The Future of Modern Application Development

Chris Munns – Principal Developer Advocate – AWS Serverless

About me:

Chris Munns - munns@amazon.com, @chrismunns

- Principal Developer Advocate - Serverless
- New Yorker
- Previously:
  - AWS Business Development Manager – DevOps, July ’15 - Feb ‘17
  - AWS Solutions Architect Nov, 2011- Dec 2014
  - Formerly on operations teams @Etsy and @Meetup
  - Little time at a hedge fund, Xerox and a few other startups
- Rochester Institute of Technology: Applied Networking and Systems Administration ’05
- Internet infrastructure geek


Why are we here today?
https://secure.flickr.com/photos/mgifford/4525333972


The Future

Two main topic areas for today:

Customer facing applications

Backend / Supporting applications

aws

https://secure.flickr.com/photos/psd/4389135567/

Two main topic areas for today:

Customer facing applications

- 🚀 Reactive web interfaces
- 🚀 PWA
- 🚀 Voice

aws

Two main topic areas for today:

🚀 Serverless
🚀 APIs
🚀 GraphQL

Backend / Supporting applications

aws

Reactive Web Interfaces

Not a new concept!

Popularized by JavaScript frameworks like AngularJS from Google(2010), React from Facebook(2013), Vue.js(based on Angular concepts, 2014).

- Provide interface libraries and data request handling capabilities
- Aim to make interface development easier for multiple platforms/ devices
- Some of the most popular open source projects on GitHub
- Huge ecosystem of add-ons, tooling, books, etc.

aws

The foundation for your cloud-powered mobile & web apps

GET STARTED

Amplify your apps. Build on a flexible, scalable, and reliable serverless backend.

AWS Amplify

- JavaScript Library for frontend development
  - JavaScript & TypeScript
  - React + React Native
  - Angular, Vue
  - Ionic
- Declarative interface across different categories
- Open Source (Apache) github.com/aws/aws-amplify

✓ Authentication
✓ Analytics
✓ API
✓ Storage
✓ Cache
✓ PubSub
✓ Interactions

aws

React withAuthenticator HOC

```
Import {
      withAuthenticator
} from ‘aws-amplify-react’

…

export default withAuthenticator(App);
```

- ✓ React HOC
- ✓ 100% Scaffold UI
- ✓ State Management
- ✓ MFA / SMS / Email
- ✓ Credential Management

aws

User Authentication

The authentication component utilizes the styles outlined in the styleguide to provide a minimal styled experience out of the box.

NEW USER ACCOUNT

Create a new account

Username *

my_username

Password *

Create a password

Email Address *

janedoe@email.com

Phone Number

(555) 555-1212

Have an account? Sign in

CREATE ACCOUNT

RECOVER USER ACCOUNT

Reset your password

You will receive a verification code from Amplify to reset your password.

Username *

my_username

Forgot username

SUBMIT

Reset your password

Enter the code you received from Amazon and set a new password.

Verification Code

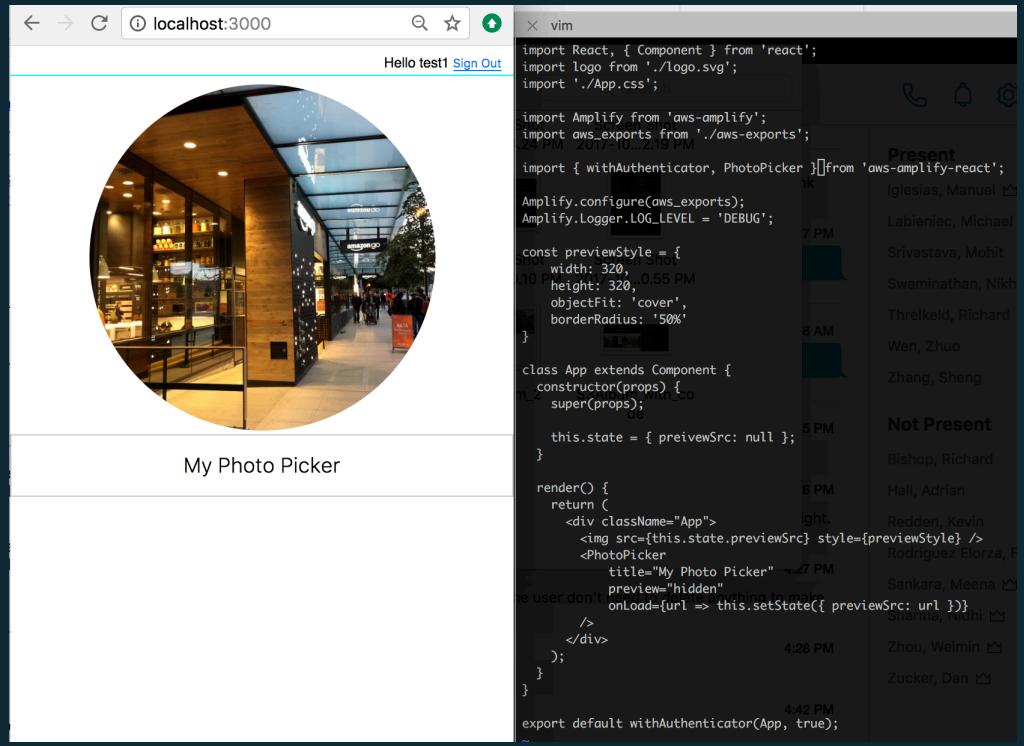
Enter code

React <PhotoPicker /> Component

```
import {
  PhotoPicker
} from ‘aws-amplify-react’

render(
  <PhotoPicker onPick={
        data => doSomething(data)
  } />
)
```

- ✓ Scaffold UI
- ✓ Customizable UI
- ✓ State Events
- ✓ Credential Management



Progressive Web Applications

Popularized by Google on Android, PWA's merge web and native applications for both mobile and desktop users:

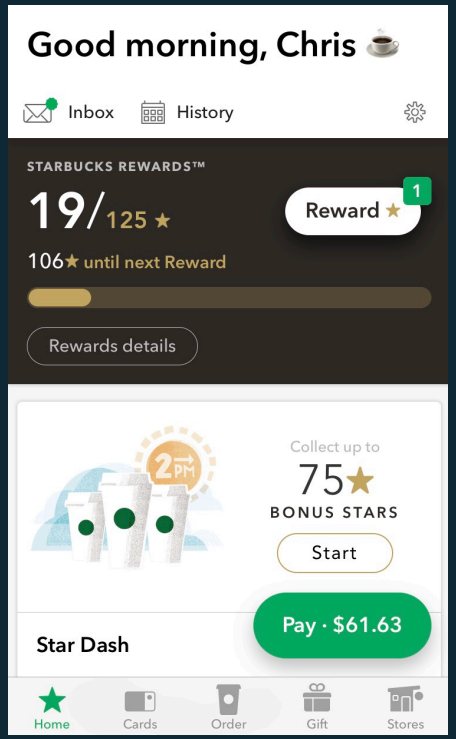
- Work via "Service worker" that runs in Android or Chrome, written in Javascript:
    - Service worker is like a proxy client that provides client side caching of web content and request management
    - Lesser support in iOS today, which is maybe the biggest blocker
- Maintain tenets of being "reliable, fast, engaging"
- Allow for off-line usage and re-sync of data
- Typically way faster, way smaller, and easier to develop than native, applications.

aws

Progressive Web Applications: Starbucks

Perhaps one of the most popular and drastic examples:

- The Starbucks mobile app is considered one of, if not the, largest mobile pay application in the U.S. ("23.4 million people aged 14 and above will use the app to make a POS purchase at least once every six months," #1)
- Moved to a PWA in late 2017 (#2)
  - New app is .4% the size of the previous Android application
  - Faster than iOS native app
  - Doubled daily active users
  - Moved to GraphQL backend
  - Shared code base for Android, Windows mobile, Desktop



#1: https://www.retailtouchpoints.com/topics/mobile/study-starbucks-mobile-app-usage-exceeds-popular-payment-apps
#2: https://formidable.com/work/starbucks-progressive-web-app/

Service Workers

AWS Amplify *ServiceWorker* class enables registering a service worker in the browser and communicating with it via *postMessage* events, so that you can create rich offline experiences with Push APIs and analytics.

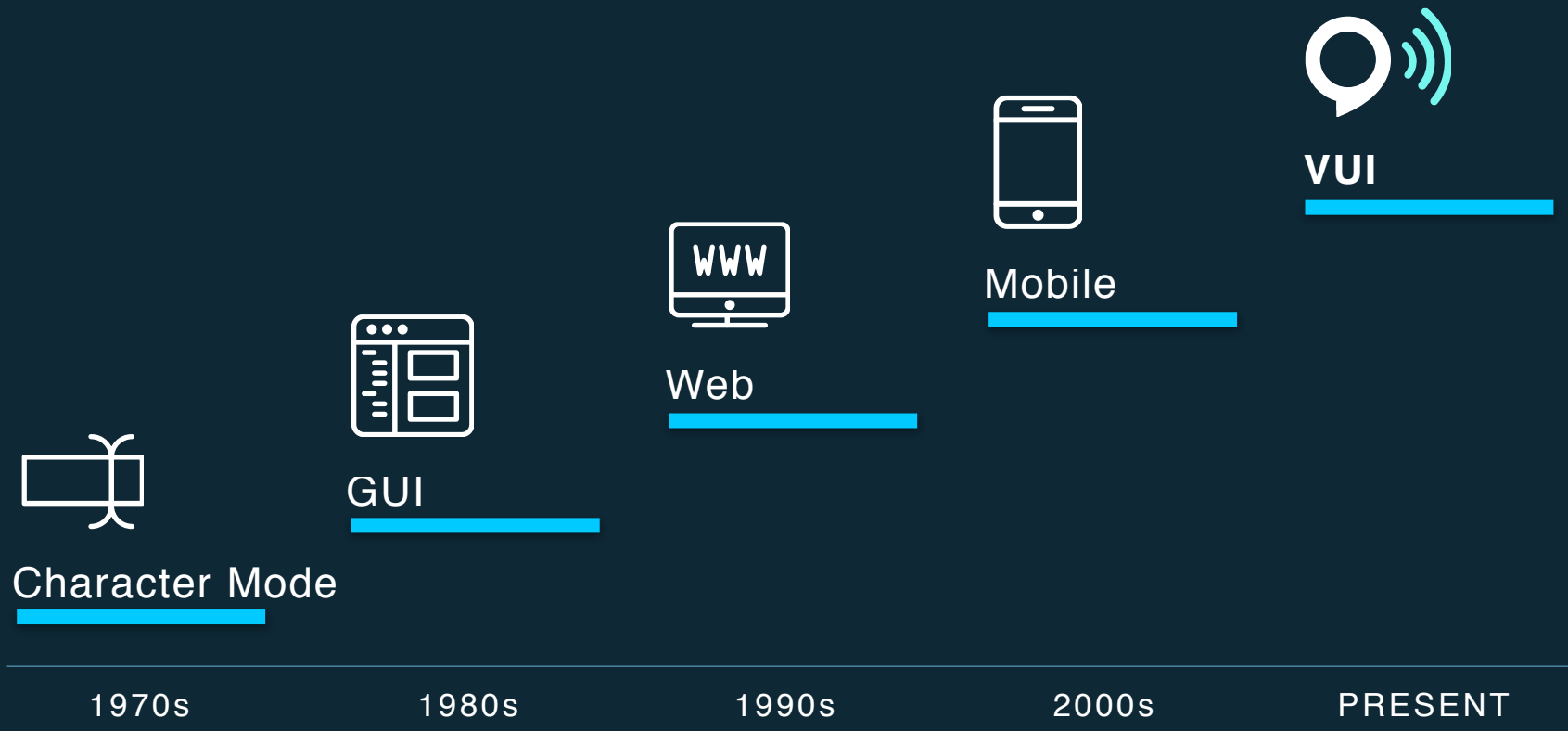
After registering the service worker, the ServiceWorker module will listen and attempt to dispatch messages on state changes, and it will record analytics events based on the service worker's lifecycle.

> *postMessage* events are currently not supported in all browsers. For details and to learn more about Service Worker API, please visit here.

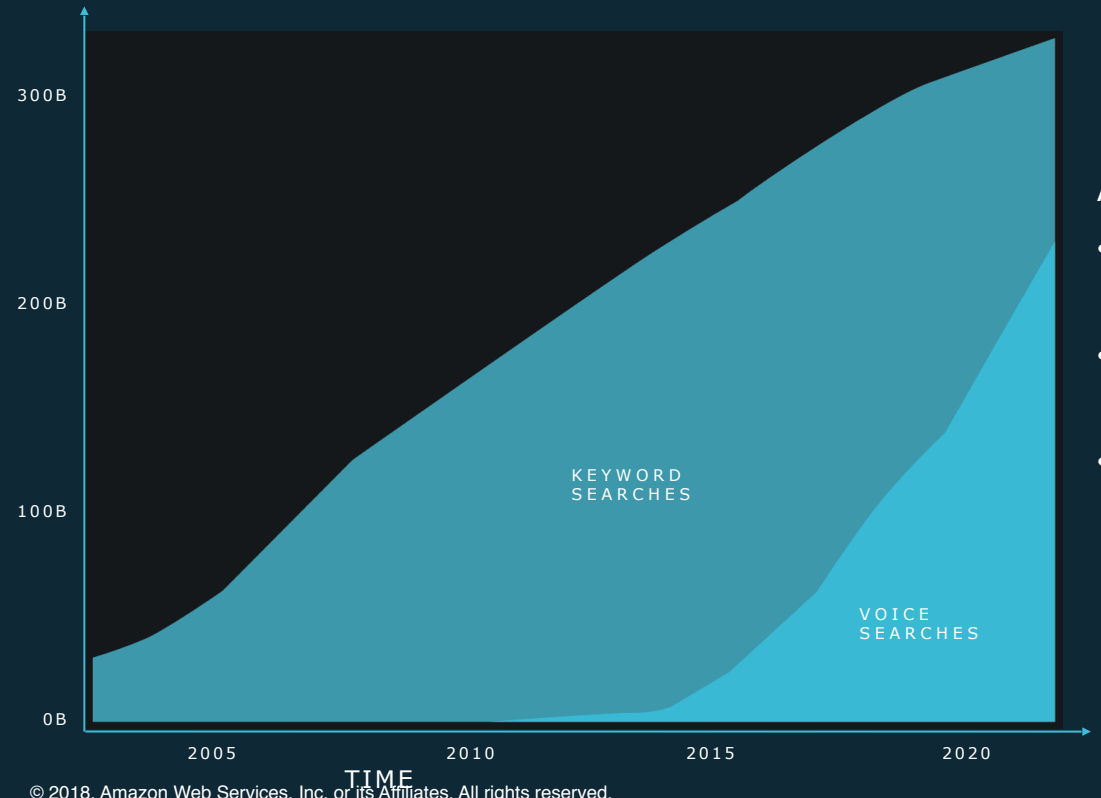
Installation

Import *ServiceWorker* and instantiate a new instance (you can have multiple workers on different scopes):

UI's Have Evolved Over the Past 5 Decades



Voice Is Now THE NEW STANDARD



WORLD WIDE SEARCHES PER MONTH

A Massive shift in voice has already begun.

- In 2014, voice search traffic was negligible. Today it exceeds 10% of all search traffic.
- Virtual assistants exceed 50B voice searches per month.
- By 2020, over 200 billion searches per month will be done with voice.

Source: MindMeld


10:45
POWER LEVEL
TIME DEFROST
WEIGHT DEFROST
1 2 3
4 5 6
7 8 9
POPCORN (WEIGHT)
0
KITCHEN TIMER
PAUSE
Stop
ASK ALEXA
START
+30sec.


10:45
POWER LEVEL
TIME DEFROST
WEIGHT DEFROST
1 2 3
4 5 6
PAUSE
Stop
ASK ALEXA
START

Alexa Made Voice the Mainstream UI at Home

Alexa for Business

Transform your workplace with voice

Smarter Workplace

- Conference Rooms
- Warehouses
- Front Desks

Workplace Productivity

- Business Calling
- Calendar Management
- 3rd Party Enterprise Application

Centralized Admin Control

- Device Deployment & Management
- User Management
- Private Skills

Alexa for Business Now Lets Users Book Conference Rooms Using Alexa

Posted On: Oct 9, 2018

Starting today, Alexa for Business lets users check availability and reserve conference rooms using Alexa. Finding and booking a conference room for a last minute meeting is frequently a stressful, time-consuming task for many people. With Alexa for Business, users can check the current or future availability of the conference room they are in by asking, “Alexa, is this room free?”, or “Alexa, is this room free at 4?”. Or, they can reserve the room by saying “Alexa, book this room for half an hour”, or “Alexa, reserve this room at 2”. Conference rooms that are open might actually be reserved for a meeting, so users can identify who owns the current reservation by asking, “Alexa, who booked this room?”, and then find out if the meeting is actually happening or not.

IT administrators can enable this feature by linking their Microsoft Exchange, Office 365, or Google G-Suite calendaring systems in the Alexa for Business console, and allowing read/write permissions. Existing customers must re-link their Office 365 and G Suite calendars, and update their Microsoft Exchange permissions.

Alexa for Business lets you efficiently introduce Alexa to your organization. It provides you the tools to manage Alexa devices, users, and skills at scale. This feature is available in the Alexa for Business console in all AWS Regions where the service is available. For more information, see the Alexa for Business documentation.

Two main topic areas for today:

Customer facing applications

- 🚀 Reactive web interfaces
- 🚀 PWA
- 🚀 Voice

aws

Two main topic areas for today:

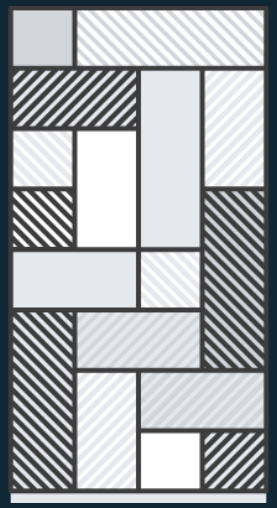
🚀 Serverless
🚀 APIs
🚀 GraphQL

Backend / Supporting applications

aws

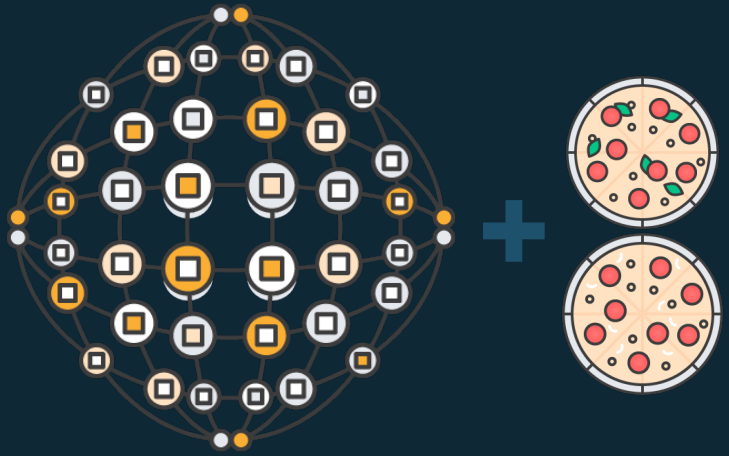
Development transformation at Amazon:

1994-2001



monolithic architecture +
hierarchical organization

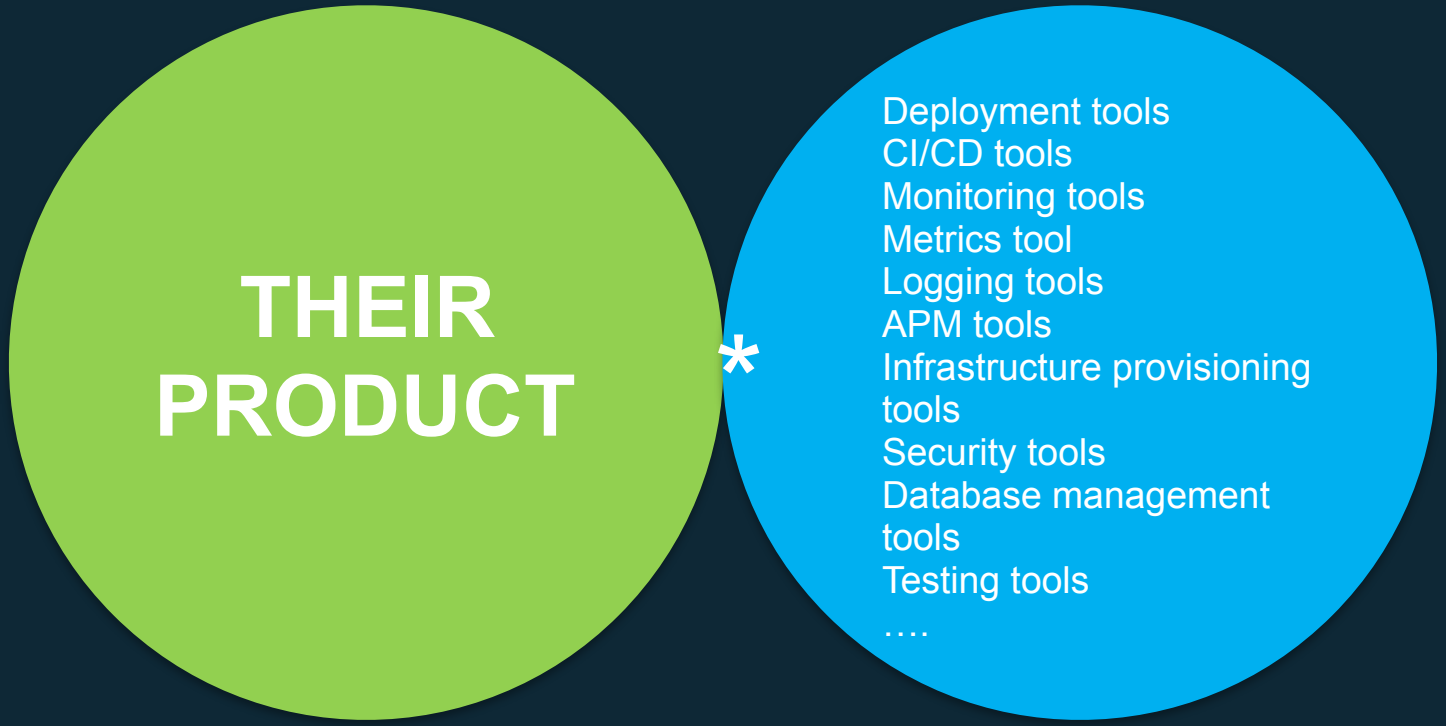
2002+



decoupled services +
2 pizza teams

aws

2 Pizza Team Responsibility Venn Diagram

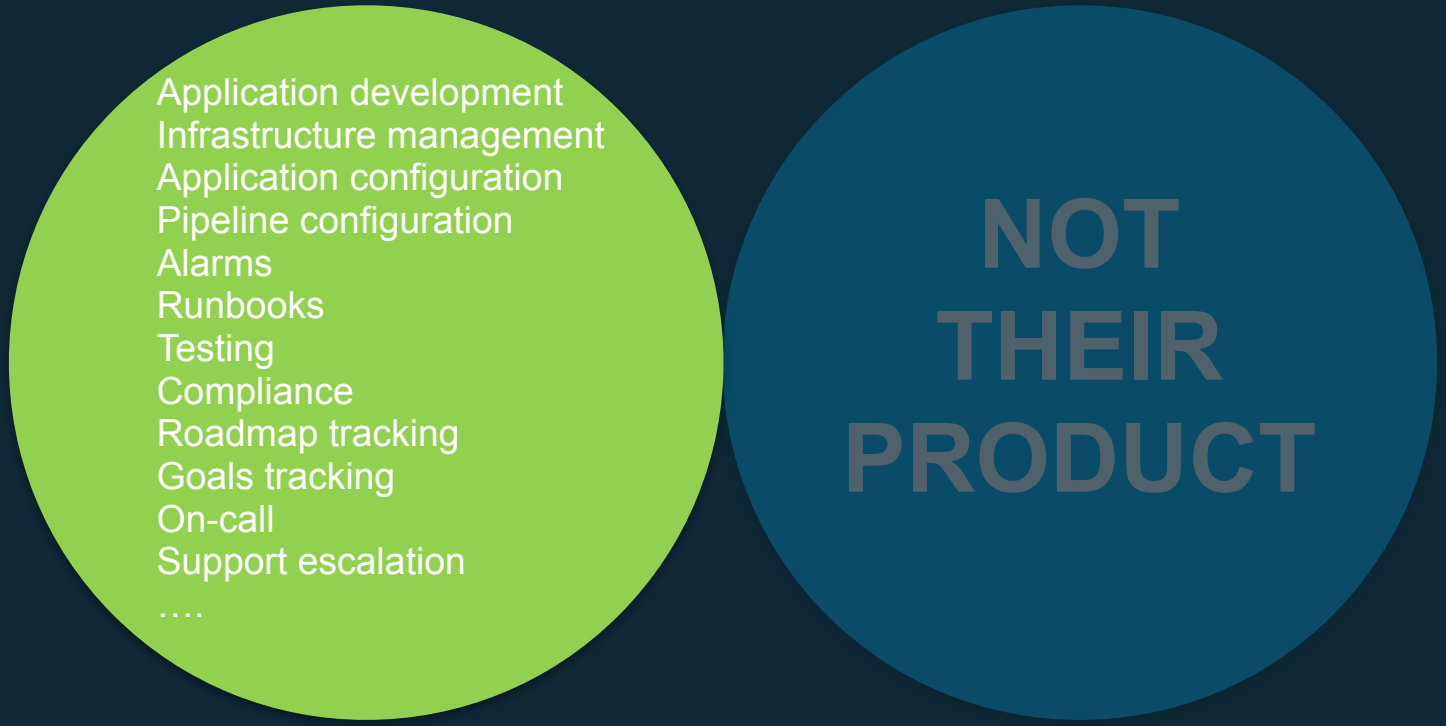


Responsible for

Not responsible for

*Unless their product belongs in the blue

2 Pizza Team Responsibility Venn Diagram



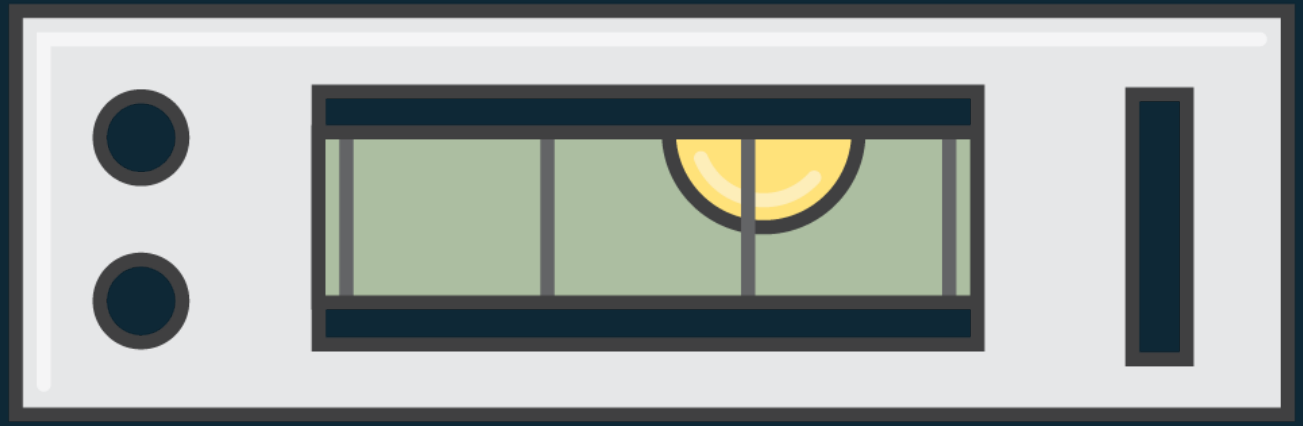
Responsible for

Not responsible for

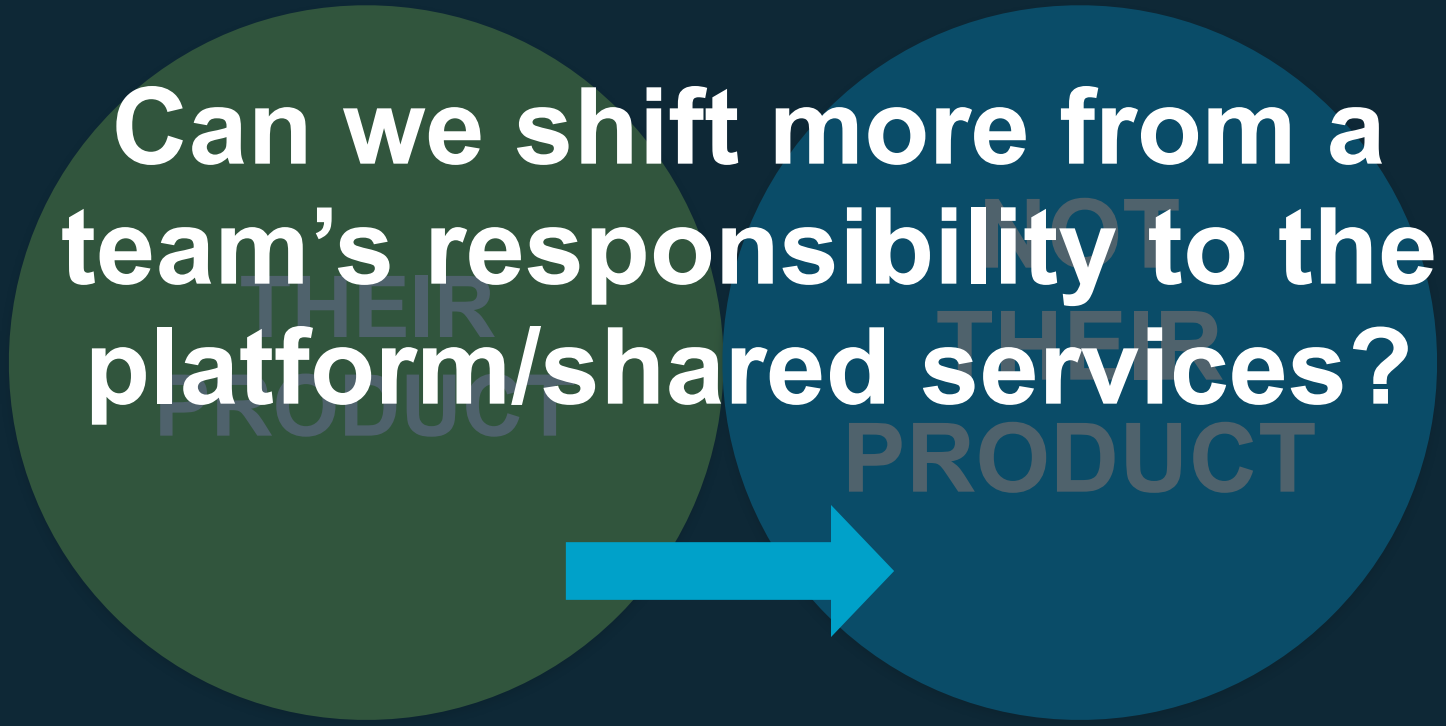
*Unless their product belongs in the blue

Determining the right balance

The more time spent on operational tasks, the less time spent on development tasks



2 Pizza Team Responsibility Venn Diagram



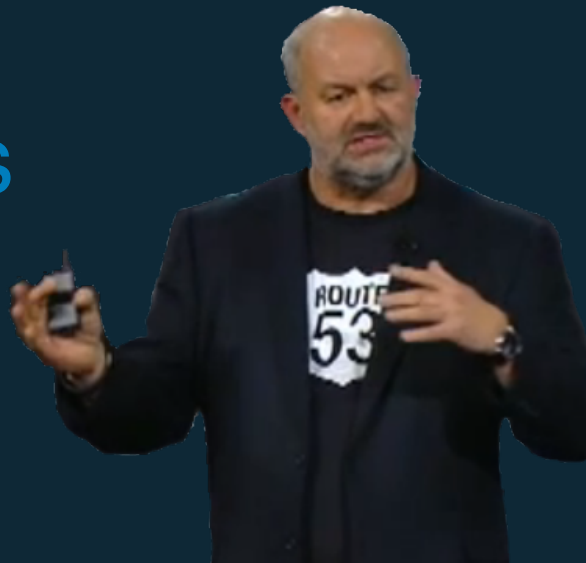
Can we shift more from a team's responsibility to the platform/shared services?

Responsible for

Not responsible for

No server is easier to manage than "no server”.

Dr. Werner Vogels
Amazon CTO



Serverless means…

No servers to provision or manage

Scales with usage

Never pay for idle

Availability and fault tolerance built in

aws

Serverless applications

EVENT SOURCE

Changes in data state

Requests to endpoints

Changes in resource state

FUNCTION

Node.js
Python
Java
C#
Go

SERVICES (ANYTHING)

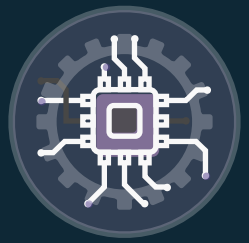
Common Lambda use cases

Web Applications

- Static websites
- Complex web apps
- Packages for Flask and Express

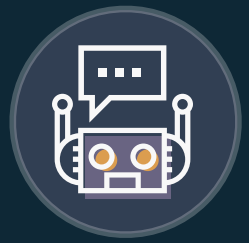
Backends

- Apps & services
- Mobile
- IoT



Data Processing

- Real time
- MapReduce
- Batch

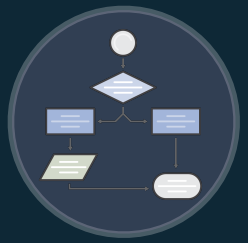


Chatbots

- Powering chatbot logic

Amazon Alexa

- Powering voice-enabled apps
- Alexa Skills Kit



IT Automation

- Policy engines
- Extending AWS services
- Infrastructure management

aws

Serverless Architectures

1. File put into bucket

2. Lambda invoked

1. Data published to a topic

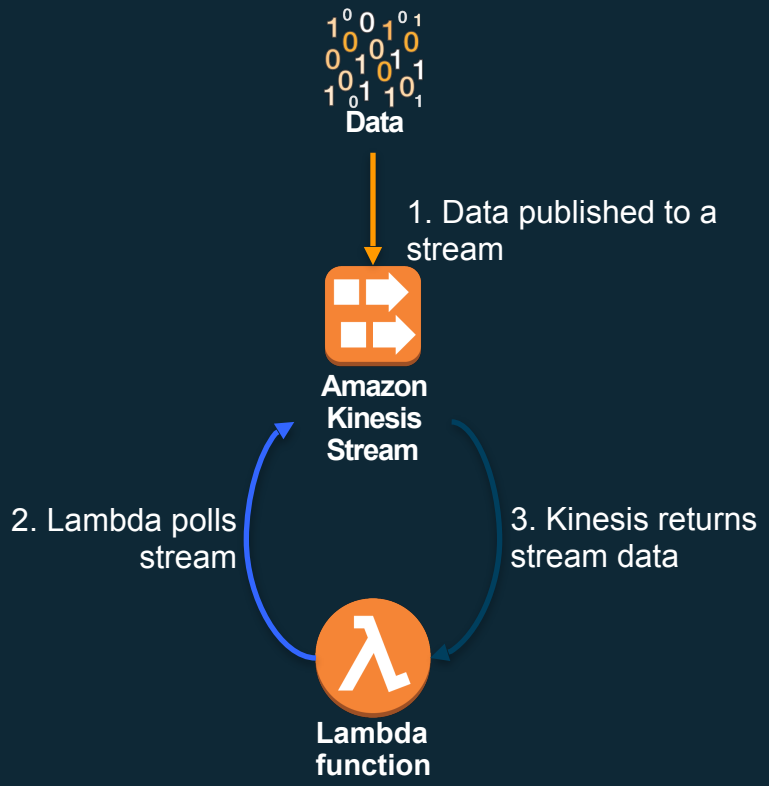
2. Lambda invoked

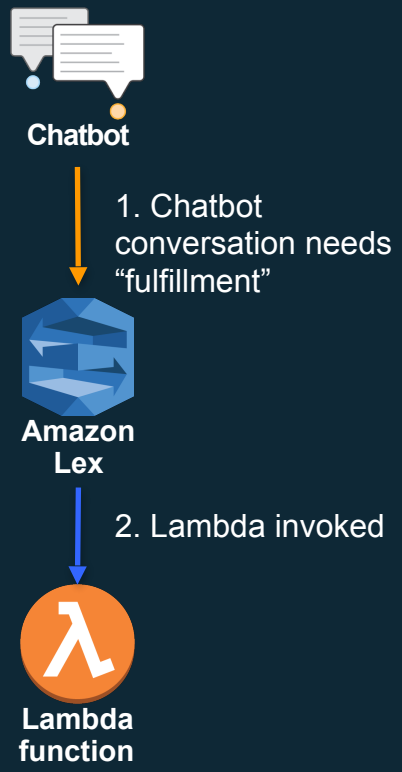
1. Message inserted into to a queue

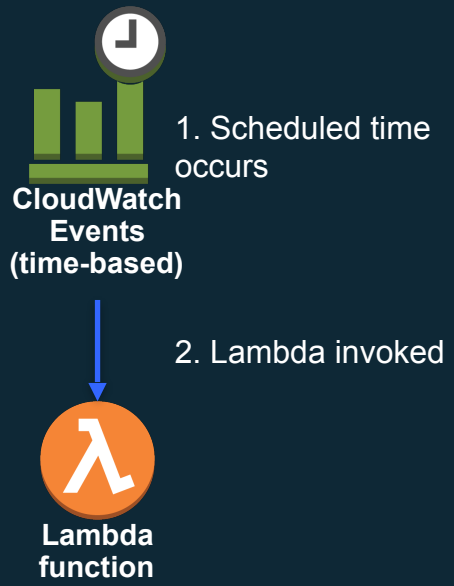
2. Lambda polls queue and invokes function

3. Function removes message from queue

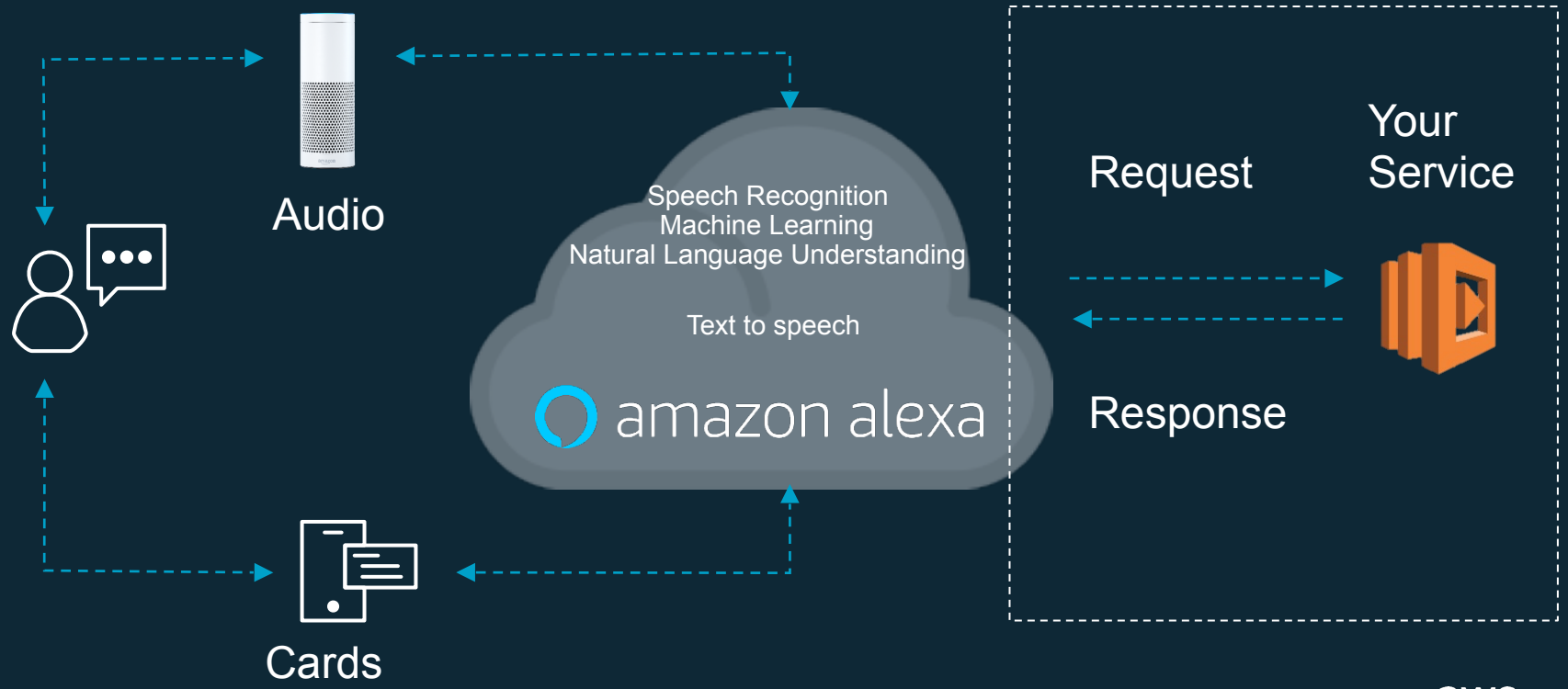
Serverless Architectures







Alexa Skills Kit: Requests and Responses

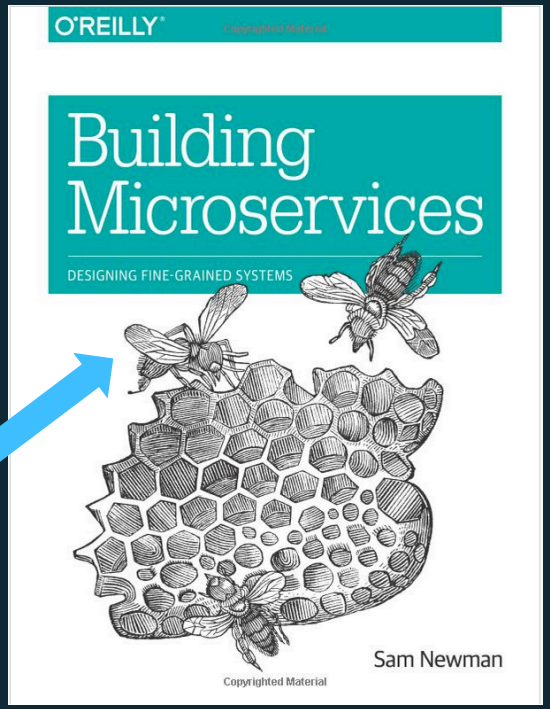


Host a Custom Skill as an AWS Lambda Function

The easiest way to build the cloud-based service for a custom Alexa skill is by using AWS Lambda, an Amazon Web Services offering that runs your code only when it's needed and scales automatically, so there is no need to provision or continuously run servers. You upload the code for your Alexa skill to a Lambda function and Lambda does the rest, executing it in response to Alexa voice interactions and automatically managing the compute resources for you.

Table of Contents

“Software is Eating the World” – M. Andreessen
“APIs are Eating Software” – Dr. S. Willmott



Fun fact: Apis is the “Genus” for Honey Bees

APIs power all of these:

iPhone
<11 years

iPad
<8 years

Tesla
Model S
<6 years

iWatch
<4 years

Echo
<4 years

Illumina DNA
Sequencer
<4 years

---

amazon.com Prime

Amazon
Prime
~13 years

NETFLIX

Netflix
Streaming
~11 years

airbnb

Airbnb
~10 years

UBER

Uber
<9 years

Square

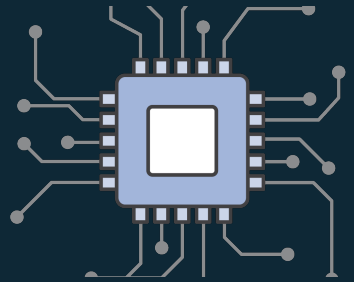
Square
<9 years

slack

Slack
< years

*Ages might be slightly off

Amazon API Gateway



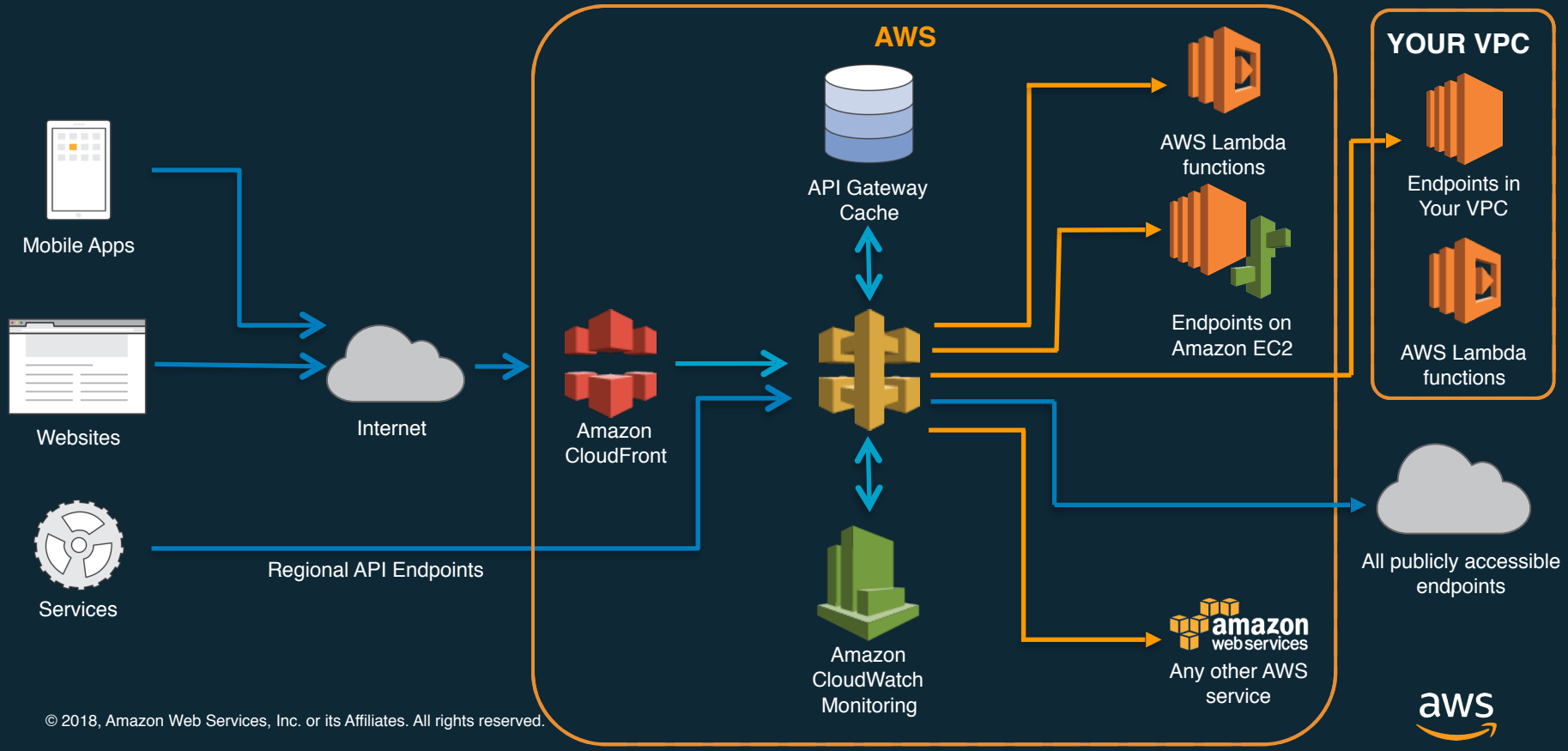
Create a unified API frontend for multiple micro-services

DDoS protection and throttling for your backend

Authenticate and authorize requests to a backend

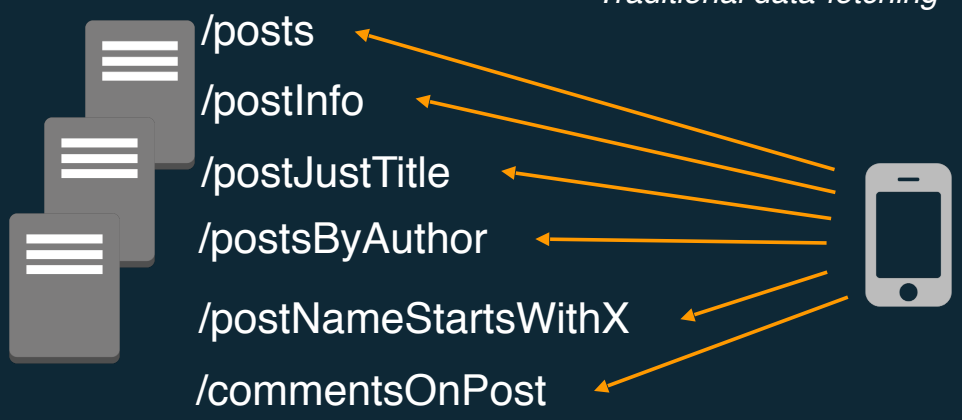
Throttle, meter, and monetize API usage by 3rd party developers

Amazon API Gateway



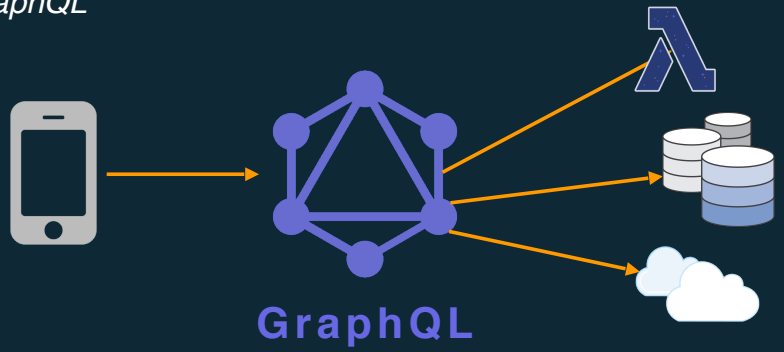
REST API vs GraphQL

Traditional data-fetching



With a REST based interface I might need to make several calls sequentially based on responses from the previous call

GraphQL




```
query {
  getPostNameComments {
    postID
    postName
    postAuthor
    commentID
    commentText
  }
}
```

With GraphQL I can make one call

How does GraphQL work?

```
type Query {
    getTodos: [Todo]
}

type Todo {
    id: ID!
    name: String
    description: String
    priority: Int
    duedate: String
}
```

Model data with application schema

```
query {
  getTodos {
    id
    name
    priority
  }
}
```

Client requests what it needs

```
{
    "id": "1",
    "name": "Get Milk",
    “priority": "1"
},
{
    "id": “2",
    "name": “Go to gym",
    “priority": “5"
},…
```

Only that data is returned

aws

Realtime and offline data using GraphQL

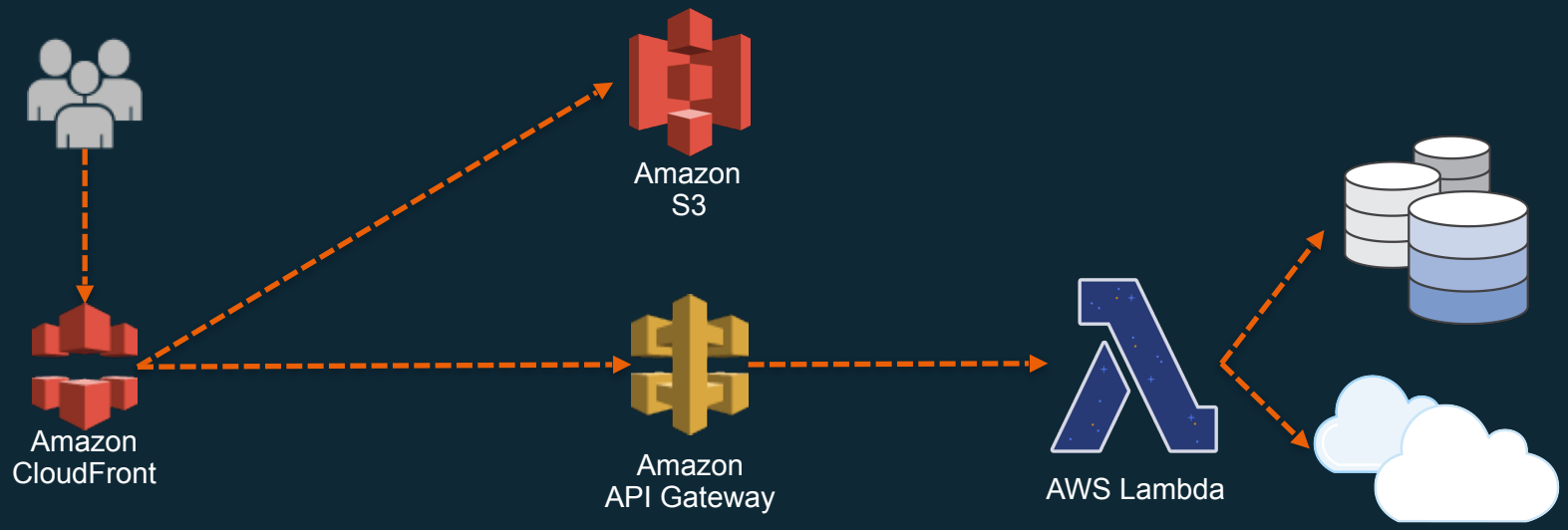
Real-time collaboration

Offline programming model with sync

Your data sources

Fine-grained access control

Serverless Web Application with API Gateway

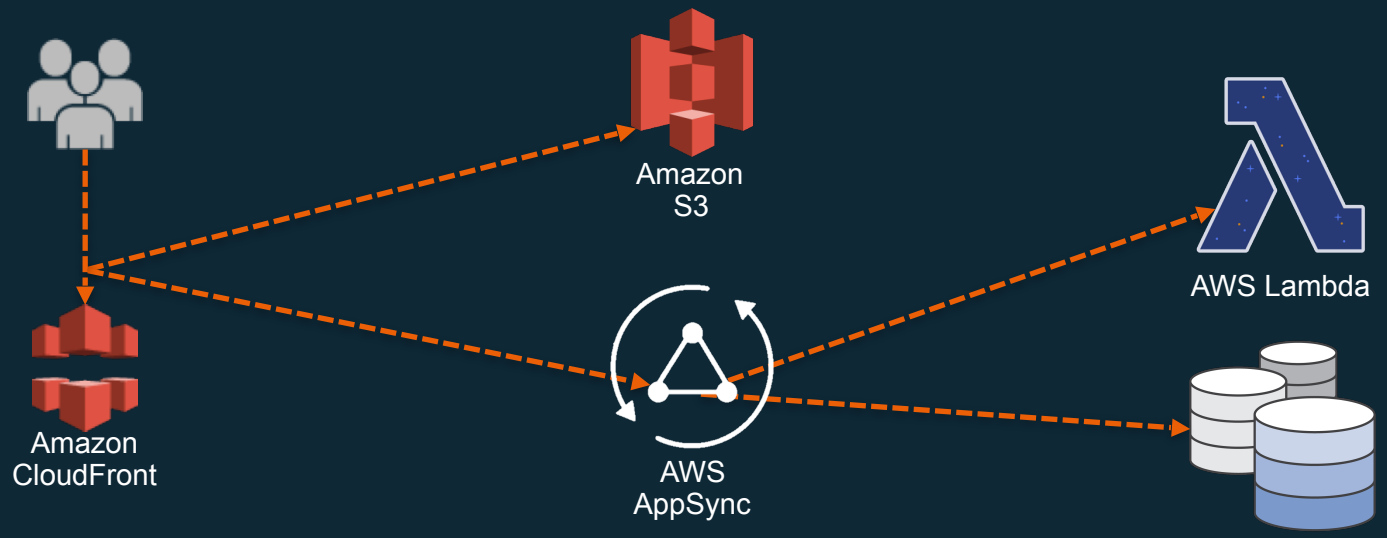


S3 stores all of your static content: CSS, JS, Images, etc. You would typically front this with a CDN such as Amazon CloudFront

API Gateway handles all of your application routing. It can handle authentication and authorization, throttling, DDOS protection and more

Lambda runs all of the logic behind your website and interfaces with databases, other backend services or anything else your site needs

Serverless Web Application with AppSync



S3 stores all of your static content: CSS, JS, Images, etc. You would typically front this with a CDN such as Amazon CloudFront

AppSync handles all of your GraphQL query resolution. It can retrieve data from data sources such as DynamoDB, Amazon ElasticSearch, Lambda, and HTTP endpoints

Data sources and/or Lambda provide customer data or backend logic

How does serverless help with development?

https://www.flickr.com/photos/lachlanhardy/4150836513/

Lines of Code (LoC)

Monolith = 10,000s to 1,000,000s

Microservices = 100s to 1000s

Lambda “nano-services” = 10s to 100s

aws

Lines of Code (LoC)

Monolith = 10,000s to 1,000,000s

Microservices = 100s to 1000s

Lambda “nano-services” = **10s to 100s**

+ Simplified testing of code
+ Easier to onboard new developers
+ Super fast deployment times
+ Orchestration in managed services not code
+ Reduced “blast radius” of issues
+ Greatly reduced technical debt

aws

What happens if we reduce operational burden massively and reduce the amount of code we have to write?

What if 2 pizzas is 1 too many?

The Future

https://www.flickr.com/photos/spacex/40143096241/

FIN/ACK

The combination of modern application user interfaces on reactive web, progressive mobile, and via voice with modern application backends powered by APIs and serverless compute will define the next decade of development

aws

aws.amazon.com/serverless

Menu | aws | Contact Sales | Products | Solutions | Pricing | More | English | My Account | Sign In to the Console

Serverless Computing | Overview | AWS Serverless Application Repository | Developer Tools | Resources | Partners

Serverless Computing and Applications

Build and run applications without thinking about servers

Find serverless applications

Serverless computing allows you to build and run applications and services without thinking about servers. Serverless applications don't require you to provision, scale, and manage any servers. You can build them for nearly any type of application or backend service, and everything required to run and scale your application with high availability is handled for you.

Building serverless applications means that your developers can focus on their core product instead of worrying about managing and operating servers or runtimes, either in the cloud or on-premises. This reduced overhead lets developers reclaim time and energy that can be spent on developing great products which scale and that

Chris Munns
munns@amazon.com
@chrismunns

https://www.flickr.com/photos/theredproject/3302110152/

https://secure.flickr.com/photos/dullhunk/202872717/