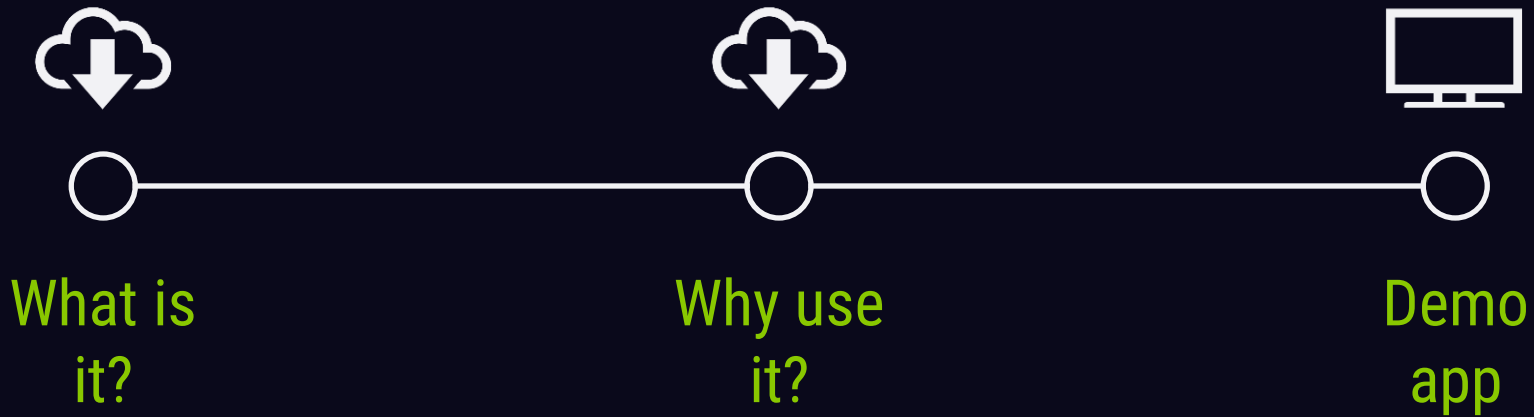
aws serverless apps

aws serverless apps
introduction



what is serverless?

parts of serverless apps

JavaScript

All of the dynamic behavior is provided by JavaScript running in the user's browser. The JavaScript itself is static, though.

API

Data is written to and retrieved from an API hosted by a cloud provider. In our case, that provider is AWS API Gateway.

Markup

Finally, the HTML markup ties the whole thing together. The markup is static and cacheable, so the initial page load for your users is fast.

how do they work together?

a user comes to your site and loads static HTML, CSS, and JavaScript

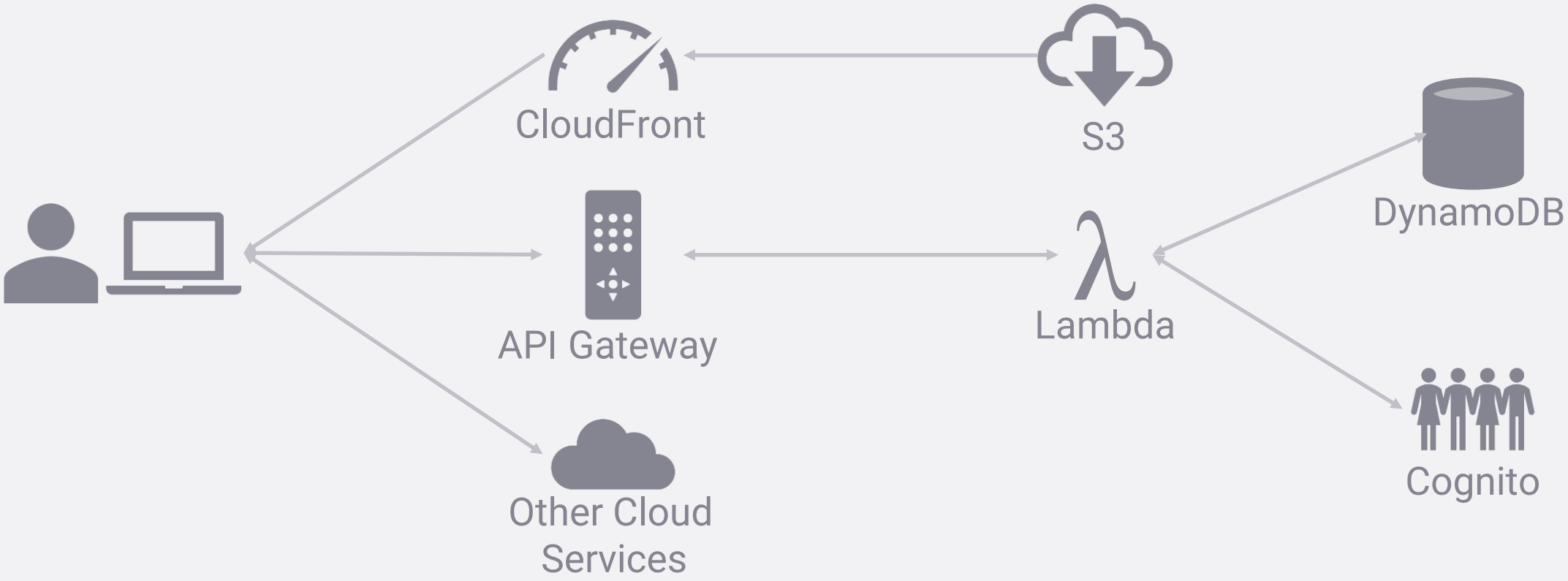
static parts of the page are in HTML while dynamic parts of the page are added by JavaScript

JavaScript retrieve dynamic data from the API

JavaScript writes data to the API

JavaScript handles some aspects of navigation

maybe a picture?



why go serverless?

why not just run a server?

server management (patches, monitoring, hardware failures, etc.)

servers can be cheap, but scaling gets expensive really fast

you don’t pay for processing time you don’t use

easier to split up development between front-end and back-end

aren’t s3, lambda, etc. servers?

yes, but…

they’re not yours

you’re not the only one on them

amazon has people with pagers to keep them working

good and bad use cases

good:
dynamic applications with lots of user interaction
most of the content in the UI is specific to the user

bad:
large applications with lots of data that would have to be loaded
anything where the time to load markup, JS, and data would create a very slow UI

demo app

getting started

getting started

sign up for AWS

Sign up at: https://portal.aws.amazon.com/billing/signup

Free tier information: https://aws.amazon.com/free/

Services we will use:
- S3
- API Gateway
- Lambda
- DynamoDB
- Cognito
- CloudFront
- IAM
- CloudWatch

understanding S3

S3 basics

S3 = Simple Storage Service

Key-Blob store

Eventually consistent

Extremely durable

S3 is not...

a file system

one more thing...

AWS policies define access control for:

- users
- other AWS services
- resources

make life easier

```
{
    "Version": "2012-10-17",
    "Id": "Policy1497053408897",
    "Statement": [
        {
            "Sid": "Stmt1497053406813",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::<your bucket name>/*"
        }
    ]
}
```

JavaScript syntax

JavaScript syntax

JavaScript is a C-style language in most respects:
- control structures
- variable naming rules
- accessing member values of an object (e.g. foo.bar)

JavaScript is not always C-like:
- variables have no type
- functions are objects
- ===

JavaScript syntax oddities

== - equivalent values
=== - equal values of the same type

Example:
1 == “1” is true
1 === “1” is false

JavaScript syntax oddities

null vs. undefined

null – an object of type null

undefined – a value of type undefined

JavaScript declarations

```
var foo = ‘bar’;
var baz = “bar”;

var object = {
    p1: 1,
    p2: 2,
    ‘this needs to be quoted’: ‘Hello, world!’
};

object.p1 is the same as object[‘p1’]
```

JQuery

$ is JQuery

$.get()
$.post()

$('.class') – CSS class selector
$('#id') – ID selector
$('div') – tag selector

Useful references

JavaScript: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

JQuery: http://api.jquery.com/

JavaScript parallelism

classic parallelism

Scenario:
You need to make two independent HTTP requests as fast as possible

Solution:
Create two threads to make the requests
Use some sort of synchronization to determine when they're done
Keep doing what you need to do

is that bad?

No, it’s perfectly normal!

Good:
Assuming you have enough processors and memory, you can be as efficient as possible.
The threads are not lies. They’re real, and you can make them do what you want.
It’s very powerful.

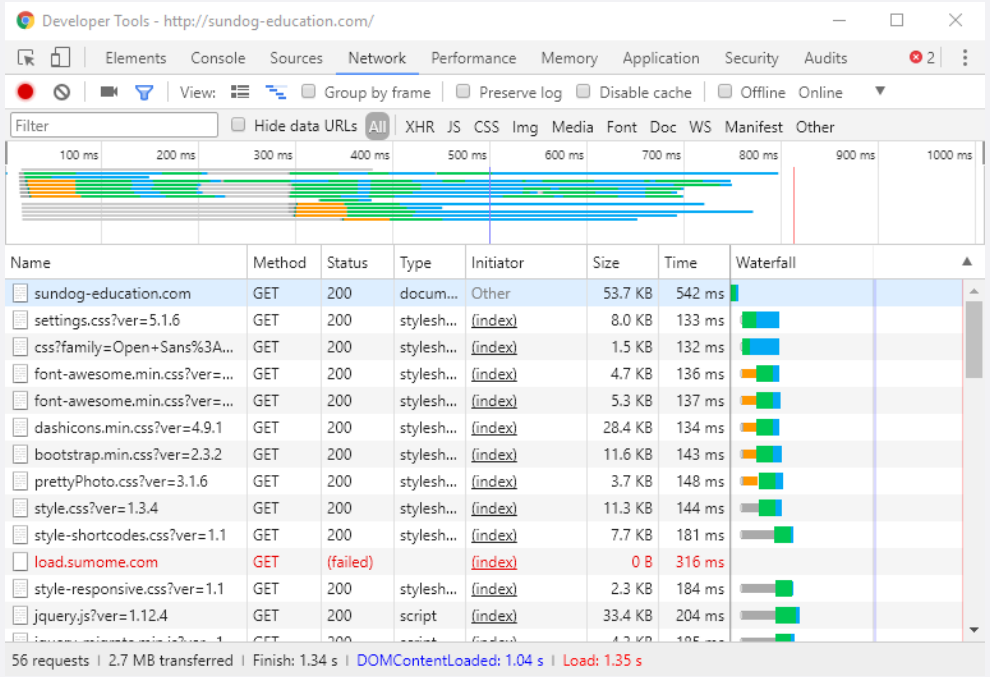
Bad:
It’s very complicated.
Most people do synchronization poorly.
There are entire books on the subject of proper multithreading.

JavaScript threading

JavaScript was made to run in the browser.
The DOM can only be manipulated in a single thread.

Since JavaScript was originally intended to manipulate the DOM, there's no reason to introduce multithreading.

what about AJAX?



Exactly!

Browsers already do a bunch of HTTP requests in parallel!

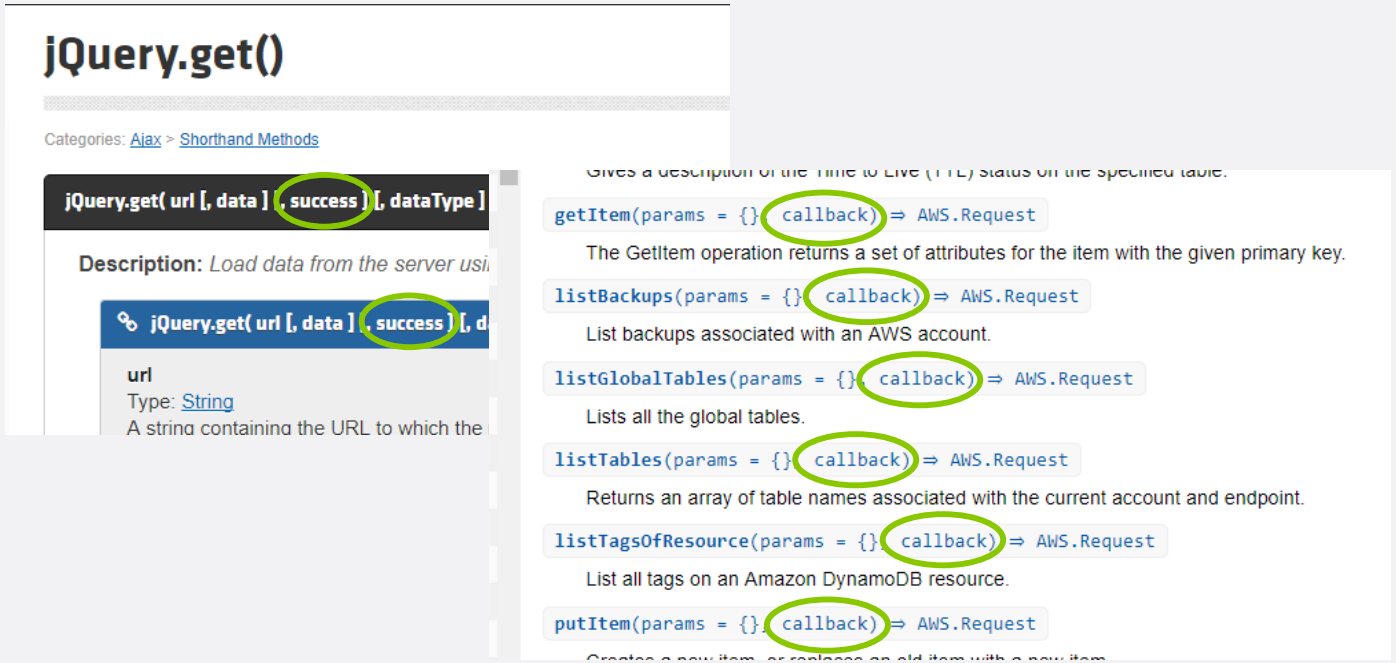
JavaScript is single-threaded!

…-ish?

All your code runs on a single thread.

Browsers and Node.js manage other threads that do things for you.

callbacks


jQuery.get()
Categories: Ajax > Shorthand Methods
jQuery.get(url [, data] [, success] [, dataType]
Description: Load data from the server usi
jQuery.get(url [, data] [, success] [, d
url
Type: String
A string containing the URL to which the
getItem(params = {}, callback) ⇒ AWS.Request
The GetItem operation returns a set of attributes for the item with the given primary key.
listBackups(params = {}, callback) ⇒ AWS.Request
List backups associated with an AWS account.
listGlobalTables(params = {}, callback) ⇒ AWS.Request
Lists all the global tables.
listTables(params = {}, callback) ⇒ AWS.Request
Returns an array of table names associated with the current account and endpoint.
listTagsOfResource(params = {}, callback) ⇒ AWS.Request
List all tags on an Amazon DynamoDB resource.
putItem(params = {}, callback) ⇒ AWS.Request

effects of event loop

Dependent calls require a lot of functions

Bookkeeping is everything for parallel behaviors

No synchronization

that’s it!

exercise #1

a) Add a message from “Student” to Frank

b) Add a new conversation

exercise solution

exercise #1 solution

```
2.json:
{
  "id": "2",
  "participants": ["Student", "Frank"],
  "messages": [
    {
      "sender": "Frank",
      "time": 1512246342159,
      "message": "This is Frank. I'm also sending you a message."
    },
    {
      "sender": "Student",
      "time": 1512247342159,
      "message": "Hello, Frank!"
    }
  ]
}
```

exercise #1 solution

```
conversations.json:
[
  {
    "participants": ["Student", "Brian"],
    "id": "1"
  },
  {
    "participants": ["Student", "Frank"],
    "id": "2"
  },
  {
    "participants": ["Student", "Alice"],
    "id": "hello-world"
  }
]
```

exercise #1 solution

hello-world.json

```
{
  "id": "hello-world",
  "participants": ["Student", "Alice"],
  "messages": [
    {
      "sender": "Alice",
      "time": 1512246299194,
      "message": "I'm Alice. I usually talk to Bob."
    }
  ]
}
```

what is lambda?

Function running on AWS

Options:
- Java
- Python
- JavaScript (Node.js)

what makes lambda serverless?

Feature	Lambda	EC2
Time to spin up	milliseconds	seconds to minutes
Billing increments	100ms	1s
Configuration	function	operating system
Scaling unit	parallel function executions	instances
Maintenance	AWS	AWS and you

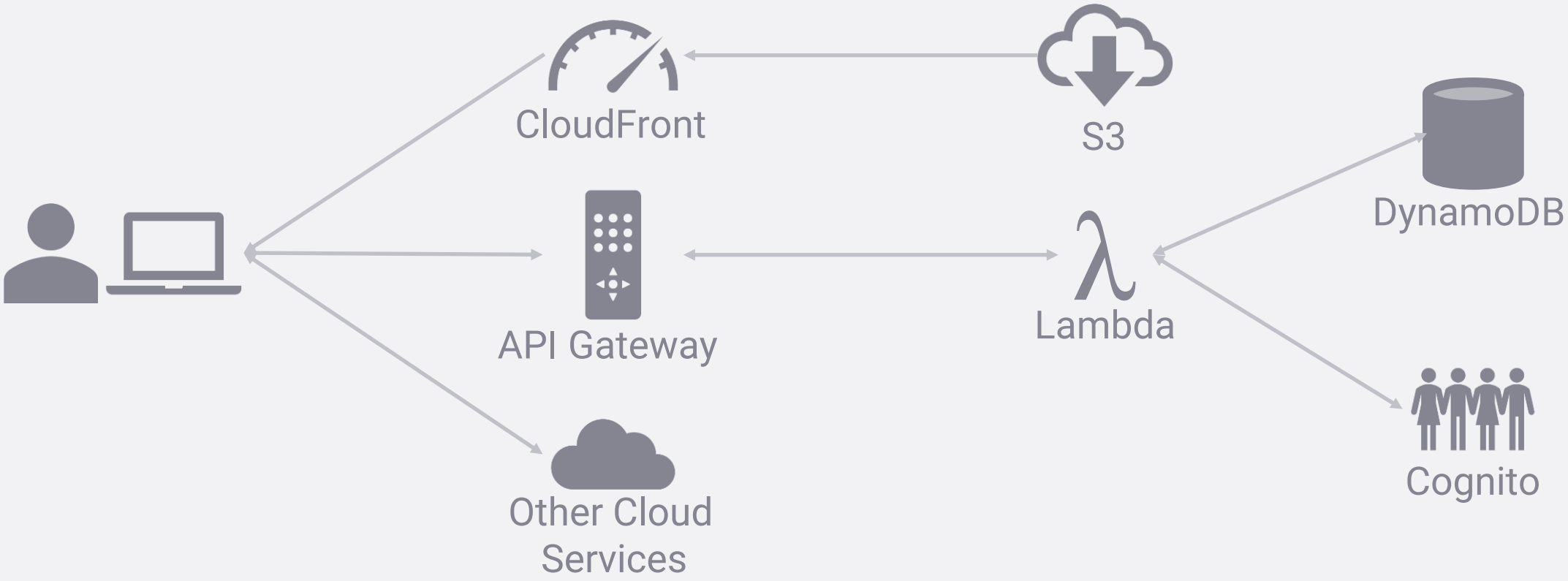
IAM and policies

IAM? i am what?

IAM – Identity and Access Management

users/groups
roles
policies
identity providers
user account settings
encryption keys

where do we need roles?



lambda triggers

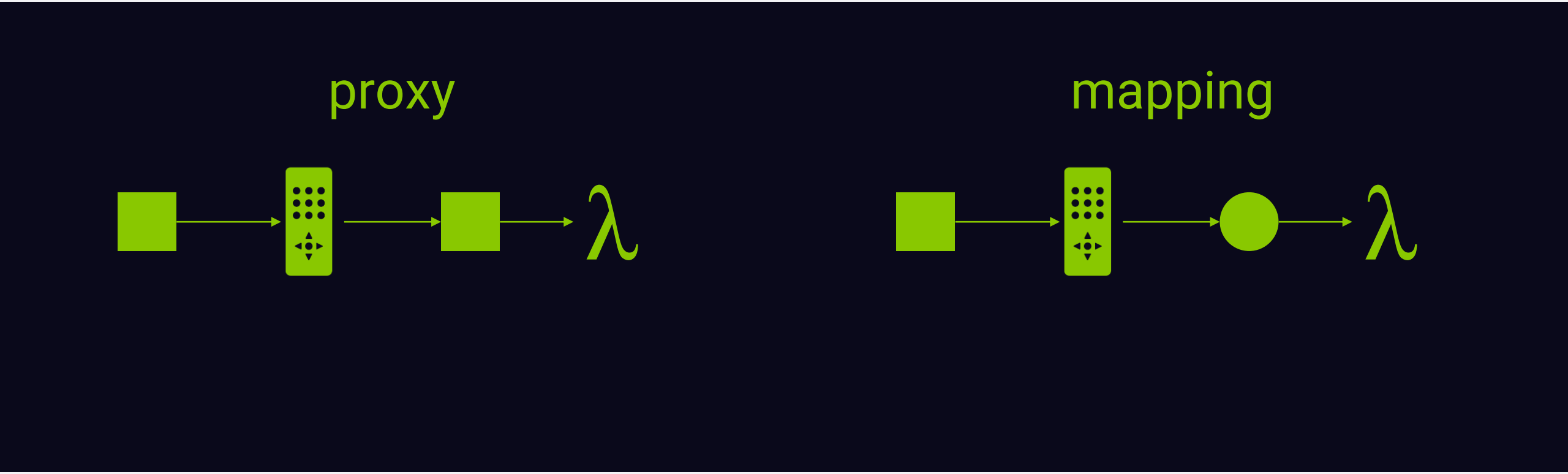
so many options

API Gateway
AWS IoT
Alexa Skills
Alexa Smart Home
CloudWatch Events
CloudWatch Logs
CodeCommit
Cognito Sync
DynamoDB
Kinesis
S3
SNS

but wait...there's more!

Lambda API
CloudFront

api gateway




cors

cors?

cross-origin resource sharing

long ago, on the internet

load this image from bar.com in an <img> tag?

sure

website
foo.com

load this JS from baz.com in a <script> tag?

browser

sure

make an XMLHttpRequest to baz.com?

NO!!!

someone got clever

load this JS from baz.com in a <script> tag?

website
foo.com

here's a new <script> tag to load from baz.com

sure

browser

sure

new standard: cors

load this JS from baz.com in a <script> tag?

website
foo.com

sure

browser

make an XMLHttpRequest to baz.com?

maybe?

maybe?

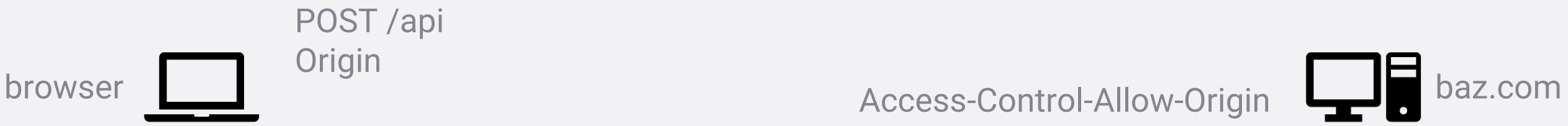
OPTIONS /api
Origin
Access-Control-Request-Method
Access-Control-Request-Headers

browser 

 baz.com

Access-Control-Allow-Origin
Access-Control-Allow-Methods
Access-Control-Allow-Headers
Access-Control-Max-Age
[Access-Control-Allow-Credentials]

then what?


POST /api
Origin
browser
Access-Control-Allow-Origin
baz.com


dynamodb

nosql

"non SQL" or "Not only SQL"

basically a key-value storage system

typically not ACID

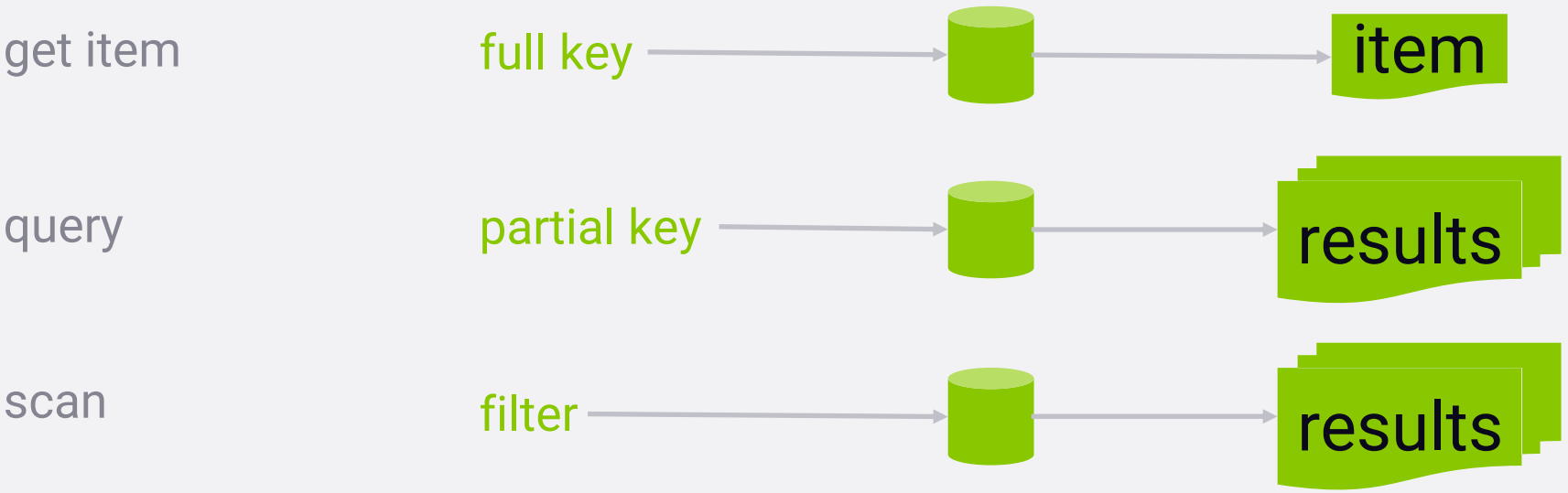
extremely scalable

dynamodb tables

types of attributes

name	type
B and BS	binary (and set)
BOOL	boolean
L	list
M	map
N and NS	number (and set)
NULL	null
S and SS	string (and set)

retrieving items



a note on pagination

provisioned throughput

1 read unit = 1 consistent read up to 4KB per second
1 read unit = 2 eventually consistent reads up to 4KB per second

1 write unit = 1 write up to 1KB per second

that’s the basics

table structures
data types
how to get items
pagination
provisioned capacity

dynamodb vs. s3

blobs vs. values

key	value
key 1	stuff
key 2	other stuff

limits

limit	S3	DynamoDB
record size	5TB	400KB
request rate	basically unlimited	80k capacity units in Virginia, 20k everywhere else
total size	unlimited	unlimited

consistency!

a few things to try

1. Make the Lambda function reject unsupported HTTP methods
   A. Create some new test configurations for these cases
2. Remove S3 access for your Lambda function

λ proxy pain

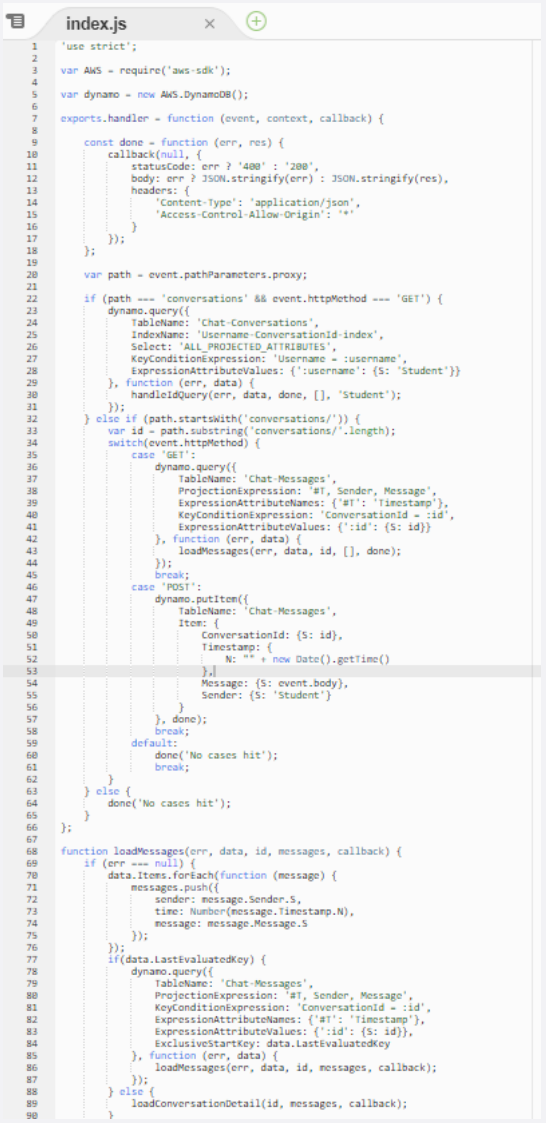
wait...what?

poorly-documented event object

having to worry about the details of HTTP requests

input comes from multiple places

over 200 lines of JavaScript to manage

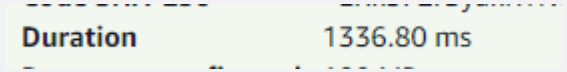


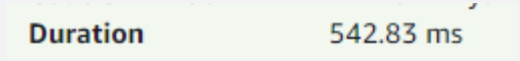
is there anything good about it?

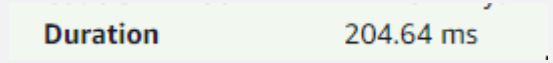
it’s easy

Lambda startup cost can be lower

less management







basically...

the power of api gateway

each resource-method combination can be its own function with its own lifecycle

each function can have simple, tailored input

ensuring API compatibility can occur without code

the plan

resources and methods

api models

request and response flows

resources and methods

http basics

```
GET / HTTP/1.1
User-Agent: curl/7.38.0
Host: www.google.com
Accept: */*
```

```
HTTP/1.1 200 OK
Date: Thu, 11 Jan 2018 03:46:43 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See
g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Set-Cookie: …
Accept-Ranges: none
Vary: Accept-Encoding
Transfer-Encoding: chunked
```

methods

Read

GET
HEAD

Write

POST
PUT
PATCH

Other

DELETE
OPTIONS

status codes

family	meaning
2xx	success
3xx	redirect
4xx	client error
5xx	server error

parts of a URL

request data

resource
method
query string
headers (includes cookies)
body (POST, PUT, PATCH)

response data

status code
headers (includes cookies)
body

what goes where?

what?	request	response
resource	what data or behavior is required	N/A
query string	filtering and subset information	N/A
headers	processing directives	processing directives
body	input data	output data

resources

Fully-defined

Some resources are one of a kind even though they may change based on user, time, etc.

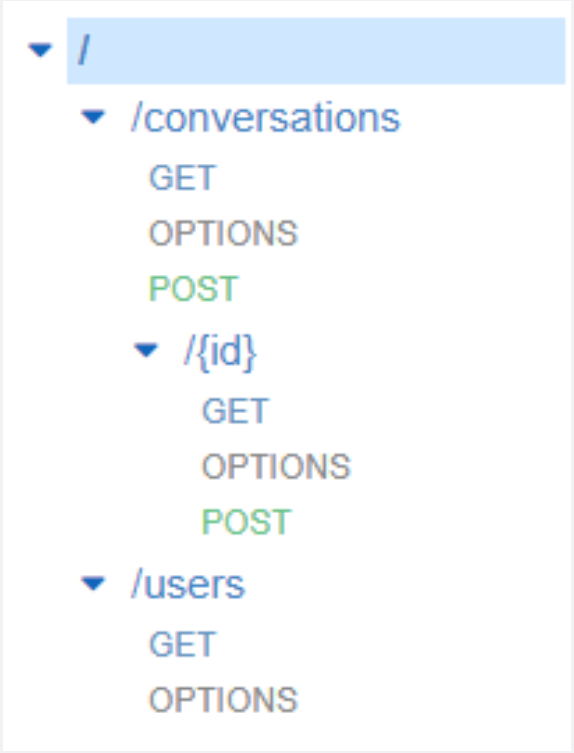
Example: /conversations

Parameterized

Other resources contain data in the URL. There can be some large number of resources that follow a pattern.

Example: /conversations/{id}

resources have methods



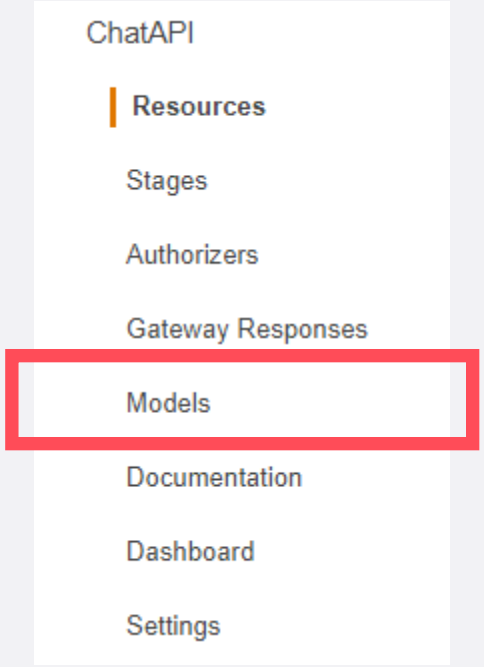
models

request data

resource
method
query string
headers (includes cookies)
body (POST, PUT, PATCH)

response data

models in api gateway



defining models

JSON schema

each model in its own schema document

json schema primitive types

type	notes
null	
boolean	
object	properties defined by “properties”
array	elements defined by “items”
number	
string	

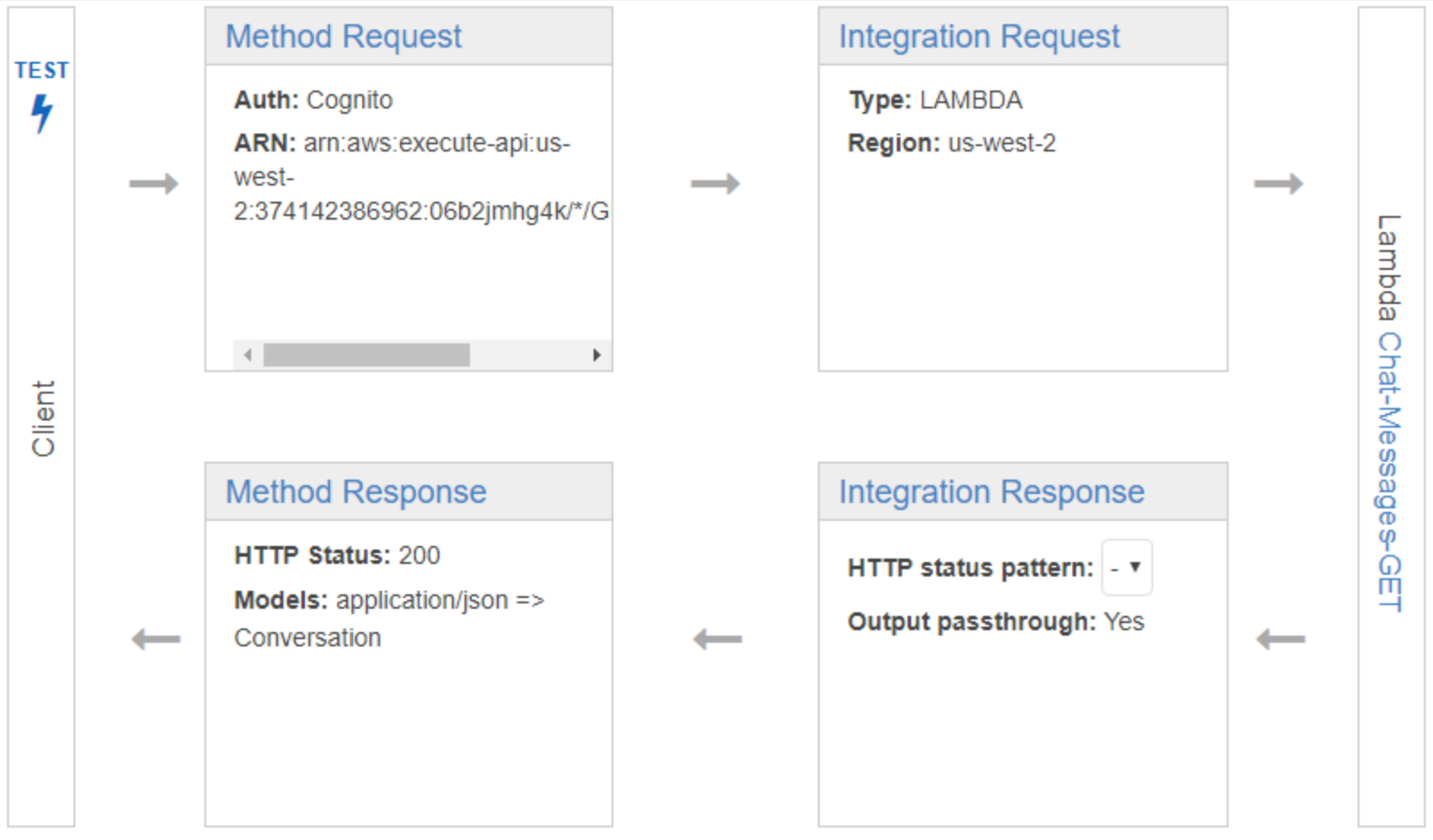
examples

```
{
  "type": "string"
}
```

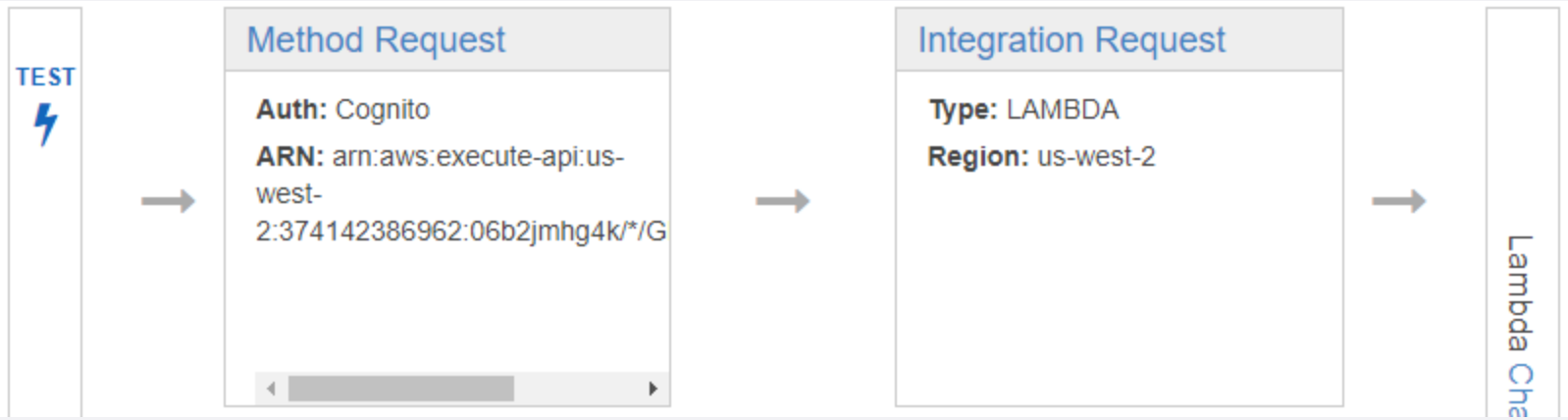
```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "id": {
        "type": "string"
      },
      "participants": {
        "type": "array",
        "items": {
          "type": "string"
        }
      },
      "last": {
        "type": "number",
        "format": "utc-millisec"
      }
    }
  }
}
```

request flow

overall flow



request flow



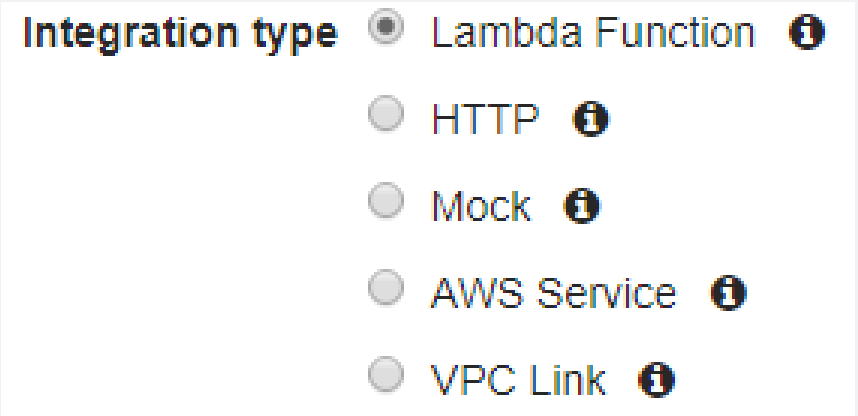
authorization
validation
caching
input:
- request path variables
- query string parameters
- request headers
- request body

integration type
remapping of input

authorization

AWS IAM
API key
Cognito
Custom

integration



lambda and mock

Lambda

Supports proxy and non-proxy modes

Just calls a Lambda function

Mock

Always returns the same thing

http, aws, and vpc

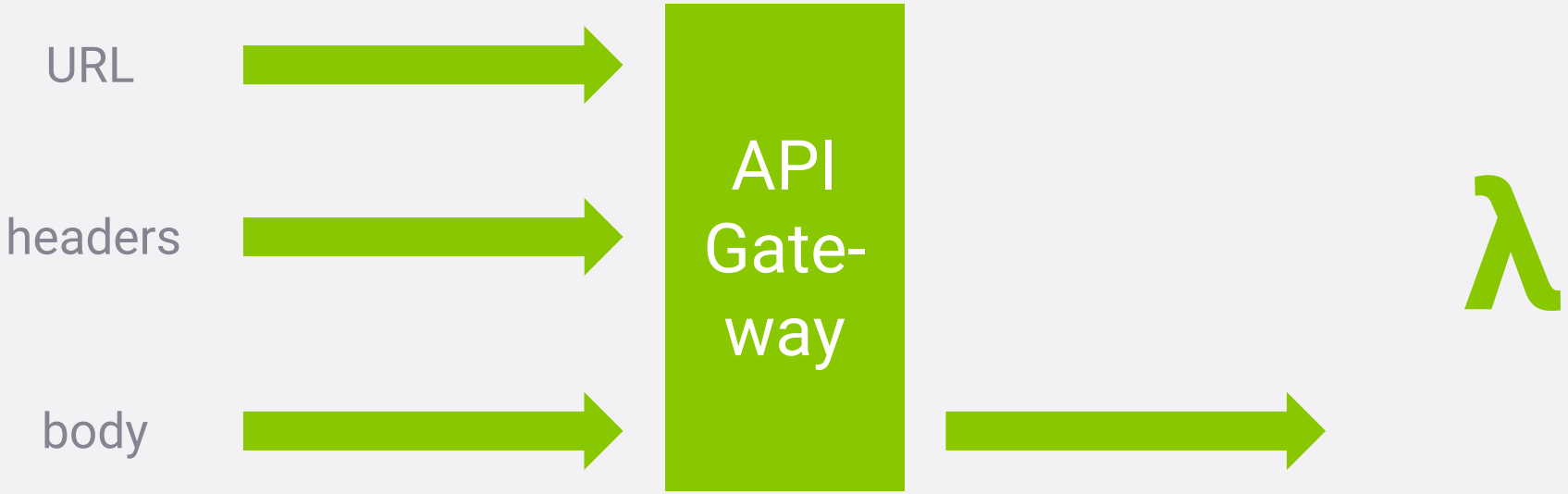
all very similar

default is effectively passthrough

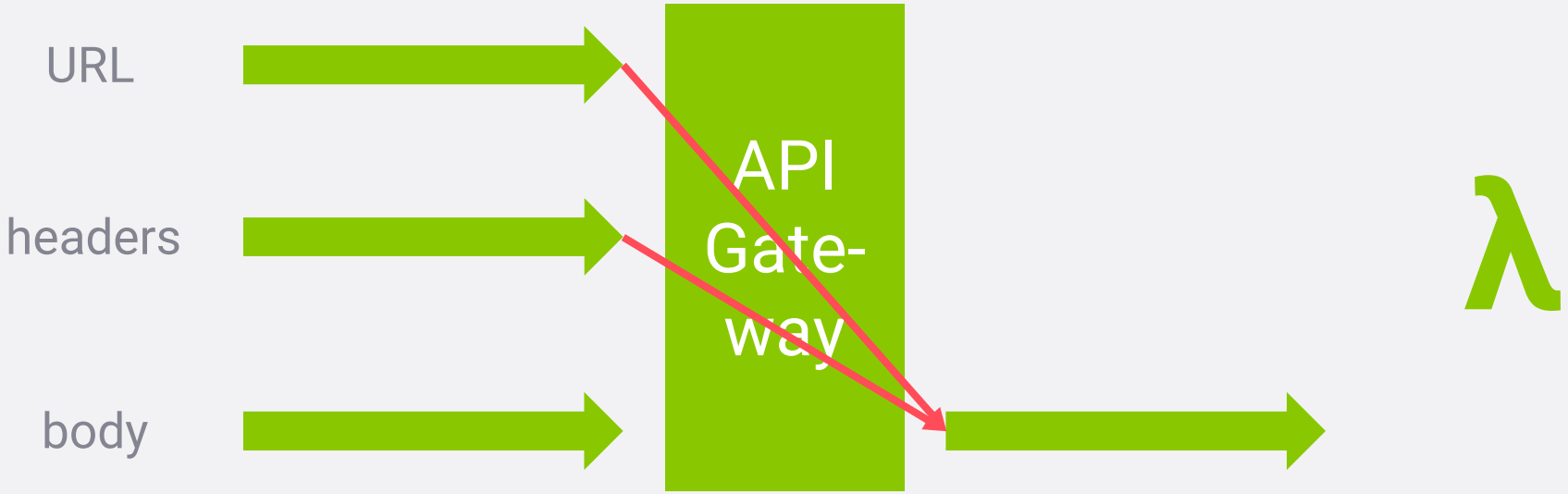
remapping can be done on almost anything

also, proxy mode

input mapping

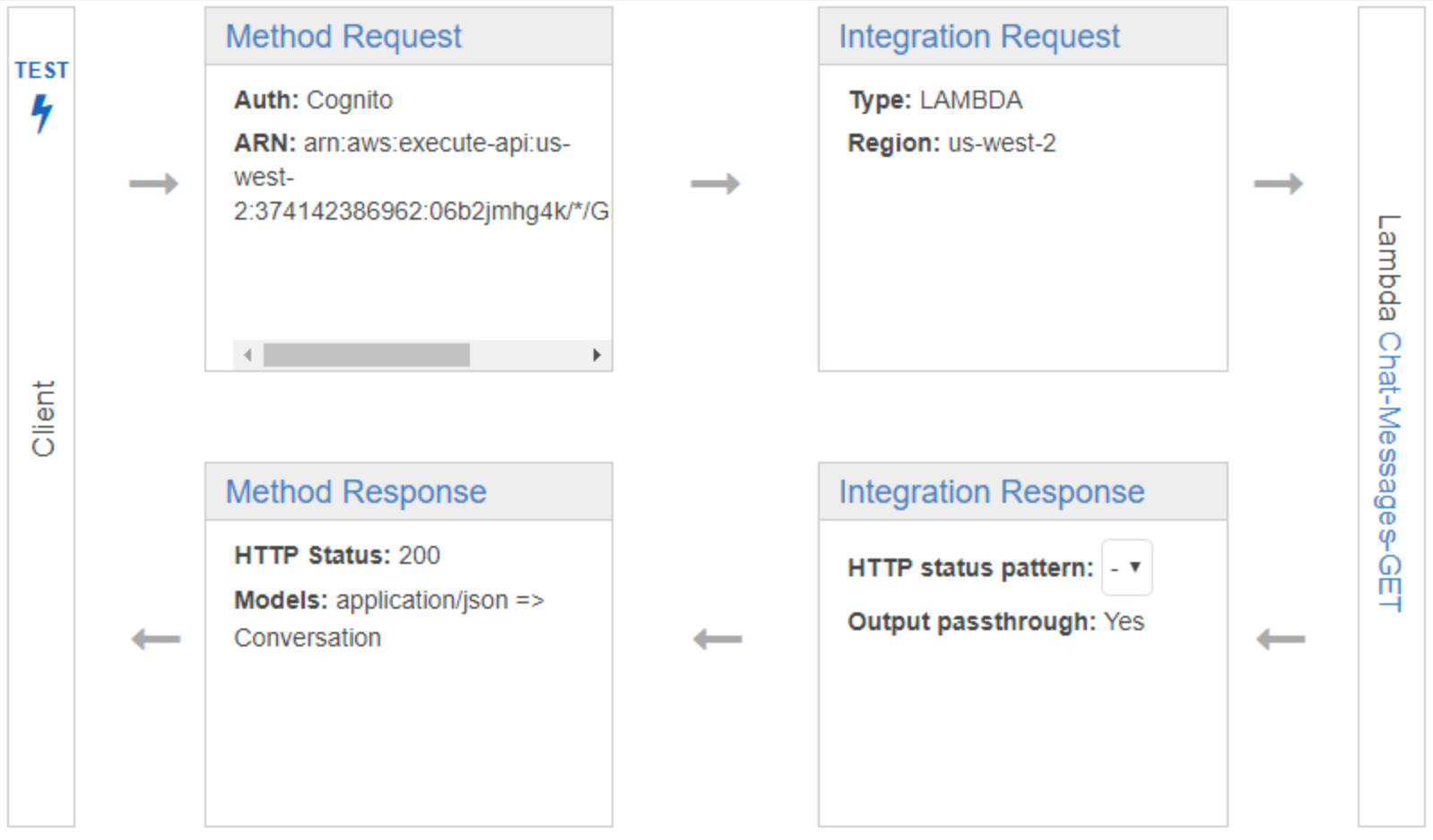


input mapping



response flow

overall flow



response flow

response definitions

response mapping
status code mapping


stages

what do they do?

snapshots of the API

public URL

operational settings

generated clients

generated swagger

Canary

operational stuff

Caching

CloudFront-based caching

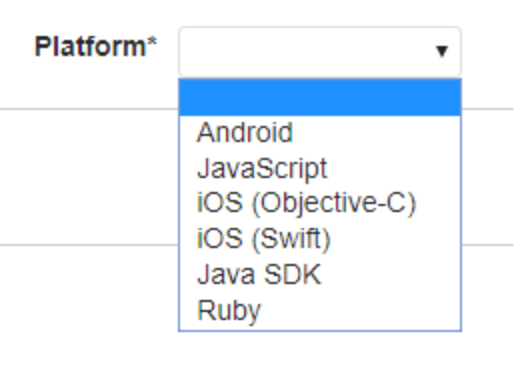
Throttling

Request rate with bursting

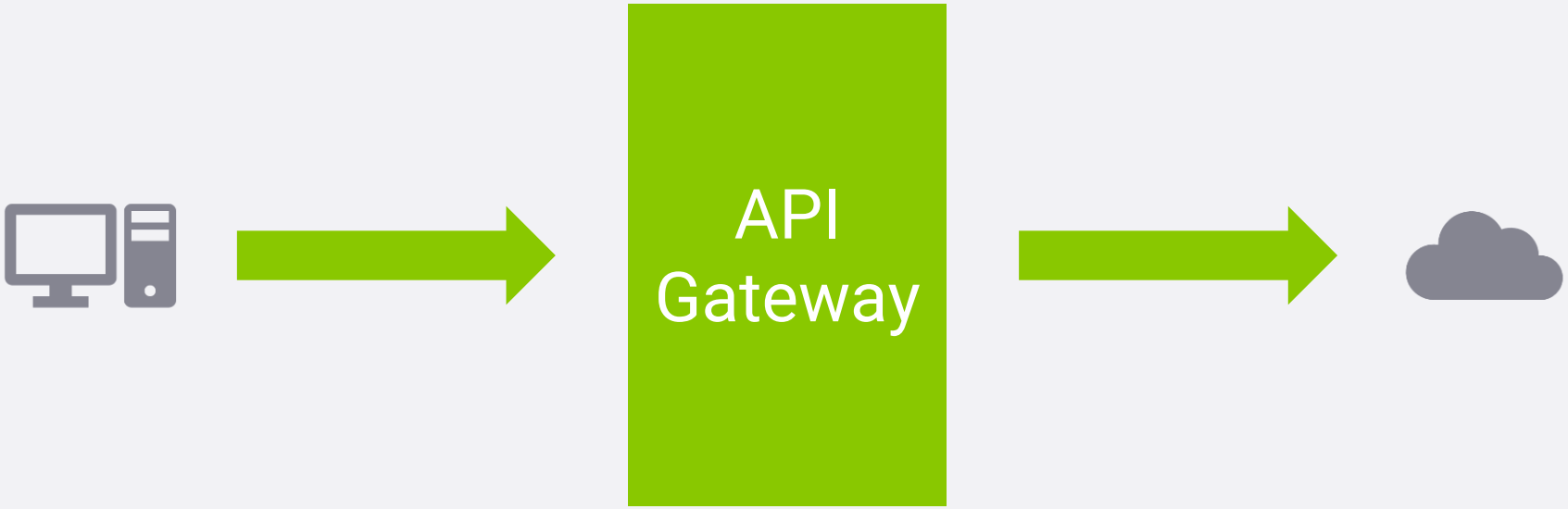
Variables

Define variables for the stage

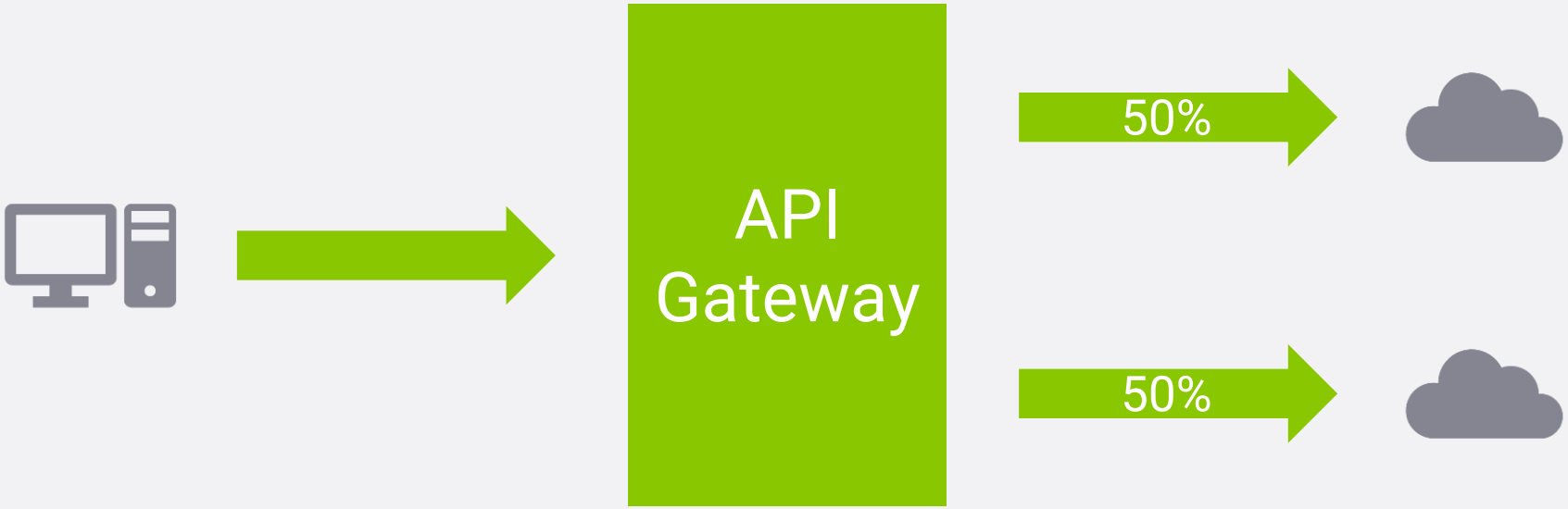
sdk generation and swagger



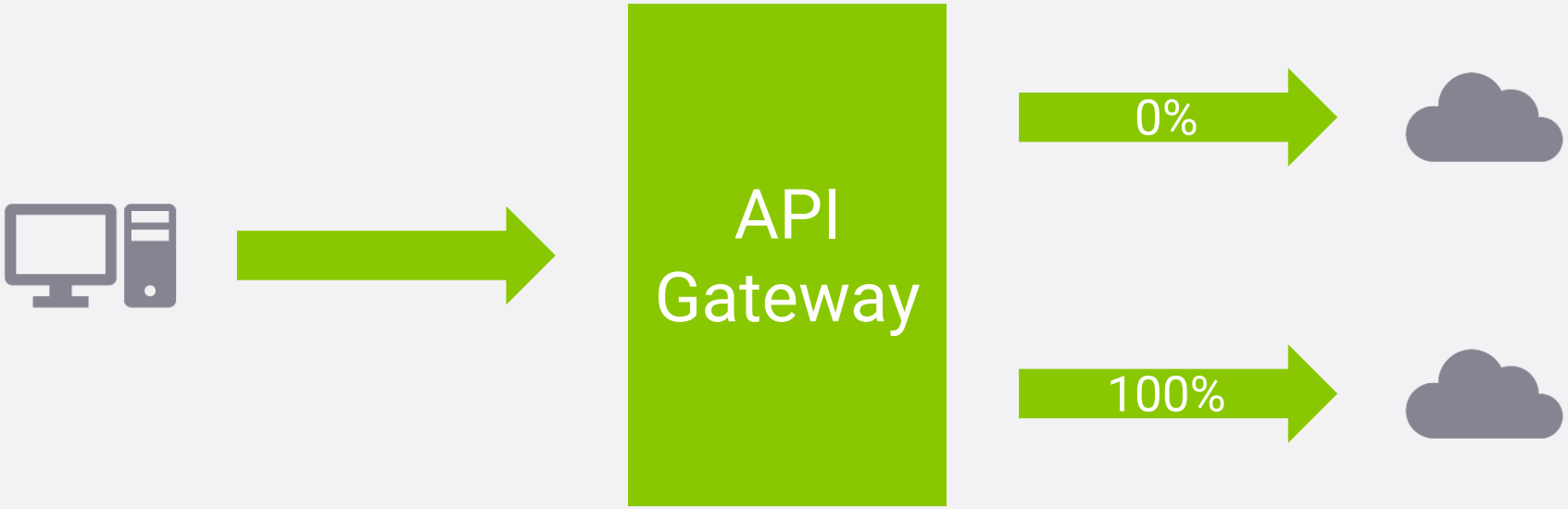
canary



canary



canary




cognito

services

User Pools

Your own usernames, passwords, attributes, etc.

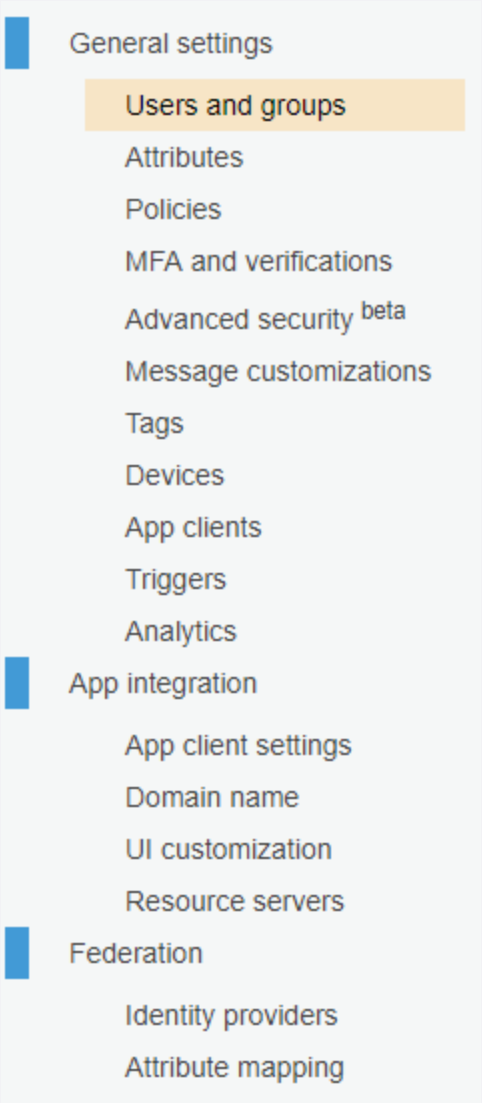
Federated

Use SSO (Google, Facebook, Amazon, etc.)

Sync

Synchronize data across devices

user pools


General settings
Users and groups
Attributes
Policies
MFA and verifications
Advanced security beta
Message customizations
Tags
Devices
App clients
Triggers
Analytics
App integration
App client settings
Domain name
UI customization
Resource servers
Federation
Identity providers
Attribute mapping

user information

users and groups

List and manage users and/or groups and see details

attributes

Define what a user looks like

security

policies

Password length, special characters, etc.

mfa and verifications

MFA
phone and email verification

advanced

Intelligent security features

customization

message

Customize messages to users

triggers

Lambda functions for user lifecycle

devices

Remember devices

other

app clients

Define clients of the user pool

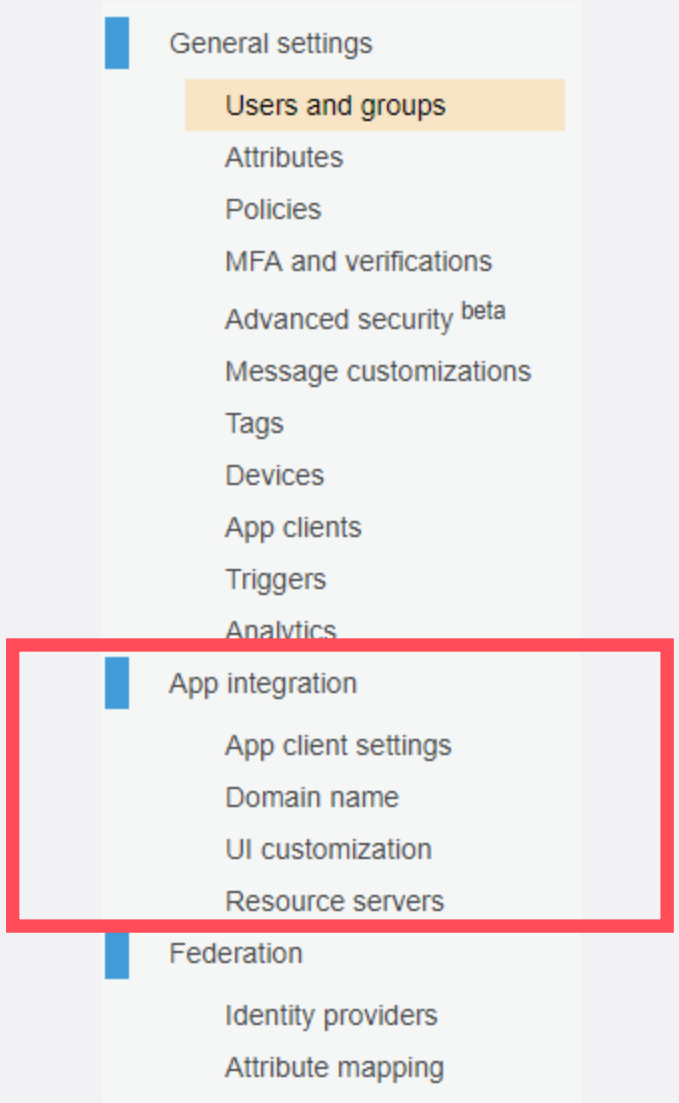
tags

AWS tagging

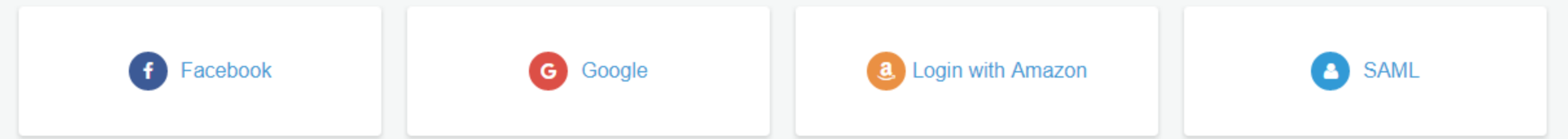
analytics

What are your users doing?

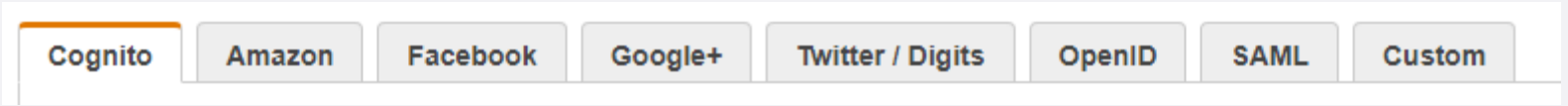
oauth 2.0


General settings
Users and groups
Attributes
Policies
MFA and verifications
Advanced security beta
Message customizations
Tags
Devices
App clients
Triggers
Analytics
App integration
App client settings
Domain name
UI customization
Resource servers
Federation
Identity providers
Attribute mapping

federation?

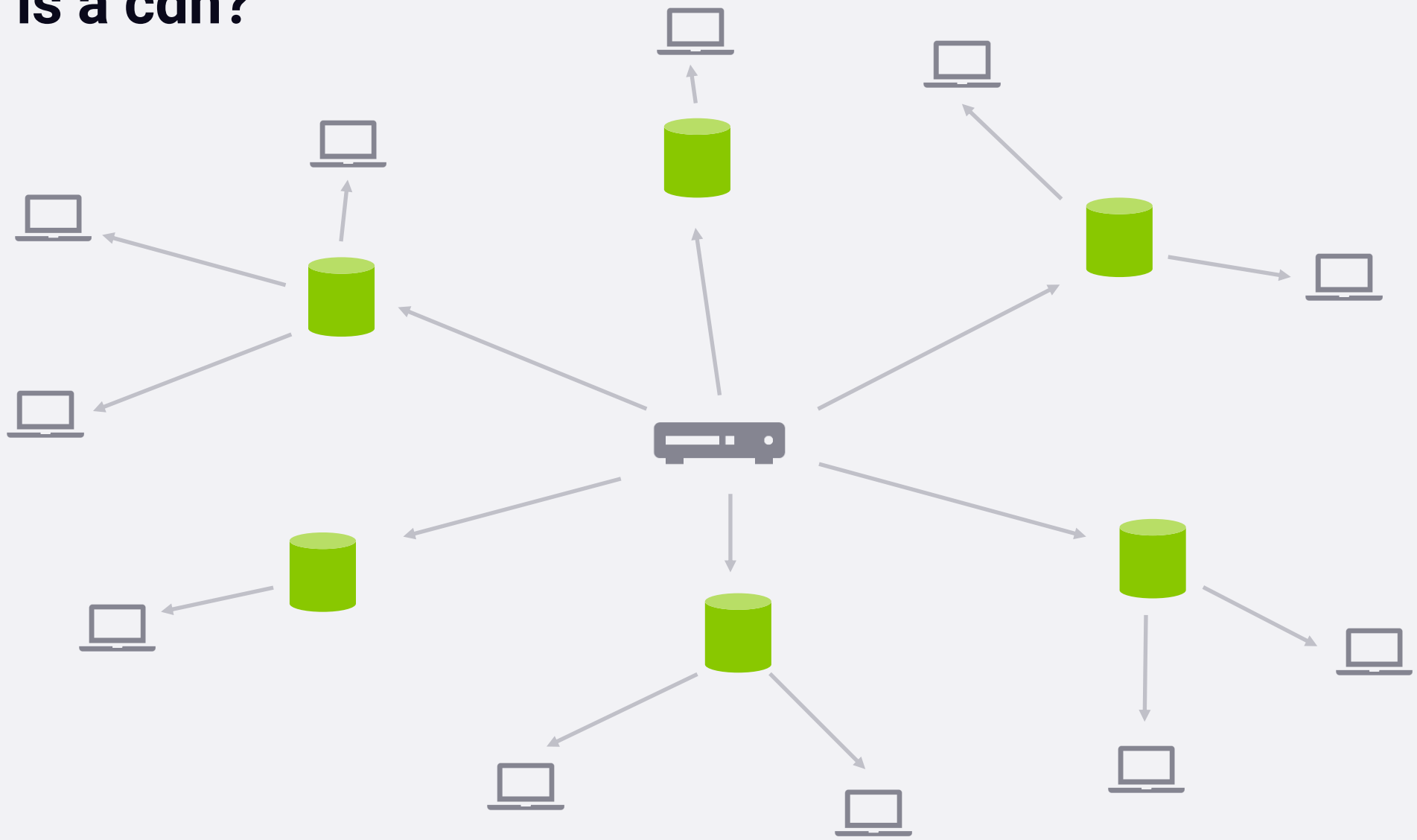


federation!
and sync!




cloudfront

what is a cdn?



origin vs. behavior

origin

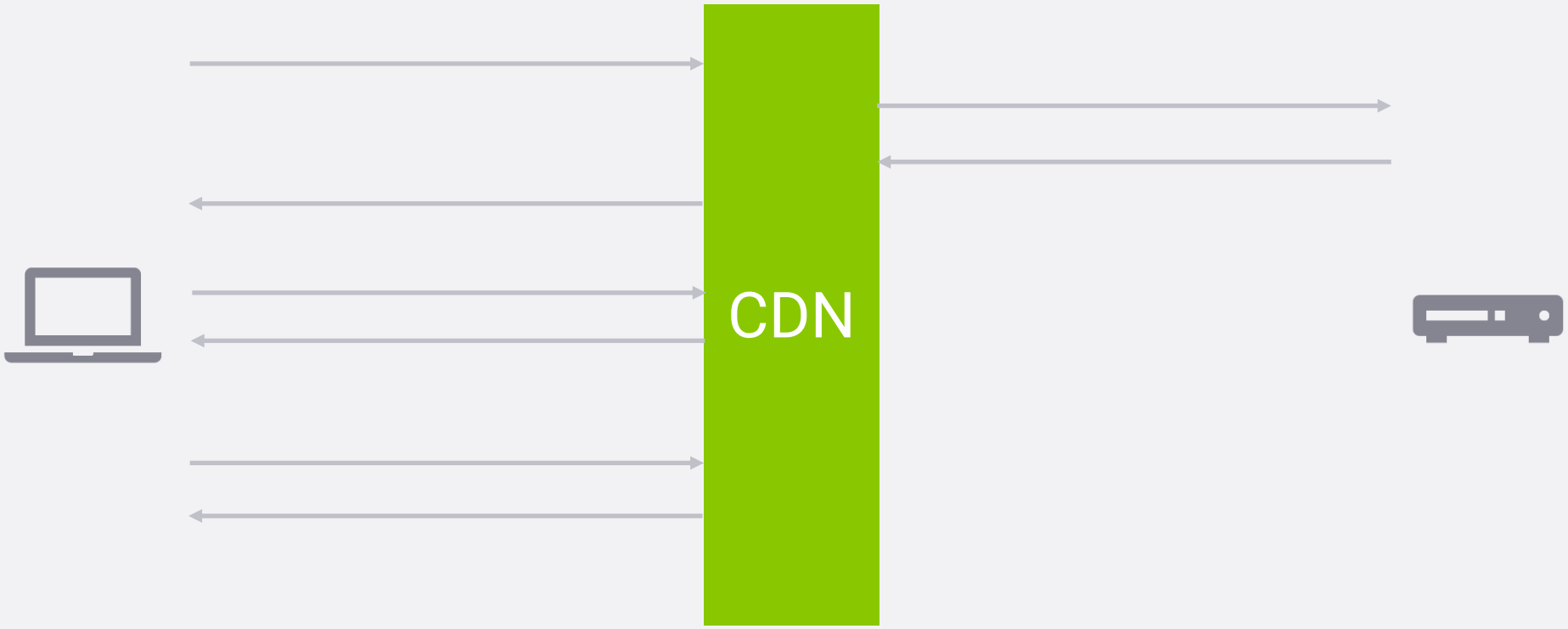
Where the content comes from

S3, Web Server, API

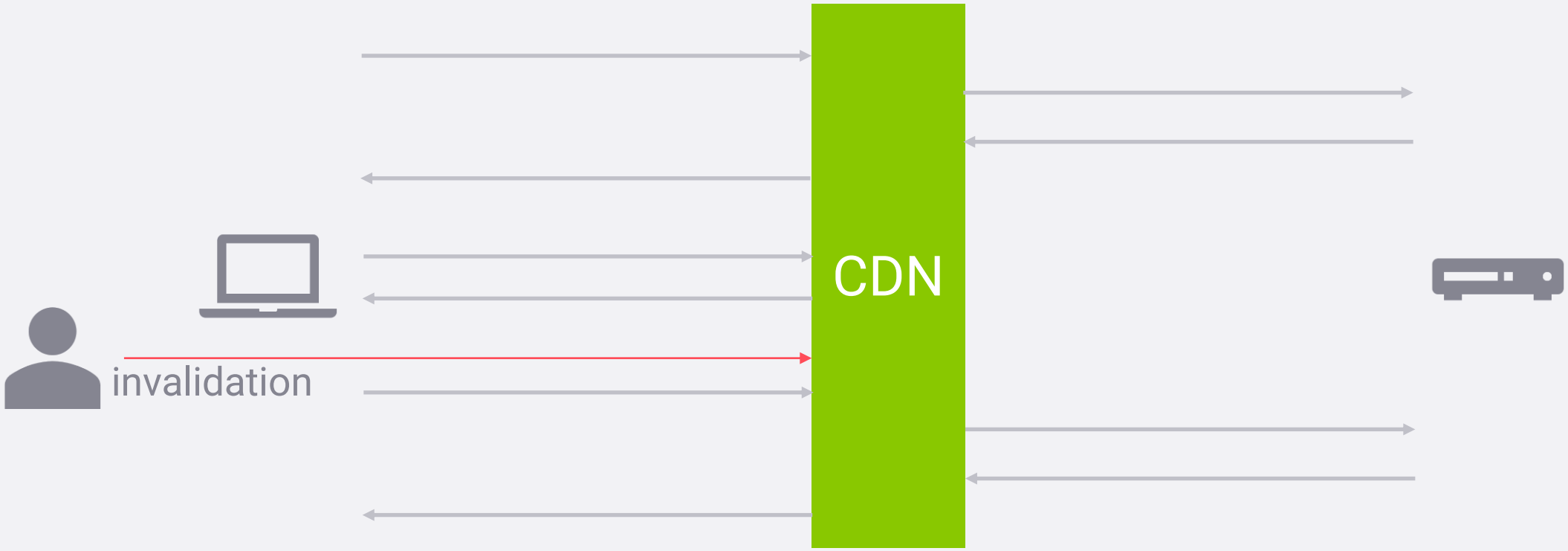
behavior

How to serve and cache a set of URIs

caching



invalidation



other stuff

compression

Node.js at the edge