Building a serverless data pipeline

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

Requirements For Internet-scale Apps

Simplicity

Scalability

Low Cost

Reliability

Low Latency

Building Blocks For Internet-scale Apps

Storage

Amazon S3

Data Store

Amazon DynamoDB

Compute

Evolution of Computing

Weeks

On-premises

Minutes

Amazon EC2

Seconds

Amazon EC2
Container Service

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

amazon web services

AWS Lambda

- No infrastructure to manage: deploy only functions in Java, Python and Node.js
- Built-in scalability and high-availability
- Works nicely with other AWS managed services

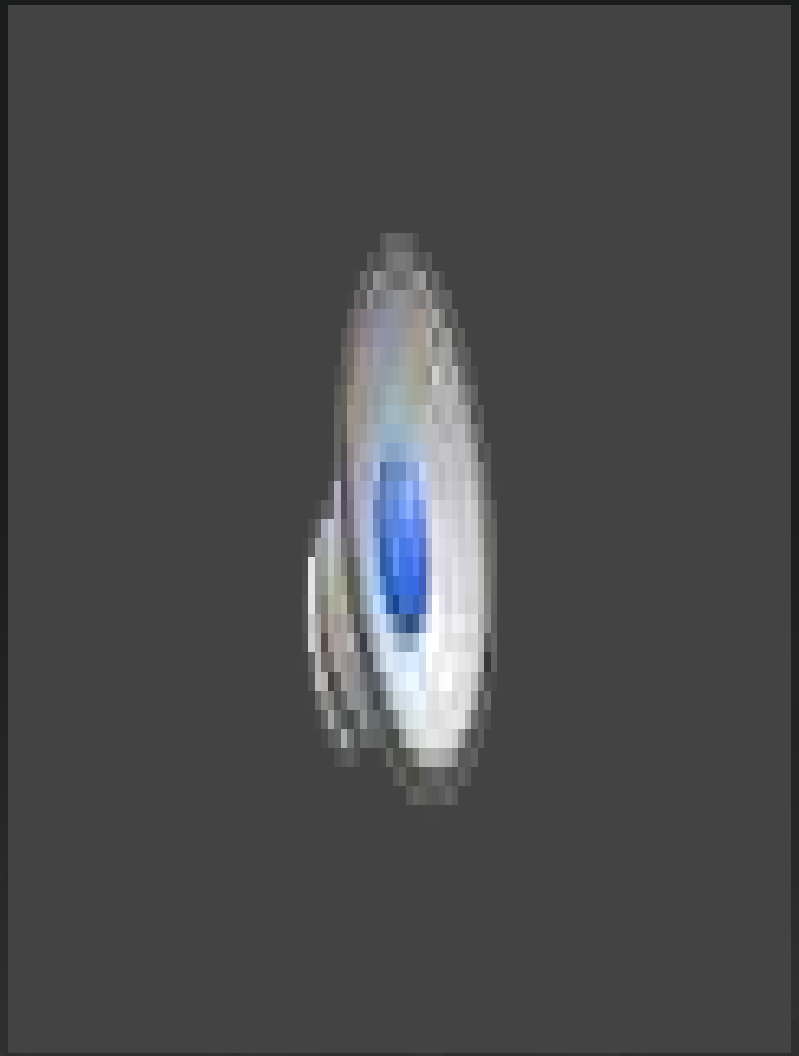
- Build event-driven applications
- Build RESTful APIs in conjunction with Amazon API Gatewa
- Pay as you go:
  number of requests + execution time (100ms slots)

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

THREAT INTELLIGENCE AND ANALYTICS

MOBILE CHAT APP

AD DATA ANALYTICS AND ROUTING

MOBILE APP ANALYTICS

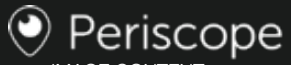


IMAGE CONTENT FILTERING

WEB APPLICATIONS

WEB APPLICATIONS

DATA PROCESSING

CLOUD TELEPHONY

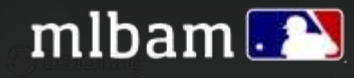
REAL-TIME VIDEO AD BIDDING

PRODUCT RECOMMANDATION

NEWS CONTENT PROCESSING

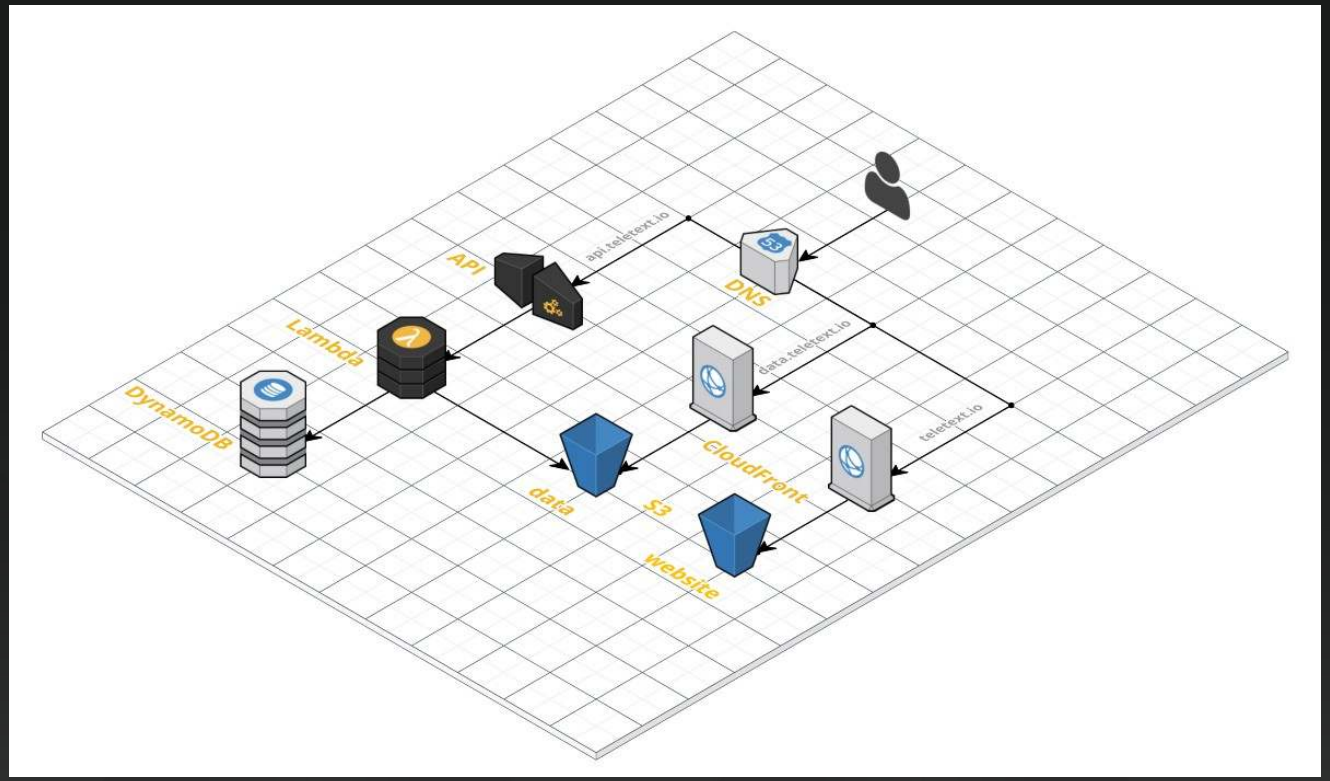
NEWS CONTENT PROCESSING

GENE SEQUENCE SEARCH



GAME METRICS ANALYTICS

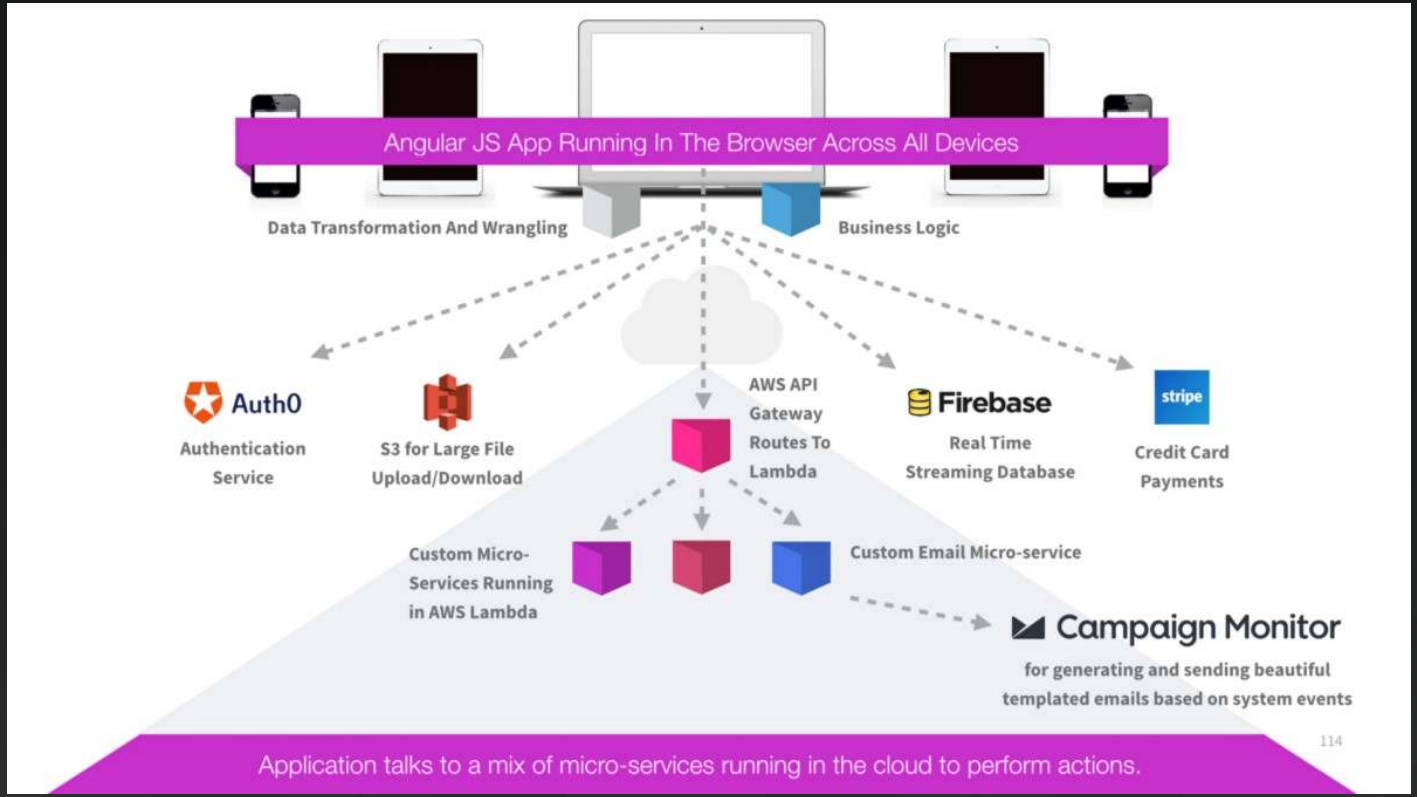
Instant.cm: 100% Serverle



https://blog.instant.cm/a-serverless-architecture-with-zero-maintenance-and-infinite-scalability-b00c2ceb4c2b
http://highscalability.com/blog/2015/12/7/the-serverless-start-up-down-with-servers.html

amazon web services

A Cloud Guru: 100% Serverless



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

AWS Lambda ‘Hello World’ (Python)

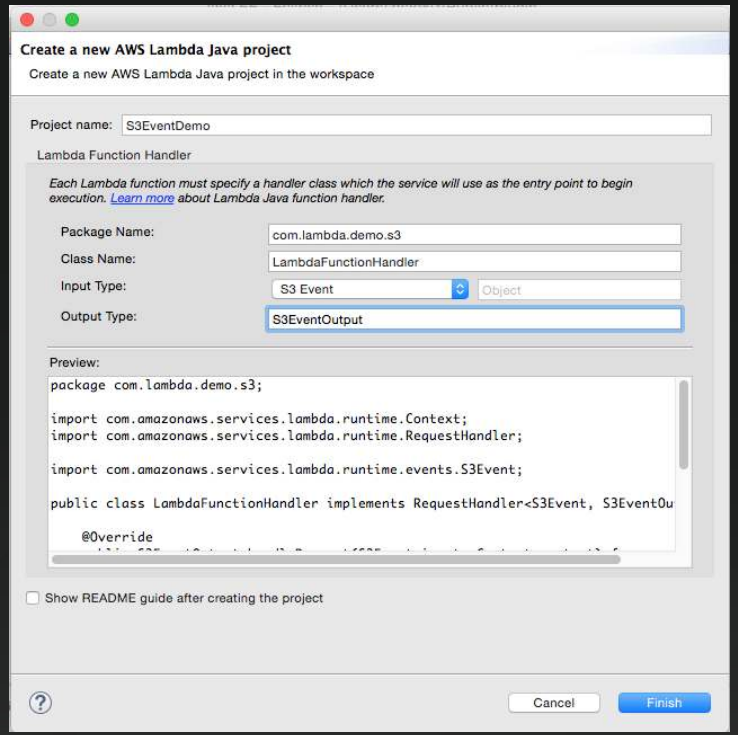
1. Write a simple Lambda function in Python
2. Create a REST API with API Gateway (resource + POST method)
3. Deploy the API
4. Invoke the API with ‘*curl*’

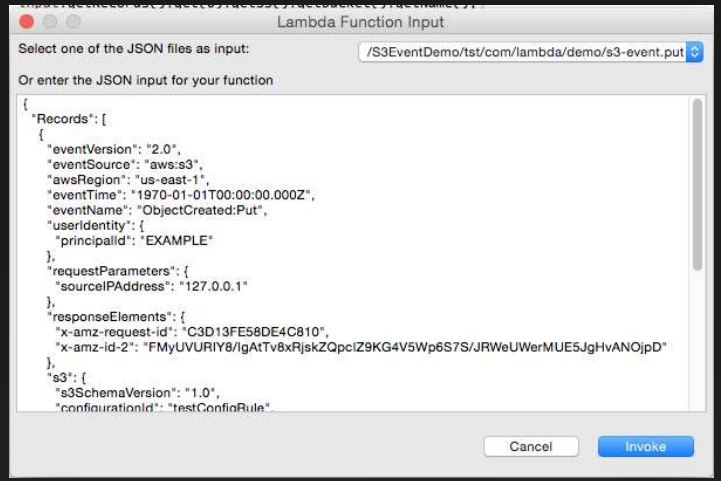
A simple Lambda function in Python

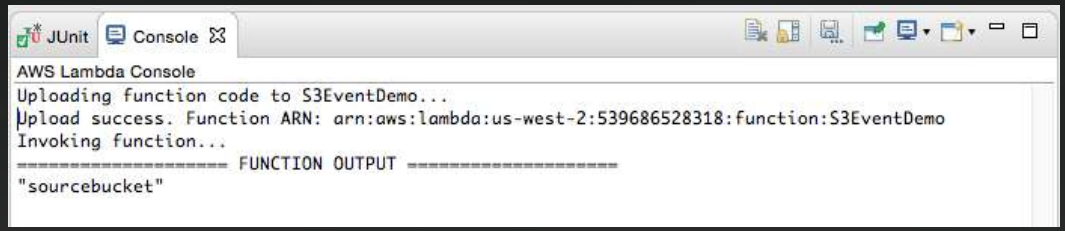
```
def lambda_handler(event,context):
   result = event['value1'] + event['value2']
   return result
```

```
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

amazon web services

AWS Lambda with the Serverless framework

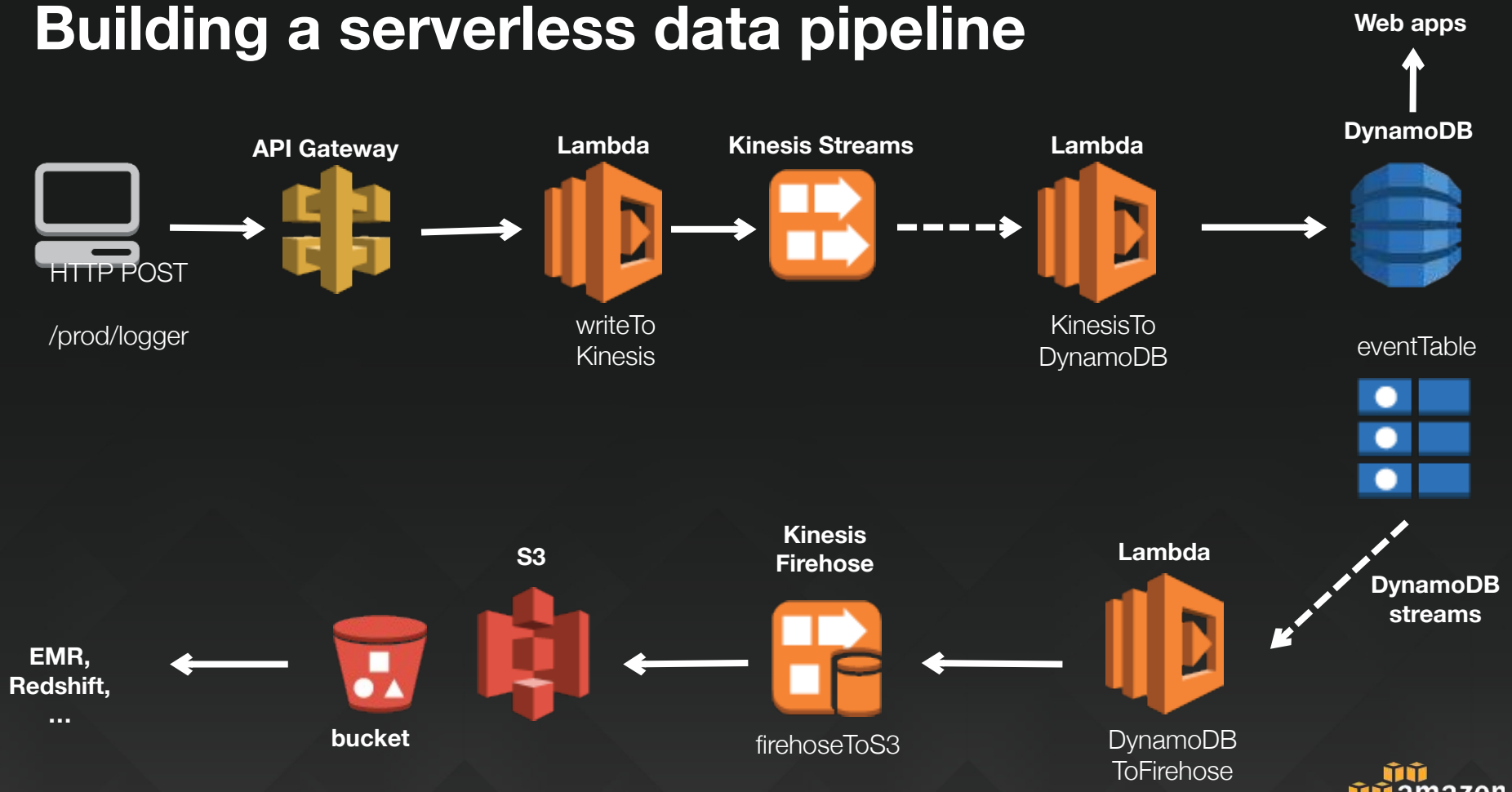
http://github.com/serverless/serverless

- 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

Other notable frameworks: Chalice (Python), Zappa (Python), Apex (golang)

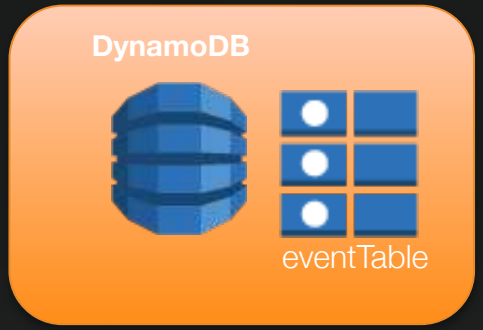
amazon web services

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
```

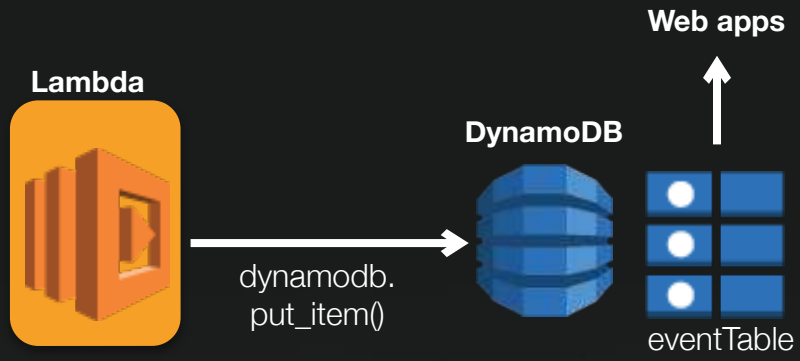


Step 2: IAM role for Lambda function

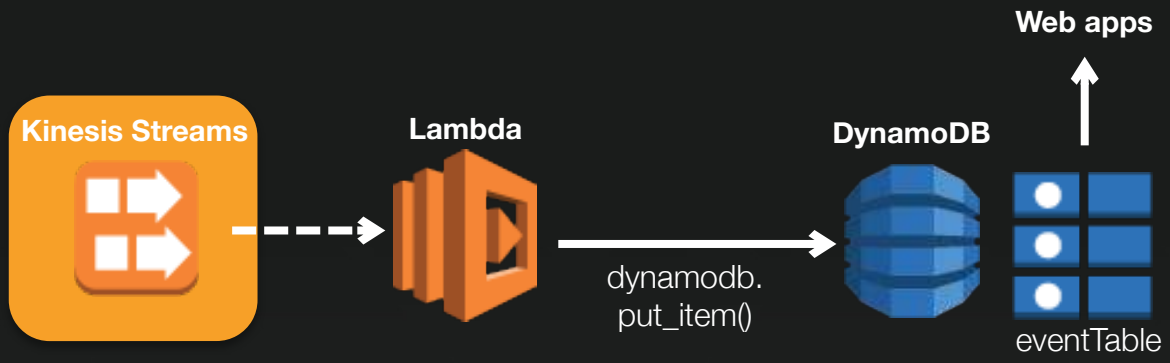
```
aws iam create-role \
--role-name writeToDynamoDB_role \
--assume-role-policy-document file://lambda_trust_policy.json

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”
```



Step 4: create Kinesis Stream

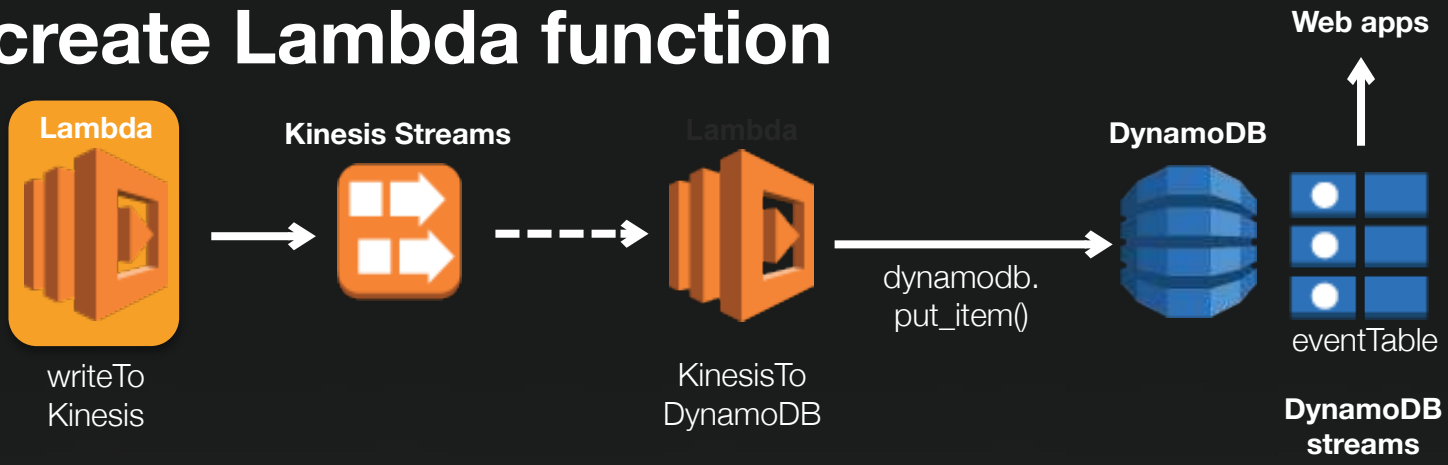



```
aws kinesis create-stream --stream-name APItoDynamoDB --shard-count 1
```

Step 5: IAM role for Lambda function

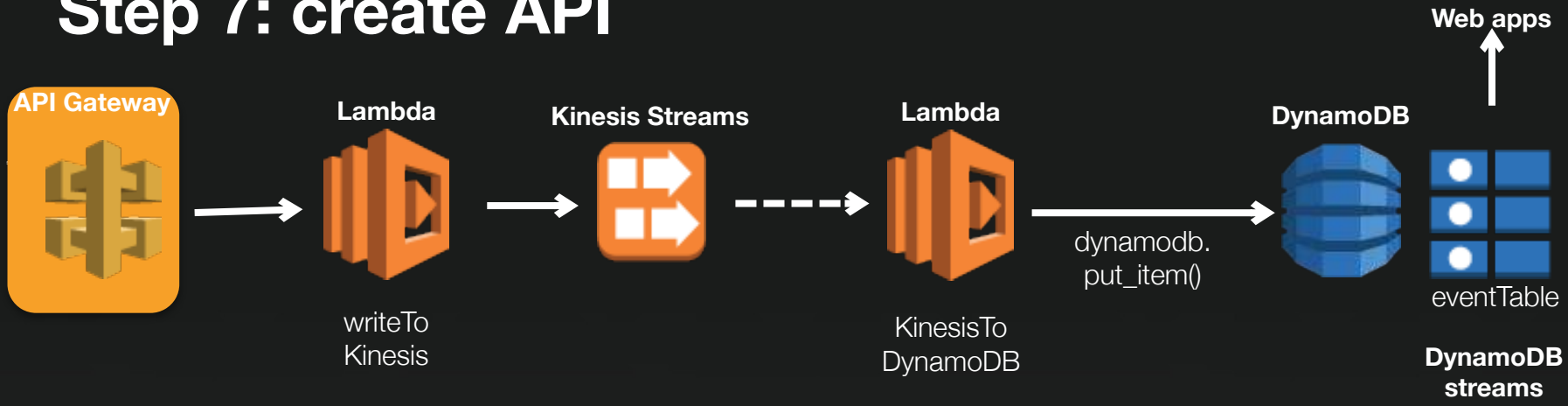
```
aws iam create-role \
--role-name writeToKinesis_role \
--assume-role-policy-document file://lambda_trust_policy.json

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



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 a framework (Serverless, Chalice, etc.)

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
```

Step 10: create IAM role

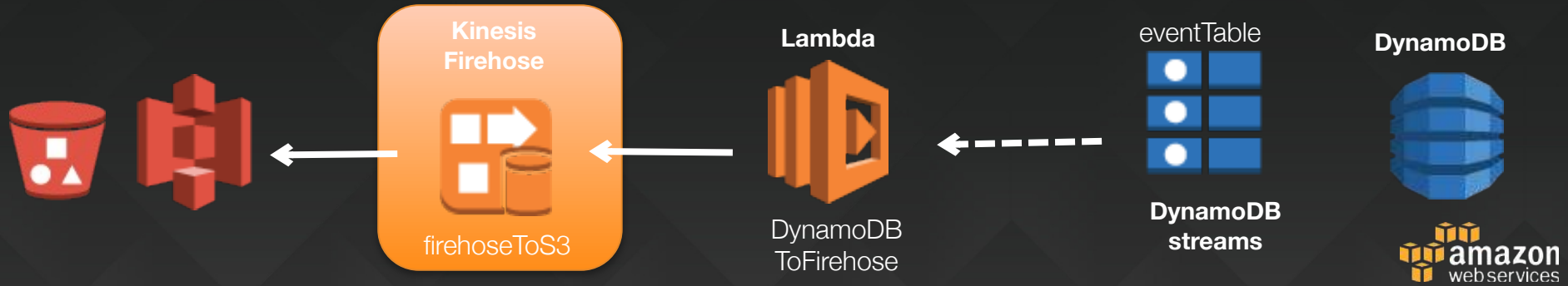
```
aws iam create-role \
--role-name firehoseToS3_role \
--assume-role-policy-document file://firehose_trust_policy.json

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

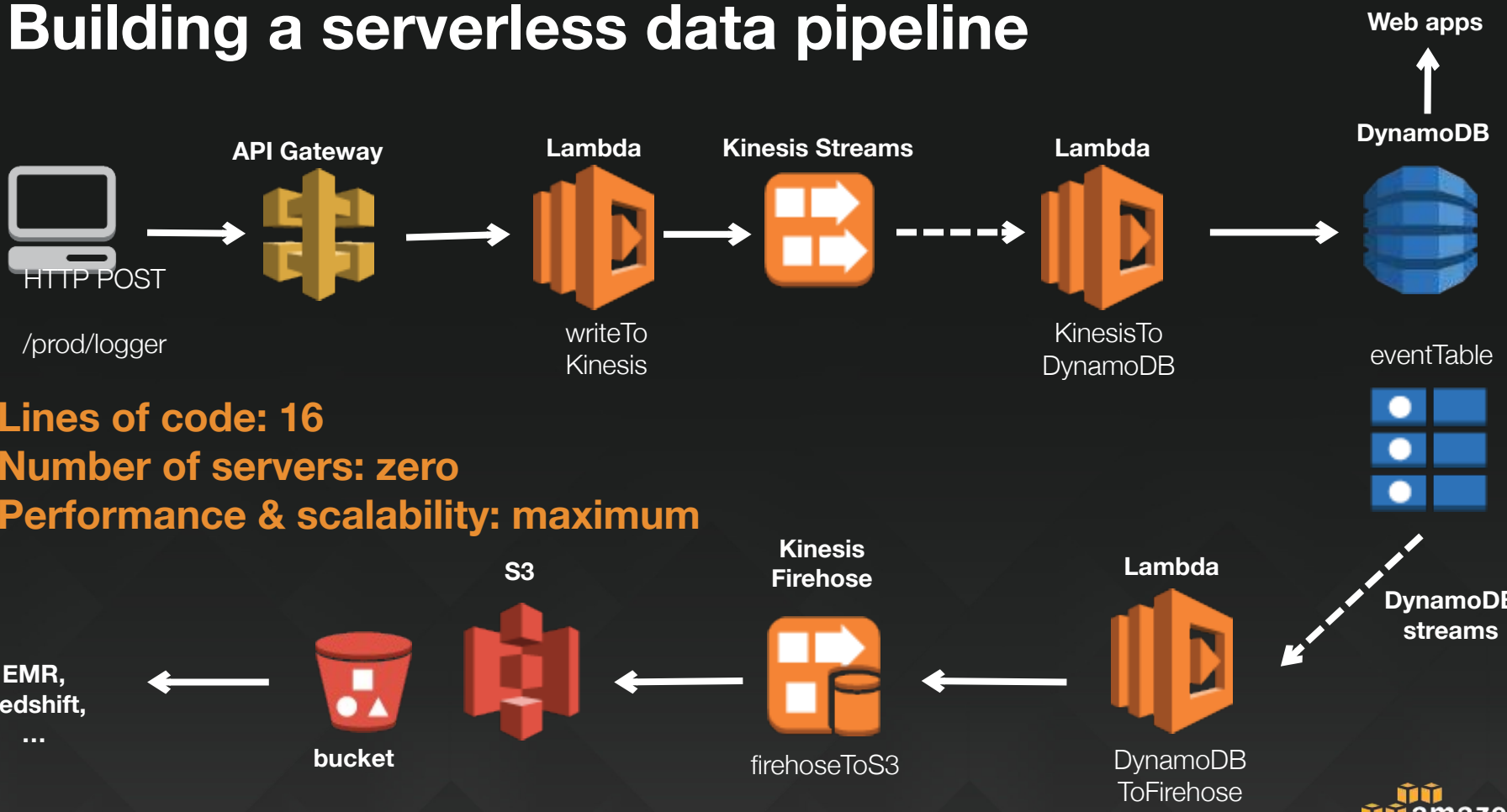
```
aws s3 mb s3://jsimon-public
```

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

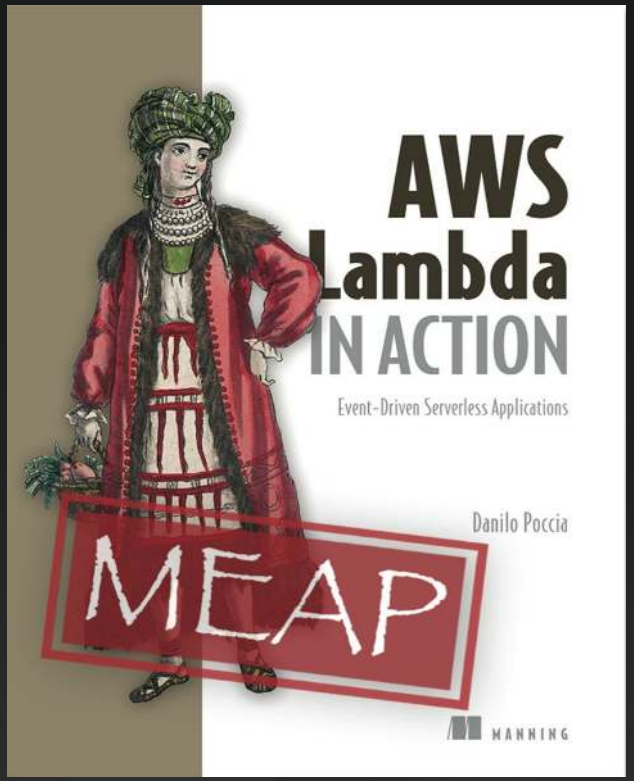


https://github.com/juliensimon/aws/tree/master/serverlessPipeline

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

amazon web services

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

Dank u wel!

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

amazon web services