AWS re:Invent

MOB318-R

AWS AppSync does that: Support for alternative data sources

Josh Kahn

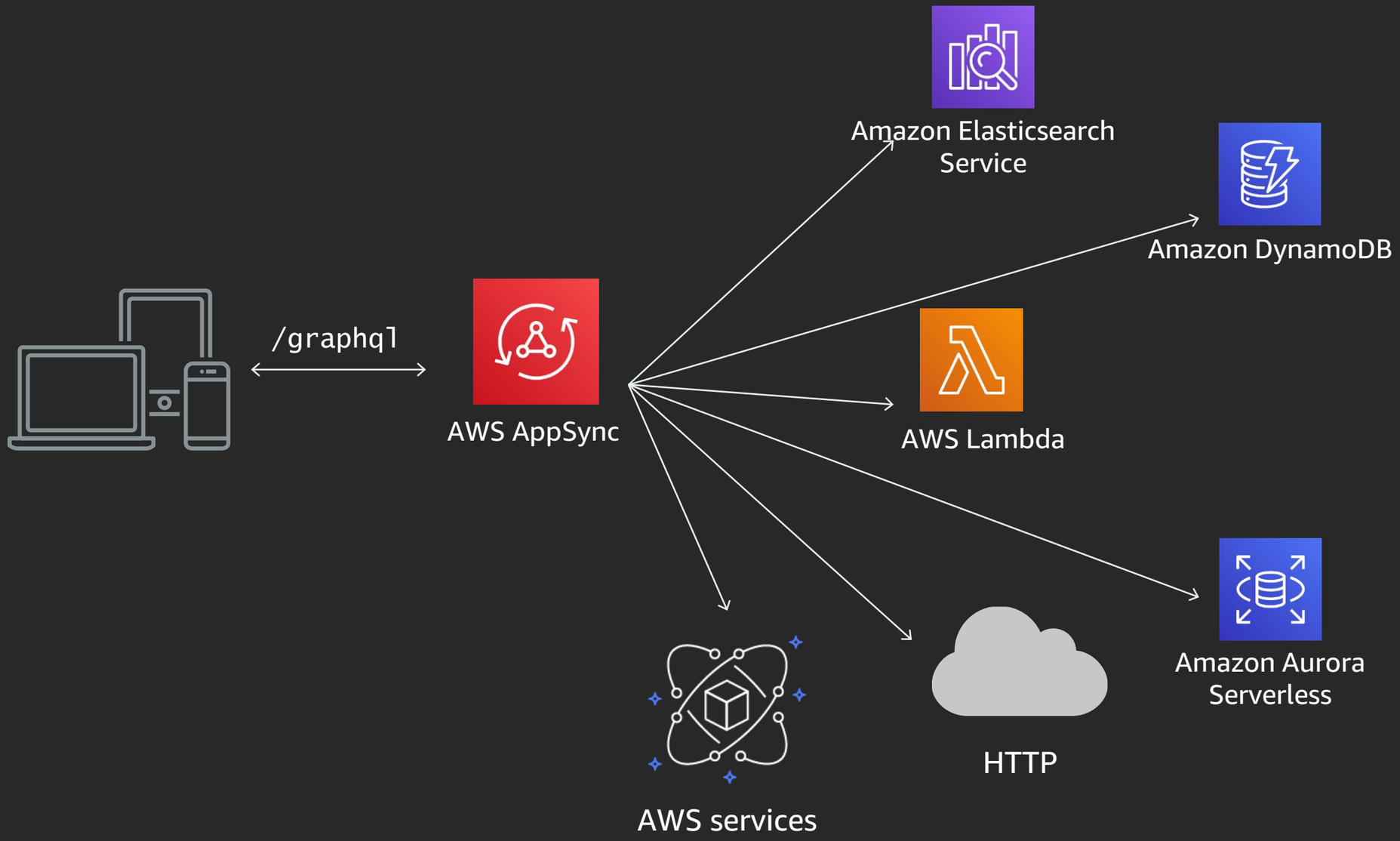
Senior Solutions Architect

Amazon Web Services

AWS AppSync

- Fully managed API layer
- Implemented in GraphQL
- Database agnostic
- Enterprise security built in
- Real-time support (via GraphQL subscriptions)
- Offline SDKs (AWS Mobile SDK for iOS, AWS Mobile SDK for Android, AWS SDK for JavaScript in the Browser)

Out-of-the-box data sources

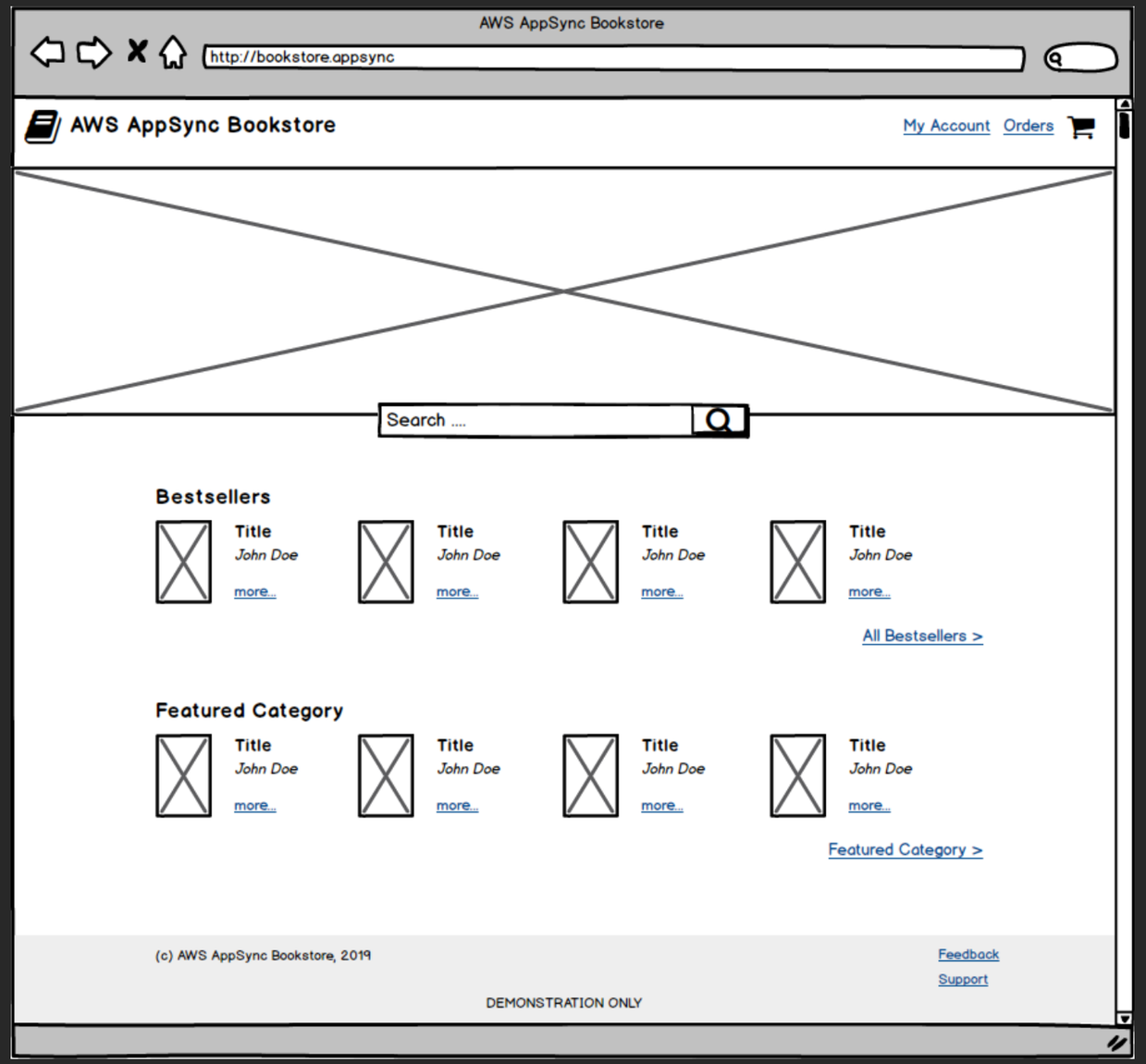


"Seldom can one database fit the needs of multiple distinct use cases. The days of the one-size-fits-all monolithic database are behind us, and developers are now building highly distributed applications using a multitude of purpose-built databases."

Dr. Werner Vogels

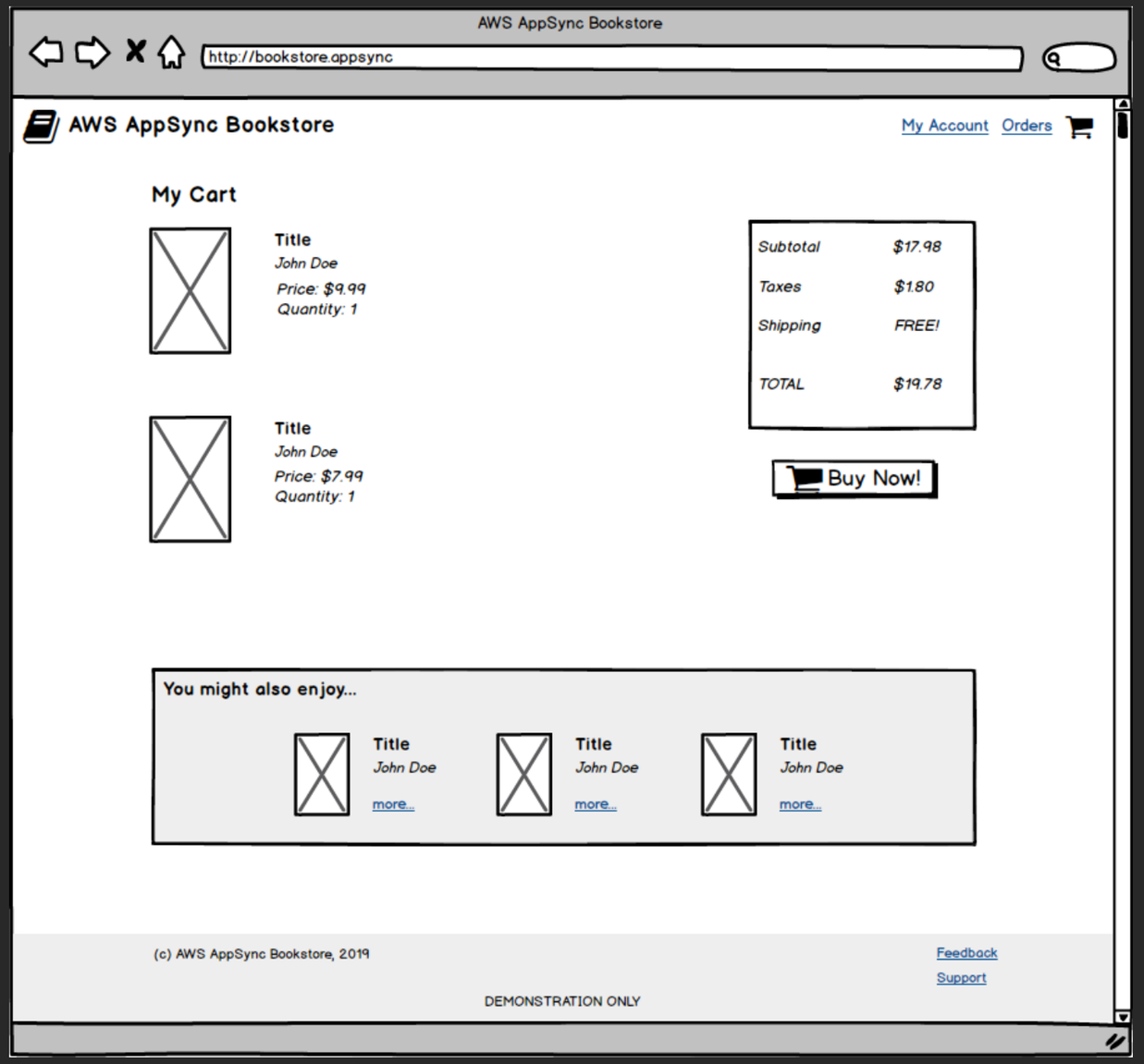
CTO of Amazon.com

AWS AppSync bookstore



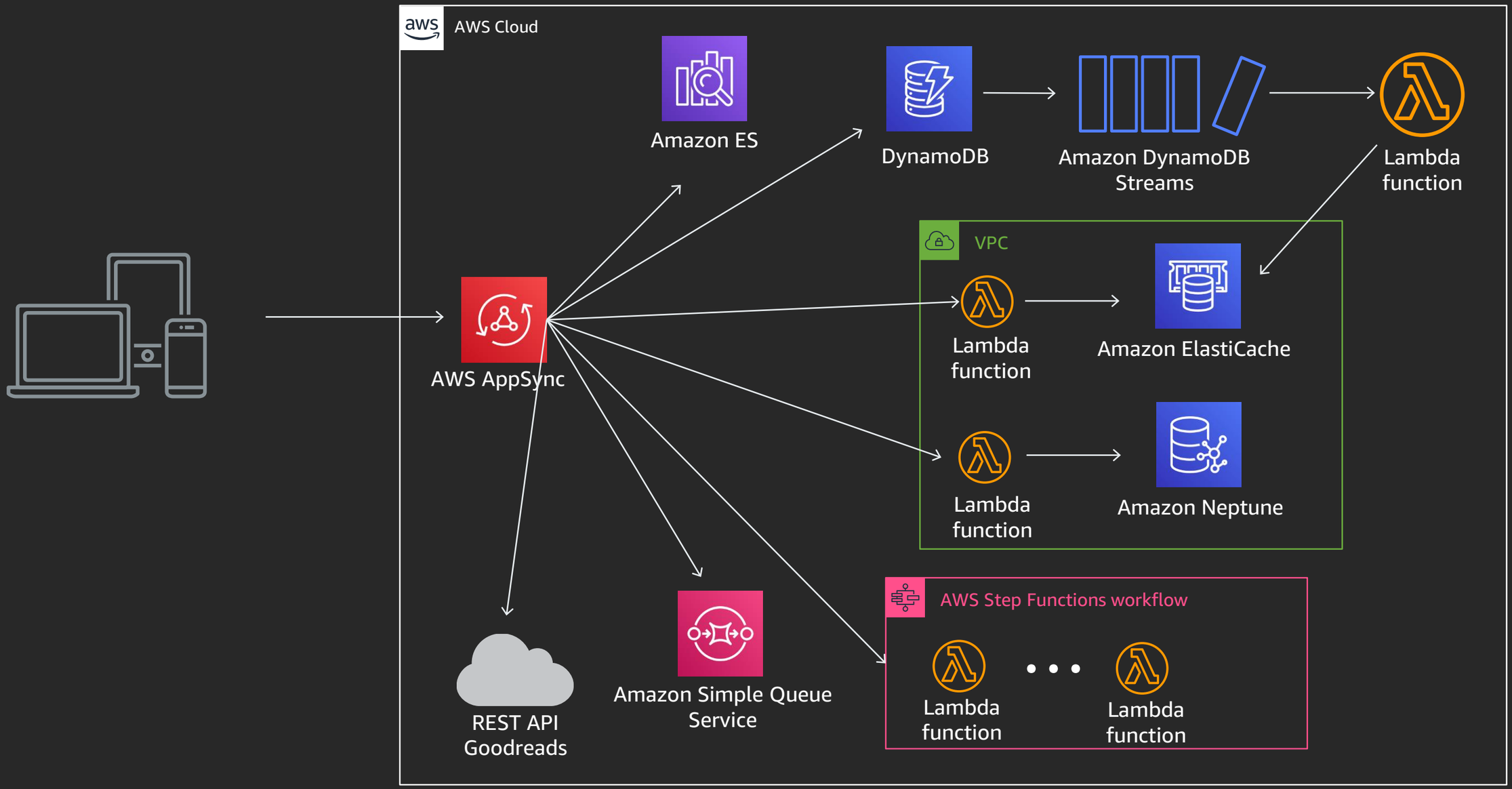
- What functionality do we need?
- What data sources might power that functionality?

AWS AppSync bookstore: Cart detail

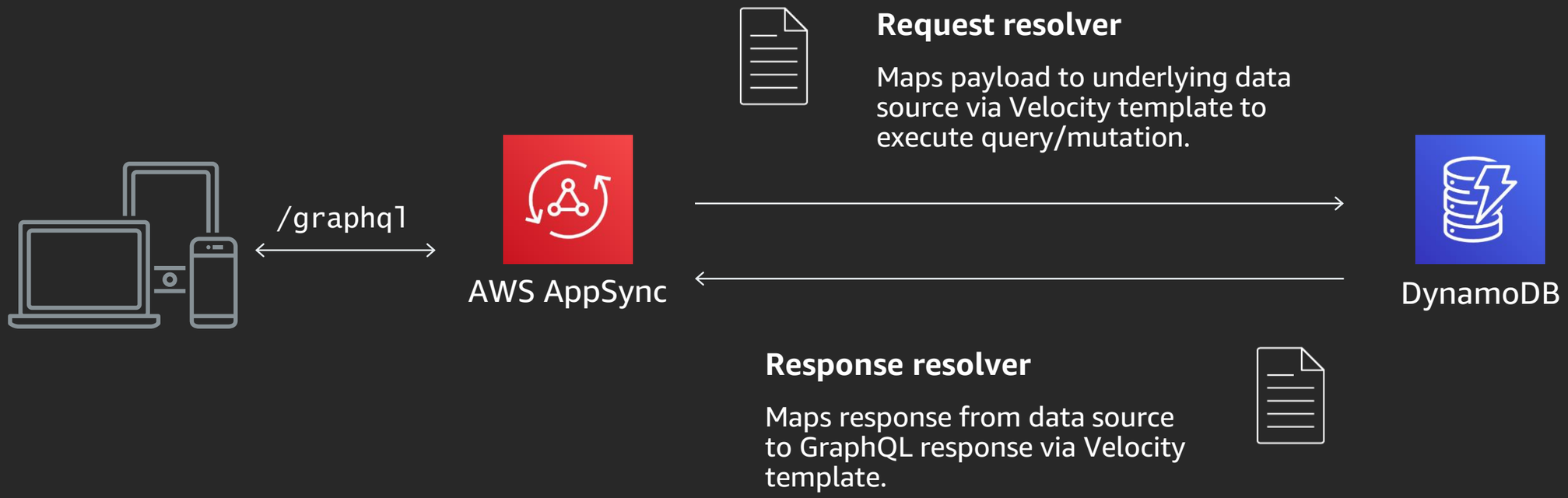


- What functionality do we need?
- What data sources might power that functionality?

AWS AppSync bookstore architecture



AWS AppSync resolvers



Request resolver

Maps payload to underlying data source via Velocity template to execute query/mutation.

Response resolver

Maps response from data source to GraphQL response via Velocity template.

Out-of-the-box data sources

Amazon DynamoDB

- Manage shopping cart (user)
- Read book catalog (public)
- Manage book catalog (administrators)

```
type Query {
  getBook(id: ID!): Book
  listBooks(limit: Int, nextToken: String): BookConnection
  listCartItems: [Cart]
}
```

Amazon Elasticsearch Service

- Search books

```
type Query {
  searchBook(filter: SearchFilter!): BookConnection
}
type BookConnection {
  items: [Book]
  nextToken: String
}
```

Modeling with AWS Amplify GraphQL Transform

```
type Book
  @model
  @searchable
  @key(name:“ByCategory”, fields:[“category”], queryField: “booksByCategory”)
  @auth(rules: [
    { allow: groups, groups: [“Admin”] },
    { allow: public, operations: [ read ], provider: iam }
  ])
{
  id: ID!
  isbn: String!
  title: String!
  ...
}
```

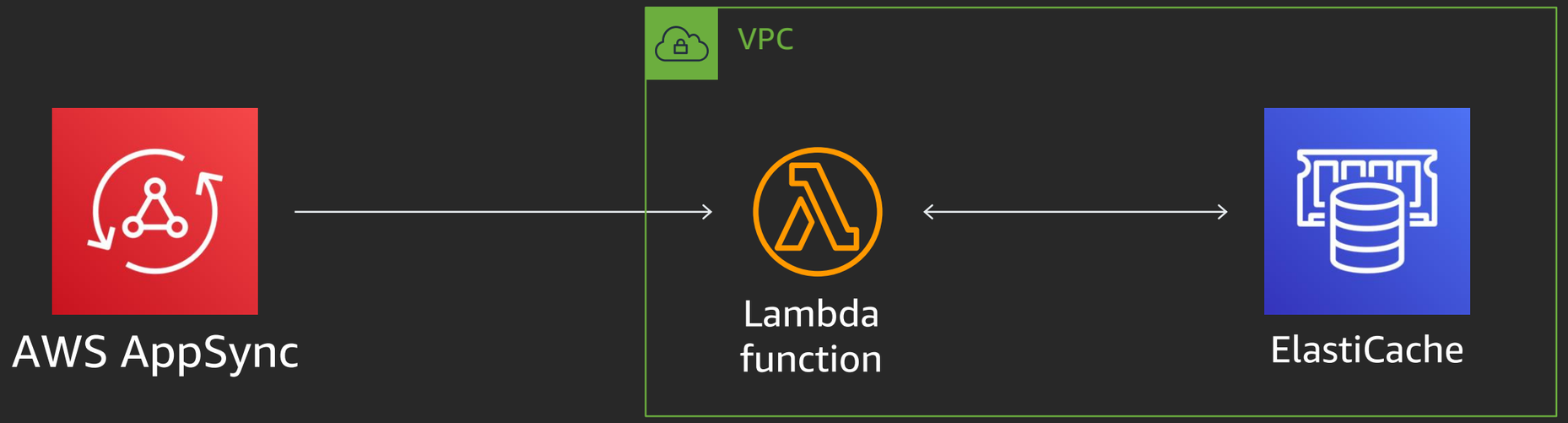
Purpose-built databases

Amazon ElastiCache

- List best-selling books

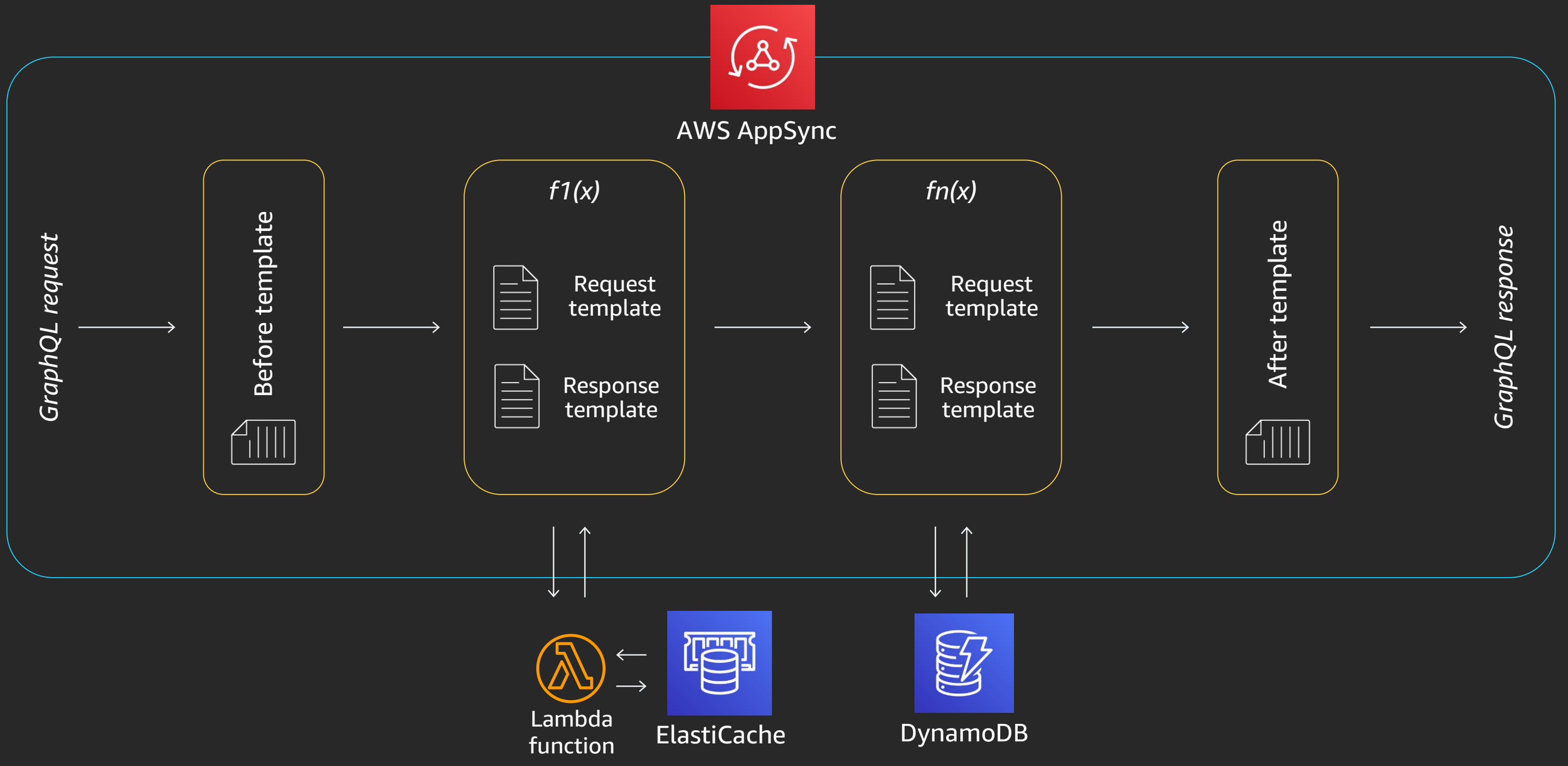
```
type Query {
  bestsellers(start: Int, end: Int): BookConnection
}
```

Utilize AWS Lambda to integrate with other databases




```
[
  “FD128CC0-670C-4ECB-96DC-30BD53C3BD74”,
  “8185DE0B-61E3-4EDD-A3F6-F377696C859A”,
  “025640FA-06AC-4D47-9CE9-7A4FD1ADB1EB”
]
```

Reusable templates with pipeline resolvers



Pipeline resolver #1: Get best-sellers

Invoke Lambda function

```
{
  "version": "2017-02-28",
  "operation": "Invoke",
  "payload": {
    "action": ”getBestsellers",
    "arguments":  $utils.toJson($ctx.args)
  }
}
```

Load data from ElastiCache

```
const Redis = require("ioredis");
const client = new Redis();
async function getBestsellers(s, e) {
  let result = await client.zrevrange(s, e);
  if (!result) { return []; }
  return result.map((r) => {
    return { id: r }
  };
}
```

Pipeline resolver #2: BatchGetItem

Request mapping template

```
#set($ids = [])
#foreach($result in $ctx.prev.result)
  #set($map = {})
  $util.qr($map.put("id", $util.dynamodb.toString($result.id)) )
  $util.qr($ids.add($map))
#end
{
  "version" : "2018-05-29",
  "operation" : "BatchGetItem",
  "tables" : {
    ”my_table_name": { "keys": $util.toJson($ids), "consistentRead": true }
  }
}
```

Amazon Neptune

- Recommendations by book
- Recommendations by shopper

```
type Query {
  recommendationsByBook(book: ID!, limit: Int, nextToken: String):
      BookConnection
  recommendationsByUser(user: ID!, limit: Int, nextToken: String):
      BookConnection
}
```

Pipeline resolver #1: Get recommendations

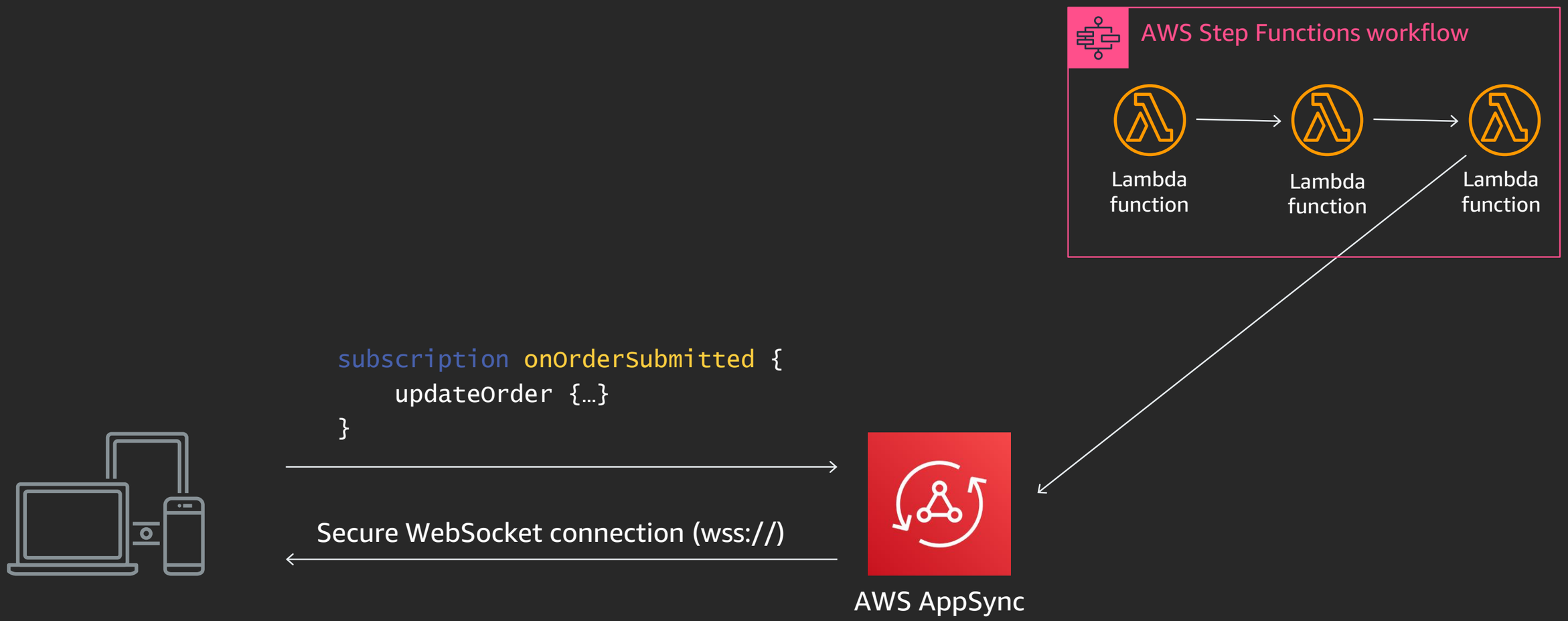
Invoke Lambda function

```
{
  "version": "2017-02-28",
  "operation": "Invoke",
  "payload": {
    "action": ”getRecommendations",
    "arguments":  $utils.toJson($ctx.args)
  }
}
```

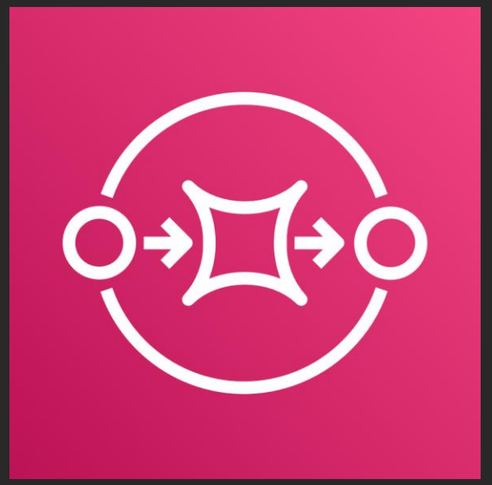
Load data from Neptune

```
const gremlin = require('gremlin’);
# connect to endpoint with gremlin client
async function getRecommendations(user) {
  # full Gremlin query not shown
  let result = g.V()...
  return result.value.map((r) => {
    return { id: r }
  };
}
```

Synchronize data in real time via subscriptions



AWS services



Amazon Simple Queue Service

- Submit feedback

```
type Mutation{
  submitFeedback(input: FeedbackInput!): Feedback
}
```

Amazon SQS: Data source

1. Configure AWS AppSync service role: AWS Identity and Access Management policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [ "sqs:SendMessage" ],
      "Effect": "Allow",
      "Resource": "arn:aws:sqs:us-east-1:444455556666:queue1"
    }
  ]
}
```

Amazon SQS: Data source

2. Configure HTTP data source **with authorization** (AWS Command Line Interface)

```
aws appsync create-data-source --api-id <API-ID> \
    --name SQS \
    --type HTTP \
    --http-config file://http.json
    --service-role-arn <ROLE-ARN>
```

http.json

```
{
  "endpoint": "https://sqs.us-east-1.amazonaws.com/",
  "authorizationConfig": {
    "authorizationType": "AWS_IAM",
    "awsIamConfig": {
      "signingRegion": "us-east-1",
      "signingServiceName": "sqs”
    }
  }
}
```

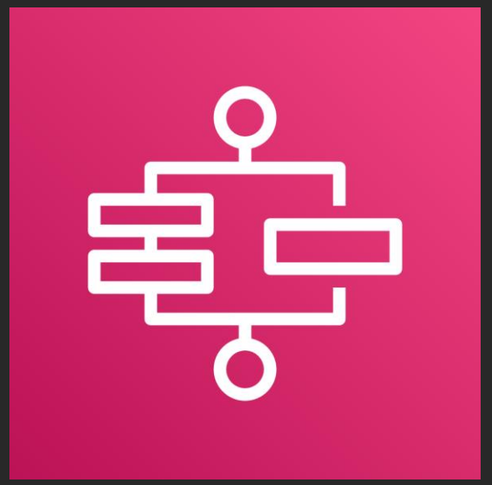
Amazon SQS: Send message resolvers

Request mapping template

```
{
  "version": "2018-05-29",
  "method": "GET",
  "resourcePath": "/ACCOUNT_ID/QUEUE_NAME",
  "params": {
    "query": {
      "Action": "SendMessage",
      "Version": "2012-11-05",
      "MessageBody": $util.toJson($ctx.args.input)
    }
  }
}
```

Response mapping template

```
#set($result = $utils.xml.toMap($ctx.result.body))
{
  "id": "$result.SendMessageResponse
              .SendMessageResult.MessageId”
}
```



AWS Step Functions

- Submit order for processing, including:
  - Calculate sales tax
  - Confirm inventory
  - Verifiy shipping selection

```
type Mutation{
  submitOrder(input: SubmitOrderInput!): Order
}
```

Step Functions: Data source

1. Configure AWS AppSync service role IAM Policy—`states:StartExecution`
2. Configure HTTP data source **with authorization** (AWS CloudFormation)

```
HttpConfig:
  Endpoint: !Sub https://states.${AWS::Region}.amazonaws.com/
  AuthorizationConfig:
    AuthorizationType: AWS_IAM
    AwsIamConfig:
      SigningRegion: !Ref AWS::Region
      SigningServiceName: states
```

Step Functions: Submit order

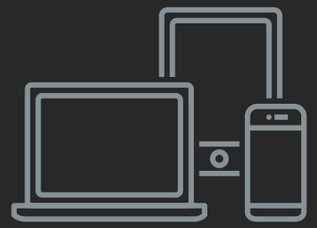
Request mapping template

```
$util.qr($ctx.stash.put(”orderId", $util.autoId()))
{
  "version": "2018-05-29",
  "method": "POST",
  "resourcePath": "/",
  "params": {
    "headers": {
      "content-type": "application/x-amz-json-1.0",
      "x-amz-target":"AWSStepFunctions.StartExecution”
    },
    "body": {
      "stateMachineArn": "arn:aws:states:REGION:ACCOUNT_ID:stateMachine:name",
      "input": "{ \"name\": \"$ctx.stash.orderId\”, \”cartId\": \"$ctx.args.cartId\” }”
    }
  }
}
```

Response mapping template

```
{
  "id": "$ctx.stash.orderId",
  "status": "PENDING”,
  ...
}
```

Updating user interface via subscription



```
mutation submitOrder(id) {...}
```

```
subscription onOrderUpdate {
    updateOrder {…}
}
```

AWS AppSync

states:startExecution

AWS Step Functions workflow

Lambda function

.

.

.

Lambda function

- HTTP POST to AWS AppSync endpoint
- GraphQL payload
- AWS IAM

```
type Mutation {
  submitOrder(cartId: ID!): Order
  updateOrder(orderId: ID!): Order @aws_iam
}
type Subscription {
  updateOrder(orderId: ID!): Order
    @aws_subscribe(mutations: [ "updateOrder" ])
}
```

AWS Secrets Manager

- It’s a secret

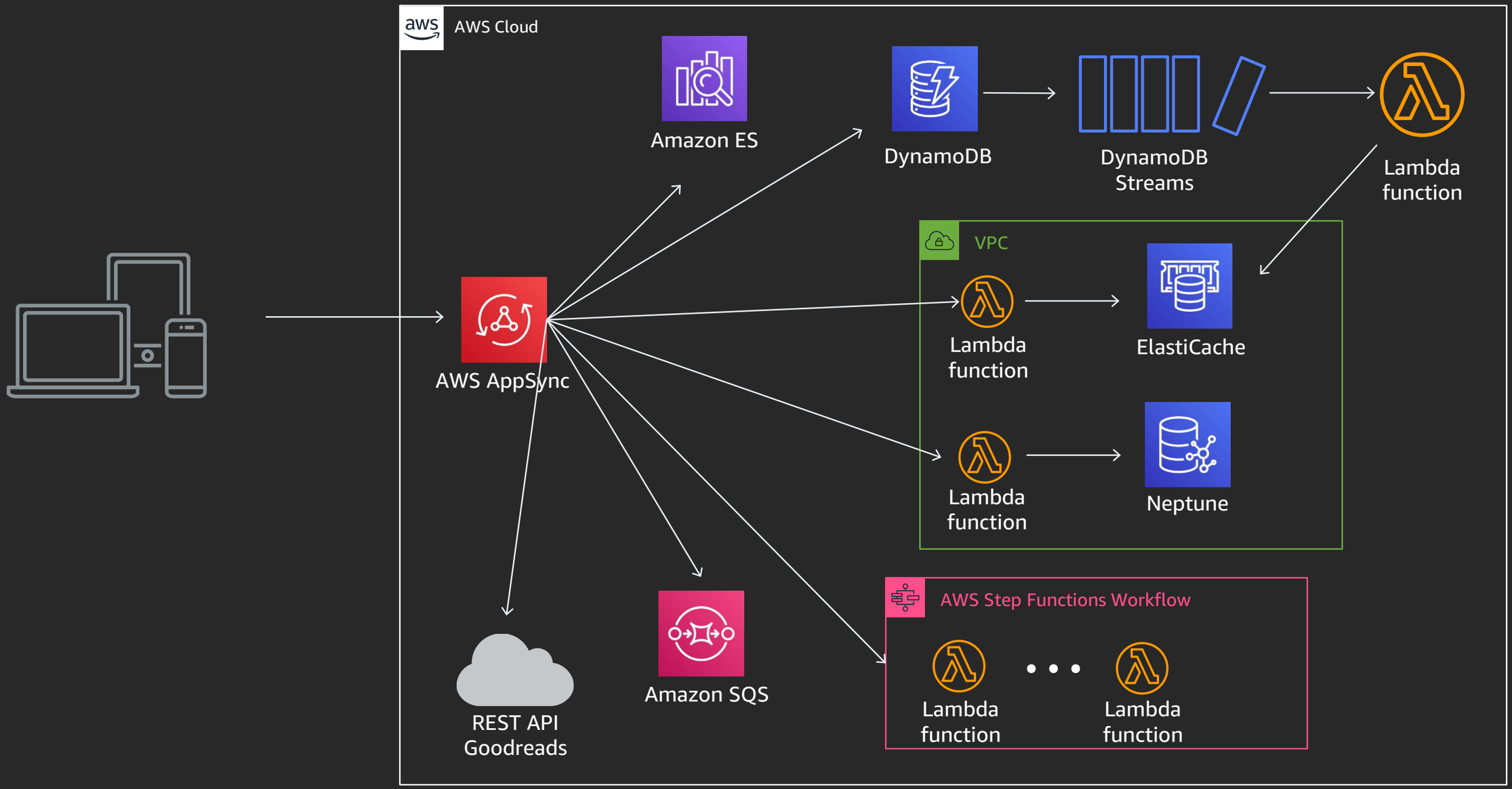
```
type Query{
  getSecret(secretId: ID!): Secret
}
```

Secrets Manager: Get secret

Request mapping template

```
{
  "version": "2018-05-29",
  "method": ”POST",
  "resourcePath": "/",
  "params": {
    "headers": {
      "content-type": "application/x-amz-json-1.1",
      "x-amz-target": "secretsmanager.GetSecretValue"
    },
    ”body”: {
      ”SecretId”: ”$ctx.args.secretId"
    }
  }
}
```

Review: AWS AppSync bookstore architecture



Thank you!

Josh Kahn

jkahn@amazon.com

AWS re:Invent

aws

Please complete the session survey in the mobile app.

References

- A one-size-fits-all database doesn't fit anyone
- Invoke AWS services directly from AWS AppSync
- AWS bookstore demo app