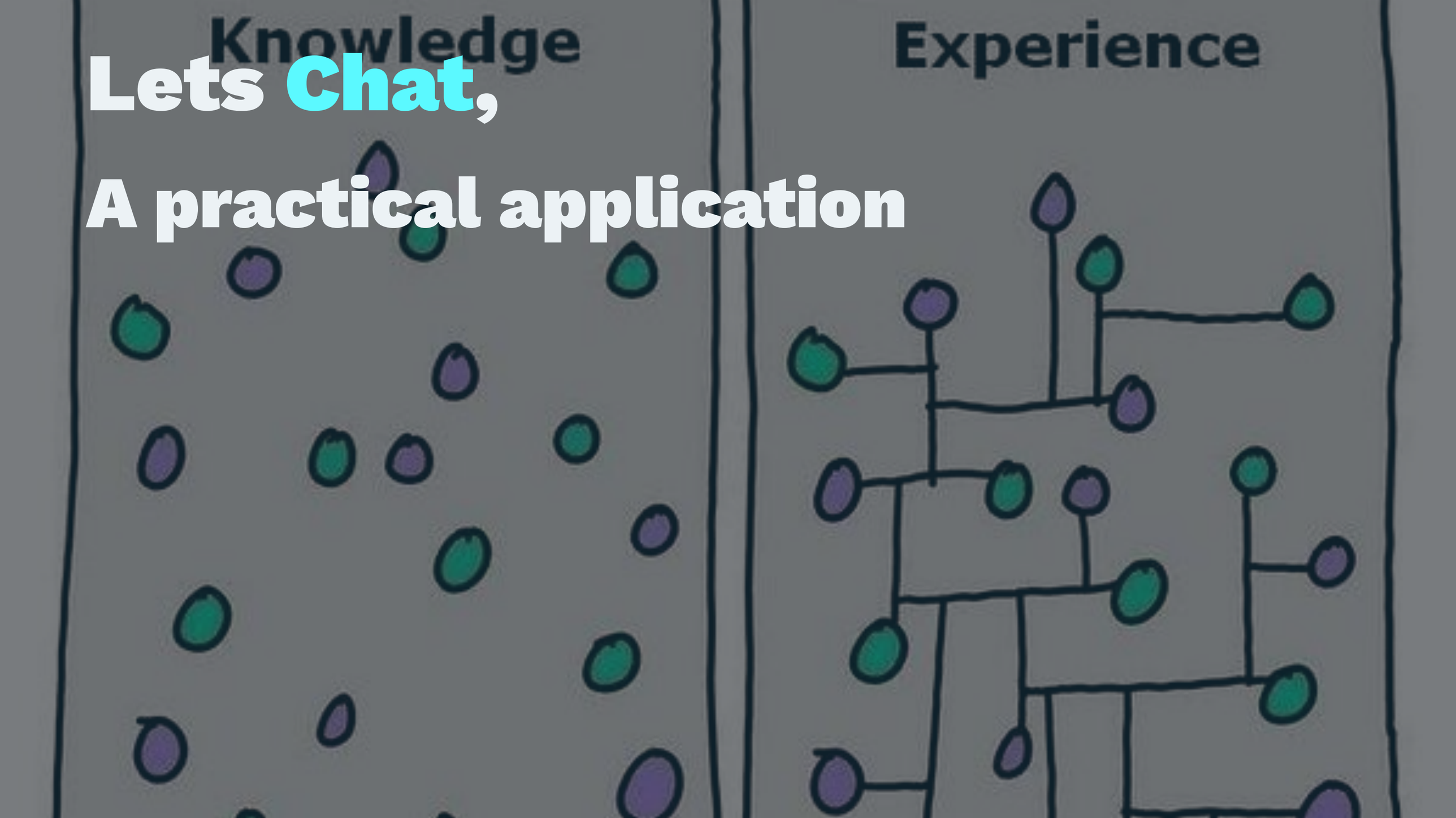

Redis Desktop Manager
Redis
truth teller
queue (2)
queue:default
queue:progeny
queues
readingeggs (40)
stat (4)
db1
db2
db3
db4
db5
db6
db7
db8
db9
db10
db11
db12
db13
db14
db15
Import / Export
Connect to Redis Server
truth teller::d...::queue:default
LIST: queue:default
Size: 2
TTL: -1
Rename
Delete
row
value
1
{"class":"MyWorker","args":[{"abc":"123","simple_as":"drm"}],"retry":true,"queue":"default","jid":"dfe1d657c841e97ef71edcbc","created_at":1505120203.7369971,"enqueued_at"...
Add row
Delete row
Reload Value
Page 1 of 1
Set Page
Value:
View as: JSON
{
"class": "MyWorker",
"args": [
{
"abc": "123",
"simple_as": "drm"
}
],
"retry": true,
"queue": "default",
"jid": "dfe1d657c841e97ef71edcbc",
"created_at": 1505120203.7369971,
"enqueued_at": 1505120203.737372
}
Save
2017-09-11 18:56:49 : Connection: truth teller > [runCommand] LLEN queue:default -> response received :
2017-09-11 18:56:49 : Connection: truth teller > [runCommand] LRANGE queue:default 0 1
2017-09-11 18:56:49 : Connection: truth teller > [runCommand] LRANGE queue:default 0 1 -> response received : Array
2017-09-11 18:56:53 : Connection: truth teller > [runCommand] LRANGE queue:default 1 1
2017-09-11 18:56:53 : Connection: truth teller > [runCommand] LRANGE queue:default 1 1 -> response received : Array
2017-09-11 18:56:53 : Connection: truth teller > [runCommand] LSET queue:default 1 ---VALUE_REMOVED_BY_RDM---
2017-09-11 18:56:53 : Connection: truth teller > [runCommand] LSET queue:default 1 ---VALUE_REMOVED_BY_RDM--- -> response received : +OK
2017-09-11 18:56:53 : Connection: truth teller > [runCommand] LREM queue:default 0 ---VALUE_REMOVED_BY_RDM---
2017-09-11 18:56:53 : Connection: truth teller > [runCommand] LREM queue:default 0 ---VALUE_REMOVED_BY_RDM--- -> response received :
System log

Server

```
class MyWorker
  include Sidekiq::Worker

  def perform(hash)
    puts hash.inspect # => {abc: "123", simple_as: "drm"}
  end
end
```

Lets Chat,

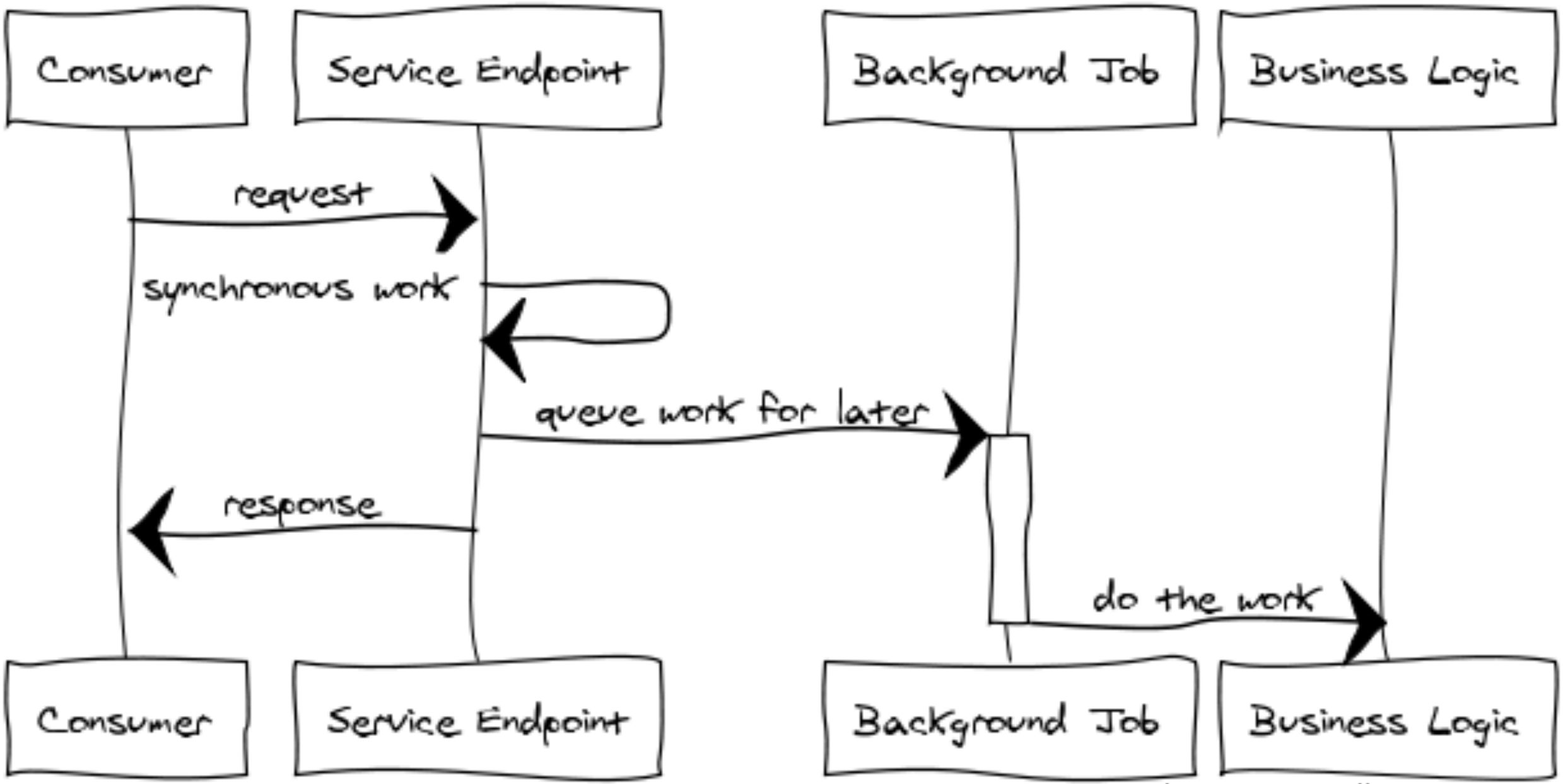
A practical application



Sidekiq - how does it work?

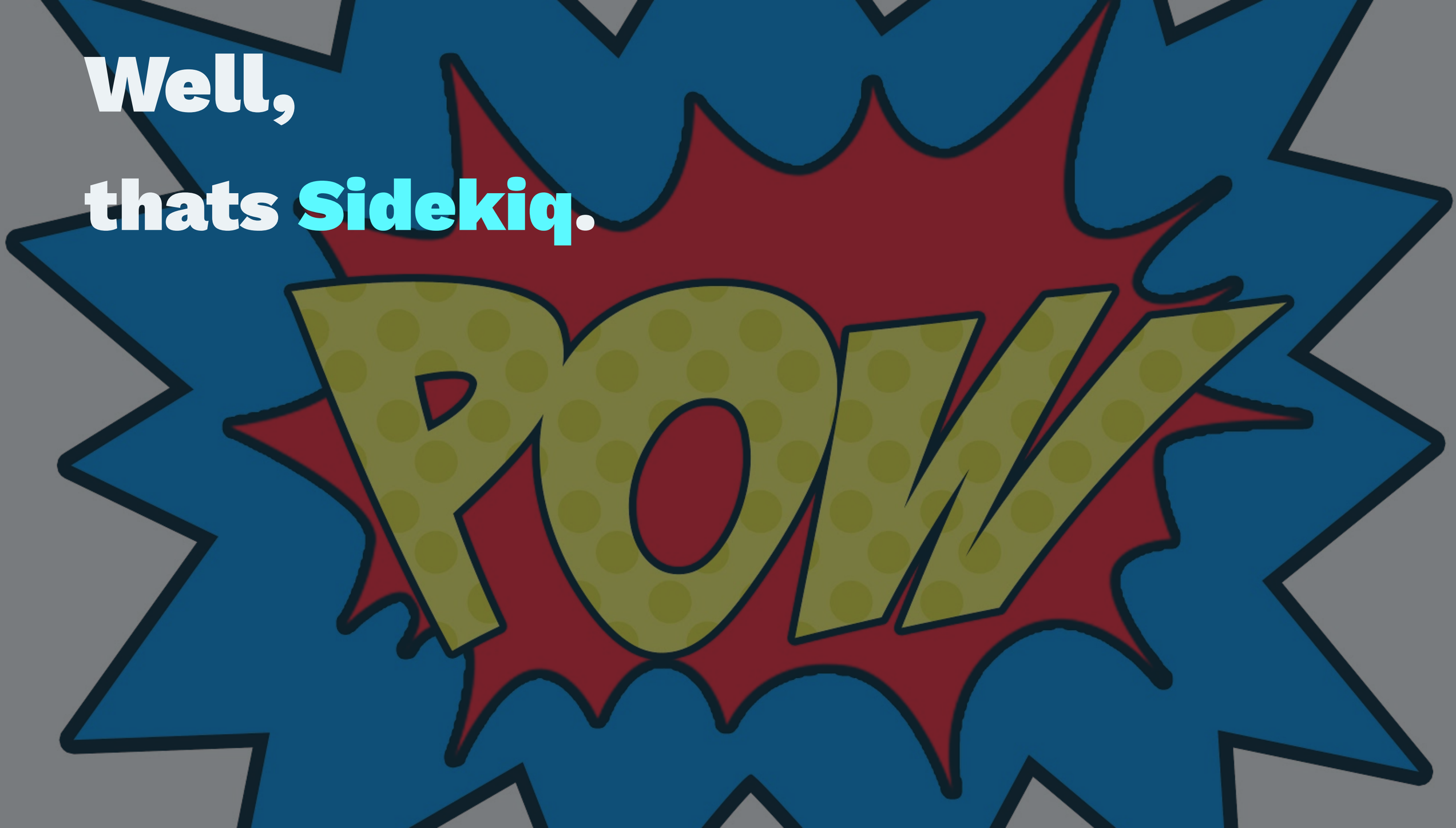
```
# clients
rpush "redis:jobs" {assignee: "Audience Member", task: "Greet your neighbour"}
rpush "redis:jobs" {assignee: "Audience Member", task: "Take a drink"}
rpush "redis:jobs" {assignee: "Audience Member", task: "Heckle louder"}
rpush "redis:jobs" {assignee: "Audience Member", task: "Do Garrett's Taxes"}

# sidekiq
while true
  job = lpop "redis:jobs"
  if job
    AudienceMember.perform(job)
  end
end
```

Background Job



Well,

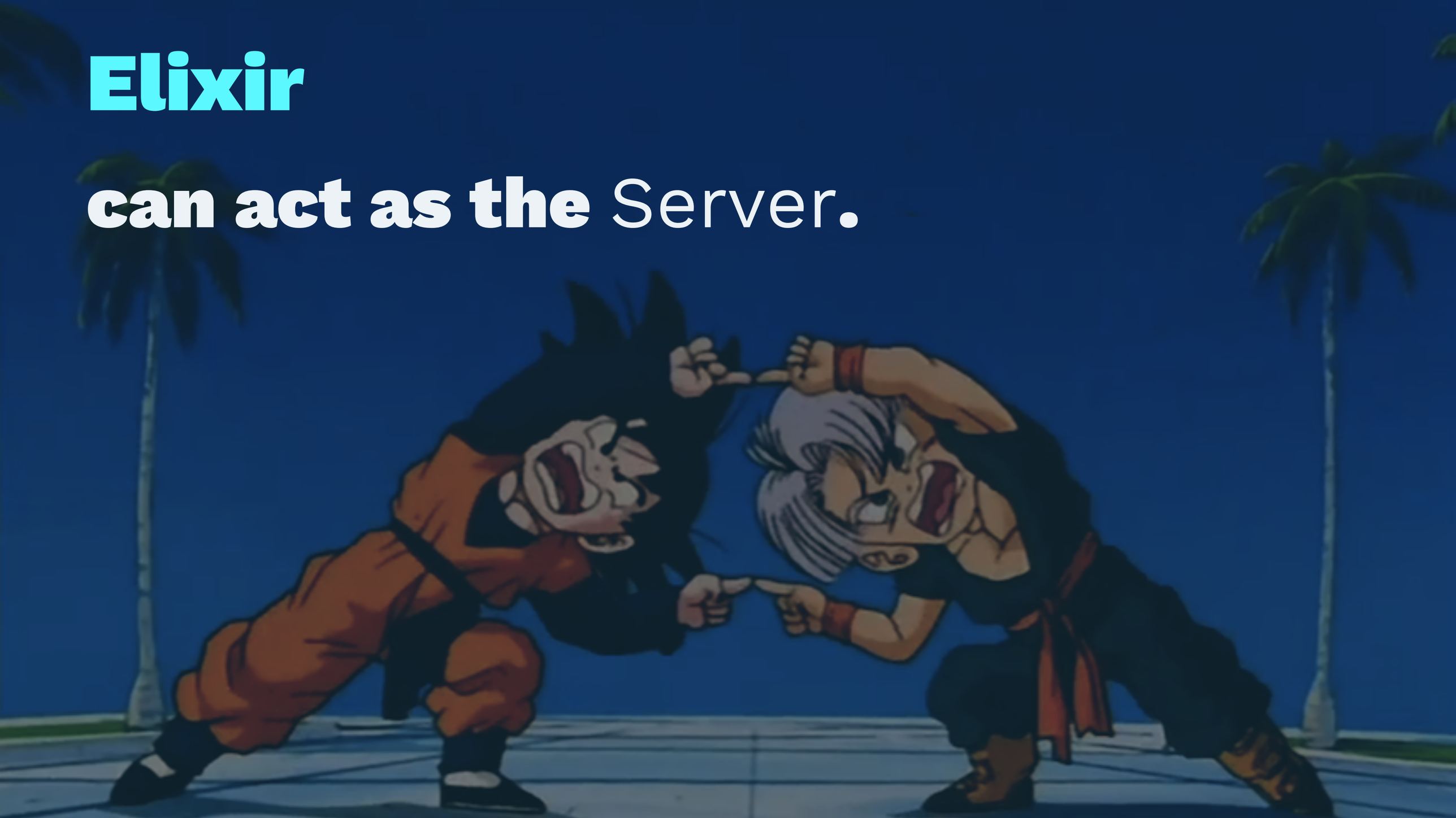
thats Sidekiq.



Introduce Elixir

Elixir

can act as the Server**.**



Exq

job processing library compatible with Sidekiq for Elixir

Configure Exq

```
config :exq,
  host: "localhost",
  port: 6379,
  namespace: "",
  concurrency: 10,
  queues: ["default"],
  poll_timeout: 200
```

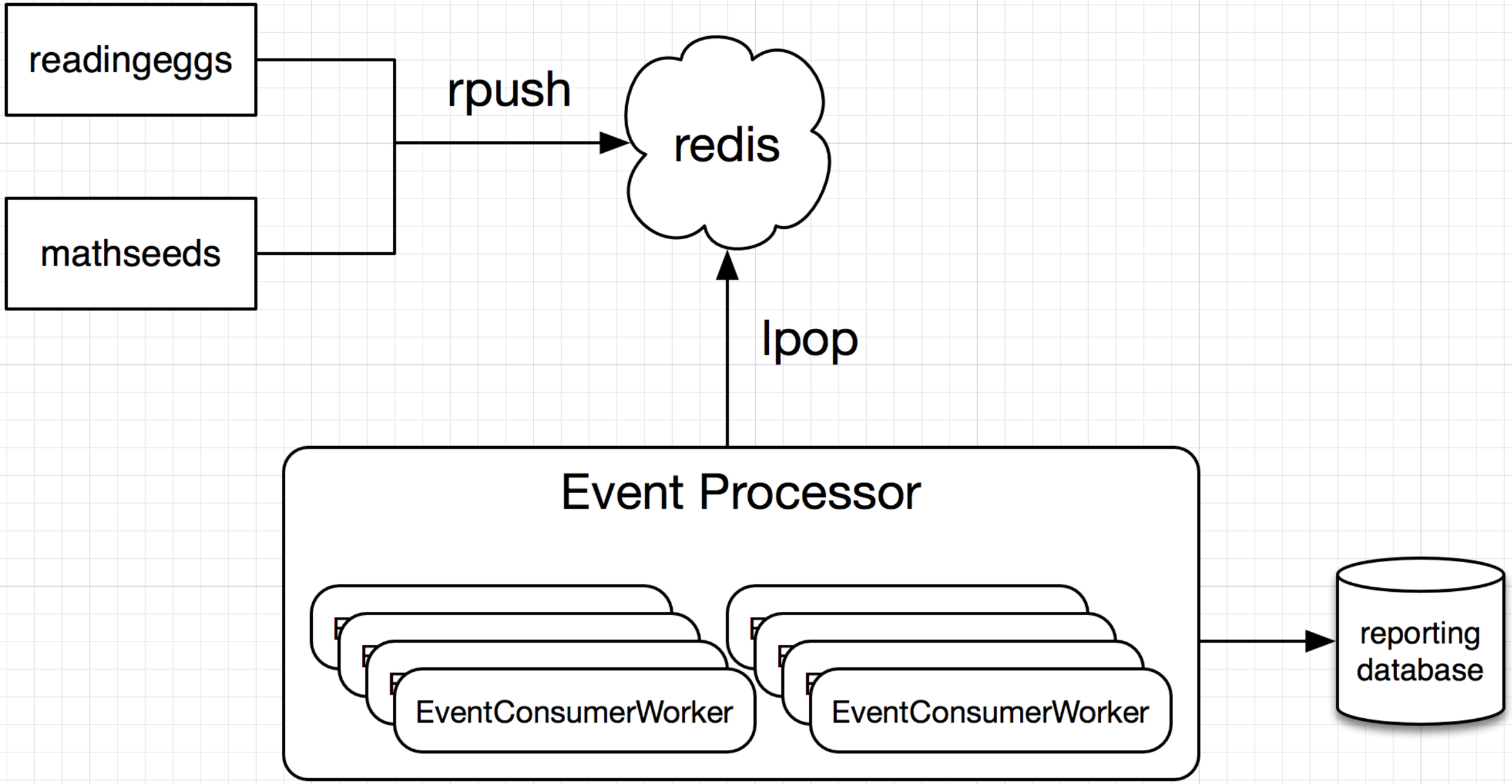
Ruby Sidekiq Server

```ruby
class MyWorker
  include Sidekiq::Worker

  def perform(hash)
    puts hash.inspect # => {abc: "123", simple_as: "drm"}
  end
end
```

Elixir Sidekiq Server

```
defmodule MyWorker do
  def perform(map) do
    IO.inspect(map) # => %{abc: "123", simple_as: "drm"}
  end
end
```

Demo Time


readingeggs
mathseeds
rpush
redis
lpop
Event Processor
EventConsumerWorker
EventConsumerWorker
reporting
database

Recap

— Redis is a Hash
— We push jobs into a list
— We pull jobs off the list
— We can process the jobs with Ruby or Elixir
— Exq is the Elixir Sidekiq

Thanks

@gogogarrett