**Knowledge Transfer( Lambda Start Stop Function )**

**Dated : 28/08/2023**

**\*\*Challenge :\*\***

Hey geeks, today I have another challenge for , are you excited so what you have to do is you have to start and stop the EC2 instance using AWS services now I am not going to tell which service because this is the challenge so let’s go and dig some information about it and give me the solution …

==============================================================================

**\*\*Task :\*\***

Ticket: Get a solution for start and stop the EC2 instance.

==============================================================================

**\*\*Solution :\*\***

Hey Guys, I have found out the solution for the challenge that toggles the state of a designated EC2 instance between running and stopped. It is done using Lambda function in ehich the use of Python Boto3 module is used to interact with AWS EC2 service.

==============================================================================

**\*\*Pre-requisites:\*\***

• Have a good knowledge about EC2 and Python

• Know the all services of AWS

==============================================================================

**\*\*Objective:\*\***

The objective is to create an AWS Lambda function that toggles the state of a designated EC2 instance between running and stopped.

.==============================================================================

**\*\*Description:\*\***

AWS Lambda and Boto3, a Python library developed by AWS, constitute a powerful combination that enables developers to create serverless applications and interact with various AWS services seamlessly.

**AWS Lambda:**

AWS Lambda is a serverless compute service that eliminates the need for provisioning or managing servers. With Lambda, you can run code in response to events and triggers from AWS services or HTTP requests. It provides an environment where you can write your code in various programming languages, including Python, without worrying about server management. Lambda functions can be triggered by various AWS services such as S3, DynamoDB, API Gateway, CloudWatch Events, and more.

**Python Boto3:**

Boto3 is the official AWS SDK for Python, designed to provide a consistent interface to AWS services. It simplifies the process of integrating applications with AWS services by providing high-level abstractions and idiomatic Pythonic APIs. Boto3 allows you to interact with a wide range of AWS resources, including EC2 instances, S3 buckets, DynamoDB tables, and more. It supports various authentication methods and provides a convenient way to manage AWS resources programmatically.

Together, AWS Lambda and Boto3 offer developers a versatile toolkit for building and deploying serverless applications that can seamlessly interact with AWS services. Python's simplicity and Boto3's well-designed APIs make it easier to write code that automates tasks, handles events, and manages cloud resources, all while taking advantage of the benefits of serverless computing. Whether it's managing EC2 instances, processing data, or responding to events, this combination empowers developers to create efficient, scalable, and cost-effective solutions in the AWS cloud.

==============================================================================

**\*\*Steps:\*\***

# Login to your AWS account and search EC2 service and Follow the steps…

**Step 1 ( Creating EC2 Instance)**

# Firstly create a AWS linux EC2 Instance as we always create it…

# After creating stop the instance and leave it…

**Step 2 ( Creating Role )**

# Now we have to make IAM role to give the permission to lambda

# Come to IAM service and select role and follow the steps…

* After coming in roles select create role
* In use case select lambda and click next
* Don’t give any permission and click next
* Now give the name as per your choice
* Lastly click create role

# In this way your role is created now we have to give some permission to it so open the role which you have created by clicking it…

# After click it follow the steps…

* Now in permission section click add permission
* In add permission click create incline policy
* Now in select service click EC2
* Next in action allowed search Start

# After searching start you will see Start Instances click on it…

* Now again in action allowed search Stop

# Then you will see Stop Instances click on it…

* Now down in resources section click Add ARN

# After clicking it a dialogue box will open enter the details init…

* In dialogue box select this account
* Now give the region your instance is in

# Give the instance region in code format not in name format **eg :- eu-west-3…**

* Next give your instance ID
* Now click on Add Arn

# By this you have entered the instance details which you have to start and stop…

* Then click next
* Now give the policy name and click create policy

# Now you have finally created the role which the lambda function will you to operate the instance…

**Step 3 ( Creating Lambda Function )**

# After creating instance search lambda service and come to it’s dashboard and follow the steps…

* On dashboard click create function
* Then click author from scratch
* Now give the function name as Start Instance
* In runtime select Python latest version

# There are many other scripts also through which you can perform this task but we will continue it in python…

* Now click on change default execution role below permission section
* Next click on use an existing role
* Now select the IAM role you have created
* Then click create function

# Now create another function named Stop Instance using above steps…

**Note : Don’t forget to add IAM role to function if it not added then function will not work…**

**Step 4 ( Uploading Python Script )**

# After Creating functions we have to upload the python script in it to make functions work…

# Don’t worry I have Attach both scripts for you below copy that scripts and follow the steps…

* Now open the start function
* Scroll down below there will be Code source

# There will be sample python script delete and paste the start script there…

**Note: After pasting make the changes in the script as stated in it…**

* After pasting click deploy
* Next click on test and a dialogue box will appear
* Then give name to event as Start and click save
* Now again click on test

# After clicking test Execution result will run, in that response result should be null and if any error occurs check the script in that any extra spaces or the replace objects are wrong…

**# After correcting the error again deploy it and test it…**

# Now go to the instance of which you have entered the ID and check it has been started…

# Next click on the configuration tab above and follow the steps…

* After clicking it click on edit
* Now scroll down to Timeout section
* Then change the 3 sec timeout – 10 sec timeout
* Now click save

# Now follow the above steps and add Stop script in Stop Function…

# After clicking test the same running instance will be stopped…

**Python Scripts :**

**( Start Script )**

import boto3

import json

region = 'us-east-1'

instance\_id = str('i-0fb27af7af2efa8fb')

def lambda\_handler(event, context):

ec2 = boto3.client('ec2', region\_name=region)

ec2.start\_instances(InstanceIds=[instance\_id])

print ('started your instance: ' + str(instance\_id))

**( Stop Script )**

import boto3

import json

region = 'us-east-1'

instance\_id = str('i-0fb27af7af2efa8fb')

def lambda\_handler(event, context):

ec2 = boto3.client('ec2', region\_name=region)

ec2.stop\_instances(InstanceIds=[instance\_id])

print ('stopped your instance: ' + str(instance\_id))

**#### THE END ####**

==============================================================================

**\*\*Explanation:\*\***

Creating a Cloud-Formation stack involves a step-by-step process to efficiently deploy AWS resources using infrastructure as code. To begin, you need to prepare a Cloud-Formation template, a YAML or JSON file that outlines the desired AWS resources and their configurations. This template encapsulates your infrastructure design, making it manageable and repeatable.

Access the AWS Management Console using your credentials and navigate to "Cloud-Formation" under "Management & Governance." Click on the "Create stack" button to initiate the stack creation process. You'll then choose how to specify the template – either by uploading a file or providing a URL.

Next, provide essential stack details. Give your stack a unique name that identifies its purpose within your AWS environment. If your template has parameters, input values for them here. Parameters enable customization of the stack without modifying the template directly.

Further configure your stack by adding tags for organizational purposes, setting permissions using IAM roles, and adjusting advanced settings if required. Once all settings are defined, review them to ensure accuracy.

With settings reviewed, click "Create stack" to start provisioning resources. The "Stacks" dashboard will display the stack's progress, initially showing "CREATE\_IN\_PROGRESS." You can monitor the advancement here.

If your template includes output values, access the "Outputs" tab to retrieve useful information about the created resources, such as URLs or IDs.

As your infrastructure needs evolve, you can update your stack by modifying the template and using the "Update stack" option. Alternatively, if resources are no longer needed, you can choose "Delete stack" to remove all associated components. Confirm deletion to prevent accidental data loss.