**This Project will provide you with the necessary information to run below Amazon services.**

* Amazon Elasticsearch
* Amazon RedShift
* Amazon Kinesis

An introduction about Amazon Services for this project

Amazon Elasticsearch Service is a fully managed service that makes it easy for you to deploy, secure, and operate Elasticsearch at scale with zero down time. The service offers open-source Elasticsearch APIs, managed Kibana, and integrations with Logstash and other AWS Services, enabling you to securely ingest data from any source and search, analyze, and visualize it in real time. Amazon Elasticsearch Service lets you pay only for what you use – there are no upfront costs or usage requirements. With Amazon Elasticsearch Service, you get the ELK stack you need, without the operational overhead.

Amazon Redshift is a fast, scalable data warehouse that makes it simple and cost-effective to analyze all your data across your data warehouse and data lake. Redshift delivers ten times faster performance than other data warehouses by using machine learning, massively parallel query execution, and columnar storage on high-performance disk. You can setup and deploy a new data warehouse in minutes, and run queries across petabytes of data in your Redshift data warehouse, and exabytes of data in your data lake built on Amazon S3. You can start small for just $0.25 per hour and scale to $250 per terabyte per year, less than one-tenth the cost of other solutions.

Amazon Kinesis makes it easy to collect, process, and analyze real-time, streaming data so you can get timely insights and react quickly to new information. Amazon Kinesis offers key capabilities to cost-effectively process streaming data at any scale, along with the flexibility to choose the tools that best suit the requirements of your application. With Amazon Kinesis, you can ingest real-time data such as video, audio, application logs, website clickstreams, and IoT telemetry data for machine learning, analytics, and other applications. Amazon Kinesis enables you to process and analyze data as it arrives and respond instantly instead of having to wait until all your data is collected before the processing can begin.

**Overview**

The lab will give you a basic understanding of Amazon Elasticsearch Service, including: creating clusters, cluster node configurations, storage configurations and Identity & Access Management Policies (IAM).

**Topics covered**

* By the end of this lab you will be able to:
* Deploy an Amazon Elasticsearch Service domain
* Create an AWS CloudWatch Log Group
* Subscribe an Amazon CloudWatch Log Group to Amazon Elasticsearch Service
* Monitor Amazon Elasticsearch Service cluster metrics

**Prerequisites**

You should have completed the *Introduction to Amazon EC2* and *Introduction to AWS Identity and Access Management (!AM) labs.* Previous knowledge of Kibana and Elasticsearch is desirable.

**Introducing the Technologies**

**Amazon Elasticsearch Service**

Amazon Elasticsearch Service (Amazon ES) is a managed service that makes it easy to deploy, operate, and scale Elasticsearch in the AWS cloud. Elasticsearch is a **popular open-source search and analytics engine** for use cases such as log analytics, real-time application monitoring, click stream analytics, and text search.

You can set up and configure your Amazon Elasticsearch Service Cluster in minutes from the AWS Management Console.

Amazon ES provisions all the resources for your cluster and launches it. Amazon ES automatically detects and replaces failed nodes, reducing the overhead associated with self-managed infrastructures. There are no upfront costs to set up Amazon ES clusters, and you pay only for the service resources that you use.

Amazon ES offers the following benefits of a managed service:

* Simple cluster scaling via API
* Self-healing clusters
* High availability on-demand
* Automatic cluster snapshots for data durability
* Security
* Cluster monitoring

Amazon ES includes built-in integration with Kibana and supports integration with Logstash. **Logstash** is an open-source data pipeline that helps you process logs and other event data and load them into Elasticsearch. **Kibana** is an open-source analytics and visualization platform that helps you get a better understanding of your data in Elasticsearch.

**Components of Amazon Elasticsearch Service**

Amazon ES contains the following components:

**Domain:** An Amazon ES domain comprises an Elasticsearch Service Cluster -hardware and software - along with additional hardware and software providing load-balancing, security, monitoring, and more. The domain is exposed by service endpoints for Amazon ES, with a name that must meet the following criteria:

Uniquely identifies a domain within an AWS account

* Starts with a letter or number
* Contains at least three characters, but not more than 28 characters
* Contains only lowercase characters a-z, the numbers 0-9, and the hyphen (-)

**Cluster:** A collection of one or more data **nodes**, optional dedicated master nodes, and storage required to run Elasticsearch Service.

**Node:** A single instance within an Elasticsearch Service cluster that has the ability to recognize and process or forward messages to other nodes.

**Storage**: Amazon ES supports two distinct storage types, Instance Storage or Elastic Block Store (EBS) that can be general purpose (SSD), provisioned lOPS (SSD) or magnetic.

**Task 1: Deploy an Amazon Elasticsearch Service Domain**

In this task, you will deploy an Amazon Elasticsearch Service Domain.

* In the **AWS Management Console**, on the Services menu, click **Elasticsearch Service.**
* Click Create a new domain then configure:
* **Elasticsearch domain name:***mydomain*
* **Elasticsearch version** 6.3
* Click **Next**

An Elasticsearch cluster can contain multiple **nodes** to provide additional storage. You can also configure the **Instance Type** for each node, which offers different compute, memory, and storage capabilities.

* On the **Configure cluster** page, under **Node configuration,** for **Instance type** select *t2.smallelasticsearch.*

A dedicated master node (which you should *not* select) is a cluster node that performs cluster management tasks but does not hold data or respond to data upload requests. This offloading of cluster management tasks increases the stability of your Amazon Elasticsearch Service clusters. It is recommended that you avoid allocating dedicated master nodes for all small and short-lived Amazon Elasticsearch Service domains.

Zone awareness (which you should *not* select) distributes replicas across multiple Availability Zones, which will increase the availability of your cluster.

* In the Storage configuration section, ensure the following are set:
  + **Storage type**: EBS
  + **EBS volume type**: General Purpose (SW)
  + **EBS volume size**: 10

**Amazon EBS** provides durable, block-level storage volumes.

**General Purpose (SSD)** storage is suitable for a wide variety of workloads that have moderate I/O requirements. The baseline of 3 10PS per GB and the ability to burst up to 3,000 lOPS will provides predictable performance well-suited to many applications.

* Scroll to the bottom of the screen, then click Next
* In the **Network configuration** section, select **Public access**

Amazon Elasticsearch Service offers several ways of controlling access to your domains. This section covers the various policy types, how they interact with each other, and how to create your own, custom policies.

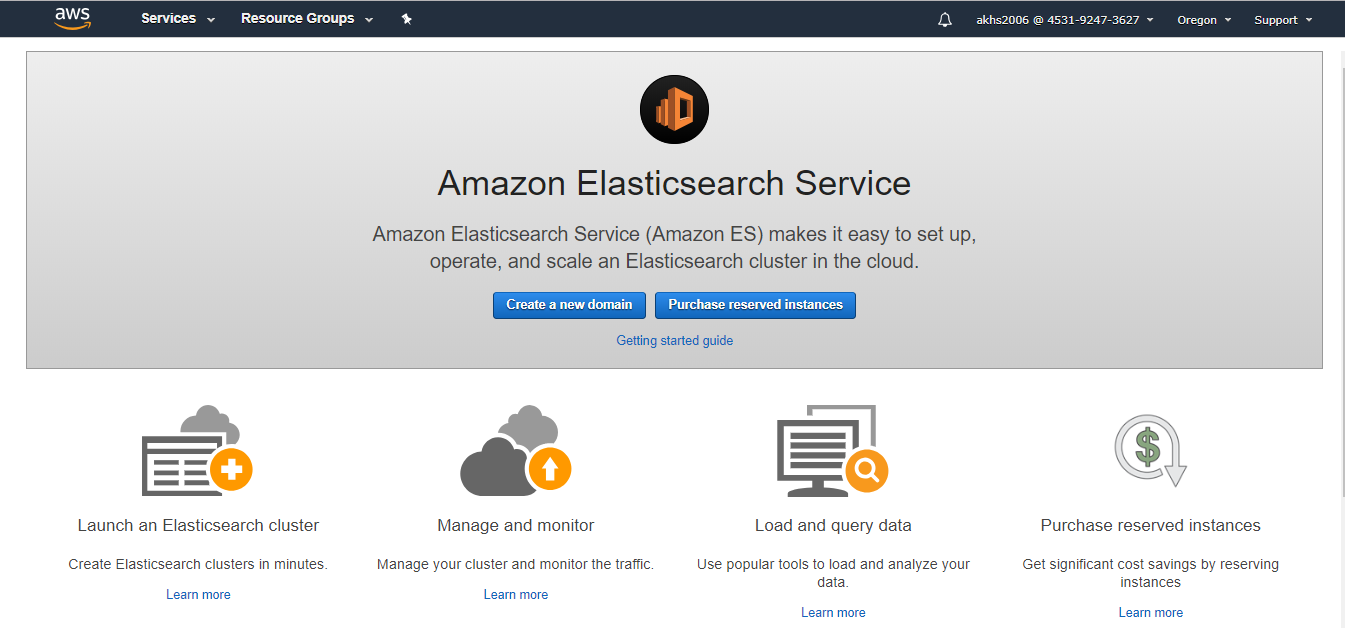
* **Resource-based Policies:** You attach resource-based policies to domains. These policies specify which actions a principal can perform on the domain's sub resources. Subresources include Elasticsearch indices and APIs.
* **Identity-based policies:** Unlike resource-based policies, which you attach to domains in Amazon ES, you attach identity-based policies to users or roles using the AWS Identity and Access Management (IAM) service. Just like resource-based policies, identity-based policies specify who can access a service, which actions they can perform, and if applicable, the resources on which they can perform those actions.
* **IP-based Policies:** IP-based policies restrict access to a domain to one or more IP addresses or CIDR blocks. Technically, IP-based policies are not a distinct type of policy. Instead, they are just resource-based policies that specify an anonymous principal and include a special Condition element.
* In the Access policy section, configure:
  + - **Set the domain access policy to**: *Allow open access to the domain*
    - Select **I accept the Risk**
    - **Click** OK

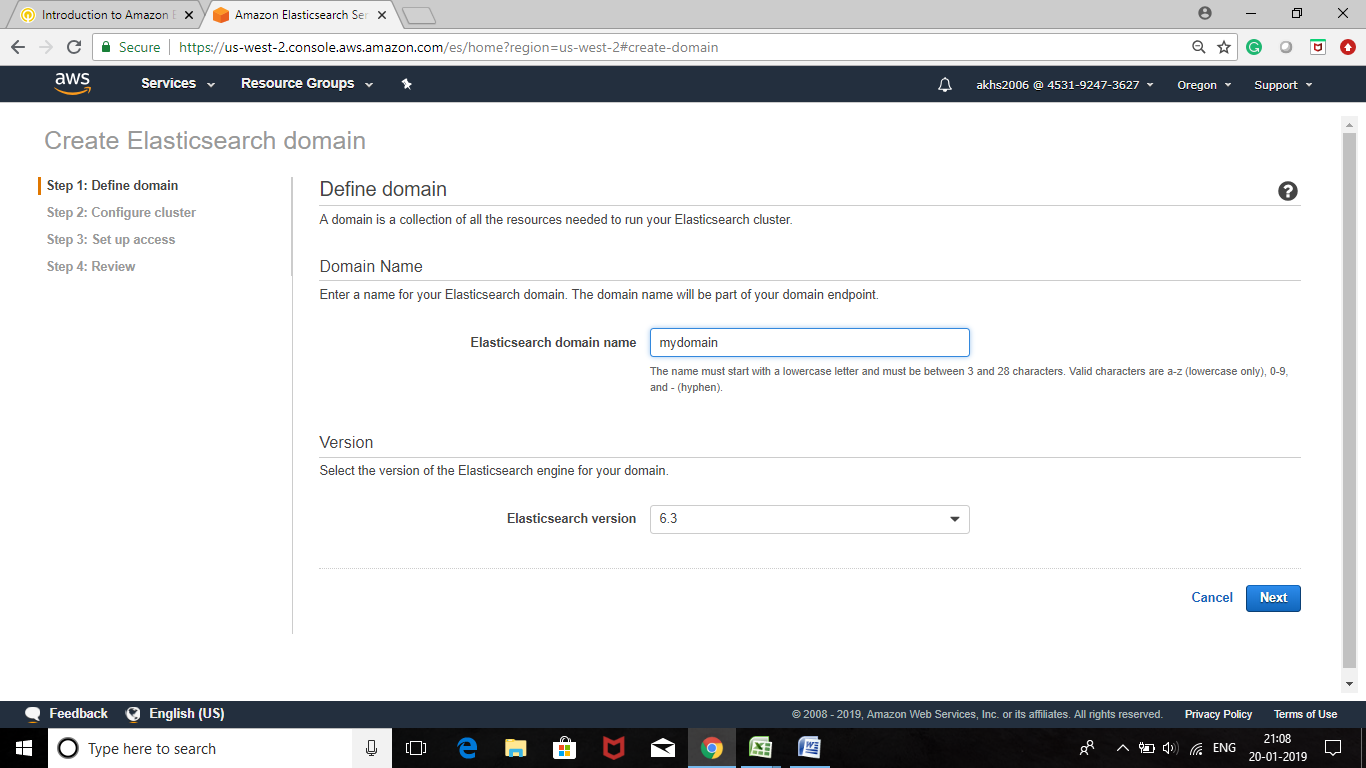
Examine the policy. It is **allowing** all Elasticsearch Service actions ( es :\* ) to anybody

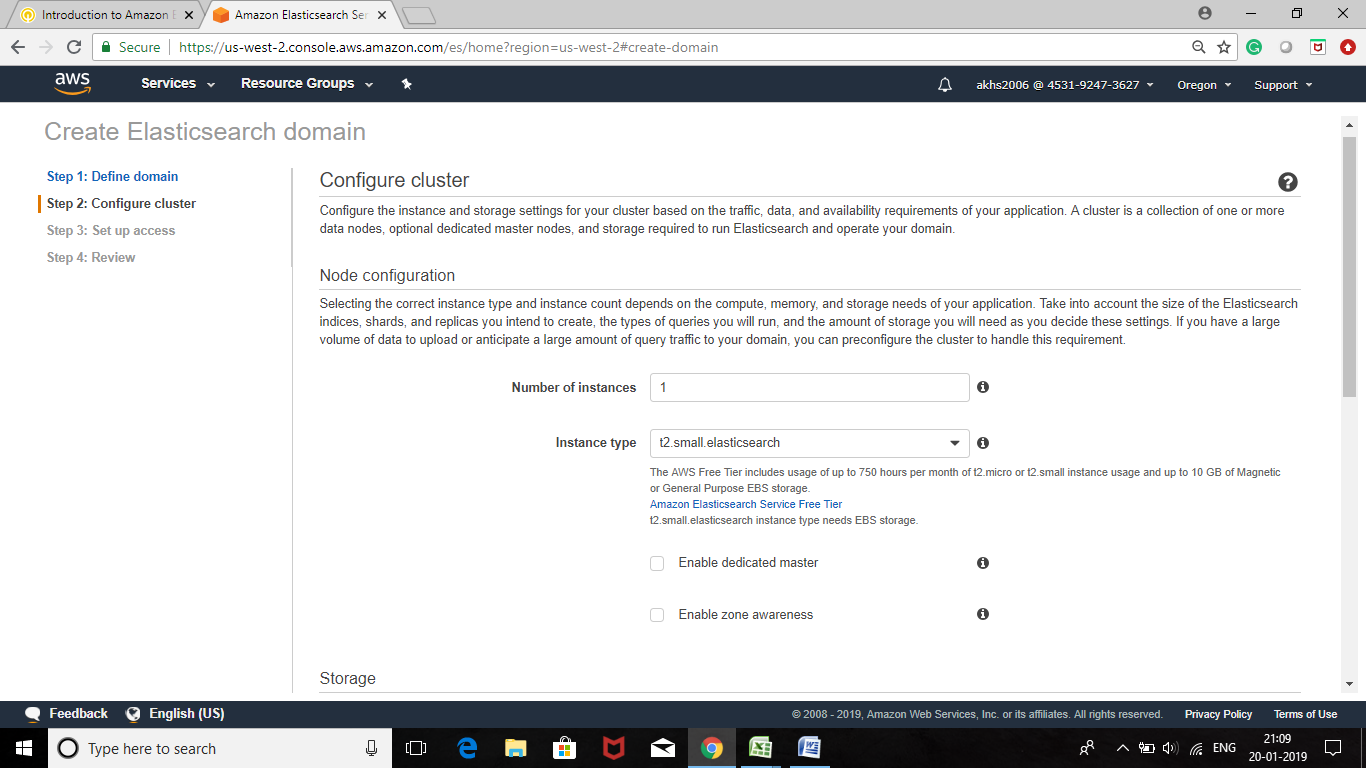
(Principal = \* ) for your *mydomain* domain.

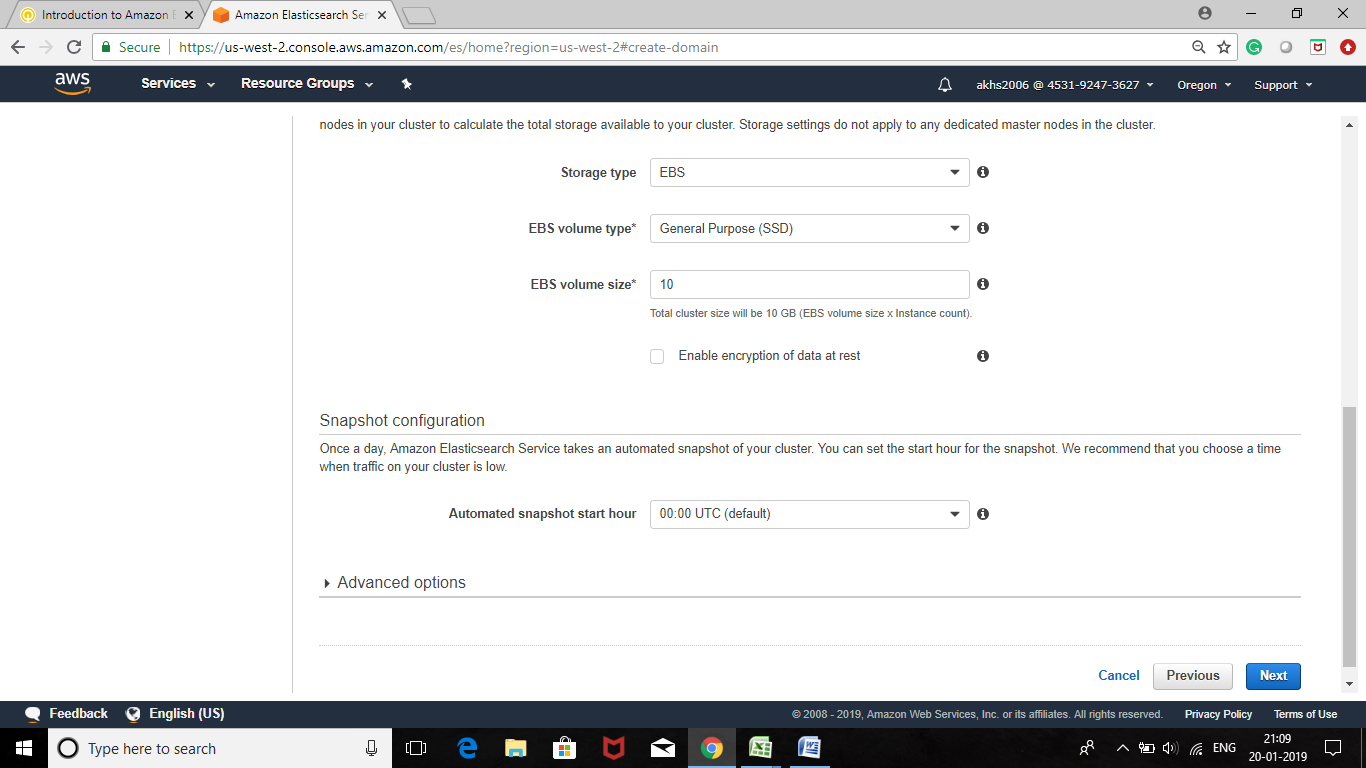
* Click Next
* Review the system configuration, then click Confirm

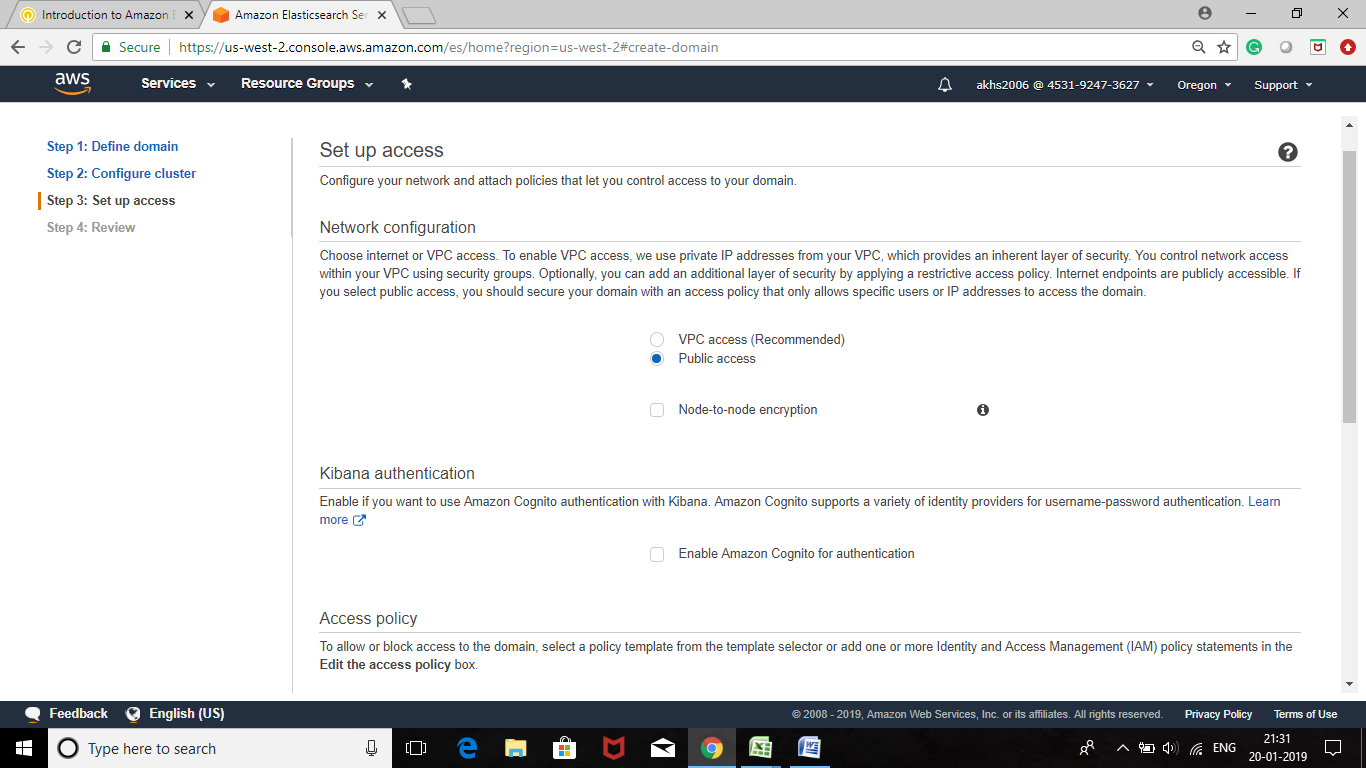
The service will take approximately 10 minutes to deploy. While waiting for the service to deploy, please **continue with the steps below**. There is no need to wait for the cluster to be ready.

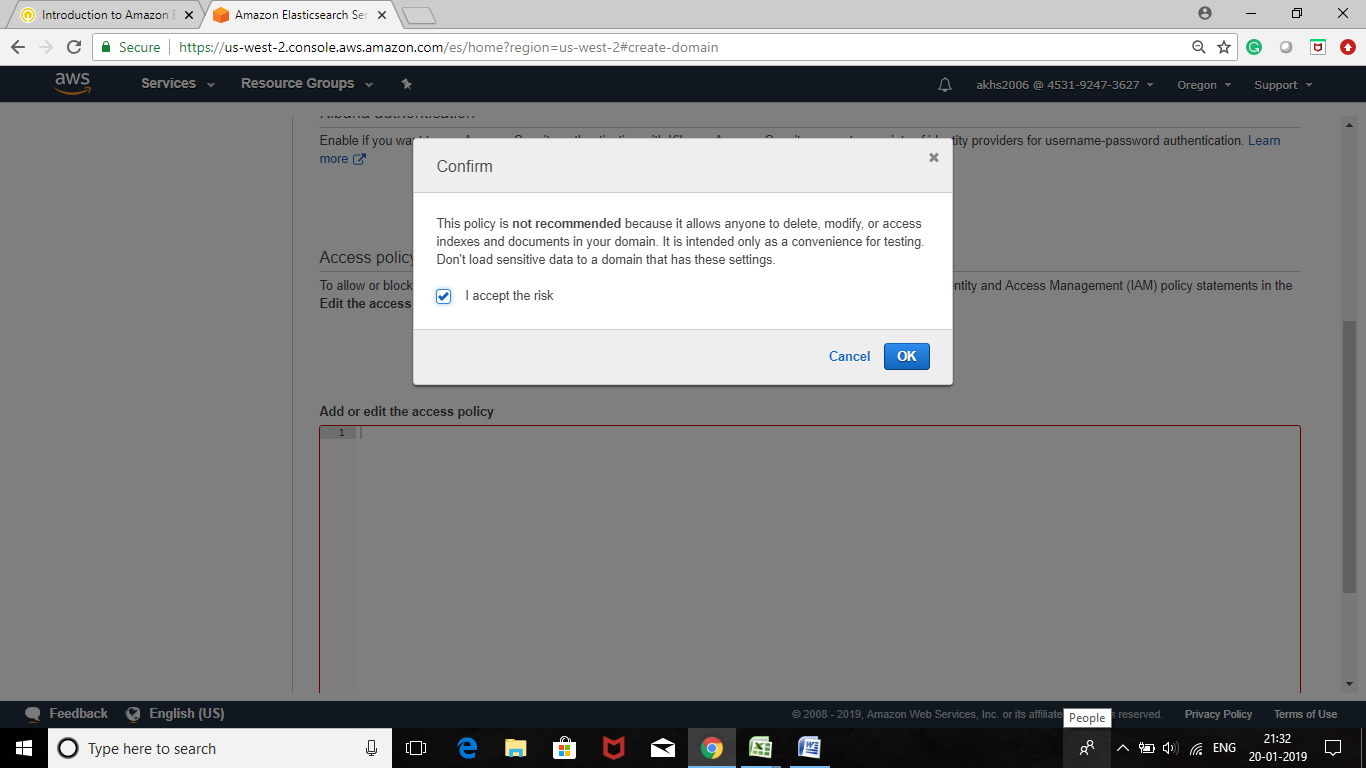


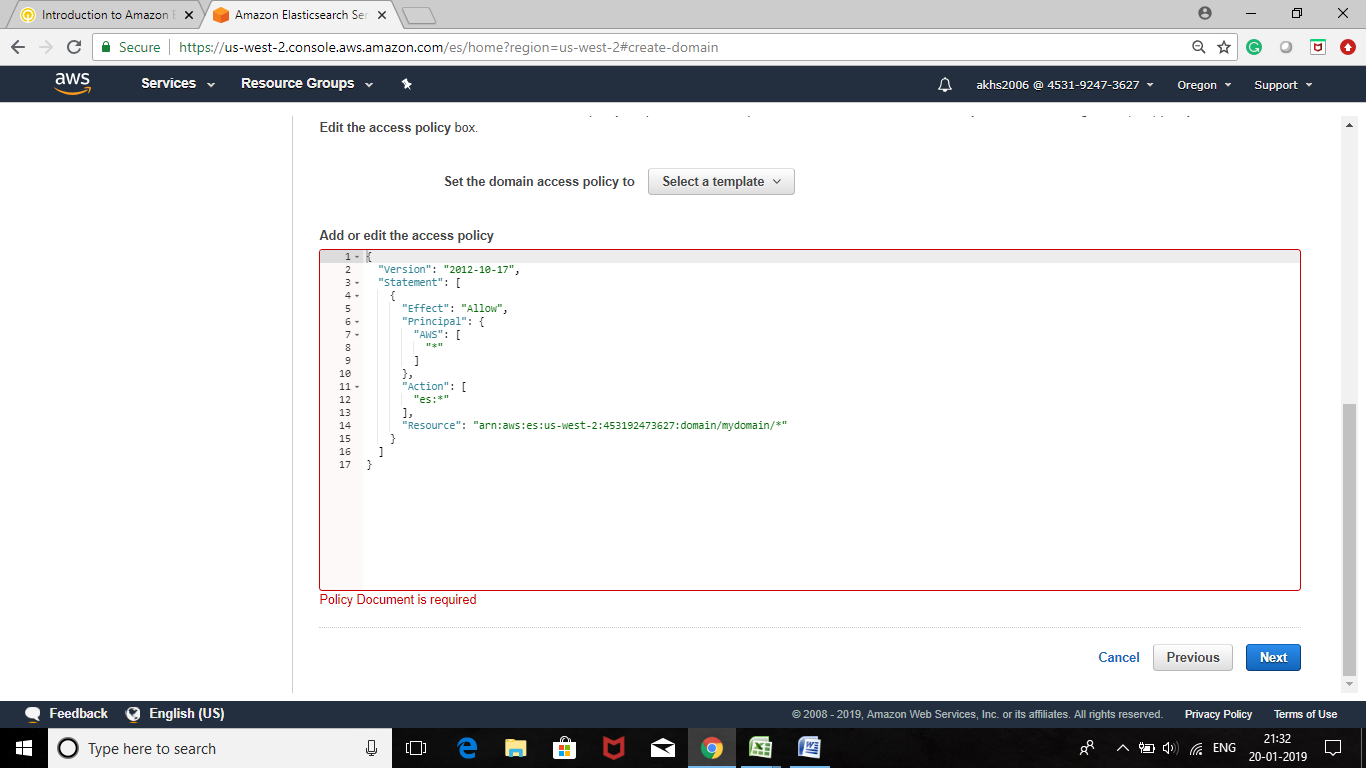


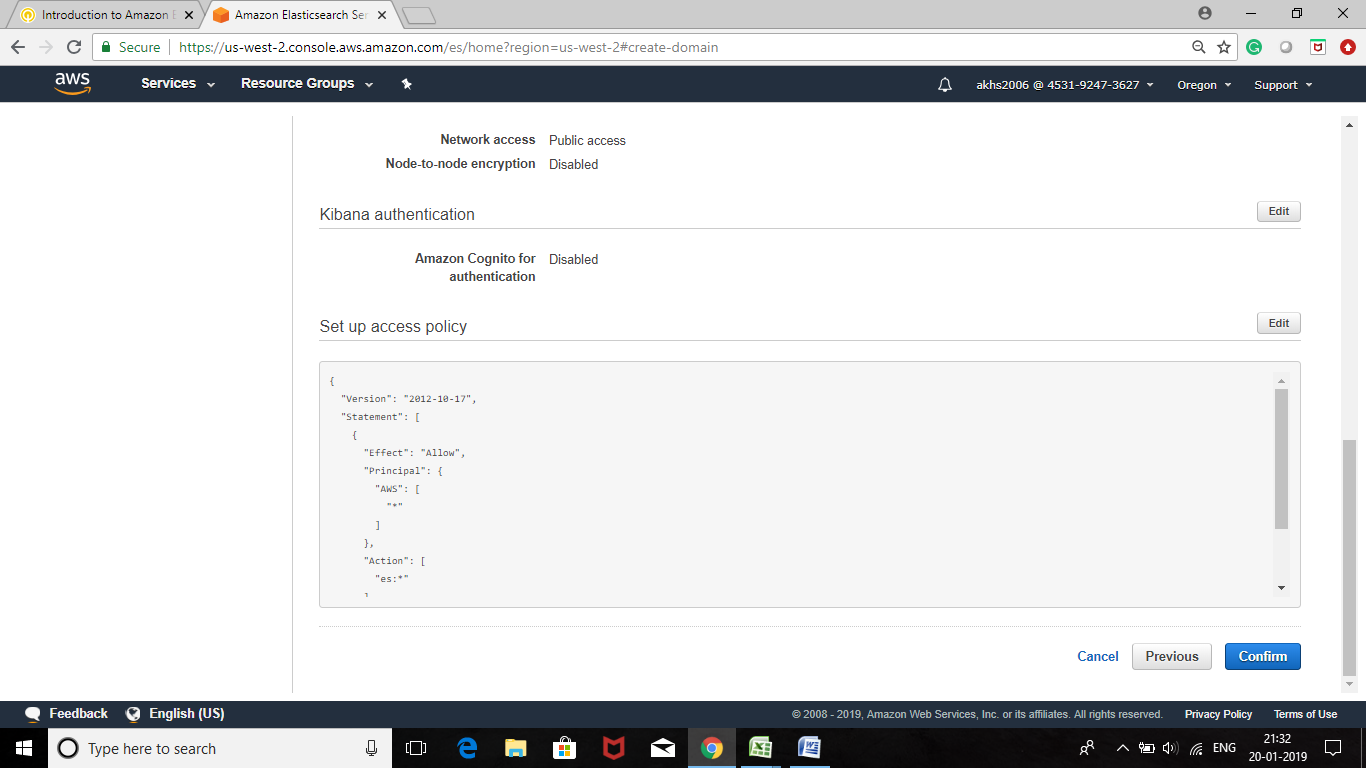












**Task 2: Generate Data from AWS CloudTrail**

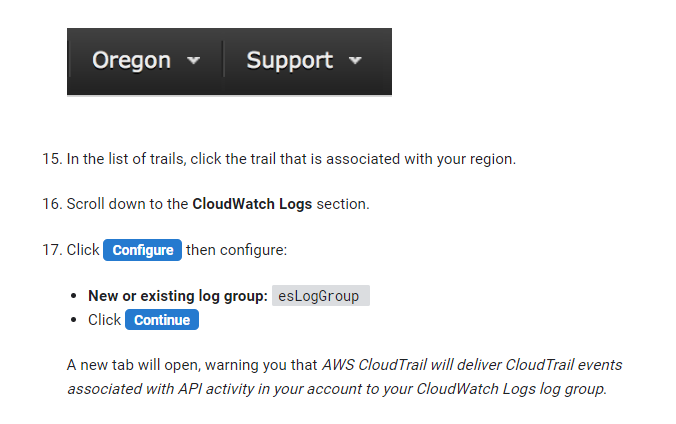
In this task, you will create an **AWS CloudTrail Log Group,** which captures information about usage of AWS services within an account. You will subscribe this Log Group to the Elasticsearch cluster to generate data for analysis.

With **AWS CloudTrail,** you can monitor your AWS deployments in the cloud by getting a history of AWS API calls for your account, including API calls made via the AWS Management Console, the AWS SDKs, the command line tools, and higher-level AWS services. You can also identify which users and accounts called AWS APIs for services that support CloudTrail, the source IP address the calls were made from, and when the calls occurred.

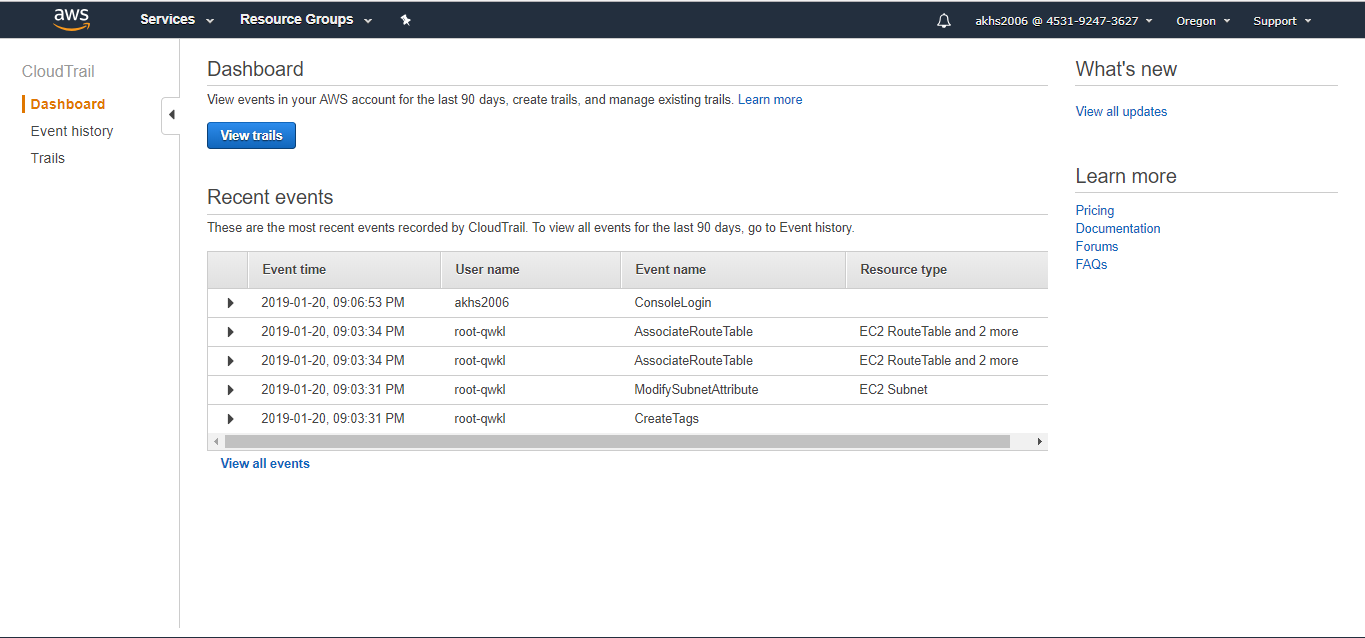
* On theServicesmenu, click **CloudTrail.**
* In the left navigation pane, click **Trails.**

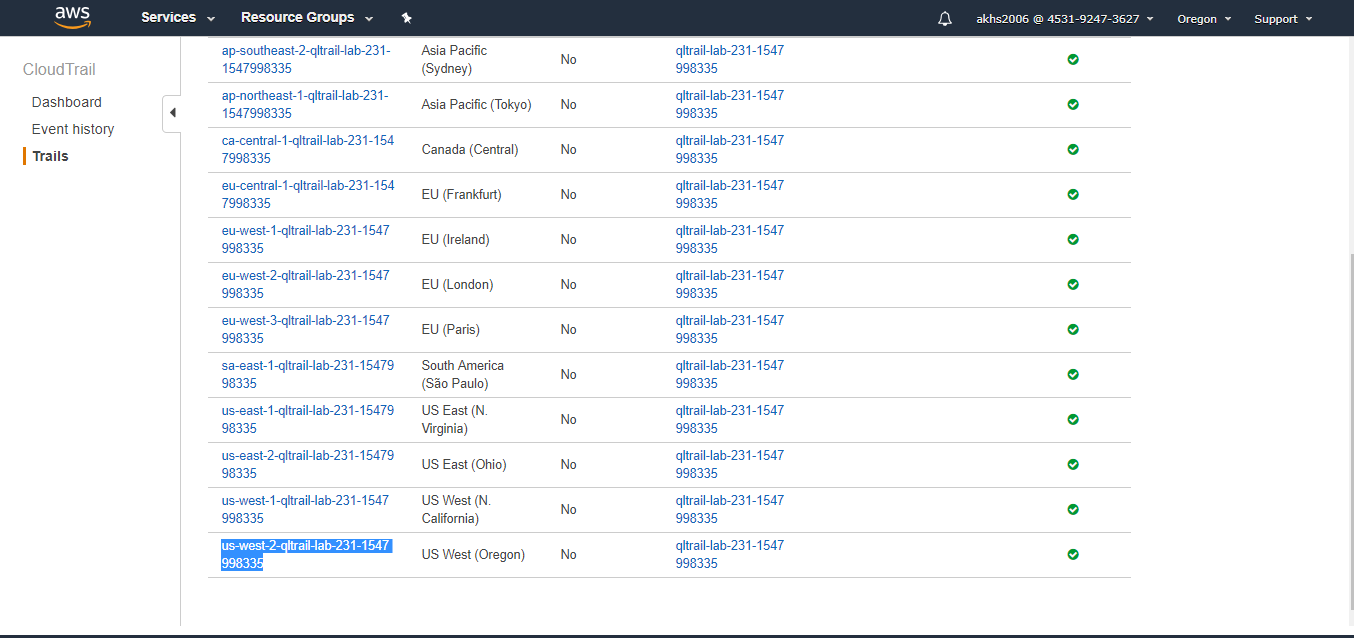
A number of trails will appear, one for each AWS region.

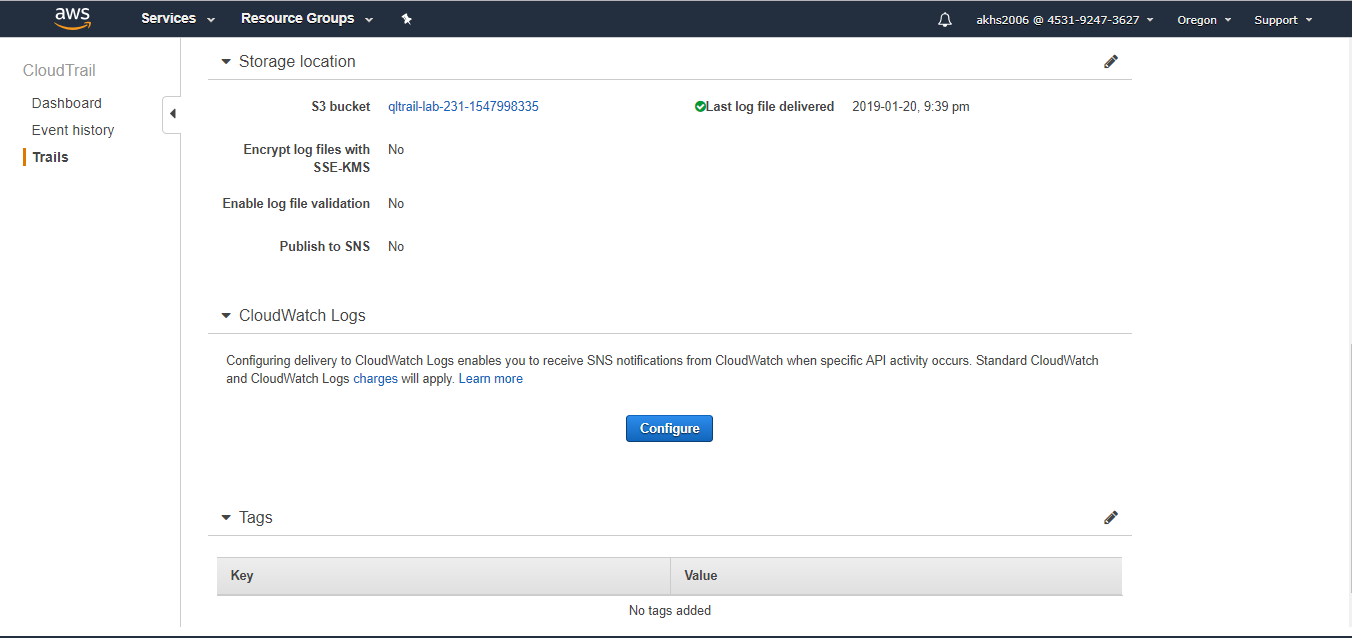
* At the top-right of your screen, locate your region. It will look similar to:

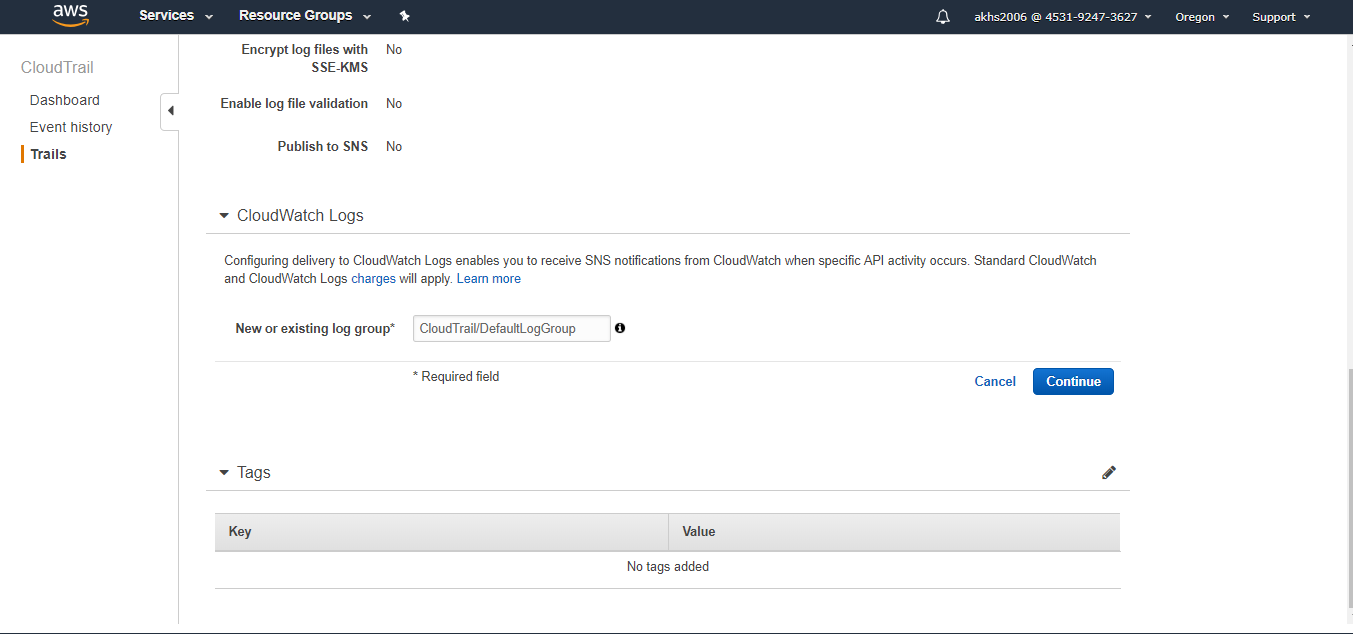


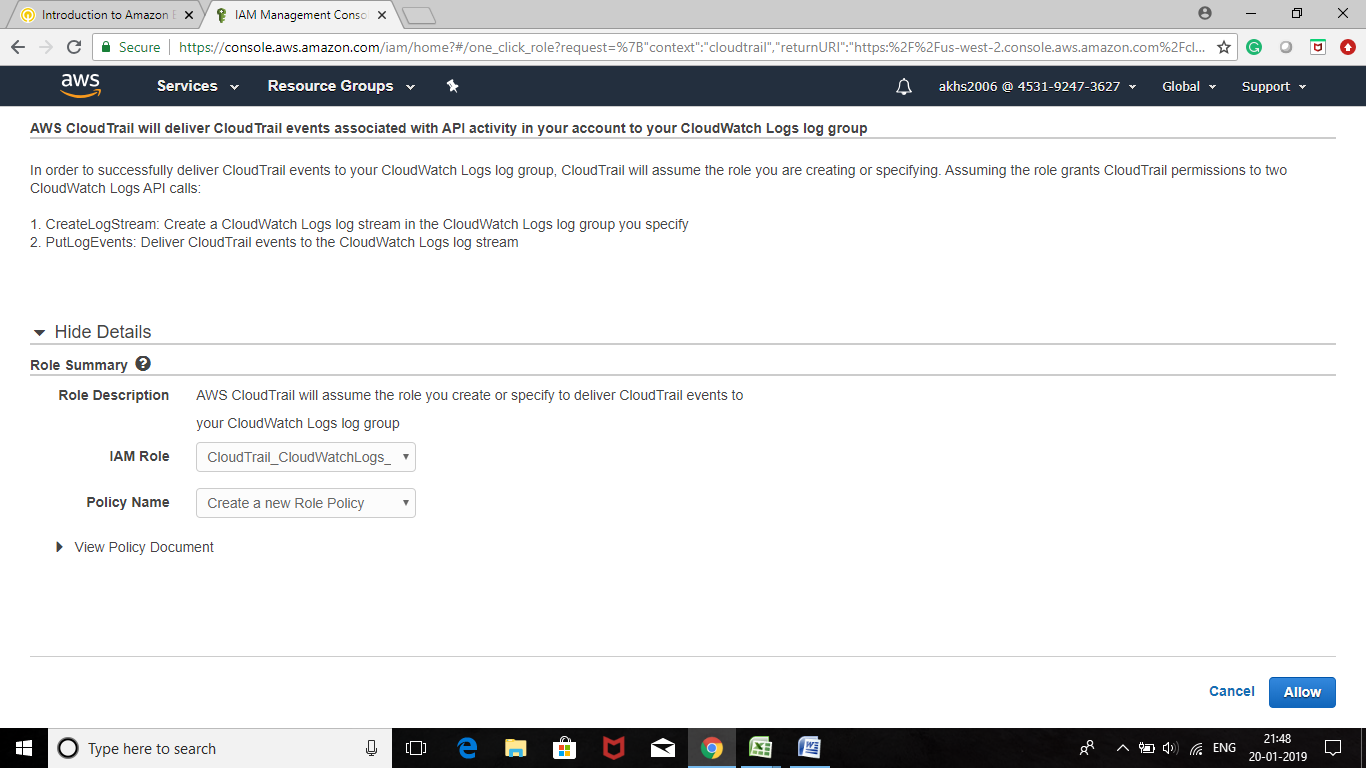
* In the list of trails, click the trail that is associated with your region.
* Scroll down to the **CloudWatch Logs** section.
* Click Configure then configure:
* **New or existing log group:** *esLogGroup*
* **Click** Next
* . Expand ► **View Details**, then configure:
* Policy Name: *Cloud TrailLogsPolicy*
* ClickAllow

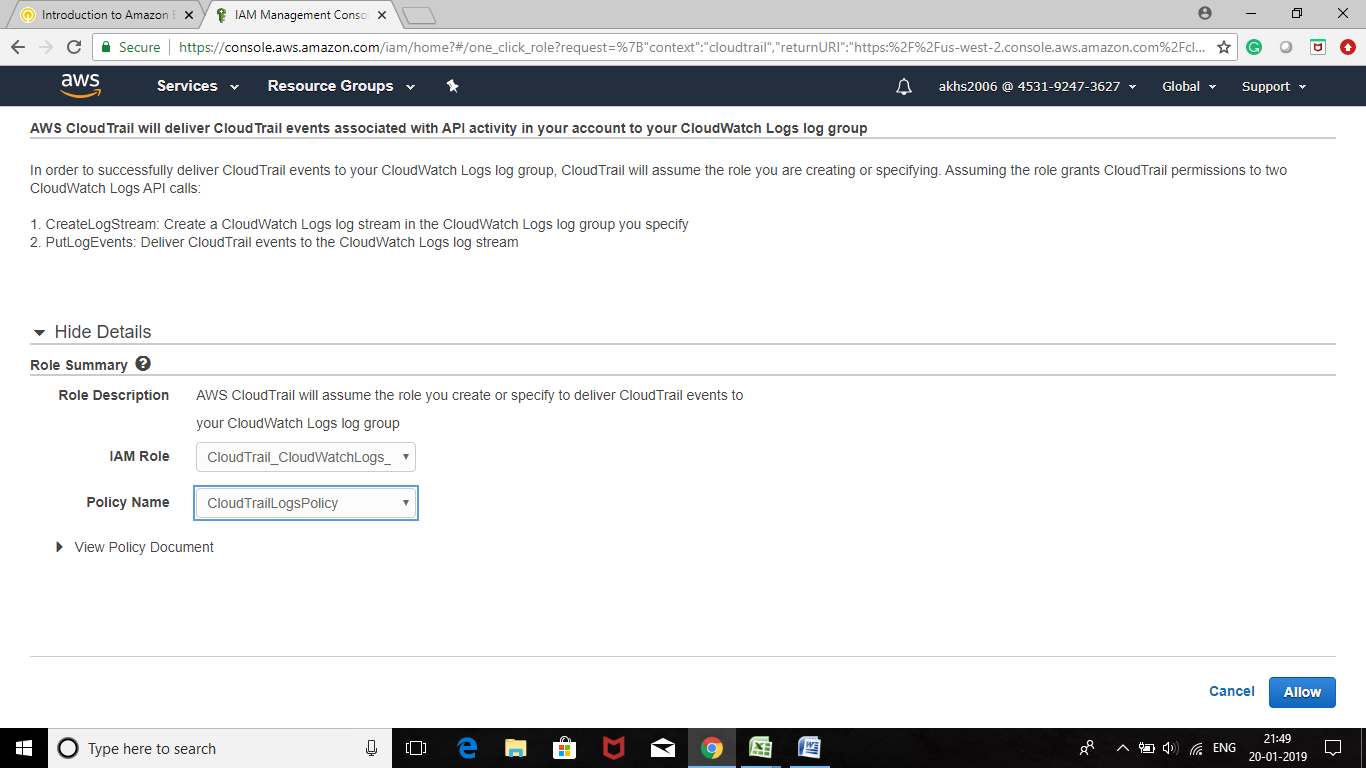












**Task 3: Subscribe the CloudWatch Log**

Group to Amazon Elasticsearch Service

In this task, you will configure Amazon CloudWatch Logs to send data to the Elasticsearch cluster.

* On the Services menu, click **Elasticsearch Service**.
* Below **Domain**, click *mydomain*.
* Wait for the **Domain status** to display Active.

If it is not yet Active, please click the refresh icon every 30 seconds until the cluster is ready. The cluster will require approximately 10 minutes to launch, starting from the time that you launched the cluster earlier in the lab.

* On the services menu, click **Cloud Watch.**
* In the left navigation pane, click **Logs**.

The CloudTrail log group named **esLogGroup** will be displayed. This is the log group that you created earlier.

* In the **Log Groups** page, select **esLogGroup.**
* In the **Actions** menu, select **Stream to Amazon Elasticsearch Service.**

You will now configured Amazon Cloud Watch Logs to stream the CloudTrail data to your Elasticsearch cluster.

* In the **Amazon ES cluster** drop-down list, select *mydomain*

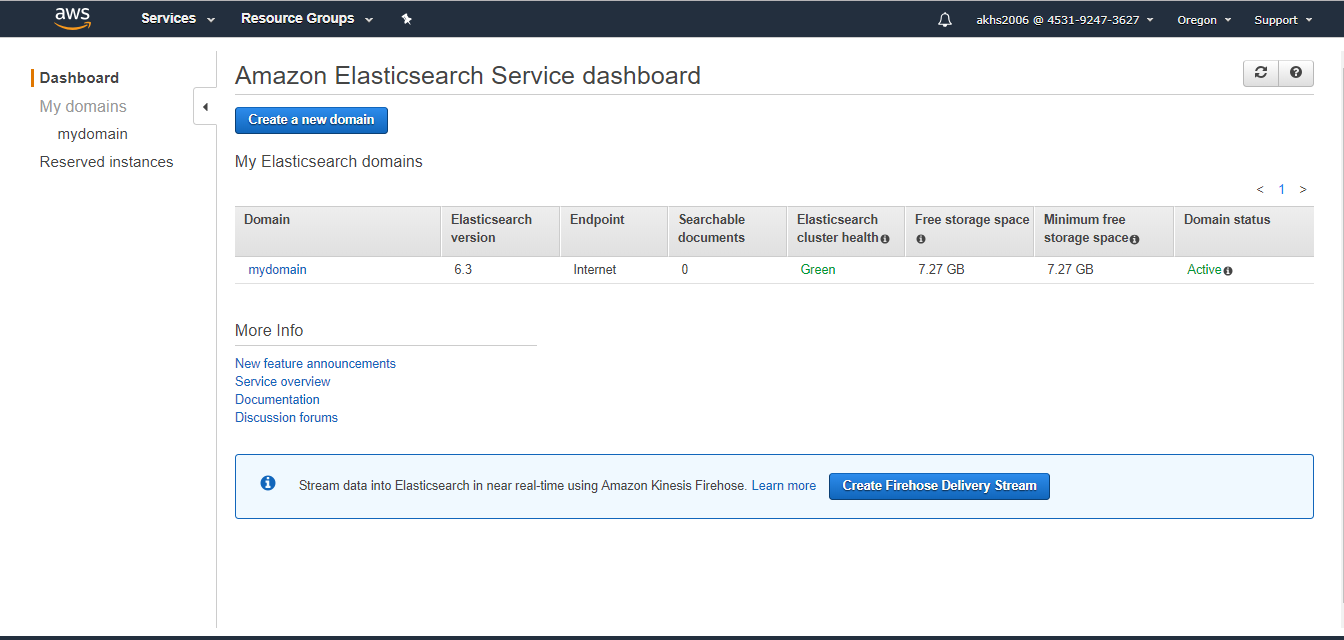
This is the Amazon Elasticsearch Service Cluster that you created earlier.

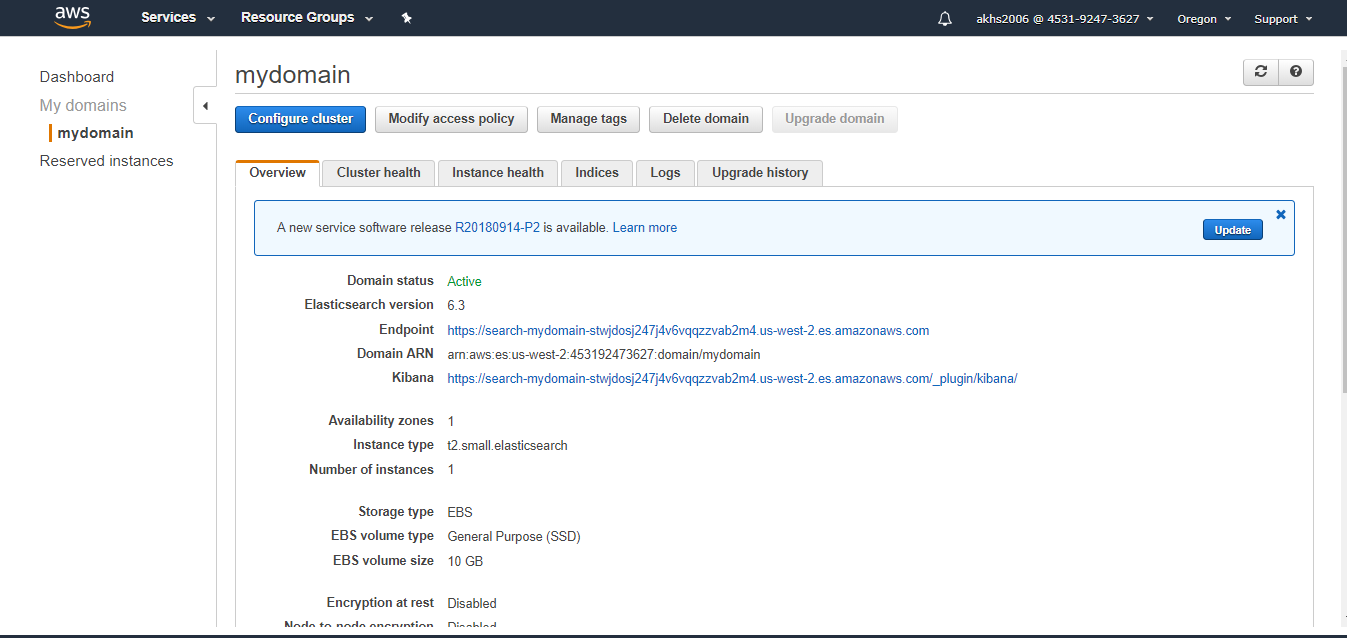
* In the **Lambda IAM Execution Role** drop-down list, select *lambda\_basic\_execution.*
* **Click**Next

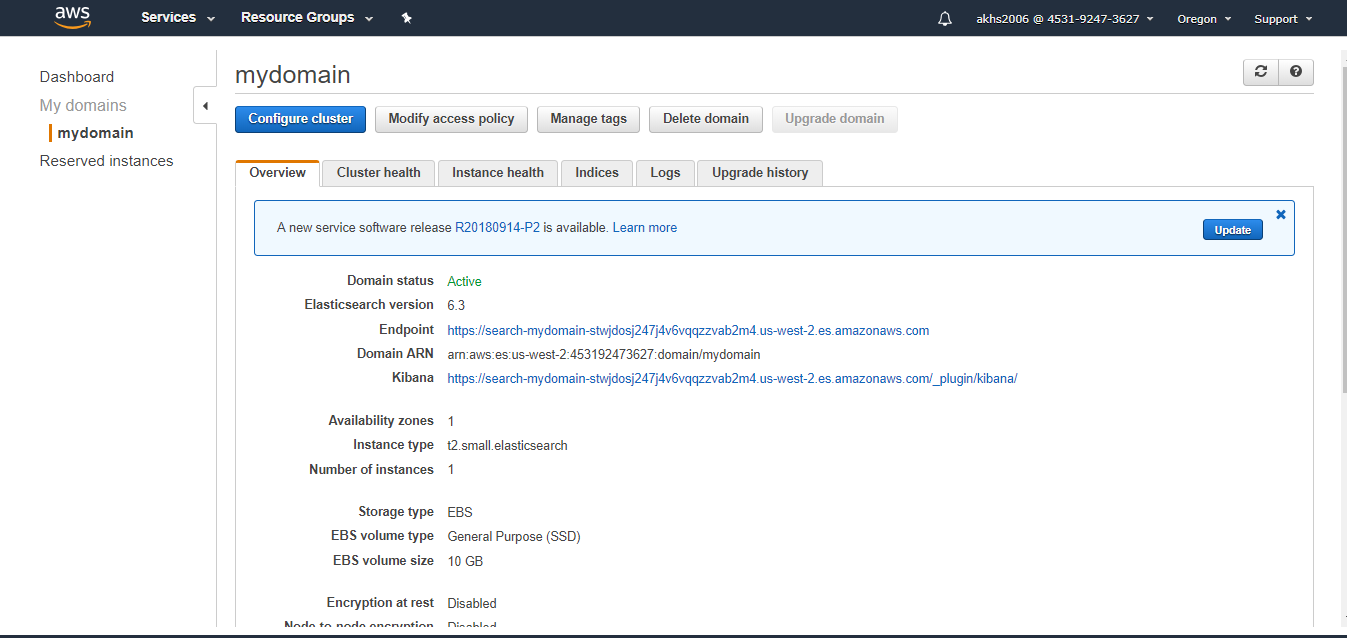
You will now be taken to the **Configure Log Format and Filters** page, where you will configure the format of data that will be sent to the Elasticsearch cluster.

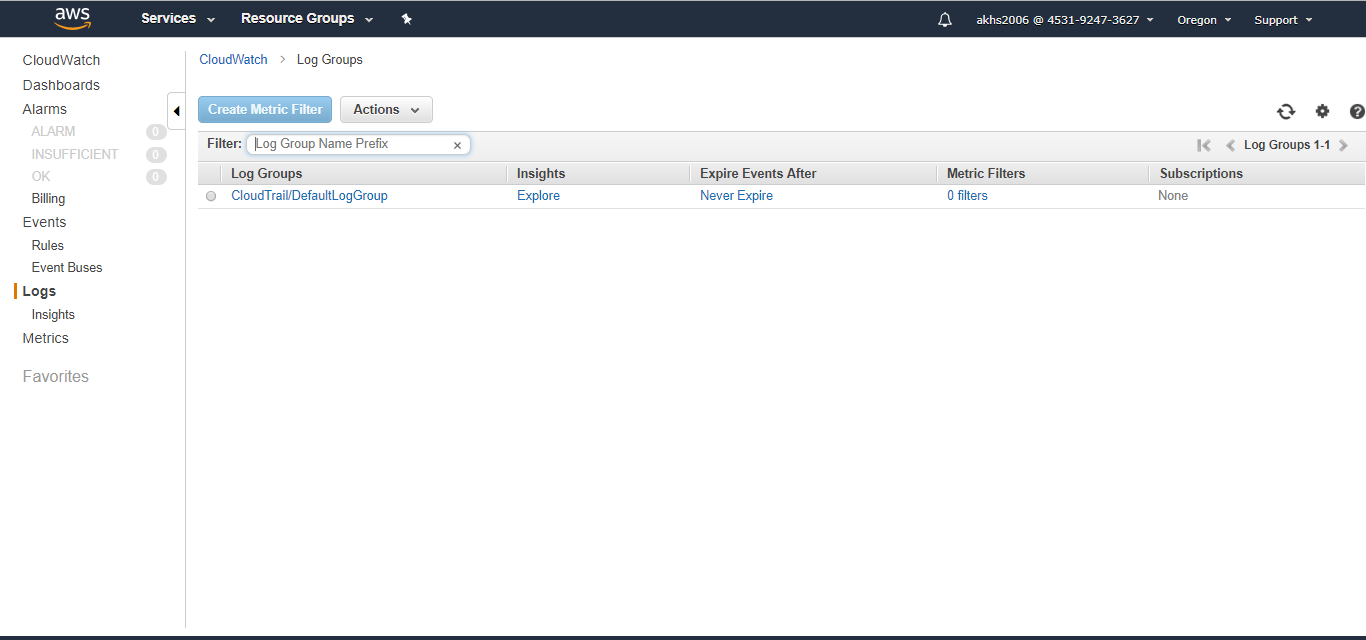
* In the **Log Format** drop-down list, select *AWS CloudTrail.*
* In the **Select Log Datato Test** section, click Test Pattern
* Click Next
* Review your settings, then click Next again.
* Click Start Streaming

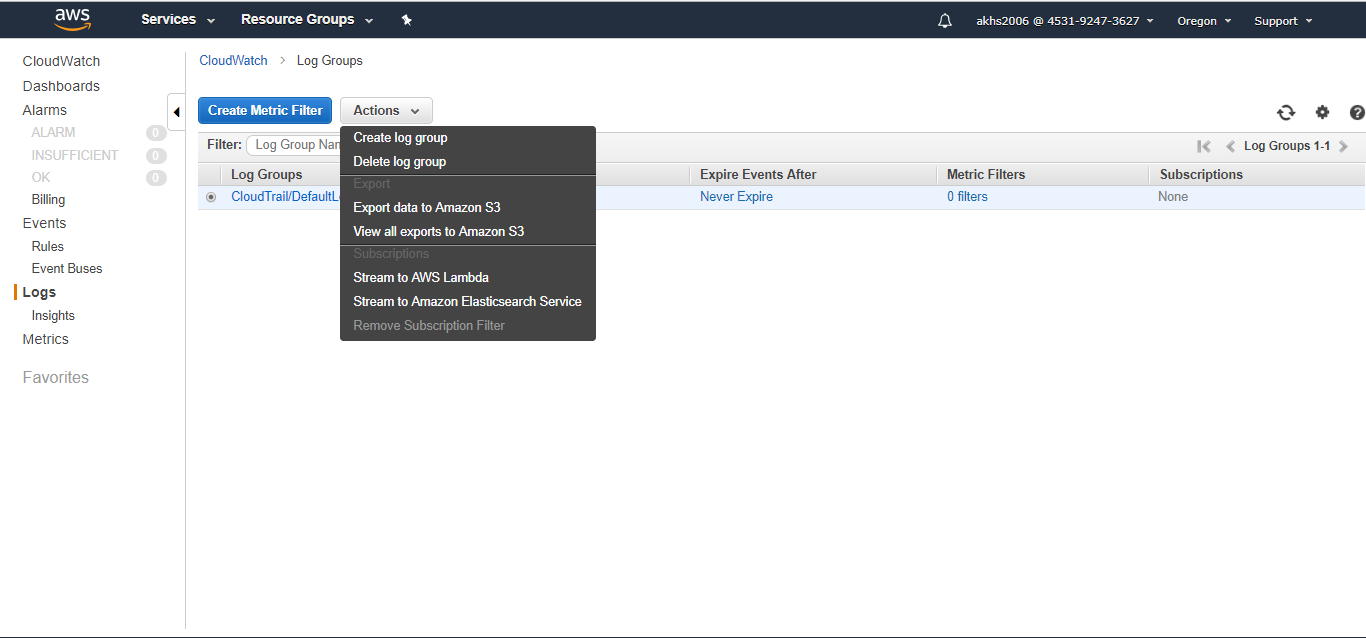
You will be informed that*yourSubscription Filter has been created.*

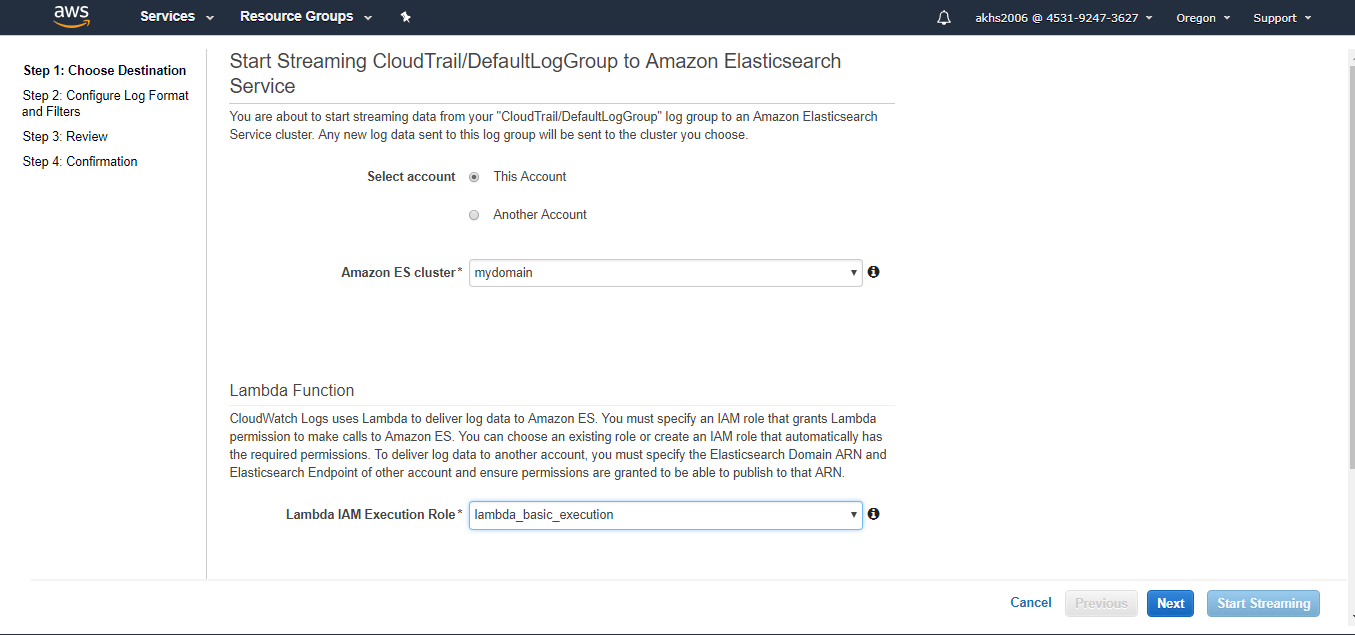


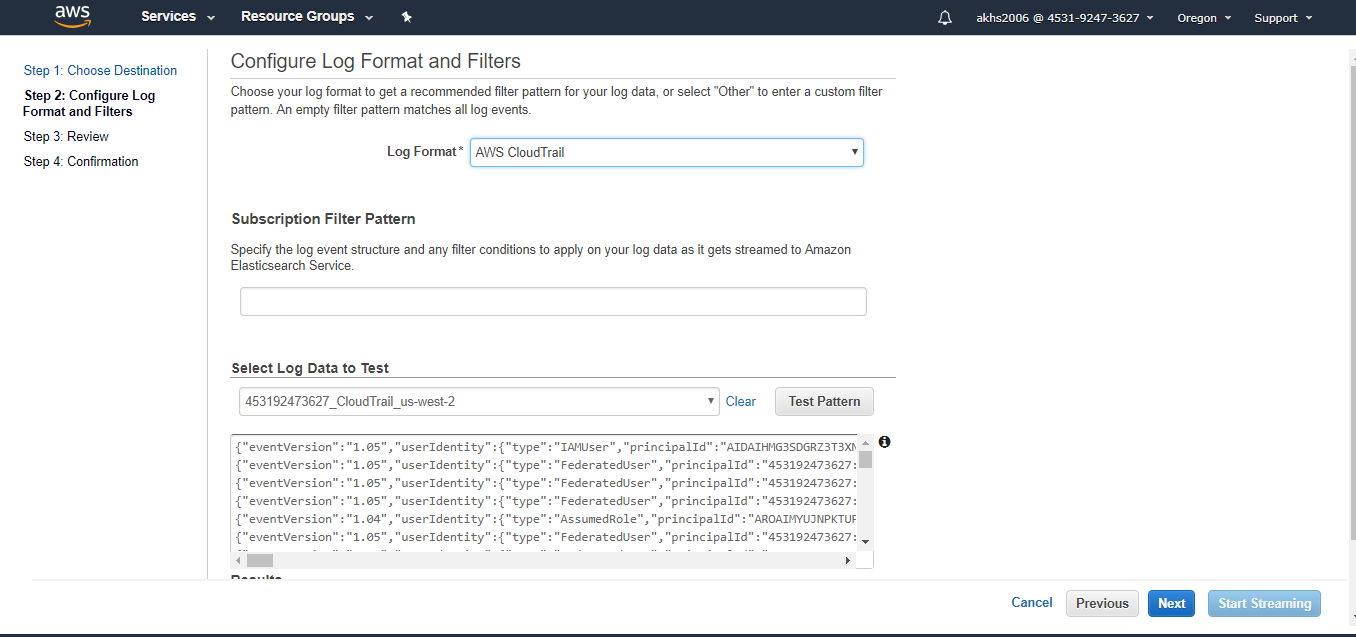


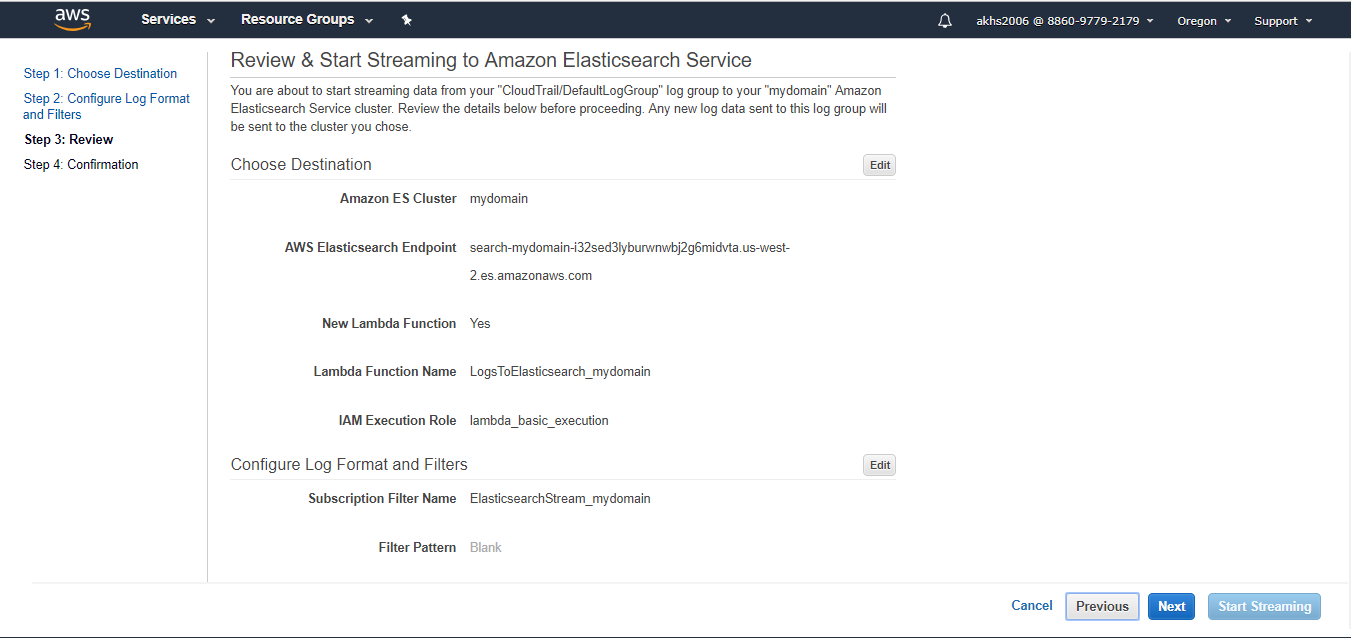


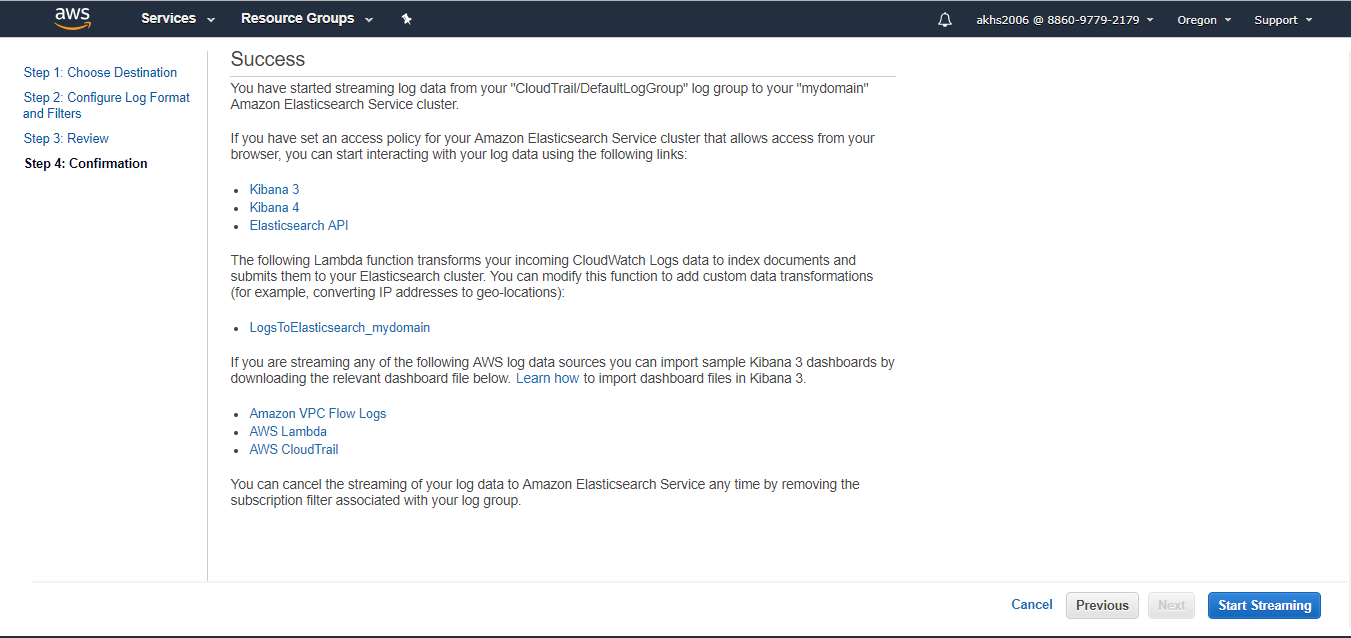


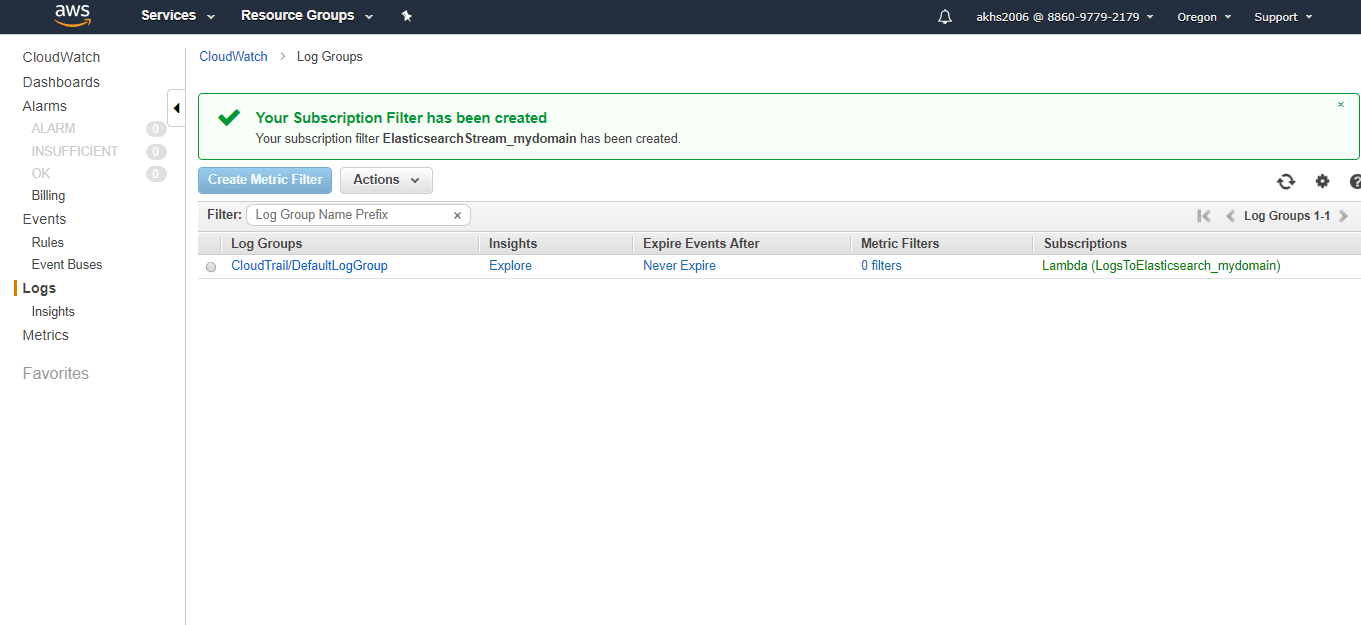












**Task 4: Configure Kibana for Reporting**

In this task, you will configure Elasticsearch and Kibana to report on the CloudTrail data.

* On the **Services** menu, click **Elasticsearch Service.**
* In the left navigation pane, click **mydomain.**
* To interact with your log data, click the **Kibana** link.

It should look similar to:[*https://vpc-lab-IvIs3xivtvkulw2dqxwymrhbo4.us-west­*](https://vpc-lab-IvIs3xivtvkulw2dqxwymrhbo4.us-west) *2.es.amazonaws.com/ plugin/kibana/A* new browser tab will open with the Kibana console.

**Kibana** is a web interface for Elasticsearch Service that provides visualization capabilities on top of the content indexed on an Elasticsearch cluster. You can create bar, line and scatter plots, or pie charts and maps on top of large volumes of data. You can also do comparisons of queries across different time ranges

**Below steps for practice:**

* In the left navigation pane, click **Visualize**.
* If you see a message similar to the following**: In order to visualize and explore data in Kibana, you'll need to create an index pattern to retrieve data from Elasticsearch**., wait a few minutes and then click, Check for new data

You may have to wait a few minutes for your data to received.

* In the **Index pattern** field, enter \*
* Click >Next step
* For **Time Filter field name**, select *@timestamp*

You will need to wait a few minutes and click **refresh fields** for this drop-down list to populate. It will eventually show the **@timestamp** field, which indicates that data has been received.

* Click Create Index Pattern

A list of available data fields will be displayed.

* In the left navigation pane, click **Discover.**
* At the top-right of the screen, change the time-interval to **Last 30 minutes.**

A sample of the CloudTrail audit data is displayed. This includes:

* **principalld:** A unique identifier for the entity that made the call.
* **accountld:** The account that owns the entity that granted permissions for the request.
* **accessKeyld:** The Access Key that signed the request.

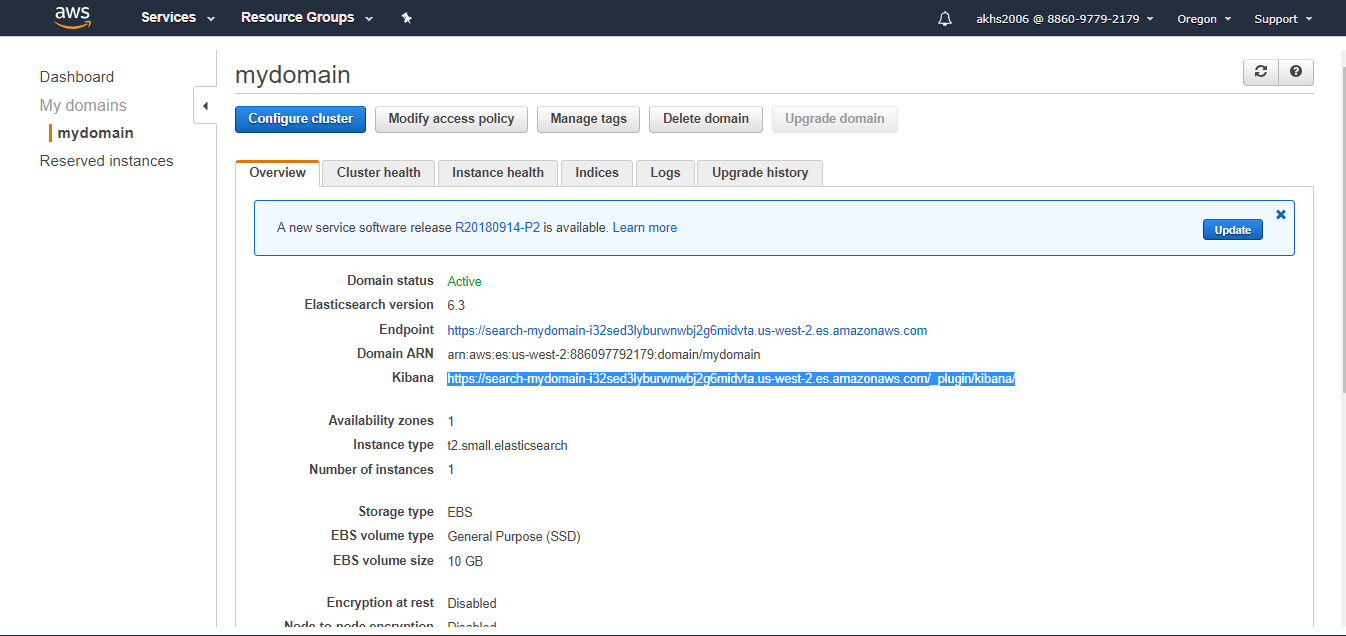
You can also visualize the log data as a line chart.

* In the left navigation pane, click **Visualize.**
* Click the **Line** chart It
* Click +Create Visualization
* In the **From aNew Search, Select Index**: column, Click \*.

You will now configure a chart to show API requests over time.

* Under **Select buckets type**, click **X-Axis**, then configure:
  + **Aggregation**: *Date Histogram*
  + **Field**: *@timestarnp*
* ClickAdd sub-buckets
* Click **Split Series**, then configure:
  + **Sub Aggregation**: *Terms*
  + **Field**: *eventName.keyword*
* Click the play button at the top of the pane to view the chart.

The chart will display API calls over time. This data is being received in real-time from AWS CloudTrail via Amazon CloudWatch Logs.



**Amazon Redshift**

**Overview**

This lab provides an overview of Amazon Redshift. In this lab, you will launch and work with an Amazon Redshift cluster to analyze USA Domestic flight data.

Amazon Redshift is a **fast, fully managed, petabyte-scale data warehouse service** that makes it simple and cost-effective to efficiently analyze all your data using your existing business intelligence tools. It is optimized for datasets ranging **from a few hundred gigabytes to a petabyte** or more and costs less than S1,000 per terabyte per year, a tenth the cost of most traditional data warehousing solutions.

Amazon Redshift delivers fast query and I/O performance for virtually any size dataset by using **columnar storage technology** and **parallelizing and distributing queries across multiple nodes.** We've made Amazon Redshift easy to use by automating most of the common administrative tasks associated with provisioning, configuring, monitoring, backing up, and securing a data warehouse.

**Topics Covered**

* By the end of this lab, you will be able to:
* Launch an Amazon Redshift cluster
* Connect to Amazon Redshift by using SQL client software
* Load data from Amazon S3 into Amazon Redshift
* Query data from Amazon Redshift
* Monitor Amazon Redshift performance

**Lab Pre-requisites**

To successfully complete this lab, you should be familiar with basic concepts of databases and SQL.

**Task 1: Launch your Amazon Redshift Cluster**

You will now launch an Amazon Redshift cluster, which starts your very own database for use in this lab.

* Wait until the lab status at the top of this page shows **Lab Running.** This indicates that required resources are available for your lab.
* In the **AWS Management Console**, on the **Services** menu, click **Amazon Redshift**.
* Click **Launch cluster**.
* Configure the following settings:
* **Cluster identifier**: lab
* **Database name**: lab
* **Database port**: 5439
* **Master user name**: master
* **Master user password**: Redshift123
* **Confirm password**: Enter the same password again
* Click **Continue**.
* Configure the following settings:
* **Node type**: *dc2.1arge*
* **Cluster type**: *Multi Node*
* **Number of compute nodes**: 2

This lab uses the **dc2.large** node size, which has 160GB of storage per node. You will be using a single node for this lab, but the type and number of nodes in a Redshift cluster can be changed at any time to provide extra storage and faster data processing.

* Click **Continue.**

You wIl be presented with a configuration page.

* Configure the following, leaving other settings at their default value:
* **Choose a VPC:** Select the VPC that is **not** the Default VPC
* **VPC security groups:** Redshift Security Group
* **Available IAM roles:** Redshift-Role

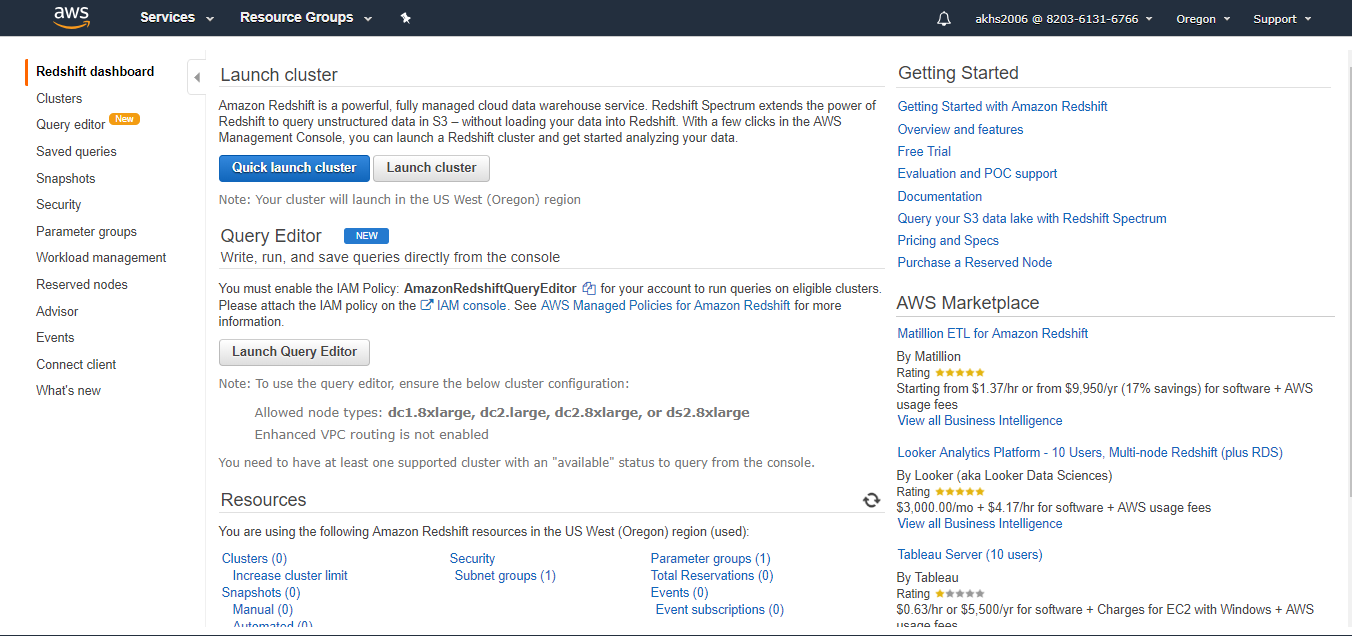
If you cannot see the *Redshift Security Group,* confirm that you have changed the VPC.

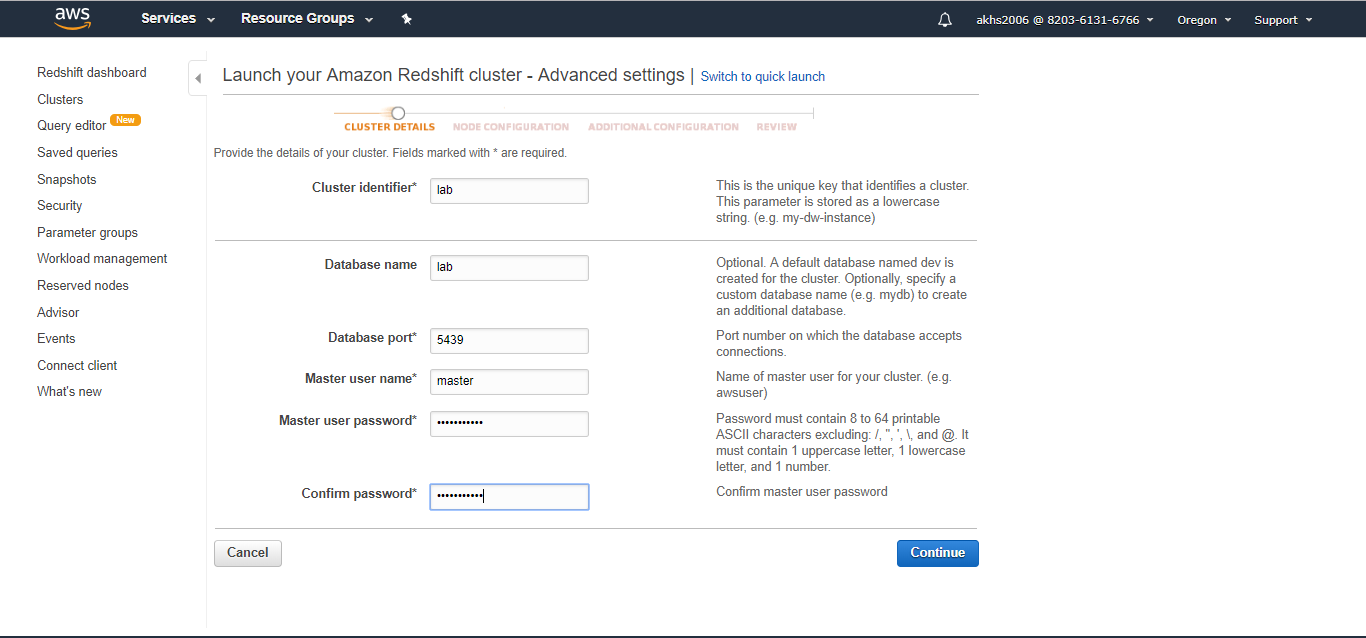
These settings define the network and security configurations for the Redshift cluster.

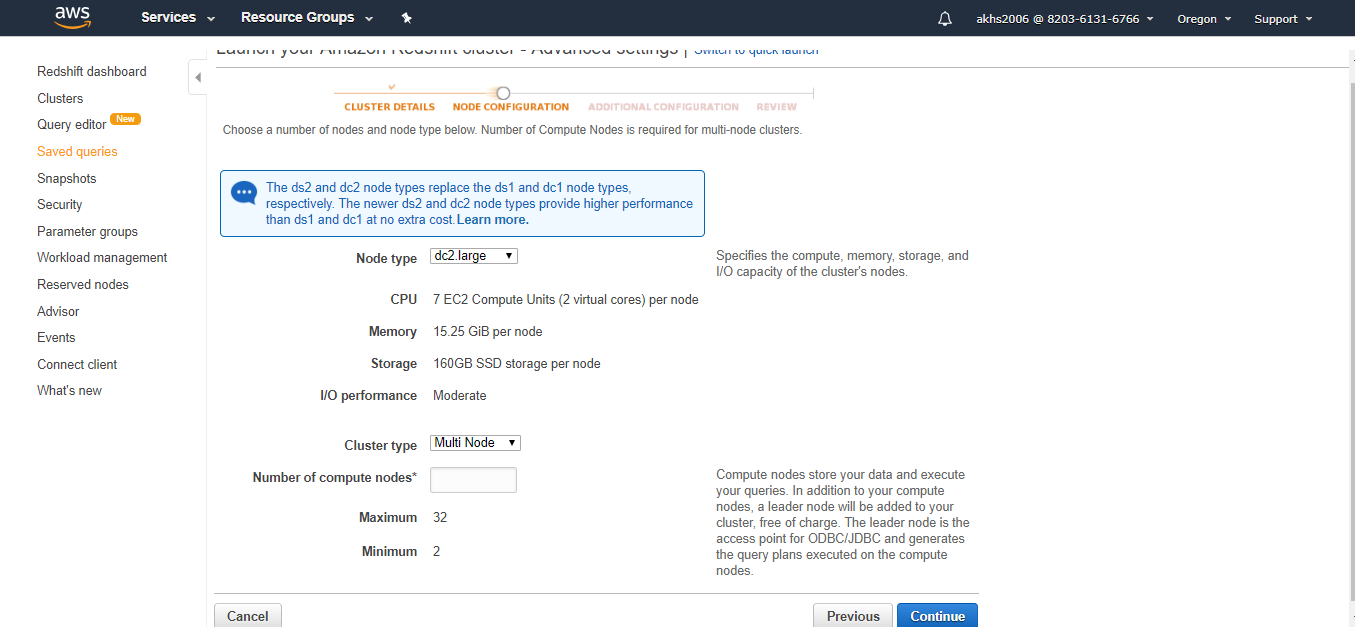
* Click **Continue**.
* Review the settings, then click **Launch cluster** (you might need to scroll down to see it).
* In the left navigation pane, click **Clusters** to view your Clusters.
* Click the **lab** link to view your cluster details.

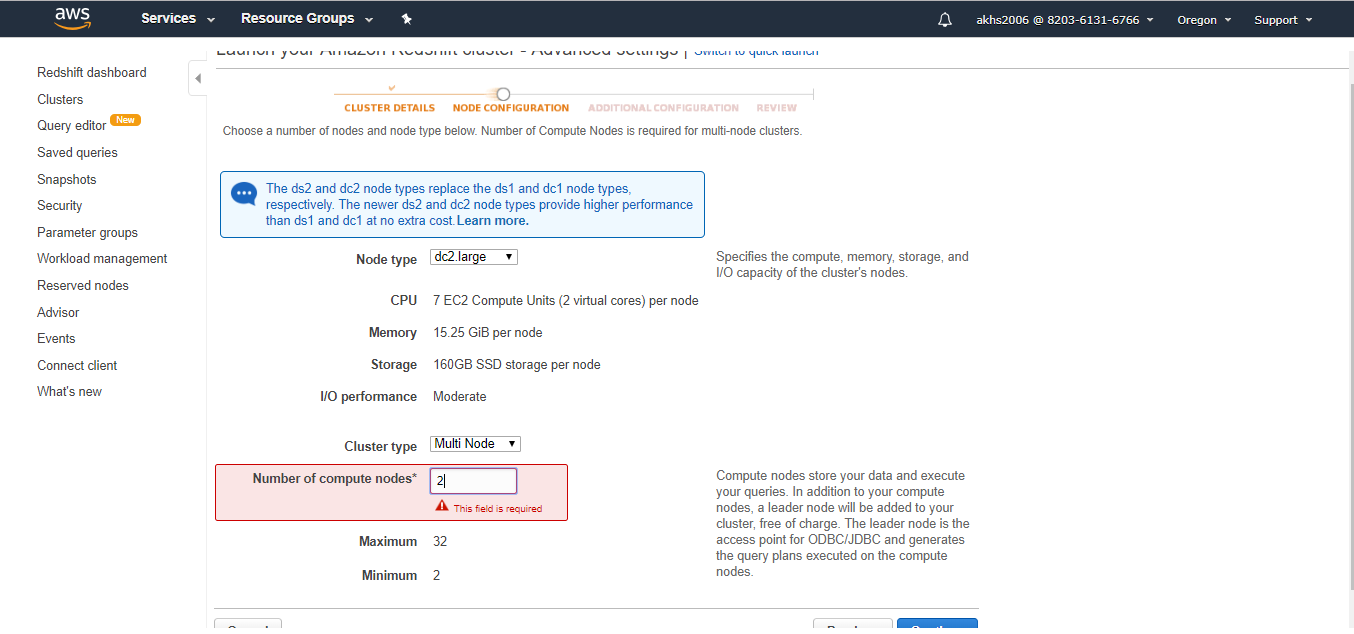
Your cluster can take up to 5 minutes to launch.

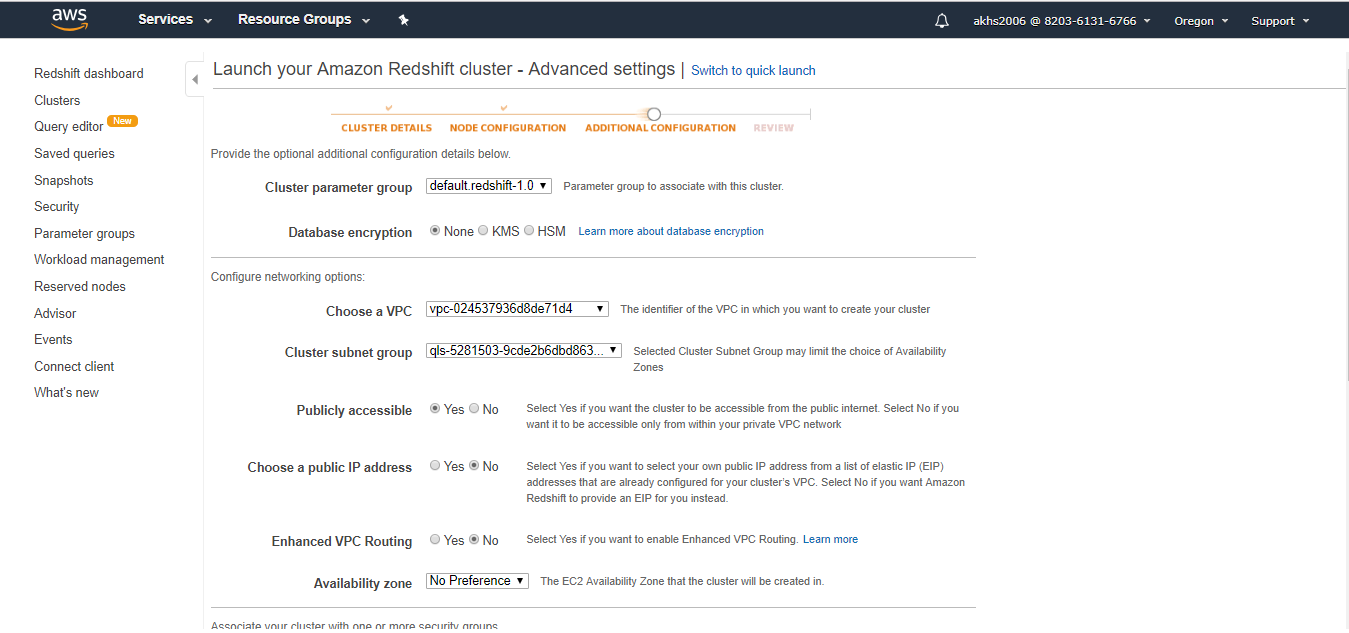
While you are waiting, please continue reading the next section.

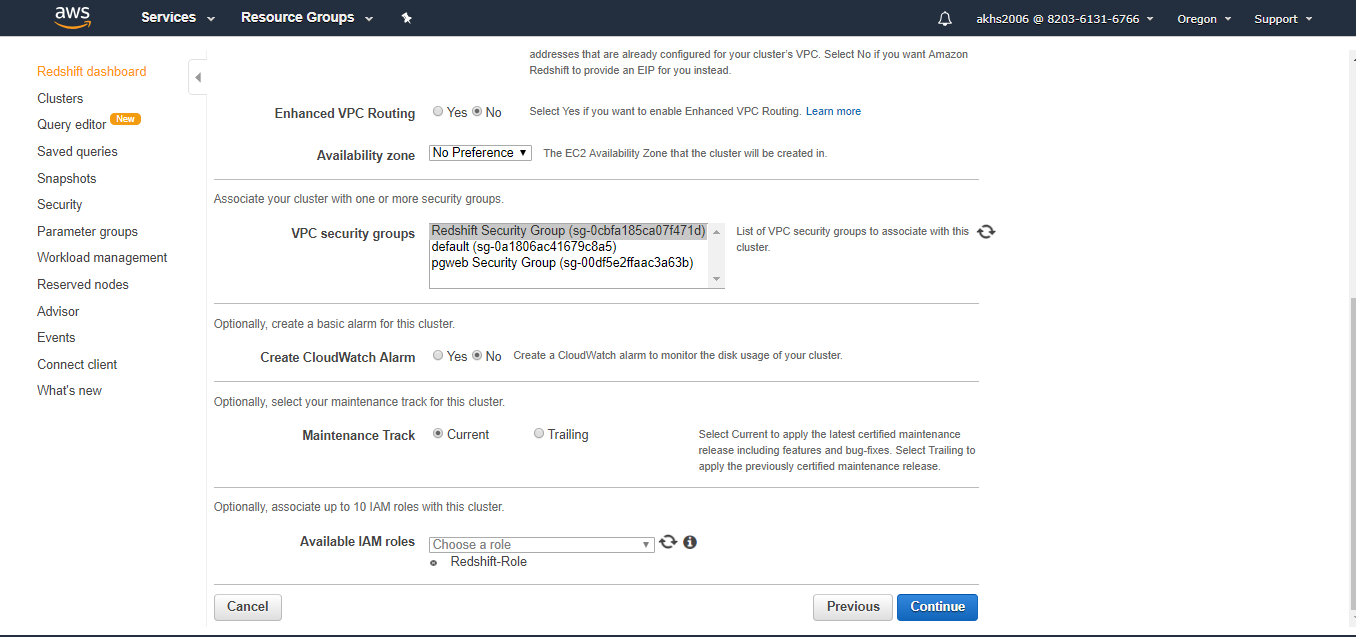


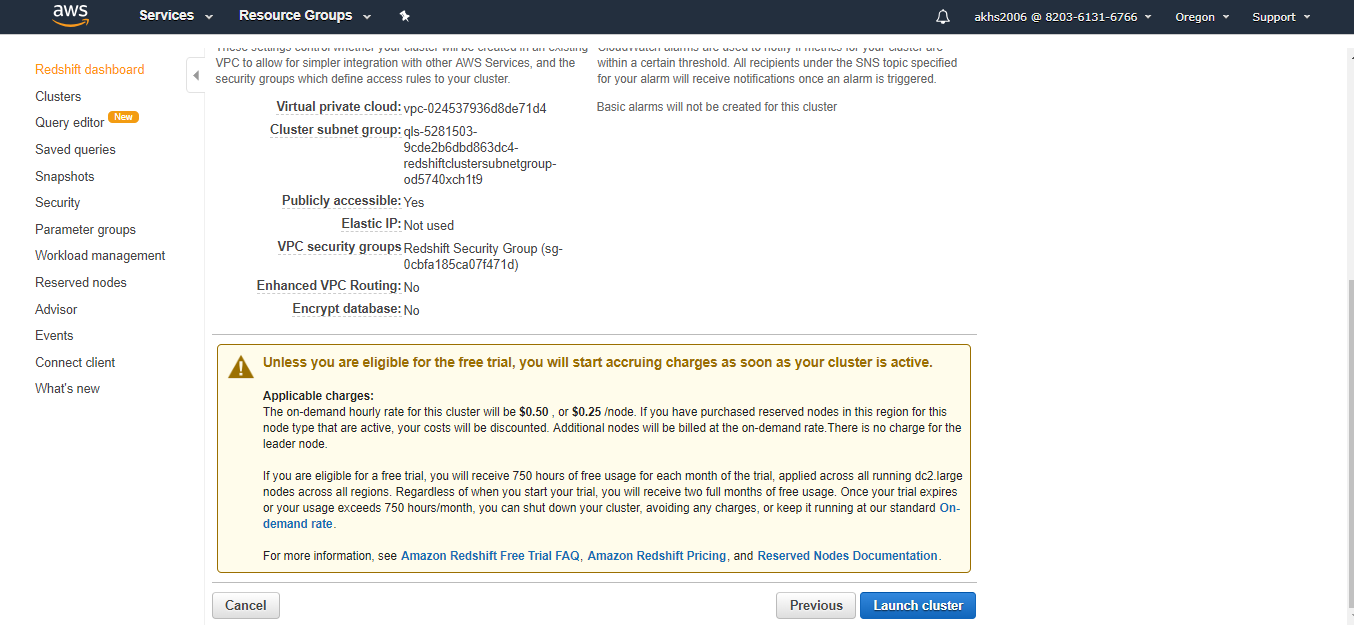


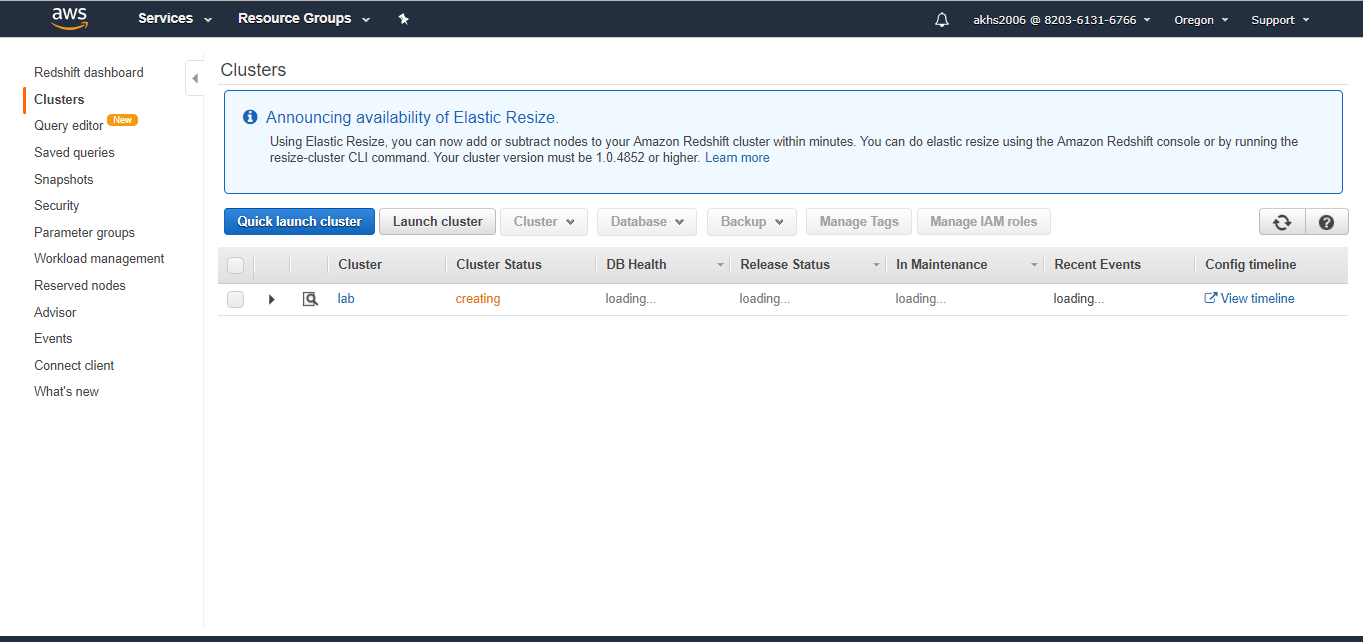


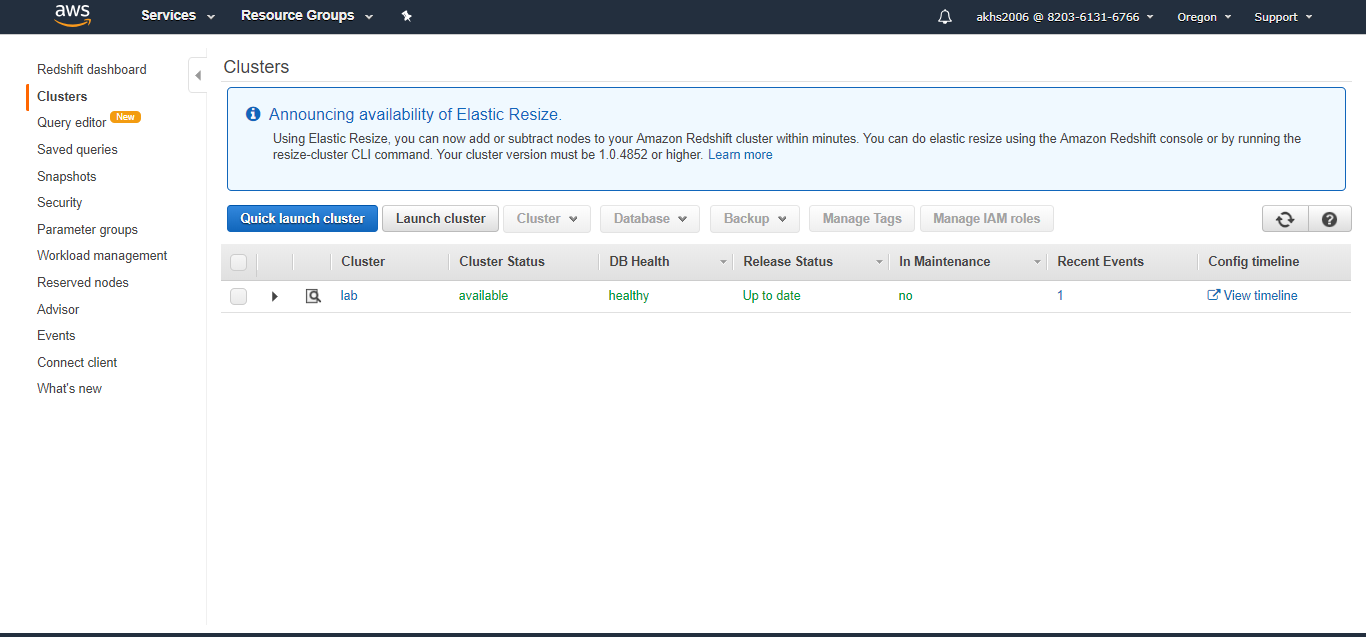










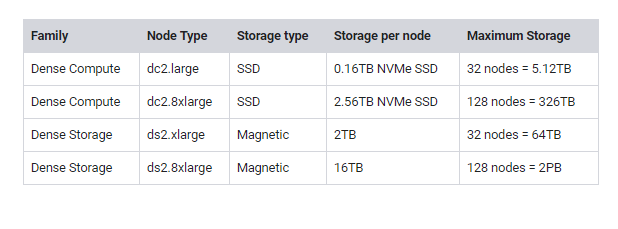


**Amazon Redshift Primer**

While you are waiting for your cluster to launch, here is some information that highlights the most important features of Amazon Redshift.

Nodes & Clusters

An Amazon Redshift data warehouse is a collection of computing resources called **nodes**. This collection of nodes is called a **cluster**. When you provision a cluster, you specify the type and the number of nodes that will make up the cluster. The node type determines the storage size, memory, CPU, and price of each node in the cluster:



Scalability

If your storage and performance needs change after you initially provision your cluster, you can always **scale** the cluster in or out by **adding or removing nodes**, scale the cluster up or down by **specifying a different node type**, or you can do both. Resizing the cluster in either way involves minimal downtime. Resizing replaces the old cluster at the end of the resize operation. When you submit a resize request, the source cluster remains in read-only mode until the resize operation is complete.

Parallel Processing

Amazon Redshift **distributes workload** to each node in a cluster and **processes work in parallel**, allowing processing speed to scale in addition to storage.

Columnar Storage

Columnar storage for database tables is an important factor in optimizing analytic query performance because it drastically **reduces the overall disk I/O requirements** and **reduces the amount of data you need to load from disk.**

Rather than storing data values together for a whole row, Amazon Redshift stores data by **column**. This means that operations on a column require less disk I/0.

Compression

Compression is a column-level operation **that reduces the size of data when it is stored**. Compression conserves storage space and reduces the size of data that is read from storage, which reduces the amount of disk I/O and therefore **improves query performance**.

Snapshots as Backups

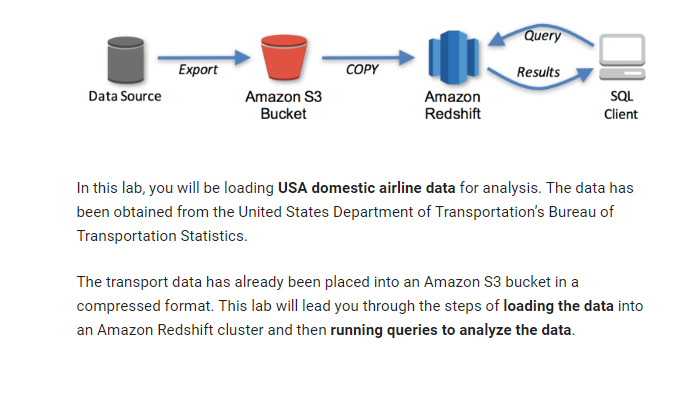
Snapshots are **point-in-time backups** of a cluster. You can create snapshots automatically or manually. Amazon Redshift stores these snapshots internally in **Amazon S3** using an encrypted Secure Sockets Layer (SSL) connection. If you need to restore a cluster, Amazon Redshift creates a new cluster and imports data from the snapshot that you specify.

Integrates with Existing Business Intelligence Tools

Amazon Redshift uses **industry-standard SQL** and is accessed using standard JDBC and ODBC drivers. Your existing Business Intelligence tools can easily integrate with Amazon Redshift.

Data Loading Process

* The typical process for loading data into Amazon Redshift is:
* Data is **exported** from a source system (for example, a company database).
* The data is placed into an **Amazon S3 bucket,** preferably in a compressed format to save storage space.
* The data is **copied into Amazon Redshift tables** via the COPY command.
* The **SQL client** is used to **query** Amazon Redshift.
* The **results** of the query will be returned to the SQL client



In this lab, you will be loading **USA domestic airline data** for analysis. The data has been obtained from the United States Department of Transportation's Bureau of Transportation Statistics.

The transport data has already been placed into an Amazon S3 bucket in a compressed format. This lab will lead you through the steps of **loading the data** into an Amazon Redshift cluster and then **running queries to analyze the data.**

**Connect to Amazon Redshift**

In this task you will use a web-based PostgreSQL client (“pgweb”) to connect to Redshift

* Copy the pgweb IP address . This is the IP address of a web

Server that is running the pgweb software

* Open a new tab in your web browser , paste the ip address and hit enter

You will be presented with the pgweb login screen

* Configure the following settings
  + - **Username**:master
    - **Password**: Redshift123
    - **Database**:lab
    - **Port**: 5439

If you cannot change the **Port** value, set **SSL** to **disable** and try again.

The **Host** value can be obtained from the Redshift management console, but you will need to wait until the cluster is available for use.

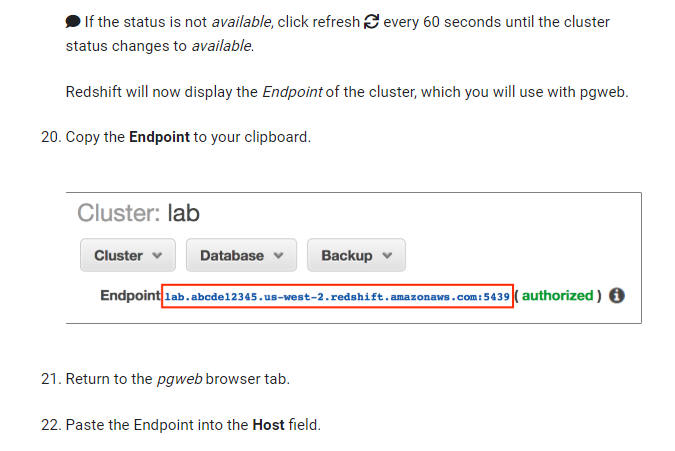
* Return to the browser tab with the Redshift management console (but do not close the pgweb tab).
* Confirm that the Cluster Status displays ***available***.



If the status is not *available,* click refresh re; every 60 seconds until the cluster status changes to *available.*

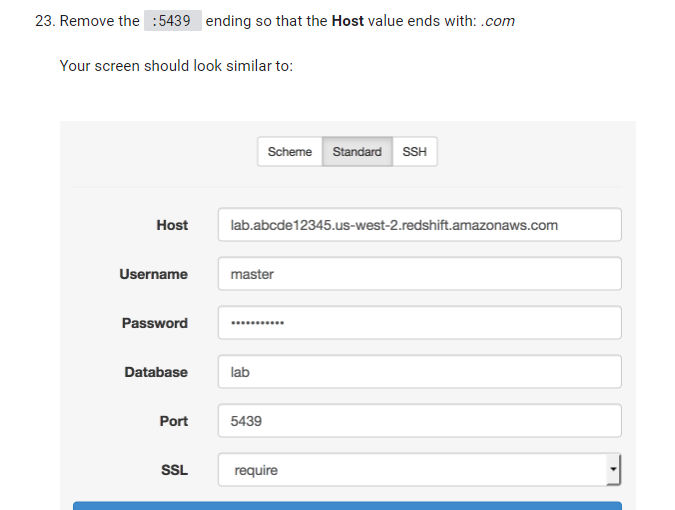
Redshift will now display the *Endpointof* the cluster, which you will use with pgweb.

* Copy the **Endpoint** to your clipboard.



* Return to the pgweb browser tab.
* Paste the Endpoint into the Host field.
* Remove the :5439 ending so that the **Host** value ends with: .com

Your screen should look similar to:



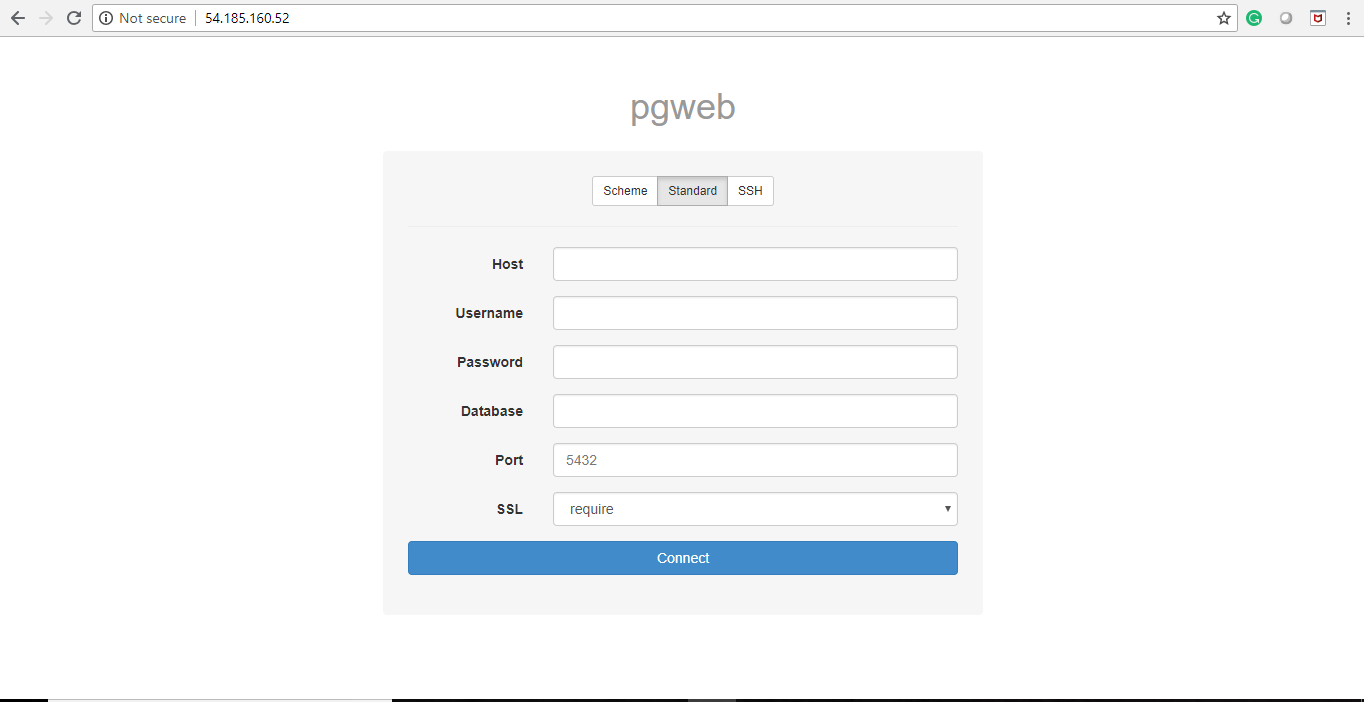
* Click Connect.

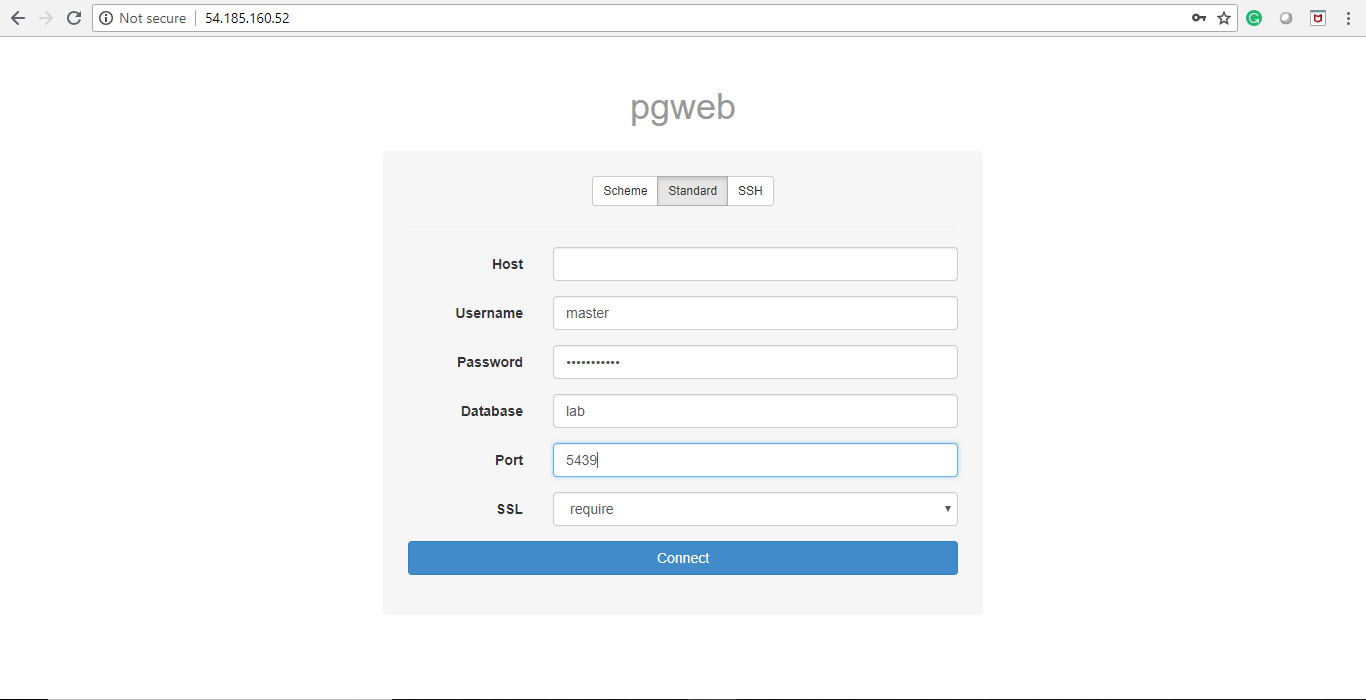
You are now ready to interact with Amazon Redshift.

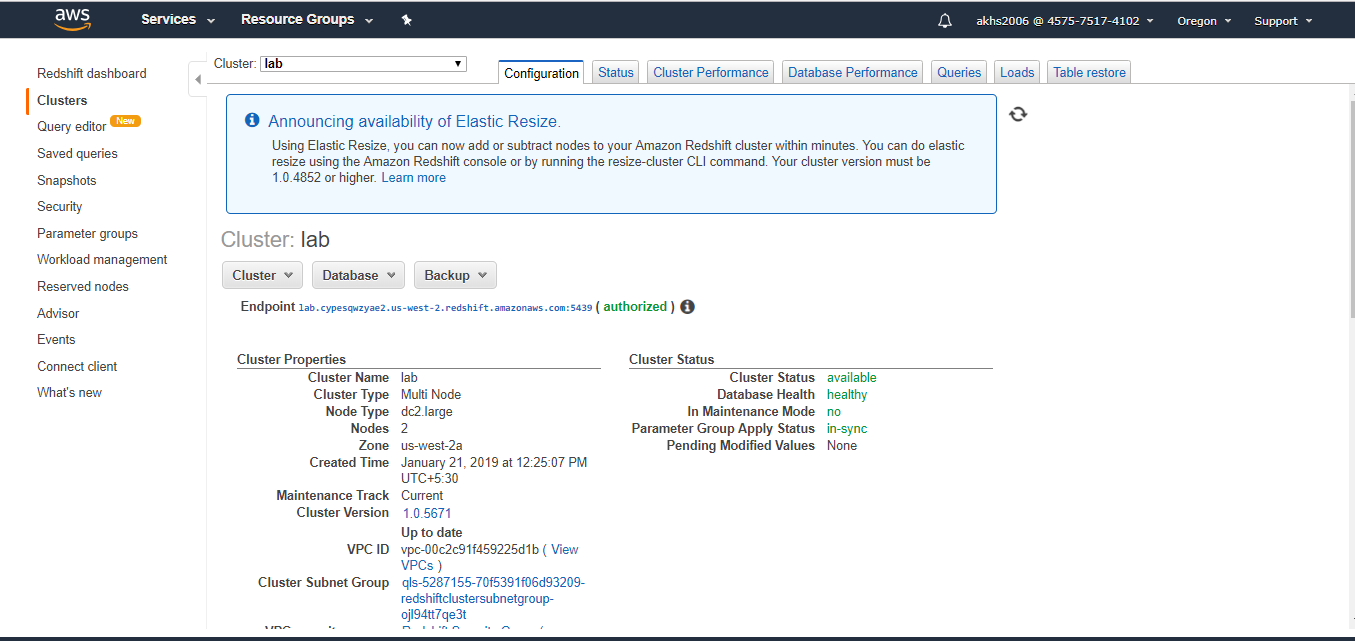
If you are not able to connect, check your configuration. Your configuration should look similar to the screenshot above.

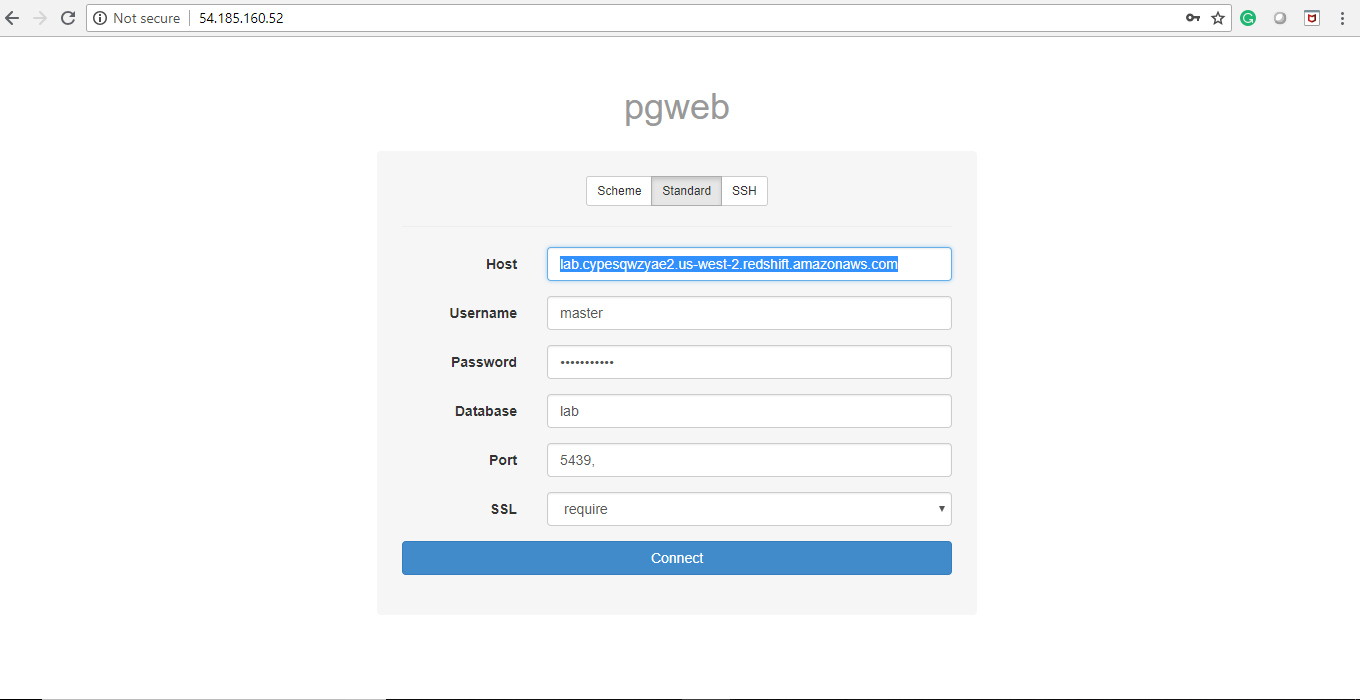
**Note:** To connect to Amazon RedShift you need to create a server and run a pgweb software on it.

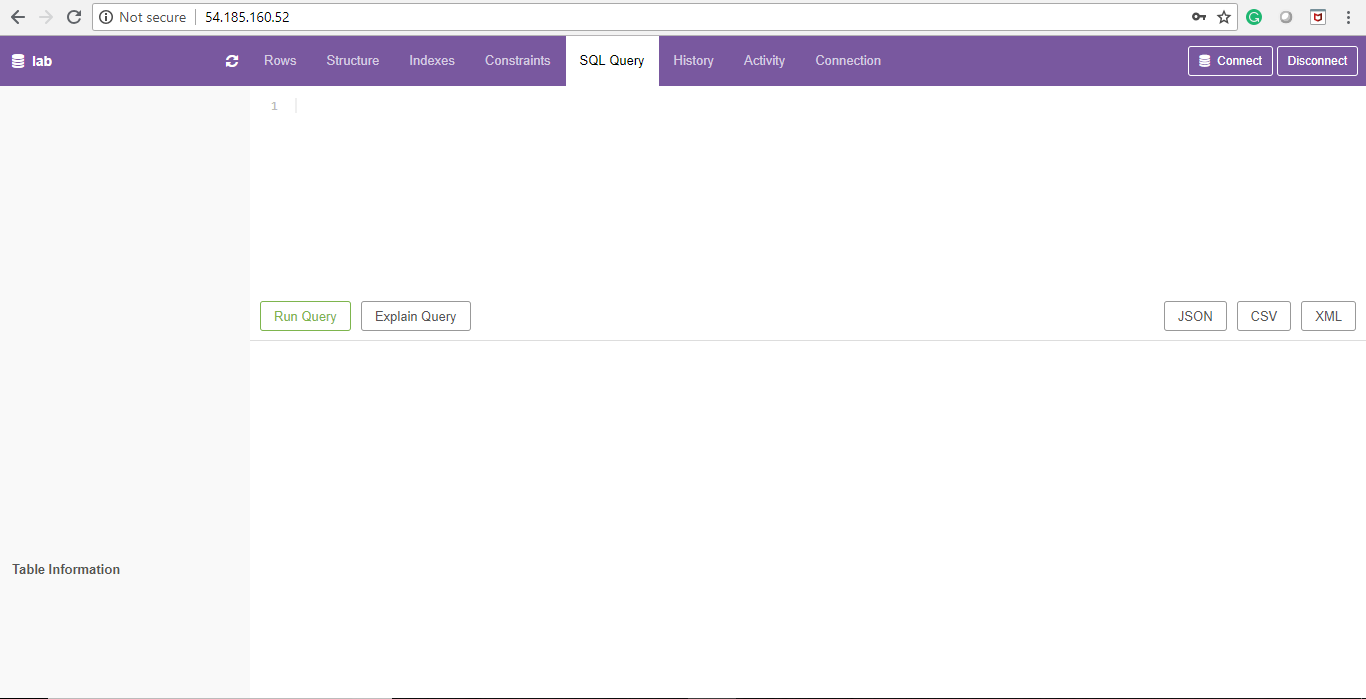
You will be presented with the pgweb login screen







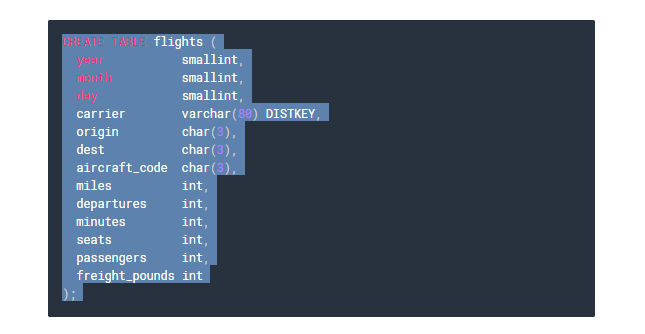




**Task 3: Load data**

In this task, you will create a Table in Amazon Redshift. Tables are used to store a particular set of information.

The first table will be used to store flight information.



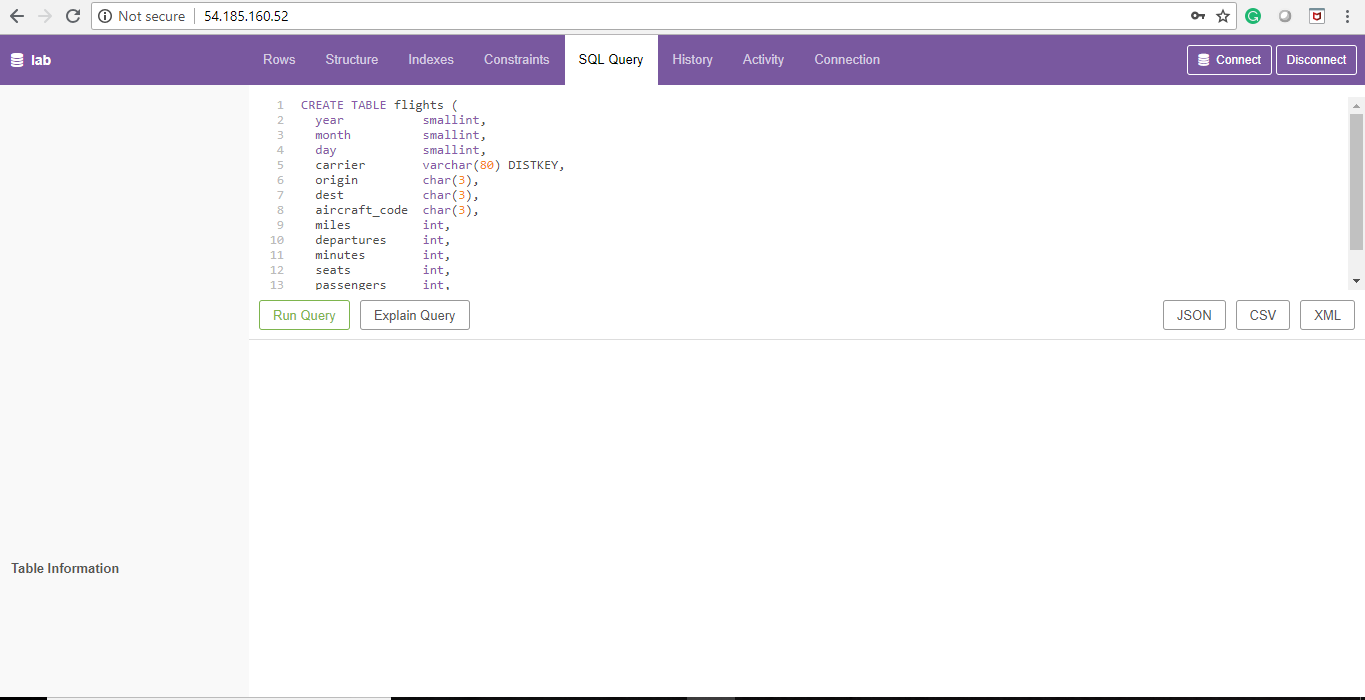
* Copy and paste the following text into pgweb (above the Run Query button):

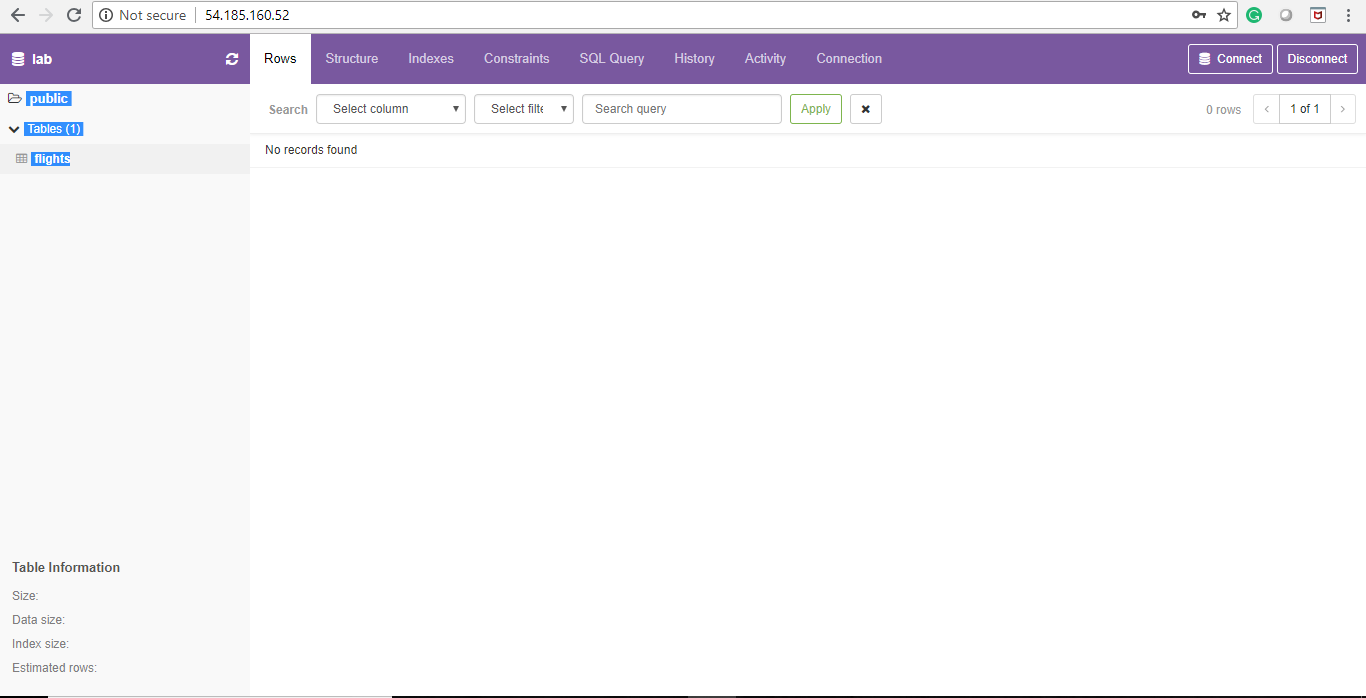
CREATETABLEflights(yearsmallint,monthsmallint,daysmallint,carriervarchar(80)DISTKEY,originchar(3),destchar(3),aircraft\_codechar(3),milesint,departuresint,minutesint,seatsint,passengersint,freight\_poundsint);

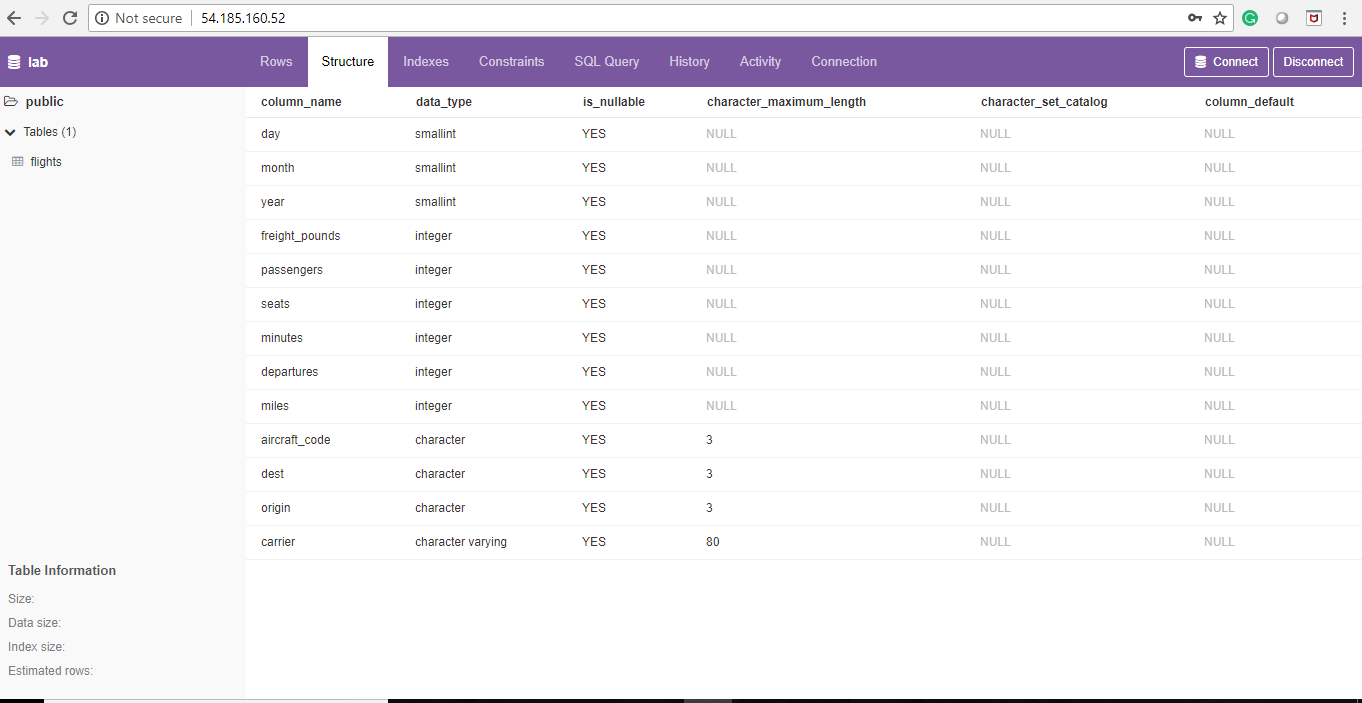
* Click **Run Query.**

A new **flights** table will appear on the left of the screen, under the Tables heading.

You can now **load data** into the table. The data has already been placed into an Amazon S3 bucket and can be loaded into Amazon Redshift by using the COPY command.







**For Assignment: Follow the below steps**

* Delete the query text.
* Paste the following text into pgweb **but do not run it yet:**



* Replace **INSERT-YOUR-REDSHIFT-ROLE** in the third line with the **RedshiftRole** value shown to the left of these instructions. The result should look similar to this:
* Click **Run Query**.

It will take **approximately 3 minutes** to load the data. Please continue reading while the data is loading.

The **COPY** command is used to load data into Amazon Redshift:

* **FROM**: Indicates where the data is located
* **IAM\_ROLE**: Provides the permissions to access the data being loaded
* **GZIP**: Indicates that the data has been compressed (zipped) — Amazon Redshift will automatically decompress the data when it is loaded
* **DELIMITER**: Indicates that data items are separated by a comma
* **REMOVEQUOTES**: Tells Amazon Redshift to remove quotation marks that are included in the data
* **REGION**: Indicates which AWS region contains the S3 bucket

The data being loaded consists of:

* 23 data files in CSV format (one for each year from 1990 - 2012)
* Comprising **6 GB of data**
* Compressed with GZIP down to only 700 MB of storage

The data files are being loaded **in parallel** from Amazon S3. This is the most efficient way to load data into Amazon Redshift since the load process is distributed across multiple slices across all available nodes.

Each slice of a compute node is allocated a portion of the node's memory and disk space, where it processes a portion of the workload assigned to the node. The leader node manages distributing data to the slices and apportions the workload for any queries or other database operations to the slices. The slices then work in parallel to complete the operation.

When you create a table, you can optionally specify one column as the **distribution key**. When the table is loaded with data, the rows are distributed to the node slices according to the distribution key. Choosing a good distribution key enables Amazon Redshift to use parallel processing to load data and execute queries efficiently.

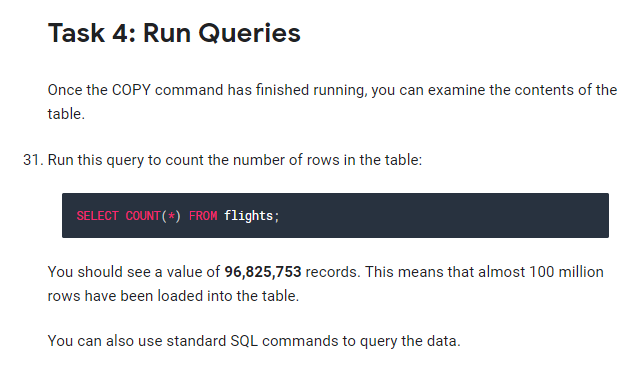
The CREATE TABLE command you ran earlier designated the carrier (airline) field as the **Distribution Key** (DISTKEY). This means the data will be split between the all available slices and nodes, but all data related to a particular carrier will always reside on the same slice. This improves processing speed when performing operations on the carrier field, such as GROUP BY and JOIN operations.

**Task 4: Run Queries**

Once the COPY command has finished running, you can examine the contents of the table.

* Run this query to count the number of rows in the table:

You should see a value of 96,825,753 records. This means that almost 100 million rows have been loaded into the table.



You can also use standard SQL commands to query the data.

**Kinesis Streaming using CloudFormation Template**

Copy the below template in the Word and upload in Cloud Formation service

{

"AWSTemplateFormatVersion" : "2010-09-09",

"Description" : "The Amazon Kinesis Data Visualization Sample Application",

"Parameters" : {

"InstanceType" : {

"Description" : "EC2 instance type",

"Type" : "String",

"Default" : "t2.micro",

"AllowedValues" : [ "t2.micro", "t2.small", "t2.medium", "m3.medium", "m3.large", "m3.xlarge", "m3.2xlarge", "c3.large", "c3.xlarge", "c3.2xlarge", "c3.4xlarge", "c3.8xlarge" ],

"ConstraintDescription" : "must be a supported EC2 instance type for this template."

},

"KeyName" : {

"Description" : "(Optional) Name of an existing EC2 KeyPair to enable SSH access to the instance. If this is not provided you will not be able to SSH on to the EC2 instance.",

"Type" : "String",

"Default" : "",

"MinLength" : "0",

"MaxLength" : "255",

"AllowedPattern" : "[\\x20-\\x7E]\*",

"ConstraintDescription" : "can contain only ASCII characters."

},

"SSHLocation" : {

"Description" : "The IP address range that can be used to SSH to the EC2 instances",

"Type" : "String",

"MinLength" : "9",

"MaxLength" : "18",

"Default" : "0.0.0.0/0",

"AllowedPattern" : "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",

"ConstraintDescription" : "must be a valid IP CIDR range of the form x.x.x.x/x."

},

"ApplicationArchive" : {

"Description" : "A publicly accessible URL to the sample application archive as produced by 'mvn package'",

"Type" : "String",

"MinLength" : "7",

"MaxLength" : "255",

"Default" : "https://github.com/awslabs/amazon-kinesis-data-visualization-sample/releases/download/v1.1.1/amazon-kinesis-data-visualization-sample-1.1.1-assembly.zip"

}

},

"Conditions": {

"UseEC2KeyName": {"Fn::Not": [{"Fn::Equals" : [{"Ref" : "KeyName"}, ""]}]}

},

"Mappings" : {

"AWSInstanceType2Arch" : {

"t2.micro" : { "Arch" : "64" },

"t2.small" : { "Arch" : "64" },

"t2.medium" : { "Arch" : "64" },

"m3.medium" : { "Arch" : "64" },

"m3.large" : { "Arch" : "64" },

"m3.xlarge" : { "Arch" : "64" },

"m3.2xlarge" : { "Arch" : "64" },

"c3.large" : { "Arch" : "64" },

"c3.xlarge" : { "Arch" : "64" },

"c3.2xlarge" : { "Arch" : "64" },

"c3.4xlarge" : { "Arch" : "64" },

"c3.8xlarge" : { "Arch" : "64" }

},

"AWSRegionArch2AMI" : {

"us-east-1" : { "64" : "ami-76817c1e" },

"us-west-2" : { "64" : "ami-d13845e1" },

"eu-west-1" : { "64" : "ami-892fe1fe" },

"ap-southeast-1" : { "64" : "ami-a6b6eaf4" },

"ap-southeast-2" : { "64" : "ami-d9fe9be3" },

"ap-northeast-1" : { "64" : "ami-29dc9228" }

}

},

"Resources" : {

"KinesisStream" : {

"Type" : "AWS::Kinesis::Stream",

"Properties" : {

"ShardCount" : "2"

}

},

"KCLDynamoDBTable" : {

"Type" : "AWS::DynamoDB::Table",

"Properties" : {

"AttributeDefinitions" : [

{

"AttributeName" : "leaseKey",

"AttributeType" : "S"

}

],

"KeySchema" : [

{

"AttributeName" : "leaseKey",

"KeyType" : "HASH"

}

],

"ProvisionedThroughput" : {

"ReadCapacityUnits" : "10",

"WriteCapacityUnits" : "5"

}

}

},

"CountsDynamoDBTable" : {

"Type" : "AWS::DynamoDB::Table",

"Properties" : {

"AttributeDefinitions" : [

{

"AttributeName" : "resource",

"AttributeType" : "S"

},

{

"AttributeName" : "timestamp",

"AttributeType" : "S"

}

],

"KeySchema" : [

{

"AttributeName" : "resource",

"KeyType" : "HASH"

},

{

"AttributeName" : "timestamp",

"KeyType" : "RANGE"

}

],

"ProvisionedThroughput" : {

"ReadCapacityUnits" : "10",

"WriteCapacityUnits" : "5"

}

}

},

"Ec2SecurityGroup" : {

"Type" : "AWS::EC2::SecurityGroup",

"Properties" : {

"GroupDescription" : "Enable SSH access and HTTP access on the inbound port",

"SecurityGroupIngress" :

[{ "IpProtocol" : "tcp", "FromPort" : "22", "ToPort" : "22", "CidrIp" : { "Ref" : "SSHLocation"} },

{ "IpProtocol" : "tcp", "FromPort" : "80", "ToPort" : "80", "CidrIp" : "0.0.0.0/0"}]

}

},

"EIP" : {

"Type" : "AWS::EC2::EIP",

"Properties" : {

"InstanceId" : { "Ref" : "Ec2Instance" }

}

},

"RootRole": {

"Type" : "AWS::IAM::Role",

"Properties" : {

"AssumeRolePolicyDocument": {

"Version" : "2012-10-17",

"Statement" : [ {

"Effect" : "Allow",

"Principal" : {

"Service" : [ "ec2.amazonaws.com" ]

},

"Action" : [ "sts:AssumeRole" ]

} ]

},

"Path" : "/"

}

},

"RolePolicies" : {

"Type" : "AWS::IAM::Policy",

"Properties" : {

"PolicyName" : "root",

"PolicyDocument" : {

"Version" : "2012-10-17",

"Statement" : [ {

"Effect" : "Allow",

"Action" : "kinesis:\*",

"Resource" : { "Fn::Join" : [ "", [ "arn:aws:kinesis:", { "Ref" : "AWS::Region" }, ":", { "Ref" : "AWS::AccountId" }, ":stream/", { "Ref" : "KinesisStream" } ]]}

}, {

"Effect" : "Allow",

"Action" : "dynamodb:\*",

"Resource" : { "Fn::Join" : [ "", [ "arn:aws:dynamodb:", { "Ref" : "AWS::Region" }, ":", { "Ref" : "AWS::AccountId" }, ":table/", { "Ref" : "KCLDynamoDBTable" } ]]}

}, {

"Effect" : "Allow",

"Action" : "dynamodb:\*",

"Resource" : { "Fn::Join" : [ "", [ "arn:aws:dynamodb:", { "Ref" : "AWS::Region" }, ":", { "Ref" : "AWS::AccountId" }, ":table/", { "Ref" : "CountsDynamoDBTable" } ]]}

}, {

"Effect" : "Allow",

"Action" : "cloudwatch:\*",

"Resource" : "\*"

} ]

},

"Roles" : [ { "Ref": "RootRole" } ]

}

},

"RootInstanceProfile" : {

"Type" : "AWS::IAM::InstanceProfile",

"Properties" : {

"Path" : "/",

"Roles" : [ { "Ref": "RootRole" } ]

}

},

"Ec2Instance": {

"Type" : "AWS::EC2::Instance",

"Metadata" : {

"AWS::CloudFormation::Init" : {

"config" : {

"packages" : {

"yum" : {

"java-1.7.0-openjdk" : []

}

},

"files" : {

"/var/kinesis-data-vis-sample-app/watchdog.sh" : {

"content" : {"Fn::Join" : ["", [

"#!/bin/bash\n",

"if ! ps aux | grep HttpReferrerCounterApplication | grep -v grep ; then\n",

" # Launch the Kinesis application for counting HTTP referrer pairs\n",

" java -cp /var/kinesis-data-vis-sample-app/lib/\\\* com.amazonaws.services.kinesis.samples.datavis.HttpReferrerCounterApplication ", { "Ref" : "KCLDynamoDBTable" }, " ", { "Ref" : "KinesisStream" }, " ", { "Ref" : "CountsDynamoDBTable" }, " ", { "Ref" : "AWS::Region" }, " &>> /home/ec2-user/kinesis-data-vis-sample-app-kcl.log &\n",

"fi\n",

"if ! ps aux | grep HttpReferrerStreamWriter | grep -v grep ; then\n",

" # Launch our Kinesis stream writer to fill our stream with generated HTTP (resource, referrer) pairs.\n",

" # This will create a writer with 5 threads to send records indefinitely.\n",

" java -cp /var/kinesis-data-vis-sample-app/lib/\\\* com.amazonaws.services.kinesis.samples.datavis.HttpReferrerStreamWriter 5 ", { "Ref" : "KinesisStream" }, " ", { "Ref" : "AWS::Region" }, " &>> /home/ec2-user/kinesis-data-vis-sample-app-publisher.log &\n",

"fi\n",

"if ! ps aux | grep WebServer | grep -v grep ; then\n",

" # Launch the webserver\n",

" java -cp /var/kinesis-data-vis-sample-app/lib/\\\* com.amazonaws.services.kinesis.samples.datavis.WebServer 80 /var/kinesis-data-vis-sample-app/wwwroot ", { "Ref" : "CountsDynamoDBTable" }, " ", { "Ref" : "AWS::Region" }, " &>> /home/ec2-user/kinesis-data-vis-sample-app-www.log &\n",

"fi\n"

]]},

"mode" : "000755",

"owner" : "ec2-user",

"group" : "ec2-user"

},

"/var/kinesis-data-vis-sample-app/crontask" : {

"content" : {"Fn::Join" : ["", [

"\* \* \* \* \* bash /var/kinesis-data-vis-sample-app/watchdog.sh\n"

]]},

"mode" : "000644",

"owner" : "ec2-user",

"group" : "ec2-user"

}

},

"sources": {

"/var/kinesis-data-vis-sample-app" : { "Ref" : "ApplicationArchive" }

}

}

}

},

"Properties" : {

"KeyName" : { "Fn::If" : [ "UseEC2KeyName", { "Ref" : "KeyName" }, { "Ref" : "AWS::NoValue" } ]},

"ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" : "AWS::Region" },

{ "Fn::FindInMap" : [ "AWSInstanceType2Arch", { "Ref" : "InstanceType" },

"Arch" ] } ] },

"InstanceType" : { "Ref" : "InstanceType" },

"SecurityGroups" : [{ "Ref" : "Ec2SecurityGroup" }],

"IamInstanceProfile": { "Ref": "RootInstanceProfile" },

"UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [

"#!/bin/bash\n",

"yum update -y aws-cfn-bootstrap\n",

"/opt/aws/bin/cfn-init -s ", { "Ref" : "AWS::StackId" }, " -r Ec2Instance ",

" --region ", { "Ref" : "AWS::Region" }, "\n",

"# Register watchdog script with cron\n",

"crontab /var/kinesis-data-vis-sample-app/crontask\n",

"# Launch watchdog script immediately so if it fails this stack fails to start\n",

"/var/kinesis-data-vis-sample-app/watchdog.sh\n",

"/opt/aws/bin/cfn-signal -e $? '", { "Ref" : "WaitHandle" }, "'\n"

]]}}

}

},

"WaitHandle" : {

"Type" : "AWS::CloudFormation::WaitConditionHandle"

},

"WaitCondition" : {

"Type" : "AWS::CloudFormation::WaitCondition",

"DependsOn" : "Ec2Instance",

"Properties" : {

"Handle" : {"Ref" : "WaitHandle"},

"Timeout" : "600"

}

}

},

"Outputs" : {

"URL" : {

"Description" : "URL to the sample application's visualization",

"Value" : { "Fn::Join" : [ "", [ "http://", { "Fn::GetAtt" : [ "Ec2Instance", "PublicDnsName" ] }]]}

},

"InstanceId" : {

"Description" : "InstanceId of the newly created EC2 instance",

"Value" : { "Ref" : "Ec2Instance" }

},

"AZ" : {

"Description" : "Availability Zone of the newly created EC2 instance",

"Value" : { "Fn::GetAtt" : [ "Ec2Instance", "AvailabilityZone" ] }

},

"StreamName" : {

"Description" : "The name of the Kinesis Stream. This was autogenerated by the Kinesis Resource named 'KinesisStream'",

"Value" : { "Ref" : "KinesisStream" }

},

"ApplicationName" : {

"Description" : "The name of the Kinesis Client Application. This was autogenerated by the DynamoDB Resource named 'KCLDynamoDBTable'",

"Value" : { "Ref" : "KCLDynamoDBTable" }

},

"CountsTable" : {

"Description" : "The name of the DynamoDB table where counts are persisted. This was autogenerated by the DynamoDB Resource named 'CountsDynamoDBTable'",

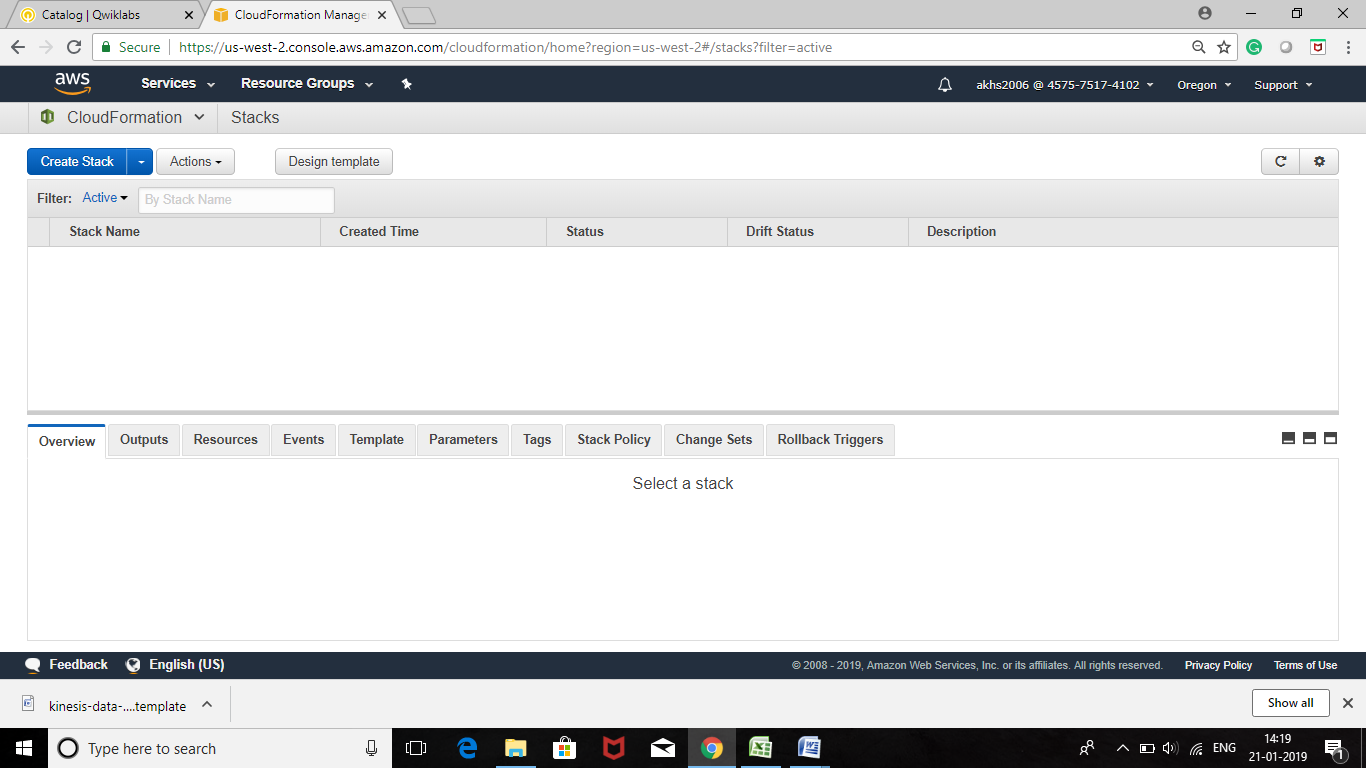
"Value" : { "Ref" : "CountsDynamoDBTable" }

}

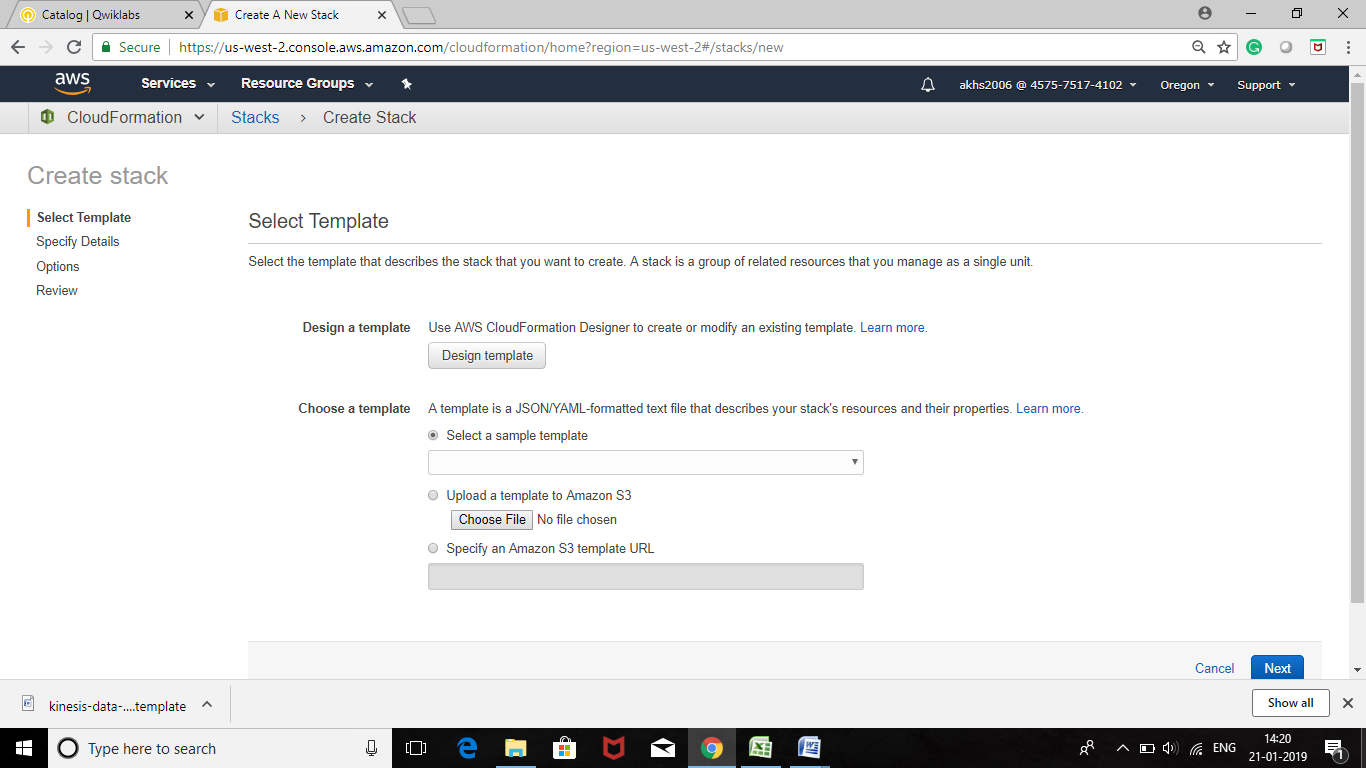
}

}

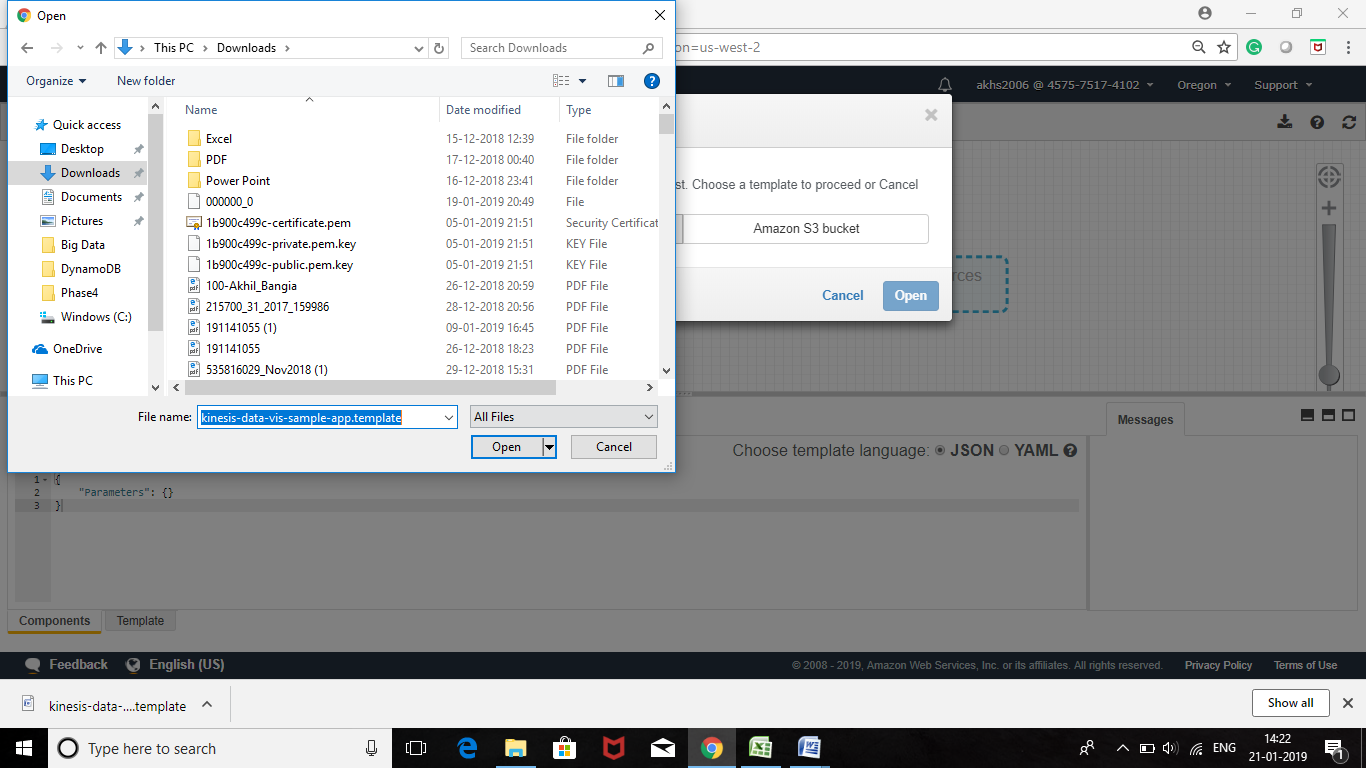
* Run the Template in Cloud Formation
* Cloud Formation will initiate an EC2 instance
* Copy the Public IP of the EC2 instance
* Open a new browser window and paste the Public IP of EC2
* You will see the stream of Analytics run by AWS Kinesis

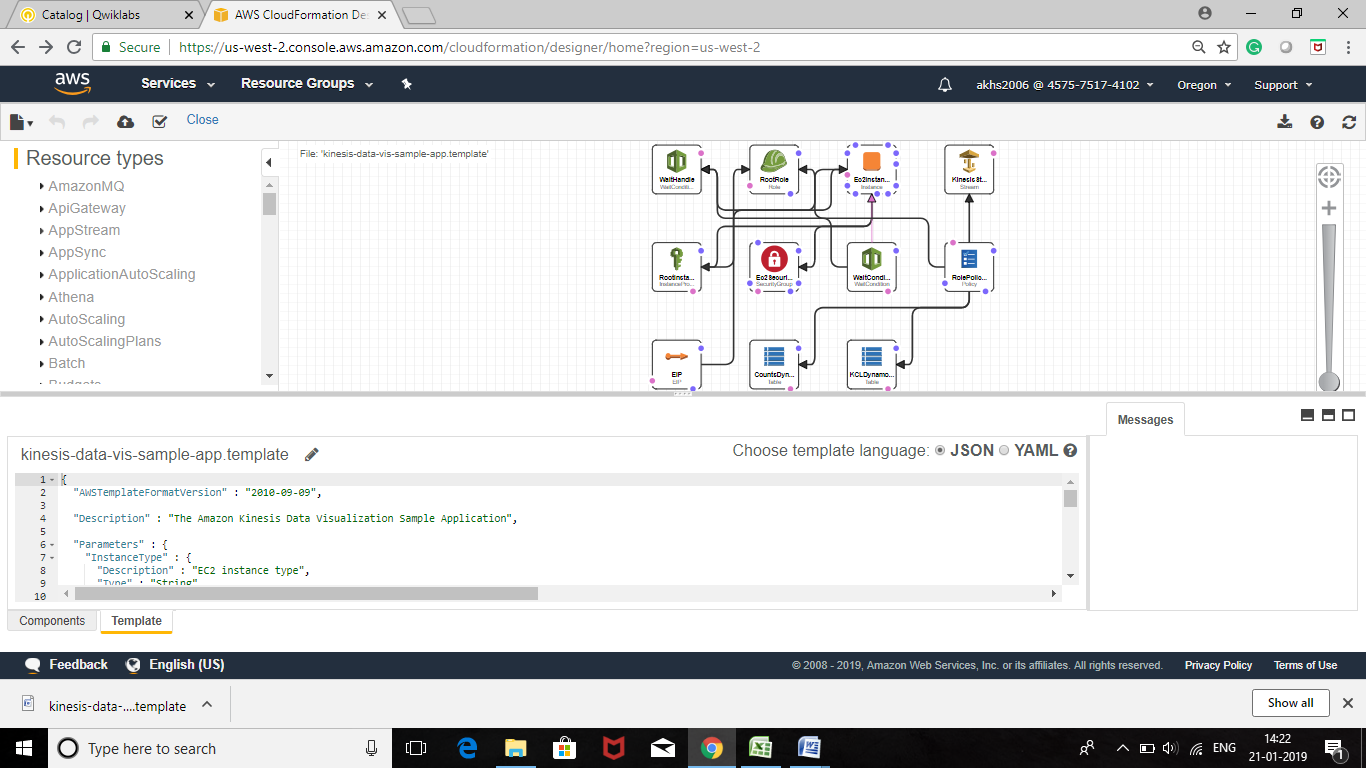


Click on Create Stack



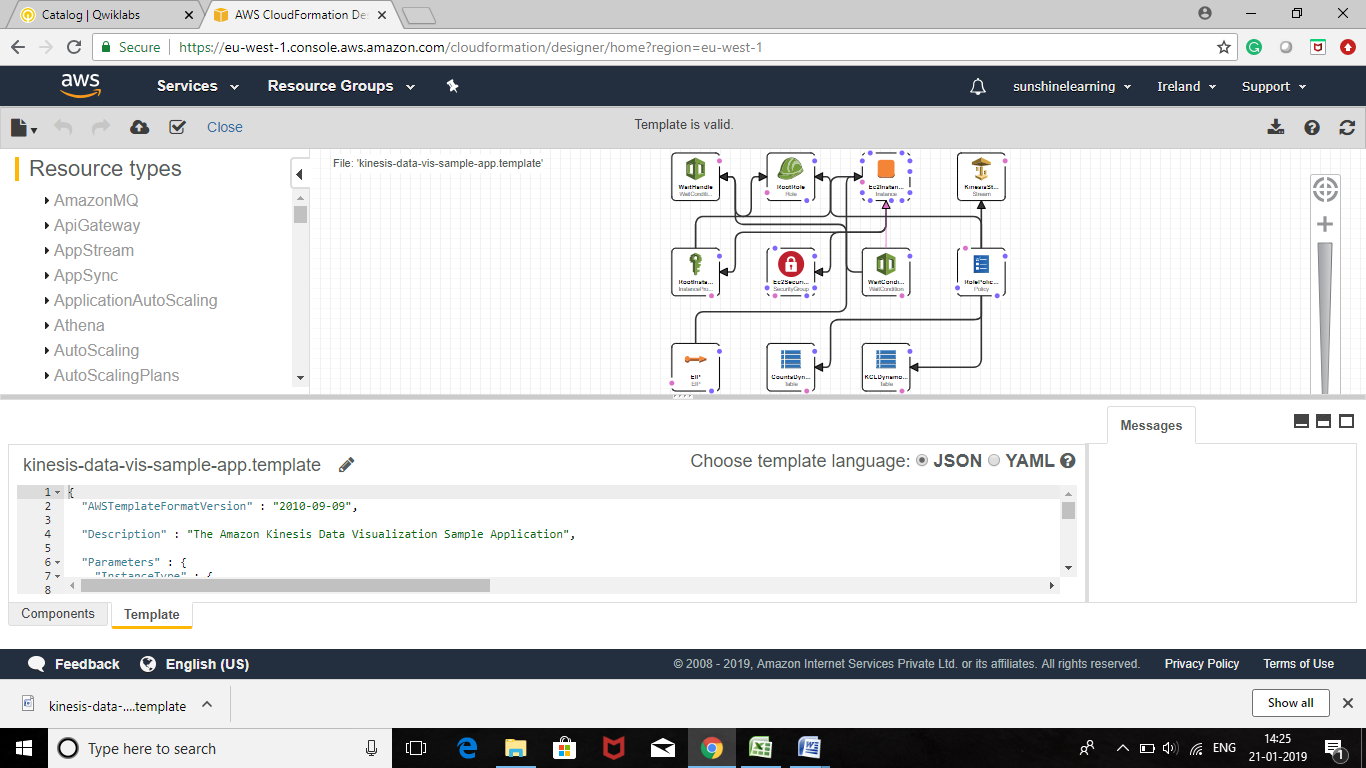
Click on Design Template and Open a Saved Template



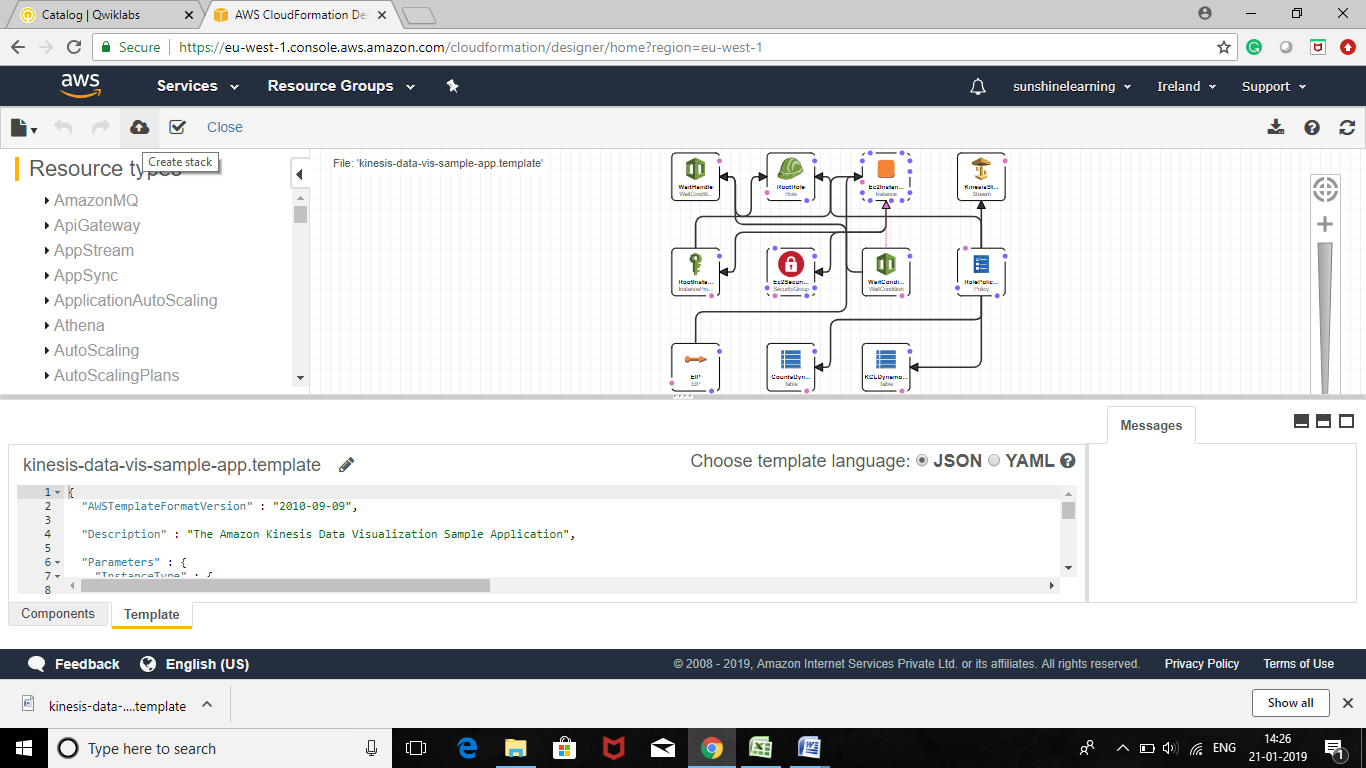


Click on Validate Template





Now click on Create Stack



Click on Next, this will create a Stack with the sample template for Kinesis

