
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

FWM316

Accelerate GraphQL API app development & collaboration with AWS AppSync

Ed Lima

Sr. Product Manager
AWS

Brice Pellé

Principal Product Manager
AWS

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

- API evolution: Why GraphQL?
- Accelerate multi-team development with GraphQL
- Accelerate development with JavaScript resolvers

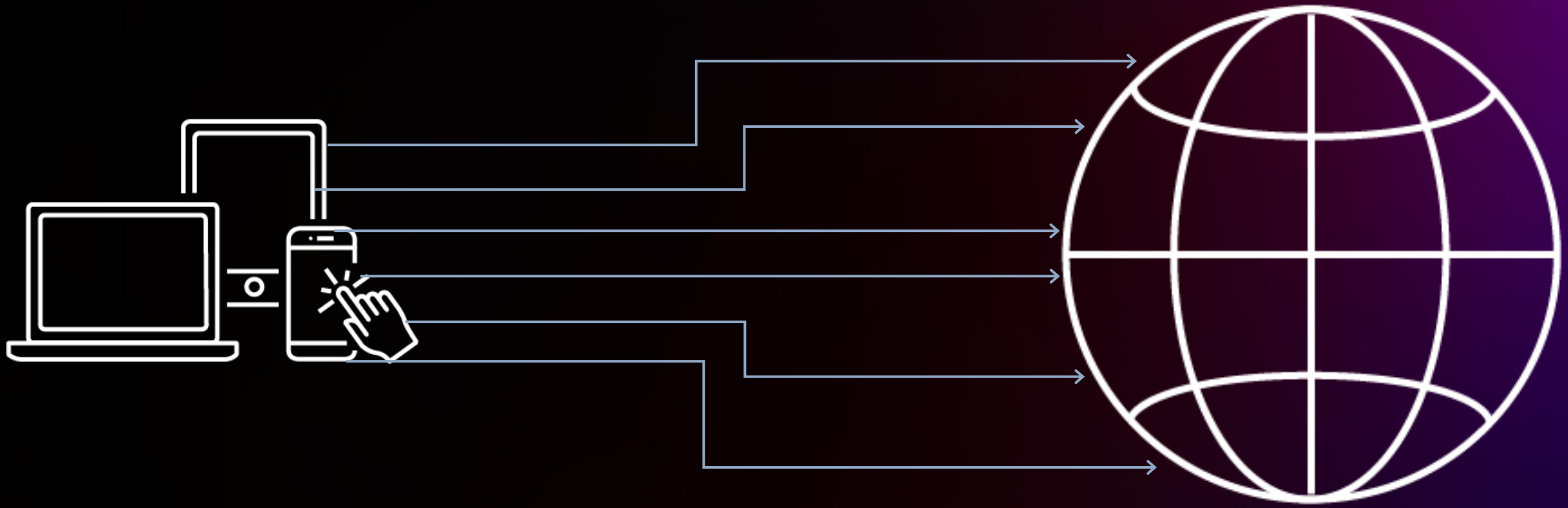


© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

API evolution: Why GraphQL?

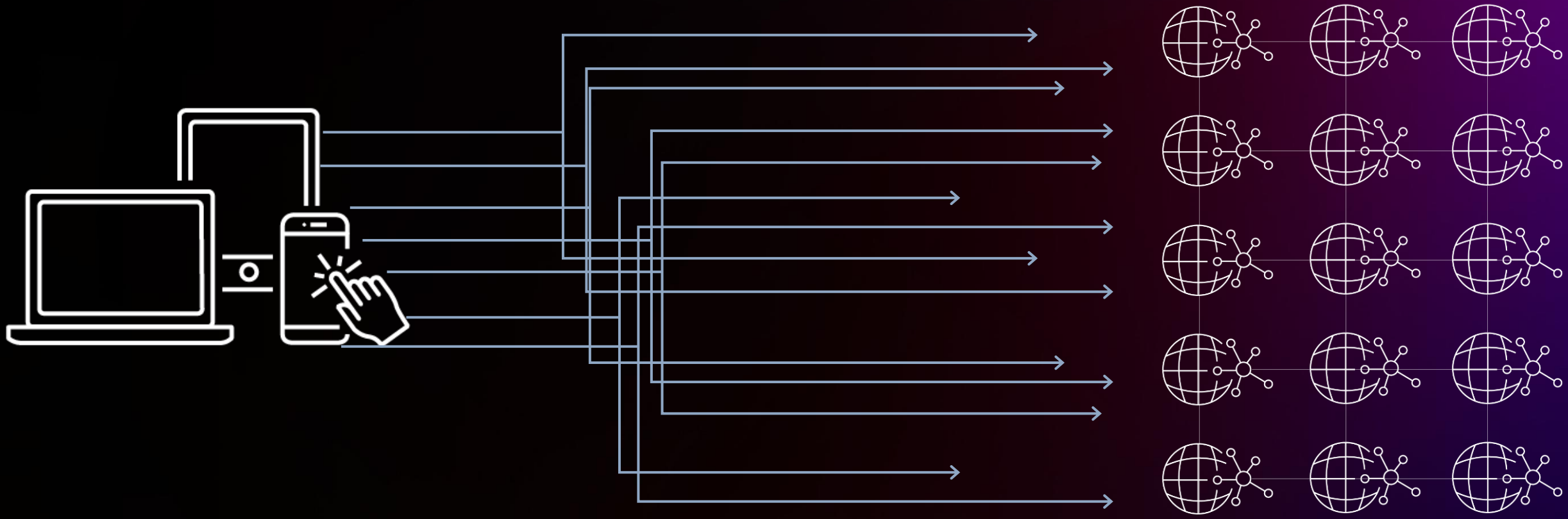
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Monolith



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Microservices: Direct access



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Microservices: Aggregation layer



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

GraphQL is a query language for APIs
and a runtime for fulfilling those queries
with your existing data



Describe your data

```
type Project {
  name: String
  tagline: String
  contributors: [User]
}
```

Ask for what you want

```
{
  project(name: "GraphQL") {
    tagline
  }
}
```

Get predictable results

```
{
  "project": {
    "tagline": "A query language for APIs"
  }
}
```

Strongly typed system

All the data you need in a single request

Real-time notifications

aws

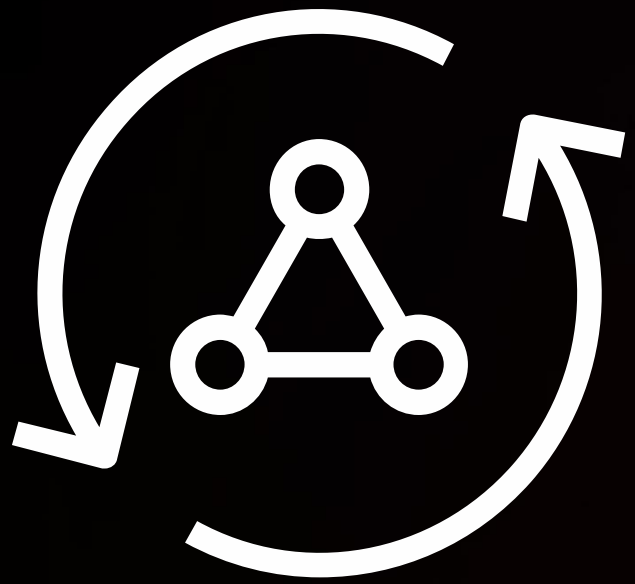
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The API market will grow to **$27 billion** by **2028**, with **more than 50%** of enterprises **using GraphQL in production by 2025**, up from less than 10% in 2021*

*Source: Gartner, “Predicts 2022: APIs demand improved security and management,” https://www.gartner.com/en/documents/4009103

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

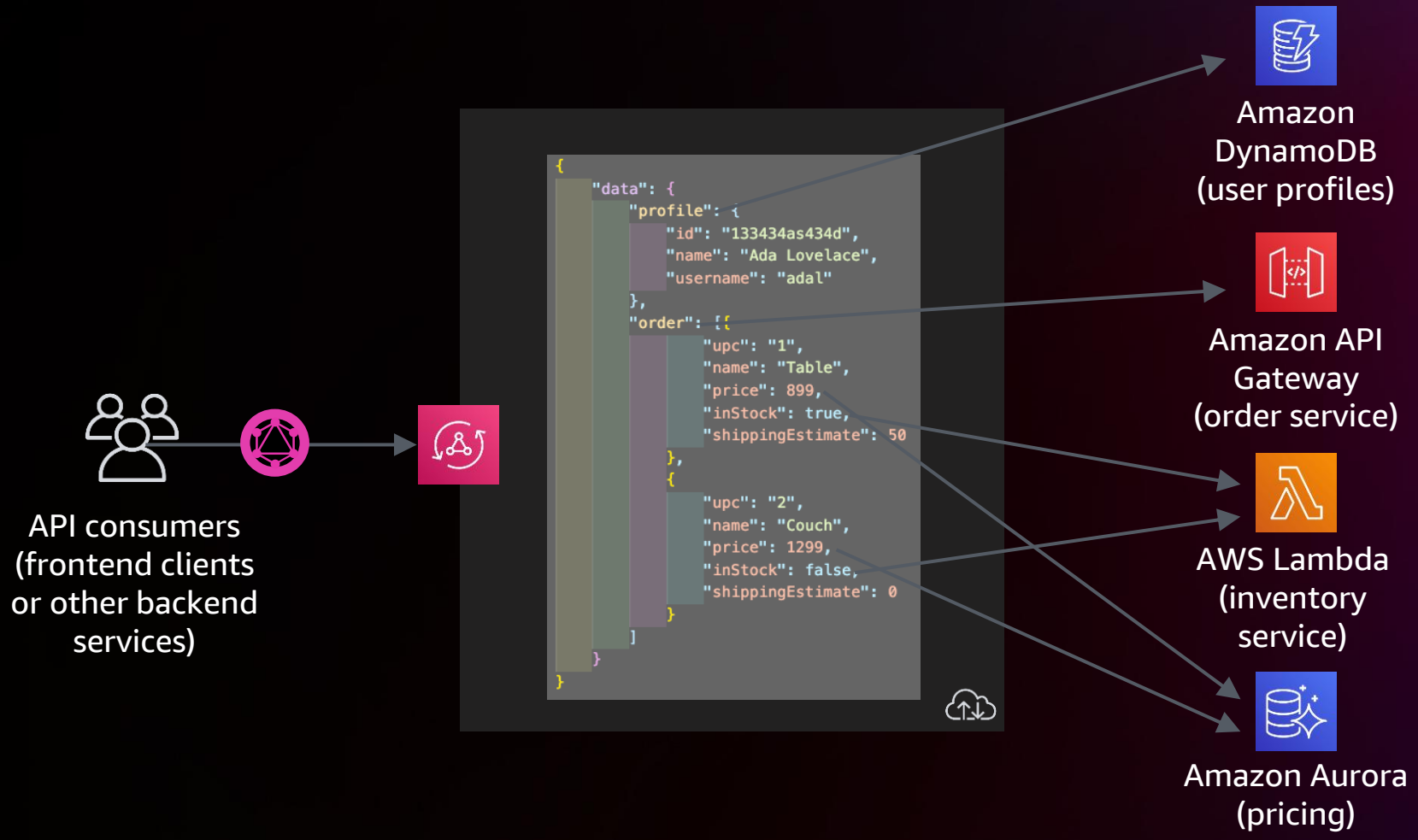


AWS AppSync

Build scalable applications on a range of data sources, including those requiring real-time updates and offline data access with **no servers to manage**

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

GraphQL as aggregation layer

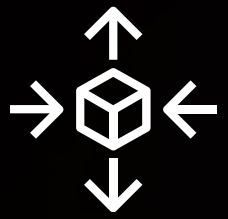


Single endpoint | Single request | Multiple datasources

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS AppSync

PRIMARY USE CASES



Serverless GraphQL

- ✓ One endpoint for any data, anywhere
- ✓ Connect apps to many data sources and microservices



Real-time apps

- ✓ Built in support for pub/sub architectures
- ✓ Example: fan engagement, location-aware notifications



Offline apps

- ✓ Built-in support for off-line synchronization
- ✓ Example: Field service apps, mobile apps, IoT devices

aws

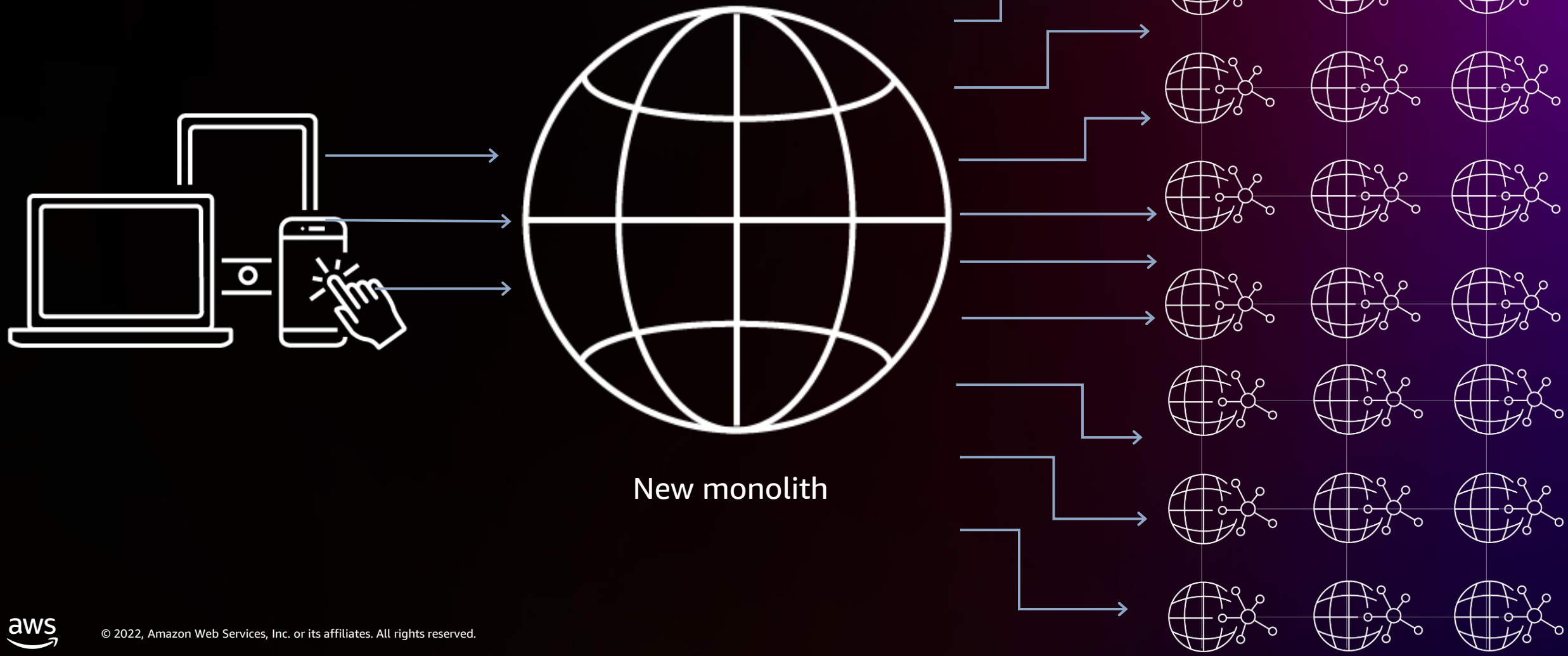
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Microservices: Aggregation layer



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Microservices: Aggregation layer



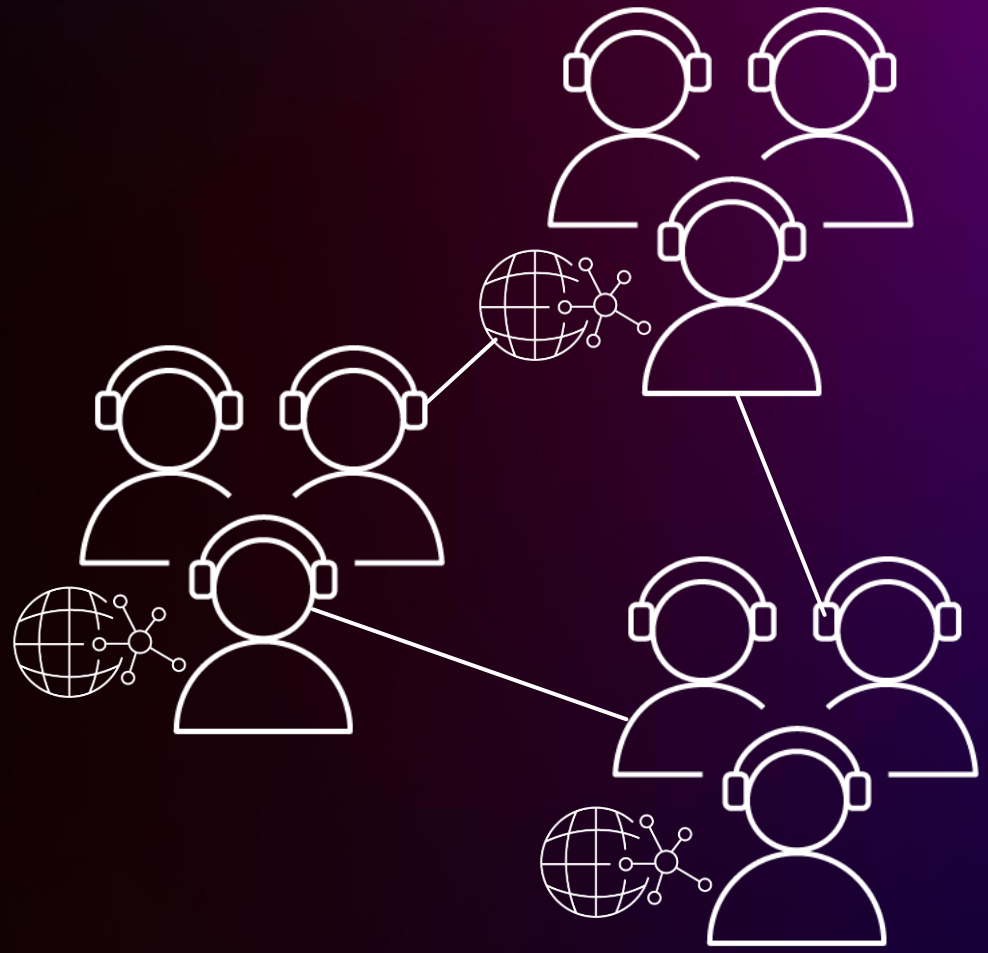
aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Accelerate multi-team development with GraphQL

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Requirements in a multi-service environment

- Velocity
- Isolation
- Ownership
- Rapid iteration
- Simplicity



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

GraphQL schema composition



Runtime composition

- Central gateway with secondary GraphQL APIs as part of query execution path
- More network hops can increase latency
- No support for real-time subscriptions

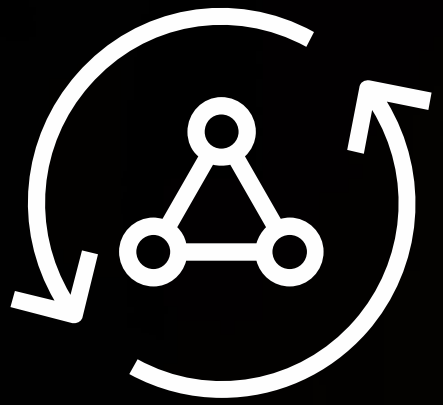
Build-time composition

- Single API composed at build time, no other APIs involved in the execution path (better performance)
- Complex deployment pipelines
- As the result is a single API, real-time subscriptions are fully supported

Single GraphQL endpoint for API consumers

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.



AWS AppSync merged APIs



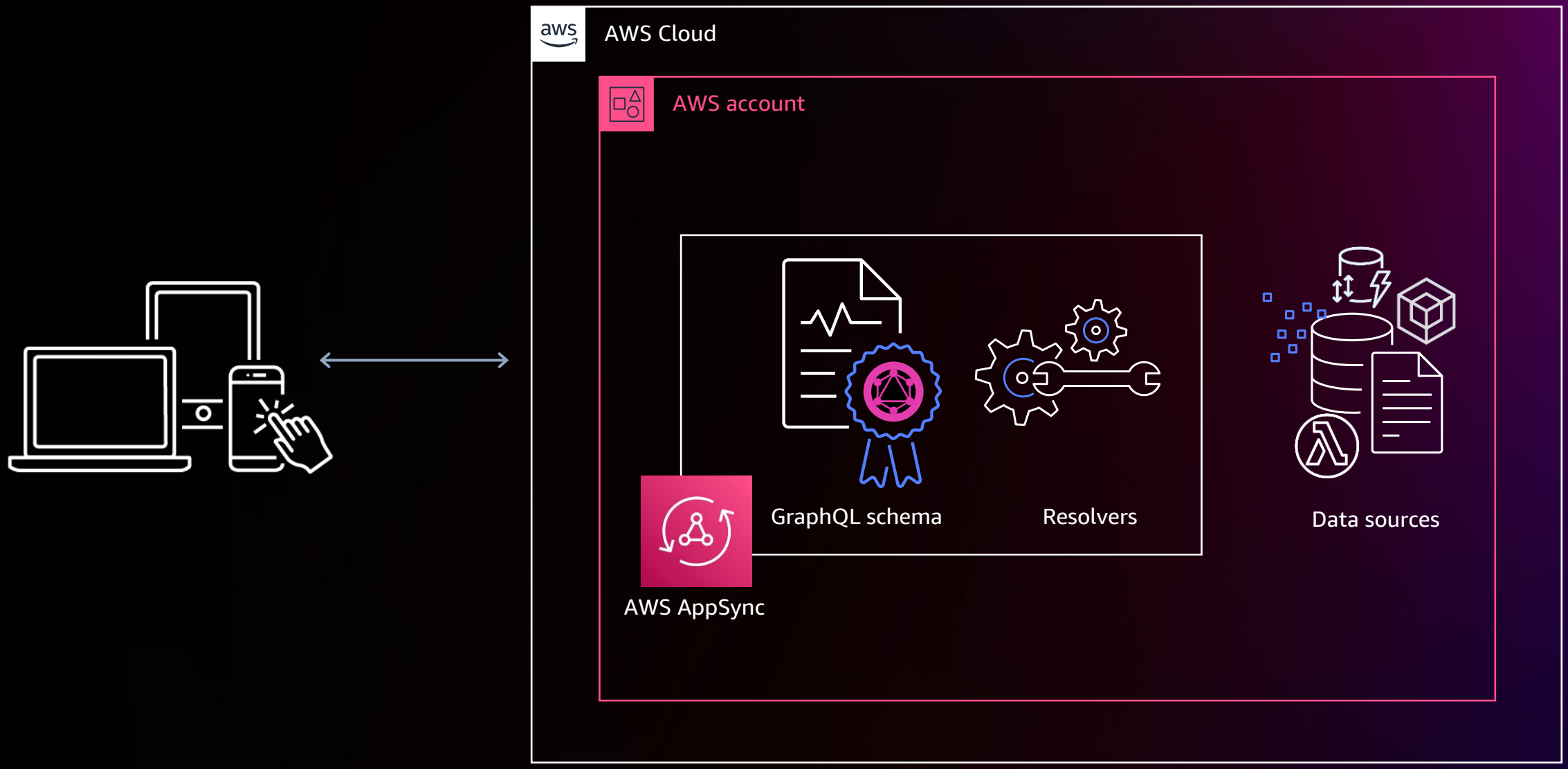
https://github.com/aws/aws-appsync-community/issues/226

Build-time GraphQL schema composition
without the complexity

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS AppSync APIs



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

```
type Order {
  id: ID!
  total: Float
  inStock: Boolean
  date: AWSDate
  items: [Product]
}

Type Query {
   getOrder(id: ID!): Order
}
```



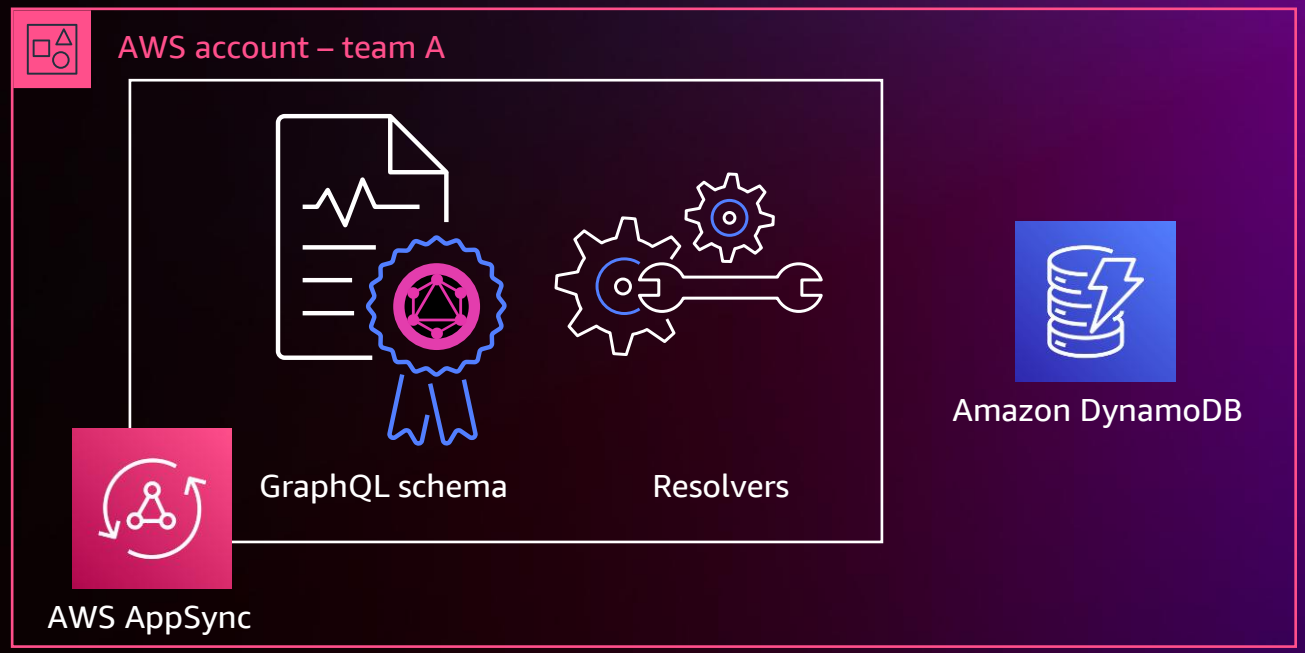
Schema A

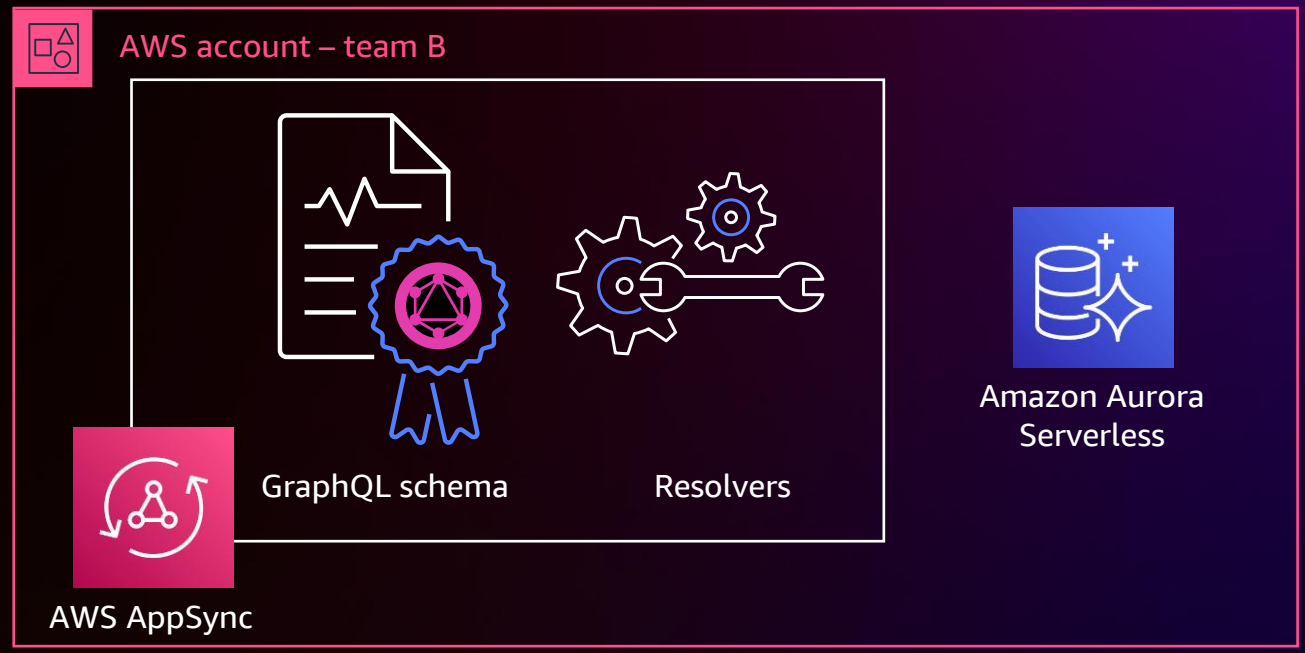
```
type Product {
  id: ID!
  name: String
  description: String
  price: Float
}

Type Query {
   getProduct(id: ID!): Product
}
```



Schema B





aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.



AWS Console/AWS CLI/AWS SDK

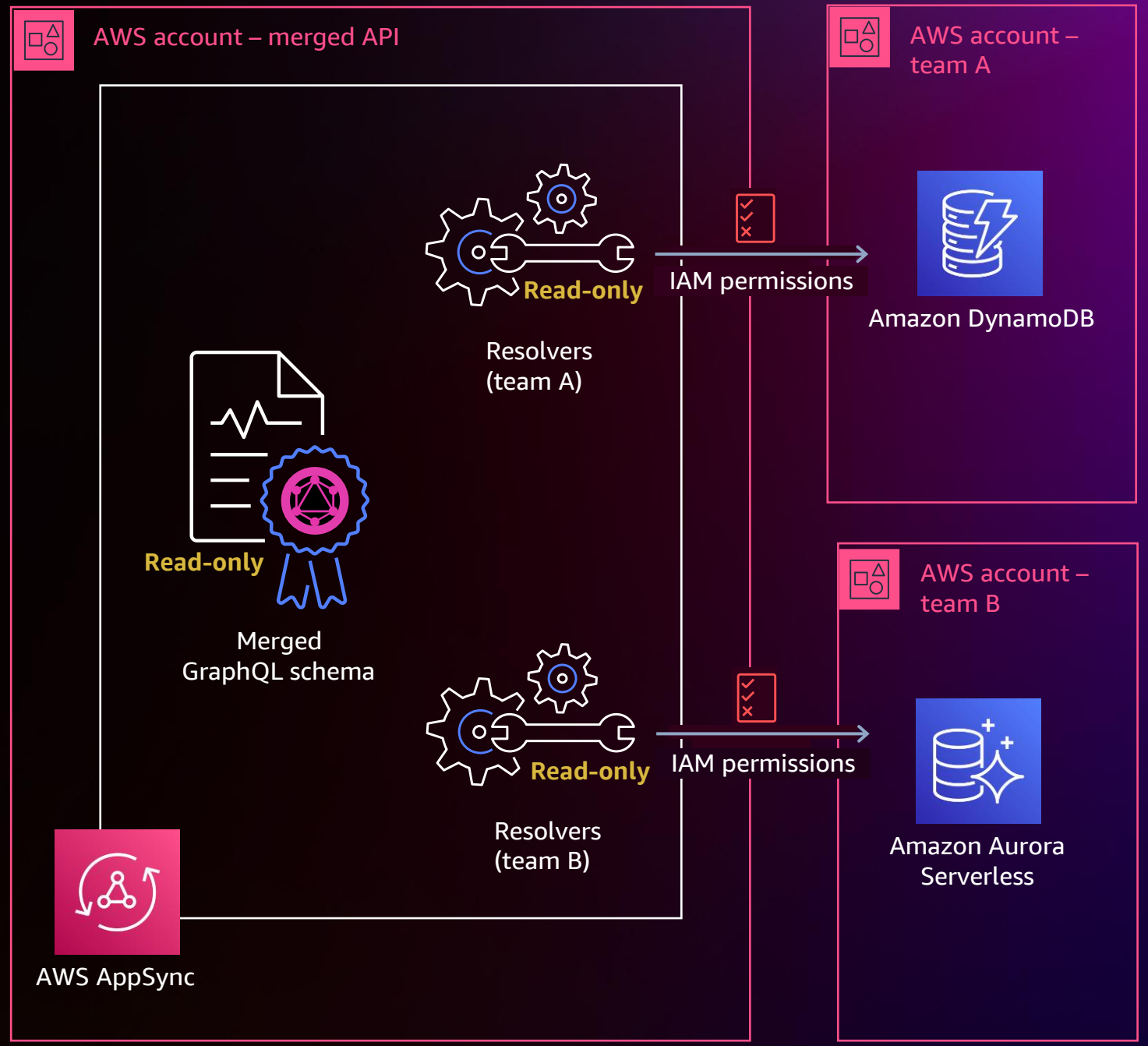
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

```
type Order {
  id: ID!
  total: Float
  inStock: Boolean
  date: AWSDate
  items: [Product]
}

type Product {
  id: ID!
  name: String
  description: String
  price: Float
}


Type Query {
   getOrder(id: ID!): Order
   getProduct(id: ID!): Product
}
```

Merged schema



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Resolving conflicts with directives

- **@canonical:** if two or more source APIs have the same **GraphQL type or field**, a resource with this directive **takes priority** when merging the schemas

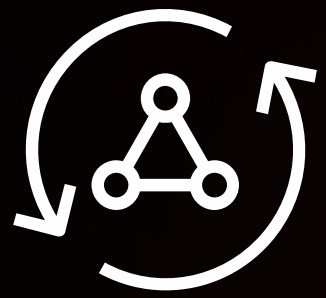
- **@hidden:** hide specific types or operations in the merged API so only source API clients can access specific typed data; with this directive attached, **types or fields are not merged** to the merged API target

- **@renamed:** there are use cases where different APIs have the same type or field name and they all need to be available in the merged schema; with this directive a type in a source API can be **renamed in the merged API**

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS AppSync merged APIs

- **Merged API: single AWS AppSync API resulting from a merge operation** where multiple GraphQL schemas, resolvers and data sources from Source APIs are imported and composed; **no extra costs**
- **Source API: single AWS AppSync API that owns** specific **GraphQL types**, **fields**, **resolvers**, and **data sources** that are later ingested/imported to a merged API
- Source APIs **are managed by different teams** and a merged API is where all **teams can collaborate safely and in isolation**
- Teams **cannot edit** GraphQL resources **directly** in the **merged API**



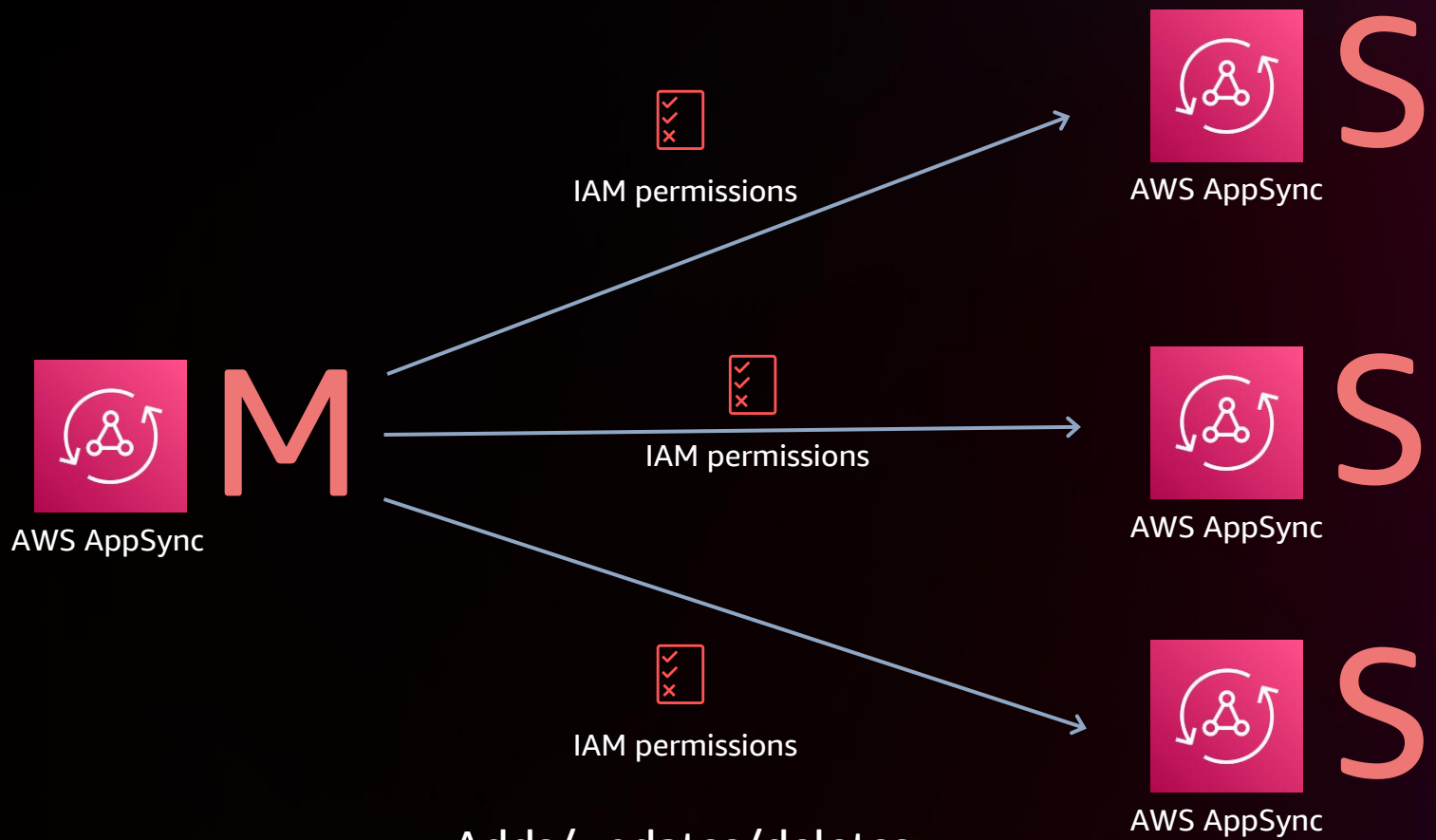




aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS AppSync merged APIs: Getting started



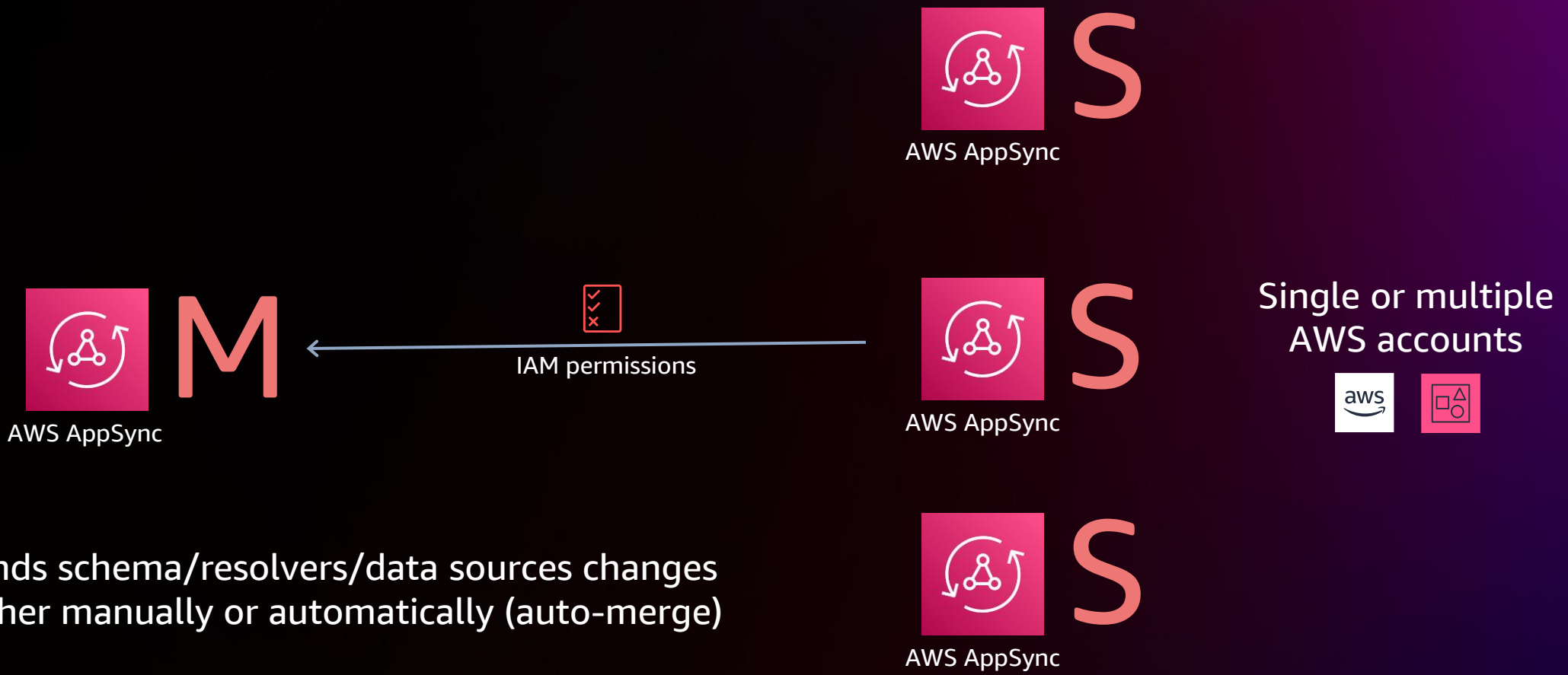
Single or multiple AWS accounts

Adds/updates/deletes

(Invite/uninvite)

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS AppSync merged APIs: Getting started



Sends schema/resolvers/data sources changes either manually or automatically (auto-merge)

Initial merge = accepts invitation

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS AppSync merged APIs



AWS AppSync
(merged API)

Build-time GraphQL
schema composition



Teams
boundaries

Multiple
schemas/
data models

Single GraphQL schema

AWS AppSync
(source APIs)

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS AppSync merged APIs



AWS AppSync
(merged API)

Build-time GraphQL
schema composition



Teams
boundaries

Multiple
schemas/
data models

Single GraphQL schema

AWS AppSync
(source APIs)

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS AppSync merged APIs: Source

Source API

- **Source of truth**, with full ownership of its own GraphQL schema and API resources
- Resources can be merged automatically whenever there's a change – **auto-merge** or **manually** (such as, deployment pipeline)
- **Full isolation:** no other teams can edit or change GraphQL types, fields, operations, or resolvers in a source API
- Source APIs are **not part of** the merged API **GraphQL query execution path** and can be accessed **independently** by their own API consumers

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS AppSync merged APIs: Target

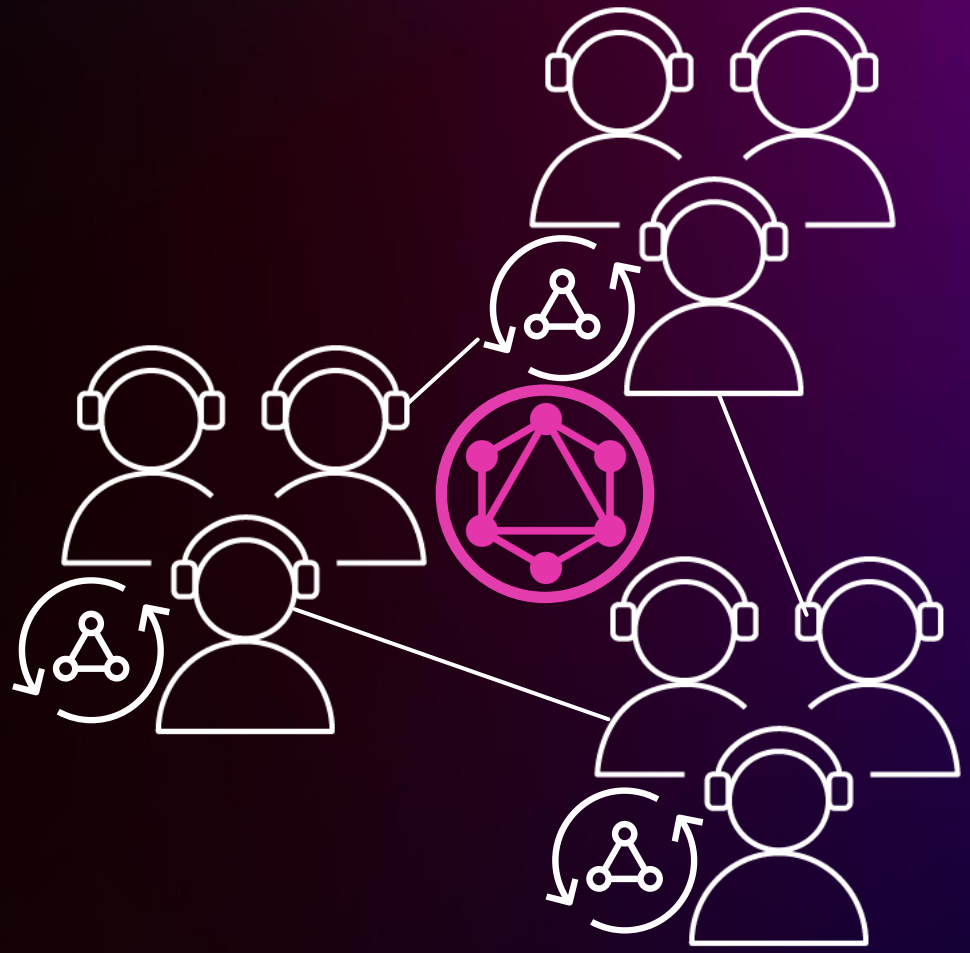
Merged API

- Automatically **composes all schemas** from source APIs; AWS AppSync API **with read-only** GraphQL schema, resolvers, and data sources (pointers)
- No need to setup complex deployment pipelines with build/merge logic; **AWS AppSync takes care of everything**
- **Authorization** – such as, merged API accessed by external clients but source API accessed by internal clients with different auth modes – **caching**, **logging**, **AWS WAF**, and **AWS X-Ray** integration are fully configurable to provide extra flexibility
- **Direct access** to data sources **secured by IAM**, no extra network hops or more API aggregation layers with **improved performance**
- **Full support for real-time subscriptions** powered by AWS AppSync **Serverless WebSockets**

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Collaboration in a multi-service environment with AWS AppSync merged APIs

- Velocity
- Isolation
- Ownership
- Rapid iteration
- Simplicity



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

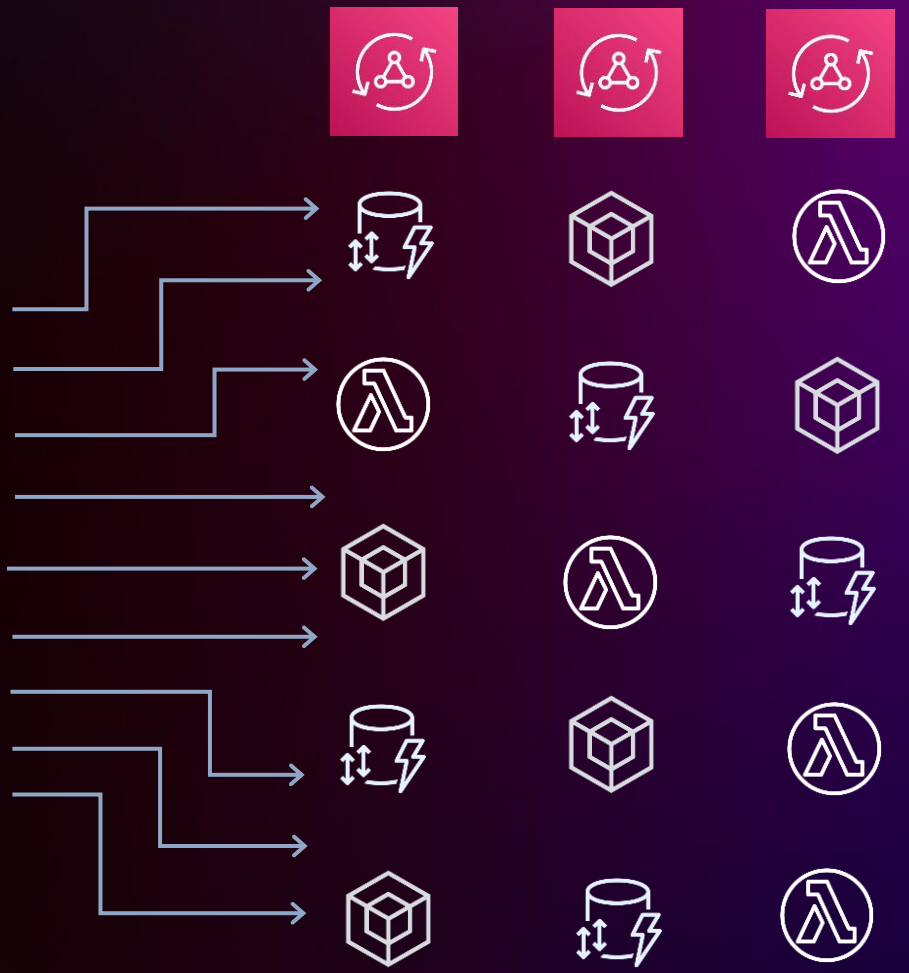
Next API evolution: AWS AppSync merged APIs



API consumers



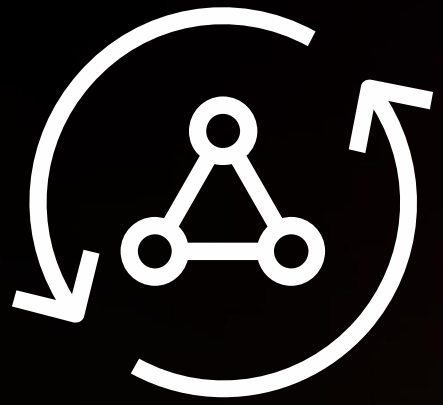
AWS AppSync
(merged API)



GraphQL
microservices
data

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.



AWS AppSync merged APIs



Build-time GraphQL schema composition
without the complexity

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Accelerate development with JavaScript resolvers

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS AppSync resolvers today

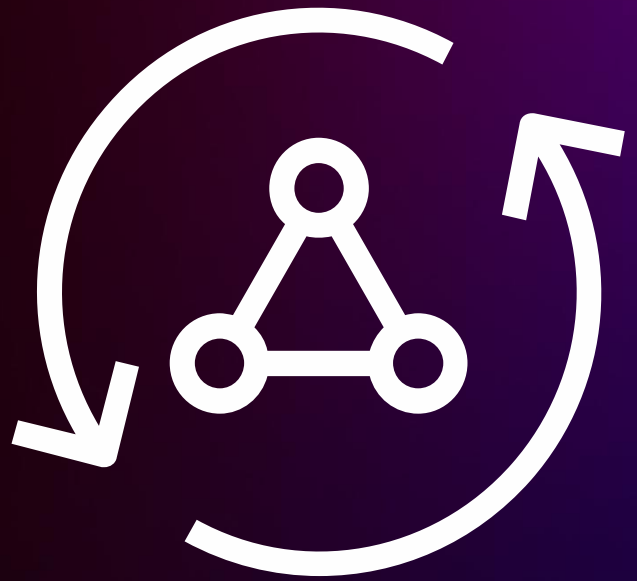
The easiest way to connect applications to your data

- Built-in integration with AWS DynamoDB, AWS Lambda
- Direct integration with AWS services via HTTP resolvers
- With AWS Lambda: easy to batch and proxy to other APIs
- Use AWS AppSync utilities

No need to worry about

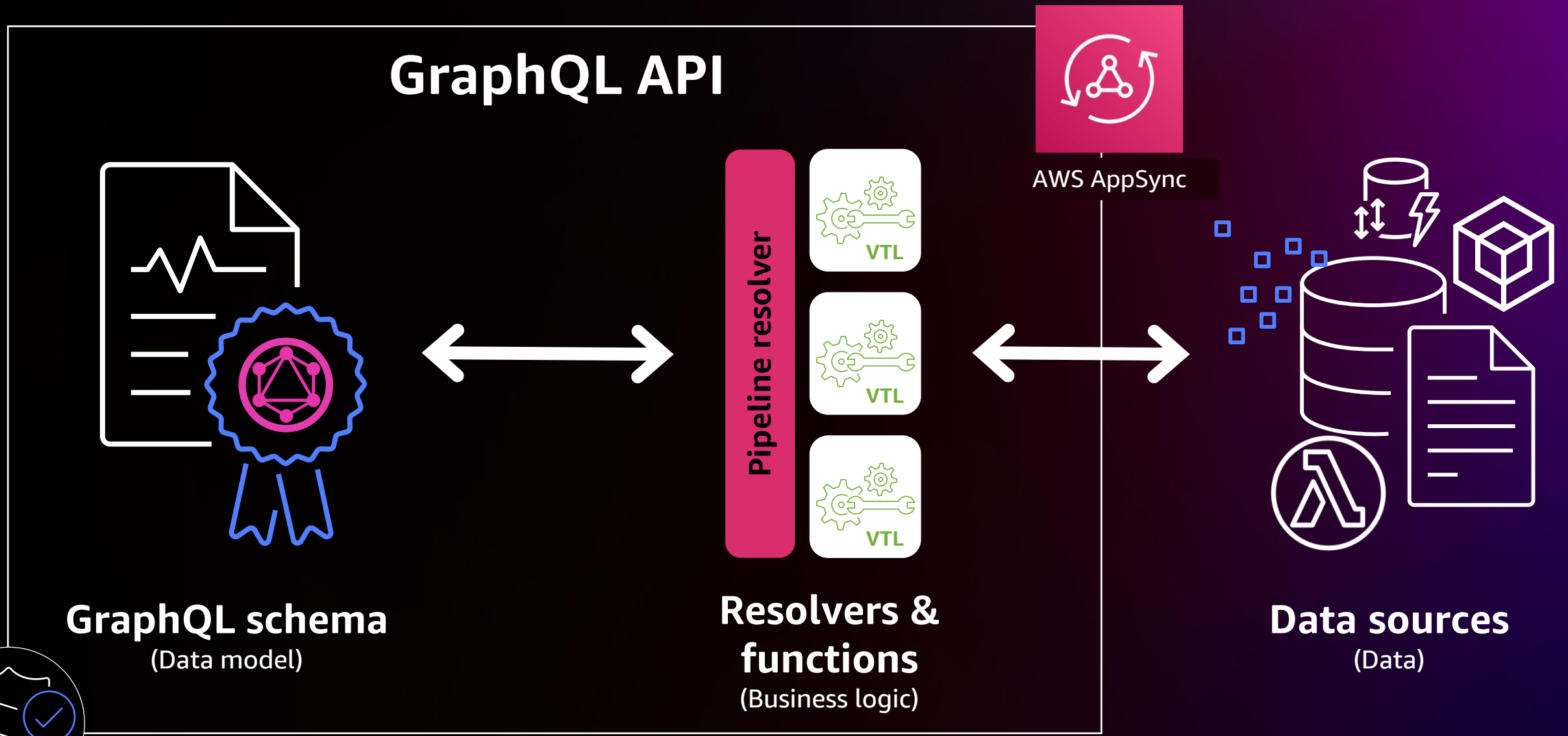
- Security, scale, performance
- Complex functionality (such as, WebSocket implementation)

Focus on your business logic and business objectives



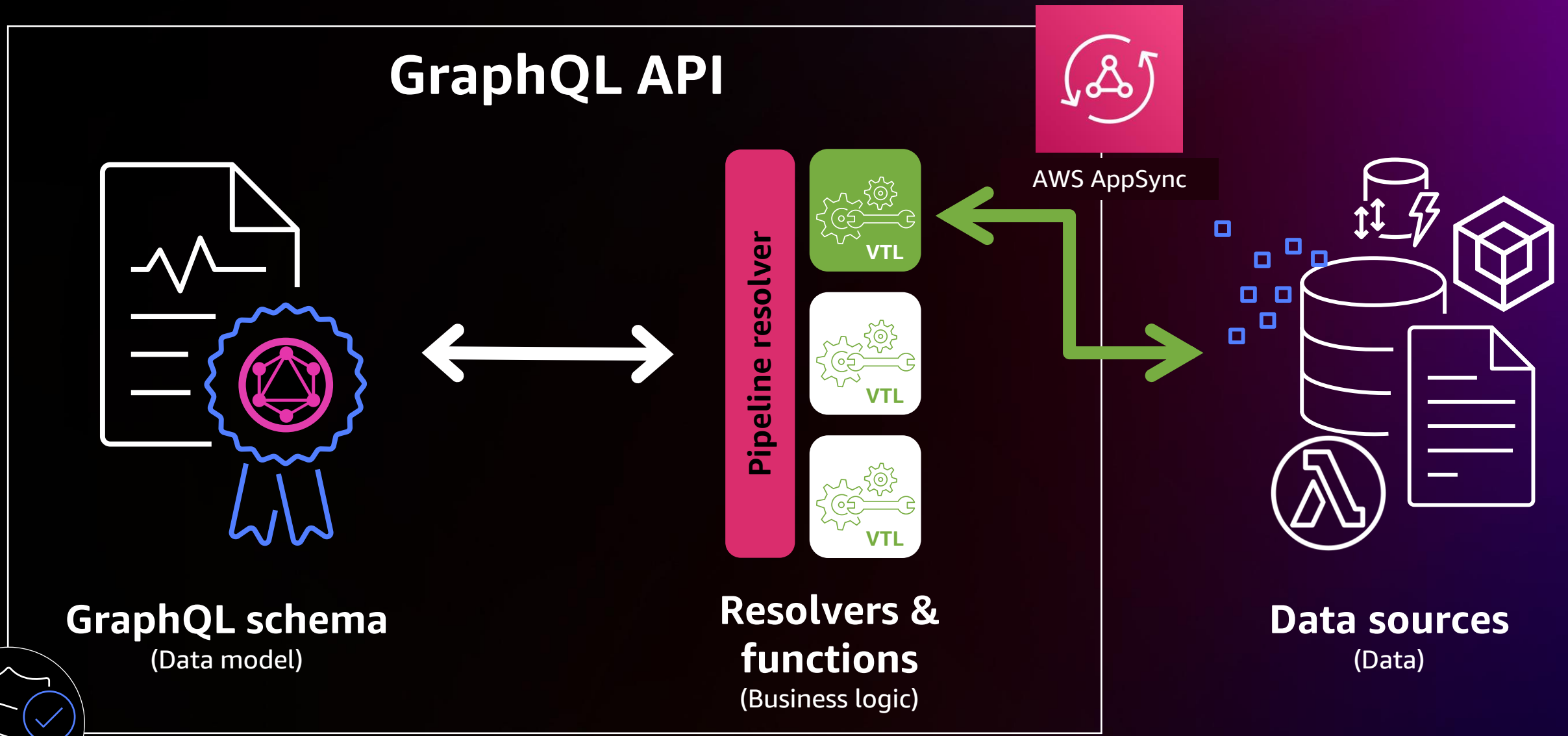
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS AppSync: How it works



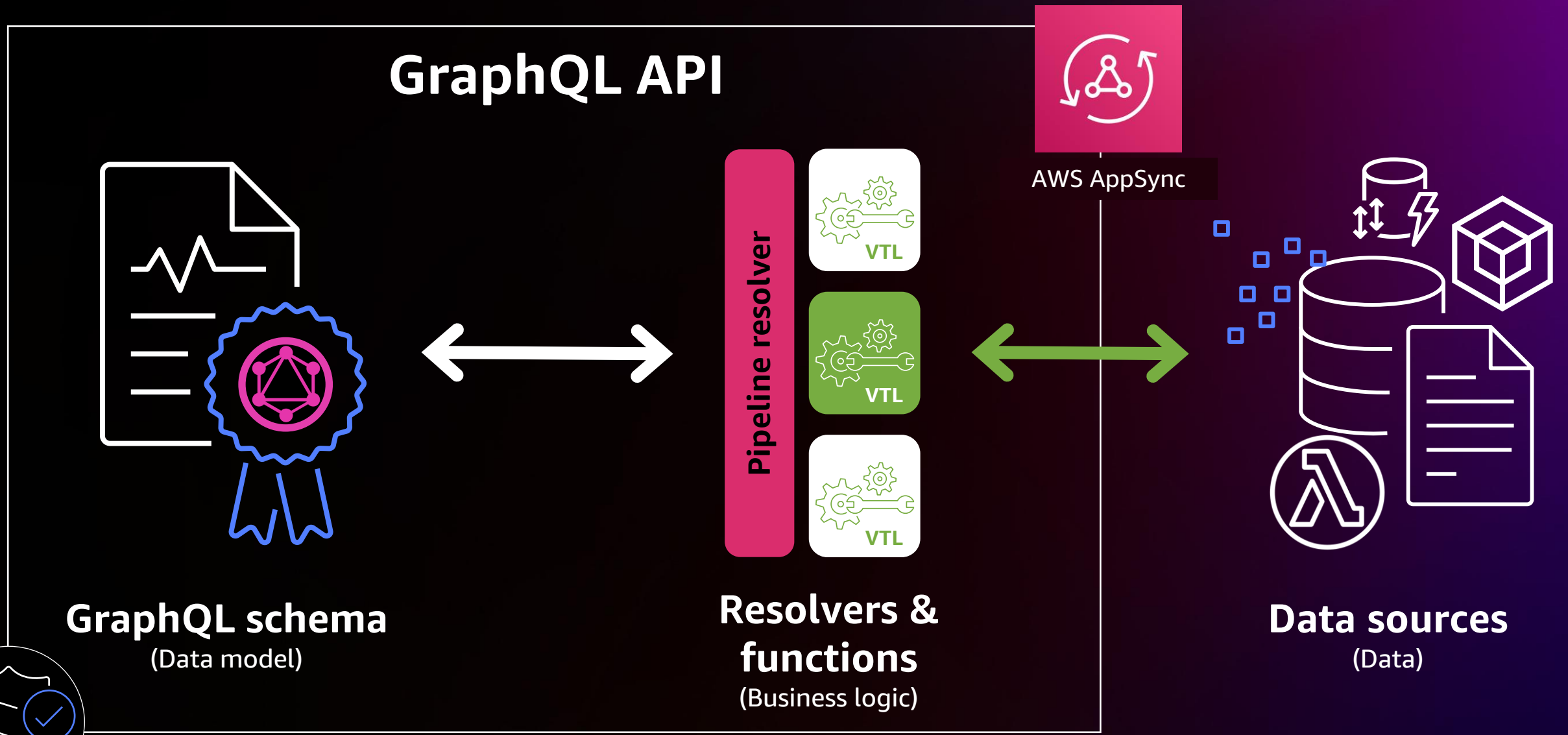
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS AppSync: How it works



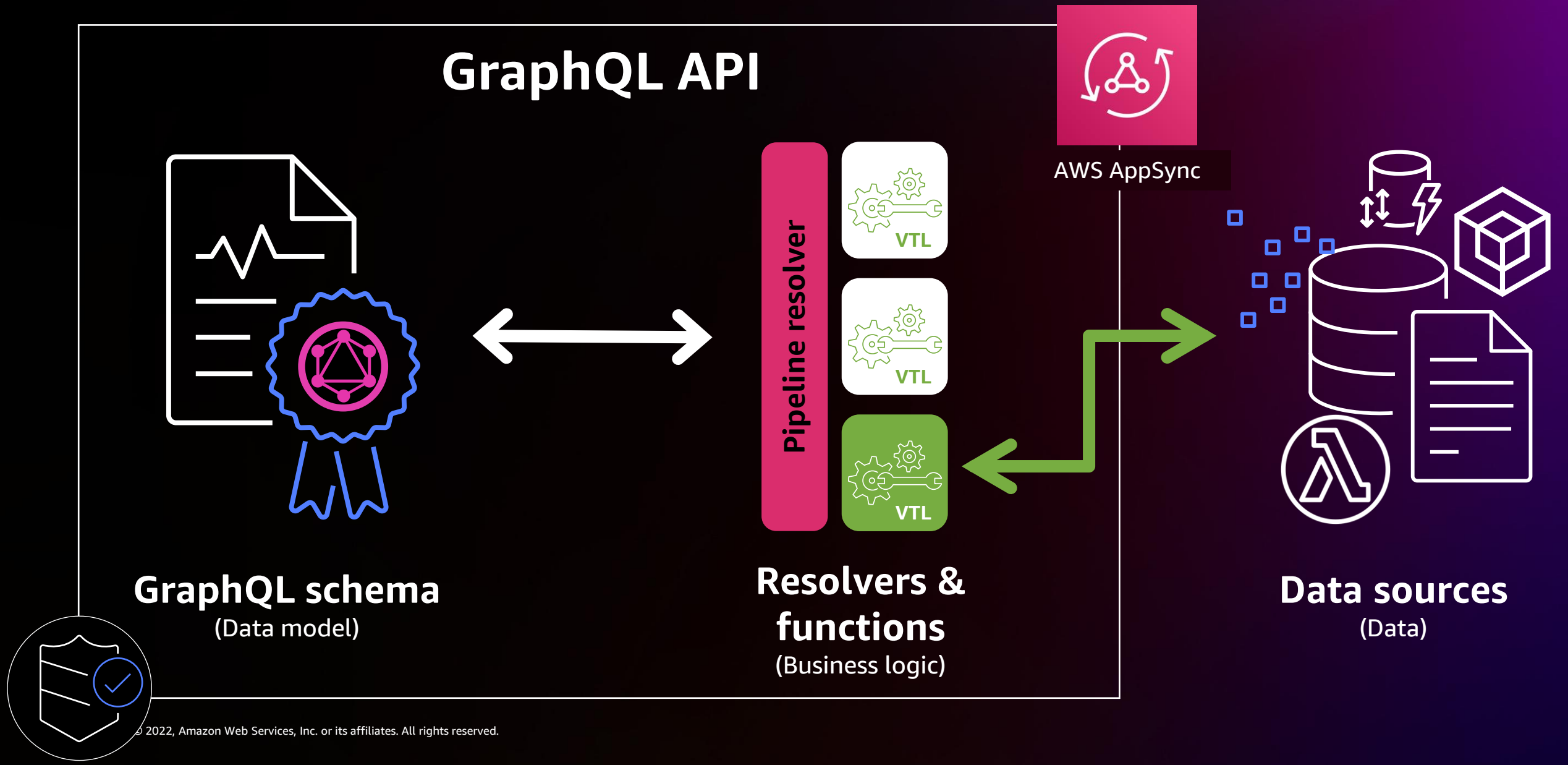
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS AppSync: How it works



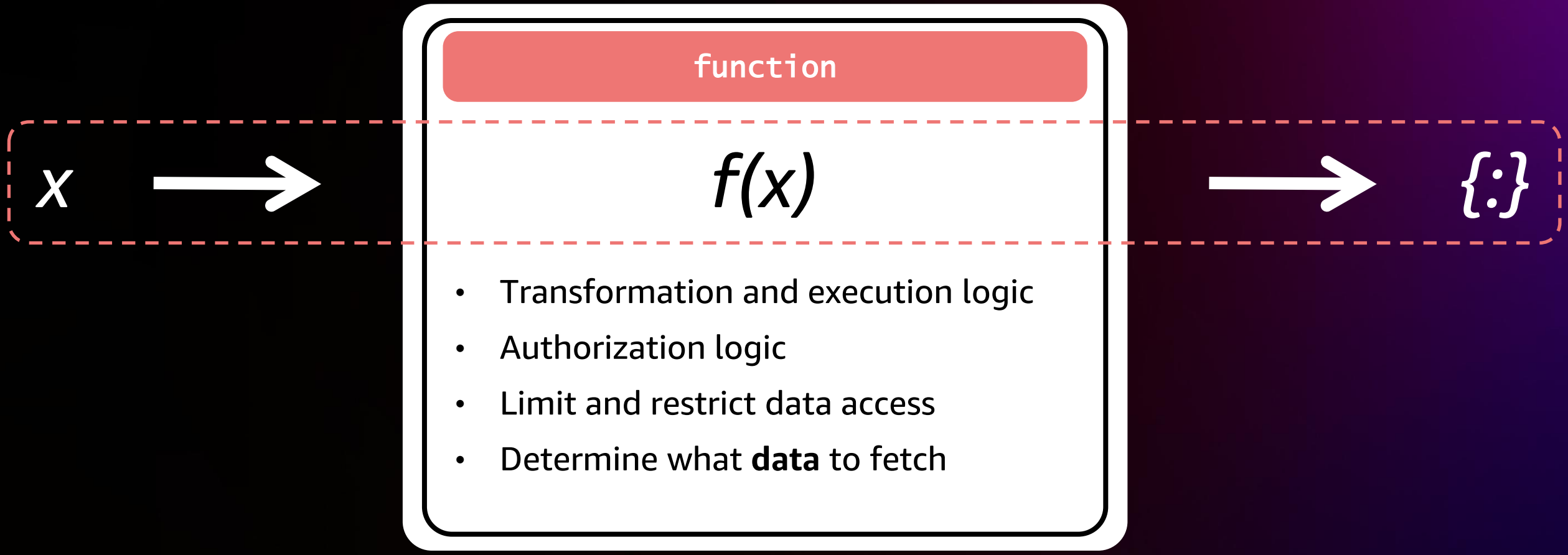
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS AppSync: How it works



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS AppSync: the role of a function



- Transformation and execution logic
- Authorization logic
- Limit and restrict data access
- Determine what **data** to fetch

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

However

- Developers want an easy way to write code
- VTL, while powerful, is mostly unknown
- Hard to write their code in VTL
- Most developers are familiar with JavaScript

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Introducing support for JavaScript

- Define data access business logic with new **APPSYNC_JS** runtime
- Write your code using JavaScript or build client-side TypeScript
- Use the native JavaScript functionality that you love
- Use types (optionally) for code completion and IntelliSense

- **You** focus on your code and business objectives
- **AWS AppSync** manages scale, security, performance

https://github.com/aws/aws-appsync-community/issues/147







aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Introducing support for JavaScript

Less code

- Just JavaScript, use AWS AppSync auth, cache, subscriptions

Less complexity

- Just JavaScript, less dependencies, focus on data access logic

Less cost

- Just JavaScript, no additional services or resources to run

- **Work in your local environment or the AWS AppSync console**
- **None of the complexities of running your own service**

https://github.com/aws/aws-appsync-community/issues/147



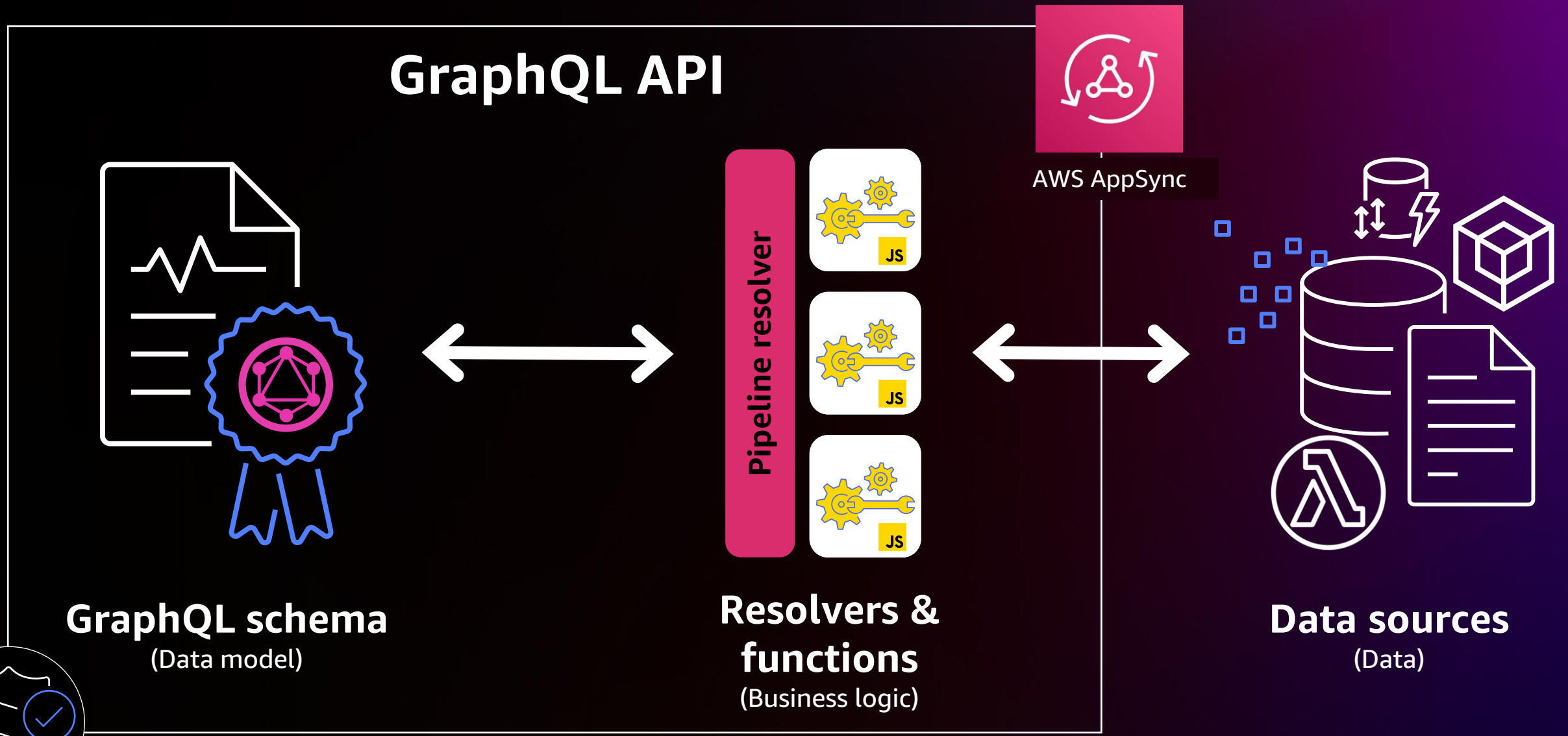




aws

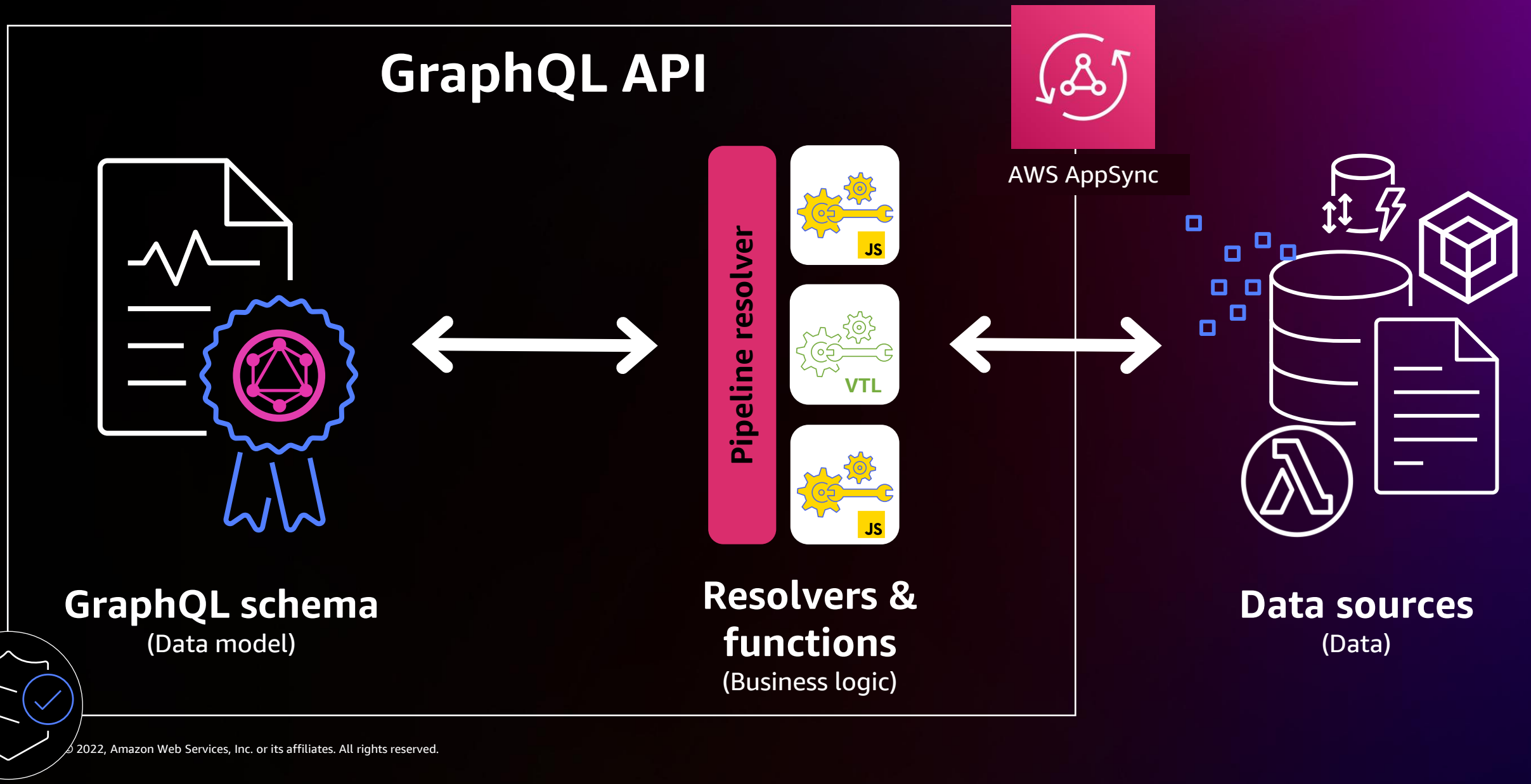
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

JavaScript in AWS AppSync functions



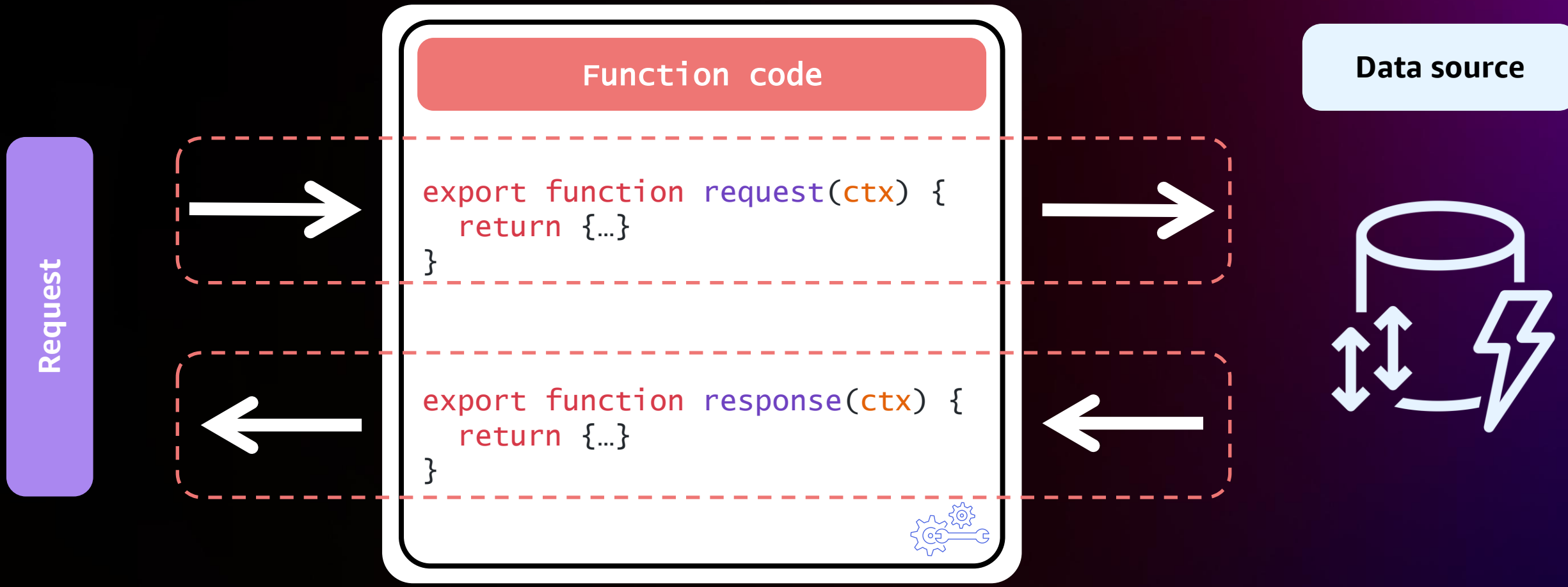
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

JavaScript functions: Mix runtime in pipeline



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

JavaScript functions: How they work



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

APPSYNC_JS runtime: Functionality

- Custom-built runtime to evaluate JavaScript in an AWS AppSync context
- ES 6 baseline support: Boolean, number, string, object, array, function
- Flow statement control
  - If/else, for of/in loops, switch/break, ternary, null coalesce, and so on
- Built-in functions: object, string, array, JSON, console, and math
- Define and re-use your own functions
- Modern JavaScript features
  - Destructuring, default assignment, spread, let/const, template literals, and so on

Do more with your code, within AWS AppSync, without extra compute infrastructure

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

From VTL to JavaScript

Before – VTL

Request mapping template

```
{
  "version": "2017-02-28",
  "operation": "GetItem",
  "key": {
    "id": {"S": $ctx.args.id}
  }
}
```

Response mapping template

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

Now – JavaScript

Code

```
export function request(ctx) {
  return {
    operation: 'GetItem',
    key: { id: { S: ctx.args.id } },
  }
}

export function response(ctx) {
  return ctx.result
}
```

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

From VTL to JavaScript

Before – VTL

```
#set( $expNames = {} )
#set( $expValues = {} )
#set( $expSet = {} )
#set( $expRemove = [] )

## Iterate through the keys of the values
#foreach( $entry in $util.map.copyAndRemoveAllKeys($values, $keyFields).entrySet() )
    #set( $entryKeyAttributeName = $entry.key )
    ## save the name
    $util.qr($expNames.put("#$entryKeyAttributeName", "$entry.key"))
    #if( $util.isNull($entry.value) )
        ## if the value is null, add to the list to be removed
        #set( $discard = $expRemove.add("#$entryKeyAttributeName") )
    #else
        ## if there is a value, add to list to be updated
        $util.qr($expSet.put("#$entryKeyAttributeName", ":$entryKeyAttributeName"))
        $util.qr($expValues.put(":$entryKeyAttributeName",
  $util.dynamodb.toDynamoDB($entry.value)))
    #end
#end
#set( $expression = "" )
#if( !$expSet.isEmpty() )
    #set( $expression = "SET" )
    #foreach( $entry in $expSet.entrySet() )
        #set( $expression = "$expression $entry.key = $entry.value" )
        #if( $foreach.hasNext() )
            #set( $expression = "$expression," )
        #end
    #end
#end
#if( !$expRemove.isEmpty() )
    #set( $expression = "$expression REMOVE" )
    #foreach( $entry in $expRemove )
        #set( $expression = "$expression $entry" )
        #if( $foreach.hasNext() )
            #set( $expression = "$expression," )
        #end
    #end
#end
```

Now – JavaScript

```
const sets = [];
const removes = [];
const expressionNames = {};
const expValues = {};

// Iterate through the keys of the values
for (const [key, value] of Object.entries(values)) {
  // save the name
  expressionNames[`#${key}`] = key;
  if (value) {
    // if there is a value, added to list to be updated
    sets.push(`#${key} = :${key}`);
    expValues[`:${key}`] = value;
  } else {
    // if the value is null, added to the list to be removed
    removes.push(`#${key}`);
  }
}

let expression = sets.length ? `SET ${sets.join(', ')}` : '';
expression += removes.length ? ` REMOVE ${removes.join(', ')}` : '';
```

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Developer experience support

JavaScript support is great, but devs need to easily **code**, **test**, and **debug** resolvers

Code

- Use tooling and types to enhance your coding experience

Test

- Unit test your resolvers to verify you code logic with existing frameworks

Debug

- Evaluate your code to see if it works, review logs, debug quickly

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Utility support

`@aws-appsync/utils`

- Provide type definitions
- Auto-completion in IDEs
- Reduce risk of using wrong utilities (such as, no more typos)
- Helps you write correct requests



```
import { Context, util } from '@aws-appsync/utils'

export function request(ctx: Context) {
  const {
    args: { key1, key2, .
  } = ctx

  const id = key1 ?? util.autoId()
  const key = { id, key2 }
  const condition = {
    key1: { attributeExists: true },
    key2: { attributeExists: true },
  }
  return dynamodbUpdateRequest({ key, values, condition })
}
```

```
(method) autoId(): string

Returns a 128-bit randomly generated UUID.

@returns — - Randomly generated UUID
```

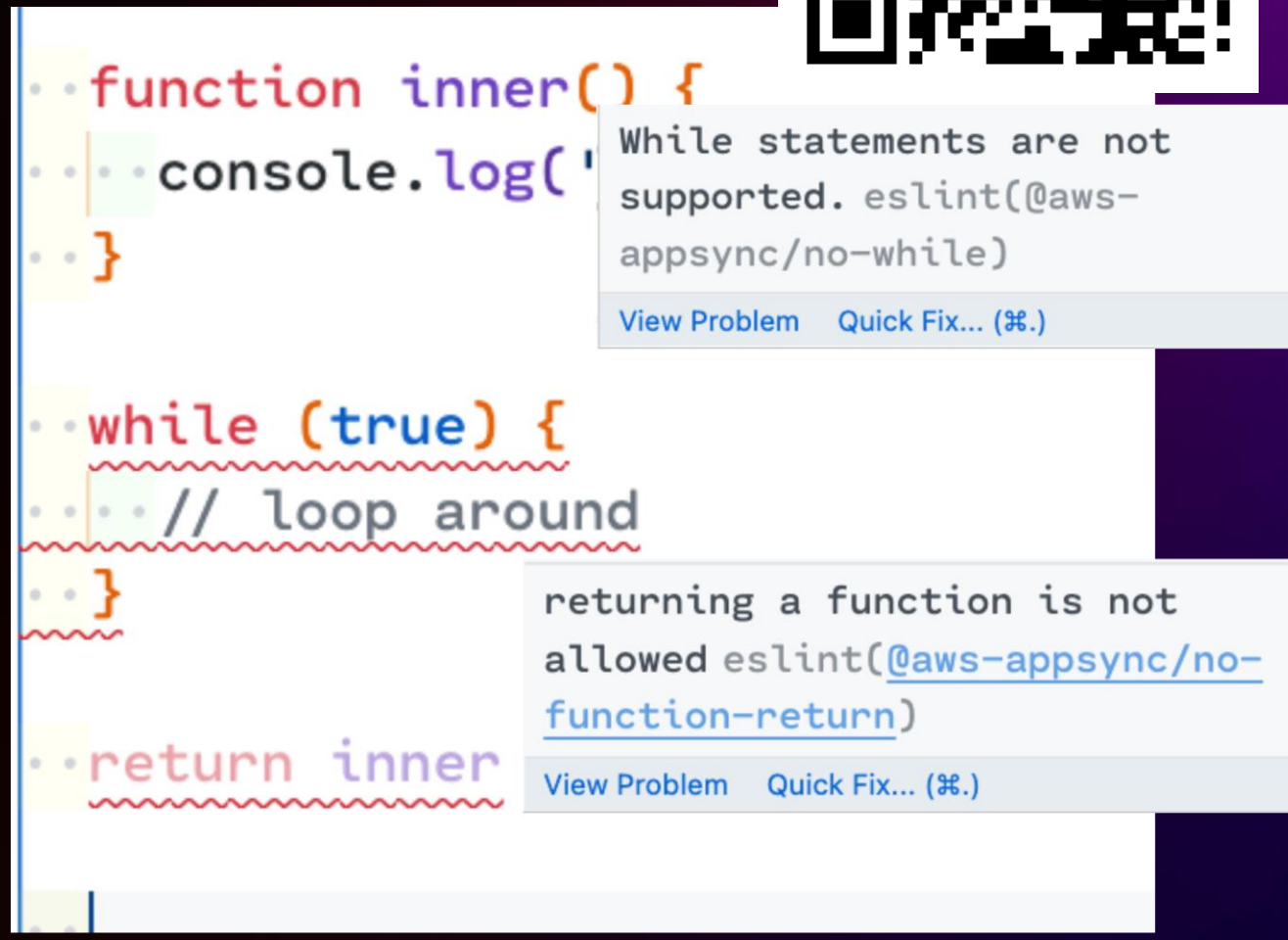
aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Eslint support

`@aws-appsync/eslint-plugin`

- Warns you of unsupported features
- Helps you write compliant code
- Unresolvable issues caught before and when you submit your code



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Testing & debugging

New `evaluateCode` command

- Quick feedback
- Accurate evaluation of your code
- Catches code issues before you save your code
- Includes logs to help debug, troubleshoot

```
// Iterate through the keys of the values
for (const key in values) {
  // save the name
  expressionNames[`#${key}`] = key
  const value = values[key]
  console.log(`Adding to update: ${key}: ${value ? value : '-'}`)
  if (value) {
    // add the value to the update list
    sets.push(`#${key} = :${key}`)
    expValues[`:${key}`] = value
  } else {
    // add the key to the delete list
    console.error(` > ${key} has not value - marking for deletion`)
    removes.push(`#${key}`)
  }
}
console.log(`Updates: ${sets.length}, Deletes: ${removes.length}`)
```

```
INFO - code.js:45:5: "Adding to update: owner: john random doe"
INFO - code.js:45:5: "Adding to update: nullKey: -"
ERROR - code.js:52:7: " > nullKey has not value - marking for deletion"
INFO - code.js:45:5: "Adding to update: limit: 13"
INFO - code.js:45:5: "Adding to update: title: hello"
INFO - code.js:45:5: "Adding to update: empty: -"
ERROR - code.js:52:7: " > empty has not value - marking for deletion"
INFO - code.js:56:3: "Updates: 3, Deletes: 2"
```

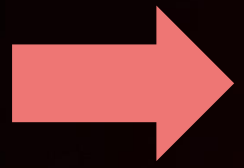
aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Improved Amazon Cloudwatch Logs

```
3cc INFO - code.js:45:5: "Adding to update: owner: brice"
3cc INFO - code.js:45:5: "Adding to update: description: update the description"
3cc INFO - code.js:45:5: "Adding to update: title: -"
3cc ERROR - code.js:52:7: " > title has not value - marking for deletion"
3cc INFO - code.js:56:3: "Updates: 2, Deletes: 1"
```

Clear logs with line numbers

Clear evaluation result



```
    "evaluationResult": {
        "operation": "UpdateItem",
        "key": {
            "id": {
                "S": "uuid"
            }
        },
        "condition": {
            "expression": "(attribute_exists(#id))",
            "expressionNames": {
                "#id": "id"
            }
        },
        "update": {
            "expression": "SET #owner = :owner, #description =
:description REMOVE #title",
            "expressionNames": {
                "#owner": "owner",
                "#description": "description",
                "#title": "title"
            },
            "expressionValues": {
                ":owner": {
                    "S": "brice"
                },
                ":description": {
                    "S": "update the description"
                }
            }
        }
    },
```

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

JavaScript: Calling an AWS service

Easiest way to call an AWS service; no SDK required

1. Setup an HTTP datasource pointing at an AWS service endpoint
2. Create an AWS AppSync function that sends an HTTPRequest
3. Use one of the functions provided at aws-samples/aws-appsync-resolver-samples



```
export function request(ctx) {
  const {text, source, target} = ctx.args
  return awsTranslateRequest(text, source, target)
}

export function response(ctx) {
  const { result } = ctx
  if (result.statusCode !== 200) {
    return util.appendError(result.body, `${result.statusCode}`)
  }

  const { TranslatedText: text } = JSON.parse(result.body)
  return text
}
```

aws © 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Direct data source vs. Lambda data source

USE THEM BOTH, SELECT THE OPTION THAT WORKS FOR YOUR USE CASE

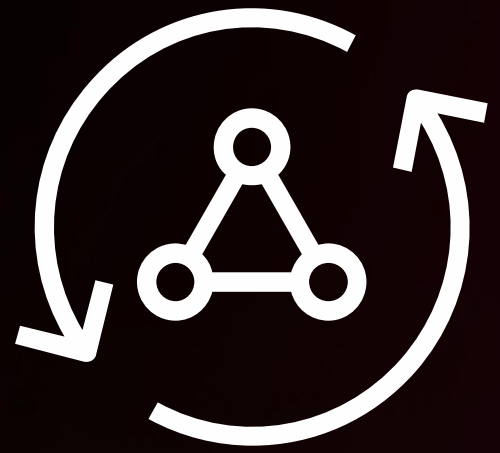
	Direct data source integration	Lambda data source as a proxy
Use case	**AWS AppSync functions interact directly with API data sources**	**AWS AppSync functions call Lambdas that interact with API data sources**
Runtime	APPSYNC_JS (JavaScript)	Any supported Lambda runtime
Maximum size of code	32,000 characters per AWS AppSync function	50 MB (zipped) per Lambda function
External modules	Limited, APPSYNC_JS features only	Yes
Call any AWS service	Yes, using AWS AppSync HTTP data source	Yes, using AWS SDK
Access request header	Yes	Yes
Network / FS access	No	Yes
Logging and metrics	Yes	Yes
Build and test entirely within AWS AppSync	Yes	No
Cold start	No	No, with provisioned concurrency
Auto-scaling	Yes, transparently by AWS AppSync	Yes, as configured in Lambda
Pricing	No additional charge	Charged for Lambda usage

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

JavaScript resolvers in AWS AppSync

- Easiest way to write GraphQL API with custom logic
- Available now
- Visit the documentation
  https://docs.aws.amazon.com/appsync/
- Checkout the samples
  https://github.com/aws-samples/aws-appsync-resolver-samples
- Get started today
  https://aws.amazon.com/appsync



aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS AppSync sessions at re:Invent 2022

Building a multi-Region serverless application with AWS AppSync

FWM310

How Amazon Location Service enabled tracking of London Marathon runners

FWM201

Create real-time, event-driven apps with Amazon EventBridge & AWS AppSync

FWM315

Modernize and gradually migrate your data model from SQL to NoSQL

BOA321

Build cross-platform interactive polls with .NET and MQTT

BOA319

Build with prototypes: Be a customer-obsessed developer

BWP201

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Ed Lima
@ednergizer

Brice Pellé
@BricePelle

Learn more and get started for free at aws.amazon.com/appsync/



Please complete the session survey in the **mobile app**

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.