# Serverless Task Chaining

for Qlik Sense Enterprise SaaS
Using AWS Lambda

Johan Bäcklin
Principal Enterprise Architect

LEAD WITH DATA™   Qlik Q

# TABLE OF CONTENTS

## SUMMARY

- Task chaining can be accomplished in different ways, for example <u>Task Chaining with qlik-cli</u>.

- This paper will showcase how to configure chained reload tasks in Qlik Sense Enterprise SaaS without the need of an additional server.

- Using AWS Lambda function but other technologies can be used.

- Schedule/reload any app in a chain through the SaaS Hub and rest of the chain will execute as defined.

## INTRODUCTION

Task chaining is useful both when there is a need to handle layered ETL or when several apps should be reloaded based on the same data sources i.e., start reloads in succession or trigger simultaneously reloads. In some cases, it is also relevant to reload different apps based on the reload result of another task i.e., if app1 succeed, reload app2a, else reload app2b.

The required code to accomplish this is quite simple and lightweight and can thus fit into a serverless architecture, alleviating the need of hosting and maintaining an additional server.

One such technology is AWS Lambda functions, used in this example.

This paper is not meant as an AWS Lambda walkthrough but will cover the necessary points to set it up.

# Qlik API key

To authorize the request from the Lambda function to Qlik we need an API key. If this is unfamiliar; please see <u>Generate your first API key</u>.

# Associated costs

At the time of writing this paper, AWS still offer a free usage tier for Lambda which includes 1M free requests per month and 400,000 GB-seconds of compute time per month. For current pricing information please see <u>AWS Lambda - Pricing</u>.

# Lambda function

This example is built on a plain Lambda function authored from scratch running Node.js 14.x. Besides the default index.js file, two more files are used to hold the definition of the app chains and the secrets. Secrets could optionally be stored in <u>AWS Secrets Manager</u> to mask them.

### Example apps.js

```
module.exports = [
{ name:'test app1',appId:'b4a3cf85-5dd8-42f7-a864-
3b2460215f8b',chain:['test app2','test app3'],condition:'ok'},
{ name:'test app2',appId:'5f74af07-ea2f-4398-b15f-23b12abf8fef'},
{ name:'test app3',appId:'6b57a80a-d9c4-4a0d-a5e0-f124dc68396e',chain:['test app4'],condition:'ok'},
{ name:'test app4',appId:'9ffe941d-8eb9-42e5-bab2-72bac5ca4b59',chain:['test app5'],condition:'ok'},
{ name:'test app5',appId:'163ca7fb-a684-4a4d-8d1c-37e3e372e1cc'},
{ name:'2nd chain1',appId:'cdaccf05-236f-4ab6-b77a-25138ba0df2f',chain:['2nd chain2'],condition:'ok'},
{ name:'2nd chain2',appId:'9f824b21-d089-4e59-8596-ad1a0d483f3f',chain:['2nd chain3a'],condition:'ok'},
{ name:'2nd chain2',appId:'9f824b21-d089-4e59-8596-ad1a0d483f3f',chain:['2nd chain3b'],condition:'error'},
{ name:'2nd chain3a',appId:'a11e3570-2dc9-4583-8417-d48bdd18602b'},
{ name:'2nd chain3b',appId:'db086243-b615-4fbe-8983-2832284670db'},
];
```

### Example secrets.js

```
module.exports = [
{ name:'qcs',data:[
    {tenant:'jbi.eu.qlikcloud.com',token:'eyJhbGciOiJFUzM4NCIsIm...'}
]}
];
```

## Example index.js

```javascript
const secrets = require('./secrets.js');
const apps = require('./apps.js');
const https = require('https');
async function reload(obj){
  return new Promise(async (res, rej) => {
    var options = {
        hostname: secrets.find((e) => { return e.name === 'qcs'; }).data[0].tenant,
        path: '/api/v1/reloads',
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'Authorization':'Bearer '+secrets.find((e) => { return e.name === 'qcs'; }).data[0].token
        }
    };
    const req = https.request(options, (r) => {
      r.on('data', (d) => { res(JSON.parse(d))});
    });
    req.write(JSON.stringify({appId:obj}));
    req.end();
  });
}
exports.handler = async (event) => {
  try {
    if(event.httpMethod==='POST'){
      const data = JSON.parse(event.body).data;
      const app = apps.find((e) => { return e.appId === data.id && e.condition === data.status});
      if(!app)
        return { statusCode: 404, body: JSON.stringify({error:'This app is not in the list.'}) };
      const reloadApps = app.chain;
      if(reloadApps) {
        var res=[];
        for (let i = 0; i < reloadApps.length; i++)
            res.push(await reload(apps.find((e) => { return e.name === reloadApps[i] }).appId));
        return { statusCode: 200,body: JSON.stringify(res) };
      }
      else
        return { statusCode: 404, body: JSON.stringify({error:'No apps are triggered on this app.'}) };
    }
    else if(event.httpMethod==='GET')
      return { statusCode: 200, body: JSON.stringify(apps) };
  }
  catch(e){
    return { statusCode: 400, body: e.message };
  }
};
```

# Expose Lambda function

Once the Lambda function has been created, it needs to be reachable from Qlik SaaS in order for Qlik to make HTTP requests that trigger the functionality.

From the function overview it is possible to add triggers; this example exposes the function as a REST endpoint secured with an API key ("Create an API > REST API > Security: API key").

This API key will be added as a header when creating the webhook, to authorize the request from Qlik. Reference Using AWS Lambda with Amazon API Gateway.

# Webhook

Recently Qlik SaaS introduced webhooks for various events. More information about webhooks.

One of those events are App Reload Finished, which sends a POST to the desired endpoint, the Lambda function in this case. This POST contains useful information, amongst other, which app that was reloaded and the result of the reload. Thus, it is possible to build chains both based on failed and successful reloads.

Webhooks are added in the SaaS console (https://<tenant>. eu.qlikcloud.com/console/webhooks/).

The webhook should use the API key, added as a header with key: x-api-key and value: "the key from AWS". The URL and API key were automatically generated by API Gateway when adding a trigger to the Lambda function.

This information is available under API Gateway, found in the AWS console > Functional overview for the Lambda function.

Also tick the checkbox for App Reload Finished, meaning that a request will be sent to the Lambda function for each app reload that has finished (failed and successful a like). The Lambda function code will ignore all requests that do not match an appId present in the apps.js list.

# Visualize the task chains

Using the code above one can make a GET request using the Qlik REST connector which will return the apps and how they relate to each other. Combining this with the Qlik RESTful Reloads API one could produce an org chart like the one below. With Qlik SaaS it is also possible to monitor a chart in the hub, like the second image.
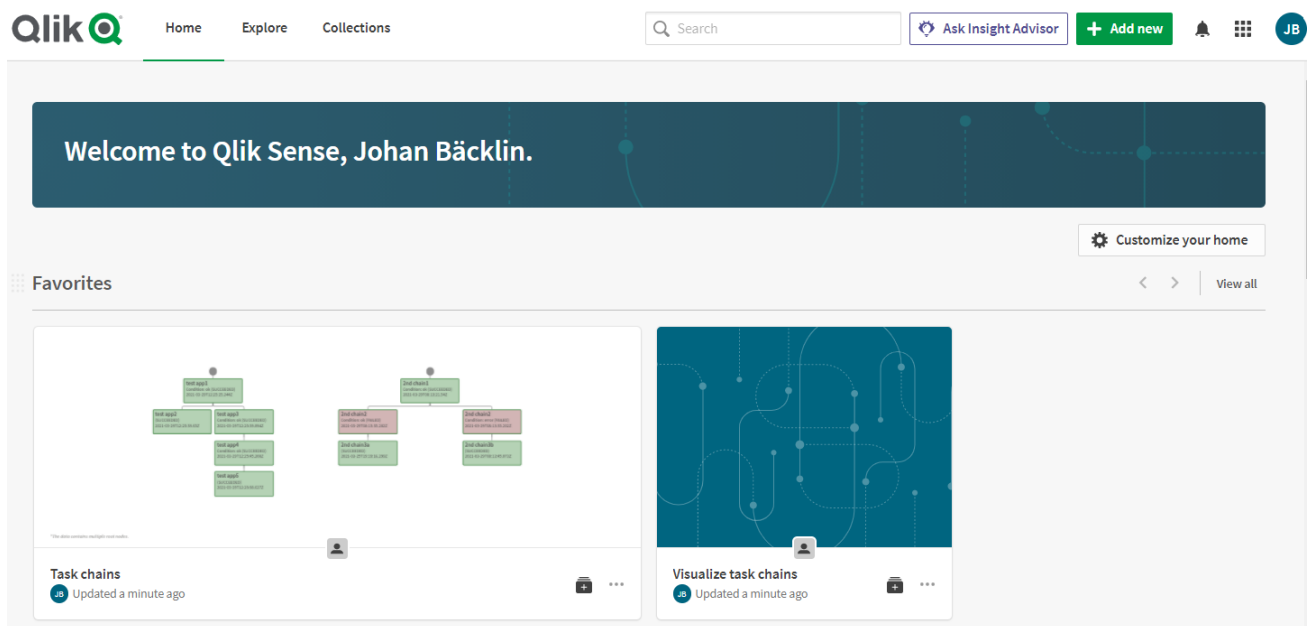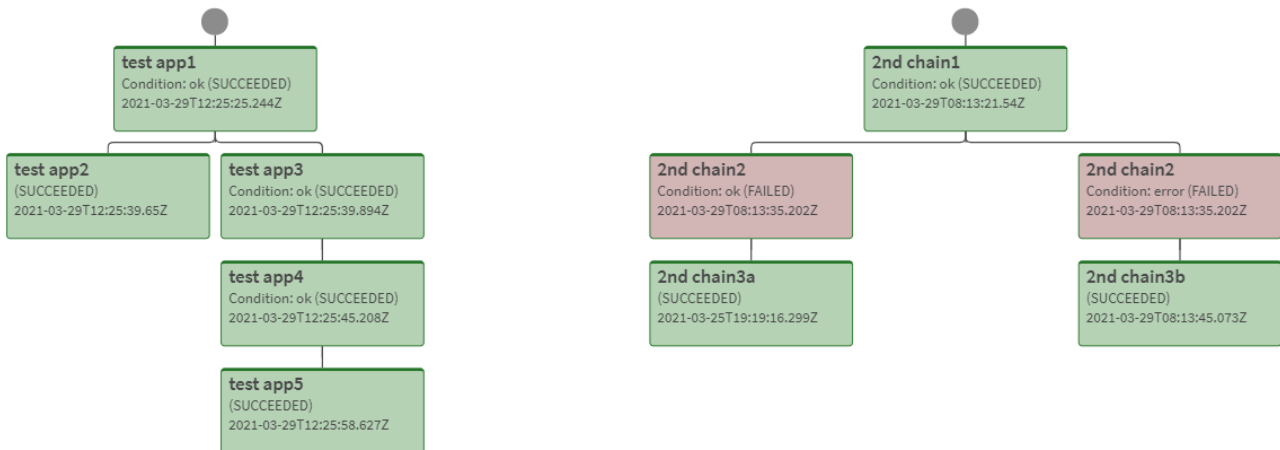




# Usage

- To start executing task chains simply modify the apps.js file with the appropriate appId's, references of which app should trigger on which and if it should trigger on successful ('ok') or failed ('error') reload. Compare the example apps.js with the picture above for reference. Schedule/reload any app in a chain through the SaaS Hub (or the Reloads API) and rest of the chain will execute as defined in apps.js.

**Qlik Q** LEAD WITH DATA™

### About Qlik

Qlik's vision is a data-literate world, where everyone can use data and analytics to improve decision-making and solve their most challenging problems. Qlik provides an end-to-end, real-time data integration and analytics cloud platform to close the gaps between data, insights and action. By transforming data into active intelligence, businesses can drive better decisions, improve revenue and profitability, and optimize customer relationships. Qlik does business in more than 100 countries and serves over 50,000 customers around the world.

**qlik.com**