AWS AppSync

GraphQL APIs for serverless webapps and real time data

Richie Schmid - Senior Developer
WebGate

13.09.2018

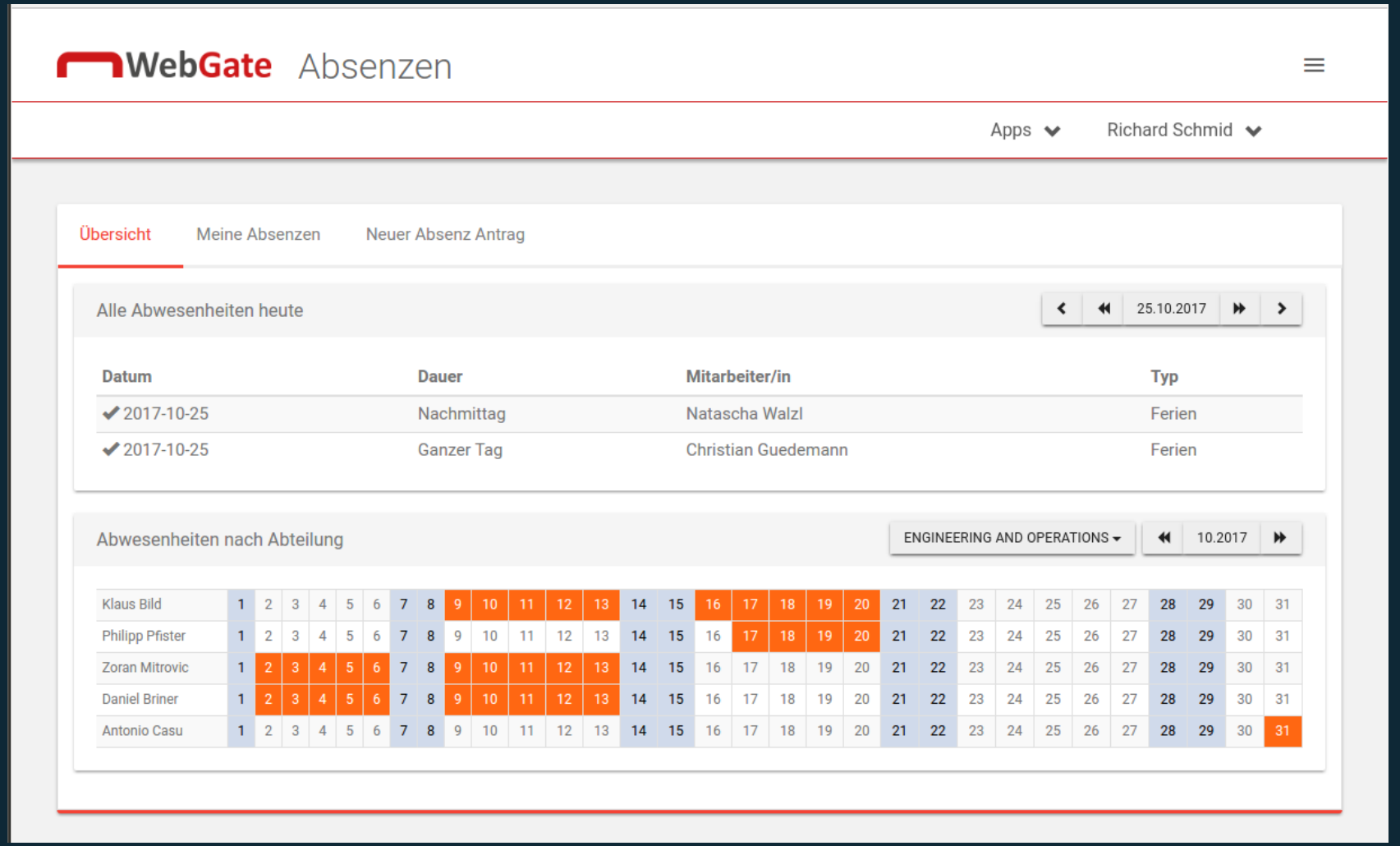
blog.webgate.biz

Serverless Apps are cool

- No servers
- No patching
- No capacity guessing
- Pricing
- etc etc.. well, you (hopefully) already know all the advantages..

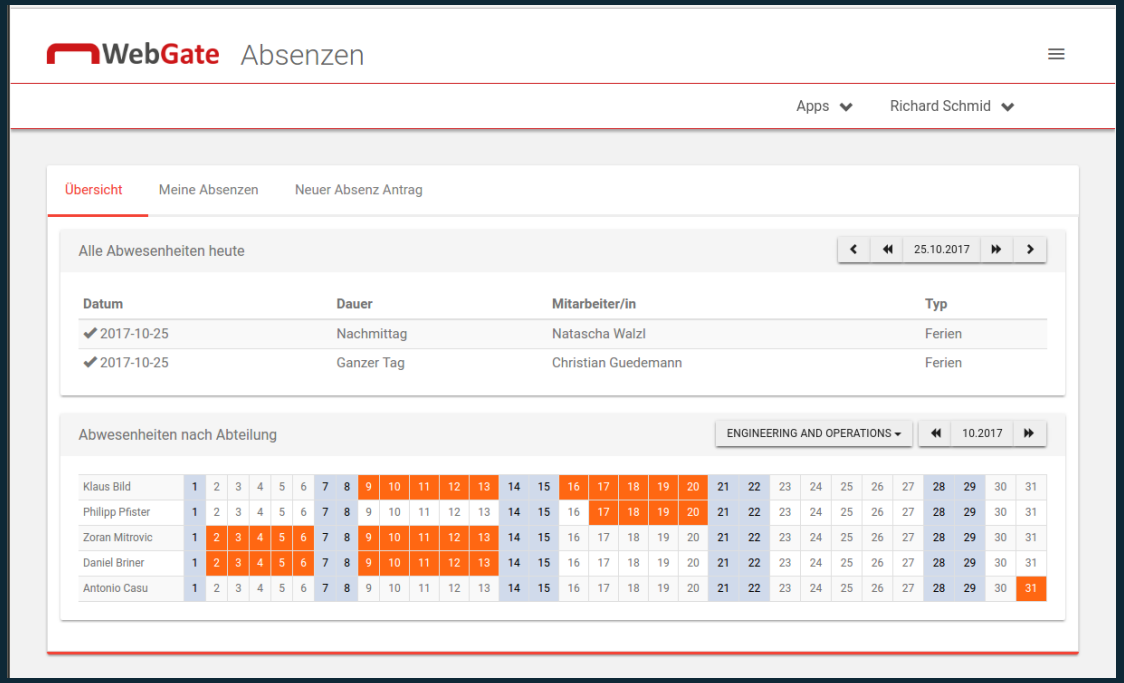
Serverless Apps with serverless components

Typical REST API combo for WebApps:


WebGate Absenzen
Apps
Richard Schmid
Übersicht
Meine Absenzen
Neuer Absenz Antrag
Alle Abwesenheiten heute
25.10.2017
Datum
Dauer
Mitarbeiter/in
Typ
2017-10-25
Nachmittag
Natascha Walzl
Ferien
2017-10-25
Ganzer Tag
Christian Guedemann
Ferien
Abwesenheiten nach Abteilung
ENGINEERING AND OPERATIONS
10.2017
Klaus Bild
Philipp Pfister
Zoran Mitrovic
Daniel Briner
Antonio Casu

WebGate

aws



Dashboard API Calls:

- Today's absences

/getAbsencesByDate

- List of departments

/getDepartments

- Team Absences, depending on user Department

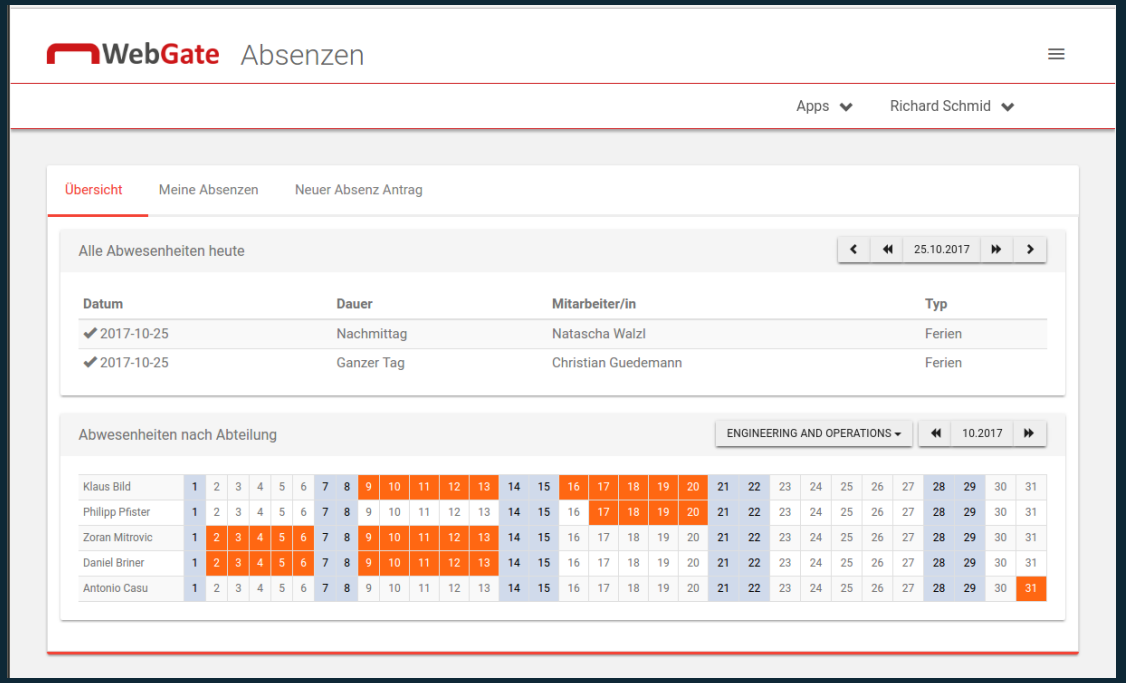
/getAbsencesByDepartment

→ Provider driven API, instead of consumer driven

→ Many REST calls instead of just one

WebGate

aws



Consolidate many microservices into single API calls?

We could combine all info into one single REST call.

- May be ok for a small app like this

- In larger apps quickly ends in countless special cases

And of course requires programming and maintaining..

WebGate

aws

REST APIs for real-time data?

If you want to build apps displaying real-time data, REST calls will always show delayed data!

With REST you have to poll for updates every *n* minutes.

Not relevant for our absence app, but may be useful in our shift planer app, a webshop ordering queue or sensor dashboards!

WebGate

aws

WebSockets

- Good for real-time data
- Permanent bidirectional connection

Looks like we need a constantly running server, as replacement for Lambda API-GW?

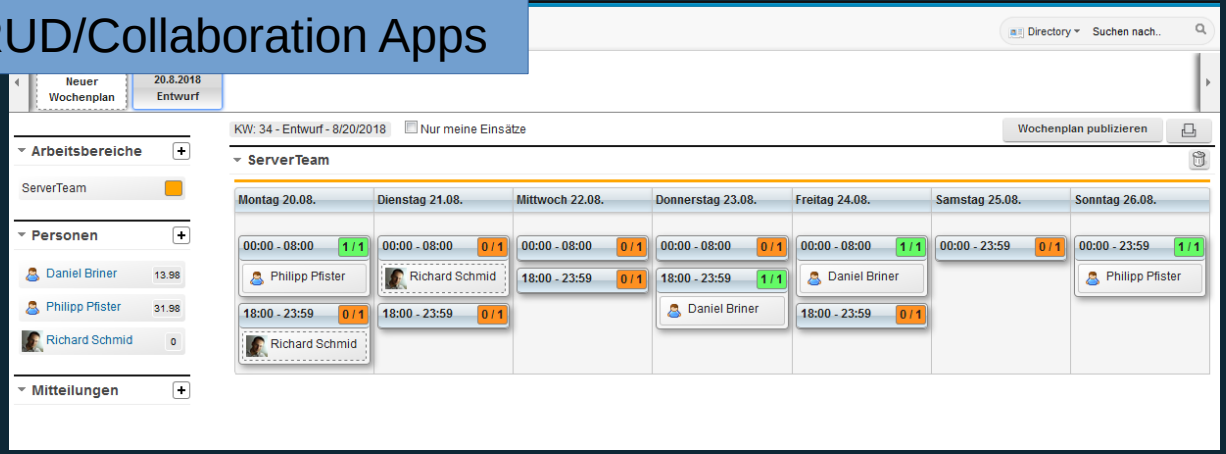
Goodbye serverless?

aws

Sensor Dashboard

AWS IoT Hub

CRUD/Collaboration Apps



AWS AppSync

WebGate

aws

AWS AppSync
Serverless GraphQL Service

- GraphQL query APIs (HTTP POST)
- WebSockets for subscription to real-time updates
- Even has offline-support!
- Seamlessly integrated into AWS serverless goodies:

  DynamoDB, Lambda, Cognito..

What is GraphQL?

GraphQL

From Wikipedia, the free encyclopedia

GraphQL is an open source data query and manipulation language, and a runtime for fulfilling queries with existing data.[2] GraphQL was developed internally by Facebook in 2012 before being publicly released in 2015.[3] It provides a more efficient, powerful and flexible alternative to REST and ad-hoc web service architectures.[2][4] It allows clients to define the structure of the data required, and exactly the same structure of the data is returned from the server, therefore preventing excessively large amounts of data from being returned.

GraphQL supports reading, writing (mutating) and subscribing to changes to data (realtime updates).[5]

Major GraphQL clients include Apollo Client[6] and Relay.[7] GraphQL servers are available for multiple languages, including Haskell, JavaScript, Python,[8] Ruby, Java, C#, Scala, Go, Elixir,[9] Erlang, PHP, R, and Clojure.

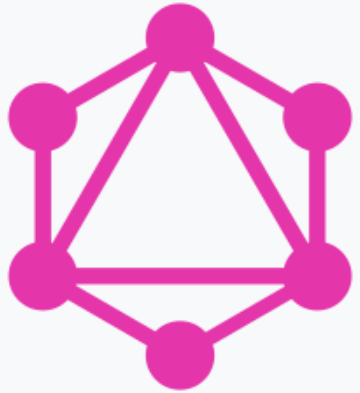
On the 9th February 2018, the GraphQL Schema Definition Language (SDL) was made part of the specification.[10]

See also [edit]

- Query by Example

References [edit]

GraphQL



Developer(s)	Facebook, and community
Initial release	2015

- Developed by Facebook, made public in 2015, SDL added in 2018.
- Allows clients to define structure of required data, and get exactly that
- GraphQL client libraries available for many languages (Apollo)

WebGate

aws

GraphQL – Query Language for APIs

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"
  }
}
```

- Ask for what you need, get exactly that
- Get many resources in a single request
- Single API endpoint for everything
- Describe what's possible with a type system
- Evolve API without versions

https://graphql.org

WebGate

aws

Ask for what you need and get exactly that

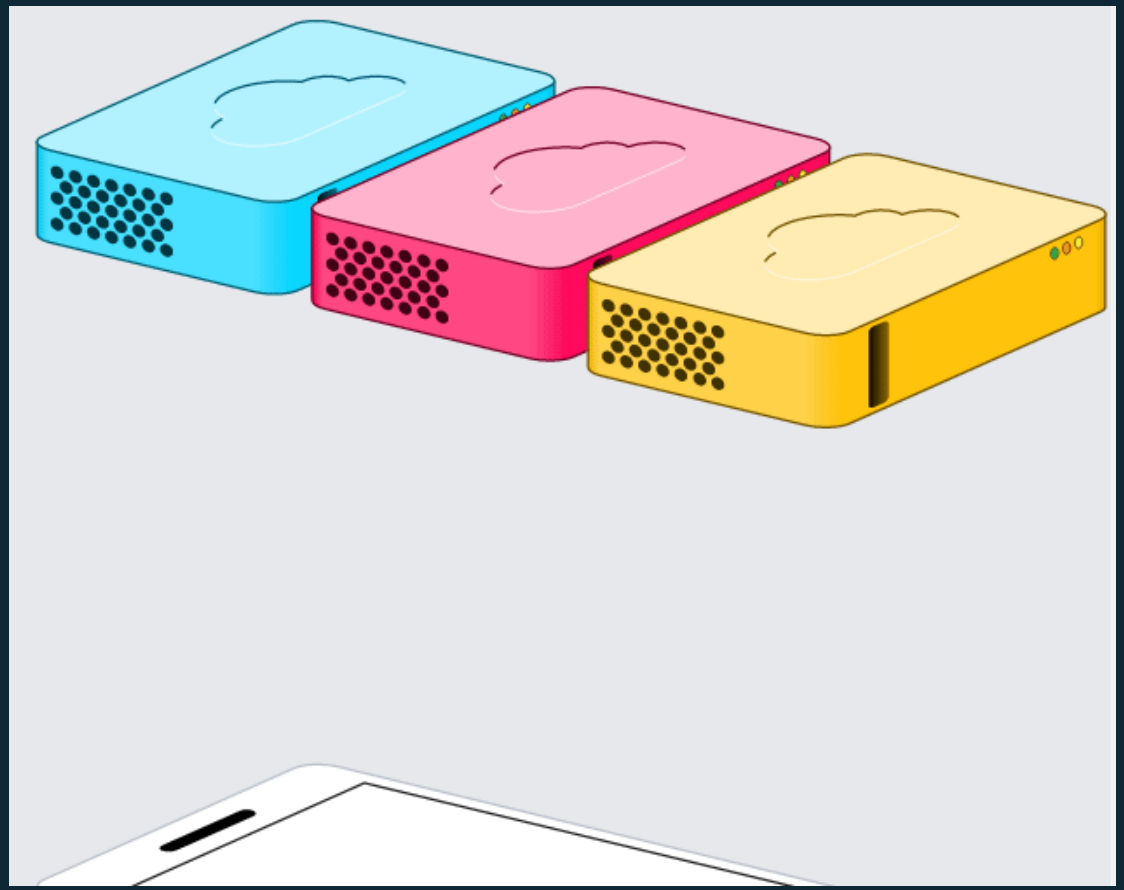
Request:

```
{
  hero {
    name
  }
}
```

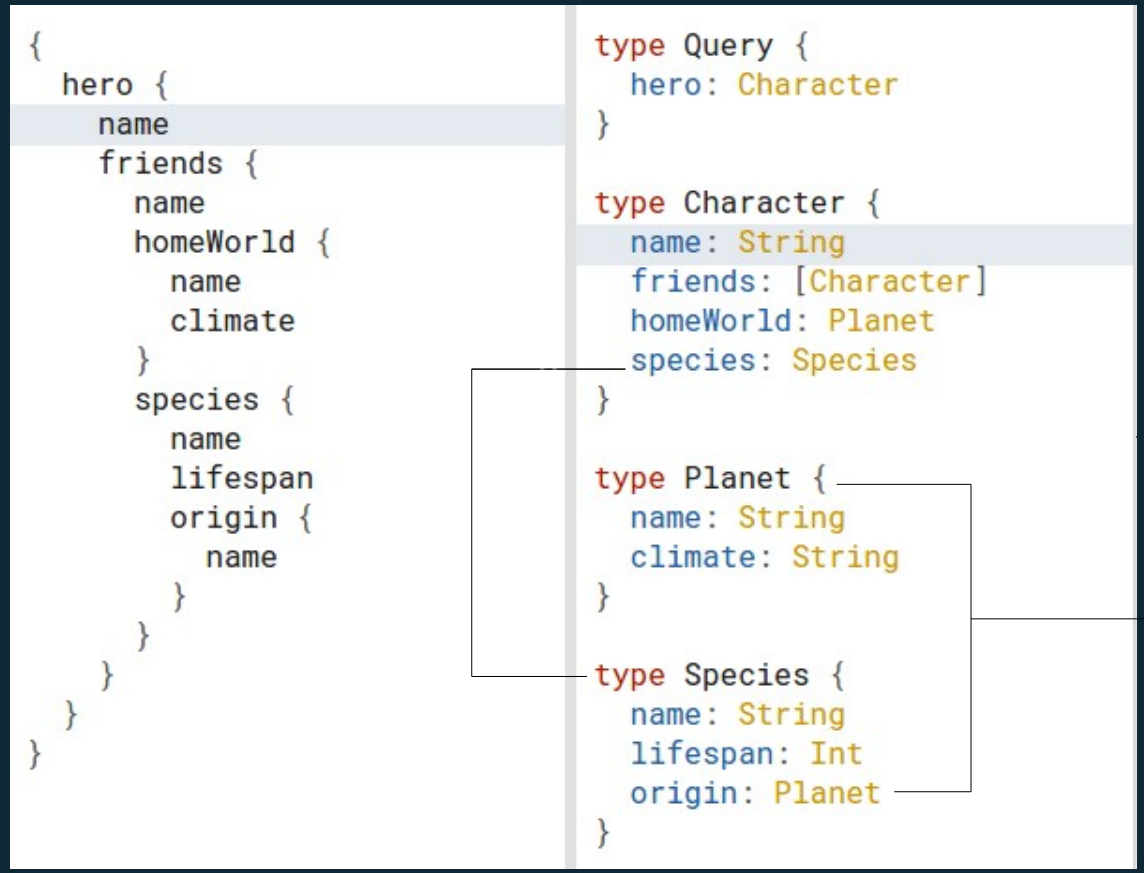
Response:

```
{
  "hero": {
    "name": "Luke Skywalker"
  }
}
```

Get many resources in a single request



Describe what's possible with a type system



```
{
  hero {
    name
    friends {
      name
      homeWorld {
        name
        climate
      }
      species {
        name
        lifespan
        origin {
          name
        }
      }
    }
  }
}
```

```
type Query {
  hero: Character
}

type Character {
  name: String
  friends: [Character]
  homeWorld: Planet
  species: Species
}

type Planet {
  name: String
  climate: String
}

type Species {
  name: String
  lifespan: Int
  origin: Planet
}
```

→ Did you notice? You can now build relations in DynamoDB without coding!

HOW COOL IS THAT!?!1!

GraphQL API interactions

Query
HTTP POST

```
query HeroNameAndFriends {
  hero {
    name
    friends {
      name
    }
  }
}
```

Mutation
HTTP POST

```
mutation {
 createTask(title:”Cleanup”){
        id
 }
}
```

Subscription
WebSocket

```
subscription {
 onCreateComment{
        id
 }
}
```

WebGate

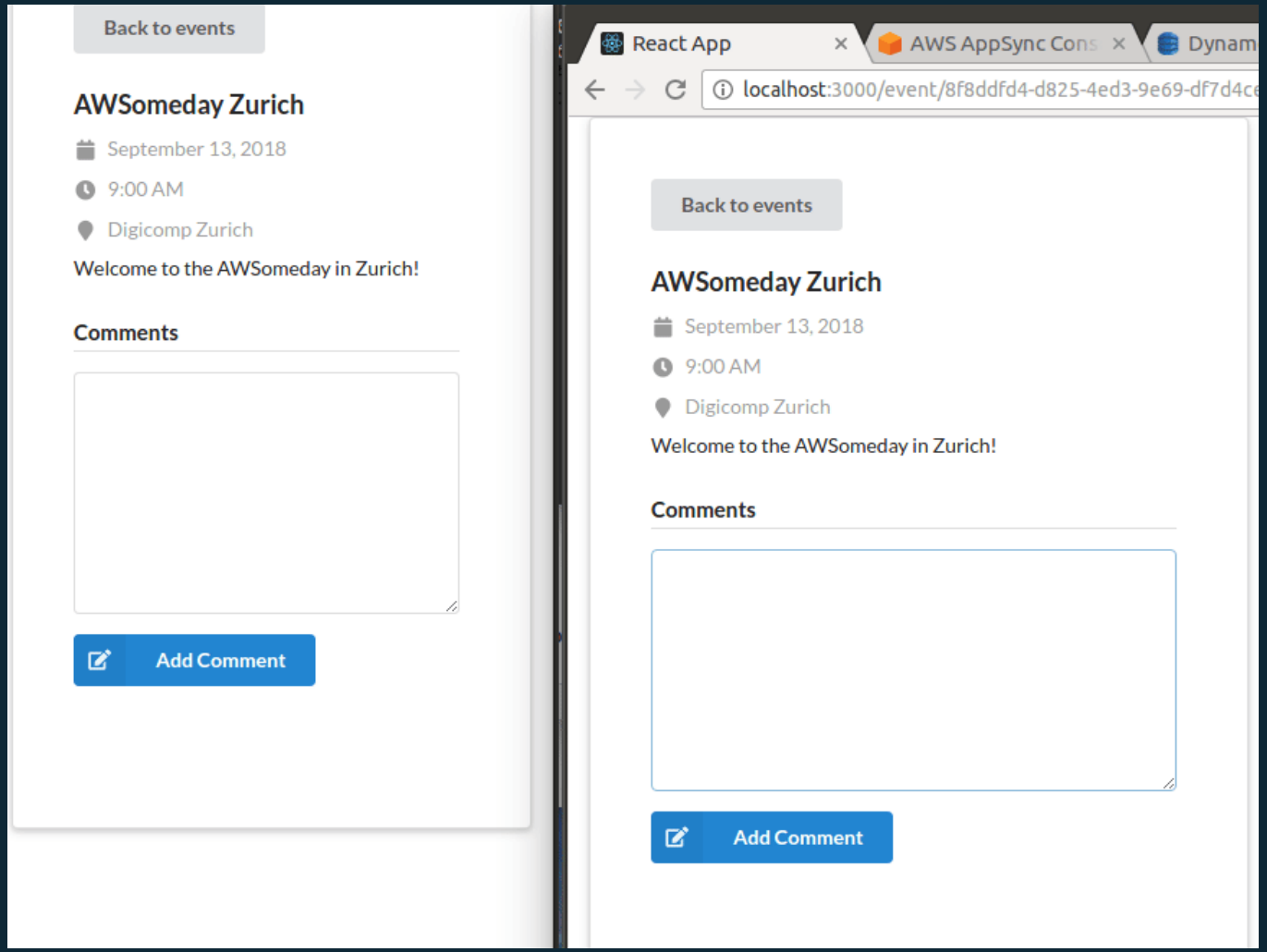
aws

Let’s see this in action

Event Demo WebApp in React

with real-time comment updates

User A



User B

WebGate

aws

AWSomeday Zurich

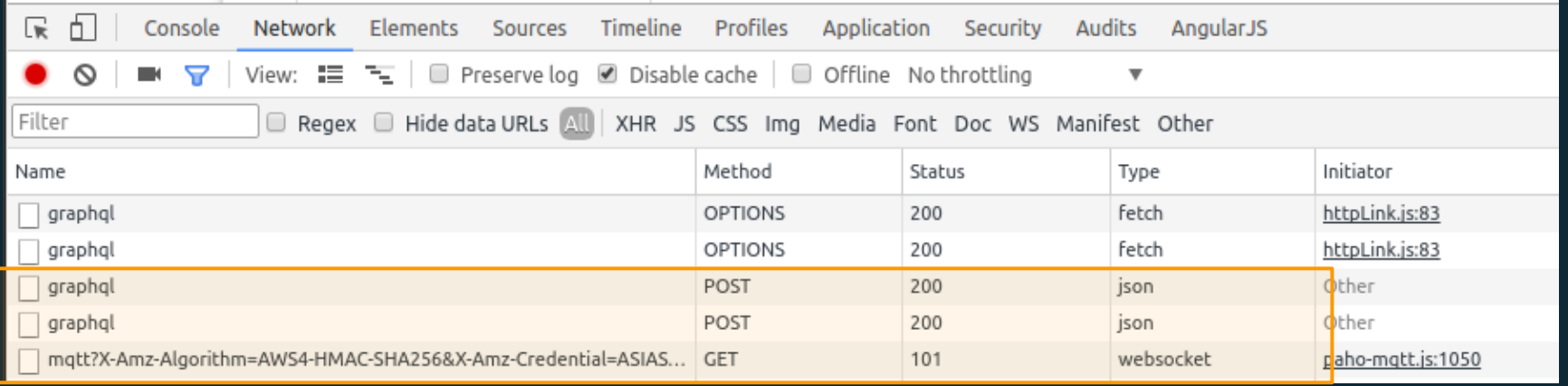
September 13, 2018

9:00 AM

Digicomp Zurich

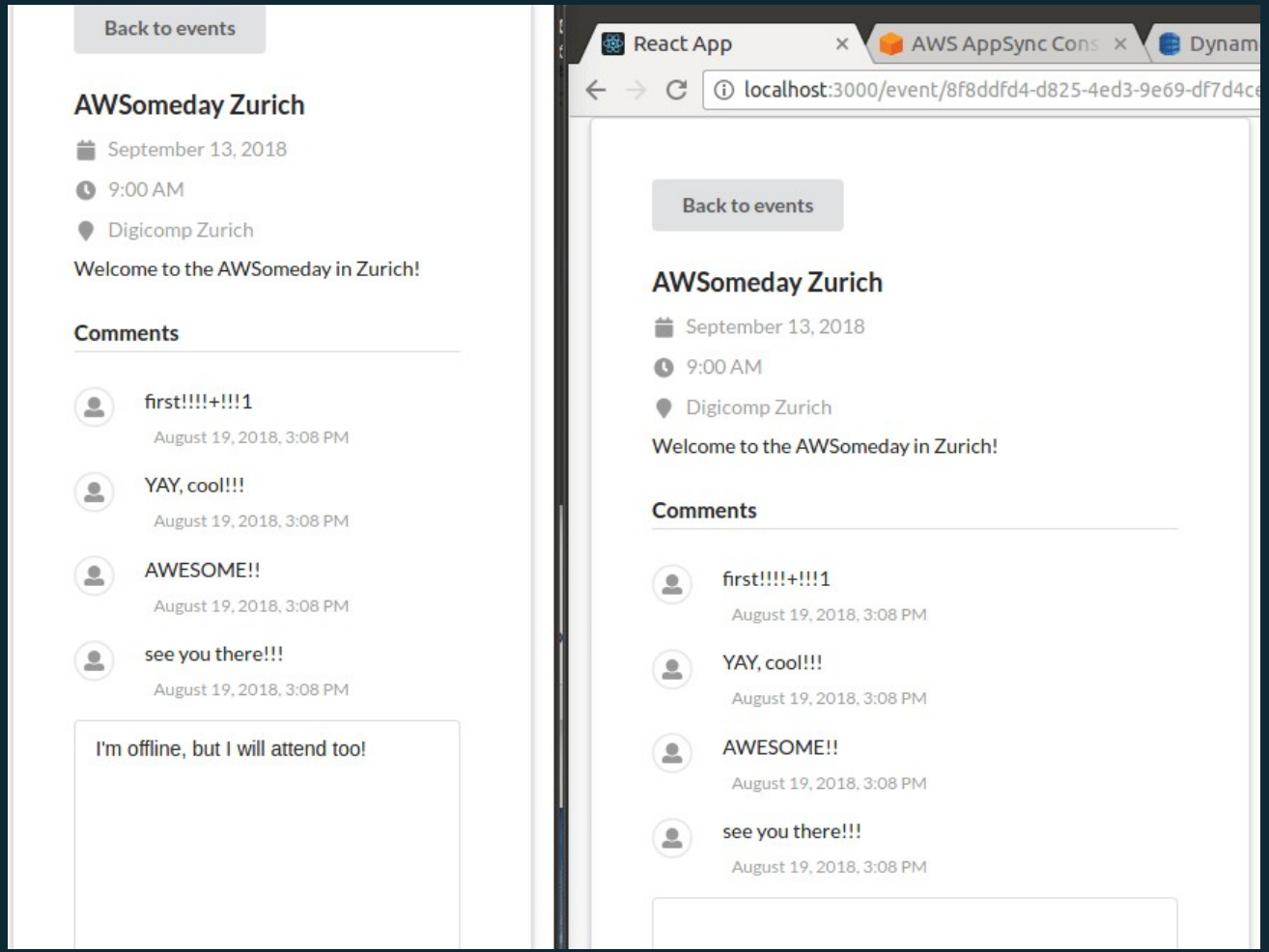
Welcome to the AWSomeday in Zurich!

Comments



Offline interaction

UI is updated immediately, but request remains in the queue



Optimistic UI

- optimisticReponse in mutate function

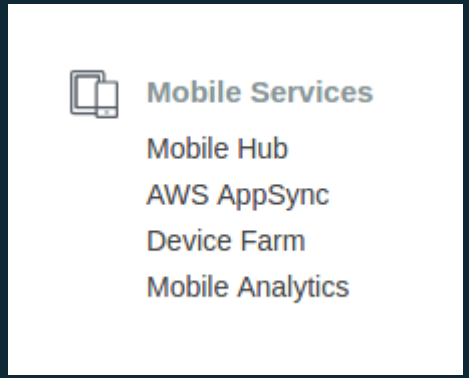
  Update the UI immediately, instead of waiting for server response.

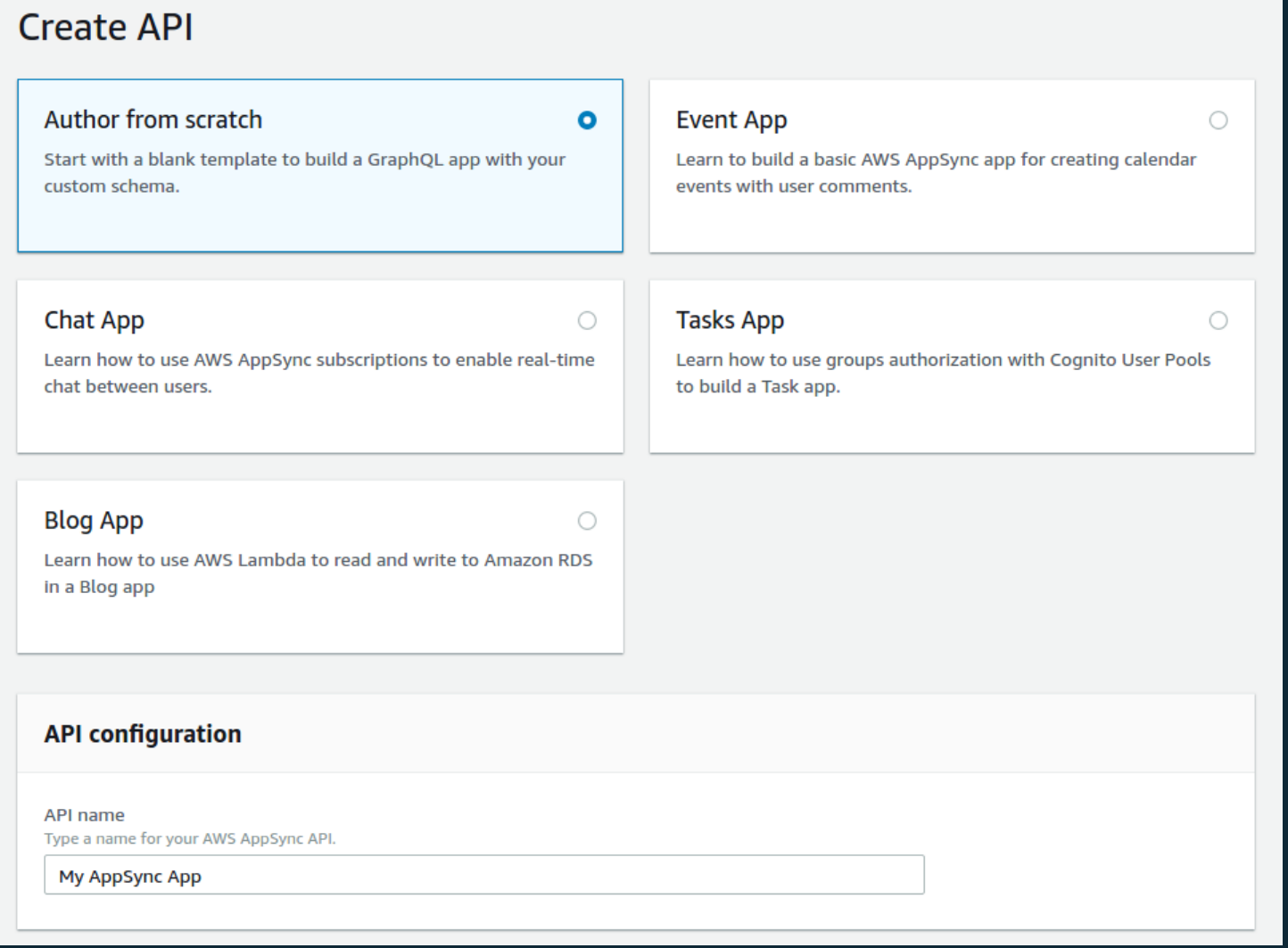
  Not just for offline mode.
  App appears to be faster for the user!

```
@Component({ ... })
class AppComponent {
  submit({ commentId, commentContent }) {
    this.apollo.mutate({
      variables: { commentId, commentContent },
      optimisticResponse: {
        __typename: 'Mutation',
        updateComment: {
          id: commentId,
          __typename: 'Comment',
          content: commentContent,
        },
      },
    }).subscribe();
  }
}
```

Create a new AWS AppSync API

- Open the AWS web console in a supported region and navigate to “Mobile Services > AppSync”

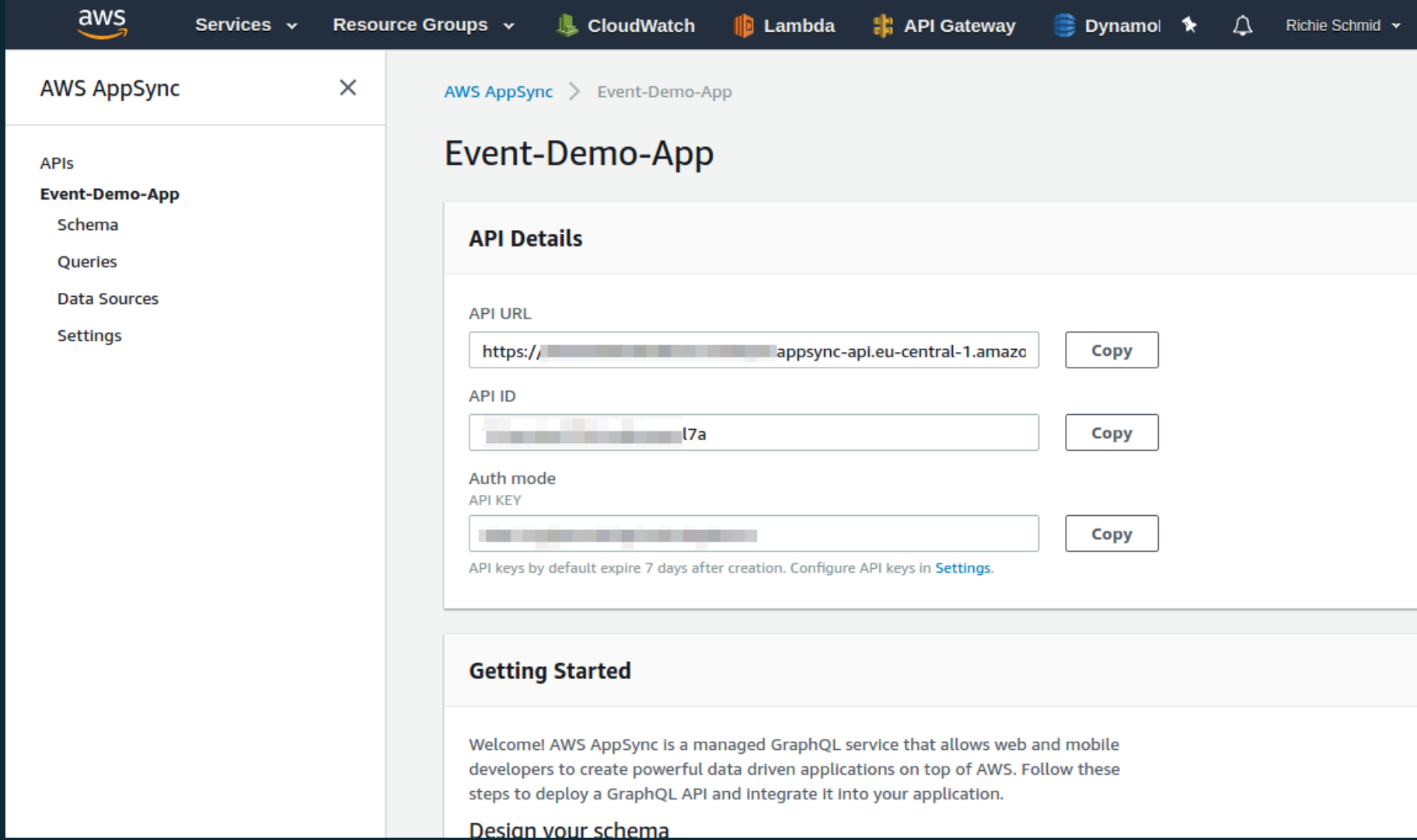


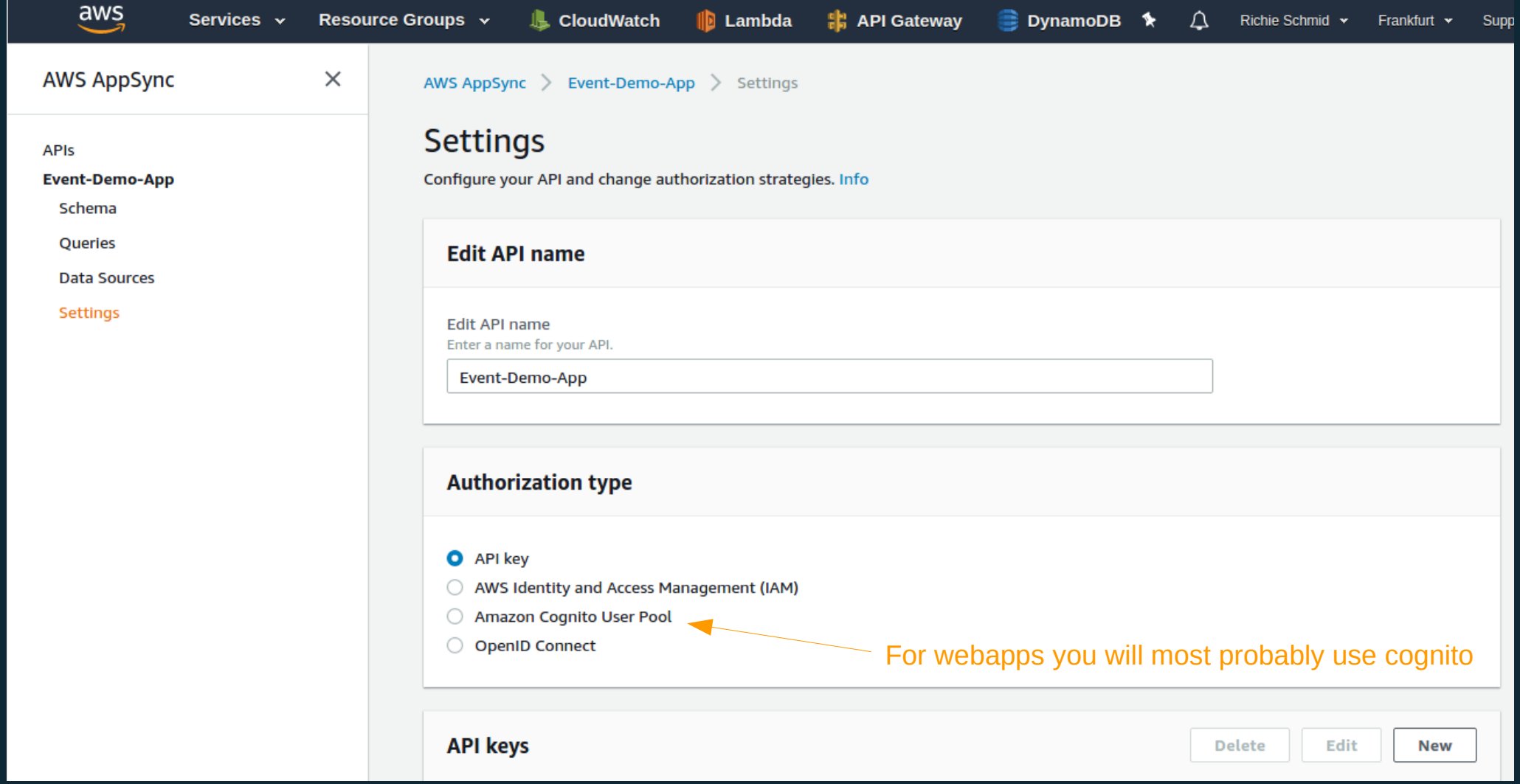


There are sample client apps available for:

- Angular (Web)
- React (Web)
- React Native
- iOS
- Android

Selecting a demo app will create required resources in the background (DynamoDB tables etc)


aws
Services
Resource Groups
CloudWatch
Lambda
API Gateway
Dynamol
Richie Schmid
AWS AppSync
APIs
Event-Demo-App
Schema
Queries
Data Sources
Settings
AWS AppSync > Event-Demo-App
Event-Demo-App
API Details
API URL
https://appsync-api.eu-central-1.amazo
Copy
API ID
l7a
Copy
Auth mode
API KEY
Copy
API keys by default expire 7 days after creation. Configure API keys in Settings.
Getting Started
Welcome! AWS AppSync is a managed GraphQL service that allows web and mobile developers to create powerful data driven applications on top of AWS. Follow these steps to deploy a GraphQL API and integrate it into your application.
Design your schema


aws
Services
Resource Groups
CloudWatch
Lambda
API Gateway
DynamoDB
Richie Schmid
Frankfurt
Supp
AWS AppSync
APIs
Event-Demo-App
Schema
Queries
Data Sources
Settings
AWS AppSync > Event-Demo-App > Settings
Settings
Configure your API and change authorization strategies. Info
Edit API name
Edit API name
Enter a name for your API.
Event-Demo-App
Authorization type
API key
AWS Identity and Access Management (IAM)
Amazon Cognito User Pool
OpenID Connect
For webapps you will most probably use cognito
API keys
Delete
Edit
New

Data Sources

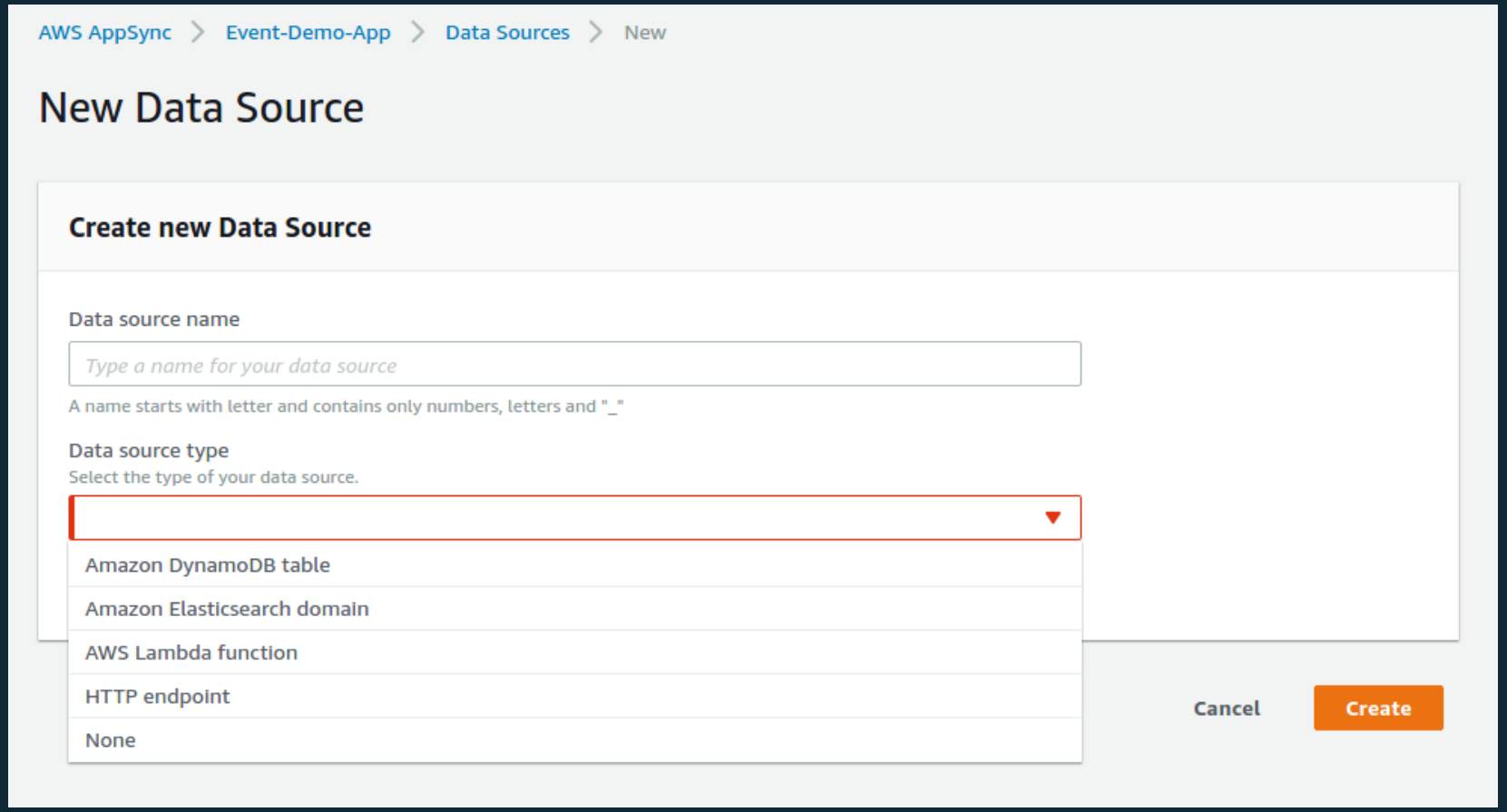
Connect existing AWS resources to your API. Info

Data Sources

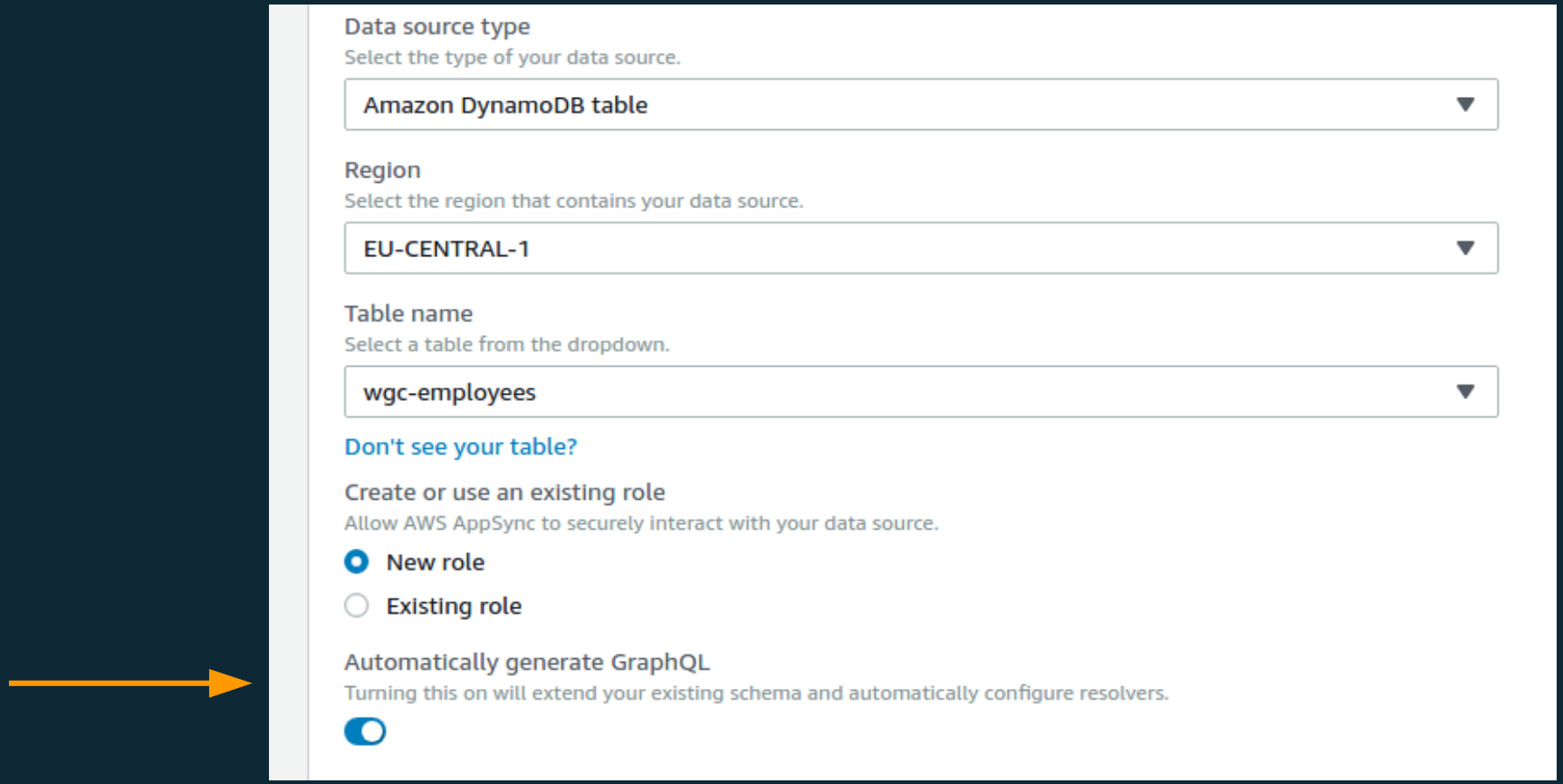
Delete Edit New

1

Name	Type	Resource
AppSyncCommentTable	AMAZON_DYNAMODB	AppSyncCommentTable-NqcjATND
AppSyncEventTable	AMAZON_DYNAMODB	AppSyncEventTable-NqcjATND

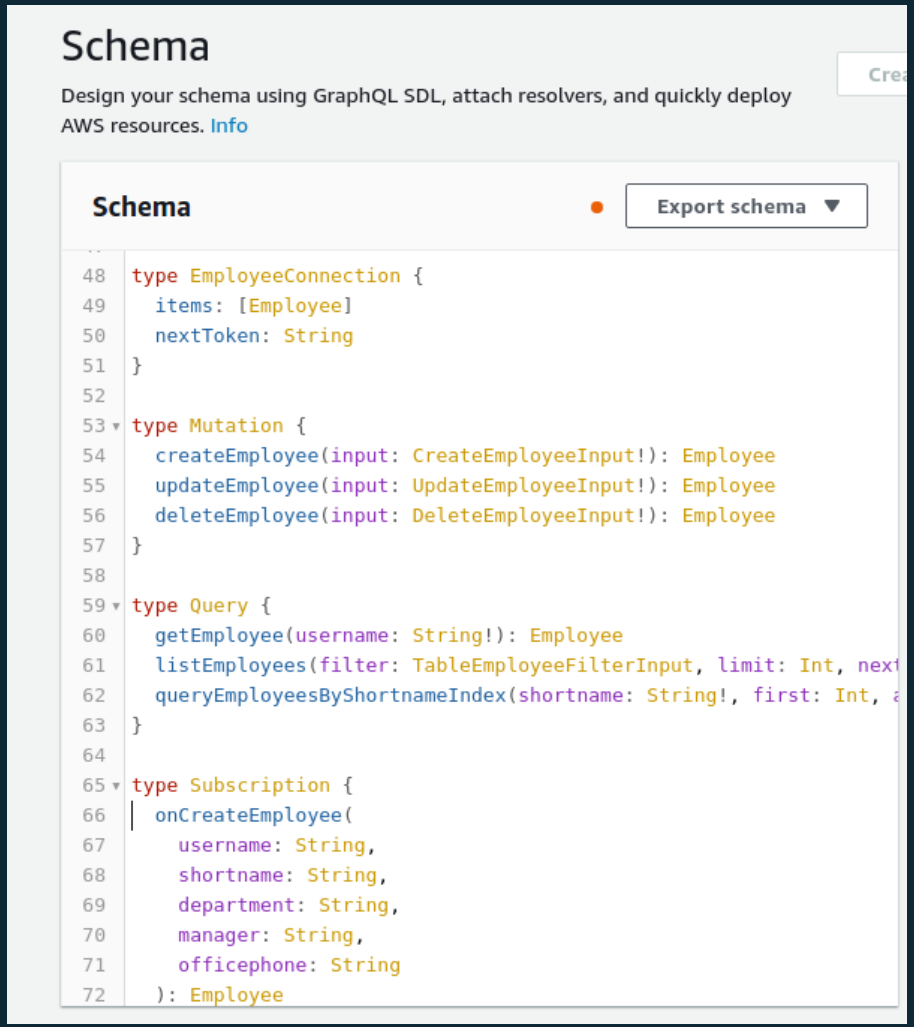

AWS AppSync
Event-Demo-App
Data Sources
New
New Data Source
Create new Data Source
Data source name
Type a name for your data source
A name starts with letter and contains only numbers, letters and "_"
Data source type
Select the type of your data source.
Amazon DynamoDB table
Amazon Elasticsearch domain
AWS Lambda function
HTTP endpoint
None
Cancel
Create

DynamoDB table as data source without Lambda



GraphQL schema automatically created from existing DynamoDB table data!

Or you can write your Schema first and click the “Create Resources” button.



WebGate

aws

Another schema example

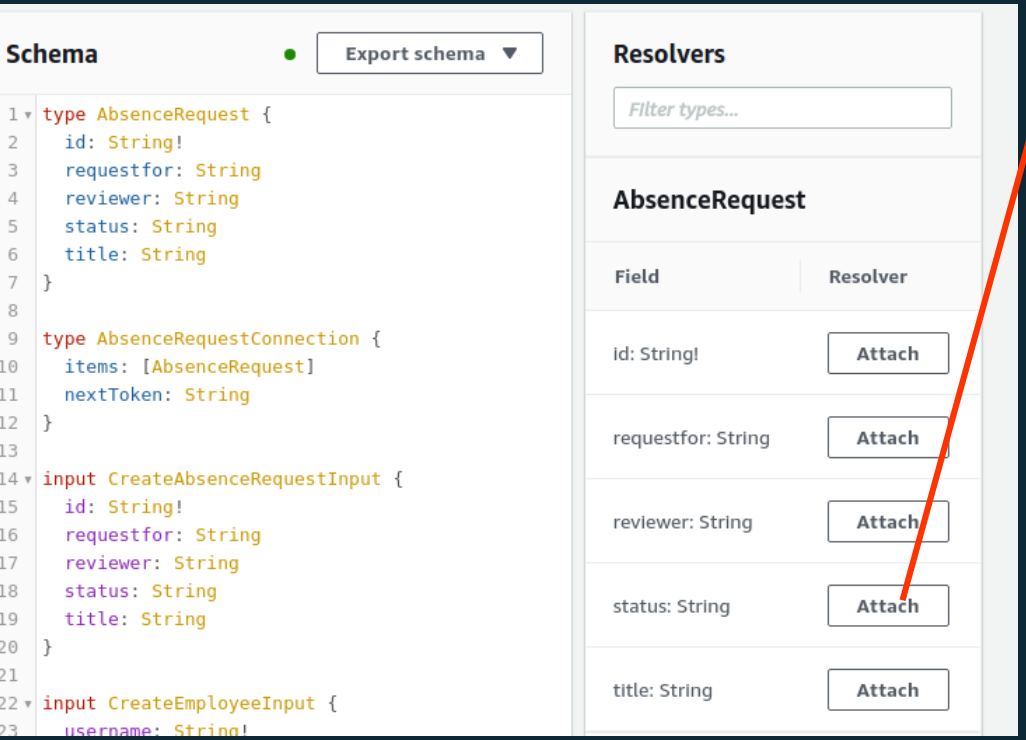
Access control

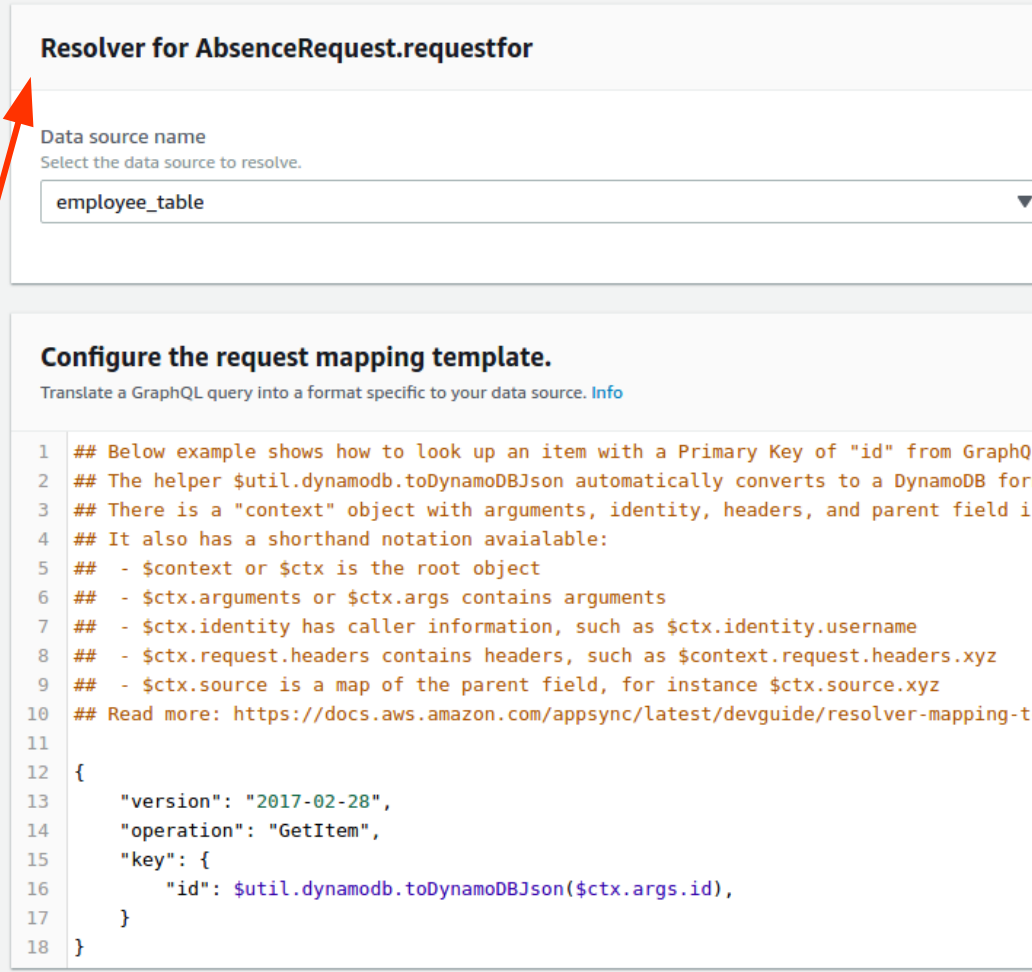
Mandatory fields

WebGate

```
schema {
  query: Query
  mutation: Mutation
}

type Mutation {
  # In this example, only users in the ManagerGroup can create tasks
  createTask(
    owner: String!,
    title: String!,
    taskStatus: String!,
    description: String!
  ): Task
    @aws_auth(cognito_groups: ["ManagerGroup"])
  # Both Employees and Managers can update a task's status
  updateTaskStatus(id: ID!, taskStatus: String!): Task
    @aws_auth(cognito_groups: ["EmployeeGroup","ManagerGroup"])
  updateTaskBody(id: ID!, title: String!, description: String!): Task
    @aws_auth(cognito_groups: ["ManagerGroup"])
}

type Query {
  # Users belonging to both EmployeesGroup and ManagerGroup can read a particular task
  getTask(id: ID!): Task
    @aws_auth(cognito_groups: ["EmployeeGroup","ManagerGroup"])
  # Only Managers can list all the Tasks
  allTasks(nextToken: String): TaskConnection
    @aws_auth(cognito_groups: ["ManagerGroup"])
}

type Task {
  id: ID!
  owner: String!
  title: String!
  description: String!
  taskStatus: String
}

type TaskConnection {
  items: [Task]
  nextToken: String
}
```

Attach resolvers to create relations





WebGate

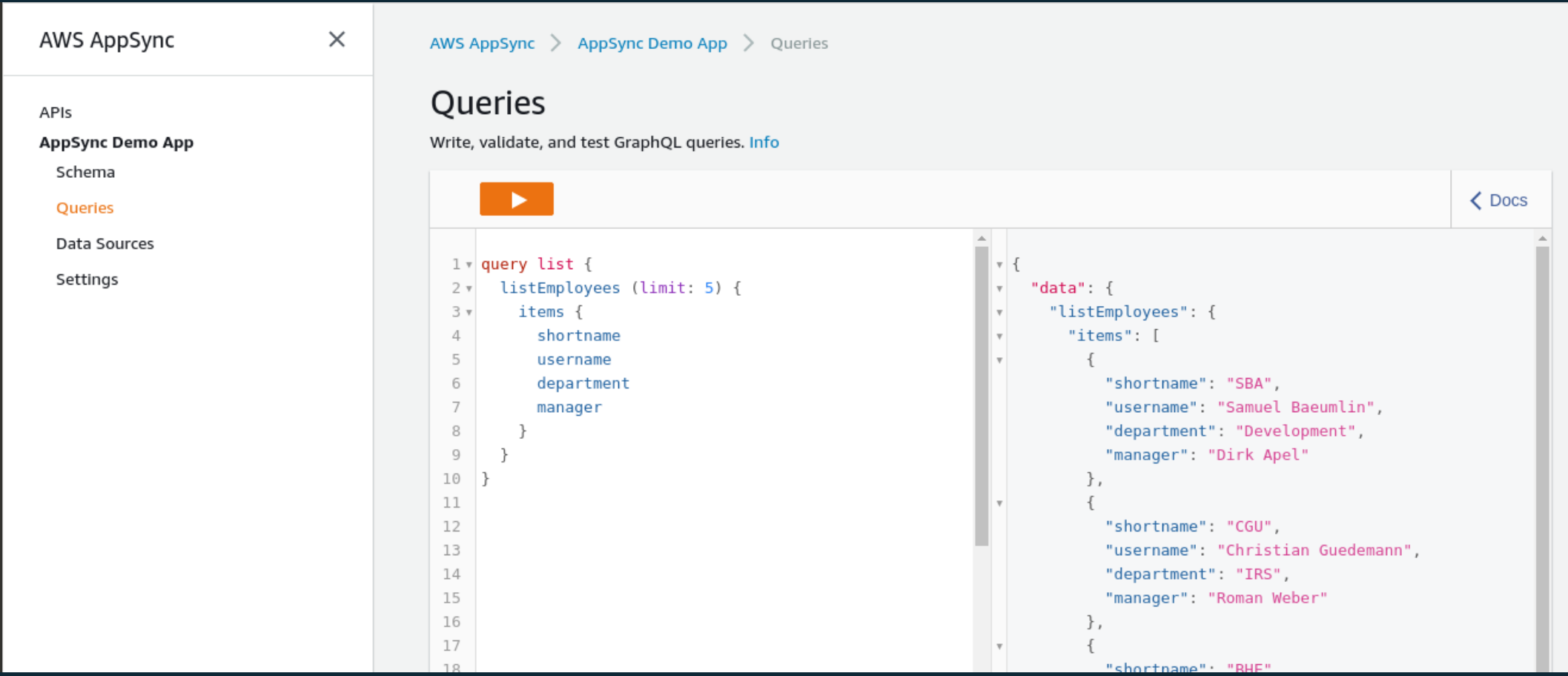
ACL using Resolver Mapping Templates

Apache Velocity Template Language (VTL)

Allow owner of document to read (based on field value in dynamoDB)

```
#if($context.result["Owner"] == $context.identity.username)
    $utils.toJson($context.result);
#else
    $utils.unauthorized()
#end
```

Query testing tool



Code samples: Query by ID

```jsx
import React, { Component } from "react";
import { graphql } from "react-apollo";
import QueryGetEvent from "../GraphQL/QueryGetEvent";

const ViewEventWithData = graphql(
    QueryGetEvent,
    {
        options: ({ match: { params: { id } } }) => ({
            variables: { id },
            fetchPolicy: 'cache-and-network',
        }),
        props: ({ data: { getEvent: event, loading} }) => ({
            event,
            loading,
        }),
    },
)(ViewEvent);

class ViewEvent extends Component {

    render() {
        const { event, loading } = this.props;
```

JS QueryGetEvent.js ×

```graphql
import gql from "graphql-tag";

export default gql(`
query($id: ID!) {
  getEvent(id: $id) {
    id
    name
    where
    when
    description
    comments {
      __typename
      items {
        commentId
        content
        createdAt
      }
    }
  }
}`);
```

WebGate

aws

Automatic conflict resolution in the cloud

Useful when using offline mode

- Client wins
- Server wins
- Silent reject
- Custom Lambda Fn

AppSync vs. API-GW+Lambda

- Almost no backend coding required
- Easy to implement access control, even on field level
- Build object relations, without coding
- Object related definition, type-save programming
- Real time subscriptions to events
- Offline support incl. conflict resolution (useful for mobile webapps)
- Perfect for CRUD. API-GW+Lambda for complex tasks.

Pricing

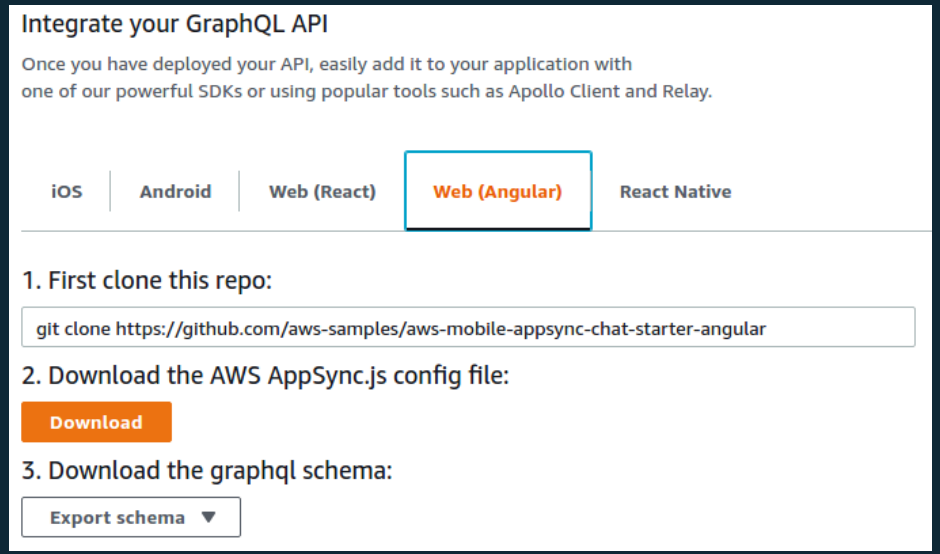
Query and Data Modification Operations

- AppSync: $4 per million Query and Data Modification Operations

  (API-GW: $3.50 per million API calls received, plus the cost of data transfer out, plus lambda execution time!)

Real-time Updates

- $2 per million Real-time Updates
- $0.08 per million minutes of connection to the AWS AppSync service



Explore the demos!

Angular2 Demo: https://github.com/aws-samples/aws-mobile-appsync-chat-starter-angular
React Demo: https://github.com/aws-samples/aws-mobile-appsync-events-starter-react

BTW: You can also use NodeJS to connect to AppSync!

Documentation: https://docs.aws.amazon.com/appsync/latest/devguide/welcome.html

Thanks for attending this AWSome Day session

Join the AWS Swiss User Group on meetup.com
for more AWS technical sessions,
or follow our blog.

richard.schmid @ webgate.biz
blog.webgate.biz

Innovationen
Etablieren

Inhalte
Vitalisieren

Prozesse
Digitalisieren

Innovation ist bei uns nicht nur ein Buzzword

Inhalte vitalisieren, Kunden überzeugen

Wettbewerbsvorteile sichern und ausbauen