SERVERLESS ARCHITECTURE

MICROSERVICES AND SCALABILITY FOR AUTOMOTIVE DATA

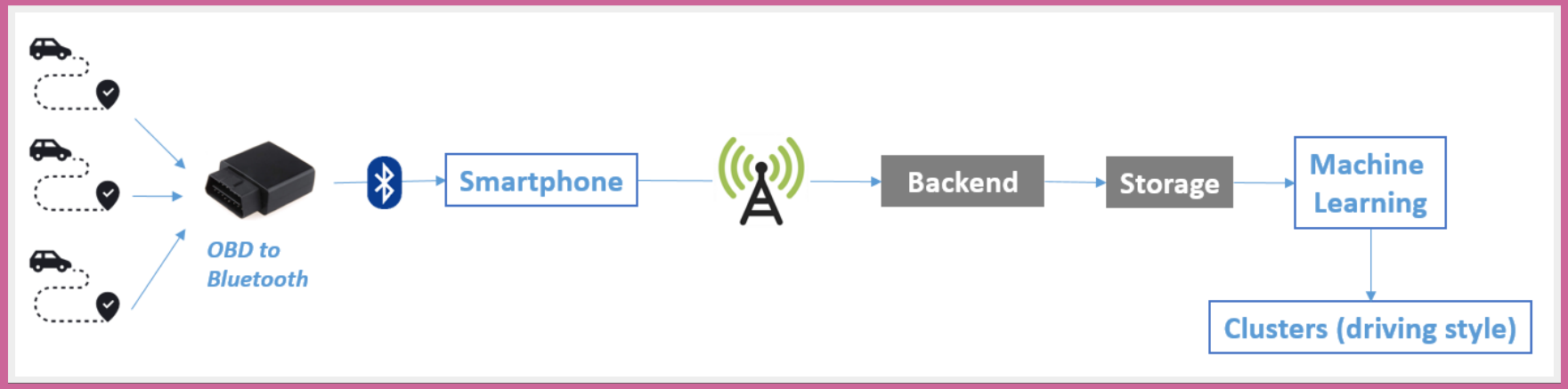
Jean-Christophe Baey

Photo: Maximilian Wachter

AGENDA

- Objectives
- Microservices
- Serverless
- POC Architecture
- Cost estimation

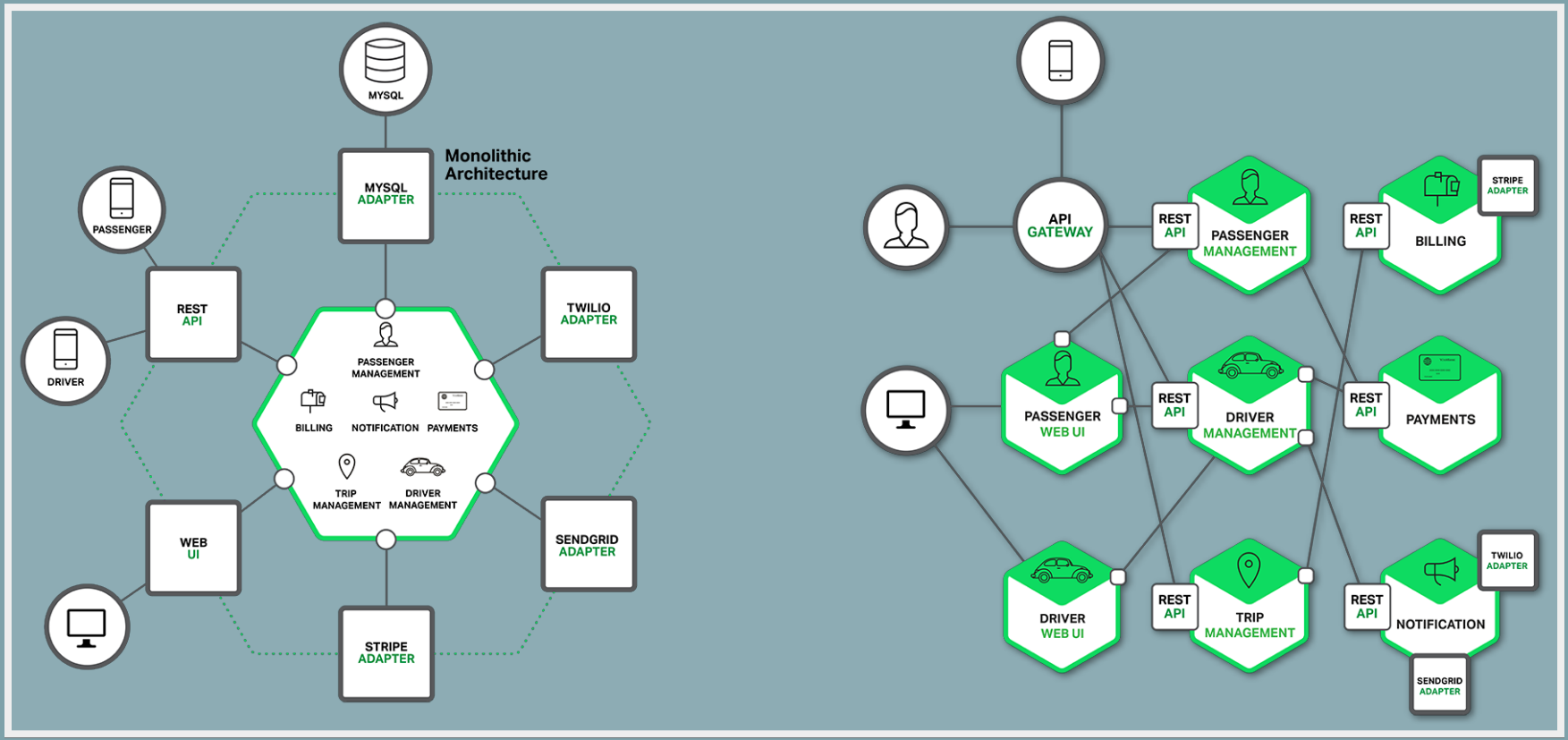
OBJECTIVES


OBD to
Bluetooth
Smartphone
Backend
Storage
Machine
Learning
Clusters (driving style)

MICROSERVICES ARCHITECTURE

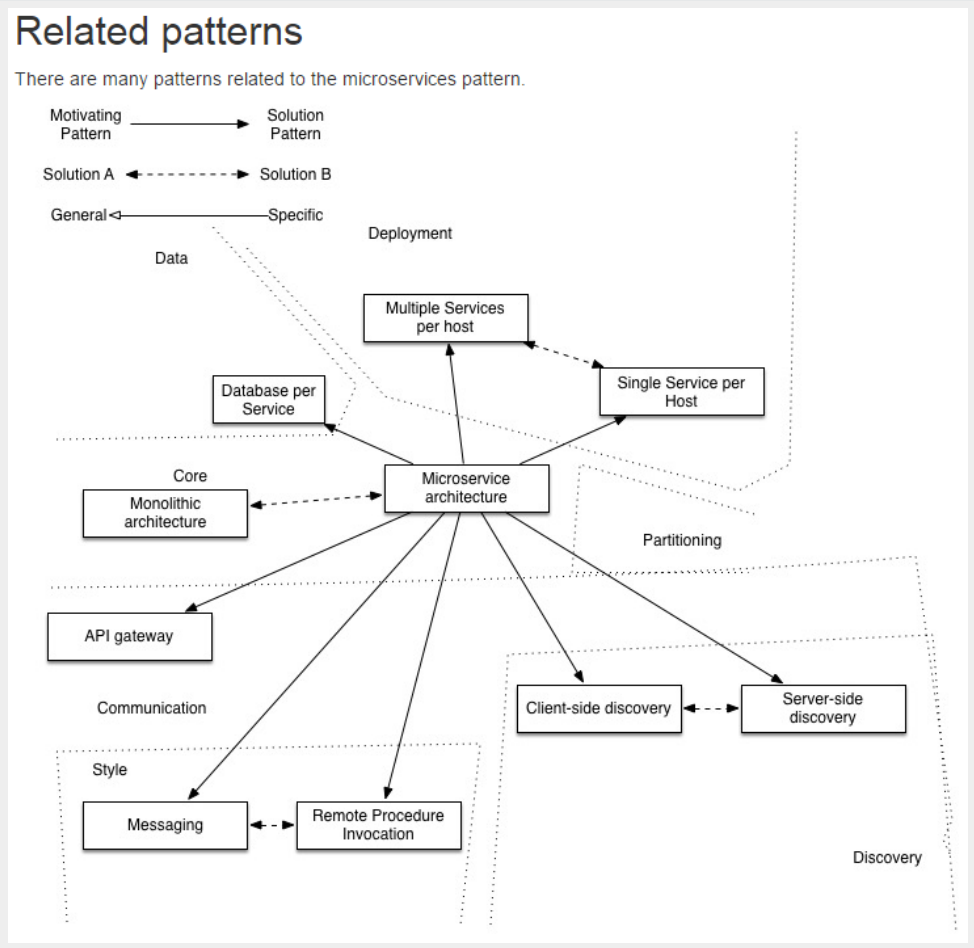
MICROSERVICES ARCHITECTURE

Monolithic Applications vs Microservices Applications



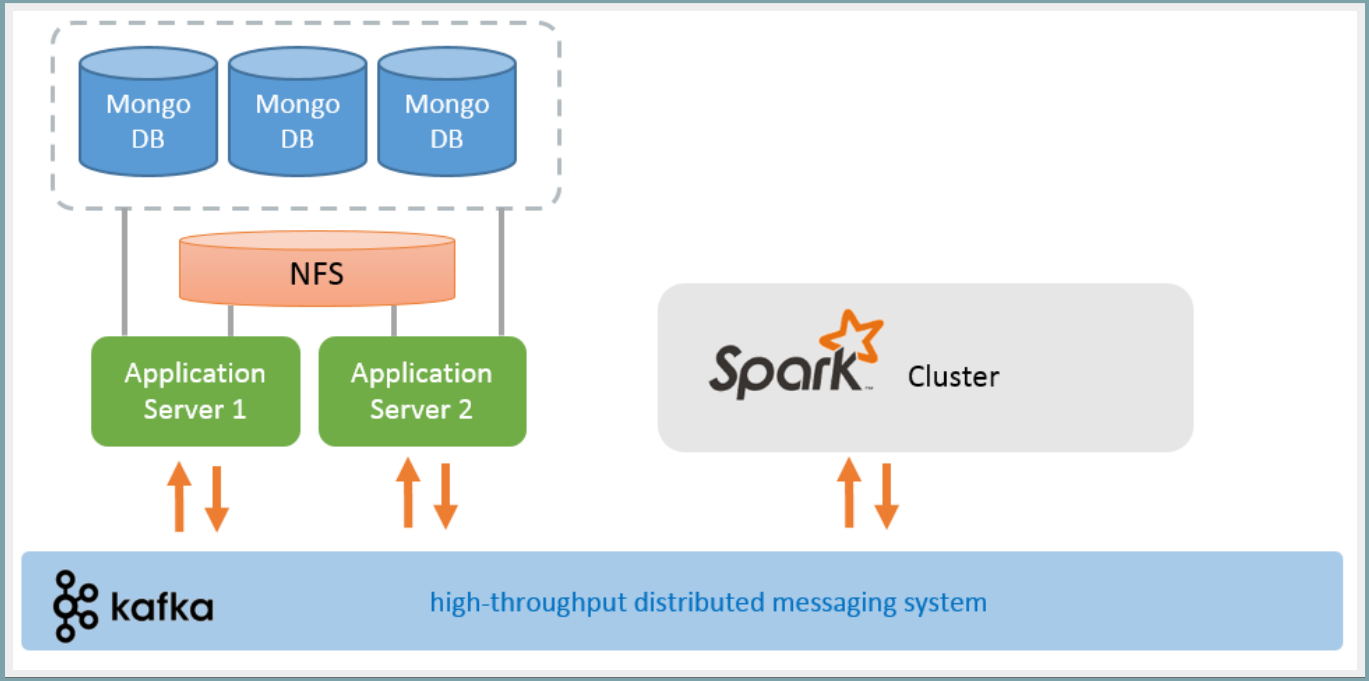
Credits: Nginx Introduction to microservices

MICROSERVICES ARCHITECTURE



Credits: Microservices.io Patterns

MICROSERVICES ARCHITECTURE



Micro-services using message bus

SERVERLESS ARCHITECTURE

SERVERLESS ARCHITECTURE

DEFINITION 1

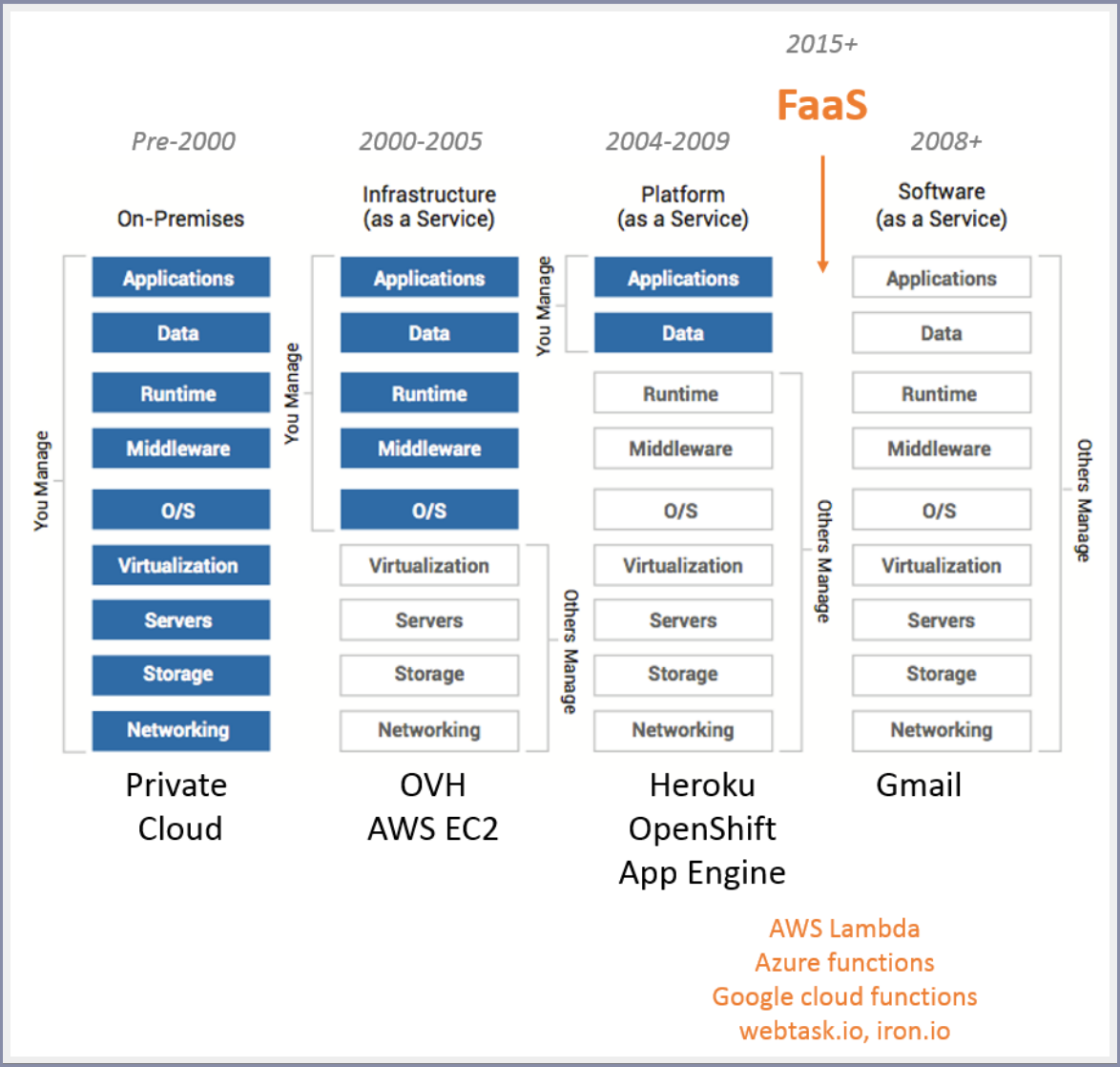
- Fully depend on third-party services in the cloud, for ex:
  - Authentication (Auth0, AWS Cognito, ...)
  - Database (AWS DynamoDB,)
  - Emailing (AWS SES, Sendgrid, ...)
  - Payments (Stripe)
  - ...
- Moving logic to front end
- Backend as a Service or BaaS

SERVERLESS ARCHITECTURE

DEFINITION 2

- Custom code that's run in stateless ephemeral containers:
  - functions are event-triggered
  - only last for one invocation
  - run code without provisioning or managing servers
- Functions as a service or FaaS
- AWS Lambdas, Azure functions, Google cloud functions

SERVICE MAP


2015+
FaaS
Pre-2000
2000-2005
2004-2009
2008+
On-Premises
Infrastructure (as a Service)
Platform (as a Service)
Software (as a Service)
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
You Manage
Others Manage
Private Cloud
OVH AWS EC2
Heroku OpenShift App Engine
Gmail
AWS Lambda
Azure functions
Google cloud functions
webtask.io, iron.io

SERVERLESS PROS

- Horizontal scaling is completely automatic, elastic, and managed by the provider
- Remove traditional **always on** server system
- Reduce operational cost and complexity

SERVERLESS CONS

- Stateless
- Limited execution Duration
  - not suitable for long lived task
- Startup Latency
  - average: 10ms to 100ms, up to 10s on JVM (if not used)
- Vendor dependencies
- Immature tooling (Serverless framework, Apex)
- Cloud only for the moment
  - **on premises** open source initiative: IBM's OpenWhisk

POC

HIGH LEVEL ARCHITECTURE

API GATEWAY

Host the API and route API calls

LAMBDA

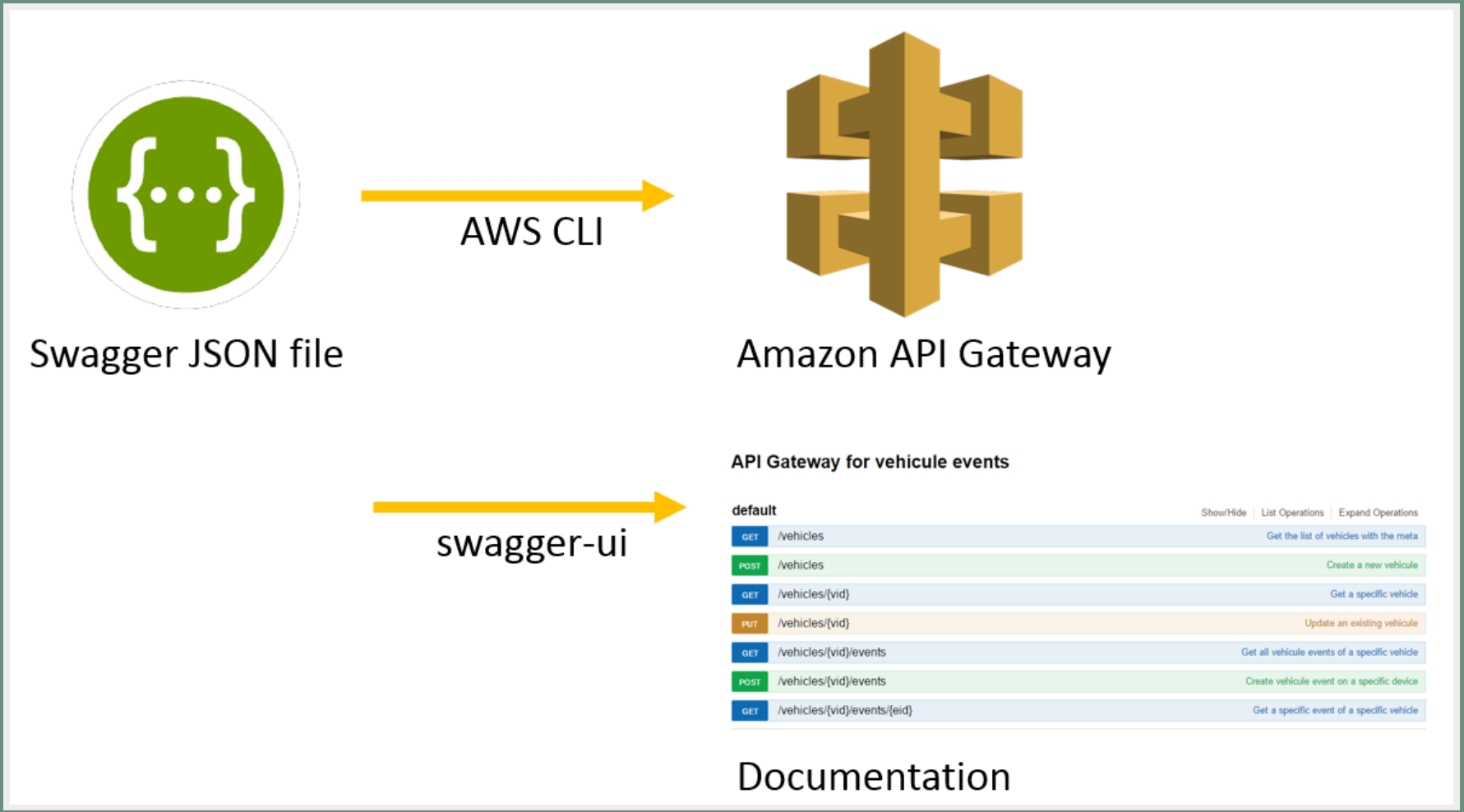
Execute our app's business logic

DYNAMODB

Data store

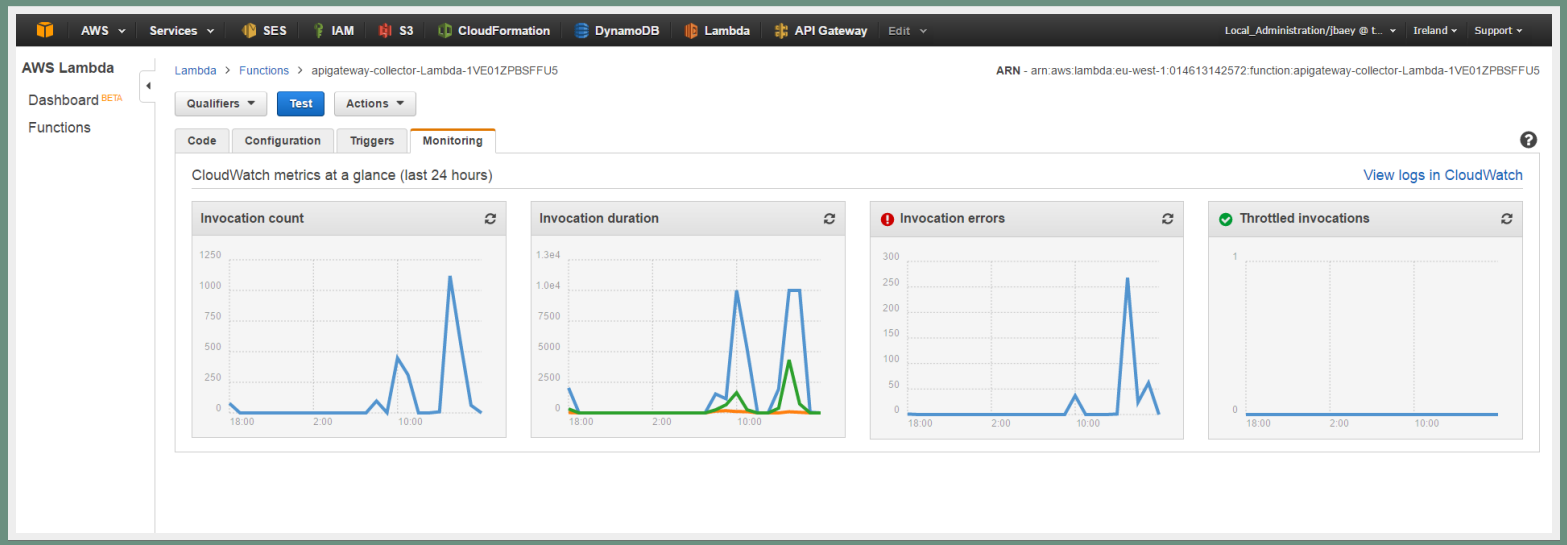
API GATEWAY

- Created from JSON/Yaml Swagger file
- Generate developper documentation



LAMBDA

- Handle request from API Gateway
- Read/Write into DynamoDB and log files.
- Monitored by CloudWatch
- Written in NodeJS 4.3

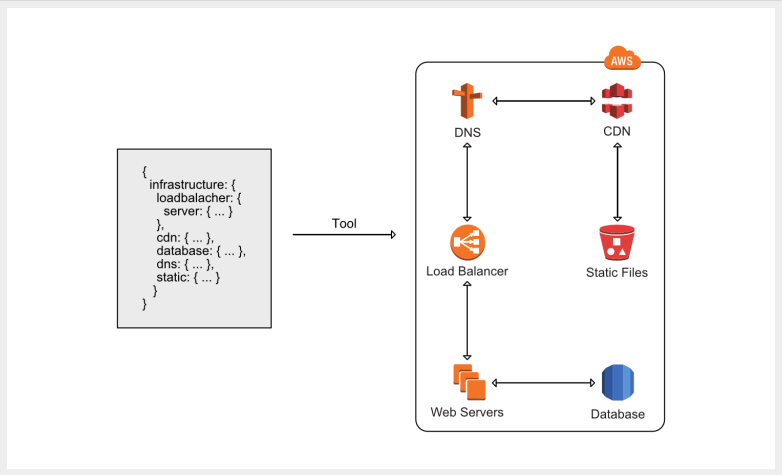


DYNAMO DB

- NoSQL database
- Seamless scalability
- Document based (JSON)
- 2 tables
  - vehicules
    - primary partition key: vid
  - vehiculeEvents
    - primary partition key: vid
    - primary sort key: eid

CLOUD FORMATION

- AWS tool to create a complete stack using CLI
- Described in template file
- Used for creating:
  - Lambda
  - DynamoDB Tables
  - IAM access rights



Example from cloudonaut.io

COST ESTIMATION

COST ESTIMATION

Assumptions per vehicule:

- 40 min drive per day
- 1 sample (520 bytes) per sec
  - 2 400 samples per day
- 1 query to AWS for 10 samples

COST ESTIMATION

Vehicule count	Lambda cost	Dynamodb cost io / db size	Total for 1 month
100	0 $	0$ 10 io/sec, 3.5GB	0$
1 000	22.54 $	54.03$ 100 io/sec, 35GB	76.57$
10 000	287.16 $	760.89$ 1 000 io/sec 349GB	1 048.05$

THANKS