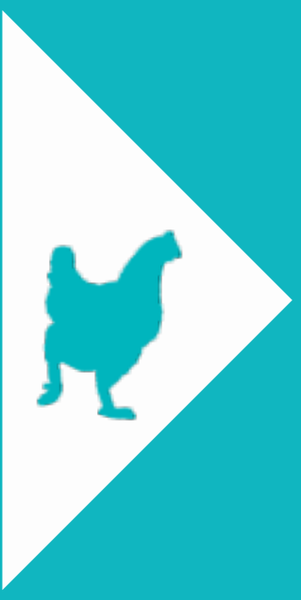


ZERO INFRASTRUCTURE*

Building Realtime* Data Pipelines with Python and AWS Lambda

**well, almost.*

scripps lifestyle network

Mercedes Coyle
@benzobot
Cloud Operations Engineer

WHAT I'LL COVER

- Realtime/Streaming Systems Architecture
- AWS Serverless Components:
  - Kinesis
  - Lambda
  - API Gateway
  - EMR

USE CASE

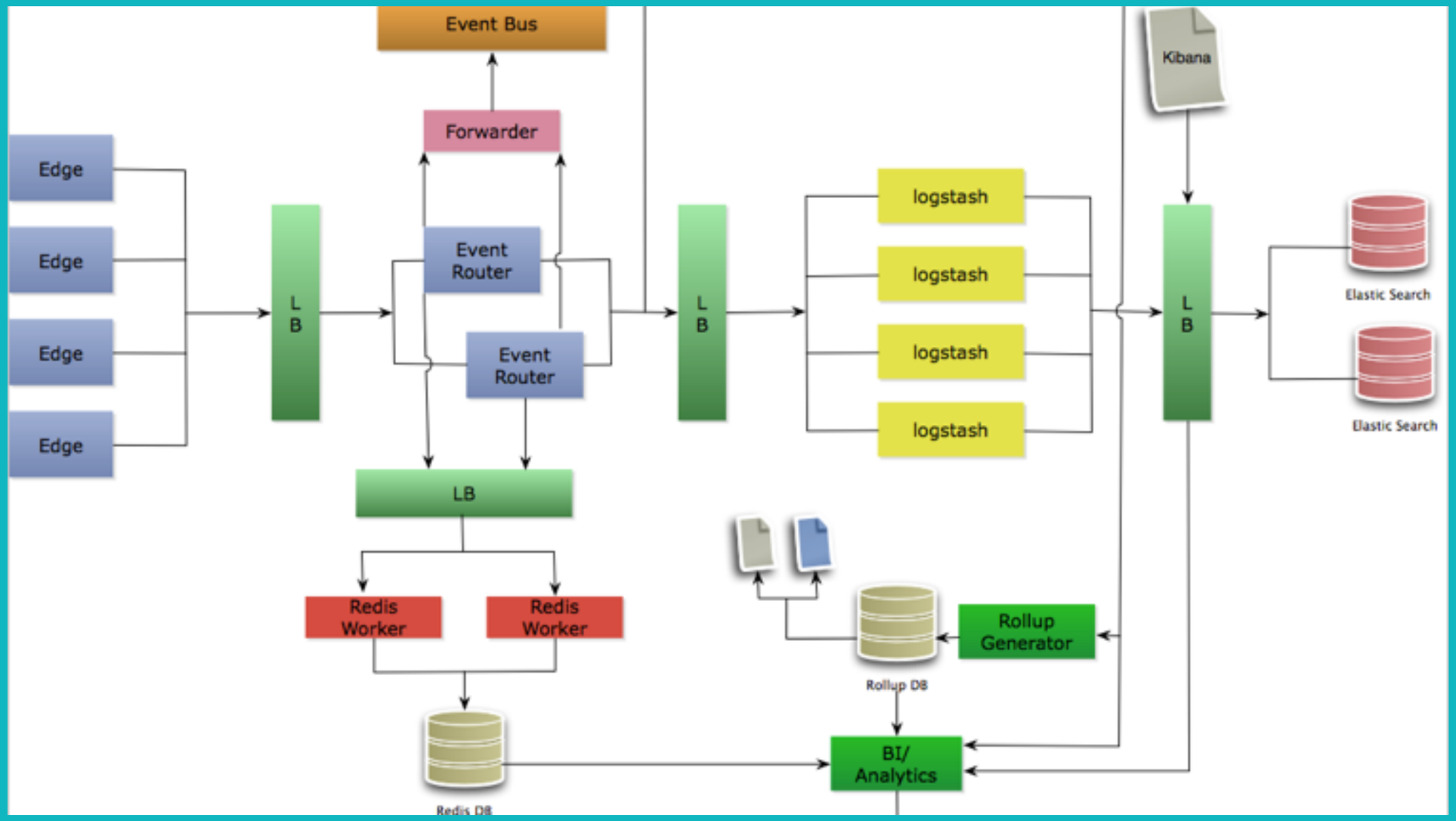
- Online video syndication platform
- Connects content providers, video publishers, and advertisers
- 2-3 million video streams per day

INTRO: REALTIME SYSTEMS

- What does “Realtime” mean?
  - Event based
  - Near realtime - up to several seconds between data origin and destination

LEGACY DATA SYSTEM

Architecture



LEGACY DATA SYSTEM

What we learned

- Need for faster data analysis
- Avoid logging to disk as a method of data collection
- Scheduled jobs are not intelligent
- Mangled data

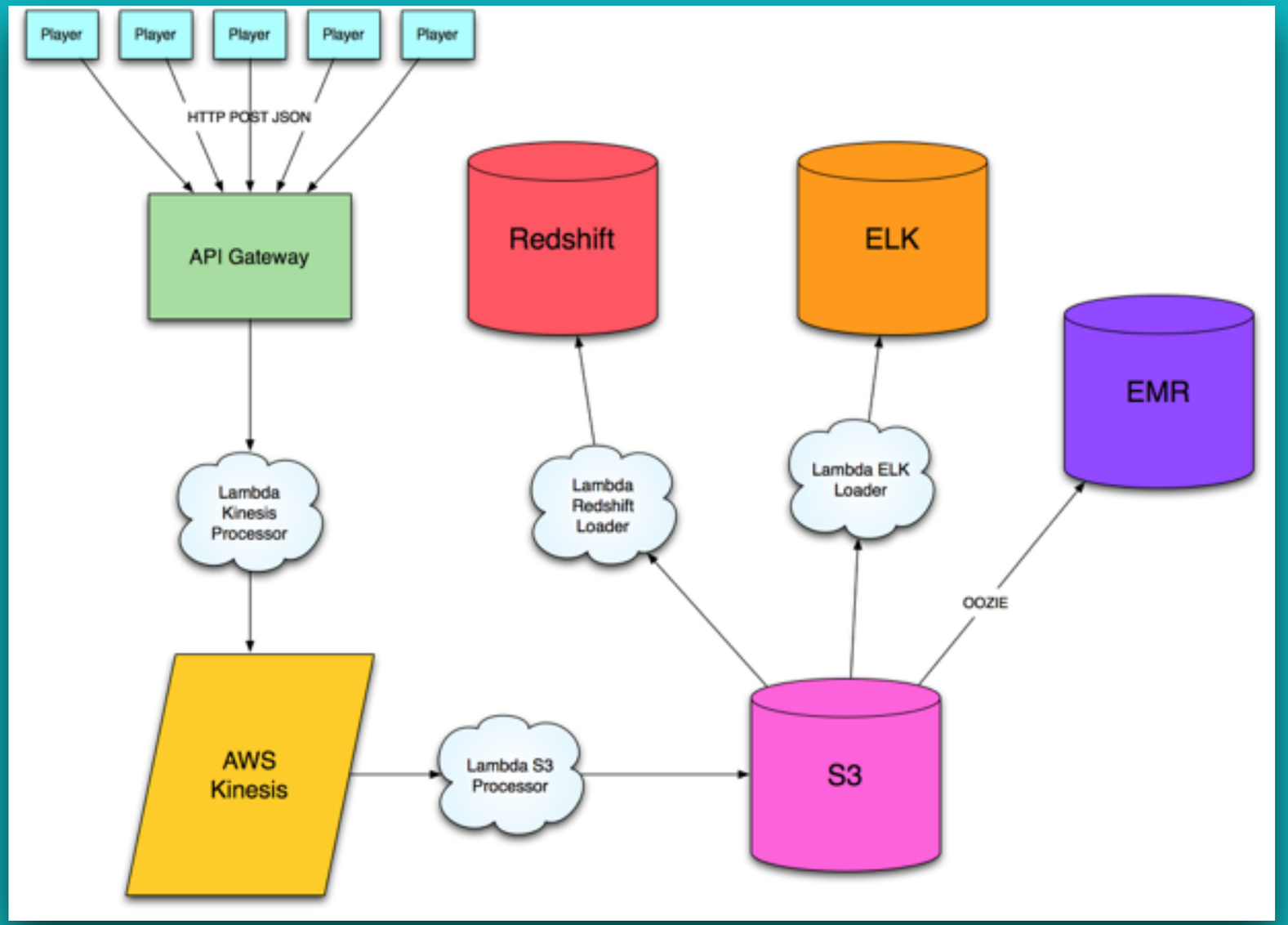
GOING SERVERLESS

System Requirements

- Allow for streaming analytics
- Reduce system complexity
- Data source and storage agnostic
- Flexibility

SERVERLESS DATA SYSTEM

Architecture



GOING SERVERLESS

AWS Services

- API Gateway
- Kinesis Streams
- Lambda
- S3
- EMR

API GATEWAY

- Quick and easy to setup
- Public HTTP interface or use API keys
- Can trigger lambda or go directly to Kinesis stream

KINESIS STREAMS

Queueing Service

- HTTP PUT single or batched records
- 7 day data retention
- Multiple subscriber
- Horizontally scalable

S3

Simple Storage Service

- Simple Storage Service
- Stores file objects, not a traditional file system
- Categorize file objects by buckets
- <bucket>/<year>/<month>/filename.bz2

EMR

Elastic Map Reduce

- Managed Hadoop cluster
- Spin up, process, destroy

AWS LAMBDA

Features

- Event driven push/pull
- Scales up/down automatically
- Supports Python, NodeJS, and Java
- Stateless and Asynchronous

LAMBDA: ANATOMY

The basics

```python
import boto3
import json
import logging

kinesis         = boto3.client('kinesis')
kinesis_stream = 'test-stream'
partition       = 'video_player'
logger          = logging.getLogger()

def event_handler(event, context):
    try:
        kinesis.put_record(StreamName=kinesis_stream,
                           Data=json.dumps(event),
                           PartitionKey=partition)
    except Exception, e:
        logger.error("Failed to add event to kinesis: error {}".format(e))
```

LAMBDA: ANATOMY

Event and Context Object

Event Data

```
"Records": [
  {
    "eventID": "shardId-000000000000:49545115243490985018280067714973144582180062593244200961",
    "eventVersion": "1.0",
    "kinesis": {
      "partitionKey": "partitionKey-3",
      "data": "SGVsbG8sIHRoaXMgaXMgYSB0ZXN0IDEyMy4=",
      "kinesisSchemaVersion": "1.0",
      "sequenceNumber": "49545115243490985018280067714973144582180062593244200961"
    },
    "invokeIdentityArn": identityarn,
    "eventName": "aws:kinesis:record",
    "eventSourceARN": eventsourcearn,
    "eventSource": "aws:kinesis",
    "awsRegion": "us-east-1"
  }
]
}
```

LAMBDA: ANATOMY

Event and Context Object

- The context object is metadata about the running function
  - context.get_remaining_time_in_millis()
  - context.aws_request_id()

LAMBDA: ANATOMY

Runtime

- Key design feature is statelessness
- Lambda functions don’t know anything about previous events
- Automatic retry on failure

LAMBDA: ANATOMY

Runtime

```
19  def bzip2_in_mem(data):
20      '''Transforms an array of strings into a newline separated, bz2 encoded string'''
21      compressed = bz2.compress("\n".join(data))
22      return compressed
23
24  def upload_s3(bucket, filename, data):
25      '''Put an s3 object in our s3 bucket'''
26      uploaded = s3.Object(bucket, filename).put(Body=data)
27      logger.info(uploaded)
28
29  def event_handler(event, context):
30      '''Handles all incoming events and logging'''
31      data_events    = []
32      file_name = "{}/{}/player_events.{}.{}.bz2".format(
33                      date.strftime("%Y"),
34                      date.strftime("%m"),
35                      date.strftime("%Y%m%d%H"),
36                      unique_id)
37      try:
38          for record in event['Records']:
39              decoded_event=base64.b64decode(record["kinesis"]["data"])
40              data_events.append(decoded_event)
41
42          upload_s3(bucket, file_name, bzip2_in_mem(data_events))
43
44          status = 'Processed {} records.'.format(len(event['Records']))
45          logger.info(status)
46
47      except Exception, e:
48          status = "Lambda function returned error {}".format(e)
49          logger.error(status)
```

LAMBDA: ANATOMY

Logging and Monitoring

- Any print or logging statement is logged to CloudWatch

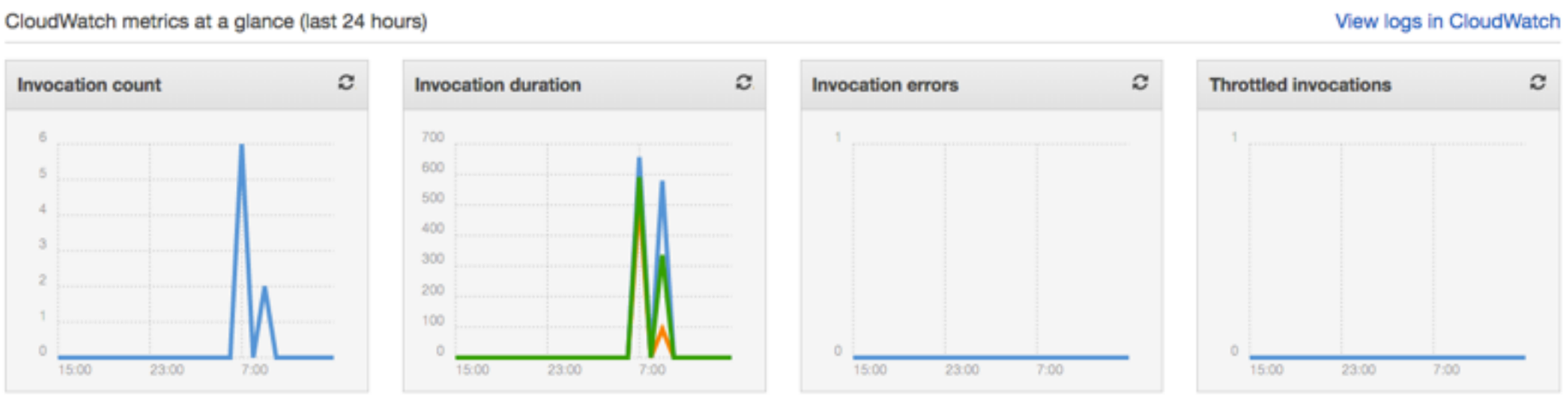
Event Data

```
START RequestId: 1fcd84d7-221c-11e6-8818-5172a2bd2595 Version: $LATEST
[INFO] 2016-05-25T01:58:07.960Z 1fcd84d7-221c-11e6-8818-5172a2bd2595 Calling s3:put_object with {'Body…
[INFO] 2016-05-25T01:58:08.22Z 1fcd84d7-221c-11e6-8818-5172a2bd2595 Starting new HTTPS connection (1):
s3.amazonaws.com
[INFO] 2016-05-25T01:58:08.231Z 1fcd84d7-221c-11e6-8818-5172a2bd2595 {u'ETag':
'"943d3db6675d4a54b7d25eea6296dfff"', 'ResponseMetadata': {'HTTPStatusCode': 200, 'HostId':
'r5o0wIKiPm5PVS3BQ7NziHa2mFk0DbxX/KvrIwdAH99FKJv9L1SzgDdKBdPWezSBKgra33vpRsU=', 'RequestId':
'9AD11EE7CD1F6DFB'}}
[INFO] 2016-05-25T01:58:08.231Z 1fcd84d7-221c-11e6-8818-5172a2bd2595 Processed 1 records.
[INFO] 2016-05-25T01:58:08.232Z 1fcd84d7-221c-11e6-8818-5172a2bd2595 Remaining execution time:
119691ms
[INFO] 2016-05-25T01:58:08.232Z 1fcd84d7-221c-11e6-8818-5172a2bd2595 Request ID: 1fcd84d7-221c-11e6-
8818-5172a2bd2595
[INFO] 2016-05-25T01:58:08.232Z 1fcd84d7-221c-11e6-8818-5172a2bd2595 Function version: $LATEST
END RequestId: 1fcd84d7-221c-11e6-8818-5172a2bd2595
REPORT RequestId: 1fcd84d7-221c-11e6-8818-5172a2bd2595 Duration: 308.56 ms Billed Duration: 400 ms
Memory Size: 128 MB Max Memory Used: 28 MB
```

LAMBDA: ANATOMY

Logging and Monitoring

- Metrics dashboard displays high level performance data



LAMBDA: TESTING

- Can test lambda code as any other python code with your preferred testing framework
- Invoke lambda functions manually from AWS CLI
  - aws lambda invoke –invocation-type DryRun –function-name put-events-kinesis –payload '{"test":"data"}' outfile

LAMBDA: PACKAGING

- Need to create a zip file of function code and any dependencies
- Can pip install <module> -t /project-dir/ and zip contents of that directory
- Or you can install the contents of <virtualenv>/lib/python2.7/site-packages/

LAMBDA: DEPLOYMENT

- AWS CLI from Travis CI job
- Cloud Formation template
- Upload to S3 and deploy via Lambda

SUMMARY

Takeaways and Lessons Learned

- Python 2.7 only
- Faster development cycles and data insights
- Code more for business goals, less for infrastructure
- Factor in maintenance and operational costs when pricing out

SERVERLESS*

**Not all servers*

RESOURCES

- https://aws.amazon.com/blogs/compute/microservices-without-the-servers/
- http://redmonk.com/fryan/2016/04/28/serverless-volume-compute-for-a-new-generation/
- http://blogs.aws.amazon.com/bigdata/post/Tx2Z24D4T99AN35/Snakes-in-the-Stream-Feeding-and-Eating-Amazon-Kinesis-Streams-with-Python
- https://docs.aws.amazon.com/lambda/latest/dg/intro-core-components.html
- https://docs.aws.amazon.com/lambda/latest/dg/limits.html
- https://github.com/spulec/moto

THANK YOU!

Mercedes Coyle
@benzobot