Building a serverless data pipeline

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

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

AWS Lambda

- Deploy pure functions in Java, Python and Node.js

- Works nicely with AWS managed services:
  Amazon S3, Amazon DynamoDB, etc.

- Build event-driven applications

- Build RESTful APIs in conjunction with Amazon API Gateway

- Pay as you go: number of requests + execution time (100ms slots)

Managed services

+

AWS Lambda

=

Serverless architecture

Another way to put it...

Tim Wagner,
General Manager,
AWS Lambda

Serverless conference, NYC, May 2016

Selected serverless customers

FireEye
THREAT INTELLIGENCE
AND ANALYTICS

kik·
MOBILE
CHAT APP

AdRoll
AD DATA ANALYTICS
AND ROUTING

Localytics
MOBILE APP
ANALYTICS
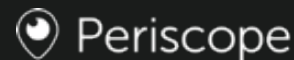
Periscope
IMAGE CONTENT
FILTERING

Quicken Loans
WEB
APPLICATIONS

Coca-Cola
WEB APPLICATIONS

Zillow.com.
Your Edge in Real Estate
DATA
PROCESSING

twilio
CLOUD
TELEPHONY

VidRoll
REAL-TIME VIDEO
AD BIDDING

NORDSTROM
PRODUCT
RECOMMANDATION

THOMSON REUTERS
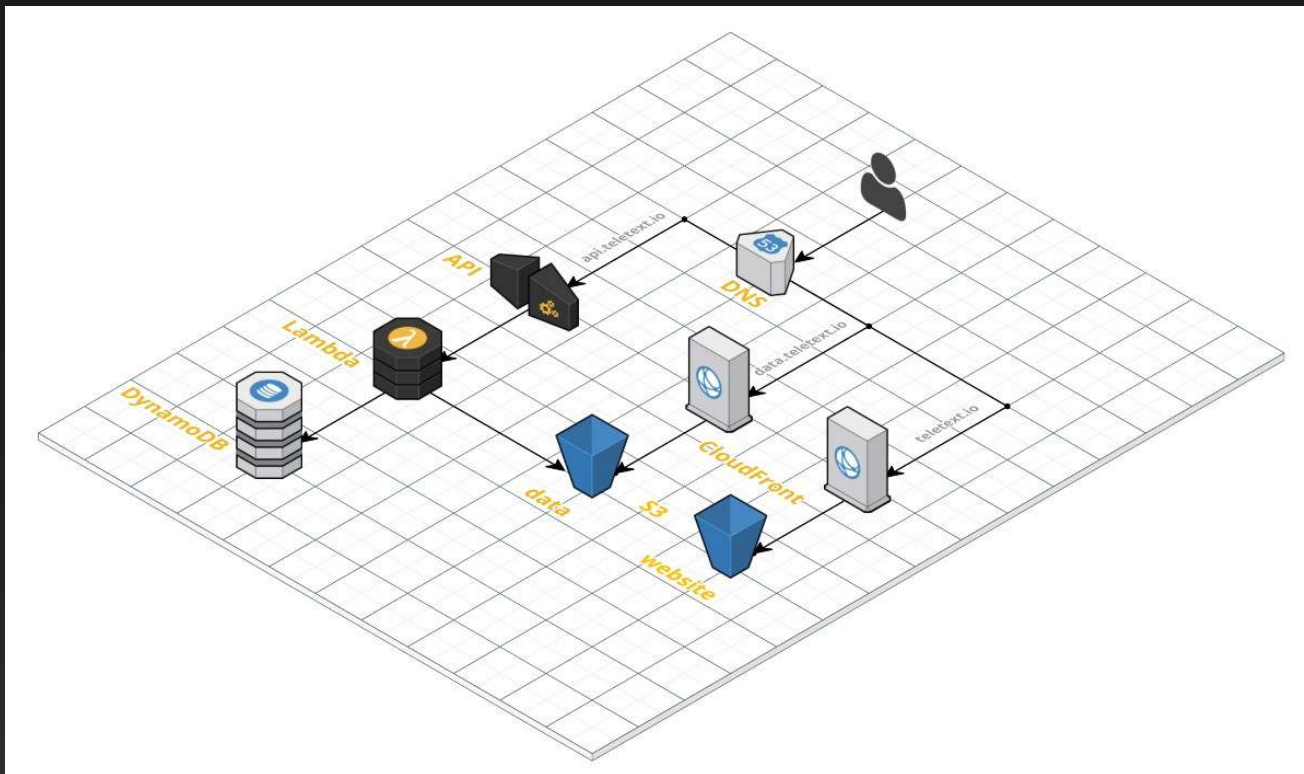NEWS CONTENT
PROCESSING

BUSTLE
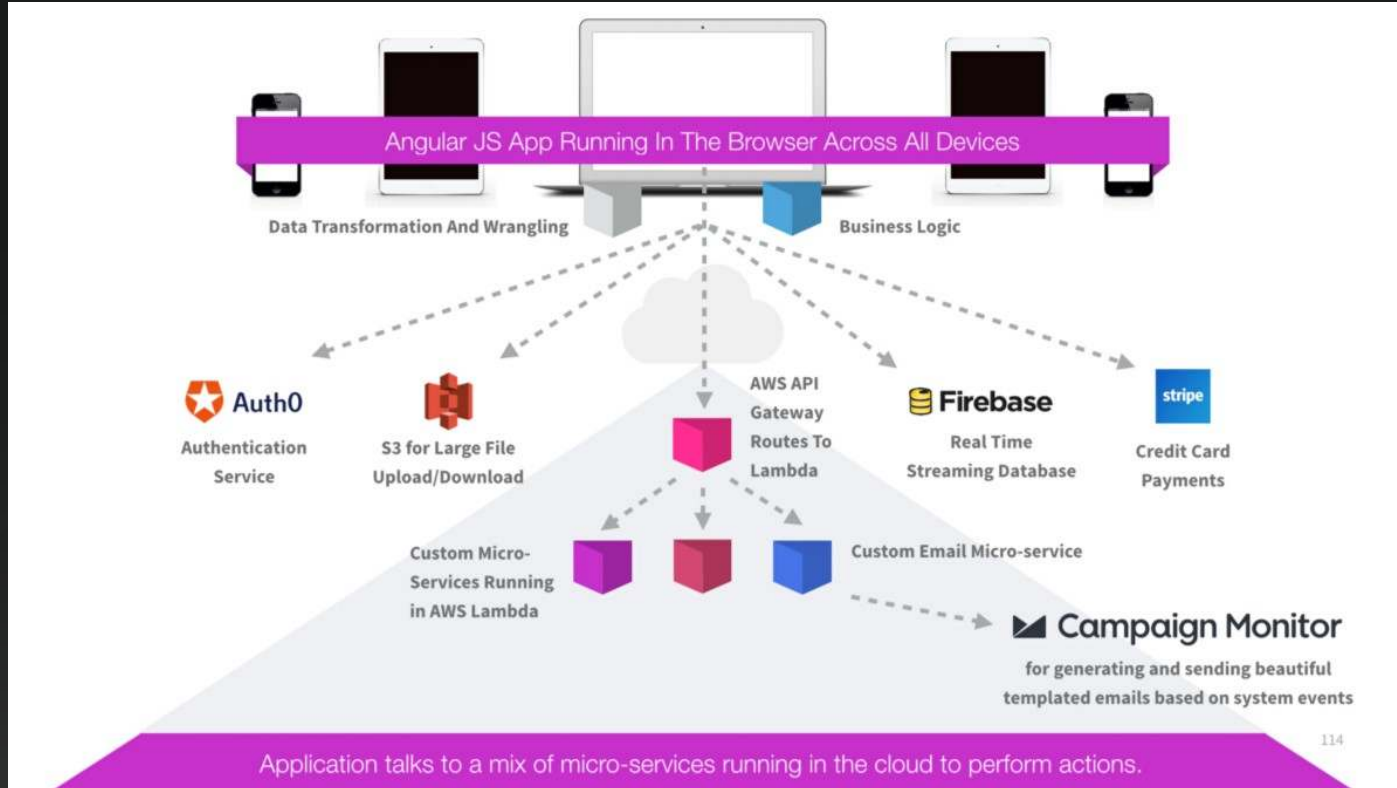NEWS CONTENT
PROCESSING

Benchling
GENE SEQUENCE
SEARCH

mlbam
GAME METRICS ANALYTICS

amazon
web services

Instant.cm: 100% Serverless

A Cloud Guru: 100% Serverless

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

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

AWS Lambda 'Hello World' (Python)

1. Write a simple Lambda function in Python

2. Create a REST API with API Gateway (resource + POST method)

3. Deploy the API
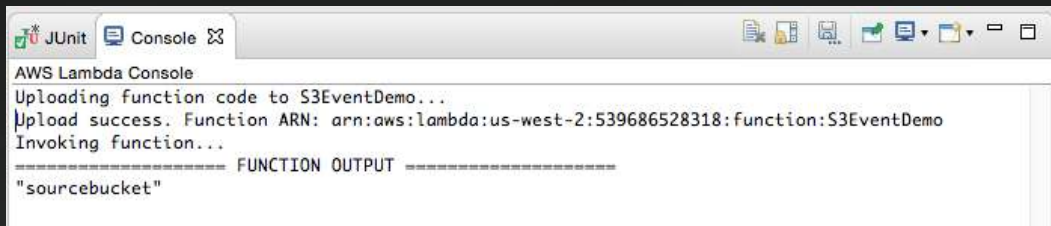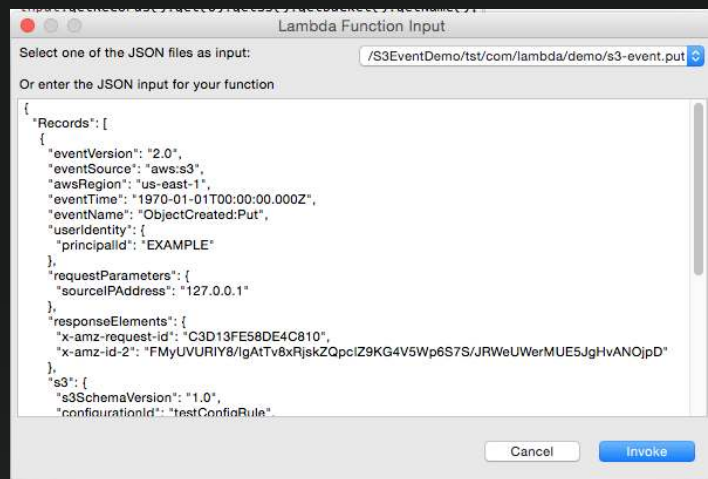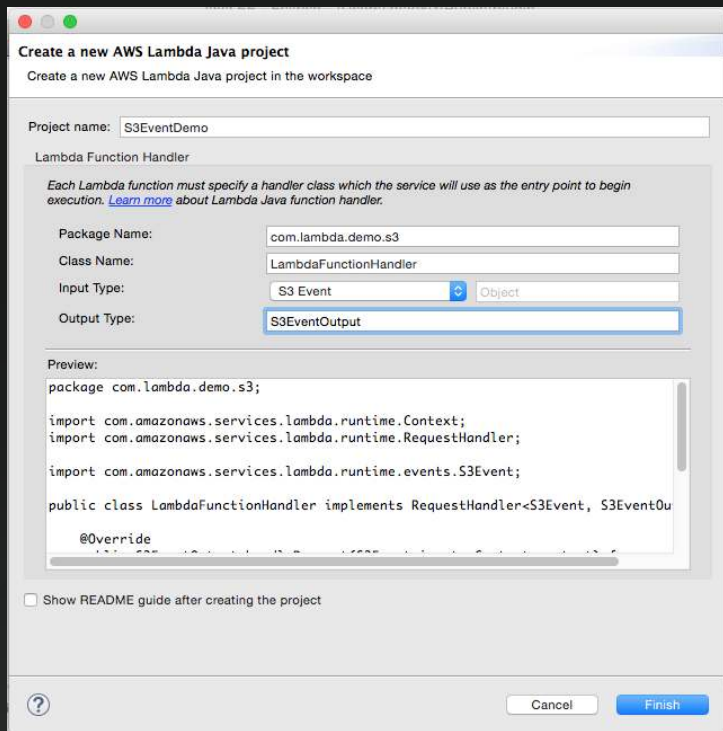
4. Invoke the API with '*curl*'

A simple Lambda function in Python

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

```
aws lambda create-function --function-name myFunc \
--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/STAGE/RESOURCE
```

AWS Lambda in Java with Eclipse

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

AWS Lambda with the Serverless framework

SERVERLESS.COM

POWERED BY
amazon
web services™

http://github.com/serverless/serverless

- Run/test AWS Lambda functions locally, or remotely

- Auto-deploys & versions your Lambda functions

- Auto-deploys your REST API to AWS API Gateway

- Auto-deploys your Lambda events

- Support for multiple stages

- Support for multiple regions within stages

- Manage & deploy AWS CloudFormation resources

amazon
web services

Building a serverless data pipeline
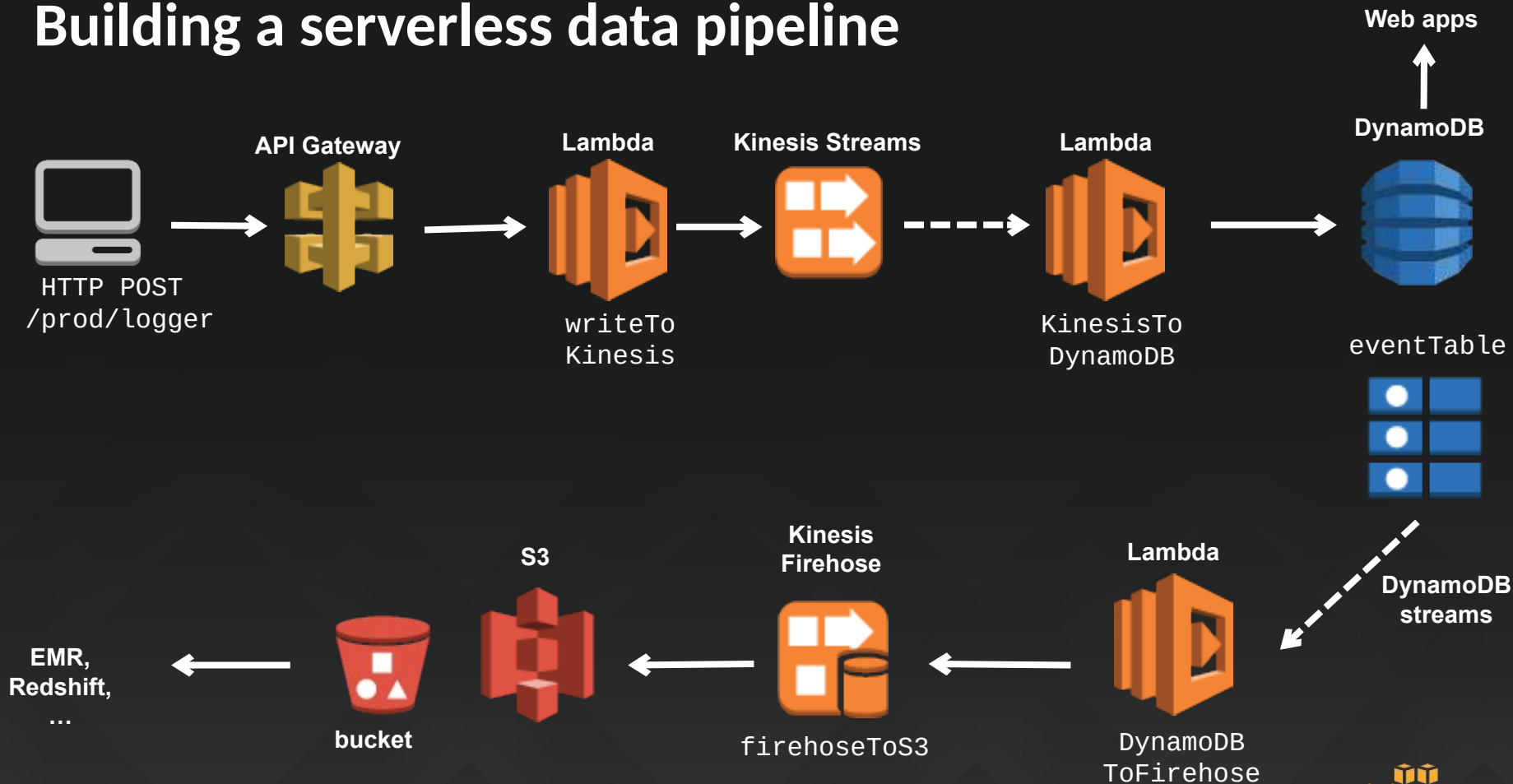
Step 1: create DynamoDB table

```
aws dynamodb create-table \
--table-name eventTable \
--attribute-definitions \
AttributeName=userId,AttributeType=N \
AttributeName=timestamp,AttributeType=N \
--key-schema \
AttributeName=userId,KeyType=HASH \
AttributeName=timestamp,KeyType=RANGE \
--provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5 \
--stream-specification StreamEnabled=true,StreamViewType=NEW_IMAGE
```


DynamoDB

eventTable
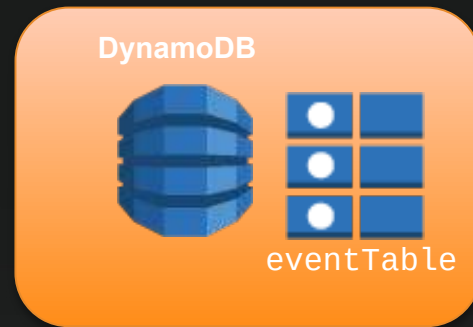
Step 2: IAM role for Lambda function

```
aws iam create-role \
--role-name writeToDynamoDB_role \
--assume-role-policy-document file://lambda_trust_policy.json

aws iam create-policy \
--policy-name writeToDynamoDB_policy \
--policy-document file://writeToDynamoDB_policy.json

aws iam attach-role-policy \
--role-name writeToDynamoDB_role \
--policy-arn WRITETODYNAMODB_POLICY_ARN
```

Step 3: create Lambda function

```
aws lambda create-function \
--function-name writeToDynamoDB \
--role WRITETODYNAMO_DB_ROLE \
--zip-file fileb://writeToDynamoDB.zip \
--handler writeToDynamoDB.lambda_handler \
--runtime python2.7 \
--memory-size 128 \
--description "Write events to DynamoDB"
```

Web apps

Lambda

DynamoDB

dynamodb.
put_item()

eventTable

Step 4: create Kinesis Stream

Kinesis Streams

Lambda

DynamoDB

Web apps

dynamodb.
put_item()

eventTable

```
aws kinesis create-stream --stream-name APItoDynamoDB --shard-count 1
```
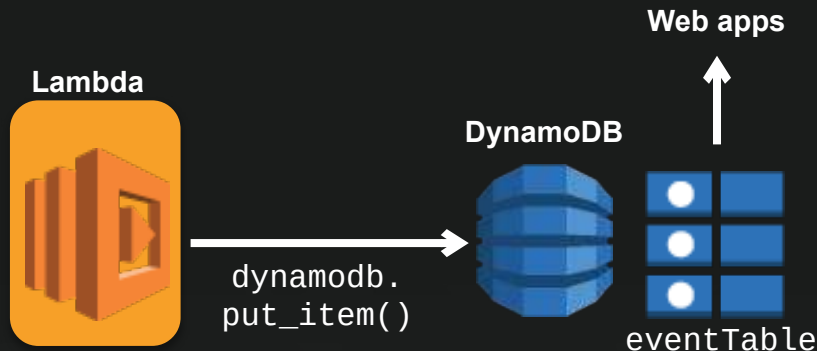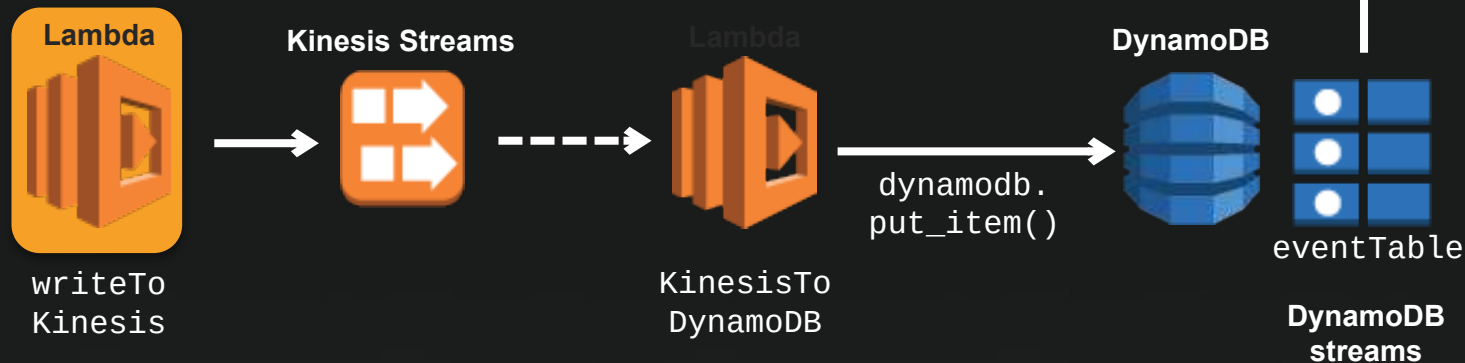
Step 5: IAM role for Lambda function

```
aws iam create-role \
--role-name writeToKinesis_role \
--assume-role-policy-document file://lambda_trust_policy.json

aws iam create-policy \
--policy-name writeToKinesis_policy \
--policy-document file://writeToKinesis_policy.json

aws iam attach-role-policy \
--role-name writeToKinesis_role \
--policy-arn WRITETOKINESIS_POLICY_ARN
```
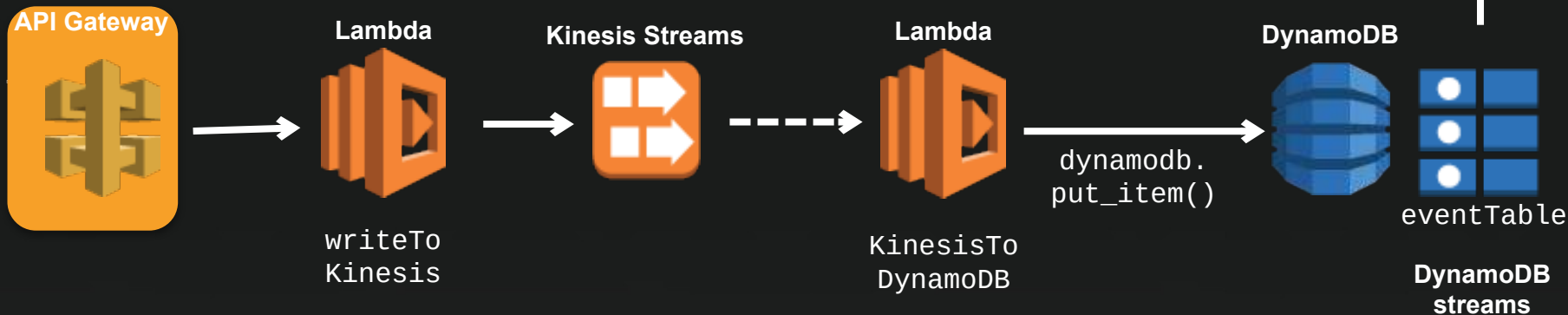
Step 6: create Lambda function


```
aws lambda create-function \
--function-name writeToKinesis\
--role WRITETOKINESIS_ROLE \
--zip-file fileb://writeToKinesis.zip \
--handler writeToKinesis.lambda_handler \
--runtime python2.7 \
--memory-size 128 \
--description "Write events to Kinesis"
```

Step 7: create API

Web apps

API Gateway   **Lambda**   **Kinesis Streams**   **Lambda**   **DynamoDB**

```
writeTo
Kinesis
```

```
KinesisTo
DynamoDB
```

```
dynamodb.
put_item()
```

```
eventTable
```

DynamoDB streams

Painful to do with the CLI: 9 aws apigateway calls :-/

→ Use the console

→ Use a Swagger File
http://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-import-api.html

→ Use the Serverless framework

amazon
web services

Step 8: create IAM role

```
aws iam create-role \
--role-name DynamoDBToFirehose_role \
--assume-role-policy-document file://lambda_trust_policy.json

aws iam create-policy \
--policy-name DynamoDBToFirehose_policy \
--policy-document file://DynamoDBToFirehose_policy.json

aws iam attach-role-policy \
--role-name DynamoDBToFirehose_role \
--policy-arn DYNAMODBTOFIREHOSE_POLICY_ARN
```

Step 9: create Lambda function and DynamoDB trigger

```
aws lambda create-function \
--function-name DynamoDBToFirehose \
--role DYNAMODBTOFIREHOSE_ROLE_ARN \
--zip-file fileb://DynamoDBToFirehose.zip \
--handler DynamoDBToFirehose.lambda_handler \
--runtime python2.7 \
--memory-size 128 \
--description "Write DynamoDB stream to Kinesis Firehose"

aws lambda create-event-source-mapping \
--function-name DynamoDBToFirehose \
--event-source DYNAMODB_STREAM_ARN \
--batch-size 10 \
--starting-position TRIM_HORIZON
```

DynamoDB

eventTable

Lambda

DynamoDB
streams

DynamoDB
ToFirehose

Step 10: create IAM role

```
aws iam create-role \
--role-name firehoseToS3_role \
--assume-role-policy-document file://firehose_trust_policy.json

aws iam create-policy \
--policy-name firehoseToS3_policy \
--policy-document file://firehoseToS3_policy.json

aws iam attach-role-policy \
--role-name firehoseToS3_role \
--policy-arn FIREHOSETOS3_POLICY_ARN
```

Step 11: create S3 bucket

```
aws s3 mb s3://jsimon-public
```


Lambda

eventTable

DynamoDB
ToFirehose
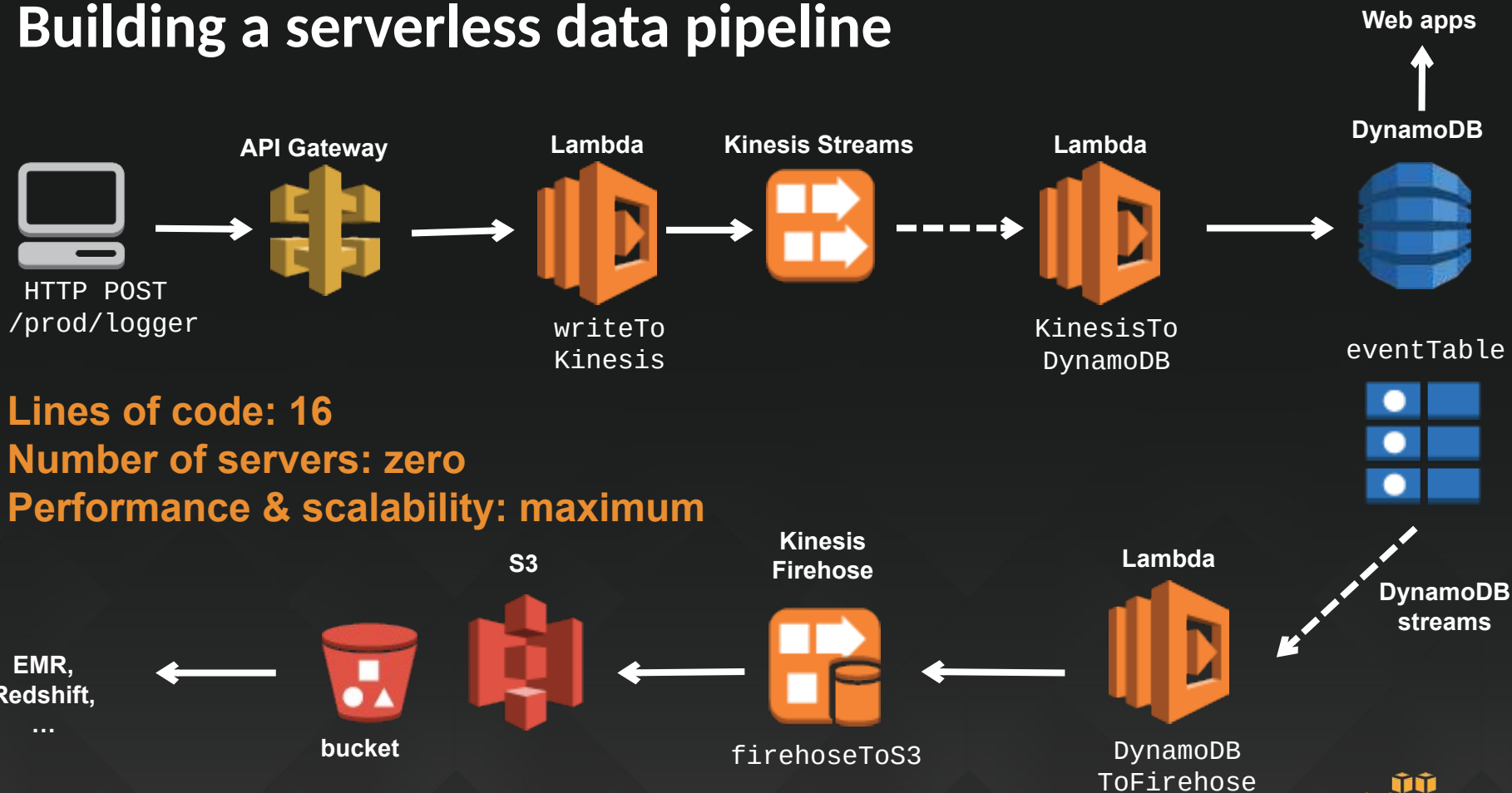
**DynamoDB
streams**

DynamoDB
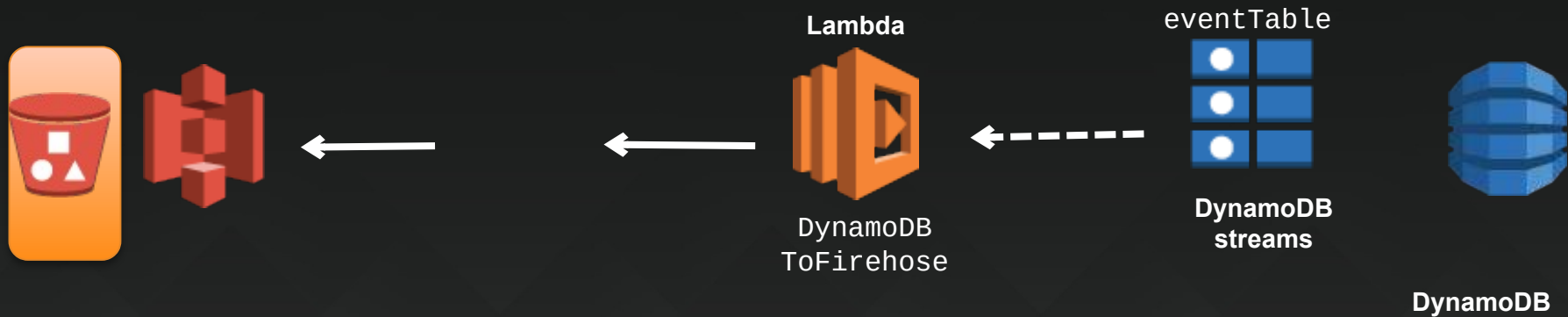
Step 12: create Kinesis Firehose stream

```
aws firehose create-delivery-stream \
--delivery-stream-name firehoseToS3 \
--s3-destination-configuration \
RoleARN=FIREHOSETOS3_ROLE_ARN, \
BucketARN="arn:aws:s3:::jsimon-public", \
Prefix="firehose", \
BufferingHints=\{SizeInMBs=1,IntervalInSeconds=60\}, \
CompressionFormat="GZIP", \
EncryptionConfiguration={NoEncryptionConfig="NoEncryption"}
```
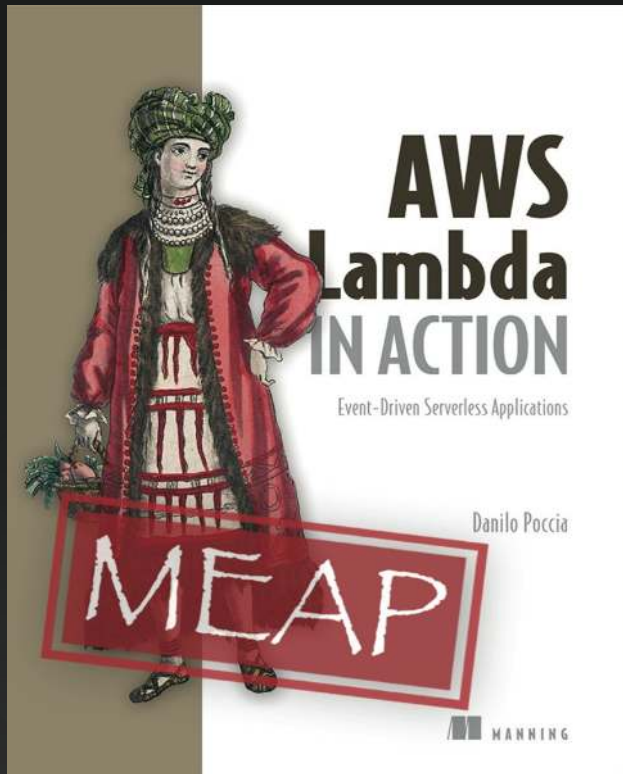
Building a serverless data pipeline

API Gateway

Lambda

Kinesis Streams

Lambda

DynamoDB

Web apps

HTTP POST
/prod/logger

writeTo
Kinesis

KinesisTo
DynamoDB

eventTable

Lines of code: 16
Number of servers: zero
Performance & scalability: maximum

S3

**Kinesis
Firehose**

Lambda

DynamoDB
streams

EMR,
Redshift,
...

bucket

firehoseToS3

DynamoDB
ToFirehose

https://github.com/juliensimon/aws/tree/master/serverlessPipeline

amazon
web services

Ready for some testing?
http://api.julien.org

Upcoming book on AWS Lambda

Written by AWS Technical Evangelist Danilo Poccia

Early release available at:

https://www.manning.com/books/aws-lambda-in-action

Going further

AWS re:Invent 2014 | (MBL202) NEW LAUNCH: Getting Started with AWS Lambda
https://www.youtube.com/watch?v=UFj27IaTWQA

AWS re:Invent 2015 | (DEV203) Amazon API Gateway & AWS Lambda to Build Secure and Scalable APIs
https://www.youtube.com/watch?v=ZBxWZ9bgd44

AWS re:Invent 2015 | (DVO209) JAWS: The Monstrously Scalable Serverless Framework
https://www.youtube.com/watch?v=D_U6luQ6I90
https://github.com/serverless/serverless

AWS re:Invent 2015 | (ARC308) The Serverless Company Using AWS Lambda
https://www.youtube.com/watch?v=U8ODkSCJpJU

AWS re:Invent 2015 | (CMP407) Lambda as Cron: Scheduling Invocations in AWS Lambda
https://www.youtube.com/watch?v=FhJxTIq81AU

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

AWS User Groups

Lille
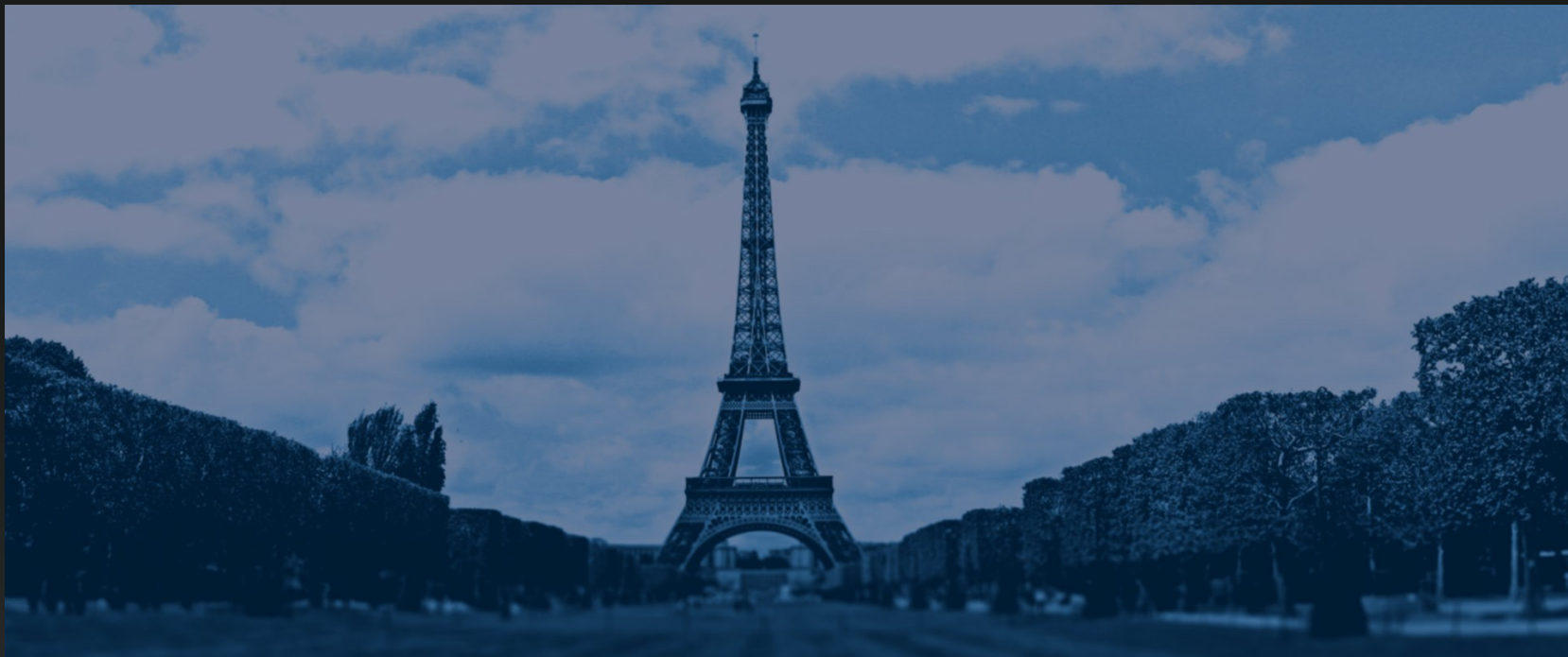Paris
Rennes
Nantes
Bordeaux
Lyon
Montpellier
Toulouse

facebook.com/groups/AWSFrance/

@aws_actus

AWS Enterprise Summit – 27/10/2016, Paris

http://amzn.to/1X2yp0i

Merci !

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