Taking Advantage of Serverless to Speed Up Python Deployment

Babajide Ogunjobi

jogunjobi@gmail.com @jide_o1

Agenda

- Serverless Concept
- Serverless Framework
- DEMO
  - Deploying SNS Module
  - Deploying a Flutterwave Payment system

About Me

CURRENT

Data Engineer Lead - Quartz Media

PREVIOUS

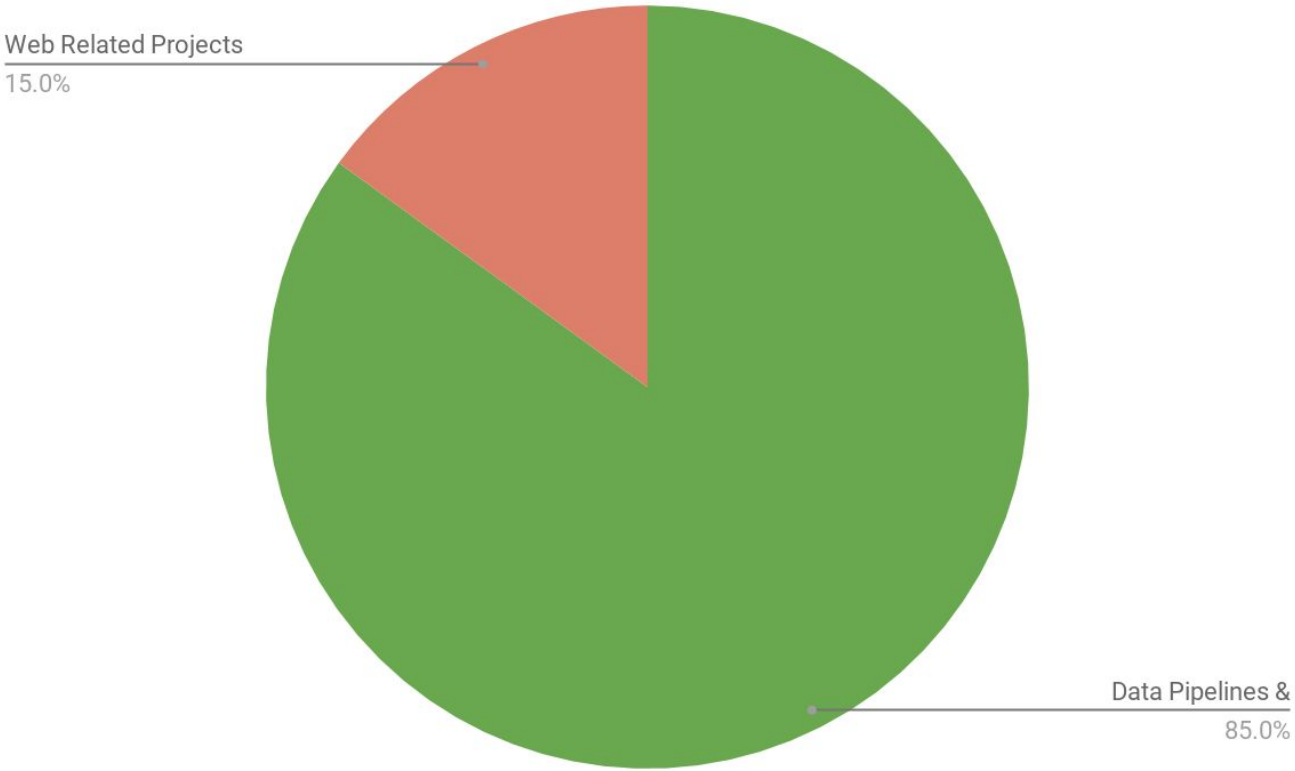
Lead Data Engineering Consultant - TGE Data

Senior Data Engineer - Grubhub

Data Solutions Architect - J.P. Morgan Chase

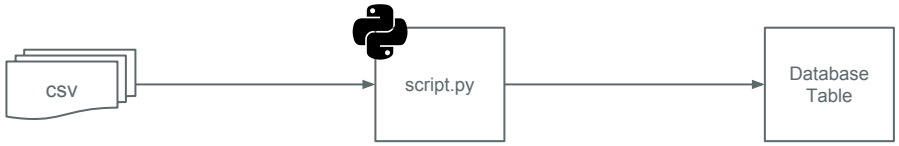
Solutions Architect Lead (ETL & Metadata) - Citigroup

My Projects

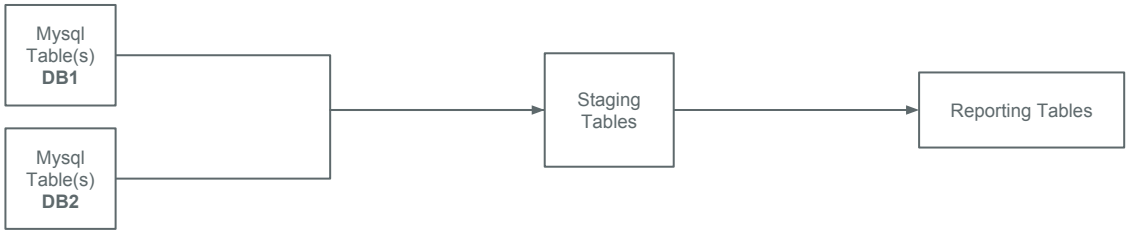


ETL & Data Pipelining

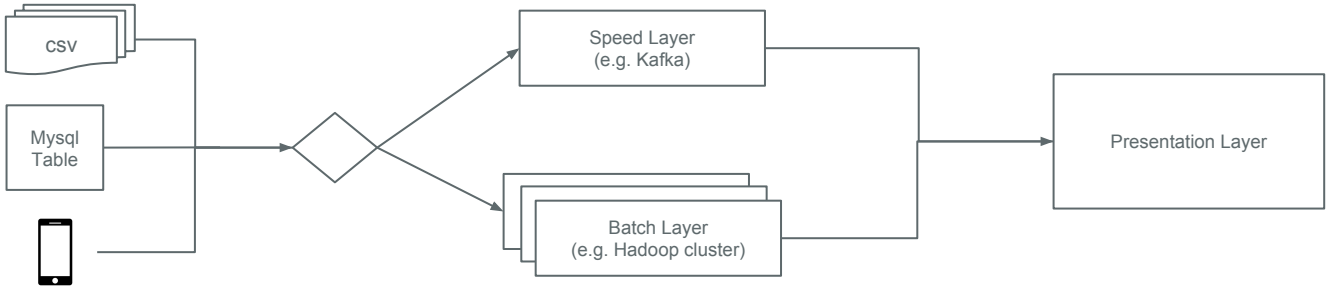
SIMPLE (within a single instance)



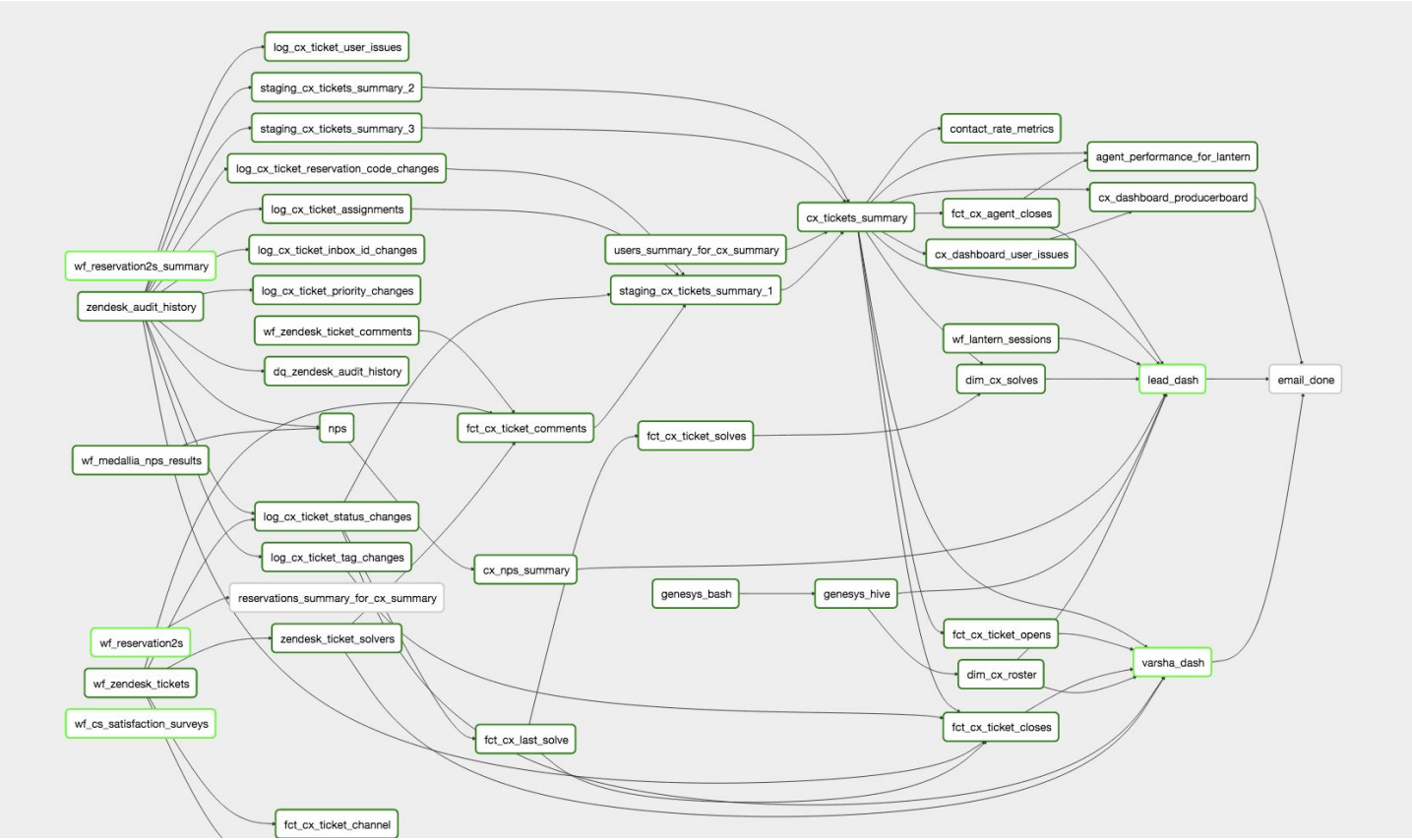
MEDIUM (across 1 or 2 instances)



COMPLICATED (across a cluster of instances)



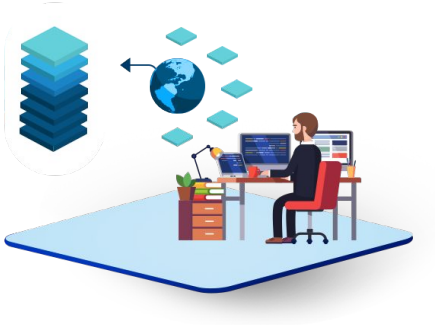
Scheduling & Orchestration



Tools

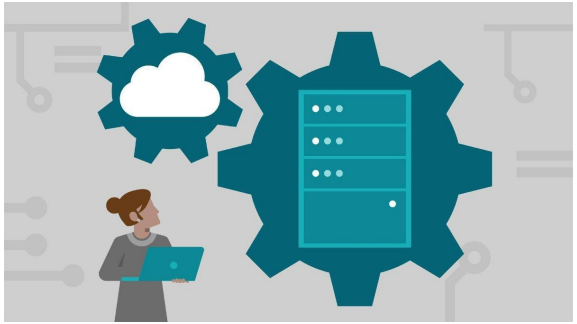
- Airflow
- Luigi
- Azkaban
- Cron

Web related projects



Development

- Writing code
- Testing using localhost & ngrok



Deployment

- Pushing code to remote server
- Production database connectivity
- Webserver configs (Nginx & Apache)

Maintenance

- Availability
- Server Costs
- Scaling
- Upgrades
- Security

TIGHTLY COUPLED - Any issues could cause total breakdown of application

The Day I discovered the serverless concept and learnt about AWS Lambda

About Serverless (the concept)

- There are still servers running (somewhere) but that’s not my business anymore
- Server scaling, availability and other management tasks have been outsourced to the cloud provider
- However, that means that you cannot access the servers and thereby have very little control over them (no SSH)

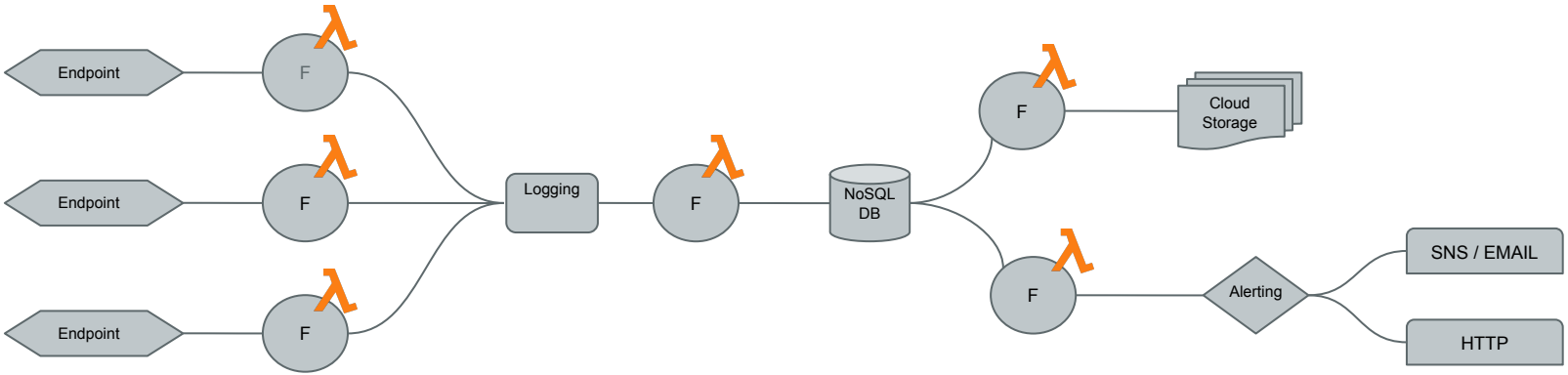
About Serverless (the concept)

Function as a Service (Faas)

This introduces the concepts of “breaking up” your application into multiple standalone modules or functions that only handle single actions or business logic where each function is executed individually.

- Loosely coupled components (connected via APIs)
- Largely event driven* architecture vs. Always up architecture
- The server management tasks are offloaded to a service provider
- Dynamic pricing model
- Developer no longer has to worry about scaling and allocation of resources
- Allows developers to shift focus from the server level to the task level
- Troubleshooting is localized to each function

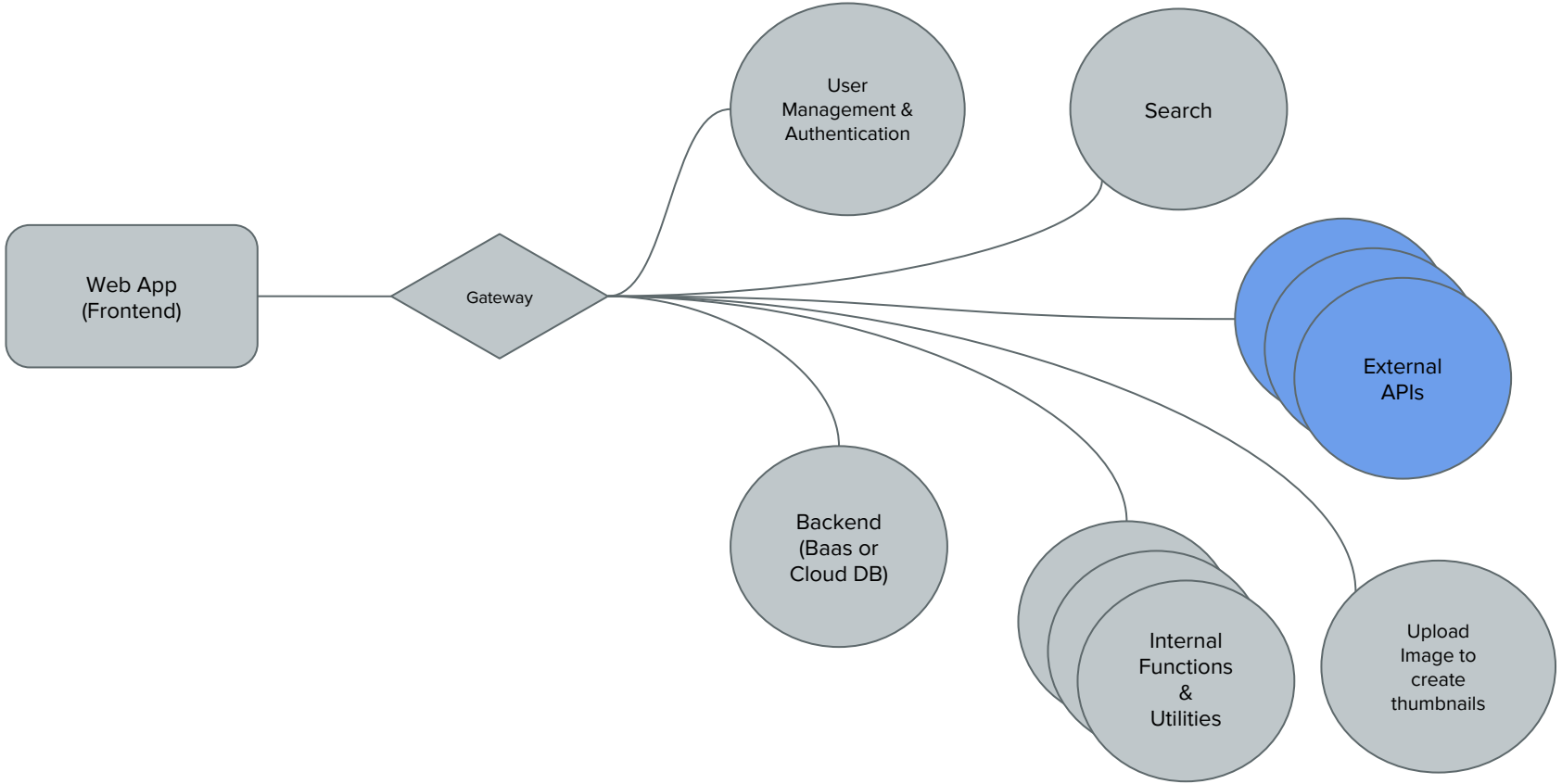
About Serverless (the concept)



Sample API Logging Framework

About Serverless (the concept)

Let’s revisit our web application



About Serverless (the concept)

While the concept of FaaS and decoupled functions is very attractive, they can quickly become very burdensome to maintain especially if there are multiple functions and associated APIs.

For example:
1). Creating the web app
2). Creating functions that need to be executed
3). Creating API endpoints which will trigger the functions
4). Creating API endpoints for internal & external services

The Serverless Framework simplifies the development, deployment and maintenance of multiple cloud functions and APIs within a few packages

The Day I discovered the Serverless Framework

About Serverless (the framework)

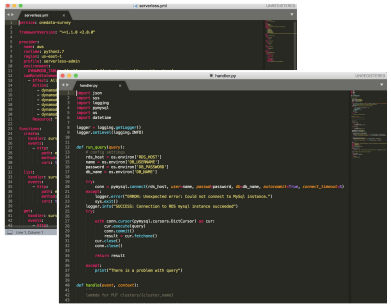
From the Serverless website (https://serverless.com/framework):

The Serverless Framework is an open-source CLI for building and deploying serverless applications.

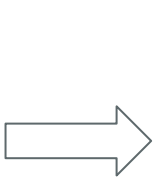
- Infrastructure As Code
- Simple Serverless Development
- Provider Agnostic

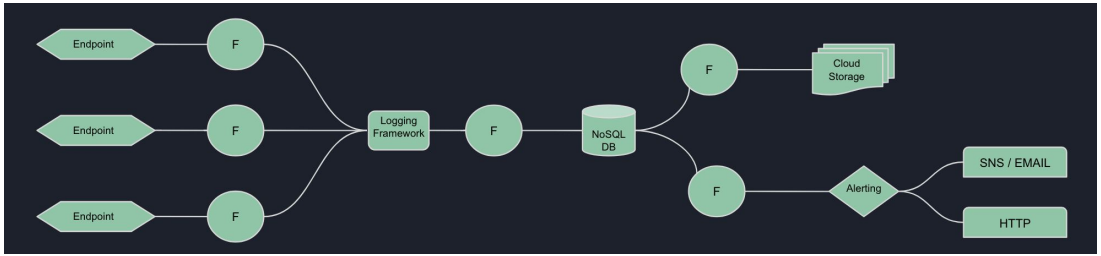
aws Azure Google Cloud Platform IBM Cloud kubernetes

handler.py

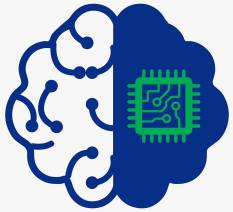


serverless.yml

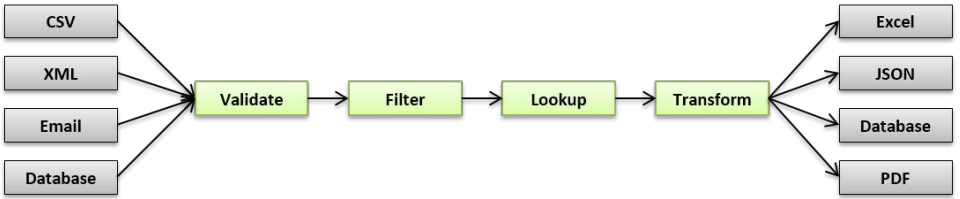




Serverless Uses



Data Science & AI



Data Pipelines

Web Apps

APIs

IOT

Installing Serverless

OSX

```
brew install node

npm install -g serverless
```

LINUX

```
sudo apt-get update

sudo apt-get install nodejs

sudo apt-get install npm

npm install -g serverless
```

WINDOWS

```
Download & install nodejs installer file from
https://nodejs.org/en/download/

npm install -g serverless
```

https://github.com/jogunjobi/pycon-nigeria

Digging into a Serverless project

handler.py

```python
import json
import logging
import os
import time
import uuid

import boto3
dynamodb = boto3.resource('dynamodb')


def create(event, context):
    data = json.loads(event['body'])
    if 'uuid' not in data:
        logging.error("Validation Failed")
        raise Exception("Couldn't create the todo item.")
        return

    timestamp = int(time.time() * 1000)

    table = dynamodb.Table(os.environ['DYNAMODB_TABLE'])

    item = {
        'id': str(uuid.uuid1()),
        'uuid': data['uuid'],
        'party': data['What is your name?'],
        'to_win': data['What is your age?'],
        'picture': data['Upload your picture'],
        'createdAt': timestamp,
        'updatedAt': timestamp,
    }

    # write the todo to the database
    table.put_item(Item=item)

    # create a response
    response = {
        "statusCode": 200,
        "body": json.dumps(item)
    }

    return response

```

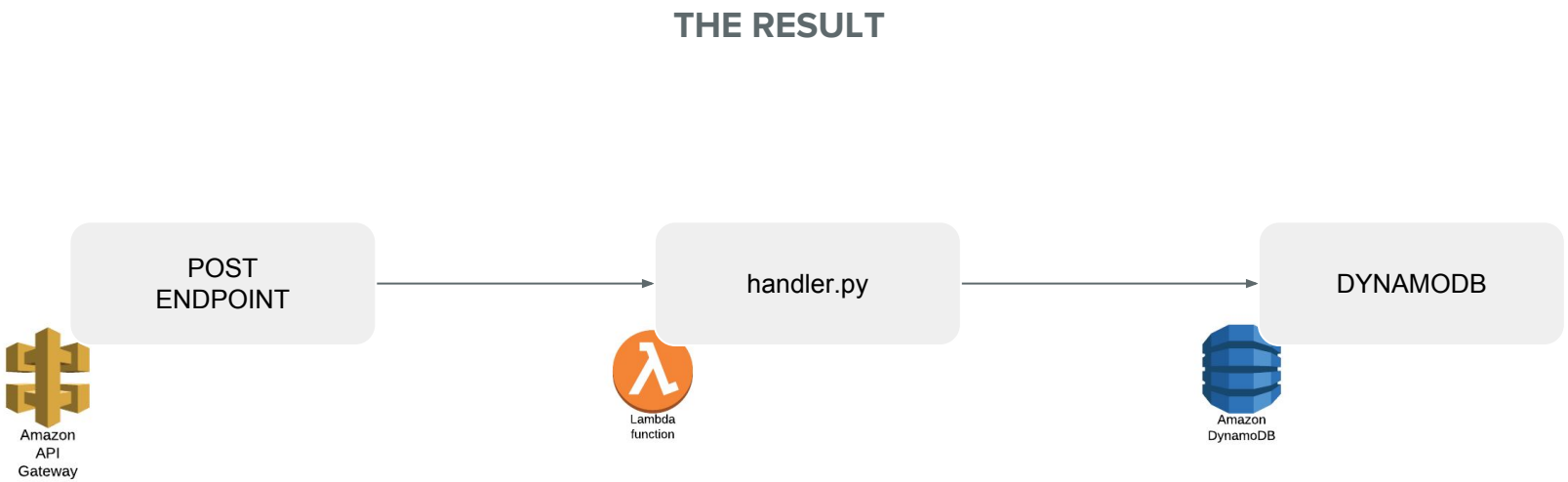
serverless.yml

```yaml
service: test

frameworkVersion: ">=1.1.0 <2.0.0"

provider:
  name: aws
  runtime: python2.7
  region: us-east-1
  profile: serverless-admin
  environment:
    DYNAMODB_TABLE: ${self:service}-${opt:stage, self:provider.stage}
  iamRoleStatements:
    - Effect: Allow
      Action:
        - dynamodb:Query
        - dynamodb:Scan
        - dynamodb:GetItem
        - dynamodb:PutItem
        - dynamodb:UpdateItem
        - dynamodb:DeleteItem
      Resource: "arn:aws:dynamodb:${opt:region, self:provider.region}:*:table/${self:provider.environment.DYNAMODB_TABLE}"

functions:
  create:
    handler: handler.create
    events:
      - http:
          path: mycreate
          method: post
          cors: true
resources:
  Resources:
    TodosDynamoDbTable:
      Type: 'AWS::DynamoDB::Table'
      #DeletionPolicy: Retain
      Properties:
        AttributeDefinitions:
          -
            AttributeName: id
            AttributeType: S
        KeySchema:
          -
            AttributeName: id
            KeyType: HASH
        ProvisionedThroughput:
          ReadCapacityUnits: 1
          WriteCapacityUnits: 1
        TableName: ${self:provider.environment.DYNAMODB_TABLE}

```

handler.py

serverless.yml

Digging into a Serverless project



DEMO

Takeaways

- The Serverless framework encapsulates the complexity of deploying and managing multiple functions, endpoints and applications.

- Serverless (concept & framework) is not a magic bullet for all deployments. Explore if it will fit into your stack before making a change.

- You can also explore replacing just a piece of your stack instead of a full replace

QUESTIONS?