Why You Should Build Your Next Product With AppSync

Marco Troisi

aws COMMUNITY DAY BELFAST

Hello there!

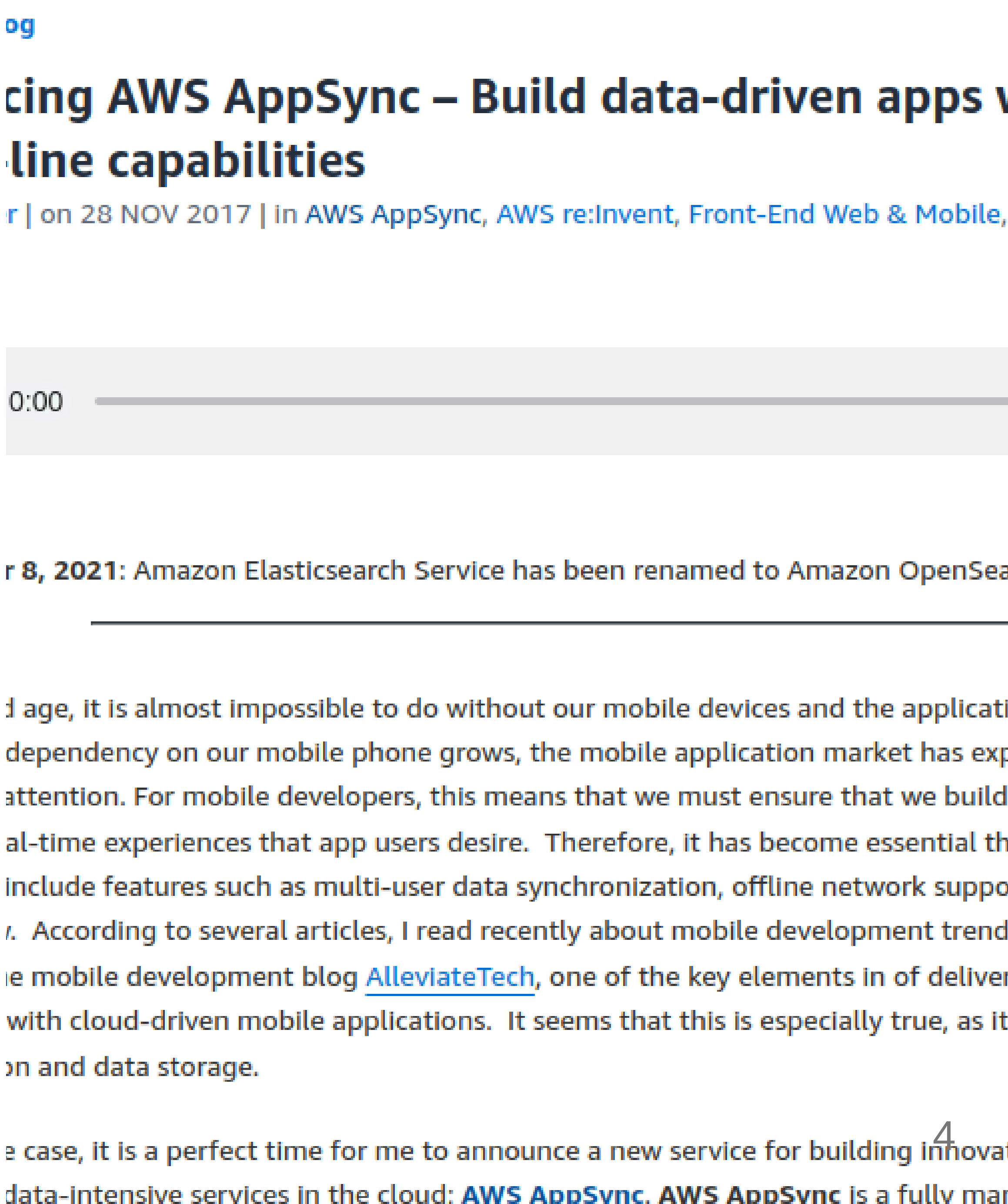
- Marco Troisi
- Born in 🇮🇹 Living in 🇬🇧
- CTO at Trilo
- Software Developer and Software Architect for over 12 years
- AWS Community Builder (Serverless)
- Writer for The Serverless Mindset newsletter

Agenda

- The evolution of AppSync
- *Reason #1*: Rapid Prototyping
- *Reason #2*: Infinite Scalability
- *Reason #3*: Team Collaboration
- *Reason #4*: Native AWS Integration
- *Reason #5*: Error Reduction
- *Reason #6*: Cloud-Native Evolution
- *Reason #7*: Empowering Small Teams

The evolution of AppSync

- The old AppSync
  - VT-*hell*
  - Not so great DX
  - Business logic very hard to test



og

cing AWS AppSync – Build data-driven apps v
line capabilities

r | on 28 NOV 2017 | in AWS AppSync, AWS re:Invent, Front-End Web & Mobile,

0:00

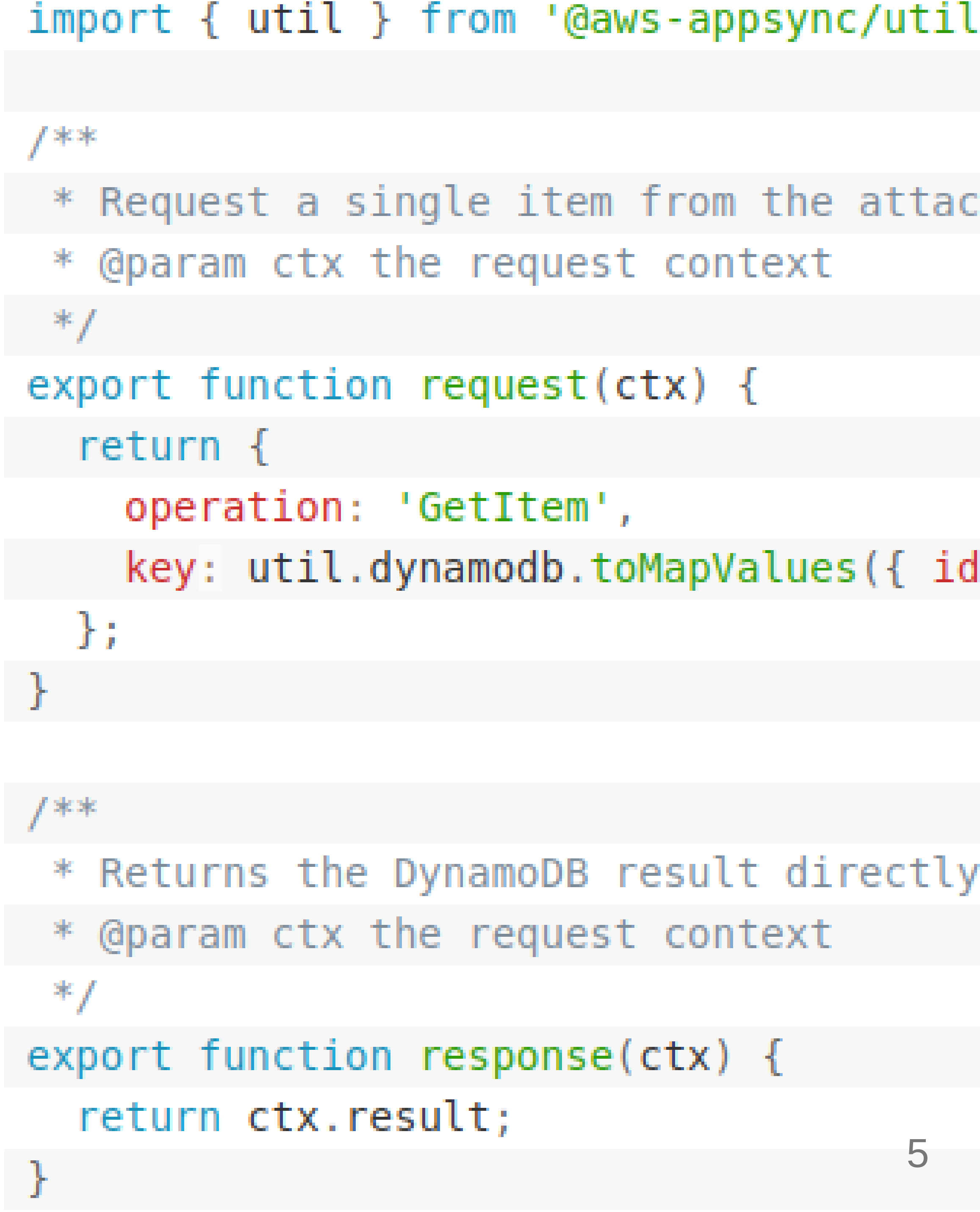
r 8, 2021: Amazon Elasticsearch Service has been renamed to Amazon OpenSea

d age, it is almost impossible to do without our mobile devices and the applicati
dependency on our mobile phone grows, the mobile application market has exp
attention. For mobile developers, this means that we must ensure that we build
al-time experiences that app users desire. Therefore, it has become essential th
include features such as multi-user data synchronization, offline network suppo
y. According to several articles, I read recently about mobile development trend
e mobile development blog AlleviateTech, one of the key elements in of deliver
with cloud-driven mobile applications. It seems that this is especially true, as it
on and data storage.

e case, it is a perfect time for me to announce a new service for building innova
data-intensive services in the cloud: **AWS AppSync**. **AWS AppSync** is a fully man

The evolution of AppSync

- The *new* AppSync
  - Javascript resolvers
  - Lambda resolvers
  - Direct connections



```
import { util } from '@aws-appsync/util

/**
 * Request a single item from the attac
 * @param ctx the request context
 */
export function request(ctx) {
  return {
    operation: 'GetItem',
    key: util.dynamodb.toMapValues({ id
  };
}

/**
 * Returns the DynamoDB result directly
 * @param ctx the request context
 */
export function response(ctx) {
  return ctx.result;
}
```

Rapid Prototyping

1. Build a UI
2. Write the interactions in GraphQL
3. Figure out the AWS services needed
4. Write the resolvers
5. Amplify codegen
6. Good to go 🎉

Infinite Scalability

- Automatic scaling
- 100% serverless
- Real-time data sync
- IAM authentication
- Multi-region support

Team Collaboration

- GraphQL as a shared language between frontend and backend
- Reduce misunderstandings
- Parallelise work

Native AWS Integrations

- Easy to connect *directly* to AWS services
- No need for glue code

Error Reduction

- Less reliance on custom code
- Fewer potential points of failure
- Less boilerplate

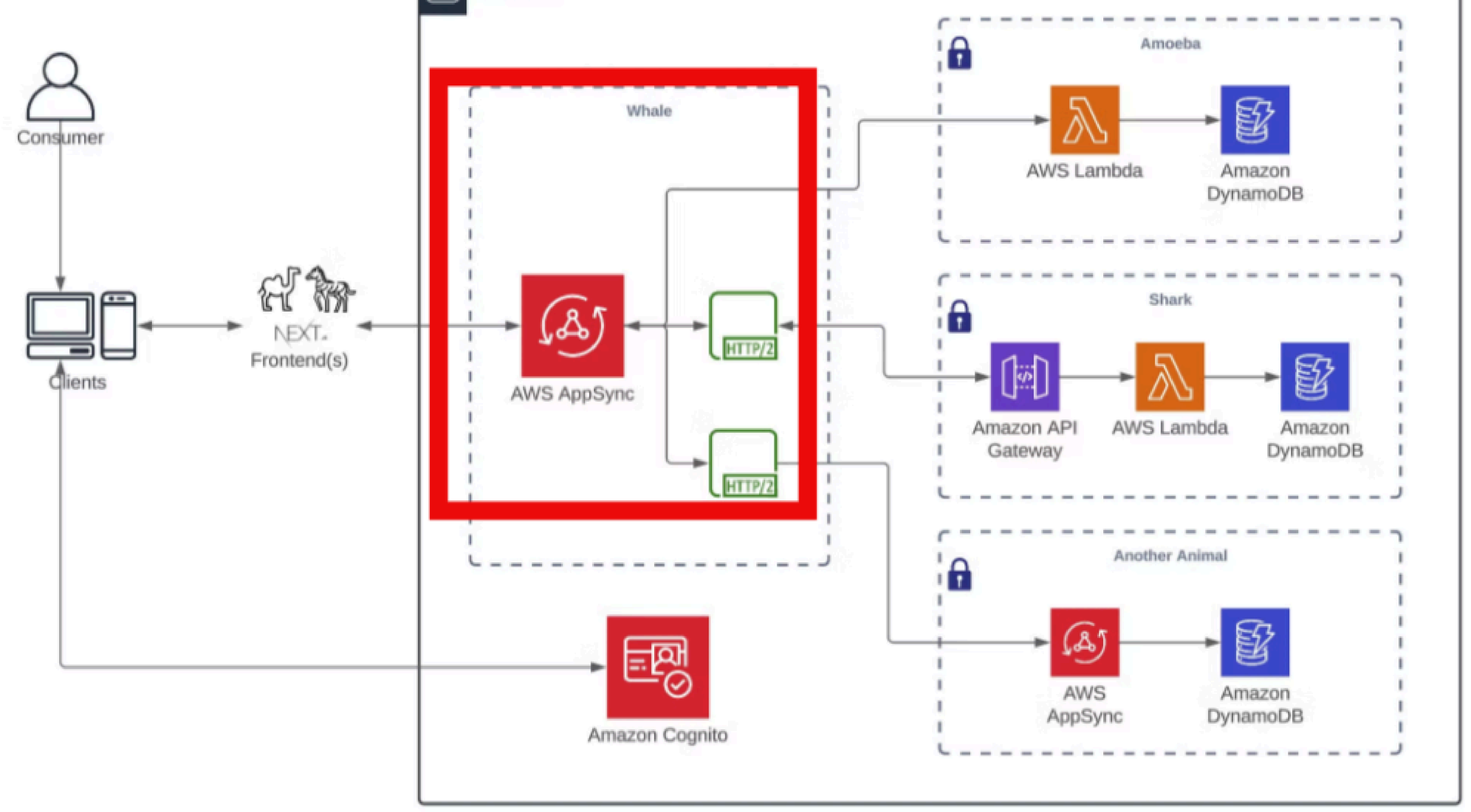
Cloud-Native Evolution

- The convenience of Django, Rails, Laravel
- ...but backed by the cloud!

(AppSync is not a battery-included framework)

Empowering Small Teams

- We rebuilt our entire API layer in a few weeks
- Made us much more productive and able to parallelise our work
- Has given us clarity over our data access patterns
- Has made it easier for frontend people to venture into the backend


Consumer
Clients
NEXT.
Frontend(s)
Whale
AWS AppSync
HTTP/2
HTTP/2
Amazon Cognito
Amoeba
AWS Lambda
Amazon DynamoDB
Shark
Amazon API Gateway
AWS Lambda
Amazon DynamoDB
Another Animal
AWS AppSync
Amazon DynamoDB

Thank You! 🙏

Useful Links

@MarcoTroisi

linkedin.com/MarcoTroisi

theserverlessmindset.com

trilo.io