

Going to Serverless the easy way



Serverless architecture

FaaS: Function as a Service

- **Full managed** computer,
 - Provisioning, patching
 - Scalability
 - Monitoring
 - Logging
 - No ops
- **Just deploy your code**
- **Pay only for you actual usage == Full utilization pay !**



Amazon API gateway

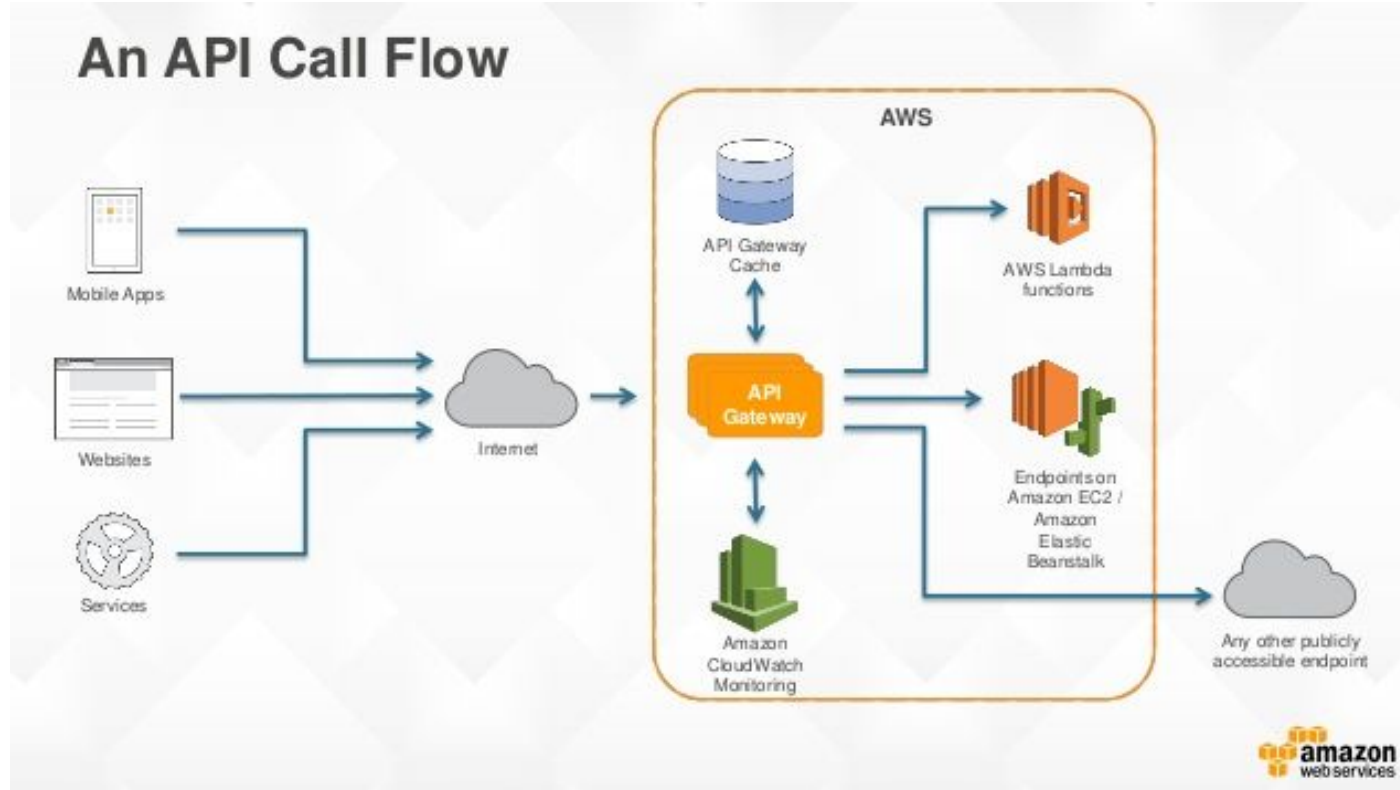


- is a fully managed service
- acts as a “front door” for applications to another
- it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale.
- including traffic management, authorization and access control, monitoring, and API version management.
- Usage plans
- You pay only for the API calls you receive and the amount of data transferred out.

Source :

<https://aws.amazon.com/api-gateway/>

Amazon API gateway



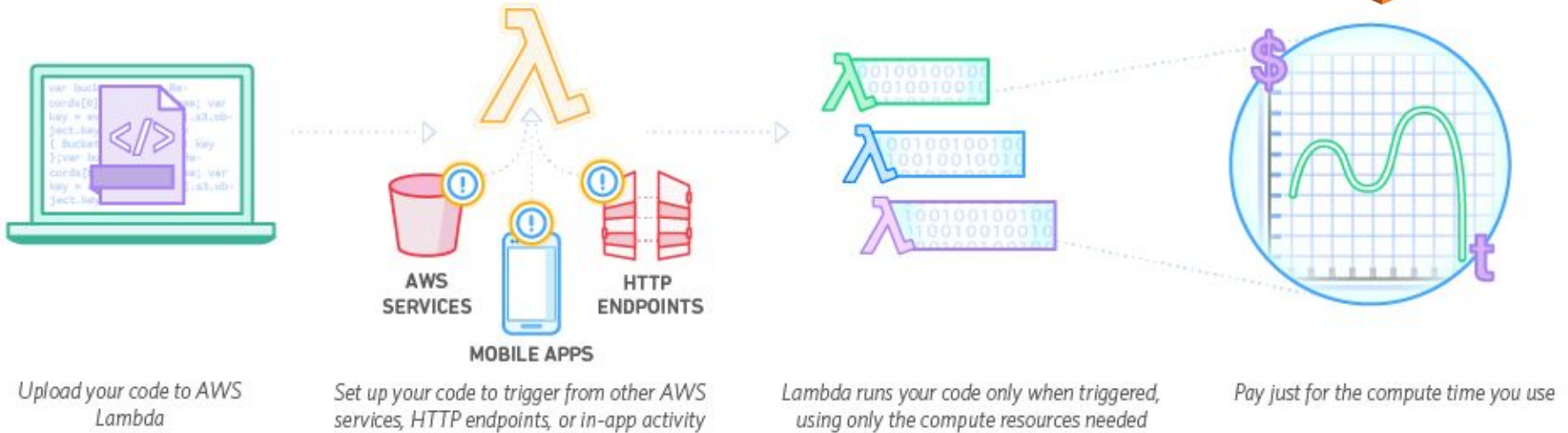
AWS Lambda



- All with **zero administration**
- you can **run code for virtually any type of application** or backend service
- lets you run code **without provisioning** or managing servers.
- **Architecture: Stateless**
 - Auto Scaled according to demand (events or request)
 - You pay only for the compute time you consume



6 | AWS Lambda: how it works



Multiple ways to put Lambda to work



7 | AWS Lambda : limitations



- Need to keep it warm
- Convenience can lead to vendor lock-in
- Languages: ... and python 2.7 and 3.6, NodeJs, Go.
- Execution time is limited to 5 min max
- Concurrent execution is limited to 1000
- Various payload and disk size limits
- Not in all AWS region
- No SSH
- How debug it ?
 - 17/Ago/2017 New – AWS SAM Local (Beta) – Build and Test Serverless Applications Locally

AWS Lambda & API Gateway are very nice, but...

- **Configuration** headache
 - IAM Roles
 - API Gateway
- **Deployment** headache
 - Packaging
 - Uploading
 - Rollback



Why use serverless frameworks

- **Simplify**

- IAM Roles
- API Gateway



AWS Chalice

- **Provisioning / Deployment**

- Continuous Delivery
- Simplify Devops
- Faster Deployments

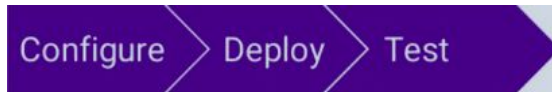


Zappa

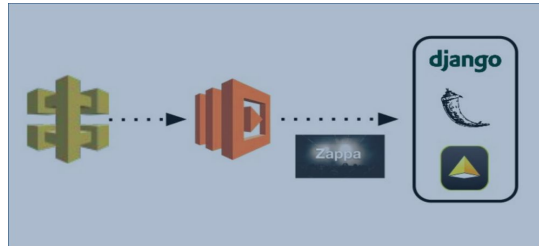


- **Description:**

- Python WSGI applications deployment on AWS (Lambda + API Gateway)
- With Zappa, each request is given its own virtual HTTP "server" by Amazon API Gateway. AWS handles the horizontal scaling automatically, so no requests ever time out. After your app returns, the "server" dies.
- **Manages application lifecycle (Creation, Deploy, Update, Undeploy)**




- **Architecture**



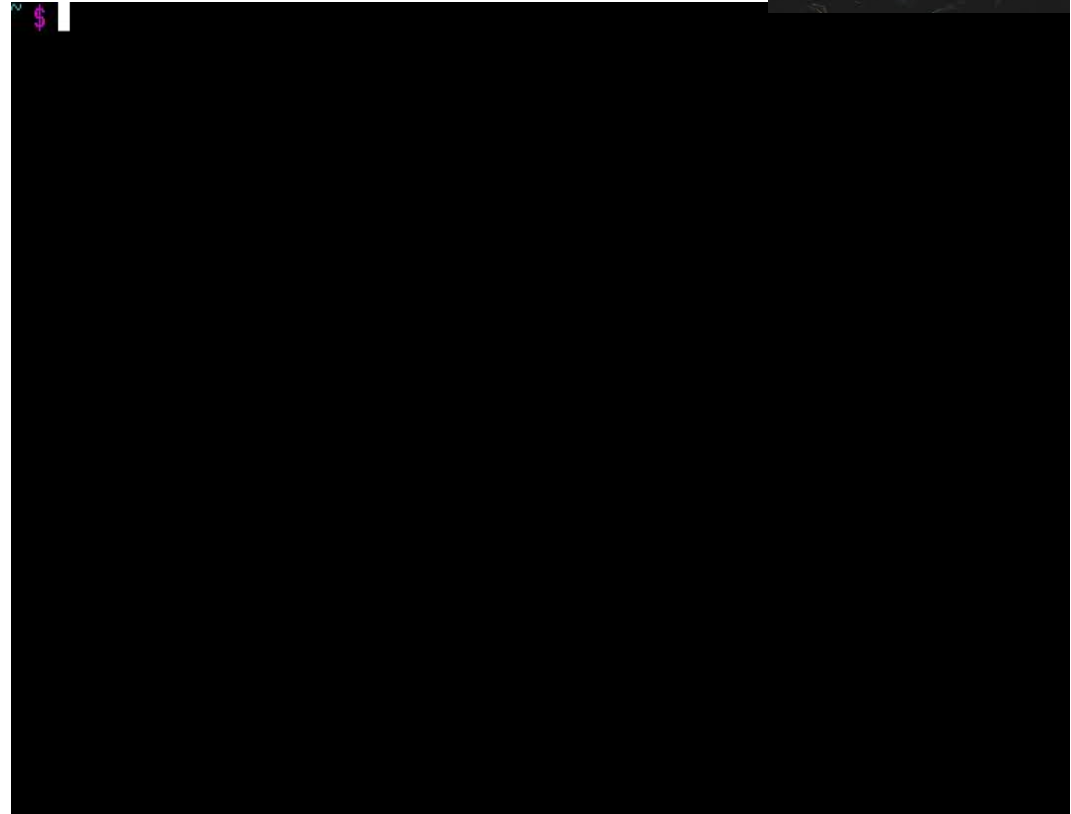
Source: <https://www.zappa.io/>

Zappa : example



```
1  from flask import Flask
2  
3  app = Flask(__name__)
4
5
6  @app.route('/')
7  def hello_world():
8      return 'Hello World!'
9
10
11  if __name__ == '__main__':
12      app.run()
13
```

```
1  {
2      "dev": {
3          "app_function": "zappa2.app",
4          "aws_region": "eu-west-1",
5          "profile_name": "staging",
6          "runtime": "python3.6",
7          "s3_bucket": "zappa-kjv8lj8oz"
8      }
9  }
```



Zappa : commands



- Install latest zappa (*optional create a new virtualenv*)
 - \$ pip install zappa
 - Use pip < 10 (9.0.3 is fine)
 - The virtual environment name should not be the same as the Zappa project name, as this may cause errors.
- Initialize project (*optional*)
 - \$ zappa init
 - This create a zappa_settings.json file
 - YAML format also available
- Deploy
 - \$ zappa deploy dev
 - #Edit code
 - \$ zappa update dev
- Test
 - \$ curl <https://x0doi1ioid.execute-api.eu-west-1.amazonaws.com/dev>

```
Uploading servless-dev-template-1506419092.json (1.6KiB)..  
100%|██████████| 1.61K/1.61K [00:00<00:00, 3.71KB/s]  
Waiting for stack servless-dev to create (this can take a bit)..  
75%|██████████| 3/4 [00:09<00:04, 4.94s/res]  
Deploying API Gateway..  
Deployment complete!  
https://x0doi1ioid.execute-api.eu-west-1.amazonaws.com/dev
```



Zappa : goodies



- Create deployment package (modules form current virtualenv)
- Performs deployments (no manual *.zip uploads)
- Support for stages (dev, stage, prod)
- Manages settings:
 - Environment variables
 - VPC
 - Logging (AWS CloudWatch)
 - Domain names
 - Security Roles
 - Certificates and Keys
- API creation

Is Serverless the silver bullet? mmmm, No.

As any technology is suitable for certain problems and not for others.

Let's review some of its limitations in general :

- **Vendor lock-in:** it is difficult to migrate from one provider to another.
- **Limitation of the maximum execution time** of a function.
- **Limitation of the maximum size of the function Initial latency:**
Enclosed environment and managed by others
- **Hard of debug**
- **No concurrency**

And that's all

Questions?



Thanks!

sergio.robles@ebury.com

 @_sroblesv_

 <https://www.linkedin.com/in/sergio-robles-708a2824>