**Stream and Analyze Log Data with Kinesis Firehose, Lambda, S3, Glue, and Athena**

Created by[Frank Kane](https://deloittedevelopment.udemy.com/user/frank-kane-2/)

**AWS Workspace**

**120-180 minutes**

In this lab, you are a data engineer assisting your system operations team. To help troubleshoot issues in near-real-time, your sysops need a central, searchable repository of syslog messages across their entire fleet of servers. Your goal is to stream log data from EC2 using Kinesis Firehose, Lambda, and a Kinesis Agent into S3, where it will be queried by Athena through a Glue Data Catalog.

Our server fleet has grown to the point where our Ops team is having trouble maintaining them. Instead of logging into each server individually to troubleshoot issues, they could respond much more quickly if they could pull up syslog messages across the fleet from one place. You need to build this in such a way that doesn't just introduce more servers for them to maintain, and minimizes costs. I think Kinesis Firehose, Lambda, S3, Glue, and Athena will provide the tools we need to build this system in a totally serverless manner where we only pay for usage.

All of our servers have a "messages" log in syslog format that we want to query from a central location. You'll set up a free EC2 instance as a test server, and build a system to stream its log into an S3 data lake. From there, you'll use Glue and Athena to interactively query this streaming log data. The Ops team is OK with data being up to one minute old, so we can use Kinesis Firehose, Lambda, and the Kinesis Agent to power and transform the stream.

**How you'll work**

Your project has been broken into a set of tasks. To complete these tasks, use the provided workspace. You can launch your workspace by clicking below or using the button in the top right of the screen.

Tasks

1-Create an s3 bucket for our data

2-Set up a Kinesis Data Firehose stream that streams syslog data in Json format in S3

3-create EC2 to test the system with

4-Create an IAM role to allow EC2 to use Fire store

5-Install and configure the AWS Kinesis agent on our EC2 host

6-Ensure our message lo has the appropriate file permission

7-Run the Kinesis agent, and confirm log data is written into S3

8-Create a table in the Glue Data Catalog for our log data using a Glue Crawlwe

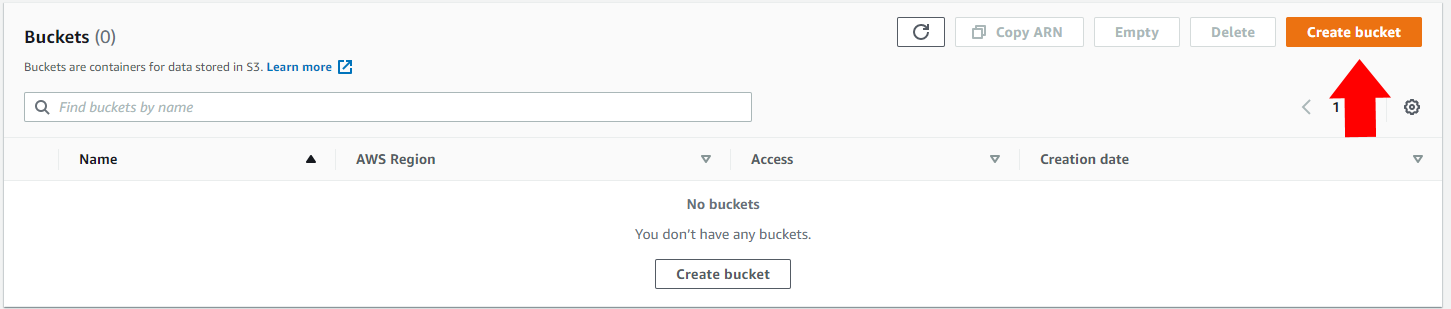
9-Use Athena to interactively query our log

10-Clean up

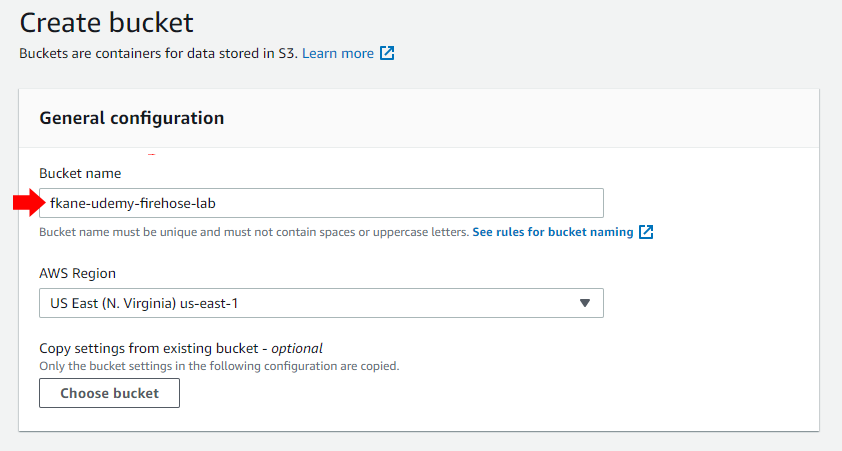
Create an s3 bucket for our data

First we'll need a central repository for our log and query data. Set up a new S3 bucket for our log data, and a folder within it for our query data. For security purposes, ensure the bucket is encrypted at rest and is not publicly accessible.

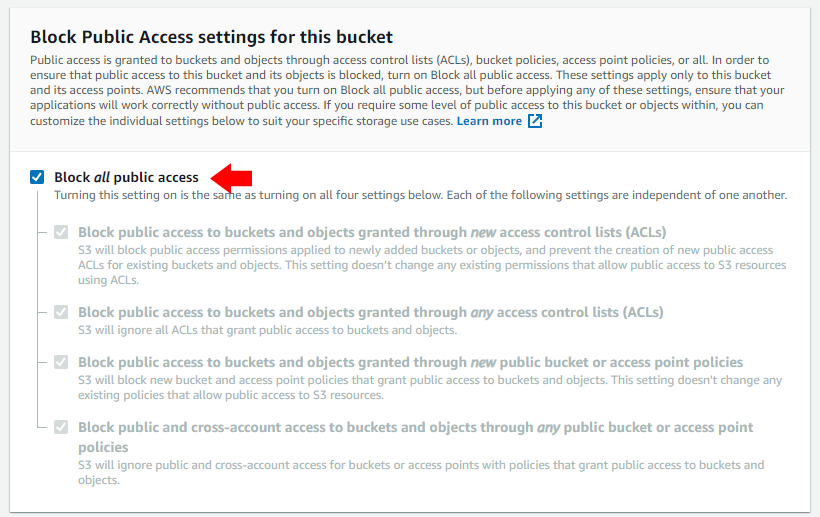
1. From the AWS Management Console, select **S3**.
2. Click on **Create bucket**.



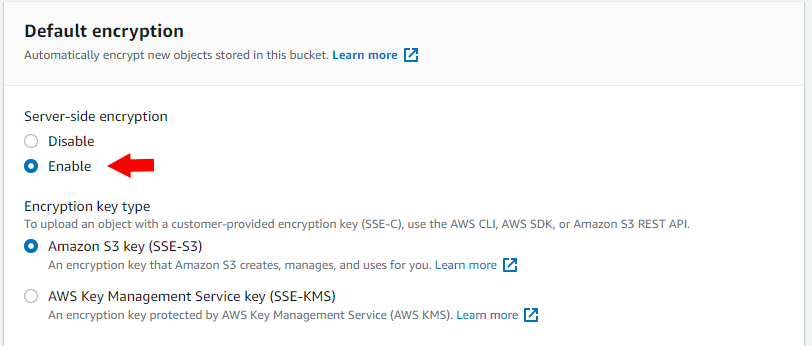
1. Give your bucket a globally unique name, such as *yourname-udemy-firehose-lab* (substituting your own unique domain or name.)



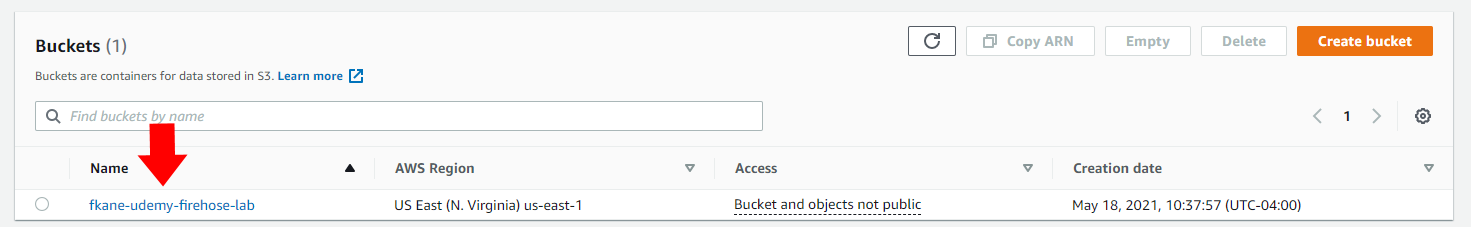
1. Ensure **Block *all* public access** is selected.



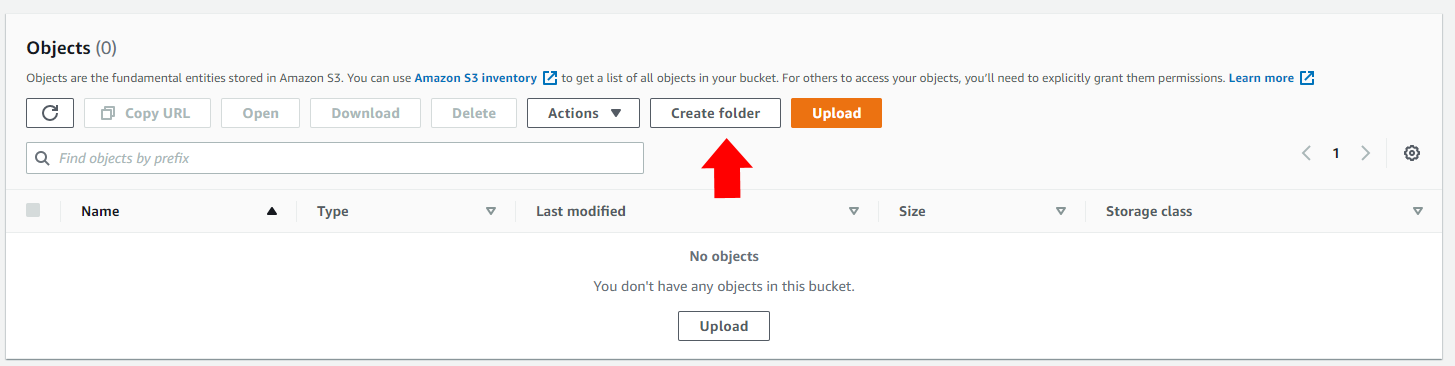
1. Under *Default encryption*, choose **Enable**. The *Encryption key type* should remain at the default *Amazon S3 key (SSE-S3)*.



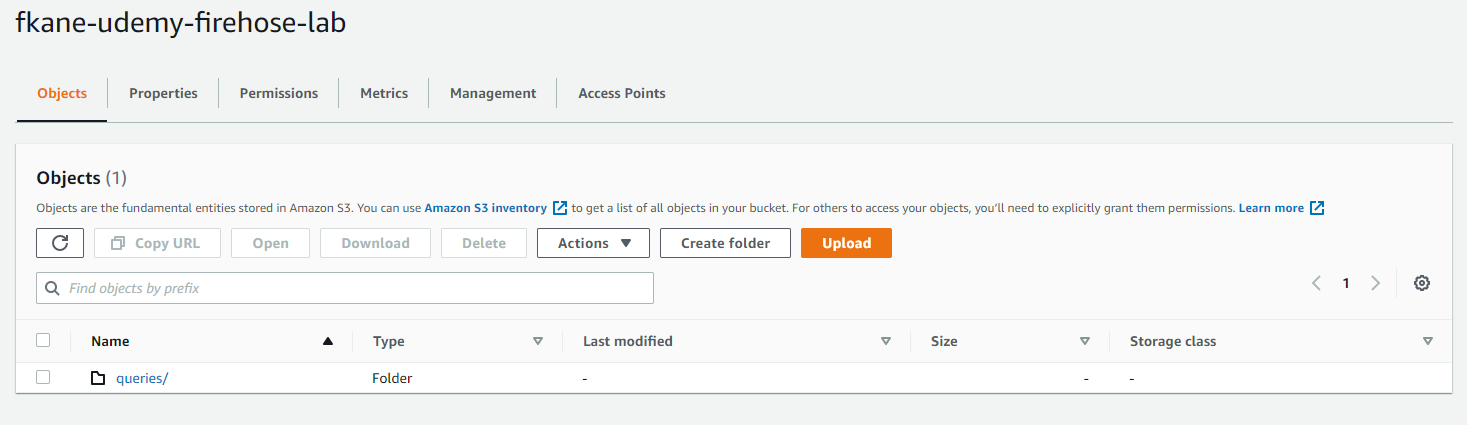
1. Click the **Create bucket** button at the bottom of the screen.
2. Select the bucket you just created (your bucket name is unique and will be different.)



1. Click the **Create folder** button.



1. Enter the *Folder name* **queries**and click **Create folder.**
2. You should now see the *queries*folder in your bucket, like this:

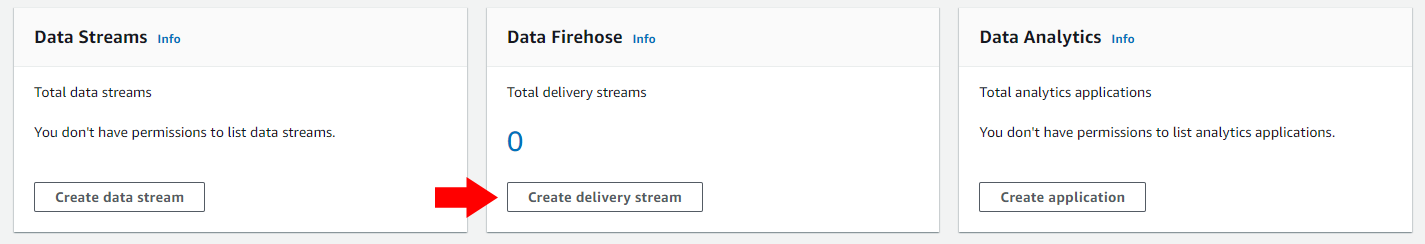


<https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html>

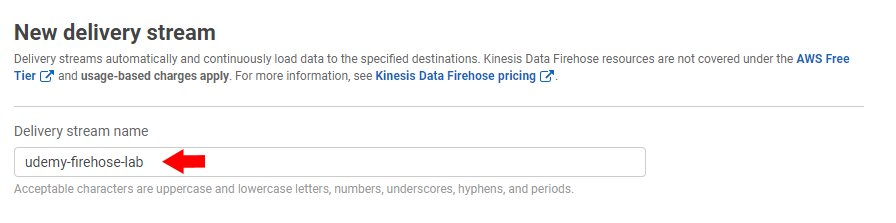
Set up a Kinesis Data Firehose stream that streams syslog data in Json format in S3

You'll need to create a Kinesis Data Firehose delivery stream to transport log data from our servers into S3. We want this data to end up structured for SQL queries, so use a Lambda function to transform the data from syslog to JSON format along the way. Watch out for the default timeout setting on the Lambda function, though - it will be processing a lot of data, and needs to be increased to at least one minute. This data will be used for troubleshooting operational emergencies, so make sure the data is transported as quickly as Firehose will allow while setting this up.

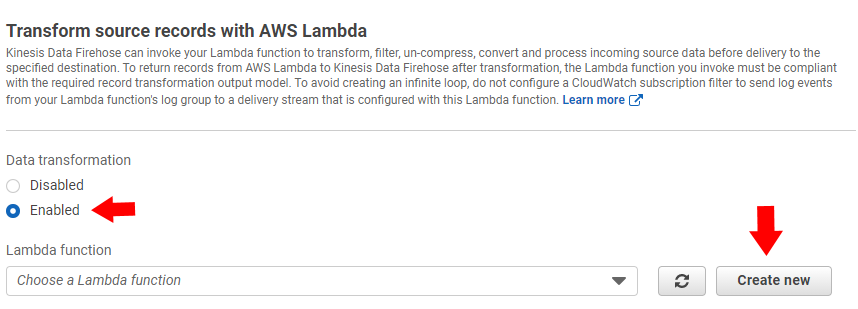
1. From the AWS Console or *Services* menu, select **Kinesis**.
2. Under *Data Firehose*, click the **Create delivery stream** button.



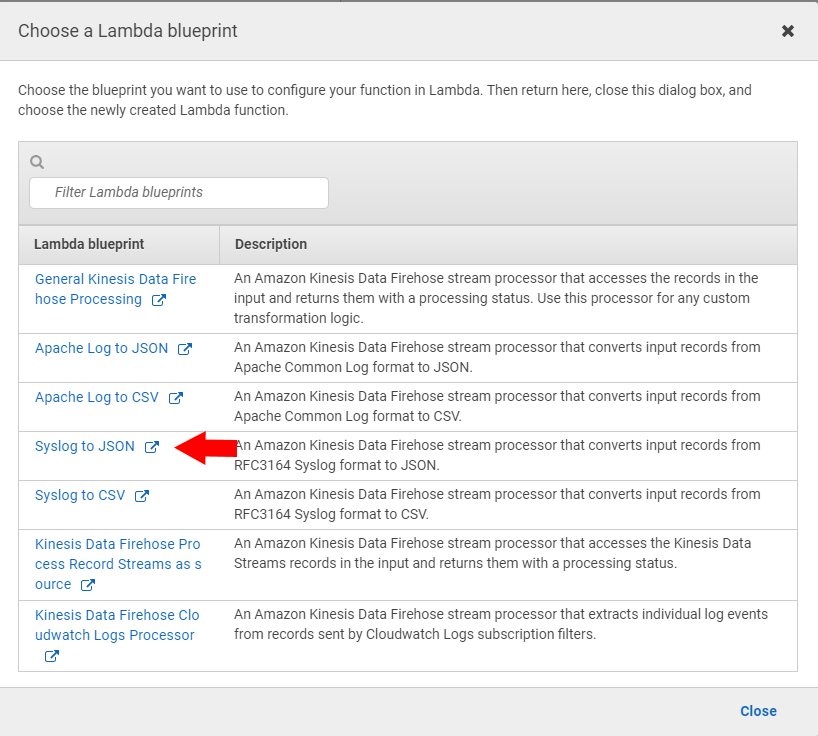
1. Give the new delivery stream a name, such as *udemy-firehose-lab*.



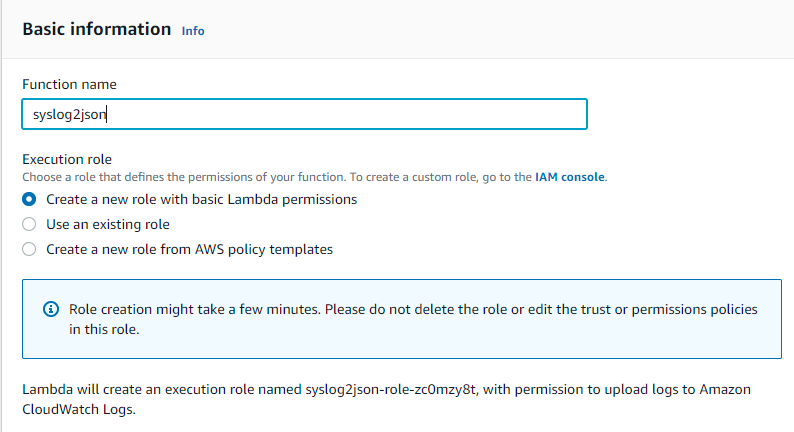
1. Leave the *Source* set to **Direct PUT or other sources** and leave *Enable server-side encryption for source records in delivery stream* unchecked. In a production setting we might choose to use a KMS key to encrypt data if it were sensitive.
2. Click the **Next** button to proceed to the *Process records* screen.
3. Under *Transform source records with AWS Lambda*, select **Enabled**, then click the **Create new** button.



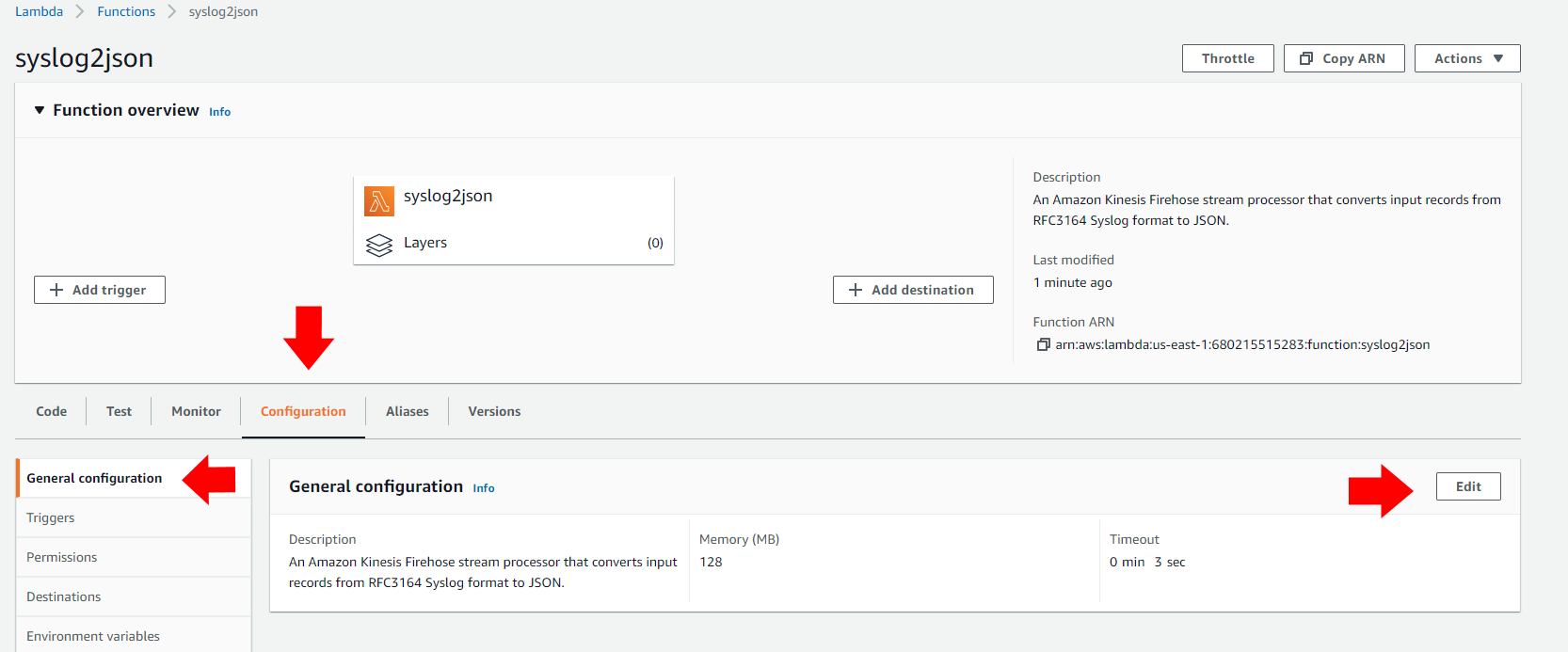
1. Select the Lambda blueprint **Syslog to JSON**; a screen for configuring your Lambda function will appear in a new window.



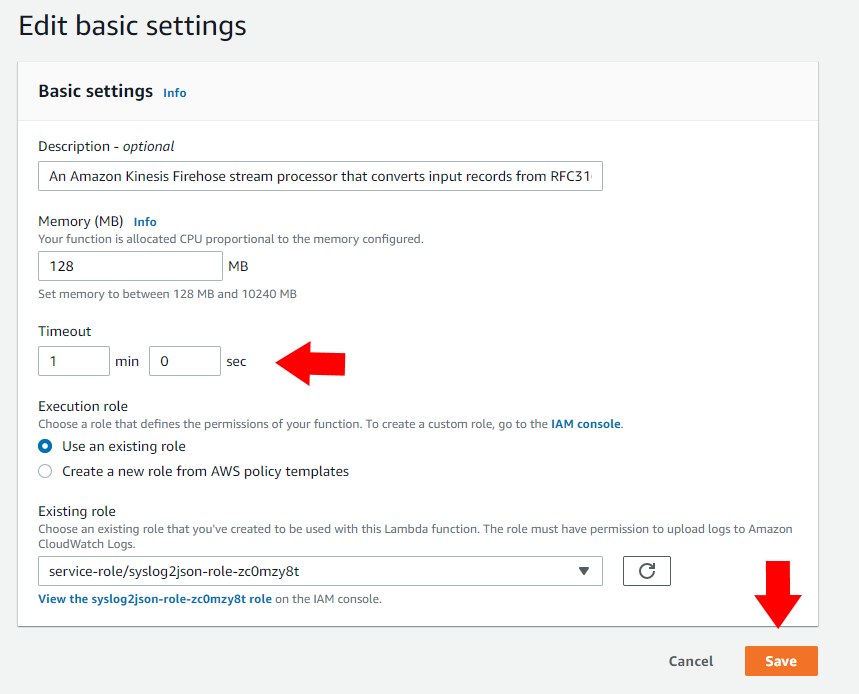
1. Give your Lambda function a name, such as *syslog2json*. Leave the *Execution role* set to **Create a new role with basic Lambda permissions**.



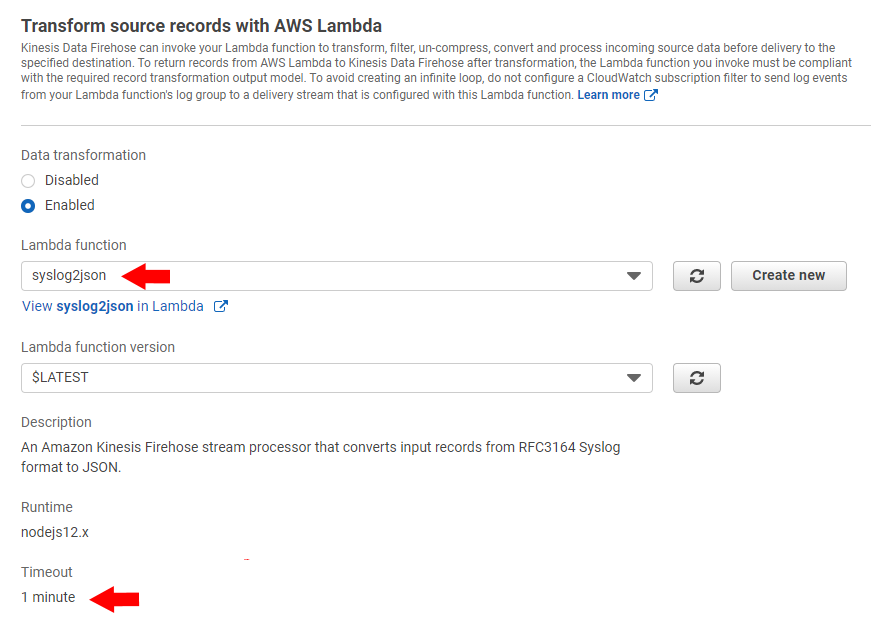
1. Click the **Create function** button at the bottom of the screen.
2. Your new Lambda function will appear. The default timeout of 3 seconds is too short, so we must change it. Choose the *Configuration* tab, and from there *General configuration*. Click the *Edit* button.



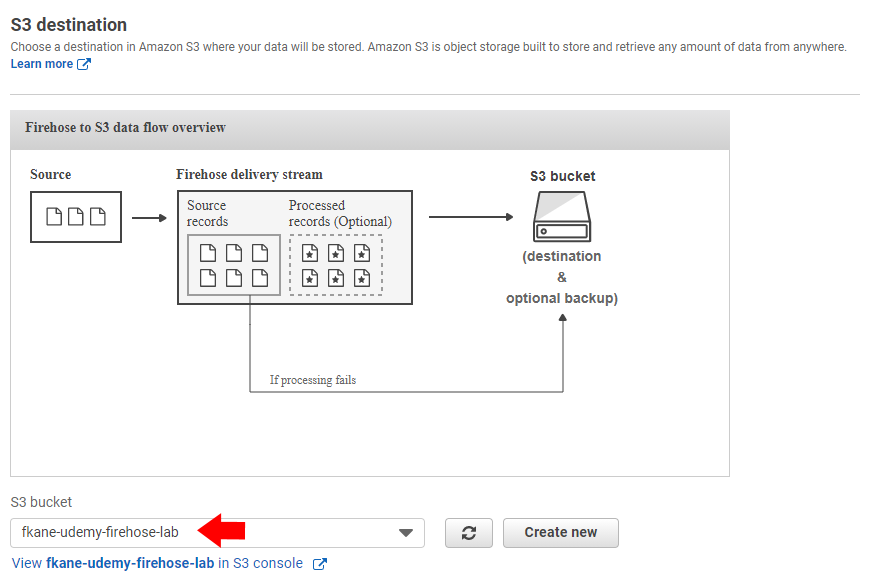
1. Change the *Timeout* value to 1 minute, and click the **Save** button.



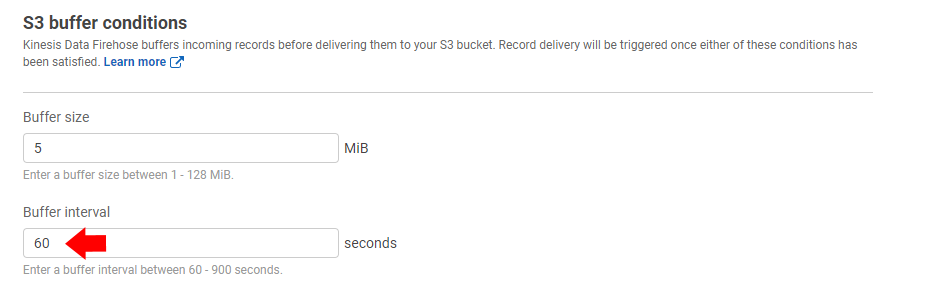
1. Return to the Kinesis Firehose screen, and close the *Choose a Lambda blueprint* pop-up window. Under *Transform source records with AWS Lambda*, choose the *syslog2json* function you just created. Confirm the *Timeout* is set to 1 minute.



1. Leave *Convert record format* set to **Disabled**, and click the **Next** button.
2. On the *Choose a destination* screen, select **Amazon S3** for the destination. Scroll down, and select the S3 bucket you created in Task 1. Your bucket name is unique and will be different from that shown below.



1. Click the **Next** button to proceed.
2. To ensure data is as up to date as possible, under *S3 buffer conditions* change the *Buffer interval* to **60** seconds.



1. The remaining settings on this screen may be left set to their defaults. In a production setting involving sensitive data, you might choose to enable S3 encryption here to encrypt data in transit to S3. Click **Next** to proceed.
2. Double-check everything on the *Review* screen, and click the **Create delivery stream** button.

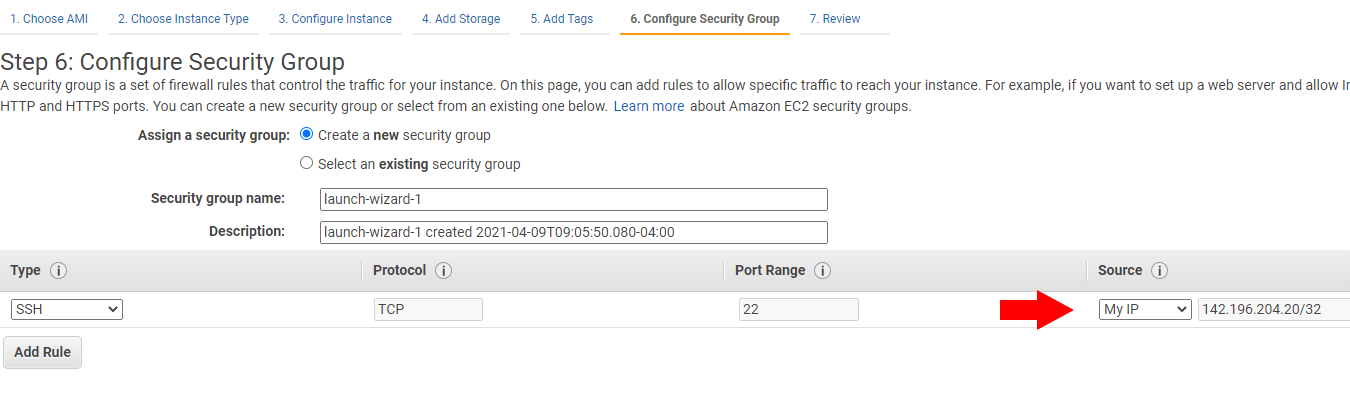
<https://docs.aws.amazon.com/firehose/latest/dev/data-transformation.html>

<https://docs.aws.amazon.com/firehose/latest/dev/basic-create.html>

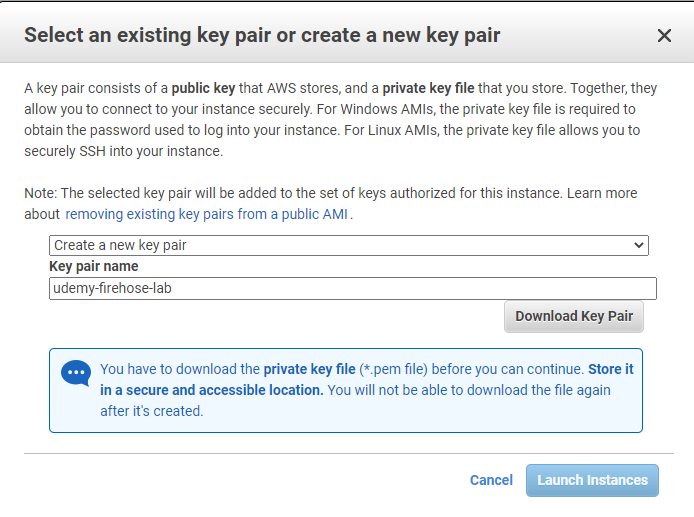
create EC2 to test the system with

Create a free-tier EC2 Amazon Linux instance so we can test the system without touching production servers. After launching the new instance, make sure you can connect to it.

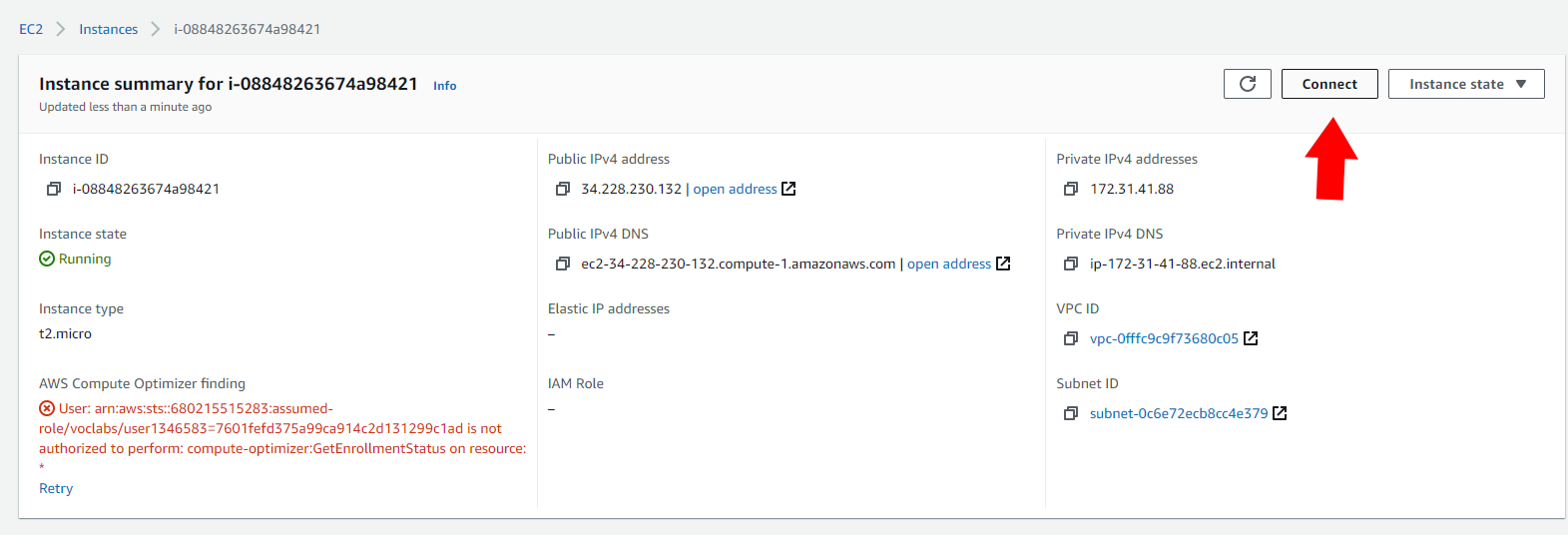
1. Select**EC2**from the AWS Management Console home page or Services menu.
2. Click the **Launch instance** button to enter the Launch Instance wizard.
3. On the first two screens, select a free-tier-eligible instance such as **Amazon Linux 2 AMI**running on a **t2.micro** instance type. Click **Next: Configure Instance Details.**
4. Ensure the instance is running within your default VPC, and select **Configure Security Group** from the top menu.
5. If you plan to connect to this instance from your desktop, change the **Source** to **My IP**instead of opening the instance up to the world.



1. Click **Review and Launch** and then **Launch**.
2. Create a new key pair for connecting to this database and download it.



1. After downloading the key pair, click **Launch Instances**. Click **View Instances** and wait a few minutes for your instance to be created.
2. Select your new Instance ID, then hit the **Connect** button.



1. Select **SSH client** and follow its instructions for connecting to your instance using your private key file. If you are connecting from Windows instead of a terminal on Linux or MacOS, you will need to install an SSH terminal application such as PuTTY, which will first require using the PuTTYGen tool to convert the private key file into the format PuTTY needs to connect.

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html>

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>

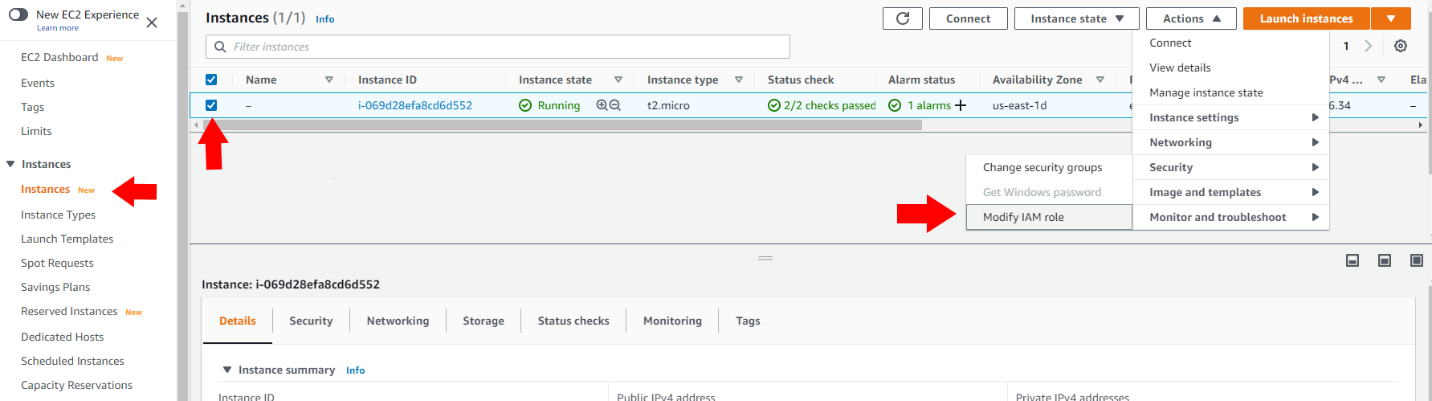
<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html>

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/security-group-rules-reference.html>

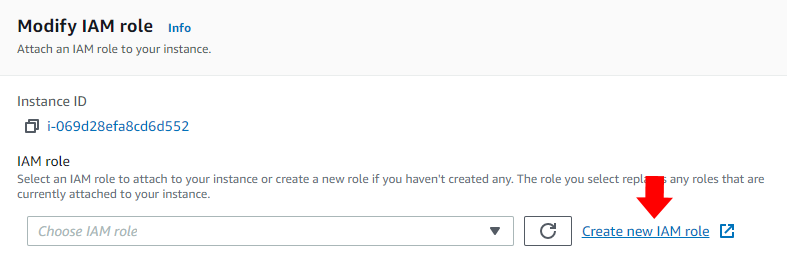
Create an IAM role to allow EC2 to use Fire store

You'll need to grant permissions for your EC2 instance to talk to other AWS services, or the Kinesis Agent won't be able to connect to the services it needs. Modify the IAM role on your EC2 instance to grant it Administrator access while we are developing.

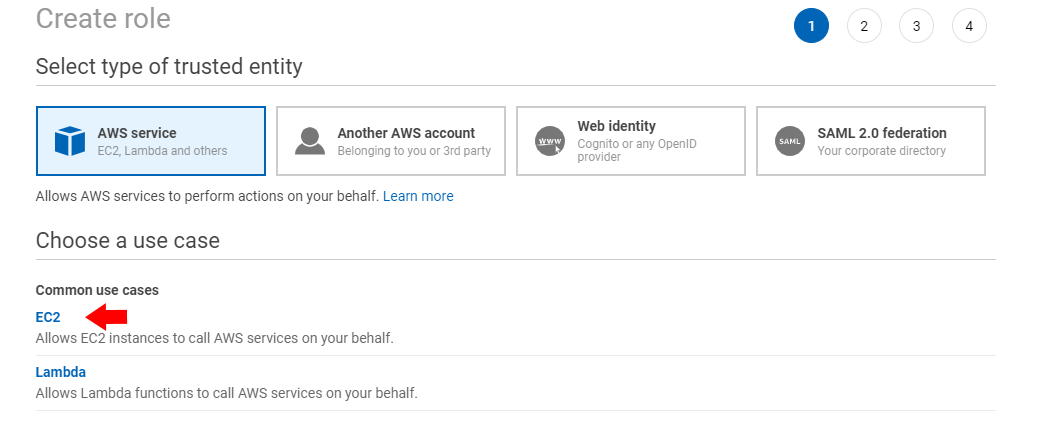
1. From the EC2 *Instances* dashboard, check the instance you just created, and from the *Actions* menu select *Security / Modify IAM role*



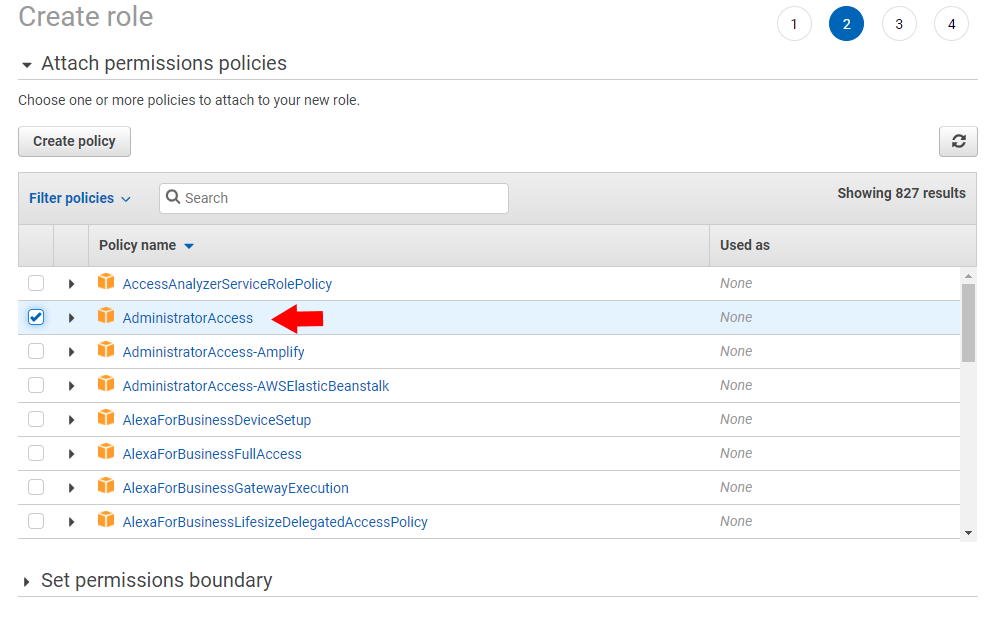
1. Click **Create new IAM role**. IAM will open in a new window.



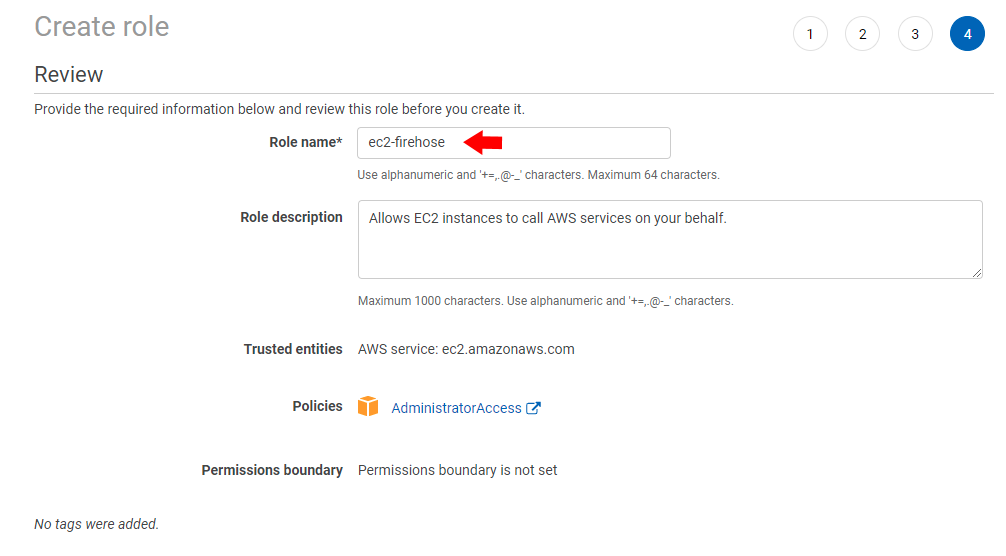
1. Click **Create role**.
2. Set the type of trusted entity to *AWS service* and select **EC2** for the use case.



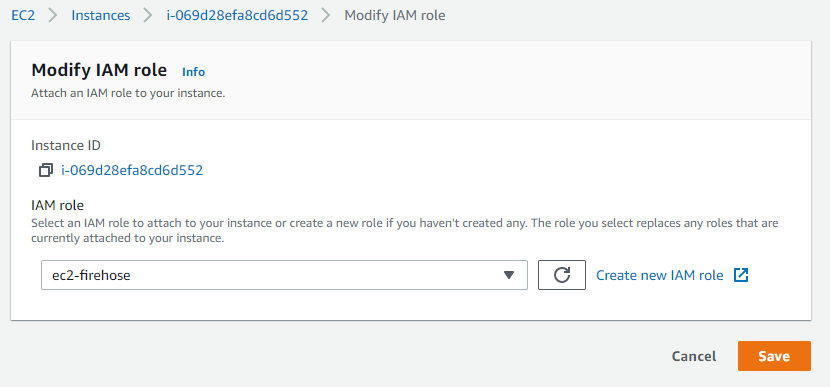
1. Click **Next: Permissions**
2. Select the **AdministratorAccess** permission. This gives your EC2 instance broad access to AWS services in your account; in a production setting, you might want to explicitly allow only the services needed by your instance, such as Firehose and CloudWatch.



1. Click on **Next: Tags** to proceed to the Tags screen, and then **Next: Review**.
2. Give your role a name, such as *ec2-firehose*.



1. Click the **Create role** button.
2. Back at the *Modify IAM role*window for your EC2 instance, click the refresh button, and select the IAM role you just created. Click**Save** when done.

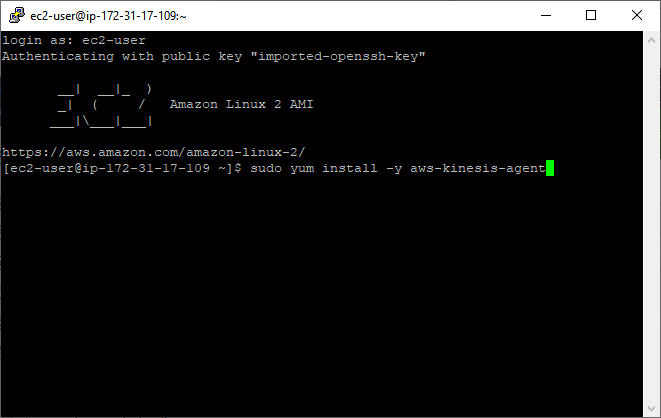


<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html>

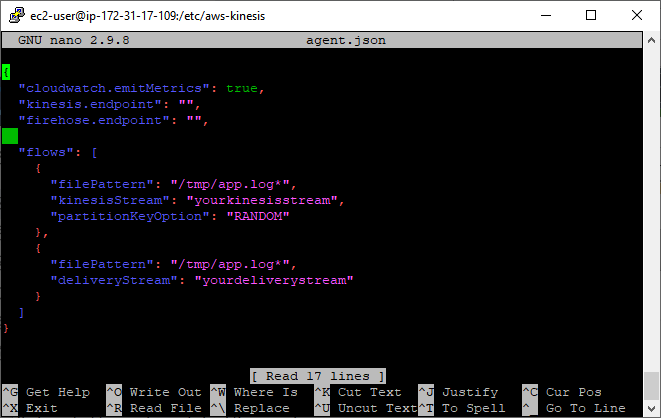
Install and configure the AWS Kinesis agent on our EC2 host

You'll need to install the AWS Kinesis agent on your EC2 test host, which will monitor the /var/log/messages file and feed it into our Firehose delivery stream. Once installed, edit its configuration file to specify the appropriate firehose endpoint for your stream's region, the log file we'll be streaming, and the delivery stream it's going to.

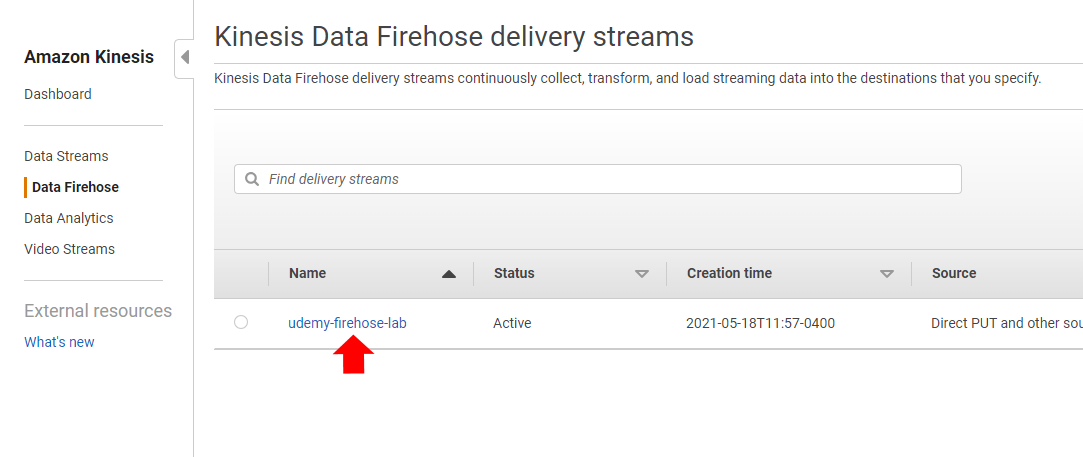
1. Log into your EC2 instance if you haven't already.
2. Install the Kinesis Agent with the command  
   sudo yum install -y aws-kinesis-agent



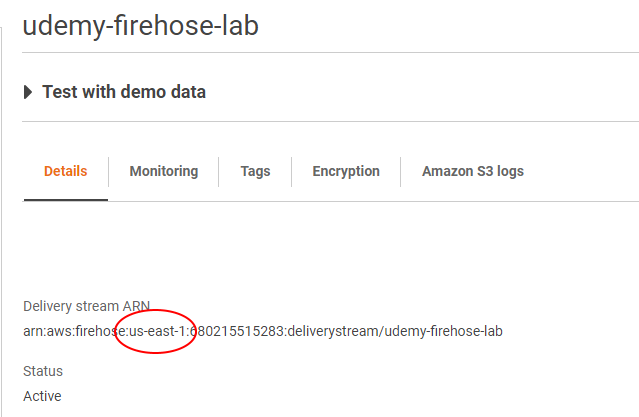
1. Edit the file /etc/aws-kinesis/agent.json:
   1. cd /etc/aws-kinesis
   2. sudo nano agent.json



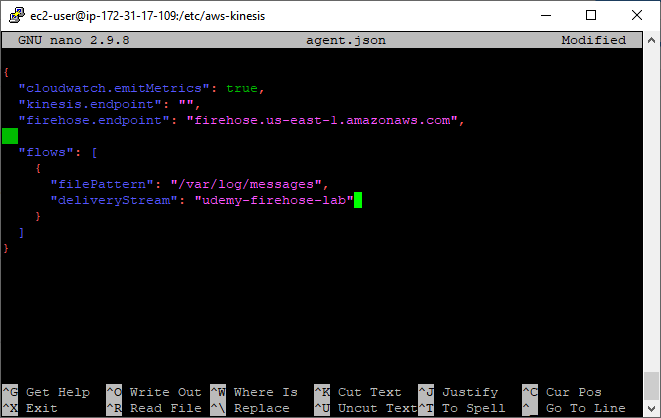
1. To determine the proper Firehose endpoint, we must determine what region our Firehose delivery stream was created in. From the Amazon Kinesis dashboard, select *Data Firehose*, and select the delivery stream you created earlier:



1. Note the region in your *Delivery stream ARN*; in this example, it is *us-east-1*. Also take note of your delivery stream's name (in this case, *udemy-firehose-lab*.)



1. Edit the agent.json file as shown below, substituting in your region within the *firehose.endpoint* setting, and your delivery stream name in *deliveryStream*. We have edited the *flows* block to only include a single Firehose flow, and set its *filePattern* to the path to the log we want to stream, which is */var/log/messages*.



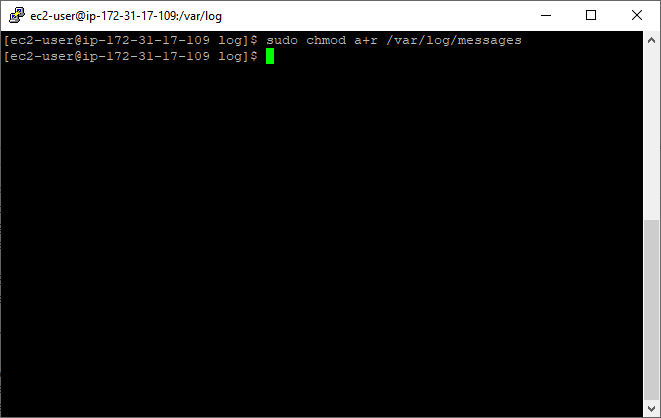
1. Save the file and quit the editor (in the nano editor, click control-O and enter, followed by control-X.)

<https://docs.aws.amazon.com/firehose/latest/dev/writing-with-agents.html>

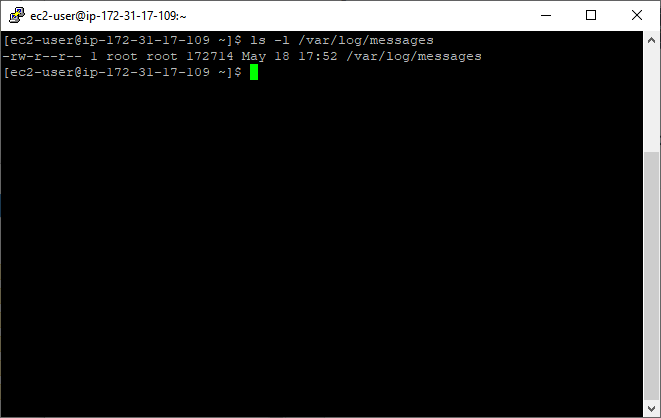
Ensure our message lo has the appropriate file permission

The default file permissions on */var/log/messages* won't allow the AWS Kinesis agent to read it, so you'll have to make that file readable by others.

1. From your terminal, enter:  
   sudo chmod a+r /var/log/messages  
   In a production setting, you would consult with your IT security team to ensure this is not a security risk. They may prefer to set up a user group that includes the *aws-kinesis-agent-user* and restrict access more tightly.



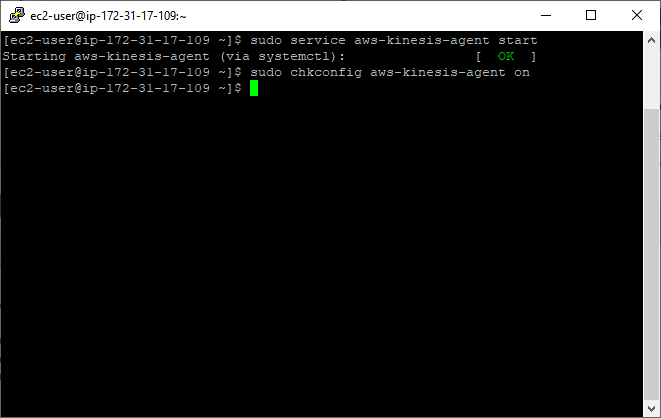
1. You may confirm the desired read permissions are set with  
   ls -l /var/log/messages



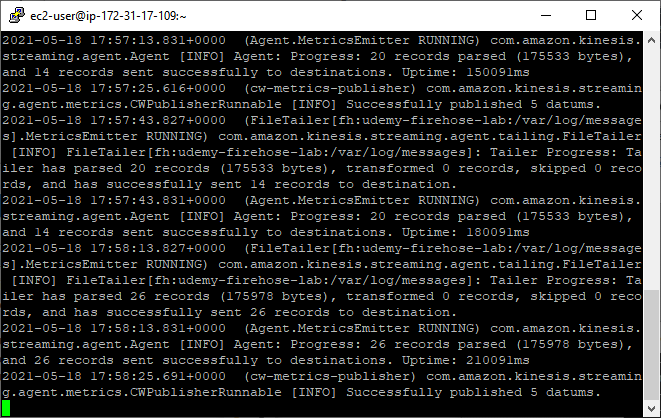
Run the Kinesis agent, and confirm log data is written into S3

Make sure the streaming of your data is working now. Start the *aws-kinesis-agent*service on your EC2 instance, and tail its log to confirm data starts getting sent to your Firehose delivery stream at least every 60 seconds. Check your S3 bucket and confirm log data is being written into S3 successfully.

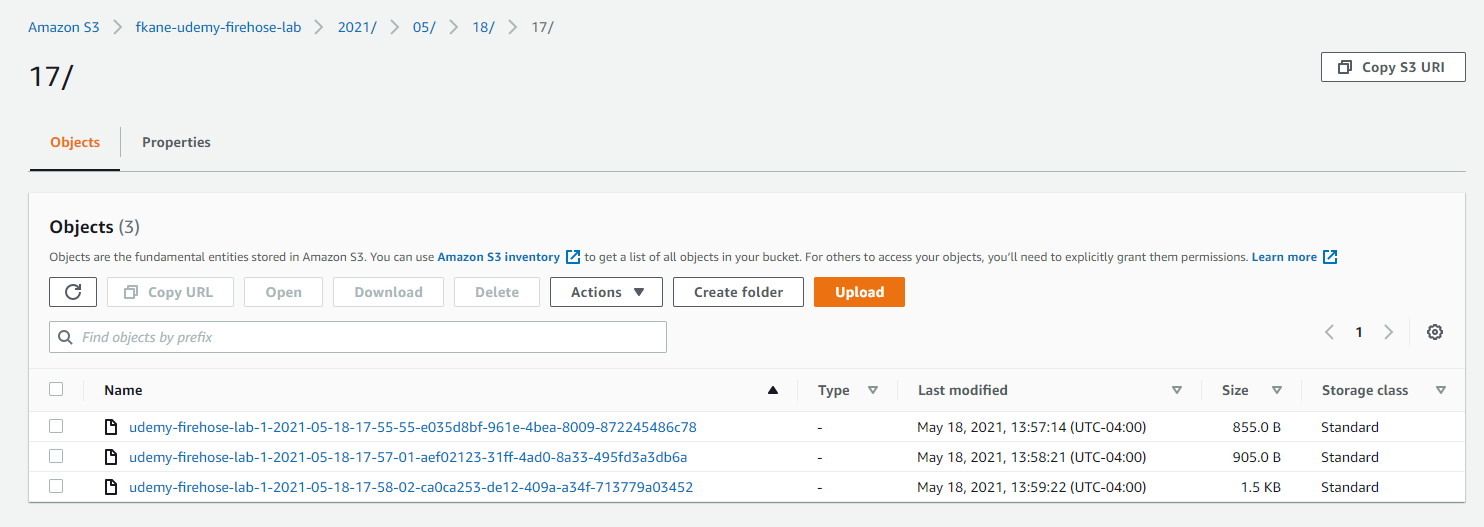
1. Enter the following commands to start the Kinesis agent:
   1. sudo service aws-kinesis-agent start
   2. sudo chkconfig aws-kinesis-agent on



1. Tail the agent's log with the command:  
   tail -f /var/log/aws-kinesis-agent/aws-kinesis-agent.log  
   Ensure there are no errors that require your attention. After a minute, you should start to see messages indicating that records were parsed and sent to their destinations. Press control-C when finished to return to the command prompt. Alternatively, logs could also be monitored using CloudWatch.



1. From the AWS Console or Services menu, select **S3.**
2. Select the bucket you created in Task 1.
3. Confirm that log data is present in the S3 bucket, partitioned automatically by date and hour. Upon navigating to the sub-folder for the current hour, your screen will look similar to this, although the dates and hour will be different:



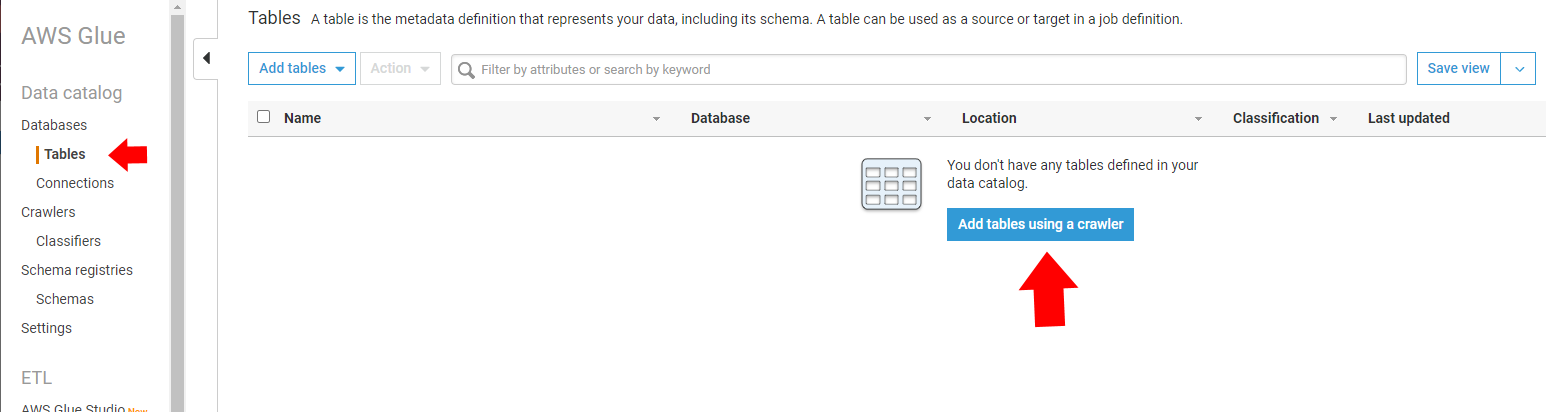
<https://docs.aws.amazon.com/firehose/latest/dev/writing-with-agents.html>

<https://docs.aws.amazon.com/firehose/latest/dev/monitoring.html>

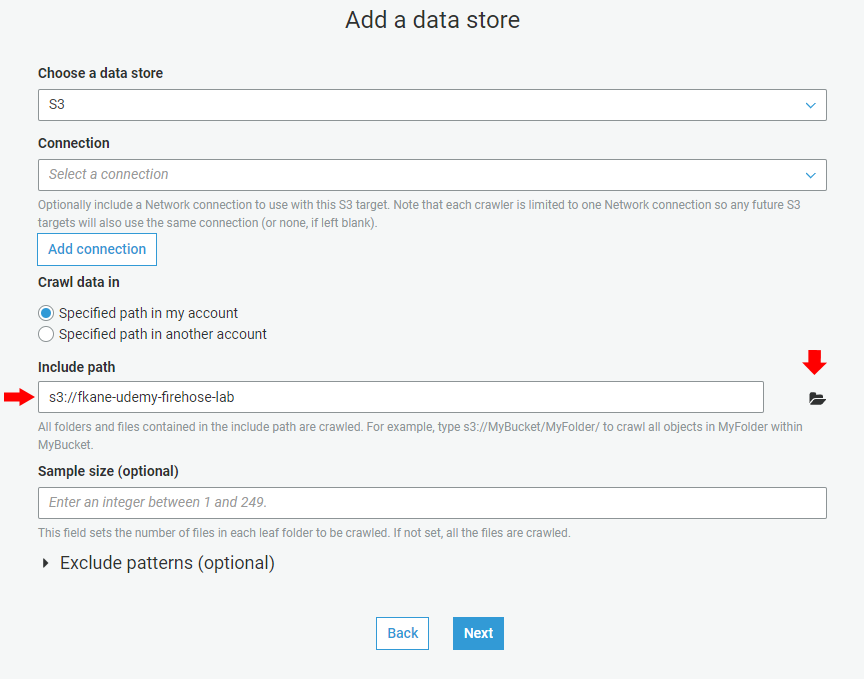
Create a table in the Glue Data Catalog for our log data using a Glue Crawlwe

Before we can query the log data you've streamed in, you'll need to create a virtual database table for it using an AWS Glue Crawler and the Glue Data Catalog. Set up a Glue Crawler for the S3 bucket containing our streamed logs, and run it once on demand. The data in the bucket is partitioned by date and hour, so you'll probably want to edit the resulting schema to give it meaningful names for those partitioned fields.

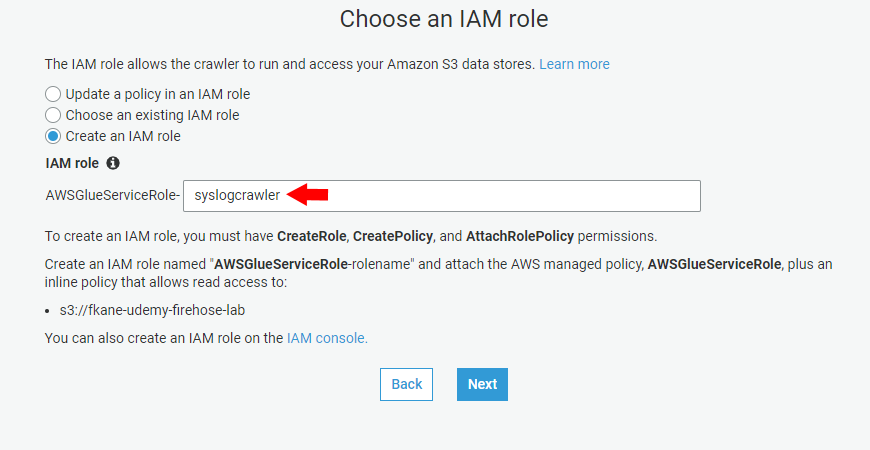
1. From the AWS Console or Services menu, select **AWS Glue**.
2. From the AWS Glue dashboard, select *Data catalog / Databases / Tables* and click the **Add tables using a crawler** button.



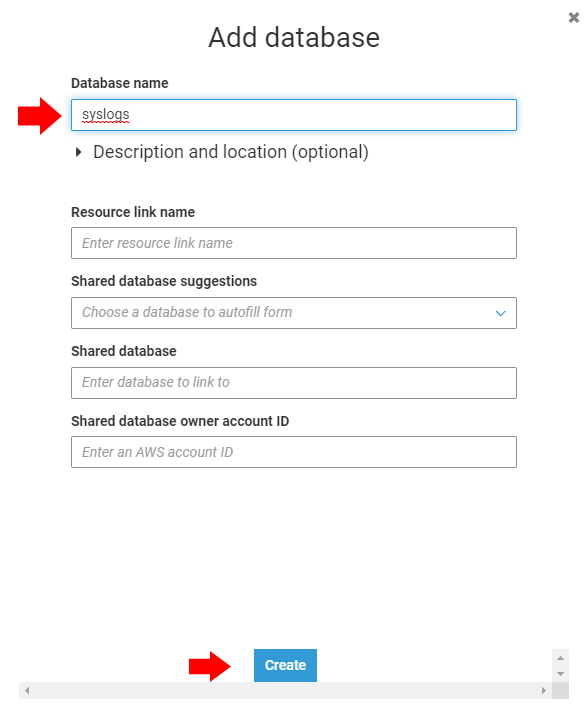
1. Give your crawler a name, such as *syslog-crawler*, and click **Next** to continue.
2. On the *Specify crawler source type* screen, leave the default settings for a *Data stores*source type and to *Crawl all folders*. Click **Next** to continue.
3. On the *Add a data store* screen, choose the path to the S3 bucket created in Task 1. Then click **Next** to continue.



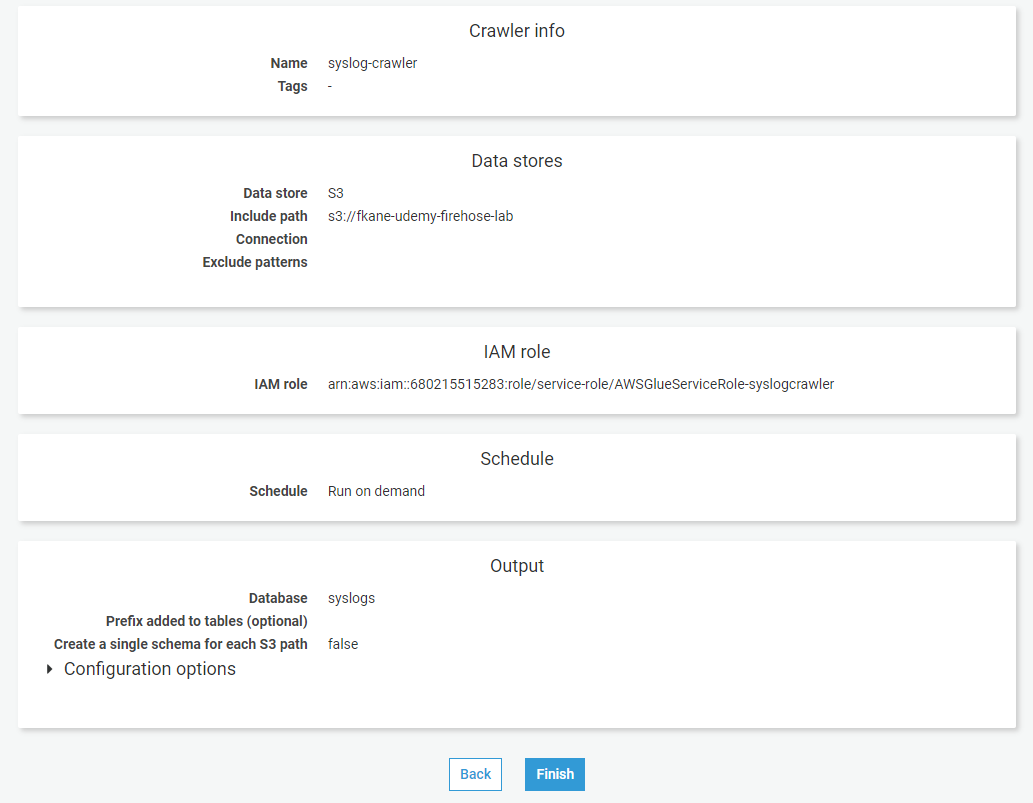
1. Select **No** on the *Add another data store* screen and click **Next**.
2. Select **Create an IAM role** on the *Choose an IAM role* screen, give the role a descriptive name such as *syslogcrawler* and click **Next**.



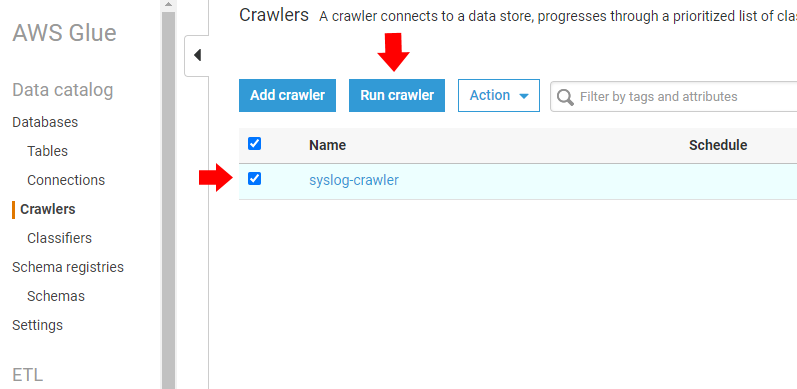
1. On the *Create a schedule for this crawler* set the *Frequency* to *Run on demand*, and click **Next**.
2. On the *Configure the crawler's output* screen, click the **Add database** button. Give the database a name, such as *syslogs*, and hit the **Create** button.



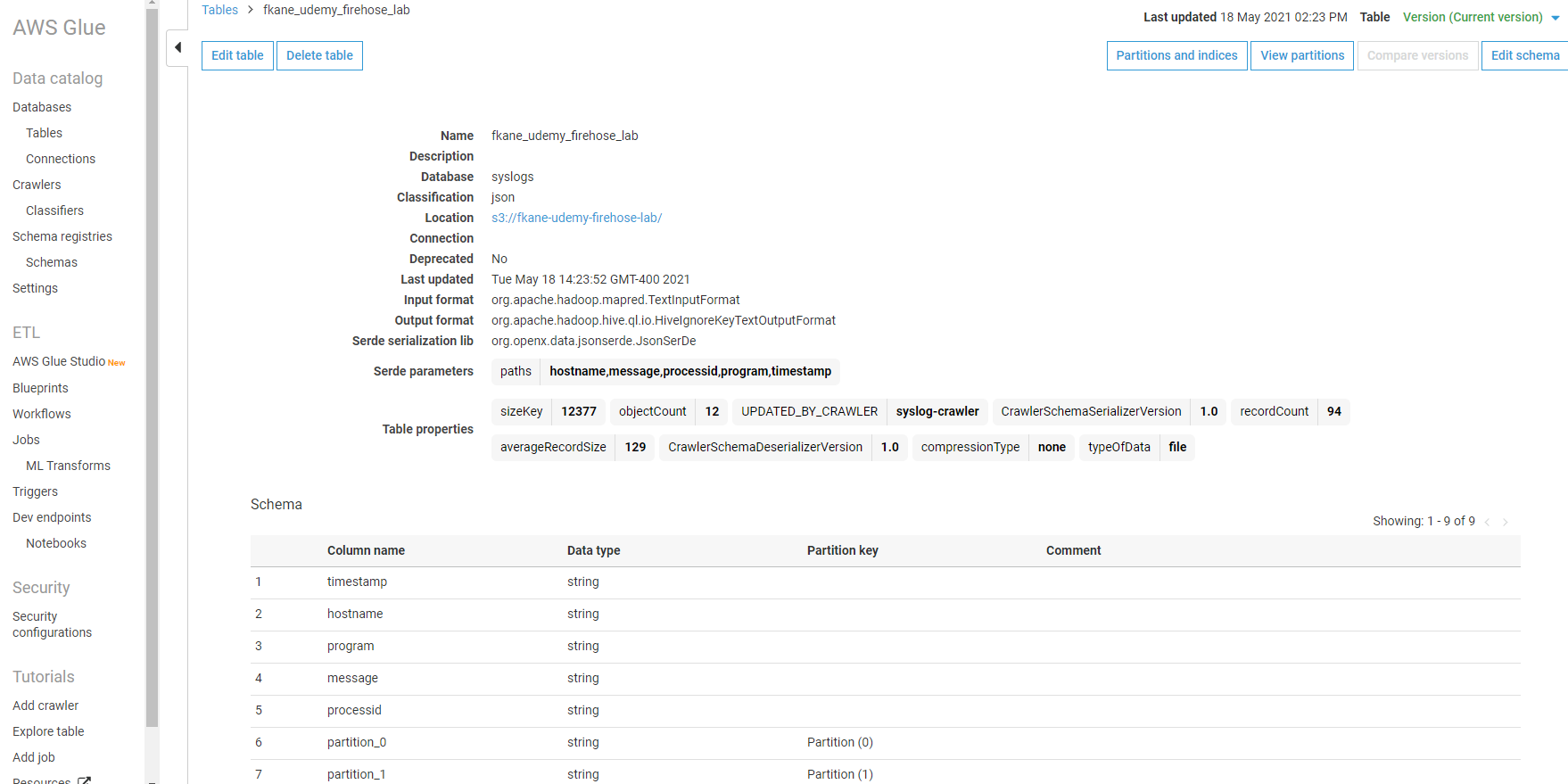
1. Click the **Next** button to proceed to the review screen.
2. Confirm the crawler configuration looks correct, and click **Finish**. Note your Data store include path will be unique and different from what is shown below.



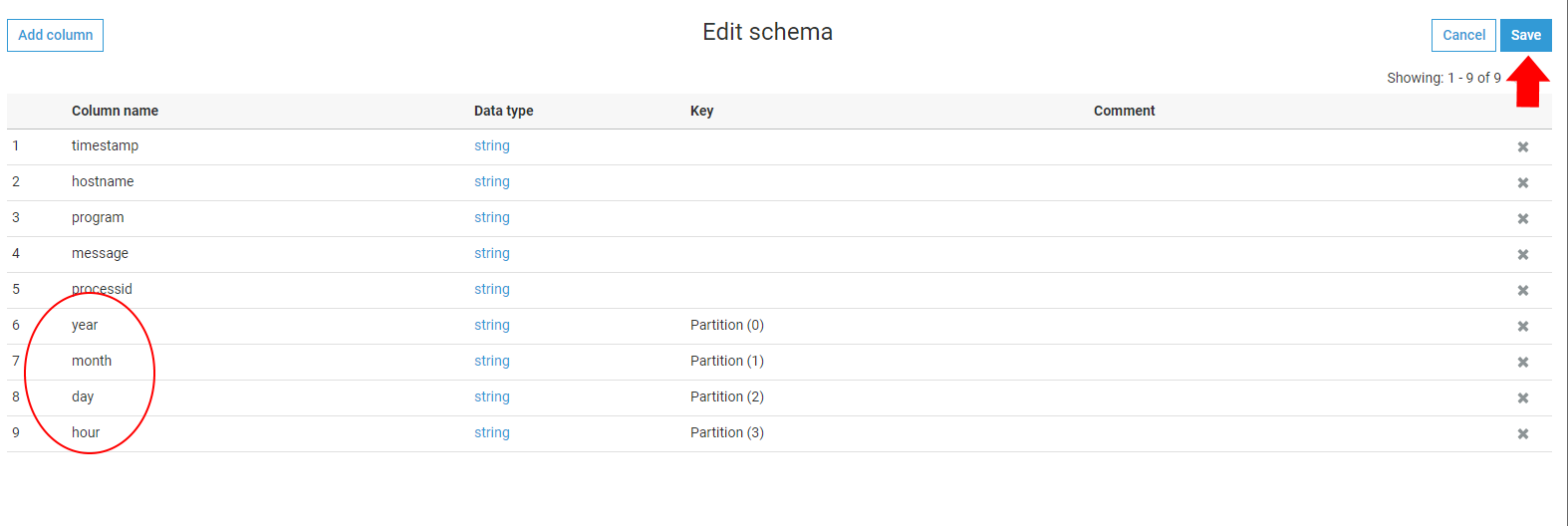
1. Select the crawler you just created and click the **Run crawler** button.



1. Wait for the status of your crawler to read *Ready*.
2. From the AWS Glue dashboard, select *Data catalog / Database / Tables* and select the table you just created. You should see a screen similar to this, substituting your own unique S3 bucket name.



1. Click the **Edit schema** button in the upper-right, and rename the partitions to year, month, day, and hour. Click **Save** when done.



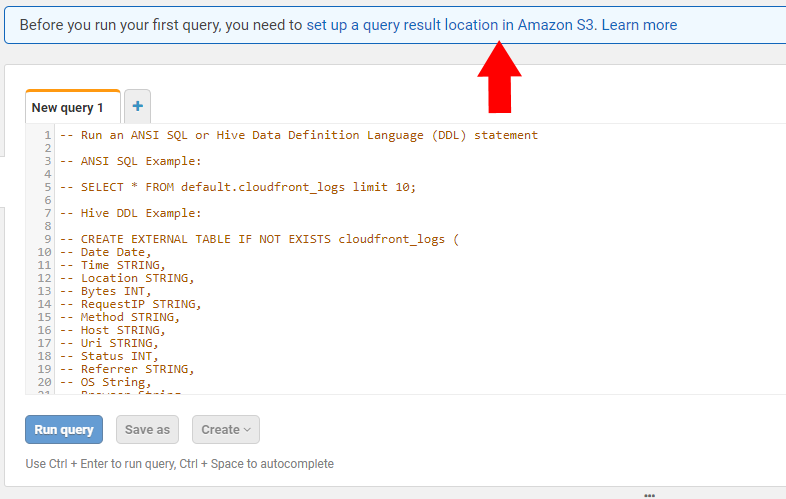
<https://docs.aws.amazon.com/glue/latest/dg/console-crawlers.html>

<https://docs.aws.amazon.com/glue/latest/dg/what-is-glue.html>

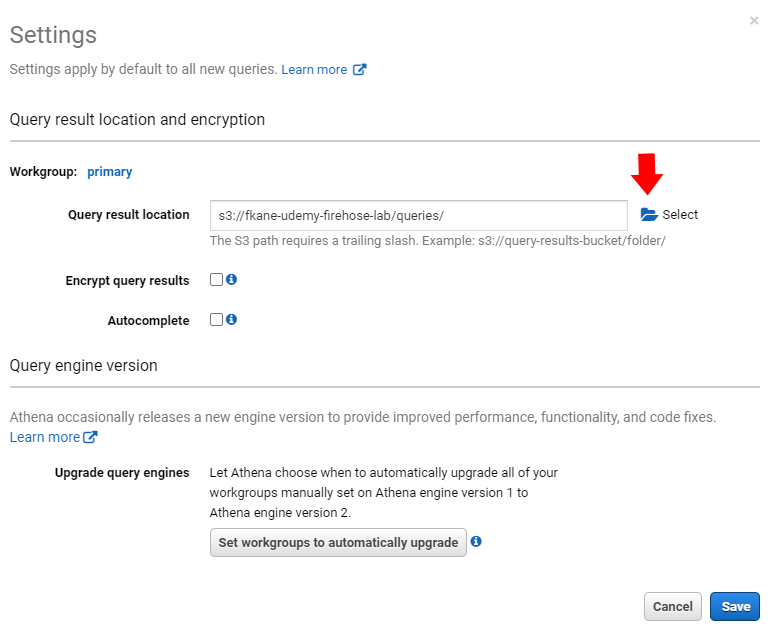
Use Athena to interactively query our log

Now that you have a schema for our streaming log data, see if you can query it successfully. Launch Amazon Athena and set up its query editor to query the syslogs database you just created. Test it out with a few simple SQL queries to confirm you can retrieve the log data that's streaming in.

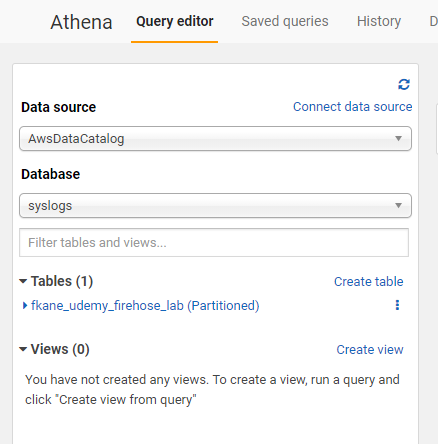
1. From the AWS Console or Services menu, select **Athena**. Navigate to the *Query editor* tab of the Athena dashboard; you may need to proceed through a "Get Started" button first.
2. Click on the prompt to *set up a query result location in Amazon S3*.



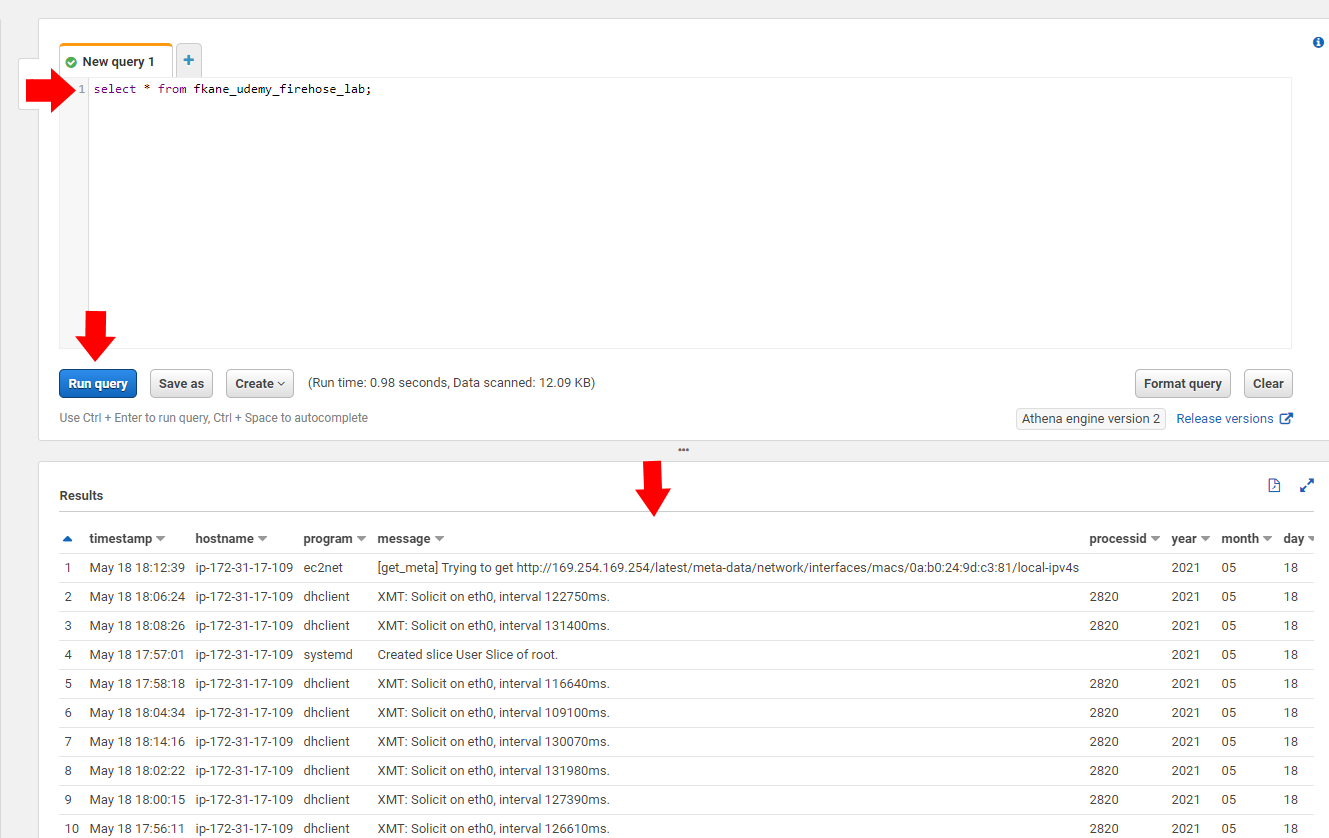
1. Navigate to the *queries* folder within your S3 bucket for the *Query result location* setting. If your data were sensitive, you might opt to also *Encrypt query results*. Click **Save** when done.



1. From the Athena Query editor screen, confirm the *Data source, Database, and Tables* are set to the database and table you just created with your Glue Crawler.



1. Confirm you are able to query the log data. Use the *New query* window and *Run query* button to execute a simple SQL command, such as  
   select \* from fkane\_udemy\_firehose\_lab;  
   (substitute in your own table name.) You should see query results in a format similar to the image below:



1. Experiment with different queries to narrow down the selection by program, hostname, or time. For example  
   select \* from fkane\_udemy\_firehose\_lab where program='systemd';  
   At this point we have an operational system!

<https://docs.aws.amazon.com/athena/latest/ug/querying-athena-tables.html>

<https://docs.aws.amazon.com/athena/latest/ug/ddl-sql-reference.html>

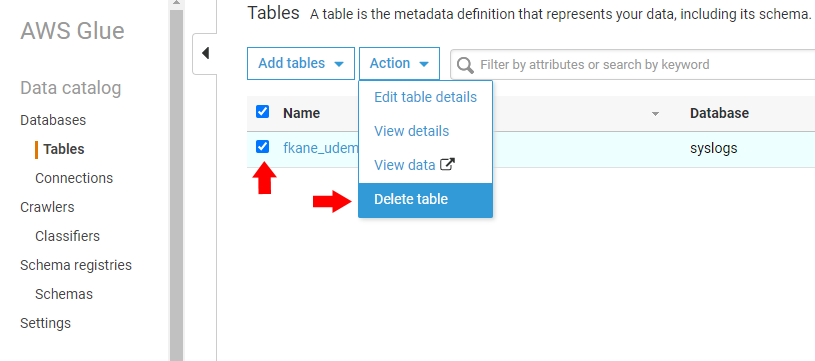
Clean up

Since we were only building a prototype for now, be sure to clean up the AWS resources you used now that you know it works! Delete any queries in the Athena query editor, our table in the Glue Data Catalog, the Glue crawler, our S3 bucket, and also the EC2 instance and Kinesis Firehose delivery stream we created.

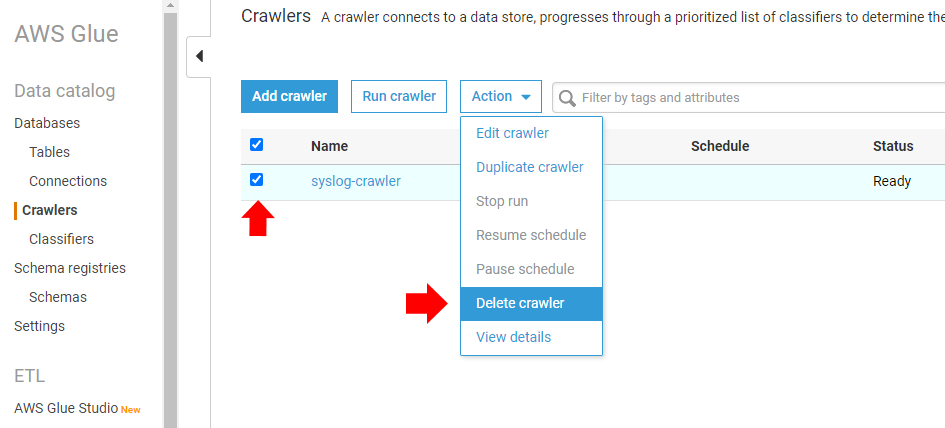
1. From the Athena query editor, close any extraneous "New query" windows, and use the **Clear** button to clean out the final one.



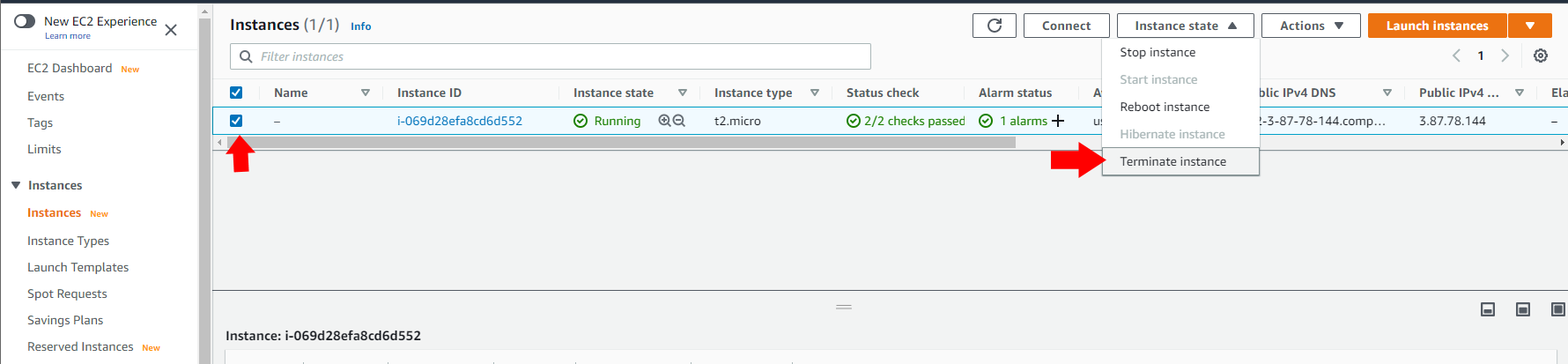
1. From the Services menu, select **AWS Glue**.
2. Select our Table, and from the **Action** button select **Delete table**. You will be prompted to confirm your choice.



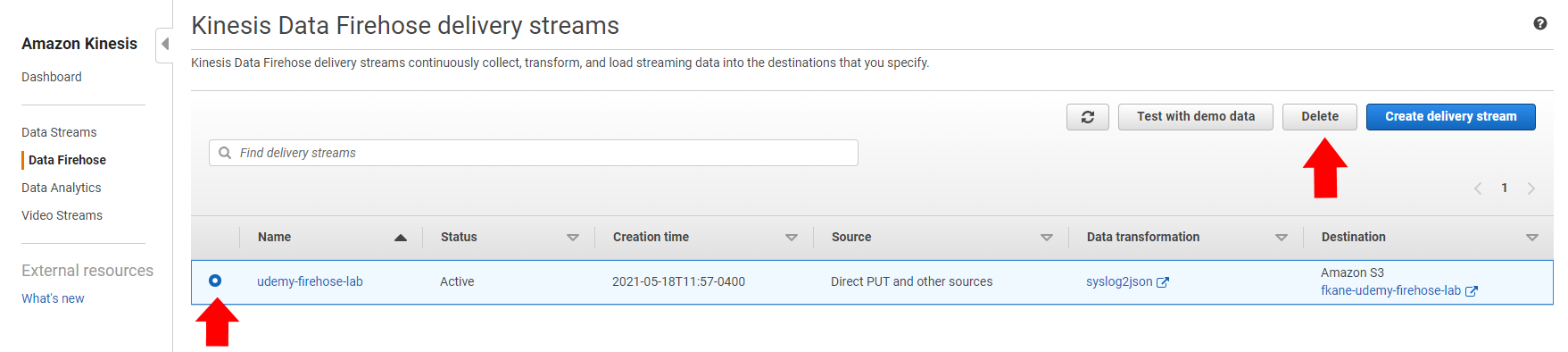
1. Select our Crawler, and from the **Action** button select **Delete crawler**. You will be prompted to confirm your choice.



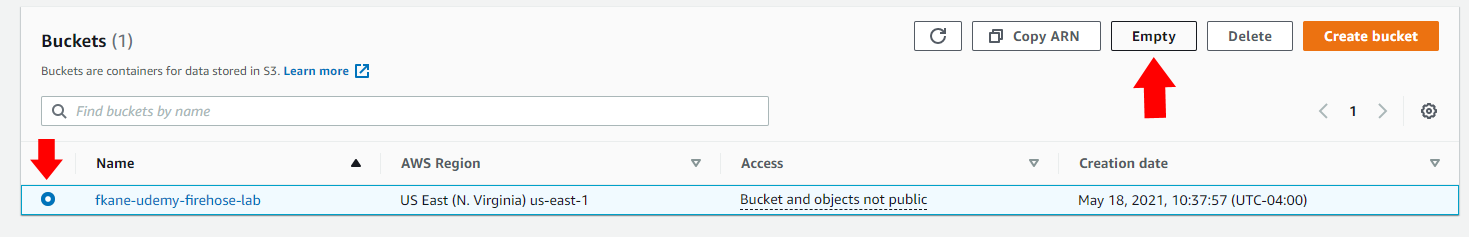
1. From the Services menu select **EC2**.
2. Navigate to the *Instances* view, select your instance, and from the **Instance state** button select **Terminate instance**.



1. From the Services menu select **Kinesis**.
2. From the Amazon Kinesis dashboard, select *Delivery streams*.
3. Select your stream and click the **Delete** button. You will be prompted to confirm your choice.



1. From the Services menu select **S3**.
2. Select your S3 bucket and click the **Empty** button. You will be prompted to confirm your choice.



1. Click the **Exit** button to return to the S3 Buckets view. With your bucket still selected, click **Delete**. You will be prompted to confirm your choice.

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/delete-bucket.html>

<https://aws.amazon.com/premiumsupport/knowledge-center/delete-terminate-ec2/>