PROJECT:4

Orchestrate Queue-based Microservices:-

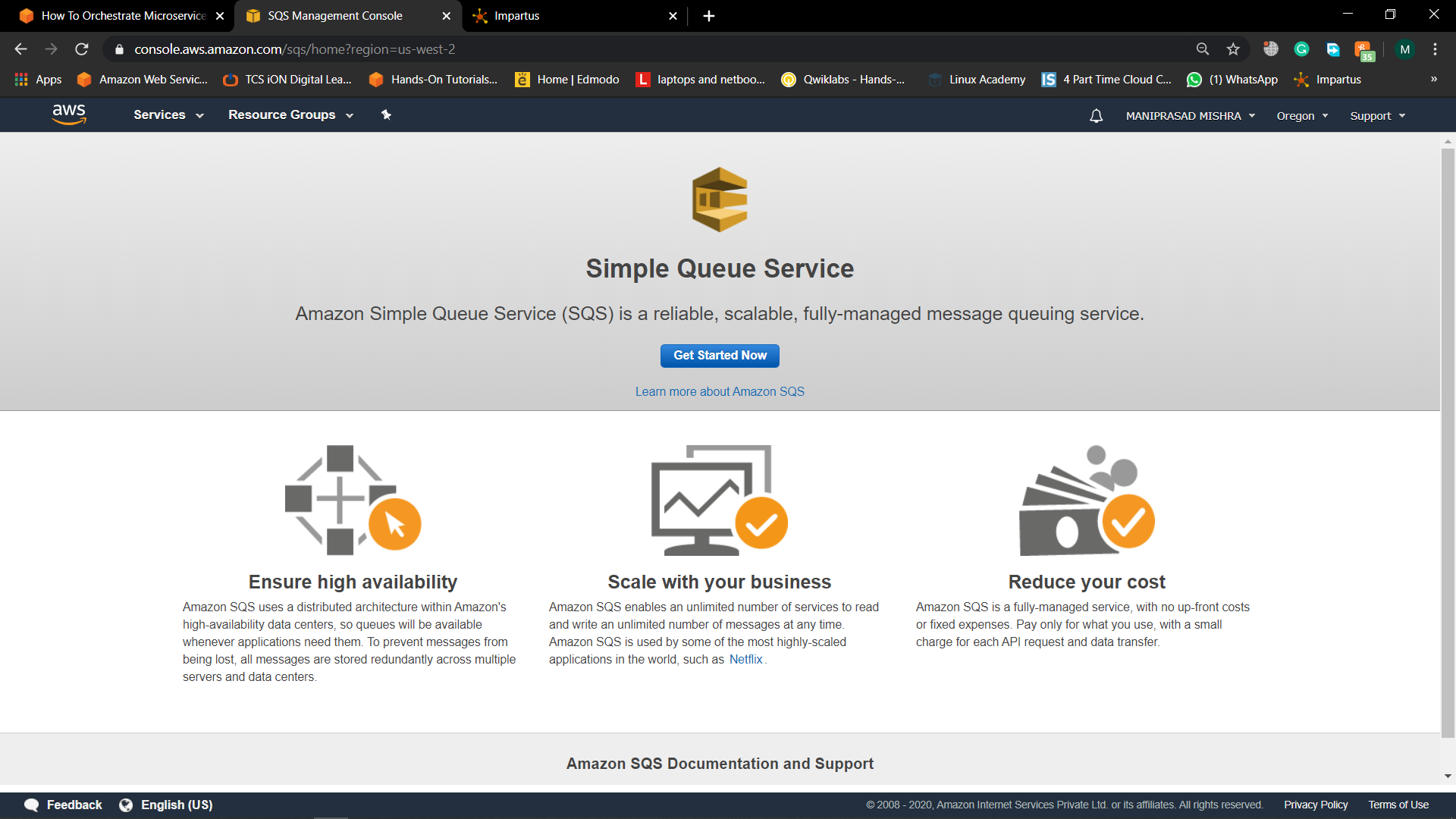
In this project, we will learn how to use AWS Step Functions and Amazon SQS to design and run a serverless workflow that orchestrates a message queue-based microservice. Step Functions is a serverless orchestration service that lets you easily coordinate multiple AWS services into flexible workflows that are easy to debug and easy to change. Amazon SQS is the AWS service that allows application components to communicate in the cloud.

In this case it will simulate inventory verification requests from incoming orders in an e-commerce application as part of an order processing workflow. Step Functions will send inventory verification requests to a queue on SQS. An AWS Lambda function will act as your inventory microservice that uses a queue to buffer requests. When it retrieves a request, it will check inventory and then return the result to Step Functions. When a task in Step Functions is configured this way, it is called a callback pattern. Callback patterns allow you to integrate asynchronous tasks in your workflow, such as the inventory verification microservice of this project.

## **Step 1: Enter the Amazon SQS Console:-**

1. Open the [AWS Management Console](https://us-east-2.console.aws.amazon.com/console/home?region=us-east-2), so you can keep this step-by-step guide open. When the screen loads, enter your user name and password to get started. In the search text box, type *SQS* and select *Simple Queue Service* to open the service console.

Pic:1(Login to Amazon SQS)

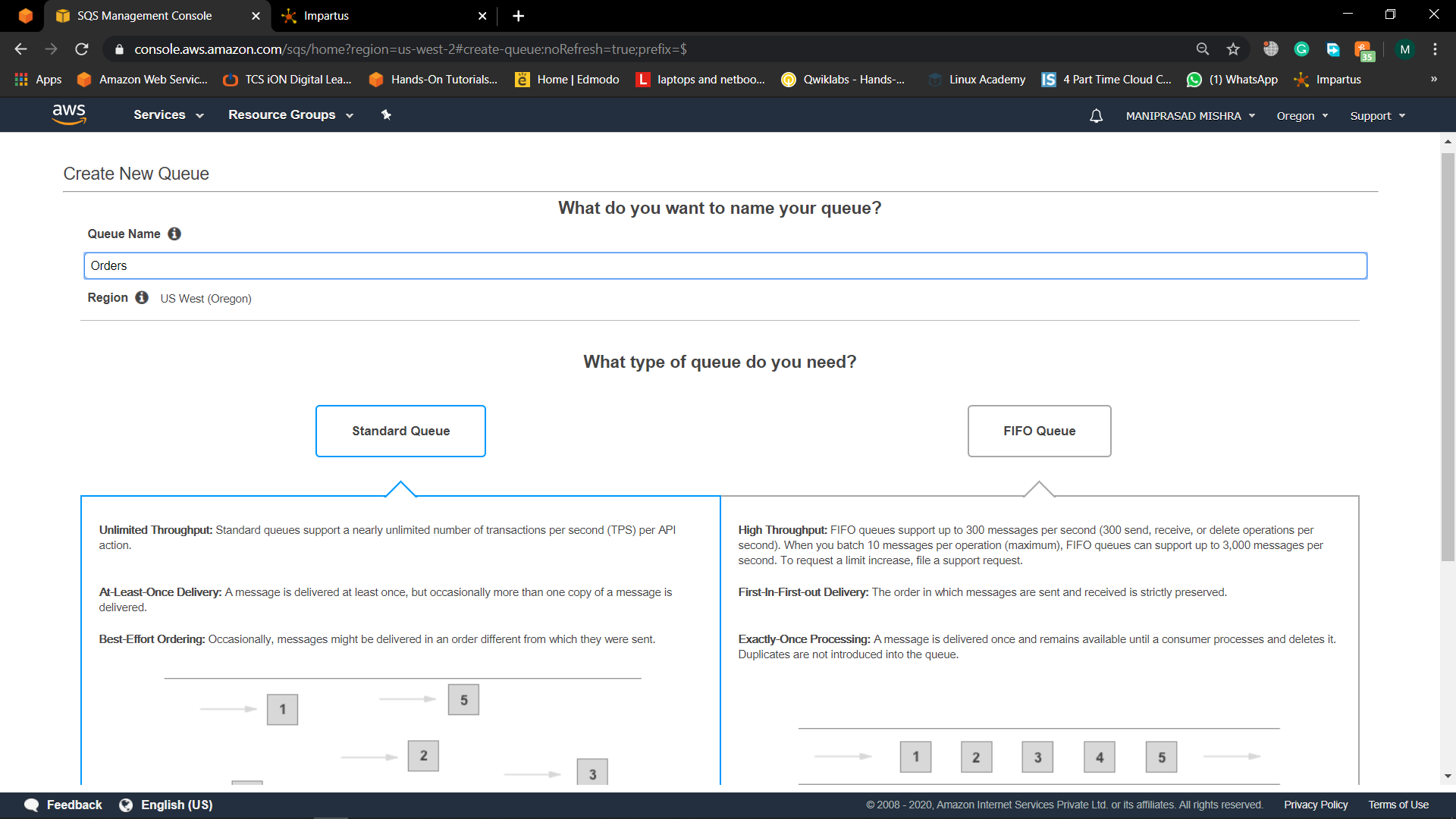


1. If the SQS console landing page appears, as shown on by the screenshot, click Get Started Now. If you don't see this page, skip to the next step.

## **Step 2: Create an Amazon SQS Queue:-**

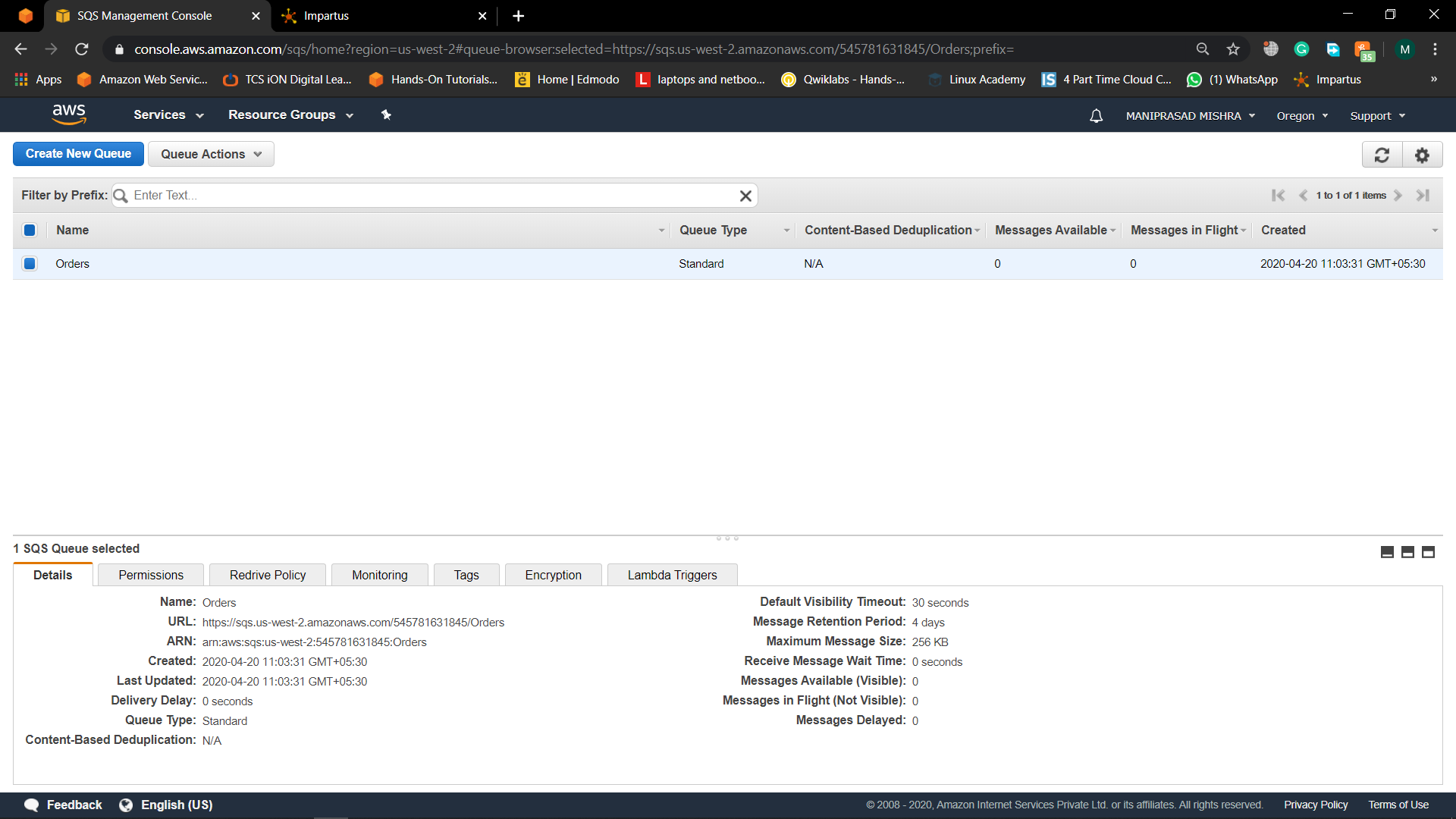
1. First, we will create a simple queue that stores orders that are placed on the store. Enter *Orders* in the Queue Name field

Pic:2(Create a SQS queue)



1. For this tutorial, we do not require strict ordering, so we won’t make any changes to the queue type. Leave Standard Queue selected.

Pic:3(Choose the Queue we created)

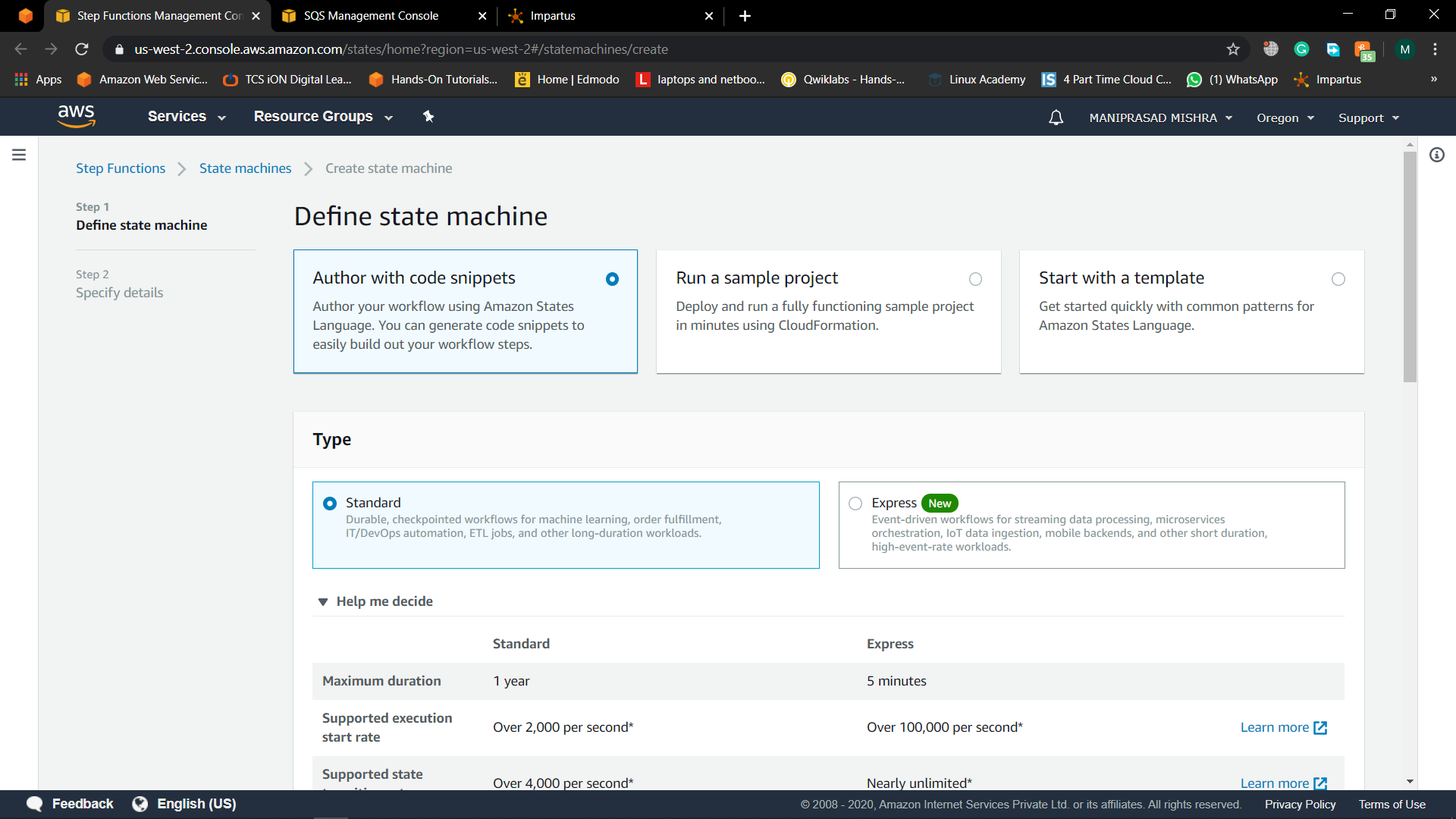


1. Your new queue is created and selected in the queue list.

**Step 3: Create a Workflow with a State Machine:-**

a.Open the [AWS Step Functions console](https://console.aws.amazon.com/states/home#/statemachines/create). Select Author with code snippets, then name your state machine *InventoryStateMachine*.

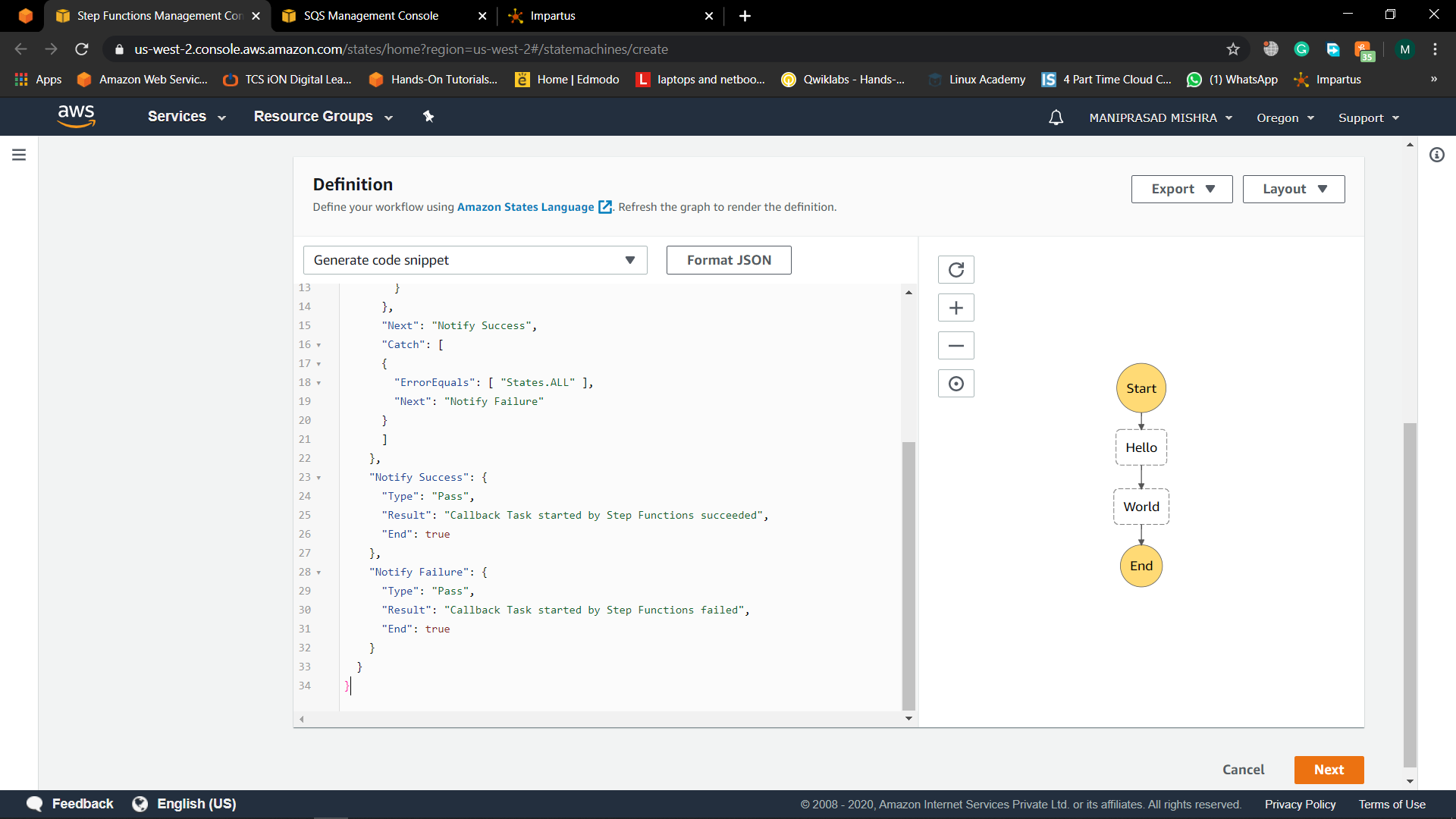
Pic:4(Create a workflow machin)



b. Replace the contents of the State machine definition window with the [Amazon States Language (ASL)](https://states-language.net/spec.html) state machine definition below. Amazon States Language is a JSON-based, structured language used to define your state machine.

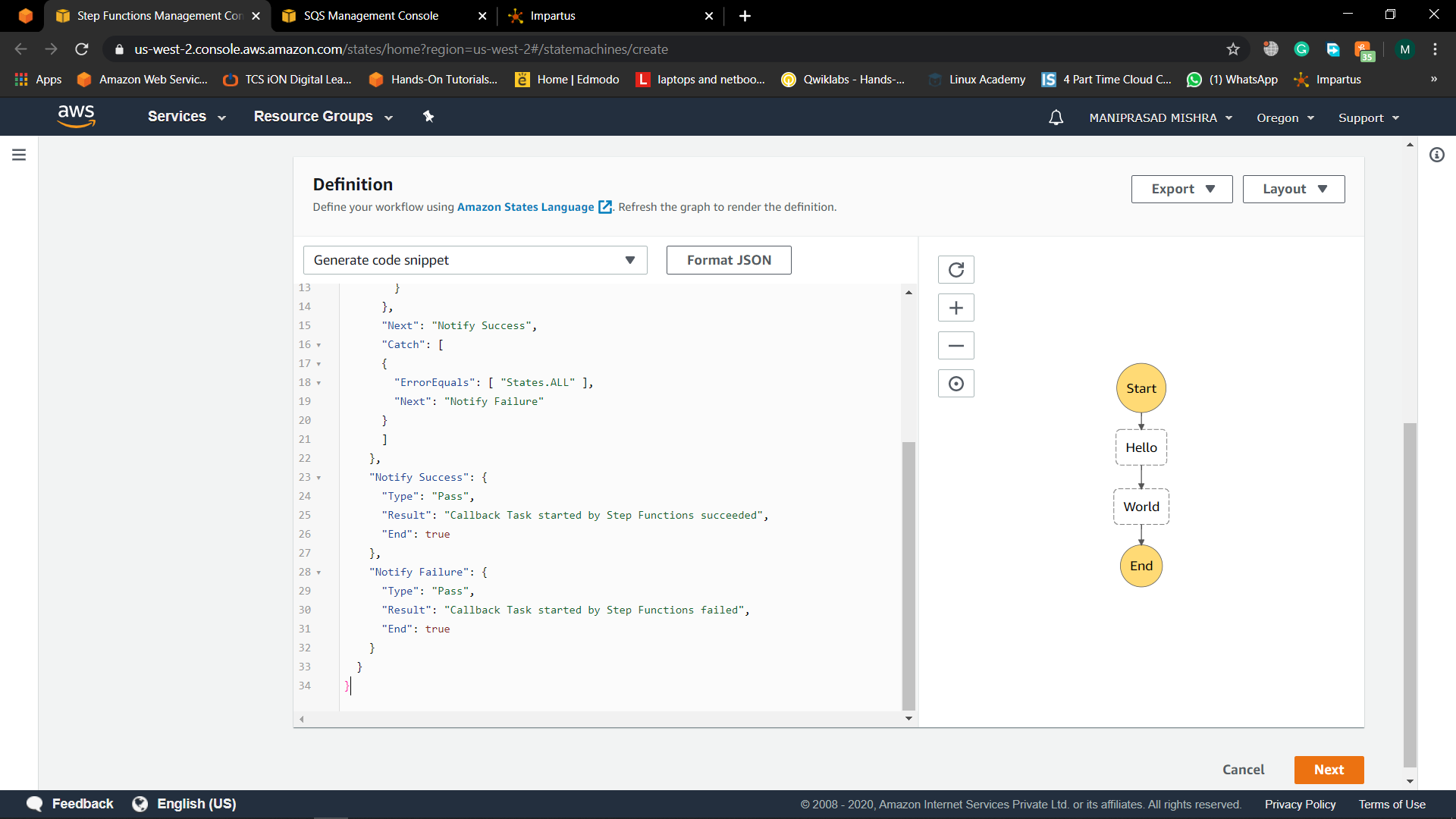
This state machine uses a task state to put a message on an SQS queue. This task state is configured for a callback pattern. When you append .waitForTaskToken to your resource, Step Functions will add a task token to the JSON payload and wait for a callback. The microservice can return a result to Step Functions by calling the Step Functions API.

Pic:5(Change the json code default to the usual code)



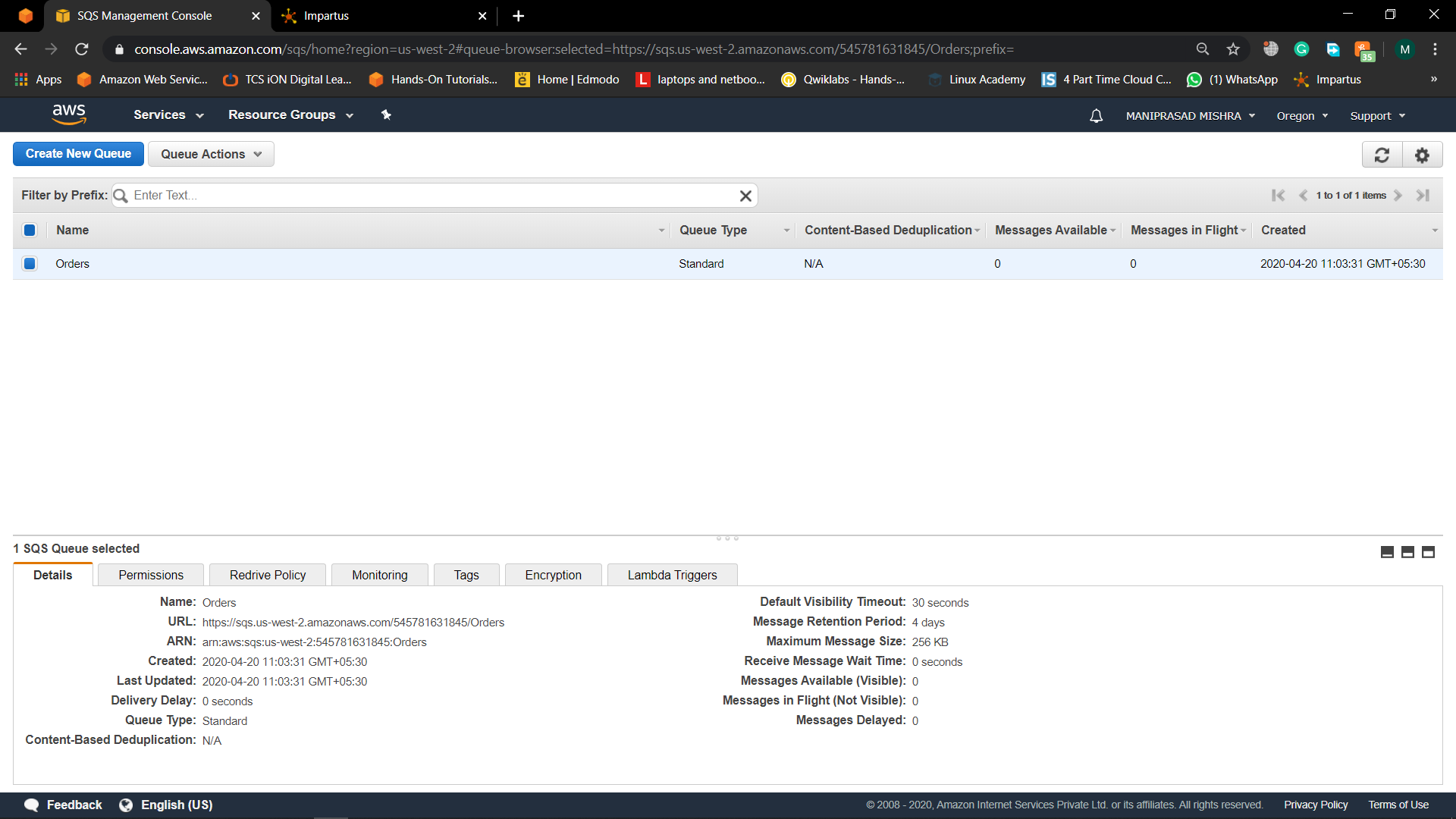
Change the code to the undergiven json code

Pic:4(After changing the code then the workflow is changed)



1. Copy the URL of your SQS queue from the SQS console and paste into your State Machine definition.

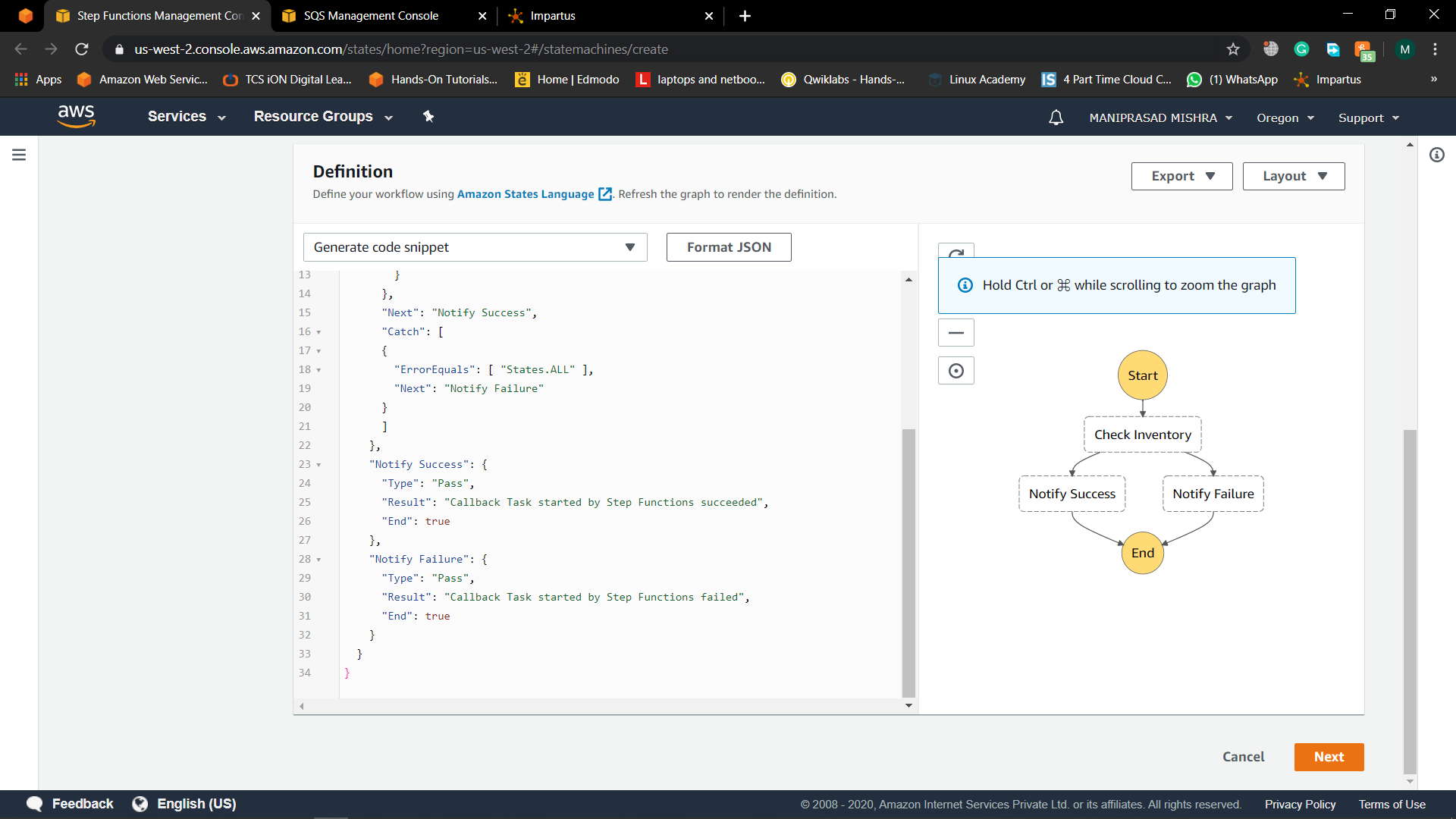
Pic:5(Then copy the ARN and paste it to the give code)



d. Click the refresh button to have Step Functions translate the ASL state machine definition into a visual workflow. You are able to easily verify that the process is described correctly by reviewing the visual workflow. When the microservice returns a result, the state machine will progress down the success branch. If something goes wrong, you can catch the exception and go down the right branch to take corrective action.

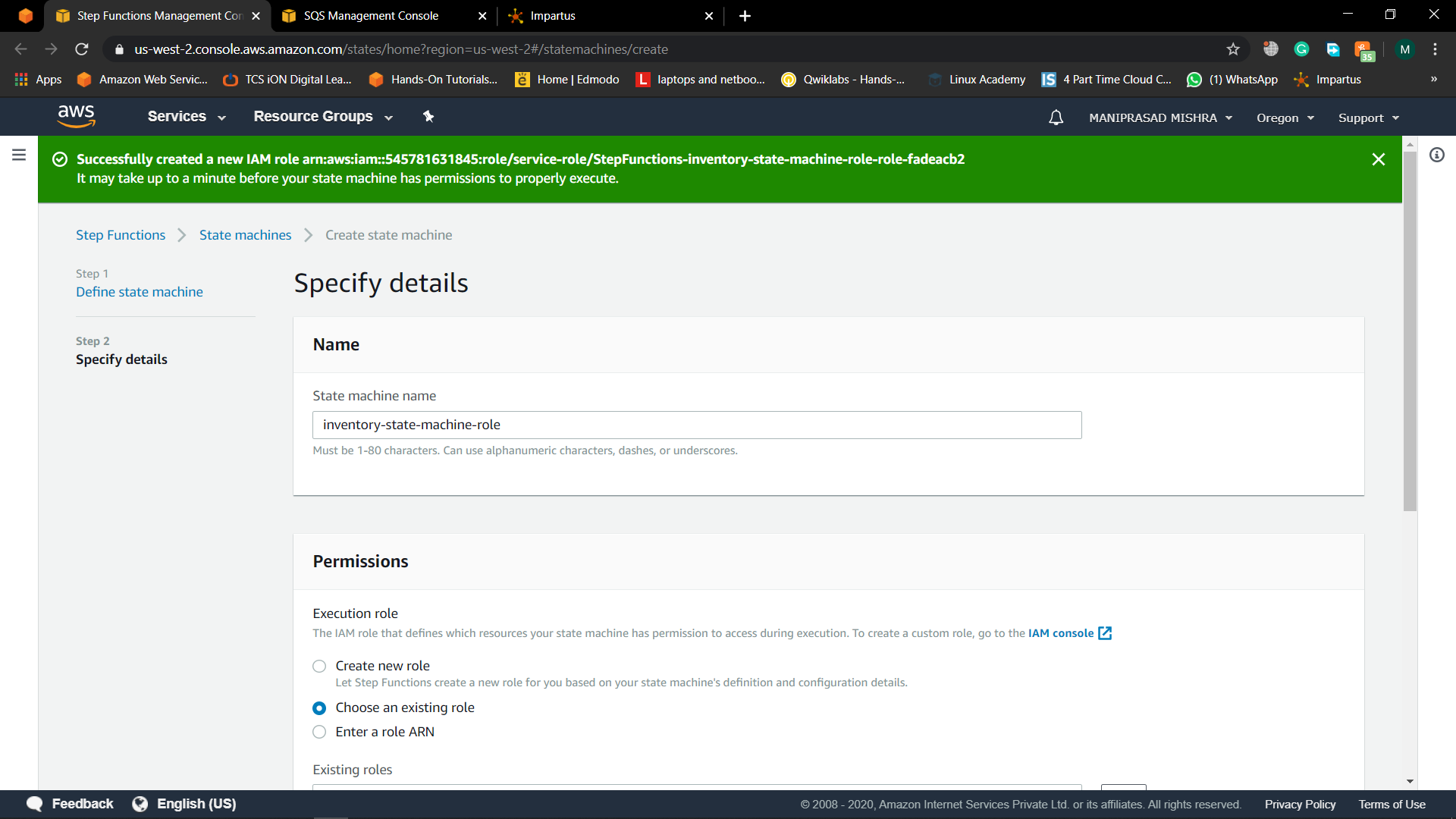
Click Next.

Pic:5(Click on refresh the workflow)



1. Next, you will add an IAM role to your workflow. Select Create an IAM role for me and name it *inventory-state-machine-role*. Step Functions will analyze your workflow and generate an IAM policy that includes resources used by your workflow. Click Create state machine. You should see a green banner indicating your state machine was successfully created.

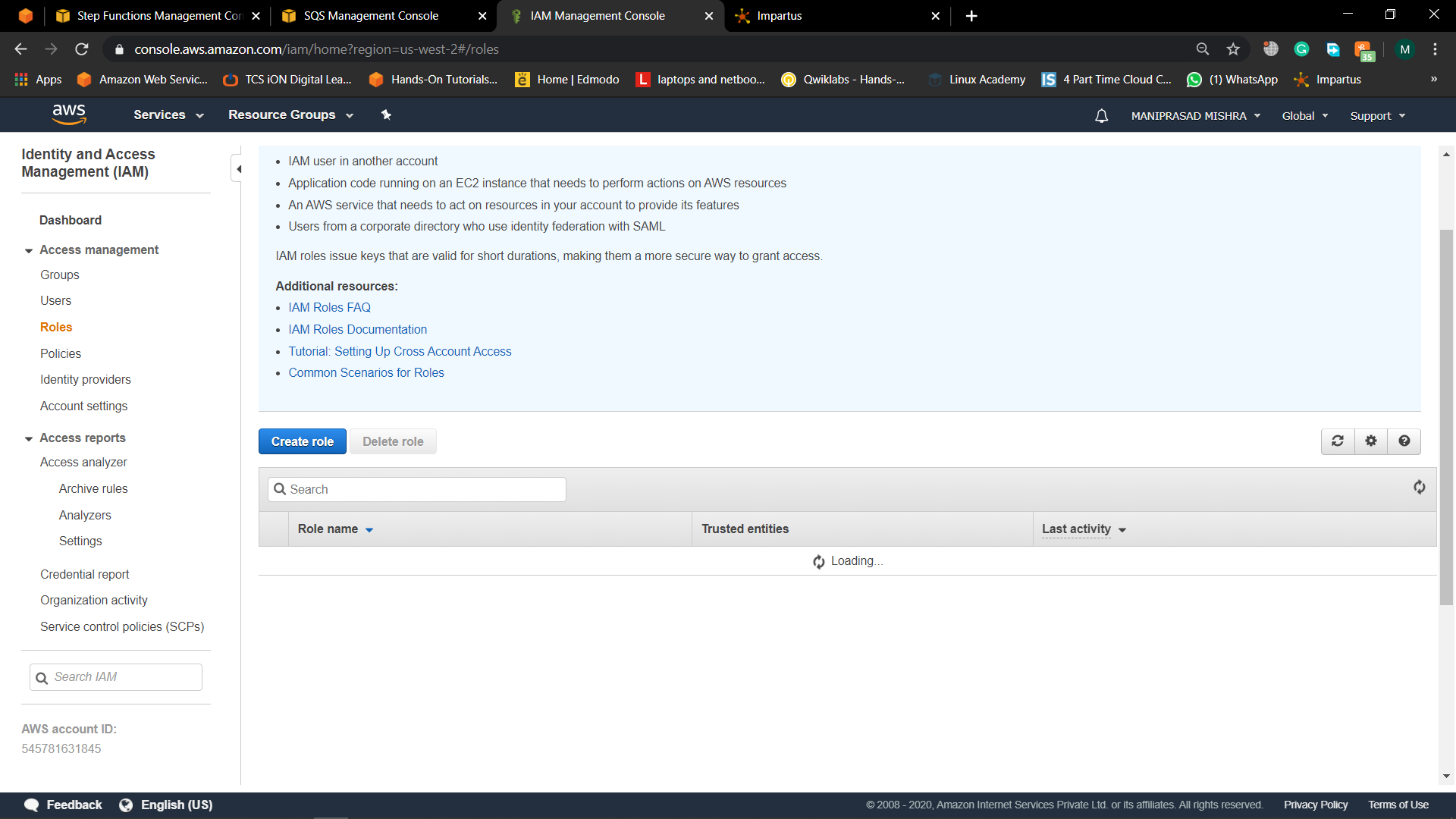
Pic:6(Specify the role as existing role)



## **Step 4: Create an AWS Identity and Access Management (IAM) Role:-**

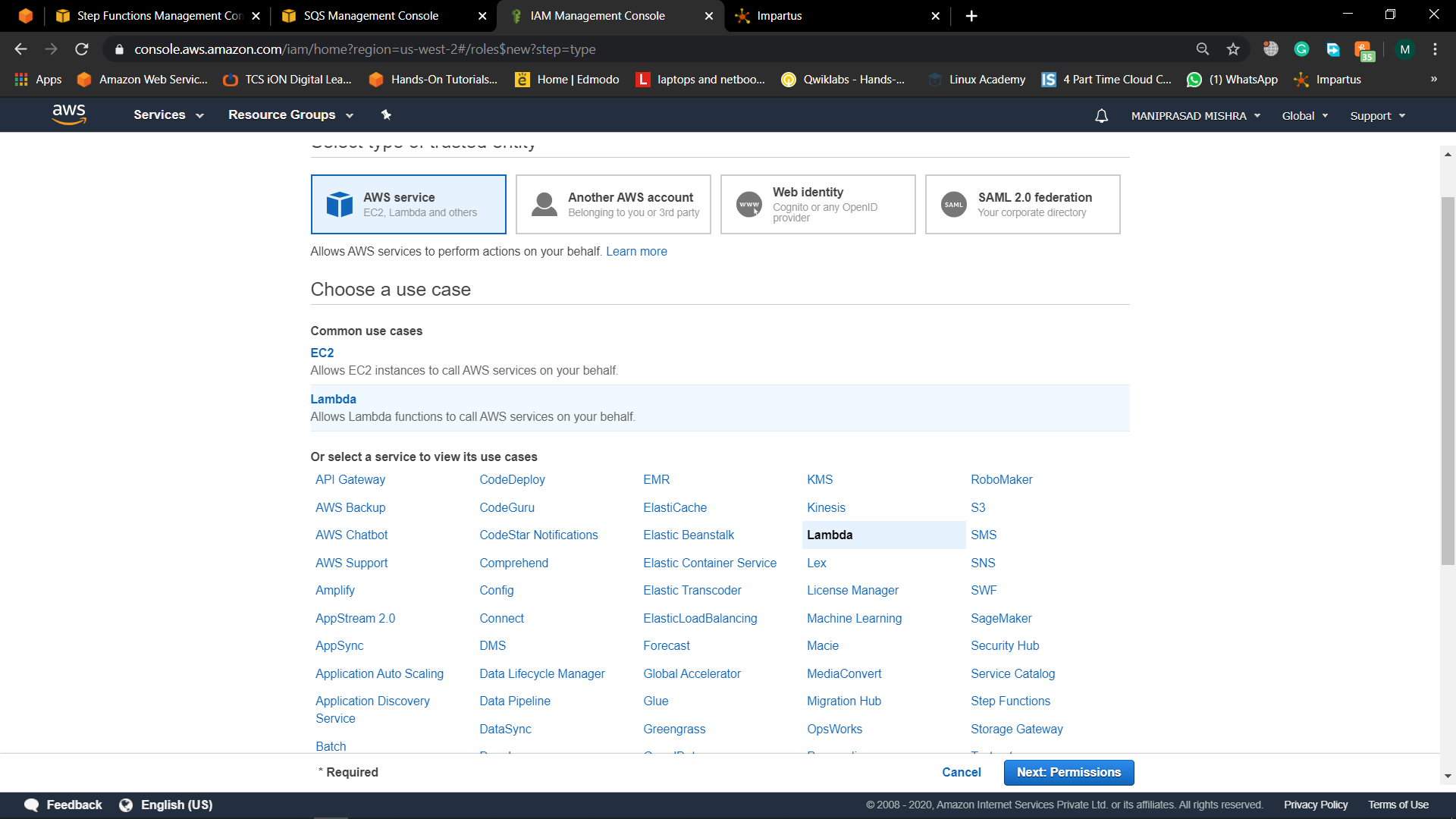
1. In another browser window, open the [AWS Management Console](https://console.aws.amazon.com/console/home). When the screen loads, type *IAM* in the search bar, then select *IAM* to open the service console.
2. Click Roles and then click Create Role.

Pic:7(Create a IAM role)



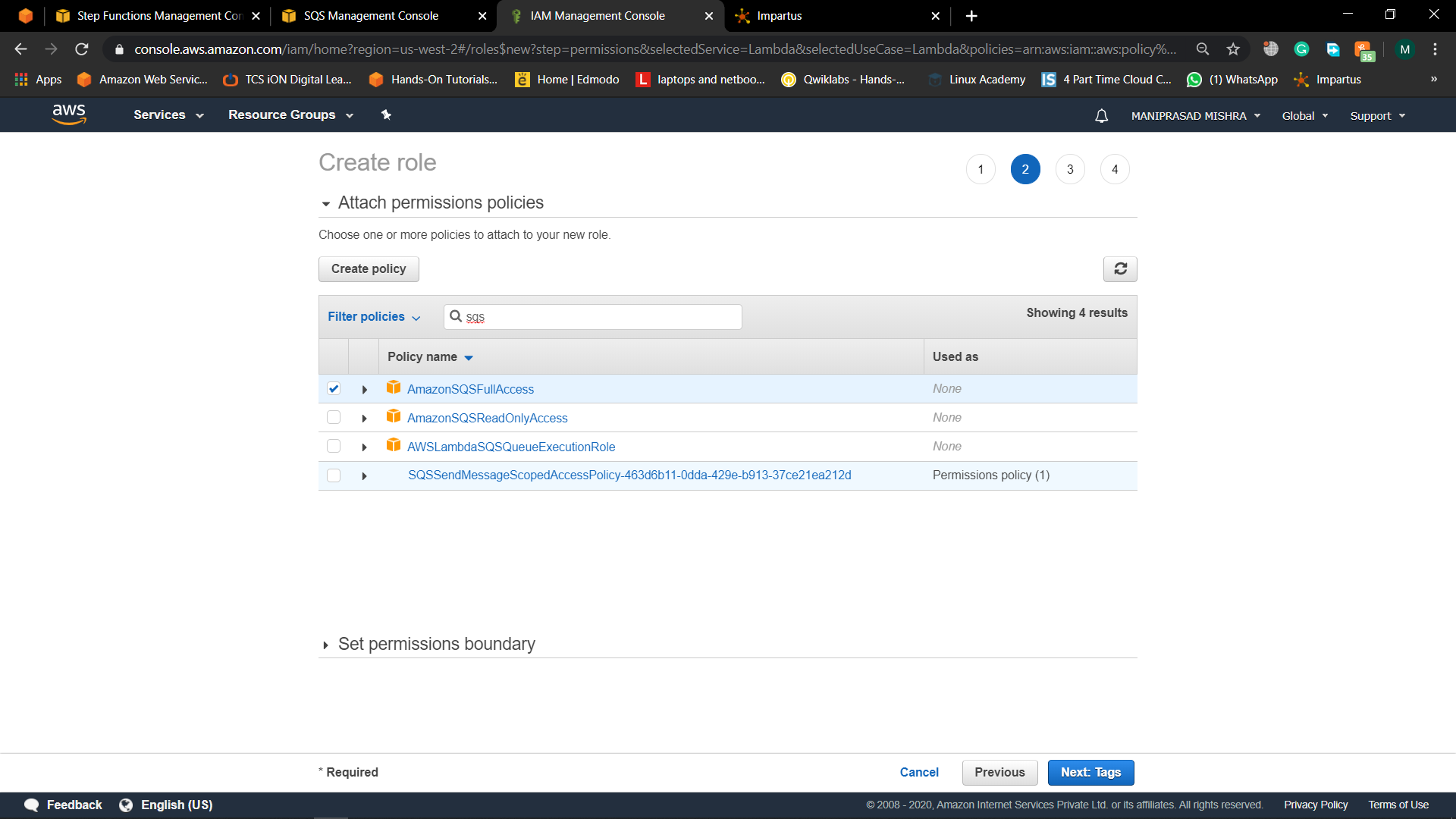
1. On the Create Role screen, leave AWS Service selected, select *Lambda* then click Next: Permissions.

Pic:8(Create a IAM role for Lambda function)

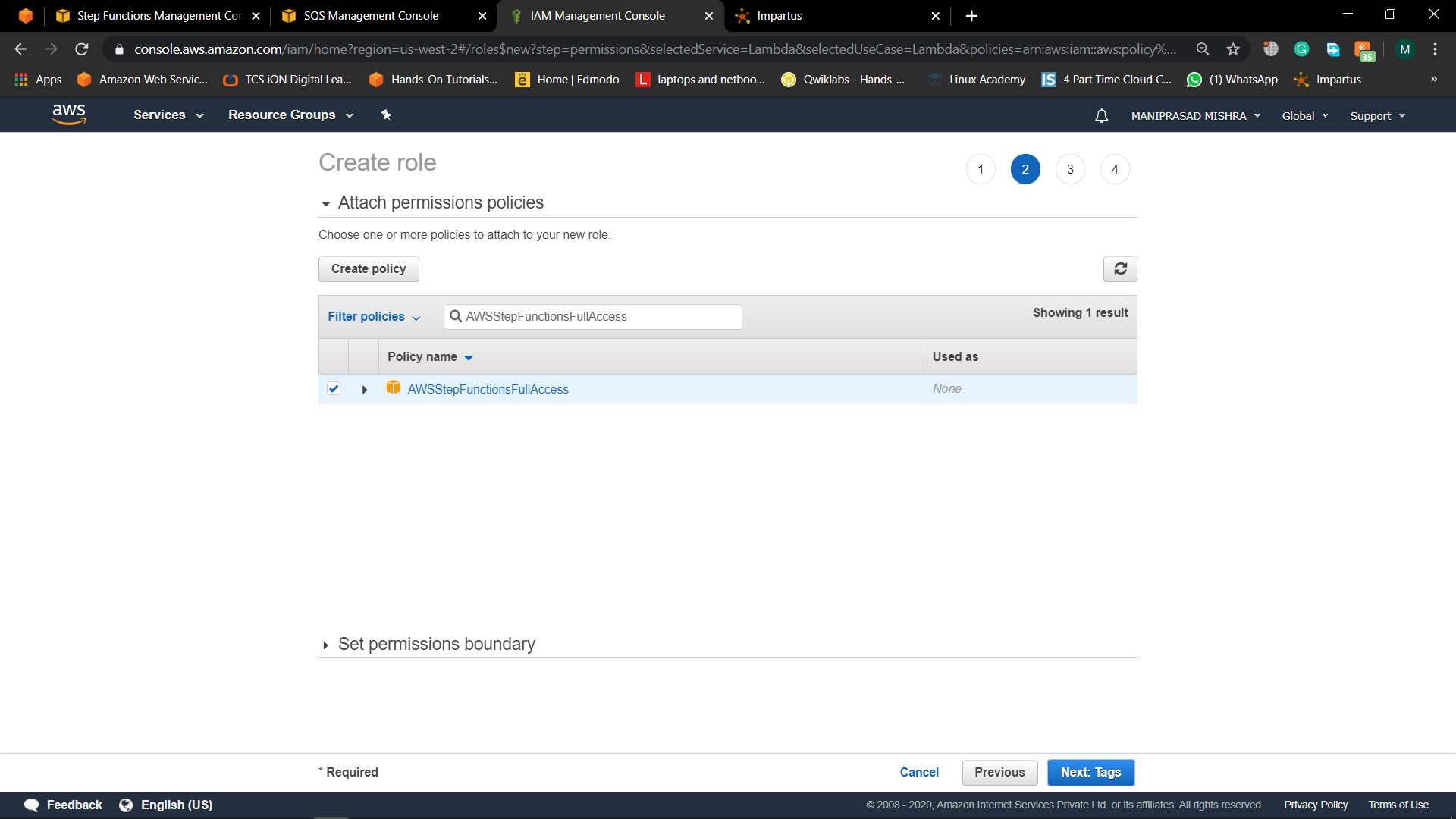


1. On the Create role screen, attach  
   *AmazonSQSFullAccess* and *AWSStepFunctionsFullAccess* policies. Click Next: Tags and then click Next: Review.

Pic:9(Attach permission and policy)

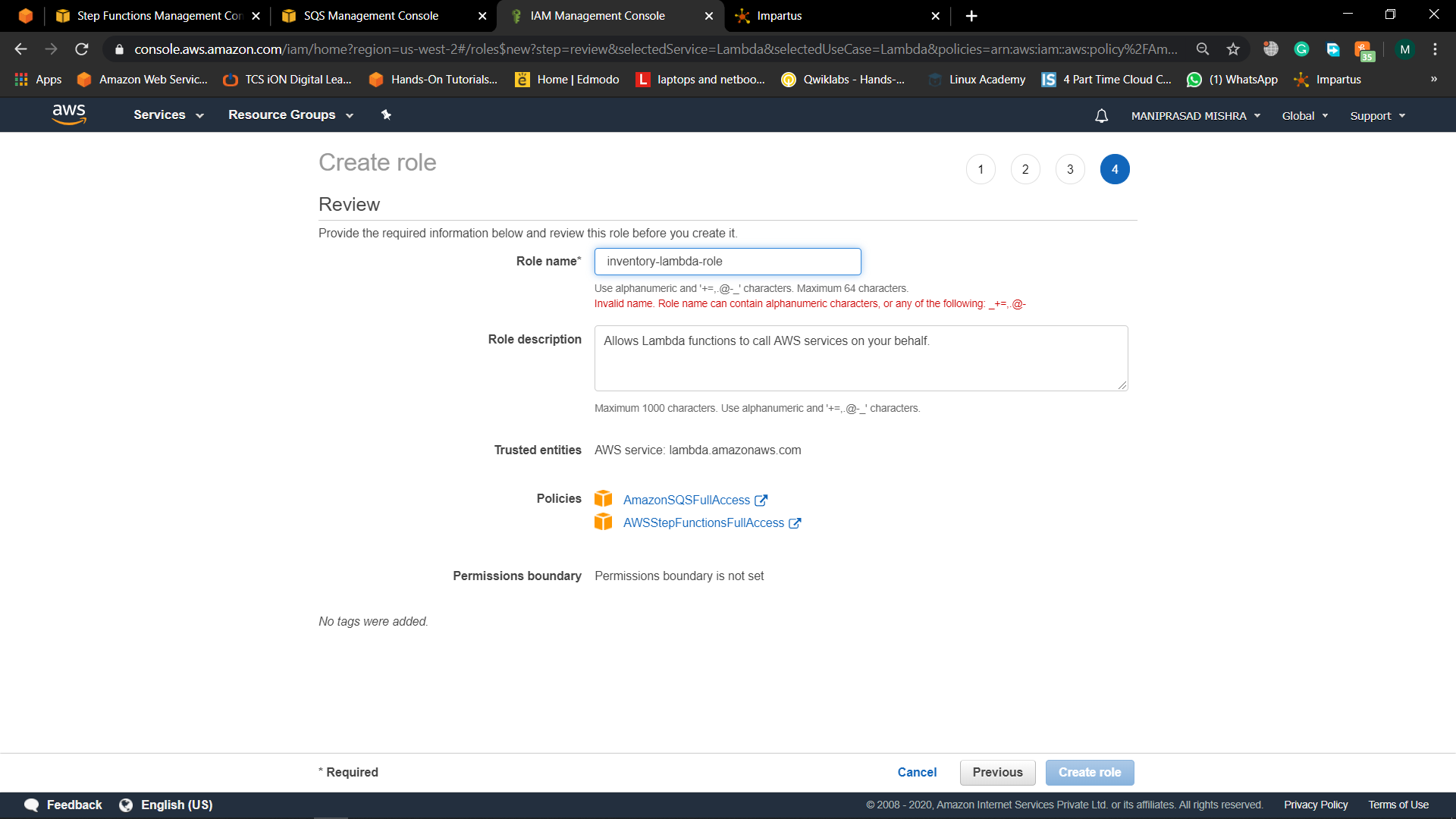


Pic:10(Choose the other permission)



1. Enter Role name as*inventory-lambda-role* and click Create role.

Pic:11(Give the role a name)



## **Step 5: Create your Microservice with AWS Lambda Functions:-**

1. Click on Services, type *Lambda* in the search bar, then select *Lambda* to open the service console.

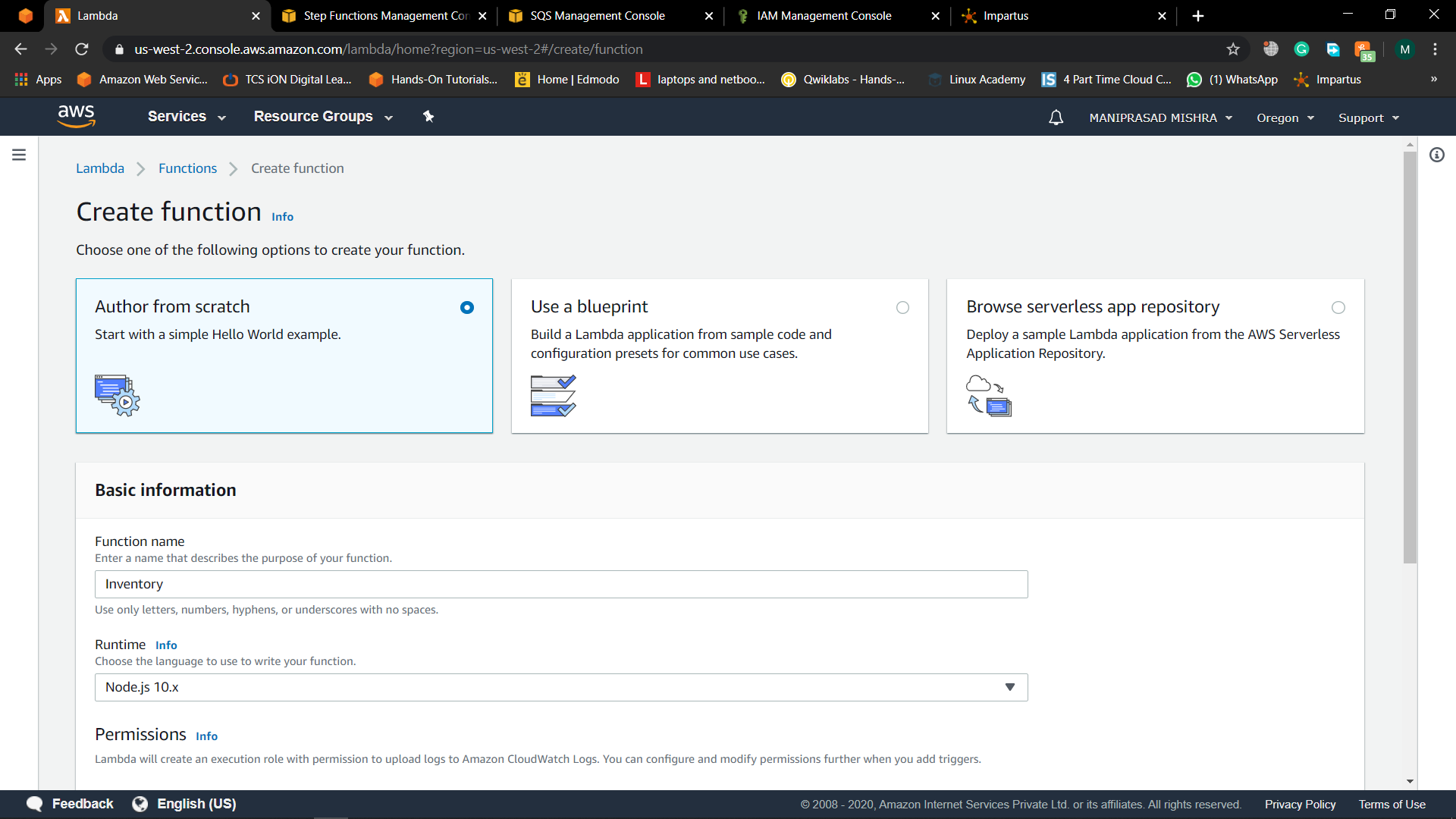
b. Choose Create function

c. Leave Author from scratch selected. Next, configure your first Lambda function as follows

For Name, type *Inventory*.  
 For Runtime, choose *Node.js 10*.x  
 For Role, select *Use an existing role*.

Select *inventory-lambda-role* from the list  
 Click Create function.

Pic:12(Create a lambda function and give the function a name)



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

console.log('Loading function');

const aws = require('aws-sdk');

exports.handler = (event, context, callback) => {

const stepfunctions = new aws.StepFunctions();

for (const record of event.Records) {

const messageBody = JSON.parse(record.body);

const taskToken = messageBody.TaskToken;

const params = {

output: "\"Callback task completed successfully.\"",

taskToken: taskToken

};

console.log(`Calling Step Functions to complete callback task with params ${JSON.stringify(params)}`);

stepfunctions.sendTaskSuccess(params, (err, data) => {

if (err) {

console.error(err.message);

callback(err.message);

return;

}

console.log(data);

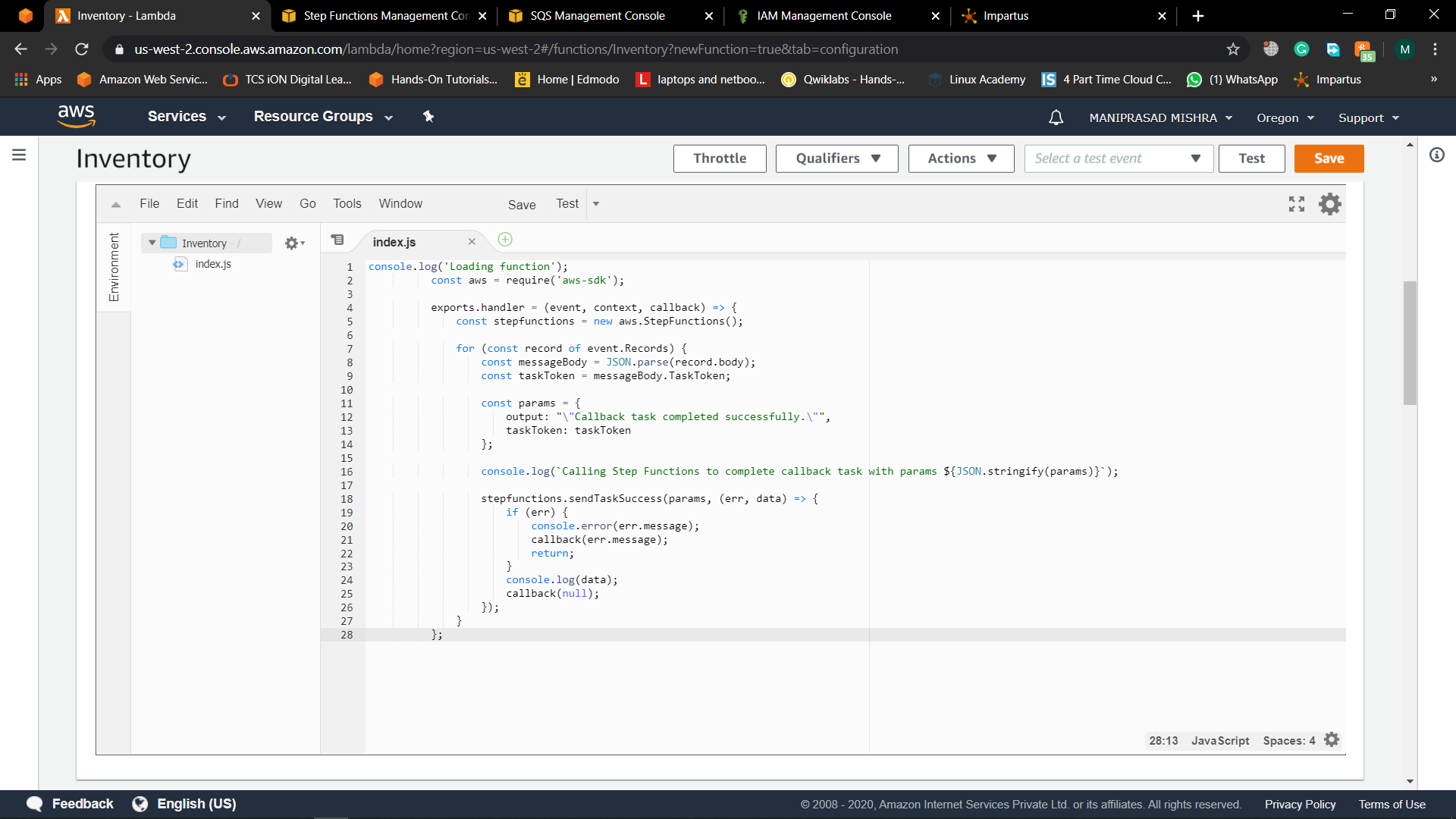
callback(null);

});

}

};

Pic:13(Change the default code of lambda to the javascript program)

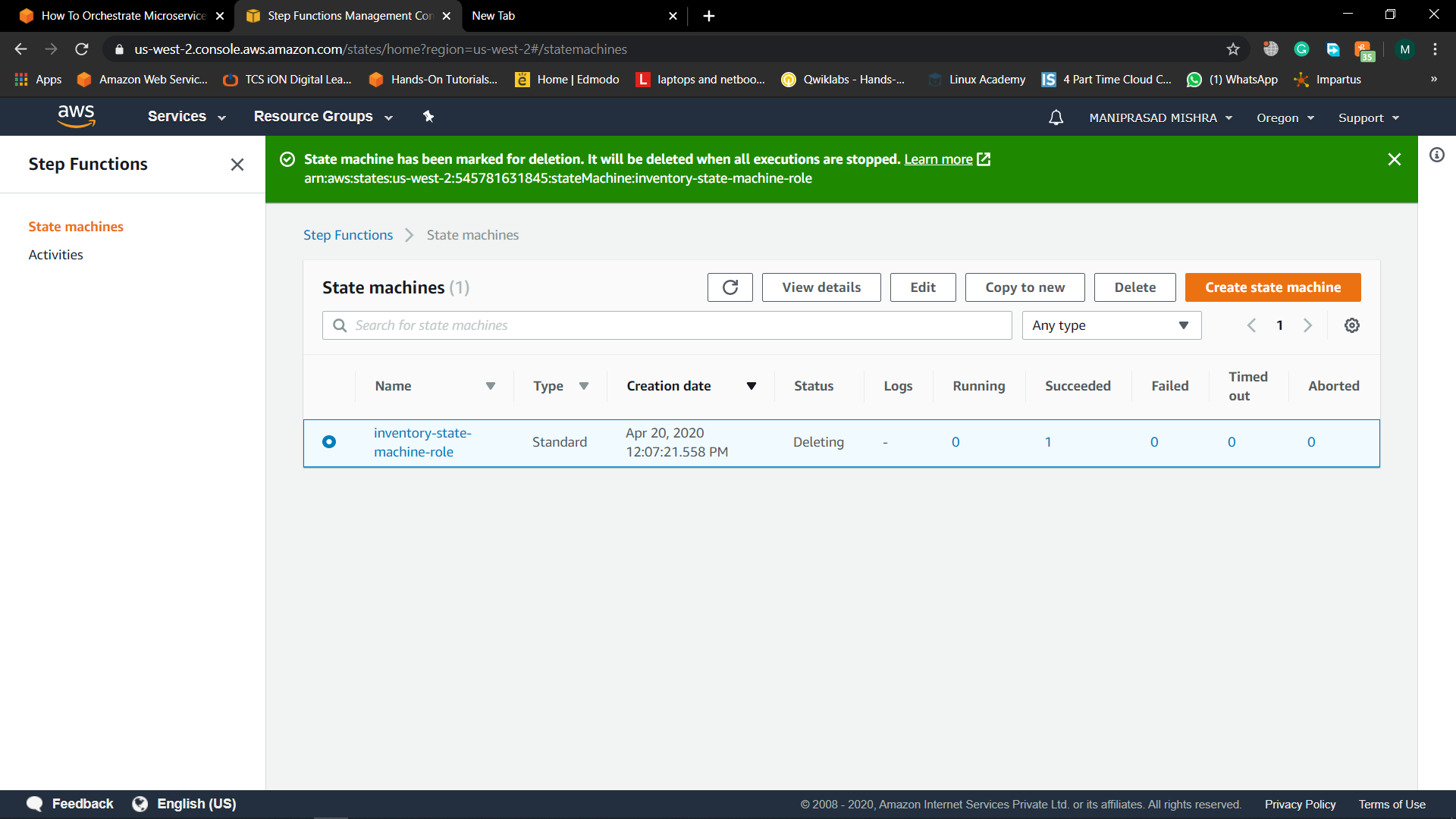


1. Select the SQS tigger. Set the toggle to *Enable* for your *Orders* queue. Click Add and then click Save.

## **Step 6: Execute your workflow:-**

1. Click on Services, type *Step* in the search bar, then select *Step Functions* to open the service console.

Pic:14(Choose the state machine and review it)

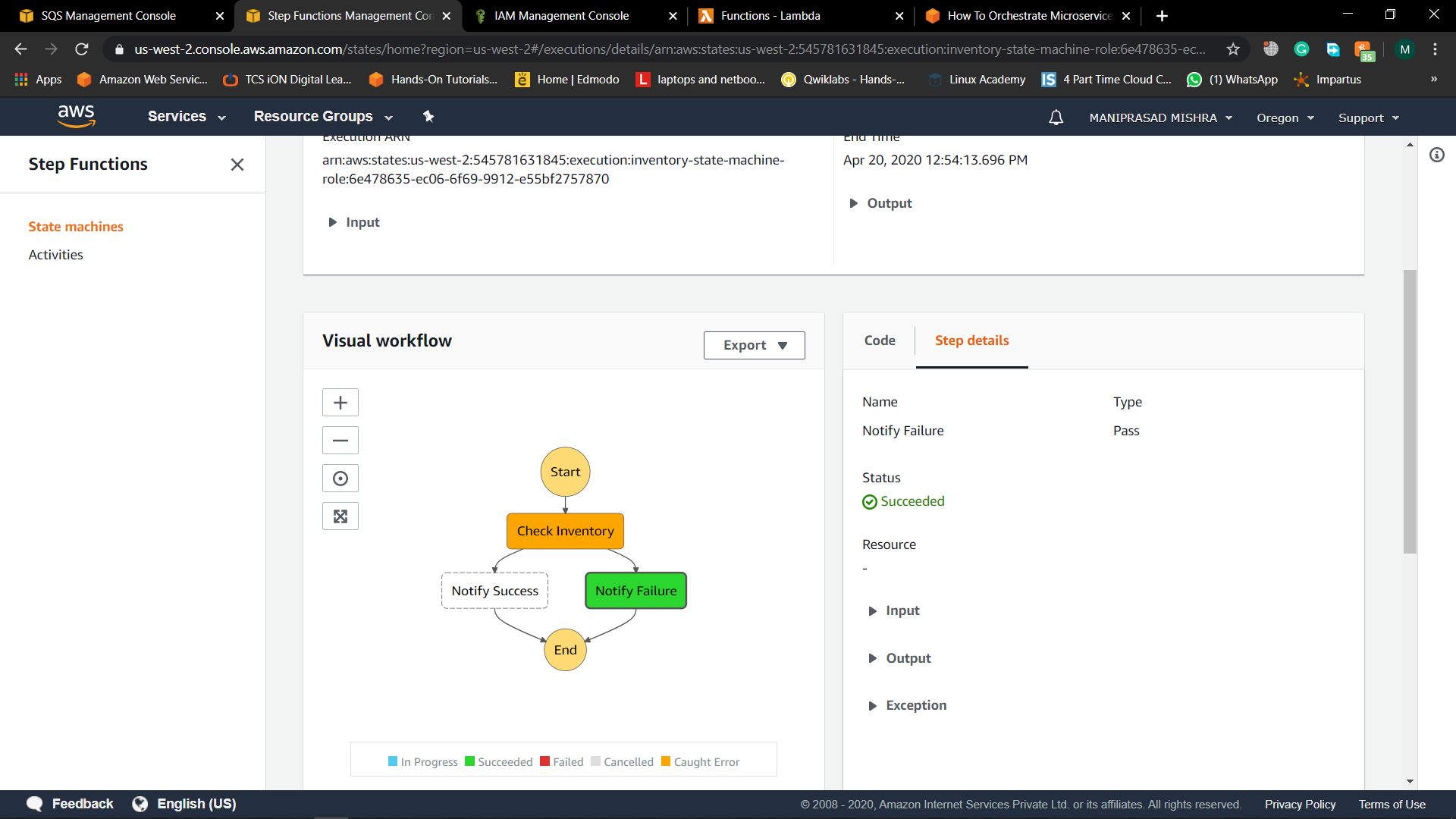
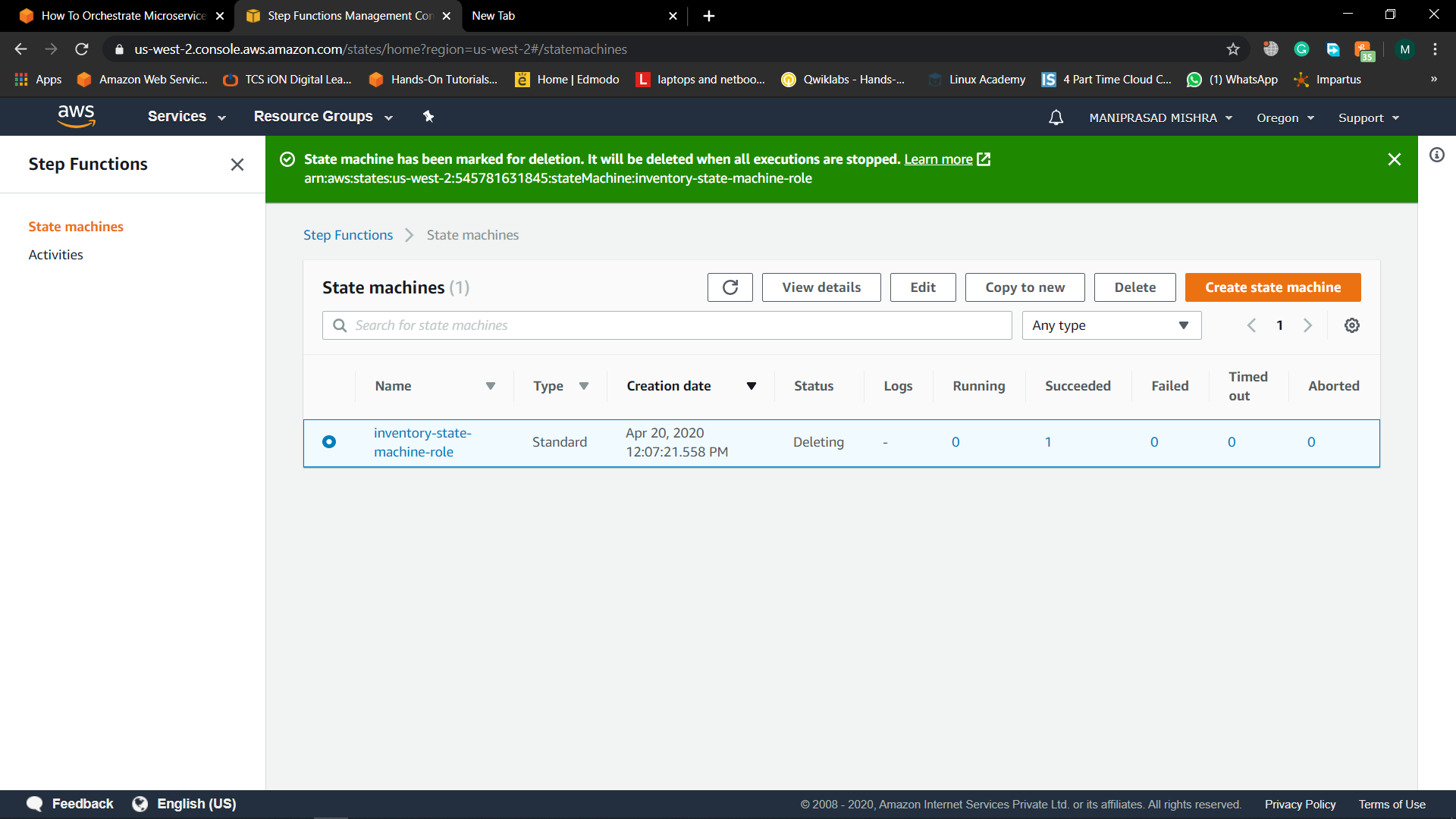


1. Click on Start execution.

d. A new execution dialog box appears, where you can enter input for your state machine. This state machine does not depend on the input. You can use the default input. Click on Start execution.

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

Pic:15(Choose the machine and see if the workflow is doing well or not )



Pic:16(Choose every step and see its info if the status of the step is good or not)

## **Step 7: Terminate your Resources:-**

1. At the top of the AWS Step Functions console window, click on State machines
2. In the State machines window, click on your *InventoryStateMachine* and select Delete. Confirm the action by selecting *Delete state machine* in the dialog box. Your state machine will be deleted in a minute or two once 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*.

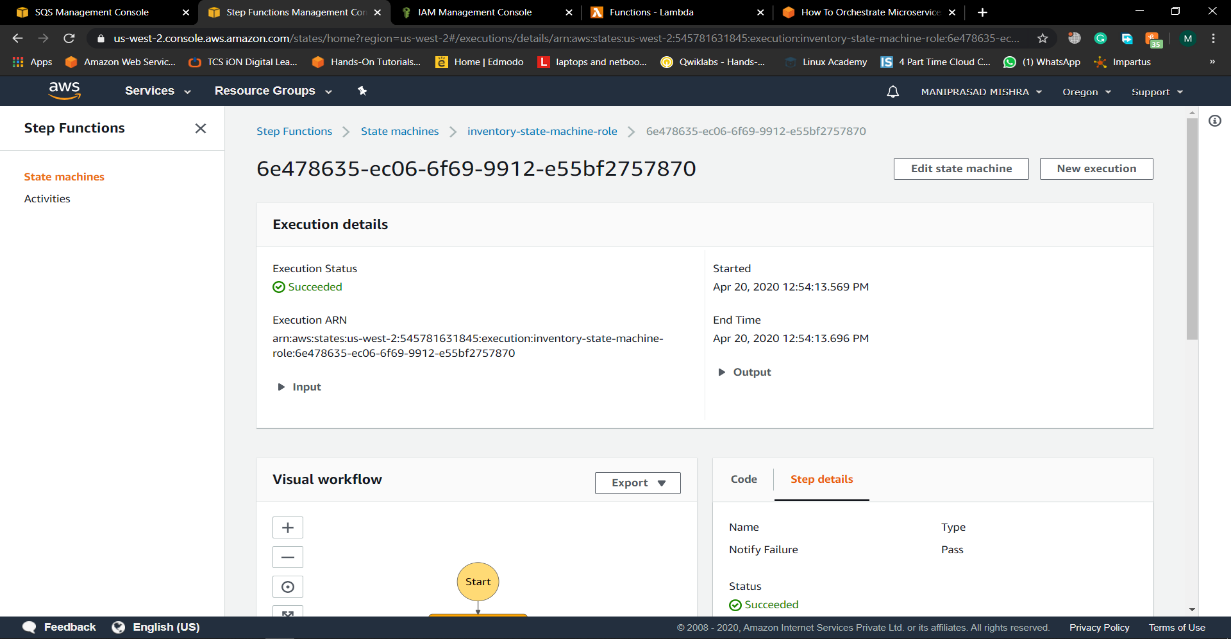
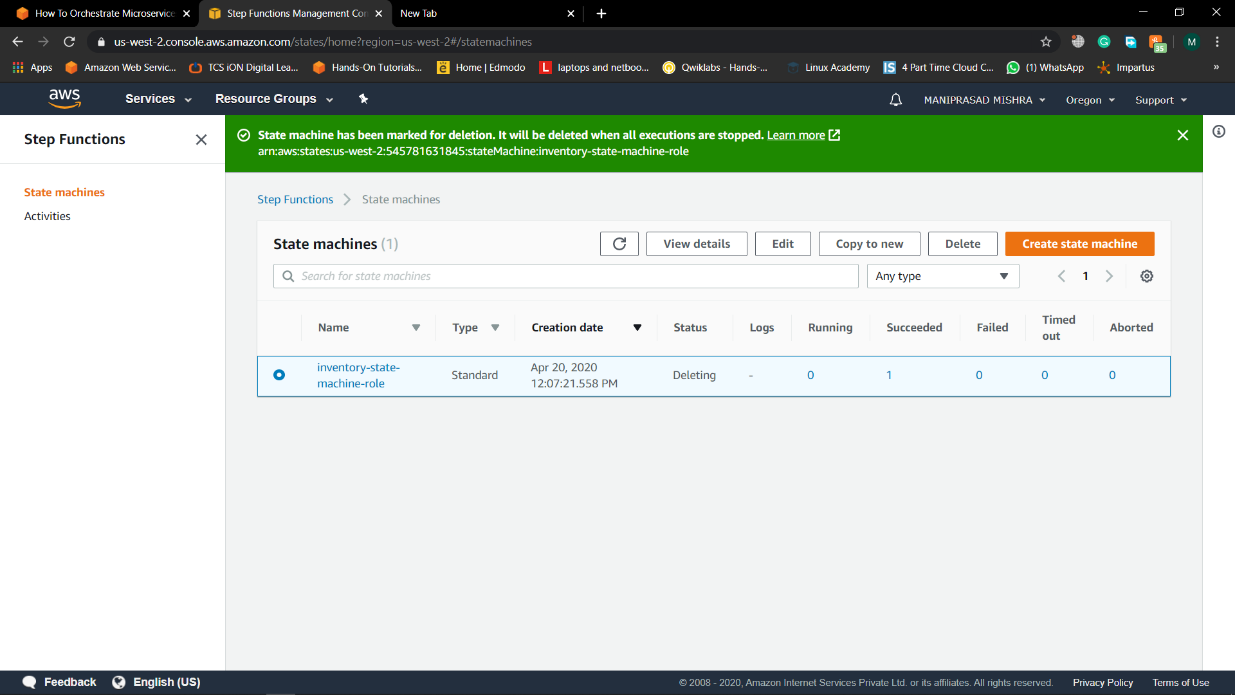
d. In the Functions screen, click the *Inventory* function you created for this tutorial and then select Actions and then *Delete*. Confirm the deletion by clicking Delete again.

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

f. In the queue list, select the *Orders* queue. Then, from Queue Actions, select Delete Queue.

h. The Delete Queues dialog box is displayed. You can delete your queue, even though you may still have messages in it. Choose Yes, Delete Queue. The queue is deleted.

Pic:17(Delete all the services which are created like IAM role,State Machin and Lambda function)



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*