Serverless frameworks on AWS

Julien Simon, Principal Technical Evangelist, AWS
julsimon@amazon.fr
@julsimon

AWS Lambda

- Announced at re:Invent 2014
- Deploy functions in Java, Python, Node.js and C#
- Just code, without the infrastructure drama
- Built-in scalability and high availability
- Well integrated with other AWS services
- Pay as you go
  - Combination of execution time (100ms slots) & memory used
  - Free tier available

What can you do with AWS Lambda?

- Grow '**connective tissue**' in your AWS infrastructure
  - Example: http://www.slideshare.net/JulienSIMON5/building-a-serverless-pipeline
- Build event-driven applications
- Build APIs together with Amazon API Gateway
  - RESTful APIs
  - Resources, methods
  - Stages

http://aws.amazon.com/lambda
http://aws.amazon.com/apigateway

AWS Lambda

+

Managed services

=

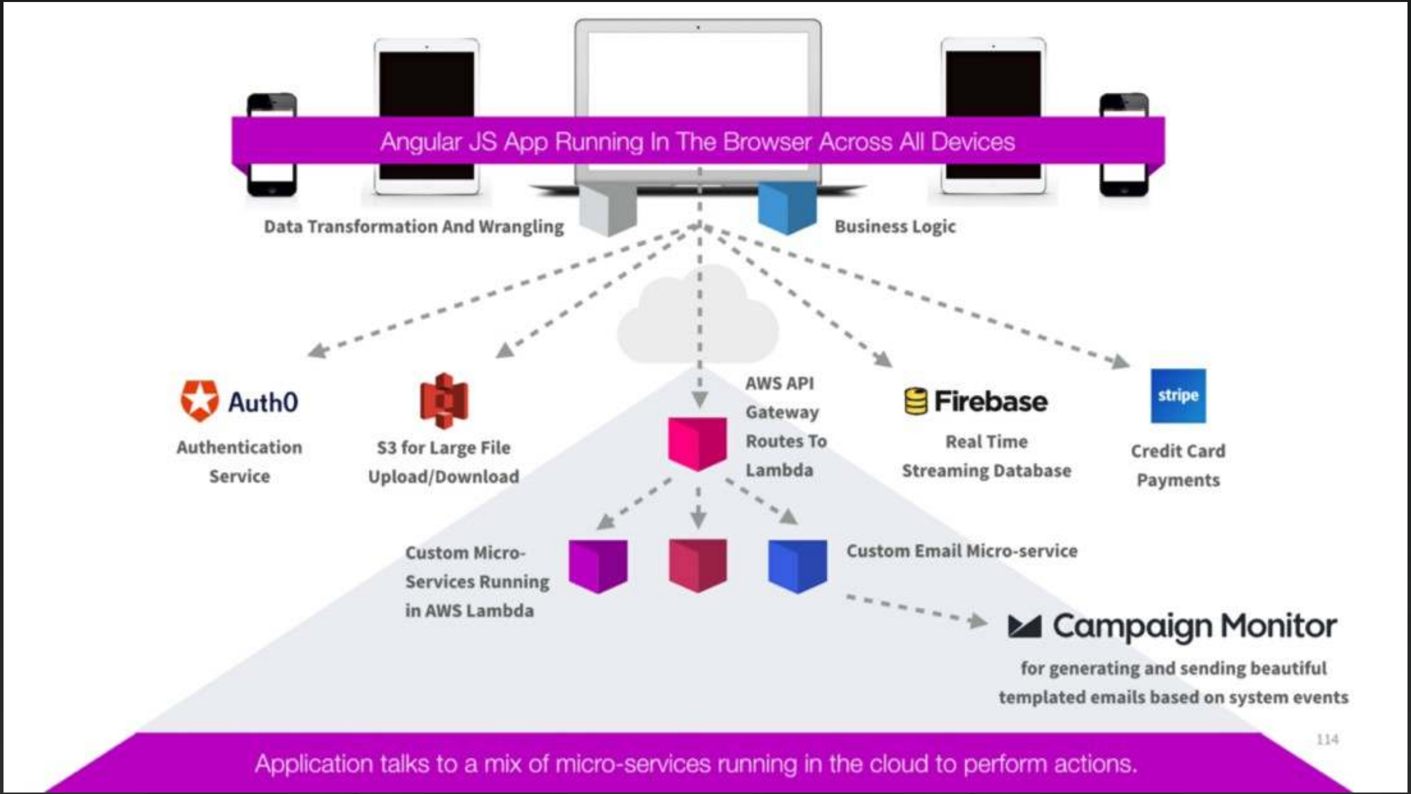
Serverless architecture

Reference architectures: http://www.allthingsdistributed.com/2016/06/aws-lambda-serverless-reference-architectures.html

Werner Vogels, CTO, Amazon.com
AWS re:Invent 2015

A Cloud Guru: 100% Serverless

A CLOUD GURU


Angular JS App Running In The Browser Across All Devices
Data Transformation And Wrangling
Business Logic
Auth0
Authentication Service
S3 for Large File Upload/Download
AWS API Gateway Routes To Lambda
Firebase
Real Time Streaming Database
stripe
Credit Card Payments
Custom Micro-Services Running in AWS Lambda
Custom Email Micro-service
Campaign Monitor
for generating and sending beautiful templated emails based on system events
114
Application talks to a mix of micro-services running in the cloud to perform actions.

https://read.acloud.guru/serverless-the-future-of-software-architecture-d4473ffed864

amazon web services

Typical development workflow with AWS Lambda

1. Write and deploy a Lambda function
2. Create and deploy a REST API with API Gateway
3. Connect the API to the Lambda function
4. Invoke the API
5. Test, debug and repeat ;)

A simple Lambda function in Python

```
def lambda_handler(event,context):
   result = event['value1'] + event['value2']
   return result
```

```
aws lambda create-function --function-name add \
--handler myFunc.lambda_handler --runtime python2.7 \
--zip-file fileb://myFunc.zip --memory-size 128 \
--role arn:aws:iam::ACCOUNT_NUMBER:role/lambda_basic_execution

curl -H "Content-Type: application/json" \
     -X POST -d "{\"value1\":5, \"value2\":7}" \
     https://API_ENDPOINT
12
```

That’s great, but…

- No one wants to code in the AWS console (right?)
- Managing functions with the AWS CLI isn’t dev-friendly
- Managing APIs with the AWS CLI quite complex (low-level calls)
- CloudFormation doesn’t make it easy to deploy and manage serverless applications (custom resources)
- So what are the options?

Serverless tools

- Development
  - Serverless Framework
  - Gordon
  - AWS Chalice
  - More frameworks: Kappa, Apex, Zappa, Docker-lambda
  - AWS Lambda plugin for Eclipse
- Deployment
  - AWS Serverless Application Framework (SAM)

Development

Code samples available at https://github.com/juliensimon/aws/tree/master/lambda_frameworks

The Serverless framework

formerly known as JAWS: Just AWS Without Servers

- Announced at re:Invent 2015 by Austen Collins and Ryan Pendergast
- Supports Node.js, as well as Python and Java (with restrictions)
- Auto-deploys and runs Lambda functions, locally or remotely
- Auto-deploys your Lambda event sources: API Gateway, S3, DynamoDB, etc.
- Creates all required infrastructure with CloudFormation
- Simple configuration in YML

http://github.com/serverless/serverless
https://serverless.com
AWS re:Invent 2015 | (DVO209) https://www.youtube.com/watch?v=D_U6luQ6I90 & https://vimeo.com/141132756

Serverless: standalone function

```
$ serverless create --template aws-python
```

Edit handler.py, serverless.yml and event.json

```
$ serverless deploy
```

```
$ serverless invoke
    [ local ]                    Only supported for Node.js :-/
    --function function_name
    [ --path event.json ]
```

```
$ serverless logs --function function_name
```

Serverless: API + function

Update serverless.yml, add JSON processing in handler.py

```
$ serverless deploy --stage stage_name

$ serverless info

$ curl -H "Content-Type: application/json" \
     -X POST -d "{\"value1\":5, \"value2\":7}" \
     https://API_ENDPOINT
```

Gordon

- Released in Oct'15 by Jorge Batista
- Supports Python, Javascript, Golang, Java, Scala, Kotlin (including in the same project)
- Auto-deploys and runs Lambda functions, locally or remotely
- Auto-deploys your Lambda event sources: API Gateway, CloudWatch Events, DynamoDB Streams, Kinesis Streams, S3
- Creates all required infrastructure with CloudFormation
- Simple configuration in YML

https://github.com/jorgebastida/gordon
https://news.ycombinator.com/item?id=11821295

Gordon: “Hello World” API

```
$ gordon startproject helloworld

$ gordon startapp helloapp

Write hellofunc() function

$ gordon build

$ echo '{"name":"Julien"}' | gordon run helloapp.hellofunc

$ gordon apply --stage stage_name

$ http post $API_ENDPOINT name=Julien
```

AWS Chalice

Think of it as a serverless framework for Flask apps

- Released in Jul'16, still in beta
- Just add your Python code
  - Deploy with a single call and zero config
  - The API is created automatically, the IAM policy is auto-generated
- Run APIs locally on port 8000 (similar to Flask)
- Fast & lightweight framework
  - 100% *boto3* calls (AWS SDK for Python) → fast
  - No integration with CloudFormation → no creation of event sources

https://github.com/awslabs/chalice
https://aws.amazon.com/blogs/developer/preview-the-python-serverless-microframework-for-aws/

AWS Chalice: “Hello World” API

```
$ chalice new-project helloworld

Write your function in app.py

$ chalice local

$ chalice deploy

$ export API_ENDPOINT = `chalice url`

$ http $API_ENDPOINT

$ http put $API_ENDPOINT’hello/julien’

$ chalice logs [ --include-lambda-messages ]
```

AWS Chalice: PUT/GET in S3 bucket

```
$ chalice new-project s3test

Write your function in app.py

$ chalice local

$ http put http://localhost:8000/objects/doc.json value1=5 value2=8

$ http get http://localhost:8000/objects/doc.json

$ chalice deploy stage_name

$ export API_ENDPOINT=`chalice url`

$ http put $API_ENDPOINT/objects/doc.json value1=5 value2=8

$ http get $API_ENDPOINT/objects/doc.json
```

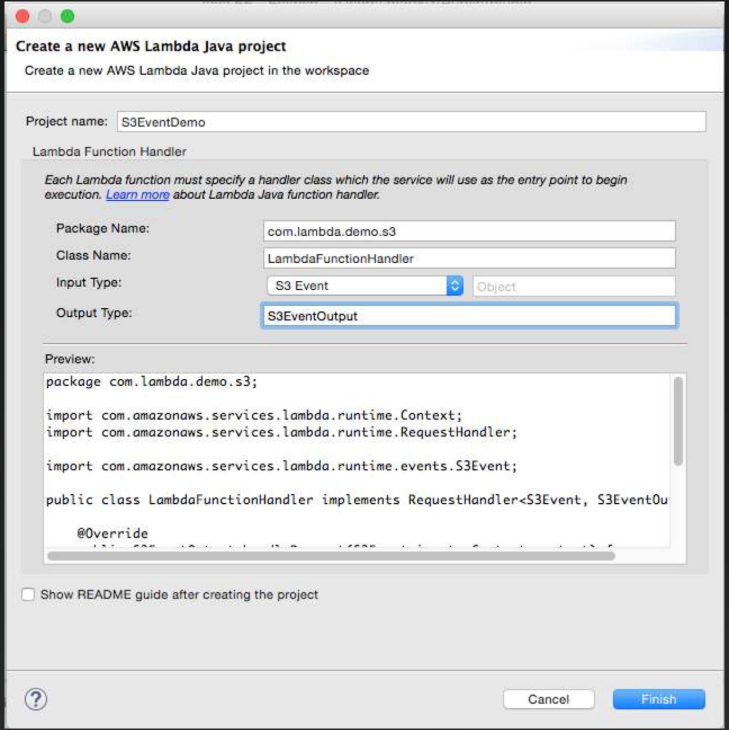
Summing things up

Serverless	Gordon	Chalice
The most popular serverless framework	Great challenger!	AWS project, in beta
Built with and for Node.js. Python and Java: YMMV	Node.js, Python, Java, Scala, Golang	Python only
Rich features, many event sources	Comparable to Serverless feature-wise	Does only one thing, but does it great
Not a web framework	Not a web framework	Dead simple, zero config
		Flask web framework

More Lambda frameworks

- Kappa https://github.com/garnaat/kappa
  - Released Dec'14 by Mitch Garnaat, author of boto and the AWS CLI (still maintained?)
  - Python only, multiple event sources
- Apex https://github.com/**apex**/**apex**
  - Released in Dec'15 by TJ Holowaychuk
  - Python, Javascript, Java, Golang
  - Terraform integration to manage infrastructure for event sources
- Zappa https://github.com/Miserlou/Zappa
  - Released in Feb'16 by Rich Jones
  - Python web applications on AWS Lambda + API Gateway
- Docker-lambda https://github.com/lambci/docker-lambda
  - Released in May'16 by Michael Hart
  - Run functions in Docker images that "replicate" the live Lambda environment

AWS Lambda plug-in for Eclipse



Create a new AWS Lambda Java project

Create a new AWS Lambda Java project in the workspace

Project name: S3EventDemo

Lambda Function Handler

Each Lambda function must specify a handler class which the service will use as the entry point to begin execution. Learn more about Lambda Java function handler.

Package Name:	com.lambda.demo.s3
Class Name:	LambdaFunctionHandler
Input Type:	S3 Event
Output Type:	S3EventOutput

Preview:

```
package com.lambda.demo.s3;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

import com.amazonaws.services.lambda.runtime.events.S3Event;

public class LambdaFunctionHandler implements RequestHandler<S3Event, S3EventOu

    @Override
```

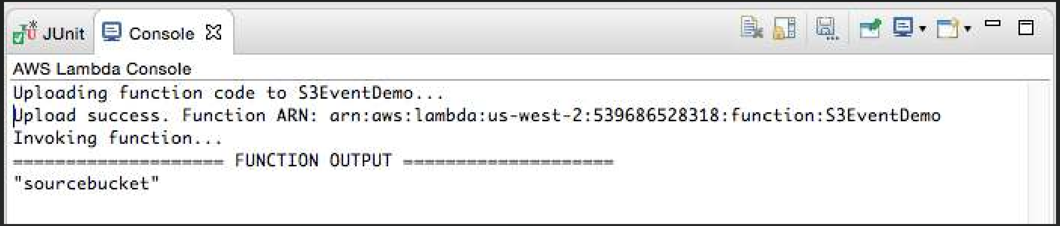
Show README guide after creating the project

Code, test and deploy Lambdas from Eclipse

Run your functions locally and remotely

Test with local events and Junit4

Deploy standalone functions, or with the AWS Serverless Application Model (Dec'16)



```
AWS Lambda Console
Uploading function code to S3EventDemo...
Upload success. Function ARN: arn:aws:lambda:us-west-2:539686528318:function:S3EventDemo
Invoking function...
==================== FUNCTION OUTPUT ====================
"sourcebucket"
```

https://java.awsblog.com/post/TxWZES6J1RSQ2Z/Testing-Lambda-functions-using-the-AWS-Toolkit-for-Eclipse

https://aws.amazon.com/blogs/developer/aws-toolkit-for-eclipse-serverless-application

Deployment

AWS Serverless Application Model (SAM)

formerly known as Project Flourish

- CloudFormation extension released in Nov'16 to bundle Lambda functions, APIs & events
- 3 new CloudFormation resource types
  - *AWS::Serverless::Function*
  - *AWS::Serverless::Api*
  - *AWS::Serverless::SimpleTable*
- 2 new CloudFormation commands
  - *'aws cloudformation package'*
  - *'aws cloudformation deploy'*
- Integration with CodeBuild and CodePipeline for CI/CD
- Expect SAM to be integrated in most / all frameworks

https://aws.amazon.com/fr/blogs/compute/introducing-simplified-serverless-application-deplyoment-and-management
https://github.com/awslabs/serverless-application-model/

amazon web services

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: Get items from a DynamoDB table.
Resources:
 GetFunction:
  Type: AWS::Serverless::Function
  Properties:
   Handler: index.get
   Runtime: nodejs4.3
   Policies: AmazonDynamoDBReadOnlyAccess
   Environment:
    Variables:
     TABLE_NAME: !Ref Table
   Events:
    GetResource:
     Type: Api
     Properties:
      Path: /resource/{resourceId}
      Method: get
 Table:
  Type: AWS::Serverless::SimpleTable
```

Sample SAM template for:

- Lambda function
- HTTP GET API
- DynamoDB table

Going further

Latest Lambda features

- 18/11 Environment variables
- 01/12 New service: AWS Lambda@Edge
- 01/12 New service: AWS Step Functions
- 01/12 New service: AWS Greengrass
- 01/12 Dead letter queues
- 01/12 C# support

https://aws.amazon.com/fr/blogs/compute/simplify-serverless-applications-with-environment-variables-in-aws-lambda/
https://aws.amazon.com/blogs/aws/coming-soon-lambda-at-the-edge/
https://aws.amazon.com/blogs/aws/new-aws-step-functions-build-distributed-applications-using-visual-workflows/
https://aws.amazon.com/blogs/aws/aws-greengrass-ubiquitous-real-world-computing/
https://aws.amazon.com/fr/blogs/compute/robust-serverless-application-design-with-aws-lambda-dlq/
https://aws.amazon.com/fr/blogs/compute/announcing-c-sharp-support-for-aws-lambda/

New Lambda videos from re:Invent 2016

AWS re:Invent 2016: What's New with AWS Lambda (SVR202)https://www.youtube.com/watch?v=CwxWhyGteNc

AWS re:Invent 2016: Serverless Apps with AWS Step Functions (SVR201)
https://www.youtube.com/watch?v=75MRve4nv8s

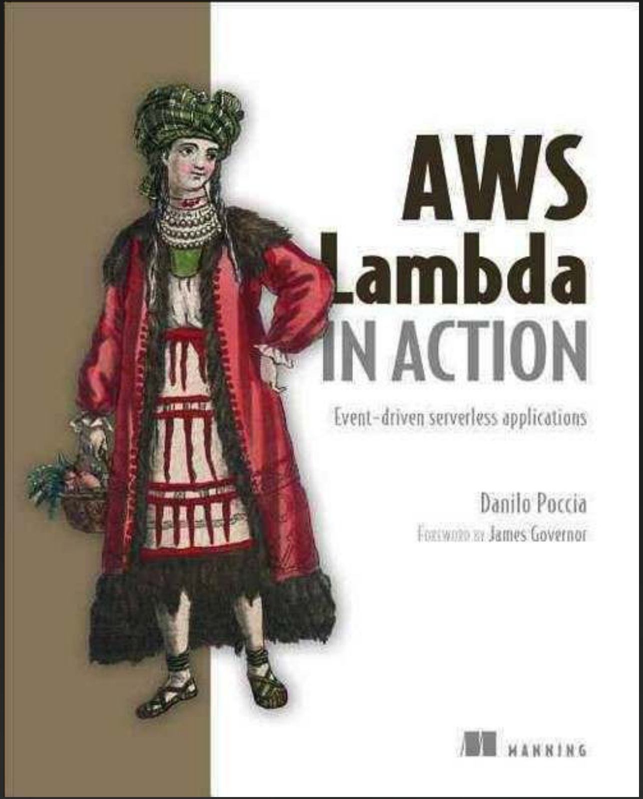
AWS re:Invent 2016: Real-time Data Processing Using AWS Lambda (SVR301)
https://www.youtube.com/watch?v=VFLKOy4GKXQ

AWS re:Invent 2016: Serverless Architectural Patterns and Best Practices (ARC402)
https://www.youtube.com/watch?v=b7UMoc1iUYw

AWS re:Invent 2016: Bringing AWS Lambda to the Edge (CTD206)
https://www.youtube.com/watch?v=j26novaqF6M

AWS re:Invent 2016: Ubiquitous Computing with Greengrass (IOT201)
https://www.youtube.com/watch?v=XQQjX8GTEko

The only Lambda book you need to read


AWS
Lambda
IN ACTION
Event-driven serverless applications
Danilo Poccia
Foreword by James Governor
MANNING

Written by AWS Technical Evangelist Danilo Poccia

Released in Nov'16

https://www.amazon.com/Aws-Lambda-Action-Event-driven-Applications/dp/1617293717/

Thank you!

Julien Simon, Principal Technical Evangelist, AWS
julsimon@amazon.fr
@julsimon