Building a serverless data pipeline

API Gateway

Lambda

DynamoDB

Kinesis Firehose

S3

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

amazon web services

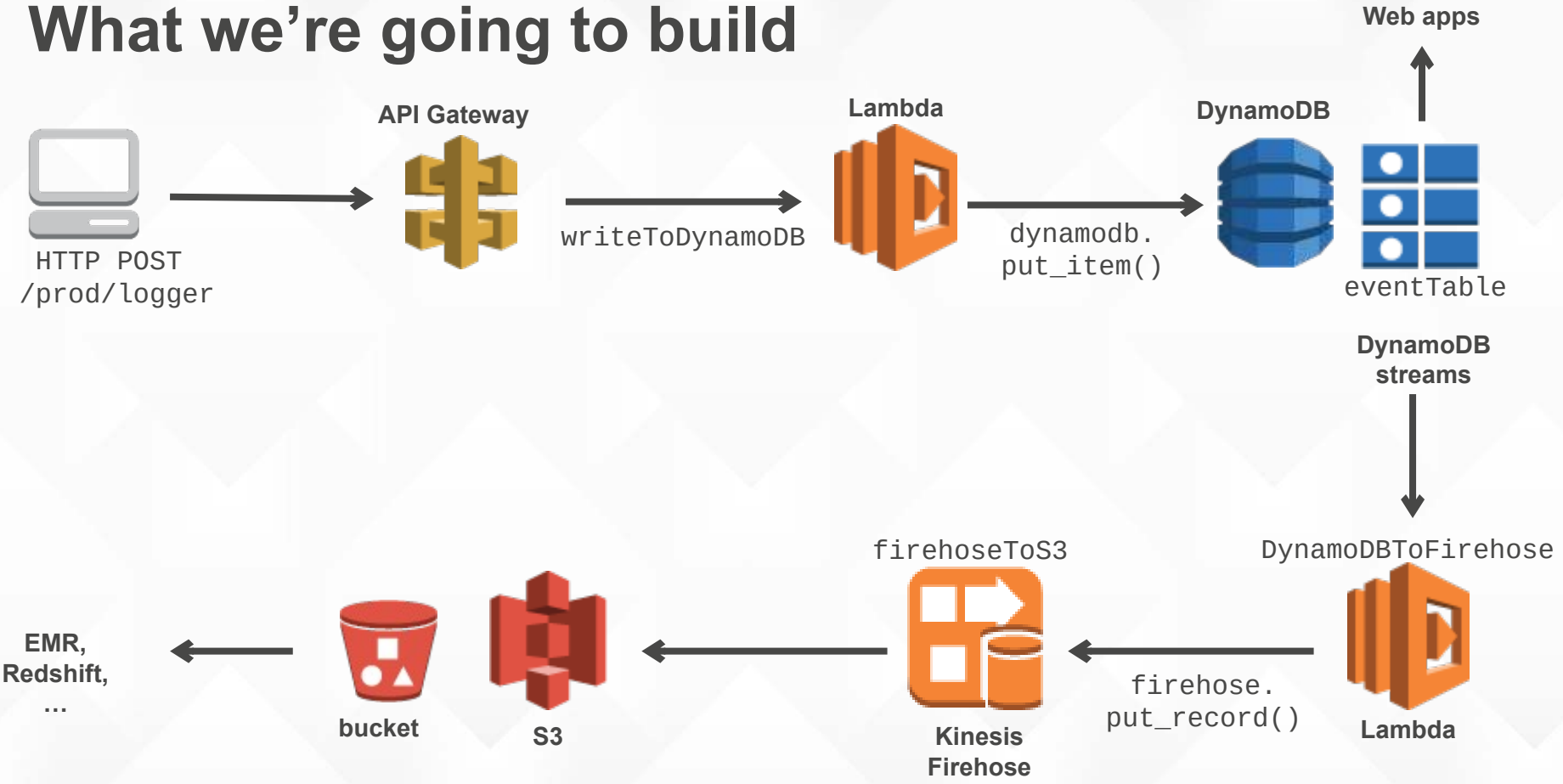
AWS Lambda

- Code as a Service, launched in 2014
- Supports Java 8, Python 2.7 and Node.js v0.10.36

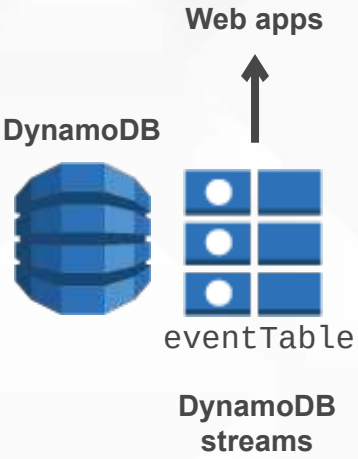
- Build event-driven applications
- Build APIs in conjunction with Amazon API Gateway
- Interact with other AWS services (S3, DynamoDB, etc)
- Log automatically to CloudWatch Logs

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

What we're going to build



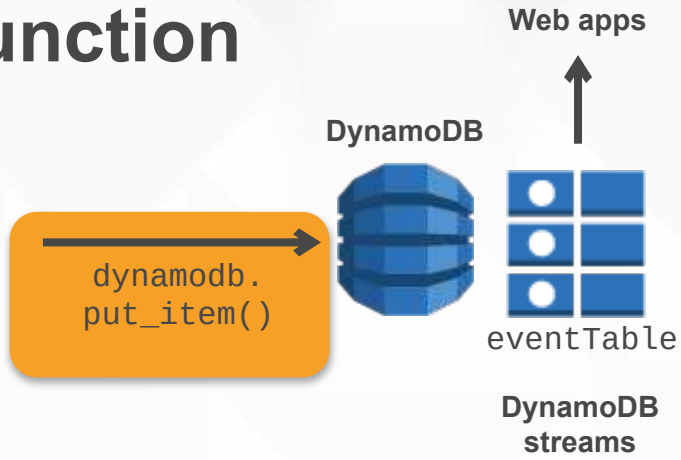
Step 1: create DynamoDB table



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



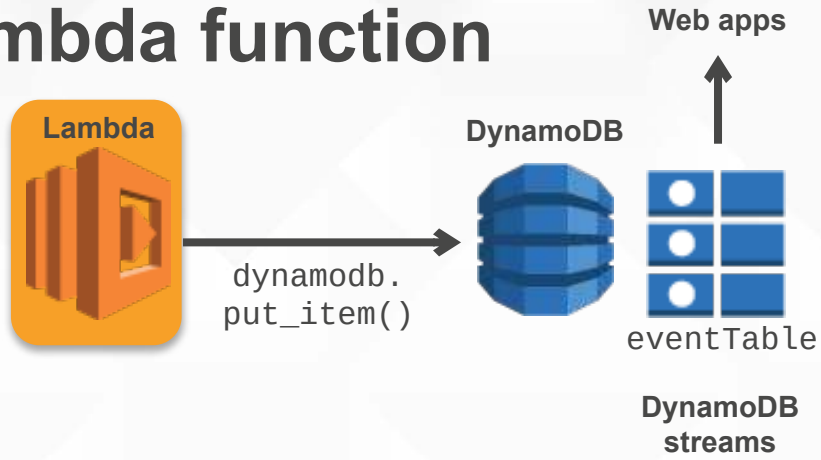
Step 2: IAM role for Lambda function

```
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 and test Lambda function

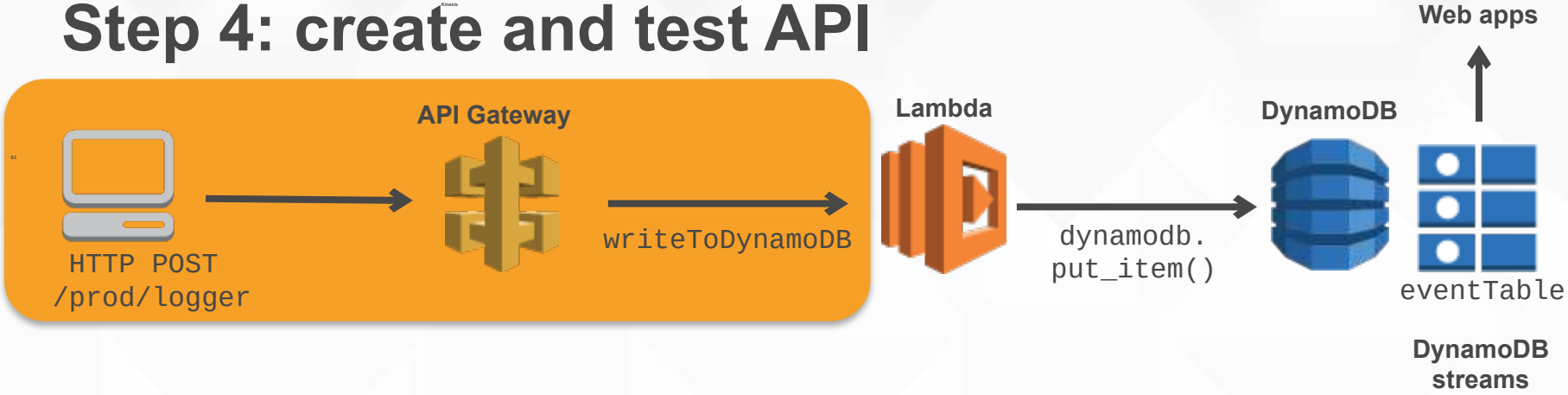


Step 3: create Lambda function

```
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"

aws lambda invoke --function-name writeToDynamoDB \
--payload "{\"userId\":99999999, \"value\":7}" \
--invocation-type RequestResponse output.txt
```

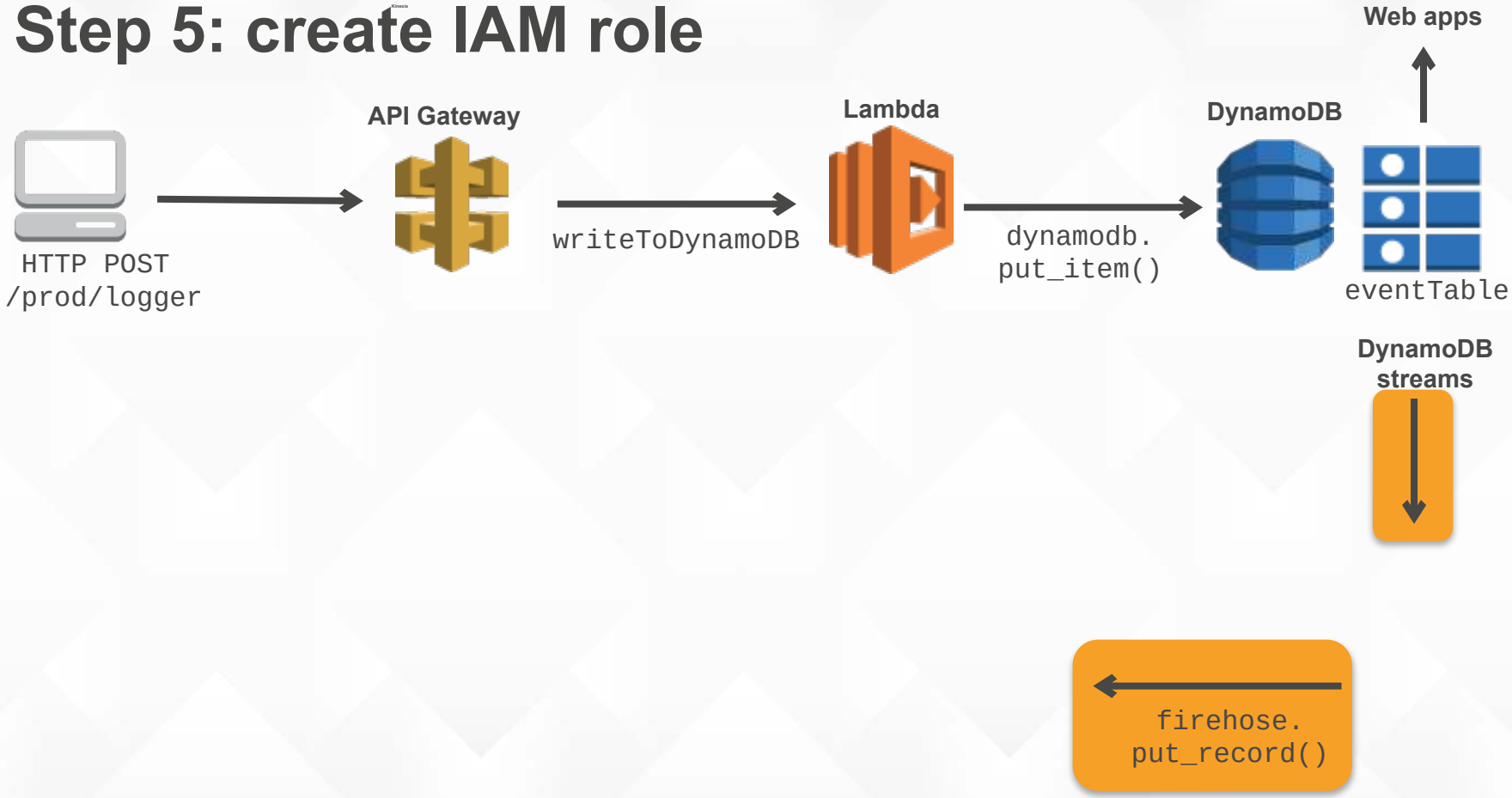
Step 4: create and test API



Step 4: create and test API

```
curl -H "Content-Type: application/json" -X POST -d
"{\"userId\":99999999, \"value2\":7}" https://8rzdhuccp7.execute-
api.eu-west-1.amazonaws.com/prod/logger
```

Step 5: create IAM role



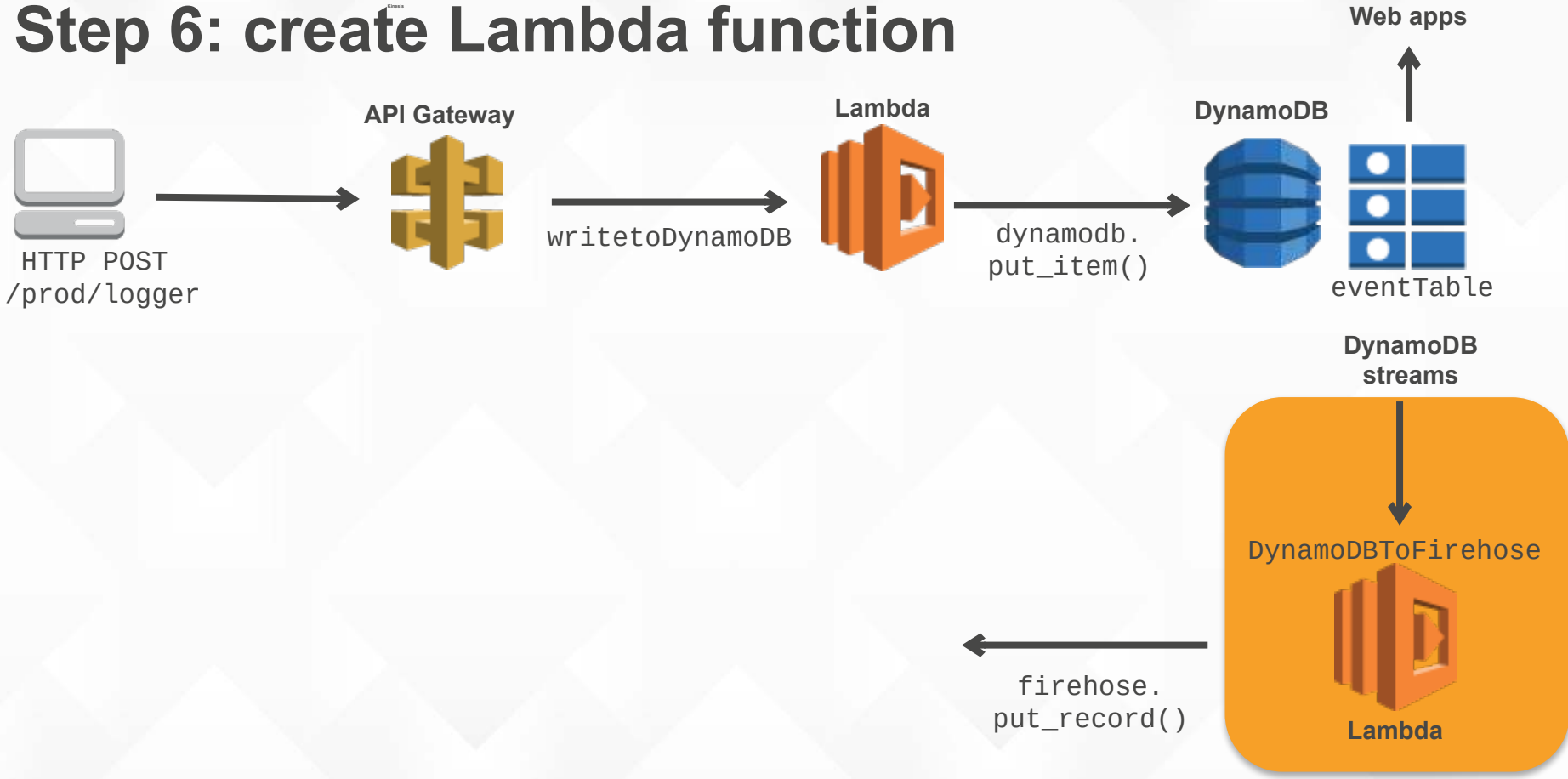
Step 5: create IAM role

```
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 6: create Lambda function

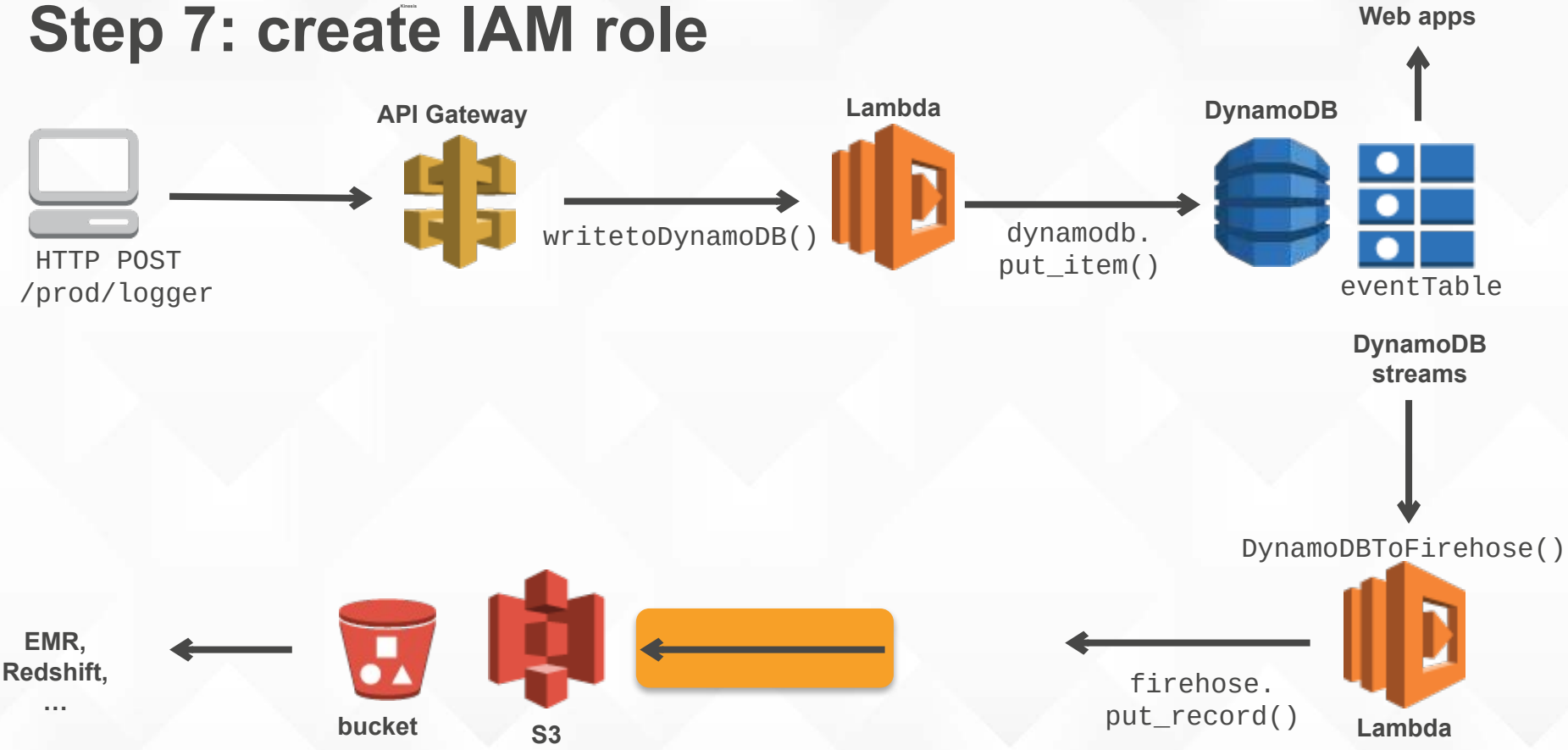


Step 6: create Lambda function

```
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 7: create IAM role



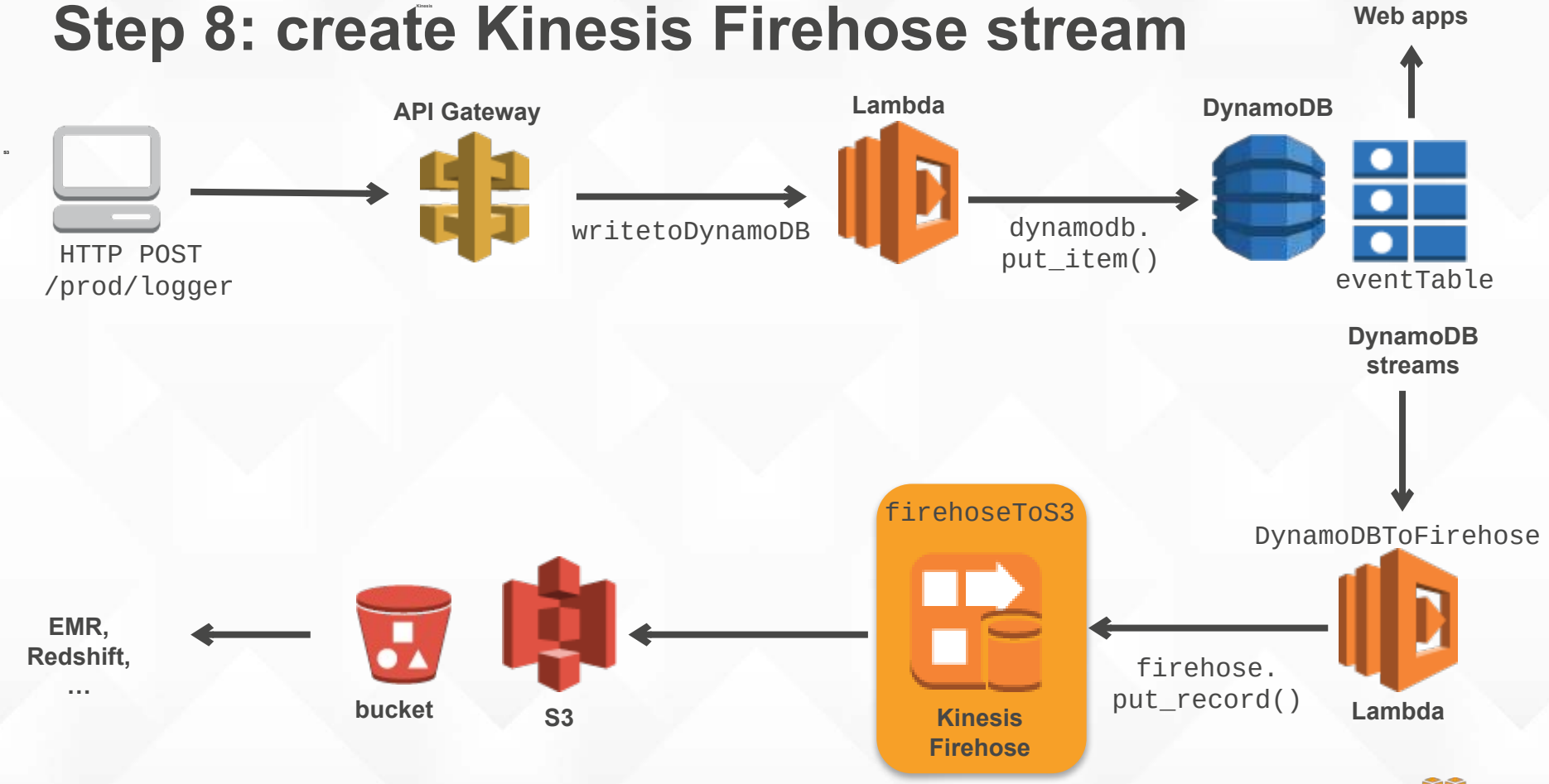
Step 7: create IAM role

```
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 8: create Kinesis Firehose stream



Step 8: create Kinesis Firehose stream

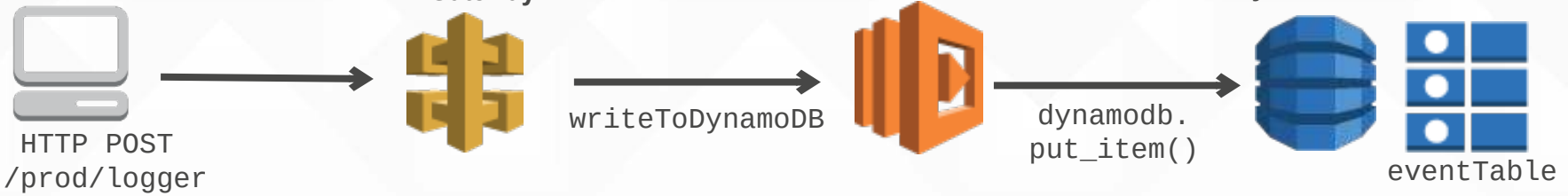
```
aws firehose create-delivery-stream \
--delivery-stream-name firehoseToS3 \
--s3-destination-configuration \
RoleARN=FIREHOSETOS3_ROLE_ARN, \
BucketARN="arn:aws:s3:::jsimon-public", \
Prefix="firehose", \
BufferingHints=\{SizeInMBs=1,IntervalInSeconds=60\}, \
CompressionFormat="GZIP", \
EncryptionConfiguration={NoEncryptionConfig="NoEncryption"}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

Time to test!

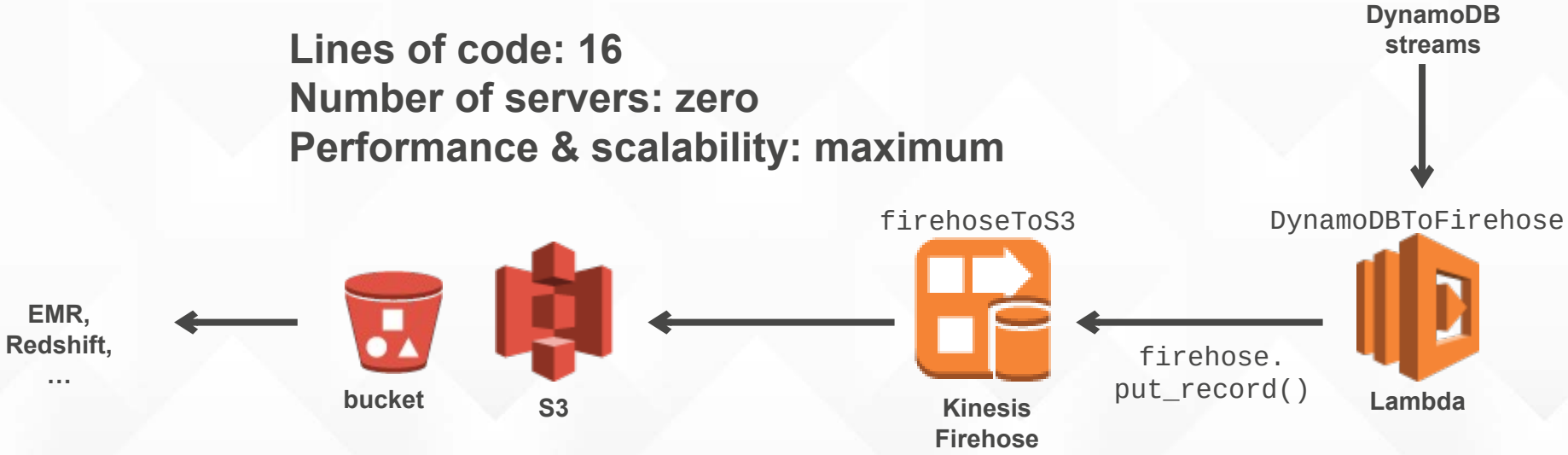
```
for i in {1..1000} \
do \
curl -H "Content-Type: application/json" \
-X POST -d "{\"userId\":$i, \"value\":$i}" \
https://API_ENDPOINT/prod/logger
done
```

```
aws s3 ls s3://jsimon-public/firehose2016/MM/DD/
```

AWS serverless demo

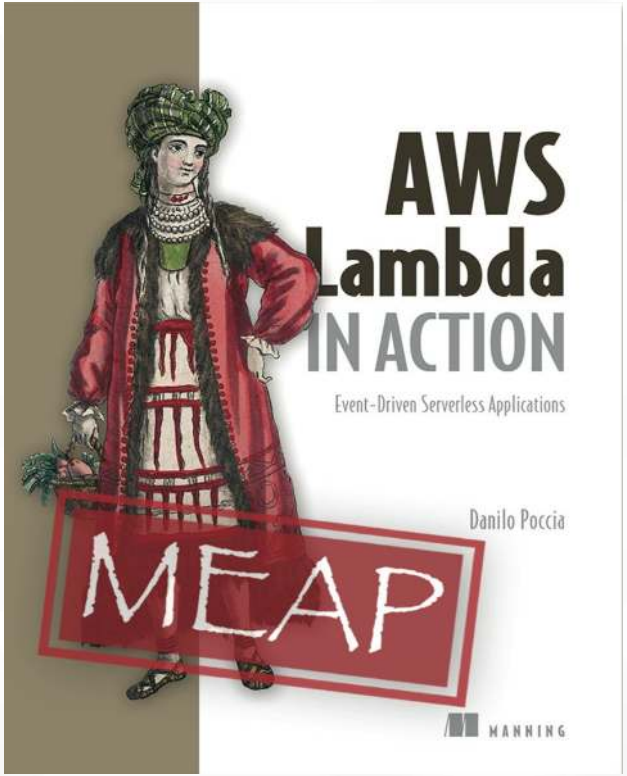


Lines of code: 16
Number of servers: zero
Performance & scalability: maximum



amazon web services

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

Merci !

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

amazon web services