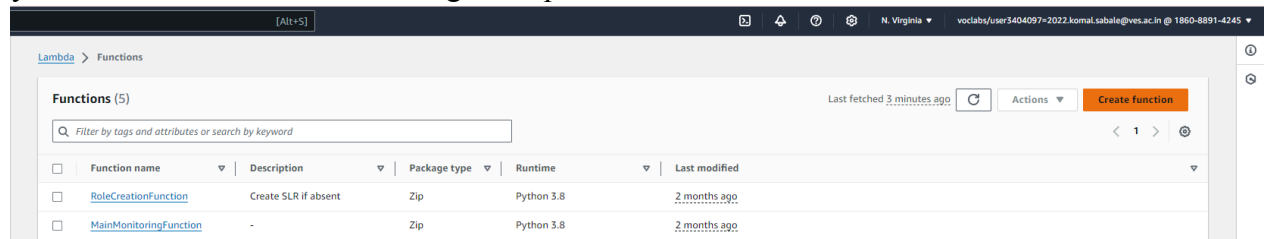


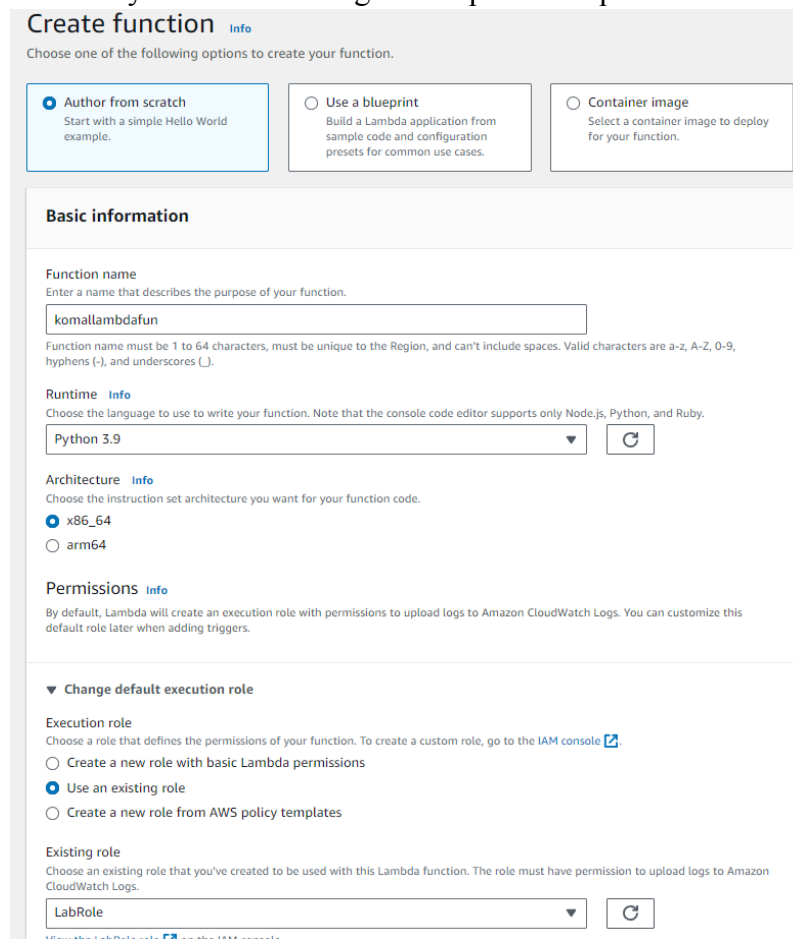
Experiment 11

Steps to create an AWS Lambda function

1. Open up the Lambda Console and click on the Create button. Be mindful of where you create your functions since Lambda is region-dependent.



2. Choose to create a function from scratch or use a blueprint, i.e templates defined by AWS for you with all configuration presets required for the most common use cases.



The screenshot shows the 'Create function' wizard in the AWS Lambda console. It has three tabs: 'Basic information', 'Permissions', and 'Change default execution role'. The 'Basic information' tab is active.

Create function [Info](#)

Choose one of the following options to create your function.

- ☒ **Author from scratch**
Start with a simple Hello World example.
- ☐ **Use a blueprint**
Build a Lambda application from sample code and configuration presets for common use cases.
- ☐ **Container image**
Select a container image to deploy for your function.

Basic information

Function name
Enter a name that describes the purpose of your function.

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.
☒ x86_64
☐ arm64

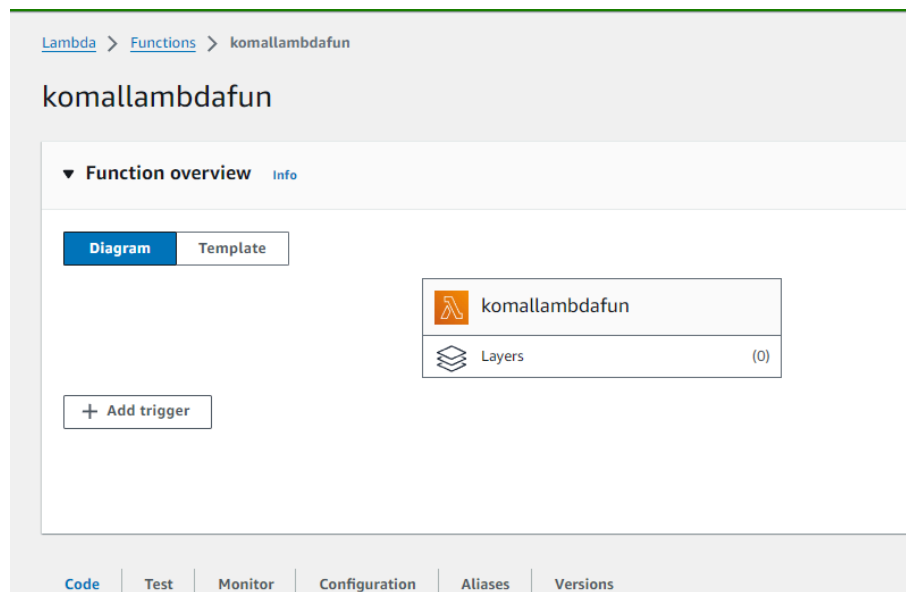
Permissions [Info](#)
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

Change default execution role

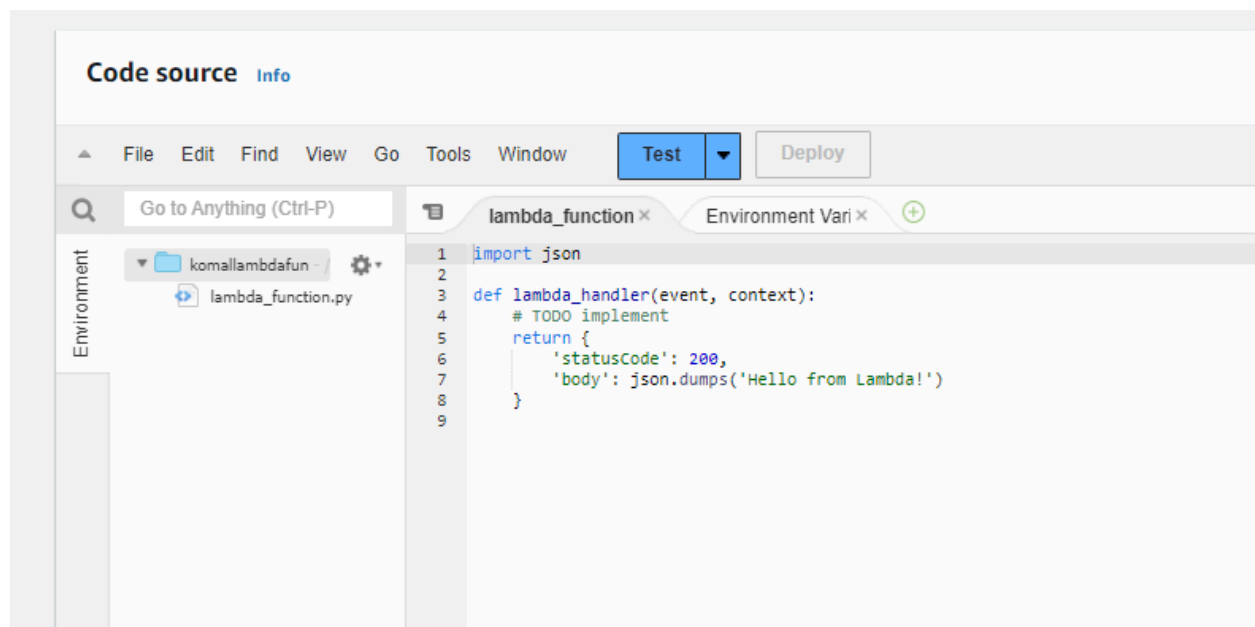
Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

- ☐ Create a new role with basic Lambda permissions
- ☒ Use an existing role
- ☐ Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.



3. This process will take a while to finish and after that, you'll get a message that your function was successfully created.



4. To change the configuration, open up the Configuration tab and under General Configuration, choose Edit.

Edit basic settings

Basic settings [Info](#)

Description - *optional*

Memory [Info](#)
Your function is allocated CPU proportional to the memory configured.
 MB
Set memory to between 128 MB and 10240 MB

Ephemeral storage [Info](#)
You can configure up to 10 GB of ephemeral storage (/tmp) for your function. [View pricing](#) [↗](#)
 MB
Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

SnapStart [Info](#)
Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the [SnapStart compatibility considerations](#) [↗](#)
None ▼
Supported runtimes: Java 11, Java 17, Java 21.

Timeout
 min sec

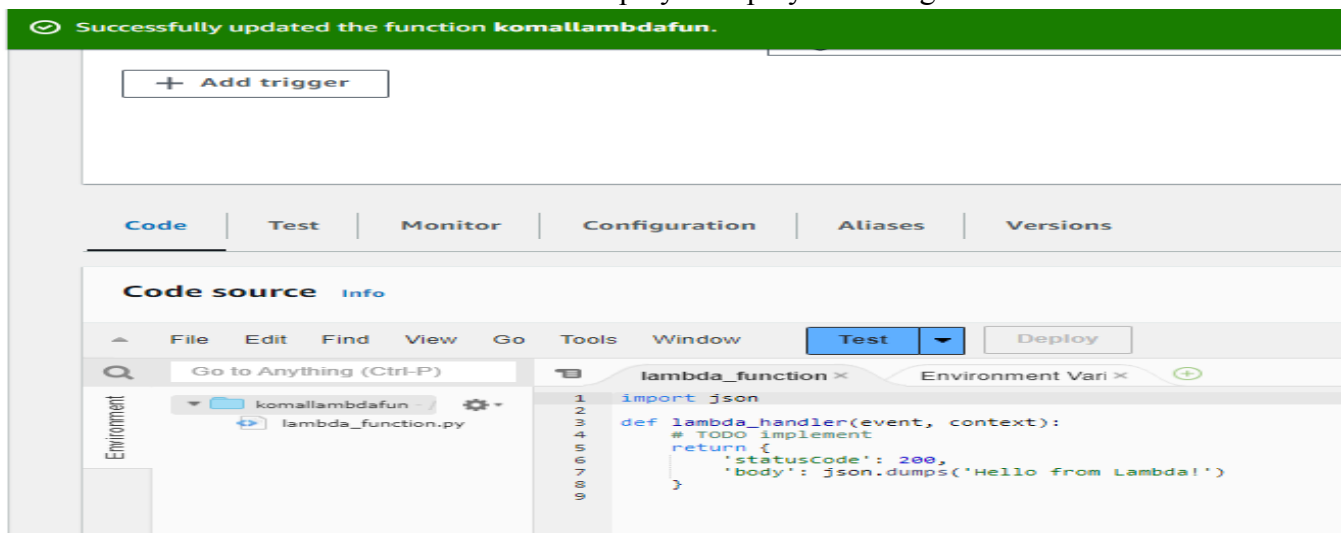
Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#) [↗](#).

☒ Use an existing role
☐ Create a new role from AWS policy templates


Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

[View the LabRole role](#) [↗](#) on the IAM console.

5. You can make changes to your function inside the code editor. You can also upload a zip file of your function or upload one from an S3 bucket if needed. Press Ctrl + S to save the file and click Deploy to deploy the changes



- Click on Test and you can change the configuration, like so. If you do not have anything in the request body, it is important to specify two curly braces as valid JSON, so make sure they are there

 Executing function: succeeded ([logs](#))
▶ Details

Test event [Info](#)

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

☒ Create new event

Event name

komalevent

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

☒ Private
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☐ Shareable
This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

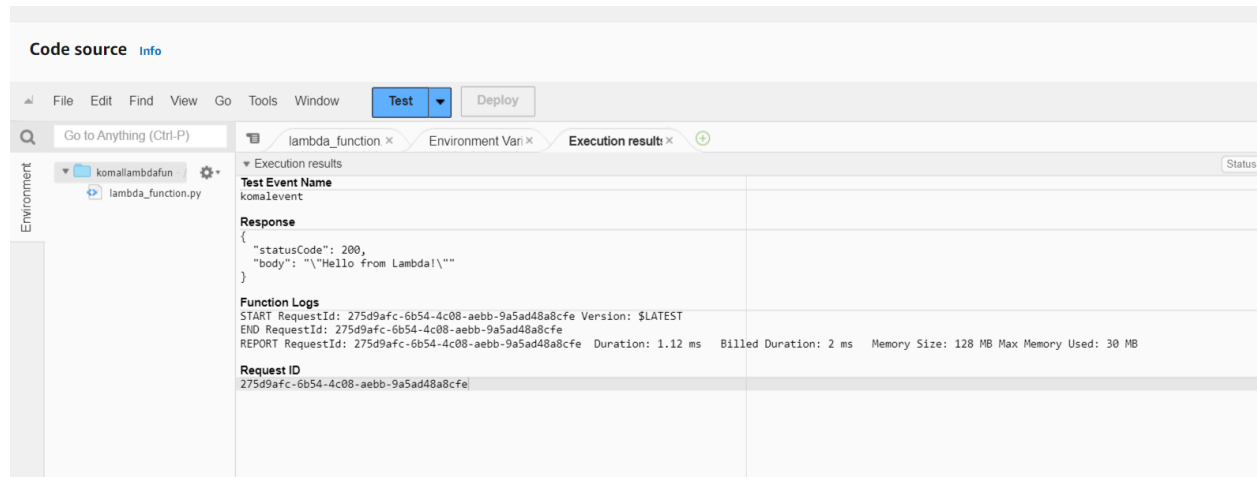
Template - optional

hello-world

Event JSON

```
1 {  
2   "key1": "value1",  
3   "key2": "value2",  
4   "key3": "value3"  
5 }
```

7. Now click on Test and you should be able to see the results.



Conclusion:

AWS Lambda automatically manages the compute resources, executes your code in response to specific events such as API calls, file uploads, or database updates, and scales based on the demand. The workflow of AWS Lambda involves defining a function with specific logic, configuring triggers that will invoke the function, and setting permissions to control access. Lambda supports multiple programming languages, including Python, Java, and Node.js, enabling developers to choose the best fit for their applications. Creating your first Lambda function is straightforward: you write the code, define triggers, and deploy, allowing you to quickly build and run applications without the overhead of managing infrastructure. This simplicity and flexibility make AWS Lambda an excellent tool for building modern, event-driven applications.