
AWS
re:Invent

BLD206

# The pragmatic cloud developer

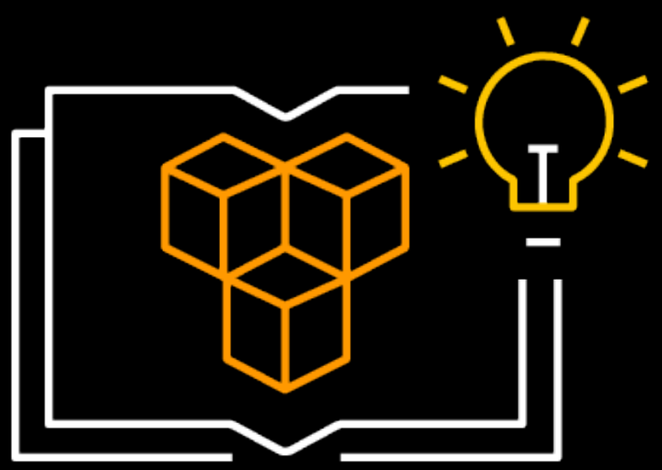
Colm MacCárthaigh
Sr. Principal Engineer
AWS

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# The Amazon Builders' Library

How Amazon builds and operates software



Architecture, software delivery and operations

By Amazon's senior technical executives and engineers

Real-world practices with detailed explanations

Content available for free on the website

# The Amazon Builders' Library

ARCHITECTURE

LEVEL 300



## Leader election in distributed systems

Author: Marc Brooker

Improving efficiency, reducing coordination, and simplifying architectures by using leader election.

SOFTWARE DELIVERY AND OPERATIONS

LEVEL 300



## Going faster with continuous delivery

Author: Mark Mansour

Automating the software testing and deployment process for speed and reliability

SOFTWARE DELIVERY AND OPERATIONS

LEVEL 400



## Implementing health checks

Author: David Yanacek

Automatically detecting and mitigating server failures without unintended consequences from fleet-wide false positives.

ARCHITECTURE

LEVEL 400



## Workload isolation using shuffle-sharding

Author: Colm MacCarthaigh

Shuffle Sharding is one of our core techniques for drastically limiting the scope of impact of operational issues

Many more at aws.amazon.com/builders-library



**Colm MacCárthaigh** @colmmacc · Oct 1

What are your favorite examples of **technical** things that don't make sense for everyone just because Amazon, Google, Microsoft, Facebook, or whoever does them?

77 42 119

# Infrastructure is not the thing, it is a means to a thing. Focus on the thing.





© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# Focus on how infrastructure choices serve your business

Measure customer-facing outcomes

Beware of ”tinkering” cultures, they can eat themselves

“Not invented here” and “Invented over there” are both extremes

Be pragmatic and questioning

# Think about your trade-offs

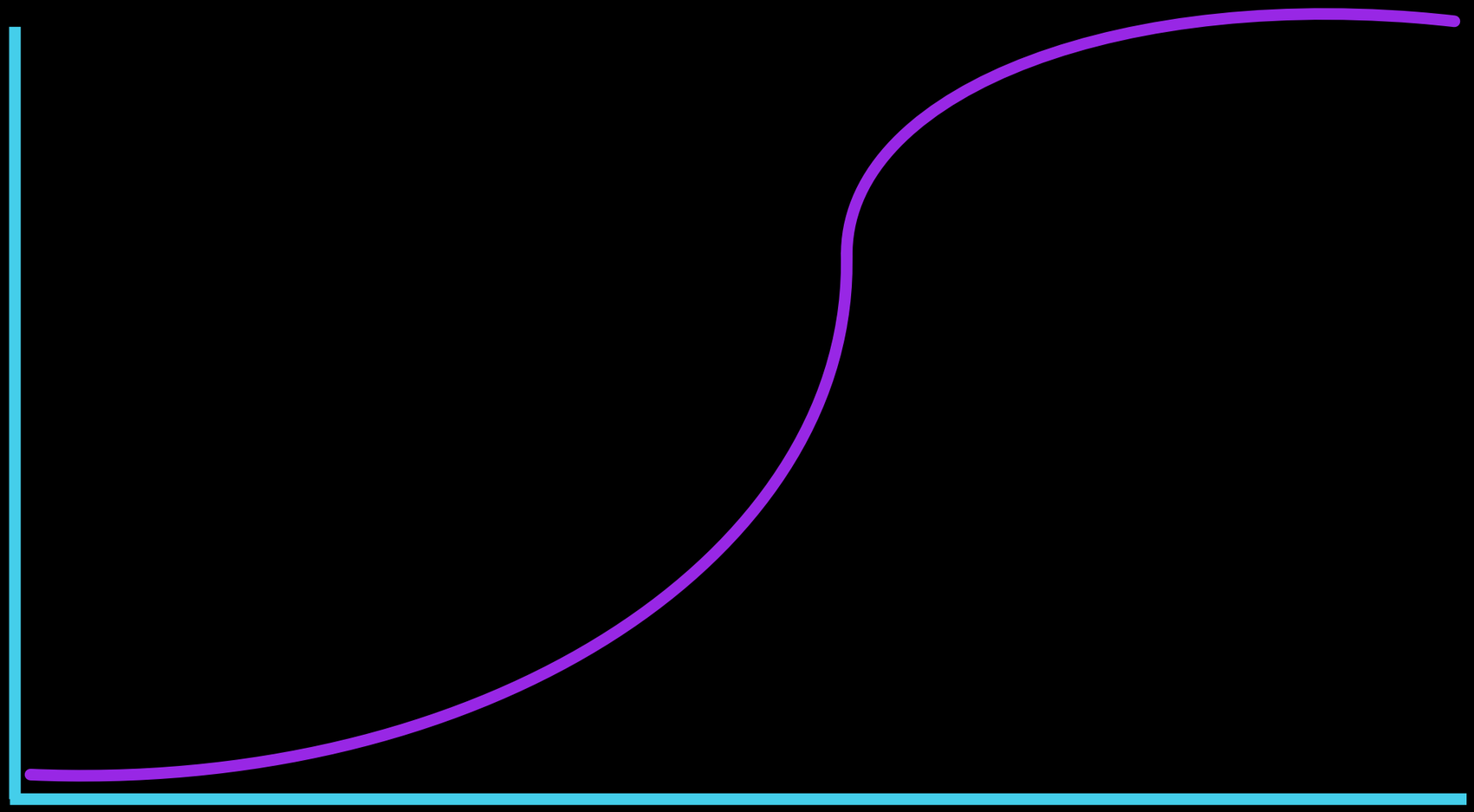
Availability

Scalability

Productivity

Cost

# Think about structure and growth



# Think about structure and growth

Startups

SME

Government agency

Enterprise

# What almost always makes sense?



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# What almost always makes sense?

Security scanning and patching

Working backups and tested restore

Resilience and redundancy

Serverless

# Five examples





© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# 1. Wrangling eventual consistency

## All Things Distributed

Werner Vogels' weblog on building scalable and robust distributed systems.

### Eventually Consistent - Revisited

By Werner Vogels on 22 December 2008 04:15 PM | Permalink | Comments ()

*I wrote a first version of this posting on consistency models about a year ago, but I was never happy with it as it was written in haste and the topic is important enough to receive a more thorough treatment. ACM Queue asked me to revise it for use in their magazine and I took the opportunity to improve the article. This is that new version.*

**Eventually Consistent - Building reliable distributed systems at a worldwide scale demands trade-offs between consistency and availability.**

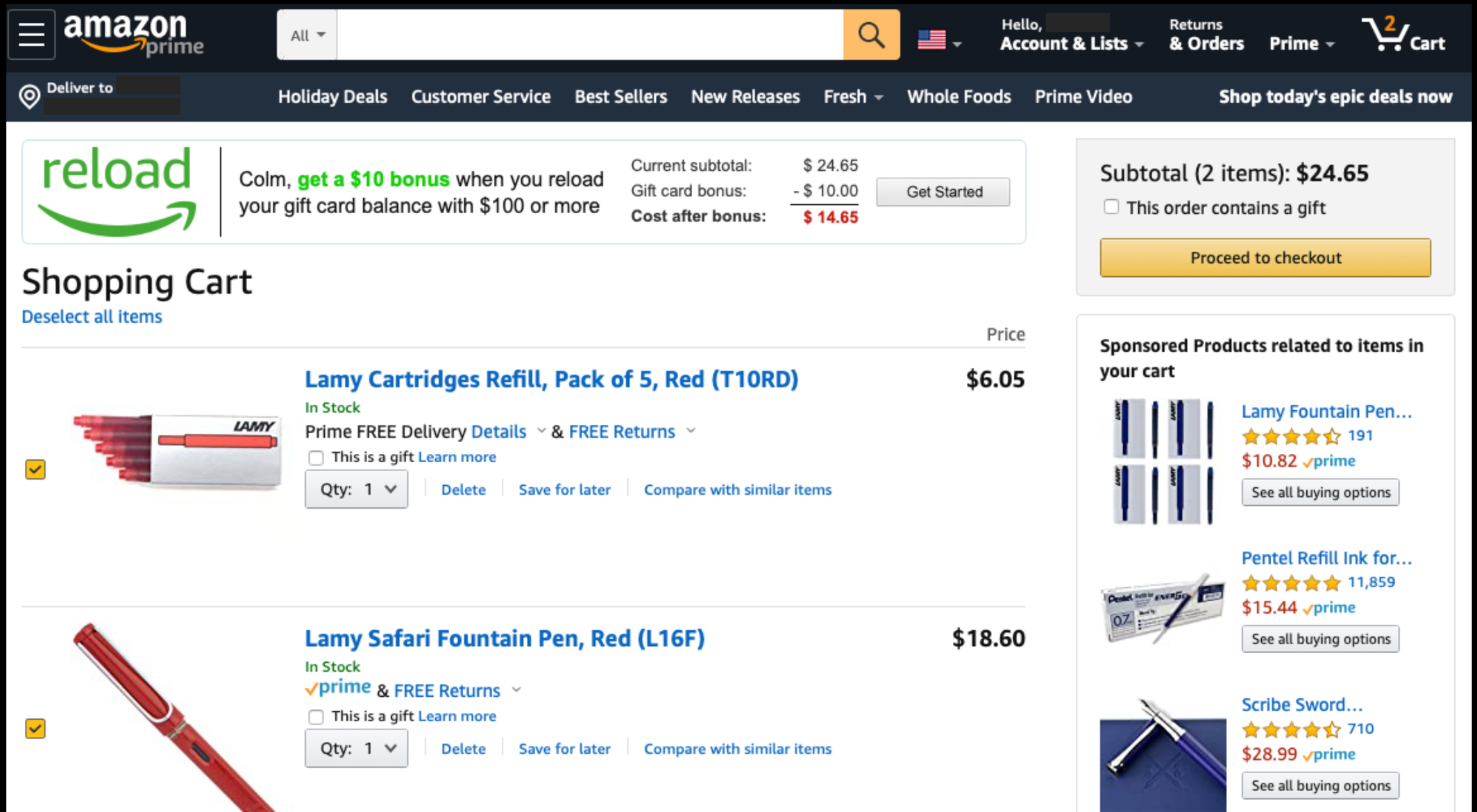
At the foundation of Amazon's cloud computing are infrastructure services such as Amazon's S3 (Simple Storage Service), SimpleDB, and EC2 (Elastic Compute Cloud) that provide the resources for constructing Internet-scale computing platforms and a great variety of applications. The requirements placed on these infrastructure services are very strict; they need to score high marks in the areas of security, scalability, availability, performance, and cost effectiveness, and they need to meet these requirements while serving millions of customers around the globe, continuously.



Contact Info

**Werner Vogels**
*CTO - Amazon.com*

# 1. Wrangling eventual consistency



# 1. Wrangling eventual consistency

Amazon learned a lot from building Amazon.com

A website of that scale and velocity requires eventual consistency

Many early AWS services emphasized eventual consistency and eventually consistent patterns

That doesn’t mean you need them; strong consistency is easier to program against

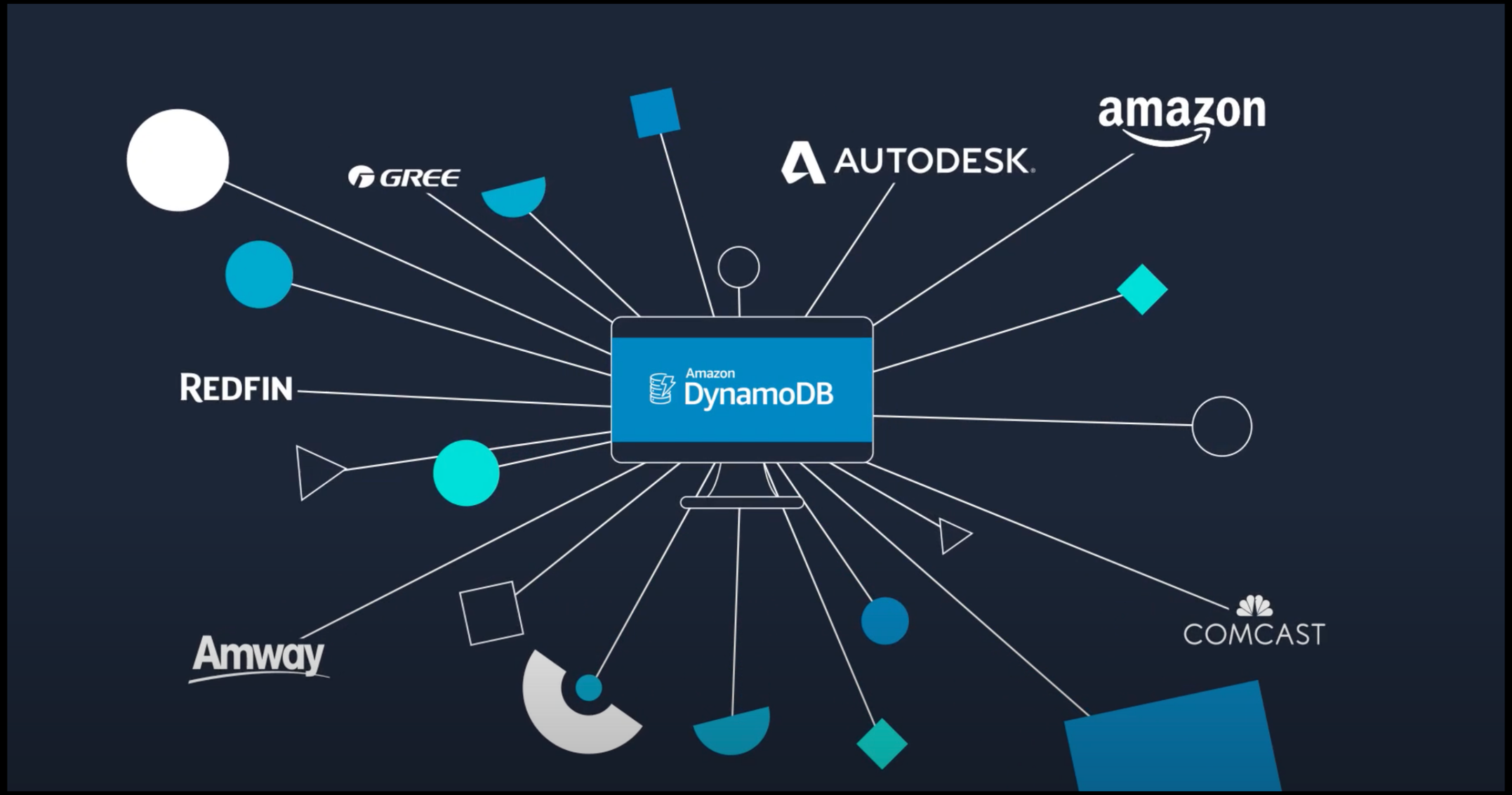
# 1. Wrangling eventual consistency

Amazon Simple Storage Service, Amazon Aurora, Amazon DynamoDB offer read-after-write consistency

Asynchronous replication available for workloads that demand it

Global replication

# 2. NoSQL


GREE
AUTODESK
amazon
REDFIN
Amazon
DynamoDB
Amway
COMCAST

# 2. NoSQL

AWS services are required to be ultra-scalable. NoSQL is more scalable and safe than SQL. It lends itself to horizontal scaling and it makes scaling points and bottlenecks explicit and more obvious.

NoSQL is not just encouraged at AWS, but all new uses of SQL require approval from Charlie Bell.

We still use SQL for many non-critical backend applications.

# 2. NoSQL

SQL is still awesome and solves a lot of problems

There’s hidden complexity in the database, but there’s a lot of convenience too

If you’re comfortably away from the margins, SQL can be a great solution

We have many, many happy customers using SQL with Amazon Relational Database Service and Aurora

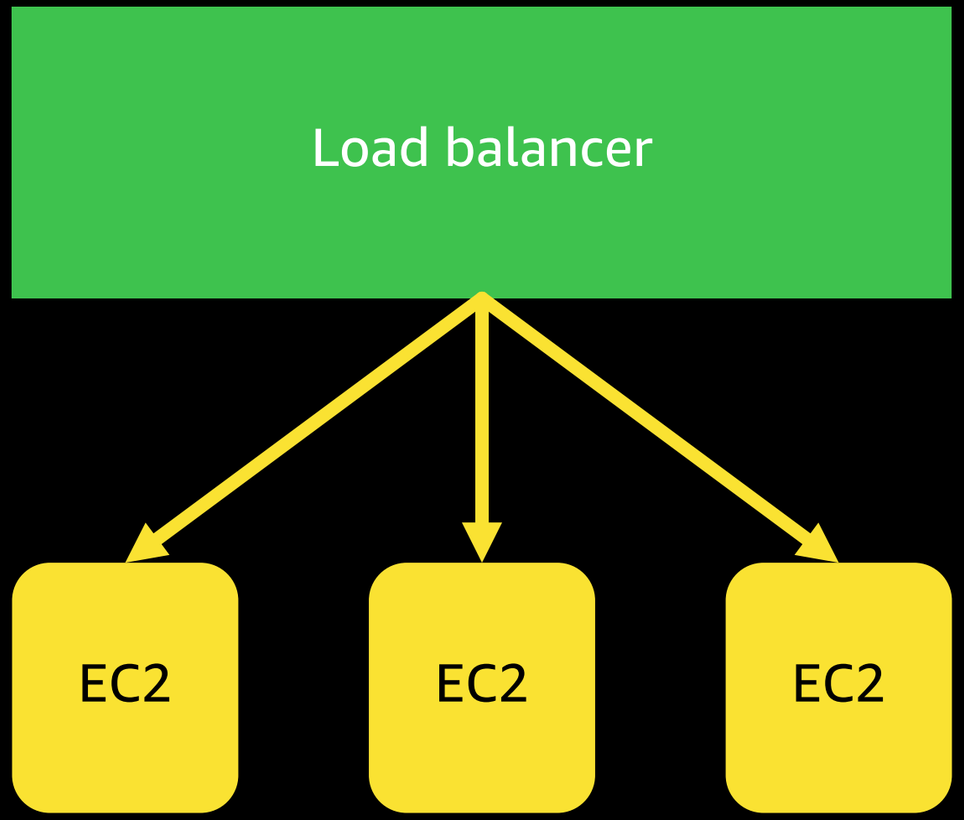
# 3. Microservices

Microservices are an organizational tool as much as they are a technical tool

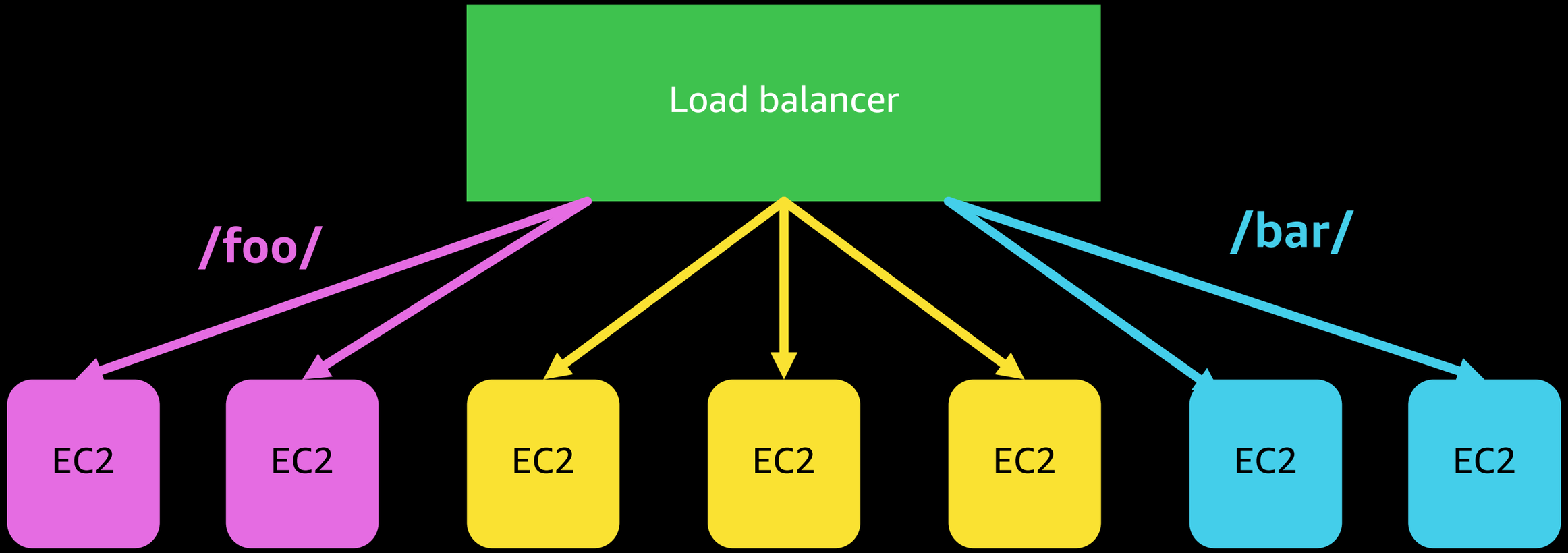
Microservices don’t make sense for all organizational models

Not a one way door; Amazon API Gateway, Application Load Balancer, and Amazon CloudFront each make it possible to split up a monolithic service into microservices

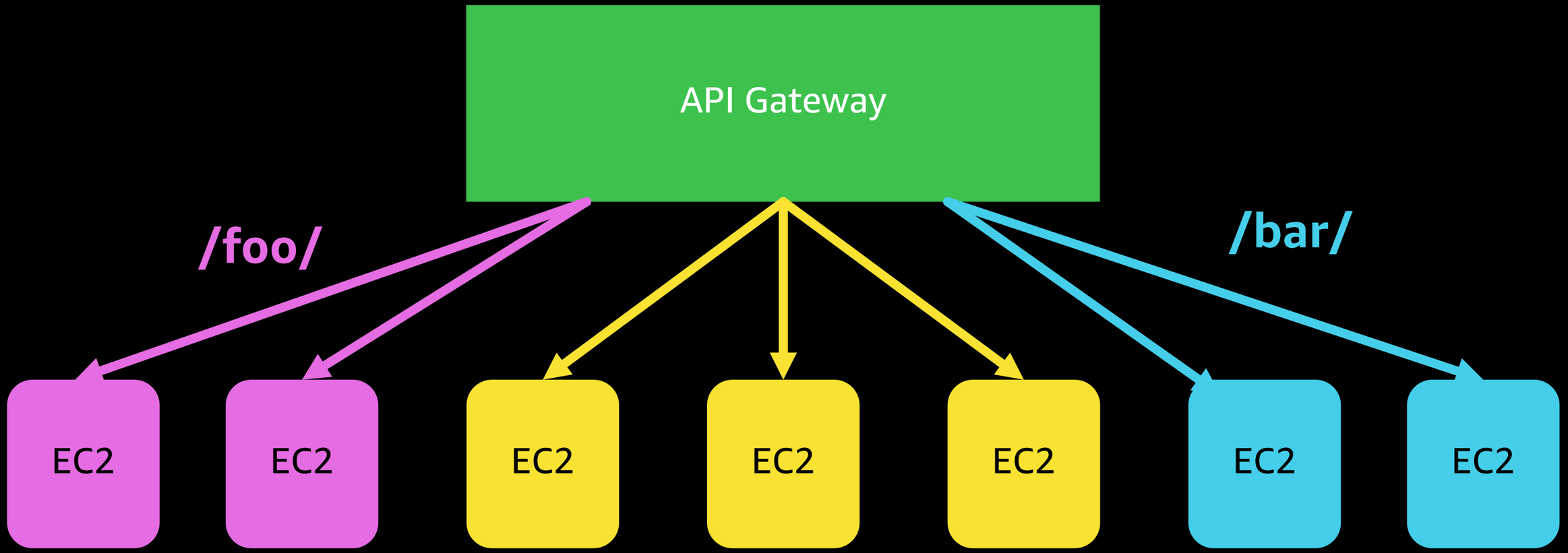
# 3. Microservices



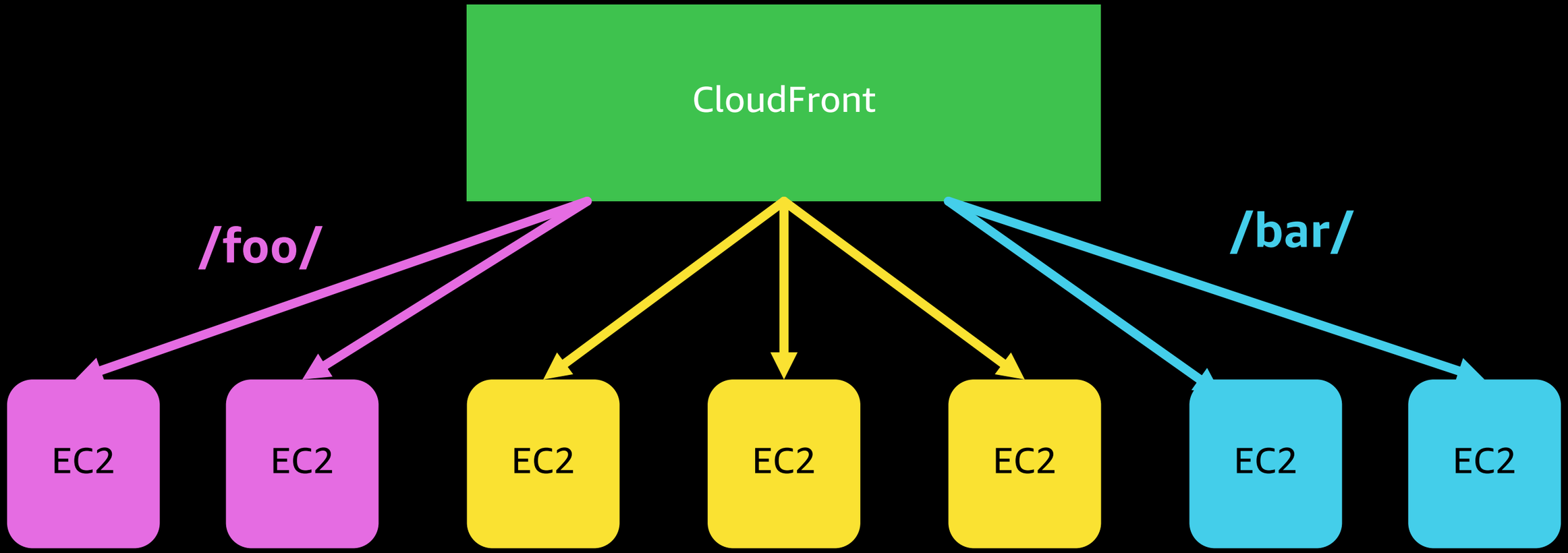
# 3. Microservices



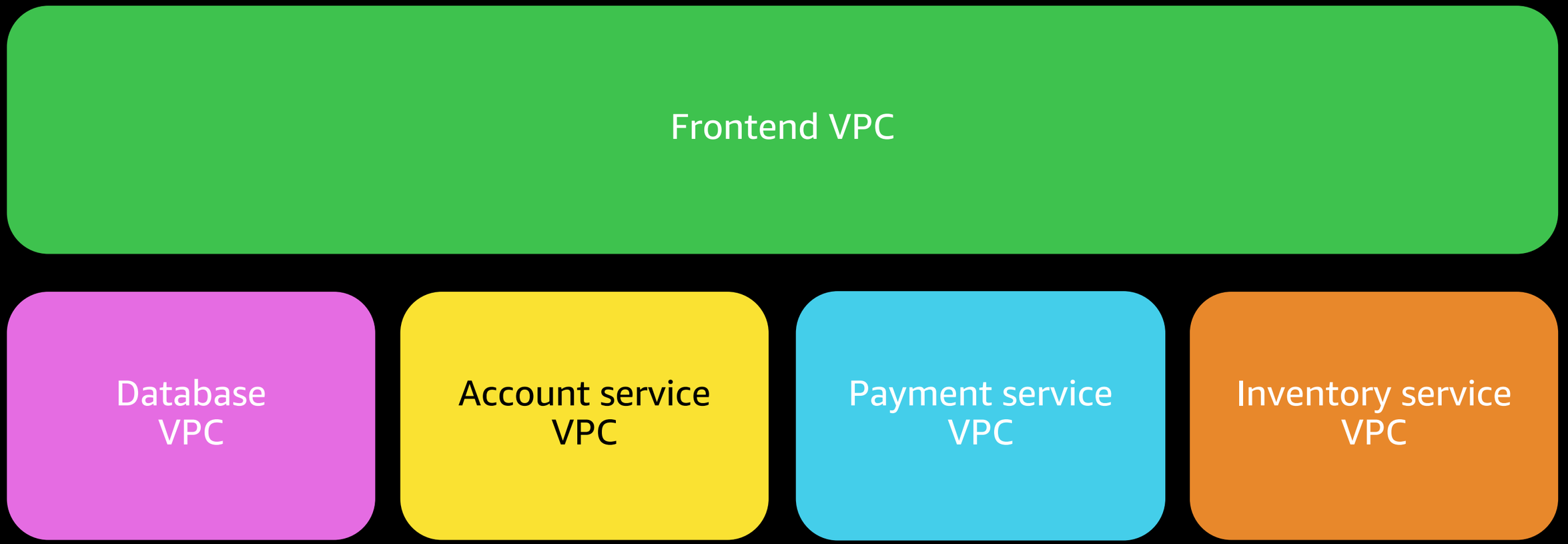
# 3. Microservices



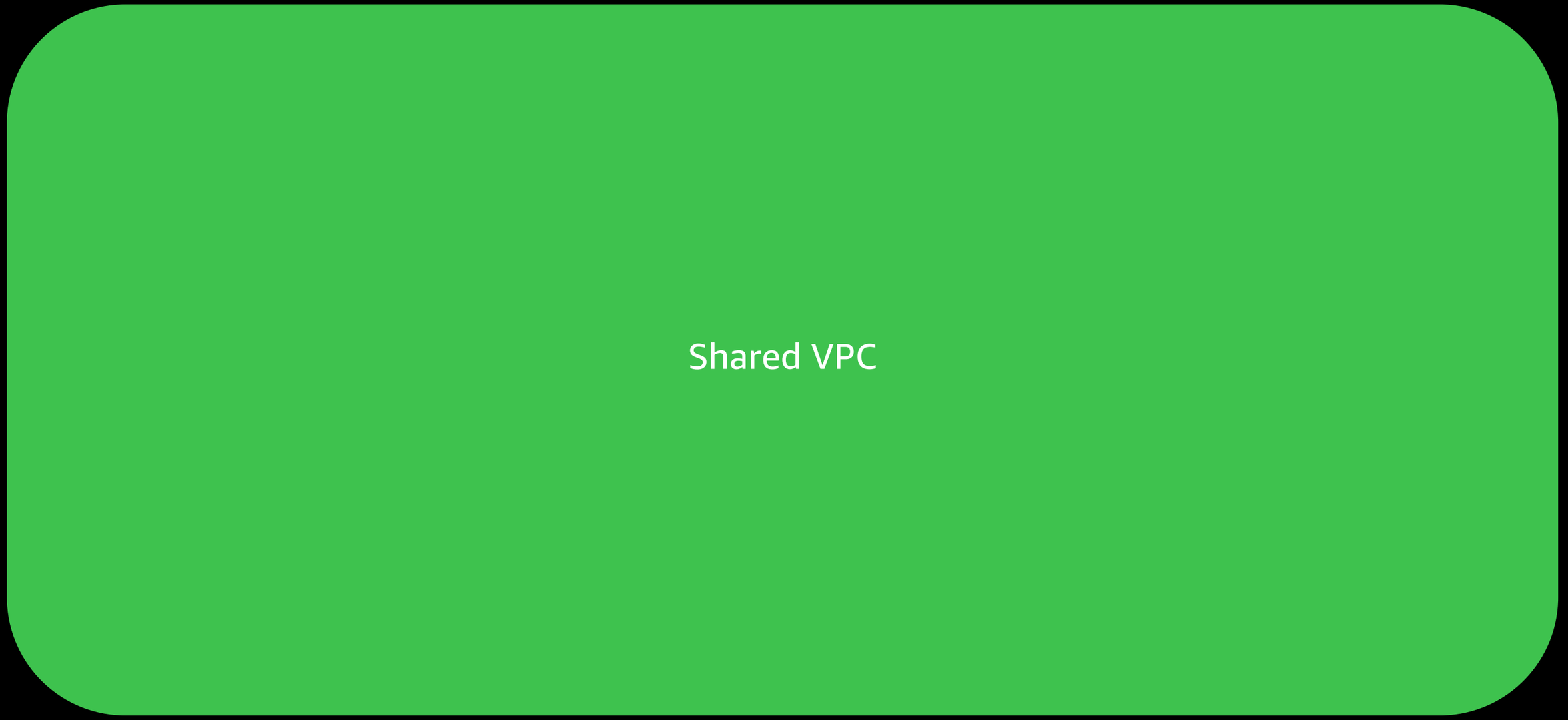
# 3. Microservices



# 4. Network segmentation



# 4. Network segmentation



# 5. SRE

Site Reliability Engineering (SRE) is awesome, but you don’t need it to use cloud services

Many customers have dedicated infrastructure ops teams

Many customers leave it to development teams

Many customers have something in between

# 5. SRE

Organizational history and expertise matter a lot

Path dependence plays a role

Organizational structure can be as diverse and creative as architectures

No one “right” answer or size for everyone

# Honerable mentions

Kubernetes

Monorepos

Full CI/CD

Multiple regions

Infrastructure as code

# Thank you!



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



# Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.