AUTOMATE START-STOP EC2 INSTANCES USING LAMBDA

AWS PROJECT

Presented by: Shaik Mohammed Shabaz

PROJECT OVERVIEW

Objective:

- To automate the start and stop of Amazon EC2 instances using AWS Lambda functions based on specific triggers or schedules.
- The purpose of this project is to implement a cost-effective automation mechanism using AWS Lambda to **automatically start and stop Amazon EC2 instances** at predefined times, eliminating the need for manual intervention and reducing unnecessary AWS usage charges.
- In many organizations or training environments, EC2 instances are used during working hours and remain idle during off-hours, leading to unnecessary costs. Manually managing their lifecycle is inefficient. This project leverages AWS serverless computing (Lambda) to automate the process based on a defined schedule using CloudWatch Events.

Goals of the Project:

- Automate the EC2 start and stop process.
- Reduce operational costs by turning off idle resources.
- Eliminate manual efforts and human errors.
- Use serverless (pay-as-you-go) architecture.

PROJECT ARCHITECTURE

Components Used :

1. AWS Lambda
2. Amazon EC2
3. IAM Roles, Policies
4. CloudWatch (EventBridge)

```
+---------------------------+          +-------------+-------------+          +---- --------+------ ------+
|  CloudWatch Events  |          |     AWS Lambda      |          |      EC2 Instances      |
|   (Scheduled Rule)    |   >      |  (Start/Stop Script)  |   >      |  (Start / Stop by ID)   |
+------------+-------------+          +-------------+------------+          +---------------------------+
```

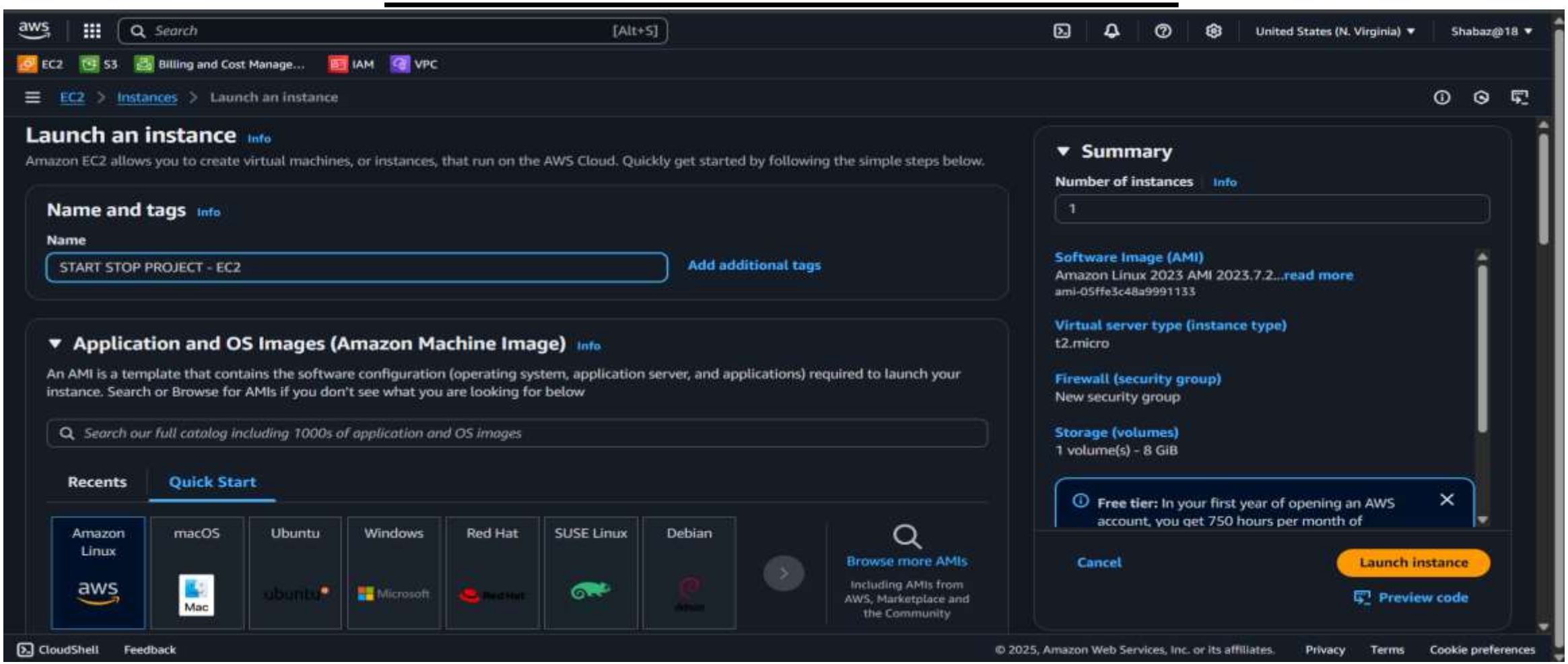
TOOLS & TECHNOLOGIES USED

- AWS LAMBDA (Serverless Compute)
- AMAZON EC2 (Virtual Machines)
- CLOUDWATCH EVENTS / EVENTBRIDGE (Scheduling)
- IAM (For Permissions)
- PYTHON (Boto3 SDK)

PROJECT WORKFLOW

- Created EC2 Instance
- Created IAM Role
- Created IAM Policies
- Created Lambda Functions
- Configured CloudWatch Events
- Used Python (Boto3) Code
- Triggered Lambda via CloudWatch Scheduler

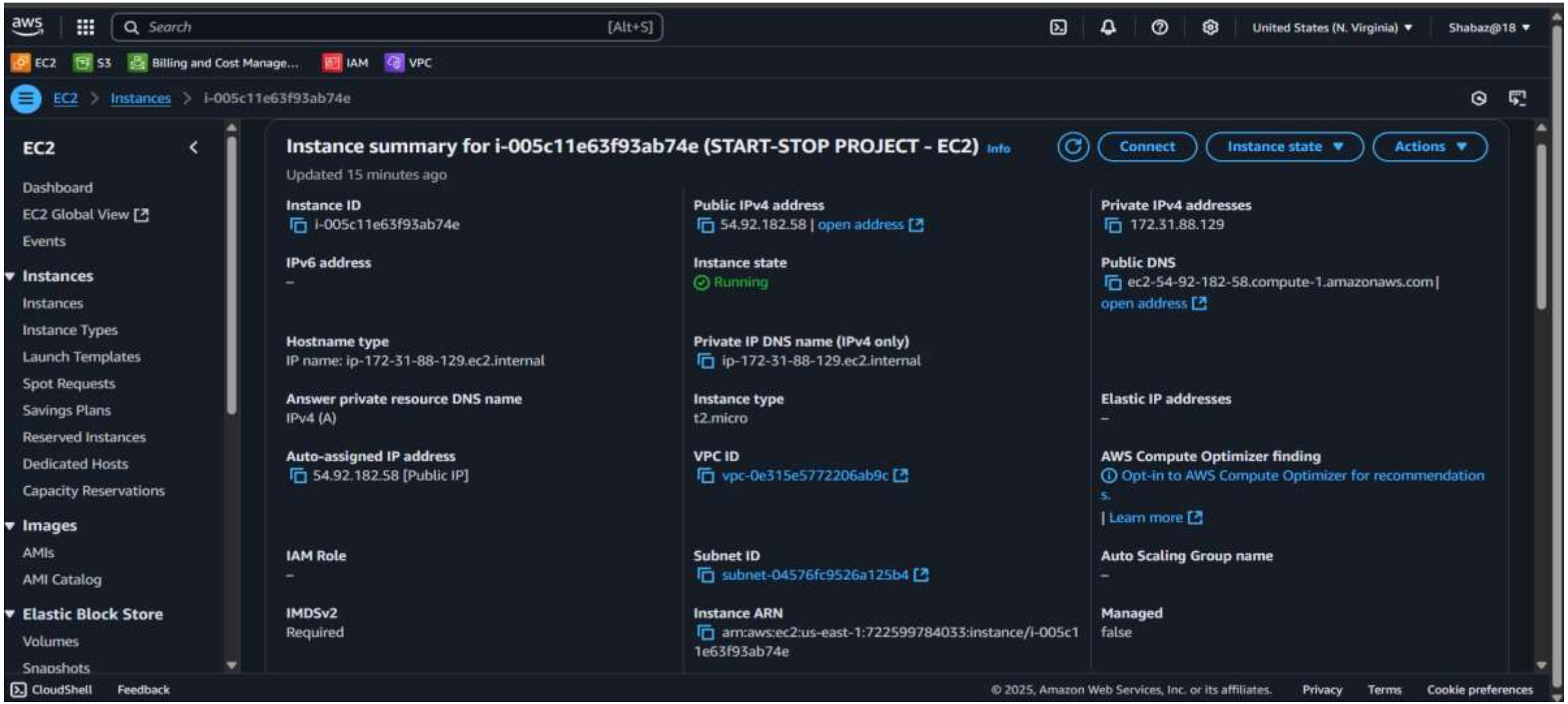
IMPLEMENTATION



>>> **LAUNCH AN EC2 INSTANCE**

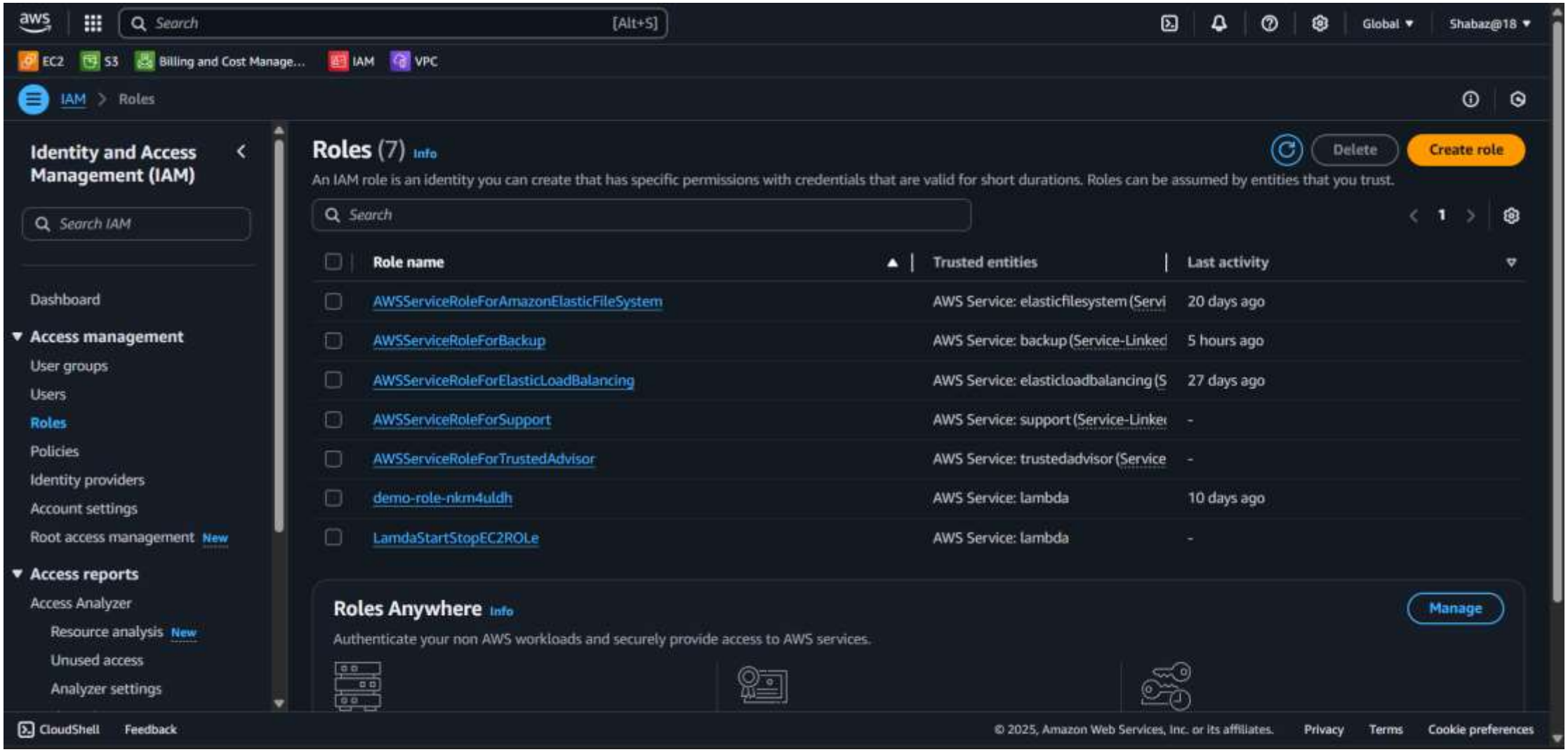
>>> With the name of – Start Stop Project – EC2 (whatever you want)

- With these EC2 Details

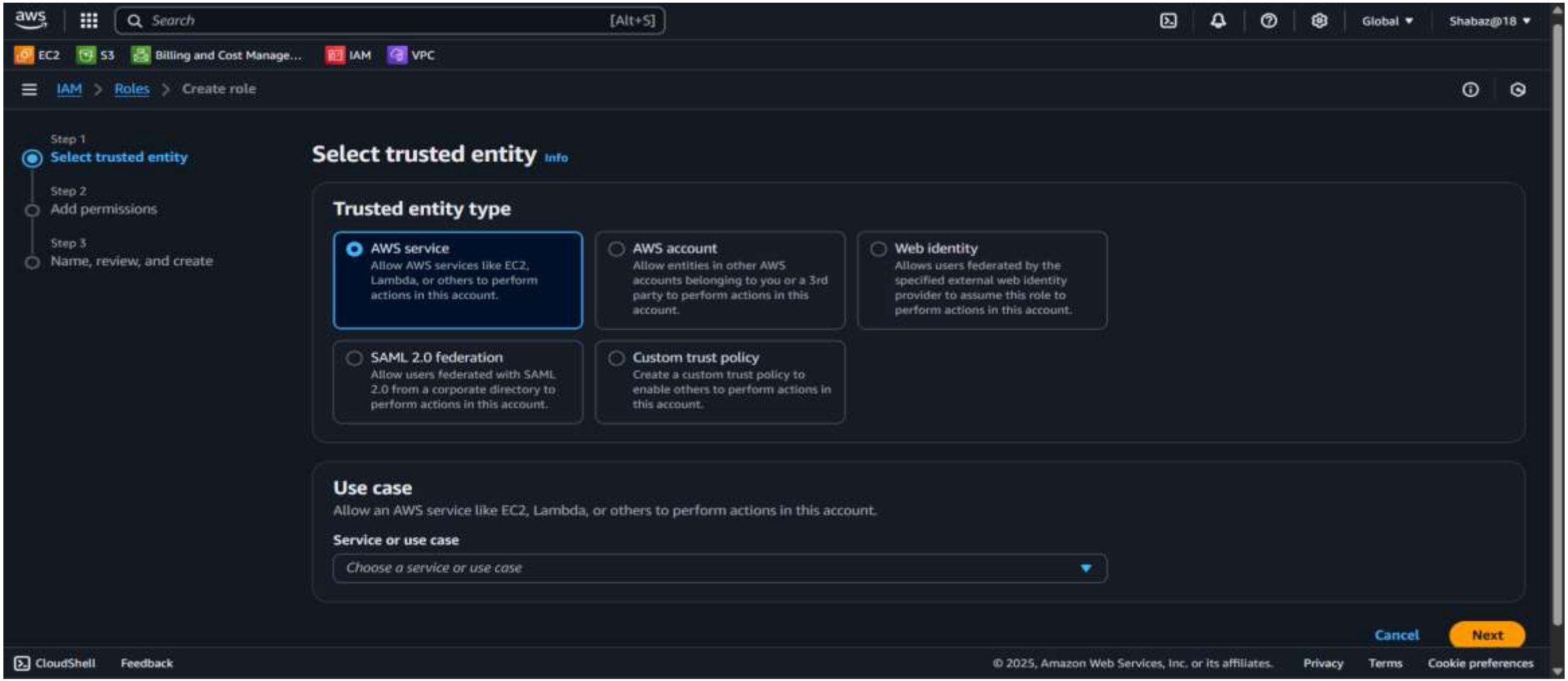


>>> CREATE AN IAM ROLE for LAMBDA

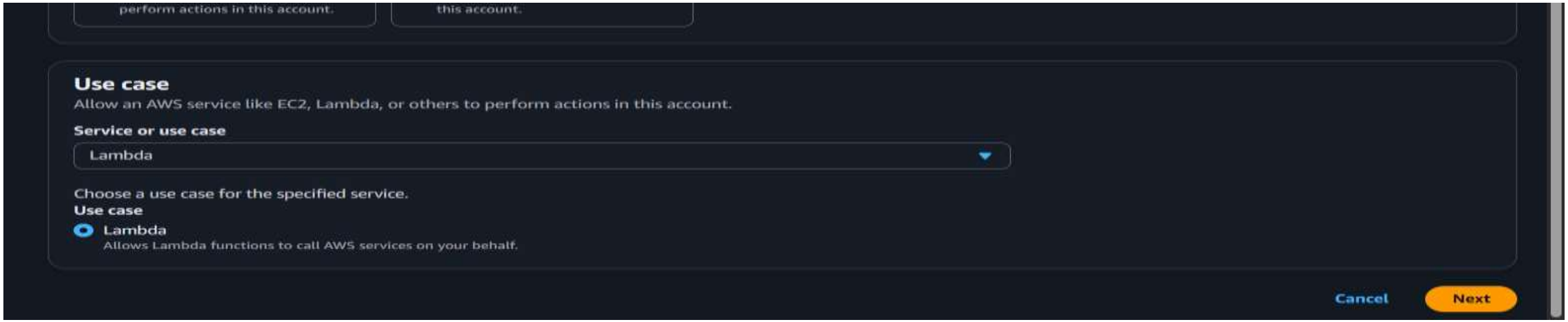
1. Here remember one thing - your Lambda Functions need permission to manage EC2 instances
2. This is why we attach EC2-related policies to the IAM Role assigned to the Lambda.“



- Process to create IAM ROLE --- >>> STEP : 1
    1. Search for IAM in the search bar , Open it
    2. Click on “ROLES” , Then Click on <u>Create role</u>

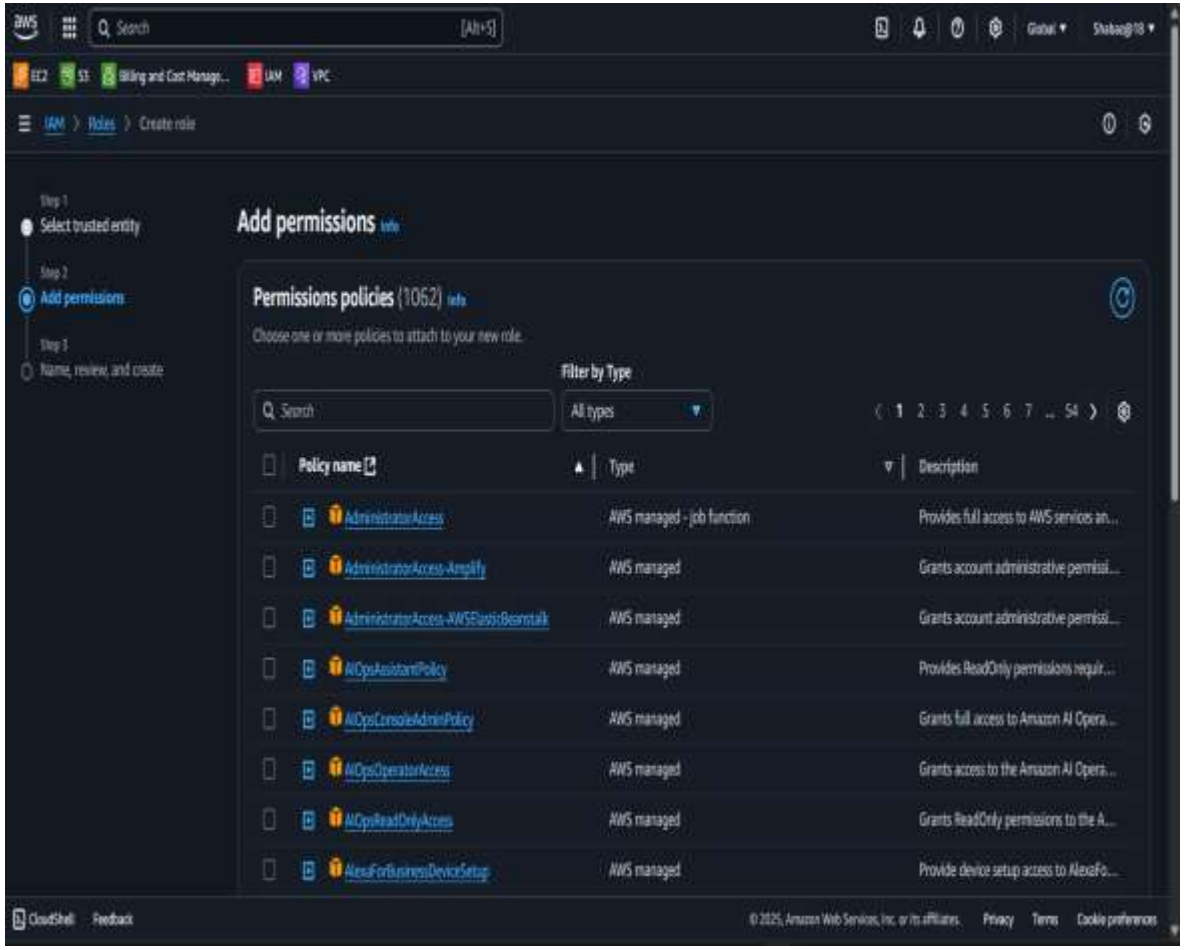


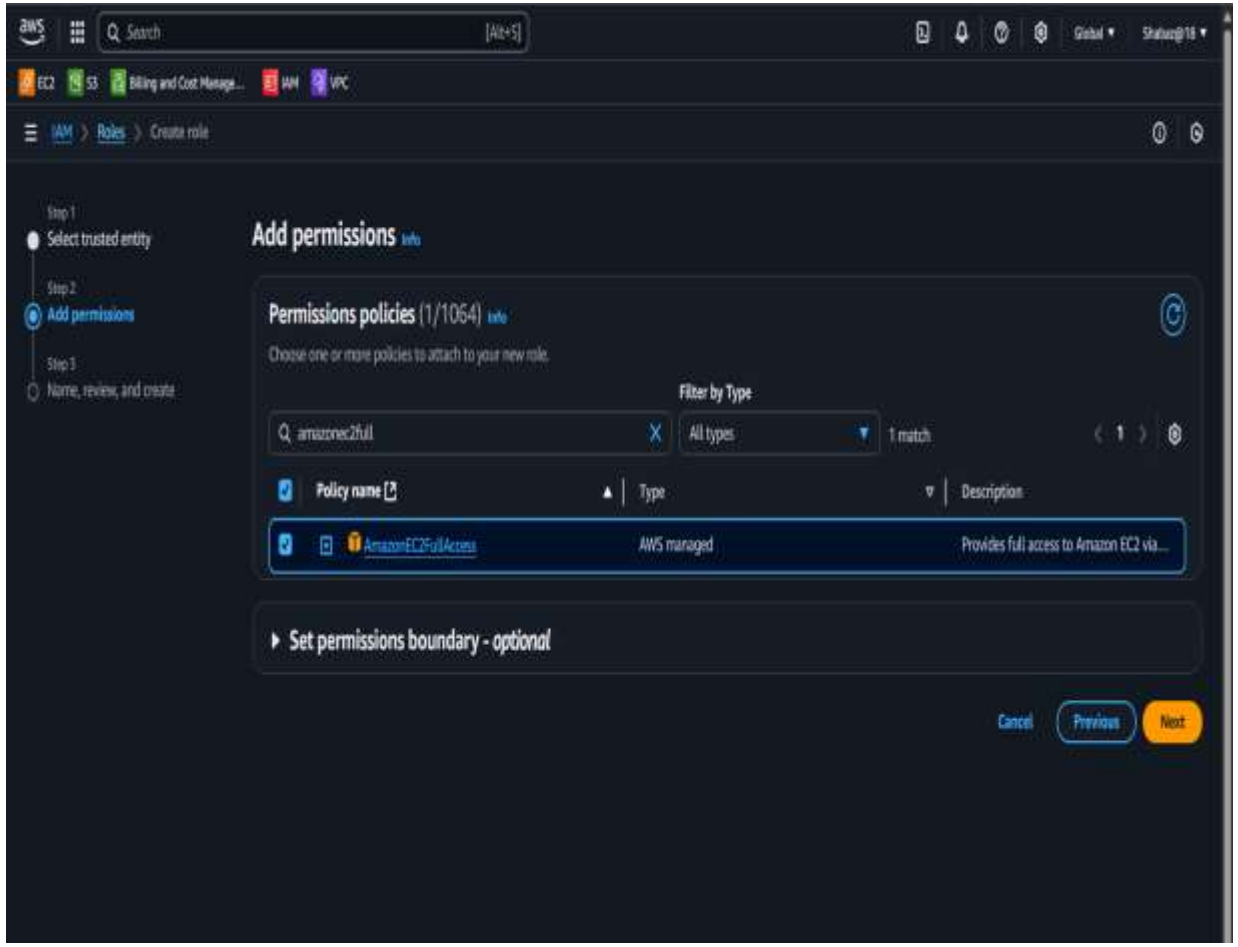
- Select AWS Service at Trusted Entity Type
- At USE CASE, search for LAMBDA & Select it & Click on Next



>>> STEP : 2

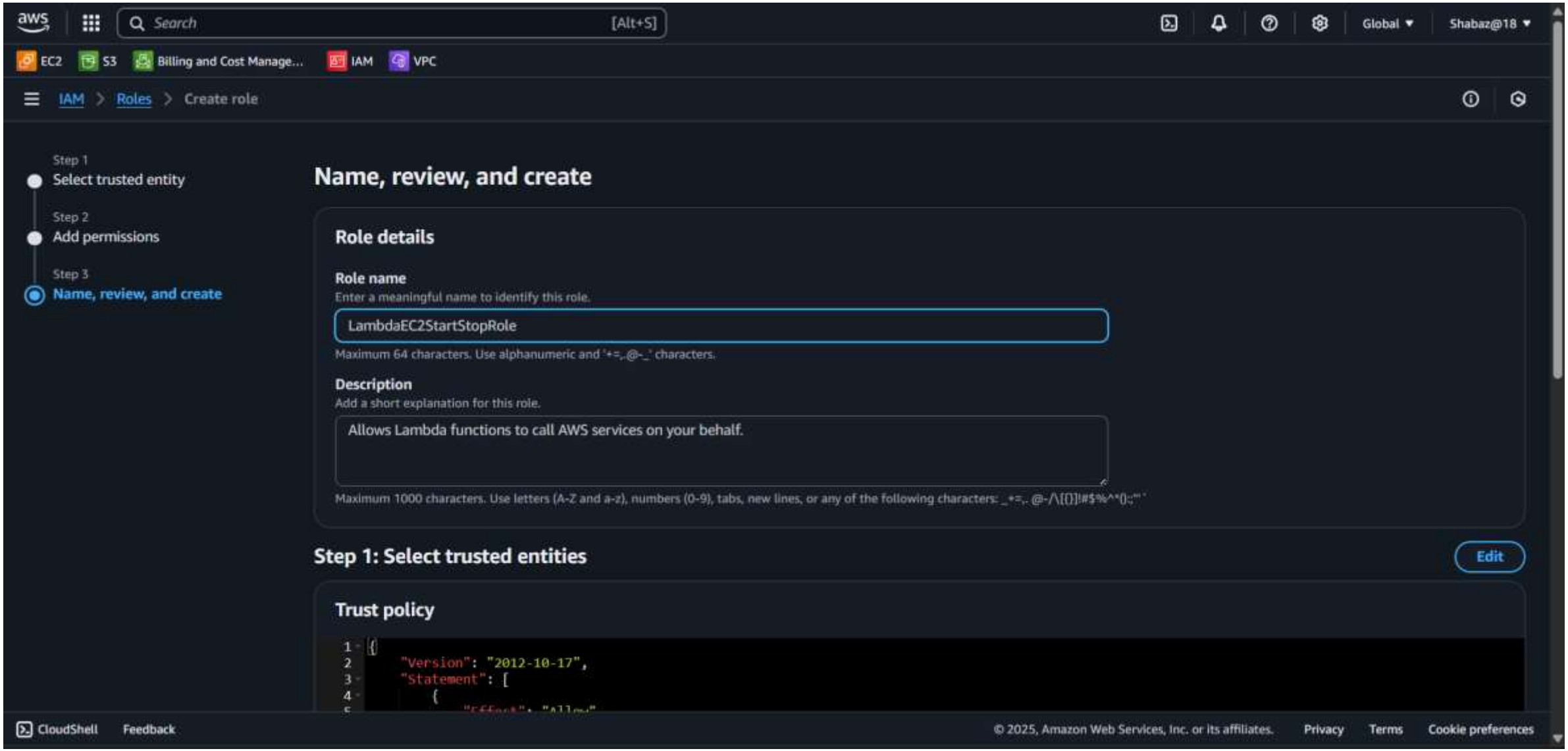
- Now we have to add the AmazonEC2FullAcess Permission at ADD PERMISSION section
- Search for AmazonEC2FullAcess & select it,Scroll down & click on next



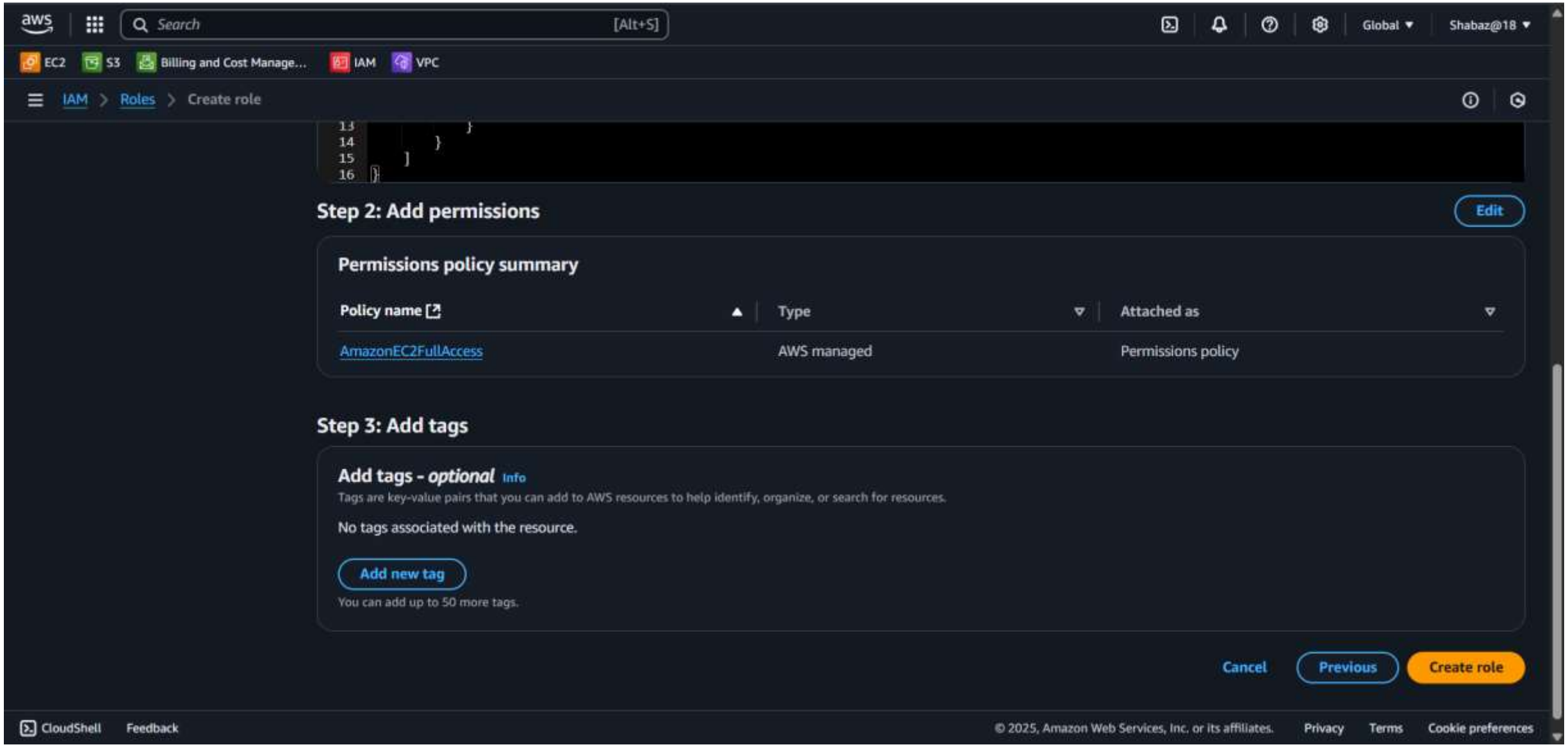


>>> STEP : 3

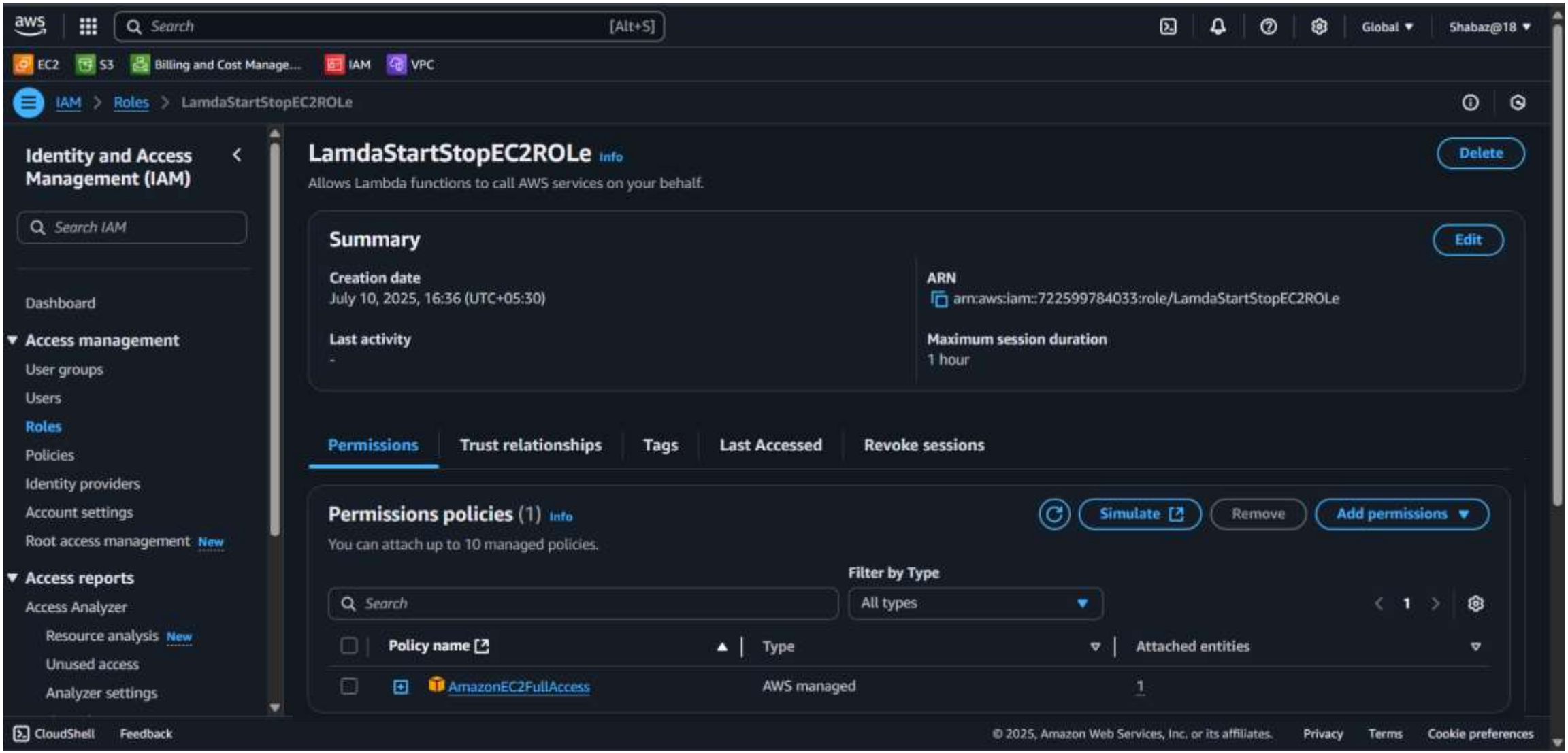
- Give ROLE NAME like : LambdaEC2startstopROLE (whatever you want)



- Scroll Down and Click on <u>CREATE ROLE</u>

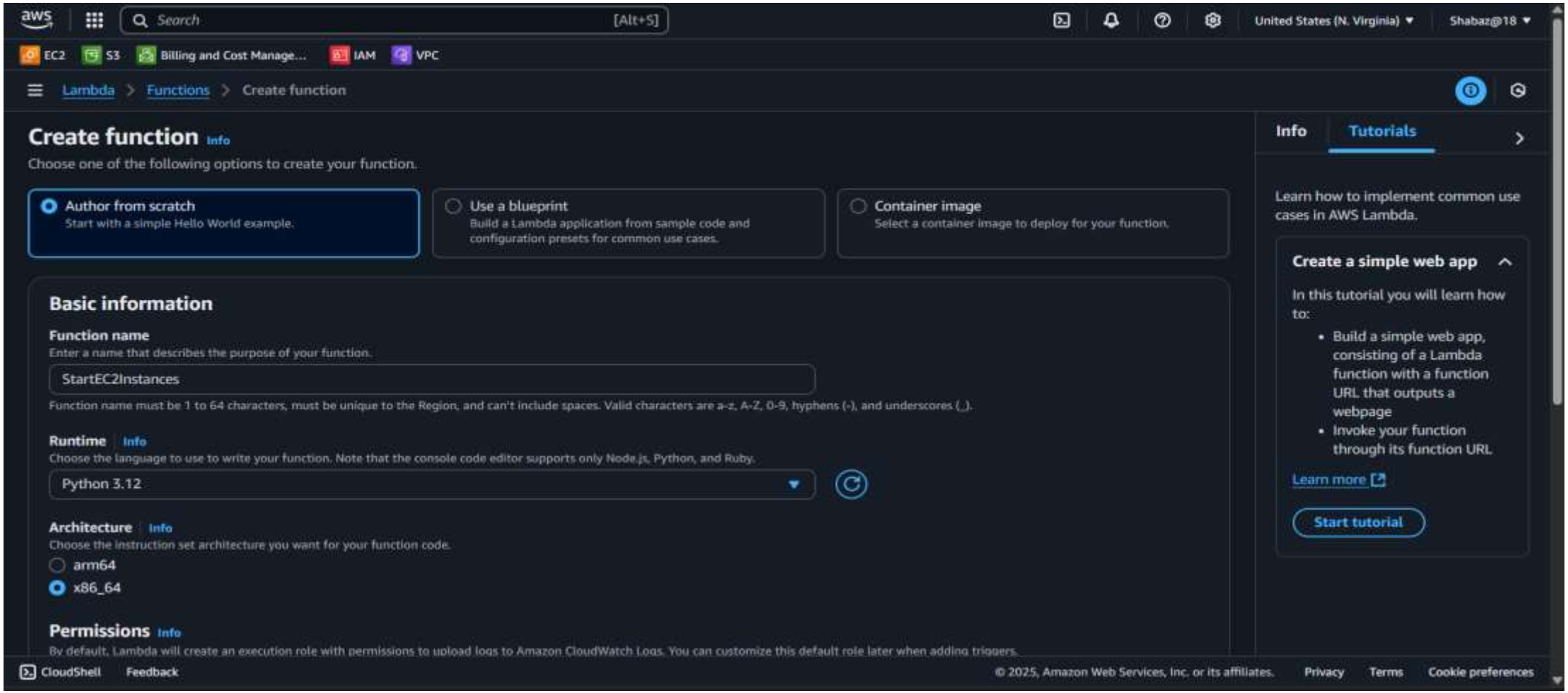


- You can see your created IAM ROLE by clicking on Roles and click on your customized name to see in detail summary

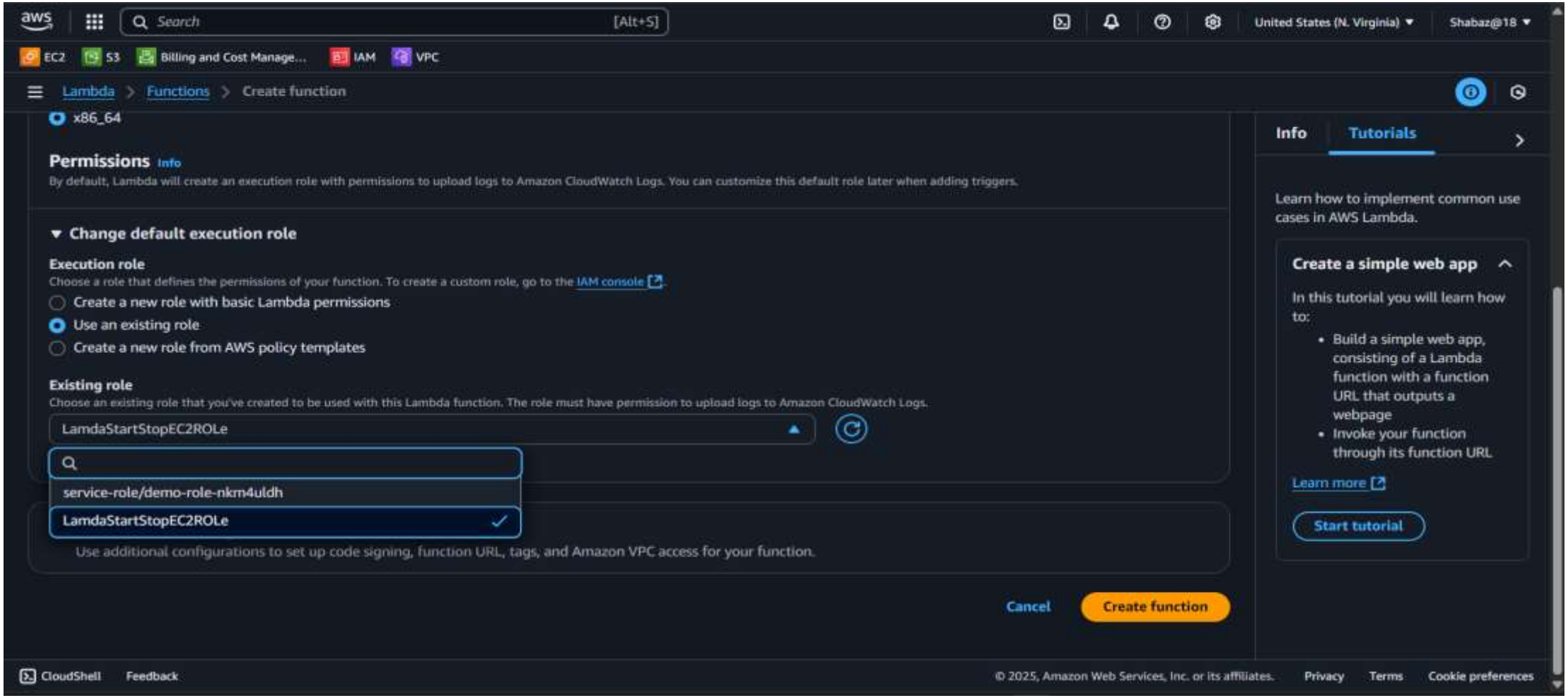


>>> Next , **CREATE THE START LAMBDA FUNCTION**

- Search for LAMBDA in the search bar, click on create Function
- Give name as StartEC2Function & Select RUNTIME : PYTHON 3.12



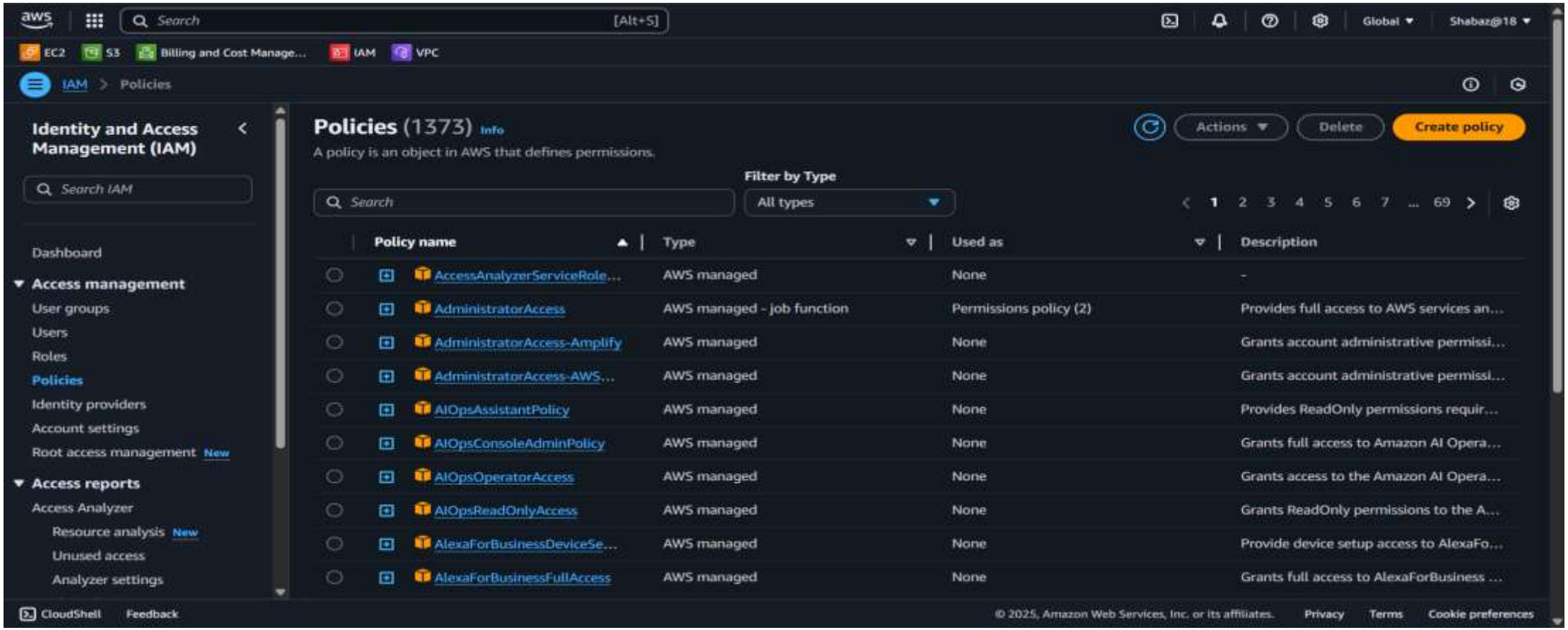
- Scroll Down and come to <u>EXECUTIVE ROLE</u>
- Here Choose – Use an Existing role
- Next, Select your Created IAM ROLE & click on Create Function

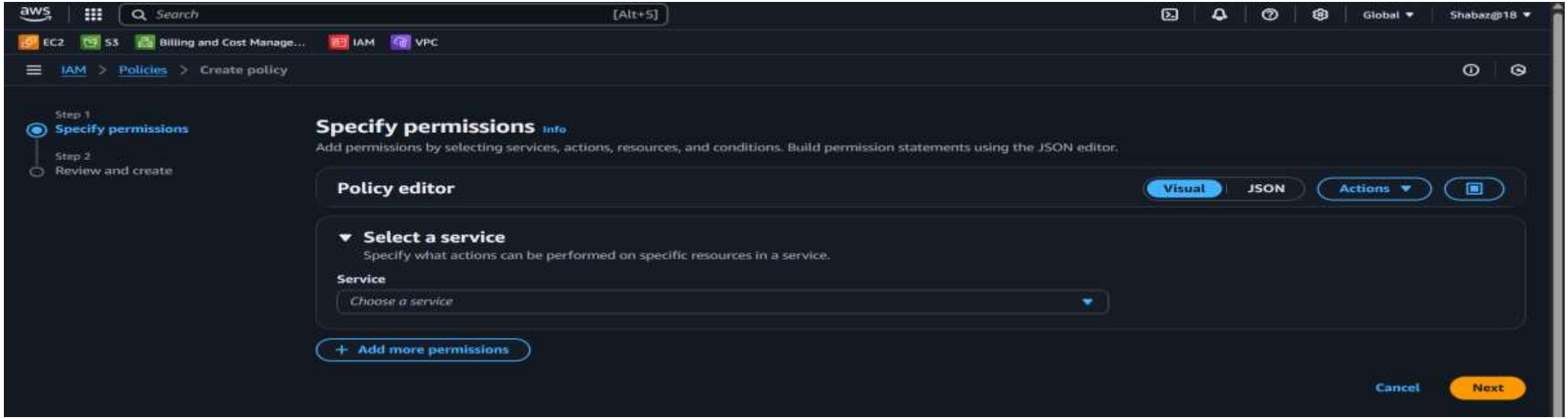


NEXT ,

>>> **<u>CREATE POLICIES</u>**

1. Go to Search bar & search for IAM
2. Select Policies and Click on Create Policies. Here, at Step 1 : You have to Specify the Permission

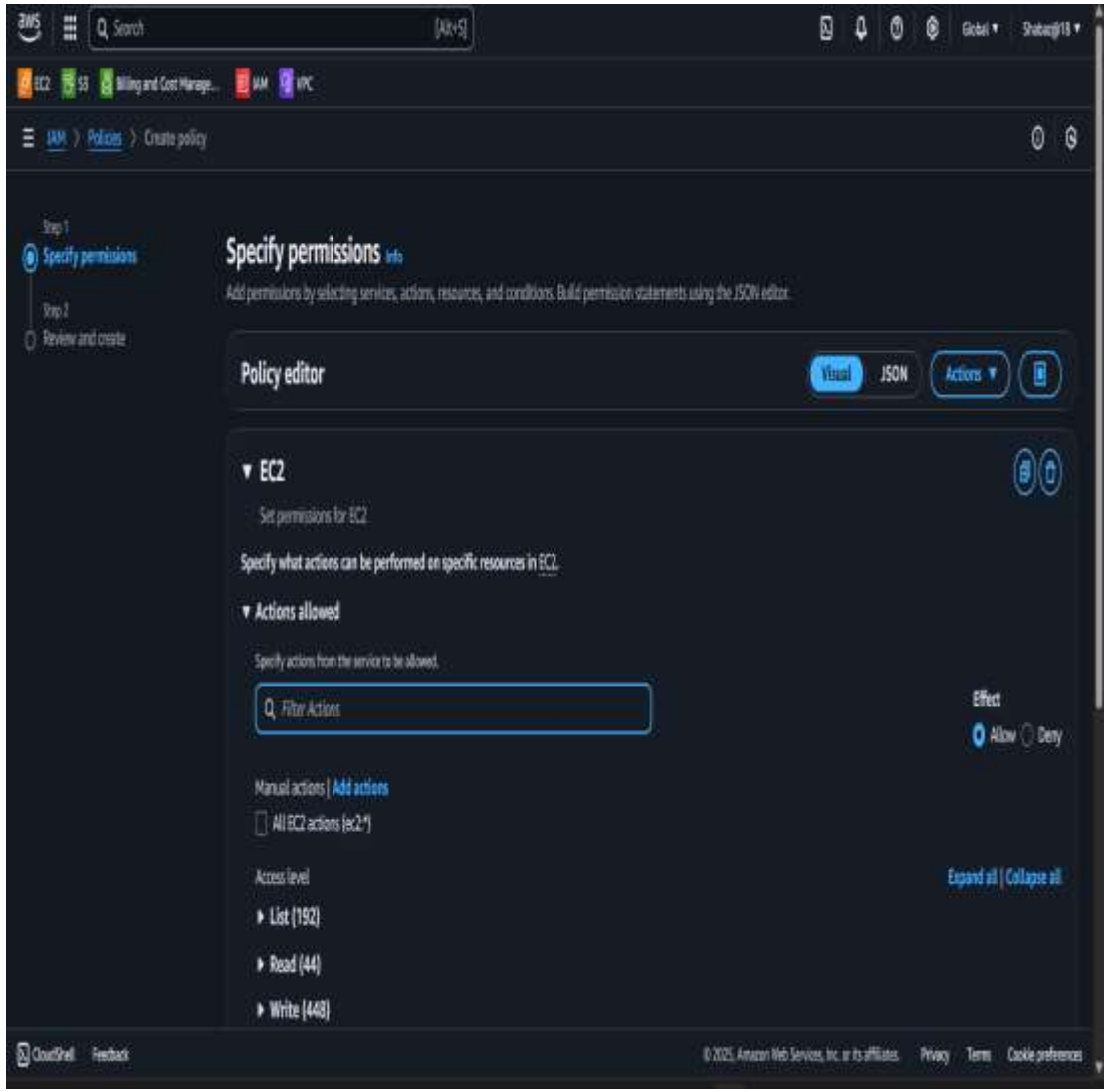


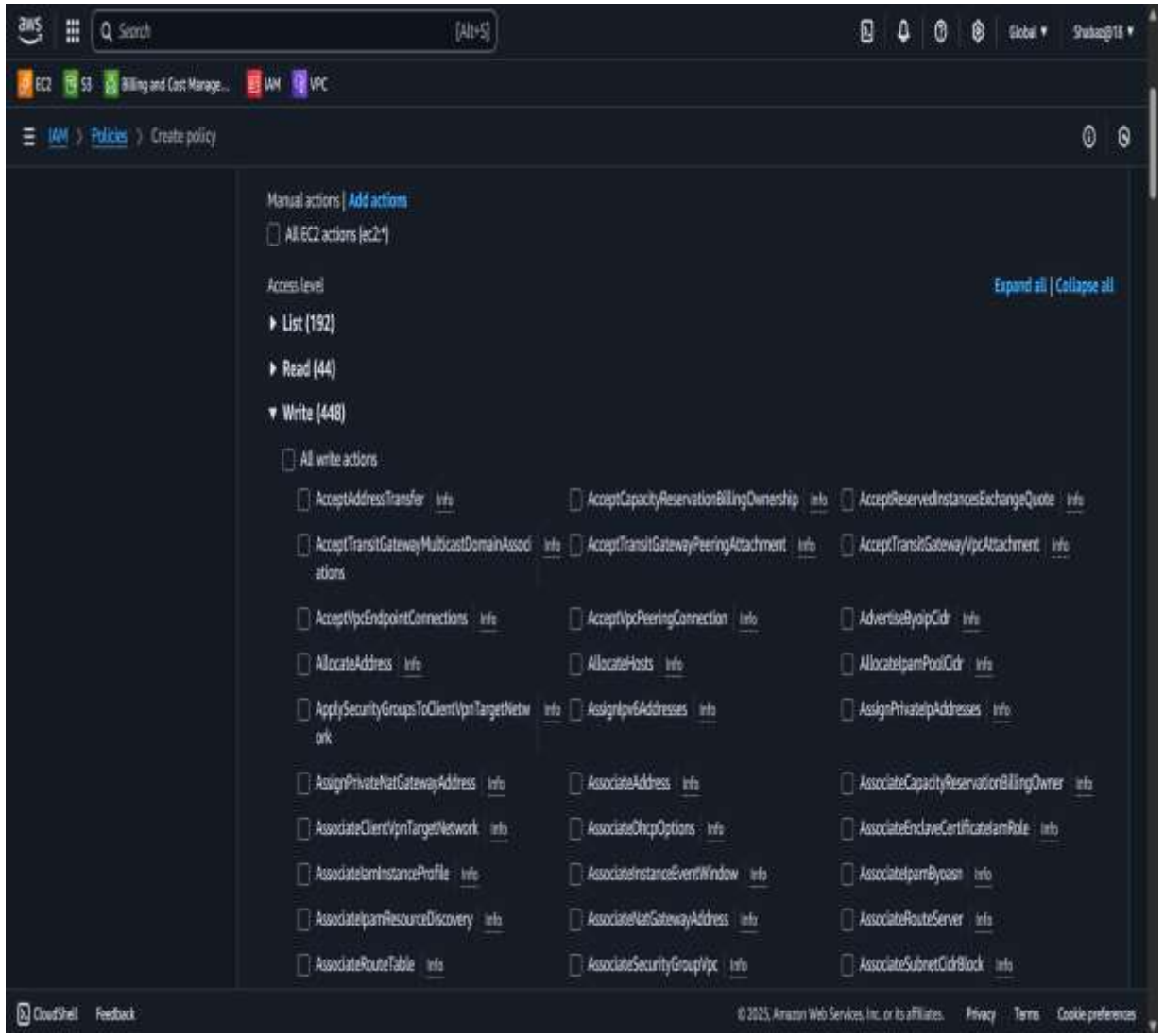

aws
Search
[Alt+S]
Global
Shabaz@18
EC2
S3
Billing and Cost Manage...
IAM
VPC
IAM > Policies > Create policy
Step 1
Specify permissions
Step 2
Review and create
Specify permissions Info
Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.
Policy editor
Visual
JSON
Actions
Select a service
Specify what actions can be performed on specific resources in a service.
Service
Choose a service
+ Add more permissions
Cancel
Next

- Select EC2

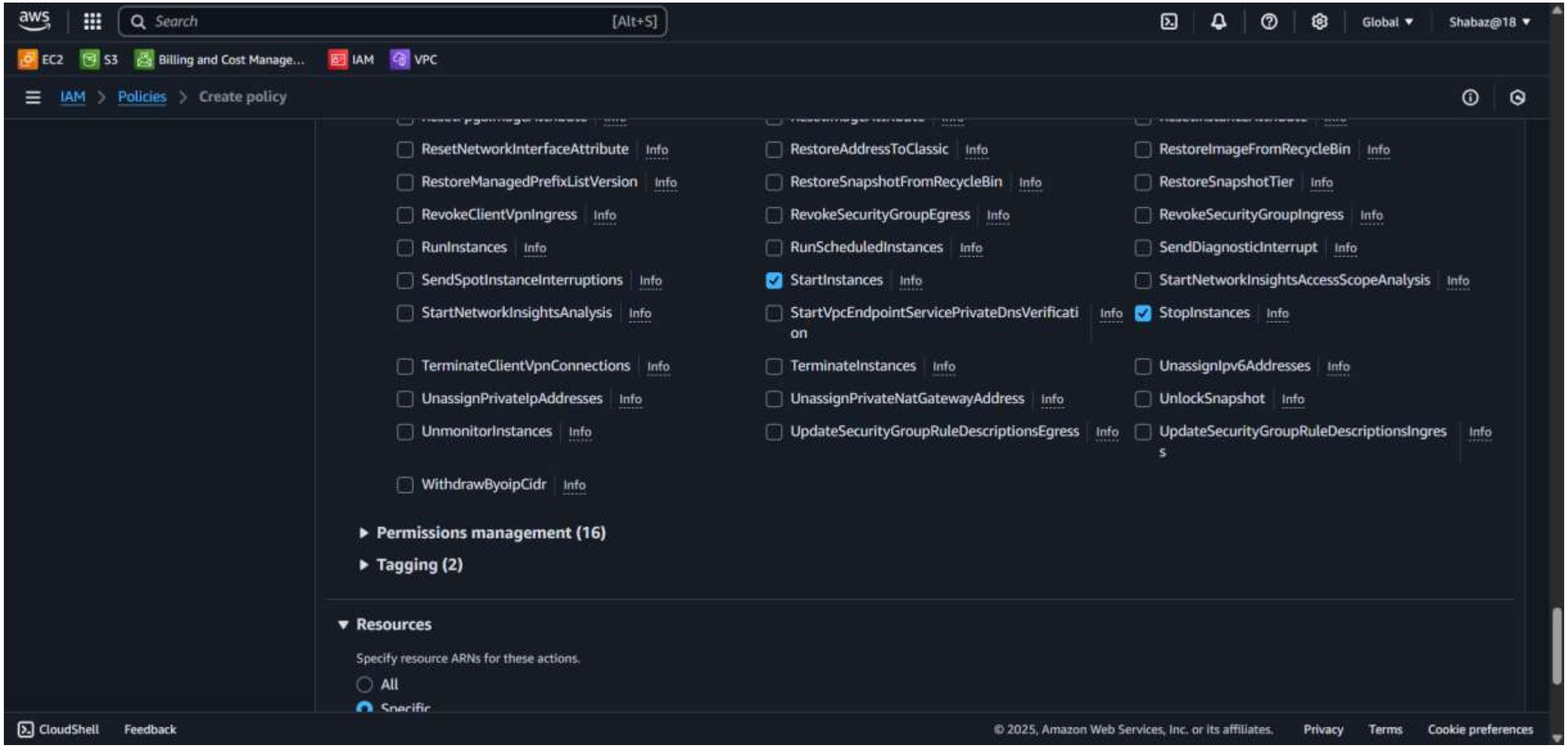

aws
Search
[Alt+S]
Global
Shabaz@18
EC2
S3
Billing and Cost Manage...
IAM > Policies > Create policy
Filter services
Commonly used services
Auto Scaling
CloudFront
EC2
IAM
Lambda
RDS
S3
SNS
Other services
Step 1
Specify permissions
Step 2
Review and create
Visual
JSON
Actions
Choose a service
+ Add more permissions
Cancel
Next

- After Selecting EC2 & Click on WRITE -- It will expand

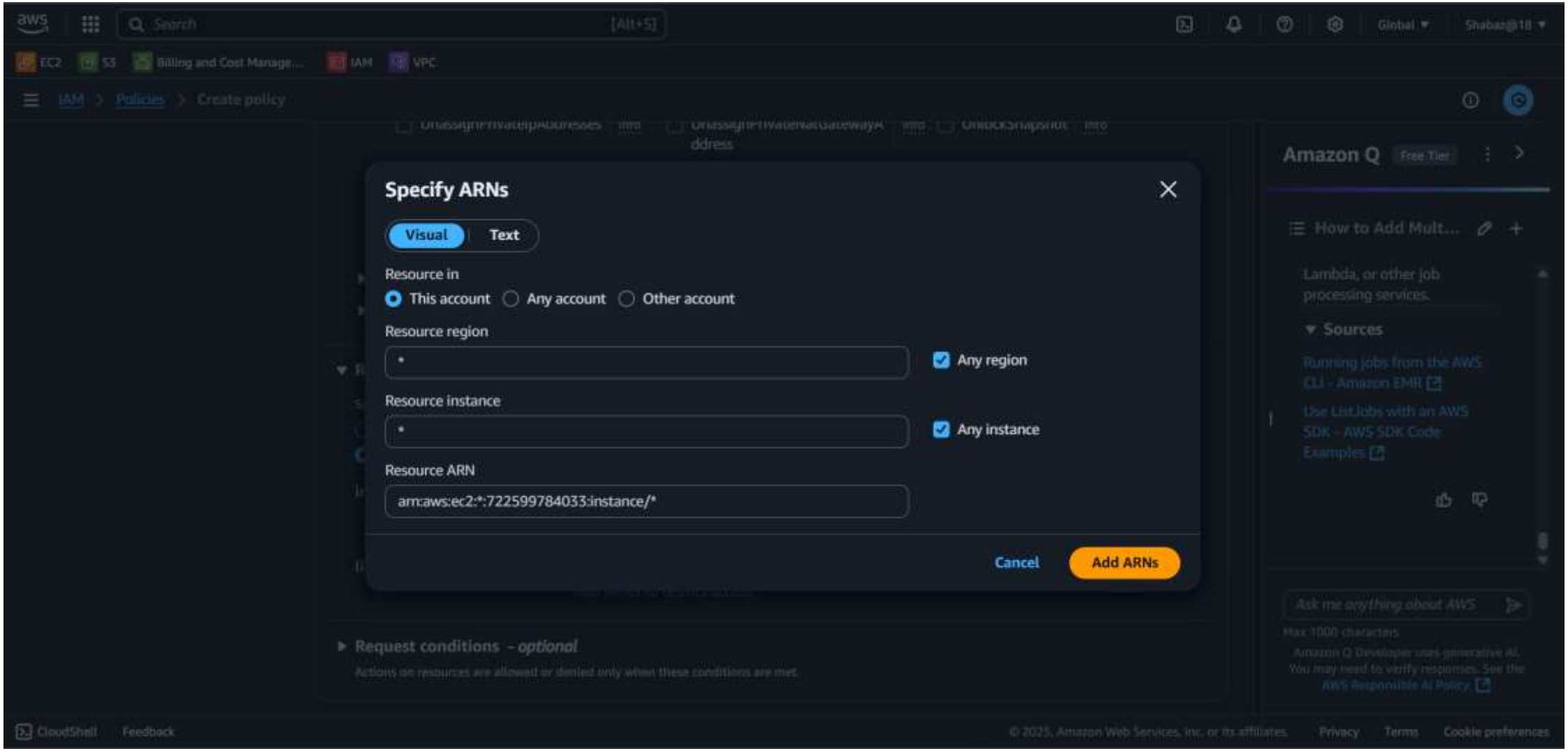




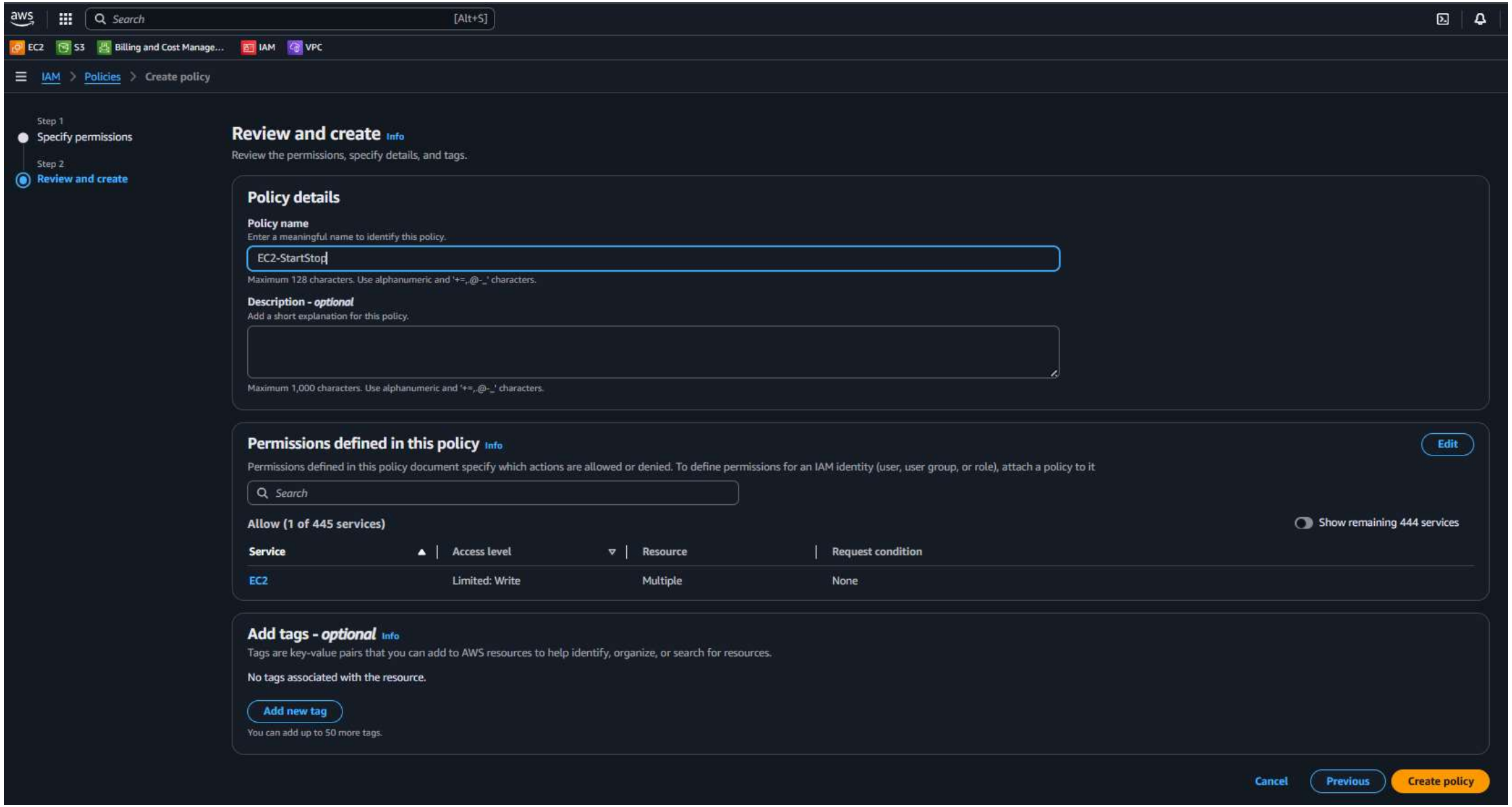
- Here Select StartInstance & Stopinstance



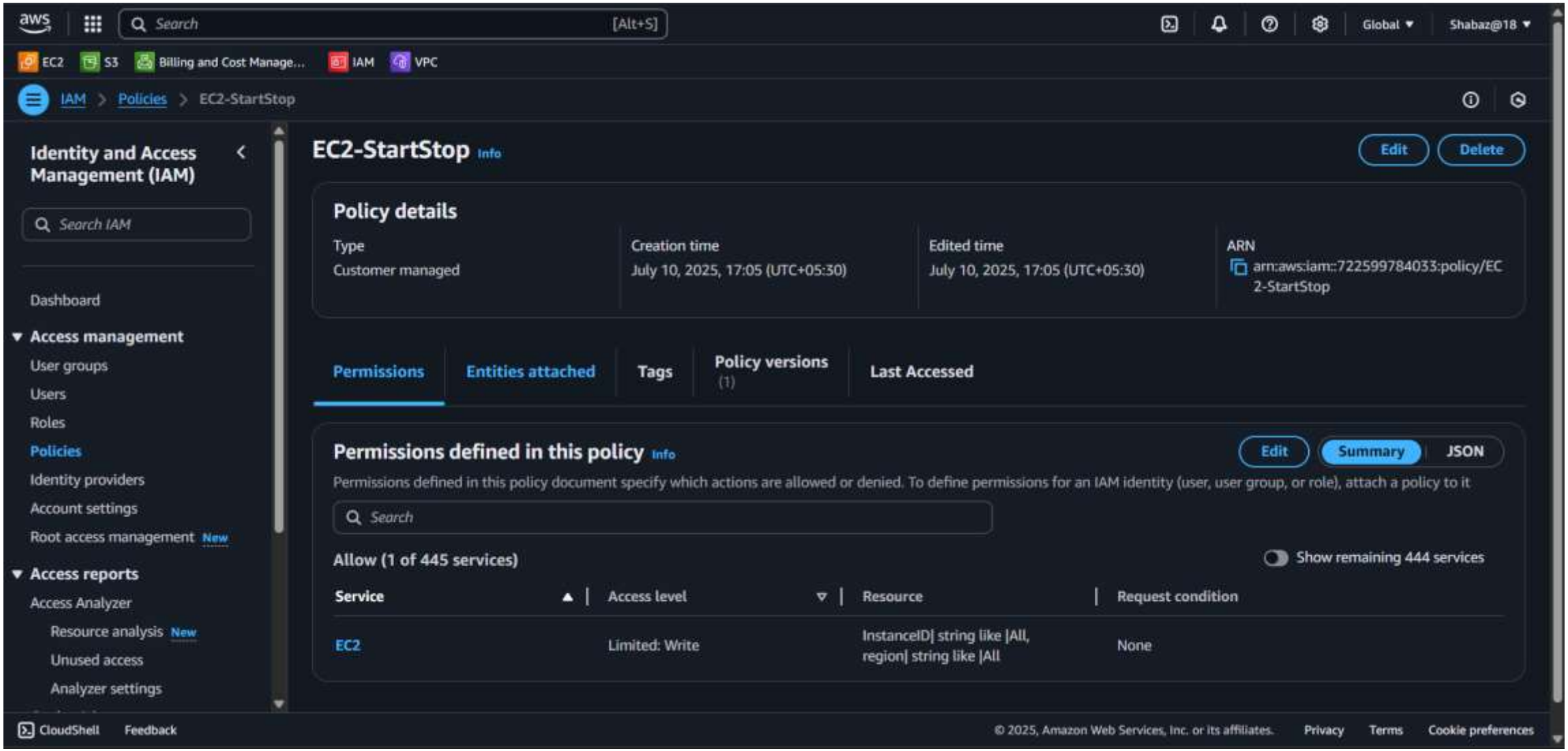
- Scroll Down & click on Add ARN, Give tick mark for both Resources and Click on Add ARN , Do the same thing to second ARN also & Then click on next



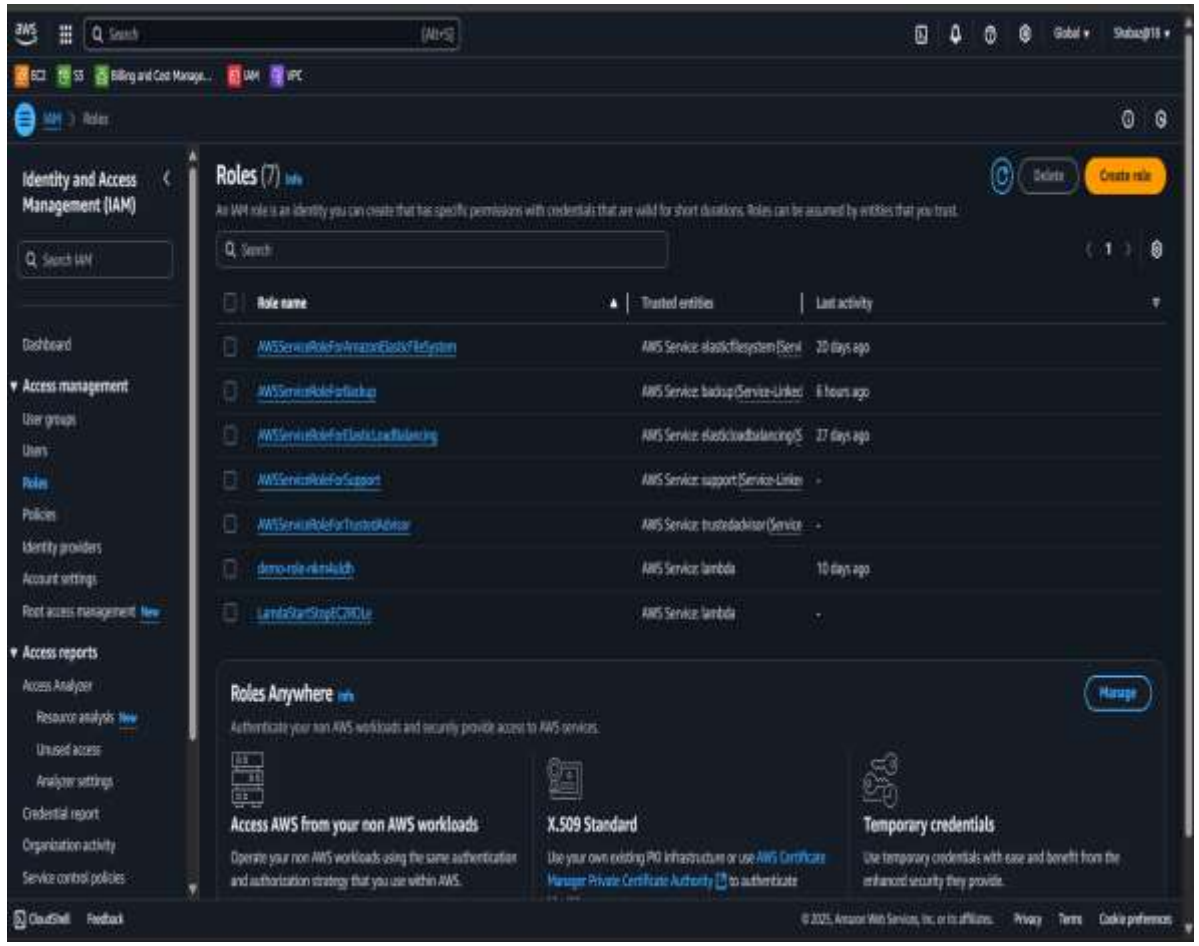
- Here, at Step 2 : you have to give Policy name as EC2-StartStop & click on Create Policy

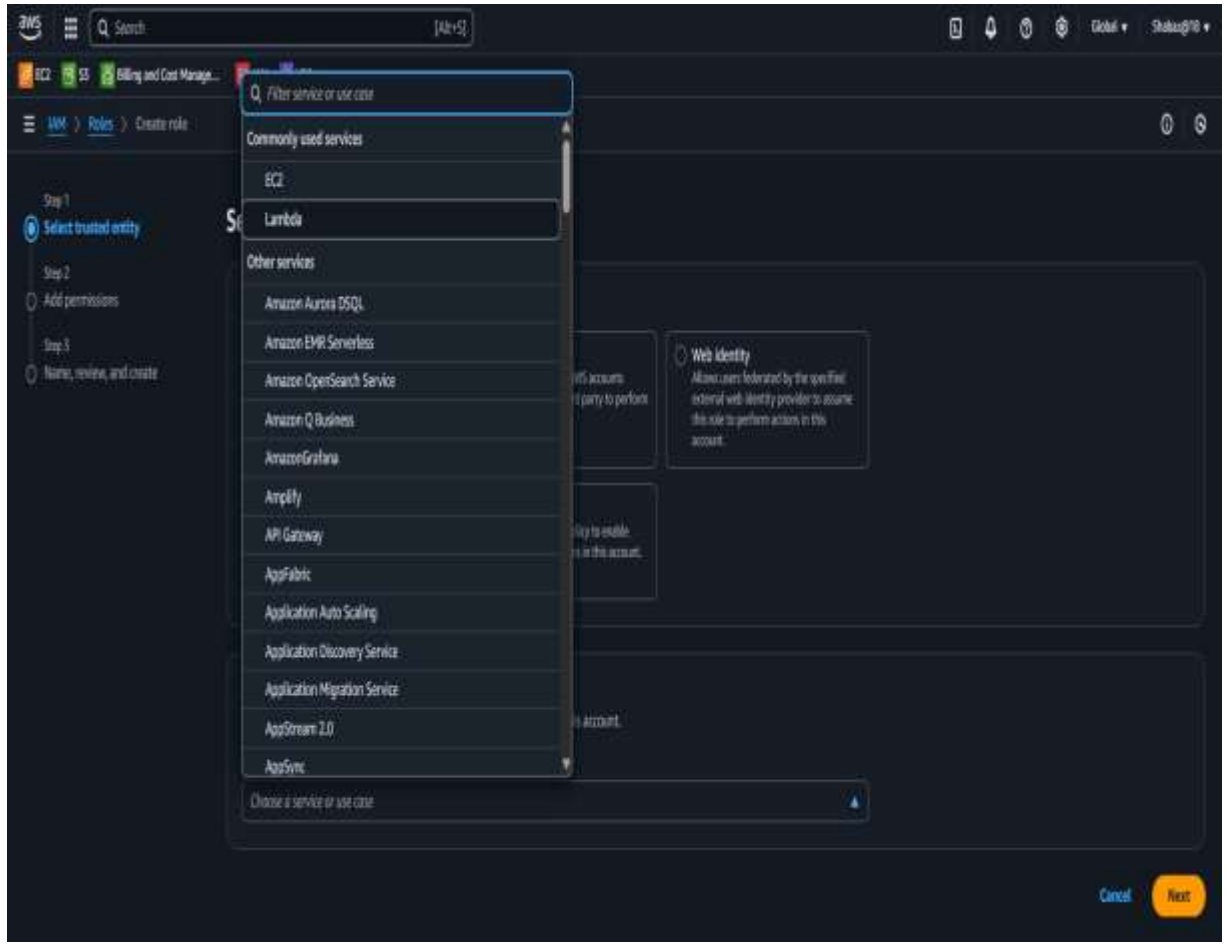


- Here, At (IAM > Policies) you can see, your Created Policies by clicking on the created Policy name



- Now, we have to Create <u>STOP LAMBDA FUNCTION</u>
- Repeat the same Process again with little bit changes, create an IAM ROLE as it is like :

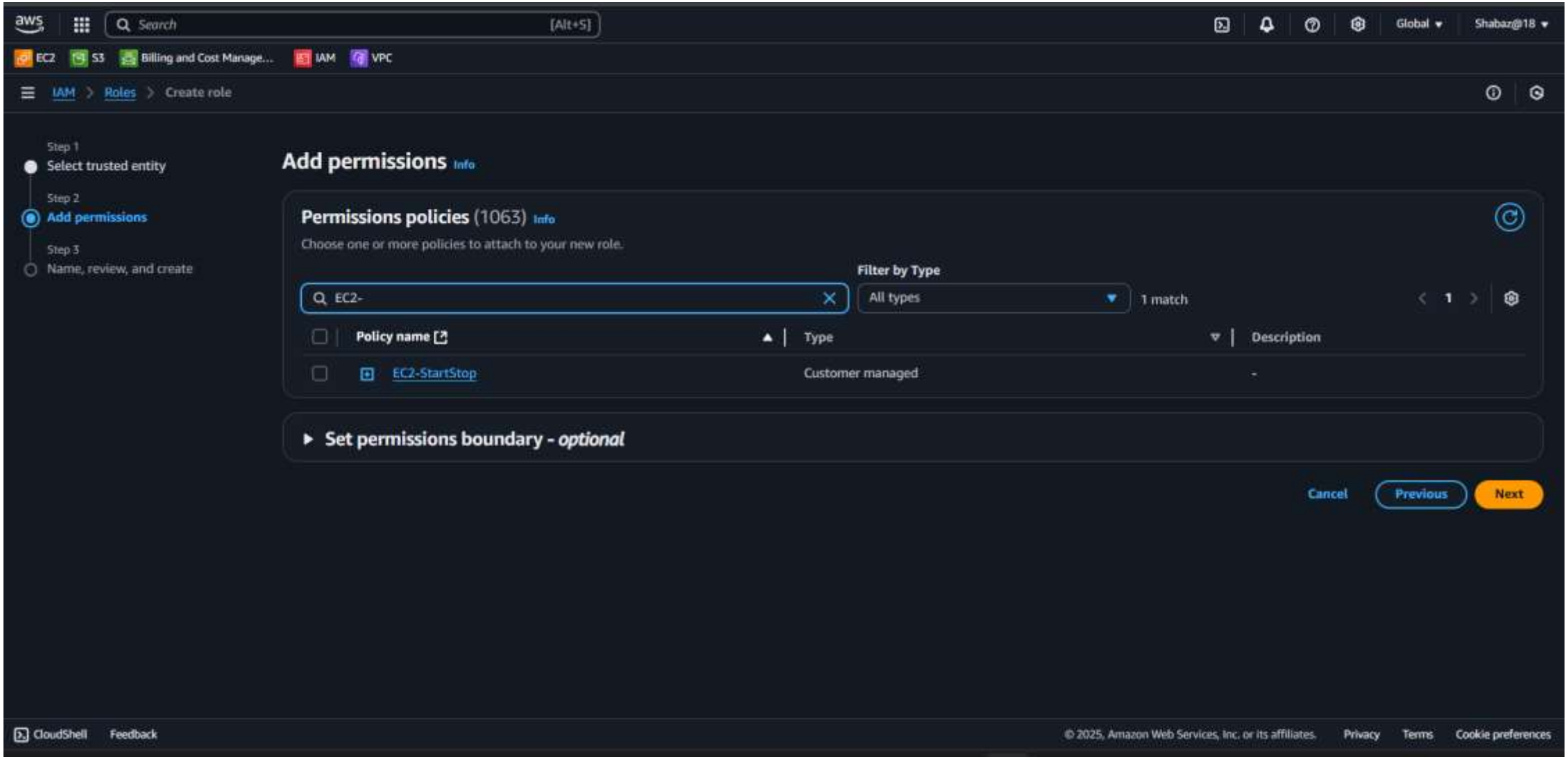




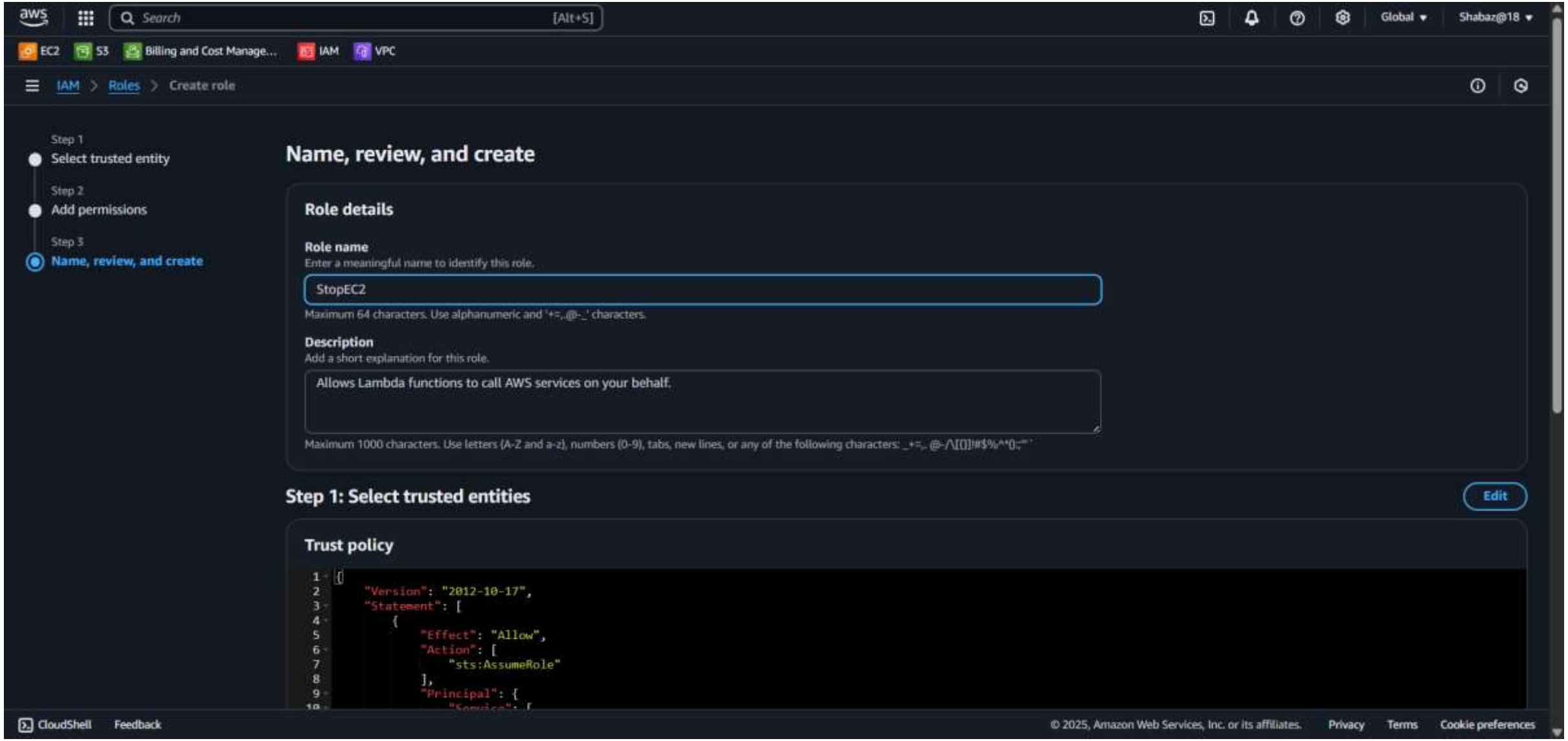
1. Click on CREATE ROLE , Click on next

2. Select Amazon service , Select Lambda at the USE CASE

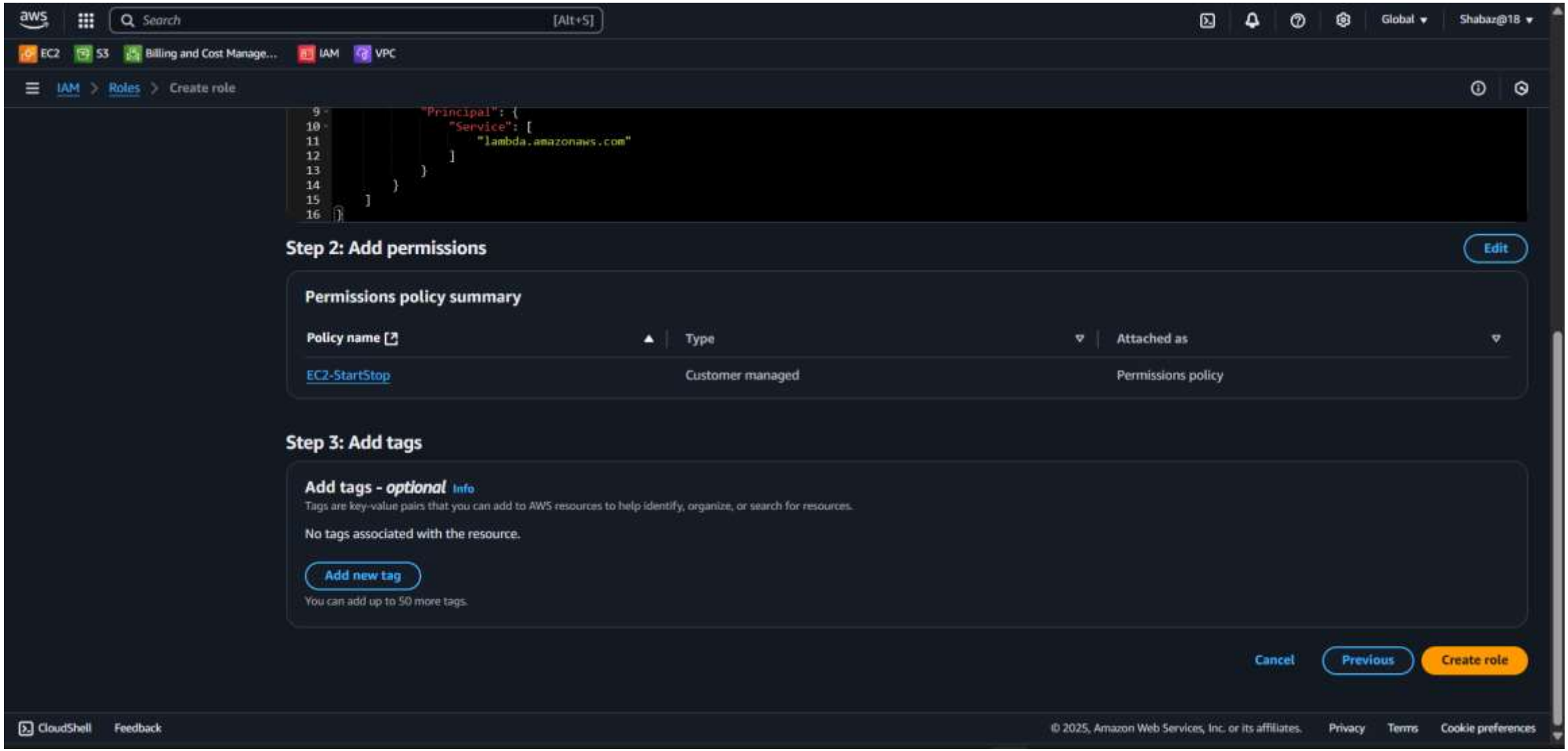
3. At ADD permission policies section search for your created policy and select it, then click on Next



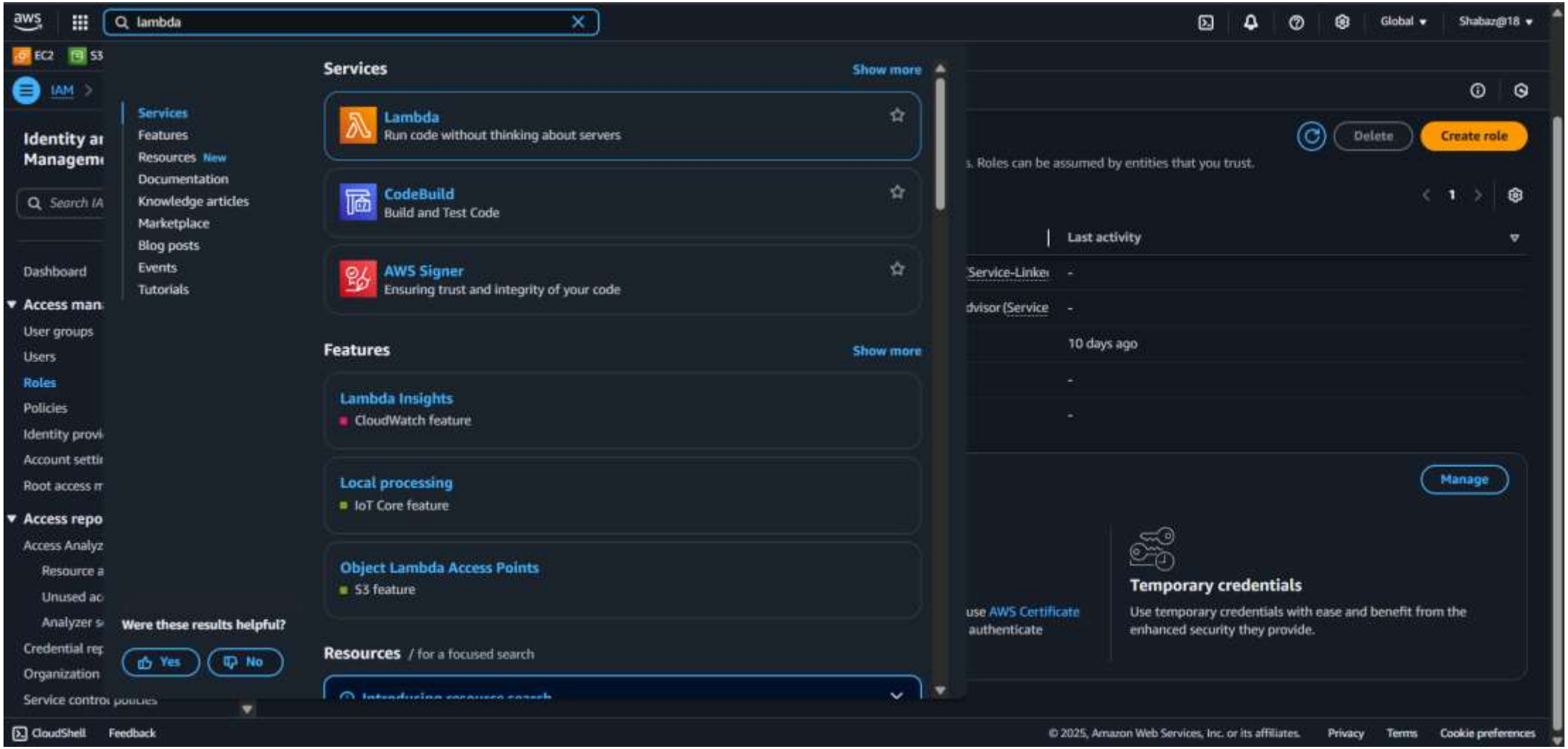
- Next, At the ROLE DETAILS Give role name as STOP EC2 and at Description add a short explanation like : Processing Service (like that)



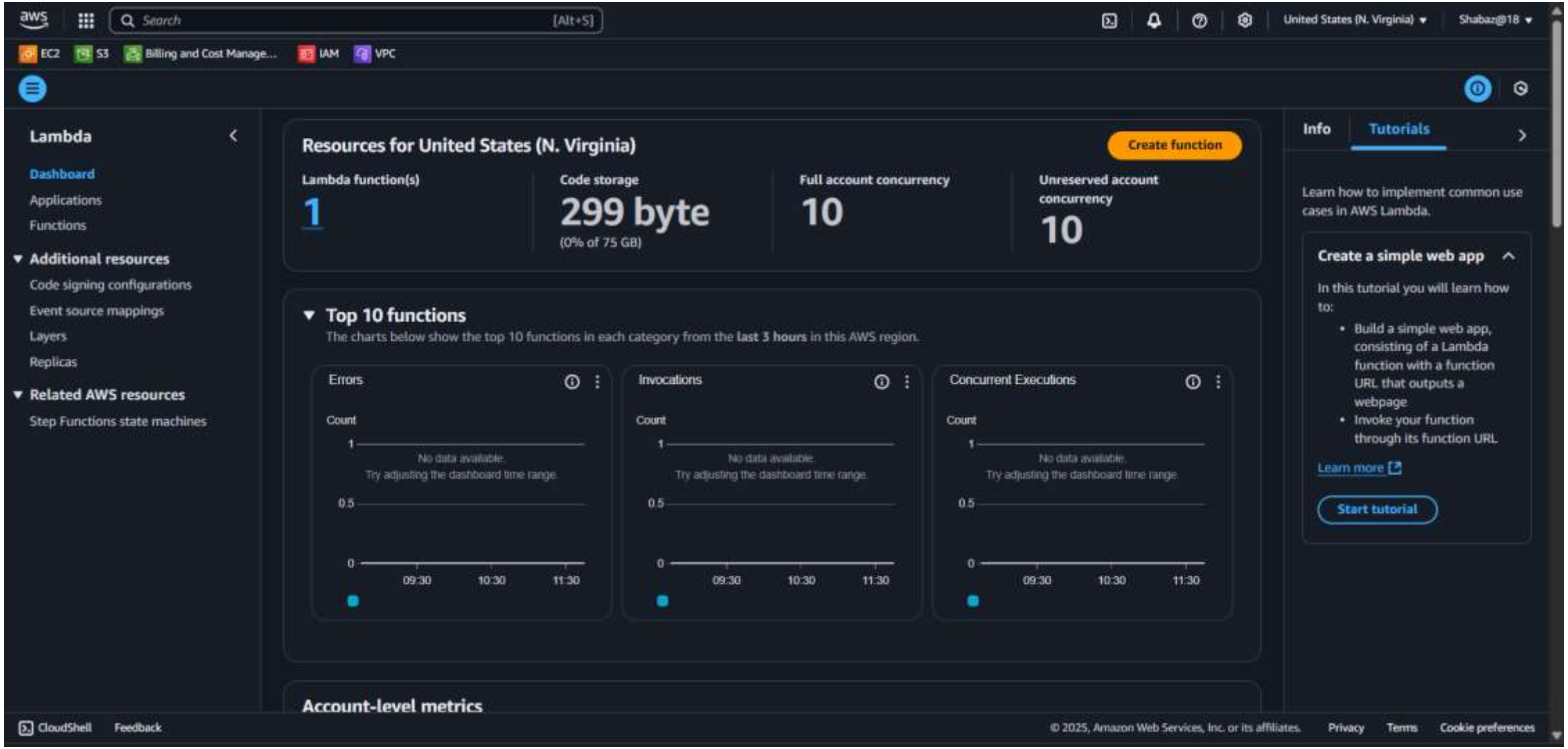
- The Display will be Appear like below shown figure, Then Click on <u>CREATE ROLE</u>



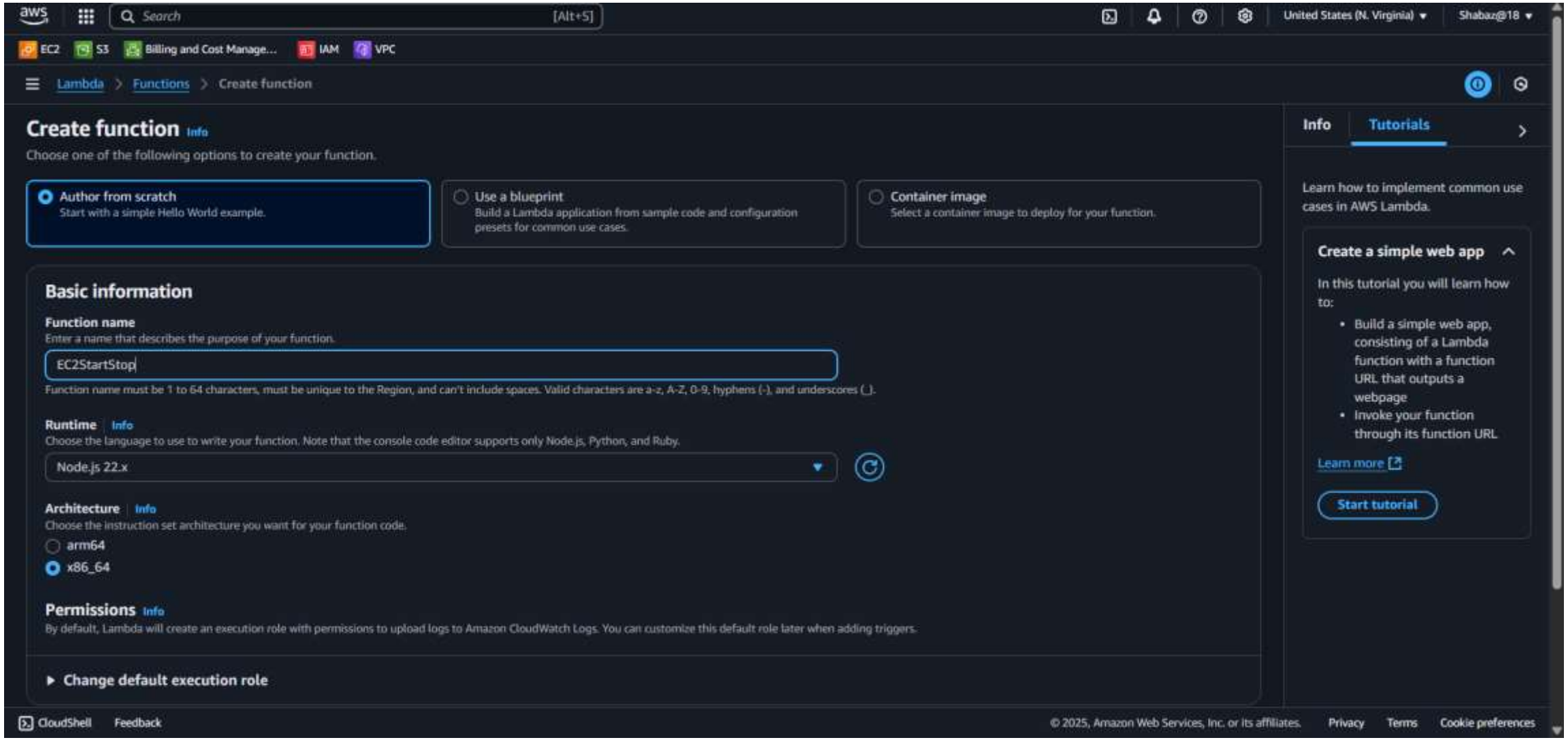
- After That Create a <u>LAMBDA Function</u>
- At the search bar, Search Lambda open it



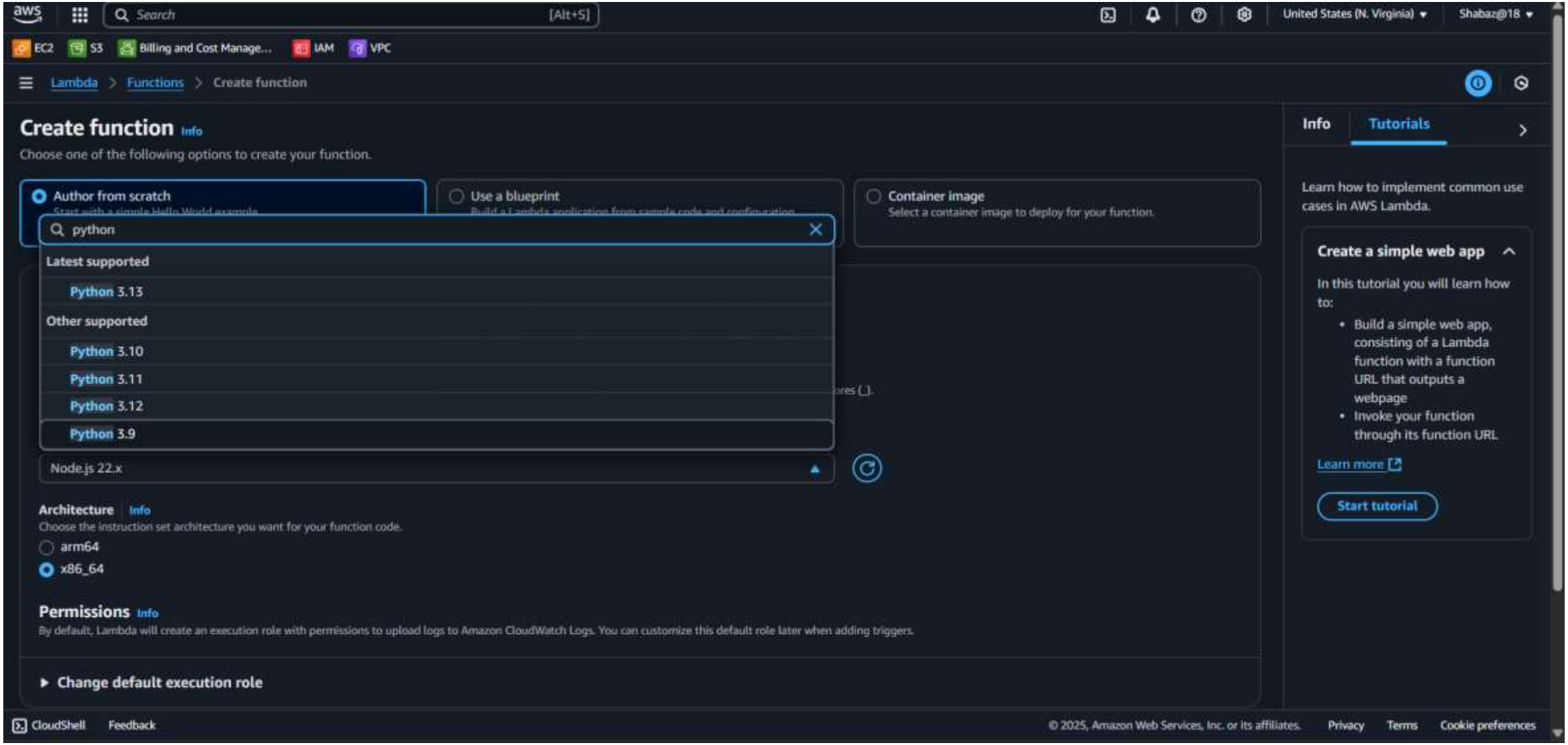
- Here, If you click on DASHBOARD , You can see the how many Functions we created till now
- We have created 1 Already & Now we are going to create another
- Click on CREATE FUNCTION



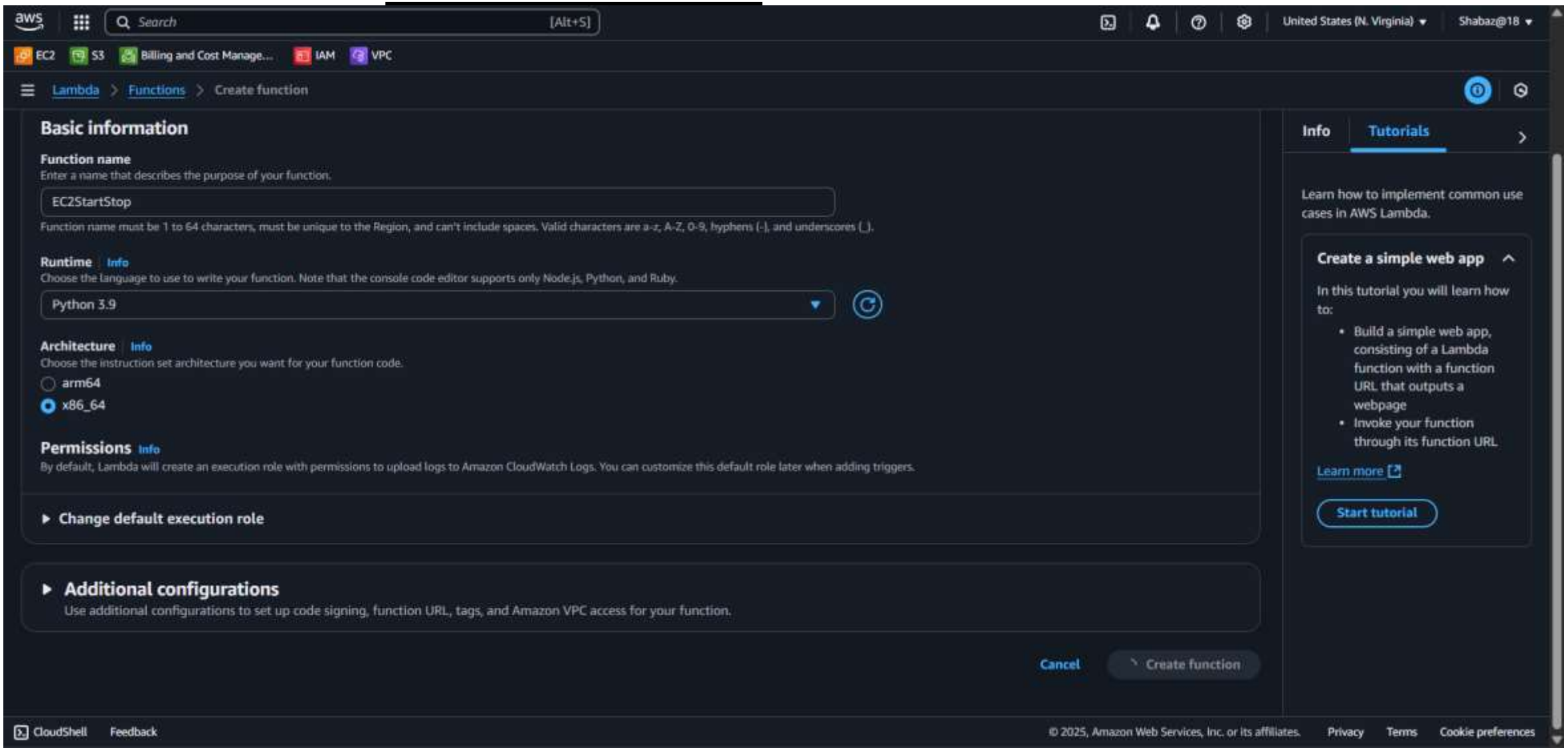
- Give Basic information like : Function name as EC2StartStop



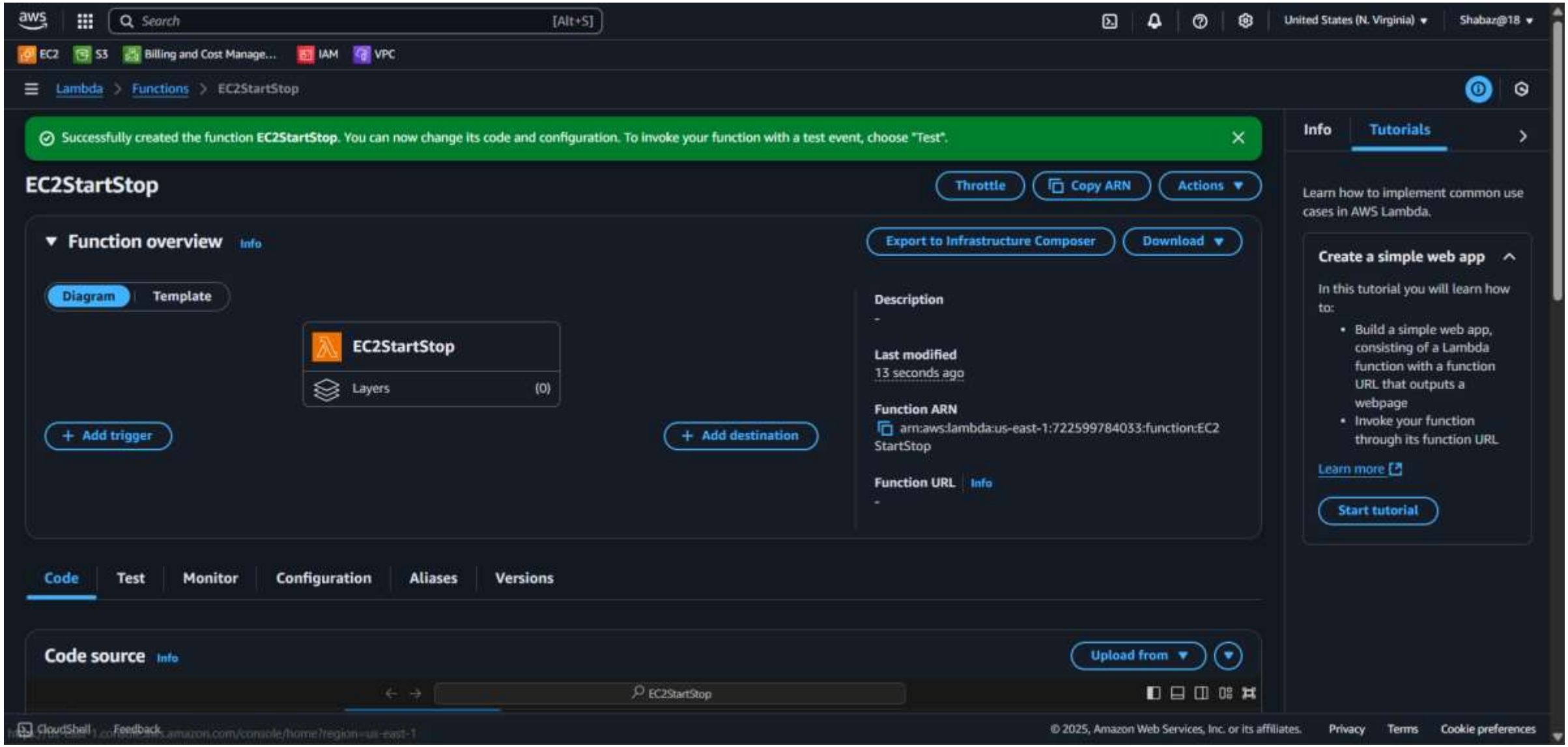
- Select RUNTIME as PYTHON 3.9



- Select ARCHITECTURE as x86_64
- Then Click on CREATE FUNCTION

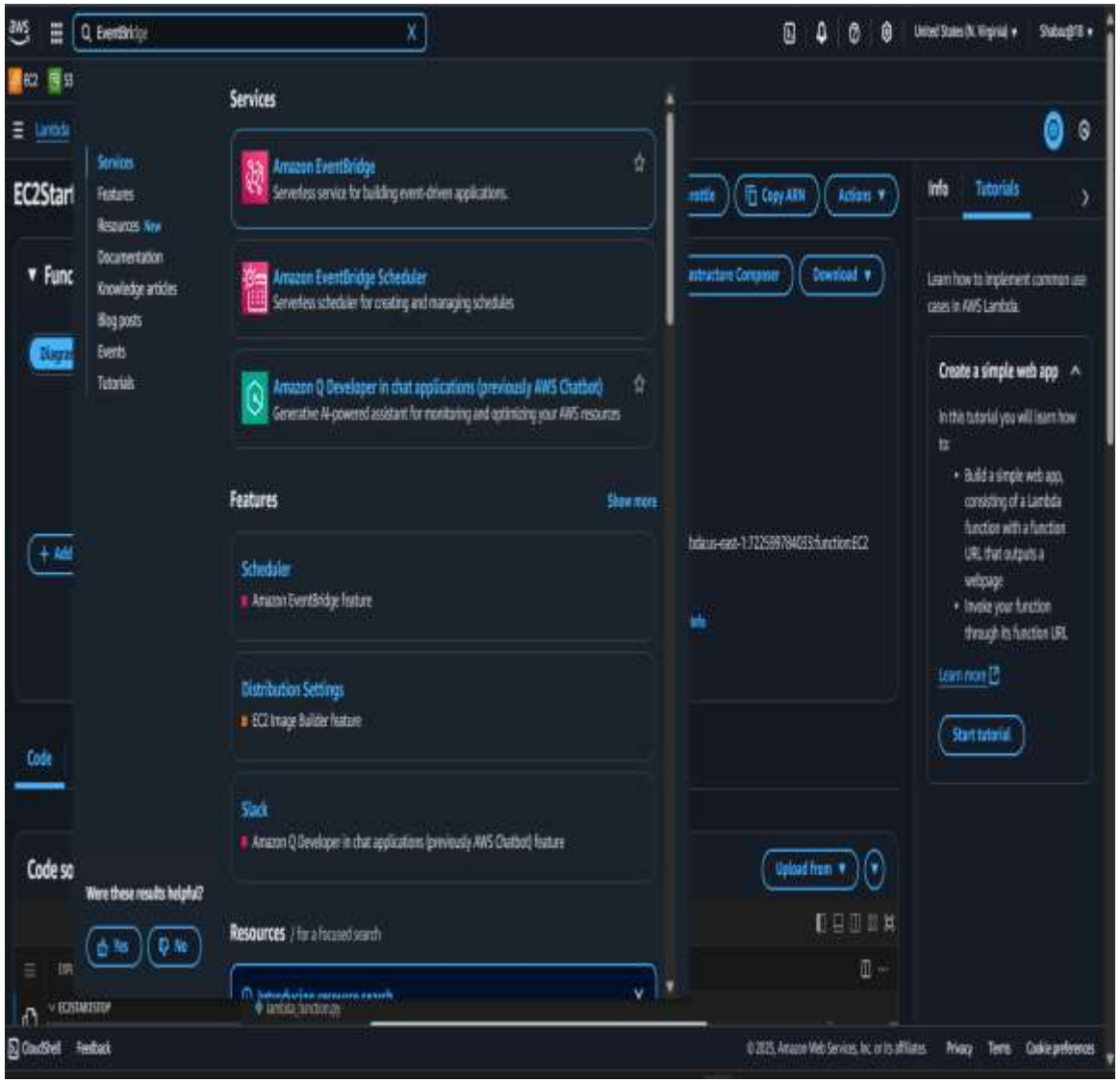


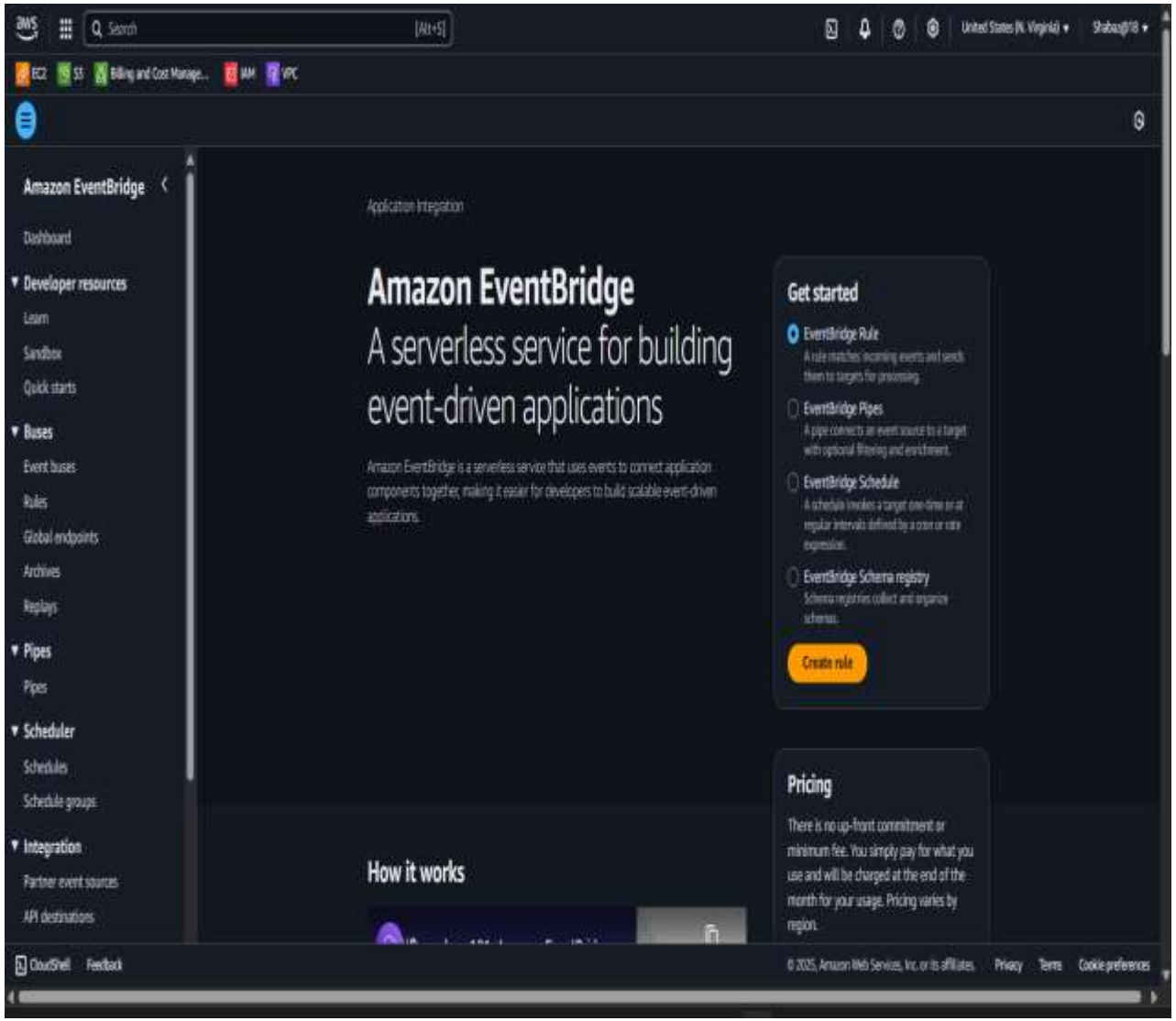
- LAMBDA FUNTION CREATED



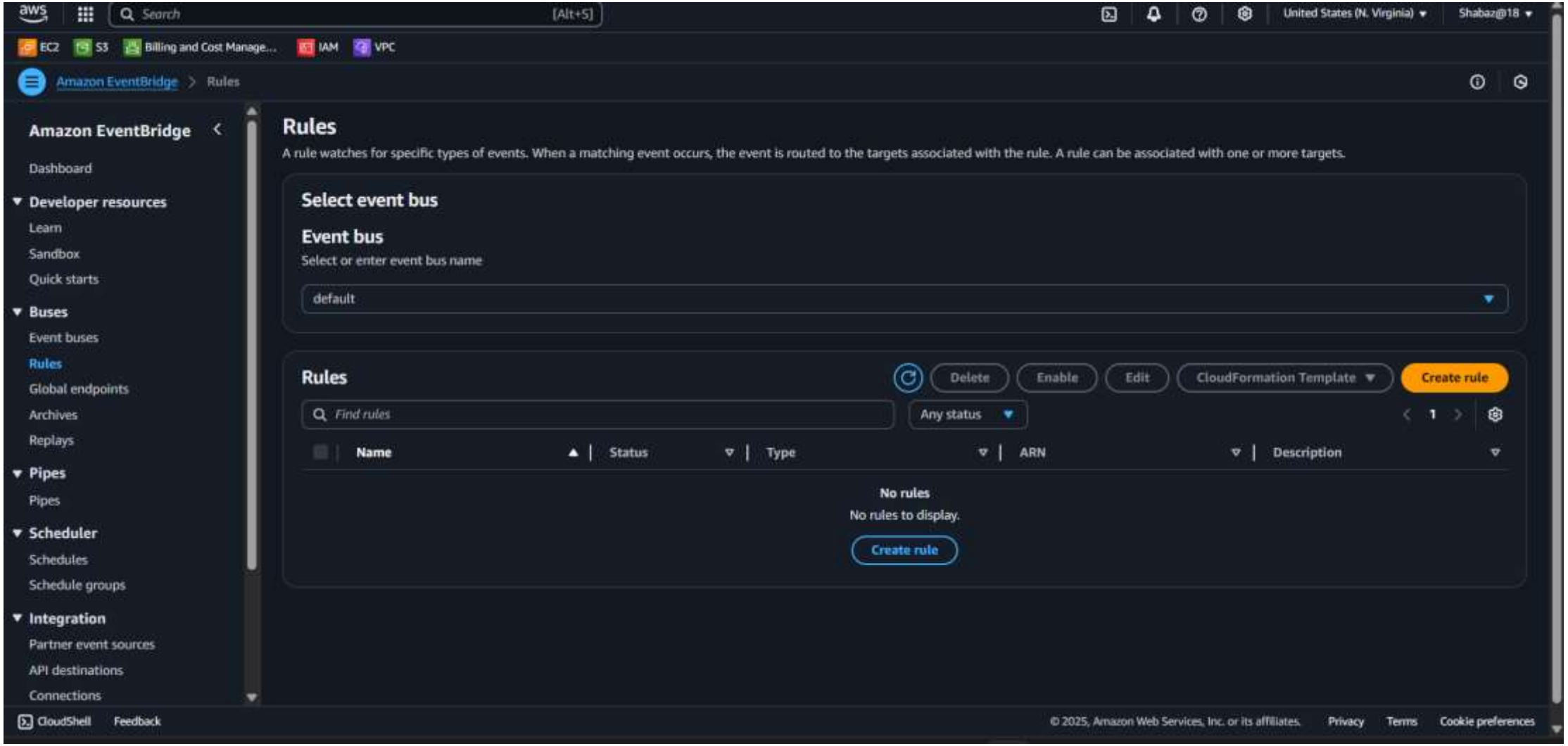
>>> NEXT CREATING **AMAZON EVENTBRIDGE**

- In Search bar, Search for AMAZON EVENTBRIDGE & Open it





- Click on <u>RULE</u> and <u>CREATE NEW RULE</u>



- Give RULE DETAILS like : NAME as **StartEC2Schedule**

Step 1
Define rule detail
Step 2
Build event pattern
Step 3
Select target(s)
Step 4 - *optional*
Configure tags
Step 5
Review and create

Define rule detail Info

Rule detail

Name

StartEC2Schedule

Maximum of 64 characters consisting of numbers, lower/upper case letters, .,-,_.

Description - *optional*

Enter description

Event bus Info

Select the event bus this rule applies to, either the default event bus or a custom or partner event bus.

default

Enable the rule on the selected event bus

Rule type Info

Rule with an event pattern
A rule that runs when an event matches the defined event pattern. EventBridge sends the event to the specified target.

Schedule
A rule that runs on a schedule

Cancel Next

- At RULE TYPE Select SCHEDULE and Click on Continue in EventBridge Scheduler

Step 2
Define schedule

Step 3
Select target(s)

Step 4 - *optional*
Configure tags

Step 5
Review and create

Rule detail

Name

StartEC2Schedule

Maximum of 64 characters consisting of numbers, lower/upper case letters, .,-,_.

Description - *optional*

Enter description

Event bus | Info

Select the event bus this rule applies to, either the default event bus or a custom or partner event bus.

default

Enable the rule on the selected event bus

Rule type | Info

Rule with an event pattern
A rule that runs when an event matches the defined event pattern. EventBridge sends the event to the specified target.

Schedule
A rule that runs on a schedule

EventBridge Scheduler - A new AWS scheduling capability! New

A new EventBridge scheduling functionality that provides one-time and recurring scheduling functionality independent of Event buses and rules. You can create a schedule to invoke targets such as a Lambda function.

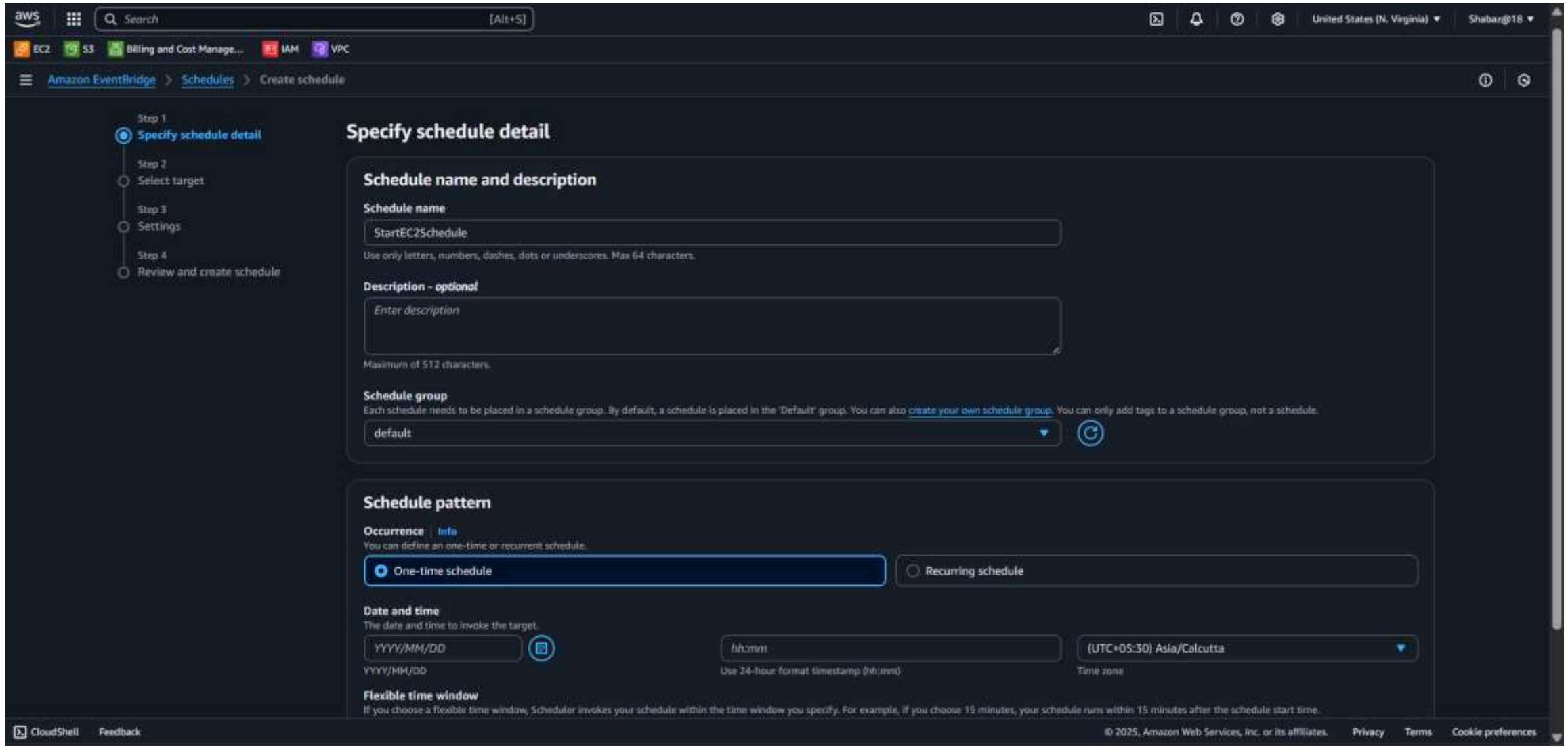
Learn More

Continue to create rule

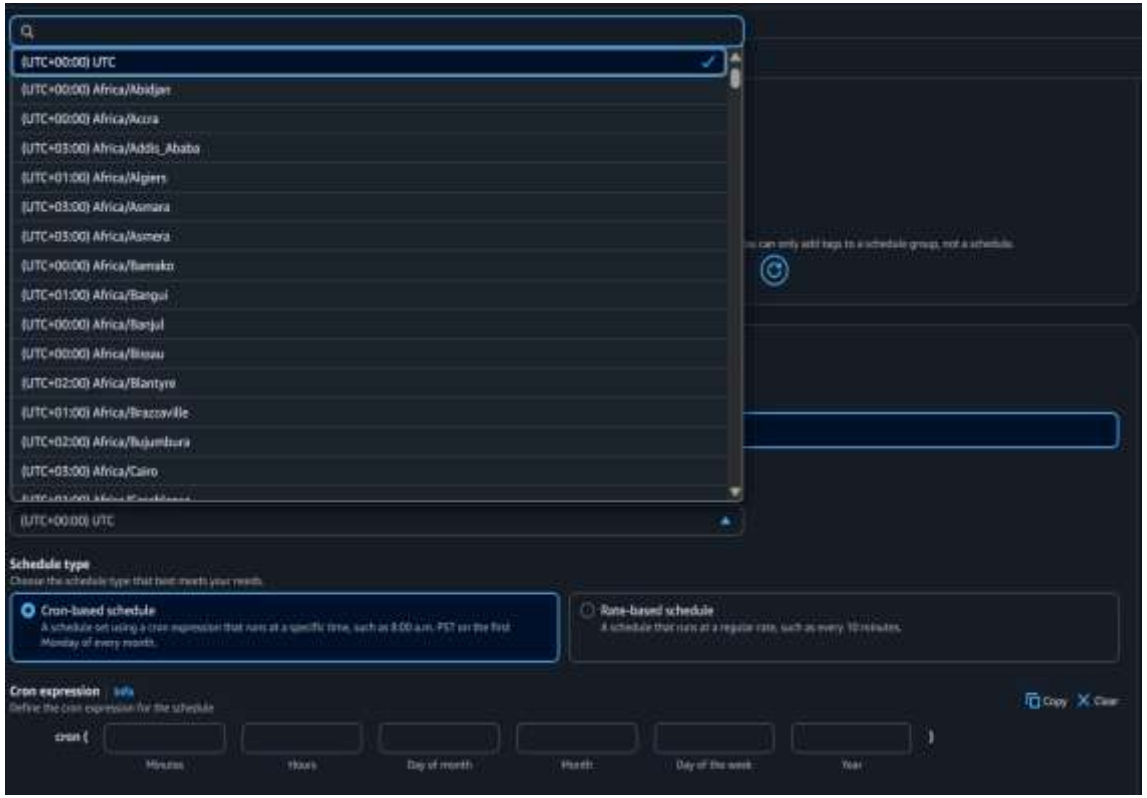
Cancel

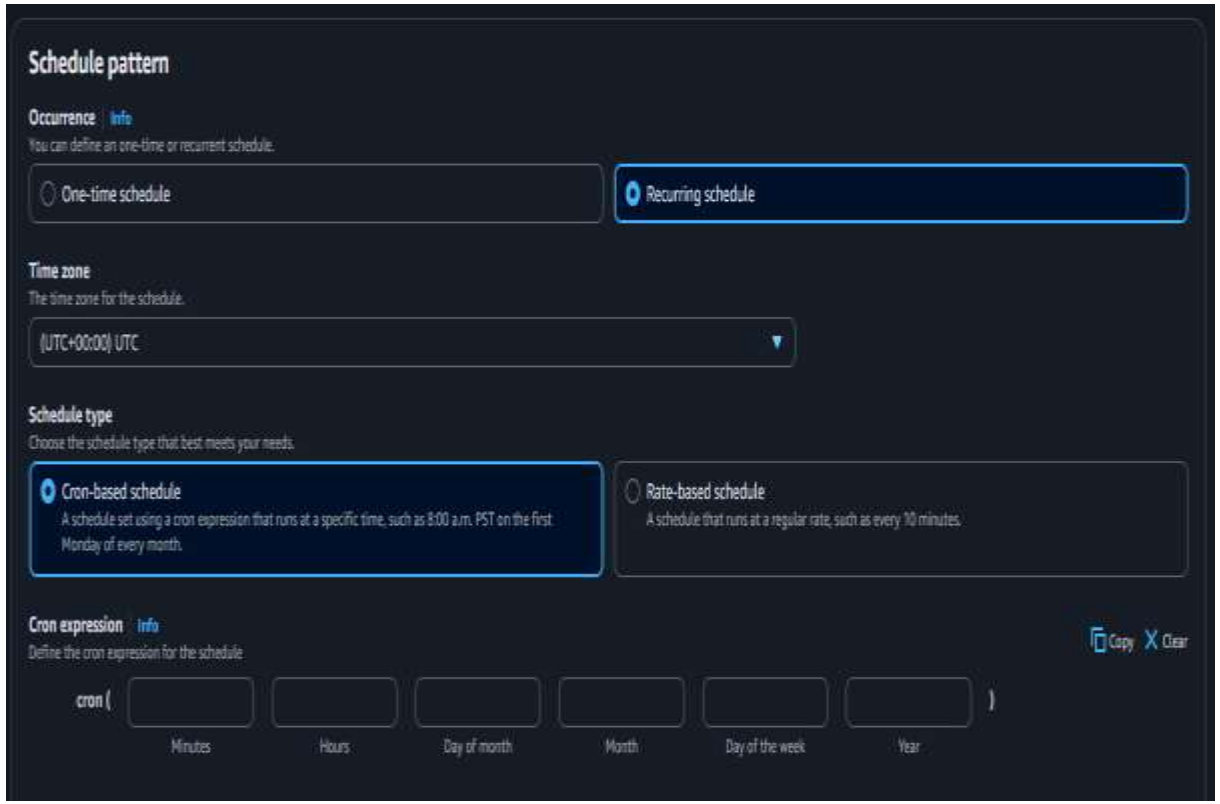
Continue in EventBridge Scheduler

- After Clicking on (Continue in EventBridge Scheduler) It will appear a new Page
- Give <u>SCHEDULE NAME</u> as <u>StartEC2Schedule</u>

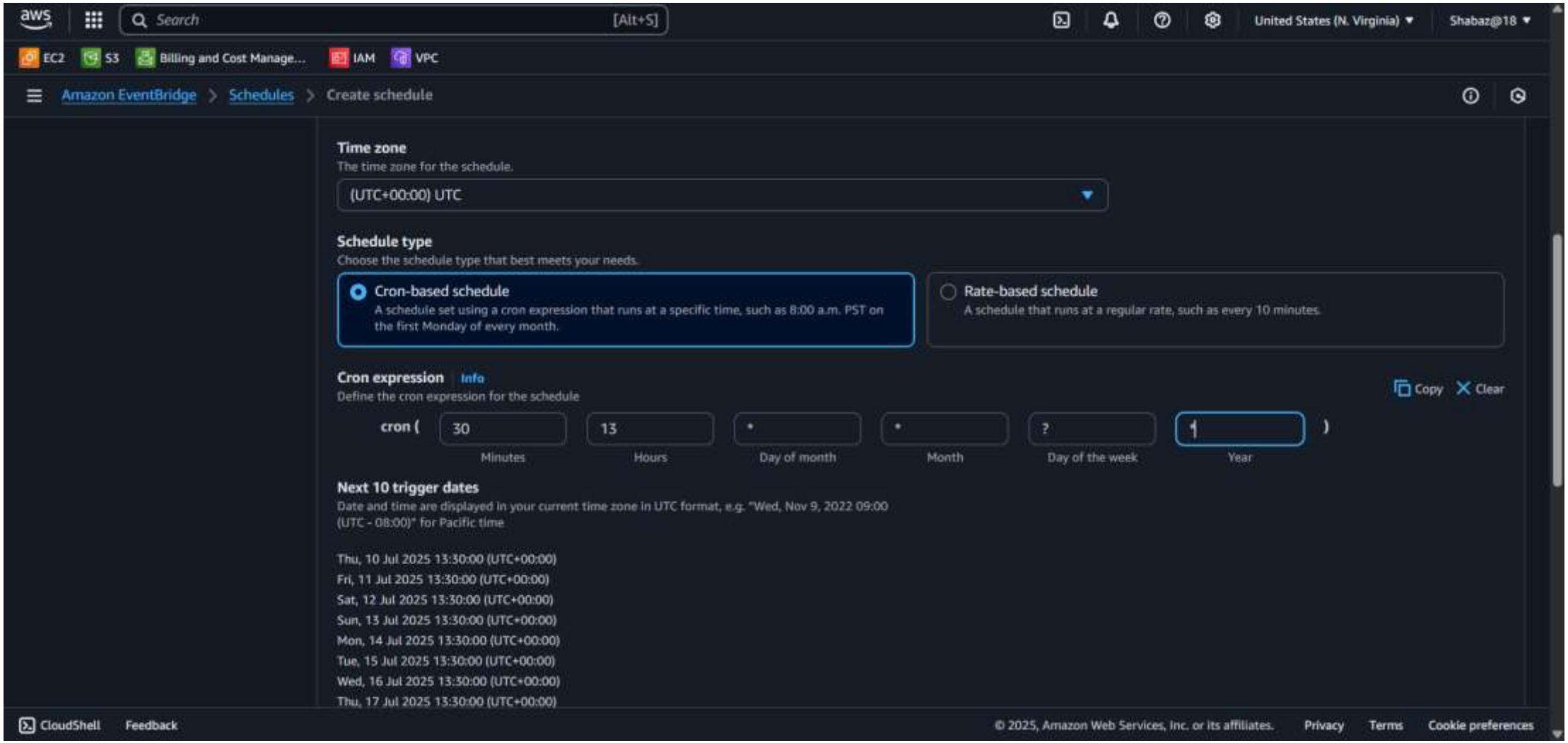


- Select Occurrence to RECURRING SCHEDULE at SCHDEULE PATTERN
- And select TIME ZONE as (UTC+00:00) UTC)
- Next Select CRON-Based Schedule at SCHEDULE TYPE



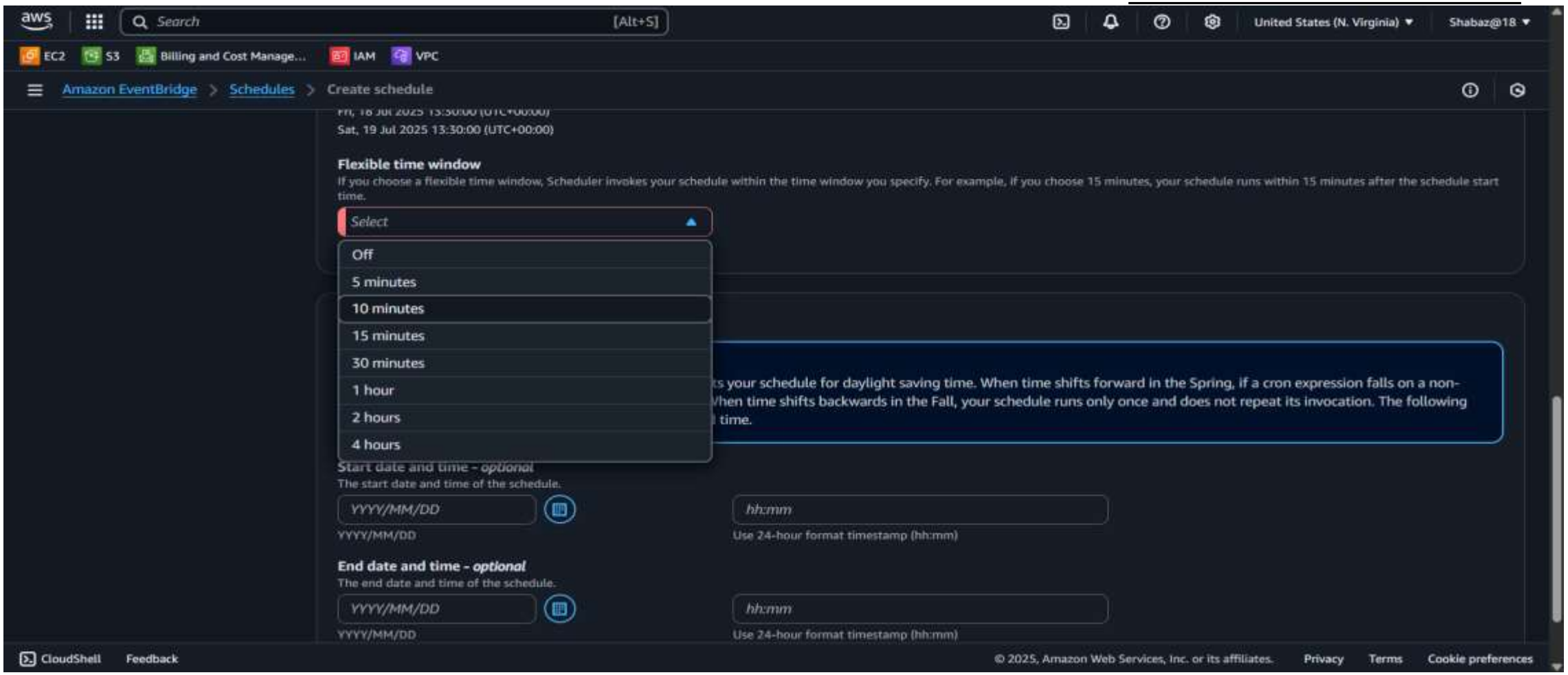


- Set what Schedule you want to - At <u>CRON EXPRESSION</u>
- I have Scheduled at 1:30 Today

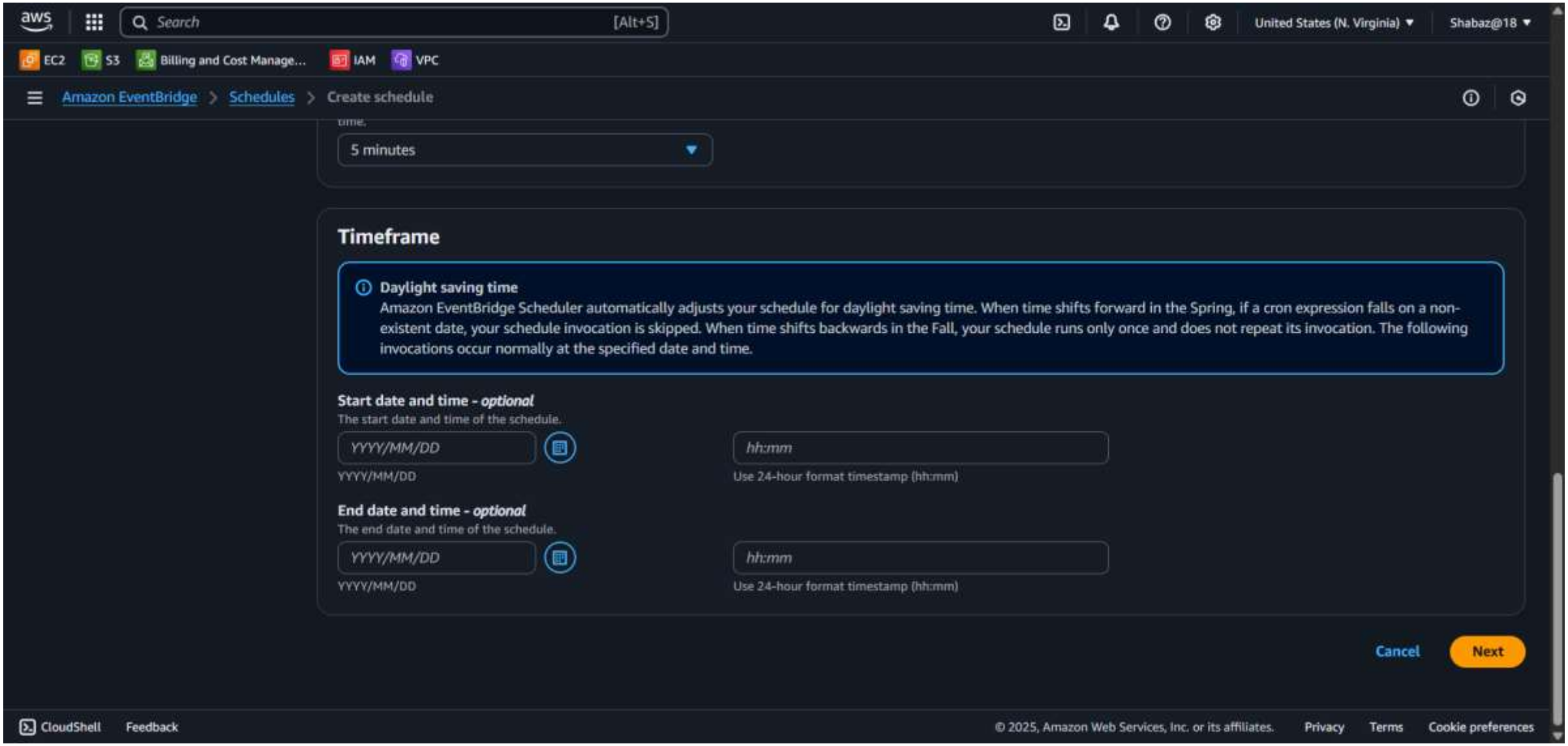


- Scroll down & Choose a Flexible time windows – What is it ?
- If you choose a flexible time windows, Scheduler invokes your schedule within the time window you specify.
- For Example: If you choose 10 minutes, your schedule runs within 10 minutes after the schedule start time

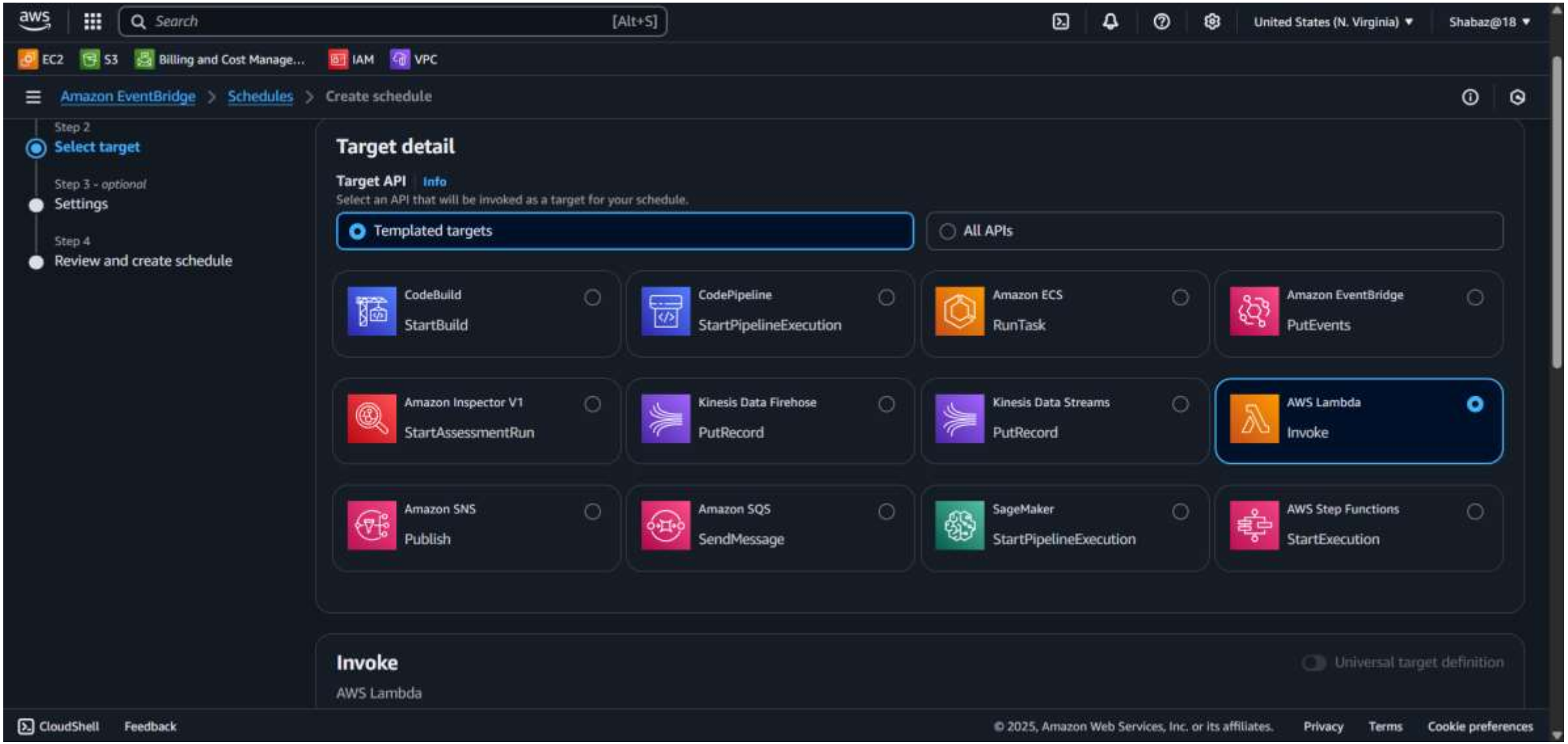
- THIS STATEMENT CLEARLY MENTIONED AT BOTTOM OF THE FLEXIBLE TIME WINDOWS



- Scroll Down And Click on Next

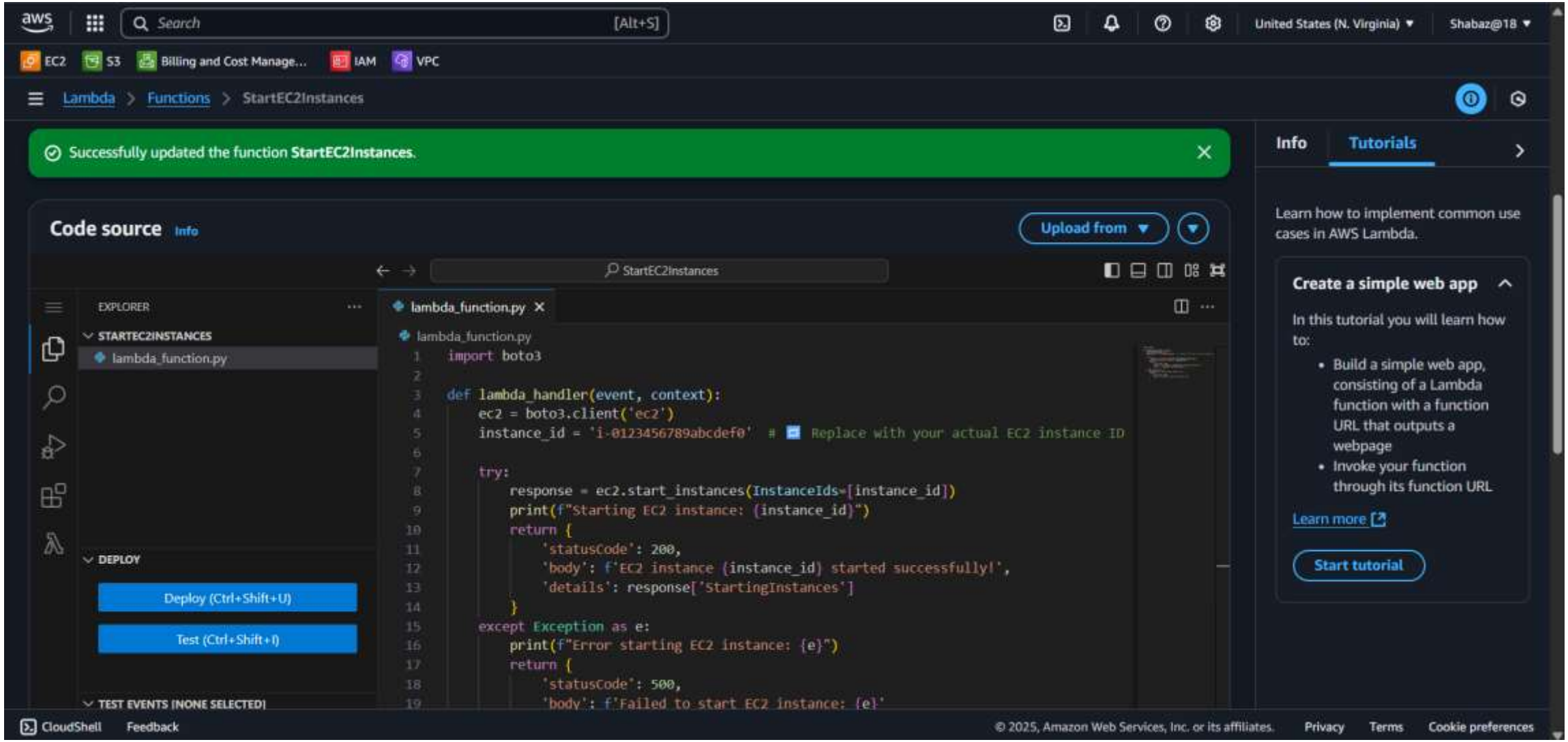


- After Clicking Next, You will be Redirected to Previous Page which is Amazon EventBridge - Step 2
- At Target detail Choose Templated targets in that Select AWS LAMBDA



- Scroll down through INVOKE
- select Start function which we named it as StartEC2instances in Lambda function
- Then Scroll down

- Here You have to Write the Python code of Start EC2 Instance
- Next Click on Deploy to Deploy the code & then Click on Test to Test the code, After that click on enter



- Same like that you have to Write the Python code of Stop EC2 Instance
- Here are PYTHON CODE for START EC2 INSTANCE & STOP EC2 INSTANCE

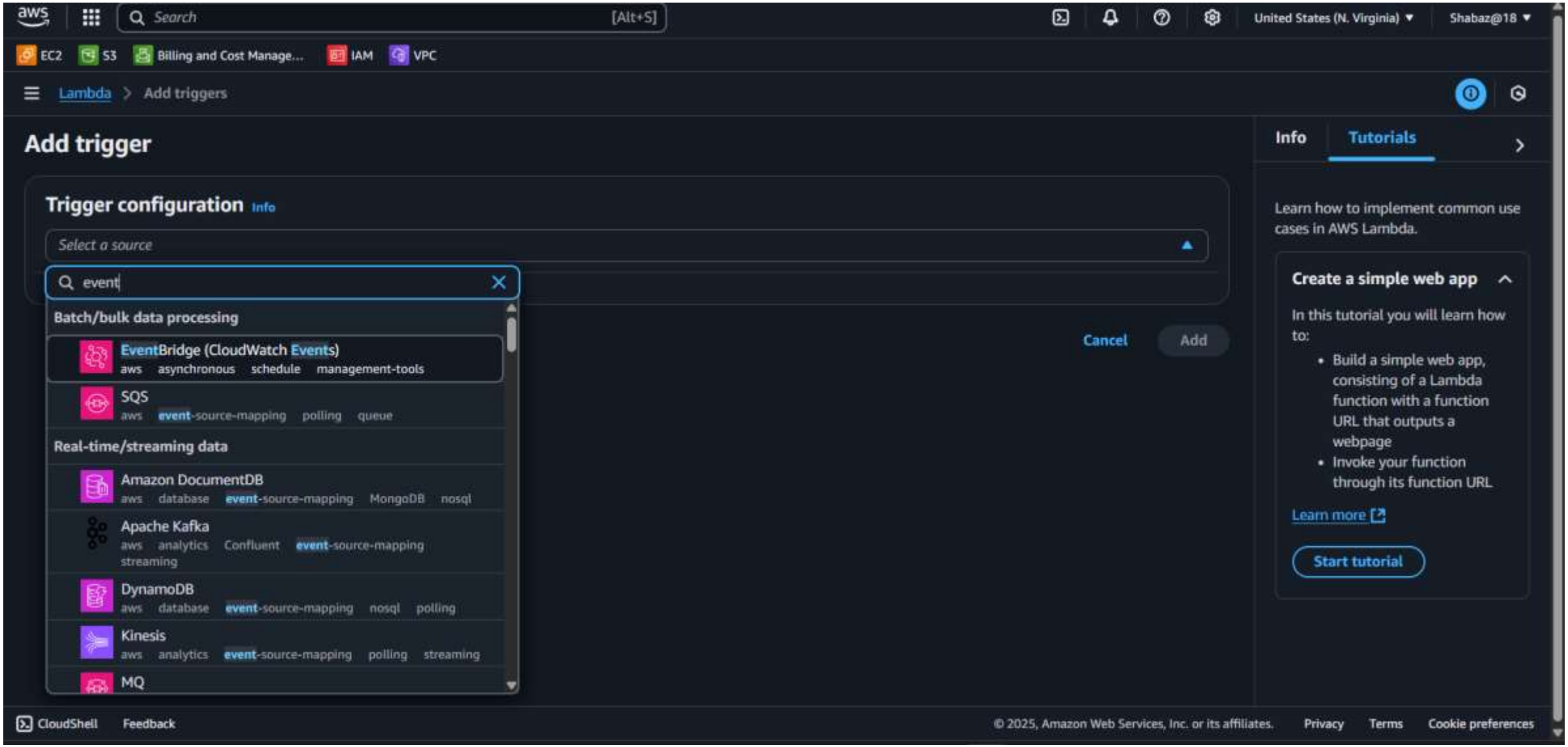
>>> **PYTHON CODE to START EC2 INSTANCE**

```
import boto3def lambda_handler(event, context):    ec2 = boto3.client('ec2')
instance_id = 'i-0123456789abcdef0'  # 🔁 Replace with your actual EC2 instance ID
try:        response = ec2.start_instances(InstanceIds=[instance_id])
print(f"Starting EC2 instance: {instance_id}")        return {            'statusCode': 200,
'body': f'EC2 instance {instance_id} started successfully!',            'details':
response['StartingInstances']        }    except Exception as e:        print(f"Error starting
EC2 instance: {e}")        return {            'statusCode': 500,            'body': f'Failed to
start EC2 instance: {e}'        }
```

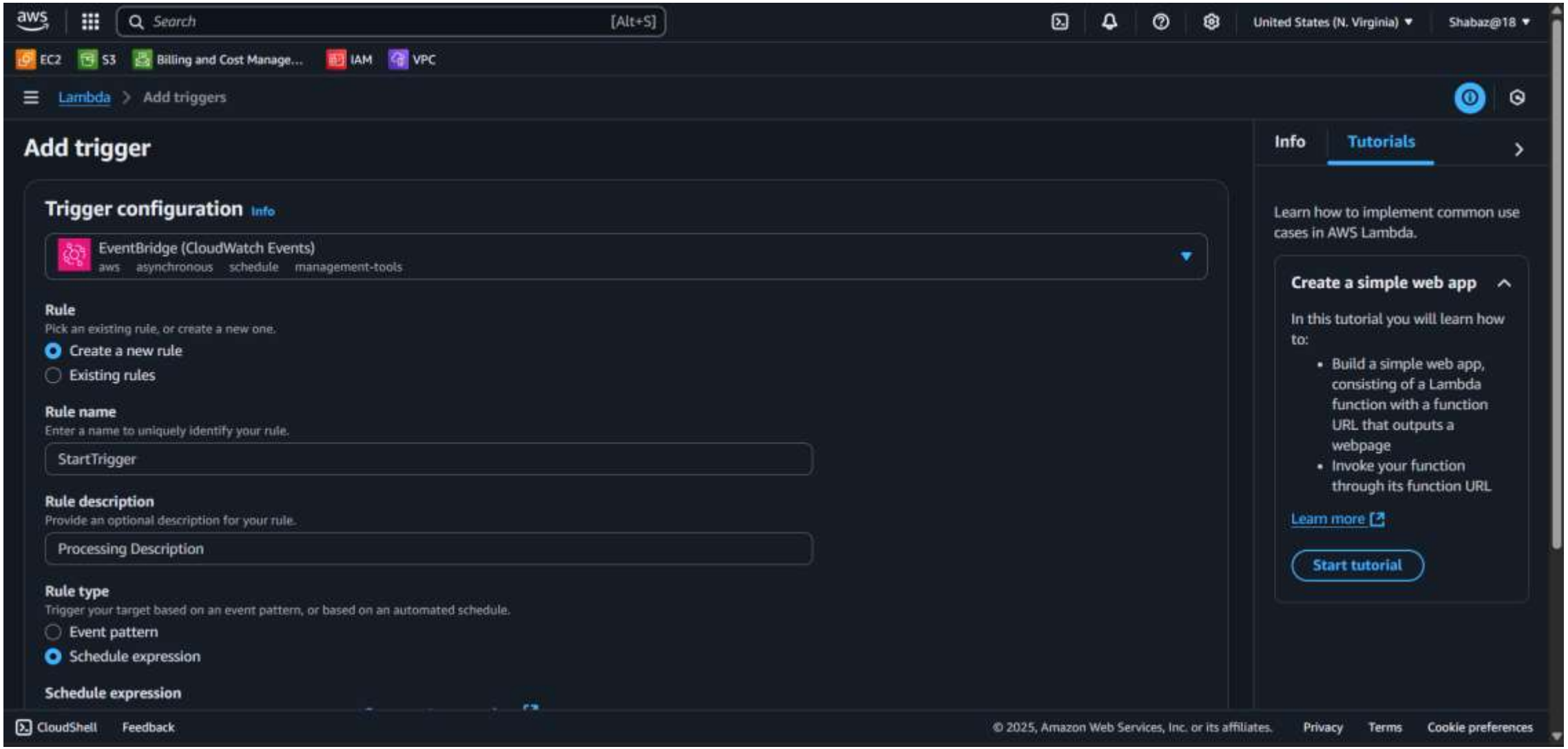
>>> **PYTHON CODE to STOP EC2 INSTANCE**

```
import boto3def lambda_handler(event, context):    ec2 =
boto3.client('ec2')    instance_id = 'i-0123456789abcdef0'  # 🔁 Replace
with your actual EC2 instance ID    try:        response =
ec2.stop_instances(InstanceIds=[instance_id])        print(f"Stopping EC2
instance: {instance_id}")        return {            'statusCode': 200,
'body': f'EC2 instance {instance_id} stopped successfully!',
'details': response['StoppingInstances']        }    except Exception as e:
print(f"Error stopping EC2 instance: {e}")        return {
'statusCode': 500,            'body': f'Failed to stop EC2 instance: {e}'        }
```

- Next Open LAMBDA & click on ADD TRIGGERS
- Select EVENTBRIDGE (CloudWatch Events) at Trigger configuration



- Choose Create a New Rule at RULE , Then Give Rule name as StartTrigger
- Next give RULE DESCRIPTION (anything you want) like : Processing Description



- Then choose Rule type as SCHEDULE EXPRESSION
- And enter Schedule Expression like **CRON(30 13 * * ? *)** like you given Before at the Amazon EventBridge > Schedulers
- Then Click on ADD

aws | Search [Alt+S] | United States (N. Virginia) | Shabaz@18

EC2 | S3 | Billing and Cost Manage... | IAM | VPC

Lambda > Add triggers

Existing rules

Rule name
Enter a name to uniquely identify your rule.
StartTrigger

Rule description
Provide an optional description for your rule.
Processing Description

Rule type
Trigger your target based on an event pattern, or based on an automated schedule.
Event pattern
Schedule expression

Schedule expression
Self-trigger your target on an automated schedule using Cron or rate expressions. Cron expressions are in UTC.
cron(30 13 * * ? *)
e.g. rate(1 day), cron(0 17 ? * MON-FRI *)

Lambda will add the necessary permissions for Amazon EventBridge (CloudWatch Events) to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

Cancel | Add

Info | Tutorials

Learn how to implement common use cases in AWS Lambda.

Create a simple web app

In this tutorial you will learn how to:

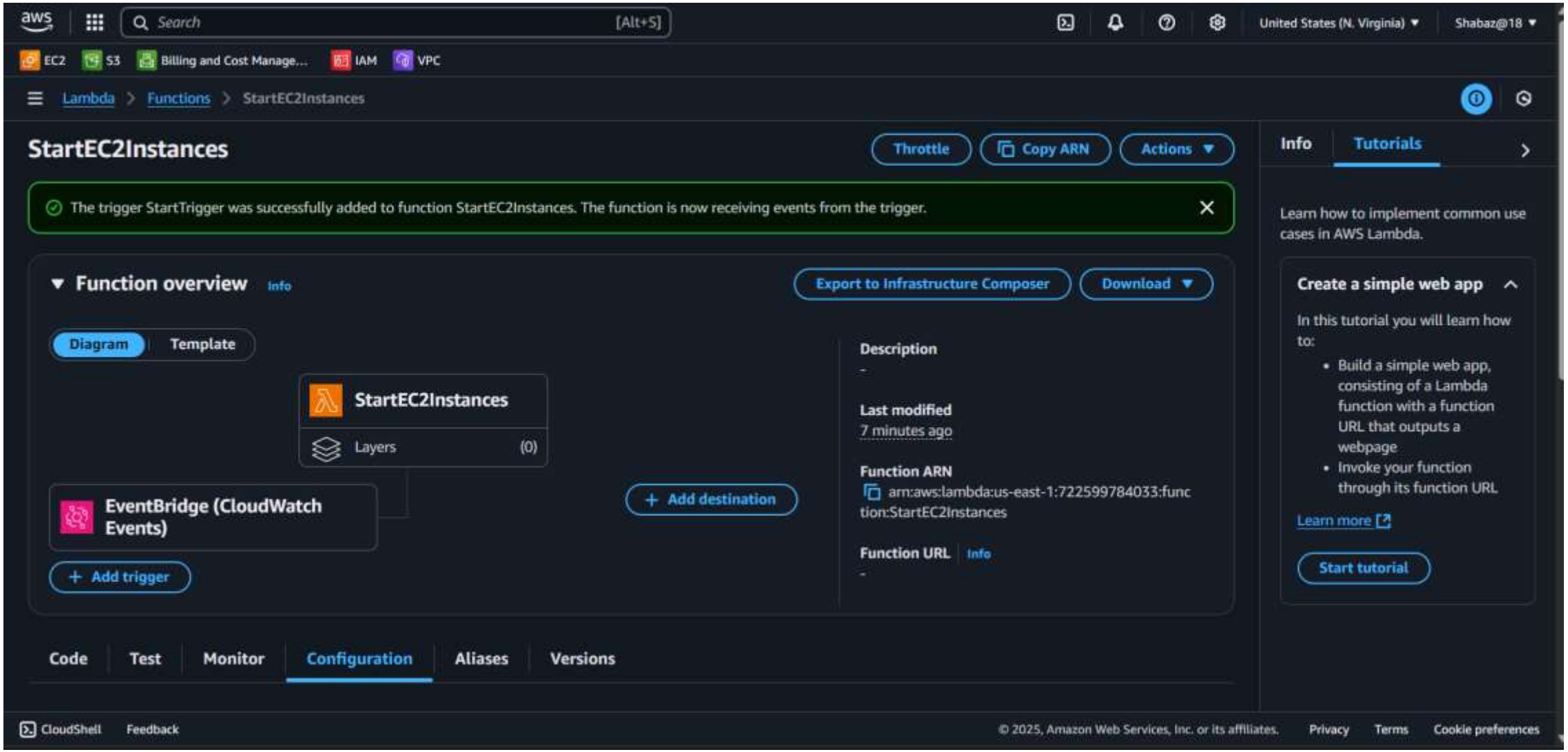
- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

Learn more

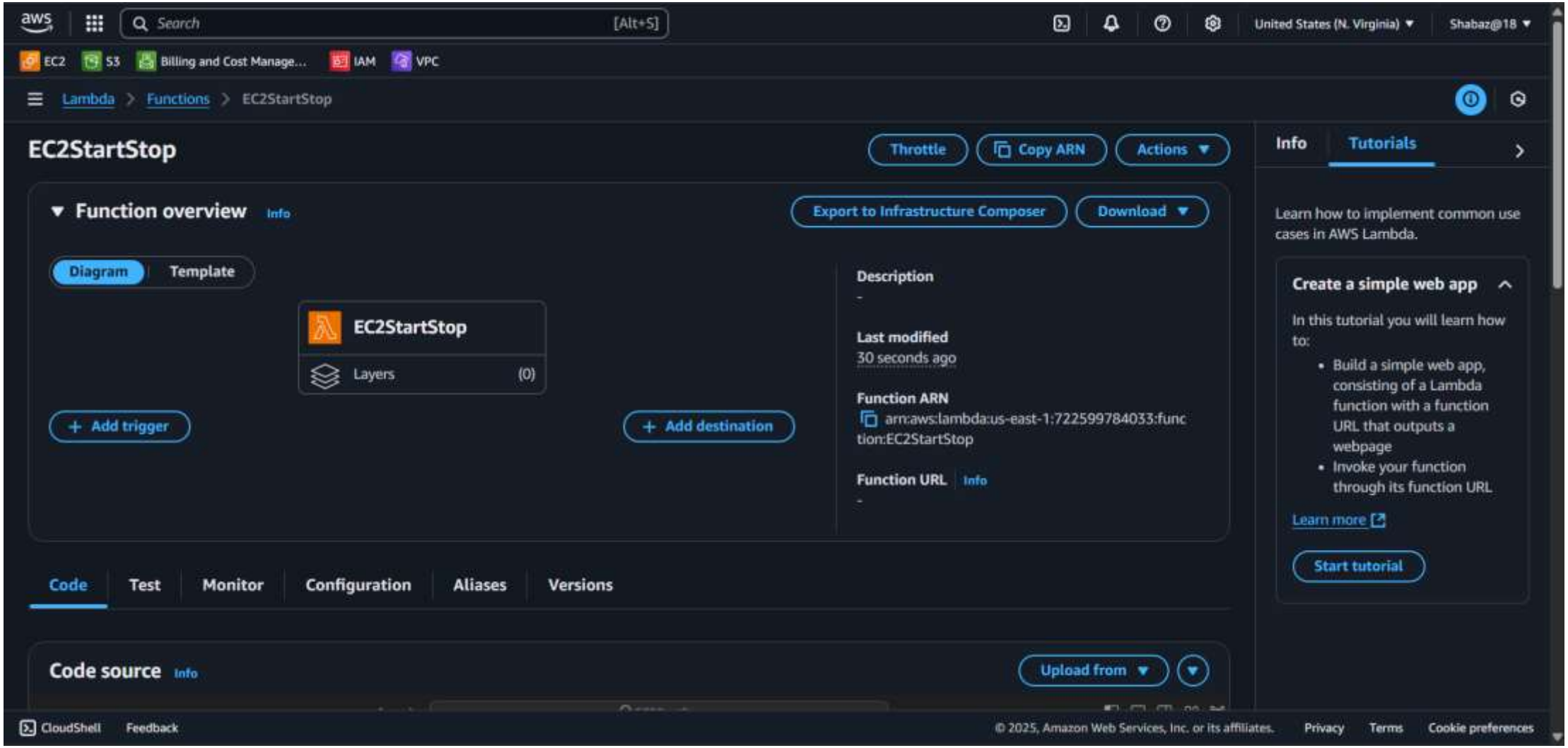
Start tutorial

CloudShell | Feedback | © 2025, Amazon Web Services, Inc. or its affiliates. | Privacy | Terms | Cookie preferences

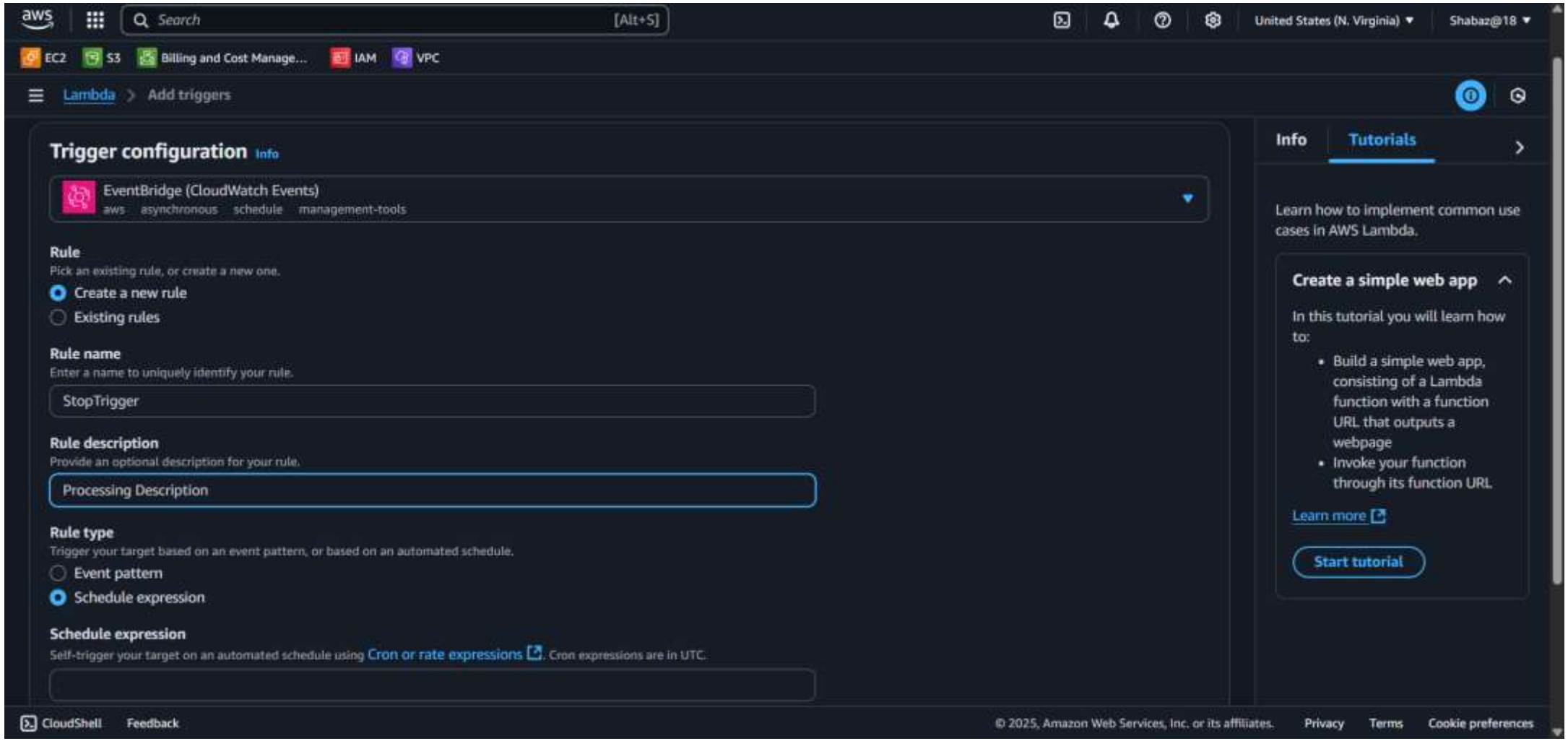
- Here You can see the Added trigger to our Start EC2 instance



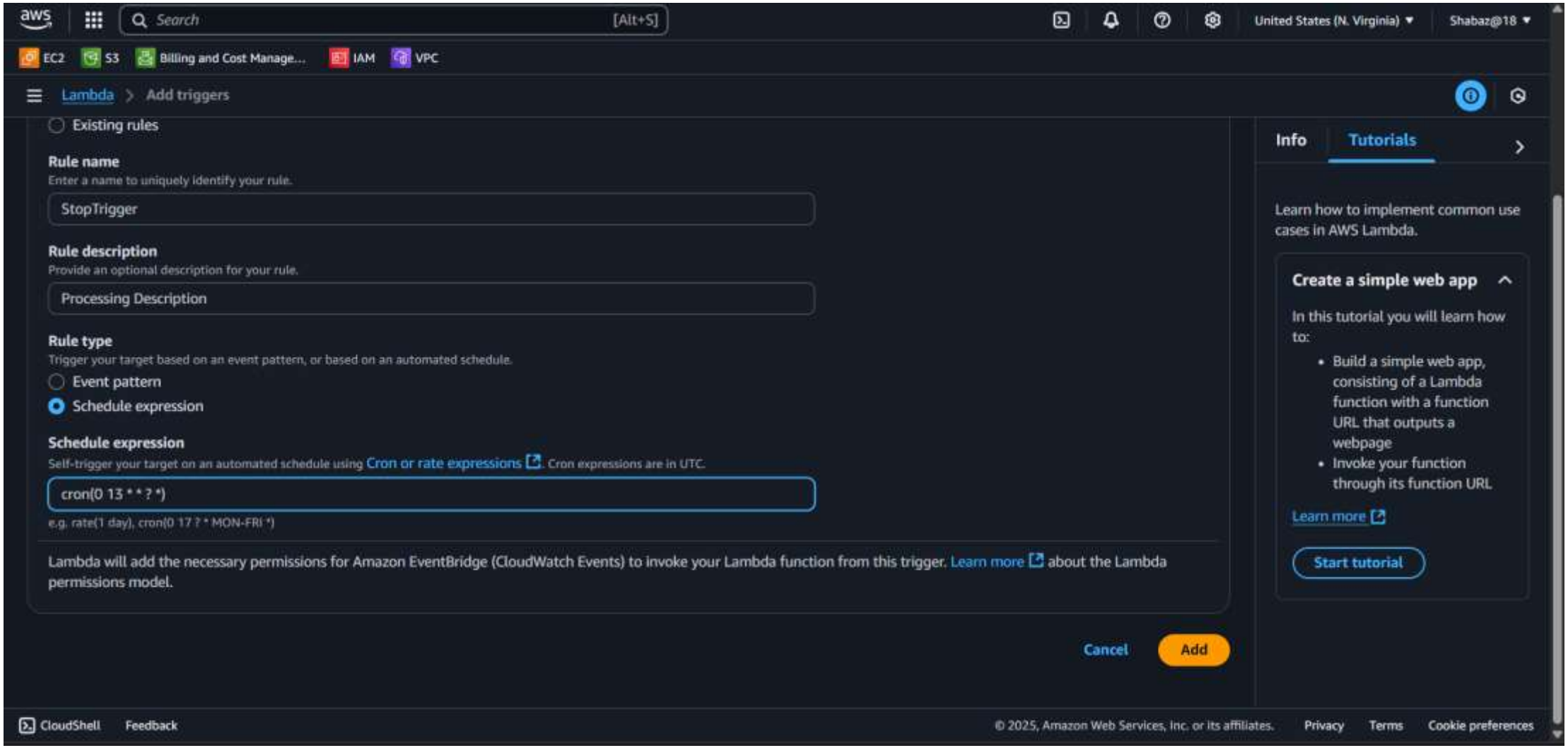
- Same like that, Open another LAMBDA function which is Stop function that we named as EC2StartStop Function
- Click on ADD Trigger



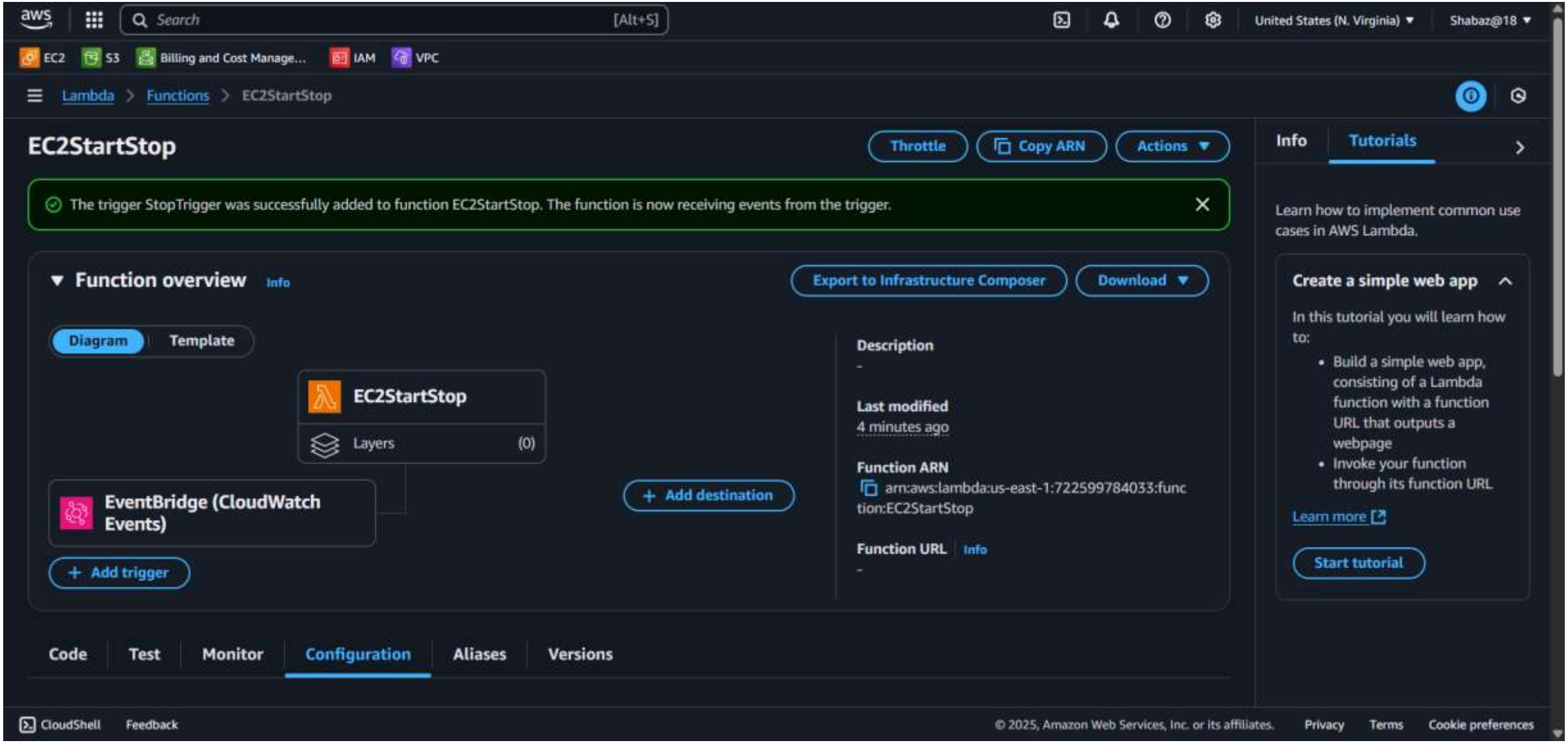
- Select EVENTBRIDGE (CloudWatch Events) at Trigger configuration
- Choose Create a New Rule at RULE , Then Give Rule name as StopTrigger
- Next give RULE DESCRIPTION (anything you want) like : Processing Description



- Then choose Rule type as SCHEDULE EXPRESSION
- And enter Schedule Expression like **CRON(0 13 * * ? *)** like you given Before at the Amazon EventBridge > Schedulers
- Then Click on ADD



- Here You can see the Added trigger to our Start EC2 instance



- **AWS CloudWatch Events** were configured to **automate the timing** of EC2 start and stop actions.

<u>Two scheduled rules were created:</u>

- **Start Trigger:** Every day at 1 PM
- **Stop Trigger:** Every day at 1:30 PM

- These rules are **connected to respective Lambda functions**, enabling **automated EC2 management** without human intervention.

SUMMARY :

- The **key advantages** of the project:
- **Automation:** EC2 operations run without manual action
- **Cost Efficiency:** Idle resources are shut down on schedule
- **Scalability:** Supports managing multiple EC2 instances
  It also concludes that this solution offers a **simple, serverless, and effective approach** to cloud automation using core AWS services.

CONCLUSION :

- This project demonstrates how powerful and cost-effective cloud automation can be when the right AWS services are integrated.
- By combining **Lambda**, **CloudWatch**, and **IAM**, we've built a hands-free system that reflects the future of **efficient cloud resource management**.

"In the cloud, efficiency isn't an option — it's a strategy."