Serverless Java Applications in AWS-Cloud

with S3, Lambda, API Gateway, SNS, DynamoDB and Aurora Serverless

by Elmar Warken and Vadym Kazulkin, ip.labs GmbH

Contact

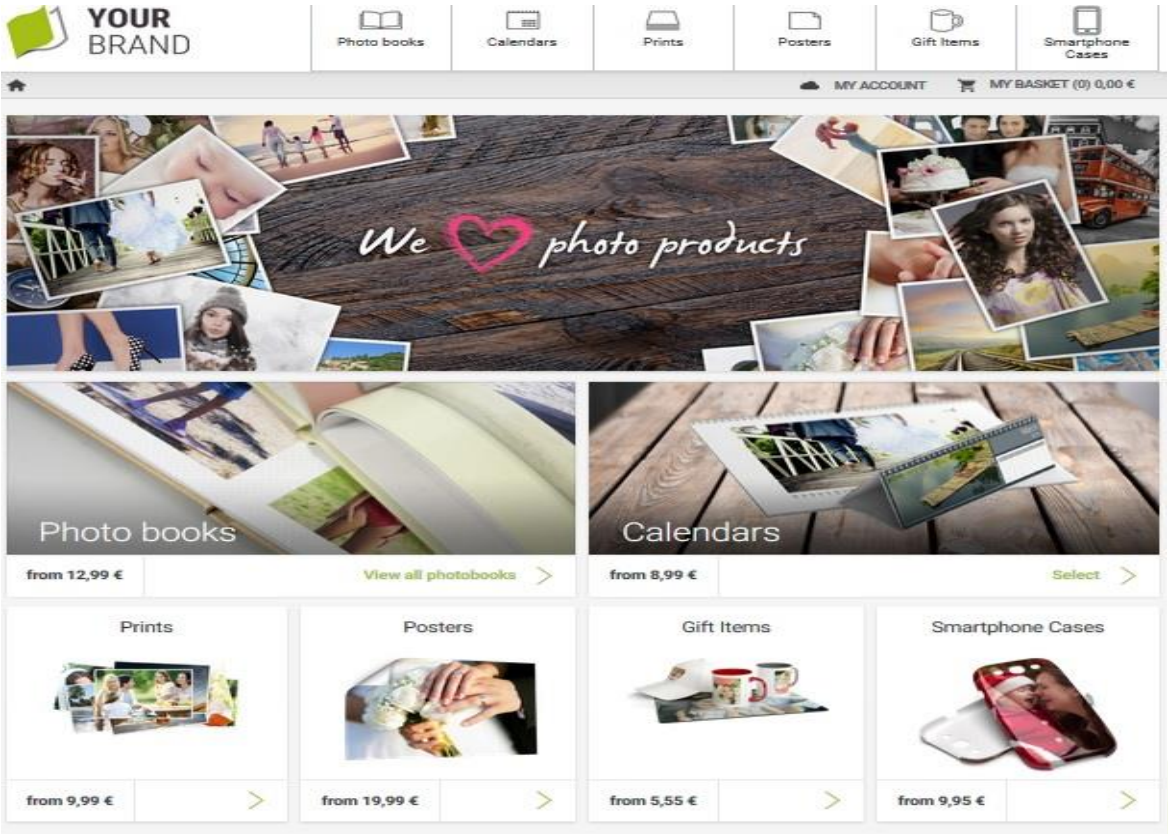
Vadym Kazulkin, ip.labs GmbH

v.kazulkin@iplabs.de
www.xing.com/profile/Vadym_Kazulkin
@VKazulkin

Elmar Warken, ip.labs GmbH

e.warken@iplabs.de
www.xing.com/profile/Elmar_Warken

ip.labs GmbH


YOUR BRAND
Photo books
Calendars
Prints
Posters
Gift Items
Smartphone Cases
MY ACCOUNT
MY BASKET (0) 0,00 €
We photo products
Photo books
from 12,99 €
View all photobooks
Calendars
from 8,99 €
Select
Prints
from 9,99 €
Posters
from 19,99 €
Gift Items
from 5,55 €
Smartphone Cases
from 9,95 €

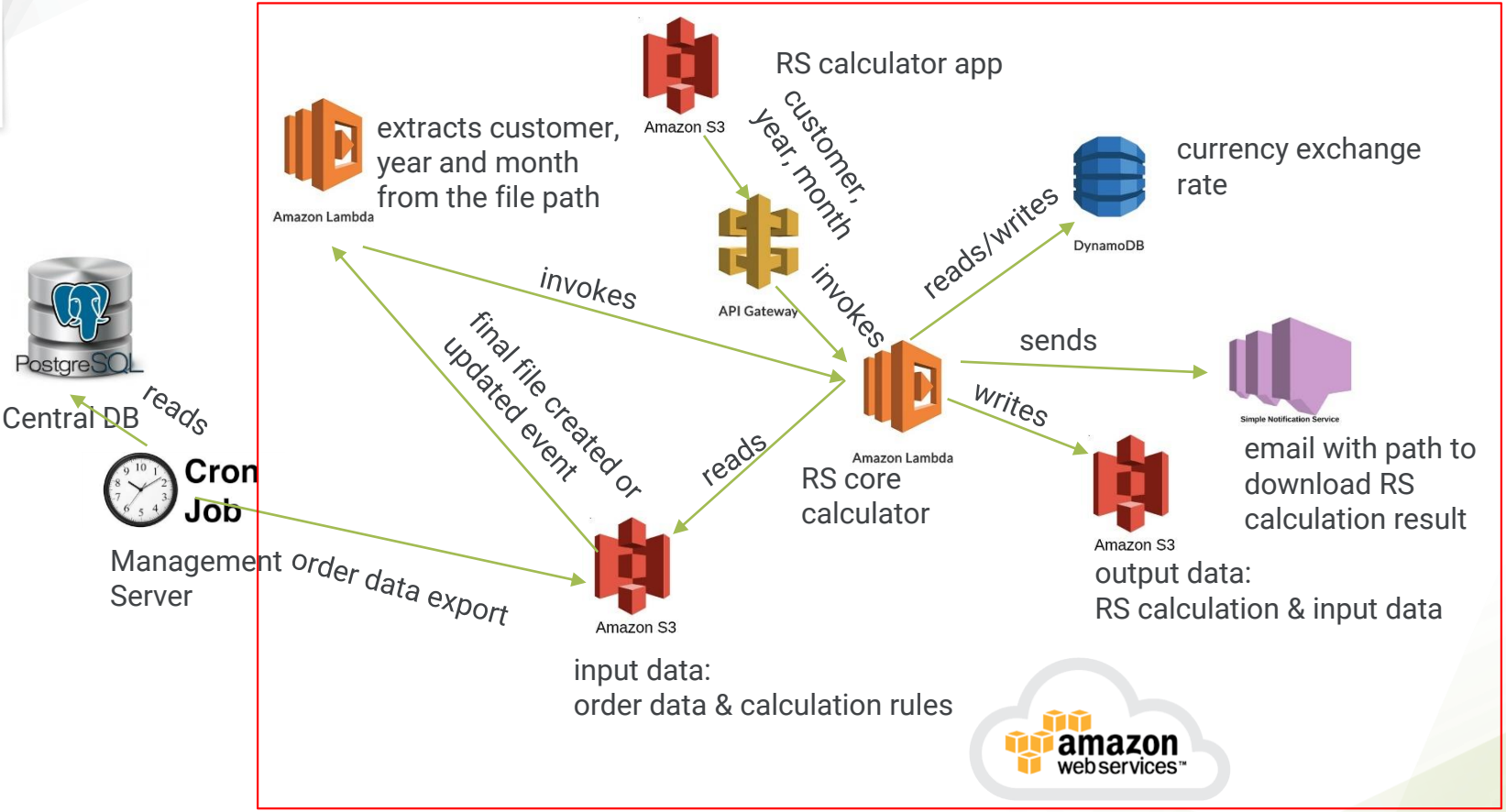
Revenue Share Project

Revenue Share Tool Requirements

1. Automatic calculation for the complete previous month on the 1st of each month for each customer

2. Calculation via a simple GUI for each customer
   - For the current month so far
   - Recalculation for the previous months

3. Notification with the link to download the calculation results for each customer for
   - Automatic calculation
   - Calculation via GUI



Key example: revsharetool-storage-demo/input-data/1000000/import/201808/revshare-final.json

What is Serverless ?

What is Serverless ?

- No servers to provision or manage
- Never pay for idle
- Scales with its usage
- Availability and fault tolerance built in

Amazon
S3

S3 = Simple Storage Service

Storage Services in the cloud

- unlimited storage, no minimum fee / reservation costs
- automatic data replication
- copies in multiple data centers
  - availability of 99% – 99.99%
  - durability: 99.999999999% per year (1 of 100 billion objects lost)
- storage classes for different access frequencies

Simple Storage Service – costs vs. EFS

as of June 2018	**S3 Standard Storage Class**	**EFS**
cost / GB * month	0,023 USD	0,30-0,36 USD

S3 costs can be reduced further:

- quantity discount: 0,021 USD from 500 TB
- less expensive storage classes:
  - S3 standard infrequent access: 0,0125 USD + higher traffic costs
  - one zone: 0,01 USD + higher traffic costs
  - Glacier: 0,004 USD

S3 structure

Buckets with globally unique names

- **Key-value-store**
- Keys can be used to map a directory structure
- files are called objects
- Objects can be **versioned**
- each object has a set of **metadata** (name-value pairs)

S3 Management console: bucket contents

Amazon S3 > revsharetool-storage-demo / input-data / 1000000 / import

Overview

Type a prefix and press Enter to search. Press ESC to clear.

Upload | Create folder | More

US East (N. Virginia)

Viewing 1 to 3

Name	Last modified	Size	Storage class
201805	--	--	--
201806	--	--	--
201807	--	--	--

Viewing 1 to 3

RevShare demo permissions

Amazon S3 > revsharetool-storage-demo

Overview | Properties | Permissions | Management

Access Control List | Bucket Policy | CORS configuration

Bucket policy editor ARN: arn:aws:s3:::revsharetool-storage-demo

Type to add a new policy or edit an existing policy in the text area below.

Delete | Cancel | Save

```
1  {
2      "Version": "2012-10-17",
3      "Id": "RevShareBucketAccessPolicy",
4      "Statement": [
5          {
6              "Sid": "All on objects in bucket lambda",
7              "Effect": "Allow",
8              "Principal": {
9                  "AWS": "arn:aws:iam::598473618090:role/revshare-lambda-demo"
10             },
11             "Action": "s3:*",
12             "Resource": "arn:aws:s3:::revsharetool-storage-demo/*"
13         },
14         {
15             "Sid": "All on bucket by lambda",
16             "Effect": "Allow",
17             "Principal": {
18                 "AWS": "arn:aws:iam::598473618090:role/revshare-lambda-demo"
19             },
20             "Action": "s3:*",
21             "Resource": "arn:aws:s3:::revsharetool-storage-demo"
22         },
23         {
24             "Sid": "AllowIPmix",
```

Using S3 Java SDK (1)

```
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;

 * The lambda's main class.
public class RevShareToolFunctionHandler implements RequestHandler<RevShareToolRequest, RevShareToolResponse> {

    private final static Regions REGION = Regions.US_EAST_1;

    public final static AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
            .withRegion(REGION)
//          .withCredentials(new ProfileCredentialsProvider())
            .build();
```

Using S3 Java SDK (2)

```java
/**
 * Builds an S3 key and stores a placeholder file with example text under this key.
 * In practice, the example text could be replaced by a ZIP file.
 */
public String putRevenueShareCalculationResultFileIntoS3() {
    final String key = buildS3KeyString(this.revShareCalculationResult.getRevShareToolRequest());
    this.feedbackService.addMessage("Output key: " + key);
    final AccessControlList bucketAcl = this.s3Client.getBucketAcl(REVENUE_SHARE_BUCKET_NAME);
    final InputStream value =
            new ByteArrayInputStream(revShareCalculationResult.getAsText().getBytes(StandardCharsets.UTF_8));

    PutObjectRequest putRequest = new PutObjectRequest(REVENUE_SHARE_BUCKET_NAME, key, value, null)
            .withAccessControlList(bucketAcl);

    this.s3Client.putObject(putRequest);
    return String.format(DOWNLOAD_FILE_URL_TEMPLATE, REVENUE_SHARE_BUCKET_NAME, key);
}

/**
 * @return example: "output-data/1000003/output/201701/1000003_201701_final.txt"
 */
private static String buildS3KeyString(RevShareToolRequest revShareToolRequest) {
    final StringBuilder sb = new StringBuilder();
    sb.append(OUTPUT_DATA_FOLDER).append("/")
      .append(revShareToolRequest.getCustomerId()).append("/")
```


AWS Lambda

Why Lambda ?

- **Cost**: Services with low traffic, or cron jobs, are cheaper with serverless. don’t pay for them when they’re idle

- **Scalability**: scale faster and more aggressively because not constrained by the amount of spare capacity you reserve (and have to pay for) for scaling

- **Resilience**: Multi-AZ out of the box with AWS Lambda

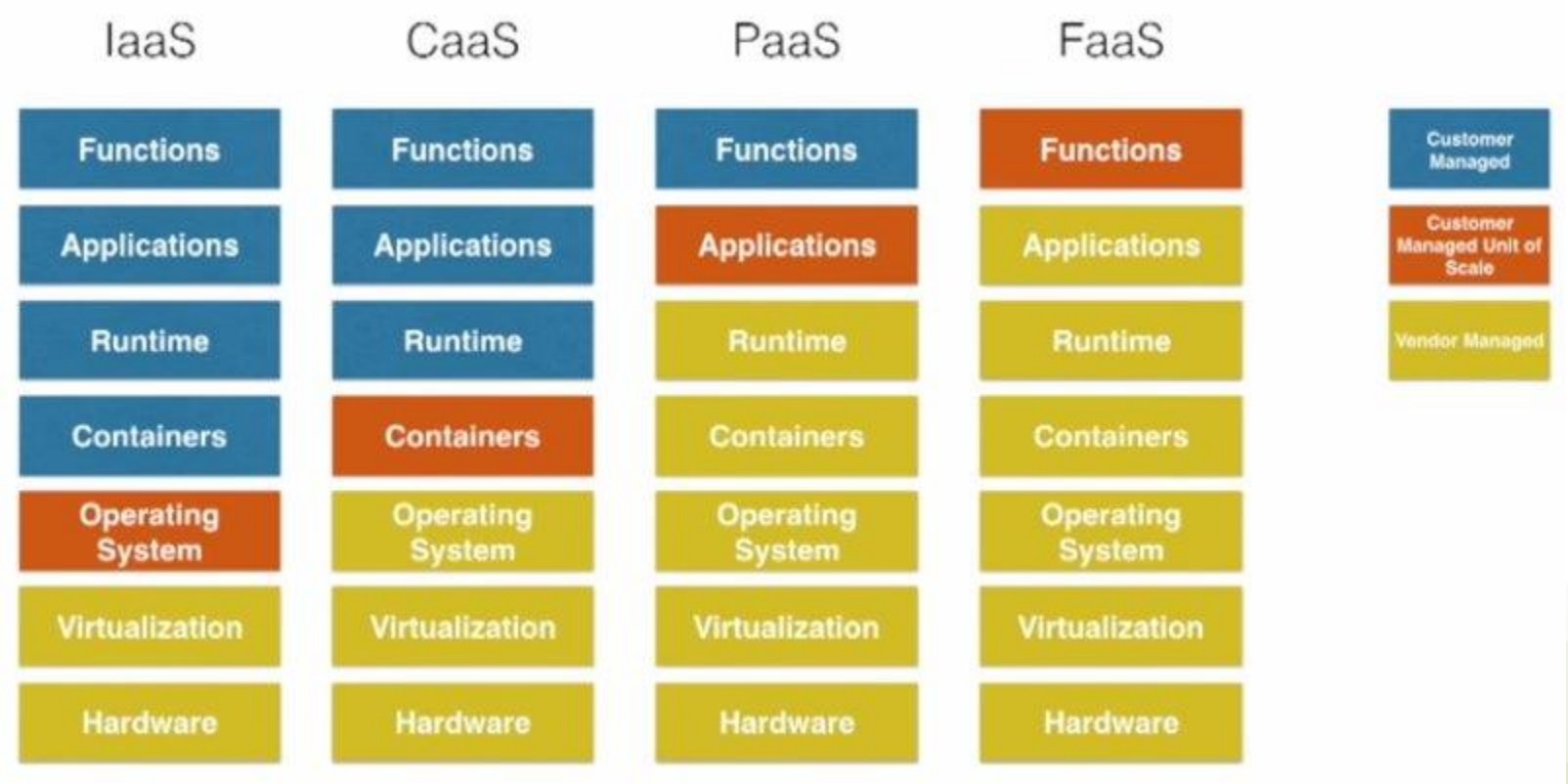
Why Lambda ?

- **Time to market**: get more done, faster. Free yourself of managing infrastructure. Focus on your users' needs instead

- **Easier OPs**: logging and monitoring out of the box. Integration with other services for tracing, alerting, and visualization

- **Security**: more fine-grained control of access. Give each function only the permissions it needs to minimize attack surface

IaaS vs CaaS vs FaaS



„Monthly Calculation Trigger“ Lambda creation

revshare-finaltrigger-demo

Drosselung | Qualifizierer ▼ | Aktionen ▼ | Testereignis auswählen ▼ | Test | Speichern

Codeeingabetyp

ZIP- oder JAR-Datei hochladen ▼

Funktionspaket*

Hochladen

Ziehen Sie bei Dateien, die größer als 10 MB sind, das Hochladen über S3

Laufzeit

Java 8 ▼

- C# (.NET Core 1.0)
- C# (.NET Core 2.0)
- Go 1.x
- Java 8
- Node.js 4.3
- Node.js 6.10
- Node.js 8.10
- Python 2.7
- Python 3.6

Handler Informationen

de.iplabs.de.revsharetool.finaltrigger.Mon

Umgebungsvariablen

Sie können Umgebungsvariablen als Schlüssel-Wert-Paare definieren, auf … nn. Diese Variablen sind nützlich, um Konfigurationseinstellungen zu speichern, ohne den Funktionscode ändern zu müssen. Weitere Informationen.

„Monthly Calculation Trigger“ Lambda creation

revshare-finaltrigger-demo

Drosselung | Qualifizierer ▼ | Aktionen ▼ | Testereignis auswählen ▼ | Test | Speichern

Ausführungsrolle

Definiert die Berechtigungen der Funktion. Beachten Sie, dass die neuen Rollen nach der Erstellung möglicherweise für einige Minuten nicht verfügbar sind. Weitere Informationen zu den Lambda-Ausführungsrollen.

Wählen Sie eine vorhandene Rolle aus ▼

Vorhandene Rolle
Sie können auch eine vorhandene Rolle mit dieser Funktion verwenden. Beachten Sie, dass die Rolle von Lambda annehmbar sein muss und über Cloudwatch Logs-Berechtigungen verfügt.

revshare-lambda-demo ▼

Grundlegende Einstellungen

Beschreibung

Arbeitsspeicher (MB) Informationen
Die Funktion wird dem konfigurierten Arbeitsspeicher CPU-proportional zugewiesen.

384 MB

Timeout Informationen

0 Min. 20 Sek.

Lambda Pricing

AWS Lambda | Übersicht | Funktionen | Preise | Erste Schritte | Ressourcen | Häufig gestellte Fragen | Partner

Arbeitsspeicher (MB)	Sekunden pro Monat im kostenlosen Kontingent	Preis pro 100 ms (USD)
128	3 200 000	0,000000208
192	2 133 333	0,000000313
256	1 600 000	0,000000417
320	1 280 000	0,000000521
384	1 066 667	0,000000625
448	914 286	0,000000729
512	800 000	0,000000834
576	711 111	0,000000938
640	640 000	0,000001042
704	581 818	0,000001146
768	533 333	0,000001250

Calling Lambda as S3 Event

Amazon S3 > revsharetool-storage-demo

Overview | Properties | Permissions | Management

Versioning

Keep multiple versions of an object in the same bucket.

Learn more

Disabled

Server access logging

Set up access log records that provide details about access requests.

Learn more

Disabled

Static website hosting

Host a static website, which does not require server-side technologies.

Learn more

Disabled

Object-level log

Record object-level API activit
CloudTrail data events feature
cost).

Learn more

Disabled

Advanced settings

Tags

Use tags to track your cost against projects or other criteria.

Learn more

0 Tags

Transfer acceleration

Enable fast, easy and secure transfers of files to and from your bucket.

Learn more

Suspended

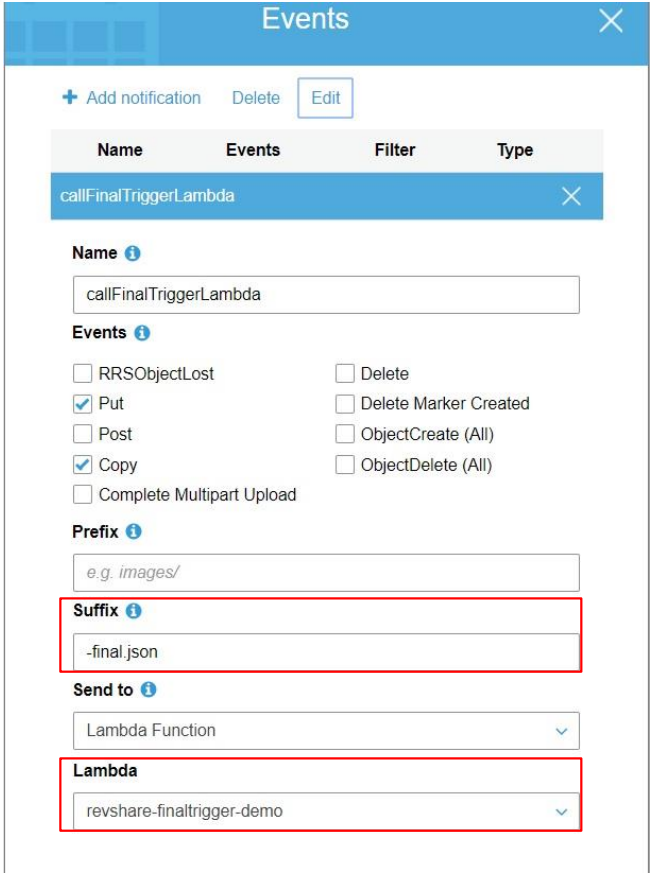
Events

+ Add notification | Delete | Edit

Name	Events	Filter	Type
callFinalTriggerLambda			

Name

Calling Lambda as S3 Event

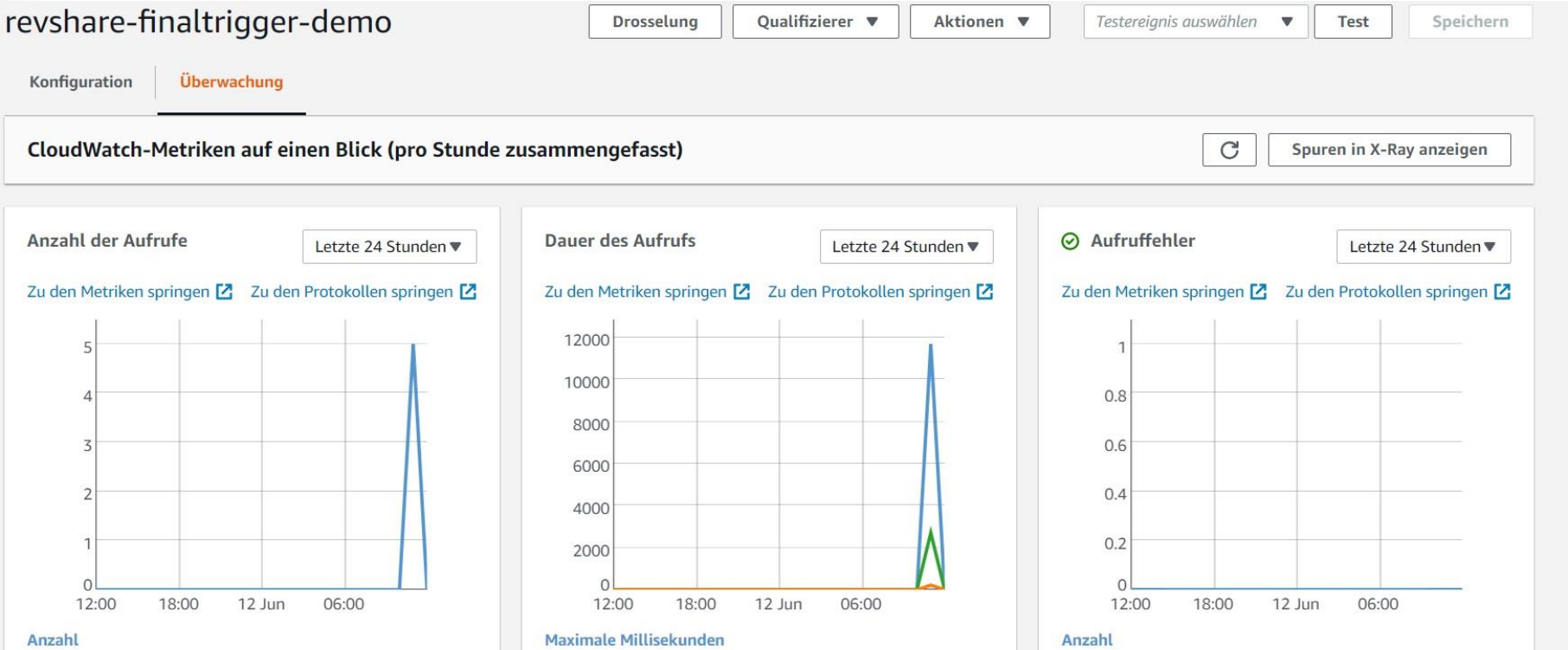


Implementing „Monthly Calculation Trigger“ Lambda with Java SDK

```java
public class MonthlyCalculationTrigger implements RequestHandler<S3Event, Void> {

    private static final String  REVENUSHARE_CALCULATION_LAMBDA= "revsharetool-calculator-demo";
    public MonthlyCalculationTrigger() {}

    @Override
    public Void handleRequest(S3Event event, Context context) {
        context.getLogger().log("Received event: " + event);

        final String key = event.getRecords().get(0).getS3().getObject().getKey();

        final RevShareToolRequest rsToolRequest= fromS3Event(key, context.getLogger());

        final ObjectMapper   mapper = new ObjectMapper();
        mapper.setVisibility(PropertyAccessor.FIELD, JsonAutoDetect.Visibility.ANY);
        mapper.configure(SerializationFeature.FAIL_ON_EMPTY_BEANS, false);

        String json=null;
        try {
            json = mapper.writeValueAsString(rsToolRequest);
        } catch (JsonProcessingException e) {
            throw new RuntimeException("can't serialize json ");
        }
        context.getLogger().log("JSON before build Async client: "+ json);

        final AWSLambda lambda= AWSLambdaClientBuilder.defaultClient();

        final InvokeRequest request = new InvokeRequest();
        request.withInvocationType(InvocationType.Event) // fire&forget
        .withFunctionName(REVENUSHARE_CALCULATION_LAMBDA)
        .withPayload(json);

        lambda.invoke(request);
        context.getLogger().log("rs calculation lambda invoked ");
        return null;
    }
```

Key example: revsharetool-storage-demo/input-data/1000000/import/201808/revshare-final.json

Lambda Monitoring



Lambda Cloud Watch Logging

CloudWatch
Dashboards
Alarms
ALARM 0
INSUFFICIENT 1
OK 0
Billing
Events
Rules
Event Buses
Logs
Metrics

Favorites
Add a dashboard

CloudWatch > Log Groups > /aws/lambda/revshare-finaltrigger-demo > 2018/06/12/[$LATEST]bfee6bfe811048daa63bf17def2d84b1

Expand all Row Text

Filter events all 2018-06-11 (09:28:03)

Time (UTC +00:00)	Message
2018-06-12	
09:16:24	START RequestId: 46a4ee23-6e21-11e8-9bc4-a5bc1d7b422e Version: $LATEST
09:16:24	Received event: com.amazonaws.services.lambda.runtime.events.S3Event@4e515669
09:16:24	customerId 1000000 year 2018 month 5
09:16:26	JSON before build Async client: {"customerId":1000000,"year":2018,"month":5}
09:16:31	created lambda client
09:16:35	rs calculation lambda invoked
09:16:35	END RequestId: 46a4ee23-6e21-11e8-9bc4-a5bc1d7b422e
09:16:35	REPORT RequestId: 46a4ee23-6e21-11e8-9bc4-a5bc1d7b422e Duration: 11682.26 ms Billed Duration: 11700 ms Memory Size: 384 MB Max Memory Used: 102 MB
09:16:45	START RequestId: 53df774e-6e21-11e8-90df-03f363aec285 Version: $LATEST
09:16:45	Received event: com.amazonaws.services.lambda.runtime.events.S3Event@754ba872
09:16:45	customerId 1000000 year 2018 month 5
09:16:45	JSON before build Async client: {"customerId":1000000,"year":2018,"month":5}
09:16:45	created lambda client
09:16:45	rs calculation lambda invoked
09:16:45	END RequestId: 53df774e-6e21-11e8-90df-03f363aec285
09:16:45	REPORT RequestId: 53df774e-6e21-11e8-90df-03f363aec285 Duration: 386.66 ms Billed Duration: 400 ms Memory Size: 384 MB Max Memory Used: 102 MB
09:21:41	START RequestId: 046c9b83-6e22-11e8-ac34-1de2b4c1b027 Version: $LATEST
09:21:41	Received event: com.amazonaws.services.lambda.runtime.events.S3Event@1f554b06
09:21:41	customerId 1000000 year 2018 month 5
09:21:41	JSON before build Async client: {"customerId":1000000,"year":2018,"month":5}
09:21:41	created lambda client

„Revenue Share Calculation“ Lambda Creation

revsharetool-calculator-demo

Drosselung | Qualifizierer ▼ | Aktionen ▼ | Testereignis auswählen ▼ | Test

Ausführungsrolle

Definiert die Berechtigungen der Funktion. Beachten Sie, dass die neuen Rollen nach der Erstellung möglicherweise für einige Minuten nicht verfügbar sind. Weitere Informationen zu den Lambda-Ausführungsrollen.

Wählen Sie eine vorhandene Rolle aus ▼

Vorhandene Rolle

Sie können auch eine vorhandene Rolle mit dieser Funktion verwenden. Beachten Sie, dass die Rolle von Lambda annehmbar sein muss und über Cloudwatch Logs-Berechtigungen verfügt.

revshare-lambda-demo ▼

Grundlegende Einstellungen

Beschreibung

Arbeitsspeicher (MB) Informationen

Die Funktion wird dem konfigurierten Arbeitsspeicher CPU-proportional zugewiesen.

768 MB

Timeout Informationen

1 Min. 0 Sek.

Implementing „Revenue Share Calculation“ Lambda with Java SDK

```java
public class RevShareToolFunctionHandler implements RequestHandler<RevShareToolRequest, RevShareToolResponse> {

    private final static Regions REGION= Regions.US_EAST_1;

    private static final BasicAWSCredentials AWS_CREDENTIALS = new BasicAWSCredentials(RevShareConstants.REVSHARE_USER_ACCESS_KE
            RevShareConstants.REVSHARE_USER_SECRET_KEY);

    private static final AWSCredentialsProvider AWS_CREDENTIALS_PROVIDER= new AWSStaticCredentialsProvider(AWS_CREDENTIALS);

    public final static AmazonS3 S3_CLIENT = AmazonS3ClientBuilder.standard().
            withRegion(REGION).build();

    public final static AmazonSNS SNS_CLLIENT = AmazonSNSClientBuilder.standard()
            .withCredentials(AWS_CREDENTIALS_PROVIDER)
            .withRegion(REGION).build();

    @Override
    public RevShareToolResponse handleRequest(RevShareToolRequest revShareToolRequest, Context context) {
        final RevShareCalculationService revShareCalculationService = new RevShareCalculationService(revShareToolRequest);
        final RevShareCalculationResult revShareCalculationResult=revShareCalculationService.calculate();

        final FeedbackService feedbackService = new FeedbackService(context.getLogger(),
                new SNSNotificationSender(RevShareConstants.SNS_TOPIC_ARN, SNS_CLLIENT));

        final RevShareResultWriterService writer=
                new RevShareResultWriterService(revShareCalculationResult, feedbackService, S3_CLIENT);

        final String downloadUrl=writer.putRevenueShareCalculationResultFileIntoS3();
        feedbackService.addMessage("download url :" +downloadUrl);

        final LocalDate calculationDate = revShareToolRequest.getCalculationDate();
        feedbackService.sendFeedback(downloadUrl, revShareToolRequest.getCustomerId(),
                calculationDate.getMonth() + " " + calculationDate.getYear());

        return new RevShareToolResponse(downloadUrl,feedbackService.getReport());
    }
```

Lambda limits

AWS Lambda Resource Limits per Invocation

Resource	Limits
Memory allocation range	Minimum = 128 MB / Maximum = 3008 MB (with 64 MB increments). If the maximum memory use is exceeded, function invocation will be terminated.
Ephemeral disk capacity ("/tmp" space)	512 MB
Number of file descriptors	1,024
Number of processes and threads (combined total)	1,024
Maximum execution duration per request	300 seconds (5 minutes)
Invoke request body payload size (RequestResponse/synchronous invocation) NOTE: The response body payload also must adhere to this limit.	6 MB
Invoke request body payload size (Event/asynchronous invocation)	128 KB

AWS Lambda Account Limits Per Region

Resource	Default Limit
Concurrent executions (see Managing Concurrency)	1000

Source: https://docs.aws.amazon.com/lambda/latest/dg/limits.html

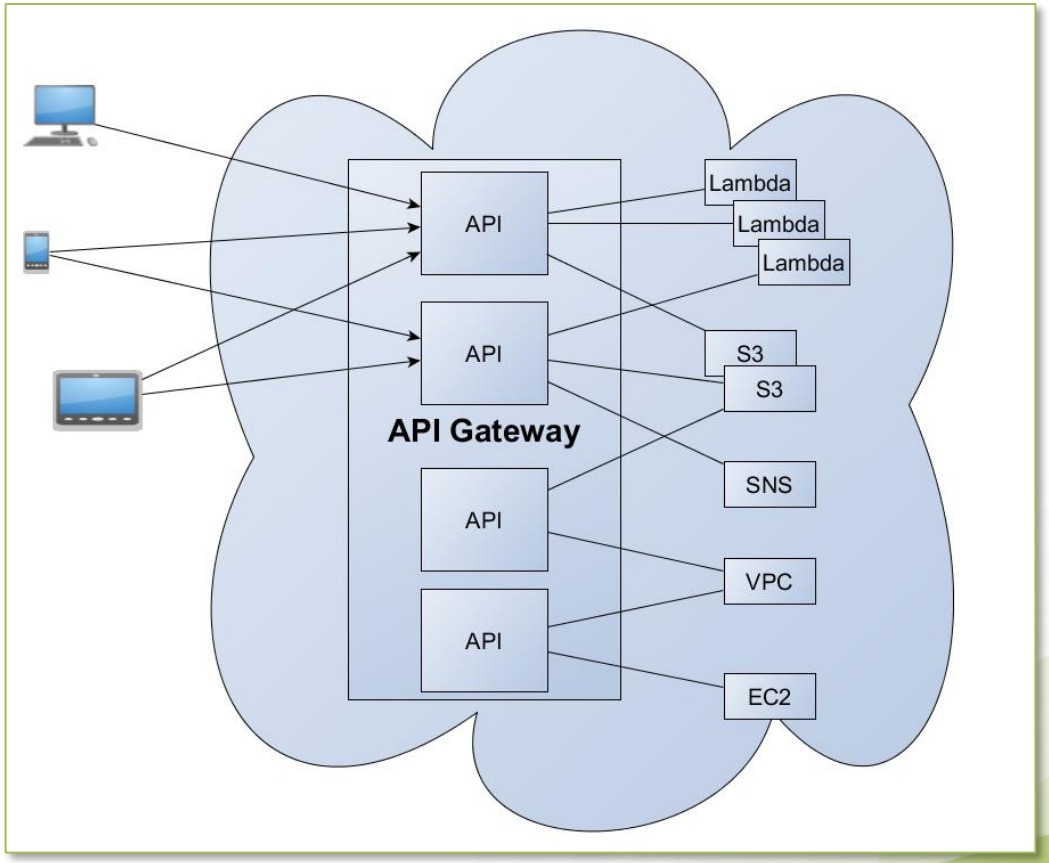
Amazon API Gateway

Features of API management platforms

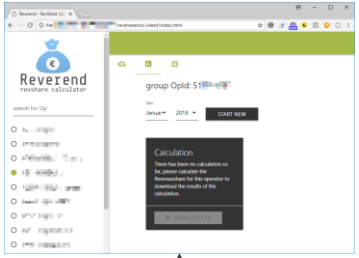
- Authorization and access control
  - IAM (Identity and Access Management: users, groups, roles)
  - Usage plans (throttling and quota)
- Monetize APIs (SaaS, AWS: Marketplace)
- Version management
  - some platforms even use git
- SDK generation
- Request monitoring and analytics
  - AWS: CloudTrail, CloudWatch

Amazon API Gateway

- Access different AWS services in a consistent manner



Calling a backend function via gateway



Application

https://pe2r5mu2n3.execute-api.us-east-1.amazonaws.com/Beta/calculate

API Gateway

Lambda function

- URL schema:
  https://***api-id***.execute-api.***region***.amazonaws.com/***stage***

AWS API Gateway: RevShare Demo

APIs > RevShareCalculatorApi (pe2r5mu2n3) > Resources > /calculate (ge89x6) > POST Show all hints

Resources Actions

- /
  - /calculate
    - POST
  - /calculateWithParams
    - /{customerId}
      - POST

/calculate - POST - Method Execution

Client

TES

Method Request

Auth: NONE

ARN: arn:aws:execute-api:us-east-1:598473618090:pe
/POST/calculate

Integration Request

Type: LAMBDA

Region: us-east-1

Lambda revsharetool-calculator-demo

Method Response

HTTP Status: 200

Models: application/json
=> Empty

Integration Response

HTTP status pattern:

-

Output passthrough:

JSON-Body

https://pe2r5mu2n3.execute-api.us-east-1.amazonaws.com/Demo/calculateWithParams/1000003?year=2017&month=1

API Method Request (1)

Amazon API Gateway | APIs > RevShareCalculatorApi (pe2r5mu2n3) > Resources > /calculate (ge89x6) > POST | Show all hints

- APIs
  - PetStore
  - RevShareCalculatorApi
    - Resources
    - Stages
    - Authorizers
    - Gateway Responses
    - Models
    - Resource Policy
    - Documentation
    - Dashboard
    - Settings
- Usage Plans
- API Keys
- Custom Domain Names
- Client Certificates
- VPC Links
- Settings

Resources | Actions

- /
  - /calculate
    - POST
  - /calculateWithParams
    - /{customerId}
      - POST

← Method Execution /calculate - POST - Method Request

Provide information about this method's authorization settings and the parameters it can receive.

Settings

Authorization	NONE
Request Validator	NONE
API Key Required	true

URL Query String Parameters

HTTP Request Headers

Request Body

Content type	Model name
application/json	RevShareCalculatorInputModel

Add model

SDK Settings

Models for JSON payload

Amazon API Gateway | APIs > RevShareCalculatorApi (pe2r5mu2n3) > Models > Create | Show all hints

- APIs
  - PetStore
  - RevShareCalculatorApi
    - Resources
    - Stages
    - Authorizers
    - Gateway Responses
    - **Models**
    - Resource Policy
    - Documentation
    - Dashboard
    - Settings
- Usage Plans
- API Keys
- Custom Domain Names
- Client Certificates
- VPC Links

Models — Create

- Empty
- Error

New Model

Provide a name, content type, and a schema for your model. Models use JSON schema.

Model name* RevShareCalculatorInputModel

Content type* application/json

Model description Model describing the JSON input of the Calculate function

Model schema*

```
1  {
2    "$schema": "http://json-schema.org/draft-04/schema#",
3    "title": "RevShareCalculatorInputModel",
4    "type": "object",
5    "properties": {
6      "customerId": { "type": "number" },
7      "year": { "type": "number" }
8      "month": { "type": "number" }
9    }
```

Expected ',' instead of '"'

* Required

Cancel | Create model

API Method Request (2)

Resources | Actions

- /
  - /calculate
    - POST
  - /calculateWithParams
    - /{customerId}
      - POST

Method Execution

/calculateWithParams/{customerId} - POST - Method Request

Provide information about this method's authorization settings and the parameters it can receive.

Settings

Authorization	NONE
Request Validator	NONE
API Key Required	true

Request Paths

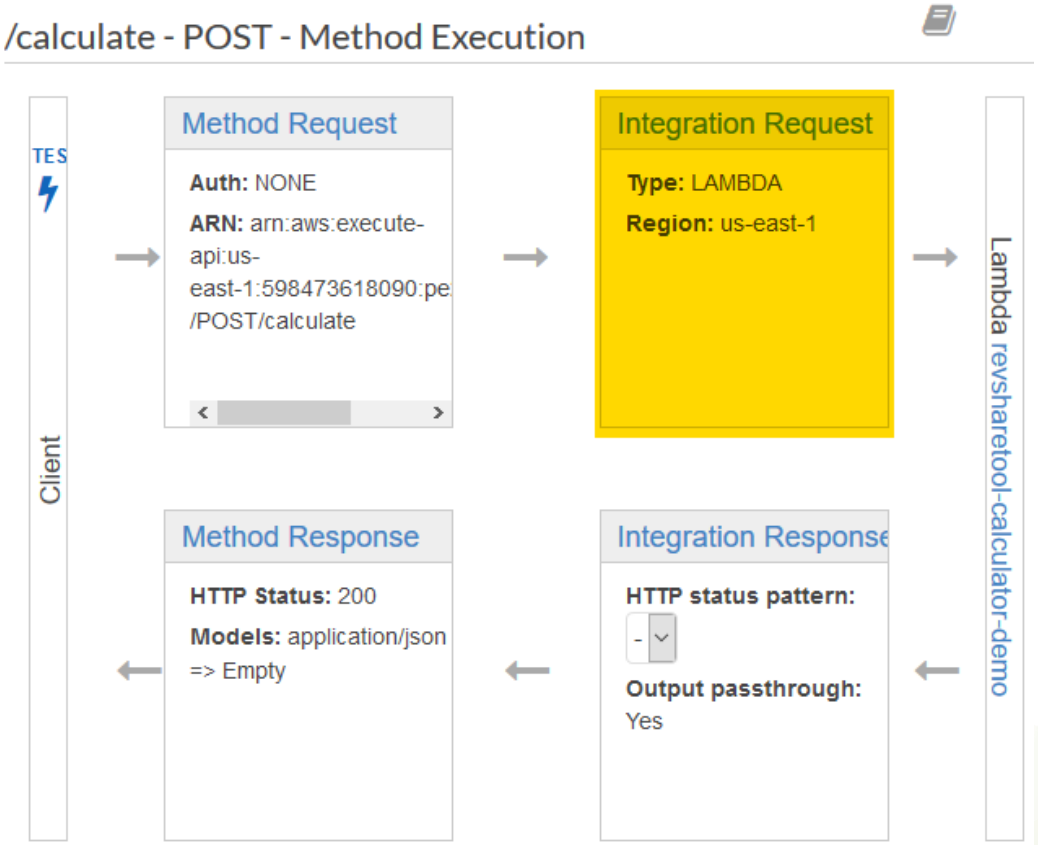
Name	Caching
customerId	☐

URL Query String Parameters

Name	Required	Caching
month	☐	☐
year	☐	☐

Add query string

Next step is …


/calculate - POST - Method Execution
TES
Client
Method Request
Auth: NONE
ARN: arn:aws:execute-api:us-east-1:598473618090:pe
/POST/calculate
Integration Request
Type: LAMBDA
Region: us-east-1
Lambda revsharetool-calculator-demo
Method Response
HTTP Status: 200
Models: application/json => Empty
Integration Response
HTTP status pattern:
-
Output passthrough:
Yes

API Integration Request

Resources | Actions

- /
  - /calculate
    - POST
  - /calculateWithParams
    - /{customerId}
      - POST

← Method Execution /calculate - POST - Integration Request

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type ◉ Lambda Function
○ HTTP
○ Mock
○ AWS Service
○ VPC Link

Use Lambda Proxy integration ☐

Lambda Region us-east-1

Lambda Function revsharetool-calculator-demo

Invoke with caller credentials ☐

Credentials cache Do not add caller credentials to cache key

Use Default Timeout ☑

▸ URL Path Parameters

▸ URL Query String Parameters

▸ HTTP Headers

▸ Body Mapping Templates

Request body mapping

- Velocity Template Language (Apache VTL)
  +
- JSON Path Expressions

```
$.store.book[0].title
$['store']['book`][0]['title']
```

Body Mapping Templates

Request body passthrough
- ○ When no template matches the request Content-Type header
- ◉ When there are no templates defined (recommended)
- ○ Never

Content-Type	
application/json	⊖

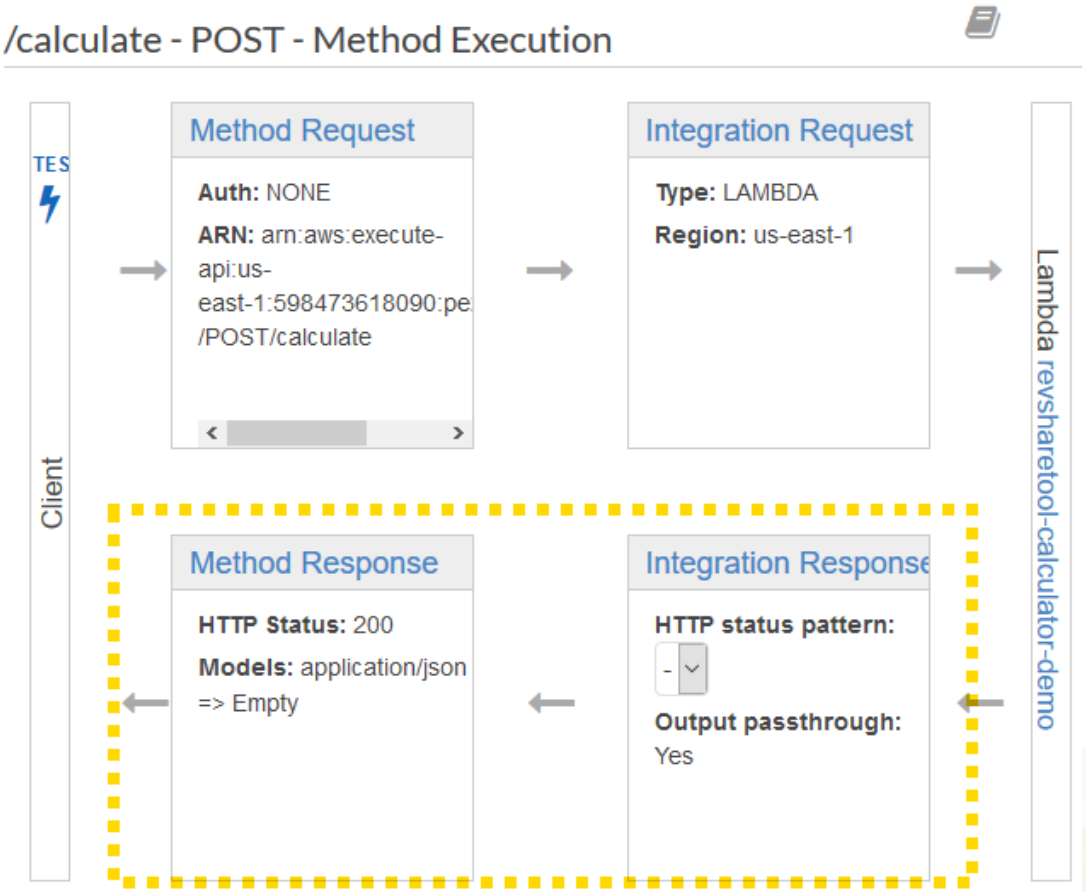
⊕ Add mapping template

application/json

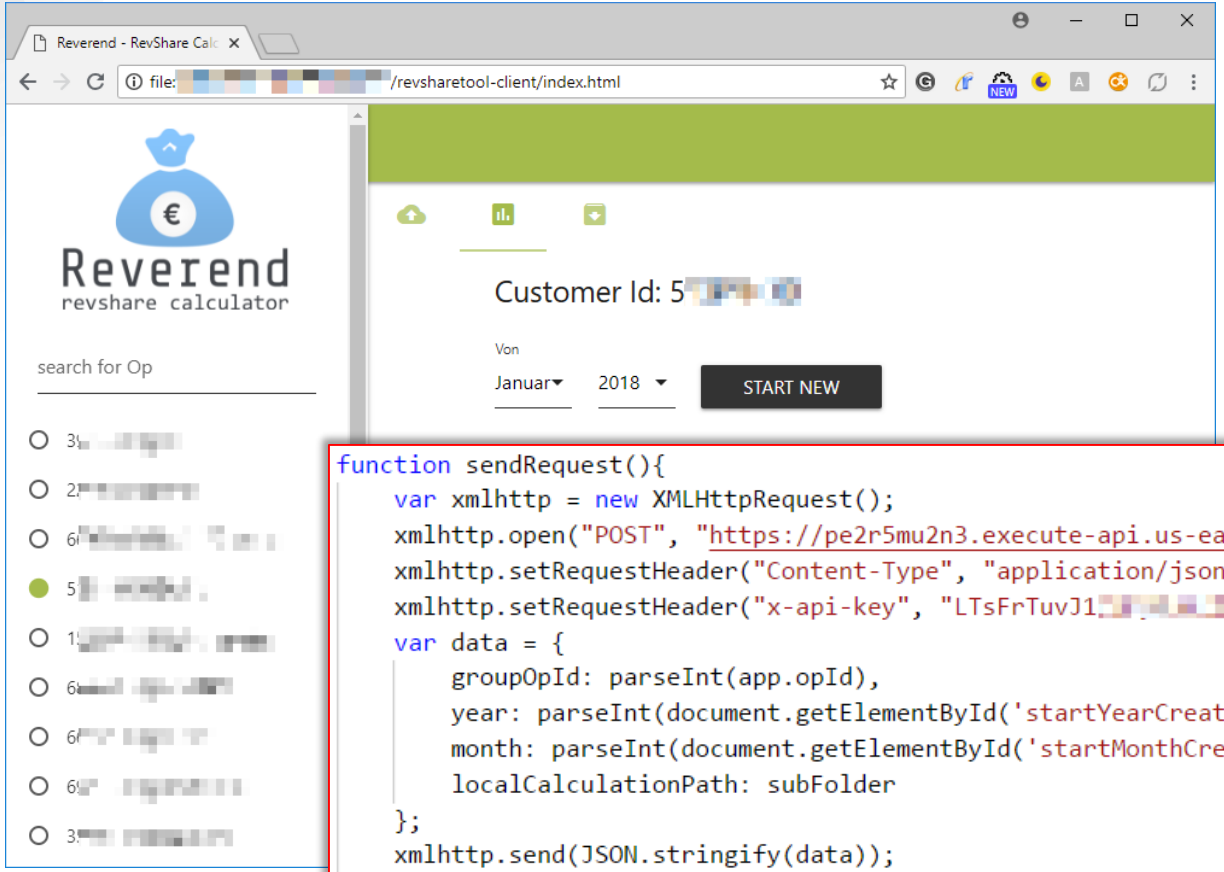
Generate template:

```
1 {
2     "customerId" : "$input.params('customerId')",
3     "year" : $input.params('year'),
4     "month" : $input.params('month')
5 }
```

Hausaufgabe ;-)



API Gateway for Lambda function in RevShare App




```
function sendRequest(){
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.open("POST", "https://pe2r5mu2n3.execute-api.us-east-1.amazonaws.com/Demo/calculate", false);
    xmlhttp.setRequestHeader("Content-Type", "application/json");
    xmlhttp.setRequestHeader("x-api-key", "LTsFrTuvJ1[redacted]");
    var data = {
        groupOpId: parseInt(app.opId),
        year: parseInt(document.getElementById('startYearCreate').value),
        month: parseInt(document.getElementById('startMonthCreate').value),
        localCalculationPath: subFolder
    };
    xmlhttp.send(JSON.stringify(data));
    console.log(JSON.parse(xmlhttp.response));
```

Amazon SNS

SNS Basics

- Publisher subscriber pattern
- Possible subscribers: Lambda functions, http(s) endpoints, email addresses, SMS receivers, SQS queues
- Optional protocol-specific message content

SNS configuration

aws Services Resource Groups

SNS dashboard
Topics
Applications
Subscriptions
Text messaging (SMS)

Topics

Publish to topic | Create new topic | Actions

Filter

Name	ARN
RevShareToolCalculation...	arn:aws:sns:us-east-1:598473618090:RevShareToolCalculationDone
dynamodb	arn:aws:sns:us-east-1:598473618090:dynamodb

Subscriptions

Create subscription | Request confirmations | Actions

Filter

Subscription ARN	Proto...	Endpoint	Topic ARN
arn:aws:sns:us-east-1:598473618090:RevShareToolCalculationDone:00882fe9-23b5-432...	email	v.kazulkin@iplabs.de	arn:aws:sns:us-east-1:598473618090:RevShareToolCalculationDone
arn:aws:sns:us-east-1:598473618090:RevShareToolCalculationDone:4f0f24d4-1526-49f2...	email	e.warken@iplabs.de	arn:aws:sns:us-east-1:598473618090:RevShareToolCalculationDone

Using SNS Java SDK

```java
private static final BasicAWSCredentials awsCredentials = new BasicAWSCredentials(RevShareConstants.REVSHARE_USER_ACCESS_KEY,
        RevShareConstants.REVSHARE_USER_SECRET_KEY);

private static final AWSCredentialsProvider awsCredentialsProvider = new AWSStaticCredentialsProvider(awsCredentials);

public final static AmazonSNS snsClient = AmazonSNSClientBuilder.standard()
        .withCredentials(awsCredentialsProvider)
        .withRegion(REGION).build();

public static final String SNS_TOPIC_ARN = "arn:aws:sns:us-east-1:598473618090:RevShareToolCalculationDone";
```

```java
    public SNSNotificationSender(final String topicArn, final AmazonSNS snsClient) {
        this.topicArn = topicArn;
        this.snsClient = snsClient;
    }

    @Override
    public void sendMessage(final String message, final String subject) {
        final PublishRequest publishRequest = new PublishRequest(this.topicArn, message, subject);
        this.snsClient.publish(publishRequest);
    }
```

Revenue Share Calculation for the customer 1000000 and for MAY 2018 - Nachricht (Nur-Text)

Datei | Nachricht | Was möchten Sie tun?

Löschen | Archivieren | Antworten | Allen antworten | Weiterleiten | Payment Captu... | An Vorgesetzte(n) | Team-E-Mail | Verschieben | Als ungelesen markieren | Kategorisieren | Nachverfolgung | Übersetzen | Zoom

AN

AWS Notifications <no-reply@sns.amazonaws.com> | Warken, Elmar | 13.06.2018

Revenue Share Calculation for the customer 1000000 and for MAY 2018

Your revenue share calculation data can be downloaded under the following URL: https://s3.us-east-1.amazonaws.com/revsharetool-storage-demo/output-data/1000000/output/201805/1000000_201805_final.txt

Calculation Report:

output key output-data/1000000/output/201805/1000000_201805_final.txt
download url :https://s3.us-east-1.amazonaws.com/revsharetool-storage-demo/output-data/1000000/output/201805/1000000_201805_final.txt

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:598473618090:RevShareToolCalculationDone:4f0f24d4-1526-49f2-923f-7910c8a3112c&Endpoint=e.warken@iplabs.de

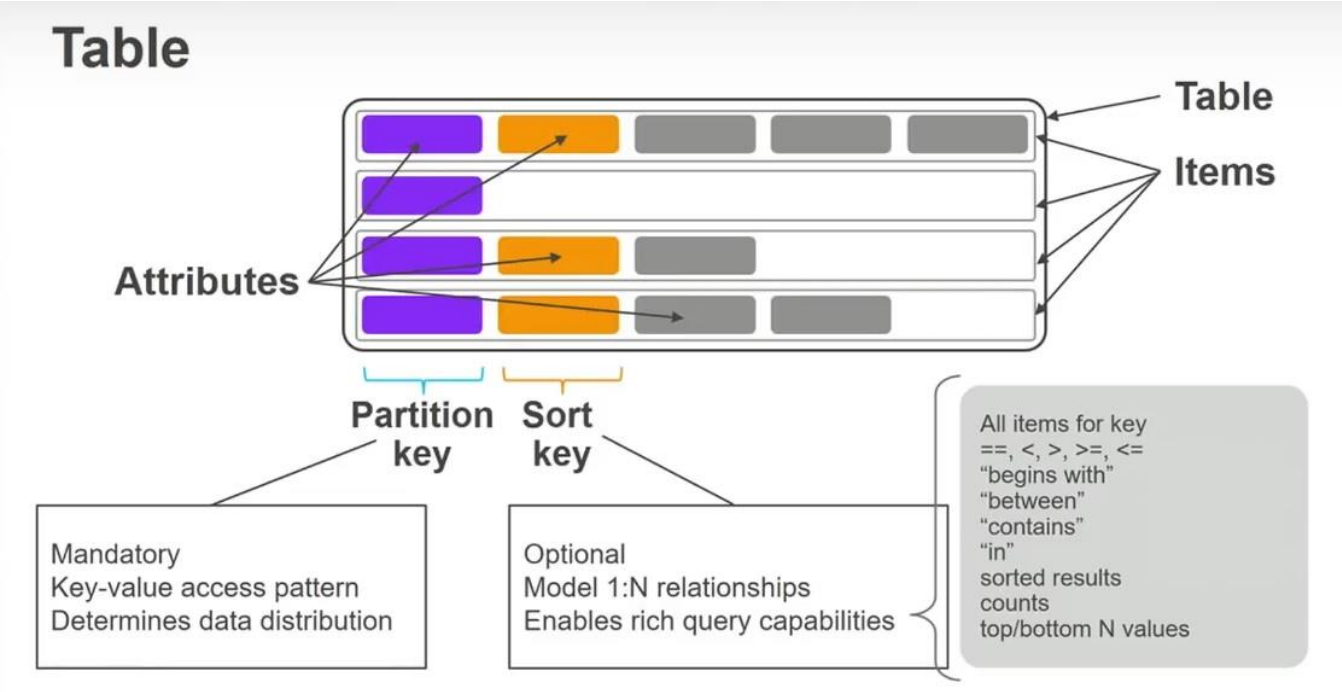
Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

Amazon **DynamoDB**

What is DynamoDB ?

- Fully managed NoSQL DB
- Document or key-value store
- Scales to any workload
- Fast and consistent

DynamoDB
Core Concepts



Source: Advanced Design Patterns for Amazon DynamoDB https://www.youtube.com/watch?v=jzeKPKpucS0

DynamoDB data types

- Scalar data types
  - number
  - string
  - binary
- Multi-value types
  - string-set
  - number-set
  - binary-set

DynamoDB data types: where is „date“ type?

String with the pattern

yyyy-MM-dd'T'HH:mm:ss.SSS‘Z‘

is automatically recognized as a **date**

DynamoDB Pricing

DynamoDB works by provisioning throughput at the table level. The throughput is set up as follows:

- Each write capacity unit (**WCU**) gives 1KB/s of write throughput
- Each read capacity unit (**RCU**) gives 4KB/s of read throughput
- Read and write throughput are independent

DynamoDB Pricing

Region: EU (Frankfurt)

Art des bereitgestellten Durchsatzes	Preis pro Stunde
Schreibkapazitätseinheit (WCU)	0,000793 USD pro WCU
Lesekapazitätseinheit (RCU)	0,0001586 USD pro RCU

Monthly free tier: 25 RCUs and 25 WCUs, 25 GB storage for all tables in AWS account

CurrencyExchangeRateInEuro DynamoDB table

DynamoDB
- Dashboard
- Tables
- Backups
- Reserved capacity
- Preferences Preview

DAX
- Dashboard
- Clusters
- Subnet groups
- Parameter groups
- Events

Create table | Delete table

Filter by table name

Name
CurrencyExchangeRateInEuro

CurrencyExchangeRateInEuro Close

Overview | Items | Metrics | Alarms | Capacity | Indexes | Global Tables | Backu

Stream details

Stream enabled	No
View type	-
Latest stream ARN	-

Manage Stream

Table details

Table name	CurrencyExchangeRateInEuro
Primary partition key	currency (String)
Primary sort key	localDate (String)
Point-in-time recovery	DISABLED Enable
Encryption	ENABLED
Time to live attribute	DISABLED Manage TTL
Table status	Active
Creation date	May 25, 2018 at 11:42:28 AM UTC+2
Provisioned read capacity units	1 (Auto Scaling Disabled)
Provisioned write capacity units	1 (Auto Scaling Disabled)
Last decrease time	-
Last increase time	-

CurrencyExchangeRateInEuro DynamoDB existing items

CurrencyExchangeRateInEuro Close

Overview | Items | Metrics | Alarms | Capacity | Indexes

Create item | Actions

Scan: [Table] CurrencyExchangeRateInEuro: currency, lo…

Scan | [Table] CurrencyExchangeRateInEuro: currency, localDate

Add filter

Start search

currency	localDate	value
CHF	2018-04-30T…	0.87
CHF	2018-05-31T…	0.85
CHF	2018-06-30T…	0.86

DynamoDB Java SDK Object Mapper creation

```
public class CurrencyExchangeRateInEuroDao {

    private final static Regions REGION= Regions.US_EAST_1;

    private static final BasicAWSCredentials AWS_CREDENTIALS = new BasicAWSCredentials(RevenueShareConstants.REVSHARE_USER_ACCESS_KEY,
            RevenueShareConstants.REVSHARE_USER_SECRET_KEY);

    private static final AWSCredentialsProvider AWS_CREDENTIALS_PROVIDER= new AWSStaticCredentialsProvider(AWS_CREDENTIALS);


    public static final AmazonDynamoDB DYNAMODB_CLIENT = AmazonDynamoDBClientBuilder.standard()
            .withCredentials(AWS_CREDENTIALS_PROVIDER)
            .withRegion(REGION).build();


    private static final DynamoDBMapper MAPPER = new DynamoDBMapper(DYNAMODB_CLIENT);
```

DynamoDB Java SDK CurrencyExchangeInEuro Entity

```java
@DynamoDBTable(tableName = CurrencyExchangeRateInEuro.TABLE_NAME)
public class CurrencyExchangeRateInEuro {
    public final static String TABLE_NAME="CurrencyExchangeRateInEuro";

    private String currency;
    private String date;
    private double value;

    @DynamoDBHashKey(attributeName = "currency")
    public String getCurrency() {
        return this.currency;
    }

    @DynamoDBRangeKey(attributeName = "localDate")
    public String getDate() {
        return this.date;
    }

    @DynamoDBAttribute(attributeName = "value")
    public double getValue() {
        return this.value;
    }
```

DynamoDB Java SDK Create/Update/Load

```java
/** creates or updates record for the currency exchange rate to euro for the given date
public void createOrUpdateCurrencyExchangeRateInEuro(final String currency, final LocalDate date,
        final double value)
{
    final CurrencyExchangeRateInEuro currencyExchangeRate= new CurrencyExchangeRateInEuro();
    currencyExchangeRate.setCurrency(currency);
    currencyExchangeRate.setValue(value);
    currencyExchangeRate.setDate(DynamoDBUtils.formatLocalDate(date));

    // save performs create and update
    MAPPER.save(currencyExchangeRate);
}

/** creates or updates record for the currency exchange rate to euro for the given date
 *
 * @param currencyExchangeRates
 */
public void createOrUpdateCurrencyExchangeRateInEuro(final List<CurrencyExchangeRateInEuro> currencyExchangeRates)
{

   // batch save performs create and update
    MAPPER.batchSave(currencyExchangeRates);
}


/** returns currency exchange rate for the last day of month for the given year
public CurrencyExchangeRateInEuro getCurrencyExchangeRateInEuro(final String currency, final LocalDate date) {

    return MAPPER.load(CurrencyExchangeRateInEuro.class, currency, DynamoDBUtils.formatLocalDate(date));
}
```

DynamoDB Java SDK query expression with a Sort Key

```
public List<CurrencyExchangeRateInEuro> getCurrenciesExchangeRateInEuro(final String currency,
        final LocalDate from, final LocalDate to) {

    Map<String, AttributeValue> eav = new HashMap<String, AttributeValue>();
    eav.put(":val1", new AttributeValue().withN(String.valueOf(currency)));

    eav.put(":val2", new AttributeValue().withS(DynamoDBUtils.formatLocalDate(from)));
    eav.put(":val3", new AttributeValue().withS(DynamoDBUtils.formatLocalDate(to)));

    DynamoDBQueryExpression<CurrencyExchangeRateInEuro> queryExpression = new DynamoDBQueryExpression<CurrencyExchangeRateInEuro>()
            .withKeyConditionExpression("currency = :val1 and localDate between :val2 and :val3")
            .withExpressionAttributeValues(eav);

    return MAPPER.query(CurrencyExchangeRateInEuro.class, queryExpression);
}
```

DynamoDB canonical use cases

- Key-value lookups on well-distributed records
- Avoiding complex queries (and joins)
- Limiting hot keys

DynamoDB is not the right choice for each solution

- Think of the access pattern in advance
- Difficult to change access patterns (partition&sort key) & table structure afterwards
- Difficult to replace DynamoDB with another even (NoSQL) database in the future

Why NoSQL database DynamoDB?

- Fixed Scheme (argument against NoSQL DB)
- Small size of the table and the entities
- Lock-in into cloud-native NoSQL DB
- Learning curve for the new technology
- No aggregate functions was sum, avg, count and no joins

Why NoSQL database DynamoDB?

Mainly for only one reason : low price for our use case

FREE USAGE TIER: New Customers get free usage tier for first 12 months

Services | Estimate of your Monthly Bill ($ 0.70)

Choose region: US East (N. Virginia)

Inbound Data Transfer is Free and Outbound Data Transfer is 1 GB free per region per month

Amazon DynamoDB is a high performance non-relational database service that is easy to set up, operate, and scale. It is designed to address the core problems of database management, performance, scalability, and reliability. It also provides predictable high performance and low latency at scale.

Clear Form

FREE TIER: Each month, Amazon DynamoDB users pay no charges on the first 25GB of storage, the first 2.5 million DynamoDB Streams read request units, as well as 25 write capacity unit and 25 read capacity unit of ongoing throughput capacity. Also free tier is applicable to deploy DynamoDB Global Tables in up to 2 AWS regions.

Indexed Data Storage:

Dataset Size: 1 GB

Provisioned Throughput Capacity *:

Item Size (All attributes): 1 KB

Number of items read per second: 2 Reads/Second

Read Consistency: Strongly Consistent / Eventually Consistent (2x cheaper)

Number of items written per second: 2 Writes/Second

DynamoDB Streams:

Read Request Units per month: 0 Units/Month

Data Transfer:

Data Transfer Out: 1 GB/Month

Data Transfer In: 1 GB/Month

Reserved Capacity:

	Description	Instances	Monthly Commitment	Term
	Add New Row			

DAX Clusters: On-Demand DAX Nodes:

	Cluster Name	Nodes	Usage	Node Type
	Add New Row			

Global Tables

Number of Regions to deploy: 2 Regions

Number of items read per second: 0 Reads/Second

Number of replicated writes per second: 0 Writes/Second

Data Storage: 1 GB

On-demand backup

Data Storage: 1 GB

Continuous backup (PITR)

Data Storage: 1 GB

Restoring a table

Data Storage: 1 GB

Preis für 1 RDS Postgres DB-Instanz

Region: EU (Frankfurt)

db.t2.micro

STANDARD – LAUFZEIT 1 JAHR					
Zahlungsoption	Vorabzahlung	Monatlich*	Tatsächlicher Stundensatz**	Ersparnis gegenüber On-Demand	On-Demand-Stundensatz
Keine Vorabzahlung	0,00 USD	13,14 USD	0,018 USD	28 %	0,025 USD
Teilweise Vorauszahlung	62,00 USD	5,84 USD	0,01508 USD	40 %	
Komplette Vorauszahlung	130,00 USD	0,00 USD	0,01484 USD	41 %	

STANDARD – LAUFZEIT 3 JAHRE					
Zahlungsoption	Vorabzahlung	Monatlich*	Tatsächlicher Stundensatz**	Ersparnis gegenüber On-Demand	On-Demand-Stundensatz
Teilweise Vorauszahlung	106,00 USD	4,38 USD	0,01003 USD	60 %	0,025 USD
Komplette Vorauszahlung	248,00 USD	0,00 USD	0,009437 USD	62 %	

db.m2 – Speicheroptimierte Instance-Klassen der vorherigen Generation.									
db.m2.4xlarge	8	26	68,4	Nein	Ja	1 000	Hoch	Nein	Nein
db.m2.2xlarge	4	13	34,2	Nein	Ja	500	Mittel	Nein	Nein
db.m2.xlarge	2	6,5	17,1	Nein	Nein	—	Mittel	Nein	Nein
db.t2 – Burstable Performance Instance-Klassen der aktuellen Generation									
db.t2.2xlarge	8	8	32	Ja	Nein	—	Mittel	Nein	Nein
db.t2.xlarge	4	4	16	Ja	Nein	—	Mittel	Nein	Nein
db.t2.large	2	2	8	Ja	Nein	—	Mittel	Nein	Nein
db.t2.Medium	2	2	4	Ja	Nein	—	Mittel	Ja	Nein
db.t2.small	1	1	2	Ja	Nein	—	Niedrig	Ja	Nein
db.t2.micro	1	1	1	Ja	Nein	—	Niedrig	Nein	Nein

Amazon Aurora Serverless

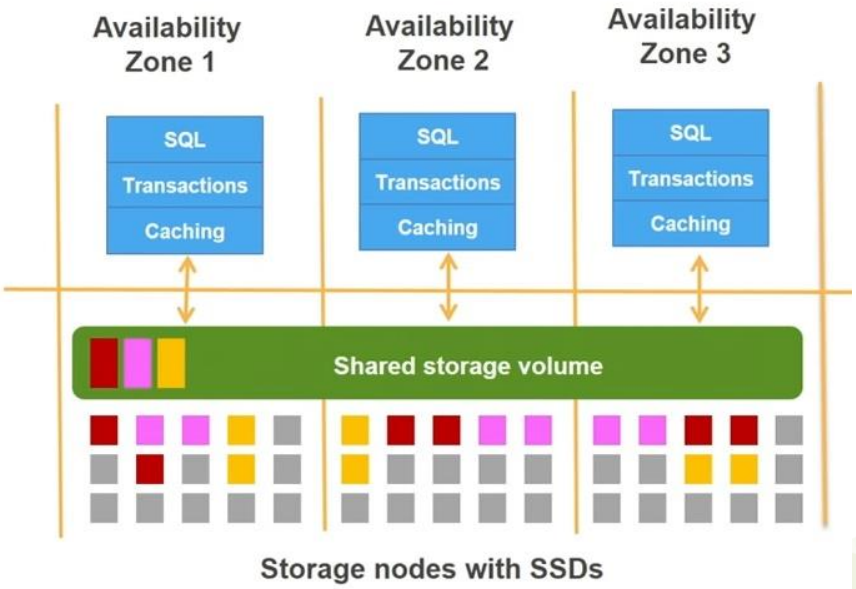
(GA for MySql in us-east-1 N. Virginia since 09.08.2018)

What Is Amazon Aurora ~~Serverless~~?

Amazon Aurora (MySql/Postgres)

Scale-out, distributed architecture

- Purpose-built log-structured distributed storage system designed for databases
- Storage volume is striped across hundreds of storage nodes distributed over 3 different availability zones
- Six copies of data, two copies in each availability zone to protect against AZ+1 failures
- Plan to apply same principles to other layers of the stack

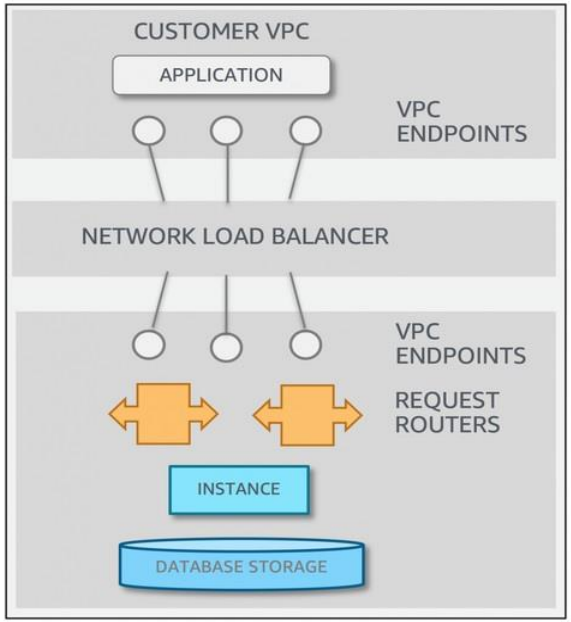


Source: “AWS Summit - London | twitch.tv/aws | Aurora Serverless Preview Update”
https://www.youtube.com/watch?v=J3A_TJZosu8

What Is Amazon Aurora Serverless?

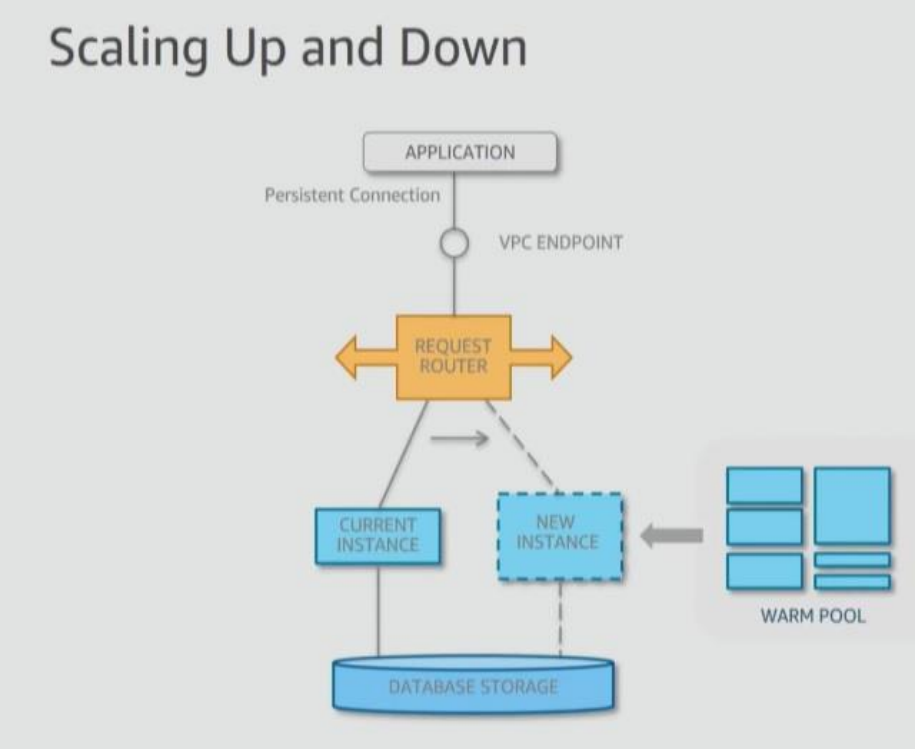
Amazon Aurora Serverless Architecture

- It creates an Aurora storage volume replicated across multiple AZs.
- It creates an endpoint in your VPC for the application to connect to.
- It configures a network load balancer (invisible to the customer) behind that endpoint.
- It configures multi-tenant request routers to route database traffic to the underlying instances.
- It provisions the initial minimum instance capacity.



Source: https://aws.amazon.com/de/blogs/aws/aurora-serverless-ga/

Amazon Aurora Serverless Architecture



Source: “AWS Summit - London | twitch.tv/aws | Aurora Serverless Preview Update”
https://www.youtube.com/watch?v=J3A_TJZosu8

Amazon Aurora Serverless Pricing & Settings

Configure advanced settings

Capacity settings

Billing estimate is based on published prices. Learn more

Minimum Aurora capacity unit Info

2 — $ 0.12 per hour
4Gb RAM

Maximum Aurora capacity unit Info

128 — $ 7.68 per hour
244Gb RAM

Additional scaling configuration

☑ Pause compute capacity after consecutive minutes of inactivity Info

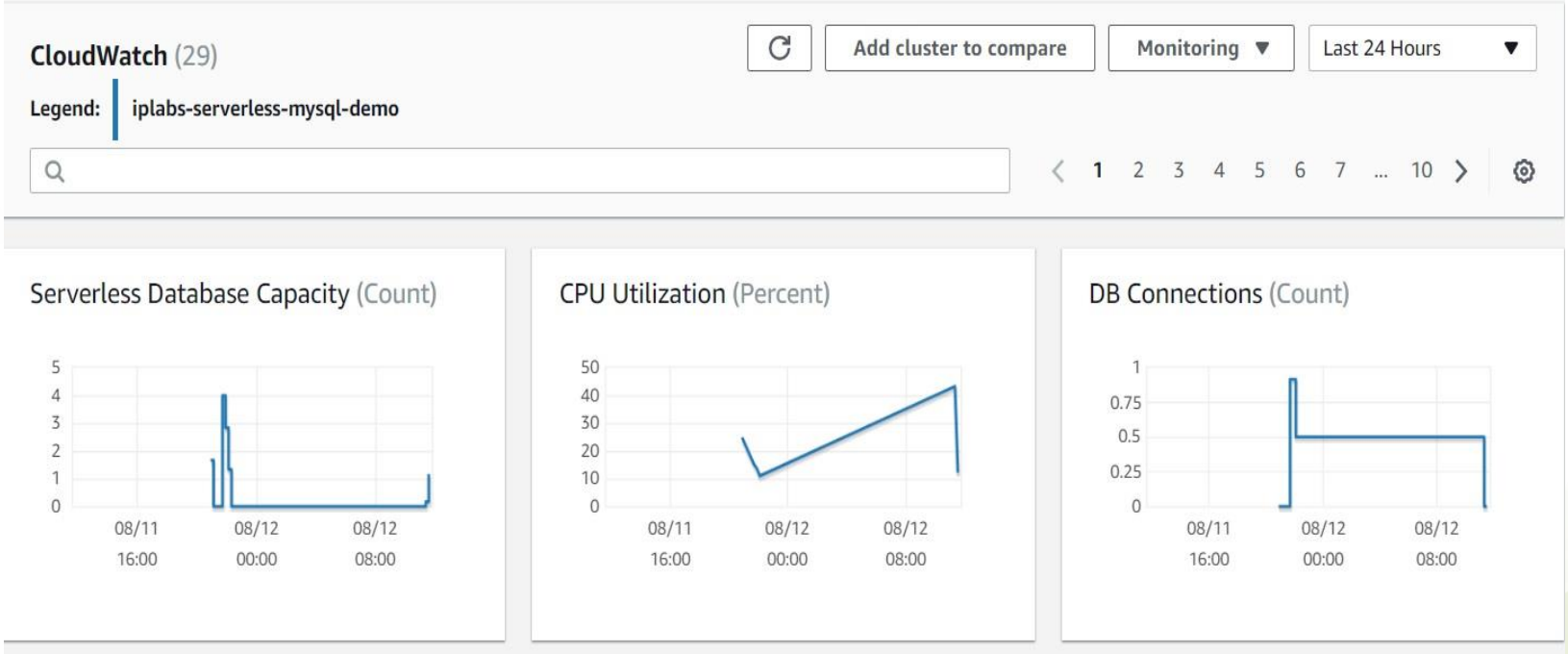
You are only charged for database storage while the compute capacity is paused

5 minutes

(up to 1440)

Source: https://aws.amazon.com/de/blogs/aws/aurora-serverless-ga/

Amazon Aurora Serverless CloudWatch Monitoring



Source: https://aws.amazon.com/de/blogs/aws/aurora-serverless-ga/

Amazon Aurora Serverless Pricing & Settings

Sun Aug 12 11:28:59 GMT+200 2018	The DB cluster is being resumed.
Sat Aug 11 22:15:31 GMT+200 2018	The DB cluster is paused.
Sat Aug 11 22:14:23 GMT+200 2018	The DB cluster is being paused.
Sat Aug 11 21:58:42 GMT+200 2018	The DB cluster has scaled from 4 capacity units to 2 capacity units.
Sat Aug 11 21:58:29 GMT+200 2018	Scaling DB cluster from 4 capacity units to 2 capacity units for this reason: Autoscaling.
Sat Aug 11 21:41:58 GMT+200 2018	The DB cluster is resumed.
Sat Aug 11 21:41:56 GMT+200 2018	DB instance restarted
Sat Aug 11 21:41:30 GMT+200 2018	The DB cluster is being resumed.
Sat Aug 11 21:02:06 GMT+200 2018	The DB cluster is paused.
Sat Aug 11 21:01:24 GMT+200 2018	The DB cluster is being paused.
Sat Aug 11 20:55:34 GMT+200 2018	DB instance restarted

Source: https://aws.amazon.com/de/blogs/aws/aurora-serverless-ga/

Use cases for Amazon Aurora Serverless

- Seldom usage (weekly jobs) + additional spikes

- Latencies acceptale

- Dev/Test Database

Questions?

Thank You!