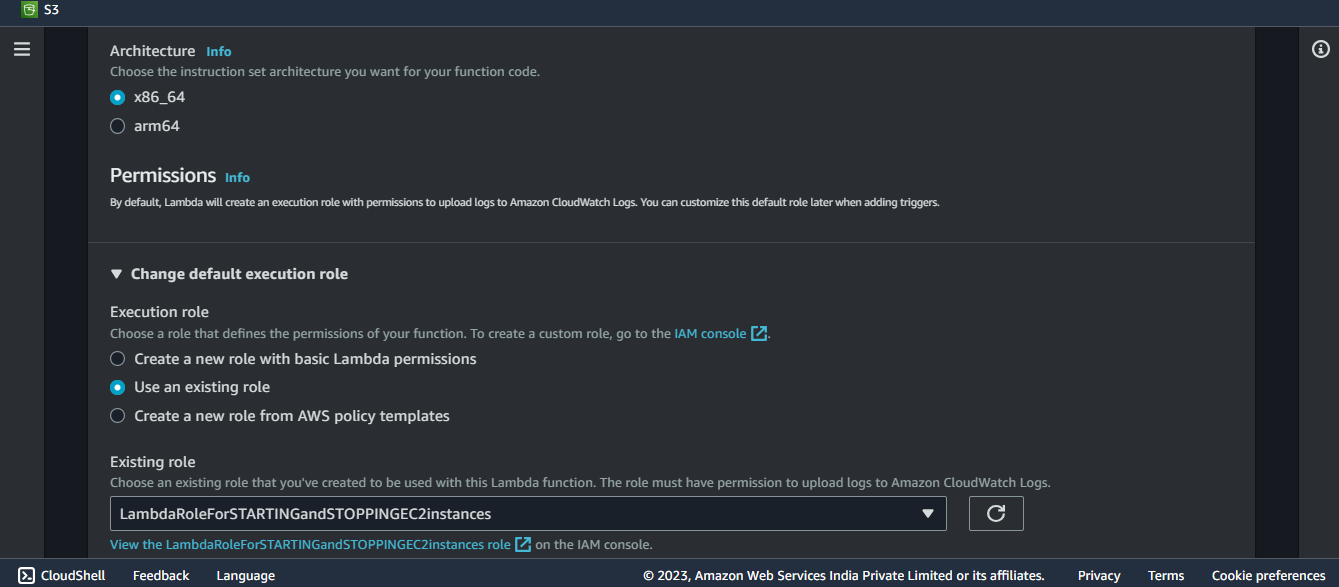
**Creating Lambda Function**

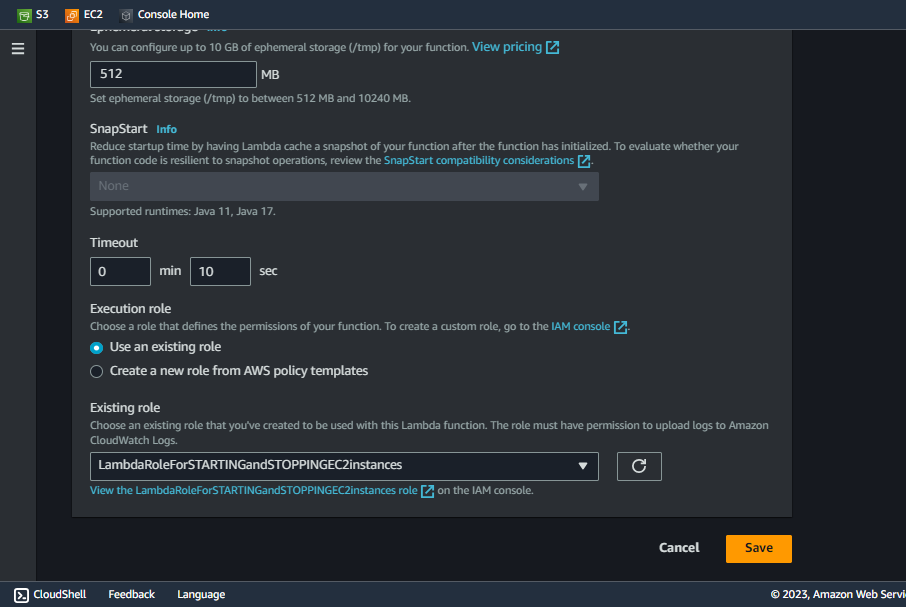
1. Go to my **GitHub** repository and copy the **Lambda function** from it.
2. Open the **Lambda console** from [here](https://console.aws.amazon.com/lambda).
3. A screenshot of a computer

   Description automatically generatedIn the left navigation pane, choose **Functions** tab**.**
4. A service page will open. Click on **Create function** button.
5. Ensure that **Author from Scratch** option is selected as the preferred. Otherwise, select it.
6. Configure the following settings:
   * + **Function name**: Enter a name for the function **LambdaEC2StartStopFunction**.
     + **Runtime**: Choose **Python 3.11**.
     + **Architecture:** x86\_64.
7. Under Permissions, expand **change default execution role** and under **Execution role**, choose ‘**Use an existing role**’.
8. From the drop-down under **Existing role**, choose the **IAM role** you made.
9. The picture below offers a clear overview of the Step 7 and Step 8.
10. Leave the other settings as default and proceed with the setup. Click on **Create function** buttonto finalize the creation of your Lambda function.
11. **A screenshot of a computer

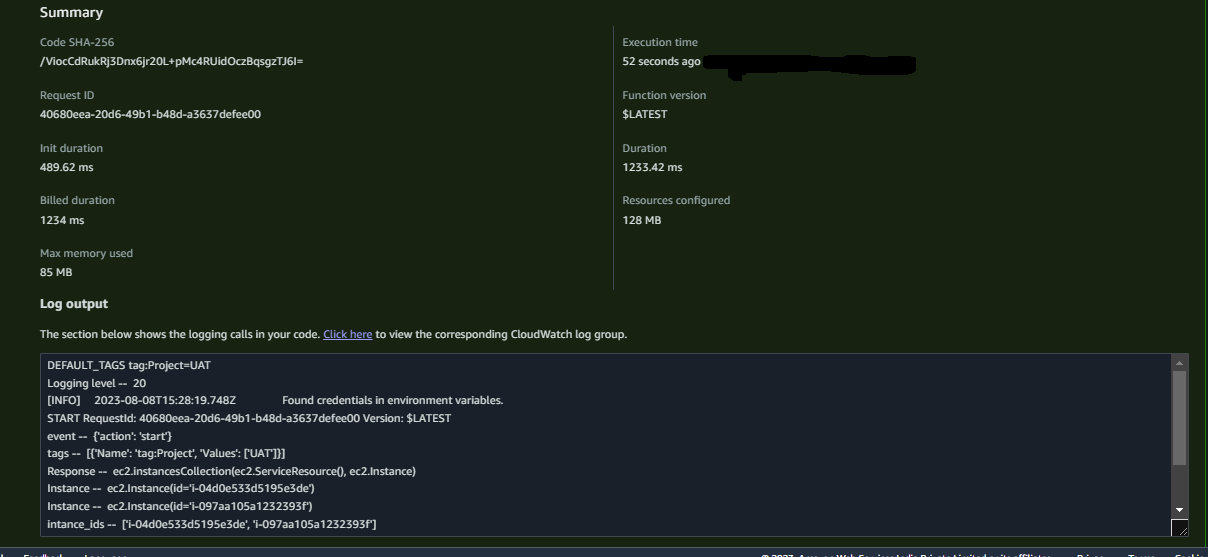
    Description automatically generated**Upon clicking **Create function** button, the **Function overview** dashboard will open, displaying a visual representation of your Lambda function.
12. Scroll down. Under **Function overview**, choose **Code** tab.
13. Under **Code source**, select and delete the pre-written code. Now, paste the copied Lambda function code from GitHub repository.
14. Choose **Deploy** button.
15. Even though the Lambda function has been deployed, it will not function properly since the necessary **environment variables**, such as **DEFAULT\_TAGS** and **LOG\_LEVEL**, have not been set.
16. So, now go to **Configuration** tab (the third tab next to **Code** tab). Choose **Environment variables** from the left pane and then click on **Edit** tab.
17. **A screenshot of a computer

    Description automatically generated**A new window will open like the one shown below, choose **Add environment variables** and add the required environment variables and their values.
18. Click on the **Save** button to save your changes.
19. Navigate to the **General configuration** section within the same **Configuration** tab and click on the **Edit** button.
20. **Scroll down the page** and set the **timeout value** to **10 seconds**.

**NOTE:** Setting the timeout value to 10 seconds provides a safety margin for the Lambda function to initiate and complete its tasks without premature timeouts. This ensures smooth execution even if the function requires a little more time to finish its operations.

1. Without making any further modifications, simply click on the **Save** button.

**Invoking functions with test events**

1. To configure a **test event**, choose **Test** tab (right next to Code tab).
2. Set the **Test event action** to the default option, which is **Create new event**.
3. For **Event name**, enter **EC2-Start-Test** or whatever you like.
4. Leave all the settings be default and scroll down to **Event JSON**. Choose **Format JSON** and write **{“action”: “start”}**.
5. If you want to save it first, choose **Save** button or else choose **Test** button.
6. You’ll get an **execution result** just below the **Test** tab under the heading **Executing function: succeeded**.
7. Navigate to the **EC2 console** and **verify** whether the **tagged EC2 instances** have **successfully started running**.
8. In the provided image, the **log output** of the **test event** vividly illustrates the process. The **5th and 6th lines** provide details about the **designated tags and the test event** itself. Meanwhile, the **8th and 9th lines** reveal the **instance IDs of EC2 instances** associated with the specified tags, which were effectively stopped in line with the function role.
9. To configure a **stop test event**, do the same steps from **2-5**.
10. Just change the **Event name** in **Step 3** to **EC2-Stop-Test** and change the **Event JSON** to **{“action”: “stop”}**.
11. Once more, access the **EC2 console** to verify the **tagged EC2 instances** have been **stopped** successfully.

**“Fantastic job, fine-tuning your Lambda function!**

**With this complete, you're all set for the next exciting step. Keep up the great momentum as you continue your journey into the world of AWS.”**

**NOTE:** There is a little surprise for you on the next page.

**Effortlessly Extending EC2 Management to Diverse Instances**

1. If you intend to **start/stop** different EC2 instances, tagged with distinct criteria (If your EC2 instances lack tags, take the initiative to tag them now), you can achieve this by specifying the unique tags associated in Format JSON as **“tags”: “tag: key=value”**, without any spaces.
2. Replace the **key** and value by your EC2 instance Key-Value pair.

A screenshot of a computer

Description automatically generated**For example:**

1. In my case, I had another EC2 instance named as Prodsrv \_1 with the tags:

**Project =Prod** (Key-Value pair)

So, I have mentioned it as you can see in the above picture.

1. Now, go back to **EC2 management console** and check the currently mentioned tagged **EC2 instance/instances** have been stopped.

**“With your newfound knowledge and the power of tailored tagging, you're now equipped to expertly manage a diverse array of EC2 instances.”**