
AWS
re:Invent

FWM304

Unify access to siloed data with AWS AppSync GraphQL resolvers

Nader Dabit

Senior Developer Advocate
AWS

AWS re:Invent

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

Agenda

Introduction to AWS AppSync

Direct resolvers

Lambda resolvers

Live coding 🚀 – Building an AppSync API with CDK



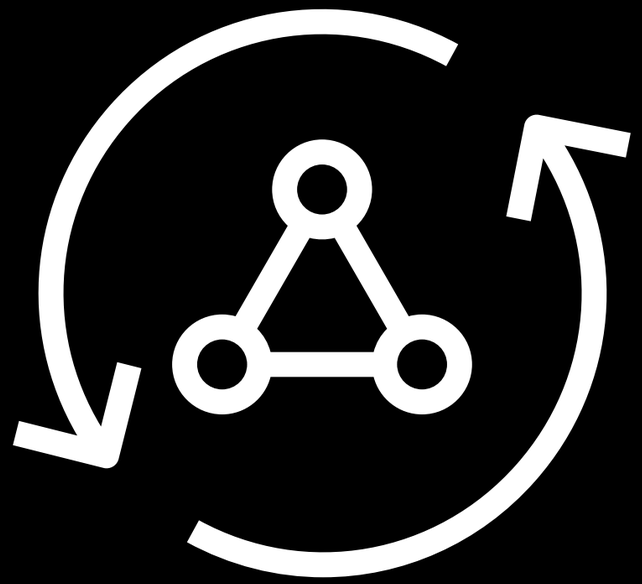
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS AppSync



AWS re:Invent

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



AWS AppSync

Build scalable applications on a range of data sources, including those requiring real-time updates and offline data access

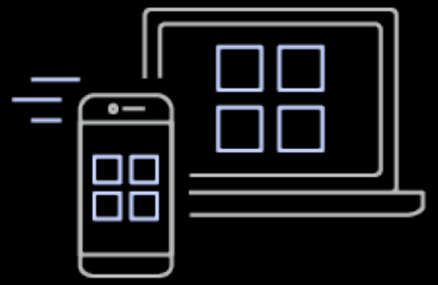
AWS AppSync



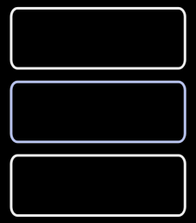
Managed serverless GraphQL service



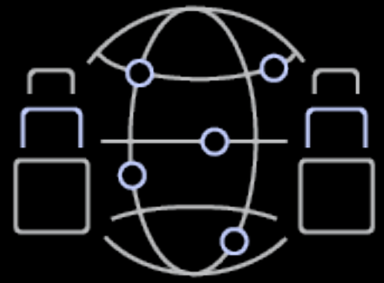
GraphQL facade for any AWS service



Add data sync, real-time, and offline capabilities for any data source or API



Server-side data caching

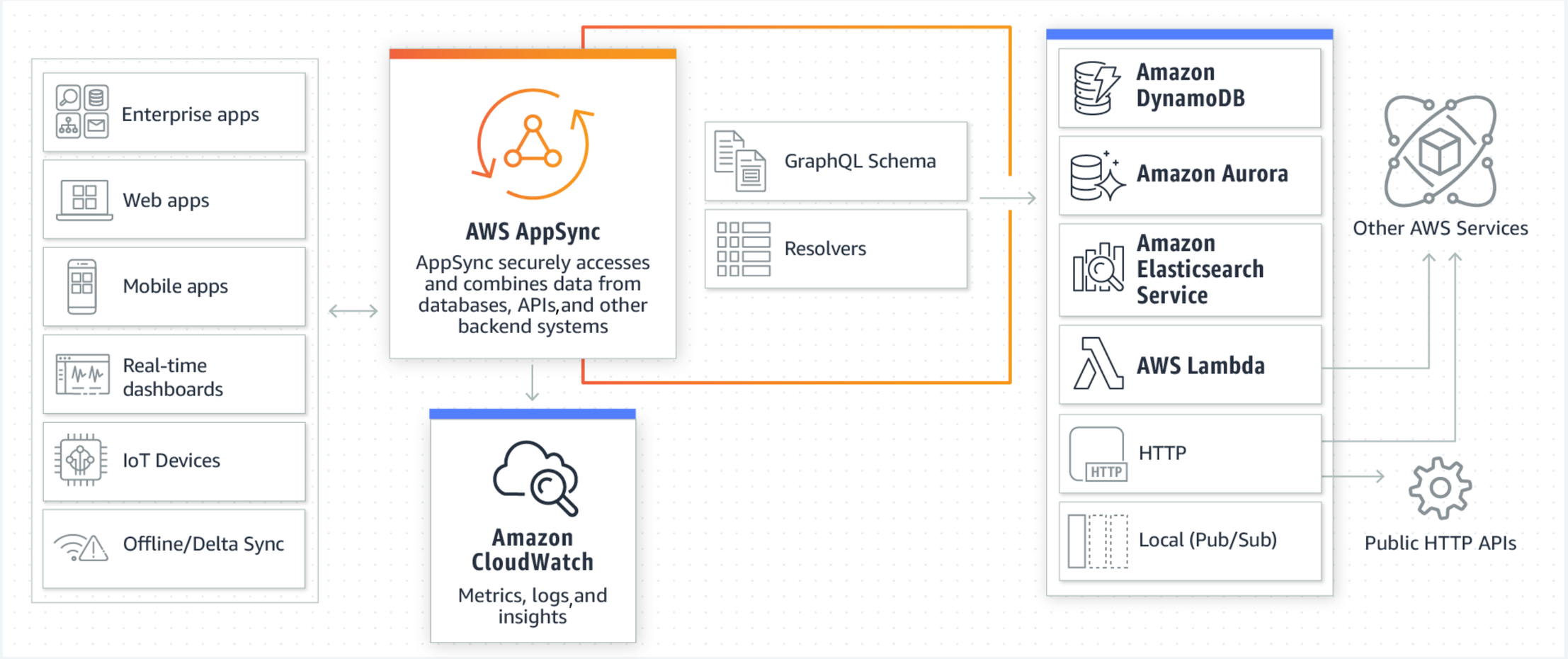


Conflict detection and resolution in the cloud

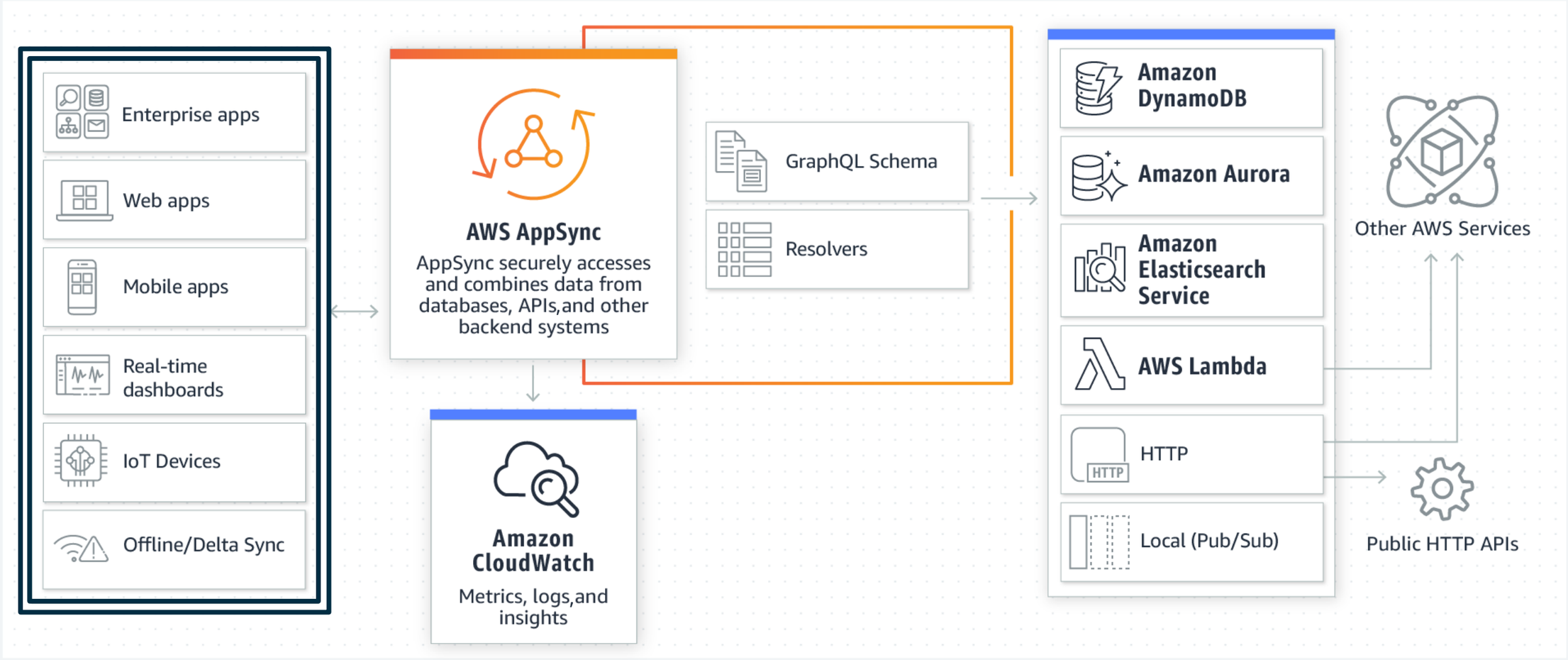


Enterprise security features: IAM, Amazon Cognito, OIDC, API keys

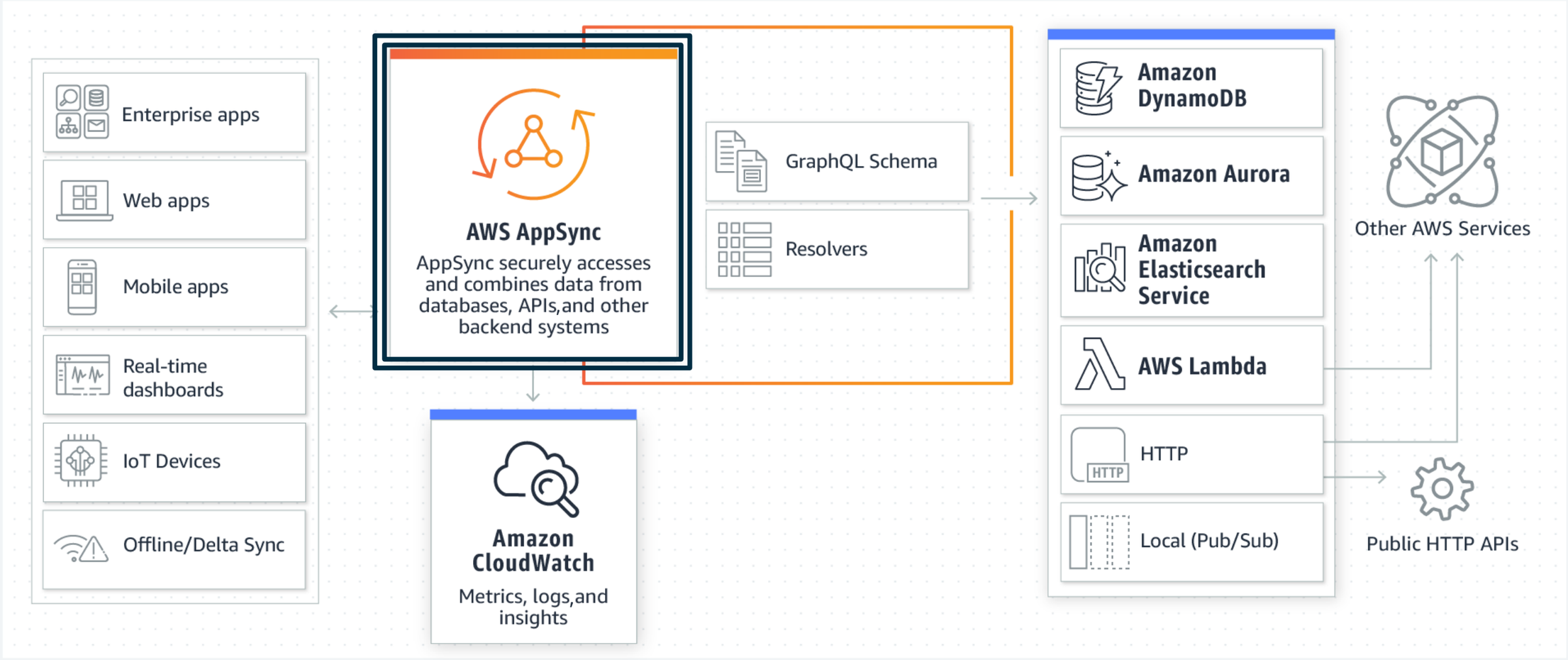
How does AWS AppSync work?



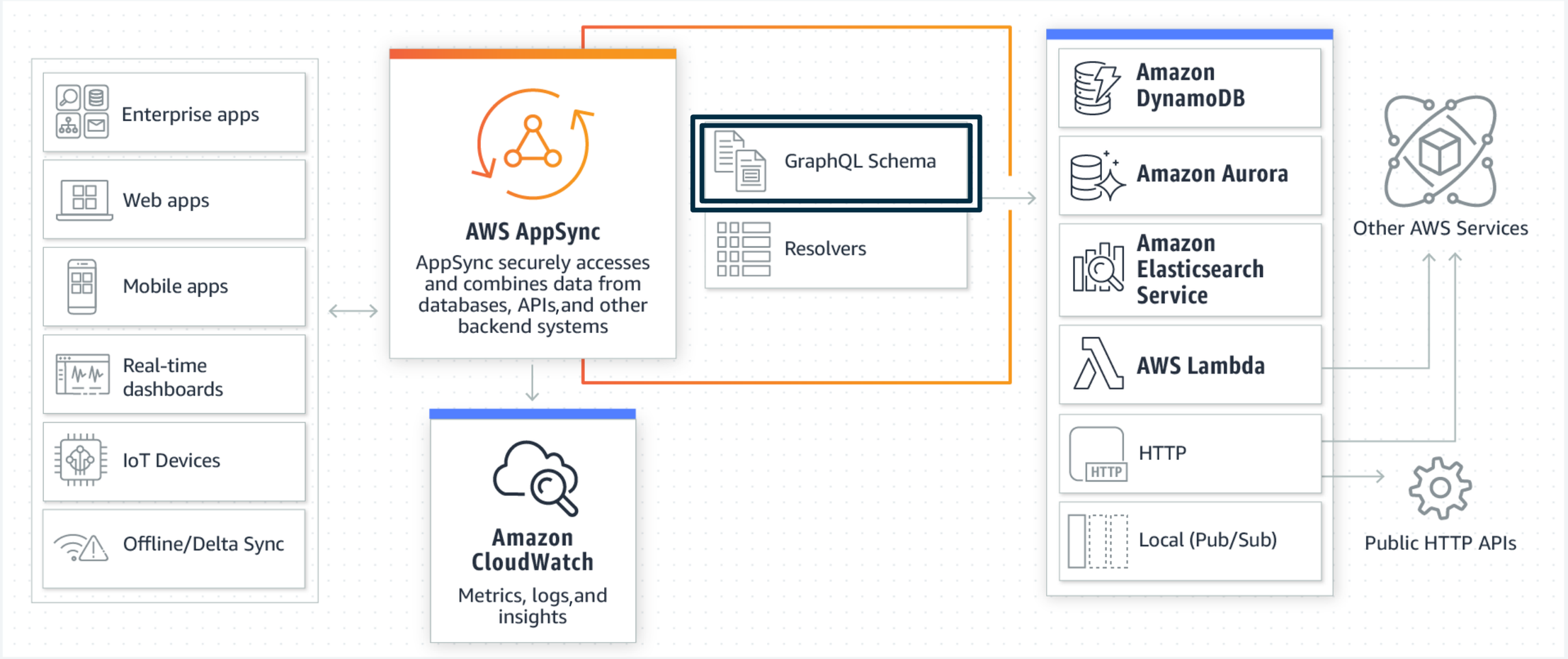
How does AWS AppSync work?



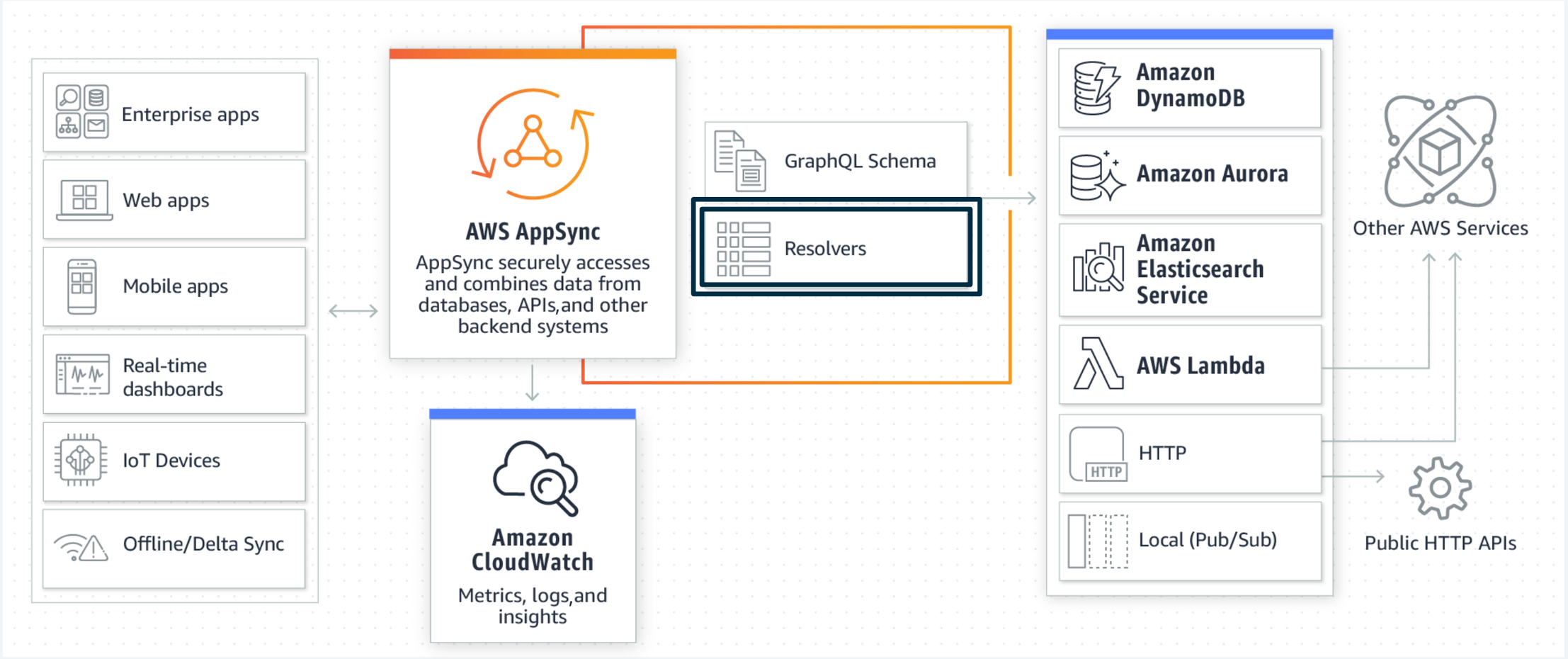
How does AWS AppSync work?



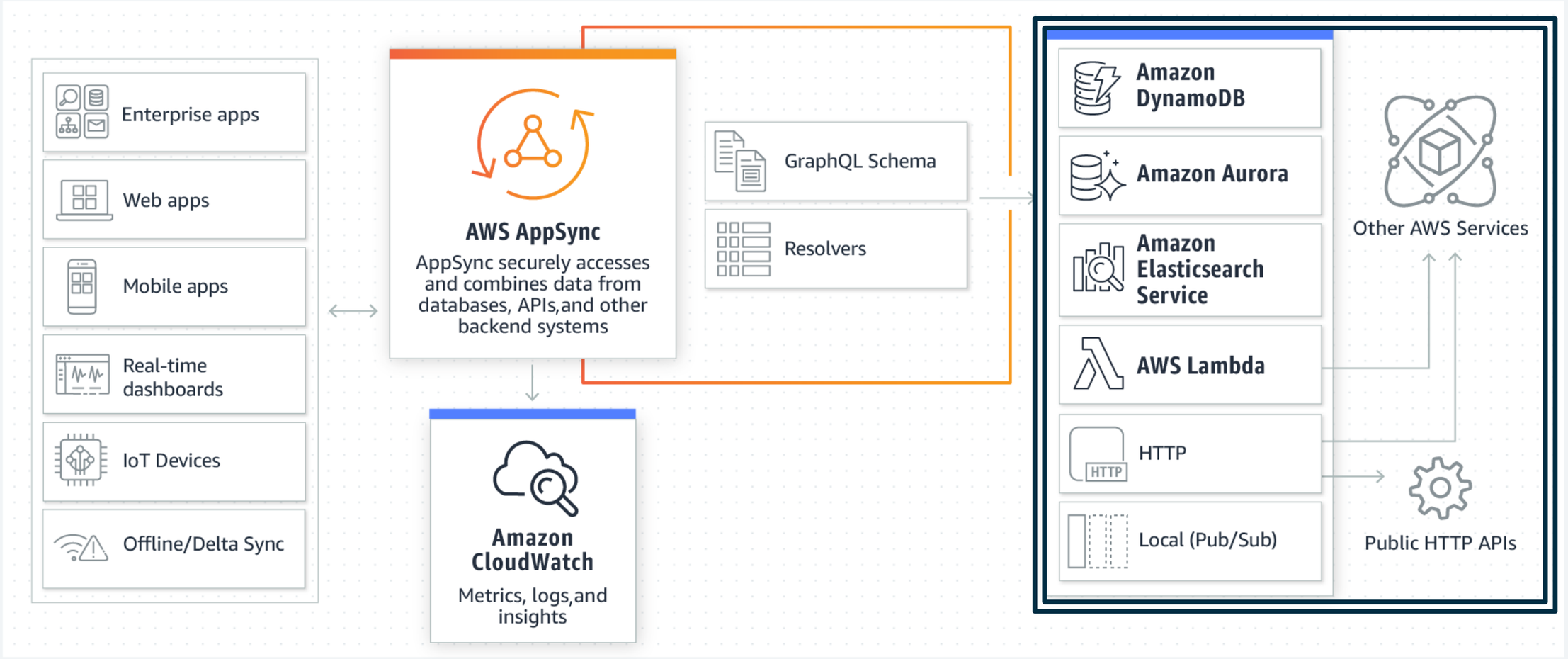
How does AWS AppSync work?



How does AWS AppSync work?



How does AWS AppSync work?



Anatomy of a GraphQL API



AWS re:Invent

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

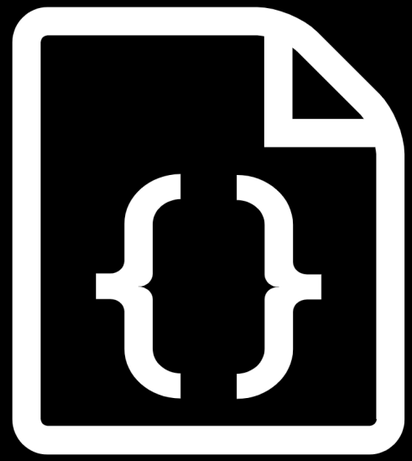
Anatomy of a GraphQL API

1. Schema
2. Resolvers
3. Data sources



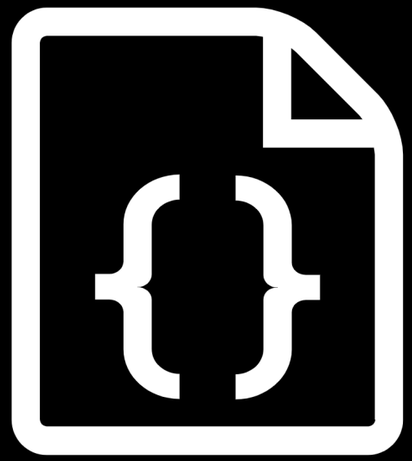
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Anatomy of a GraphQL API



Schema

Anatomy of a GraphQL API



Schema

Data

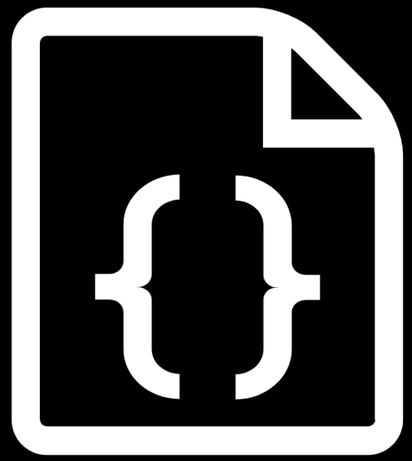
```
type Todo {
  id: ID!
  name: String!
  completed: Boolean!
}
```

Operations

```
type Query {
  getTodo(id: ID!): Todo
}

type Mutation {
  createTodo(id: ID! name: String! completed: Boolean): Todo
}
```

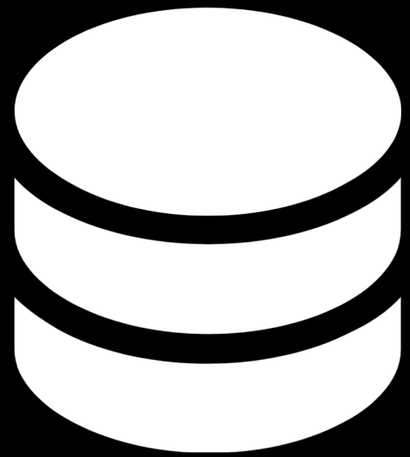
Anatomy of a GraphQL API



Schema

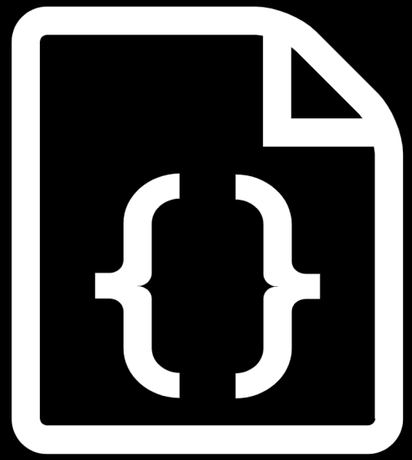
Query / mutation

Data

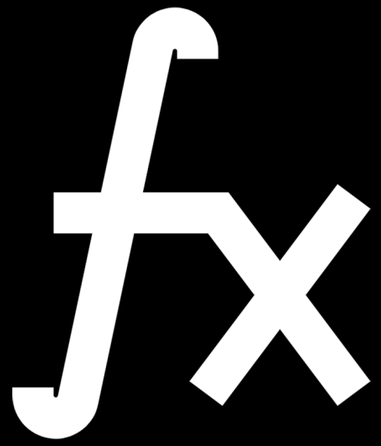


Data sources

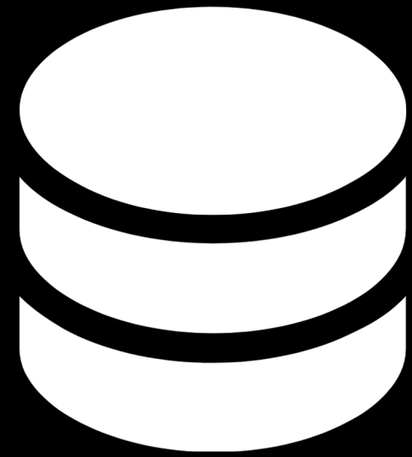
Anatomy of a GraphQL API



Schema



Resolvers



Data sources

AWS AppSync resolvers

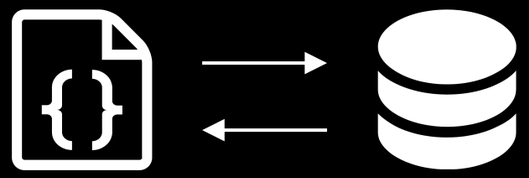


AWS re:Invent

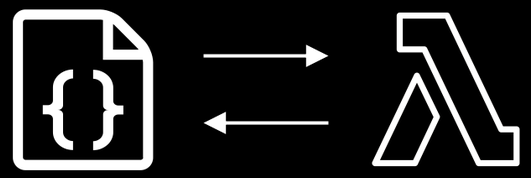
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS AppSync resolvers

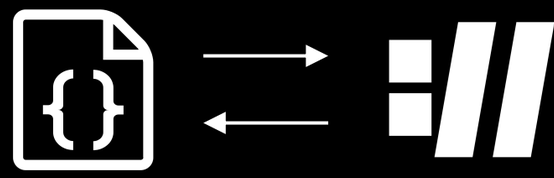
Direct



AWS Lambda



HTTP



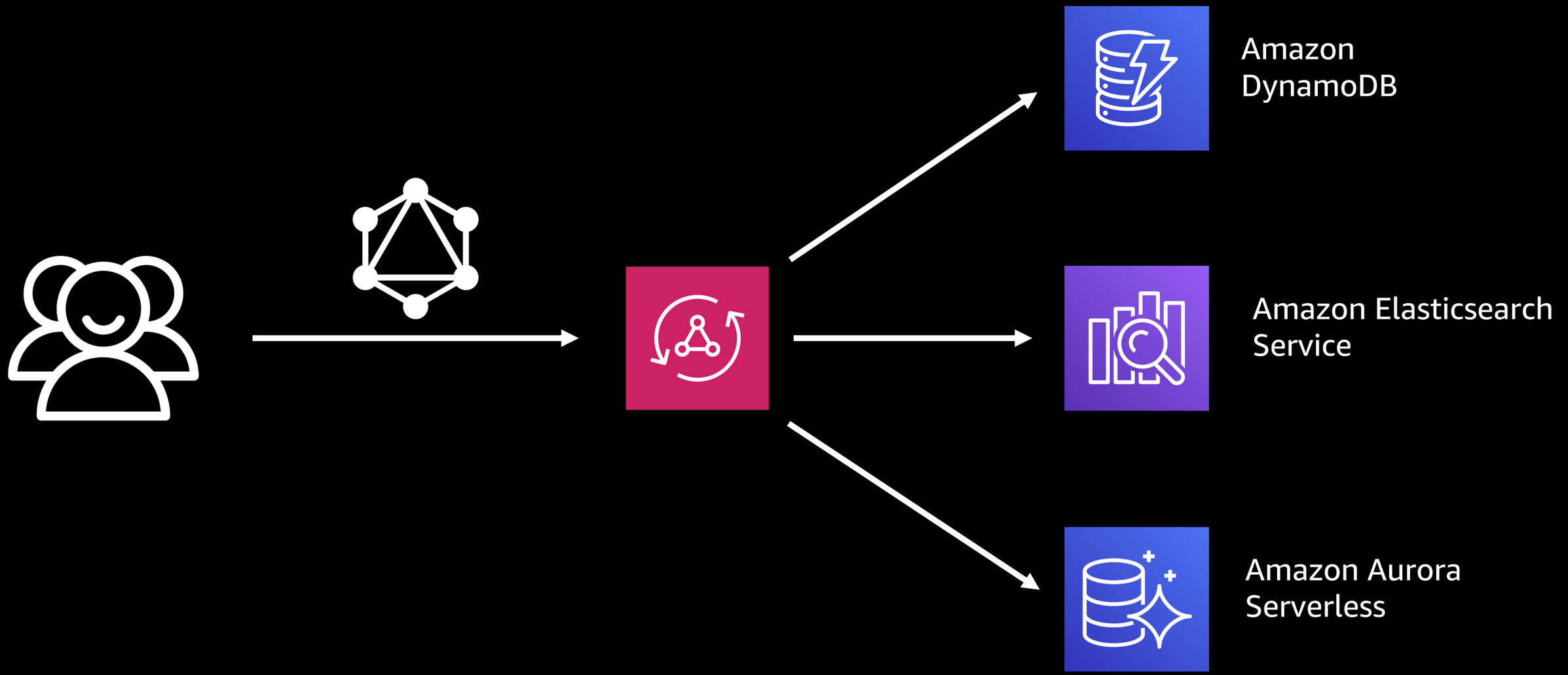
AWS AppSync resolver types



Direct resolvers

AWS AppSync resolver types

DIRECT RESOLVERS



AWS AppSync resolver types

DIRECT RESOLVERS

```
type Product {
  id: ID!
  name: String!
  price: Float
  warehouse: Warehouse
}

type Warehouse {
  warehouseId: ID!
  name: String!
  location: String!
}

type Query {
  getProduct(productId: ID!): Product
}
```

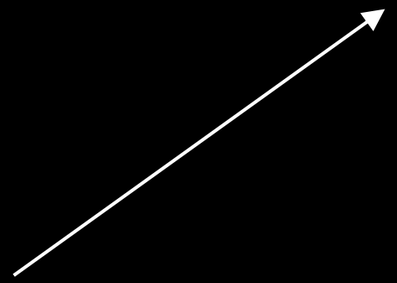
AWS AppSync resolver types

DIRECT RESOLVERS

```
type Product {
  id: ID!
  name: String!
  price: Float
  warehouse: Warehouse
}

type Warehouse {
  warehouseId: ID!
  name: String!
  location: String!
}

type Query {
  getProduct(productId: ID!): Product
}
```



```
{
  "data": {
    "getProduct": {
      "id": "001",
      "name": "Yeezys",
      "price": 300.00,
      "warehouse": {
        "warehouseId": 103,
        "name": "SFBay",
        "location": "San Francisco"
      }
    }
  }
}
```

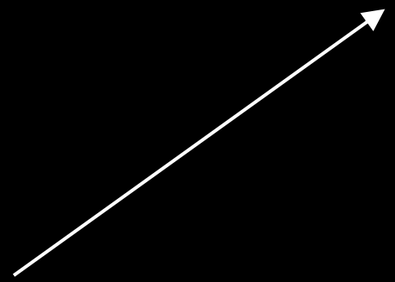
AWS AppSync resolver types

DIRECT RESOLVERS

```
type Product {
  id: ID!
  name: String!
  price: Float
  warehouse: Warehouse
}

type Warehouse {
  warehouseId: ID!
  name: String!
  location: String!
}

type Query {
  getProduct(productId: ID!): Product
}
```



```
{
  "data": {
    "getProduct": {
      "id": "001",
      "name": "Yeezys",
      "price": 300.00,
      "warehouse": {
        "warehouseId": 103,
        "name": "SFBay",
        "location": "San Francisco"
      }
    }
  }
}
```

AWS AppSync resolver types

DIRECT RESOLVERS





Amazon DynamoDB



AWS Lambda



Amazon Aurora Serverless

AWS AppSync resolver types

DIRECT RESOLVERS



AWS AppSync

Request mapping template

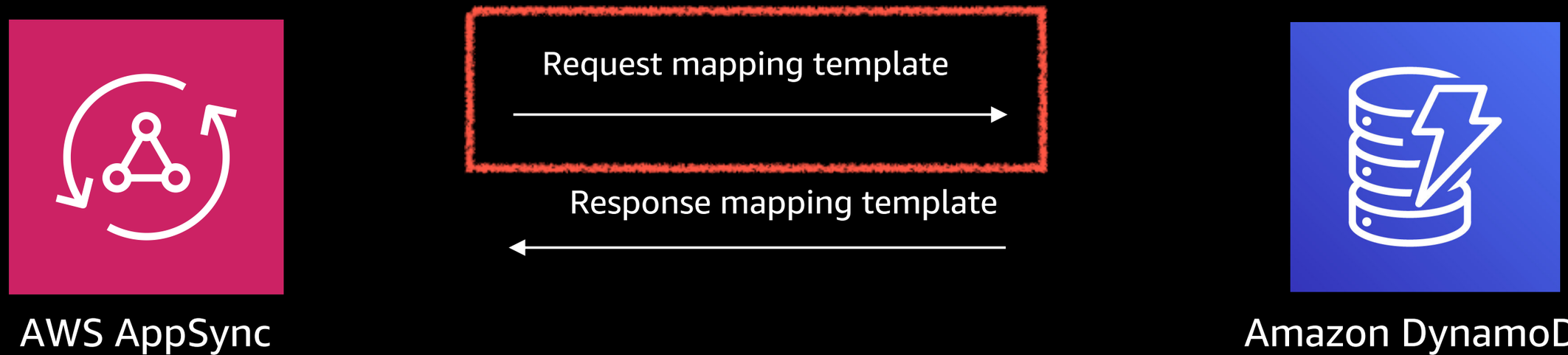
Response mapping template



Amazon DynamoDB

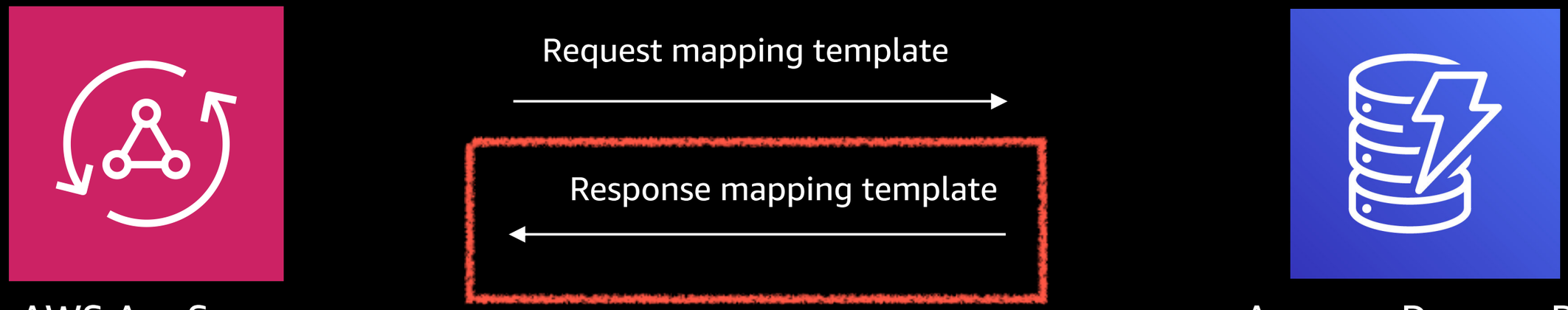
AWS AppSync resolver types

DIRECT RESOLVERS



AWS AppSync resolver types

DIRECT RESOLVERS



AWS AppSync resolver types

DIRECT RESOLVERS



AWS AppSync

```
{
  "version": "2017-02-28",
  "operation": "PutItem",
  "key": {
    "id": $util.dynamodb.toDynamoDBJson($util.autoId()),
  },
  "attributeValues": $util.dynamodb.toMapValuesJson($ctx.args.input)
}
```

Request mapping template





Amazon DynamoDB

AWS AppSync resolver types

DIRECT RESOLVERS



AWS AppSync

```
{
  "version": "2017-02-28",
  "operation": "PutItem",
  "key": {
    "id": $util.dynamodb.toDynamoDBJson($util.autoId()),
  },
  "attributeValues": $util.dynamodb.toMapValuesJson($ctx.args.input)
}
```

Request mapping template





Amazon DynamoDB

AWS AppSync resolver types

DIRECT RESOLVERS



AWS AppSync

```
$util.toJson($context.result)
```

Response mapping template



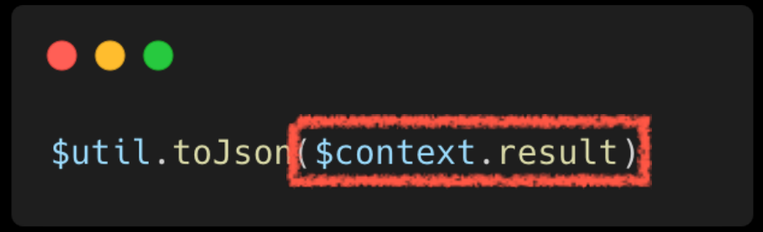
Amazon DynamoDB

AWS AppSync resolver types

DIRECT RESOLVERS



AWS AppSync



Response mapping template





Amazon DynamoDB

AWS AppSync resolver types

DIRECT RESOLVERS

Context object

The $context object is a map that holds all of the contextual information for your resolver invocation.

```
{
  "version": "2017-02-28",
  "operation": "PutItem",
  "key": {
    "id": $util.dynamodb.toDynamoDBJson($util.autoId()),
  },
  "attributeValues": $util.dynamodb.toMapValuesJson($ctx.args.input)
}
```

AWS AppSync resolver types

DIRECT RESOLVERS

Context object

The $context object is a map that holds all of the contextual information for your resolver invocation

```
{
  "version": "2017-02-28",
  "operation": "PutItem",
  "key": {
    "id": $util.dynamodb.toDynamoDBJson($util.autoId()),
  },
  "attributeValues": $util.dynamodb.toMapValuesJson($ctx.args.input)
}
```

AWS AppSync resolver types

DIRECT RESOLVERS

Context object



```
{
    "arguments" : { ... },
    "source" : { ... },
    "result" : { ... },
    "identity" : { ... },
    "request" : { ... },
    "info": { ... }
}
```

AWS AppSync resolver types

DIRECT RESOLVERS

Context object



Arguments

A map that contains all GraphQL arguments for this field

AWS AppSync resolver types

DIRECT RESOLVERS

Context object



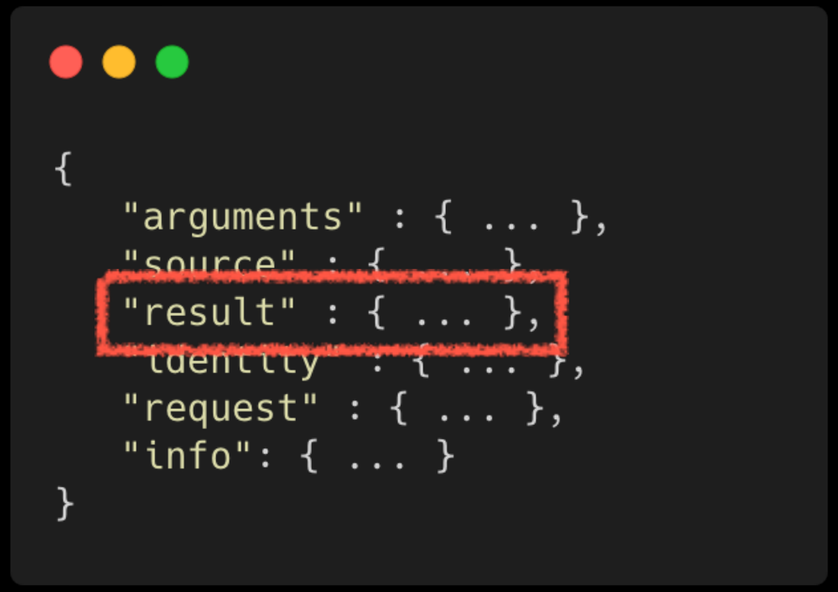
Source

A map that contains the resolution of the parent field

AWS AppSync resolver types

DIRECT RESOLVERS

Context object



Result

A container for the results of this resolver

This field is only available to response mapping templates

AWS AppSync resolver types

DIRECT RESOLVERS

Context object



Identity

An object that contains information about the caller

AWS AppSync resolver types

DIRECT RESOLVERS

Context object



Request

Additional metadata about the request, including the headers

AWS AppSync resolver types

DIRECT RESOLVERS

Context object



Info

An object that contains information about the GraphQL request

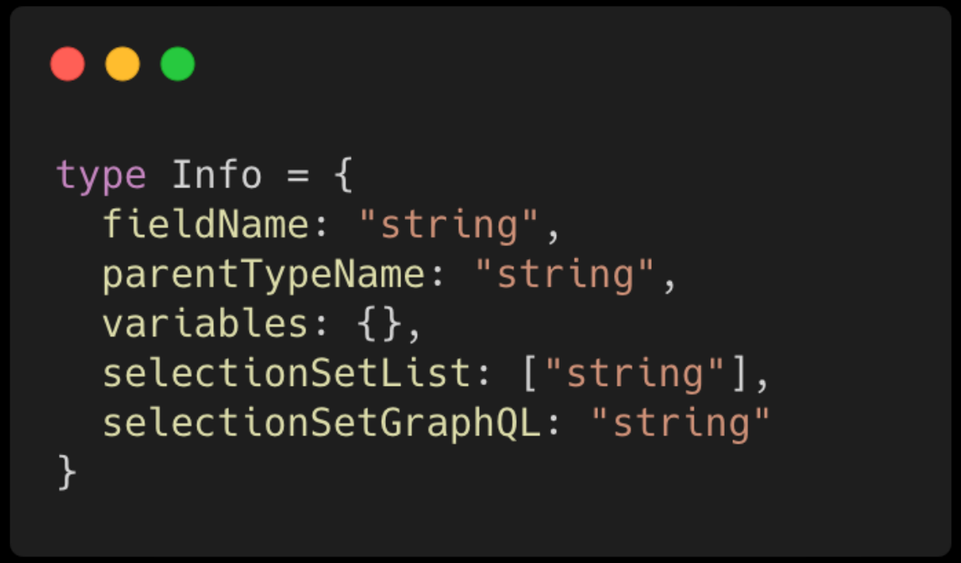
AWS AppSync resolver types

DIRECT RESOLVERS

Context object

Info

An object that contains information about the GraphQL request



```
type Info = {
  fieldName: "string",
  parentTypeName: "string",
  variables: {},
  selectionSetList: ["string"],
  selectionSetGraphQL: "string"
}
```

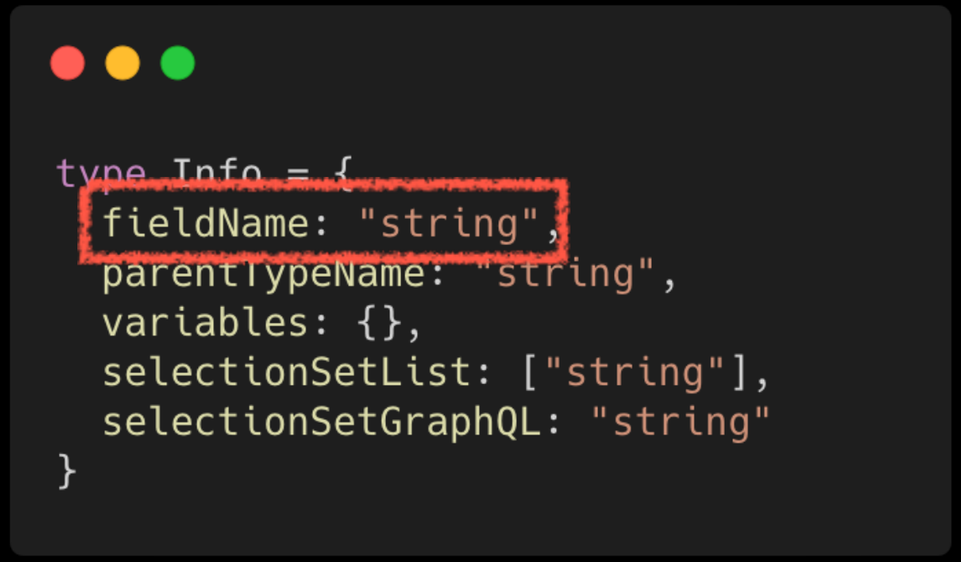
AWS AppSync resolver types

DIRECT RESOLVERS

Context object

Info

An object that contains information about the GraphQL request



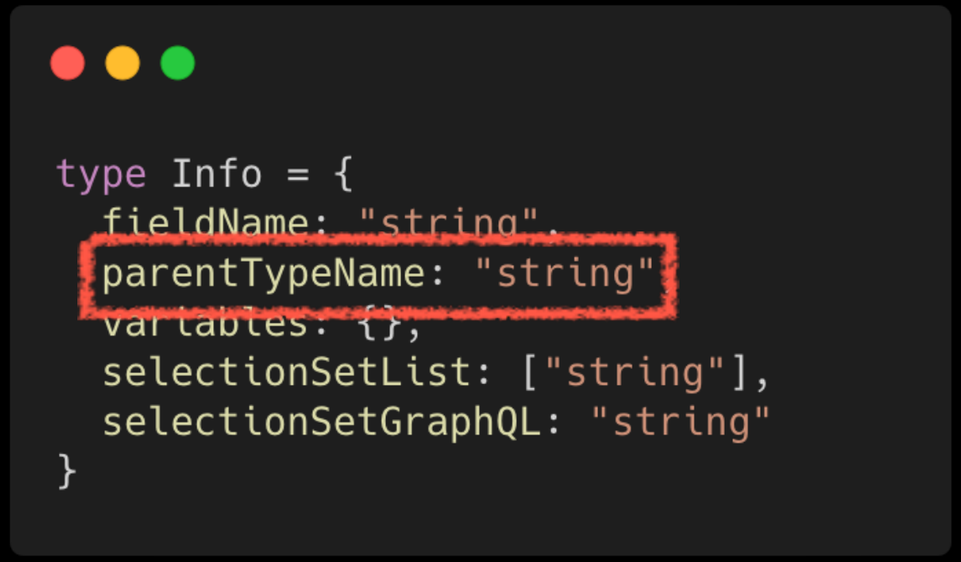
AWS AppSync resolver types

DIRECT RESOLVERS

Context object

Info

An object that contains information about the GraphQL request



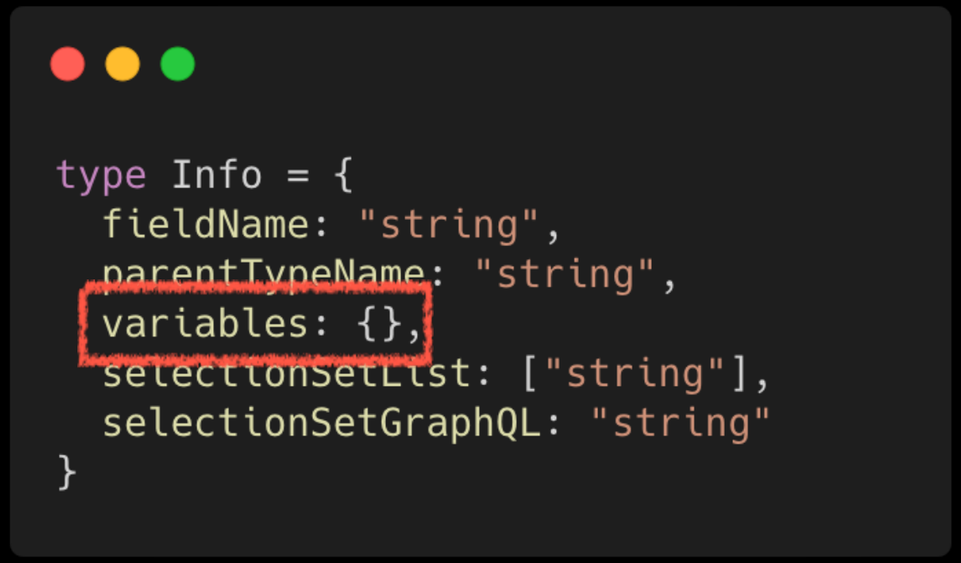
AWS AppSync resolver types

DIRECT RESOLVERS

Context object

Info

An object that contains information about the GraphQL request



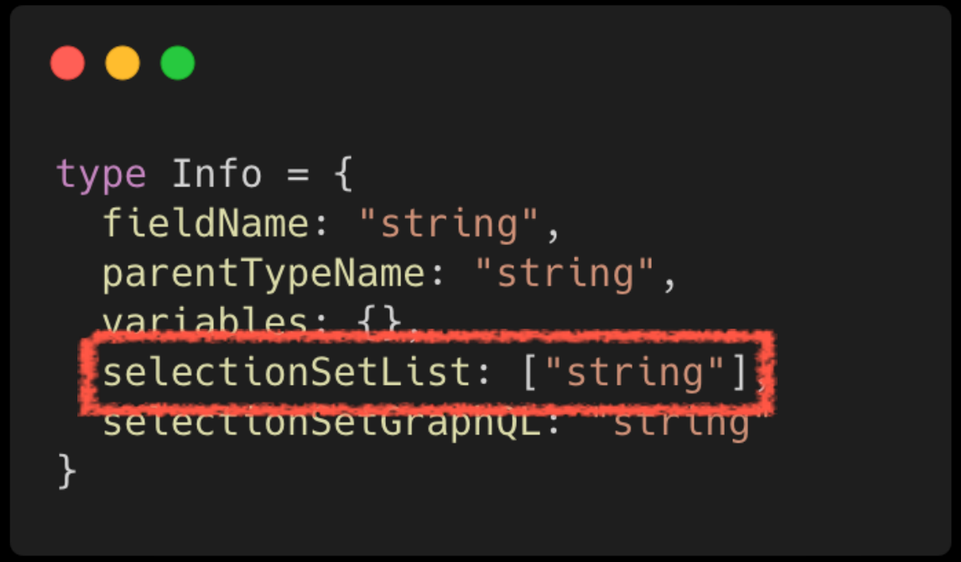
AWS AppSync resolver types

DIRECT RESOLVERS

Context object

Info

An object that contains information about the GraphQL request



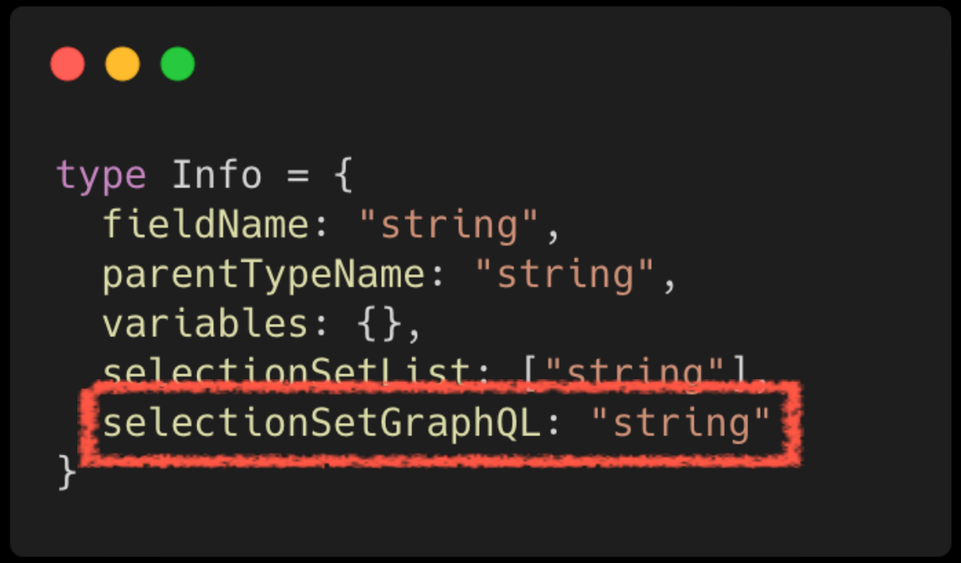
AWS AppSync resolver types

DIRECT RESOLVERS

Context object

Info

An object that contains information about the GraphQL request



AWS AppSync resolver types



AWS Lambda resolvers

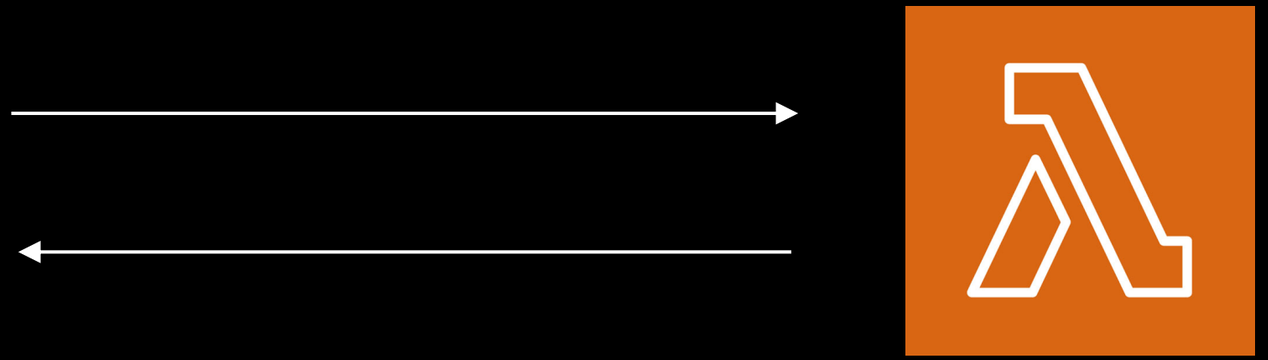
AWS AppSync resolver types

AWS LAMBDA RESOLVERS

```
type Product {
  id: ID!
  name: String!
  price: Float
  warehouse: Warehouse
}

type Warehouse {
  warehouseId: ID!
  name: String!
  location: String!
}

type Query {
  getProduct(productId: ID!): Product
}
```



AWS AppSync resolver types

AWS LAMBDA RESOLVERS

```
type Product {
  id: ID!
  name: String!
  price: Float
  warehouse: Warehouse
}

type Warehouse {
  warehouseId: ID!
  name: String!
  location: String!
}

type Query {
  getProduct(productId: ID!): Product
}
```



```
import createProduct from './createProduct'
import listProducts from './listProducts'
import getProduct from './getProduct'

exports.handler = async (event:AppSyncEvent) => {
    switch (event.info.fieldName) {

      case "getProduct":
        return await getProduct(event.arguments.productId)

      case "createProduct":
        return await createProduct(event.arguments.product);

      case "listProducts":
        return await listProducts();

      default:
        return null;
    }
}
```

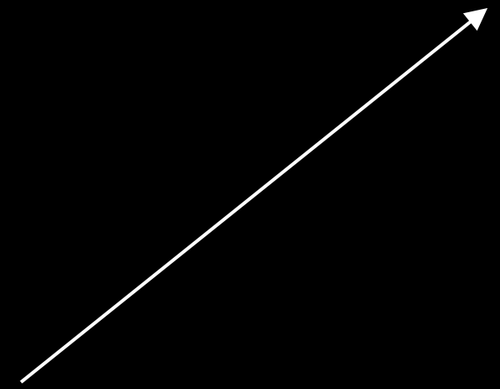
AWS AppSync resolver types

AWS LAMBDA RESOLVERS

```
type Product {
  id: ID!
  name: String!
  price: Float
  warehouse: Warehouse
}

type Warehouse {
  warehouseId: ID!
  name: String!
  location: String!
}

type Query {
  getProduct(productId: ID!): Product
}
```



```
import createProduct from './createProduct'
import listProducts from './listProducts'
import getProduct from './getProduct'

exports.handler = async (event:AppSyncEvent) => {
    switch (event.info.fieldName) {

      case "getProduct":
        return await getProduct(event.arguments.productId)

      case "createProduct":
        return await createProduct(event.arguments.product);

      case "listProducts":
        return await listProducts();

      default:
        return null;
    }
}
```

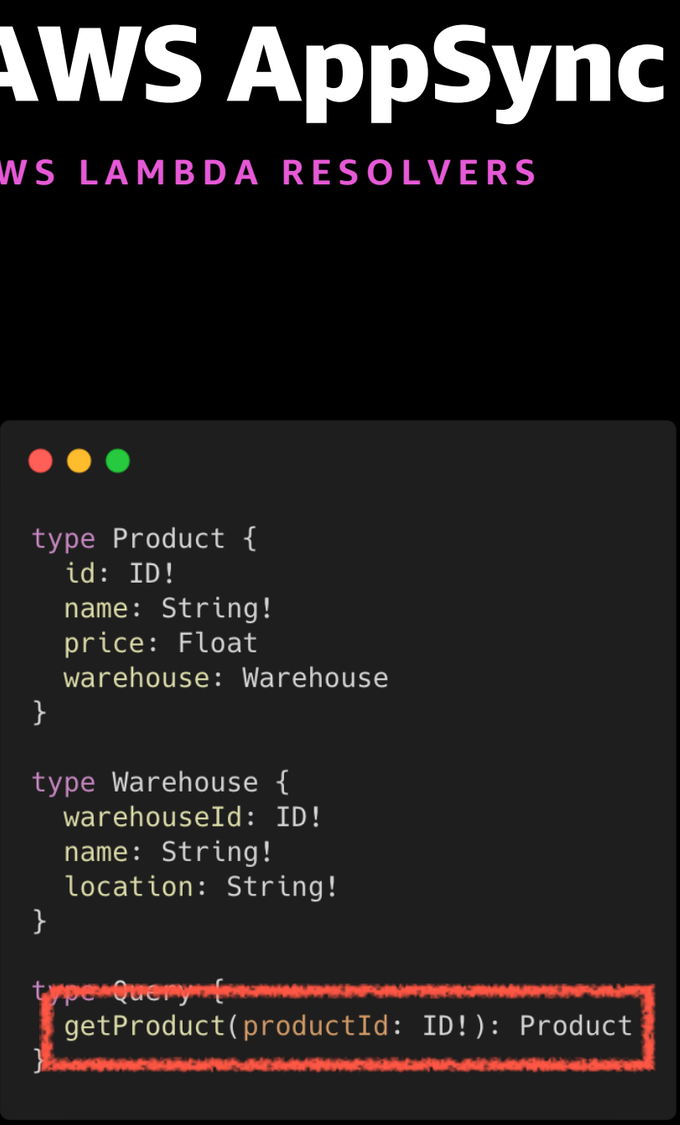
AWS AppSync resolver types

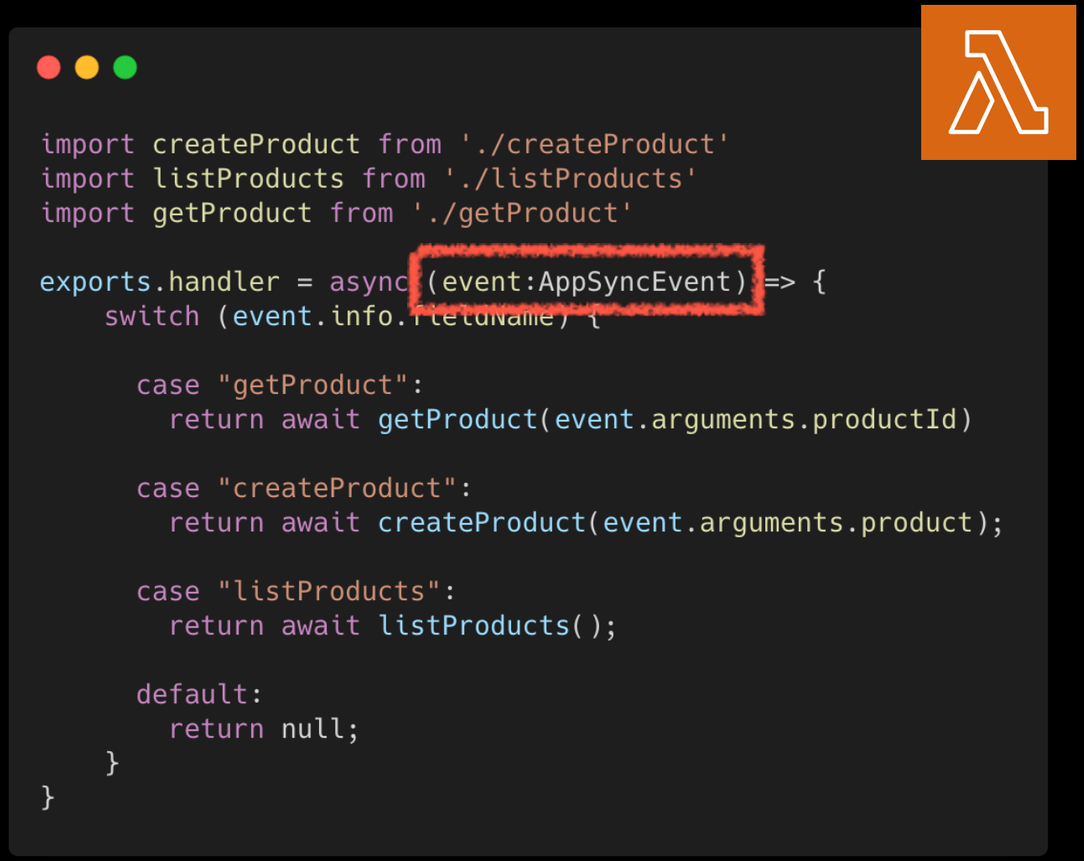
AWS LAMBDA RESOLVERS

```
type Product {
  id: ID!
  name: String!
  price: Float
  warehouse: Warehouse
}

type Warehouse {
  warehouseId: ID!
  name: String!
  location: String!
}

type Query {
  getProduct(productId: ID!): Product
}
```





```
import createProduct from './createProduct'
import listProducts from './listProducts'
import getProduct from './getProduct'

exports.handler = async (event:AppSyncEvent) => {
    switch (event.info.fieldName) {

      case "getProduct":
        return await getProduct(event.arguments.productId)

      case "createProduct":
        return await createProduct(event.arguments.product);

      case "listProducts":
        return await listProducts();

      default:
        return null;
    }
}
```

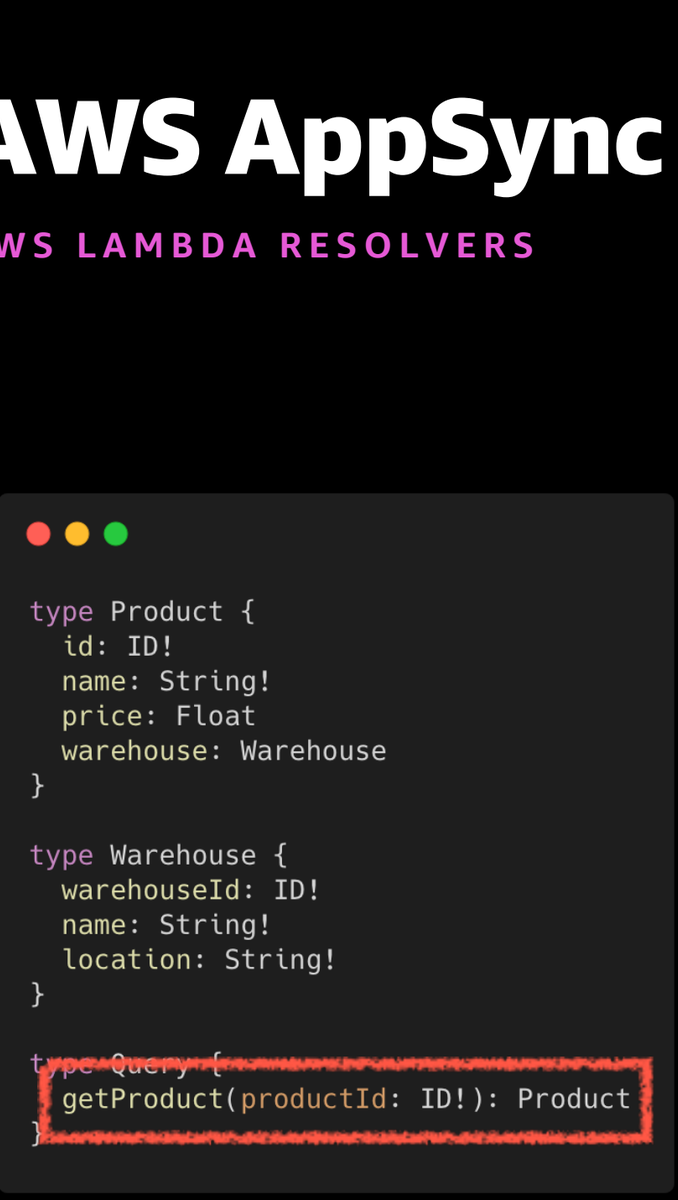
AWS AppSync resolver types

AWS LAMBDA RESOLVERS

```
type Product {
  id: ID!
  name: String!
  price: Float
  warehouse: Warehouse
}

type Warehouse {
  warehouseId: ID!
  name: String!
  location: String!
}

type Query {
  getProduct(productId: ID!): Product
}
```



```
import createProduct from './createProduct'
import listProducts from './listProducts'
import getProduct from './getProduct'

exports.handler = async (event:AppSyncEvent) => {
    switch (event.info.fieldName) {

      case "getProduct":
        return await getProduct(event.arguments.productId)

      case "createProduct":
        return await createProduct(event.arguments.product);

      case "listProducts":
        return await listProducts();

      default:
        return null;
    }
}
```

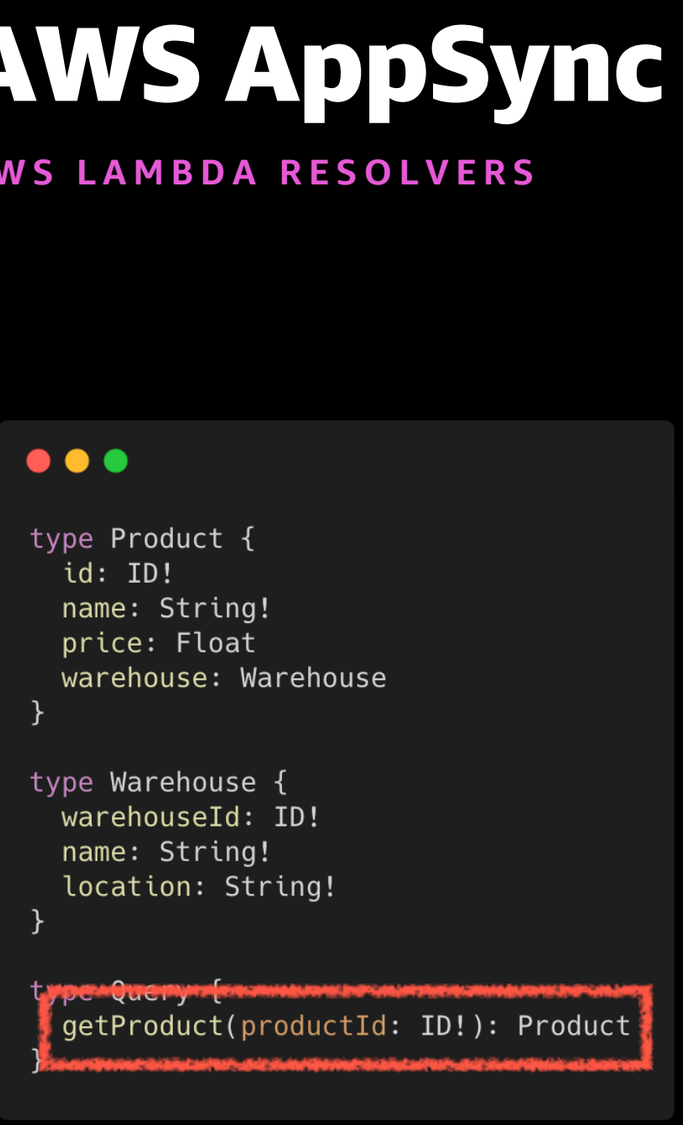
AWS AppSync resolver types

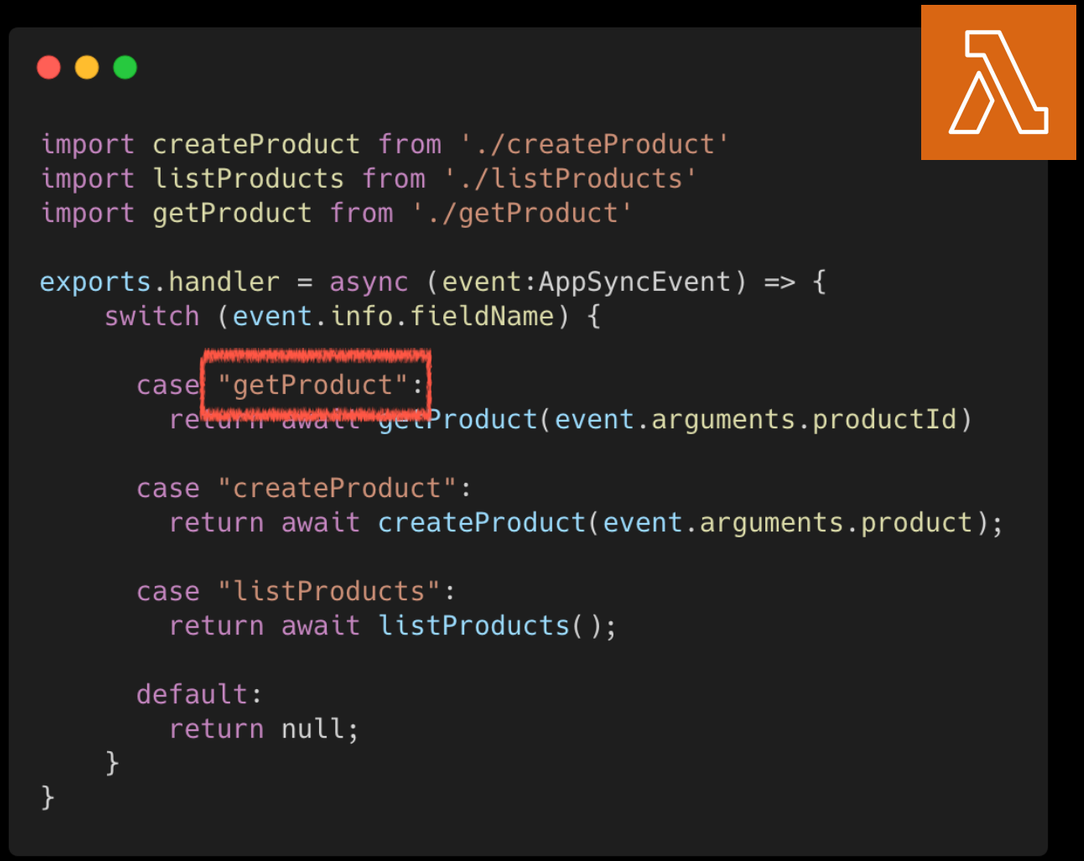
AWS LAMBDA RESOLVERS

```
type Product {
  id: ID!
  name: String!
  price: Float
  warehouse: Warehouse
}

type Warehouse {
  warehouseId: ID!
  name: String!
  location: String!
}

type Query {
  getProduct(productId: ID!): Product
}
```





```
import createProduct from './createProduct'
import listProducts from './listProducts'
import getProduct from './getProduct'

exports.handler = async (event:AppSyncEvent) => {
    switch (event.info.fieldName) {

      case "getProduct":
        return await getProduct(event.arguments.productId)

      case "createProduct":
        return await createProduct(event.arguments.product);

      case "listProducts":
        return await listProducts();

      default:
        return null;
    }
}
```

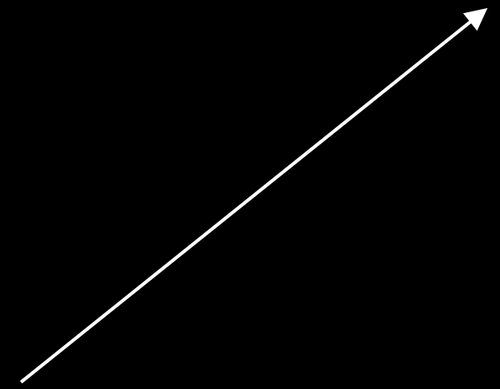
AWS AppSync resolver types

AWS LAMBDA RESOLVERS

```
type Product {
  id: ID!
  name: String!
  price: Float
  warehouse: Warehouse
}

type Warehouse {
  warehouseId: ID!
  name: String!
  location: String!
}

type Query {
  getProduct(productId: ID!): Product
}
```



```
import createProduct from './createProduct'
import listProducts from './listProducts'
import getProduct from './getProduct'

exports.handler = async (event:AppSyncEvent) => {
    switch (event.info.fieldName) {

      case "getProduct":
        return await getProduct(event.arguments.productId)

      case "createProduct":
        return await createProduct(event.arguments.product);

      case "listProducts":
        return await listProducts();

      default:
        return null;
    }
}
```

AWS AppSync resolver types

AWS LAMBDA RESOLVERS

```
import createProduct from './createProduct'
import listProducts from './listProducts'
import getProduct from './getProduct'

exports.handler = async (event:AppSyncEvent) => {
    switch (event.info.fieldName) {

      case "getProduct":
        return await getProduct(event.arguments.productId)

      case "createProduct":
        return await createProduct(event.arguments.product);

      case "listProducts":
        return await listProducts();

      default:
        return null;
    }
}
```

```
{
  "event": {
    "arguments" : { ... },
    "source" : { ... },
    "identity" : { ... },
    "request" : { ... },
    "info": { ... }
  }
}
```

AWS AppSync resolver types

AWS LAMBDA RESOLVERS

```
import createProduct from './createProduct'
import listProducts from './listProducts'
import getProduct from './getProduct'

exports.handler = async (event:AppSyncEvent) => {
    switch (event.info.fieldName) {

      case "getProduct":
        return await getProduct(event.arguments.productId)

      case "createProduct":
        return await createProduct(event.arguments.product);

      case "listProducts":
        return await listProducts();

      default:
        return null;
    }
}
```

```
{
  "event": {
    "arguments" : { ... },
    "source" : { ... },
    "identity" : { ... },
    "request" : { ... },
    "info": { ... }
  }
}
```

AWS AppSync resolver types

AWS LAMBDA RESOLVERS

```
import createProduct from './createProduct'
import listProducts from './listProducts'
import getProduct from './getProduct'

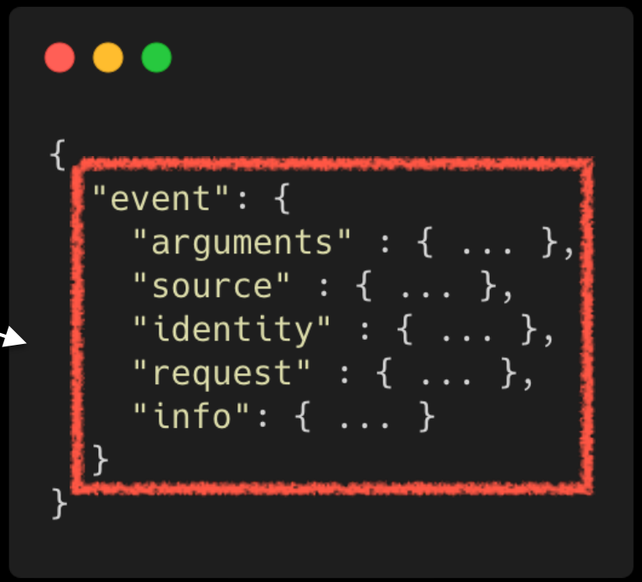
exports.handler = async (event:AppSyncEvent) => {
    switch (event.info.fieldName) {

      case "getProduct":
        return await getProduct(event.arguments.productId)

      case "createProduct":
        return await createProduct(event.arguments.product);

      case "listProducts":
        return await listProducts();

      default:
        return null;
    }
}
```



Single-table design with Amazon DynamoDB and AWS AppSync



AWS re:Invent

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

Single-table design with Amazon DynamoDB and AWS AppSync

Implement using either direct resolvers or AWS Lambda resolvers

Access Pattern	Index	Conditions
Get employee by employee ID	Main table	pk = employee ID
Get direct reports for an employee	GS1	sk = employee ID
Get discontinued products	GS1	sk = "PRODUCT, data = "1"
List all orders of a given product	GS1	sk = product ID
Get the most recent 25 orders	GS1	sk = "ORDER", Limit = 25
Get shippers by name	GS1	sk = shipper name
Get customers by contact name	GS1	sk = contact name
List all products included in an order	Main table	pk = order ID, sk begins with "product"
Get suppliers by country and region	GS1	sk = "SUPPLIER", data begins with "country#region"

Single-table design with Amazon DynamoDB and AWS AppSync

Implement using either direct resolvers or AWS Lambda resolvers

Access Pattern	Index	Conditions
Get employee by employee ID	Main table	pk = employee ID
Get direct reports for an employee	GS1	sk = employee ID
Get discontinued products	GS1	sk = "PRODUCT, data = "1"
List all orders of a given product	GS1	sk = product ID
Get the most recent 25 orders	GS1	sk = "ORDER", Limit = 25
Get shippers by name	GS1	sk = shipper name
Get customers by contact name	GS1	sk = contact name
List all products included in an order	Main table	pk = order ID, sk begins with "product"
Get suppliers by country and region	GS1	sk = "SUPPLIER", data begins with "country#region"

Single-table design with Amazon DynamoDB and AWS AppSync

Implement using either direct resolvers or AWS Lambda resolvers

Access Pattern	Index	Conditions
Get employee by employee ID	Main table	pk = employee ID
Get direct reports for an employee	GS1	sk = employee ID
Get discontinued products	GS1	sk = "PRODUCT, data = "1"
List all orders of a given product	GS1	sk = product ID
Get the most recent 25 orders	GS1	sk = "ORDER", Limit = 25
Get shippers by name	GS1	sk = shipper name
Get customers by contact name	GS1	sk = contact name
List all products included in an order	Main table	pk = order ID, sk begins with "product"
Get suppliers by country and region	GS1	sk = "SUPPLIER", data begins with "country#region"

Single-table design with Amazon DynamoDB and AWS AppSync

Implement using either direct resolvers or AWS Lambda resolvers

Access Pattern	Index	Conditions
Get employee by employee ID	Main table	pk = employee ID
Get direct reports for an employee	GS1	sk = employee ID
Get discontinued products	GS1	sk = "PRODUCT, data = "1"
List all orders of a given product	GS1	sk = product ID
Get the most recent 25 orders	GS1	sk = "ORDER", Limit = 25
Get shippers by name	GS1	sk = shipper name
Get customers by contact name	GS1	sk = contact name
List all products included in an order	Main table	pk = order ID, sk begins with "product"
Get suppliers by country and region	GS1	sk = "SUPPLIER", data begins with "country#region"

Deployment options



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

Deployment options

1. AWS Amplify
2. AWS CDK
3. Serverless framework
4. AWS CloudFormation



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

Deployment options

AWS AMPLIFY



- GraphQL code generation
- GraphQL Transform library
- Template generation
- Amplify DataStore

Deployment options

AWS AMPLIFY



```
~ amplify add api
```

Deployment options

AWS CDK

```
import * as appsync from '@aws-cdk/aws-appsync';

const api = new appsync.GraphqlApi(this, 'Api', {
  name: 'cdk-notes-appsync-api',
  schema: appsync.Schema.fromAsset('graphql/schema.graphql'),
  authorizationConfig: {
    defaultAuthorization: {
      authorizationType: appsync.AuthorizationType.API_KEY,
    },
  }
});
```

Deployment options

AWS CDK

```
import * as appsync from '@aws-cdk/aws-appsync';

const api = new appsync.GraphqlApi(this, 'Api', {
  name: 'cdk-notes-appsync-api',
  schema: appsync.Schema.fromAsset('graphql/schema.graphql'),
  authorizationConfig: {
    defaultAuthorization: {
      authorizationType: appsync.AuthorizationType.API_KEY,
    },
  }
});
```

Deployment options

AWS CDK

```
import * as appsync from '@aws-cdk/aws-appsync';

const api = new appsync.GraphqlApi(this, 'Api', {
  name: 'cdk-notes-appsync-api',
  schema: appsync.Schema.fromAsset('graphql/schema.graphql'),
  authorizationConfig: {
    defaultAuthorization: {
      authorizationType: appsync.AuthorizationType.API_KEY,
    },
  }
});
```

Deployment options

AWS CDK

graphql/schema.graphql

```
type Note {
  id: ID!
  name: String!
  completed: Boolean
}

input NoteInput {
  id: ID!
  name: String!
  completed: Boolean
}

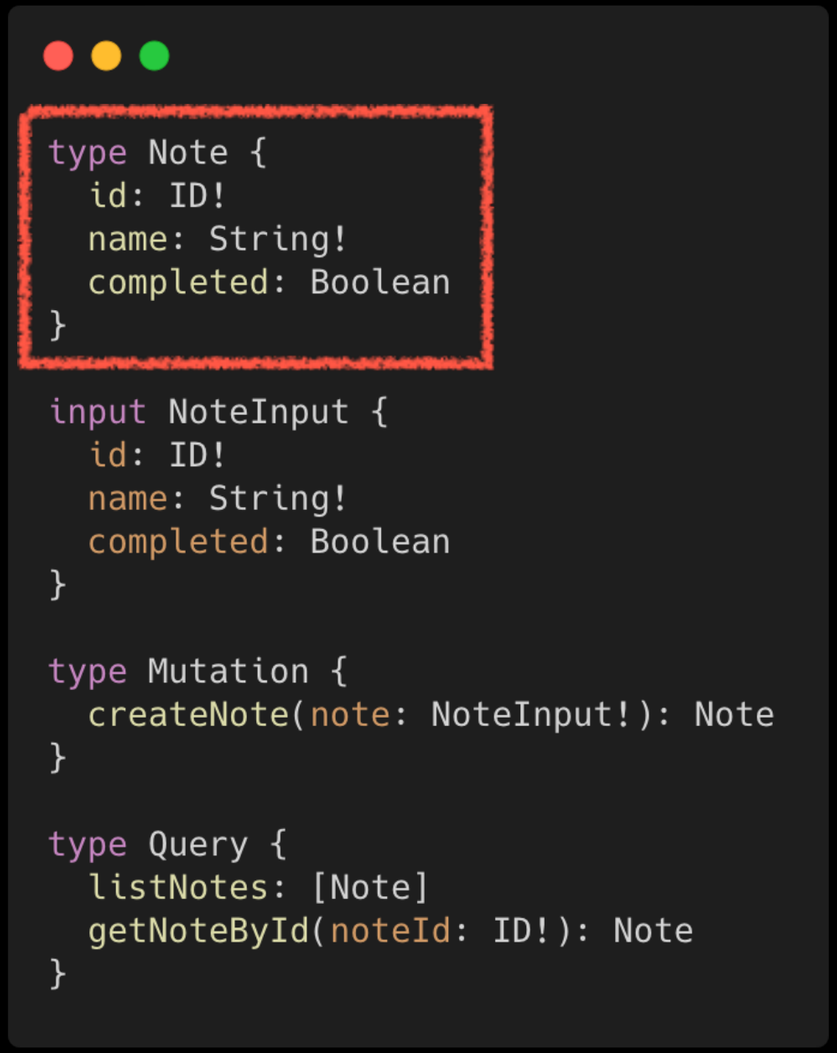
type Mutation {
  createNote(note: NoteInput!): Note
}

type Query {
  listNotes: [Note]
  getNoteById(noteId: ID!): Note
}
```

Deployment options

AWS CDK

graphql/schema.graphql



```
type Note {
  id: ID!
  name: String!
  completed: Boolean
}

input NoteInput {
  id: ID!
  name: String!
  completed: Boolean
}

type Mutation {
  createNote(note: NoteInput!): Note
}

type Query {
  listNotes: [Note]
  getNoteById(noteId: ID!): Note
}
```

Deployment options

AWS CDK

graphql/schema.graphql

```
type Note {
  id: ID!
  name: String!
  completed: Boolean
}

input NoteInput {
  id: ID!
  name: String!
  completed: Boolean
}

type Mutation {
  createNote(note: NoteInput!): Note
}

type Query {
  listNotes: [Note]
  getNoteById(noteId: ID!): Note
}
```

Deployment options

AWS CDK

graphql/schema.graphql

```
type Note {
  id: ID!
  name: String!
  completed: Boolean
}

input NoteInput {
  id: ID!
  name: String!
  completed: Boolean
}

type Mutation {
  createNote(note: NoteInput!): Note
}

type Query {
  listNotes: [Note]
  getNoteById(noteId: ID!): Note
}
```

Deployment options

AWS CDK

```
const notesLambda = new lambda.Function(this, 'AppSyncNotesHandler', {
  runtime: lambda.Runtime.NODEJS_12_X,
  handler: 'appsync-ds-main.handler',
  code: lambda.Code.fromAsset('lambda-fns'),
});

// set the new Lambda function as a data source for the AppSync API
const lambdaDs = api.addLambdaDataSource('lambdaDatasource', notesLambda);
```

Deployment options

AWS CDK

```
const notesLambda = new lambda.Function(this, 'AppSyncNotesHandler', {
  runtime: lambda.Runtime.NODEJS_12_X,
  handler: 'appsync-ds-main.handler',
  code: lambda.Code.fromAsset('lambda-fns'),
});

// set the new Lambda function as a data source for the AppSync API
const lambdaDs = api.addLambdaDataSource('lambdaDatasource', notesLambda);
```

Deployment options

AWS CDK

```
const notesLambda = new lambda.Function(this, 'AppSyncNotesHandler', {
  runtime: lambda.Runtime.NODEJS_12_X,
  handler: 'appsync-ds-main.handler',
  code: lambda.Code.fromAsset('lambda-fns'),
});

// set the new Lambda function as a data source for the AppSync API
const lambdaDs = api.addLambdaDataSource('lambdaDatasource', notesLambda);
```

Deployment options

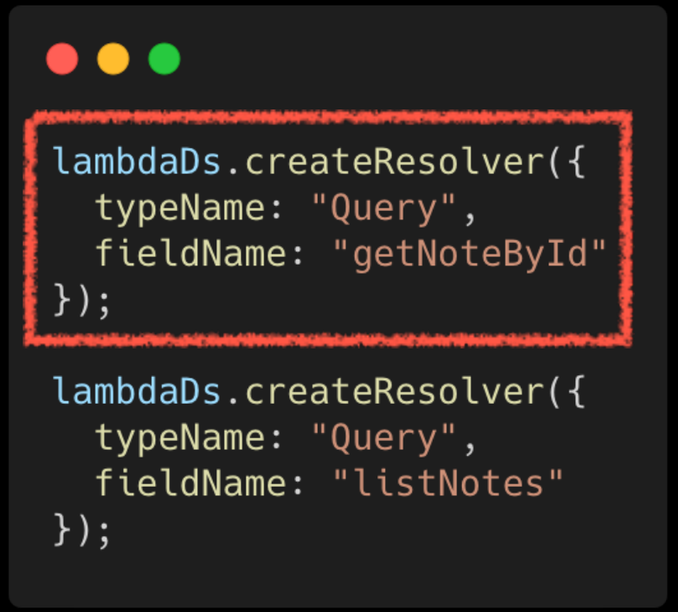
AWS CDK

```
lambdaDs.createResolver({
  typeName: "Query",
  fieldName: "getNoteById"
});

lambdaDs.createResolver({
  typeName: "Query",
  fieldName: "listNotes"
});
```

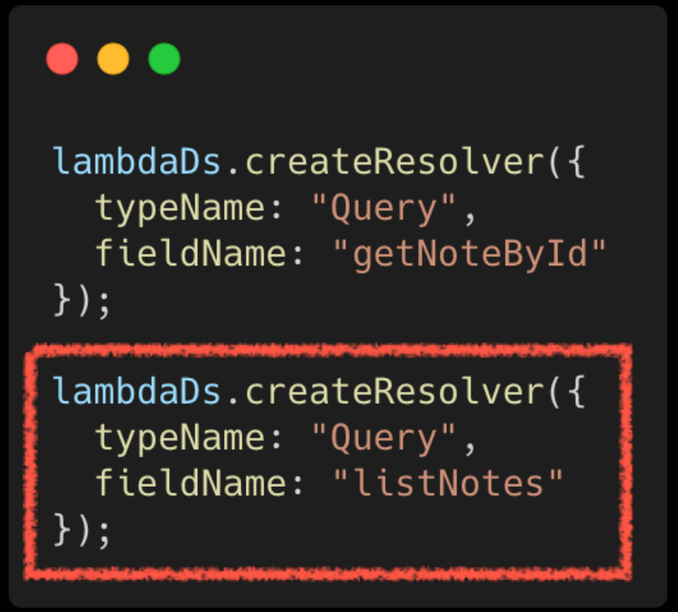
Deployment options

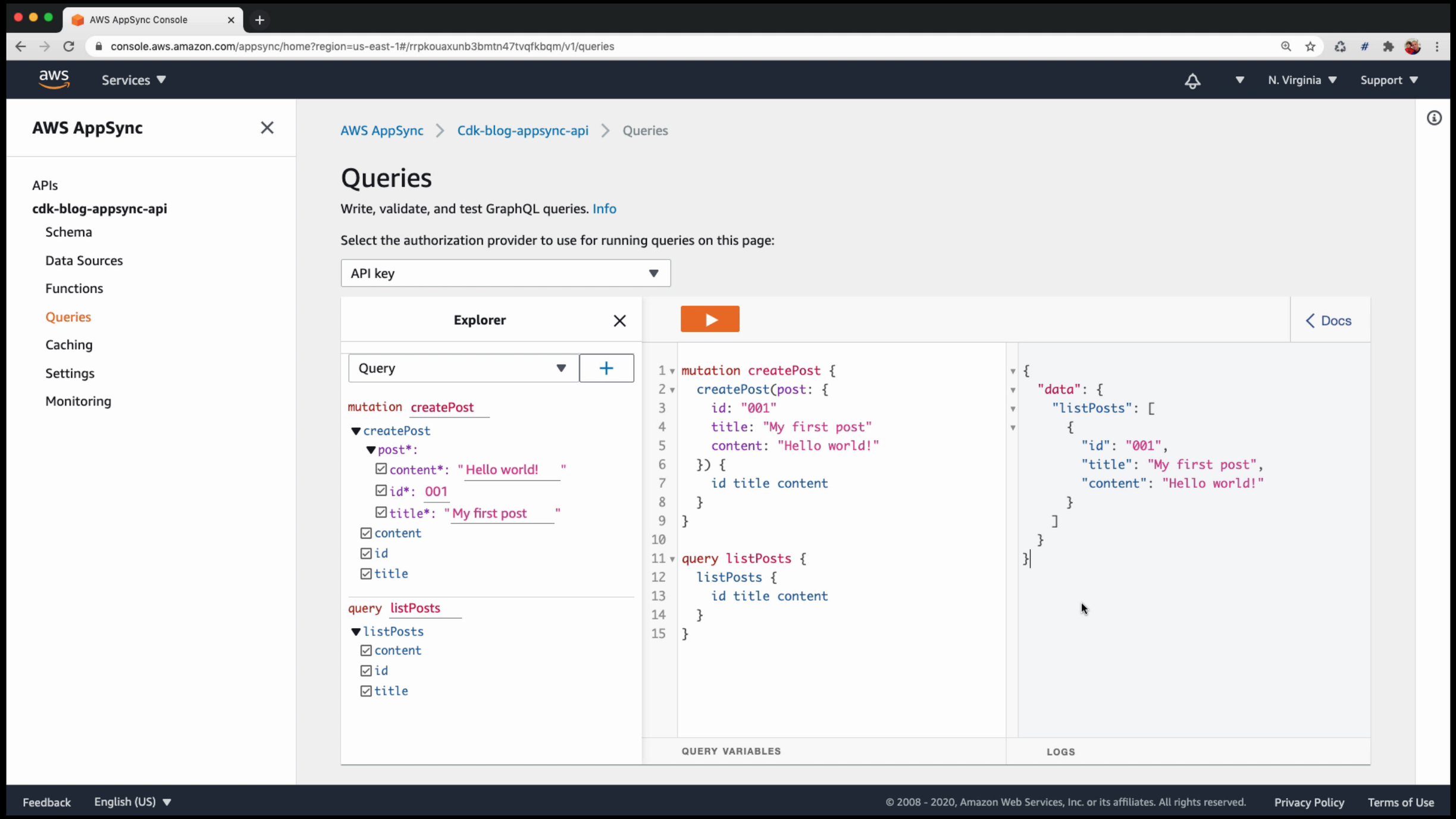
AWS CDK



Deployment options

AWS CDK




AWS AppSync Console
console.aws.amazon.com/appsync/home?region=us-east-1#/rrpkouaxunb3bmtn47tvqfkbqm/v1/queries
aws
Services
N. Virginia
Support
AWS AppSync
APIs
cdk-blog-appsync-api
Schema
Data Sources
Functions
Queries
Caching
Settings
Monitoring
AWS AppSync > Cdk-blog-appsync-api > Queries
Queries
Write, validate, and test GraphQL queries. Info
Select the authorization provider to use for running queries on this page:
API key
Explorer
Query
mutation createPost
createPost
post*:
content*: "Hello world! "
id*: 001
title*: "My first post "
content
id
title
query listPosts
listPosts
content
id
title
Docs
mutation createPost {
createPost(post: {
id: "001"
title: "My first post"
content: "Hello world!"
}) {
id title content
}
}
query listPosts {
listPosts {
id title content
}
}
{
"data": {
"listPosts": [
{
"id": "001",
"title": "My first post",
"content": "Hello world!"
}
]
}
}
QUERY VARIABLES
LOGS
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

AWS AppSync resources

Website
aws.amazon.com/appsync

Docs
docs.aws.amazon.com/appsync

Github
github.com/aws/aws-appsync-community

Blog
aws.amazon.com/appsync/blog/

More resources
aws.amazon.com/appsync/resources/

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.