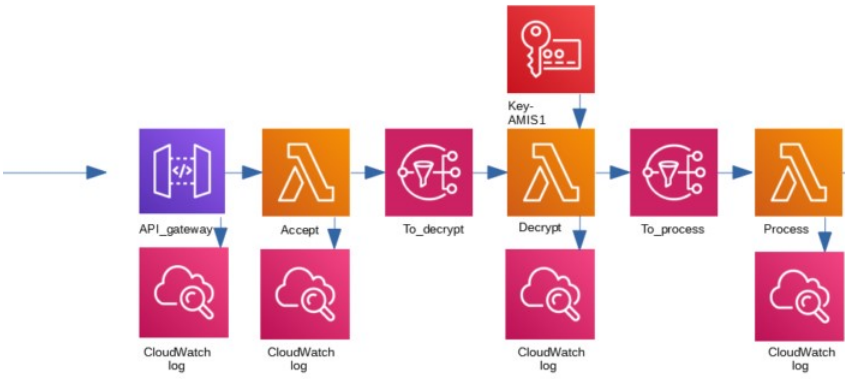


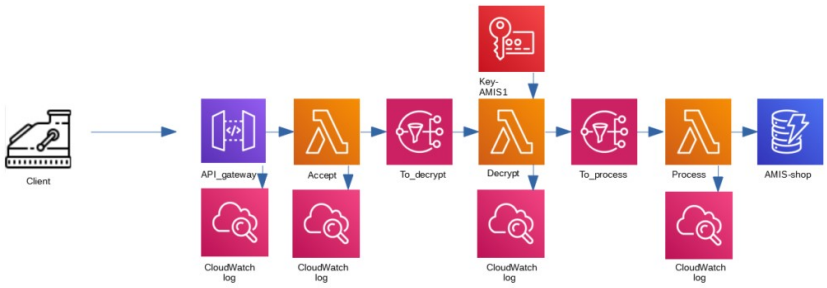
AWS Shop: about the AWS Simple Notification Service (SNS)

0

BY FREDERIQUE RETSEMA ON MAY 2, 2020 AWS, CLOUD

Introduction

Today [1] we'll look at the AWS Simple Notification Service. We have two of them in our shop: one to get messages from the accept-lambda function and send them to the decrypt-Lambda function, and the decrypt-Lambda function will send the decrypted sales information via SNS to the process-Lambda. The SNS objects are called "topics". In my shop example, these topics have the names "to_decrypt" and "to_process".



I already discussed briefly in the introduction of this series why SNS can be useful: we like to decouple the accepting of the messages from the processing of those messages. That will speed up the cash machine: customers don't have to wait for all the things we might do with the sales information.

The second advantage is, that we can send one message to multiple destinations. In SNS, those destinations are called subscribers. Let's play a little bit with the ideas behind SNS and add a subscriber to one of the SNS topics.

Add an SMS to the SNS topic

First, go to the SNS service: choose Services in the top menu, and type "SNS". You will see the SNS service come up, click on it:

ABOUT AUTHOR

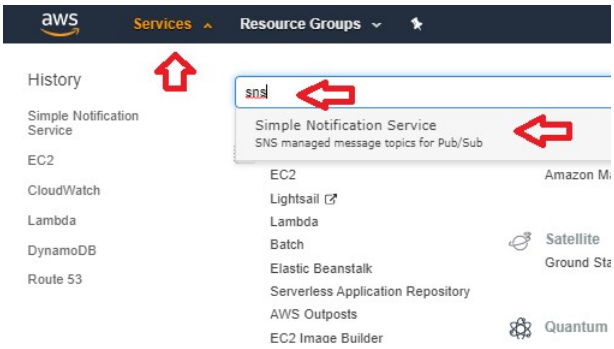
Frederique Retsema

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.

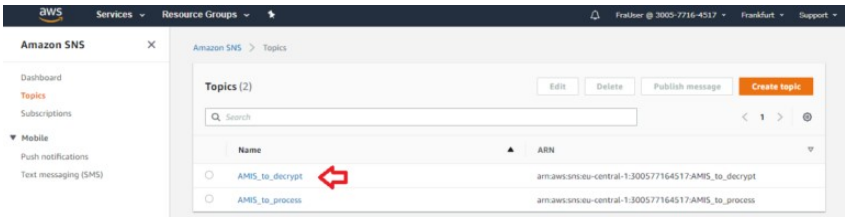
View all posts

POPULAR TAGS

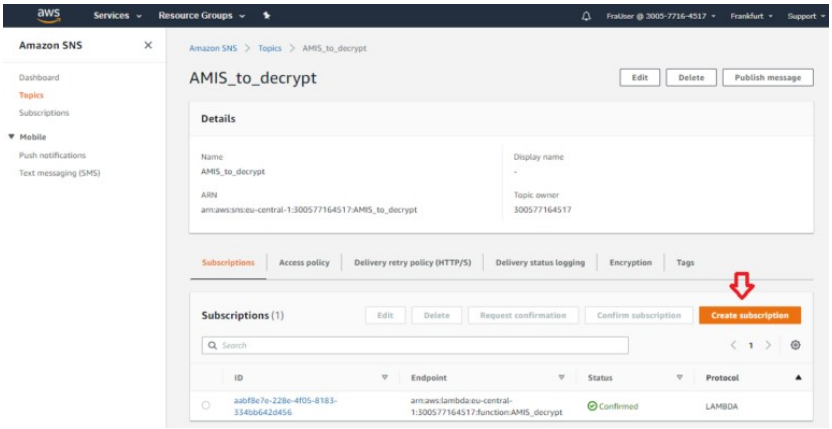
Agile analytical function apex api Architecture Azure BPEL bpm cloud container Database docker DVT html5 integration iot Java javascript jms json kafka kubernetes linux maven minikube monitoring node oci oracle cloud oracle cloud infrastructure oracle xml db OSB paas performance tuning pipeline plsql provisioning puppet push python REST saas Scrum soa sql Vagrant virtual box visualization vm XML



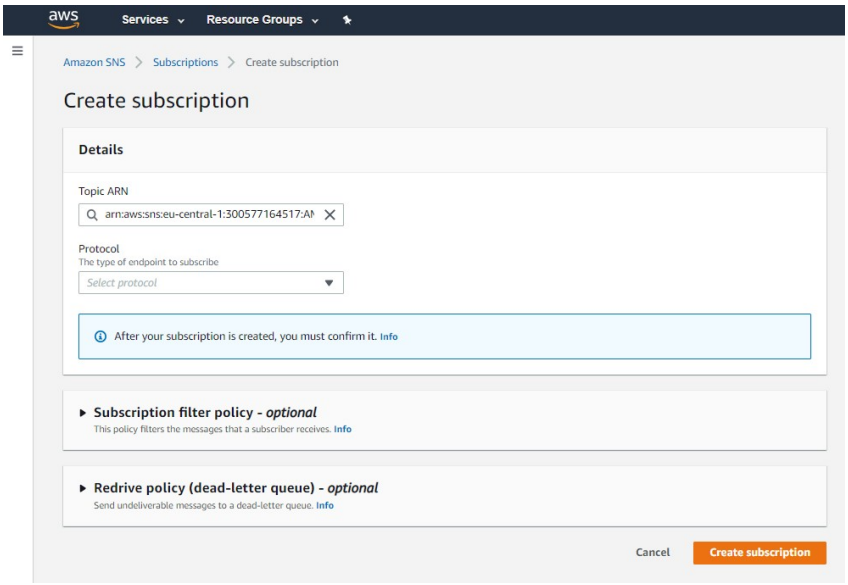
In this screen, you will see our two SNS topics. Click on the AMIS_to_decrypt link:



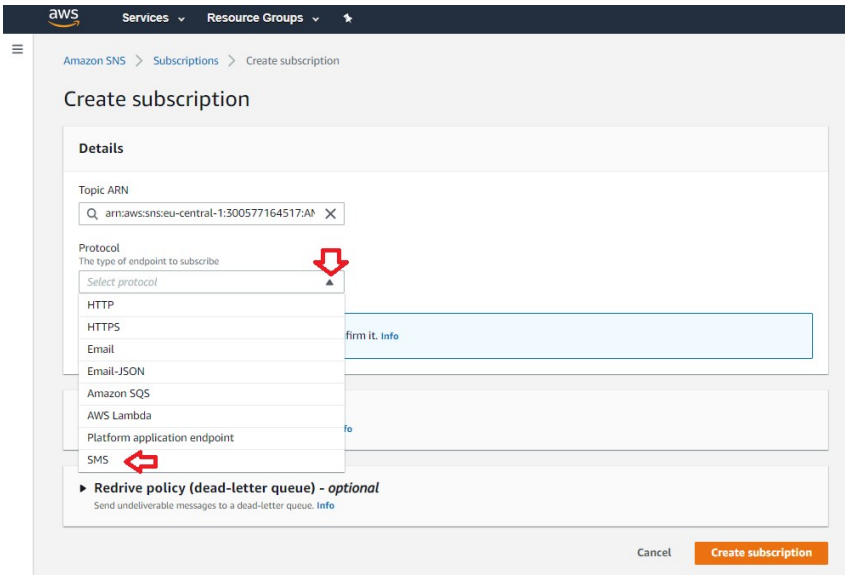
In the default tab, the subscriptions are shown. You see the subscription for the decrypt-function. Let's add a second subscription: let's assume that we want an SMS for every message that we get on this topic. Click on "Create subscription":



You can see now the options. You can send all messages, or just a selection of them – based on the keywords in the (json) message. Sometimes, messages cannot be delivered. SNS will retry up to 6 hours for mail, SMS or mobile push (or 23 days for Lambda functions) [2]. When that time limit is reached, the message will be send to the dead letter queue. That queue is defined in AWS SQS, the Simple Queue Service of AWS. When there is no dead letter queue, the message is discarded.



Click on the arrow next to the textbox for “Protocol”: you can see all the different type of end points to send messages to. Choose for SMS:



Fill in your phone number. For a telephone number in the Netherlands, with country code +31, this could be a number like +31612345678. When you play along, please fill in your own phone number and try this out! “Click on Create subscription” when you entered your phone number.

Tweets by @AMISnl

AMIS Conclusion
@AMISnl

Focus op resultaat! In de nieuwe ‘know-why-weekly’ heeft @Andre_van_Dalen het over de essentie van onderlinge samenhang bij projecten en het maken van de juiste keuze bij het type informatieverkeer in IT-systemen. Lees het volledige artikel 👉 amis.nl/nieuws/focus-o…

Focus op resultaat: de essentie van onderling…
Focus op resultaat: de essentie van onderlinge samenhang en de performance van
amis.nl

20h

AMIS Conclusion
@AMISnl

New Article : Add tests for Angular and Node.js webapp in Azure Pipelines (part 3) ift.tt/2VSLmif by Joost Luijben

Add tests for Angular and Node.js webapp in …
Adding tests for webapp in Azure Pipelines with Cobertura, karma and puppetteer for Angular and
technology.amis.nl

22h

AMIS Conclusion
@AMISnl

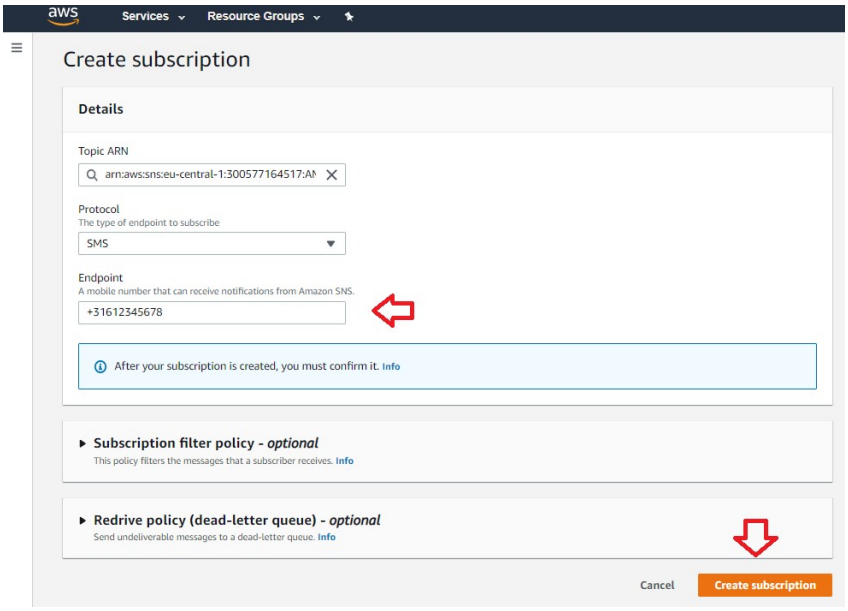
New Article : Creating a re-usable Vagrant Box from an existing VM with Ubuntu and k3s (with the Kubernetes Dashboard) and adding mysql, using Vagrant and Oracle VirtualBox ift.tt/2yUTgi3 by Marc Lameriks

Creating a re-usable Vagrant Box from an exist…
In a previous article, I shared with you the steps I took, to get k3s installed (with the Kubernetes
technology.amis.nl

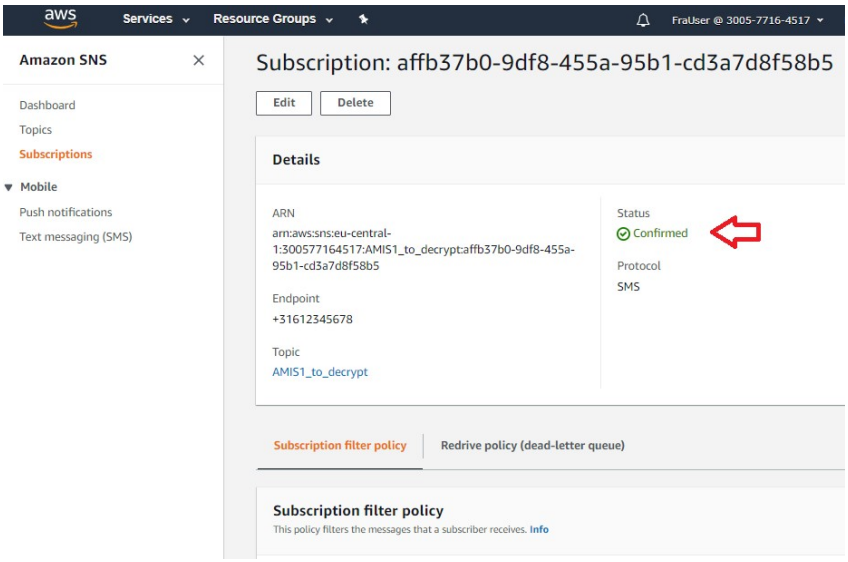
Apr 30, 2020

AMIS Conclusion
@AMISnl

New Article : Quick and easy: A multi-node Kubernetes cluster on CentOS 7 + QEMU/KVM (libvirt) ift.tt/3bSzsue by Maarten Smeets

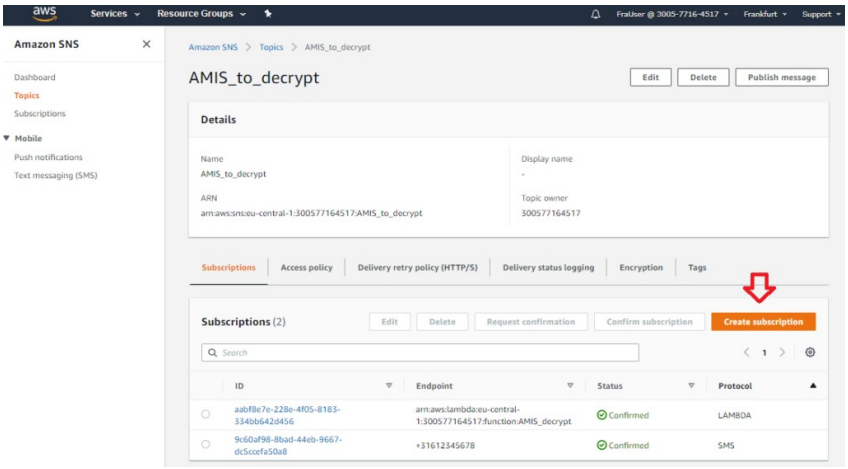


For SMS, AWS assumes that the phone number is correct: you see the status “Confirmed”.

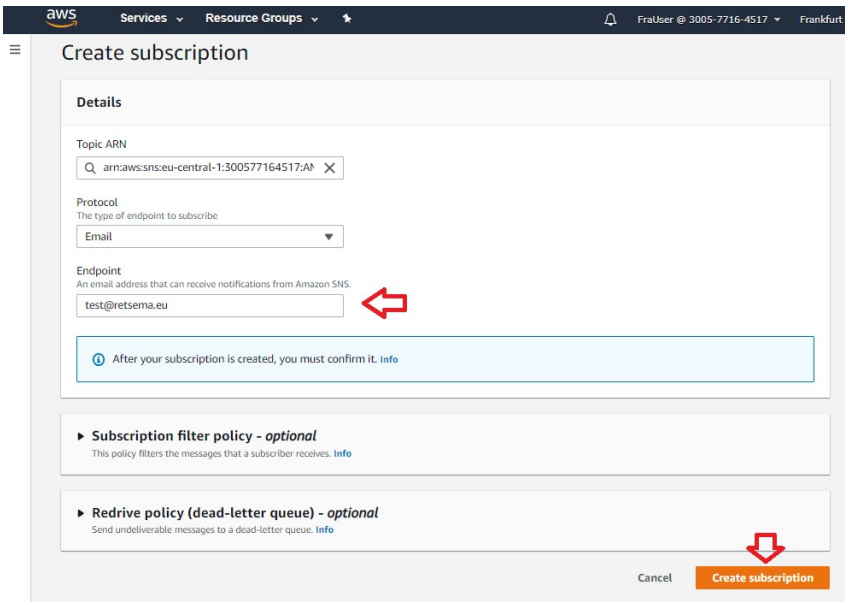


Add an e-mail address to the SNS topic

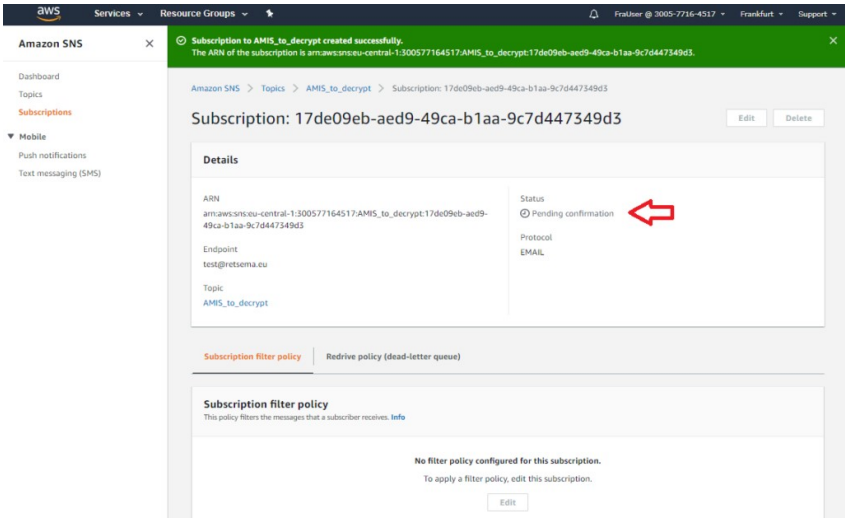
For E-mail, this works slightly different. Let’s try this out: click on Topics in the left menu, and click on AMIS_to_decrypt again. You see both subscriptions, one of type LAMBDA and the other of type SMS, both are confirmed.



Click on “Create subscription”, and click on the arrow under Protocol again. Choose for Email now. You can fill in your e-mail address. I filled in my test mailbox, please don’t send mail to that mailbox when you want to reach out to me, I will not read it. Now, click on Create subscription.

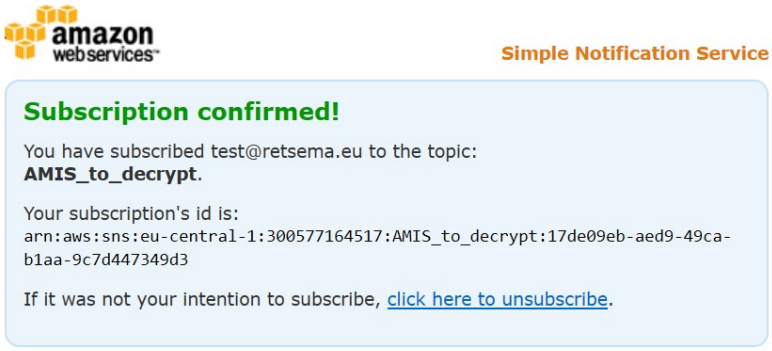


This subscription isn’t confirmed yet:

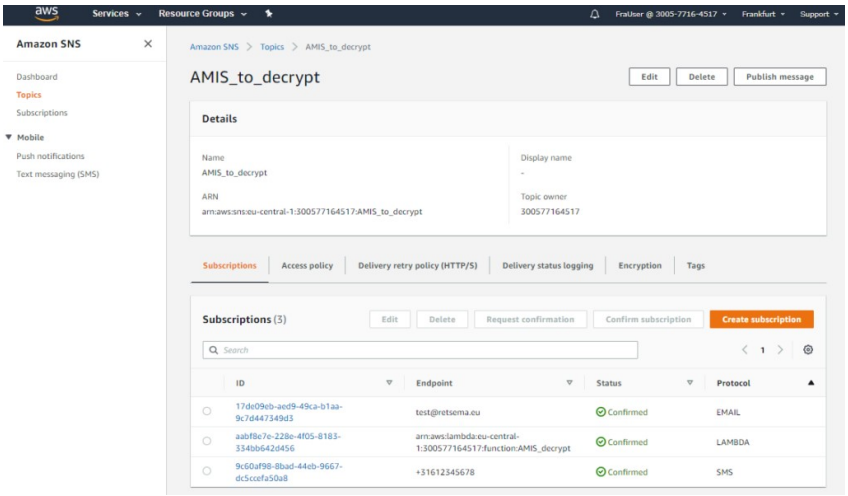


And because it isn't confirmed yet, SNS will not send e-mail to this e-mail address until it is confirmed. The E-mail that is sent looks like this one:

Click on Confirm subscription. A simple web page will be shown:



When you go to Topic and to the AMIS_to_decrypt SNS topic again, you will see that all three subscribers have been confirmed:



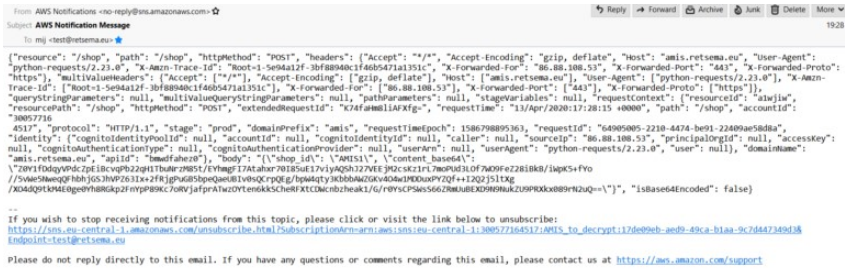
Let's try it out!

When you play along, go to the ~/AMIS-Blog-AWS/Shop-1/client directory and type the following commands (replacing the URL with your own URL or with the URL that is provided by the install script). This will send one message to the API Gateway:

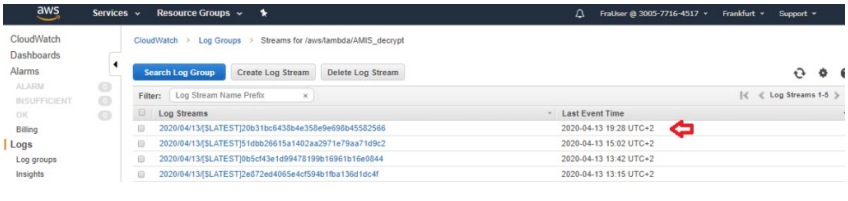
```
[vagrant@localhost client]$ ./encrypt_and_send.py AMIS1 https://amis.retsema.eu/shop

Shop id = AMIS1

Key alias = alias/KeyK-AMIS1

URL = https://amis.retsema.eu/shop

Data = {"shop_id": "AMIS1", "content_base64":
"Z0Y1fDdqyVPdcZpEiBcvqPb22qH1TbuNrzM85t/EYhmgFI7Atahxr70I85uE17viyAQShJ27VEEjM2csKz1rL7moPUd3LOf7WO9F+Z28iBkB/iWpK5+fYo//5vWe5NweqQFhbhjGSJhVPZ63Ix+2fRj

request_number = 1
```

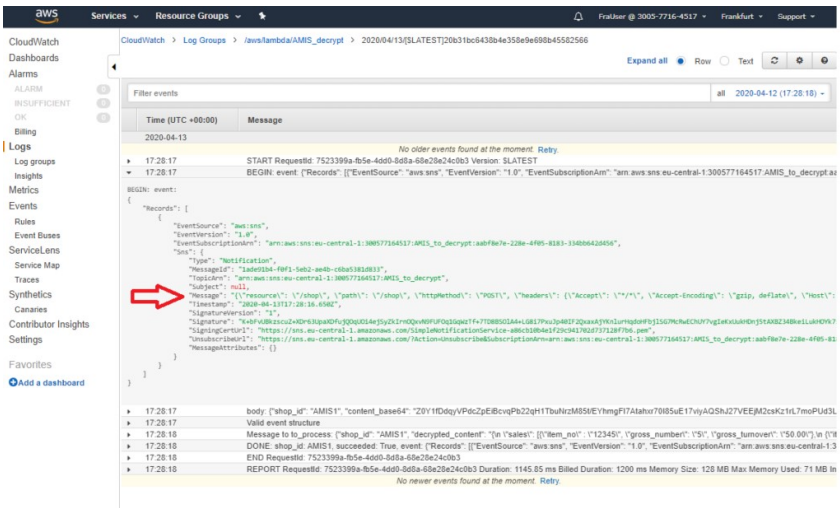
```
Status code = 200

Content = b'"OK"'
```

When you see status code 200 and OK, then you know that the delivery of the message to the Lambda accept function went well. Look in your e-mail folder: you most probably got an e-mail with all kind of information about the HTTP request.



When you look at the CloudWatch logs from the decrypt function, you will see that that Lambda function also got this message: go to the CloudWatch service and click in the left menu on “Log groups”, and click on the most recent link in AMIS_decrypt (please mind that the dates in CloudWatch are in GMT – so 2 hours earlyer than the dates in the email). The content of Message is the same as in the email:



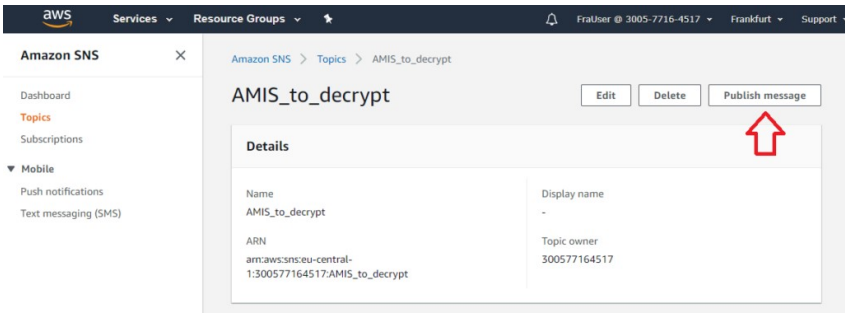


But what happened with the SMS? We didn’t get any (and yes, I used my own phonenumber instead of the bogus one in the images)…

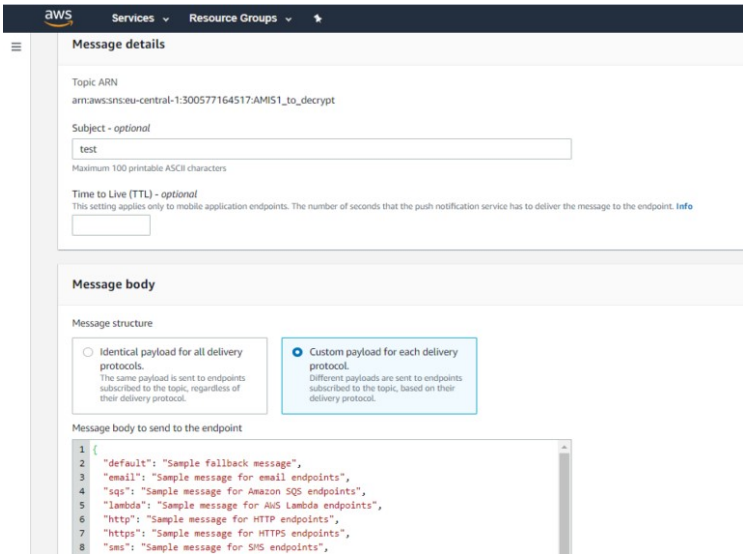
Why is this?

Sending different messages to different types of endpoints

Let’s go back to the AMIS_to_decrypt topic, and click on Publish Message:

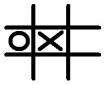


You can fill in a subject, let's say test. Further down, click on the button "Custom payload for each delivery protocol". You now see json code with different messages for each subscriber type. Change some of the texts if you wish, and scroll down to the bottom of the screen. Press on "Publish message": you now will have received both an SMS message and an e-mail. You will see subject "test" in the e-mail and the text behind email (which is by default "Sample message for email endpoints"). In the SMS, you will get the text "Sample message for SMS endpoints", the subject "test" isn't visible in SMS.



When the text is small enough, the text is sent to the SMS service. When the text is too large, the SMS is not sent.

Play along



I scripted the solution [3], and will go into more detail about other services (using screen prints of AWS) in other blogs. You can follow along and create this solution in your own environment, see the README.md file in the vagrant directory.

Links

[1] This is the third blog about this shop. Links to the previous blogs:

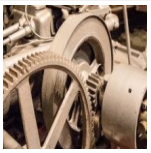
– Introduction: https://technology.amis.nl/2020/04/26/example-application-in-aws-using-lambda/

– Lambda and IAM: https://technology.amis.nl/2020/04/29/aws-shop-example-lambda/

[2] https://docs.aws.amazon.com/sns/latest/dg/sns-message-delivery-retries.html

[3] https://github.com/FrederiqueRetsema/AMIS-Blog-AWS , directory shop-1, see for a more detailed discription also the introduction blog.

Related Posts:



Differences between CloudFormation,

Creating policy's, groups and users in

Policies in AWS (2)

Example application in AWS using

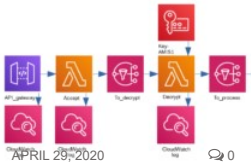
AWS shop example: Lambda

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

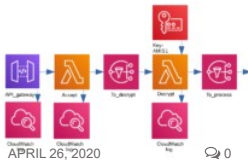


APRIL 29, 2020 0

AWS shop example: Lambda

APRIL 27, 2020 0

A Free Apache Kafka Cloud Service – and how to quickly get started with it



APRIL 26, 2020 0

Example application in AWS using Lambda

Leave a Reply

Enter your comment here...