AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

SVS204

Getting started with serverless

Heeki Park
Principal Solutions Architect
Amazon Web Services

Agenda

Step 0: Overview

Step 1: Build a serverless backend: Lambda, SAM

Step 2: Configure API authorization: API Gateway

Step 3: Build and deploy a web application: Amplify

Step 4: Test the application

Step 5: Configure image metadata extraction: Rekognition

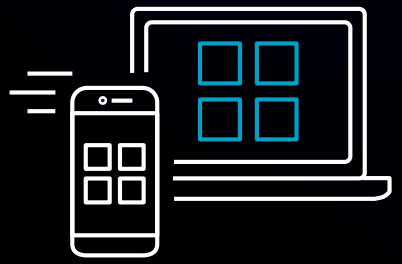
Step 6: Terminate the resources

Workshop link

https://bit.ly/3ARsuRZ

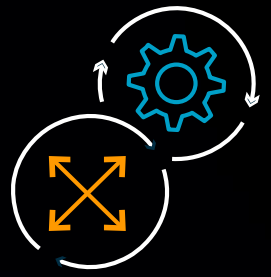
Step 0: Overview

What do our customers need to drive success?

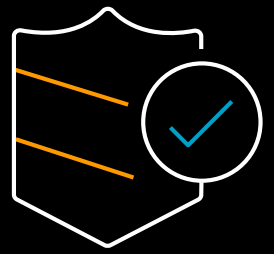


Get to market faster

Lower total cost of ownership



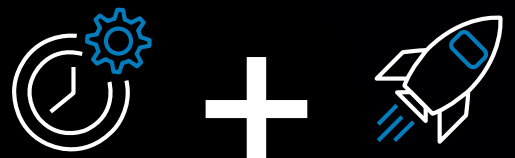
High performance and scalability



Security and isolation by design

How are customers adopting AWS?

REDUCE
the amount of DIY

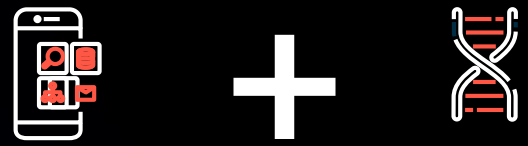


Retire + SaaS

MIGRATE
to AWS

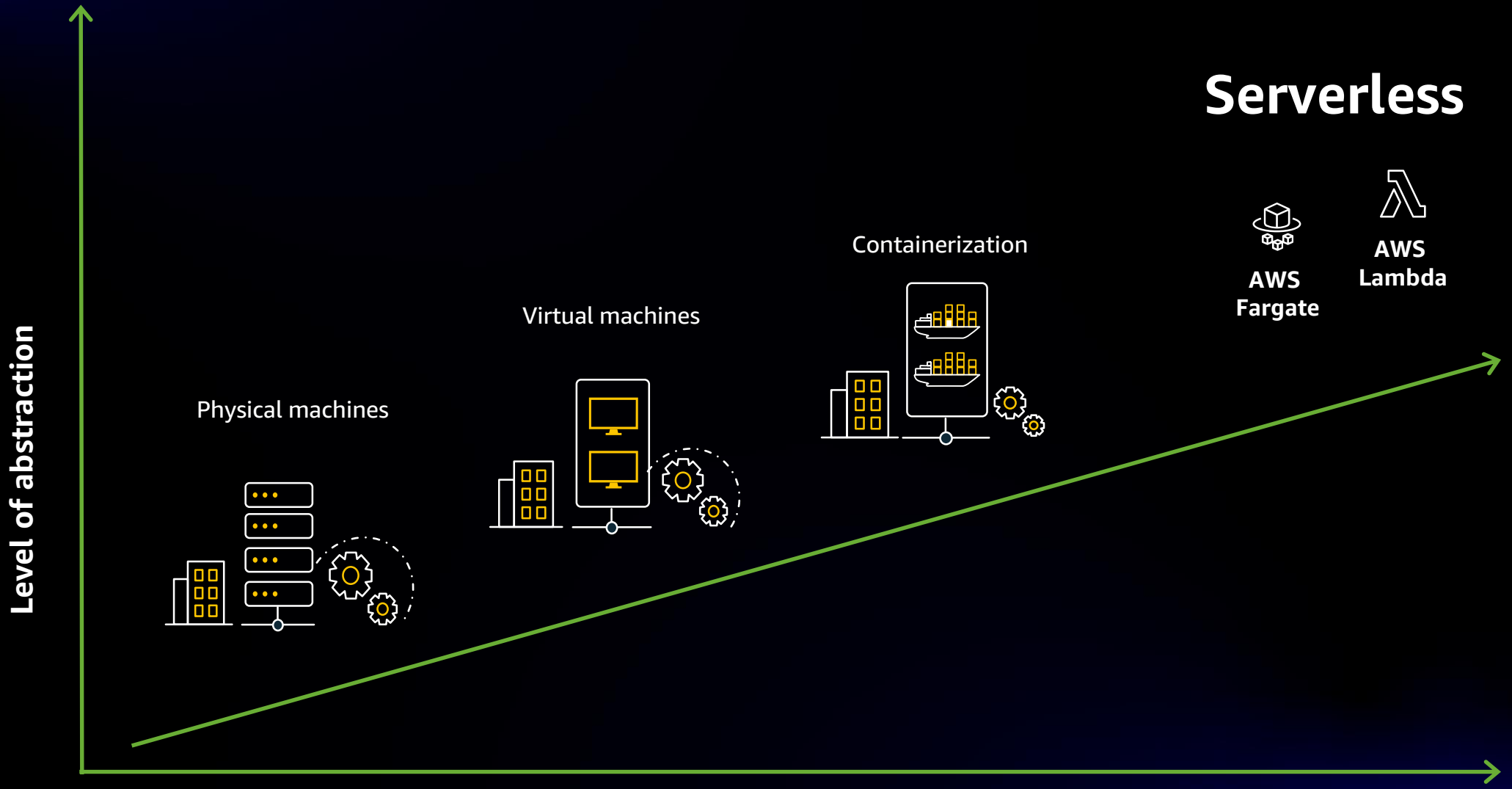
Lift and shift

MODERNIZE
on AWS



Replatform + Refactor

There’s a paradigm shift happening

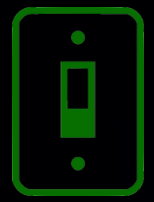


What is serverless?

No infrastructure provisioning, no management

Automatic scaling

Pay for value



Highly available and secure

aws

Serverless spans many different categories

Compute


AWS Lambda


AWS Fargate

Data stores


Amazon Simple Storage Service (Amazon S3)


Amazon Aurora Serverless

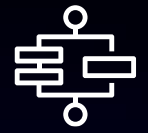

Amazon DynamoDB

Integration


Amazon API Gateway


Amazon Simple Queue Service (Amazon SQS)


Amazon Simple Notification Service (Amazon SNS)


AWS Step Functions


AWS AppSync

Serverless in the operational landscape

Compute	Virtual machine	Amazon EC2	AWS Elastic Beanstalk		Fargate	Lambda
Databases	MySQL	MySQL on Amazon EC2	Amazon RDS for MySQL	Amazon RDS	Amazon Aurora	DynamoDB
Storage	Storage					S3
Messaging	ESBs		Amazon MQ	Amazon Kinesis		EventBridge/SQS/SNS
Analytics	Hadoop	Hadoop on EC2	EMR	Amazon Elasticsearch Service		Amazon Athena

aws

What does the future look like?

ALL THE CODE YOU EVER WRITE IS BUSINESS LOGIC

Step 1:
Build a serverless backend

Serverless architecture

Event source

Changes in data state

Requests to endpoints

Changes in resource state

Function

Node.js
Python
Java
C#
Go
Ruby
Bring Your Own

Services / other

Anatomy of a function

Handler function

- Function executed on invocation
- Processes incoming event

Event

- Invocation data sent to function
- Shape differs by event source

Context

- Additional information from Lambda service
- Examples: request ID, time remaining

app.py

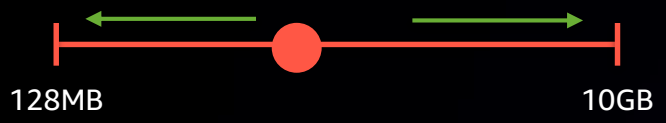
```
def handler(event, context):
  msg = ‘Hello {}’.format(
    event[‘name’]
  )
  return { ‘message’: msg }
```

aws

Function configuration

Power rating

- Select between 128 MB and 10 GB
- CPU allocated proportionally
- Power tune to balance cost and speed



Permissions model

- **Execution role** grants function access to resources via IAM
- **Function policy** controls invocation

Function configuration

Timeout

- Up to 15 minutes

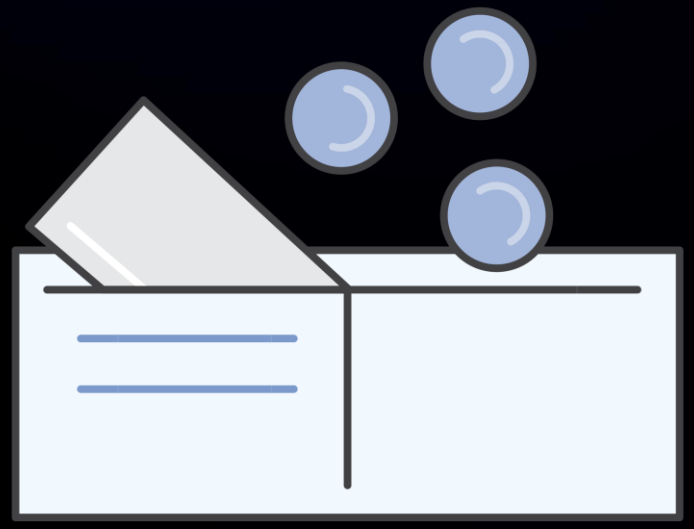
- Synchronous vs. asynchronous
- API Gateway timeout = 30 seconds

Network access

- Configure access to VPC
- Security group rules apply

- VPC does **not** enhance security of function

Fine-grained pricing



Free tier
1 million requests and 400,000 GB of compute
Every month, every customer

Pay for value
- Priced by power rating
- Charged in **1 ms** increments
- Low per-request charge

No minimum

Never pay for idle

AWS Serverless Application Model (SAM)

- CloudFormation extension optimized for serverless
- Shorthand syntax to express functions, APIs, databases, and event source mappings
- Simplifies IAM policy and event trigger management
- Model with YAML, deploy using AWS CloudFormation
- Open source!

https://aws.amazon.com/serverless/sam/
https://github.com/awslabs/serverless-application-model

AWS SAM template example

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  GetHtmlFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://sam-demo-bucket/todo_list.zip
      Handler: index.handler
      Runtime: nodejs12.x
      Policies: DynamoDBReadPolicy
      Events:
        GetToDo:
          Type: Api
          Properties:
            Path: /todo/{id}
            Method: GET
  Listable:
    Type: AWS::Serverless::Simple Table
```

SAM template transform

Creates Lambda function
- Runtime
- Execution policy
- Code
- Handler

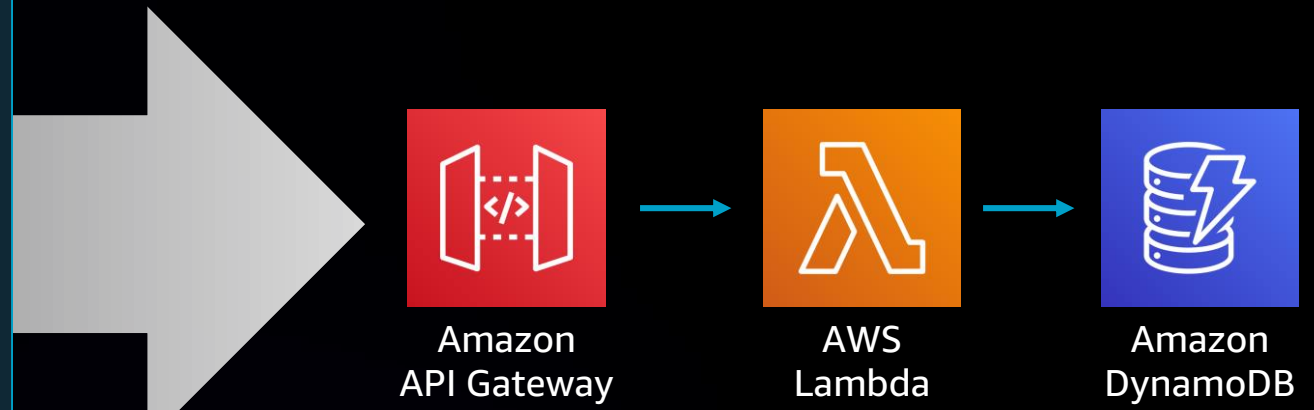
Creates API Gateway
- API endpoint
- Permissions

Create DynamoDB table with sane defaults

AWS SAM template example

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  GetHtmlFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://sam-demo-bucket/todo_list.zip
      Handler: index.handler
      Runtime: nodejs12.x
      Policies: DynamoDBReadPolicy
      Events:
        GetToDo:
          Type: Api
          Properties:
            Path: /todo/{id}
            Method: GET

  ListTable:
    Type: AWS::Serverless::SimpleTable
```



AWS SAM CLI

- CLI tool for local building, validating, testing of serverless apps
- Works with Lambda functions and “proxy-style” APIs
- Response object and function logs available on your local machine
- Mimic Lambda’s execution environment with Dockers images
- Emulates timeout, memory limits, runtimes

https://github.com/aws/aws-sam-cli

Getting started with SAM CLI

sam init

Generates a preconfigured AWS SAM template and example application code in the language you choose

sam package

Bundles your application code and dependencies into a "deployment package"

sam build

Prepares it for subsequent steps like deploy or local testing

sam deploy

Deploys your serverless application to the AWS Cloud

sam local

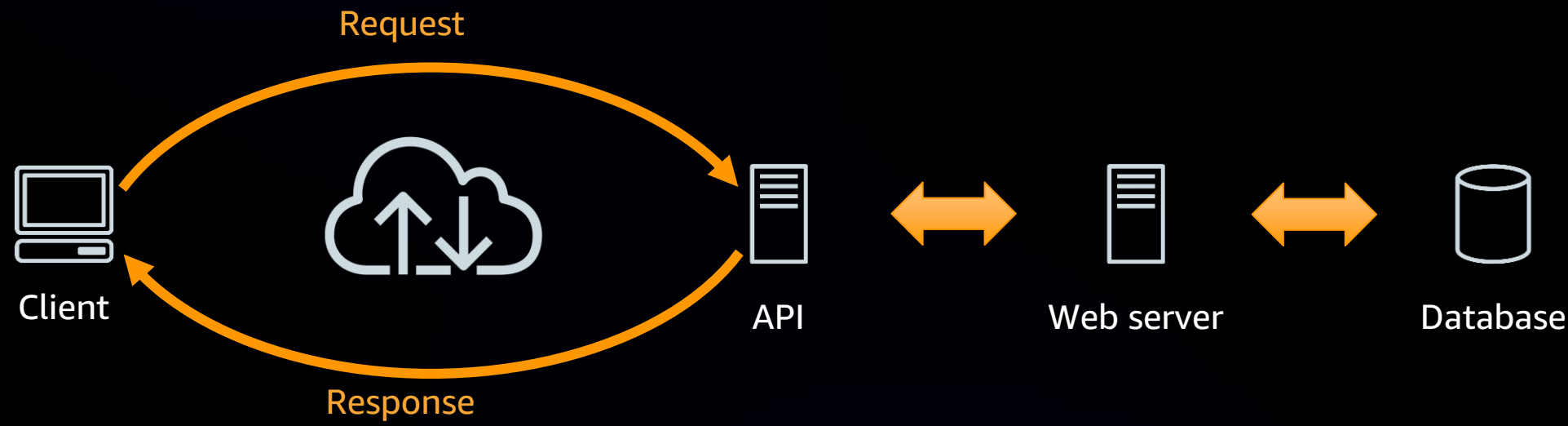
Test your application code locally

aws

Step 2:
Configure API authorization

What's an API?

> In building applications, an API simplifies programming by abstracting the underlying implementation and only exposing objects or actions the developer needs.

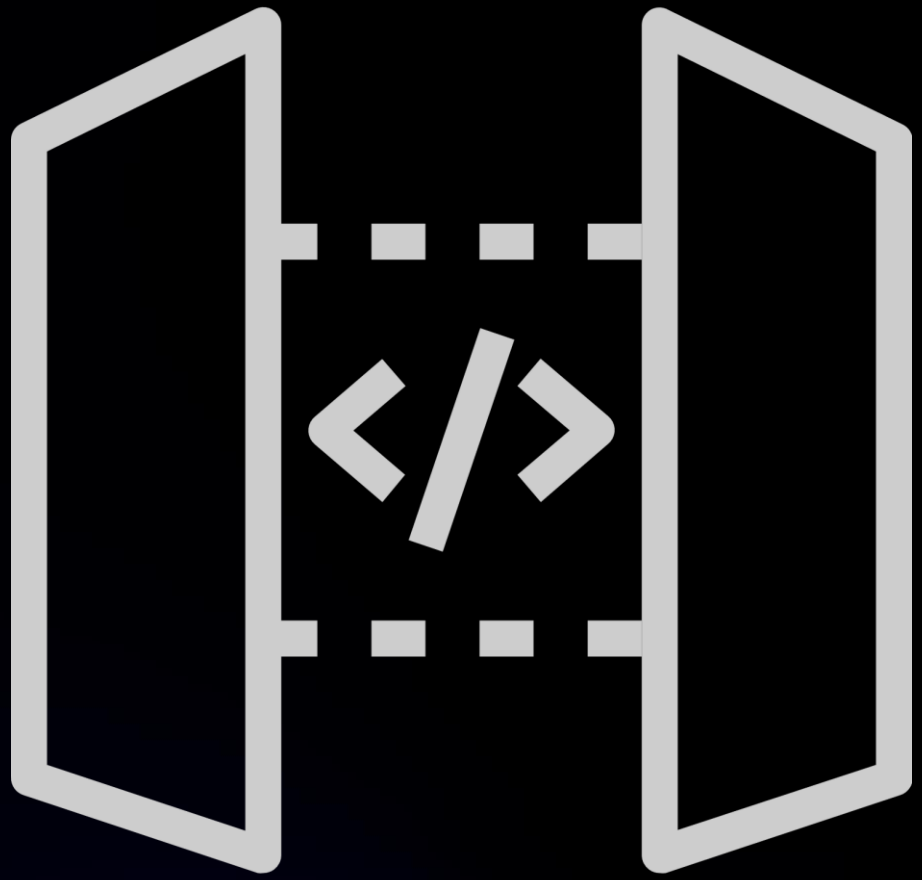


Web-based companies and services offer APIs for developers to use, such as:

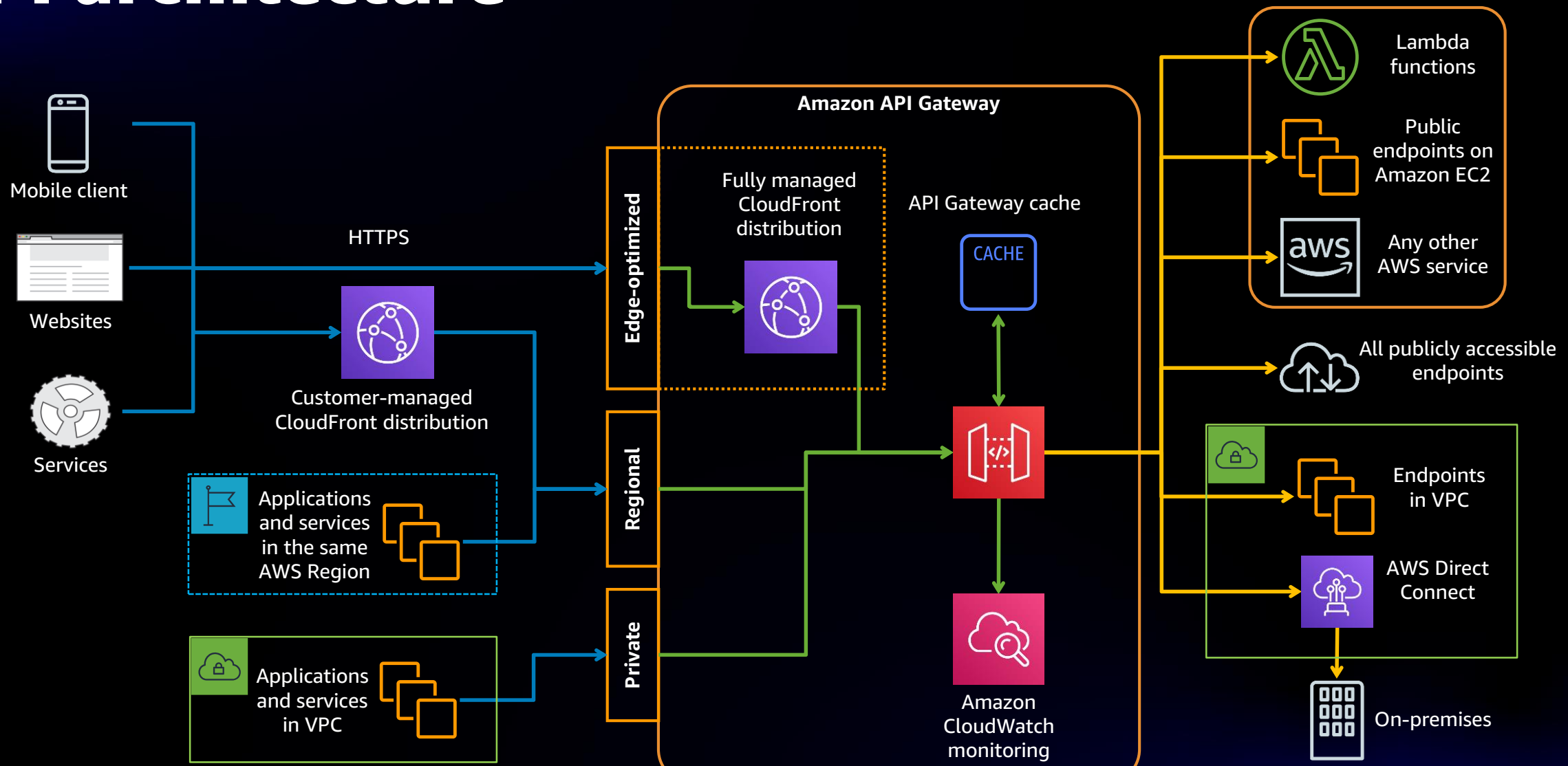
- Social networks – Facebook, Twitter, and others
- Payment processing – Amazon Pay, PayPal, and others

Amazon API Gateway

Amazon API Gateway is a fully managed (serverless) service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale.



API architecture



Endpoint types

Edge-optimized

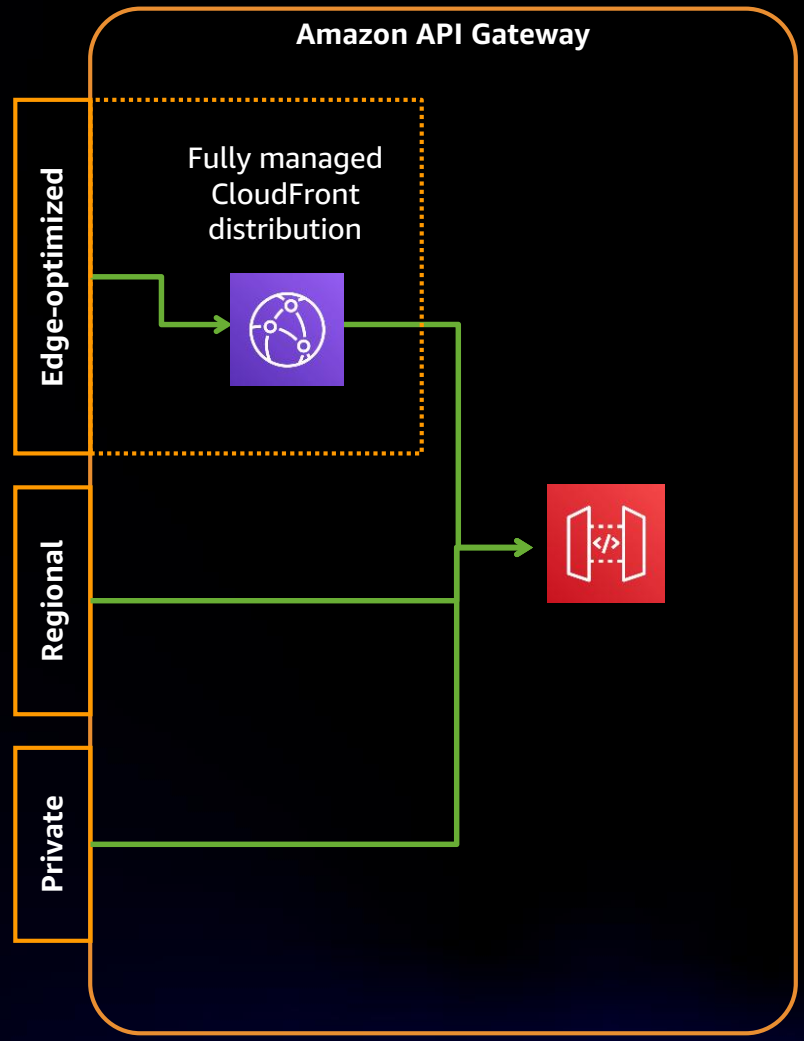
- Uses CloudFront to reduce TLS connection overhead (reduces roundtrip time)
- Designed for a globally distributed set of clients

Regional

- Recommended API type for general use cases
- Designed for building APIs for clients in the same region

Private

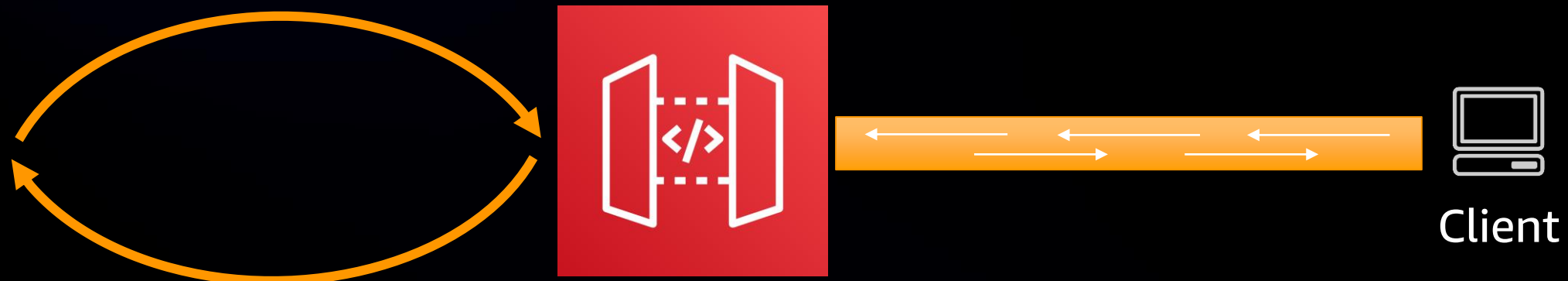
- Only accessible from within VPC (and networks connected to VPC)
- Designed for building APIs used internally or by private microservices



Supported protocols

RESTful APIs

WebSocket APIs



- Request / response
- HTTP methods like GET, POST, and so on
- Short-lived communication
- Stateless

- Serverless WebSocket
- 2-way communication channel
- Long-lived communication
- Stateful

Step 3: Build and deploy a web application

AWS Amplify

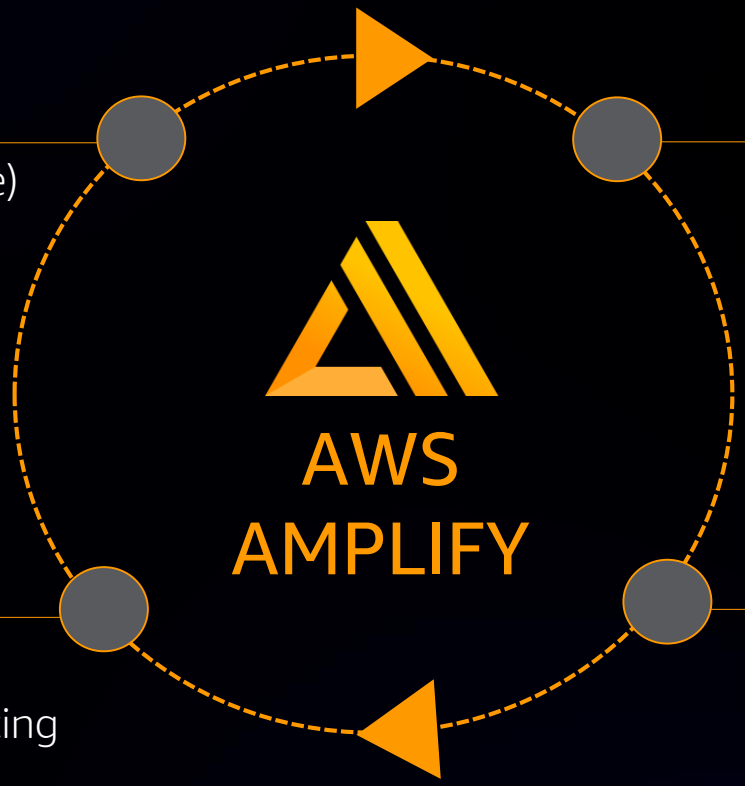
Designed for the development lifecycle

ENGAGE / MEASURE

- Multichannel (push/SMS/email/voice)
- Behavior-based and personalized audience segments
- Engagement performance measurements
- Real-time customer data for immediate optimization

DEVELOP

- Open source libraries
- iOS/Android native
- JavaScript/React/Vue/Angular
- UI components
- Escape hatches



DEPLOY / HOST

- AWS infrastructure
- Ease of deployment with CLI or hosting
- CI/CD capability
- Fully managed global hosting

TEST

- Device farm
- Test on real devices
- Test on real browsers
- Integrate testing with CI/CD

AWS Amplify CLI

- Create, update, and delete cloud services
- Manage multiple environments
- GraphQL Transform
- GraphQL Codegen

AWS Amplify Client

```
// import Amplify components
import { API } from 'aws-amplify'

// call Amazon API Gateway endpoint
const data = await API.get('orderApi', '/orders')
```

```
// import React component
import { withAuthenticator } from 'aws-amplify-react'

// main App component definition
class App extends React.Component {
    // your beautiful code
}

// add authentication
export default withAuthenticator(App)
```

Android

iOS

React Native

Ionic

JavaScript

React

Angular

Vue

AWS Amplify Libraries

Authentication

Authentication APIs with prebuilt UI components for your app

DataStore

On-device persistent storage that automatically synchronizes data between your apps and the cloud

API

HTTP requests using REST and GraphQL with support for real-time data

Analytics

Track user sessions, custom user attributes, and in-app metrics

PubSub

Connect your app to message-oriented middleware on the cloud

Predictions

Add MI/ML capabilities to your app powered by cloud services

Interactions

Conversational bots powered by deep learning technologies

Push Notifications

Push notifications with campaign analytics and targeting

Storage

Manage user content securely in public, protected, and private storage

XR

Work with augmented reality and virtual reality content in your apps

aws

AWS Amplify Console

Build, deploy, and host cloud-powered modern web apps

Connect your repository

Configure build settings

Deploy your app

Step 4:
Test the application

Step 5: Configure image metadata extraction

Amazon Rekognition

Deep learning-based image and video analysis

Amazon Rekognition is a service that can identify objects, people, text, scenes, and activities in images and videos

Facial analysis

Demographic data

Age range:	29–45
Gender: Male	96.5%

Facial landmarks

EyeLeft, EyeRight, Nose, RightPupil, LeftPupil, MouthRight, LeftEyeBrowUp Bounding Box...

Image quality

Brightness	23.6%
Sharpness	99.9%

Emotion expressed

Happy	83.8%
Surprised	0.65%

General attributes

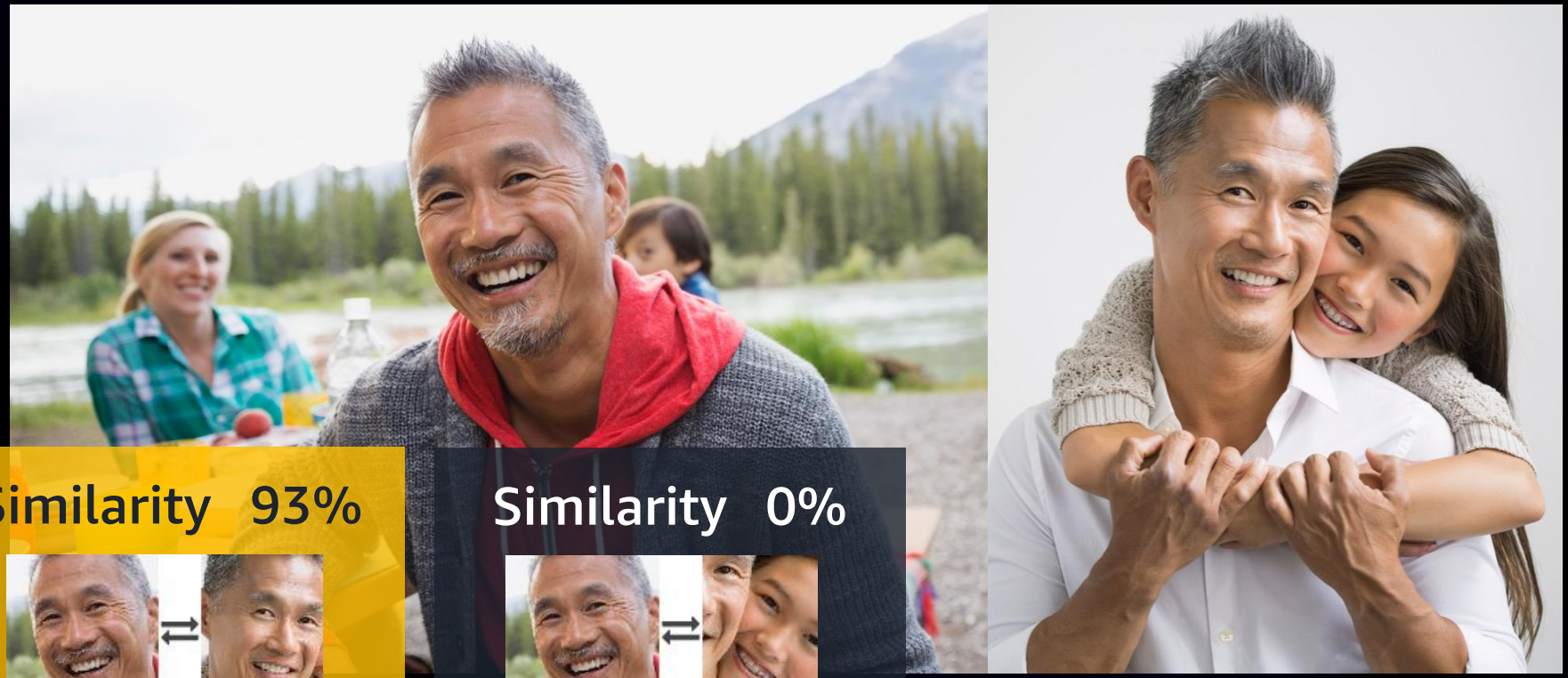
Smile: True	23.6%
Eyes Open: True	99.8%

Facial pose

Pitch	1.446
Roll	5.725
Yaw	4.383

Face comparison

Measure the likelihood that faces are of the same person



Text detection

Extract text content from real-world images and videos in various layouts, fonts, and styles

Step 6: Terminate the resources

Thank you!

Workshop Speaker

Heeki Park
heeki@amazon.com

Workshop Author

Mark Richman
mrkrchm@amazon.com

Workshop Support

Avichal Chum
Bhushan Tomar
Daniel Ness
Matt Diamond
Zack Anderson
Vijay Injam
Jake Walker