APRS Notify Documentation

2024.04.15R1

CHRIS NIGGEL

Description

Monitor Firmware

Prerequisites

You must have a functioning Arduino IDE on your development system. See the readme located at https://github.com/zzaxusl0a/APRS_notify for a parts list and schematic for additional components.

Installation

Instructions on how to configure the LightAPRS board can be found here:
https://github.com/lightaprs/LightAPRS-2.0

At the time of publishing, you must replace the ZeroAPRS library provided by QRP Labs with the one provided in this repository (v1.0.2) in order to handle messages without GPS coordinates correctly. Simply replace the ZeroAPRS directory and its contents.

Server-Side Components

Prerequisites

Your development environment will need to have the following libraries installed:

(use *pip install <library>*)

- Boto3 (https://boto3.amazonaws.com/v1/documentation/api/latest/guide/quickstart.html)
- Twilio (https://www.twilio.com/docs/libraries/reference/twilio-python/index.html)
- Urllib3 (https://pypi.org/project/urllib3/)

You will also need to have the AWS CLI available. More information on configuring the CLI can be found in the AWS documentation:
https://docs.aws.amazon.com/cli/latest/userguide/getting-started-prereqs.html

You will need an AWS account and a paid-for Twilio account in order to send and receive SMS messages. SMS Messaging in the US and Canada requires an approved 10DLC Campaign. Read https://help.twilio.com/articles/1260800720410-What-is-A2P-10DLC for more information.

Installation

Create the SDB database

The following command must be run using the AWS CLI. SimpleDB does not have a graphical or web interface:

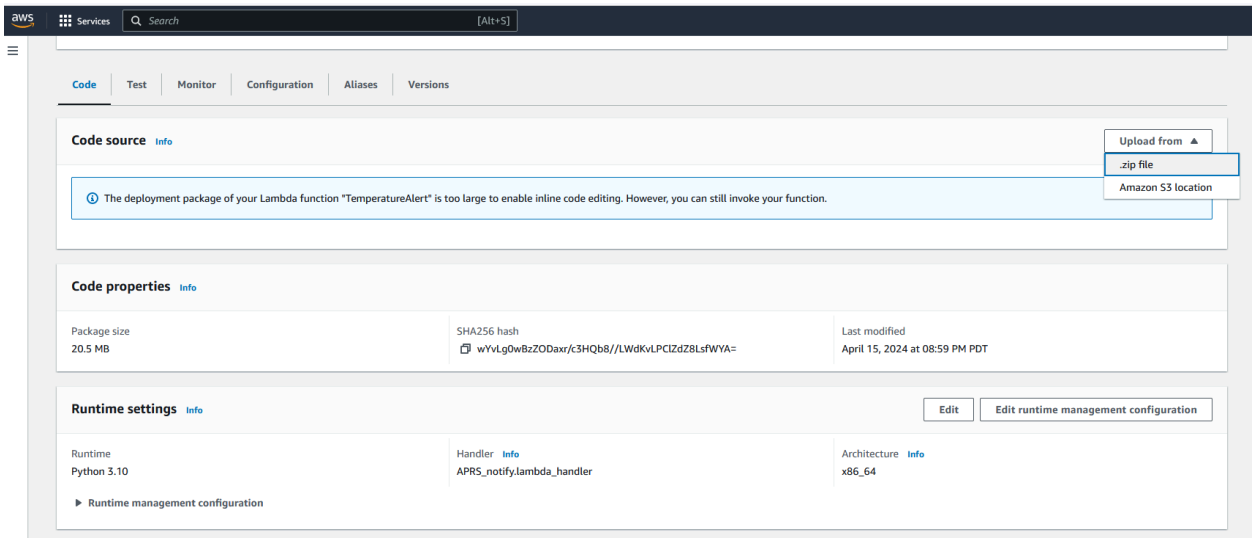
```
aws sdb create-domain –domain-name APRS_tracker
```

Attributes are automatically added on an as-needed basis by the application.

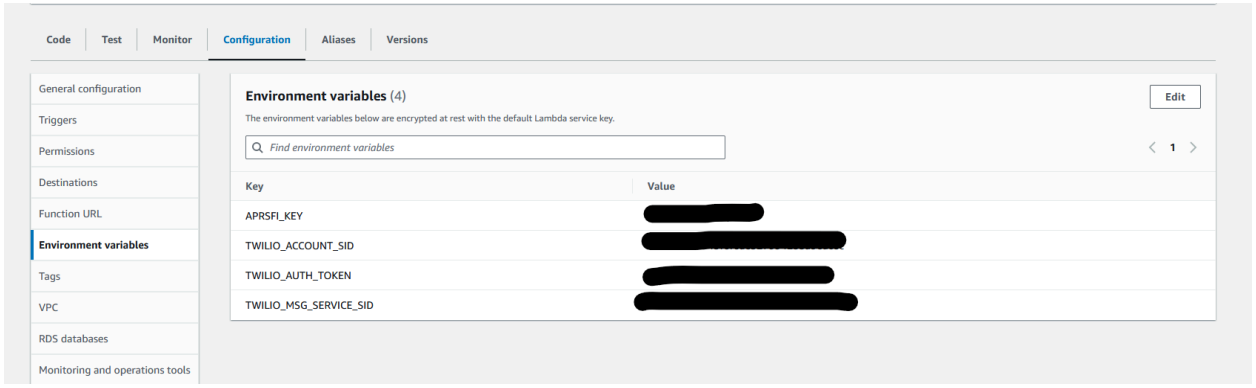
Install the Lambdas

Open the AWS Console and find the TemperatureAlert Lambda. If it doesn't exist, create it using x86_64 architecture and the Python 3.10 or greater execution environment. 128MB memory is more than enough.

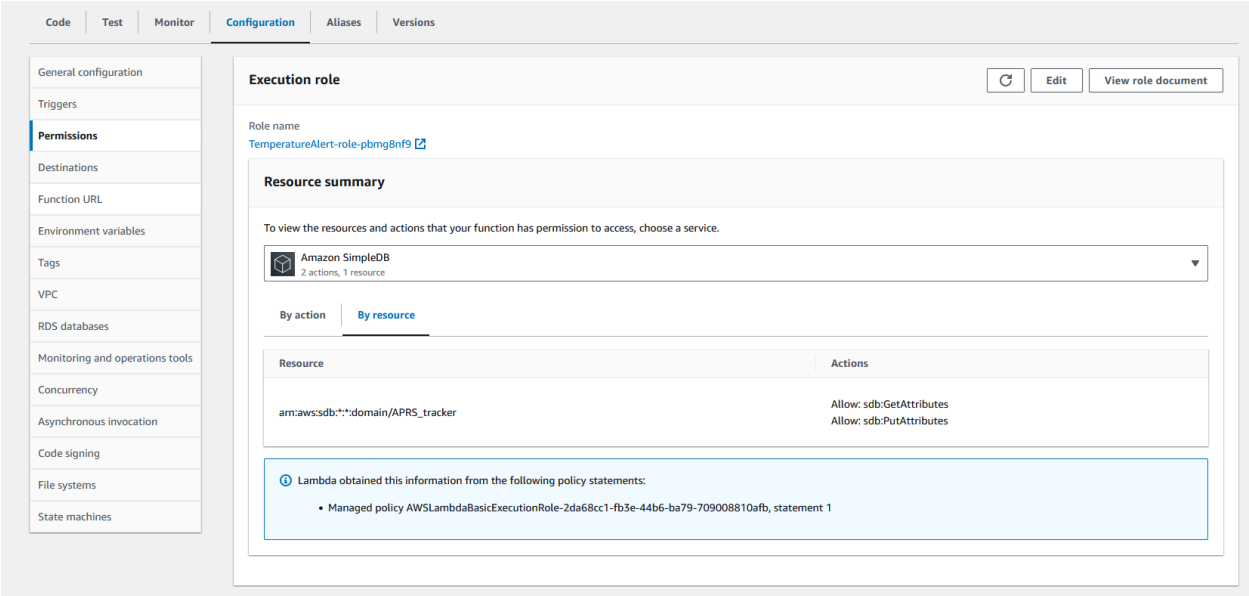
Click on “Upload From” and select ZIP. Upload the zipped packages (for information on how to create AWS Lambda deployment packages, see appendix)



Next, create the required environment variables



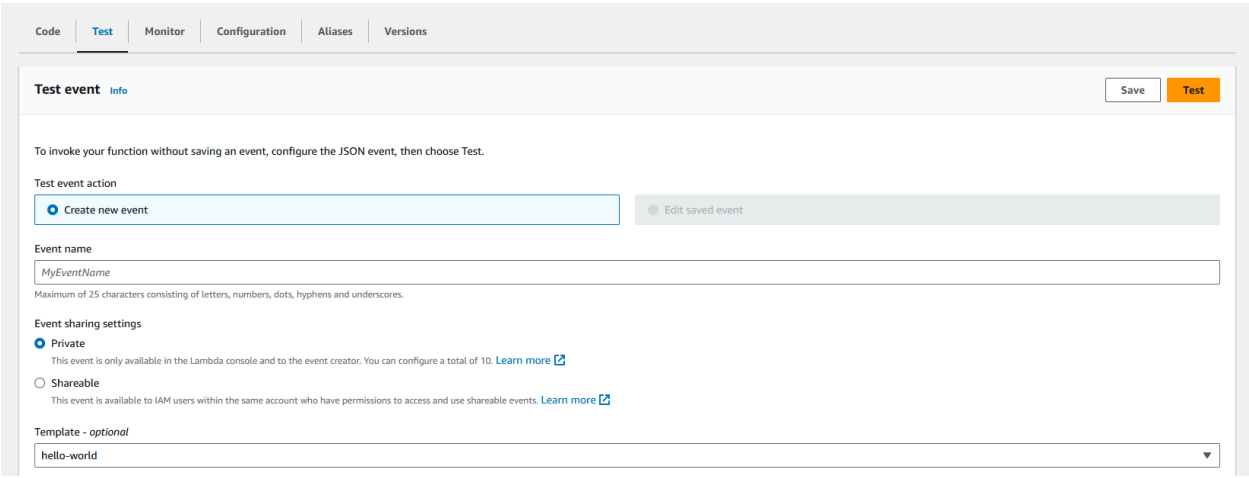
CloudWatch permissions are automatically granted by AWS. Set up permissions to access the SDB database



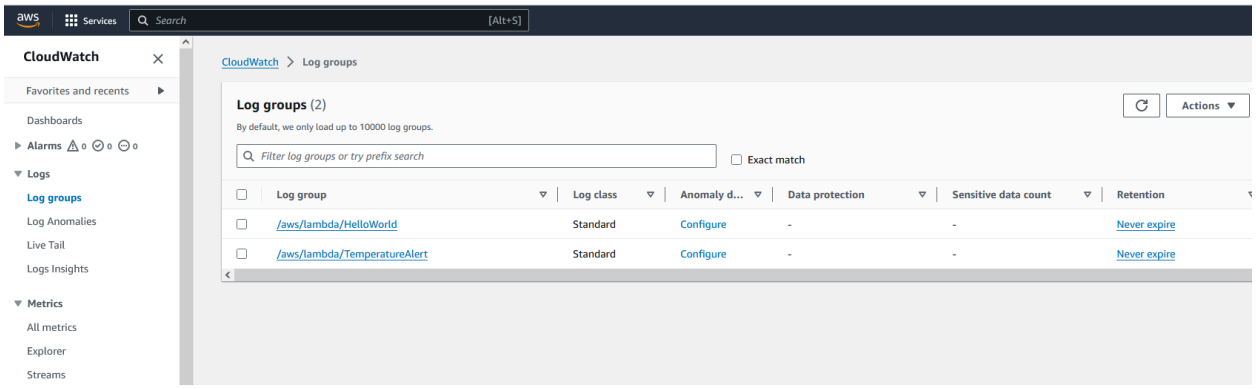
Now, create the APRS_Log_Watchdog Lambda using the same process. Additional SimpleDB permissions are not required for the APRS Log Watchdog Lambda.

Subscribe the Watchdog to the Event Stream

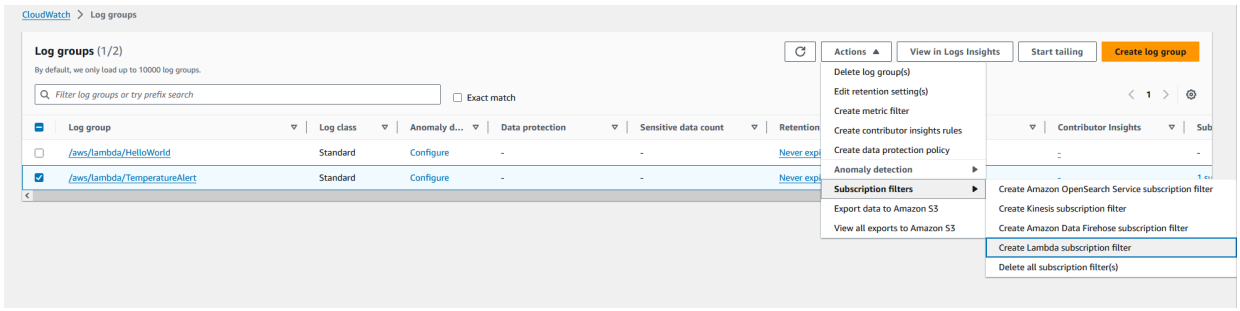
Begin by testing the TemperatureAlert Lambda if you haven't already. The "hello-word" template is sufficient.



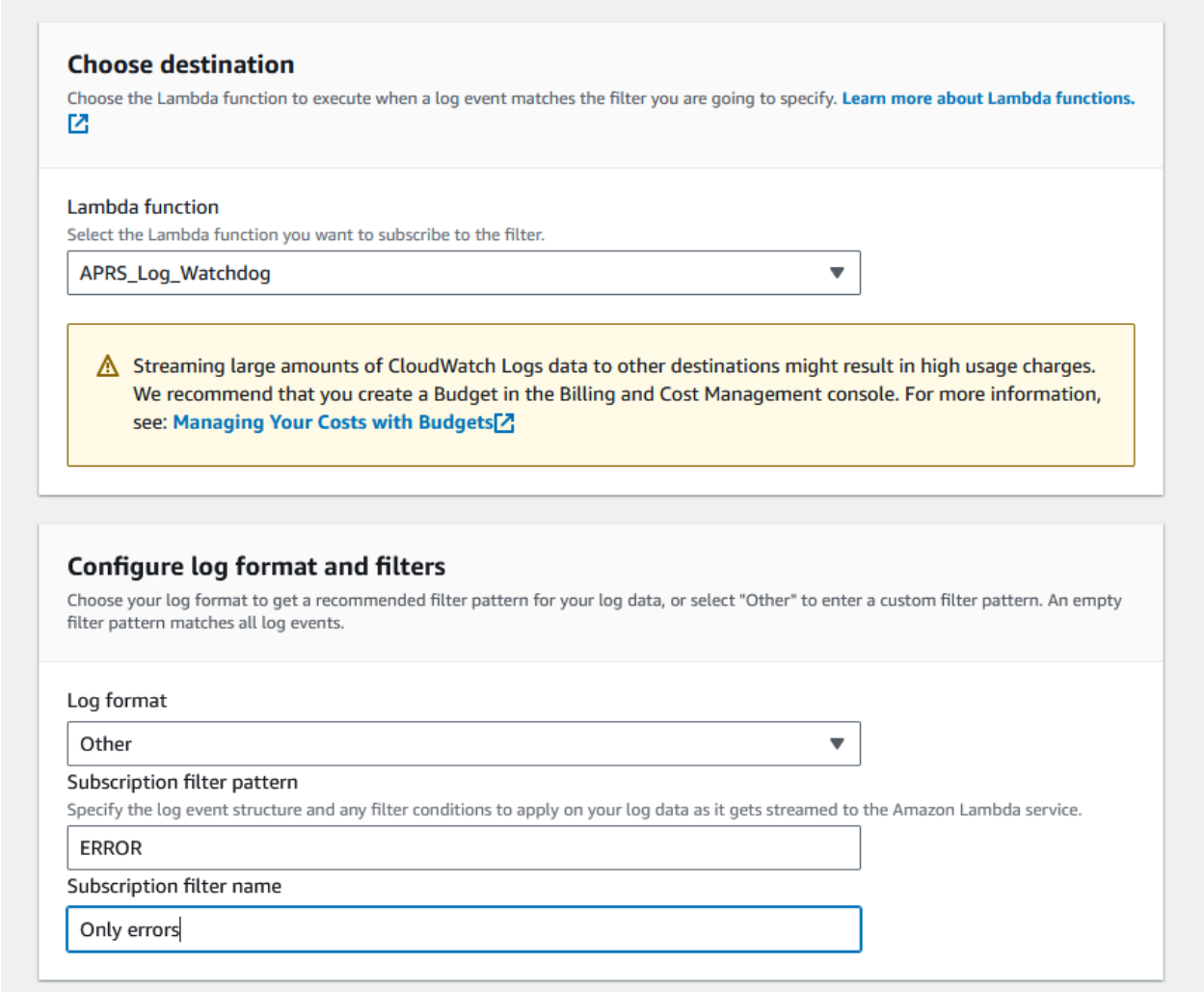
Now, move to the CloudWatch console, and find the TemperatureAlert Log Group



Select the Checkbox next to the Log Group, then go to the Actions menu and navigate to *Subscription filters / Create Lambda subscription filter.*



Subscribe the Watchdog Lambda to the CloudWatch log stream and filter for the word ERROR



You can use the Test pattern section at the bottom to validate the configuration. Once done, click the **Start Streaming** button at the bottom to save.

Appendix

Packaging Lambdas

AWS requires that libraries be included in the package for deployment.

Resource: https://docs.aws.amazon.com/lambda/latest/dg/python-package.html

To create the deployment package (project directory)

Navigate to the project directory containing your lambda_function.py source code file. In this example, the directory is named my_function.

```
cd my_function
```

Create a new directory named package into which you will install your dependencies.

```
mkdir package
```

Note that for a .zip deployment package, Lambda expects your source code and its dependencies all to be at the root of the .zip file. However, installing dependencies directly in your project directory can introduce a large number of new files and folders and make navigating around your IDE difficult. You create a separate package directory here to keep your dependencies separate from your source code.

Install your dependencies in the package directory. The example below installs the Boto3 SDK from the Python Package Index using pip. If your function code uses Python packages you have created yourself, save them in the package directory.

```
pip install --target ./package boto3
```

Create a .zip file with the installed libraries at the root.

```
cd package
zip -r ../my_deployment_package.zip .
```

This generates a my_deployment_package.zip file in your project directory.

Add the lambda_function.py file to the root of the .zip file

```
cd ..
zip my_deployment_package.zip lambda_function.py
```

The Zip file can now be uploaded to AWS through the console.

Additional Resources

http://www.aprs.org/doc/APRS101.PDF

https://medium.com/@osaimola/creating-a-serverless-sms-twitter-app-with-aws-lambda-twilio-b33f63254cbb

https://docs.aws.amazon.com/lambda/latest/dg/python-logging.html

https://docs.aws.amazon.com/lambda/latest/dg/python-package.html

https://awscli.amazonaws.com/v2/documentation/api/latest/reference/sdb/index.html#cli-aws-sdb

https://www.twilio.com/docs/messaging/guides/track-outbound-message-status#how-to-track-outbound-message-status