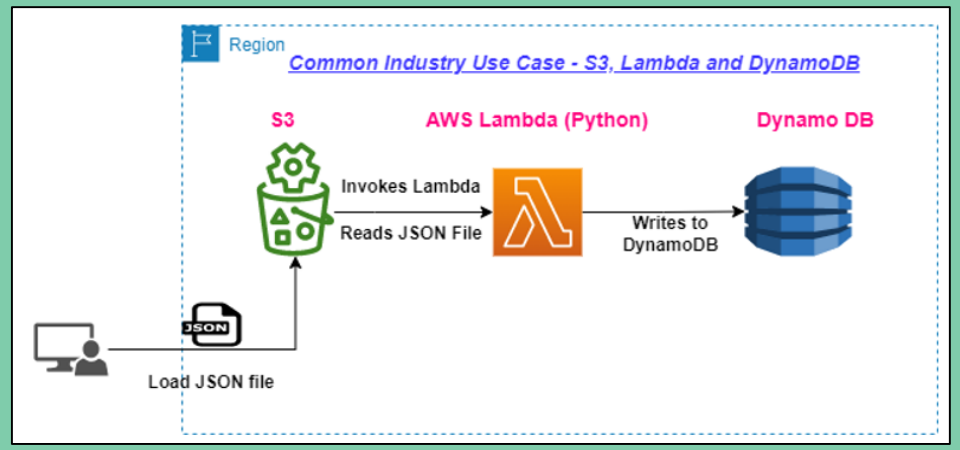
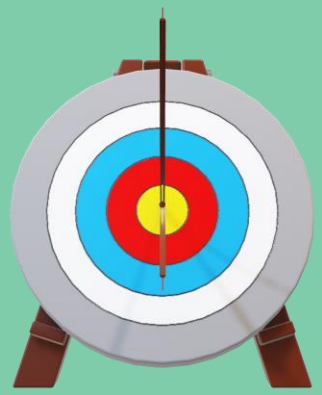
AWS Lambda and Python - Full Course
Absolute Beginner to Advanced

No previous Coding Experience Required





Develop End to End Solution with AWS Lambda

AWS Lambda and Python – Beginner to Advanced

7 Months on…. **800+ Learners** **100+ Ratings**

D
DineshEzeikel
Updated 2 months ago

★★★★★

Excellent course.

BN
basam nath
Updated 4 months ago

★★★★★

Just superb videos

EK
Ebrahim Kamari
Updated 7 months ago

★★★★★

This course helped me alot. Thanks Rahul

TA
Terence Atarah
Updated 2 months ago

★★★★★

Yes it was. It gives a clear insight on how to use lambda with python. I have struggled with this over the years and i am finally understanding it from a 3 hour course...Amazing

SJ
Shashank j
Updated 5 months ago

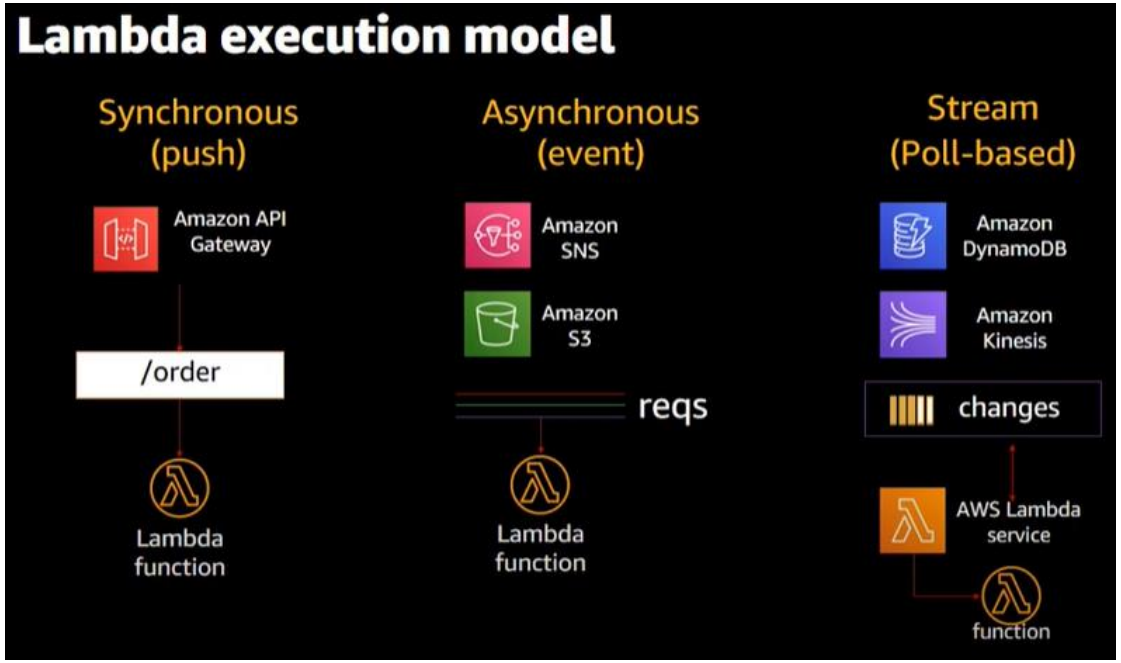
★★★★★

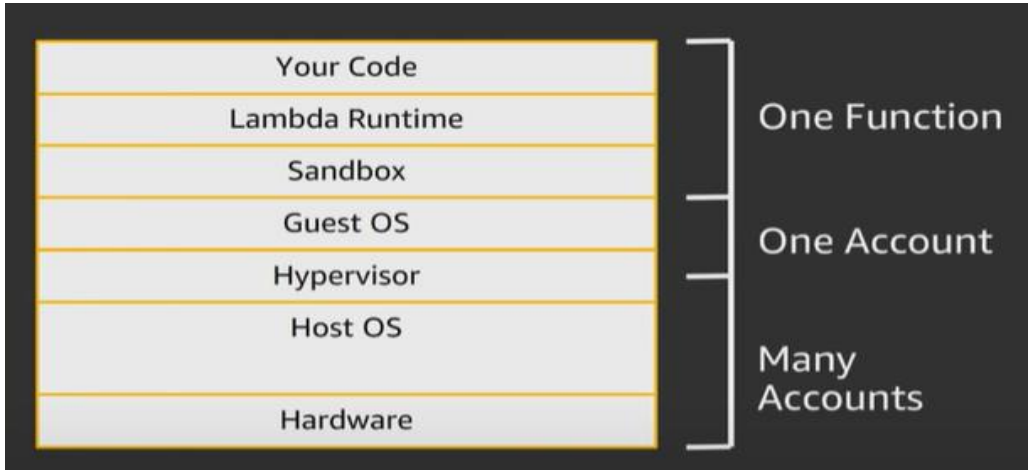
best course i have ever taken on AWS lambda. The instructor taught me lot of good stuff that i use every day

AWS Lambda and Python – Beginner to Advanced

Section 2 : AWS Lambda – Basic Concepts 1

- **Evolution from Physical Servers to AWS Lambda**
- **What is AWS Lambda - Architecture and Use Cases**
- **Lambda Console Walkthrough**
- **AWS Lambda Invocation Models - Theory**
- **AWS Lambda Invocation Models - HandsOn**
- **AWS Lambda Limits and Pricing**





AWS Lambda and Python – Beginner to Advanced

Section 3 : AWS Lambda - Python Basics

Python Basics – 1 (Pre-Req– Install PyCharm, Print Function, Variables, .Format, User Input,)

Python Basics – 2 (Data Types Intro, Loops – For, and Data Type – Dictionary Deep Dive)

Python Basics – 3 (Data Type – List and Functions)

```python
####Nested Dictionary

nesteddictionary = {1: 'Python', 2: {'books': 'cloud', 'aws': 'Lambda'}}
print(nesteddictionary[2])
print(nesteddictionary[2]['books'])

### Loops and Dictionary Methods

for k in nesteddictionary.values():
    print(k)

for k in nesteddictionary.keys():
    print(k)
#######8 user input

name = input('Please enter your name')
grade = input('Please enter your grade')
print('My name is {} and my grade is {}'.format(name, grade))


##### 9 String Slicing
data = 'ASTRING'
print(data[2:4:1])

for a in data:
    print(a)


# A simple Python function to check
# whether x is even or odd
```

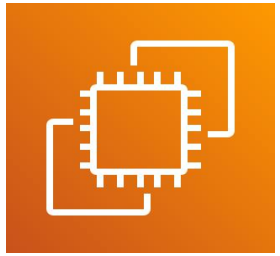
```python
def evenOdd(x):
    if (x % 2 == 0):
        print("even")
    else:
        print("odd")


# Driver code to call the function
evenOdd(2)
evenOdd(3)


def square_value(num):
    """This function returns the square
    value of the entered number"""
    return num ** 2

print(square_value(2))
print(square_value(-4))
```

AWS Lambda and Python – Beginner to Advanced

Section 4 : AWS Lambda – Create S3, EC2 and DynamoDB resources using Lambda (Boto3)

- **(Pre – Req) How to set up AWS Free Account and AWS Lambda Basics**
- **AWS Lambda with S3 (List all the buckets, Create new Bucket and Delete Bucket)**
- **AWS Lambda with EC2 (Create EC2 and Start/Stop)**
- **AWS Lambda with DynamoDB (Create Table and Put Items)**

S3

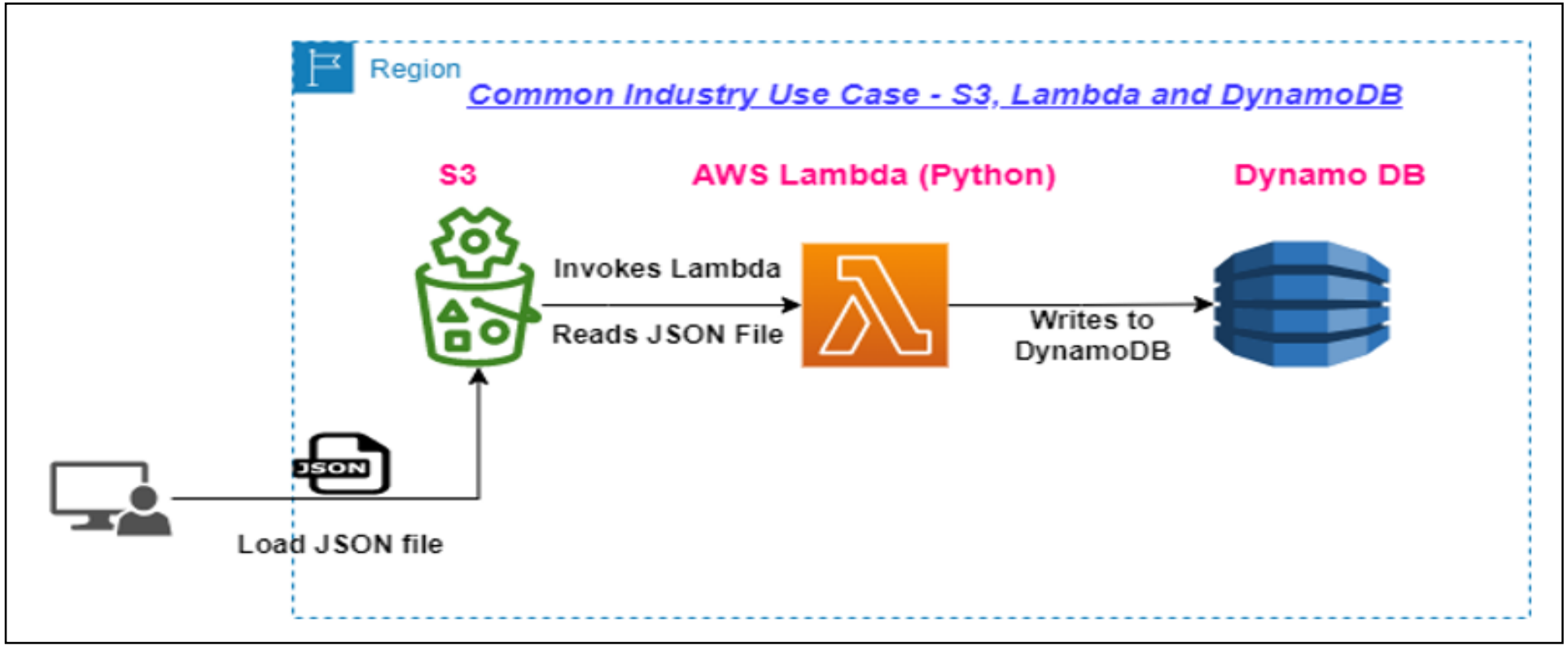


EC2

DynamoDB

AWS Lambda and Python – Beginner to Advanced

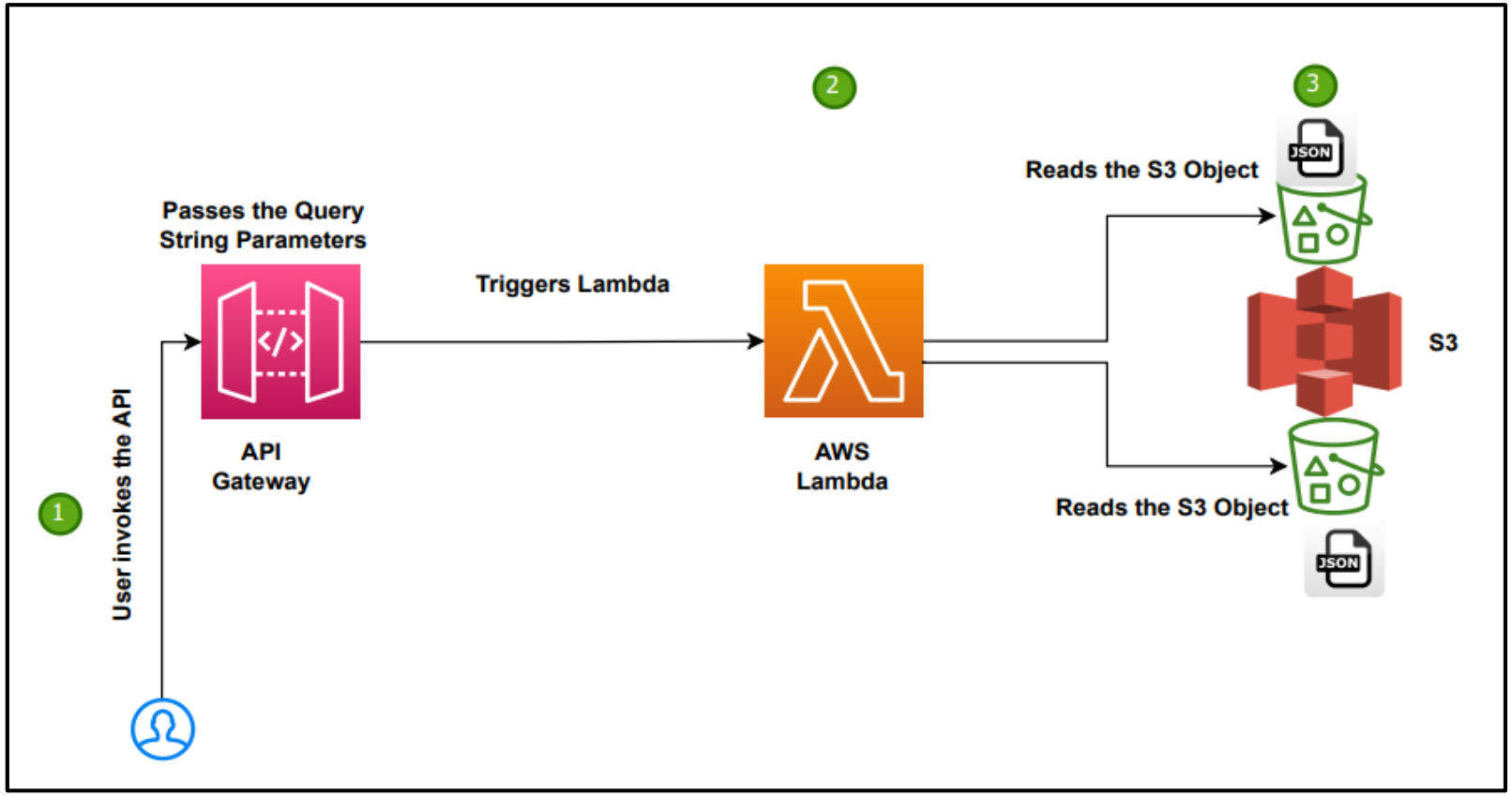
Section 5 : Real World Serverless Use Case 1 - using S3, AWS Lambda and DynamoDB



- *S3*
- *AWS Lambda*
- *DynamoDB*

AWS Lambda and Python – Beginner to Advanced

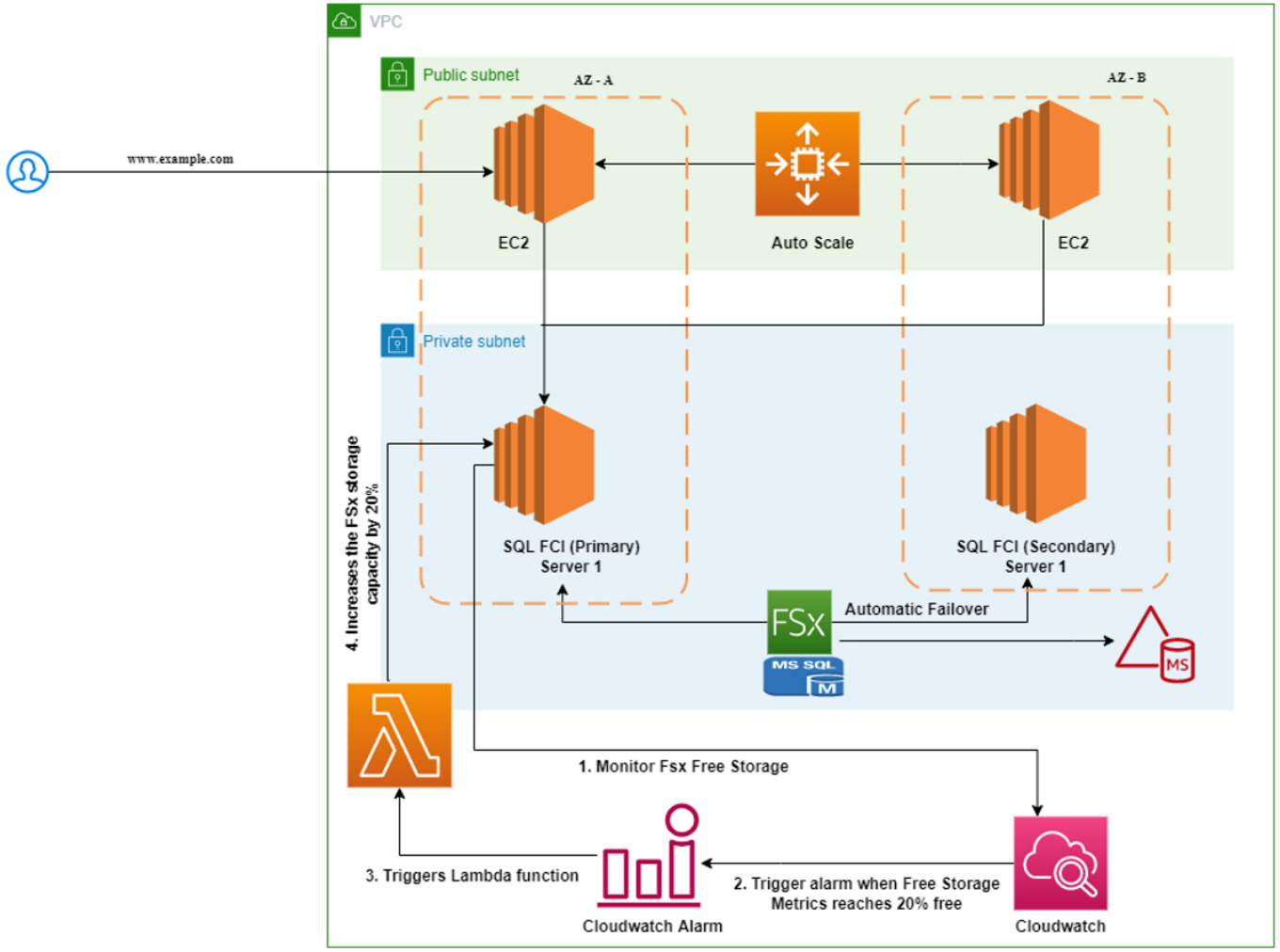
Section 6 : Real World Serverless Use Case 2 - using API Gateway, AWS Lambda and S3



- *API Gateway*
- *AWS Lambda*
- *S3*

AWS Lambda and Python – Beginner to Advanced

Section 7 : Use Case 3 - Monitor & increase free storage for SQL Server FCI Cluster using AWS Lambda- MS AD, EC2, FSx, CloudWatch and CloudWatch Alarm



- *Microsoft AD*
- *EC2*
- *FSx*
- *CloudWatch Metrics*
- *CloudWatch Alarm*

AWS Lambda and Python – Beginner to Advanced

Section 8 : AWS Lambda – Basic Concepts 2

- **AWS Lambda Concurrency - Cold Start Problem, Provisioned concurrency and Reserved concurrency**
- **Handler Function, Events and Context**
- **AWS Lambda VPC**
- **Lambda function – Use Cloud Watch Logs and CloudWatch Metrics**
- **Environment Variables**
- **Setting up DLQ and lambda Layers**

Section 2

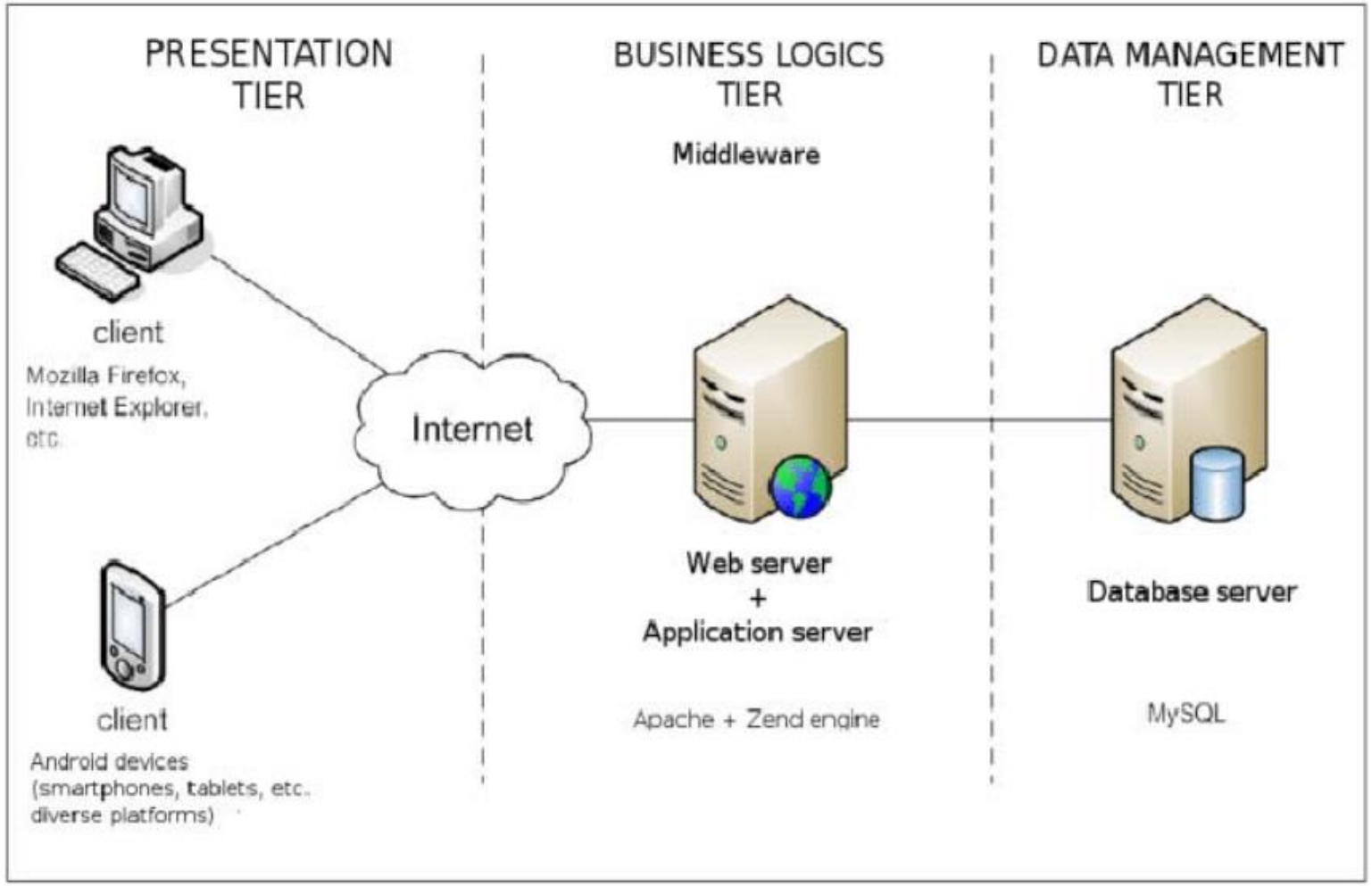
AWS Lambda – Basic Concepts 1

AWS Lambda and Python – Beginner to Advanced

AWS Lambda – Basic Concepts

- Evolution from Physical Servers to AWS Lambda
- What is AWS Lambda and Use Cases
- Lambda Console Walkthrough
- AWS Lambda Invocation Model
- AWS Lambda Limits and Pricing
- Handler Function, Events and Context
- Lambda function – Use Cloud Watch Logs and CloudWatch Metrics

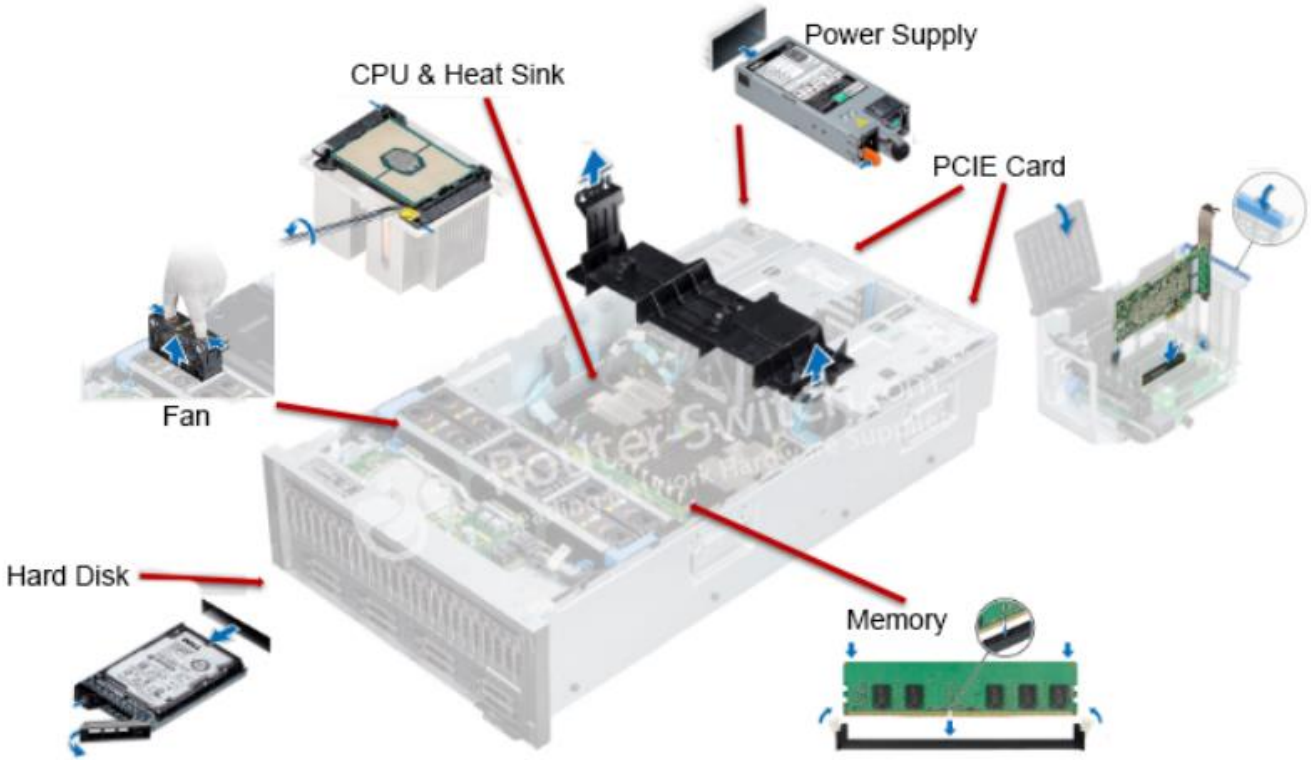
Three-Tier Architecture



3-tier architecture

Source : Internet

Server



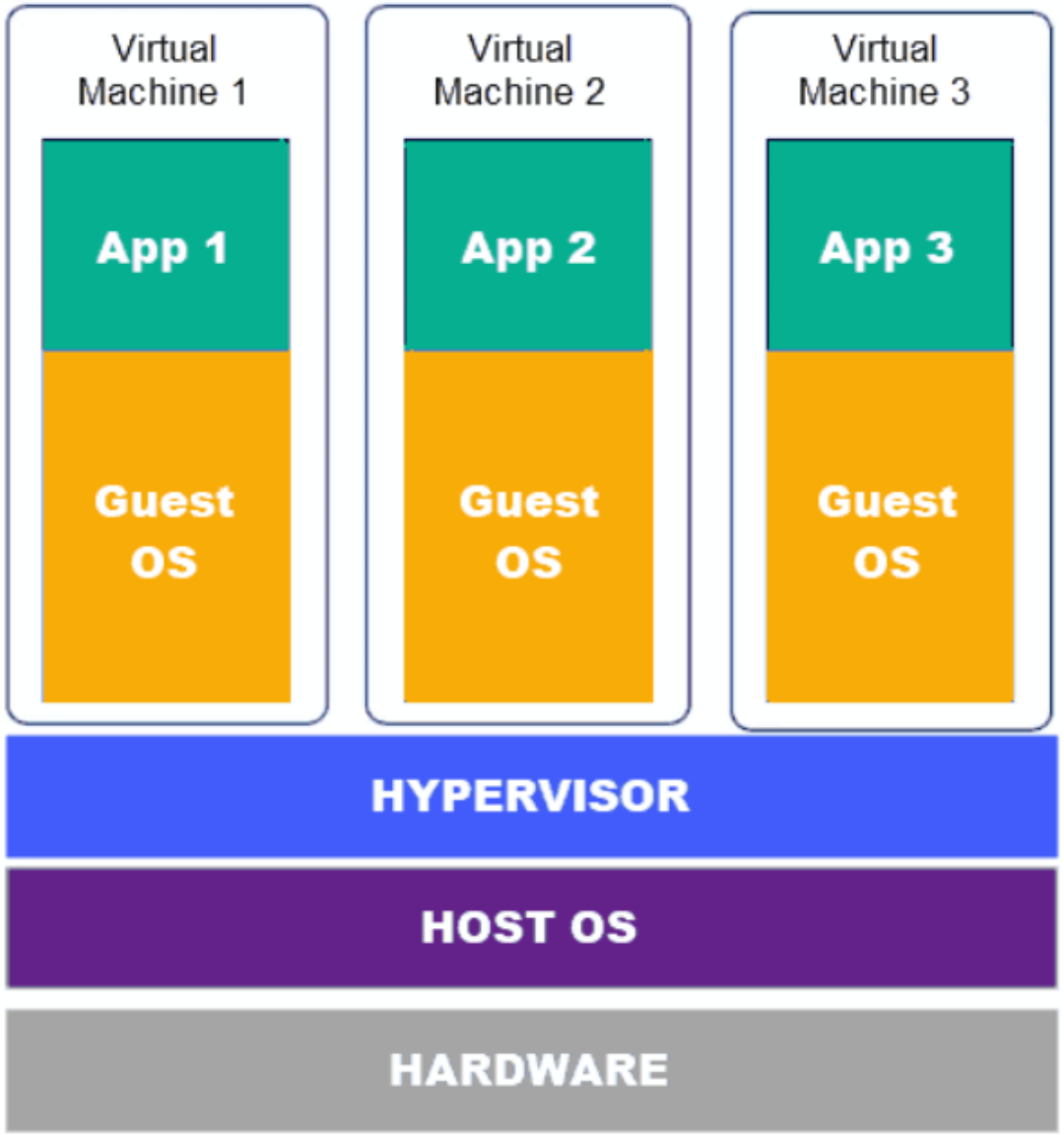
Source : Internet

Source : Internet

Source : Internet

Google data center in Eemshaven, Netherlands | Image credit: Google

Virtualization

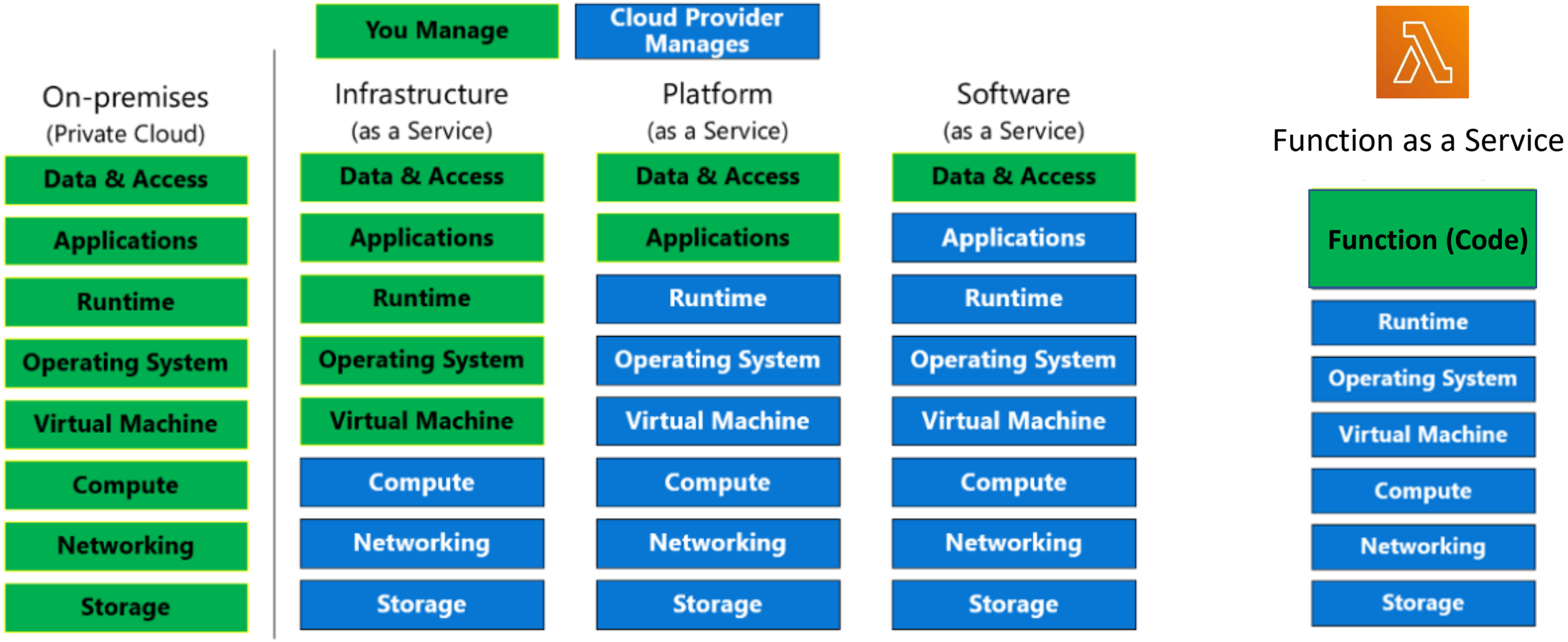


What is Cloud Computing ?

Cloud computing is **the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing**.

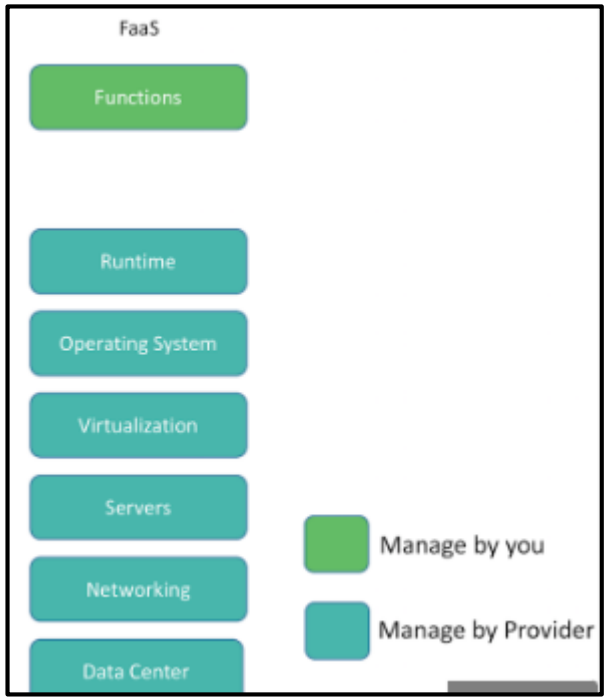
Source : Internet

IaaS vs PaaS vs SaaS and FaaS



Source : Internet

Function as a Service – AWS Lambda



Source : Internet

AWS Lambda and Python – Beginner to Advanced

AWS Definition of Lambda

Lambda is a **compute service** that lets you **run code without provisioning or managing servers**. Lambda **runs your code on a high-availability** compute infrastructure and performs all of the **administration of the compute resources**, including server and **operating system maintenance, capacity provisioning and automatic scaling, and logging.**

Key Features of Lambda

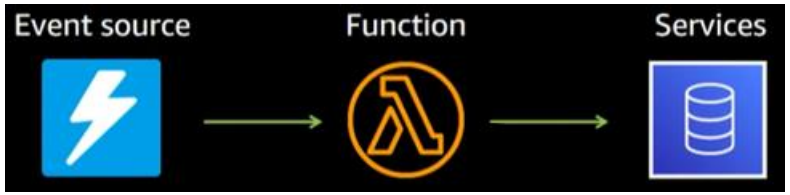
- Compute Service
- Highly Available
- All of the administration of the compute resources, including server and operating system maintenance, automatic scaling, and logging.
- Provisioning done by AWS and Pay as you use
- Bring your code and run on Lambda
- **Lambda is a serverless, event-driven compute service**

Source : AWS

AWS Lambda and Python – Beginner to Advanced

Use Case for AWS Lambda

- **Event Driven**
- **Unpredictable demand**



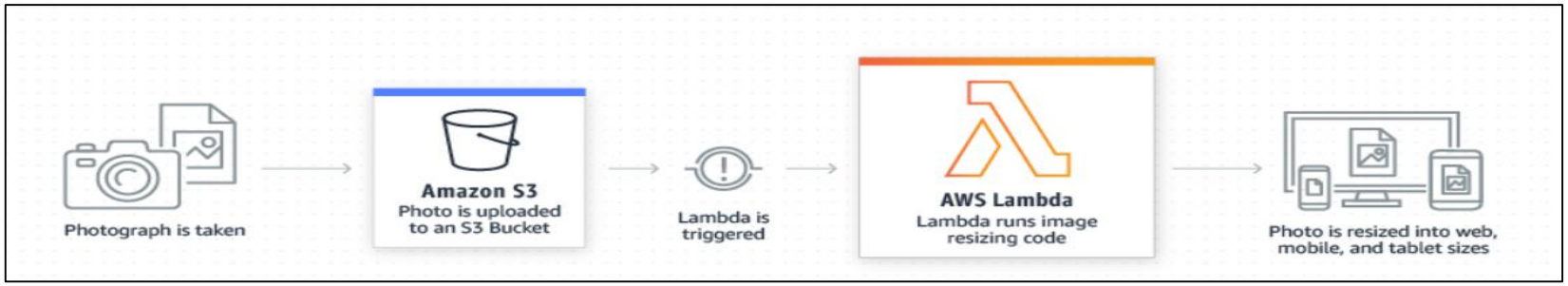
When not to use a AWS Lambda

- **When you need to manage the Infrastructure**
- **Need to run the Server Continuously instead of event driven**

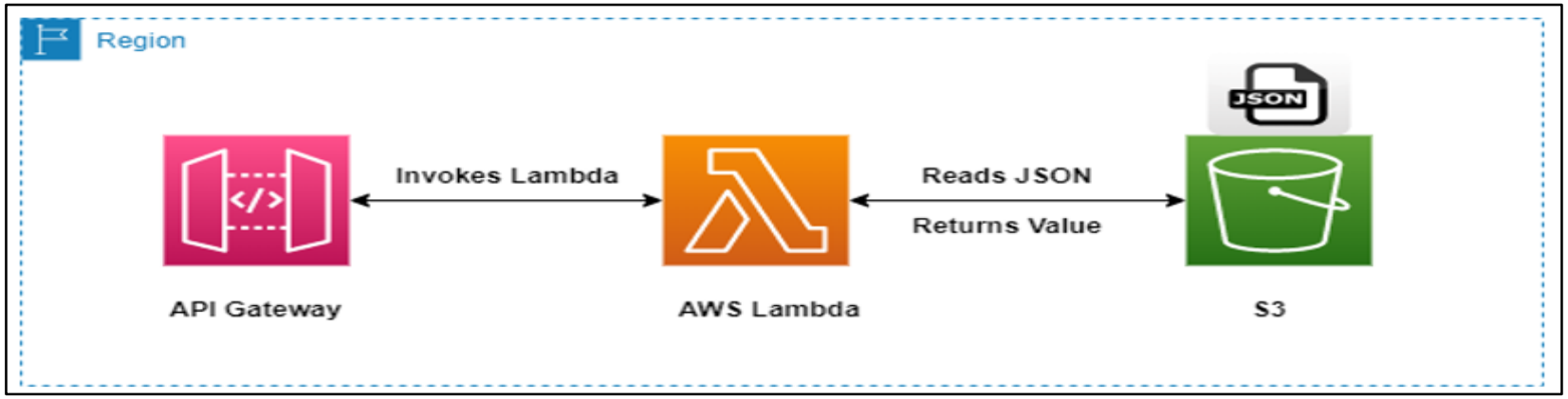
Source : AWS

AWS Lambda and Python – Beginner to Advanced

Image Processing – S3 and AWS Lambda



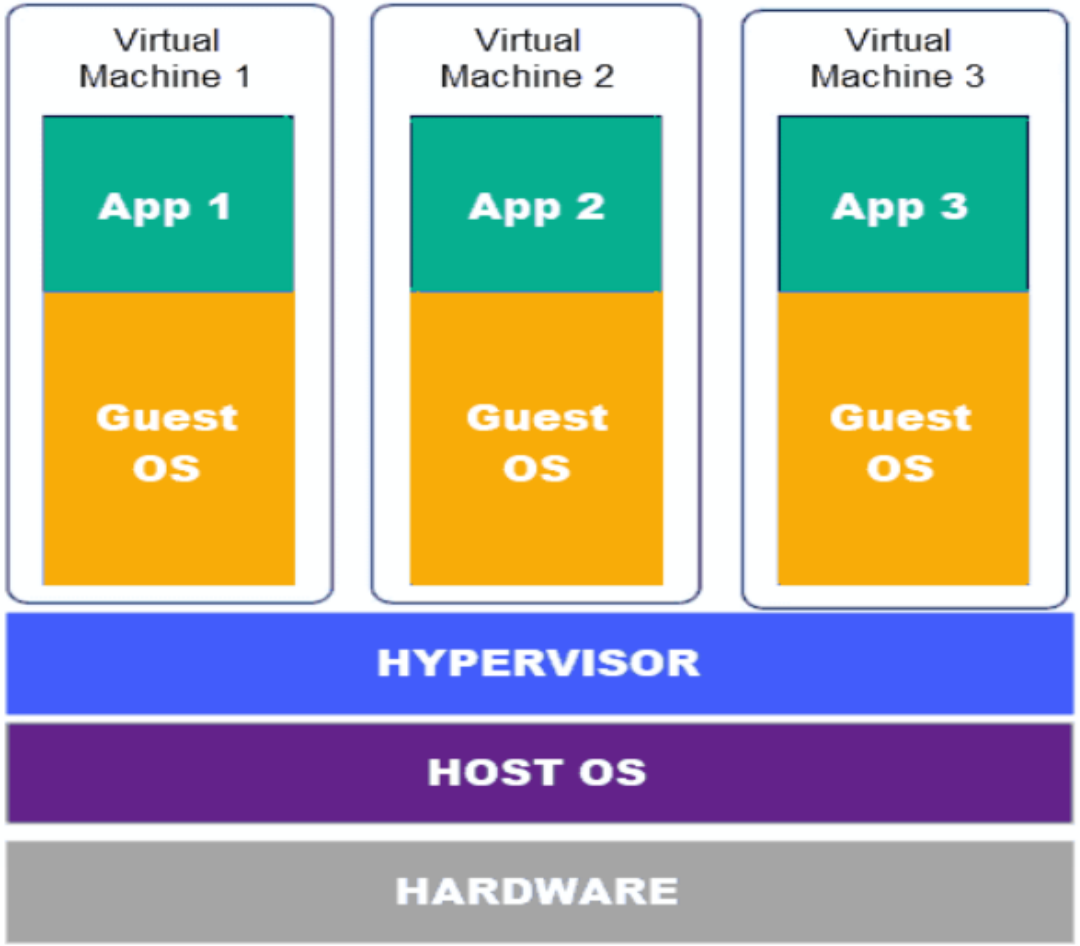
Build REST API with API Gateway, Lambda and S3



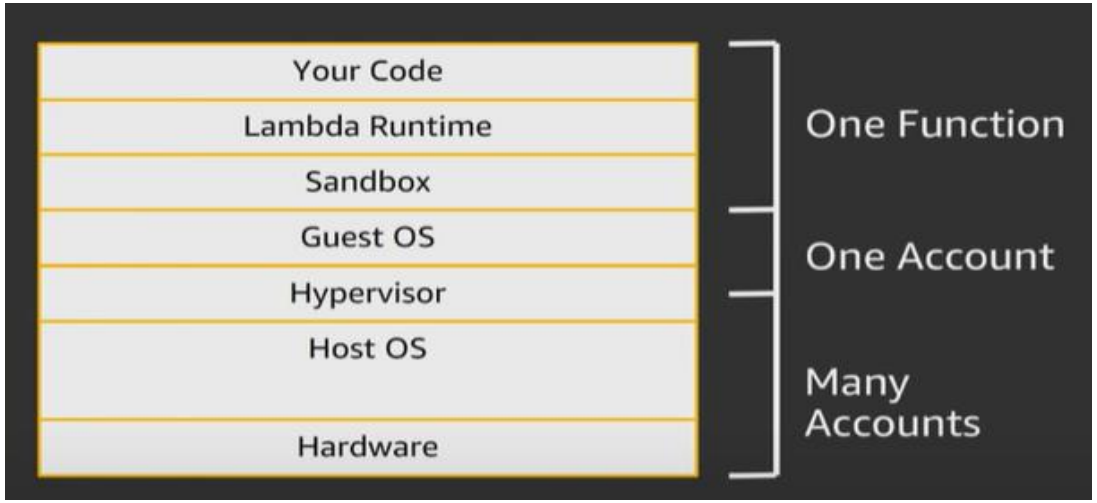
Source : AWS

Virtualization

EC2 Instance



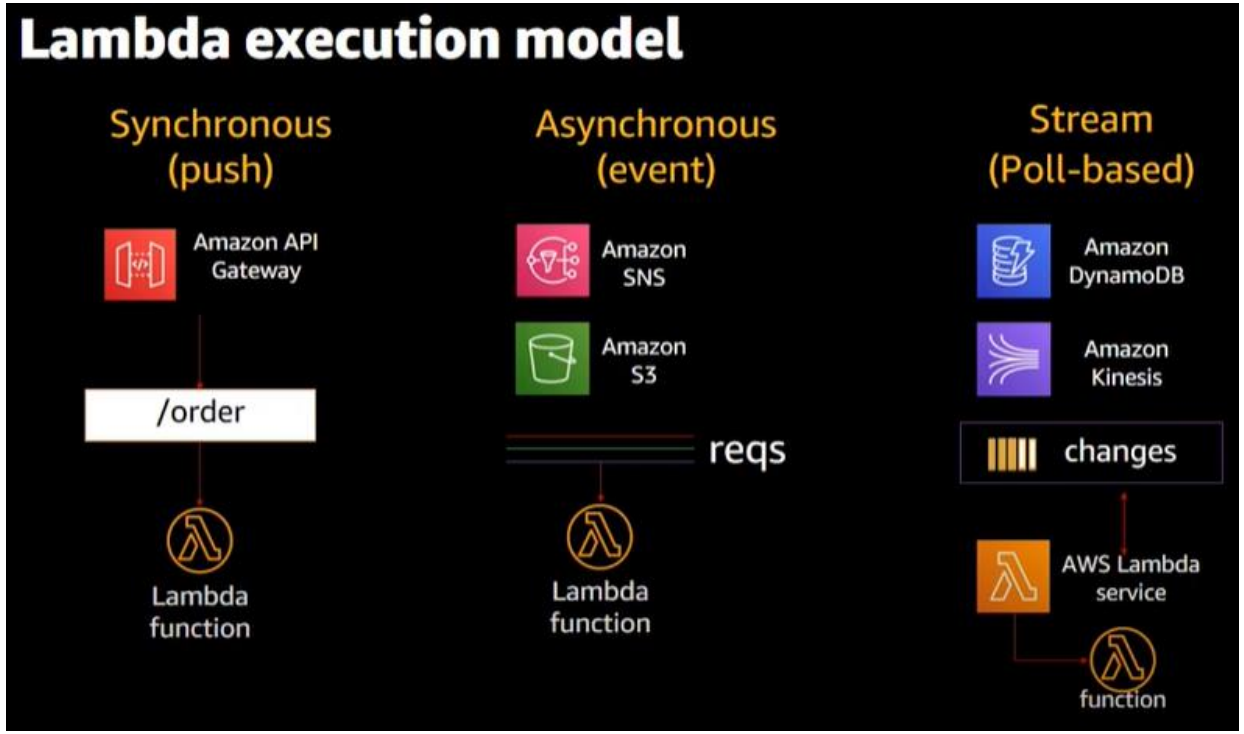
AWS Lambda



Source : Internet

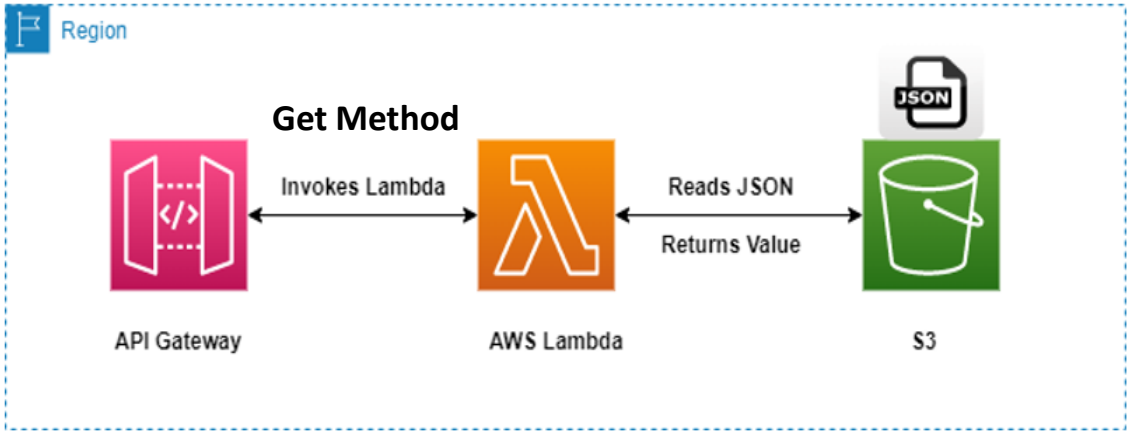
AWS Lambda and Python – Beginner to Advanced

3. AWS Lambda Execution/Invocation Model

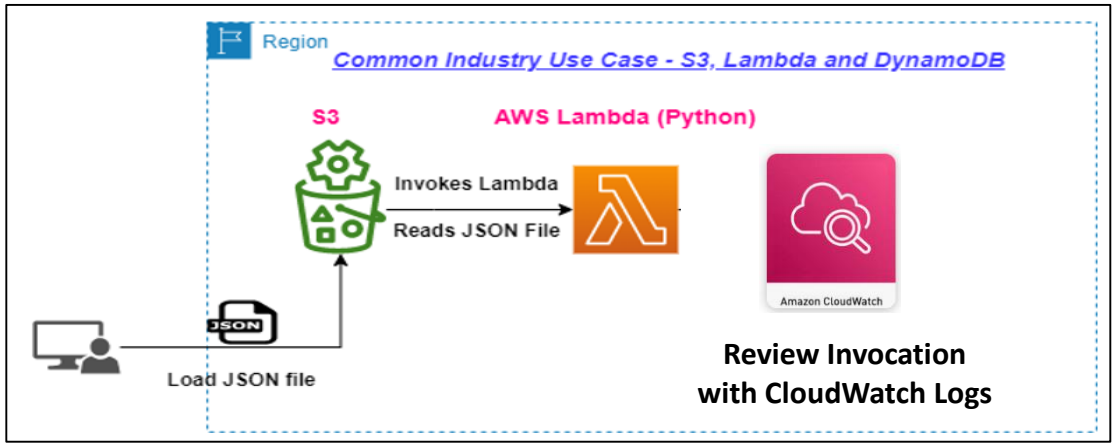


Source : AWS

1. Synchronous Invocation – API Gateway with AWS Lambda

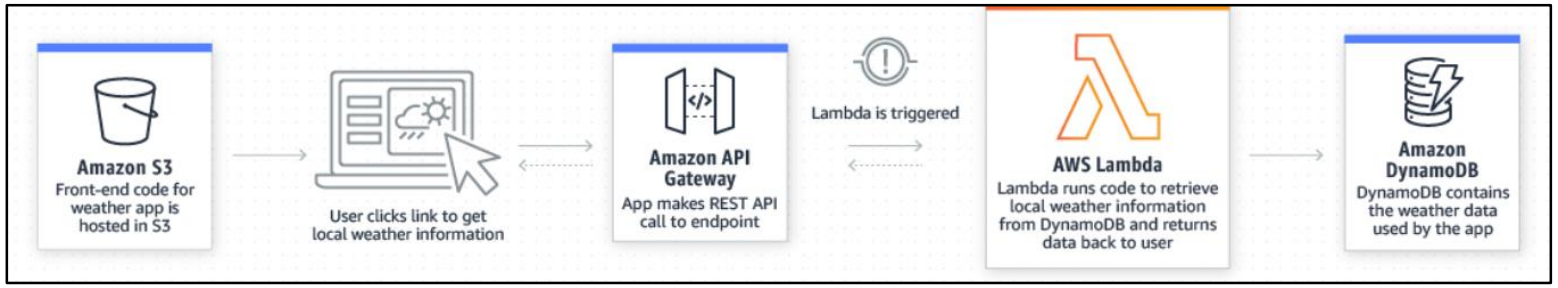


2. Asynchronous Invocation – S3 with AWS Lambda

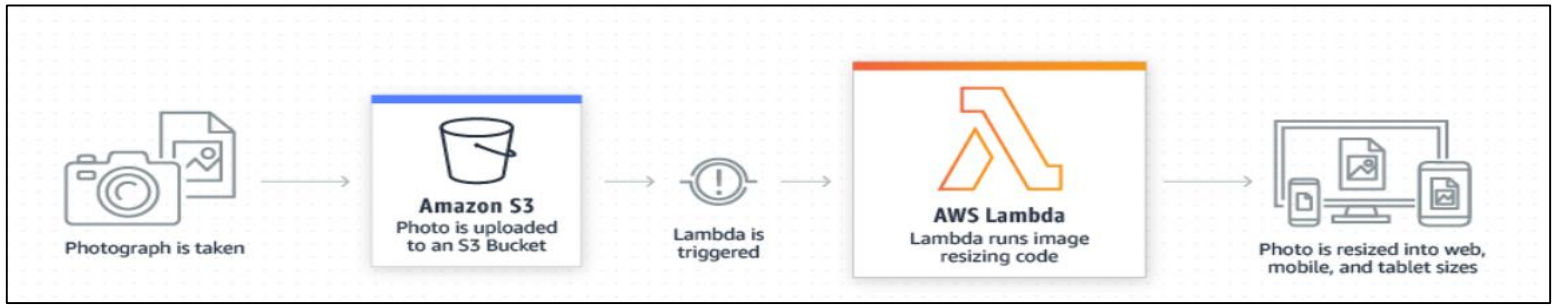


AWS Lambda and Python – Beginner to Advanced

1. Synchronous Invocation – API Gateway with AWS Lambda



2. Asynchronous Invocation – S3 with AWS Lambda



3. Stream based (Polling) Invocation – Kinesis or SQS with AWS Lambda

AWS Lambda and Python – Beginner to Advanced

4. AWS Lambda Limits and Pricing

- **Timeout – 15 min**
- **Memory – 10 GB (more memory = more CPU)**
- **Concurrent Invocations – 1000 for most Regions (500 to 3000 varies per Region)**
- **Lambda Pricing – Memory Allocated * Execution time of function*number of invocations**

Read further Links

- **Lambda Limits - https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html**
- **Lambda Pricing - https://aws.amazon.com/lambda/pricing/**

Section 3

AWS Lambda - Python Basics

Python Basics – 1 (Pre-Req– Install PyCharm)

1. **Install PyCharm(Community edition) – Free**

https://www.jetbrains.com/pycharm/download/#section=windows

2. **Get a free tier AWS account – Free**

https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all

Python Basics – 1 (Print Function, Variables, .Format, User Input)

1. Print Function – print the message to screen or any interface; Syntax : print( )

```
>> print("Hello Rahul")
>> print(30.5)
```

2. Variable - Containers for storing data values string, float or integers and no need to declare; Syntax : x = 3, greeting = "hello" etc.

```
>> demo = "hi"
>> x = 30
```

3. User Input - Allow user to provide an input; Syntax - input(" ")

```
>> schoolName = input("Please enter your School Name")
>> grade = input("Please enter your Grade")
>>  print("My schoolName is : {}  and grade is {}".format(schoolName, grade))
```

4. Print variables in Strings---- 'format()' method - .format(a,b)

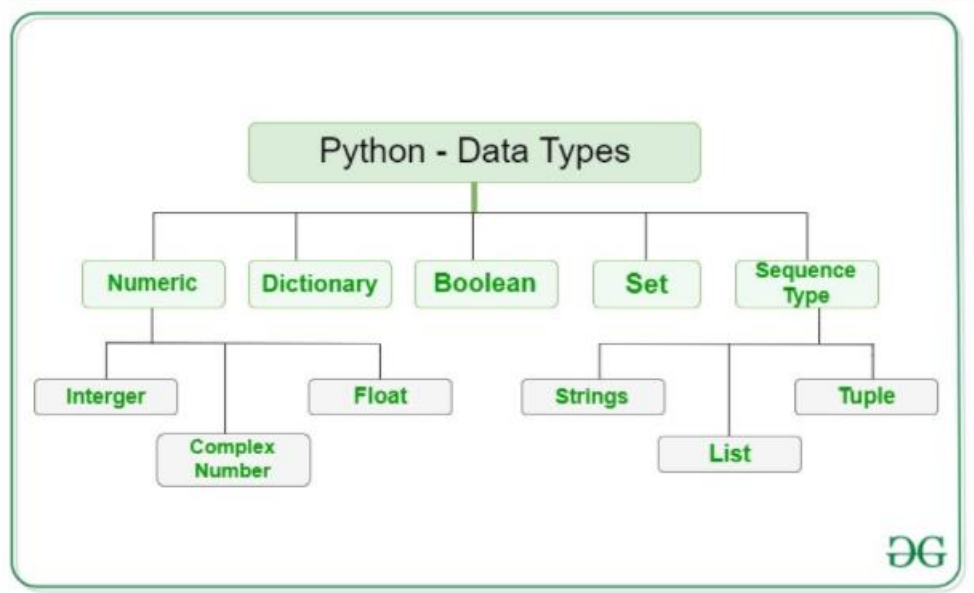
```
>>  grade = 4,
>> section = 'A'
>>  print("My class is : {}  and section is {}".format(grade, section))
```

Python Basics – 2 (Data Types Introduction)

5. Data Types Introduction

Data types are the **classification or categorization of data items**. It represents the **kind of value that tells what operations can be performed on a particular data**.

- **Numeric (Integer, Complex Number, Float)**
- **Dictionary – Most Important;** Key-value pairs; Dict = {1: ‘Rahul', 2: ‘John', 3: ‘Joy’}
- Boolean (True or False)
- Set - Sets are used to store multiple items in a single variable; fruits = {"apple", "banana", "cherry"}
- Sequence Type (String, List and Tuple)
- **Strings** is a sequence of characters internally stored as binary (“Aaron”)
    - ASCII value of the letter 'A' is 65.
- **List -** Lists in Python can be created by just placing the sequence inside the square brackets[]
    - [“Rahul”, “John”, “Joy”]



Source : Internet

6. Determine Data Type – **Syntax - type(variable)**

```
>> type(variable)
```

Python Basics – 2 (Loops and Slicing)

7. Loop (for loop)

For loops are used for sequential traversal. For example: traversing a list or string or array etc.

>> data = “length”

Syntax :

>> **for k in data :**

print(k) **-----> k, colon, spacing (4 or 5)**

8. String length function - returns the length of a string: **Syntax – len()**

a = "Hello“

print(len(a))

9. String Slicing in Python

Python slicing is about obtaining a sub-string from the given string by slicing it respectively from start to end. **Syntax: var[start: stop: step]**

- data = "john"
- print(data[0:2:1])

0	1	2	3	4	5	6
A	S	T	R	I	N	G

Python Basics – 2 (Dictionary)

10. Data Types – Dictionary

- curly brackets
- key: pair values
- Its mutable but keys immutable
- Nested Dictionary

……

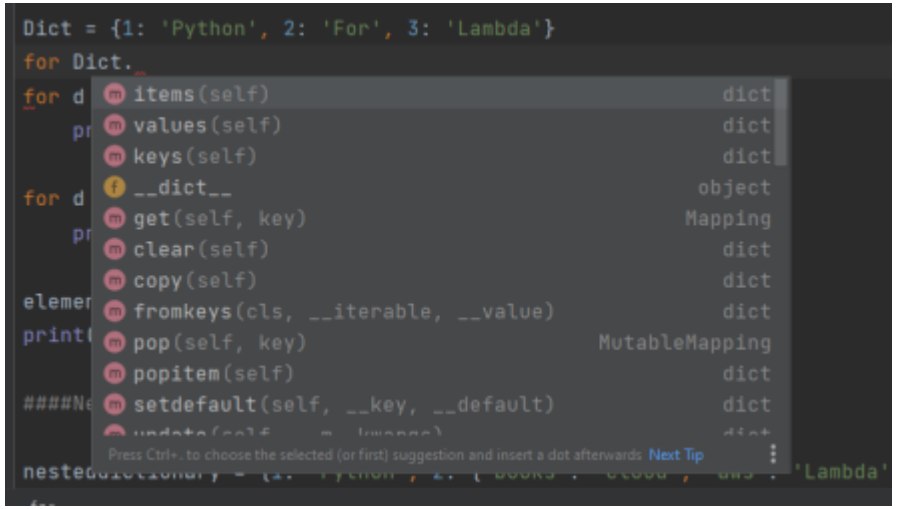
d = {1: 'Python', 2: 'For', 3: 'Lambda'}

Nested Dictionary

nd = {1:'Python', 2:{'books': 'arch' , 'aws':'Lambda'}}

……

- Items
- Keys
- Values
- **Elements in Dictionary – Getting specific values from Key Names**
- **Elements in Dictionary – Getting specific values from Key Names in a Nested Dictionary**
- **Adding an element to dictionary - d[3] = "red"**
- **Dictionary Methods**



Python Basics – 3 (List) with Loop and if/else statement

11. Data Types – List

- Lists in Python can be created by just placing the sequence inside the **square brackets []**
- A single list may contain Data Types like Integers, Strings, as well as Objects.
- List in Python are ordered and have a definite count. The elements in a **list are indexed with 0 being the first index.**
- slice(start, stop, step)
- Reverse [: : -1]

……

- l = [1, 4, 'For', 6, 'Anisha’]
- **Nested List** => nestedList = nestedList = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

……

- By positional value
  - list = [2,3,4] or [2]
  - k = list[0:2:1]
  - print(k)
- For Loop and if statement

AWS Lambda with Python – Basics - 3

12. Function

Python Functions is a block of related statements designed to perform a computational, logical, or evaluative task.

Syntax:

def function_name(argument/parameters):

return expression or value

Example

```
# A simple Python function to check whether x is even or odd

def evenOdd(x):
    if (x % 2 == 0):
        print("even")
    else:
        print("odd")
# Driver code to call the function
evenOdd(2)
evenOdd(3)
```

Section 4

AWS Lambda – Create S3, EC2 and DynamoDB resources using Lambda (Boto3)

Create S3, EC2 and DynamoDB – Introduction -1

- AWS' **Boto3 library** is used commonly to **integrate Python applications with various AWS services.**
- The two most commonly used features of boto3 are **Clients and Resources**.
- **Event driven** – S3 File Upload, S3 Event notification triggers the Lambda.

Client

- Clients provide a **low-level interface to the AWS service.**
- All AWS service operations supported by clients

Syntax

```
import boto3
s3 = boto3.client("s3")
```

…….

Resource

- Resources are a higher-level abstraction compared to clients.
- Few AWS service operations not supported

Syntax

```
import boto3
s3 = boto3.resource("s3")
```

Create S3, EC2 and DynamoDB – Introduction -1

Lambda function handler in Python

The **Lambda function *handler* is the method in your function code that processes events**. When your **function is invoked**, Lambda runs the **handler method**.

```
def handler_name(event, context):
    ...
    return some_value
```

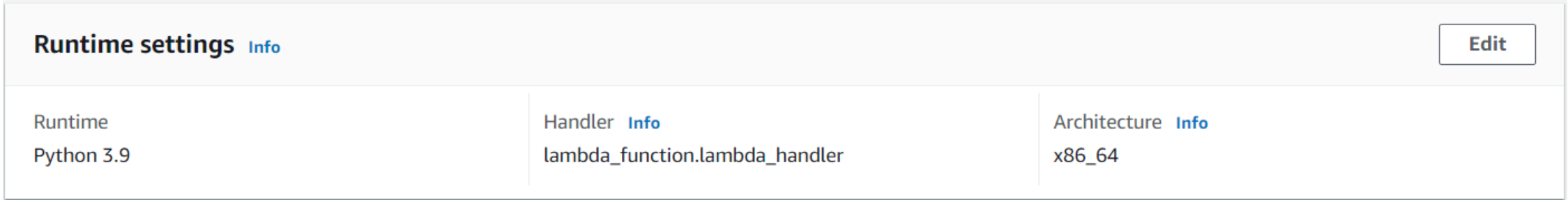
Naming

The Lambda function handler name specified at the time that you create a Lambda function is derived from:

The name of the file in which the Lambda handler function is located.

The name of the Python handler function.

A function handler can be any name; however, the default name in the Lambda console is lambda_function.lambda_handler. This function handler name reflects the function name (lambda_handler) and the file where the handler code is stored (lambda_function.py).



Create S3, EC2 and DynamoDB – Introduction - 1

Context properties

- function_name – The name of the Lambda function.
- function_version – The version of the function.
- invoked_function_arn – The Amazon Resource Name (ARN) that's used to invoke the function. Indicates if the invoker specified a version number or alias.
- memory_limit_in_mb – The amount of memory that's allocated for the function.
- aws_request_id – The identifier of the invocation request.
- log_group_name – The log group for the function.
- log_stream_name – The log stream for the function instance.
- identity – (mobile apps) Information about the Amazon Cognito identity that authorized the request.
- cognito_identity_id – The authenticated Amazon Cognito identity.
- cognito_identity_pool_id – The Amazon Cognito identity pool that authorized the invocation.
- client_context – (mobile apps) Client context that's provided to Lambda by the client application.
- client.installation_id
- client.app_title
- client.app_version_name
- client.app_version_code
- client.app_package_name
- custom – A dict of custom values set by the mobile client application.
- env – A dict of environment information provided by the AWS SDK.

Create S3, EC2 and DynamoDB

Create Below Resources through Lambda

S3

- Create new Bucket
- List all the buckets
- Delete Bucket

EC2

- List all the buckets
- Create new Bucket
- Delete Bucket

DynamoDB

- Create Resources
- Add Items

Create S3, EC2 and DynamoDB – Introduction - 1

1. **Create the Lambda function in the console**

2. **Search for Boto3 Library Client or Resource method**

3. **Increase timeout limit**

4. **Create/Enhance the IAM Role**

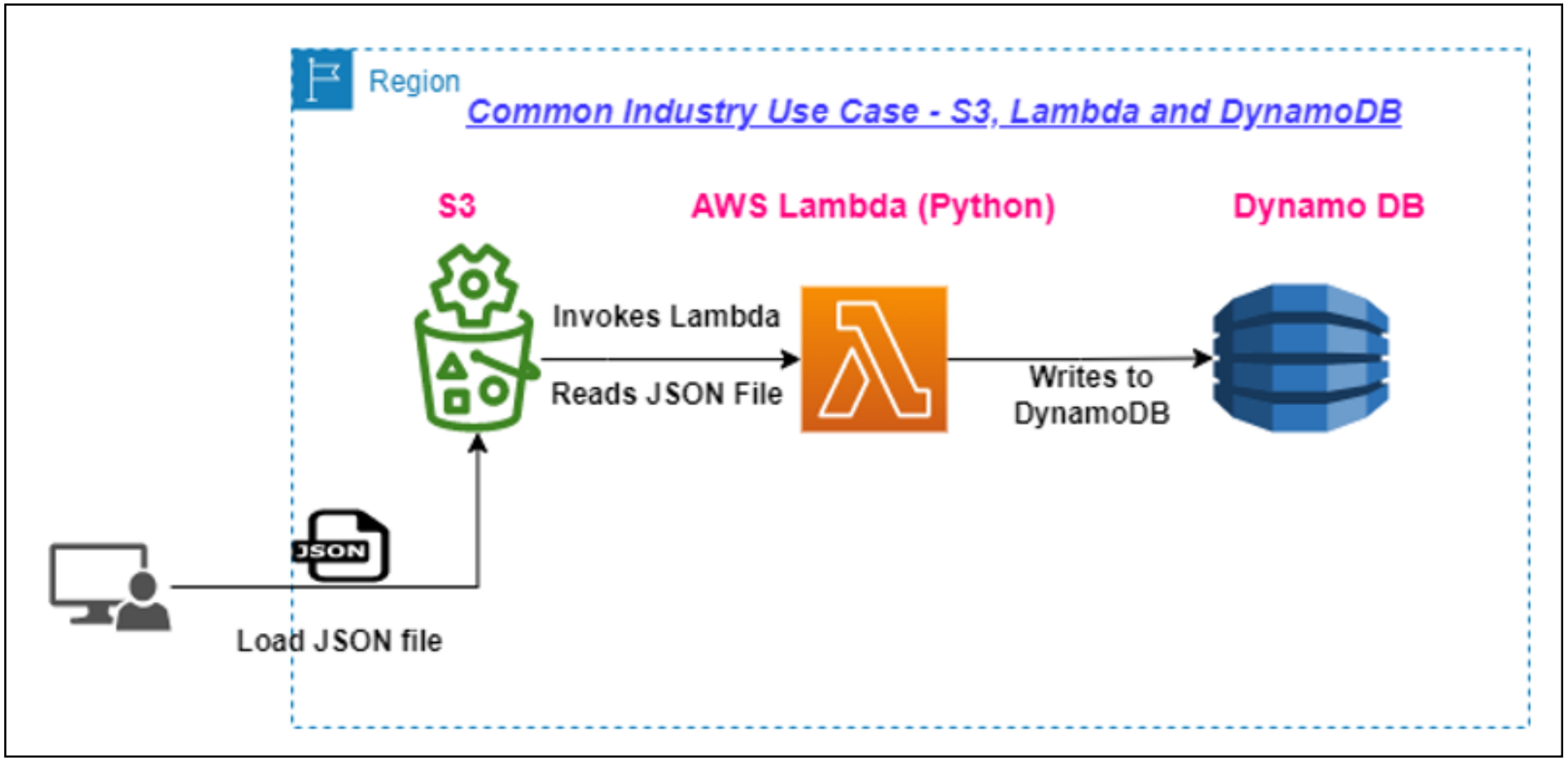
5. **Copy values from console for resource being created**

Section 5

Real World Serverless Use Case 1 - using S3, AWS Lambda and DynamoDB

Serverless Architecture – 1

AWS Lambda and Python – Target Architecture



S3 (Video 5)

- S3demo01022022abc

DynamoDB Table (Video 7)

- RetailSales02032022

Very Important :

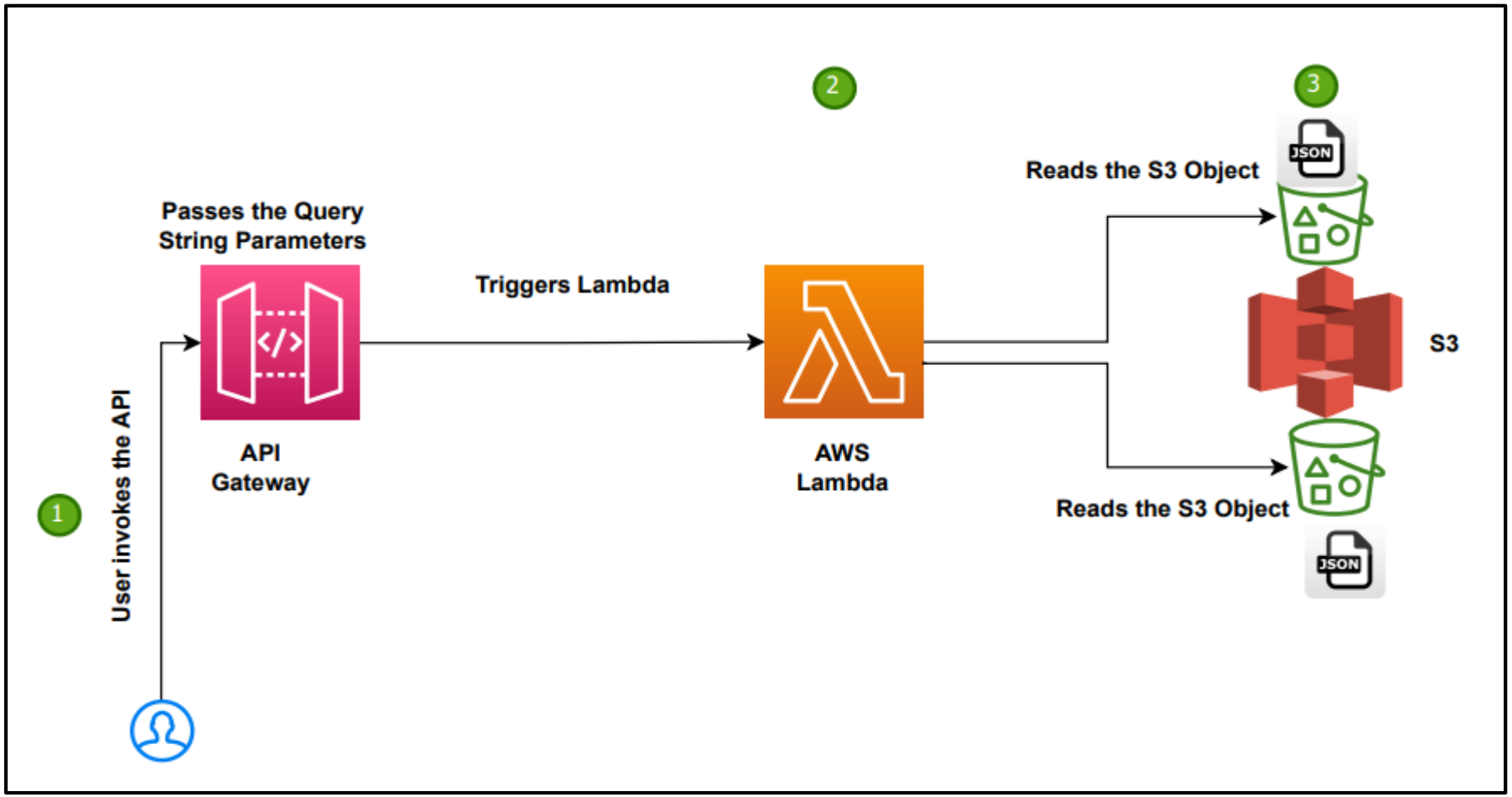
Please change the name of the S3 Bucket and DynamoDB Table with your own and substitute the names in the Lambda Python Code. The S3 bucket names need to be globally unique and same names that were highlighted in the video will not work.

Section 6

Real World Serverless Use Case 2 - using API Gateway, AWS Lambda and S3

Serverless Architecture – 2:

Target Architecture – API Gateway, Lambda and S3



Resources to be Created

- **S3 buckets – 2**
- **Lambda Function**
- **API Gateway**

Very Important :

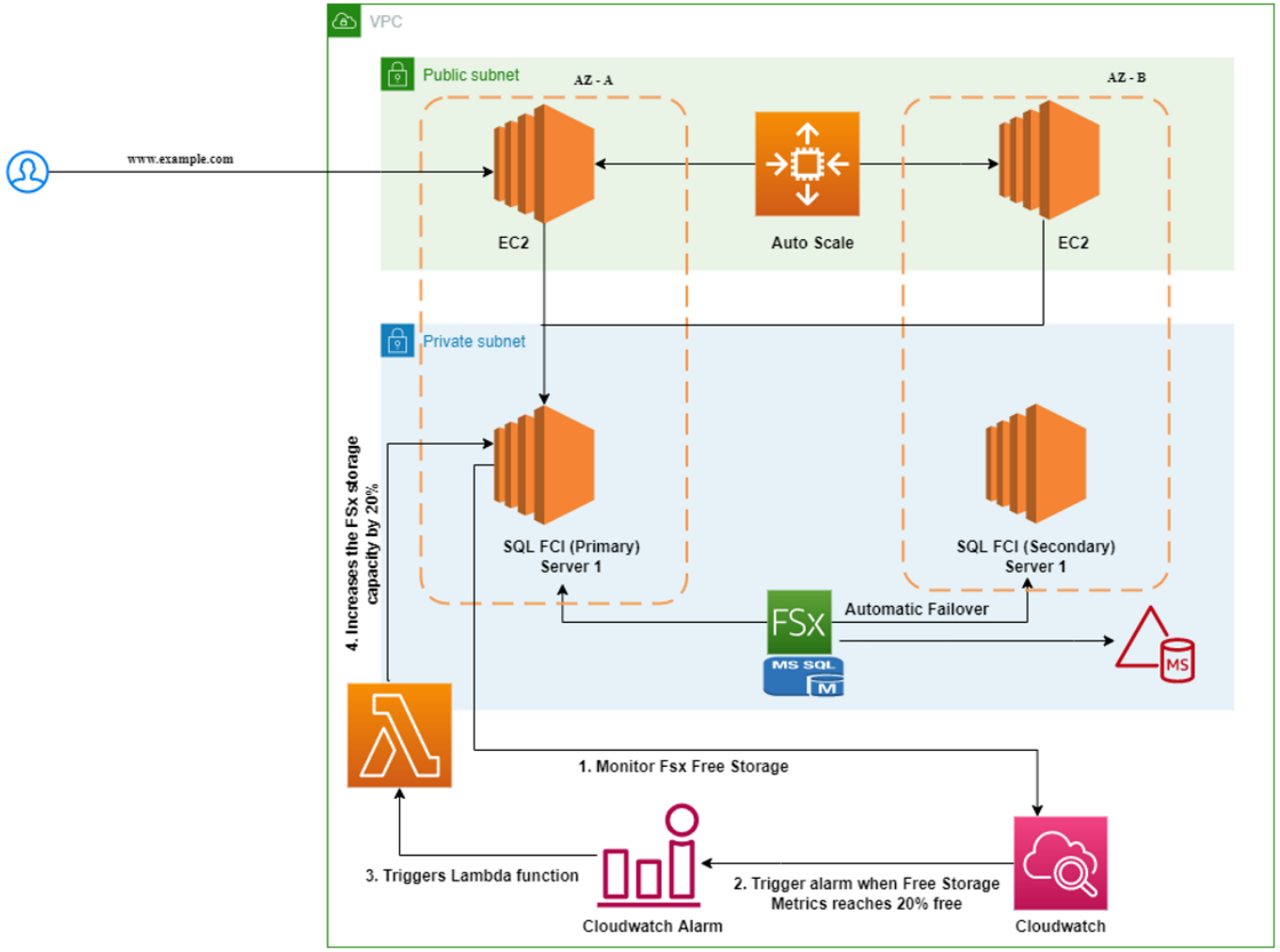
Please change the name of the S3 Bucket with your own and substitute the names in the Lambda Python Code.
The S3 bucket names need to be globally unique and same names that were highlighted in the video will not work.

Section 7

Real World Serverless Use Case 2 - using API Gateway, AWS Lambda and S3

AWS Lambda and Python – Beginner to Advanced

Monitor & increase free storage for SQL Server FCI Cluster using AWS Lambda- MS AD, EC2, FSx, CloudWatch and CloudWatch Alarm



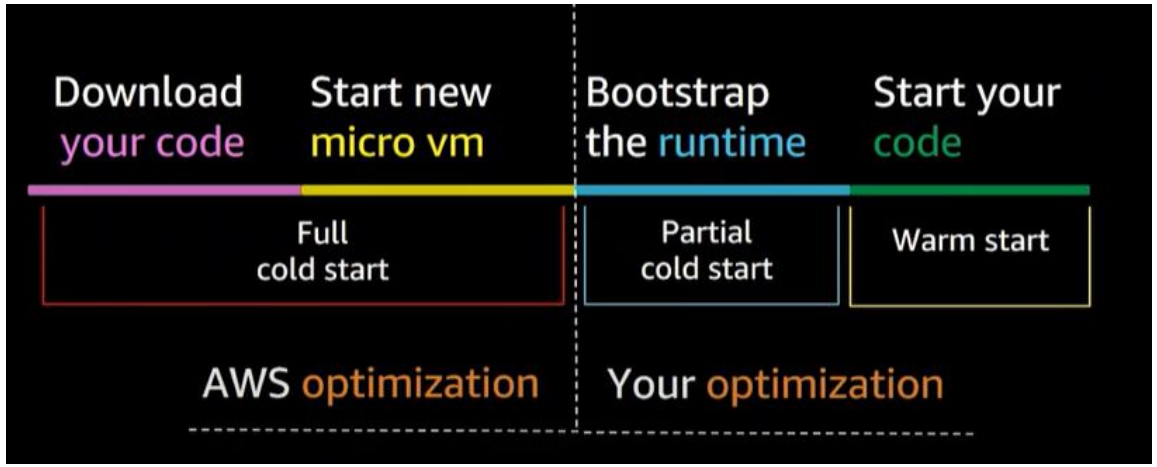
- *Microsoft AD*
- *EC2*
- *FSx*
- *CloudWatch Metrics*
- *CloudWatch Alarm*

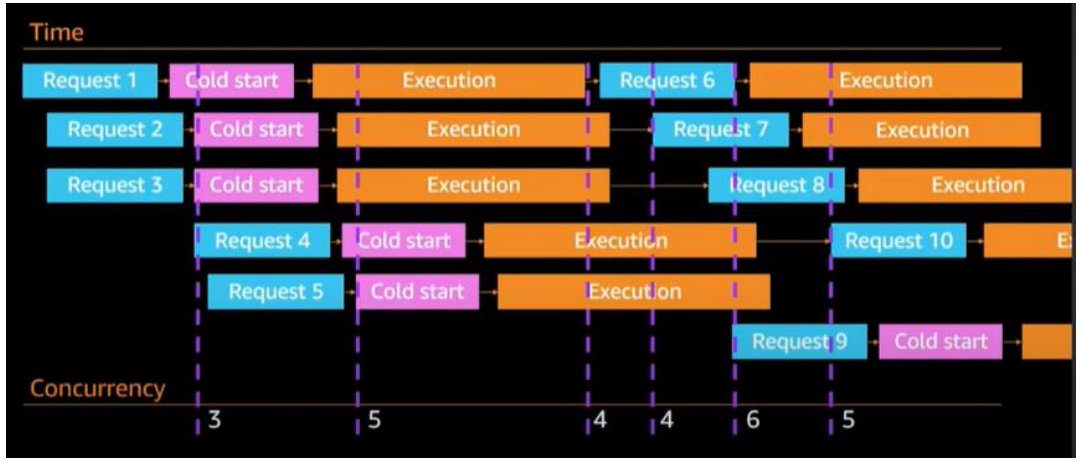
Section 8

AWS Lambda – Basic Concepts 2

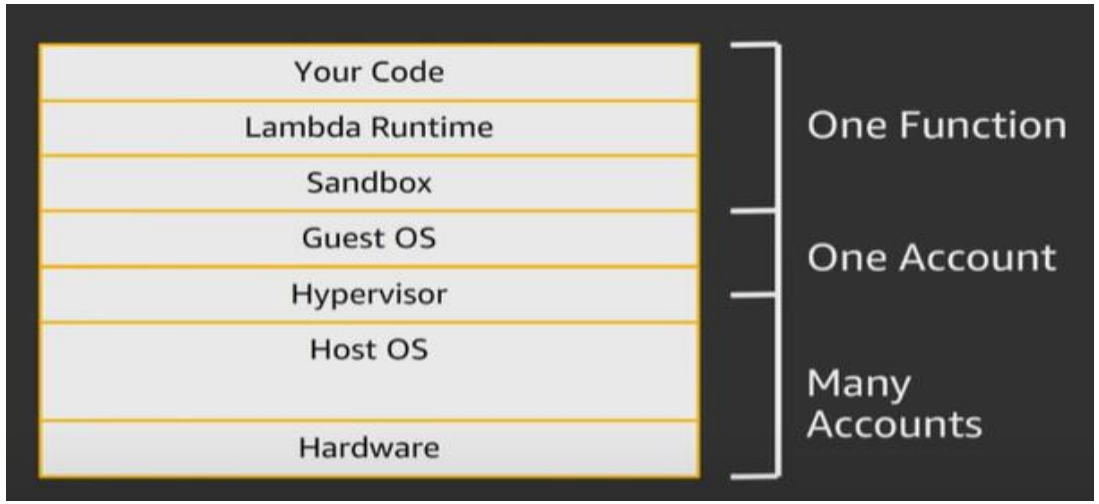
AWS Lambda and Python – Beginner to Advanced

5. AWS Lambda Concurrency - Cold Start Problem, Provisioned concurrency and Reserved concurrency





- **Provisioned Concurrency**
  - **Provisioning ramp up 500/min**
- **Reserved Concurrency**



Source : AWS

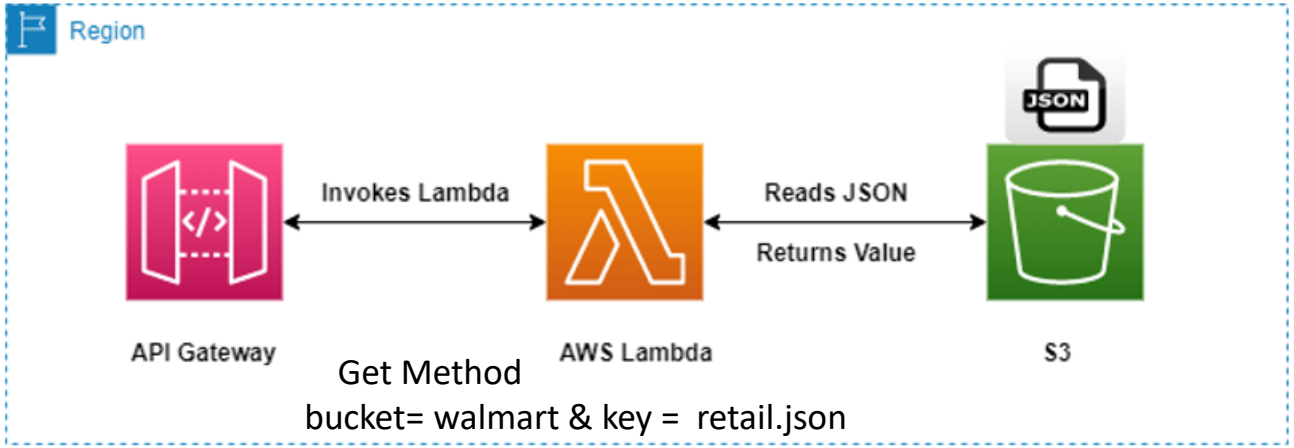
AWS Lambda and Python – Beginner to Advanced

6. Handler Function, Events and Context

The Lambda function handler is the method in your function code that processes events.

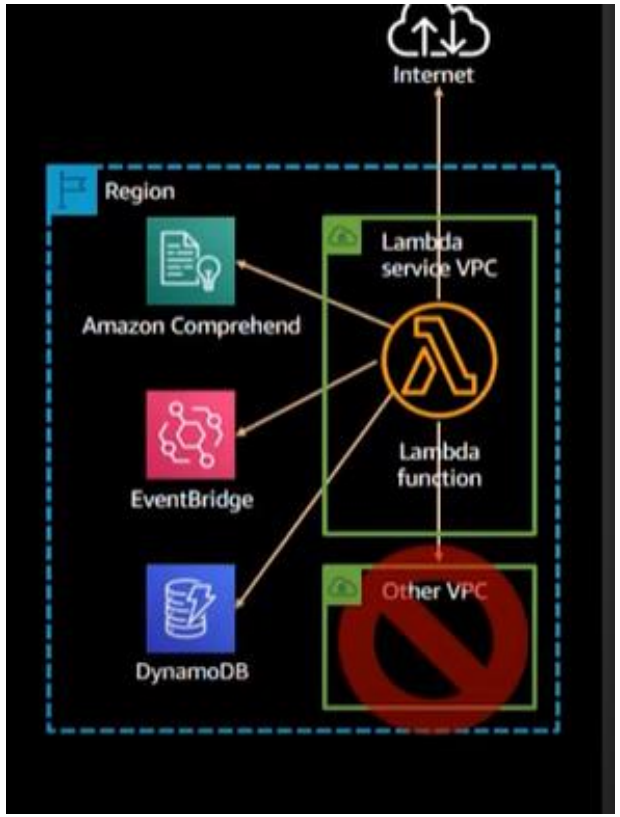
When your function is invoked, Lambda runs the handler method.

```
def handler_name(event, context):
    ...
    return some_value
```

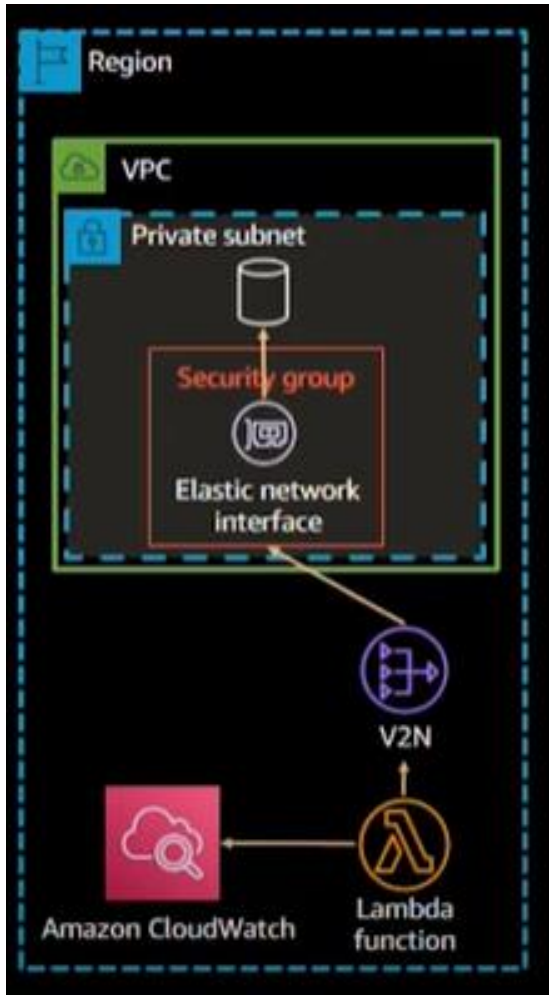


AWS Lambda and Python – Beginner to Advanced

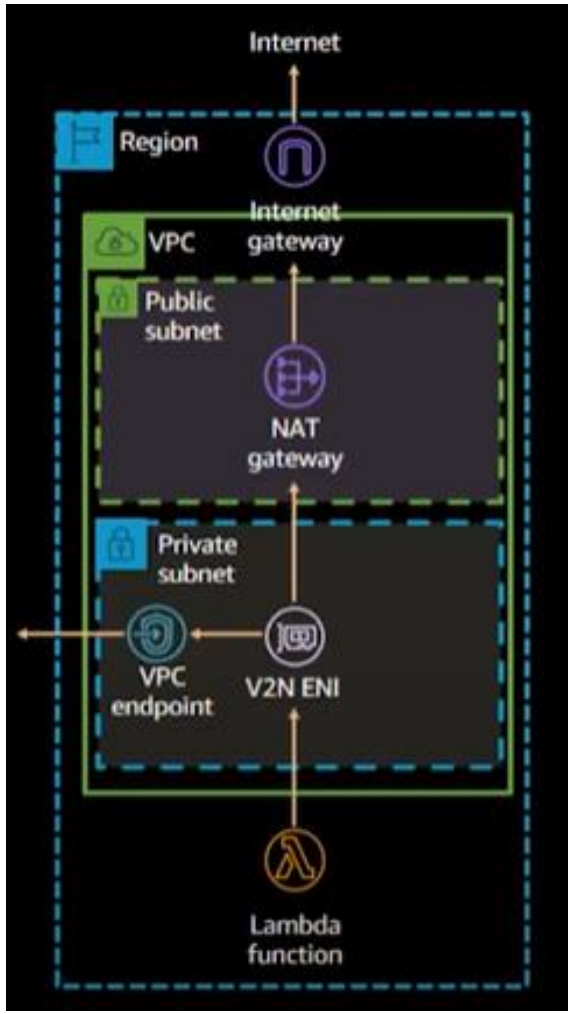
7. AWS Lambda VPC



Scenario 1



Scenario 2



Scenario 3

Source : AWS

AWS Lambda and Python – Beginner to Advanced

8. Lambda function – Use Cloud Watch Logs and CloudWatch Metrics

- CloudWatch Logs and Metrics for Lambda
- On processing an event, Lambda sends metrics about the invocation to CloudWatch.
- Graphs and dashboards can be built with these metrics on the CloudWatch console
- Set alarms to respond to changes in utilization, performance, or error rates.
- Lambda sends metric data to CloudWatch in 1-minute intervals.

9. Environment Variables

- Don't want to sensitive information in the code - password, access keys
- Store dynamic values – Prod, Dev, QA

10. Setting up DLQ and lambda Layers

- Dead letter queue (DLQ) is used to handle un processed messages for asynchronous invocations
- DLQ helps to prevent message loss in the case of failure during the lambda function invocations after 3 tries (configurable)

Thank You