
LUA AS
A SECOND LANGUAGE

@lmuzinic

LUKA MUŽINIĆ


Lua

WHERE CAN WE FIND LUA?

LIGHTROOM

MYSQL WORKBENCH

WIRESHARK

ASTERISK

WORLD OF WARCRAFT

LUA IS LIGHT IN FEATURES

DYNAMICALLY TYPED

FEW ATOMIC STRUCTURES

EVEN LESS COMPOUND STRUCTURES

FUNCTIONS AS FIRST CLASS CITIZENS

DATA TYPES

SCALAR

BOOLEAN

INTEGER

FLOAT

STRING

BOOLEAN

NUMBER

STRING

DATA TYPES

COMPOUND

ARRAY

OBJECT

CALLABLE

ITERABLE

TABLE

FUNCTION

DATA TYPES

SPECIAL

NULL

RESOURCE

NIL

USERDATA

THREAD

LUA IS LIGHT IN FEATURES

COERCION

CLOSURES

COROUTINES

GARBAGE COLLECTION

OBJECTS?

```php
class Player
{
    public $name = '';

    public function __construct($name)
    {
        $this->name = $name;
    }
}
```


```lua
Player = {
  name = ''
}

function Player:new(object)
  object = object or {};
  setmetatable(object, self);
  self.__index = self;

  return object;
end
```

INHERITANCE?

```php
class Npc extends Player
{

}
```

```lua
Npc = Player:new()

o = Npc:new{name="Ogre"}
```

NAMESPACES?

```php
namespace Player\Npc;
```

```lua
Player = {}
Player.Npc = {}
```

LUA STANDARD LIBRARY

BASE

PACKAGE

COROUTINE

STRING

UTF8

TABLE

MATH

IO

OS

DEBUG

ARRAY

array_change_key_case, array_chunk, array_column, array_combine, array_count_values, array_diff_assoc, array_diff_key, array_diff_uassoc, array_diff_ukey, array_diff, array_fill_keys, array_fill, array_filter, array_flip, array_intersect_assoc, array_intersect_key, array_intersect_uassoc, array_intersect_ukey, array_intersect, array_key_exists, array_key_first, array_key_last, array_keys, array_map, array_merge_recursive, array_merge, array_multisort, array_pad, array_pop, array_product, array_push, array_rand, array_reduce, array_replace_recursive, array_replace, array_reverse, array_search, array_shift, array_slice, array_splice, array_sum, array_udiff_assoc, array_udiff_uassoc, array_udiff, array_uintersect_assoc, array_uintersect_uassoc, array_uintersect, array_unique, array_unshift, array_values, array_walk_recursive, array_walk, array, arsort, asort, compact, count, current, end, extract, in_array, key_exists, key, krsort, ksort, list, natcasesort, natsort, next, pos, prev, range, reset, rsort, shuffle, sizeof, sort, uasort, uksort, usort,

TABLE

concat, insert, maxn, remove, sort

ECOSYSTEM

ECOSYSTEM

TIOBE INDEX

#8

#38

ECOSYSTEM

GITHUB REPOSITORIES

1M+

62K+

ECOSYSTEM

STACK OVERFLOW QUESTIONS

1.2M+

15K+

ECOSYSTEM

IDE

INTELLIJ

ECLIPSE

ZEROBRANE STUDIO

VISUAL STUDIO

EMACS

ATOM

SUBLIME

ECOSYSTEM

PACKAGE MANAGEMENT

LUAROCKS

LUADIST

ECOSYSTEM

PACKAGE MANAGEMENT

BUSTED

SPECL

LUASOCKET

REDIS-LUA

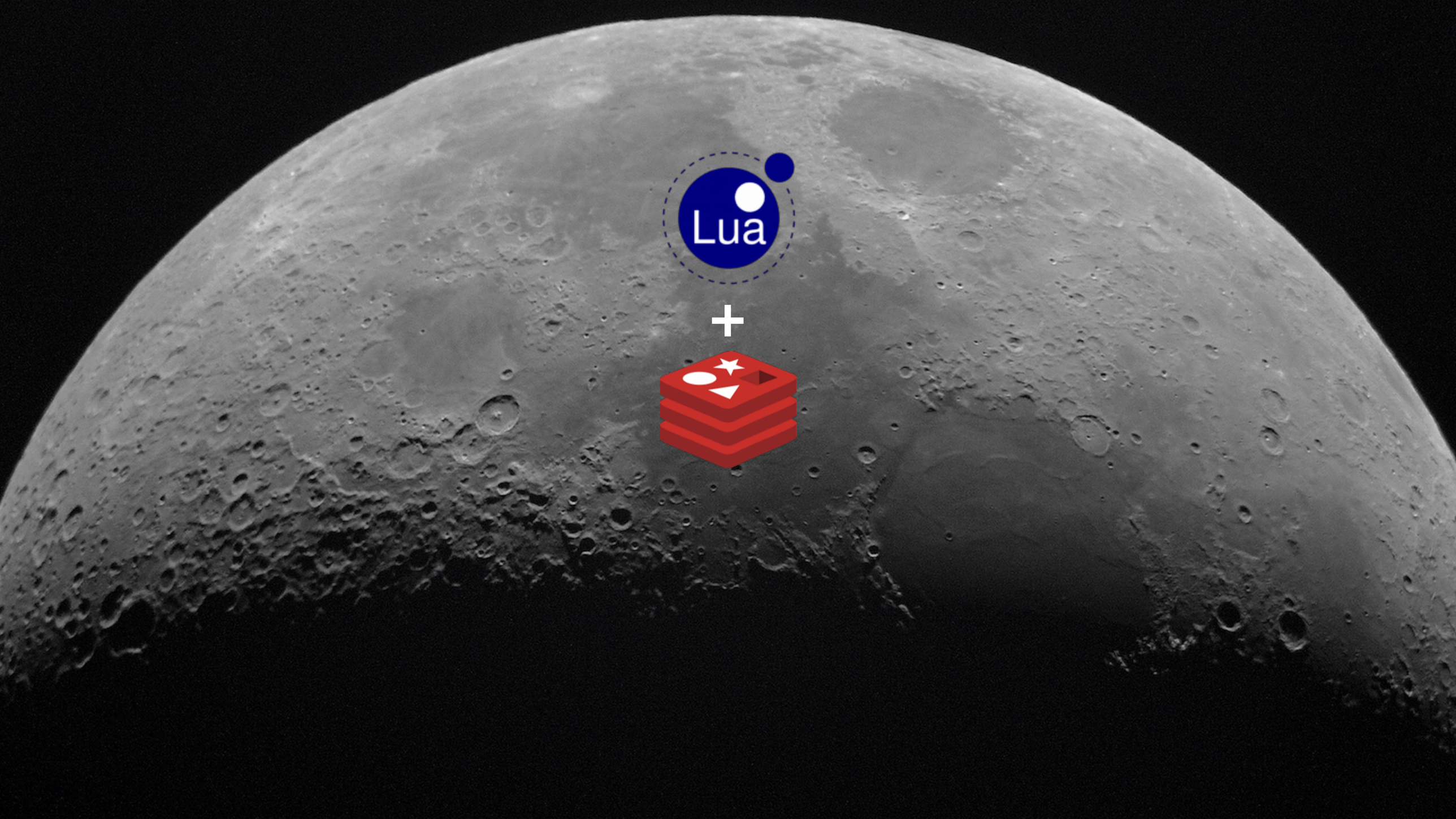
LUA-CJSON

UUID

LUASQL-MYSQL

LUA-GEOIP


Lua


Lua
+

REDIS

MOST LOVED DB

THIRD YEAR IN A ROW (via STACK OVERFLOW INSIGHTS)

REDIS

DATA STRUCTURE STORE

CACHE

DATABASE

MESSAGE BROKER

ATOMIC

LUA SCRIPTING

REDIS

LUA SCRIPTING

PROMISE OF ATOMICITY

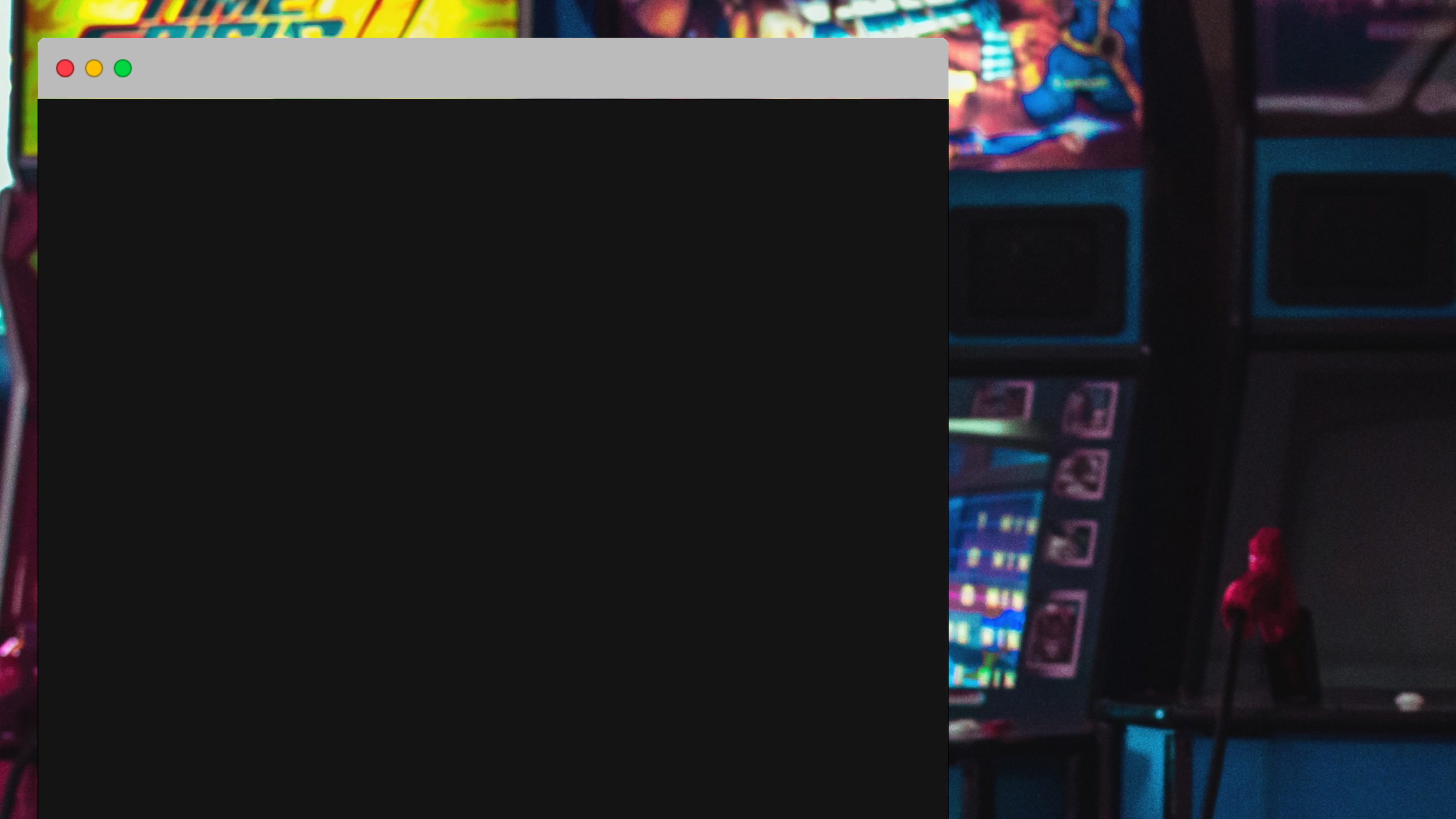
ACCESS REDIS ITSELF WITH `redis.call()` OR `redis.log()`

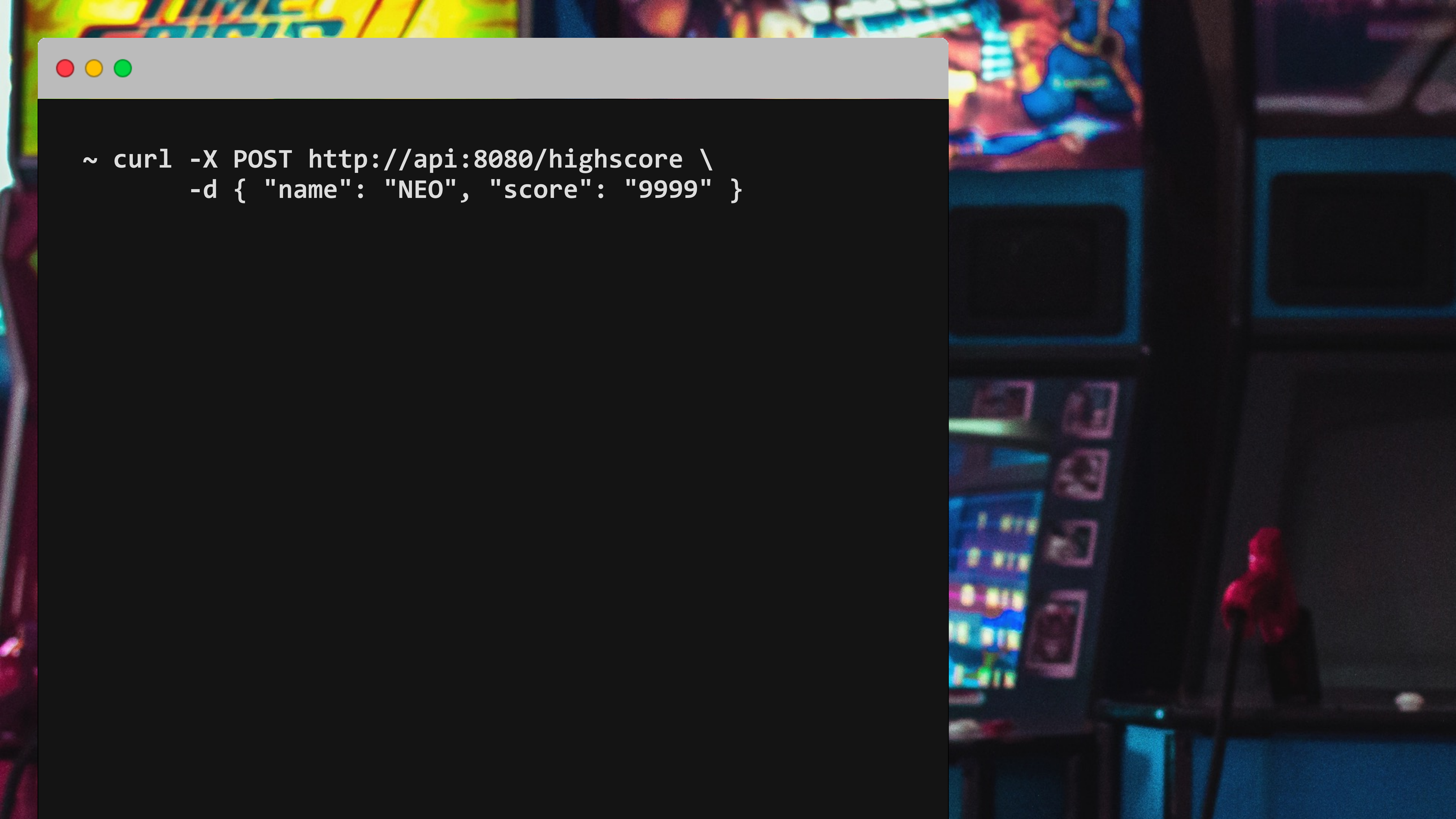
LEARN CONVERSION BETWEEN DATATYPES

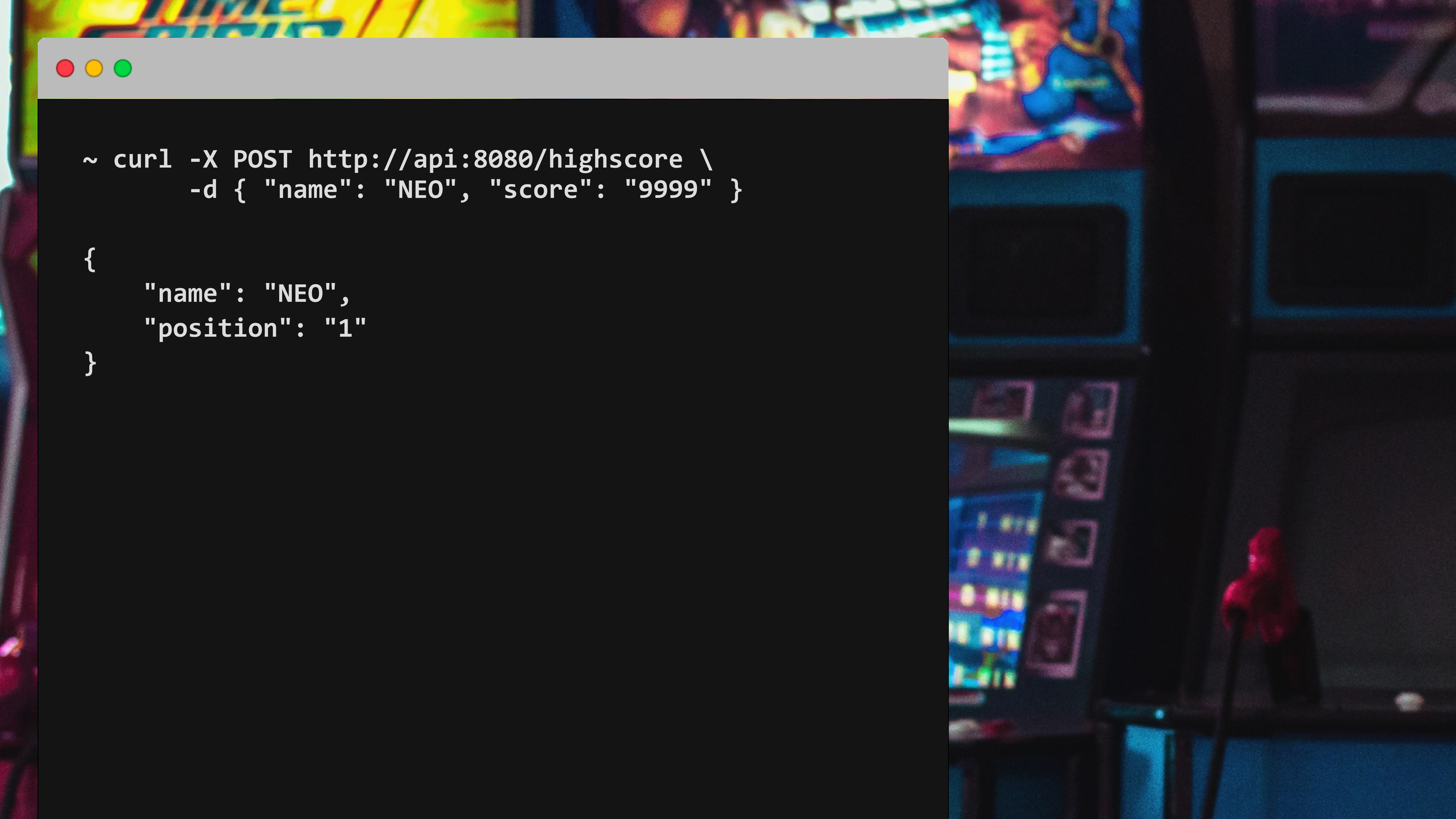
NO GLOBAL VALUES

BASE LIBRARIES + STRUCT, CJSON & CMSGPACK


TRY 2P COOP




~ curl -X POST http://api:8080/highscore \
-d { "name": "NEO", "score": "9999" }



```
~ curl -X POST http://api:8080/highscore \
       -d { "name": "NEO", "score": "9999" }

{
    "name": "NEO",
    "position": "1"
}
```

REDIS

HOW TO RUN LUA IN REDIS

REDIS

HOW TO RUN LUA IN REDIS

```
redis.call('zadd', KEYS[1], ARGV[1], ARGV[2])
```

REDIS

HOW TO RUN LUA IN REDIS

```
redis.call('zadd', KEYS[1], ARGV[1], ARGV[2])

local rank = redis.call('zrevrank', KEYS[1], ARGV[2])
local position = tostring(rank + 1)
```

REDIS

HOW TO RUN LUA IN REDIS

```
redis.call('zadd', KEYS[1], ARGV[1], ARGV[2])

local rank = redis.call('zrevrank', KEYS[1], ARGV[2])
local position = tostring(rank + 1)

return position
```

REDIS

HOW TO RUN LUA IN REDIS

```
redis.call('zadd', KEYS[1], ARGV[1], ARGV[2])

local rank = redis.call('zrevrank', KEYS[1], ARGV[2])
local position = tostring(rank + 1)

return position
```

REDIS

HOW TO RUN LUA IN REDIS

DON’T BE SCARED :)

REDIS

HOW TO RUN LUA IN REDIS

EVAL

REDIS

HOW TO RUN LUA IN REDIS

```php
$client = new Predis\Client($parameters, $options);

$position = $client->eval($lua, 1, 'highscores', 9999, 'NEO');
$position = $client->evalsha(sha1($lua), 1, 'highscores', 9999, 'NEO');
```

REDIS

HOW TO RUN LUA IN REDIS

```php
$client = new Predis\Client($parameters, $options);

$position = $client->eval($lua, 1, 'highscores', 9999, 'NEO');
$position = $client->evalsha(sha1($lua), 1, 'highscores', 9999, 'NEO');
```


```lua
redis.call('zadd', KEYS[1], ARGV[1], ARGV[2])

local rank = redis.call('zrevrank', KEYS[1], ARGV[2])
local position = tostring(rank + 1)

return position
```

REDIS

POSSIBLE USAGES

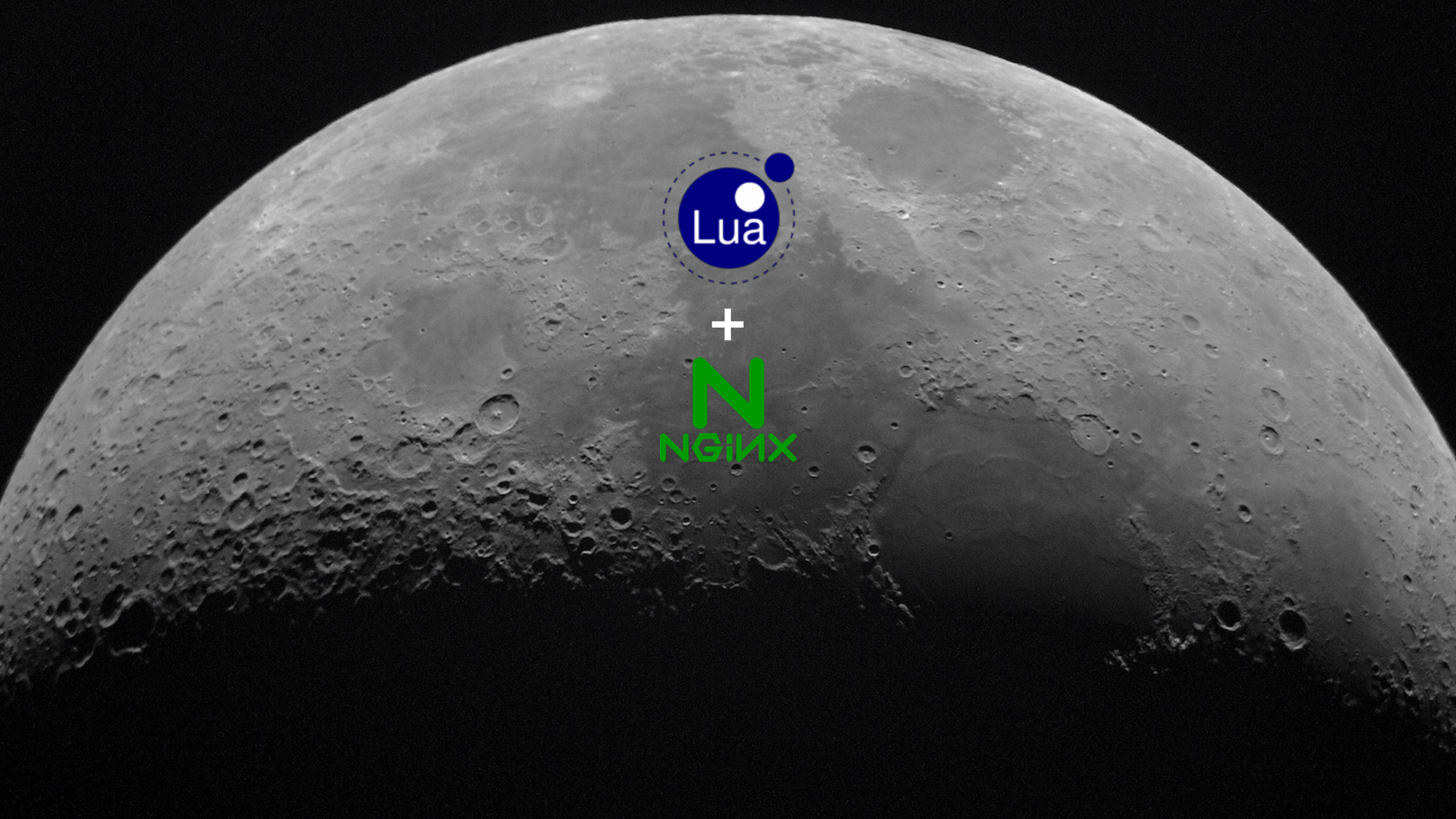
THINK MORE HELPER SCRIPTS, LESS BUILD A FRAMEWORK IN REDIS

SOMETHING + LOGS

ALL STATISTICS STUFF

TRANSACTION LIKE THINGS


Lua


Lua
+
NGINX

NGINX

WHERE CAN WE FIND IT?

PROXY

BALANCER

CACHE

WEB SERVER

NGINX

WHY DO WE USE IT?

VERY LOW MEMORY FOOTPRINT

EVENT DRIVEN

POOL OF WORKERS

NGINX

+ PHP-FPM

VERY LOW MEMORY FOOTPRINT

EVENT DRIVEN

POOL OF WORKERS

SHARED NOTHING ARCHITECTURE

SYNCHRONOUS CODE WITH BLOCKING I/O

NGINX

+ LUA

VERY LOW MEMORY FOOTPRINT

EVENT DRIVEN

POOL OF WORKERS

ALLOWS FOR SHARED DATA BETWEEN REQUESTS

SYNCHRONOUS CODE WITH NON-BLOCKING I/O

NGINX + LUA

GET STARTED WITH OPENRESTY

BUNDLE, NOT A FORK

NGINX + LUAJIT + MODULES

LEVERAGE EVENT MODEL AND NON-BLOCKING I/O

NUMEROUS MODULES

NGINX

PHASES (SIMPLIFIED)

REQUEST

REWRITE

ACCESS

CONTENT

LOG

RESPONSE

NGINX + LUA

PHASES (SIMPLIFIED)

REWRITE_BY_LUA_BLOCK

ACCESS_BY_LUA_BLOCK

CONTENT_BY_LUA_BLOCK

LOG_BY_LUA_BLOCK

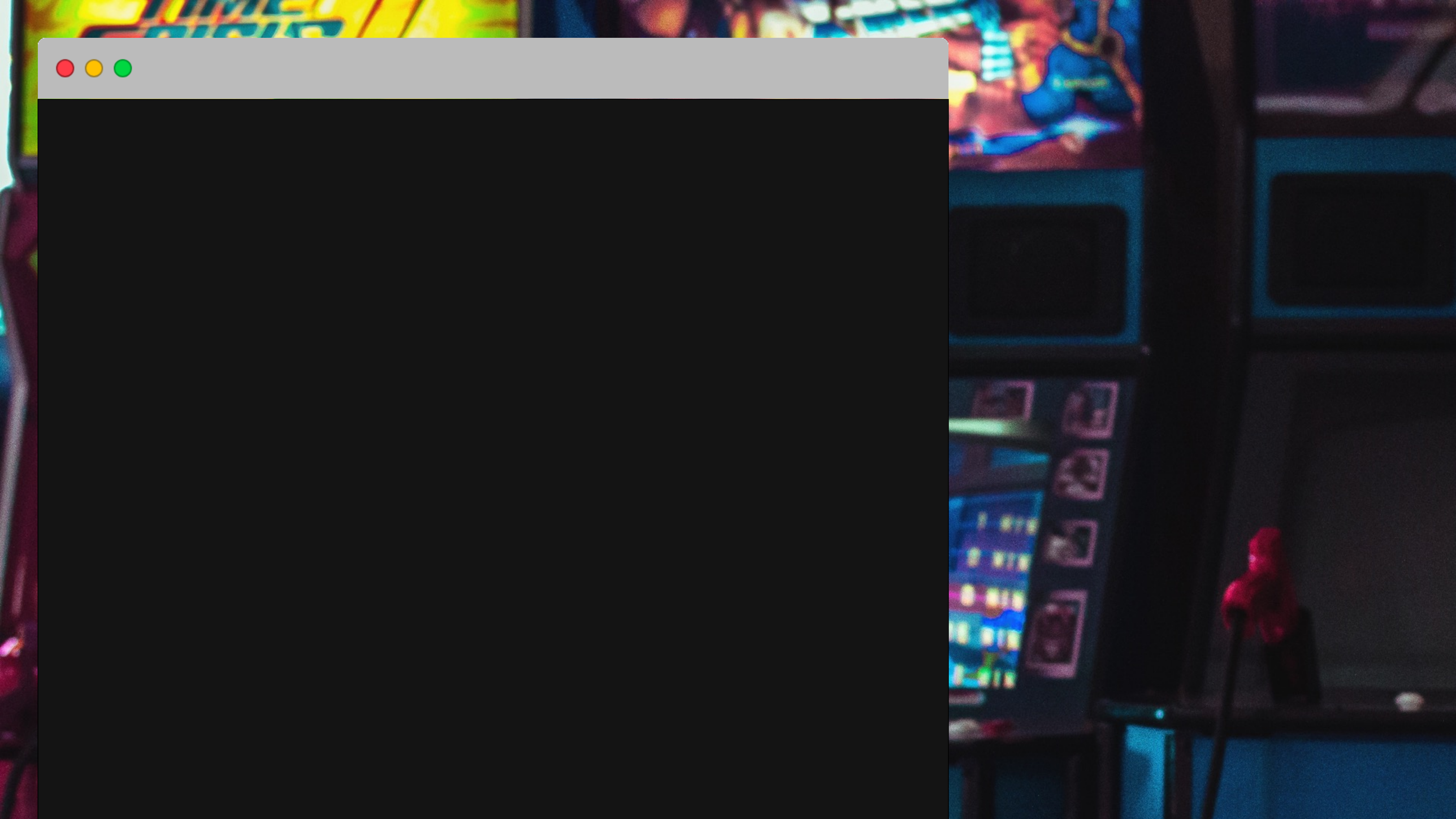
REWRITE_BY_LUA_FILE

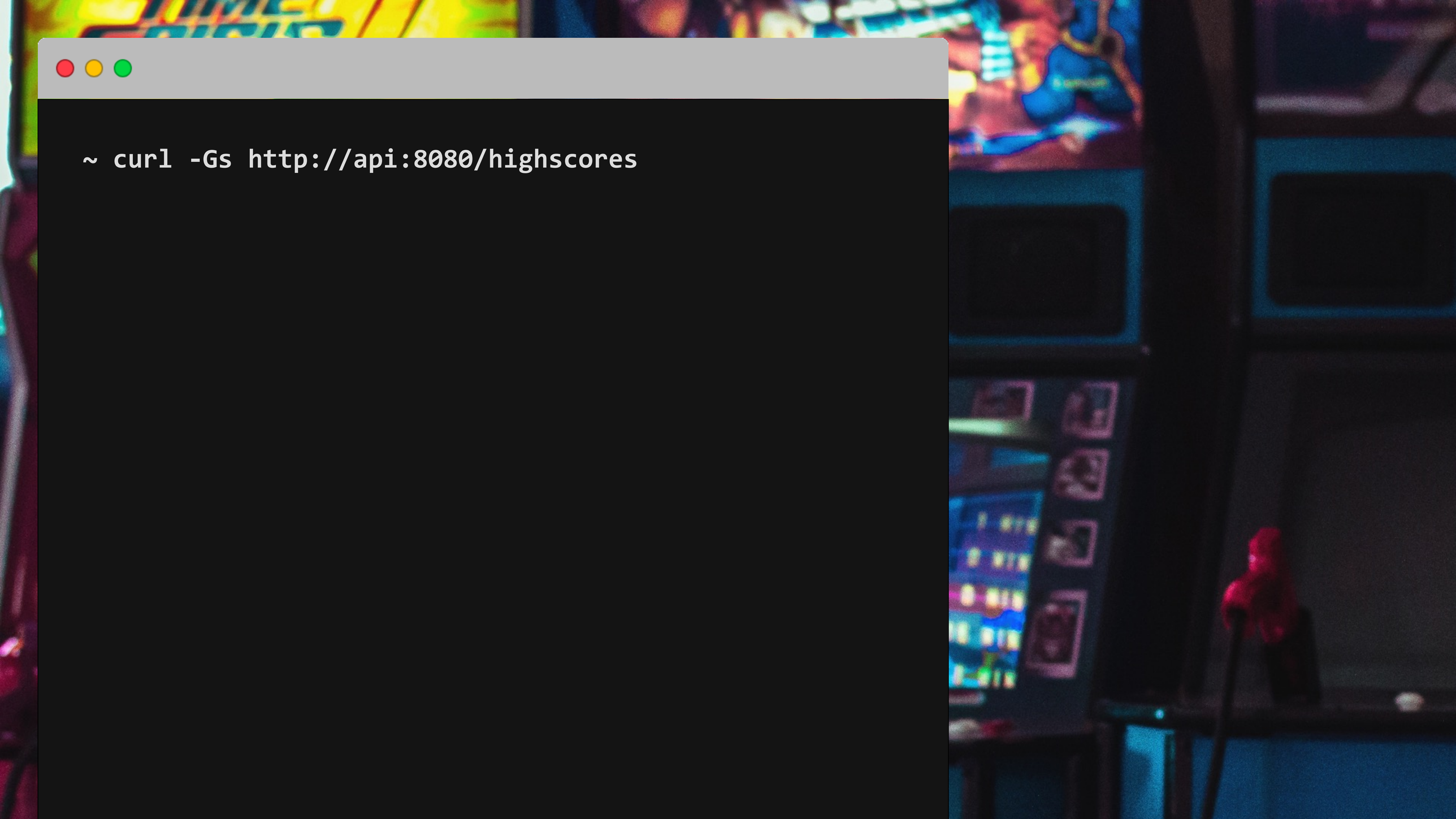
ACCESS_BY_LUA_FILE

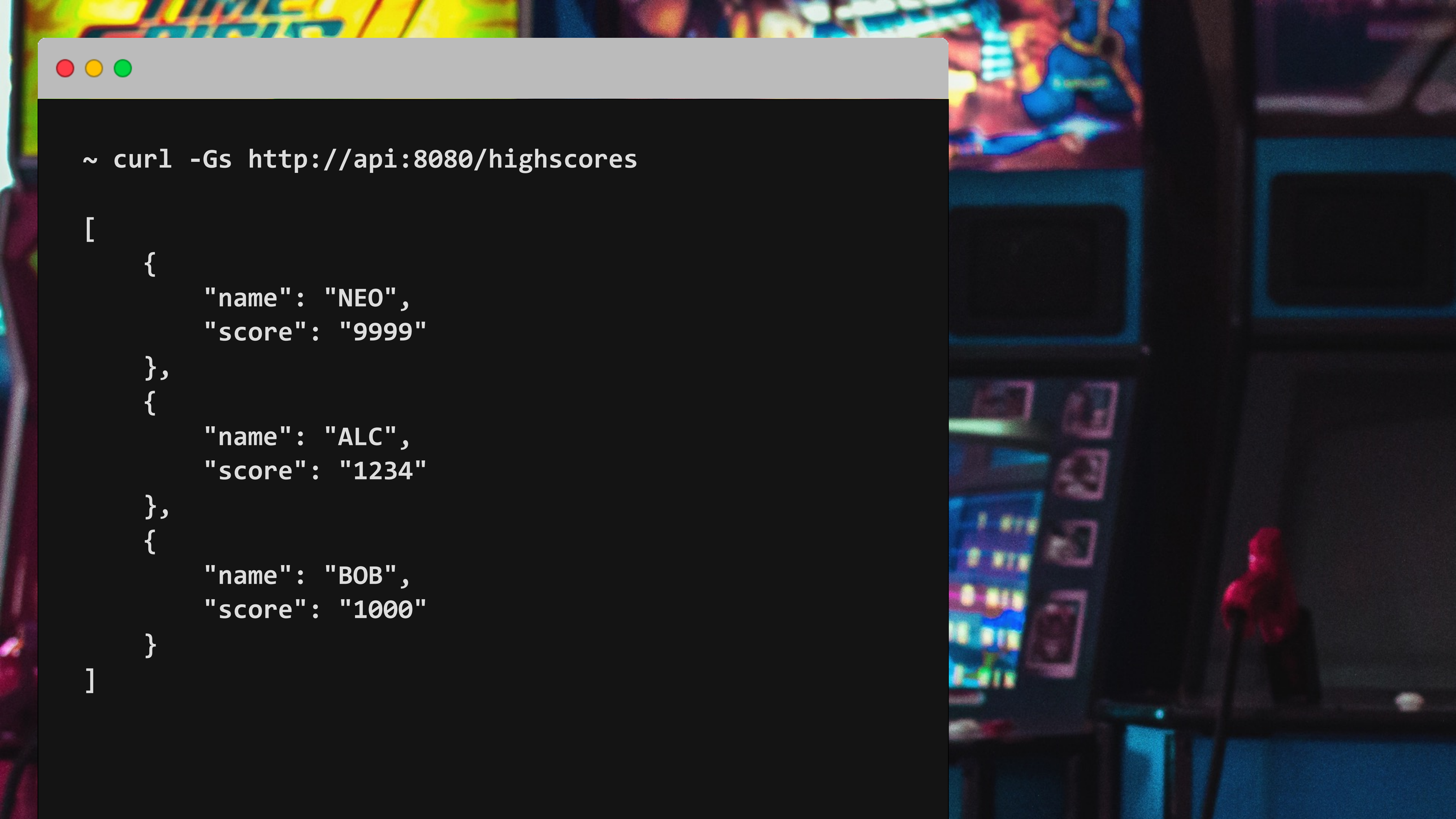
CONTENT_BY_LUA_FILE

LOG_BY_LUA_FILE


TRY 2P COOP




~ curl -Gs http://api:8080/highscores


~ curl -Gs http://api:8080/highscores
[
{
"name": "NEO",
"score": "9999"
},
{
"name": "ALC",
"score": "1234"
},
{
"name": "BOB",
"score": "1000"
}
]

NGINX + LUA

CONFIGURATION

```
location /highscores {
    default_type application/json;
    content_by_lua_block {
        # ...
    }
}
```

NGINX + LUA

CONFIGURATION

NGINX + LUA

CONFIGURATION

```
local cjson = require "cjson"
local redis = require "resty.redis"
```

NGINX + LUA

CONFIGURATION

```
local cjson = require "cjson"
local redis = require "resty.redis"
local red = redis:new()
local ok, err = red:connect("127.0.0.1", 6379)
```

NGINX + LUA

CONFIGURATION

```
local cjson = require "cjson"
local redis = require "resty.redis"
local red = redis:new()
local ok, err = red:connect("127.0.0.1", 6379)

local highscores, err = red:zrevrange("highscores", 0, 10, "withscores")
```

NGINX + LUA

CONFIGURATION

```
local cjson = require "cjson"
local redis = require "resty.redis"
local red = redis:new()
local ok, err = red:connect("127.0.0.1", 6379)

local highscores, err = red:zrevrange("highscores", 0, 10, "withscores")

local response = {}
for i = 1, #highscores, 2 do
    local player = { name = highscores[i], score = highscores[i + 1] }

    table.insert(response, player)
end
```

NGINX + LUA

CONFIGURATION

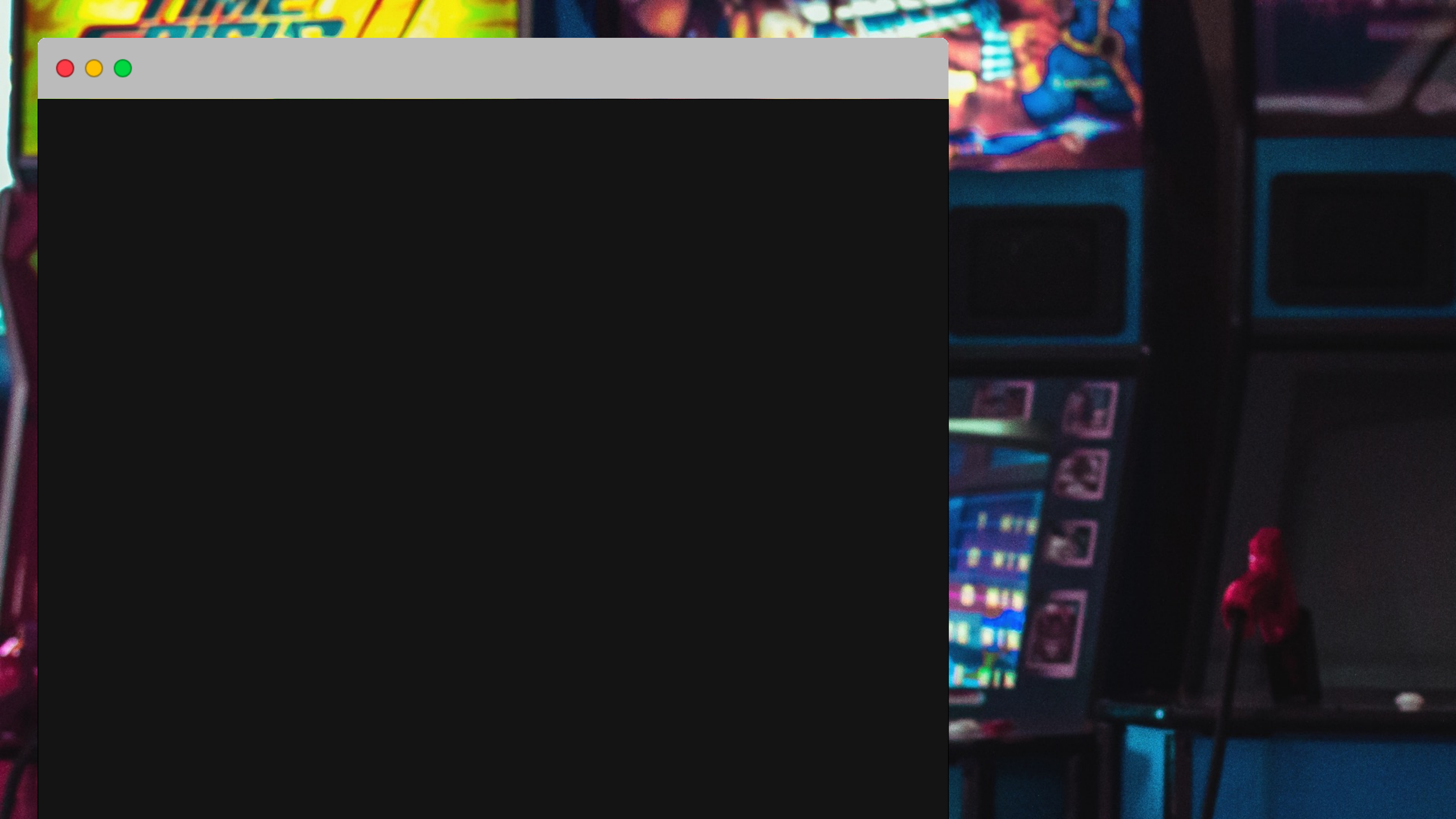
```lua
local cjson = require "cjson"
local redis = require "resty.redis"
local red = redis:new()
local ok, err = red:connect("127.0.0.1", 6379)

local highscores, err = red:zrevrange("highscores", 0, 10, "withscores")

local response = {}
for i = 1, #highscores, 2 do
    local player = { name = highscores[i], score = highscores[i + 1] }

    table.insert(response, player)
end

ngx.say(cjson.encode(response))
```

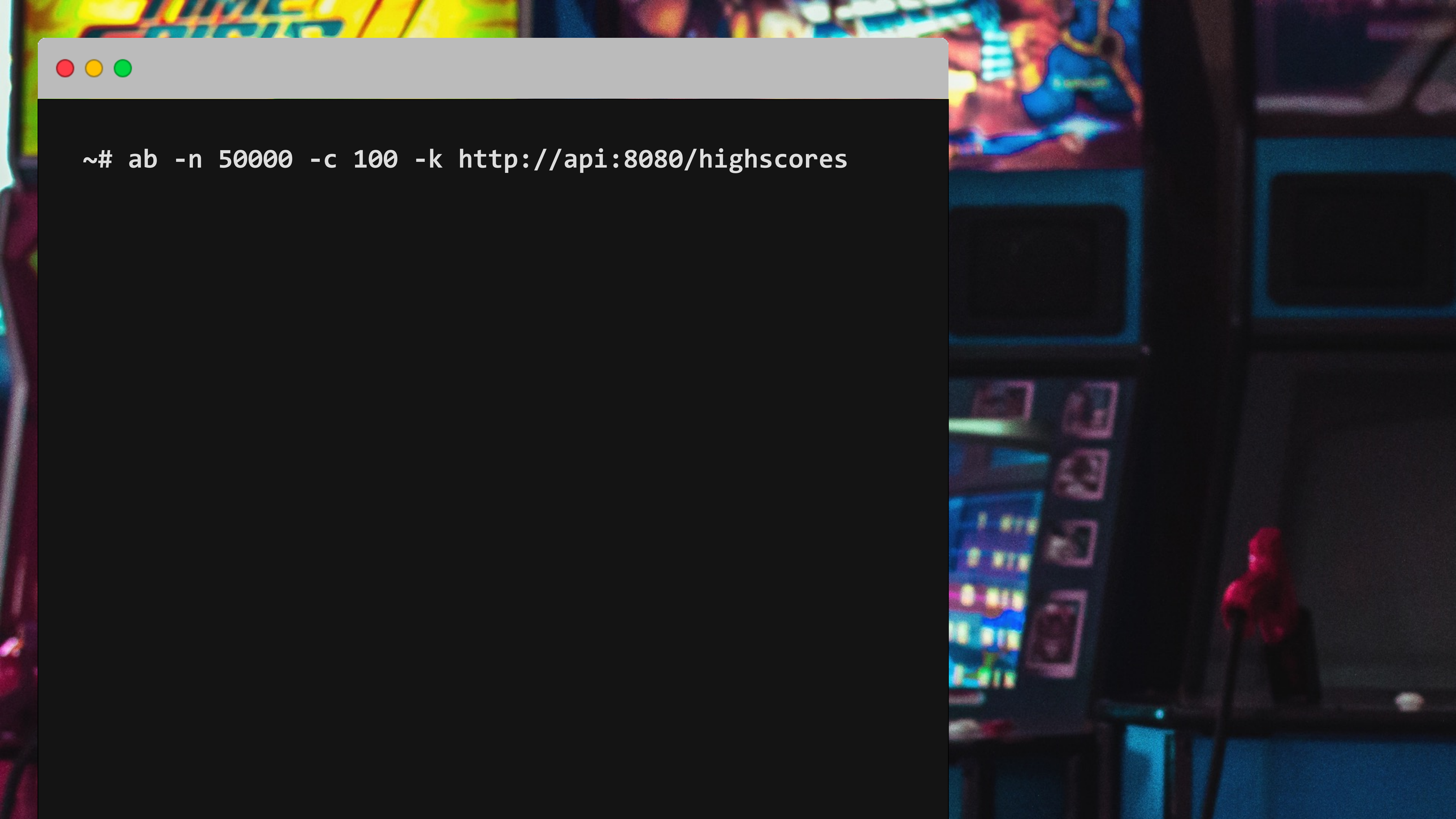
NGINX + LUA

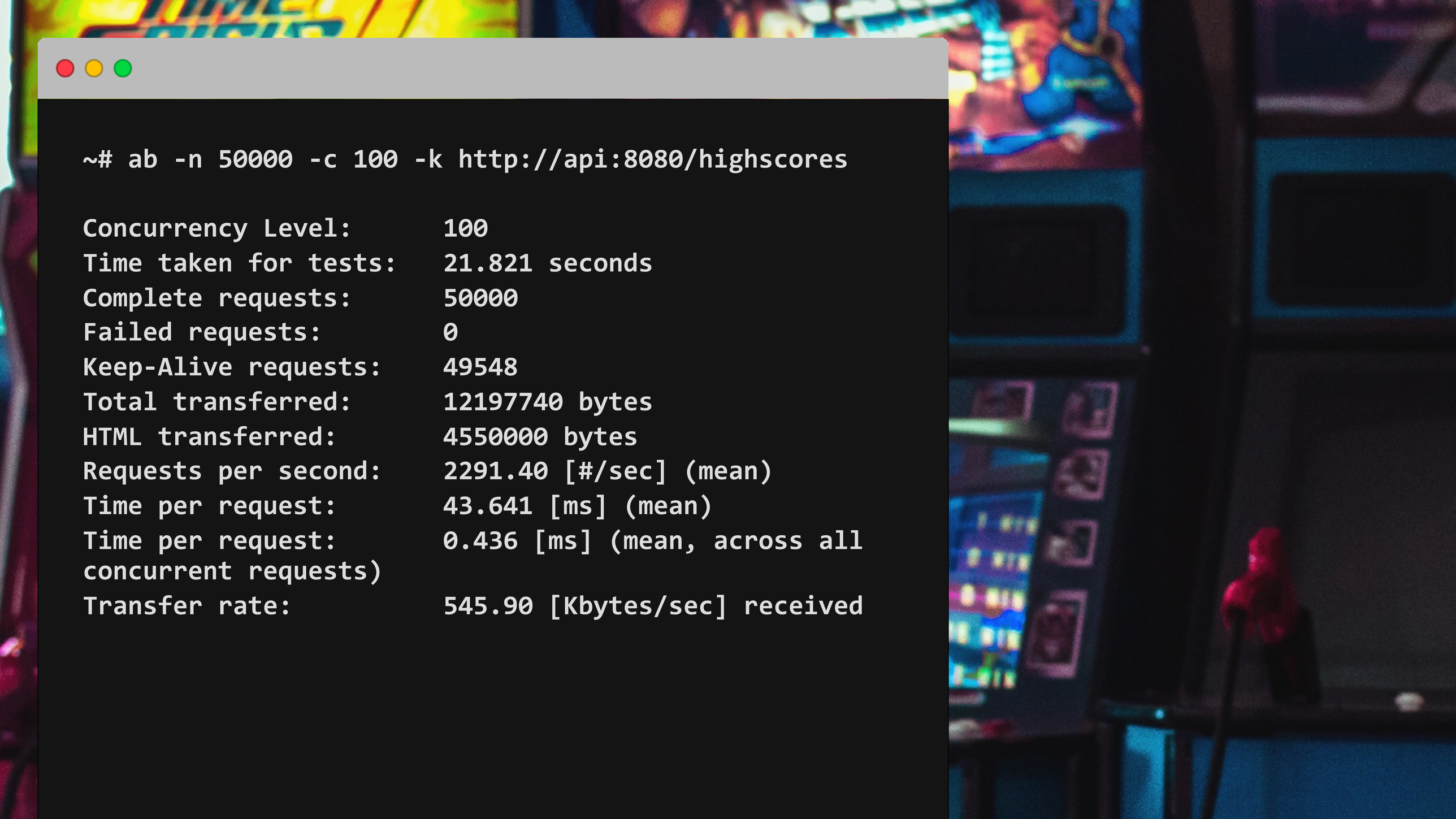
CONFIGURATION

```lua
local cjson = require "cjson"
local redis = require "resty.redis"
local red = redis:new()
local ok, err = red:connect("127.0.0.1", 6379)

local highscores, err = red:zrevrange("highscores", 0, 10, "withscores")

local response = {}
for i = 1, #highscores, 2 do
    local player = { name = highscores[i], score = highscores[i + 1] }

    table.insert(response, player)
end

ngx.say(cjson.encode(response))
```

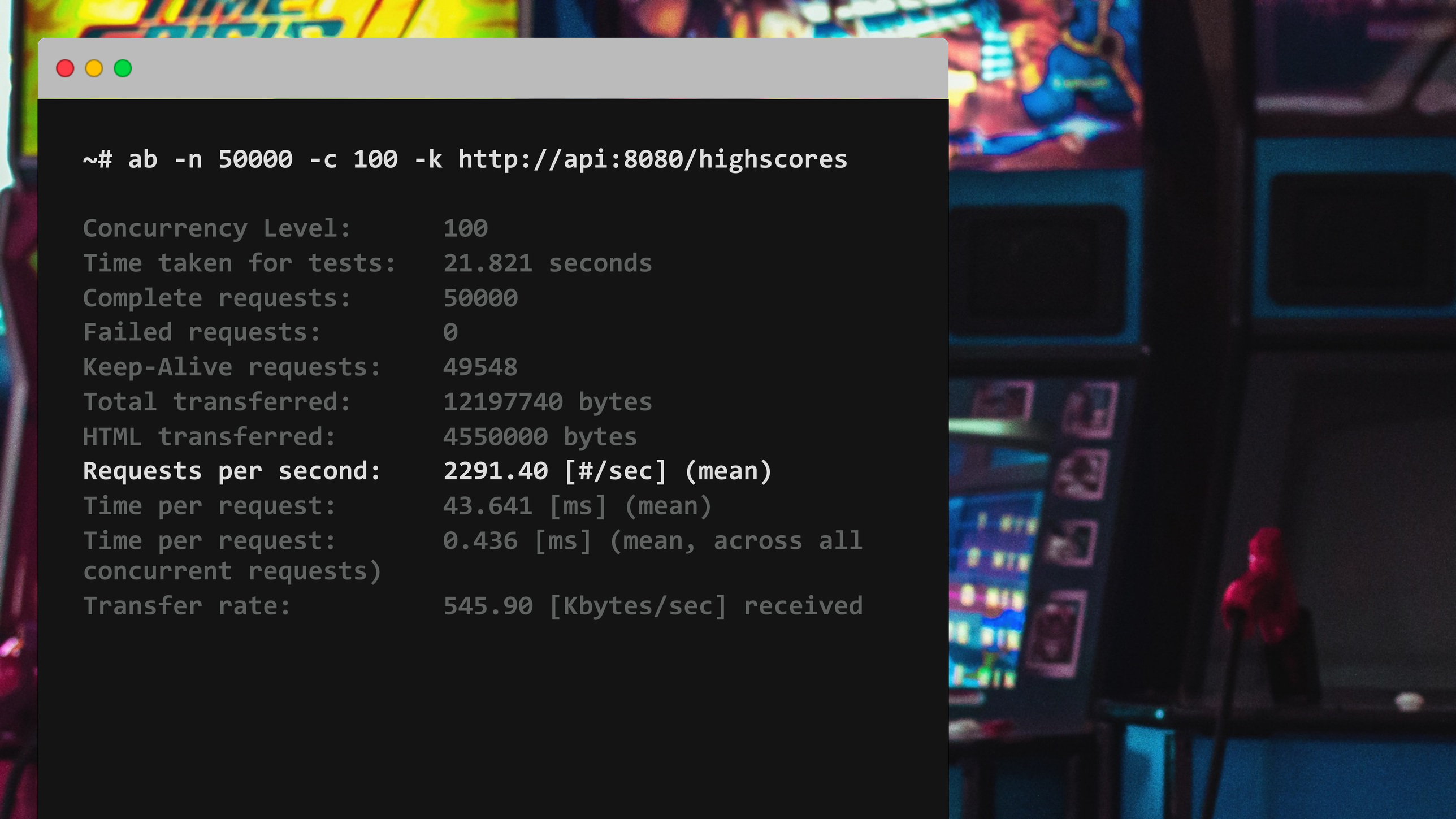

TRY 2P COOP



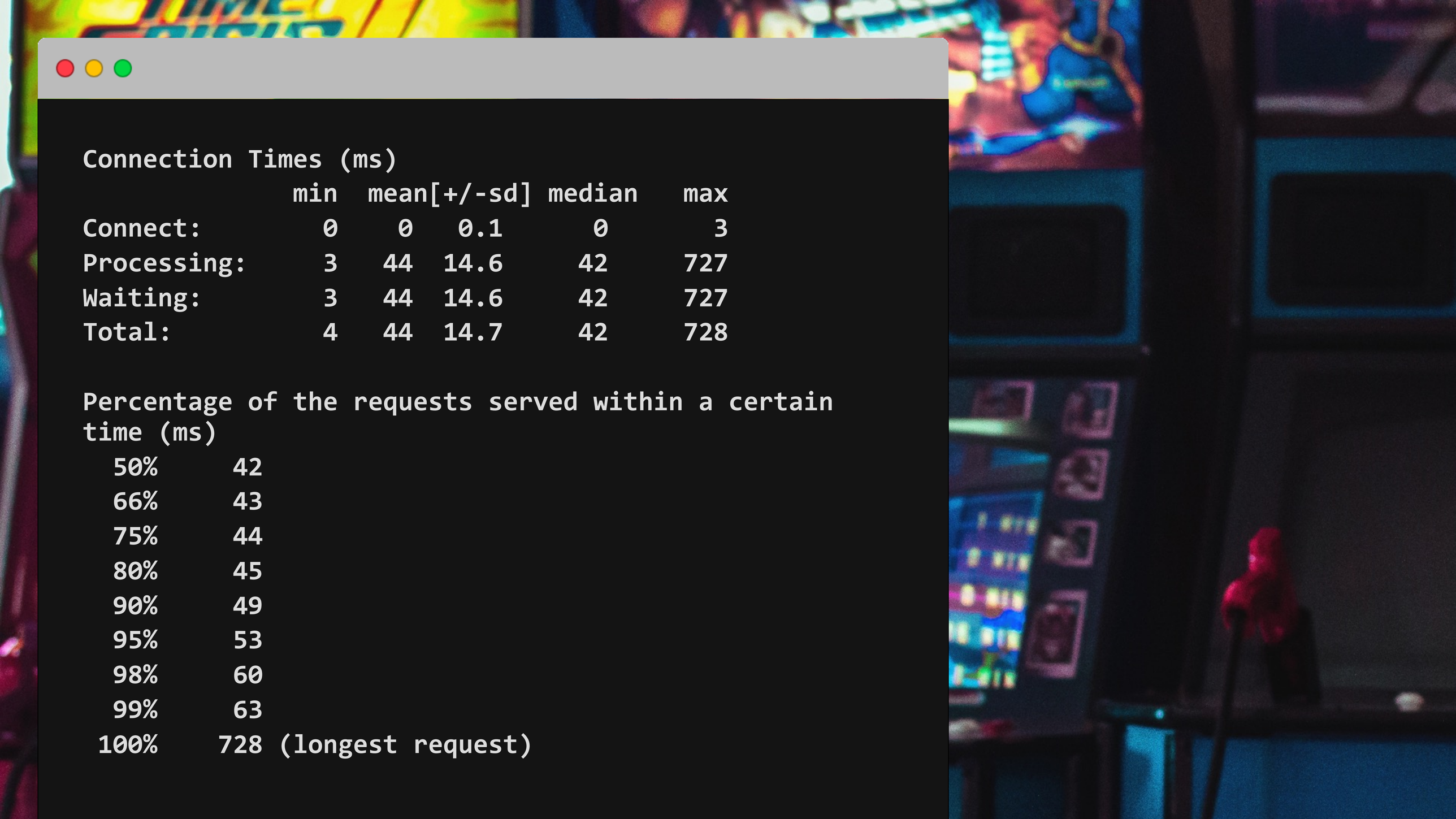

~# ab -n 50000 -c 100 -k http://api:8080/highscores



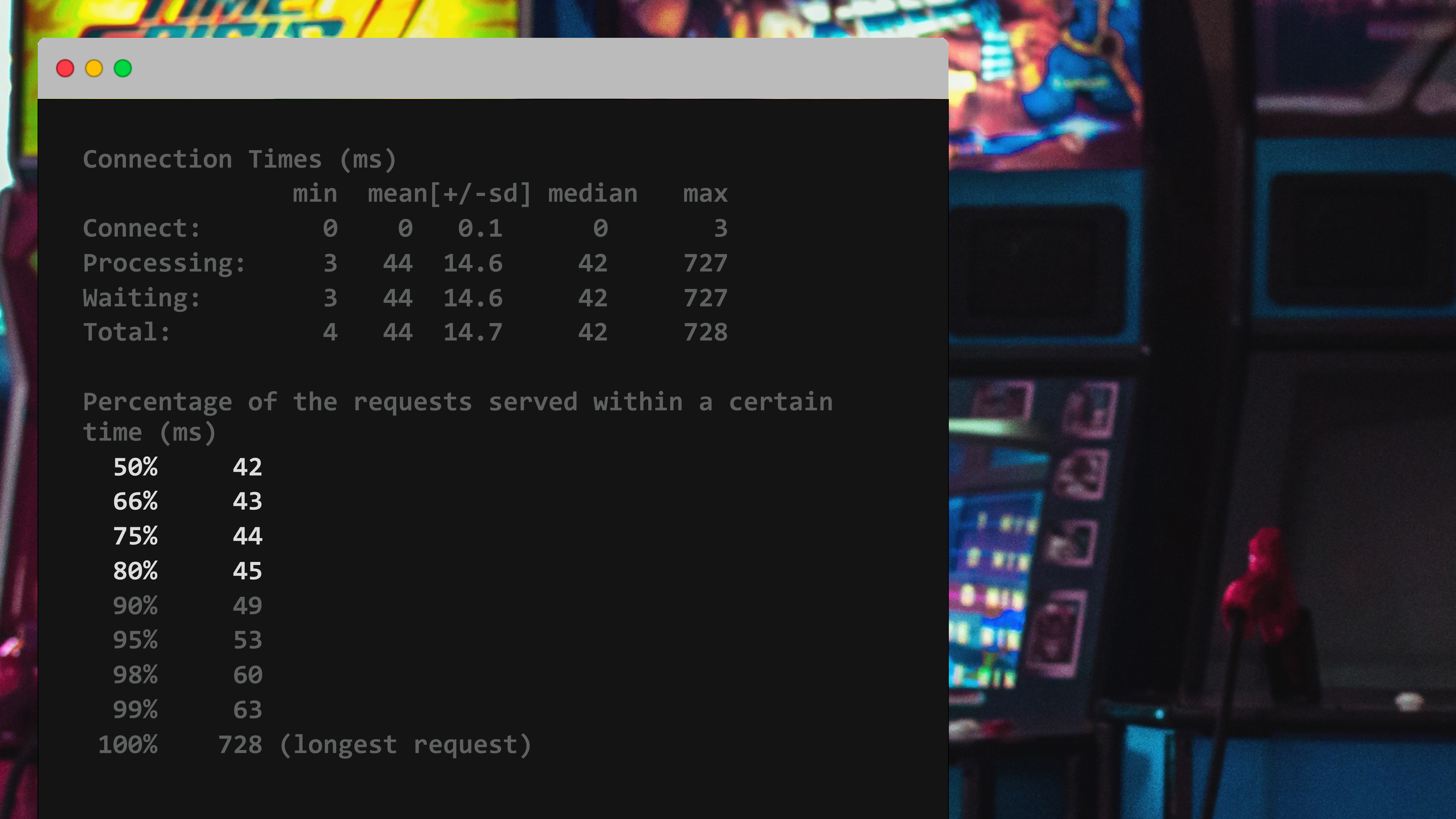
```
~# ab -n 50000 -c 100 -k http://api:8080/highscores

Concurrency Level:      100
Time taken for tests:   21.821 seconds
Complete requests:      50000
Failed requests:        0
Keep-Alive requests:    49548
Total transferred:      12197740 bytes
HTML transferred:       4550000 bytes
Requests per second:    2291.40 [#/sec] (mean)
Time per request:       43.641 [ms] (mean)
Time per request:       0.436 [ms] (mean, across all
concurrent requests)
Transfer rate:          545.90 [Kbytes/sec] received
```



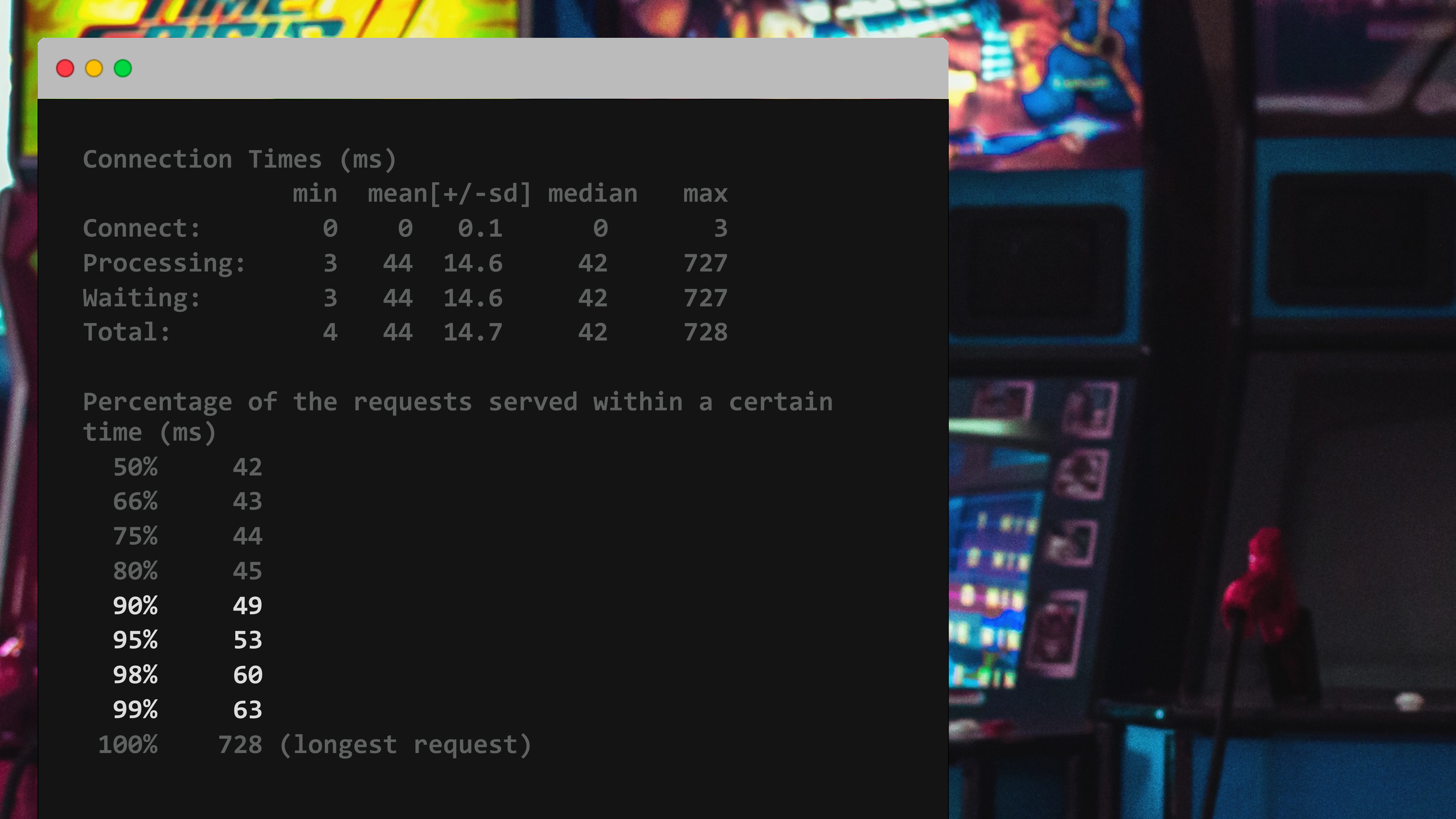
```
~# ab -n 50000 -c 100 -k http://api:8080/highscores

Concurrency Level:      100
Time taken for tests:   21.821 seconds
Complete requests:      50000
Failed requests:        0
Keep-Alive requests:    49548
Total transferred:      12197740 bytes
HTML transferred:       4550000 bytes
Requests per second:    2291.40 [#/sec] (mean)
Time per request:       43.641 [ms] (mean)
Time per request:       0.436 [ms] (mean, across all
concurrent requests)
Transfer rate:          545.90 [Kbytes/sec] received
```



```
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       3
Processing:     3   44  14.6     42     727
Waiting:        3   44  14.6     42     727
Total:          4   44  14.7     42     728

Percentage of the requests served within a certain
time (ms)
  50%     42
  66%     43
  75%     44
  80%     45
  90%     49
  95%     53
  98%     60
  99%     63
 100%    728 (longest request)
```



```
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       3
Processing:     3   44  14.6     42     727
Waiting:        3   44  14.6     42     727
Total:          4   44  14.7     42     728

Percentage of the requests served within a certain
time (ms)
  50%     42
  66%     43
  75%     44
  80%     45
  90%     49
  95%     53
  98%     60
  99%     63
 100%    728 (longest request)
```



```
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       3
Processing:     3   44  14.6     42     727
Waiting:        3   44  14.6     42     727
Total:          4   44  14.7     42     728

Percentage of the requests served within a certain
time (ms)
  50%     42
  66%     43
  75%     44
  80%     45
  90%     49
  95%     53
  98%     60
  99%     63
 100%    728 (longest request)
```


TRY 2P COOP

NGINX + LUA

USAGES

ACCESS CONTROL & SECURITY CHECKS

REPONSE HEADER MANIPULATION

FETCH BACKEND INFO DIRECTLY FROM MYSQL OR REDIS

CACHE

FORMAT LOG & ADD MORE INFOMATION


HOMEWORK

THANK YOU

Luka Muzinic
@lmuzinic

luka.muzinic.net/talks

Feedback
joind.in/talk/b8183

Photos by Nicolas Thomas and zhan zhang on Unsplash