



Linux Academy

Hands-On Training

Using SNS
and SQS for
Infrastructure
Automation

Contents

Introduction.....	3
Scenario.....	3
Why Use SQS for Automation?.....	3
Workflow.....	3
Connect to the Console and Server.....	4
Don't Lose Your LOLCATS!: Set the SNS Topic.....	4
Create Subscription Endpoints.....	4
SQS Endpoints: Alternative Method.....	5
Automation Overview.....	5
Now Let's Save the Cats!.....	6

Introduction

In this lab, we are going to simulate sending an **s3:ReducedRedundancyLostObject** event to **Amazon Simple Notification Service** (SNS). When this event is sent, it contains information about a “lost” object. This is useful because objects stored with Amazon Reduced Redundancy Storage class only have a 99.99% durability, compared to S3’s standard “eleven nines” durability.

It is considered best practice to store easily reproducible items with the RRS class for cost reduction, and then automate a “reaction” in the event one is lost, and the SNS s3:ReducedRedundancyLostObject event is triggered.

Scenario

You run an image manipulation service for cat pictures. The service allows users to upload photos of cats, that are then filtered with a black and white filter by your service. According to best practices, the original cat pictures should be stored on S3 standard storage class for eleven nines durability, while the black and white images can be stored with RRS to reduce costs.

With s3:ReducedRedundancyLostObject enabled, we will then need to automate the process of recreating the black and white photos from the source if the photo is lost. The process should be automated so users can show off their cat pictures without any interruption of service.

In our scenario, we will have a lolcat image that was uploaded and “lost” due to RRS. We will then create the automation process that will recreate the object in our S3 bucket.

Why Use SQS for Automation?

SQS allows us to create a distributed environment. If our worker instances that are supposed to process the automation request are busy or fail, we do not lose the request, which is stored inside messages in SQS. Any worker instance can poll SQS and be able to process the task. This allows for fault tolerance due to outages or other issues. If the instances that need to process the automation request is unavailable, it will be stored in the queue for up to 14 days, until a worker processes it.

Workflow

To accomplish this automation task with AWS, we will use the integration AWS already provides between S3 and SNS. With SNS, you will set up an SNS topic with a subscriber endpoint directed to SQS. SQS will receive the message containing information about the lost object, and a Python worker periodically polling SQS for new messages will automatically add a black and white filter and re-upload the altered image.

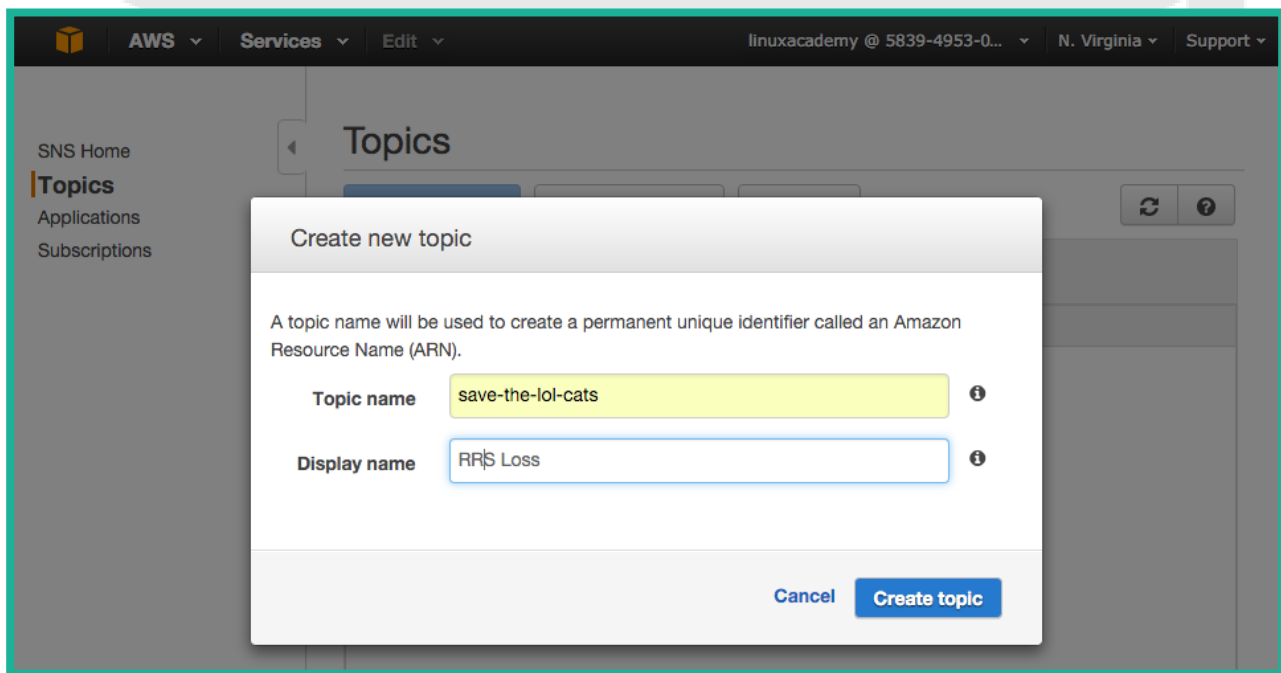
Connect to the Console and Server

Select **Start Lab** from the page for this lab on LinuxAcademy.com. It may take up to five minutes for the lab to generate.

Once ready, you will be provided with a link to the AWS console, a username and password for the console, and the public and private IPs for a server. Click on the AWS console URL and sign in with the credentials given. Open a terminal on your workstation computer, and then ssh into the provided server using the linuxacademy username, the provided public IP address, and the password provided (123456).

Don't Lose Your LOLCATS!: Set the SNS Topic

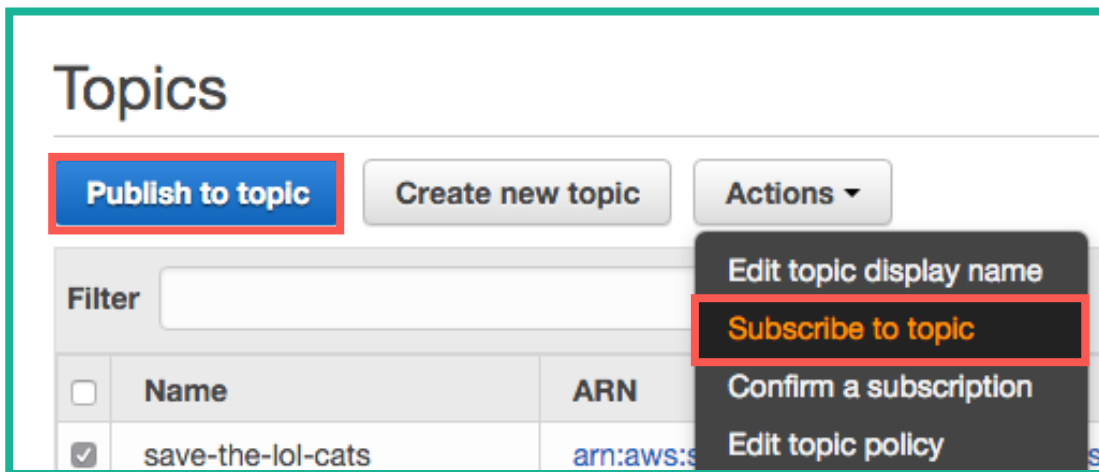
From the **AWS console**, open the **SNS** service page. From here, a new topic needs to be created. Select **Topic** from the sidebar, then click **Create New Topic**.



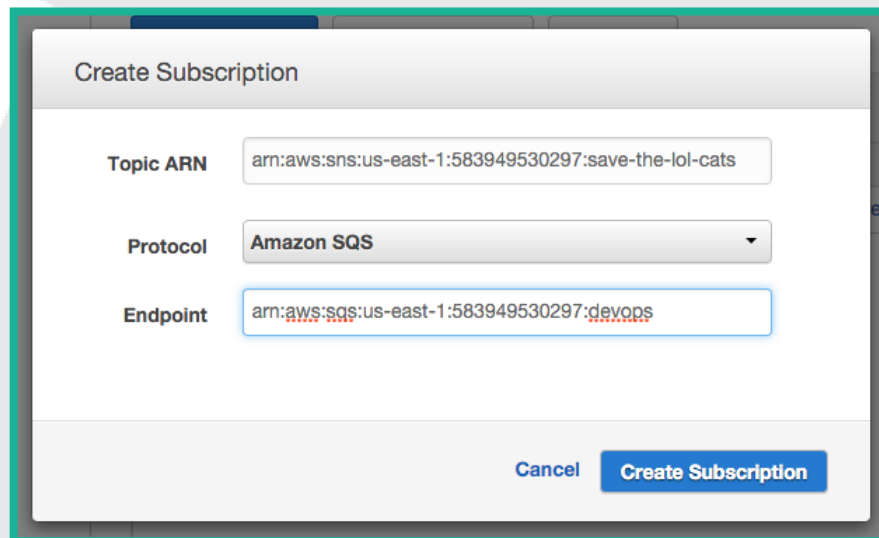
We used *save-the-lol-cats* as the **Topic Name** and *RRS Loss* for the **Display Name**.

Create Subscription Endpoints

For us to save the lolcats, we need to set up multiple subscription endpoints. First set up a subscription endpoint to an SMS or email account. This will be what notifies you when an object is lost. Choose **Subscribe to topic** from the **Action** menu to do this.



A second subscription endpoint will be created to an Amazon SQS queue that has already been created. To do this, you will need the *Amazon Resource Name* (ARN) for the queue. Open the **SQS** dashboard, and select the queue that is already there. The ARN will be available in the details. Again, select **Subscribe to topic** then set the **Protocol** to *Amazon SQS*. Paste the ARN to the endpoint.



SQS Endpoints: Alternative Method

There is a second method for creating SQS endpoints. If you have already followed the step below, you do not need to replicate them in this method, but we suggest being aware of the option.

From the **SQS** dashboard, select your *devops* queue, and from the **Actions** menu choose **Subscribe Queue to SNS Topic**. From here, you can choose a topic from the dropdown.

Automation Overview

When working with AWS, automation takes place when we glue different services together to create a process. In this example, we are simulating S3 sending an RRS lost object notification to SNS, which will then send the message to the subscriber endpoints. In this example, the endpoints are the SQS queue

that will receive the message. We will then ask our “worker” instance to poll SQS for any messages, and process these messages by executing Python code. This will create the black and white lolcat and put it right back into S3 as though it were never lost.

From here, connect to your given server, the information of which is provided on the LiveLab page.

Once connected use `ls` to see the images directory and `restoreobject` script.

In a real environment, the script would be set to run via Cron to consistently poll the SQS queue for new messages. A message will contain the information concerning the object that needs to be restored.

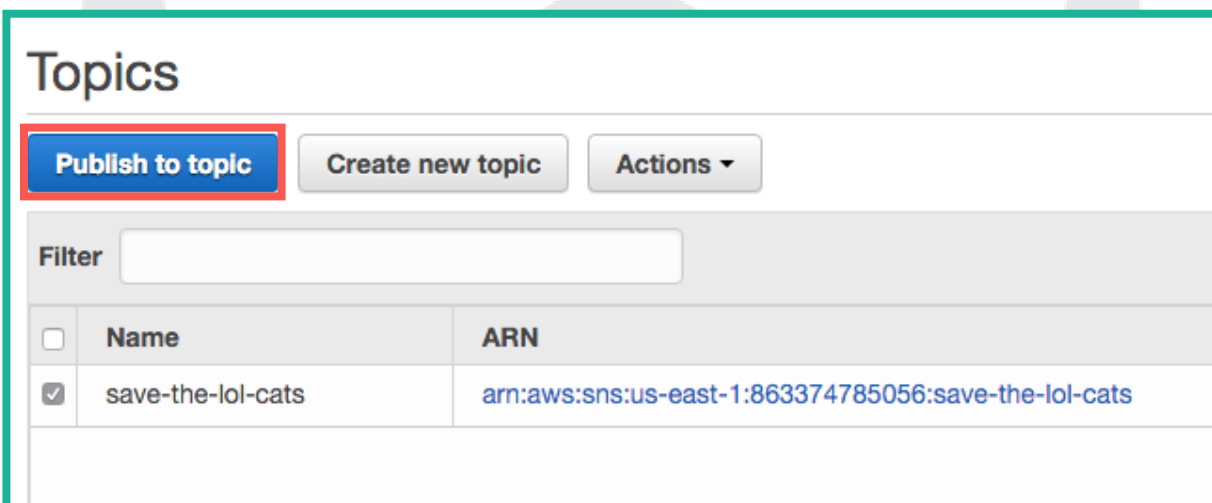
Type `cat restoreobject` to review the script’s code, which contains the following steps:

1. Polls the SQS queue that receives SNS notifications.
2. Processes any message in the queue by parsing the data received from RRS.
3. Recreates the lost object by adding a filter to the original image.
4. Reuploads the image to the same location on S3.

Now Let’s Save the Cats!

With SNS and SQS set up, and having reviewed the Python automation script, everything is prepared for use. Now, to make it work.

From your **SNS** dashboard, select your topic and press **Publish to topic**.



The screenshot shows the AWS SNS console 'Topics' page. At the top, there are three buttons: 'Publish to topic' (highlighted with a red box), 'Create new topic', and 'Actions' with a dropdown arrow. Below these buttons is a 'Filter' input field. Underneath the filter is a table with two columns: 'Name' and 'ARN'. The table contains one entry with a checked checkbox, the name 'save-the-lol-cats', and the ARN 'arn:aws:sns:us-east-1:863374785056:save-the-lol-cats'.

	Name	ARN
<input checked="" type="checkbox"/>	save-the-lol-cats	arn:aws:sns:us-east-1:863374785056:save-the-lol-cats

In the **Message** area, copy and paste the **Solution** found on the lab page at the Linux Academy website. Leave the **Message format** set to *Raw* and press **Publish message**.

Publish a message

Amazon SNS enables you to publish notifications to all subscriptions associated with a topic as well as to an individual endpoint associated with a platform application.

Topic ARN:

Subject:

Message format: ☒ Raw ☐ JSON

Message:

```
{
  "Service": "Amazon S3",
  "Event": "s3:ReducedRedundancyLostObject",
  "Time": "2014-10-13T15:57:02.089Z",
  "Bucket": "bucketname",
  "Key": "lol_cat.jpg",
  "RequestId": "5582815E1AEA5ADF",
  "HostId": "8cLeGAmw098X5cv4Zkwcmo8vvZa3eH3eKxsPzbB9wrR+YstdA6Knx4lp8EXAMPLE"
}
```

Time to live (TTL):

Now open your **SQS** dashboard, and review the **Messages Available** column — there will be one waiting under your devops queue. This means that SQS has received the message, and all that needs to happen is for the worker to poll the queue and replace the object.

To do this, return to your terminal and run the `restoreobject` script, with a unique bucket name:

```
./restoreobject lol_cats
```

Normally, the bucket will not need to be defined and will be parsed from the JSON value, but in this sample instance, we need to create a bucket.

To ensure it worked, you can either return to the SQS dashboard — notice there is no longer a message available. You can also view your newly-created S3 bucket, and see the resulting `lol_cat.jpg`!