Putting Node.js Serverless Apps into Production without the Pitfalls

Eóin Shanaghy

@eoins

fourTheorem

I’m going to talk about

- Modern Applications
- JavaScript and Serverless
- How to Build Them
- Pitfalls and how we overcome them
- Serverless
- Recipes for Effective Serverless with JS

@eoins

The Modern Application

Scalable

User-focused

Reliable

Intelligent

Fast to market

Experimental!

@eoins

The best way to build a modern application

@eoins

Build it with Wordpress!

Build it with Haskell!

Build it with Rails!

@eoins

Build it with Erlang!

@eoins

Build it with Java

Microservices

on Kubernetes

Build it with Serverless

using JavaScript

Why would you do this?!!

THE PURSUIT OF PERFECTION

REALITY

@eoins

There are always trade-offs

The pursuit of perfection

David Redfern/Redferns/Getty Images

KEITH JARRETT
THE KÖLN CONCERT

ECM

CLOUD

MICROSERVICES

IaC

INFRASTRUCTURE COMPLEXITY

FUNCTIONS AS A SERVICE

MANAGED SERVICES

@eoins

Serverless

#1 - Managed Services (incl. FaaS)

#2 - Event-driven

#3 – Pay only for what you use

#4 – No idling infrastructure

#5 – Less code

JavaScript

The Success of Node.js

Developer Productivity

Event-Driven I/O

Single thread

Modules

Easy to: Comprehend | Find | Create | Ditch

@eoins

Is JS a Good Fit for Serverless?

Maybe not…

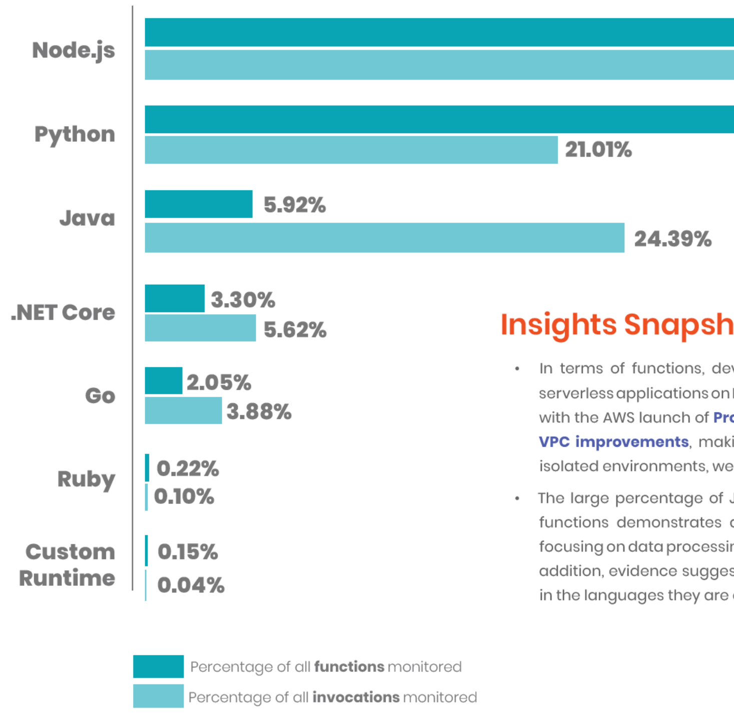
- One process per request
- No HTTP server
- Lack of types

But actually…

- Fast to start
- Fast runtime
- No compilation overhead
- Huge module ecosystem
- Familiarity, ubiquity
- Still highly productive!
- JSON

50-60% of Lambda Functions

Lambda Adoption by Runtime



Time frame July-December 2019

Insights Snapshot

- In terms of functions, developers mostly rely on Node.js and Python for building serverless applications on Lambda, with Java as the third most-used runtime. However, with the AWS launch of **Provisioned Concurrency** mitigating cold start impacts and **VPC improvements**, making Lambda more attractive for enterprises that require isolated environments, we expect the adoption trends for Java to increase in 2020.
- The large percentage of Java invocations for a relatively small number of related functions demonstrates a higher throughput based on organizations potentially focusing on data processing use cases in which cold starts could be less of an issue. In addition, evidence suggests that companies could be lift-and-shifting applications in the languages they are currently using.

New Relic, For the Love of Serverless 2020 Report

@eoins

Should We Choose JavaScript?

- It matters less than ever before
- Start with what you know
- Experiment
- Solve the problems you observe

handler.js

```javascript
const {findAccommodation} = require('./lib/accommodation');

async function lookup({queryStringParameters: {county}}) {
  const result = await findAccommodation(county);
  return {
    statusCode: 200,
    body: JSON.stringify(result),
  };
}

module.exports = {lookup};
```

@eoins

lib/accommodation.js

```javascript
const AWS = require('aws-sdk');
const s3 = new AWS.S3();

const SELECT_PARAMS = {
  Bucket: process.env.BUCKET_NAME, Key: process.env.CSV_FILE, ExpressionType: 'SQL',
  InputSerialization: {CSV: {FileHeaderInfo: 'USE', RecordDelimiter: '\r\n'}},
  OutputSerialization: {JSON: {RecordDelimiter: '\n'}},
};

async function findAccommodation(county) {
  const response = await s3.selectObjectContent({
      ...SELECT_PARAMS,
      Expression: `SELECT * FROM S3Object s WHERE s.AddressRegion = '${county}'`,
    }).promise();

  let result = '';
  for await (const event of response.Payload) {
    if (event.Records) {
      result += event.Records['Payload'].toString();
    }
  }
  return result.trim().split('\n').map(JSON.parse);
}

module.exports = {findAccommodation};
```

@eoins

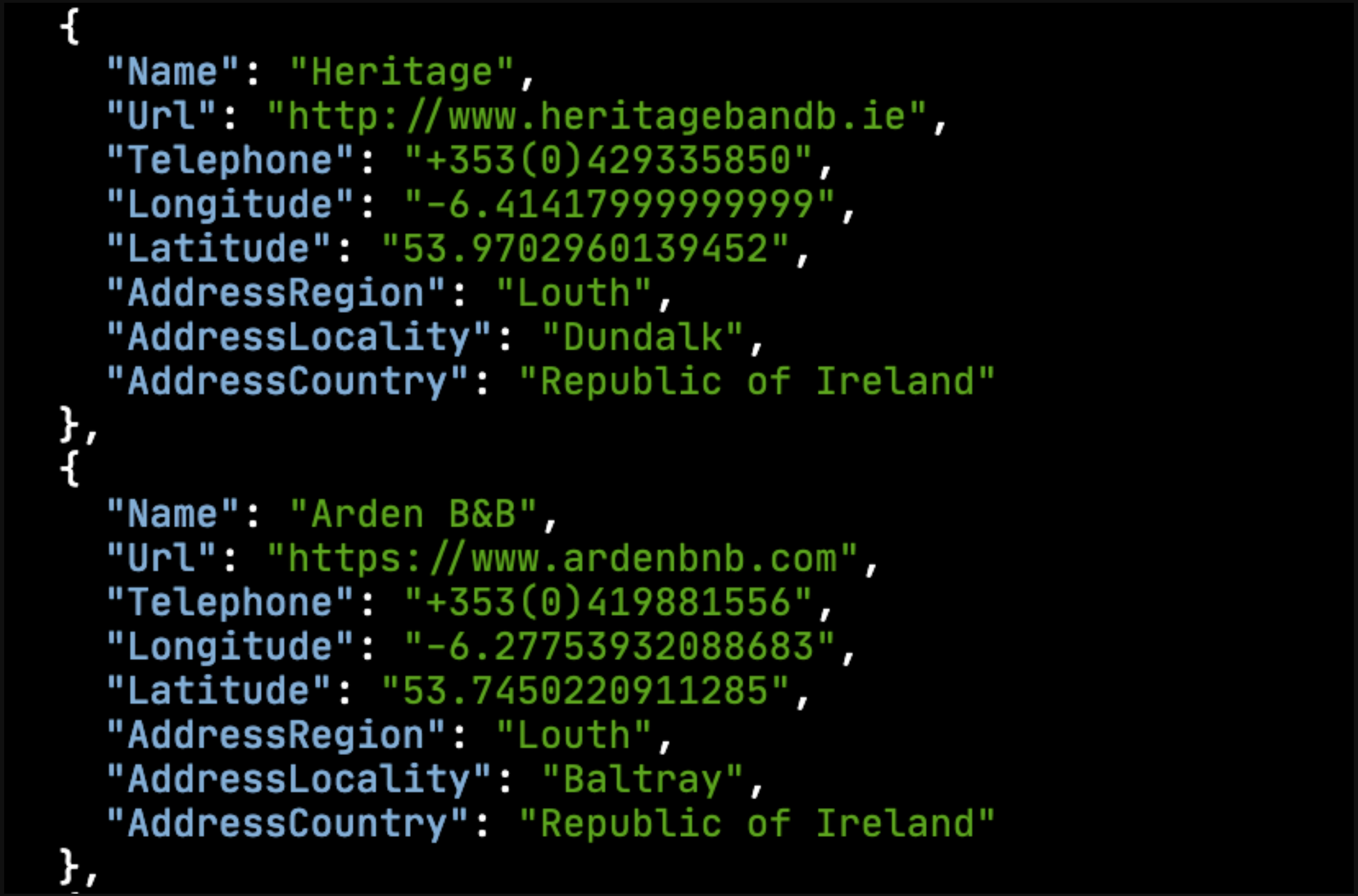
serverless.yml

```yaml
provider:
  name: aws
  runtime: nodejs12.x
  stage: dev
  region: eu-west-1
  endpointType: REGIONAL
  tracing:
    apiGateway: true
    lambda: true
  logs:
    restApi: true
  logRetentionInDays: 7
  iamRoleStatements:
    - Effect: Allow
      Action:
        - s3:GetObject
        - s3:HeadObject
      Resource:
        - arn:aws:s3:::fourtheorem-jsshow/accommodation.csv

functions:
  lookup:
    environment:
      BUCKET_NAME: fourtheorem-jsshow
      CSV_FILE: accommodation.csv
    handler: handler.lookup
    events:
      - http:
          path: accomodation
          method: get
          cors: false
```

@eoins

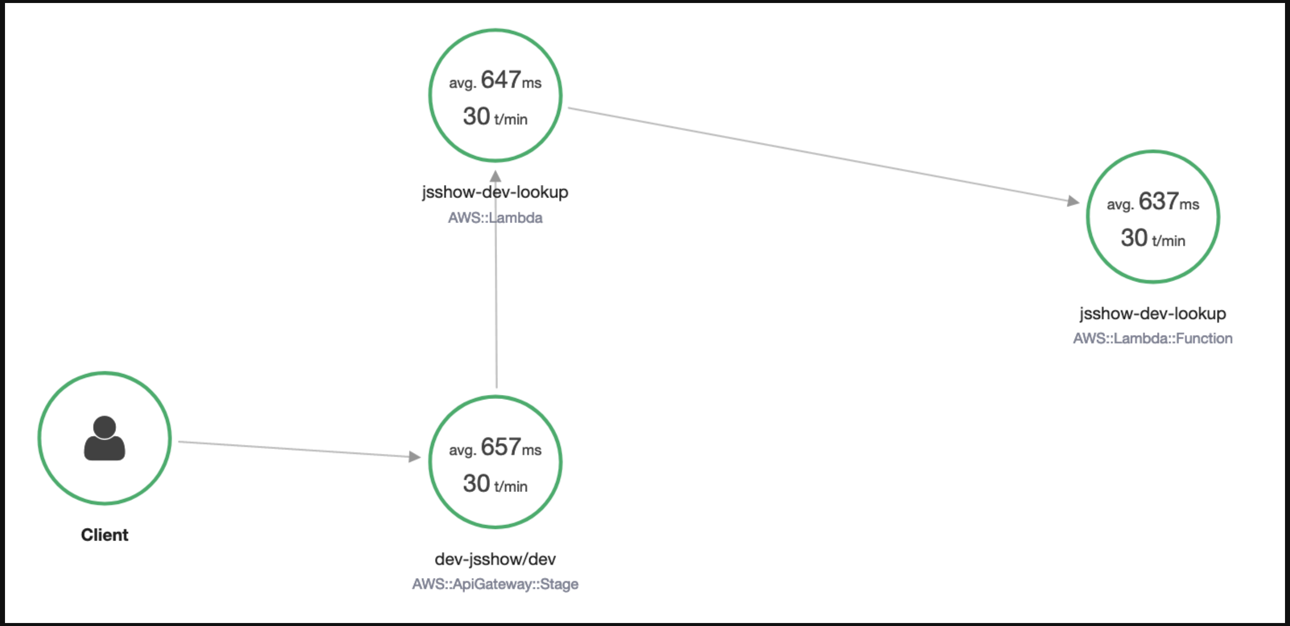
```
curl https://8dmtx7a123.execute-api.eu-west-
1.amazonaws.com/dev/accomodation\?county\=Louth
```

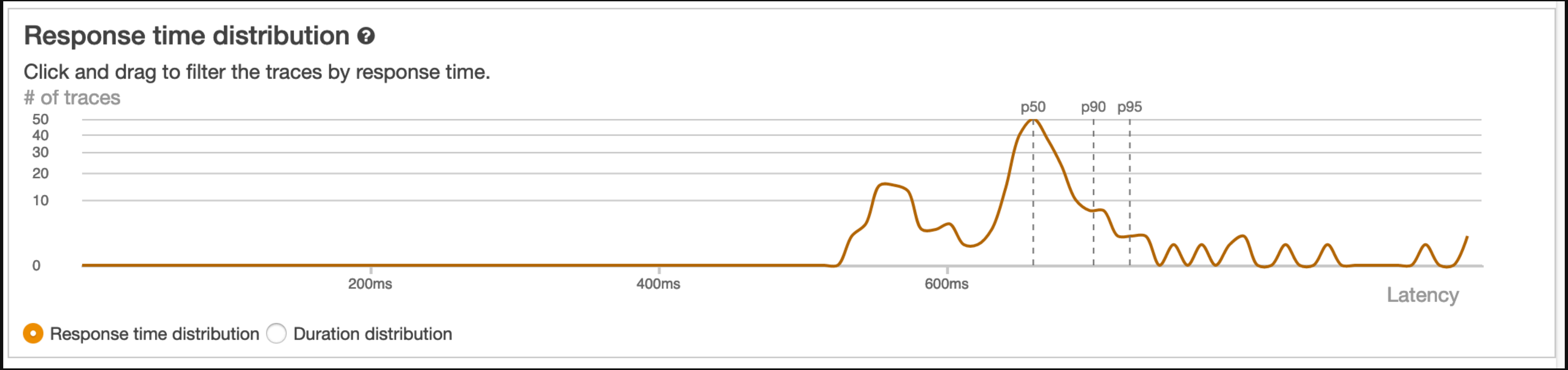


```
{
  "Name": "Heritage",
  "Url": "http://www.heritagebandb.ie",
  "Telephone": "+353(0)429335850",
  "Longitude": "-6.41417999999999",
  "Latitude": "53.9702960139452",
  "AddressRegion": "Louth",
  "AddressLocality": "Dundalk",
  "AddressCountry": "Republic of Ireland"
},
{
  "Name": "Arden B&B",
  "Url": "https://www.ardenbnb.com",
  "Telephone": "+353(0)419881556",
  "Longitude": "-6.27753932088683",
  "Latitude": "53.7450220911285",
  "AddressRegion": "Louth",
  "AddressLocality": "Baltray",
  "AddressCountry": "Republic of Ireland"
},
```

@eoins

```
requestId: d791228a-687f-45ed-
918c-3165a7c20f5c, ip:
91.123.228.33, caller: -, user: -
, requestTime:
29/Feb/2020:14:23:14 +0000,
httpMethod: GET, resourcePath:
/accomodation, status: 200,
protocol: HTTP/1.1,
responseLength: 105418
```





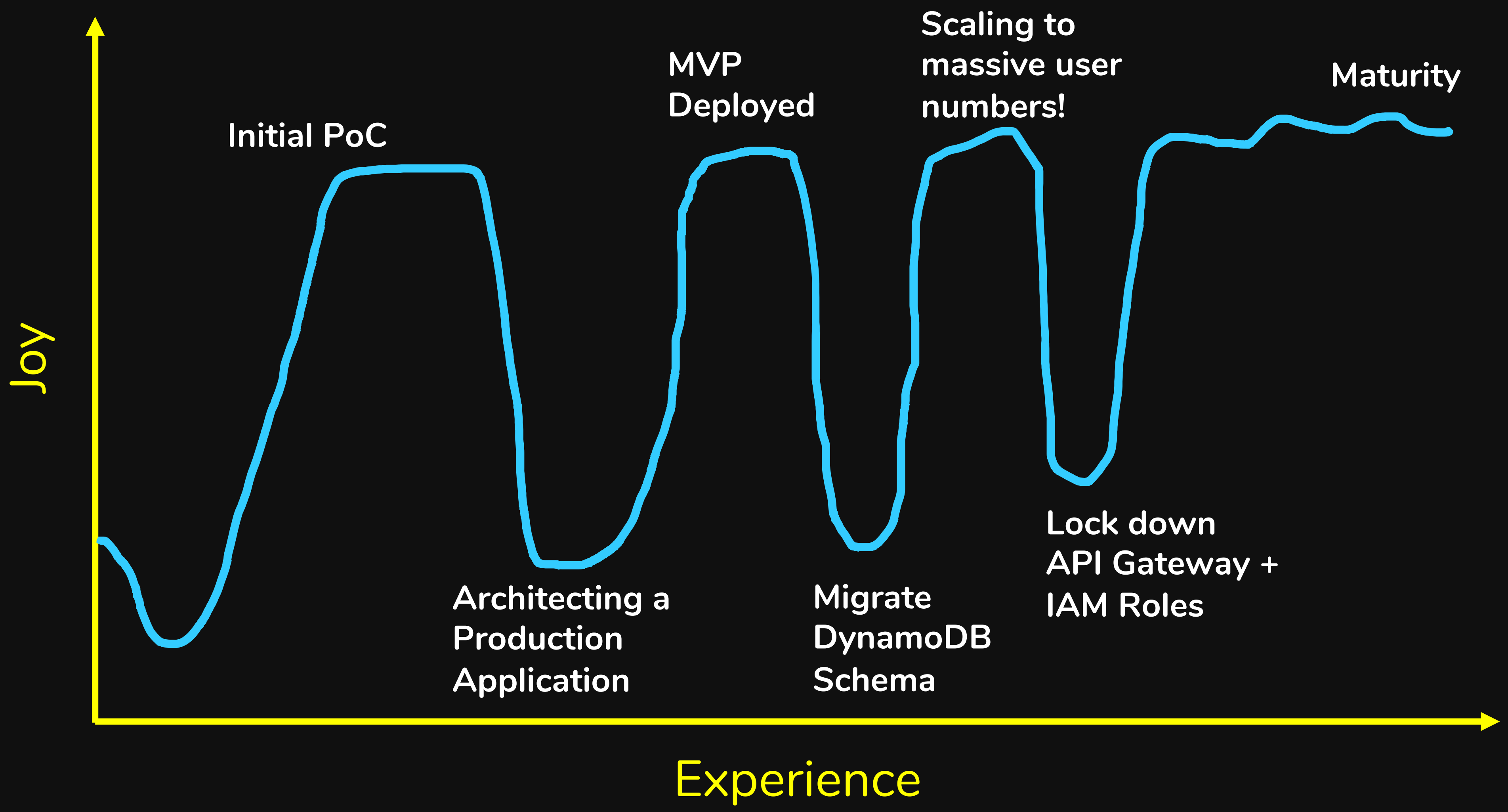
@eoins

Challenges

Learning Curve	Best Practices?	Moving Target
New Models	Organisational Change	Migration is Hard

@eoins

Serverless Adoption Rollercoaster

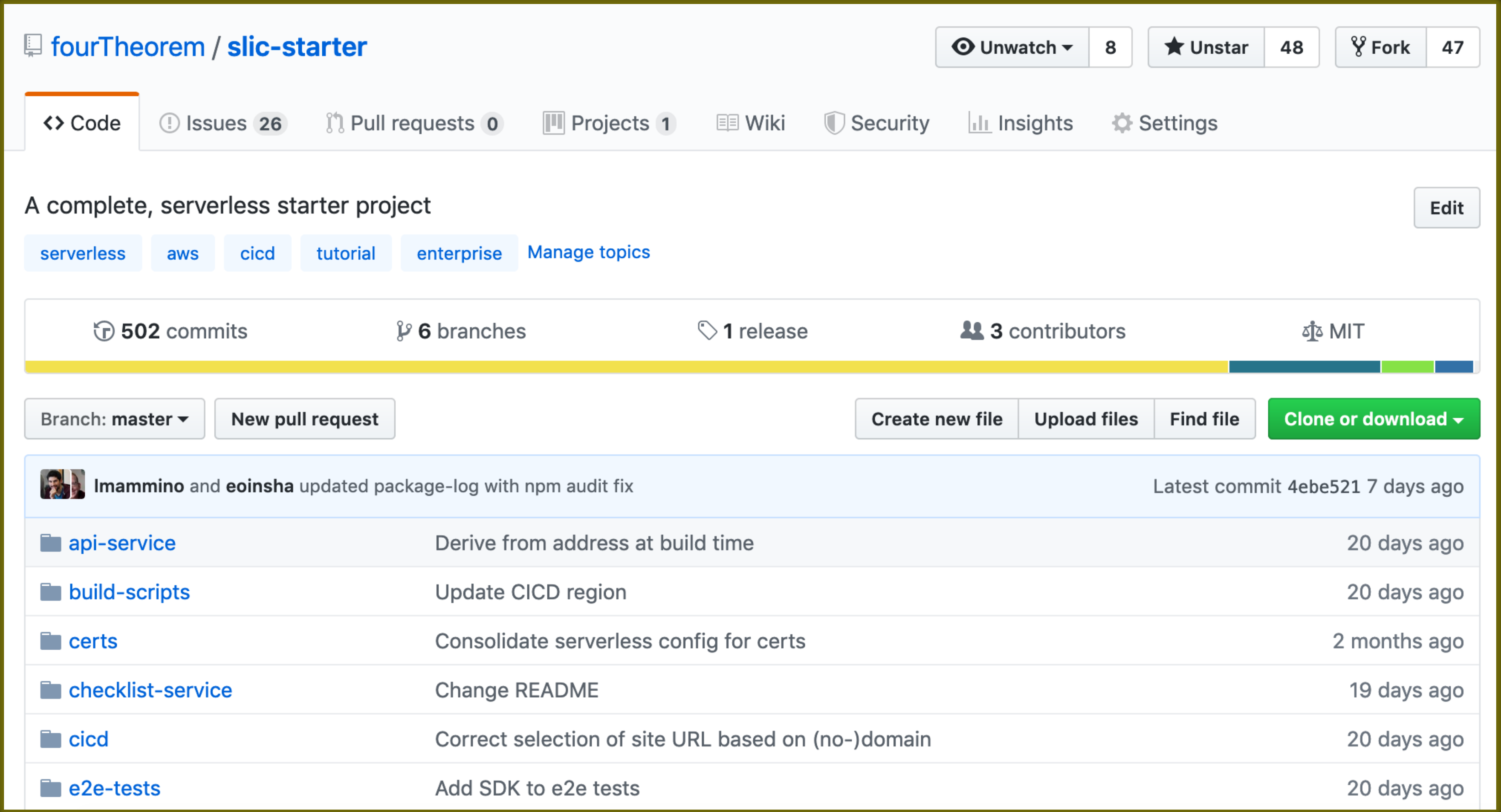


@eoins

1. Put all best practices together
2. Make opinionated decisions
3. Replicate production environment
4. Make it open source

Project Structure	CI/CD	Logging	Security
Observability	Local Development	Integration Testing	End-to-End Testing
HTTPS Certificates	Domains	Architecture	Events / Messaging
Service 'Discovery'	User Accounts	Front End	Data Access

@eoins



fourTheorem / slic-starter

Unwatch 8 | Unstar 48 | Fork 47

Code | Issues 26 | Pull requests 0 | Projects 1 | Wiki | Security | Insights | Settings

A complete, serverless starter project

serverless aws cicd tutorial enterprise Manage topics

502 commits | 6 branches | 1 release | 3 contributors | MIT

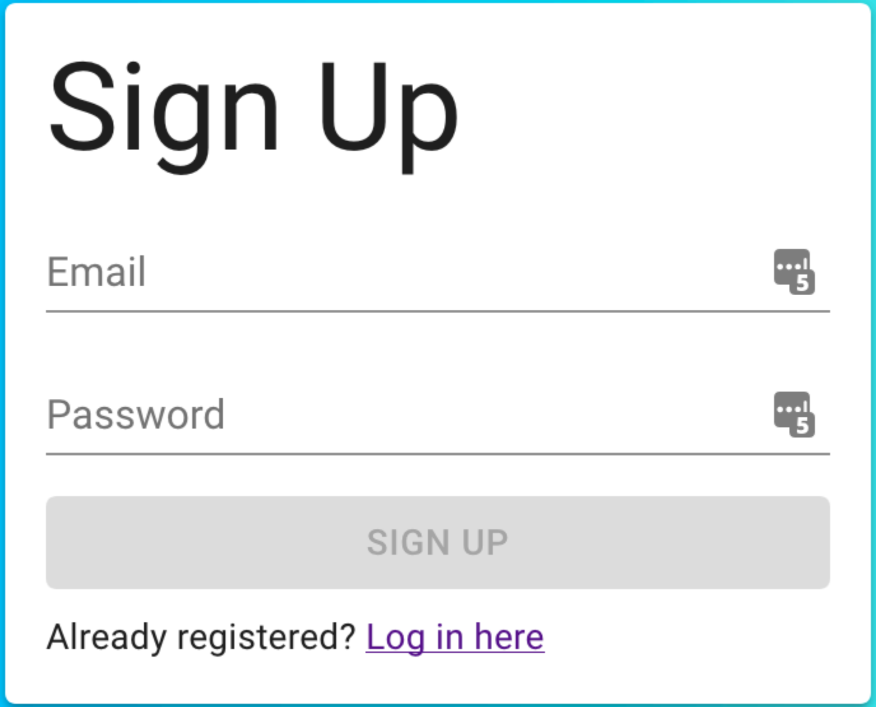
Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

Imammino and eoinsha updated package-log with npm audit fix — Latest commit 4ebe521 7 days ago

api-service	Derive from address at build time	20 days ago
build-scripts	Update CICD region	20 days ago
certs	Consolidate serverless config for certs	2 months ago
checklist-service	Change README	19 days ago
cicd	Correct selection of site URL based on (no-)domain	20 days ago
e2e-tests	Add SDK to e2e tests	20 days ago

slic.app

@eoins



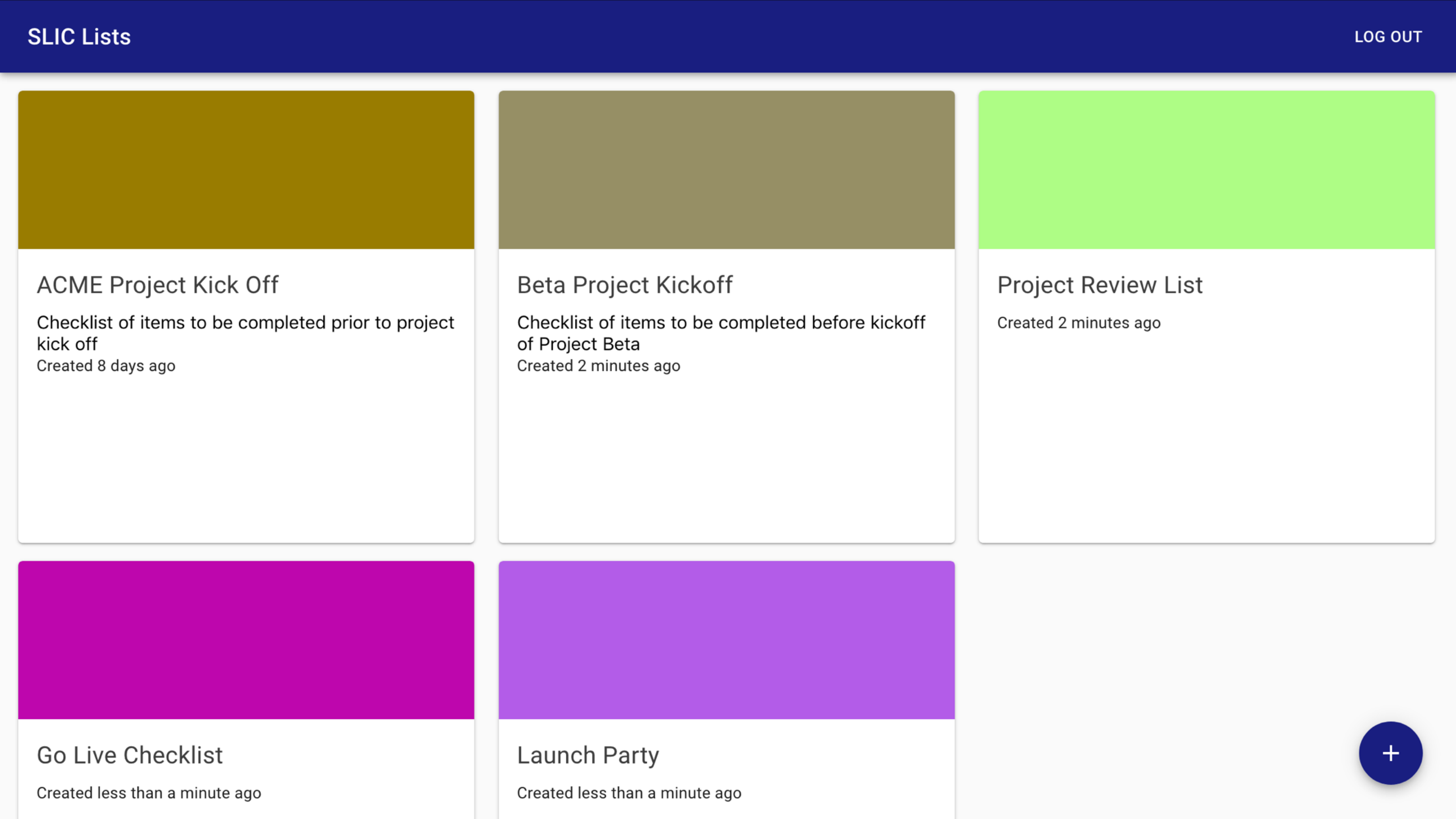
Sign Up

Email

Password

SIGN UP

Already registered? Log in here



ACME Project Kick Off

Checklist of items to be completed prior to project kick off
Created 8 days ago

Beta Project Kickoff

Checklist of items to be completed before kickoff of Project Beta
Created 2 minutes ago

Project Review List

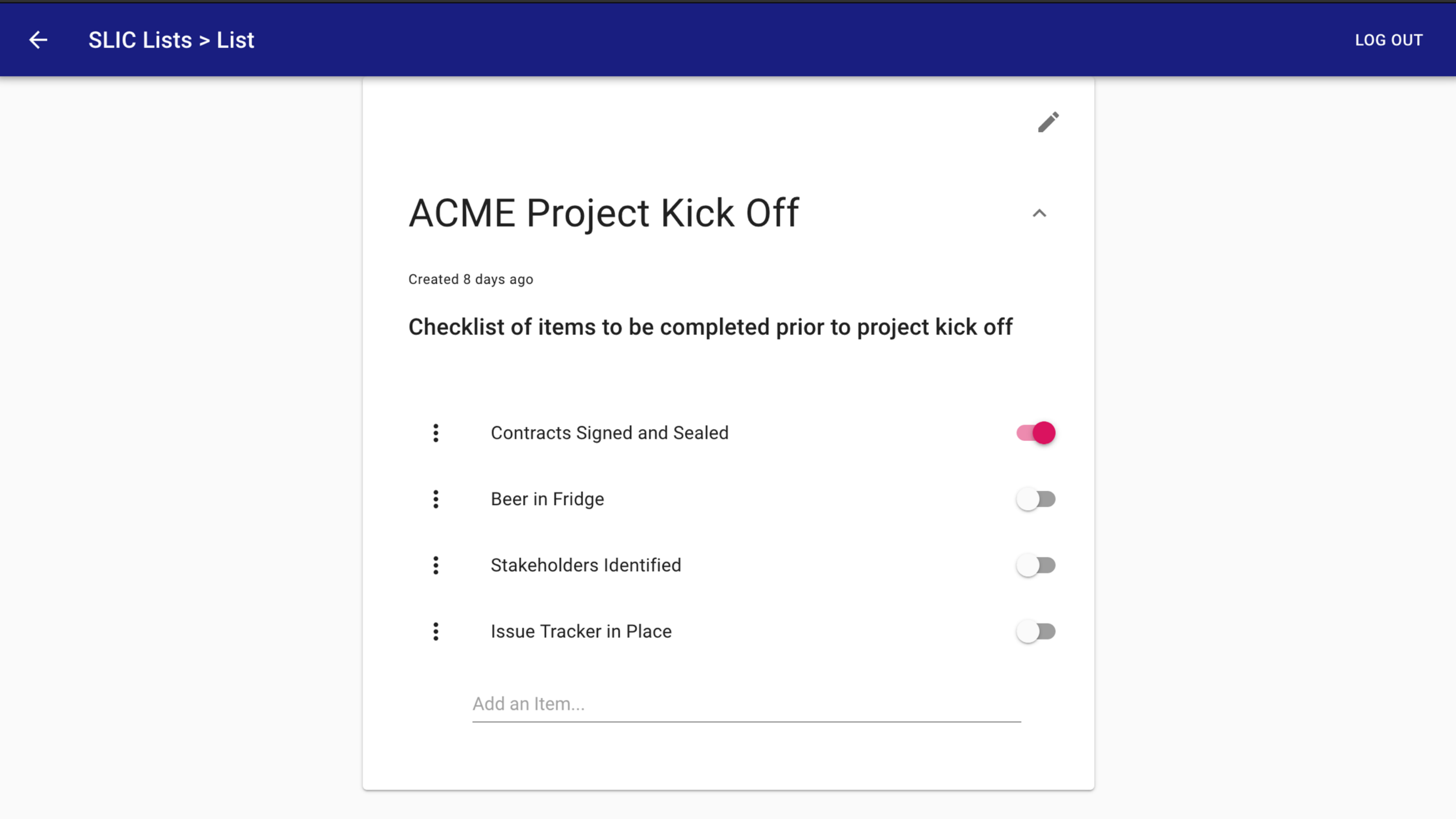
Created 2 minutes ago

Go Live Checklist

Created less than a minute ago

Launch Party

Created less than a minute ago


SLIC Lists > List

LOG OUT

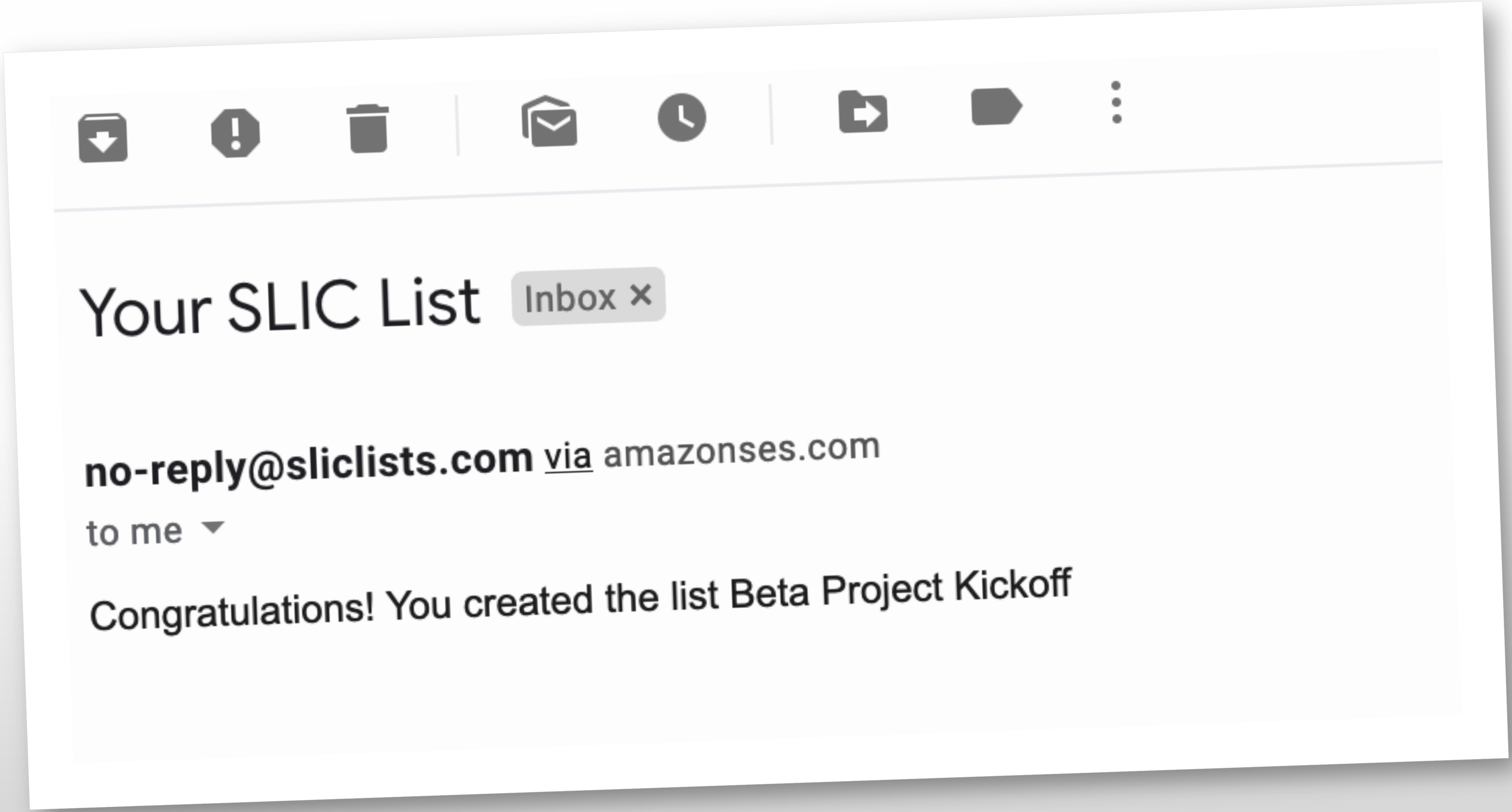
ACME Project Kick Off

Created 8 days ago

Checklist of items to be completed prior to project kick off

- Contracts Signed and Sealed ✓
- Beer in Fridge
- Stakeholders Identified
- Issue Tracker in Place

Add an Item...


Your SLIC List Inbox ×

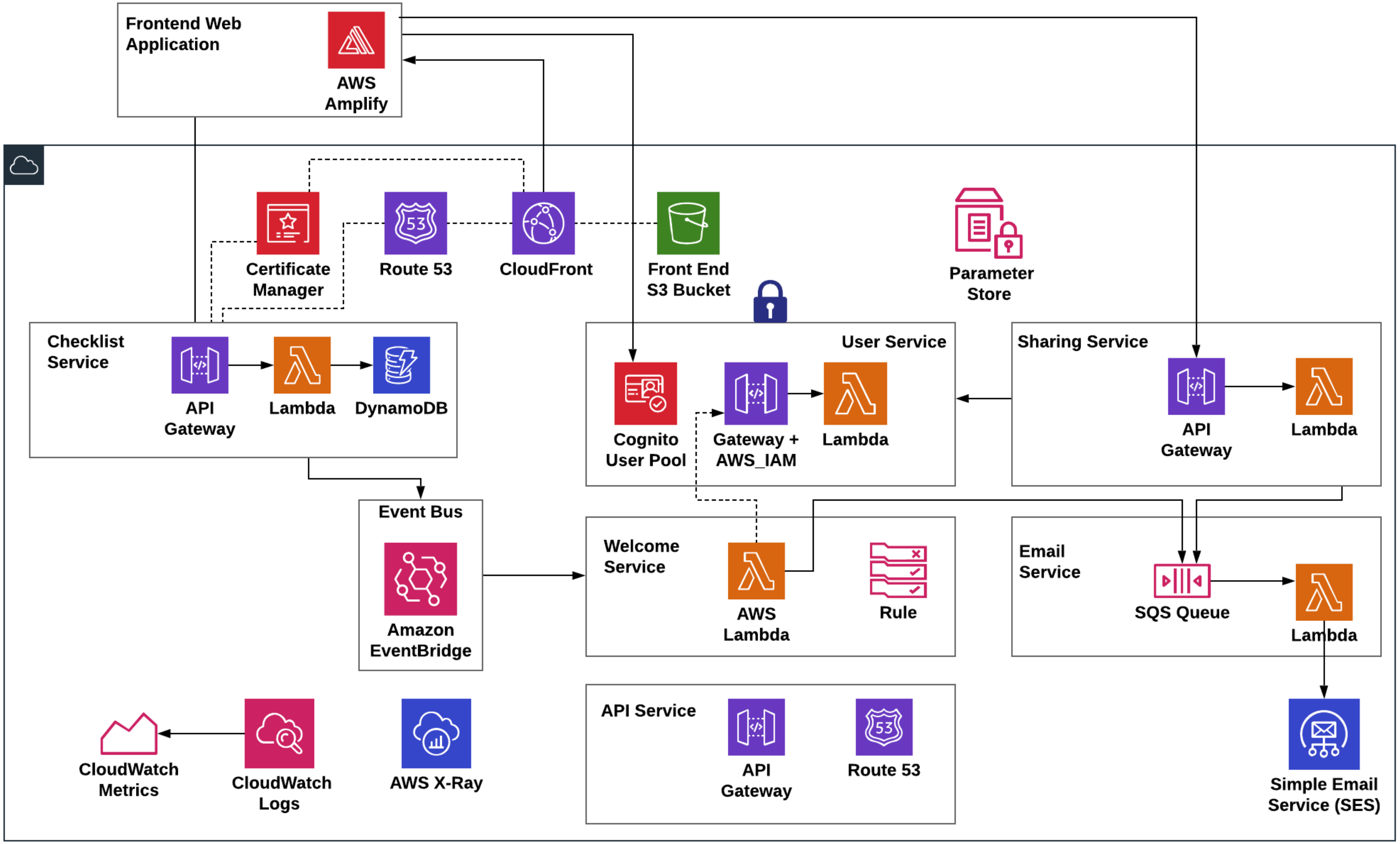
no-reply@sliclists.com via amazonses.com
to me

Congratulations! You created the list Beta Project Kickoff

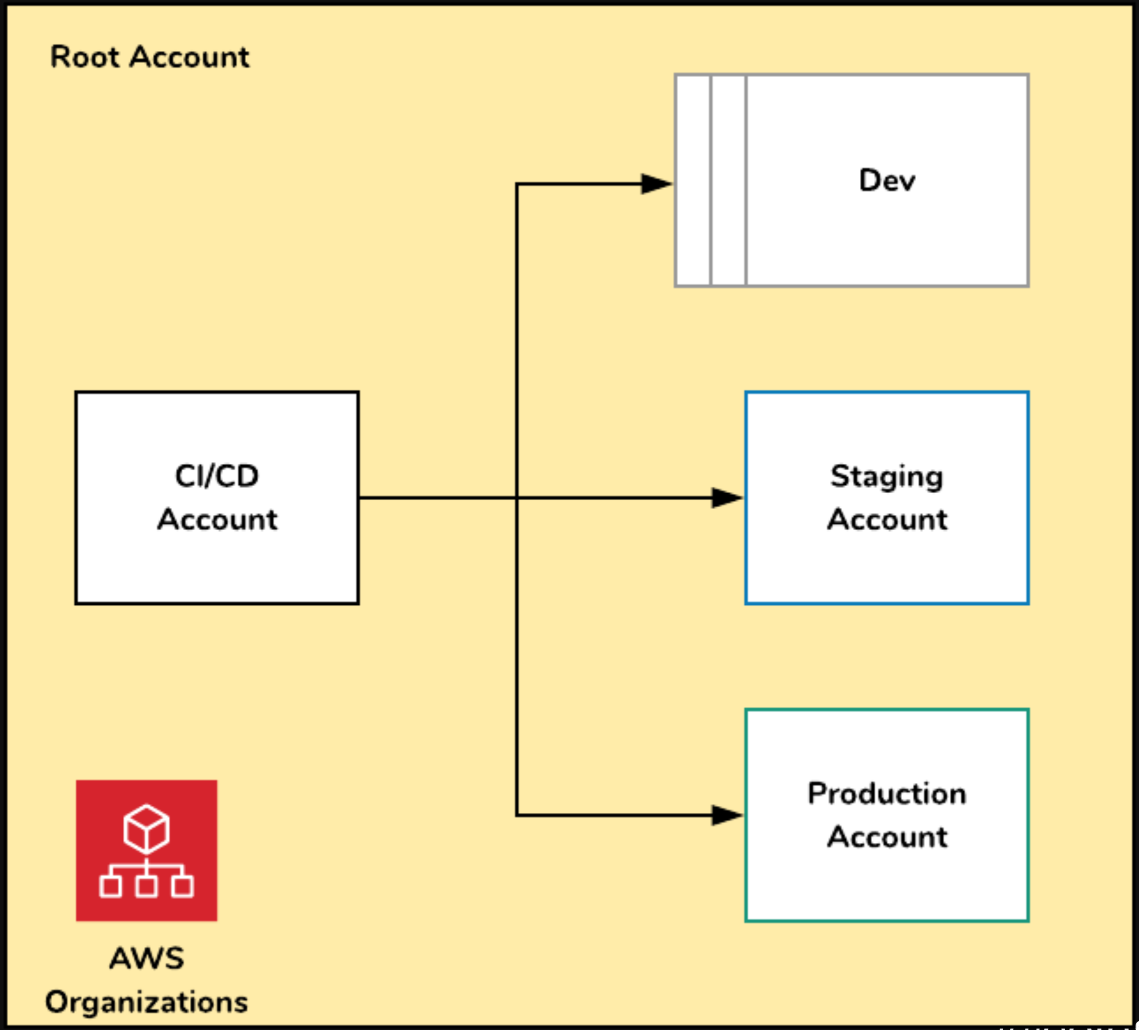
Targets AWS

aws

@eoins


Frontend Web Application
AWS Amplify
Certificate Manager
Route 53
CloudFront
Front End S3 Bucket
Parameter Store
Checklist Service
API Gateway
Lambda
DynamoDB
User Service
Cognito User Pool
Gateway + AWS_IAM
Lambda
Sharing Service
API Gateway
Lambda
Event Bus
Amazon EventBridge
Welcome Service
AWS Lambda
Rule
Email Service
SQS Queue
Lambda
API Service
API Gateway
Route 53
CloudWatch Metrics
CloudWatch Logs
AWS X-Ray
Simple Email Service (SES)

Separate Accounts



@eoins

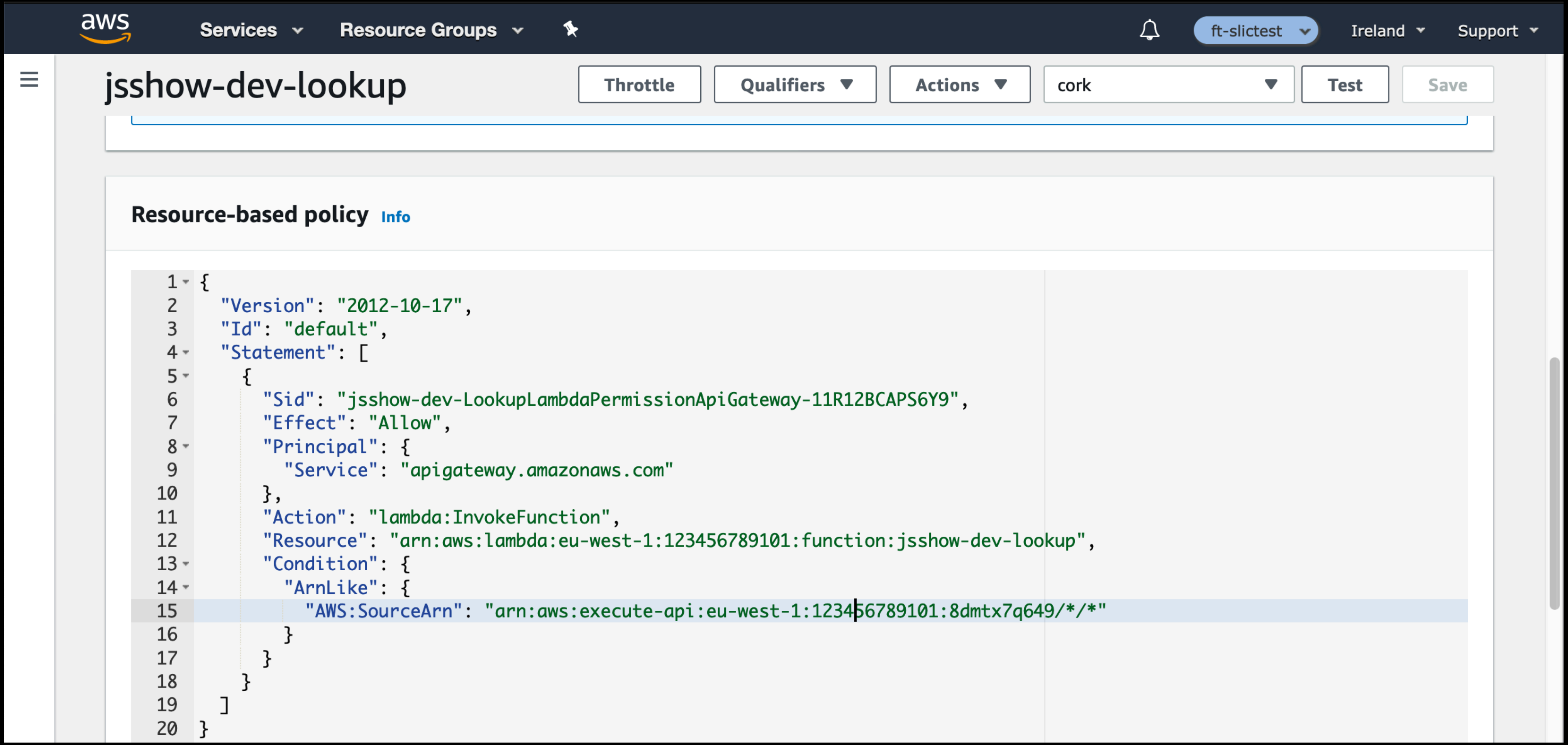
Serverless Framework

serverless

CDK

@eoins

Infrastructure



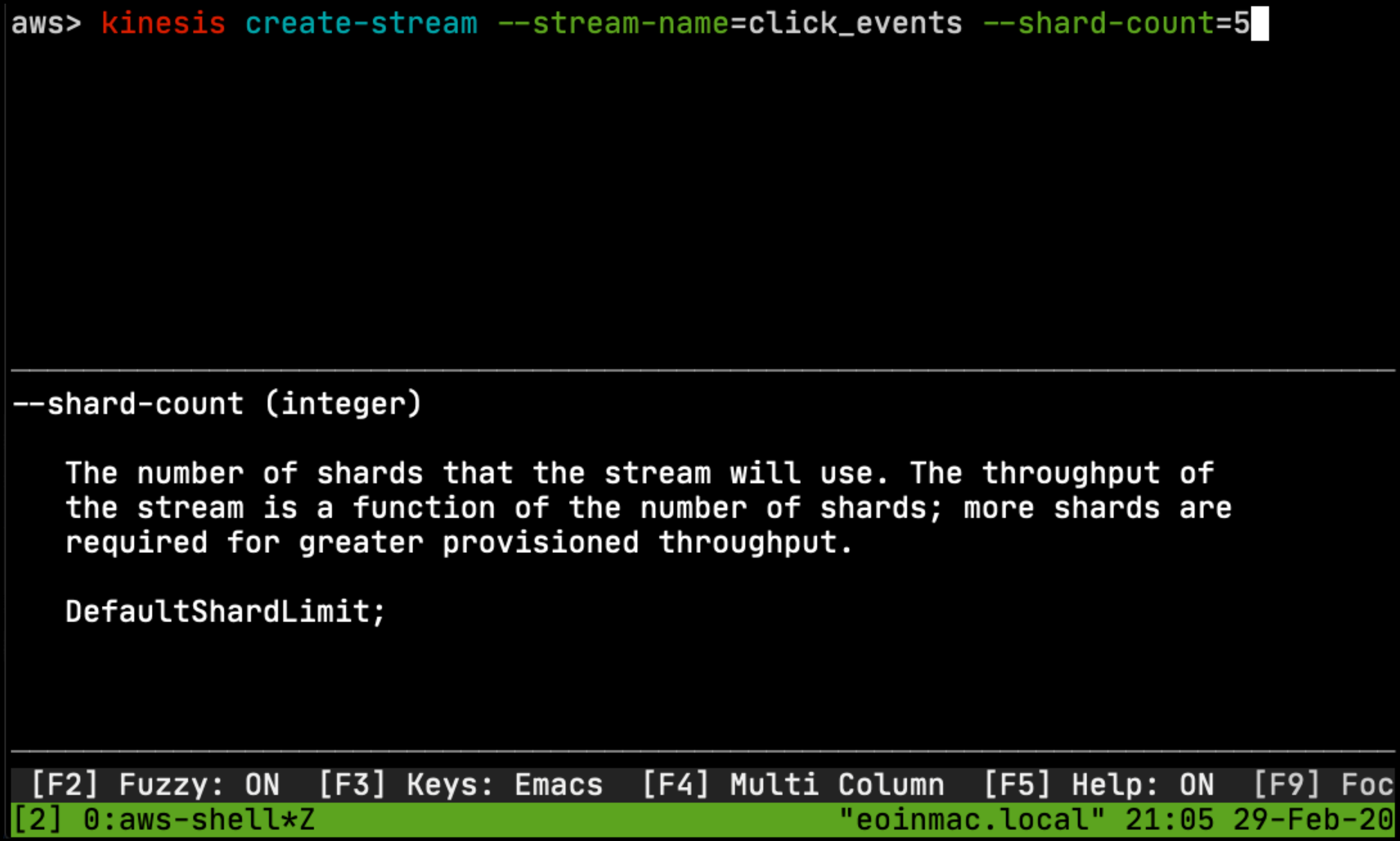
jsshow-dev-lookup

Resource-based policy Info

```
{
  "Version": "2012-10-17",
  "Id": "default",
  "Statement": [
    {
      "Sid": "jsshow-dev-LookupLambdaPermissionApiGateway-11R12BCAPS6Y9",
      "Effect": "Allow",
      "Principal": {
        "Service": "apigateway.amazonaws.com"
      },
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:eu-west-1:123456789101:function:jsshow-dev-lookup",
      "Condition": {
        "ArnLike": {
          "AWS:SourceArn": "arn:aws:execute-api:eu-west-1:123456789101:8dmtx7q649/*/*"
        }
      }
    }
  ]
}
```

@eoins

Infrastructure



```
aws> kinesis create-stream --stream-name=click_events --shard-count=5
```

```
--shard-count (integer)

   The number of shards that the stream will use. The throughput of
   the stream is a function of the number of shards; more shards are
   required for greater provisioned throughput.

   DefaultShardLimit;
```

```
 [F2] Fuzzy: ON  [F3] Keys: Emacs  [F4] Multi Column  [F5] Help: ON  [F9] Foc
[2] 0:aws-shell*Z                                  "eoinmac.local" 21:05 29-Feb-20
```

@eoins

Infrastructure

```yaml
  artifactsBucket6C289622:
    Type: AWS::S3::Bucket
    Properties:
      BucketName:
        Fn::Join:
          - ""
          - - slic-build-artifacts-
            - Ref: AWS::AccountId
            - "-"
            - Ref: AWS::Region
      VersioningConfiguration:
        Status: Enabled
    UpdateReplacePolicy: Retain
    DeletionPolicy: Retain
```

@eoins

```
const artifactsBucket = new Bucket(this, 'artifactsBucket', {
  bucketName: `slic-build-artifacts-${this.account}-${this.region}`,
  versioned: true,
})
const sourceCodeBuildRole = new CodeBuildRole(this, 'sourceCodeBuildRole')
new OrchestratorPipeline(this, 'orchestrator-pipeline', {
  artifactsBucket,
  sourceCodeBuildRole
})
const buildRole = new CodeBuildRole(this, `buildRole`)
const deployRole = new CodeBuildRole(this, `deployRole`)
const moduleBuildProject = new ModuleBuildProject(this, 'module_build', { role: buildRole
const moduleDeployProject = new ModuleDeployProject(this, `module_deploy`, {
  role: deployRole
})
;[StageName.stg, StageName.prod].forEach((stageName: StageName) => {
  const pipelineRole = new ModulePipelineRole(
    this,
    `${stageName}PipelineRole`
  )
```

CDK

Continuous Deployment

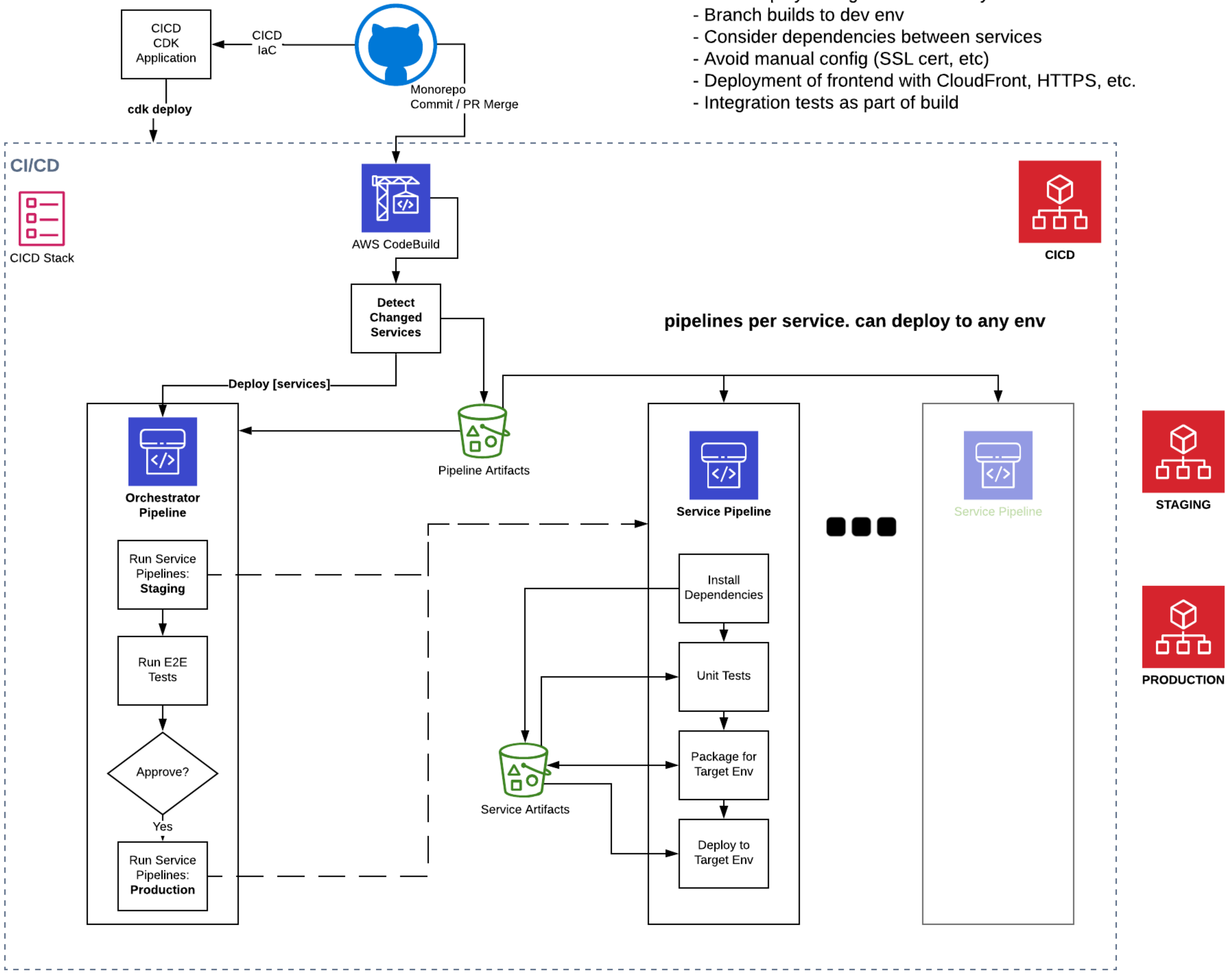
Essential

Infrastructure resources + code

Deployment time is critical

@eoins

- Build/deploy changed services only
- Branch builds to dev env
- Consider dependencies between services
- Avoid manual config (SSL cert, etc)
- Deployment of frontend with CloudFront, HTTPS, etc.
- Integration tests as part of build



@eoins

Observability

Structured Logs

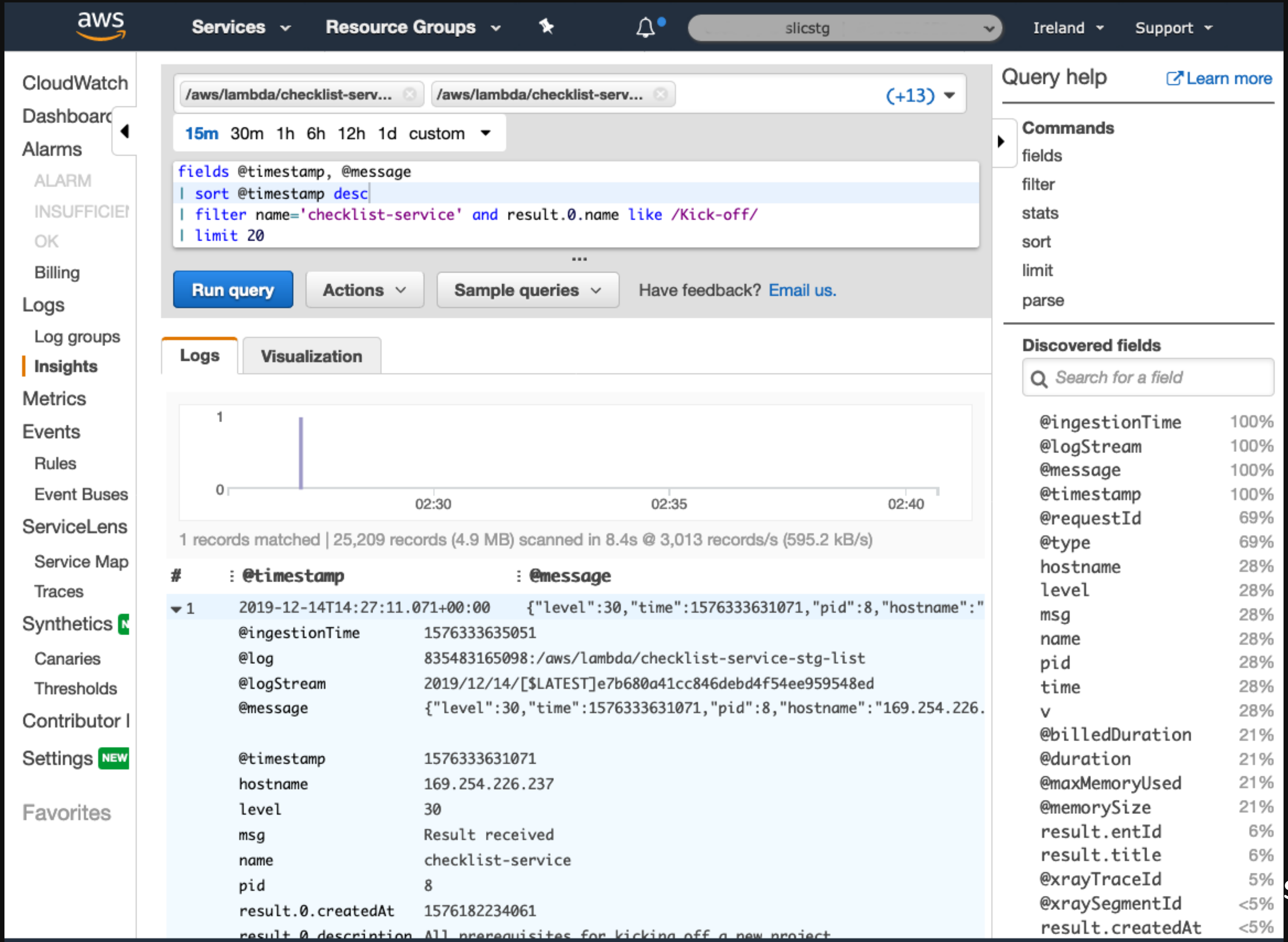
```
npm install pino --save
```

```
const pino = require('pino')
const log = pino({ name: 'pino-logging-example' })

log.info({ a: 1, b: 2 }, 'Hello world')
const err = new Error('Something failed')
log.error({ err })
```

```
{"level":30,"time":1575753091452,"pid":88157,"hostname":"
eoinmac","name":"pino-logging-
example","a":1,"b":2,"msg":"Hello world","v":1}
```

Centralized Logs



```
fields @timestamp, @message
| sort @timestamp desc
| filter name='checklist-service' and result.0.name like /Kick-off/
| limit 20
```

1 records matched | 25,209 records (4.9 MB) scanned in 8.4s @ 3,013 records/s (595.2 kB/s)

```
#    @timestamp                       @message
▾1   2019-12-14T14:27:11.071+00:00    {"level":30,"time":1576333631071,"pid":8,"hostname":"
     @ingestionTime         1576333635051
     @log                   835483165098:/aws/lambda/checklist-service-stg-list
     @logStream             2019/12/14/[$LATEST]e7b680a41cc846debd4f54ee959548ed
     @message               {"level":30,"time":1576333631071,"pid":8,"hostname":"169.254.226.

     @timestamp             1576333631071
     hostname               169.254.226.237
     level                  30
     msg                    Result received
     name                   checklist-service
     pid                    8
     result.0.createdAt     1576182234061
     result.0.description   All prerequisites for kicking off a new project
```

@ingestionTime	100%
@logStream	100%
@message	100%
@timestamp	100%
@requestId	69%
@type	69%
hostname	28%
level	28%
msg	28%
name	28%
pid	28%
time	28%
v	28%
@billedDuration	21%
@duration	21%
@maxMemoryUsed	21%
@memorySize	21%
result.entId	6%
result.title	6%
@xrayTraceId	5%
@xraySegmentId	<5%
result.createdAt	<5%

s

Service Metrics

Service	Example Metrics
Lambda	Invocations, Errors, IteratorAge, ConcurrentExecutions
DynamoDB	ReturnedBytes, ConsumedWriteCapacityUnits
Lex	MissedUtteranceCount,RuntimePollyErrors
Textract	UserErrorCount, ResponseTime
Rekognition	DetectedFaceCount,DetectedLabelCount
Polly	RequestCharacters, ResponseLatency

@eoins

Application and Service Metrics

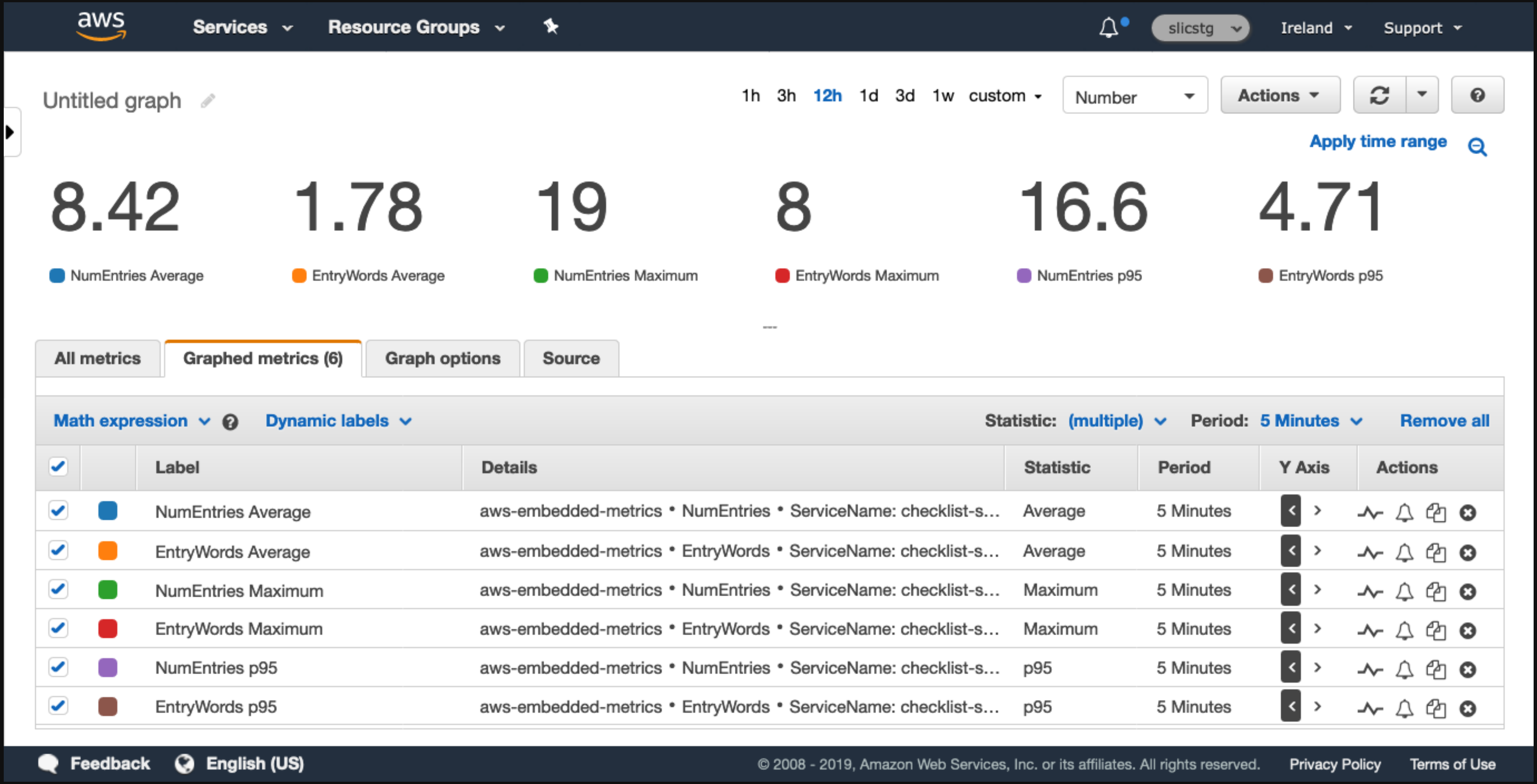
```
async function addEntry({ userId, listId, title, value }) {
  const entId = Uuid.v4()
  const params = {
    TableName: tableName,
    Key: { userId, listId },
    UpdateExpression: 'SET #ent.#entId = :entry',
    ExpressionAttributeNames: { '#ent': 'entries', '#entId': entId },
    ExpressionAttributeValues: { ':entry': { title, value } },
    ReturnValues: 'ALL_NEW'
  }
  const { Attributes: { entries } } = await dynamoDocClient().update(params).promise()

  const metrics = createMetricsLogger()
  metrics.putMetric('NumEntries', Object.keys(entries).length, Unit.Count)
  metrics.putMetric('EntryWords', title.trim().split(/s/).length, Unit.Count)
  await metrics.flush()

  return { entId, title, value }
}
```

@eoins

Application and Service Metrics

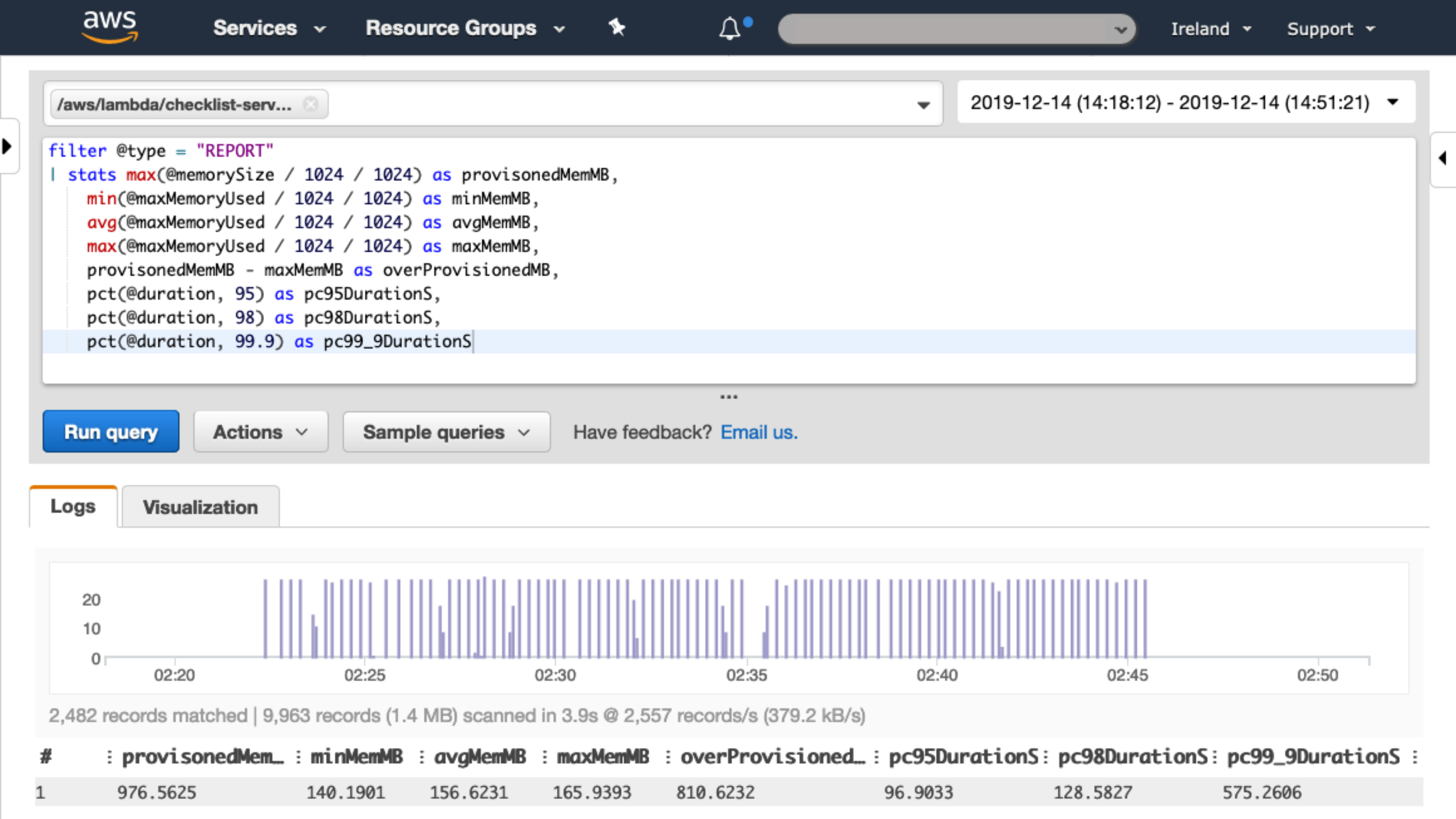


8.42	1.78	19	8	16.6	4.71
🟦 NumEntries Average	🟧 EntryWords Average	🟩 NumEntries Maximum	🟥 EntryWords Maximum	🟪 NumEntries p95	🟫 EntryWords p95

✓		Label	Details	Statistic	Period	Y Axis	Actions
✓	🟦	NumEntries Average	aws-embedded-metrics • NumEntries • ServiceName: checklist-s...	Average	5 Minutes		
✓	🟧	EntryWords Average	aws-embedded-metrics • EntryWords • ServiceName: checklist-s...	Average	5 Minutes		
✓	🟩	NumEntries Maximum	aws-embedded-metrics • NumEntries • ServiceName: checklist-s...	Maximum	5 Minutes		
✓	🟥	EntryWords Maximum	aws-embedded-metrics • EntryWords • ServiceName: checklist-s...	Maximum	5 Minutes		
✓	🟪	NumEntries p95	aws-embedded-metrics • NumEntries • ServiceName: checklist-s...	p95	5 Minutes		
✓	🟫	EntryWords p95	aws-embedded-metrics • EntryWords • ServiceName: checklist-s...	p95	5 Minutes		

Feedback
English (US)
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

@eoins



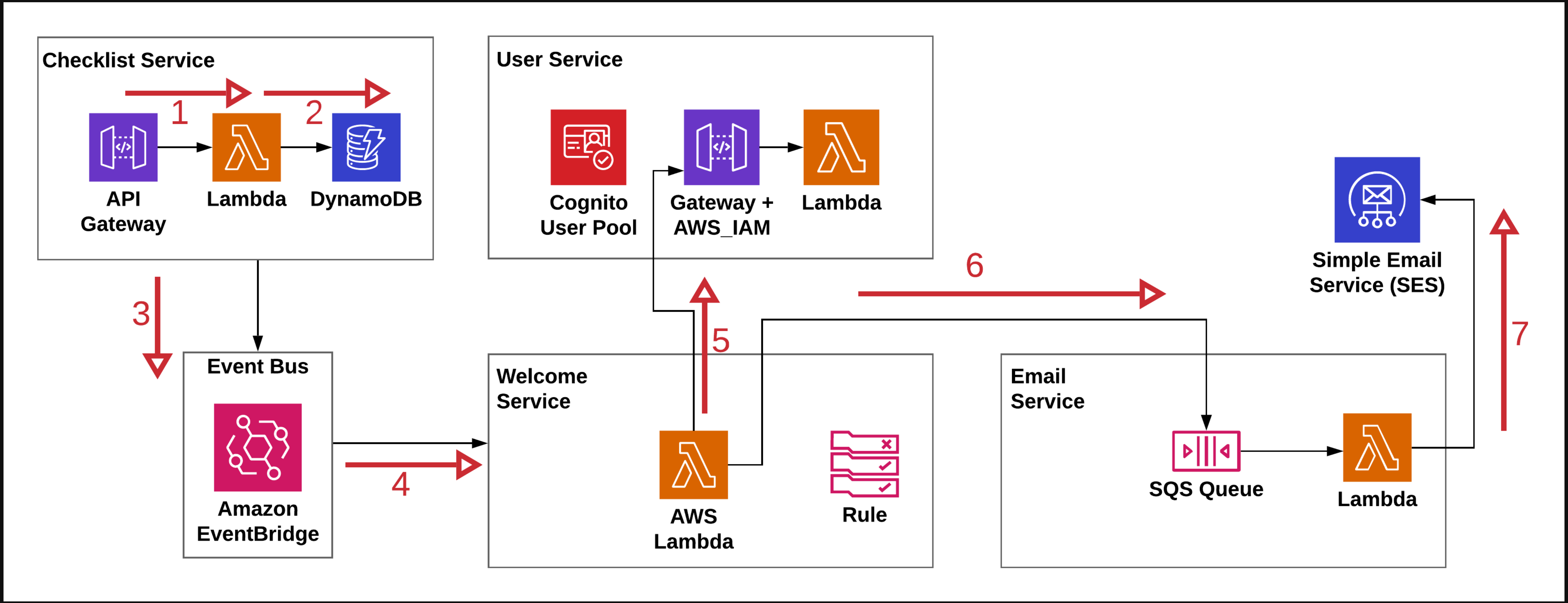
```
filter @type = "REPORT"
| stats max(@memorySize / 1024 / 1024) as provisonedMemMB,
  min(@maxMemoryUsed / 1024 / 1024) as minMemMB,
  avg(@maxMemoryUsed / 1024 / 1024) as avgMemMB,
  max(@maxMemoryUsed / 1024 / 1024) as maxMemMB,
  provisonedMemMB - maxMemMB as overProvisionedMB,
  pct(@duration, 95) as pc95DurationS,
  pct(@duration, 98) as pc98DurationS,
  pct(@duration, 99.9) as pc99_9DurationS
```

Have feedback? Email us.

2,482 records matched | 9,963 records (1.4 MB) scanned in 3.9s @ 2,557 records/s (379.2 kB/s)

#	provisonedMem...	minMemMB	avgMemMB	maxMemMB	overProvisioned...	pc95DurationS	pc98DurationS	pc99_9DurationS
1	976.5625	140.1901	156.6231	165.9393	810.6232	96.9033	128.5827	575.2606

Distributed Tracing



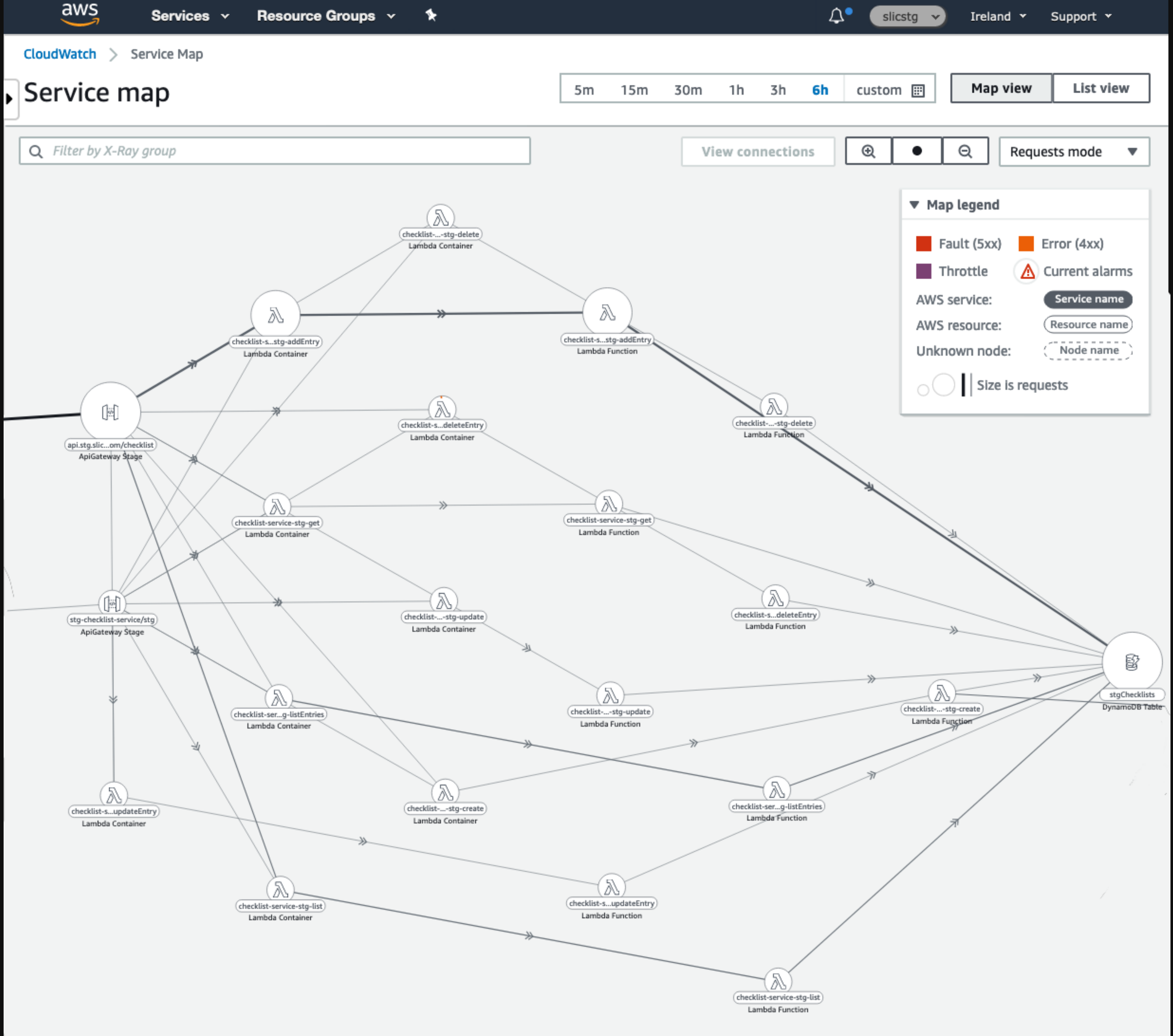
@eoins

Distributed Tracing

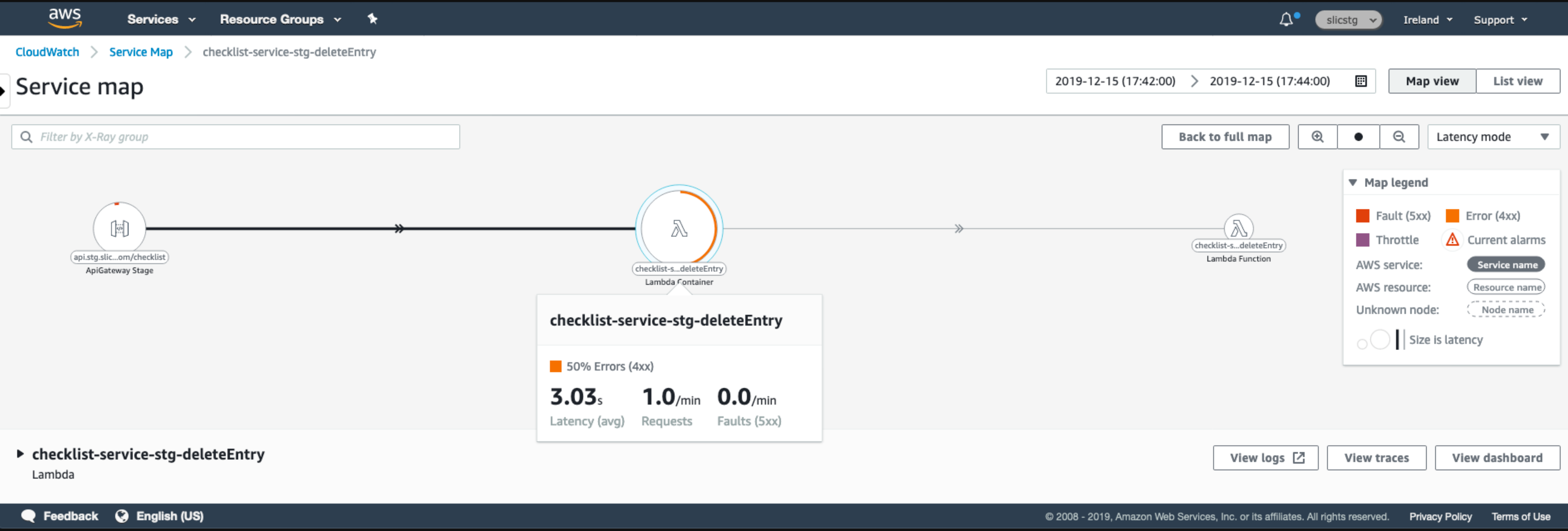
```yaml
tracing:
  apiGateway: true
  lambda: true
```

```javascript
const awsXray = require('aws-xray-sdk')
const AWS = awsXray.captureAWS(require('aws-sdk'))
```

@eoins


aws
Services
Resource Groups
slicstg
Ireland
Support
CloudWatch > Service Map
Service map
5m 15m 30m 1h 3h 6h custom
Map view
List view
Filter by X-Ray group
View connections
Requests mode
Map legend
Fault (5xx)
Error (4xx)
Throttle
Current alarms
AWS service: Service name
AWS resource: Resource name
Unknown node: Node name
Size is requests
checklist-...-stg-delete Lambda Container
checklist-s...stg-addEntry Lambda Container
checklist-s...stg-addEntry Lambda Function
api.stg.slic...om/checklist ApiGateway Stage
checklist-s...deleteEntry Lambda Container
checklist-...-stg-delete Lambda Function
checklist-service-stg-get Lambda Container
checklist-service-stg-get Lambda Function
stg-checklist-service/stg ApiGateway Stage
checklist-...-stg-update Lambda Container
checklist-s...deleteEntry Lambda Function
stgChecklists DynamoDB Table
checklist-ser...g-listEntries Lambda Container
checklist-...-stg-update Lambda Function
checklist-...-stg-create Lambda Function
checklist-s...updateEntry Lambda Container
checklist-...-stg-create Lambda Container
checklist-ser...g-listEntries Lambda Function
checklist-service-stg-list Lambda Container
checklist-s...updateEntry Lambda Function
checklist-service-stg-list Lambda Function

@eoins



Service map

2019-12-15 (17:42:00) > 2019-12-15 (17:44:00)

checklist-service-stg-deleteEntry

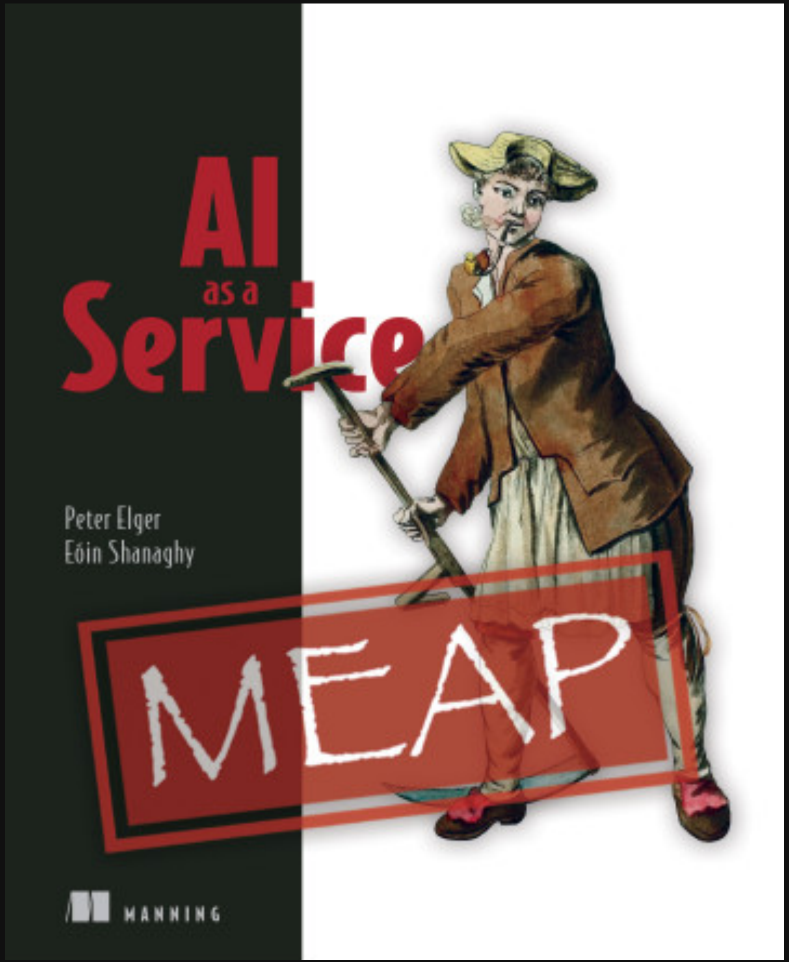
🟧 50% Errors (4xx)

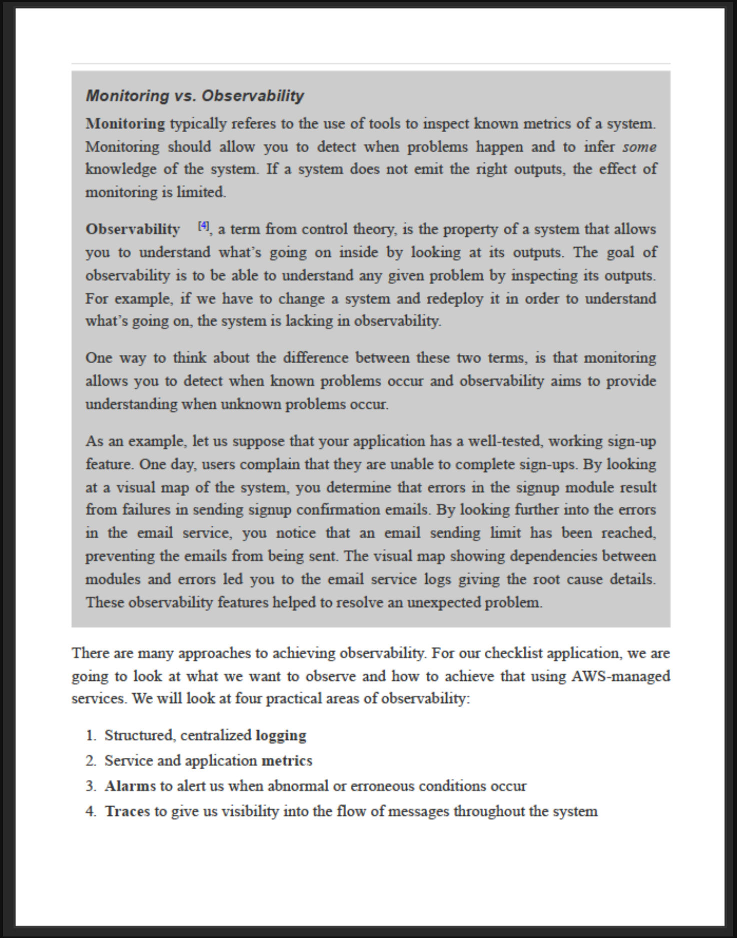
3.03s	1.0/min	0.0/min
Latency (avg)	Requests	Faults (5xx)

checklist-service-stg-deleteEntry
Lambda

@eoins

Chapter 6!





Monitoring vs. Observability

Monitoring typically referes to the use of tools to inspect known metrics of a system. Monitoring should allow you to detect when problems happen and to infer *some* knowledge of the system. If a system does not emit the right outputs, the effect of monitoring is limited.

Observability [4], a term from control theory, is the property of a system that allows you to understand what's going on inside by looking at its outputs. The goal of observability is to be able to understand any given problem by inspecting its outputs. For example, if we have to change a system and redeploy it in order to understand what's going on, the system is lacking in observability.

One way to think about the difference between these two terms, is that monitoring allows you to detect when known problems occur and observability aims to provide understanding when unknown problems occur.

As an example, let us suppose that your application has a well-tested, working sign-up feature. One day, users complain that they are unable to complete sign-ups. By looking at a visual map of the system, you determine that errors in the signup module result from failures in sending signup confirmation emails. By looking further into the errors in the email service, you notice that an email sending limit has been reached, preventing the emails from being sent. The visual map showing dependencies between modules and errors led you to the email service logs giving the root cause details. These observability features helped to resolve an unexpected problem.

There are many approaches to achieving observability. For our checklist application, we are going to look at what we want to observe and how to achieve that using AWS-managed services. We will look at four practical areas of observability:

1. Structured, centralized **logging**
2. Service and application **metrics**
3. **Alarms** to alert us when abnormal or erroneous conditions occur
4. **Traces** to give us visibility into the flow of messages throughout the system

@eoins

Serverless is about **productivity** and **agility**

Don’t seek **perfection**

Move out of your **comfort zone** enough

Check out **SLIC Starter** to avoid some Serverless pitfalls!

Thank You 😀

@eoins