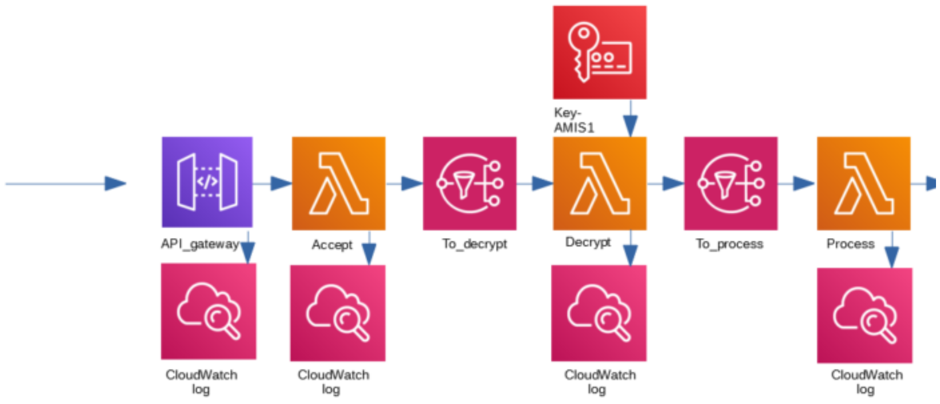


Example application in AWS using Lambda

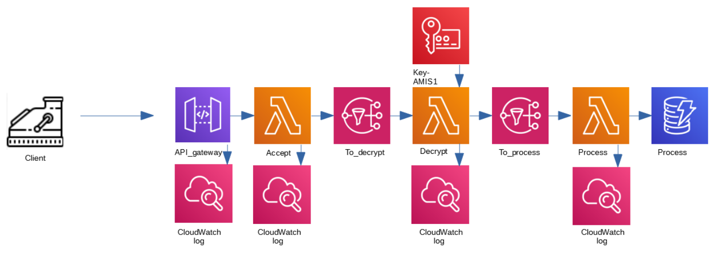
0

BY FREDERIQUE RETSEMA ON APRIL 26, 2020 AWS, CLOUD, SERVERLESS, TECHNICAL ARCHITECTURE

Introduction

I have to admit: I love serverless. Serverless computing is using the cloud as it is supposed to be used: it scales up when you need more capacity, it scales down to zero when you don't need resources. That is really good when you have, for example, a shop which is still open despite Corona, but which has less turnover because less people are buying products.

Example



In this example, I simulate a caching machine in a shop: the client application will create one API call per customer. A customer can buy multiple products. The client encrypts the content and then calls a REST API in AWS (the API Gateway is used for this). The API gateway will deliver the data to a Lambda function, which is called "accept".

The accept function accepts the request and sends it to an SNS topic. SNS stands for Simple Notification Service. SNS is a service where you can choose to send the message to zero, one or multiple subscribers. In our case, we send the message to the decrypt function.

The decrypt function uses the name of the shop to get the correct key from AWS KMS (Key Management Service). After decryption, the name of the shop and the content of the decrypted message are send to another SNS topic, which will deliver the content to the process function.

The process function uses the content of the decrypted message to update the DynamoDB (AWS NoSQL) database.

All Lambda functions use Cloudwatch for logging.

ABOUT AUTHOR

Frederique Retsema

Frederique Retsema is active in IT si
1993. Senior Consultant and develop
on diverse areas including SQL and Java. She likes work with automation tools like Bamboo, Jenkins, Ansible, Terraform and CloudFormation.

View all posts

FOLLOW US ON LINKEDIN

Follow 2,920

POPULAR TAGS

Agile analytical function apex api application container clou service Architecture Azure BPEL bpm cloud container Database docker DVT h integration iot Java javascript jms js kafka kubernetes linux maven monitoring node oracle cloud oracle cloud infrastructure oracle databa oracle xml db OSB paas performance tuning plsq provisioning puppet push python REST saas Scrum soa sql Vagrant virtual box visualization XML

FOLLOW US ON TWITTER

Tweets by @AMISnl

AMIS Conclusion
@AMISnl

New Article : Windows Sandbox – light weight playground for R&D, tutorials and workshops ift.tt/2znr
by Lucas Jellema

Advantages of using SNS

We could use just one function (and no SNS topics) instead of three to get our job done. By using SNS we achieve two goals: the first goal is to disconnect the processing of the message from the cash machine. This will speed up the process for the client: the next customer might be waiting already!

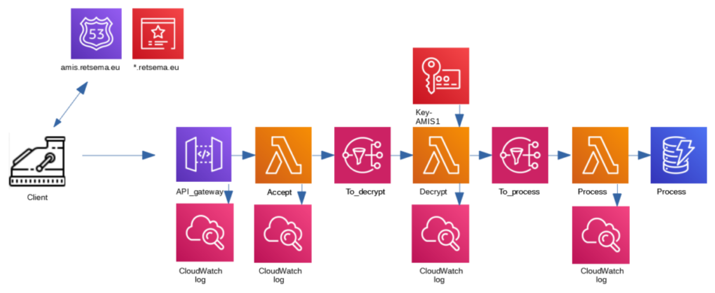
The second goal is that we can add several kinds of processing of one message, without changing the code of the program. SNS can be used with Lambda functions (as is done in this example), but SNS messages can also be send as email, as text messages (SMS) or as push messages to mobile phones. It can also send the message to an SQS (Simple Queue Service) queue in AWS and to HTTP(S) API's.

A few examples of what we might add to the SNS topics:

- We might connect an extra Lambda function to the to_decrypt SNS topic, which will look how many messages are sent with incorrect keys and what keys are used in that incorrect messages.
- We can add an extra subscriber to the SNS-topic to_process, which can change the datawarehouse based on the decrypted data. This solution might be written in Lambda to use Amazon Redshift for the datawarehouse. It might also use an on-premise solution, the subscriber can then use HTTP or HTTPS to call an on-premise web hook.
- We didn't implement it in this example, but it could be useful to have an SNS queue that collects errors in the decryption. The SNS message can then be forwarded as e-mail to a ticketing system. The same SNS topic could also send an SMS or a push message to mobile phones, to warn the ICT department (and also the store manager) that something is going wrong.

One extra step…

In the image above, the API will create an HTTPS endpoint to deliver content to. This endpoint looks like https://wmfmq9t91a.execute-api.eu-west-1.amazonaws.com/prod/shop . When we would destroy the deployment and re-create it, it will get another endpoint: the first part of this URL is completely random. It would be better to make this endpoint part of our domain. In my case, I have a domain called retsema.eu. I'd like to use an endpoint which is constant over time, for example https://amis.retsema.eu/shop.



To get this done, we need a certificate. We can use the AWS Certificate Management (ACM) for this. ACM will ask you to to add a CNAME record to the domain that the certificate belongs to, to check if you really are allowed to request certificates for this domain. When the domain is checked, the certificate is issued.

When we have the certificate, we can add a DNS record that points to the endpoint in the API.

This is just an overview of this solution. When you use Terraform, the AWS Service Development Kit (SDK) or the AWS Command Line Interface (CLI), there are multiple objects and multiple calls per service. We also need AWS Identity Access Management (IAM) for example for policies for the Lambda functions. I will walk you through the GUI to look at the objects that are used in this example. You can play along and look at/change these objects yourself in the GUI in later blogs, and will talk more about the details of each service in these images.

Play along

I scripted the solution [2], and will go into more detail (using screen prints of AWS) in other blogs. You can follow along and create this solution in your own AWS environment, see the README.md file in the vagrant directory. Install Vagrant [3] and Virtual Box [4], go to the vagrant directory and use Vagrant to install and configure the virtual machine:

D:\AMIS-Blog-AWS\shop-1\vagrant>**vagrant up**

Bringing machine 'default' up with 'virtualbox' provider…

==> default: Importing base box 'centos/7'…

Windows Sandbox - light weight playground f…
Windows Sandbox to me is a light weight Windows 10 virtual machine that I can quickly start and stop
technology.amis.nl

AMIS Conclusion
@AMISnl
New Article : Azure Pipelines: Using and set up a webapp (part 1) ift.tt/3bHsZm4 by Joost Luijben

Azure Pipelines: Using and set up a webapp (p…
Azure Pipelines: Using and set up a webapp (part 1). This article explains how you can set up a azure
technology.amis.nl

Apr 24, 2

AMIS Conclusion
@AMISnl
New Article : Azure Pipelines: How to build and test ar Angular and Node.js webapp and deploy it to on-prem VM's using multiple stages (YAML) – Overview ift.tt/2yL1GZ6 by Joost Luijben

Azure Pipelines: How to build and test an Ang…
Azure Pipelines (devops): How to build and test an Angular and Node.js webapp and deploy it to self-
technology.amis.nl

Apr 24, 2

AMIS Conclusion
@AMISnl
New Article : My Tips for Presenting an Online Meetup experiences from a would-be pro ift.tt/3ax3HFO by Lu Jellema

My Tips for Presenting an Online Meetup - exp…
After presenting a three part Online Meetup series to colleagues and a three hour webinar to just over
technology.amis.nl

Apr 24, 2

AMIS Conclusion
@AMISnl
Wij zijn trots! Vorig jaar september startte AMIS met d bouw van een IoT-platform voor een energieonderneming. Wat doen we dan en waar staan we nu? 👉 Lees hier het interview met resultaatmana Frank bit.ly/2XRqEk7 #IoT #connect

==> default: Matching MAC address for NAT networking…

[…]

100 2223 100 2223 0

default: 0 10672 0 —

default: :–:– –:–:– –:–:– 10687

D:\AMIS-Blog-AWS\shop-1\vagrant>**vagrant ssh**

[vagrant@localhost ~]$ **./init-all.sh**

Access key/secret access key: Copy/paste your access key and secret access key to the next two questions.

If you don't have (secret) access keys yet, read the README.md file in the vagrant directory

Default region : When you don't know what region to use, use eu-central-1

Default output format : Just press enter

AWS Access Key ID [None]:**ABCDEF12GHIJKLMN34OP**

AWS Secret Access Key [None]: **abCDef1ghIj2jkm+n3oPqr-STUVw4xyZaBCdE!FG**

Default region name [None]: **eu-central-1**

Default output format [None]:

Cloning into 'AMIS-Blog-AWS'…

[…]

You can choose now to use a public domain in Route53 to see the whole example, or to use the example without

public domain. The example without public domain will work with domain names from Amazon AWS, like

https://5pcn5scuq5.execute-api.eu-west-1.amazonaws.com/prod/shop

When you do have a public domain in Route53, you can also use that domain, the URL will then become something

like https://amis.retsema.eu/shop (when you would own retsema.eu, that is)

Do you have a public domain in Route53 AND do you want the example shop application to use this domain?

Only yes will use the public domain

yes

Which domain do you own?

mydomain.nl

Initializing the backend…

#beheerstvernieuwen

AMIS bouwt IoT-platform voor energieonderne…
AMIS bouwt IoT-platform voor energieonderneming
amis.nl

Apr 21, 20

Embed View on Twit

META

- Site Admin
- Log out
- Entries feed
- Comments feed
- WordPress.org

```
Initializing provider plugins…

– Checking for available provider plugins…

– Downloading plugin for provider “aws” (hashicorp/aws) 2.57.0…

[…]

Done!
```

Try for yourself: use the following commands to see if the rollout was successful:

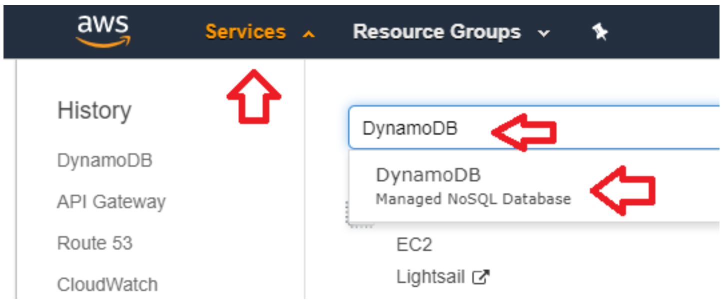
```
cd ~/AMIS-Blog-AWS/shop-1/client

./encrypt_and_send.py AMIS1 https://5ma4daqnoj.execute-api.eu-central-1.amazonaws.com/prod/shop
```

or

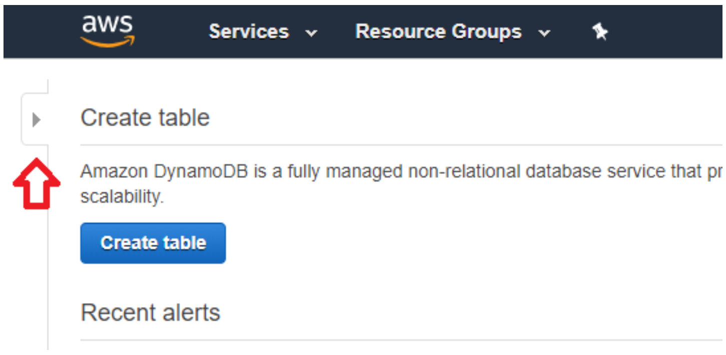
```
cd ~/AMIS-Blog-AWS/shop-1/client

./encrypt_and_send.py AMIS1 https://amis.mydomain.nl/shop
```

```
[vagrant@localhost ~]$ cd ~/AMIS-Blog-AWS/shop-1/client

[vagrant@localhost client]$ ./encrypt_and_send.py AMIS1 https://5ma4daqnoj.execute-api.eu-central-1.amazonaws.com/prod/shop

Shop id = AMIS1

Key alias = alias/KeyK-AMIS1

URL = https://5ma4daqnoj.execute-api.eu-central-1.amazonaws.com/prod/shop

Data = {“shop_id”: “AMIS1”, “content_base64”:
“KNPRXPmGLaoGaAmyIztme8BkDRB/Ga5SobJW3243arjS0PDaU3wlfKo8Zj0fnhUdorqdzZZzpUZHBIN1yZMtbizEbva+EMWn0mphuylWYC+G5mJi

request_number = 1

Status code = 200

Content = b'”OK”'

[vagrant@localhost client]$
```

Log in to the GUI, click on Services, type DynamoDB and click on DynamoDB:

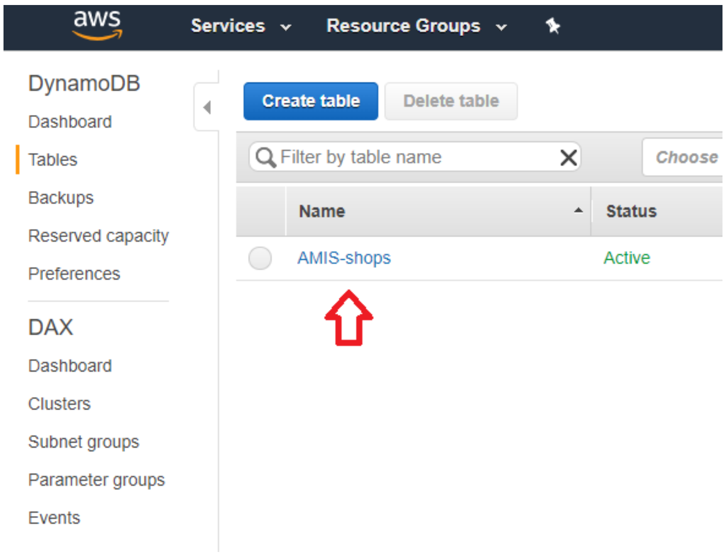


When the menu on the left is not visible, click on the arrow to make it visible:



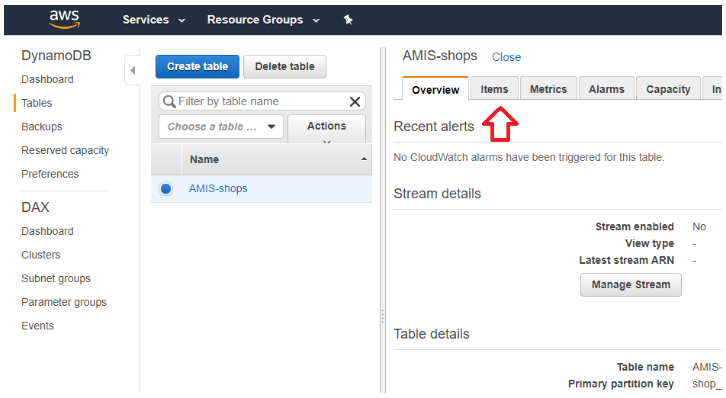
Amazon DynamoDB is a fully managed non-relational database service that pr
scalability.

In this menu, click on Tables. You will see the AMIS-shops table. Click on the link AMIS-shops:



Name	Status
AMIS-shops	Active

In the details that pops up, click on the Items tab:



Recent alerts

No CloudWatch alarms have been triggered for this table.

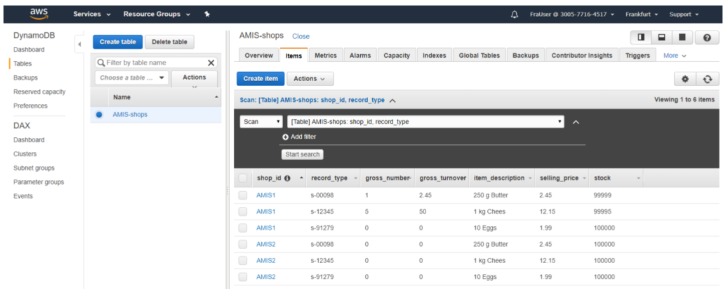
Stream details

Stream enabled	No
View type	-
Latest stream ARN	-

Table details

Table name	AMIS-
Primary partition key	shop_

You can see that the table has items for two shops: AMIS1 and AMIS2. By using the ./encrypt_and_send script, you sold some items for the AMIS1 shop:



Scan: [Table] AMIS-shops: shop_id, record_type

Viewing 1 to 6 items

[Table] AMIS-shops: shop_id, record_type

shop_id	record_type	gross_number	gross_turnover	item_description	selling_price	stock
AMIS1	s-00098	1	2.45	250 g Butter	2.45	99999
AMIS1	s-12345	5	50	1 kg Chees	12.15	99995
AMIS1	s-91279	0	0	10 Eggs	1.99	100000
AMIS2	s-00098	0	0	250 g Butter	2.45	100000
AMIS2	s-12345	0	0	1 kg Chees	12.15	100000
AMIS2	s-91279	0	0	10 Eggs	1.99	100000

(When you run into problems, look at the README.md file in the vagrant directory).

Costs

In this example, the only costs that are constant, are the costs of storage of the DynamoDB table. All other costs are going up with the number of API calls and go down when you use this solution less. Though I have a normal (non free tier) AWS account and I used AWS a lot to create and test this solution and the scripts, I only pay about $5 to $10 per month. The way to keep the costs low is to destroy all objects which you don't use. So when you stop today and want to continue tomorrow, destroy the environment today and bring it up again tomorrow.

Destroying the objects

You can destroy all items that were created by the ./init-all.sh script, by using ./destroy-all.sh script from the home directory of the vagrant user. After destroying the objects of the shop and the infra objects, you will be asked if you want to destroy the star certificate (read more about that in the README.md file in the vagrant directory). If you answer "yes" in this question, then the certificate and the DNS entry for requesting the certificate will be destroyed as well.

```
[vagrant@localhost client]$ cd ~

[vagrant@localhost ~]$ ./destroy-all.sh

data.aws_acm_certificate.domain_certificate[0]: Refreshing state…

data.aws_route53_zone.my_zone[0]: Refreshing state…

[…]

===============================================================

Did you install the certificate via the script (/home/vagrant/init-all.sh) AND do you want to delete the certificate?

(please mind that you shouldn't do this too often because of the AWS limits on the number of certificates you are allowed to request per year – this defaults to 20)

Only yes will destroy the certificate

no

Didn't destroy the certificate

Done

[vagrant@localhost ~]$
```

Links

[1] Image of the cash machine: https://icons8.com/icons/set/cash-machine , images of AWS services: https://aws.amazon.com/architecture/icons/

[2] https://github.com/FrederiqueRetsema/AMIS-Blog-AWS , directory shop-1

[3] https://www.vagrantup.com/downloads.html

[4] https://www.virtualbox.org/wiki/Downloads

Related Posts:

OOW 2011 – Oracle Enterprise Manager

AMIS vat Oracle OpenWorld samen in

The AMIS Summary of Oracle OpenWorld

new Puppet 3 Weblogic

Creating policy's, groups and users in

ABOUT AUTHOR

FREDERIQUE RETSEMA

Frederique Retsema is active in IT since 1993. Senior Consultant and developer on diverse areas including SQL and Java. She likes to work with automation tools like Bamboo, Jenkins, Ansible, Terraform and CloudFormation.

RELATED POSTS

APRIL 9, 2020 2

Migrating an old (10.2.0.4) database to Oracle Cloud with minimal downtime

MARCH 8, 2020 0

Policies in AWS (2)

MARCH 7, 2020 0

Creating policy's, groups and users in AWS

Leave a Reply

Enter your comment here...