
AWS
re:Invent

SVS314

API patterns and architectures: RESTful vs. GraphQL APIs

George Mao

Head of WW Solutions Architecture, Serverless Computing

AWS

AWS re:Invent

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

Agenda

Compare RESTful APIs and GraphQL APIs

Serverless APIs with Amazon API Gateway and AWS AppSync

How do you choose?

Getting started



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

REST vs. GraphQL APIs

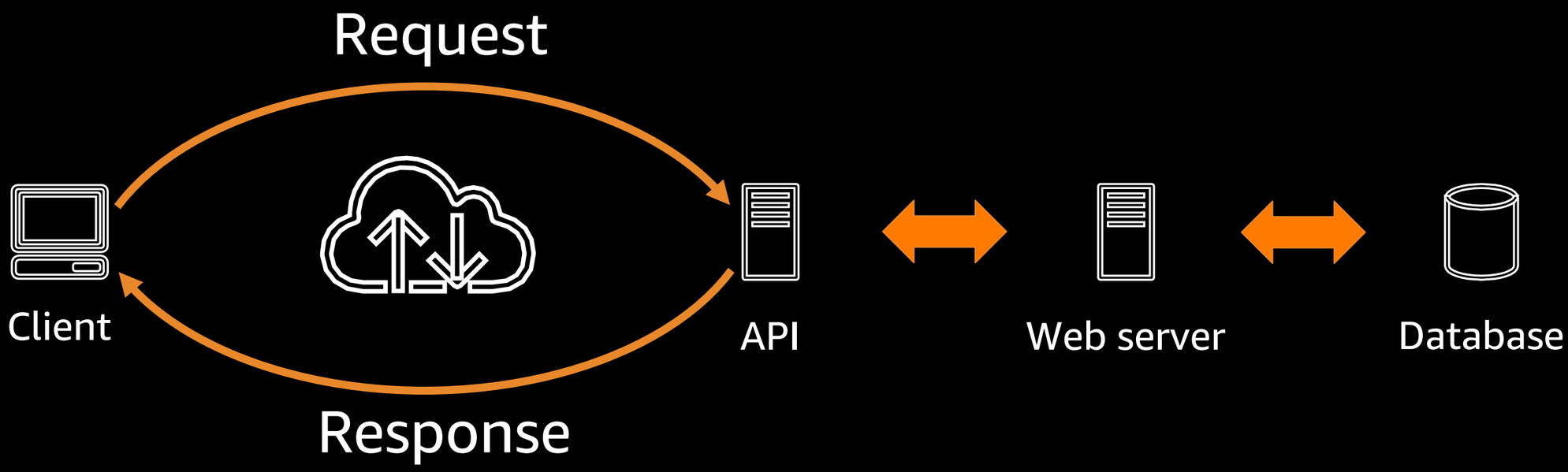


AWS re:Invent

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

Application programming interface (API)

> An API simplifies programming by abstracting the underlying implementation and only exposing objects or actions the developer needs



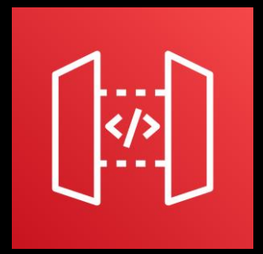
Quotation from Wikipedia: https://en.wikipedia.org/wiki/Application_programming_interface

REST and GraphQL API similarities

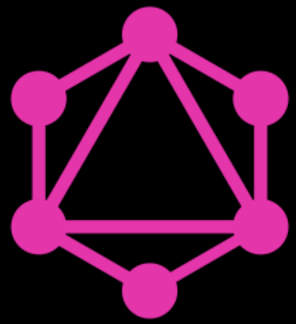
Operate over HTTP(S)
Stateless communication
Standardize client / server interactions



We all know REST, but what is GraphQL??



Representational state transfer
First introduced in 2000
Used nearly everywhere
Typically JSON based



Created by Facebook in 2015
Open sourced to GraphQL Foundation in 2018
Typically JSON based

REST and GraphQL API comparison

	REST API	GraphQL API
Endpoint URI "http://api.anycompany.com"	Base URI or domain	Base URI or domain
Resource path "/weather" or "/restaurants"	Client specifies path on server	Static path: /graphql
Parameters "?zip=11111" or "/menu"	Path parameters Query string parameters HTTP payloads	GraphQL query
HTTP method "GET, POST, UPDATE, DELETE"	Client specifies method accepted by server	POST

REST API concepts

REST is designed for servers

Clients send requests by combining:

[resource path] + [optional path/query param] + [HTTP method/verb]

/weather?zip=11111	/restaurants/11111	/login
GET	*GET*	*POST*

Servers return responses with a static schema and preprogrammed payload format

Clients must consume the responses and parse

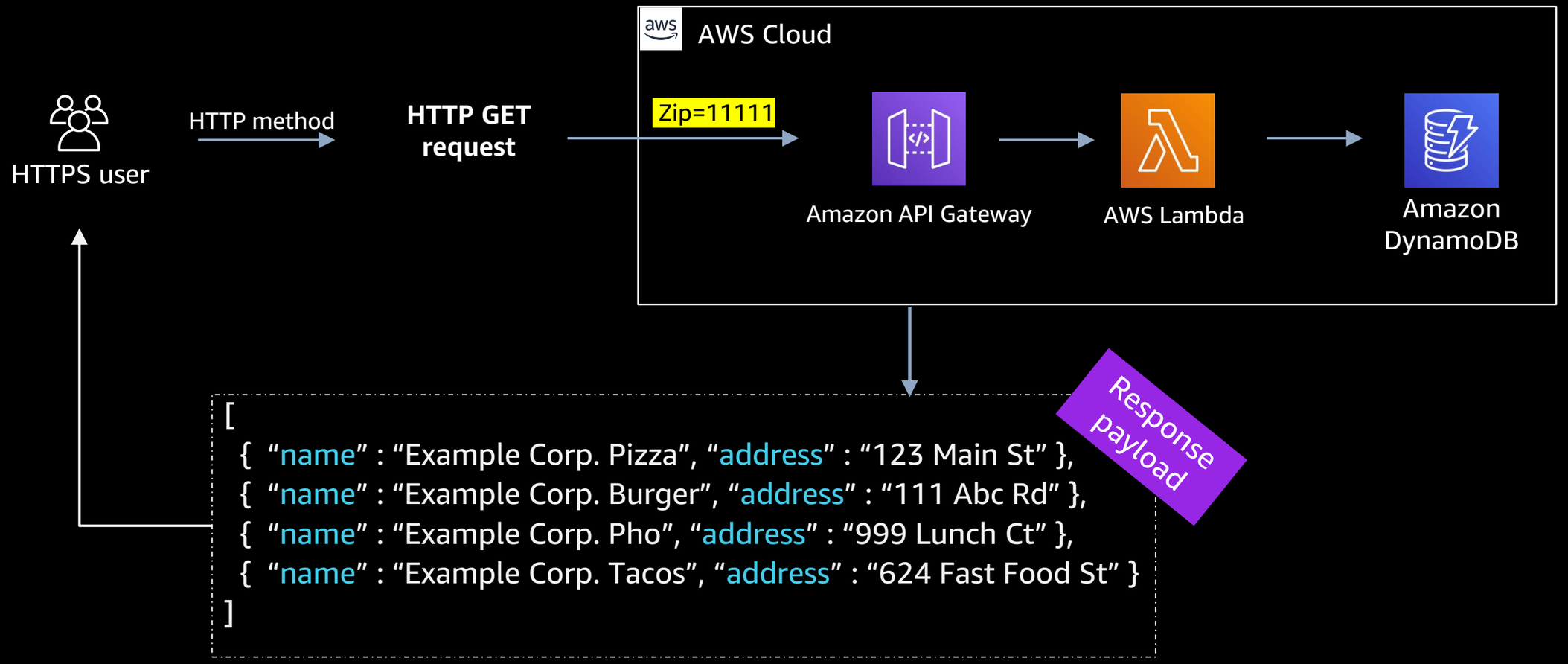
REST API concepts: HTTP methods

Client specifies the HTTP method that indicates the intended action

POST	Create records or add data
GET	Fetch or read data
PUT	Update a record
PATCH	Update or modify a record
DELETE	Delete a record or data

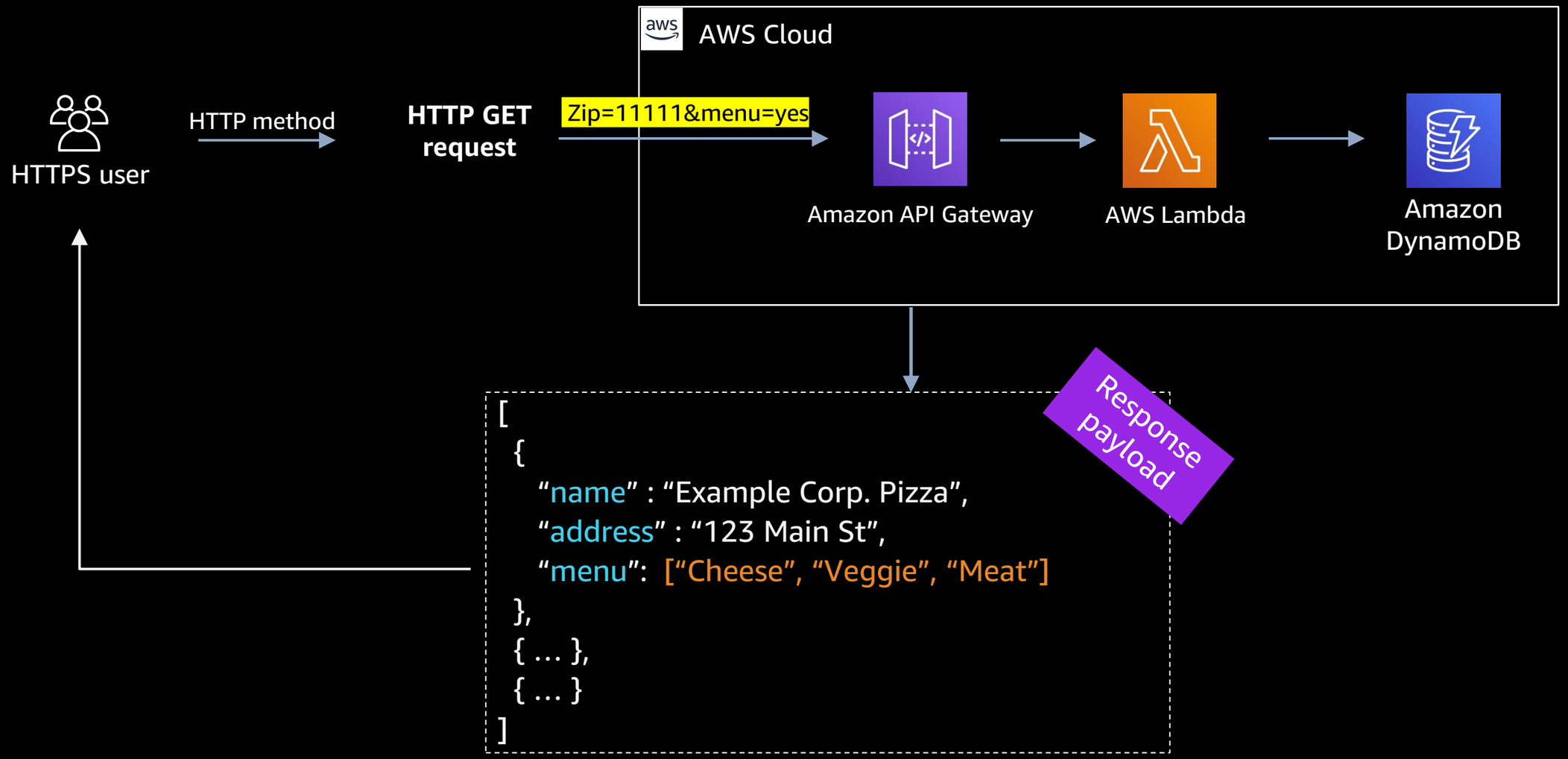
REST: List all restaurants in a ZIP code

GET https://api.anycompany.com/restaurants?zip=11111



REST: List all restaurants in a ZIP code with menu

GET https://api.anycompany.com/restaurants?zip=11111&menu=yes



REST: List all restaurants in a ZIP code with menu

Query strings

GET https://api.anycompany.com/restaurants?zip=11111&menu=yes

Path parameters

GET https://api.anycompany.com/restaurants/11111/menu

Additional resources

GET https://api.anycompany.com/restaurantsWithMenu/11111

GraphQL API concepts

GraphQL is strongly typed query language designed to allow clients to **ask** for data

Clients send requests by combining:

[/graphql] + [HTTP POST] + [Body: GraphQL query]

/graphql
POST

```
query getWeather(zip) {
  getWeather(zip) {
    currentTemp
    forecast
  }
}
```

```
query getAllRestaurants(zip) {
  getAllRestaurants(zip) {
    name
    address
  }
}
```

Servers return responses according to the requested fields specified in the query

GraphQL = Graph query language

Define a schema:

- types
- queries
- mutations



```
type Restaurant {
        id: ID!
        name: String!
        address: String!
        state: String!
        zip: String!
        type: String
        menu: [Menu]!
}
```

The GraphQL schema

Object types

```
type Restaurant {
        id: ID!
        name: String!
        address: String!
        state: String!
        zip: String!
        type: String!
        menu: [Menu]!
}


type Menu {
        name: String!
        price: String!
        glutenFree: String!
}
```

Fetch data

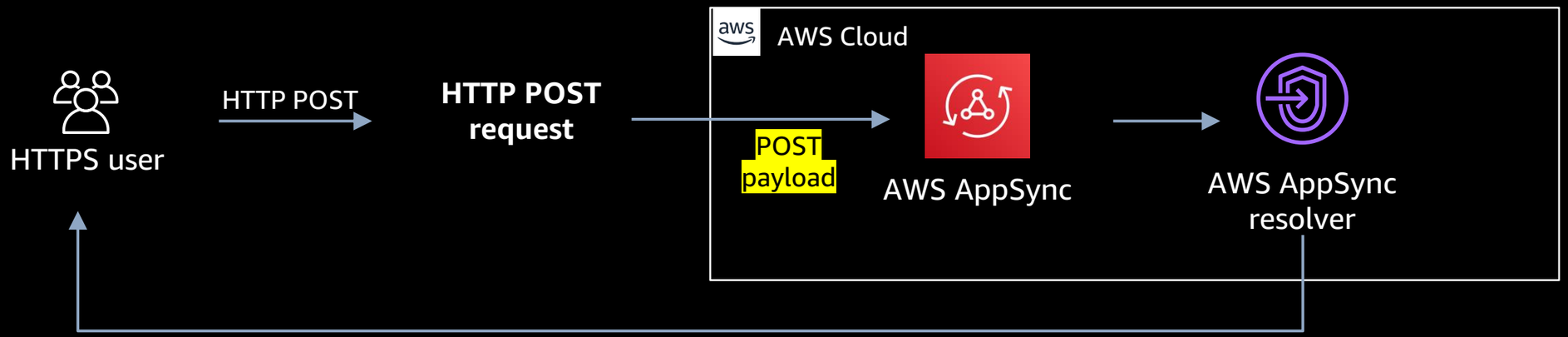
```
type Query {
        getAllRestaurants: [Restaurant]
        getRestaurant(id: ID!, zip: String): Restaurant
}
```

Modify data

```
type Mutation {
        addRestaurant: Restaurant
        deleteRestaurant: Restaurant
}
```

GraphQL: List all restaurants in a ZIP code

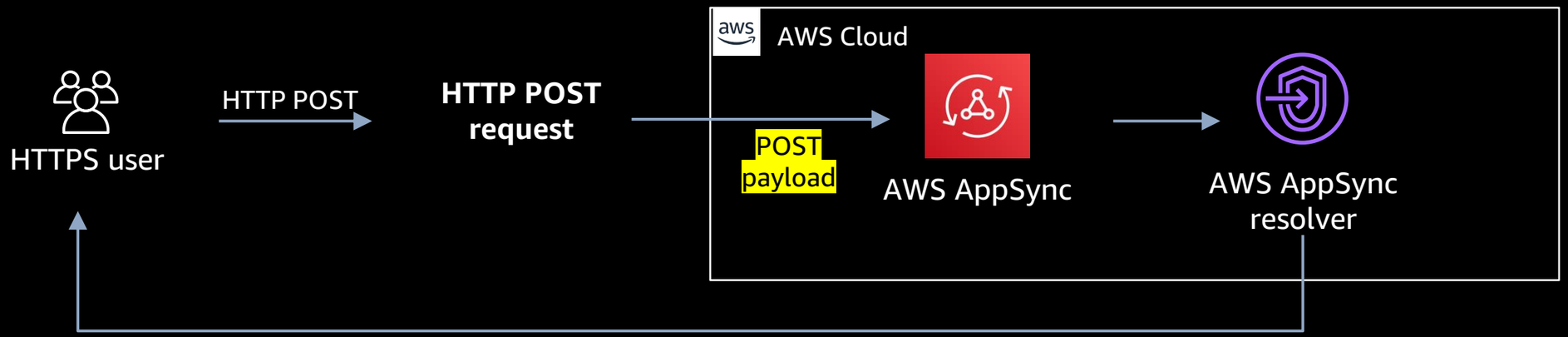
POST http://api.anycompany.com/graphql




```
query getAllRestaurants(zip) {
  getAllRestaurants(zip) {
    name
    address
  }
}
```

GraphQL: List all restaurants in ZIP with menu

POST http://api.anycompany.com/graphql




```
query getAllRestaurants(zip) {
  getAllRestaurants(zip) {
    name
    address
  }
}
```

```
query getAllRestaurants(zip) {
  getAllRestaurants(zip) {
    name
    address
    menu {
      name
      price
      glutenFree
    }
  }
}
```

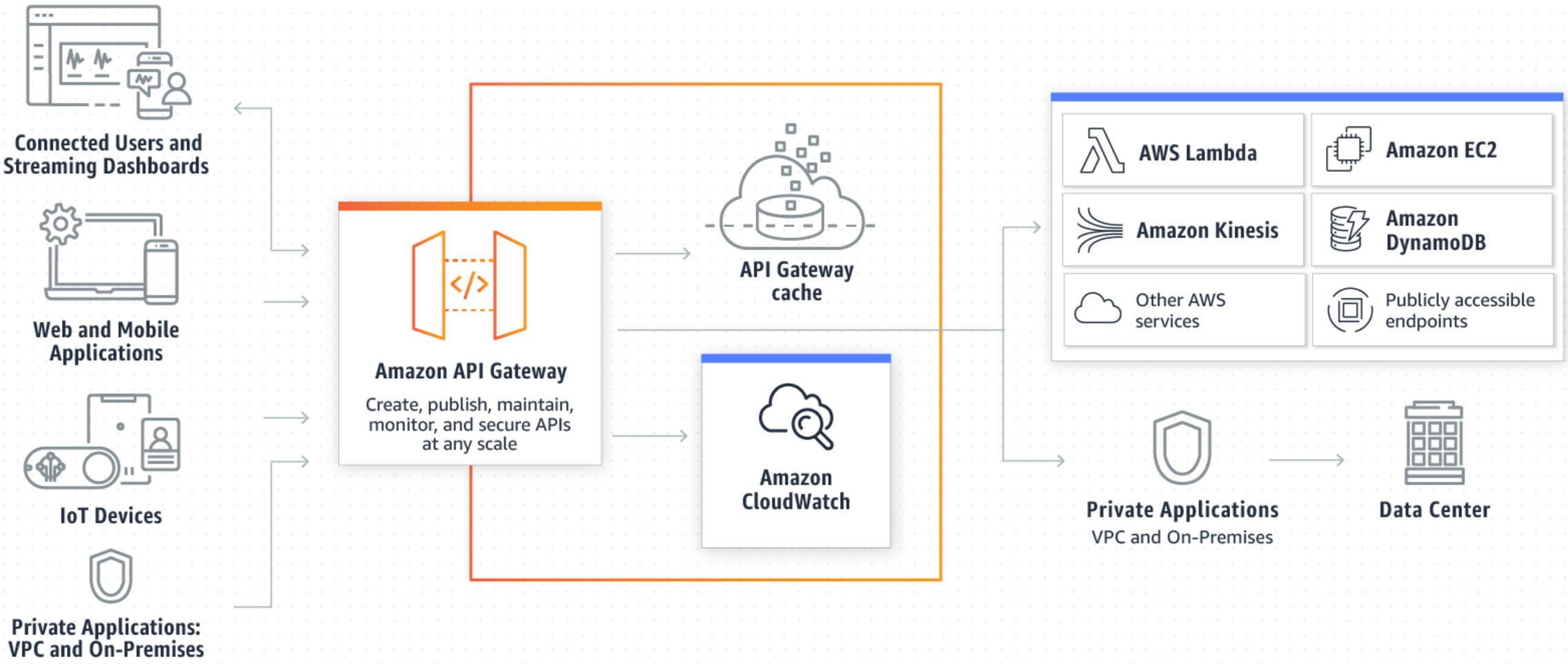
Amazon API Gateway: RESTful APIs



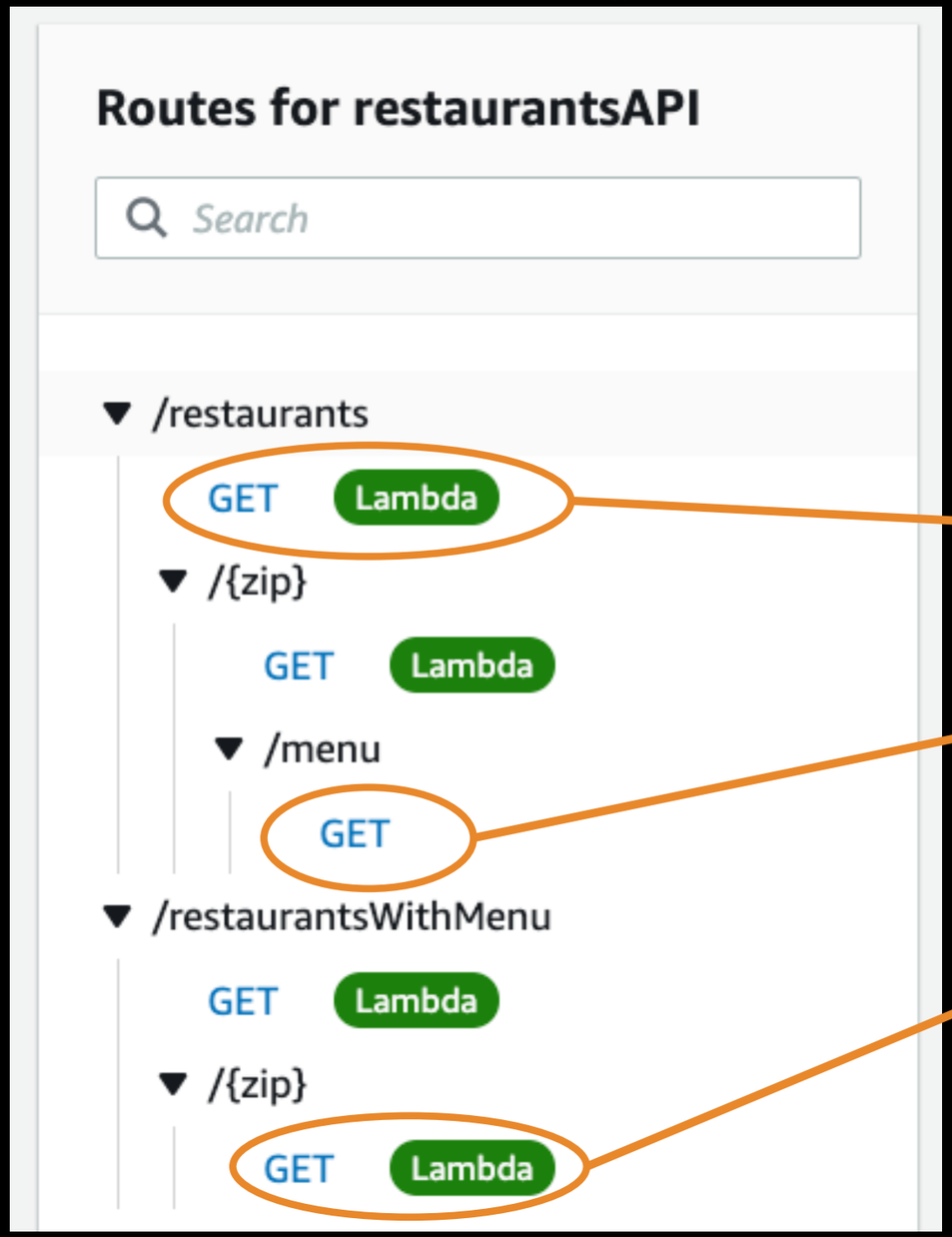
AWS re:Invent

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

Amazon API Gateway



API Gateway deep dive: REST resources



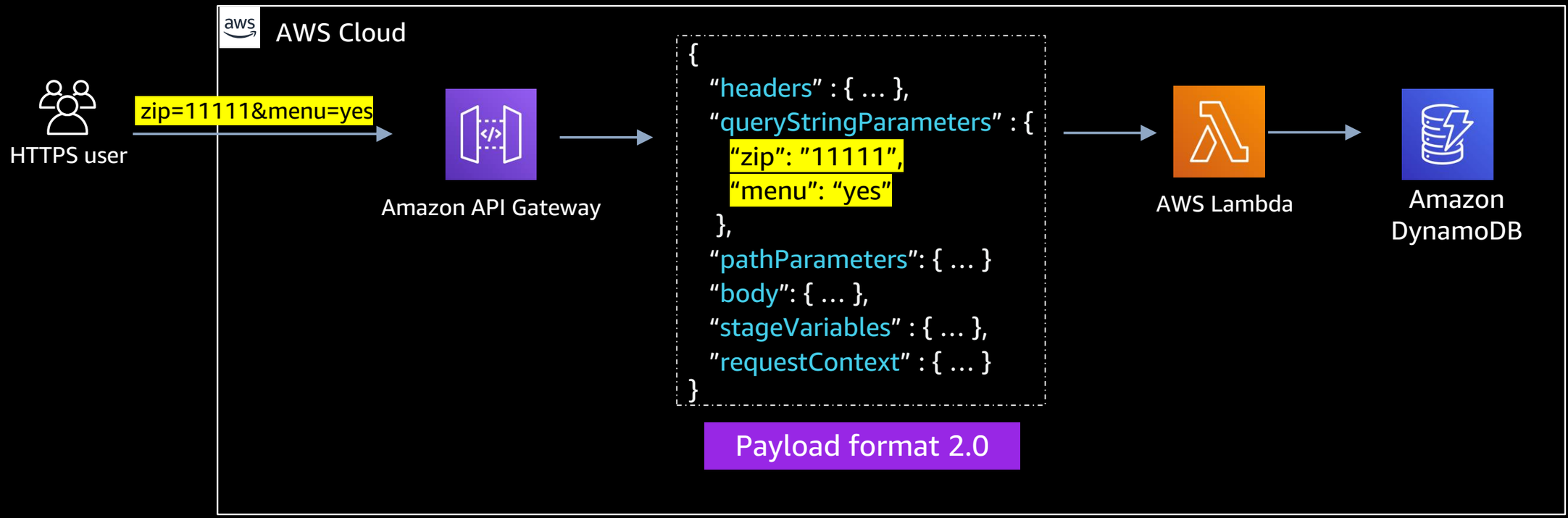
GET https://api.anycompany.com/restaurants?zip=11111&menu=yes

GET https://api.anycompany.com/restaurants/11111/menu

GET https://api.anycompany.com/restaurantsWithMenu/11111

REST: List all restaurants in a ZIP code

GET https://api.anycompany.com/restaurants?zip=11111&menu=yes



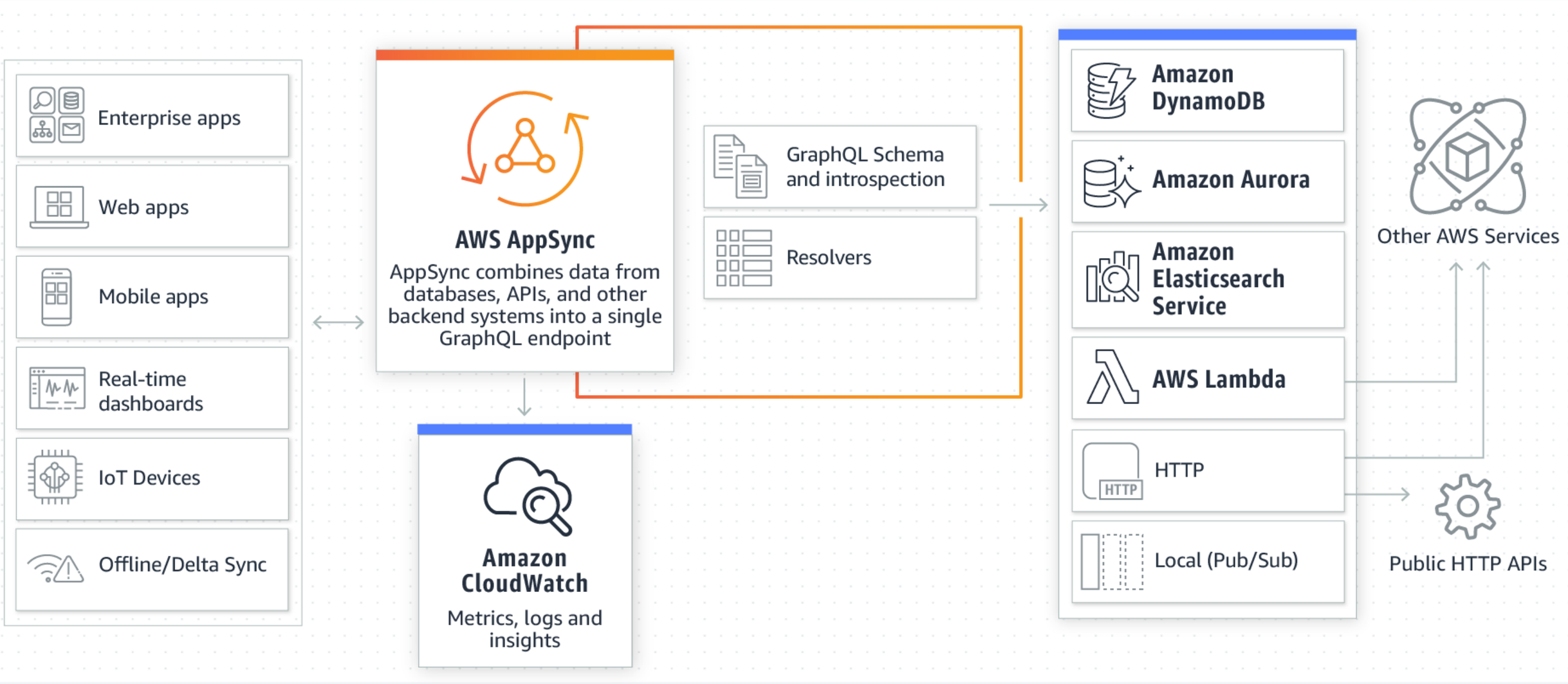
AWS AppSync: GraphQL APIs



AWS re:Invent

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

AWS AppSync


Enterprise apps
Web apps
Mobile apps
Real-time dashboards
IoT Devices
Offline/Delta Sync
AWS AppSync
AppSync combines data from databases, APIs, and other backend systems into a single GraphQL endpoint
GraphQL Schema and introspection
Resolvers
Amazon CloudWatch
Metrics, logs and insights
Amazon DynamoDB
Amazon Aurora
Amazon Elasticsearch Service
AWS Lambda
HTTP
Local (Pub/Sub)
Other AWS Services
Public HTTP APIs

AWS AppSync resolvers

Resolvers are responsible for retrieving fields from data sources and mapping to schema

```
type Restaurant {
        id: ID!
        name: String!
        address: String!
        state: String!
        zip: String!
        type: String!
        menu: Menu!
}
```

DynamoDB data source

Restaurants table

id	name	address	state	zip	type
1	Example Corp. Pizza	123 Main St	WA	11111	Fast food
2	Example Corp. Burger	111 Abc Rd	WA	11111	Fast food
3	Example Corp. Pho	999 Lunch St	VA	11111	Asian
4	Example Corp. Tacos	624 Fast Food St	MD	11111	TexMex

Lambda data source



AWS Lambda

Menu API



API Gateway

```
{
  [
    "name": "Cheese Pizza",
    "name": "Wings",
    "name": "Fries"
  ]
}
```

AWS AppSync query resolution

```
query getAllRestaurants{
        getAllRestaurants(){
                name
                address
                state
                zip
                type
                menu {
                        name
                        price
                        glutenFree
                }
        }
}
```

DynamoDB resolver

Scan

```
{
  "TableName" : "Restaurants"
}
```

DynamoDB

Restaurants Table

Lambda resolver

AWS Lambda

AWS AppSync query resolution

```
query getAllRestaurants{
        getAllRestaurants(){
                name
                address
                state
                zip
                type
        }
}
```

DynamoDB resolver

Scan

```
{
  "TableName" : "Restaurants"
}
```

DynamoDB



Restaurants Table

Modifying data: REST vs. GraphQL

```
POST /addRestaurant HTTP/1.1
Host: api.anycompany.com
Authorization: […]

{
  “name” : “A New Restaurant”,
  “address”: “000 New Street”,
  ”state”: “VA”,
  “zip” : “11111”,
  “type” : “Cafe”
}
```

```
POST /graphql HTTP/1.1
Host: api.anycompany.com
Authorization: […]

mutation addRestaurant{
  addRestaurant(
   name : “A New Restaurant”,
   address: “000 New Street”,
   state: “VA”,
   zips : “11111”,
   type : “Cafe”
  ){
    id
  }
}
```

How do you decide?



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

Development experience

	REST	GraphQL
Client code	Built-in support via JavaScript All browsers	GraphQL SDK
Tools	cURL, wget, POSTMan	POSTMan
General support	Widely accepted standard Mature ecosystem	Less common

Benefits

	REST	GraphQL
Data size	Defined by server and matches response schema	Can change based on data requested by client
Payload shape	Static response schema defined by server	Defined by what the client asks for
Client support	Natively works with modern browsers	Requires JS libraries

Considerations

How many data sources do you have?

Do you expect the data sources to change?

What are the clients for your API? Do you control the clients?

What is the skill set of your developers?

Resources

REST APIs with Amazon API Gateway

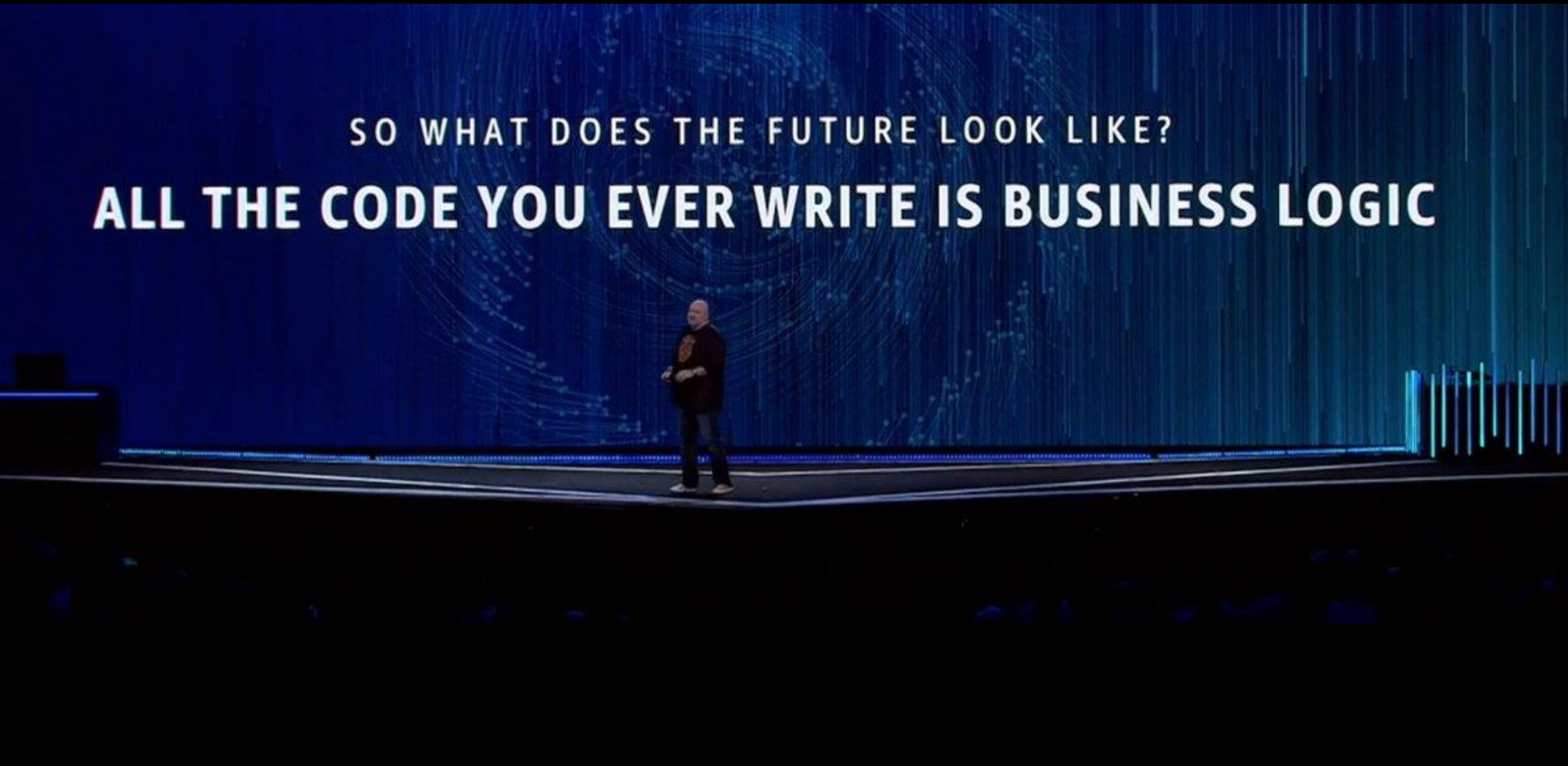
https://aws.amazon.com/api-gateway/resources/

https://aws.amazon.com/serverless-workshops/

GraphQL APIs with AWS AppSync

https://aws.amazon.com/appsync/resources/

https://aws.amazon.com/graphql/guide/


SO WHAT DOES THE FUTURE LOOK LIKE?
ALL THE CODE YOU EVER WRITE IS BUSINESS LOGIC

Check out these other sessions

SVS401 – Amazon API Gateway HTTP APIs: Beyond the proxy

FWM201 – Power modern serverless applications with GraphQL and AWS AppSync

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.