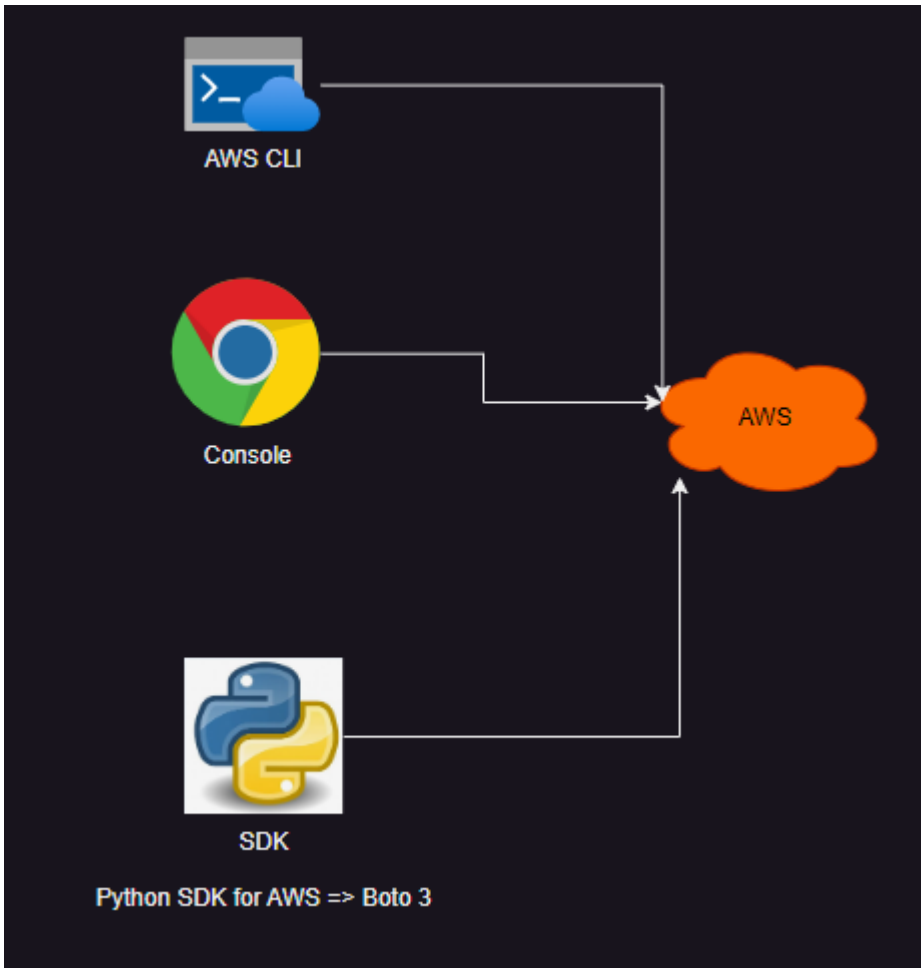
Using AWS Boto3 with python

- AWS Provides the python sdk called as boto3 which can be downloaded as a pip package



- Boto3 docs Refer Here
- Boto3 gives an API for every service, we have two major objects
    - client: low level apis
    - Resource: high level apis

Program 1: Create an s3 bucket using python

- Refer Here for the code changes

Program 2:

- Get all the buckets in your aws account (ensure atleast you have 3 buckets from three different regions)
- Get the buckets from mumbai region.
- Refer Here

Lets write python function to start the ec2 instance if the instance id is passed

- Create a free tier ec2 instance Refer Here

Lets write a python function which returns a list of ec2 instances by tag

- write a function

```
def ec2_instance_ids(tag_name, tag_value):
   pass

ec2_instance_ids('Name', 'sample')
# [ 'i-029ebe58050fd7ffd']
```

- Refer Here

Serverless Applications using python and AWS

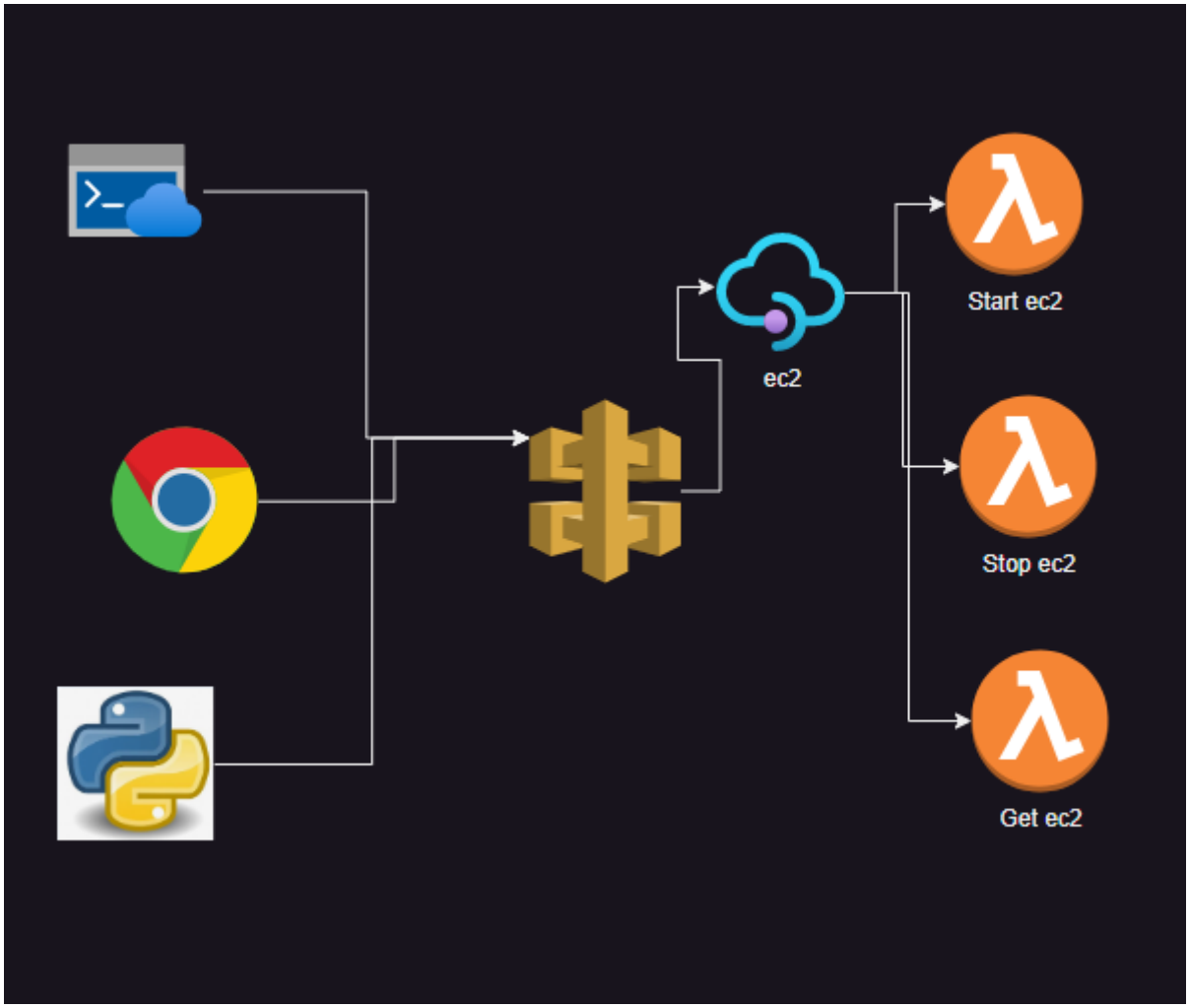
- Serverless model is defined where we break the functionality into smaller functions and you will be charged only for the duration when your function gets called.
- Serverless handles scaling
- Serverless is offered by various clouds
    - AWS: Lambda
    - Azure: Azure Functions
    - GCP: Google Functions
- Problem: Serverless is not standarized.

Lets create our first lambda function.

- For screenshots refer classroom video
- Refer Here for lambda function structure

Lambda with Api Gateway to control ec2 instances

- Overview



- The event structure

```
{
    "region": "ap-south-1",
    "tagName": "Name",
    "tagValue": "sample"
}
```

- We need a lamdba function to communicate with ec2, lets create an IAM Role for lambda to have permissions on ec2
- Lambda function to start ec2 instances

```
import json
import boto3

def start_ec2(instance_id, region = 'ap-south-1'):
    """Start ec2 instance
    """
    client = boto3.client('ec2',region_name=region)
    response = client.start_instances(InstanceIds=[instance_id])
    return response
```

```
def ec2_instance_ids(tag_name, tag_value, region = 'ap-south-1'):
    """This method will return a list of instance ids
    based on tag name and value

    Arguments:
      tag_name: str
      tag_value: str

    Returns:
      list
    """
    instance_ids = []
    client = boto3.client('ec2',region_name=region)
    response = client.describe_instances(
        Filters = [
            {
                'Name': f"tag:{tag_name}",
                'Values': [tag_value]
            }
        ]
    )
    for reservation in response['Reservations']:
        for instance in reservation['Instances']:
            instance_ids.append(instance['InstanceId'])
    return instance_ids

def start_ec2_instances_by_tag(tag_name, tag_value, region):
    """
    This method will start ec2 instances by tag value
    """
    resp_instance_ids = ec2_instance_ids(tag_name, tag_value, region)
    for resp_instance_id in resp_instance_ids:
        start_ec2(resp_instance_id, region)
    return resp_instance_ids


def lambda_handler(event, context):
    region = event['region']
    tag_name = event['tagName']
    tag_value = event['tagValue']
    instance_ids = start_ec2_instances_by_tag(tag_name, tag_value, region)


    return {
        'statusCode': 200,
        'instance_ids': ','.join(instance_ids)
    }
```

- We have created one more lambda function `stop_ec2`

```
import json
import boto3

def stop_ec2(instance_id, region = 'ap-south-1'):
    """stop ec2 instance
    """
    client = boto3.client('ec2',region_name=region)
    response = client.stop_instances(InstanceIds=[instance_id])
    return response

def ec2_instance_ids(tag_name, tag_value, region = 'ap-south-1'):
    """This method will return a list of instance ids
    based on tag name and value

    Arguments:
      tag_name: str
      tag_value: str

    Returns:
      list
    """
    instance_ids = []
    client = boto3.client('ec2',region_name=region)
    response = client.describe_instances(
        Filters = [
            {
                'Name': f"tag:{tag_name}",
                'Values': [tag_value]
            }
        ]
    )
    for reservation in response['Reservations']:
        for instance in reservation['Instances']:
            instance_ids.append(instance['InstanceId'])
    return instance_ids

def stop_ec2_instances_by_tag(tag_name, tag_value, region):
    """
    This method will stop ec2 instances by tag value
    """
    resp_instance_ids = ec2_instance_ids(tag_name, tag_value, region)
    for resp_instance_id in resp_instance_ids:
        stop_ec2(resp_instance_id, region)
    return resp_instance_ids


def lambda_handler(event, context):
    region = event['region']
    tag_name = event['tagName']
    tag_value = event['tagValue']
    instance_ids = stop_ec2_instances_by_tag(tag_name, tag_value, region)
```

```
    return {
        'statusCode': 200,
        'instance_ids': ','.join(instance_ids)
    }
```

- Now lets create API Gateway and integrate lambda functions with http methods in Resources

How to store data in dynamo db from lambda function python

- Refer Here