GraphQL with AppSync and Lambda

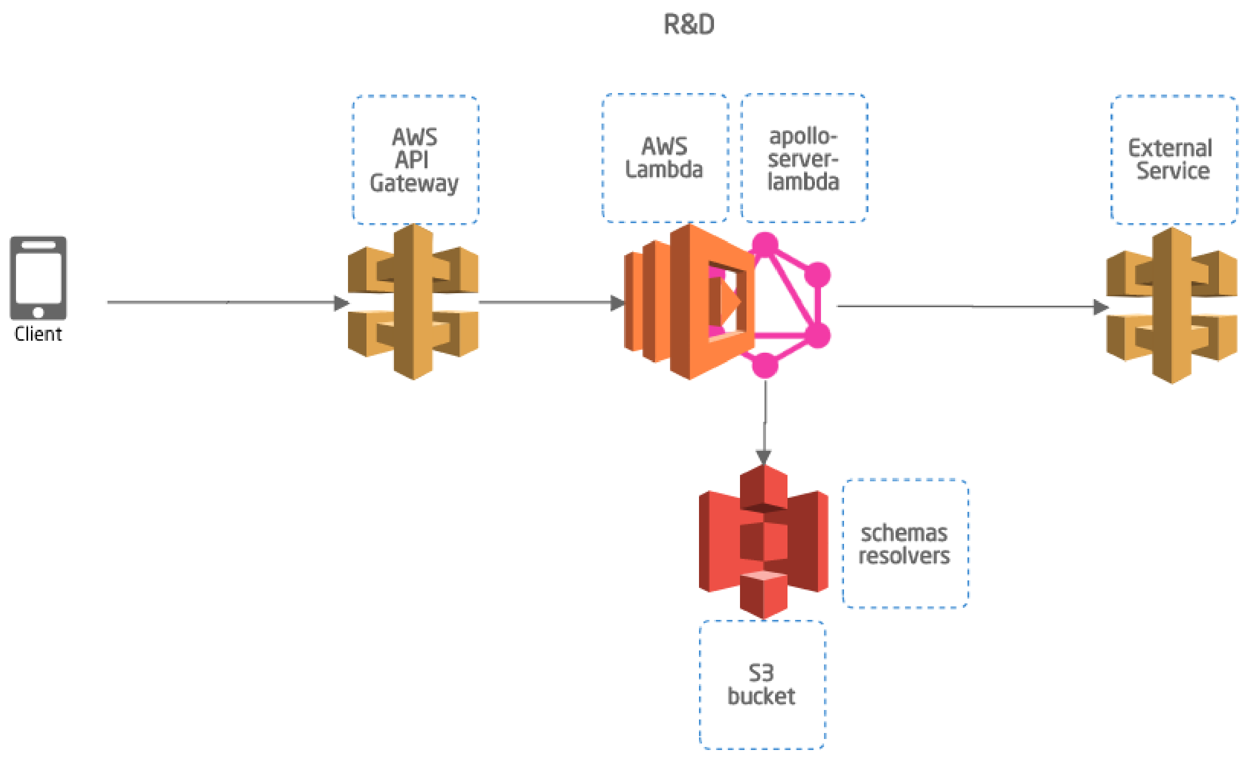
Introduction

As a part of the spike for GraphQL API for the Core Services the outcome of R&D project was looked at to see how the technology/architectural setup can fit into production level integration for the Core Services.

Internals

R&D

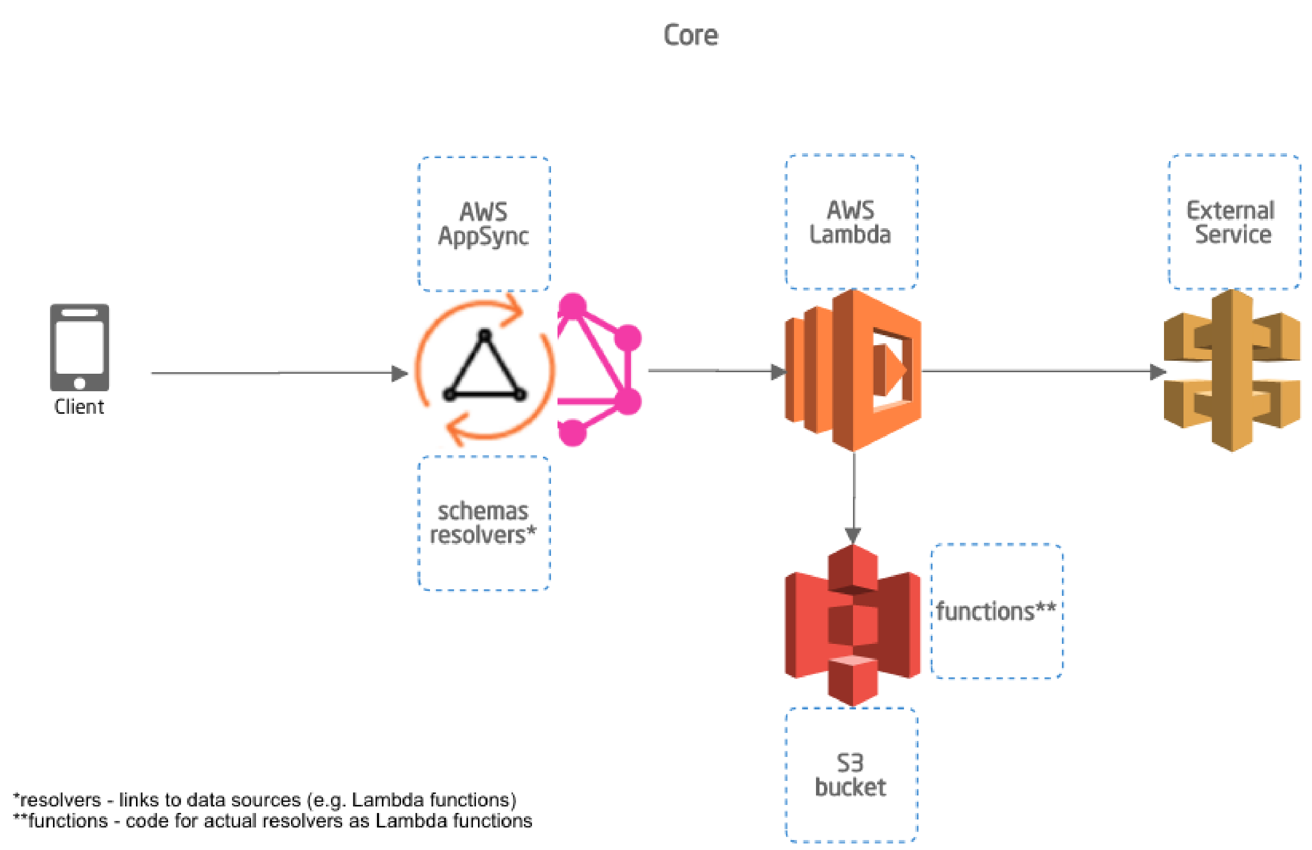
The setup that was used by the R&D team can be seen below.



In the flow of the setup presented above GraphQL requests were reaching AWS API Gateway and directing them as events to AWS Lambda. On the AWS Lambda side, apollo-server-lambda was used to run Apollo GraphQL server from within AWS Lambda for the time of the execution of the function. The code with GraphQL *schema* and *resolvers* was stored in S3 bucket and loaded via the Serverless framework.

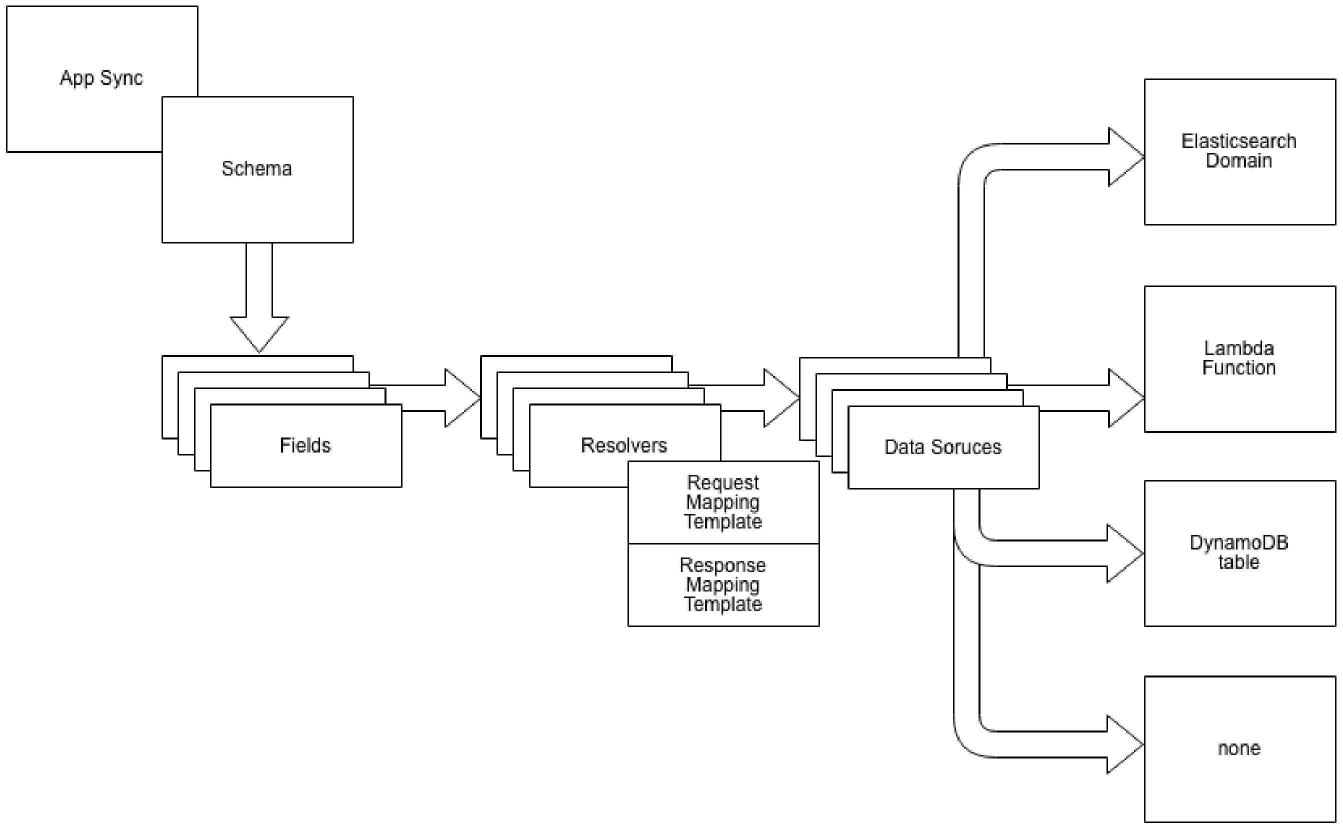
Core

At the time R&D was developed Amazon have released AWS AppSync which is Amazon configurable service providing an easy way to design, create and expose GraphQL APIs. AWS AppSync can replace apollo-server-lambda, simplifies the way API is designed and extends the list of sources that can be used to feed the API.


Core
AWS AppSync
AWS Lambda
External Service
Client
schemas resolvers*
functions**
S3 bucket
*resolvers - links to data sources (e.g. Lambda functions)
**functions - code for actual resolvers as Lambda functions

AppSync and Lambda

On the internals, AppSync is loaded with a GraphQL schema. Schema with its fields is tied to resolvers which then via Request and Response Mapping Templates are pointing to and from the data sources. The data source in AWS AppSync can point to an Elasticsearch domain, DynamoDB table or a Lambda function. The idea for Core Services is to use AWS Lambda functions to make HTTP requests to external (core) services, however GraphQL schema is defined on the AppSync side as opposed to on the Lambda function side. This is a different setup in comparison to what was used during the R&D.



Sample Code

GraphQL schema

```
type Query {
	authentication(tenant: String!, username: String!, password: 
String!): Token
}

type Token {
	token: String!
	expiresAt: String!
	createdAt: String!
	userUuid: String!
}

schema {
	query: Query
}
```

Request resolver

```
{
    "version" : "2017-02-28",
    "operation": "Invoke",
    "payload": $util.toJson($context.arguments)
}
```

Response resolver

```
$context.result
```

Lambda function

```
const https = require('https');

exports.handler = (event, context, callback) => {
     const data = {
        username : event.username,
        password : event.password
    };

    const options = {
        hostname: 'snapshot-ias.product.mttnow.com',
        port: 443,
        path: '/ias/authentication',
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'X-MTT-Tenant-ID': event.tenant
        }
    };

    const req = https.request(options, (res) => {
        let body = '';
        console.log('Status:', res.statusCode);
        console.log('Headers:', JSON.stringify(res.headers));
        res.setEncoding('utf8');
        res.on('data', (chunk) => body += chunk);
        res.on('end', () => {
            console.log('Successfully processed HTTPS response');
            if (res.headers['content-type'] === 'application/json') {
                body = JSON.parse(body);
            }
            callback(null, body);
        });
    });
    req.on('error', callback);
    req.write(JSON.stringify(data));
    req.end();
};
```

What’s next?

The next steps should answer the questions around automation, deployment and testing:

Open Questions

Links

AWS Services

AWS API Gateway

AWS AppSync

AWS Lambda

AWS Cloudformation

AWS Serverless Application Model

Other useful links

GraphQL

GraphQL and AppSync introduction

Serverless Framework

Serverless AppSync Plugin

AWS SAM HOWTO

Testing Lambda via SAM CLI

Deploying Lambda

Deploying AppSync via Cloudformation