Porting your Python web app to serverless in 30 minutes

Alex Casalboni
Technical Evangelist, AWS

@alex_casalboni

About me

- Software Engineer & Web Developer
- Data Science background
- Worked in a startup for 4.5 years
- ServerlessDays Organizer
- AWS Customer since 2013

Agenda

1. Why serverless computing
2. AWS Lambda & Python
3. Live coding

Why serverless computing

No server or container management

Flexible scaling

High availability

No idle capacity
Event-driven architectures

Event source → Serverless Function → Services (anything)
Common Use Cases

**Web apps**
- Static websites
- Complex web apps
- Packages for Flask and Express

**Backends**
- Apps and services
- Mobile
- IoT

**Data processing**
- Real time
- MapReduce
- Batch
- Machine learning inference

**Chatbots**
- Powering chatbot logic

**Amazon Alexa**
- Powering voice-enabled apps
- Alexa Skills Kit

**IT automation**
- Policy engines
- Extending AWS services
- Infrastructure management
"Alexa, buongiorno."
Ciao Milano! – An AWS Region is coming to Italy!

In early 2020 we plan to launch an AWS Region in Italy

allthingsdistributed.com

9:28 AM - 13 Nov 2018

243 Retweets  540 Likes
AWS Serverless Application Model (SAM)

- Simplified template-driven deployment model for serverless applications
- New serverless resource types
- An extension (Macro) of AWS CloudFormation
- Open specification (Apache 2.0)

github.com/awslabs/serverless-application-model
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  GetHtmlFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://sam-demo-bucket/todo_list.zip
      Handler: index.gethtml
      Runtime: nodejs4.3
      Policies: AmazonDynamoDBReadOnlyAccess
    Events:
      GetHtml:
        Type: Api
        Properties:
          Path: /{proxy+}
          Method: ANY

Tells AWS CloudFormation this is a SAM template it needs to transform.

Creates a Lambda function, API Gateway resources and all mapping and permissions necessary.
AWS SAM CLI

CLI tool for local testing of serverless apps

Works with Lambda functions and “proxy-style” APIs

Response object and function logs available on your local machine

Uses open-source Docker-Lambda images to mimic Lambda’s execution environment (timeout, memory limits, runtimes)

github.com/awslabs/aws-sam-cli
AWS Lambda & Python

Python support for AWS Lambda

- **Nov 2014**: Lambda Preview
- **Oct 2015**: Python 2.7
- **Apr 2017**: Python 3.6
- **Nov 2018**: Python 3.7
- **AWS re:Invent 2018**: Custom Runtimes!
How do we port a Python web app to AWS Lambda?

The easy way, cheating a bit!

$ pip install zappa
$ zappa init
$ zappa deploy production

https://www.zappa.io
The easy way, cheating a bit! (2)

```
$ pip install chalice
# code refactor
$ chalice deploy
```

```
from chalice import Chalice
app = Chalice(__name__)
@app.route('/

def hello_world():
    return 'Hello, World!'
```

https://chalice.readthedocs.io
NOT GOOD ENOUGH
NOT GOOD ENOUGH AT ALL
Coding time!
github.com/alexcasalboni/flask-app-serverless-porting
Thank you!

Alex Casalboni
Technical Evangelist, AWS

@alex_casalboni