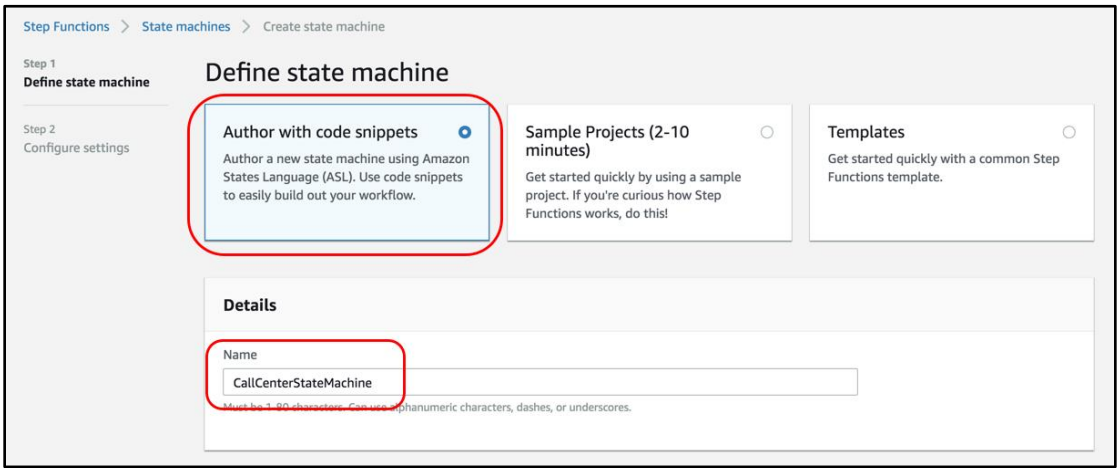
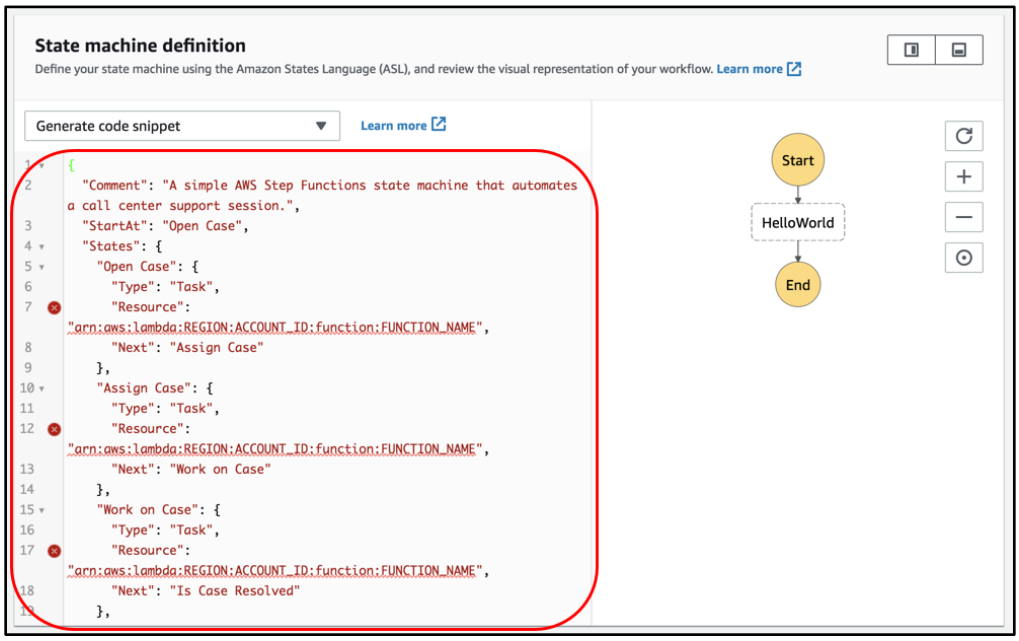
Practical 9:- Implementing a Serverless Architecture with AWS Managed Services

Step 1. Create a State Machine & Serverless Workflow

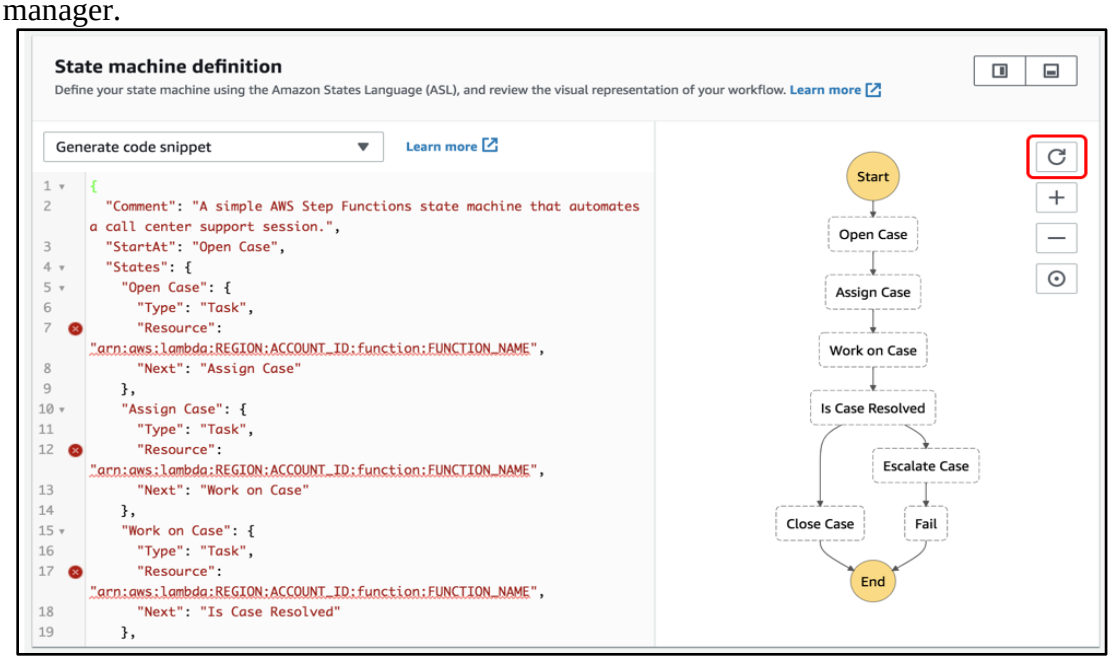
a. Open the AWS Step Functions console. Select **Author with code snippets**. In the **Name** text box, type CallCenterStateMachine.



b. Replace the contents of the State machine definition window with the Amazon States Language (ASL) state machine definition below. Amazon States Language is a JSON-based, structured language used to define your state machine.



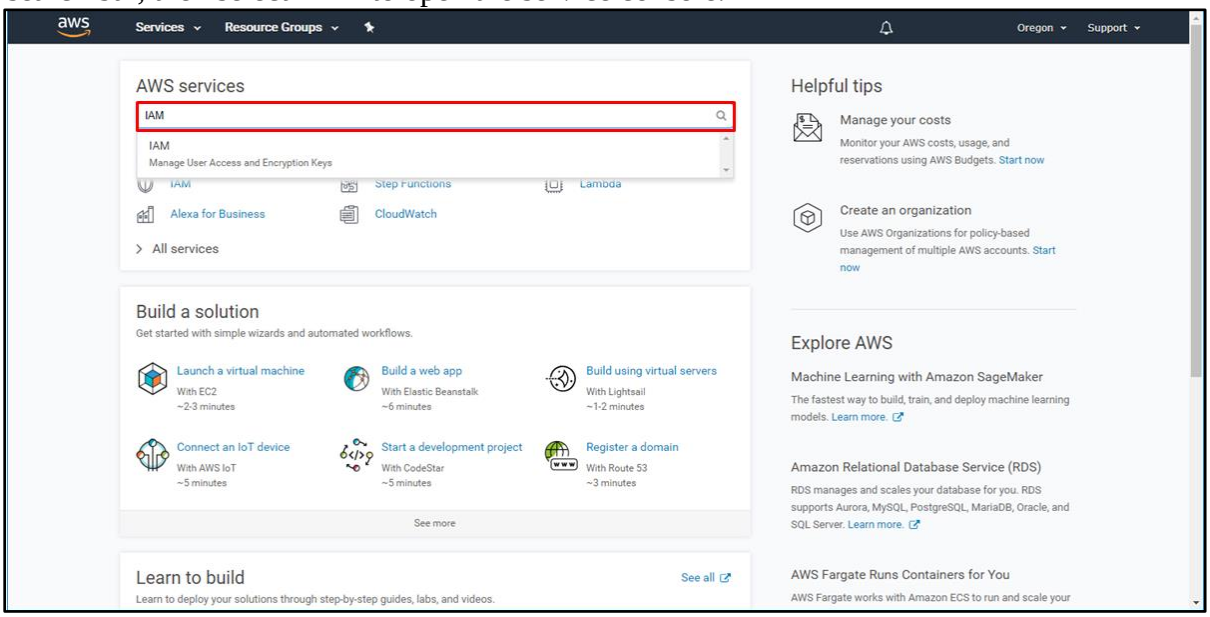
c. Click the refresh button to show the ASL state machine definition as a visual workflow. In our scenario, you can easily verify that the process is described correctly by reviewing the visual workflow with the call center manager.



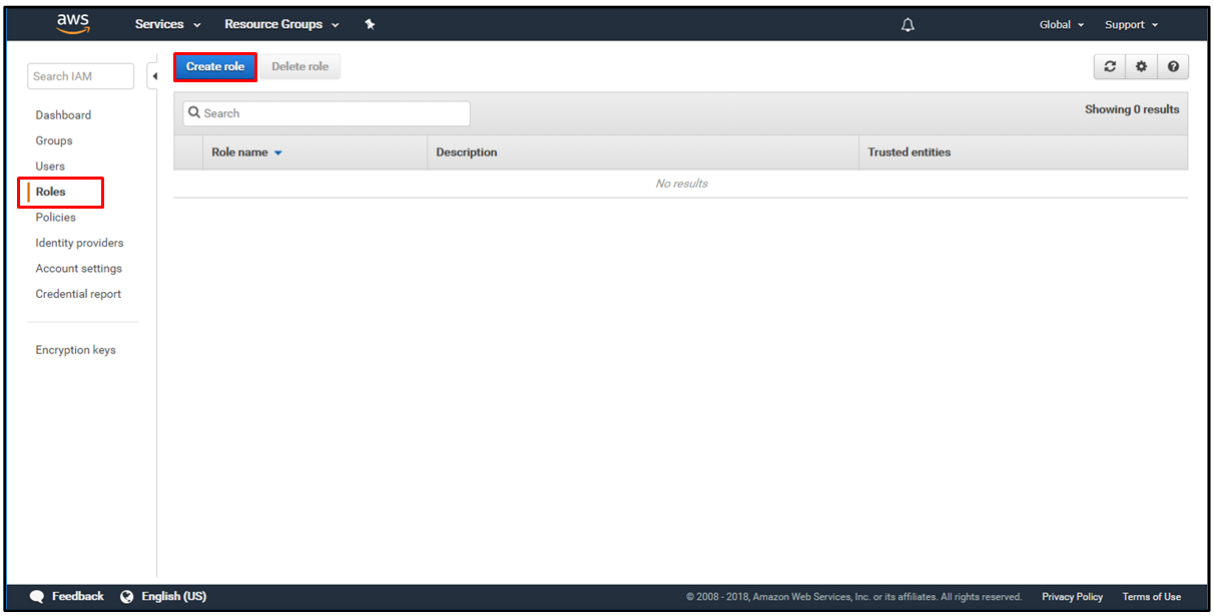
d. Click **Next.**

Step 2. Create an AWS Identity and Access Management (IAM) Role

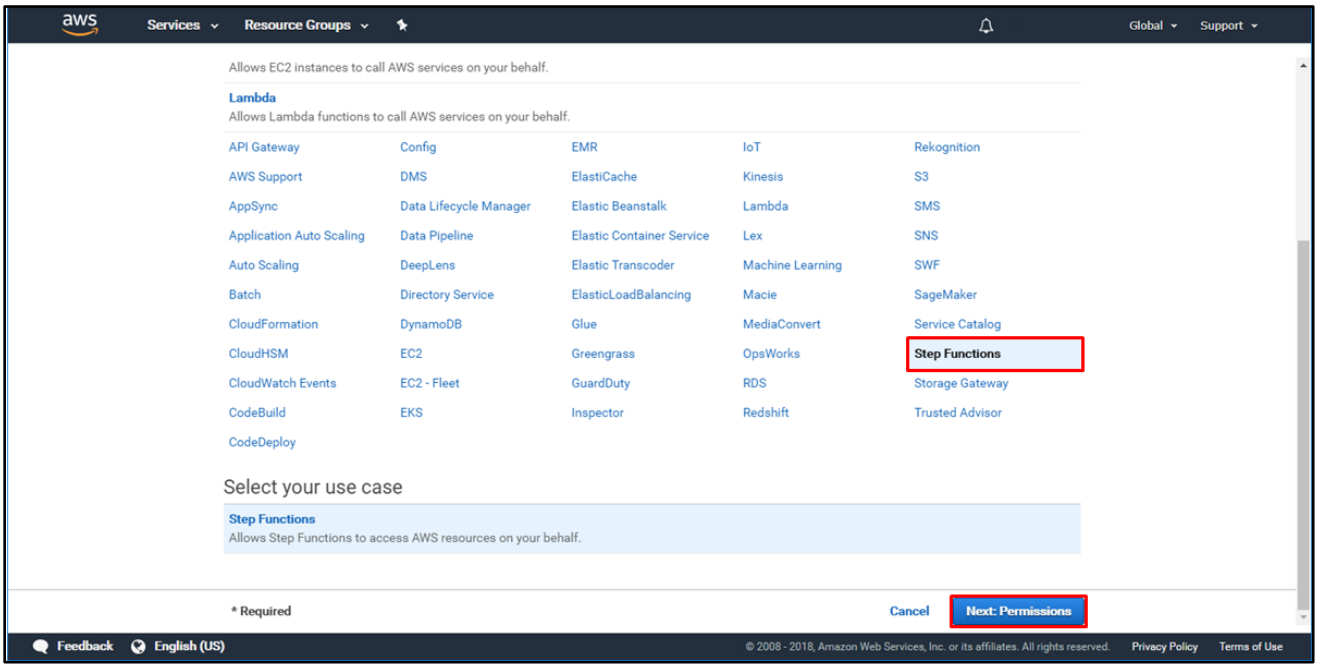
a. In another browser window, open the **AWS Management Console**. When the screen loads, type **IAM** in the search bar, then select **IAM** to open the service console.



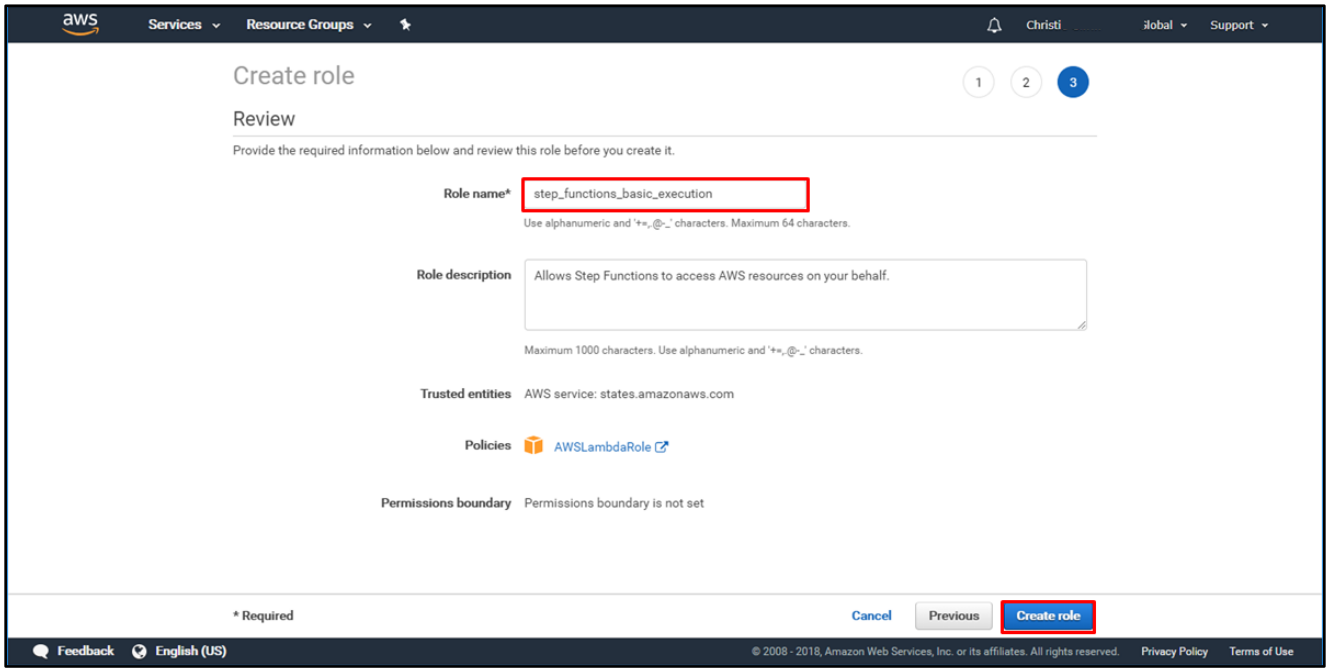
b. Click **Roles** and then click **Create Role**.



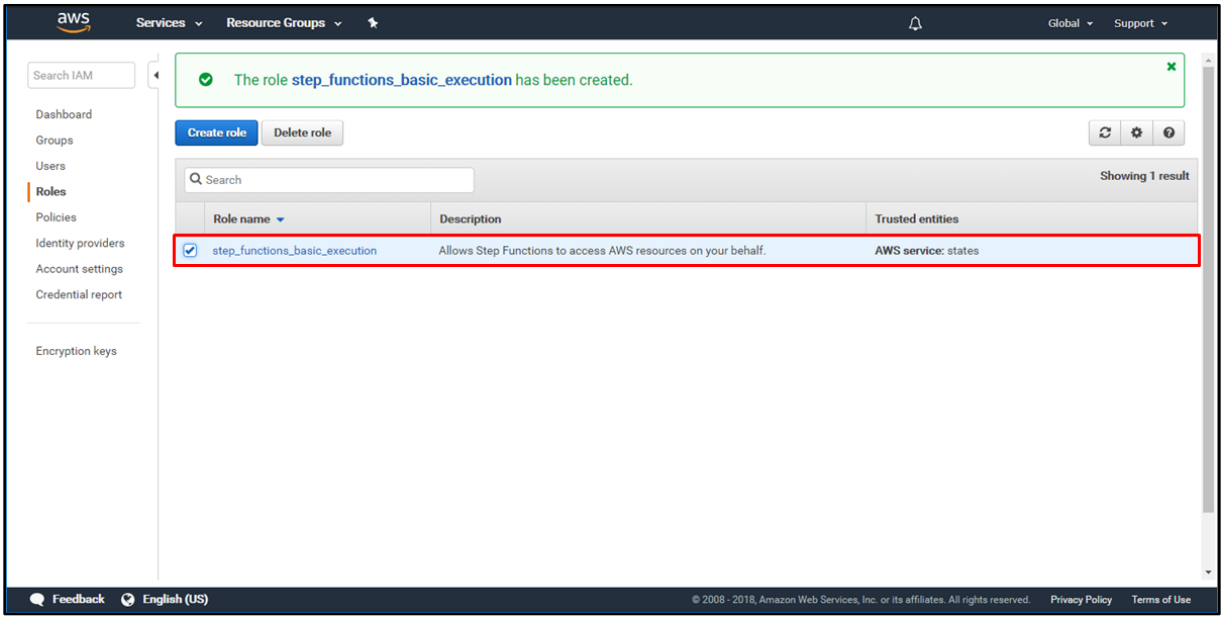
c. On the **Create Role**s screen, **leave AWS Service** selected, select **Step Functions** then click **Next: Permissions**. On the next screen, click **Next: Review**.



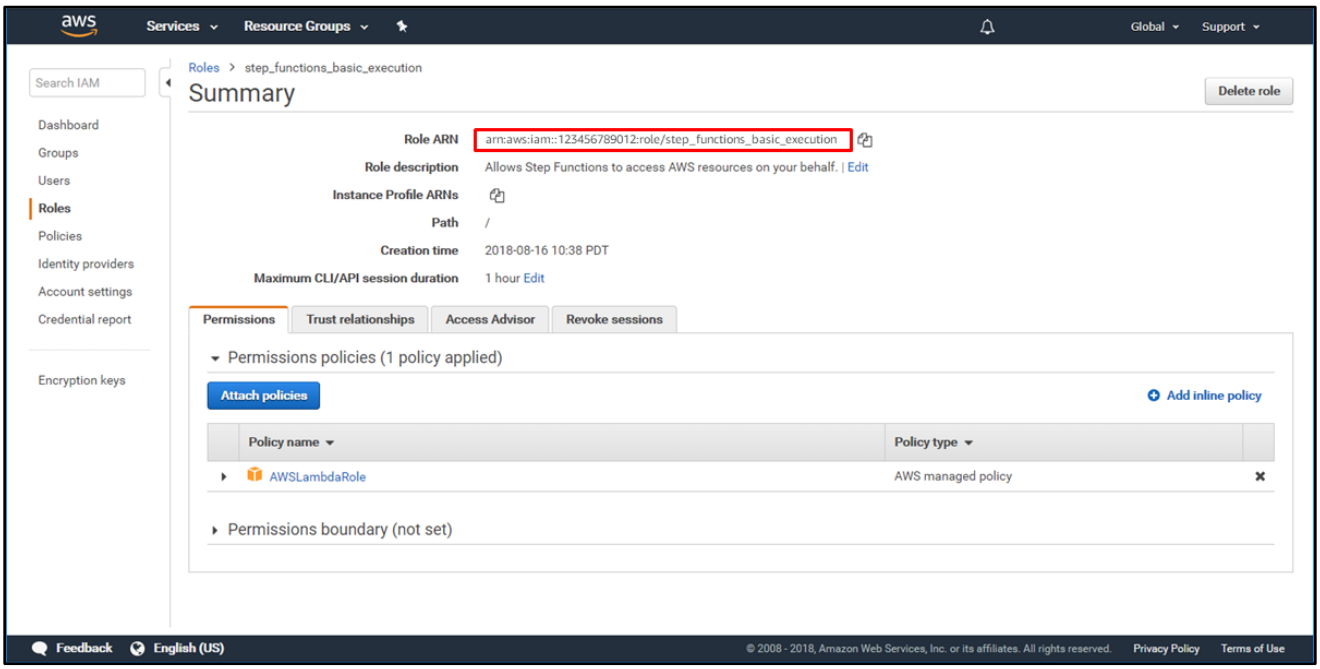
d. Enter **Role name** as step_functions_basic_execution and choose **Create role**. On the next screen, click **Next: Review.**



e. Your role is created and appears in the list. Select the name of your role to view it.

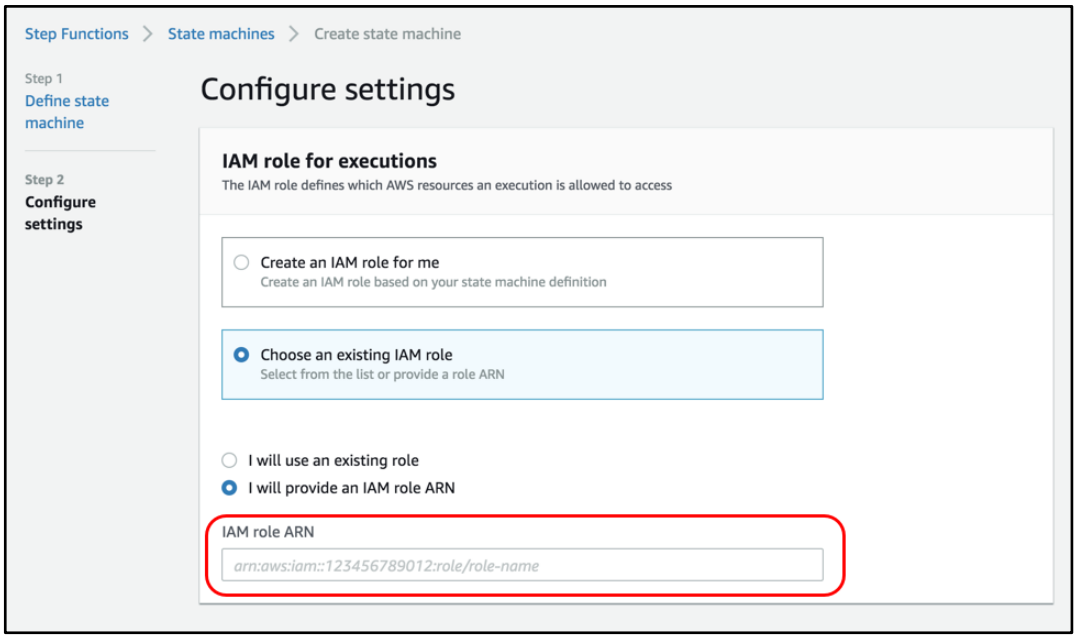


f. Copy the **Role ARN** on the next screen.



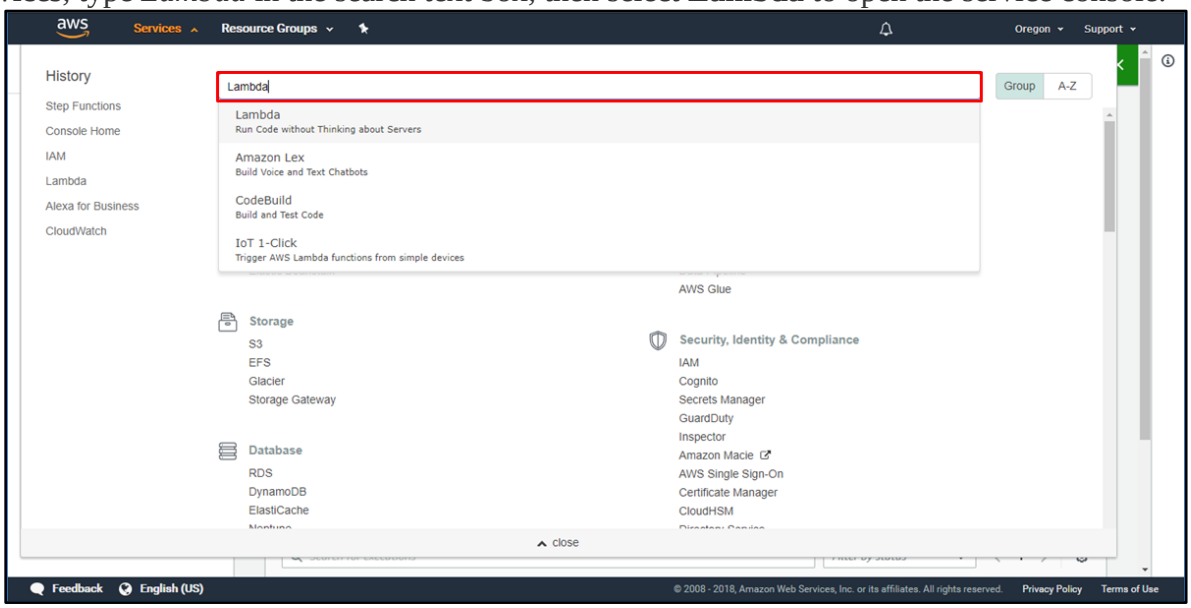
Step 3. Add the IAM Role to the State Machine

a. Select the browser tab with the Step Functions console.
b. Paste the ARN you copied in the **IAM role ARN** text box.
c. Click **Create State Machine**

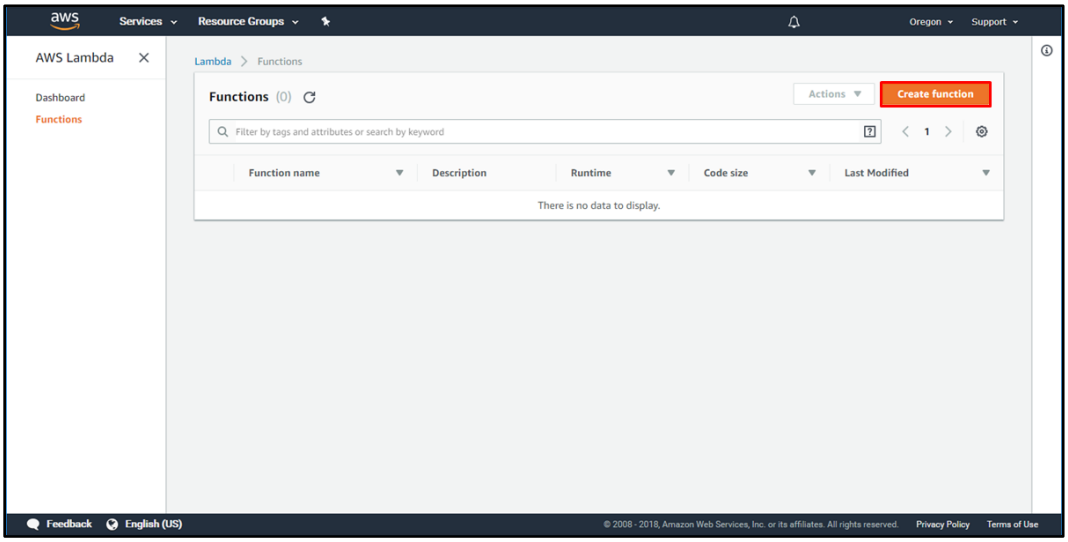


Step 4. Create your AWS Lambda Functions

a. Click **Services**, type *Lambda* in the search text box, then select **Lambda** to open the service console.



b. Click **Create function**.



c. Select **Author from scratch**.

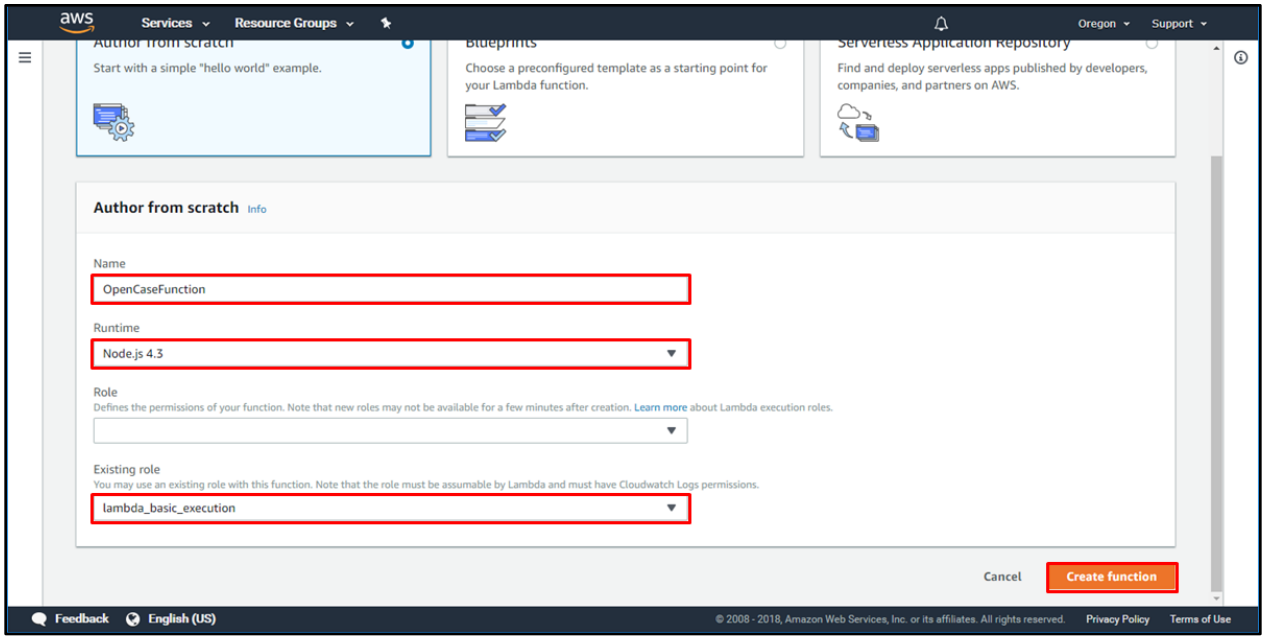
d. Configure your first Lambda function with these settings:

Name – OpenCaseFunction.
Runtime – Node.js 4.3.
Role – Create custom role.

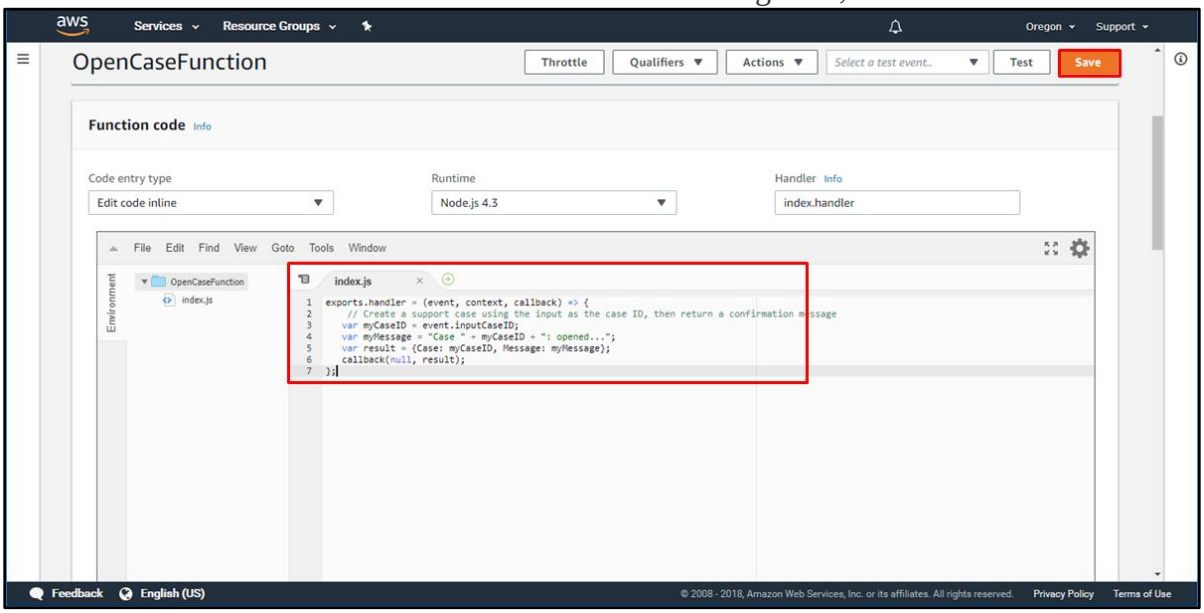
A new IAM window appears.

e. For the **Role name**, keep lambda_basic_execution and click **Allow**.
You are automatically returned to the Lambda console.

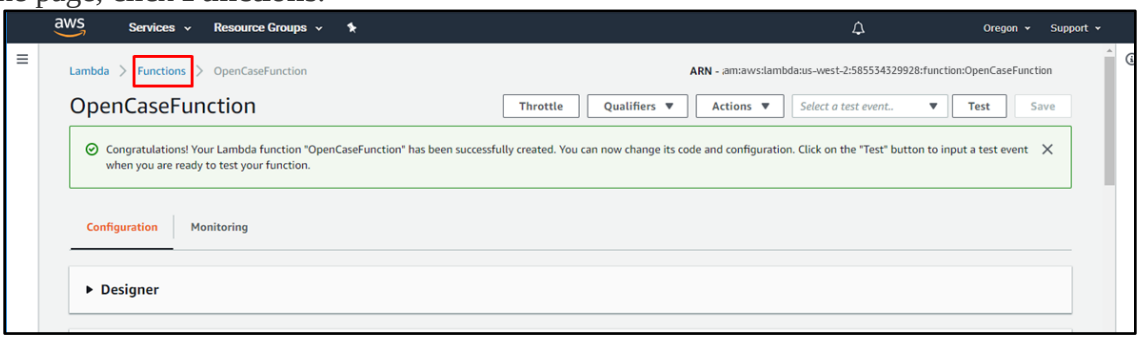
f. Click **Create function**.



g. Replace the contents of the **Function code** window with the following code, and then click **Save**



h. At the top of the page, click **Functions**.



i. Repeat steps 4b-4d to create 4 more Lambda functions, using the lambda_basic_execution IAM role you created in step 4c.

Define *AssignCaseFunction* as:

```
exports.handler = (event, context, callback) => {
  // Assign the support case and update the status message
  var myCaseID = event.Case;
  var myMessage = event.Message + "assigned...";
  var result = {Case: myCaseID, Message: myMessage};
  callback(null, result);
};
```

Define *WorkOnCaseFunction* as:

```
exports.handler = (event, context, callback) => {
  // Generate a random number to determine whether the support case has been resolved, then return that value along with the updated message.
  var min = 0;
  var max = 1;
  var myCaseStatus = Math.floor(Math.random() * (max - min + 1)) + min;
  var myCaseID = event.Case;
  var myMessage = event.Message;
  if (myCaseStatus == 1) {
    // Support case has been resolved
    myMessage = myMessage + "resolved...";
  } else if (myCaseStatus == 0) {
    // Support case is still open
    myMessage = myMessage + "unresolved...";
  }
  var result = {Case: myCaseID, Status : myCaseStatus, Message: myMessage};
  callback(null, result);
};
```

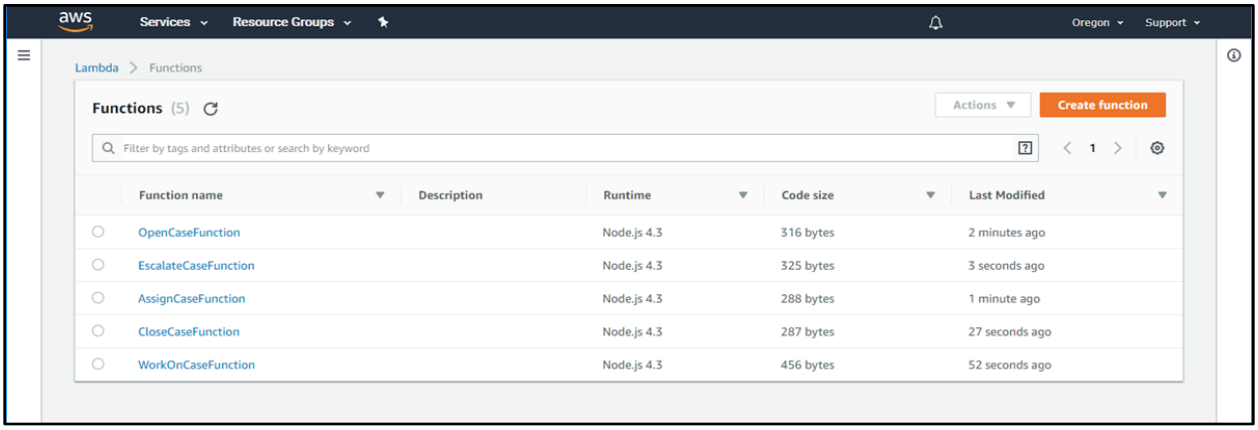
Define *CloseCaseFunction* as:

```
exports.handler = (event, context, callback) => {
  // Close the support case
  var myCaseStatus = event.Status;
  var myCaseID = event.Case;
  var myMessage = event.Message + "closed.";
  var result = {Case: myCaseID, Status : myCaseStatus, Message: myMessage};
  callback(null, result);
};
```

Define *EscalateCaseFunction* as:

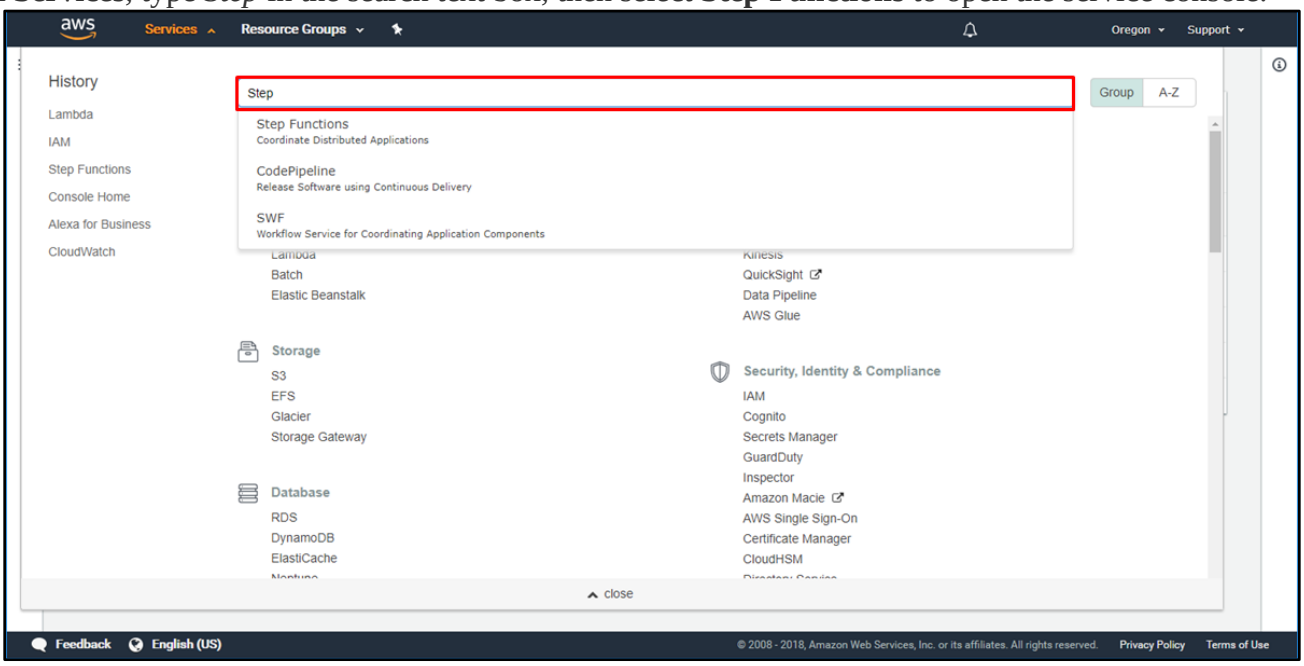
```
exports.handler = (event, context, callback) => {
  // Escalate the support case
  var myCaseID = event.Case;
  var myCaseStatus = event.Status;
  var myMessage = event.Message + "escalating.";
  var result = {Case: myCaseID, Status : myCaseStatus, Message: myMessage};
  callback(null, result);
};
```

When complete, you should have 5 Lambda functions.

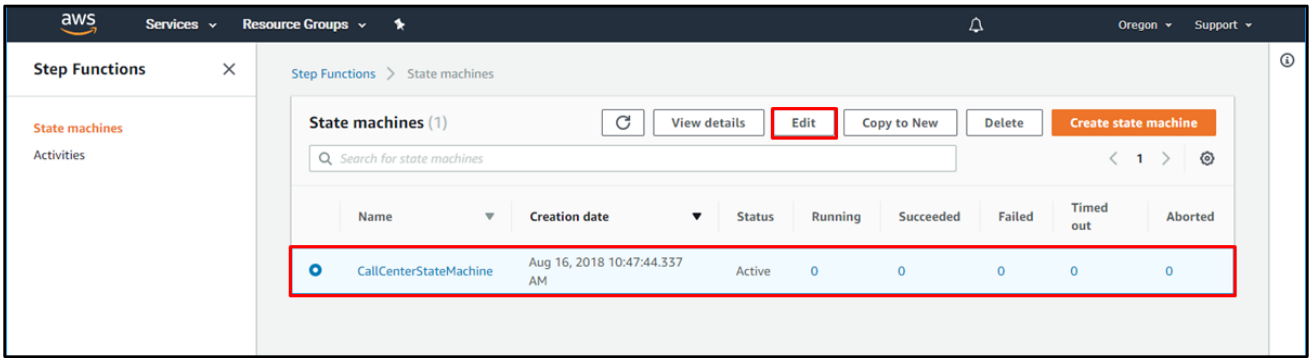


Step 5. Populate your Workflow

a. Click **Services**, type *Step* in the search text box, then select **Step Functions** to open the service console.

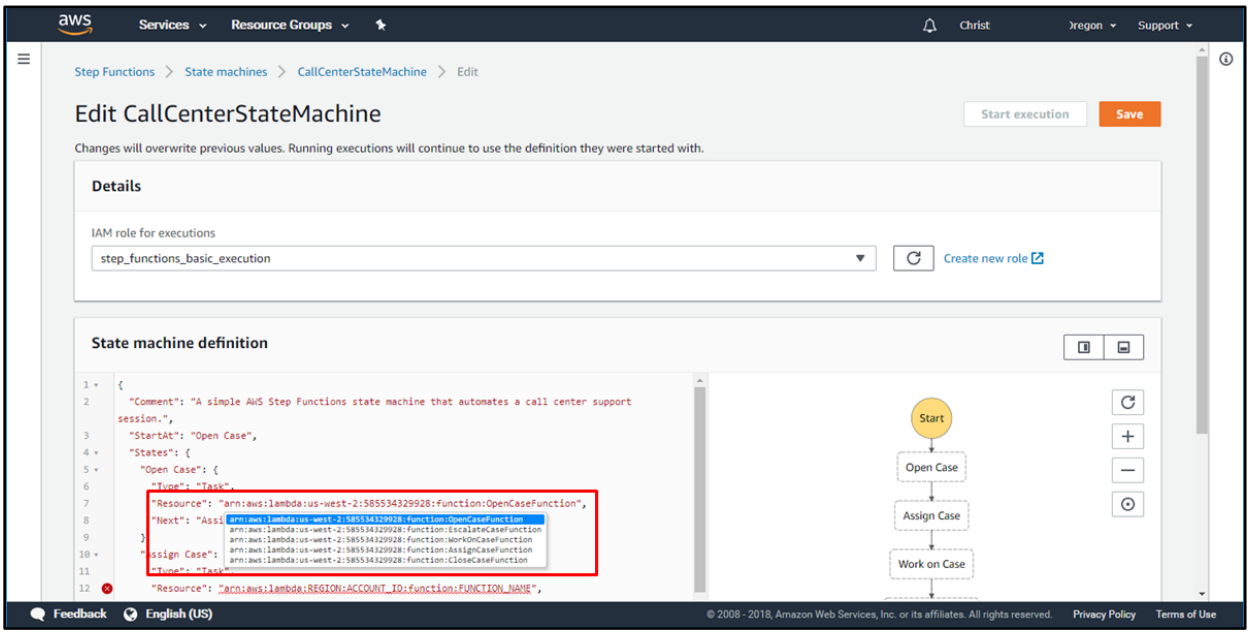


b. On the **State machines screen**, select your **CallCenterStateMachine** and click **Edit**.

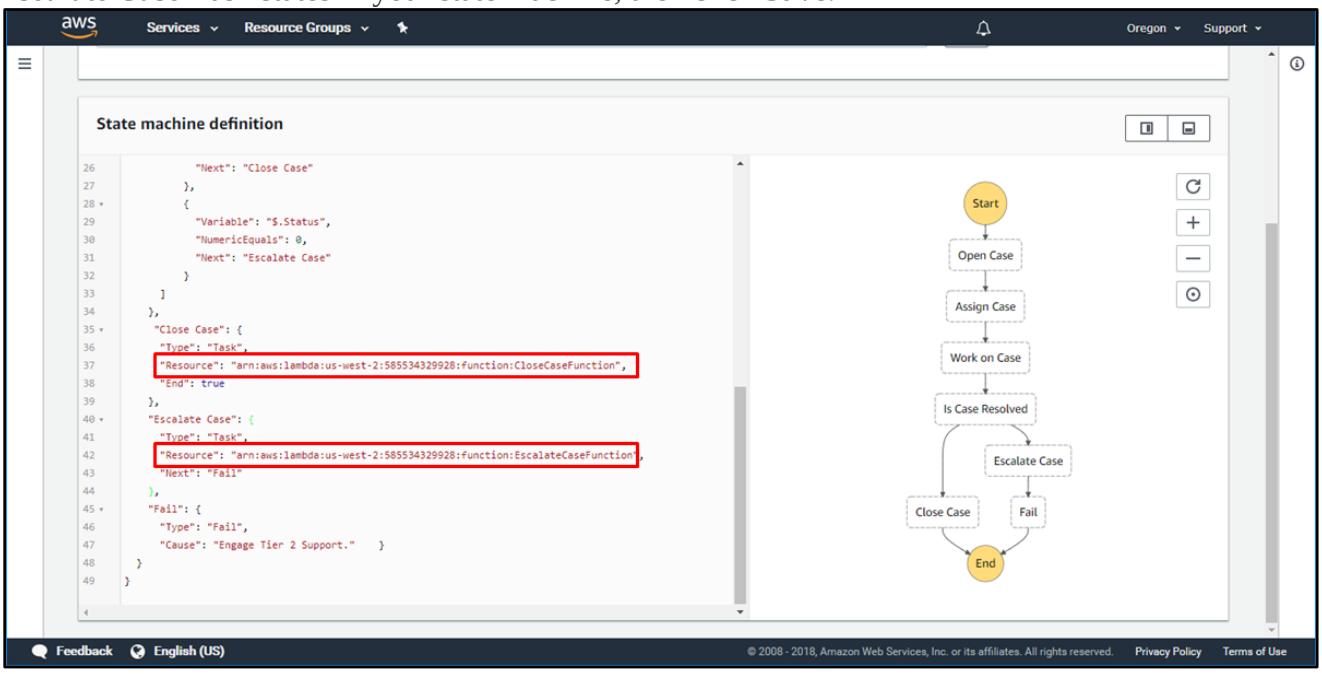


c. In the **State machine definition** section, find the line below the *Open Case* state which starts with *Resource*.

Replace the ARN with the ARN of your **OpenCaseFunction.**

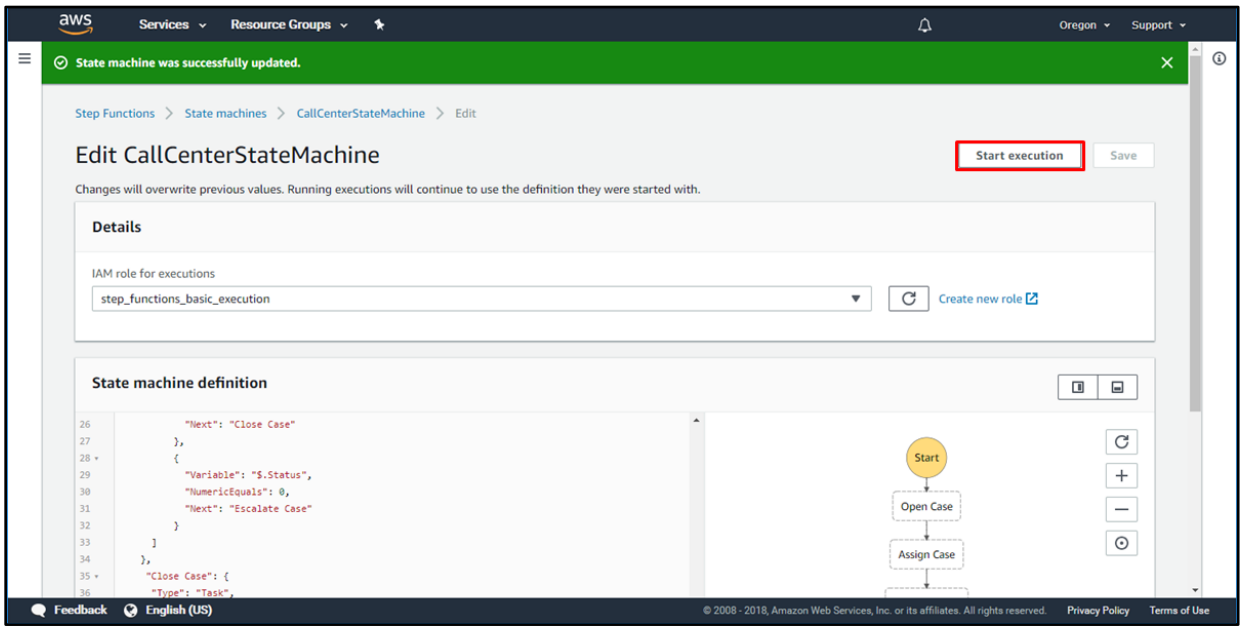


d. Repeat the previous step to update the Lambda function ARNs for the ***Assign Case, Work on Case***, ***Close Case***, and ***Escalate Case*** Task states in your state machine, then click **Save**.

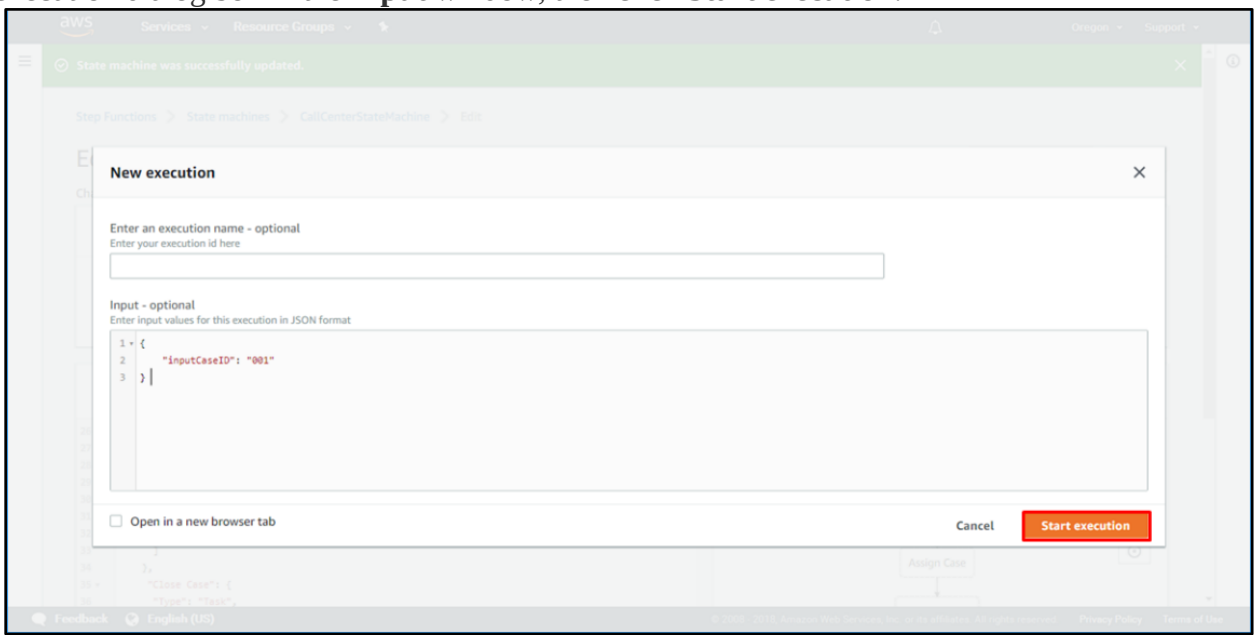


Step 6. Execute your Workflow

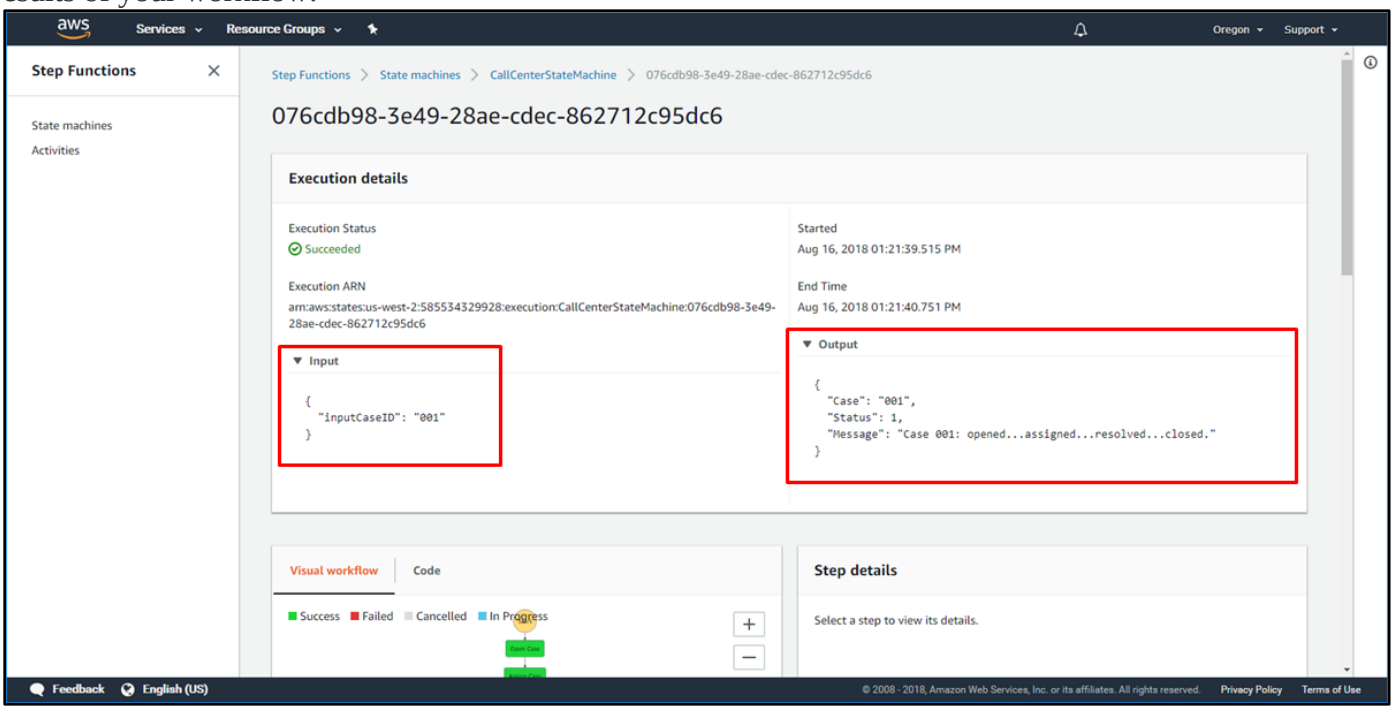
a. Click **Start execution**.



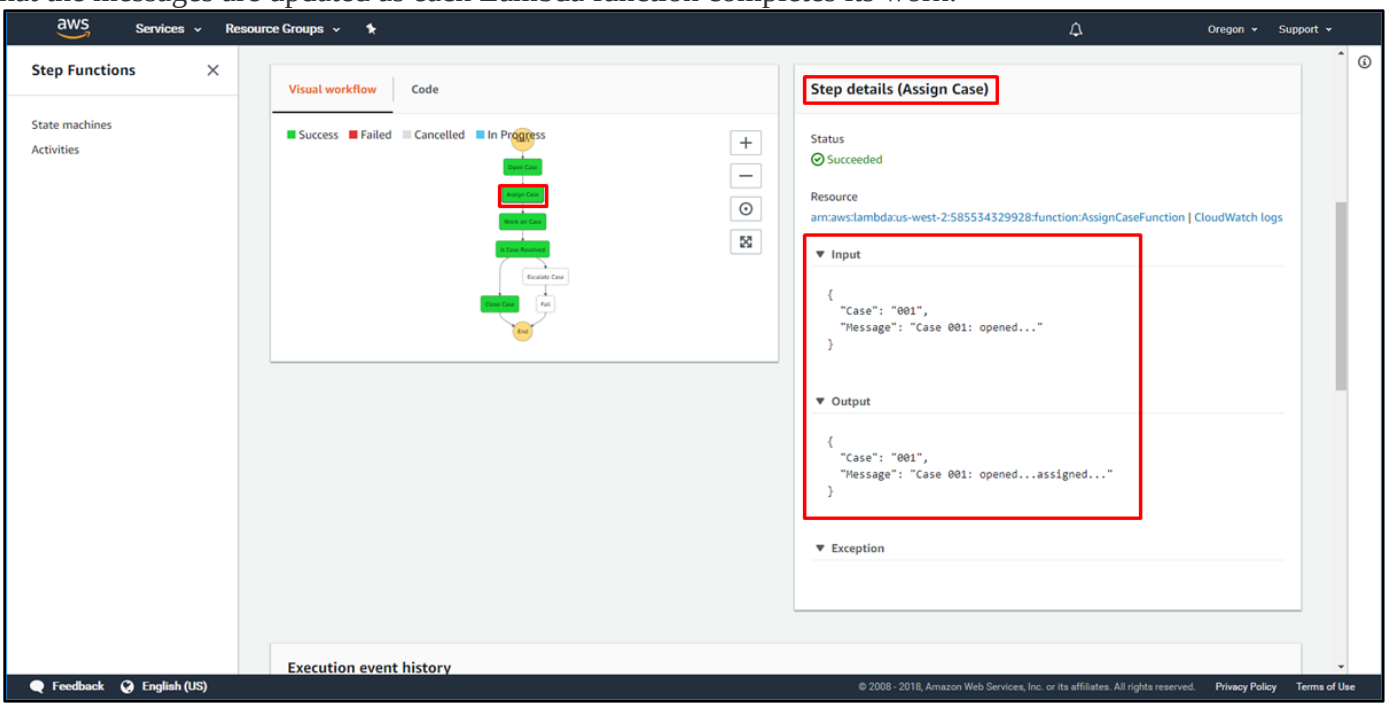
b. A New execution dialog box appears. To supply an ID for your support case, enter the content from below in the New execution dialog box in the **Input** window, then click **Start execution**.



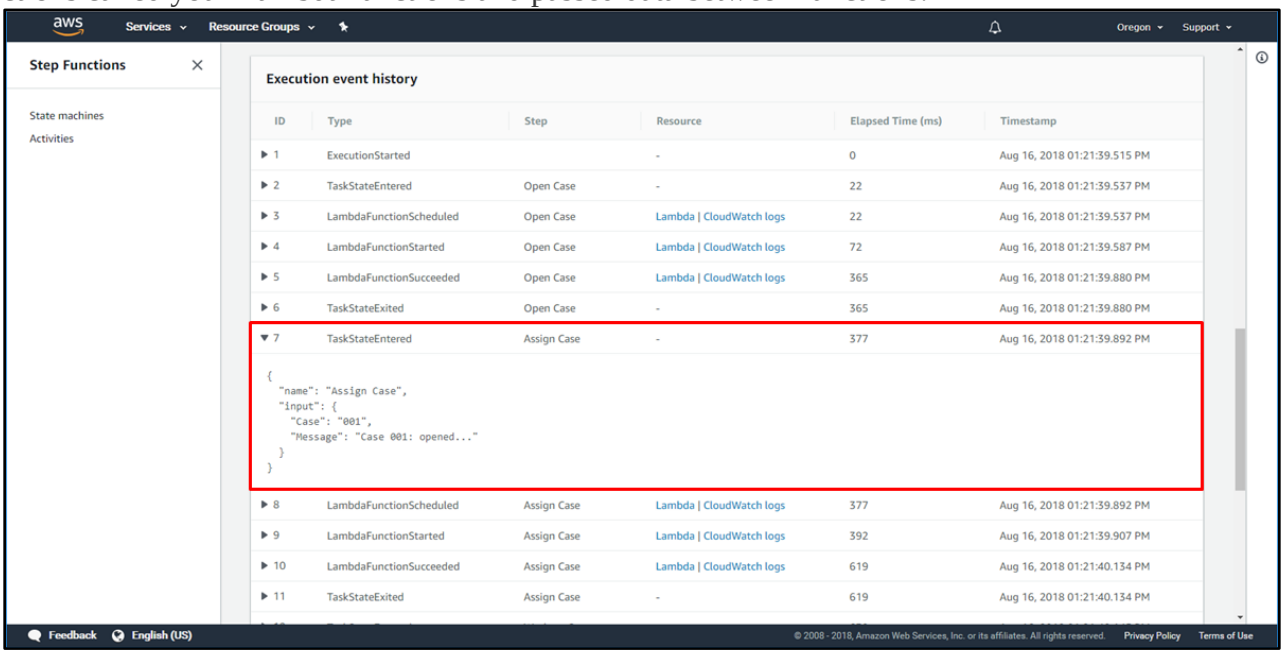
c. As your workflow executes, each step will change color in the **Visual workflow** pane. Wait a few seconds for the execution to complete. Then, in the **Execution details** pane, lick **Input** and **Output** to view the inputs and results of your workflow.



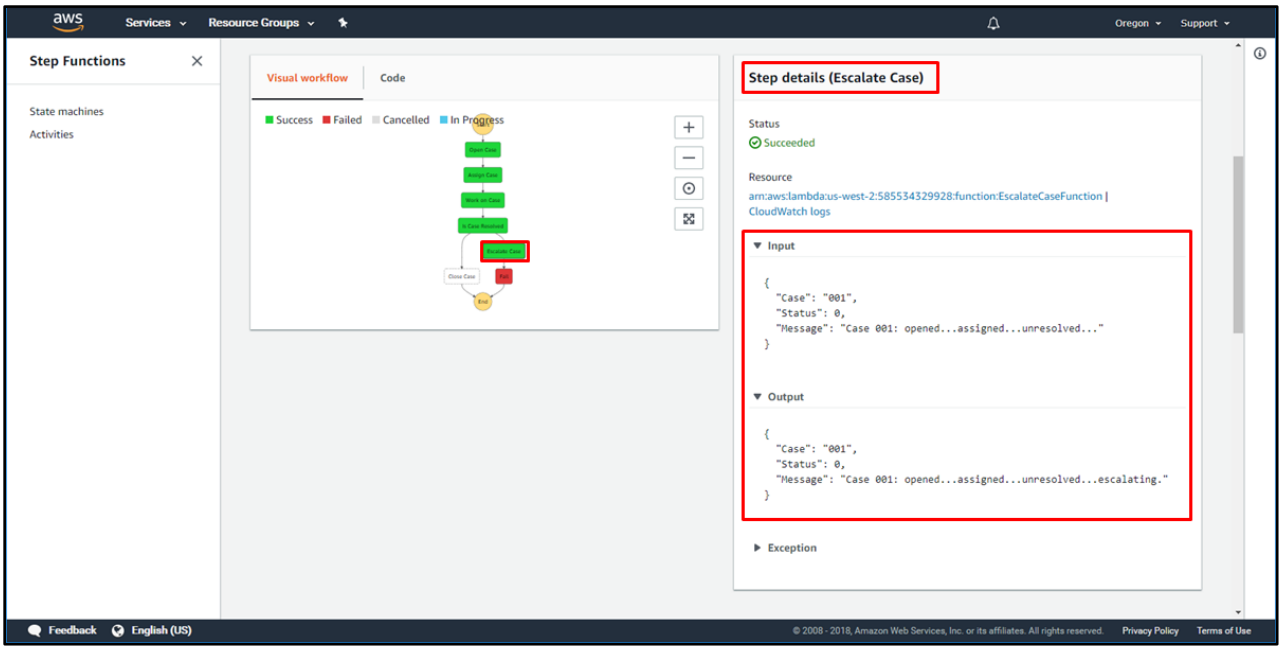
d. Step Functions lets you inspect each step of your workflow execution, including the inputs and outputs of each state. Click on each task in your workflow and expand the **Input** and **Output** fields under Step details. You can see that the case ID you inputted into your state machine is passed from each step to the next, and that the messages are updated as each Lambda function completes its work.



e. Scroll down to the **Execution event history** section. Click through each step of execution to see how Step Functions called your Lambda functions and passed data between functions.

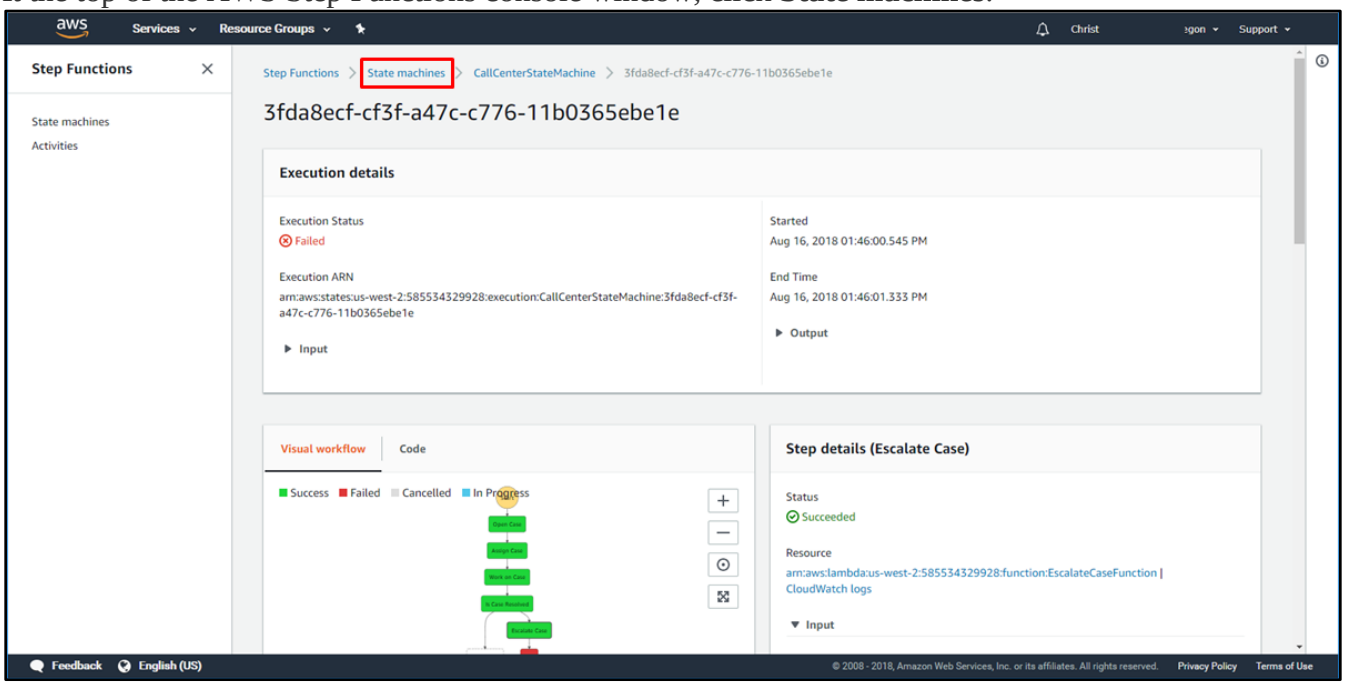


f. Depending on the output of your **WorkOnCaseFunction**, your workflow may have ended by resolving the support case and closing the ticket, or escalating the ticket to the next tier of support. You can re-run the execution a few more times to observe this different behavior. This image shows an execution of the workflow where the support case was escalated, causing the workflow to exit with a Fail state.

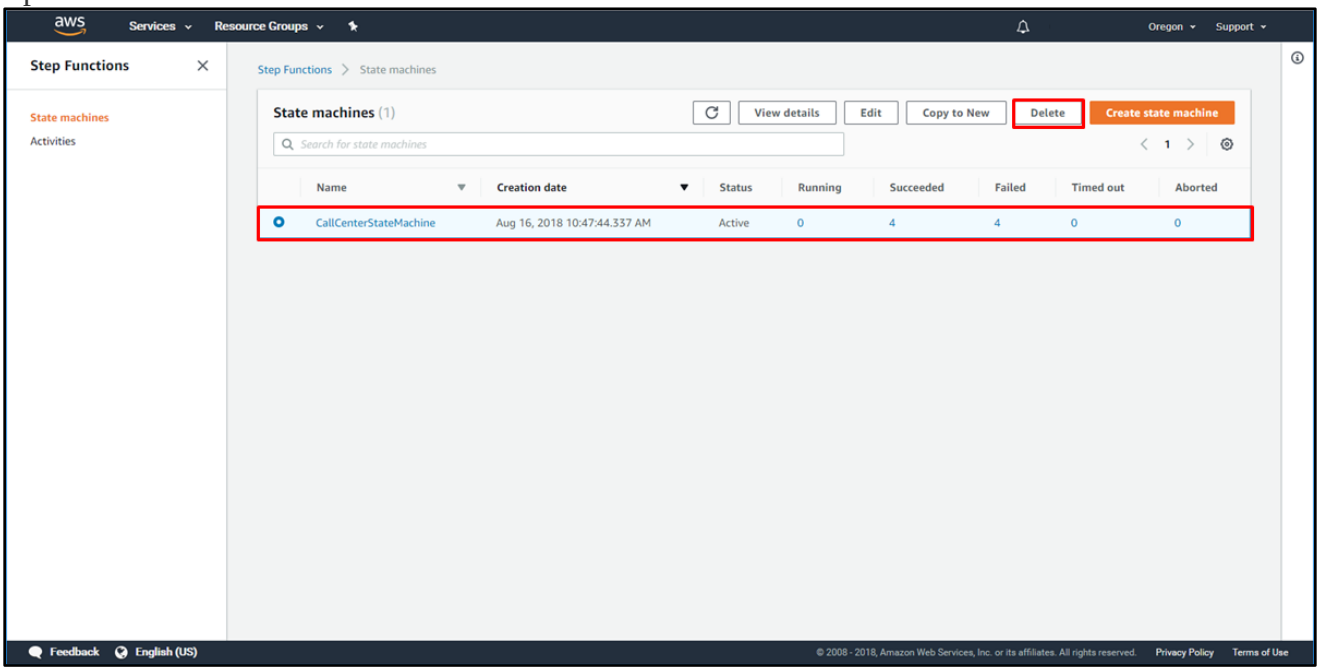


Step 7. Terminate your Resources

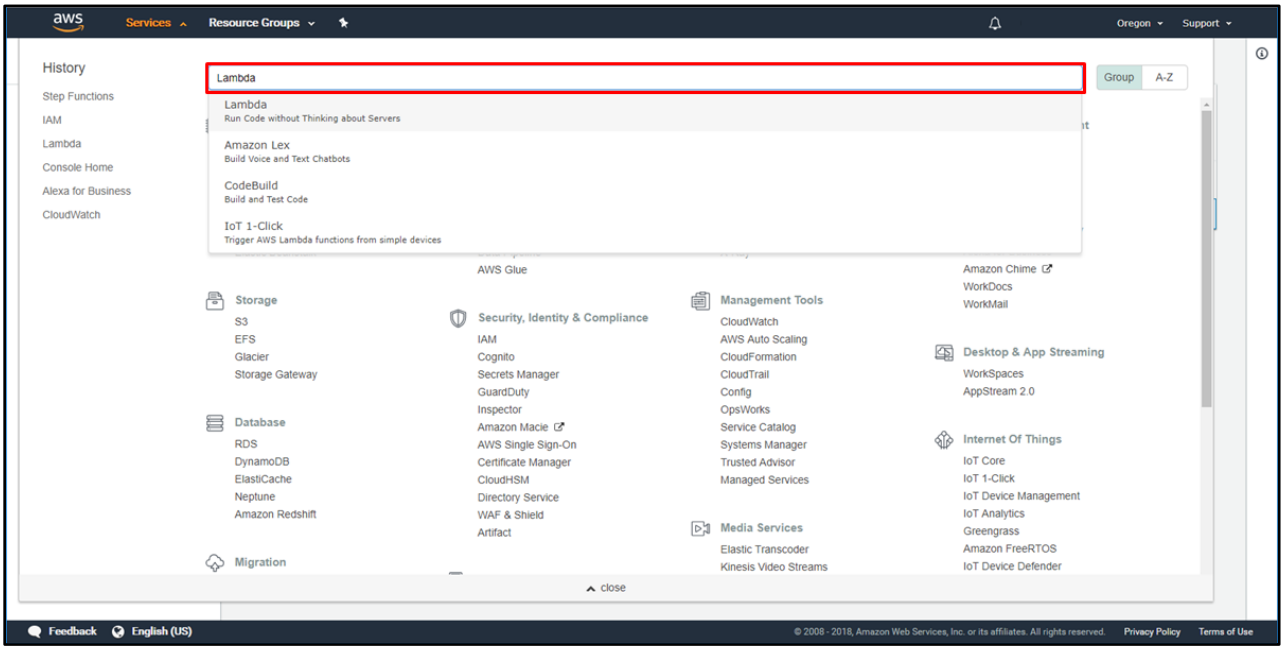
a. At the top of the AWS Step Functions console window, click **State machines**.



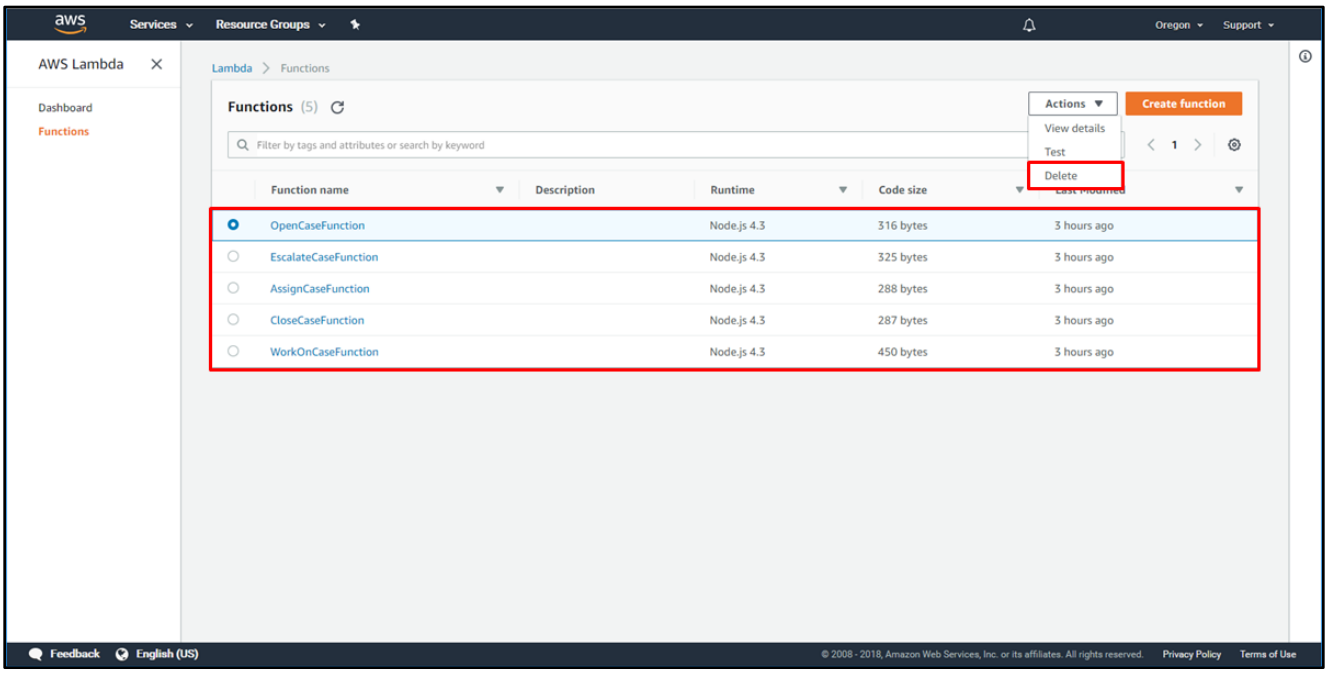
b. In the **State machines** window, select the *CallCenterStateMachine* and click **Delete**. To confirm you want to delete the state machine, in the dialog box that appears, click **Delete state machine**. Your state machine will be deleted in a minute or two, after Step Functions has confirmed that any in-process executions have completed.



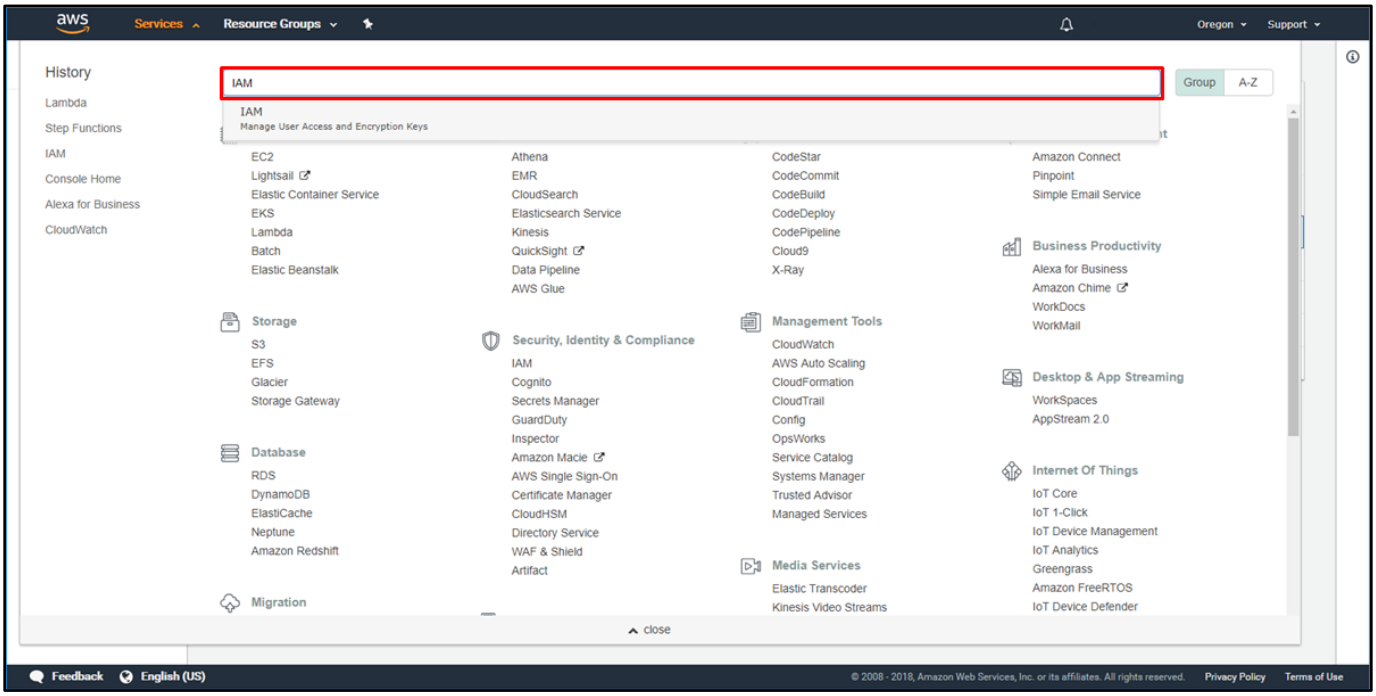
c. Next, you'll delete your Lambda functions. Click **Services** in the AWS Management Console menu, then select **Lambda**.



d. In the **Functions** screen, click each of the functions you created for this tutorial and then select **Actions** and then **Delete**. Confirm the deletion by clicking **Delete** again.



e. Lastly, you'll delete your IAM roles. Click **Services** in the AWS Management Console menu, then select **IAM**.



f. Select both of the IAM roles that you created for this tutorial, then click **Delete role**. Confirm the delete by clicking **Yes, Delete** on the dialog box.