# Observability and SLIC Watch

fourTheorem

Diren AKKOÇ

Conf42: Observability 2023

# Hi! I'm Diren

**Junior Cloud Software Developer**

fourTheorem

✉️ **Get in touch**

@AkkocDiren

Diren Akkoc

direnakkoc

fourTheorem

aws partner network

Advanced Consulting Partner

**Always re-imagining**

**We are a pioneering technology consultancy focused on AWS and serverless**

We can help with:

***Cloud Migrations***

***Training & Cloud enablement***

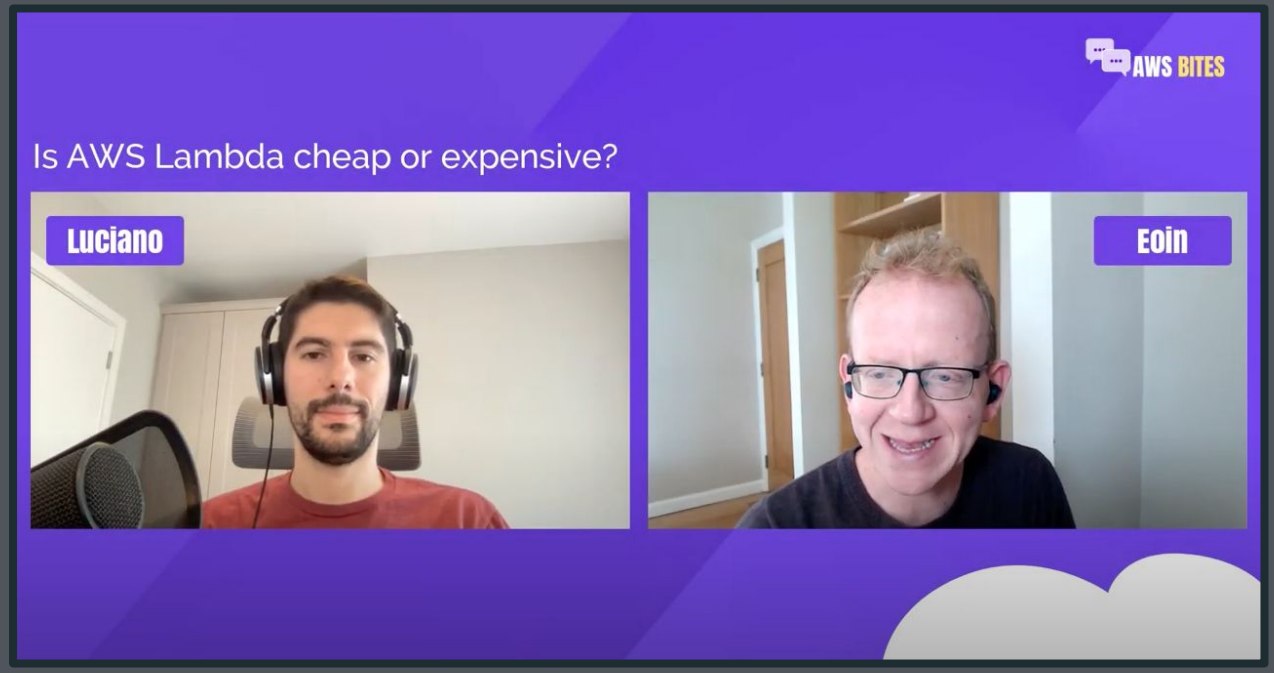
***Building high-performance serverless applications***

***Cutting cloud costs***

Middy SLIC-Starter SLIC-Watch

AWS BITES

# Check out our Podcast!




awsbites.com

# Observability in the cloud

> *a measure of how well internal states of a system can be inferred from knowledge of its external outputs*

Logs

Metrics

Tracing

# Observability in the cloud

## Logs

- It is a log of what happened

## Metrics

- Do I have a problem?

## Tracing

- Where is the problem?

# AWS native o11y = CloudWatch

Cloudwatch gives you:

- ➔ Logs with Insights
- ➔ Metrics
- ➔ Dashboards
- ➔ Alarms
- ➔ Canaries
- ➔ Distributed tracing (with X-Ray)

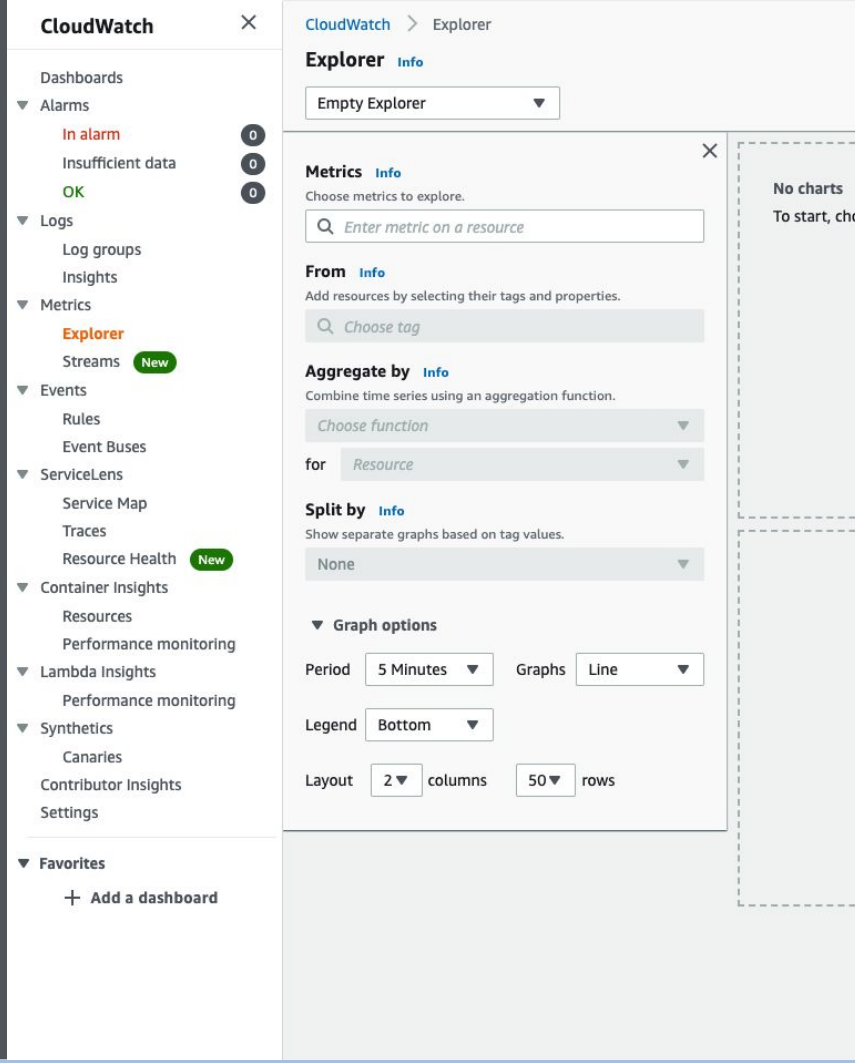
# CloudWatch out of the box

😍 A *toolkit* you can use to build observability

🤩 Metrics are *automatically* generated for all services!

😟 Lots of dashboards, but by service and not by application!

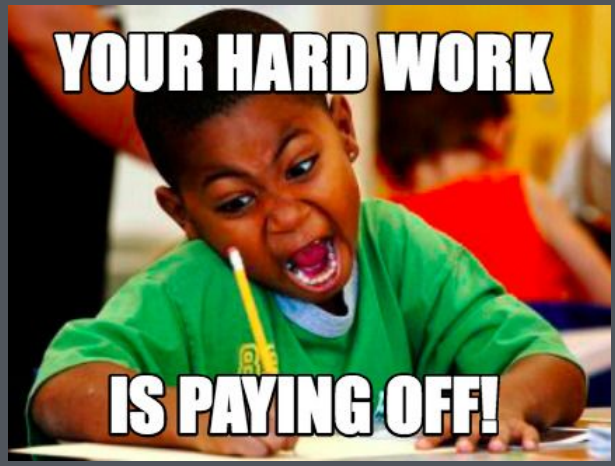
😢 **Zero alarms out of the box!**



# Setting up dashboards and alarms in CloudWatch

- 📚 **Research** and **understand** available metrics
- 📐 Decide **thresholds**
- 📊 Write IaC for application **dashboards**
- ⏰ Write IaC for service metric **alarms**
- ⏪ **Update** every time your application changes
- 📋 Copy and paste for **each stack** in your application

(a.k.a. *A LOT OF WORK!*)



# Best practices

😇 AWS Well Architected Framework

🏛️ 6 Pillars

⚙️ Operational excellence pillar covers observability

🧐 Serverless lens applies these pillars

👍 Good guidance on metrics to observe

👎 More reading and research + you still have to pick thresholds

# 4 Golden signals

## 🤭 Latency

- Response time

## 🚦 Traffic

- The amount of activity

## 🆘 Errors

- The rate of requests that fail

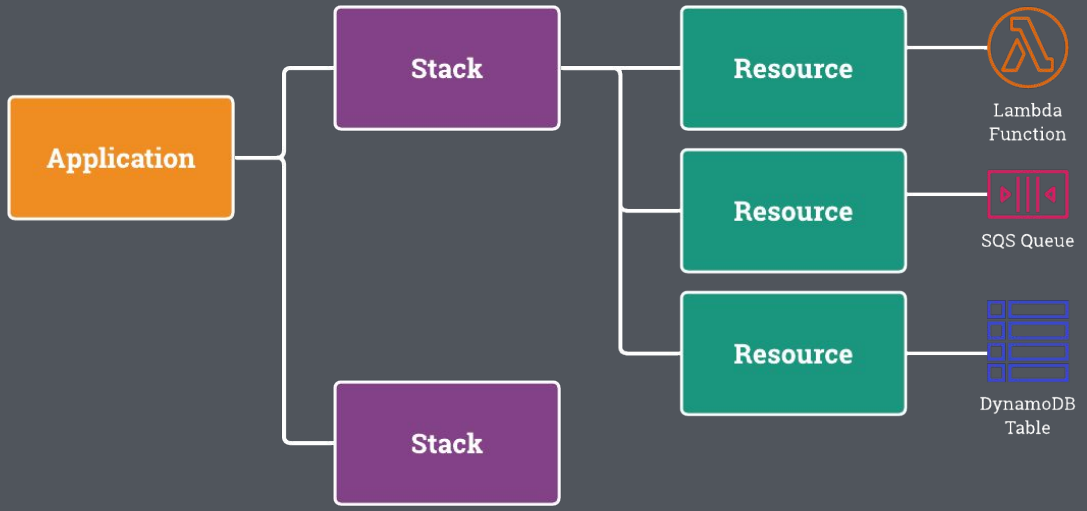
## 🫠 Saturation

- How "full" your service is.

# CloudFormation for CloudWatch Alarms 😬

```
"Type": "AWS::CloudWatch::Alarm",
"Properties": {
  "ActionsEnabled": true,
  "AlarmActions": [
    "arn:aws:sns:eu-west-1:665863320777:FTSLICAlarms"
  ],
  "AlarmName": "LambdaThrottles_serverless-test-project-dev-hello",
  "AlarmDescription": "Throttles % for serverless-test-project-dev-hello ..",
  "EvaluationPeriods": 1,
  "ComparisonOperator": "GreaterThanThreshold",
  "Threshold": 0,
  "TreatMissingData": "notBreaching",
  "Metrics": [
    {
      "Id": "throttles_pc",
      "Expression": "(throttles / throttles + invocations) * 100",
      "Label": "% Throttles",
      "ReturnData": true
    },
    {
      "Id": "throttles",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/Lambda",
          "MetricName": "Throttles",
          "Dimensions": [
            {
              "Name": "FunctionName",
              "Value": "serverless-test-project-dev-hello"
            }
          ]
        },
        "Period": 60,
        "Stat": "Sum"
      },
      "ReturnData": false
    },
    {
      "Id": "invocations",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/Lambda",
          "MetricName": "Invocations",
```

# Can we automate this?



*Magically generated alarms and dashboards for each application!*

# Introducing SLIC Watch

[fth.link/slic-watch](fth.link/slic-watch)

# slic-watch

serverless ⚡ | npm v2.1.2 | Build passing | coverage 100% | code style standard

Automatic, best-practice CloudWatch **Dashboards** and **Alarms** for your SAM, CloudFormation, CDK and Serverless Framework applications.

SLIC Watch supports: *AWS Lambda, API Gateway, DynamoDB, Kinesis Data Streams, SQS Queues, Step Functions, ECS (Fargate or EC2), SNS, EventBridge and Application Load Balancer.*

Supported tools include:

- ⚡ **Serverless Framework** v2 and v3 via the *SLIC Watch Serverless Plugin*
- **AWS SAM**, **AWS CDK** and **CloudFormation** using the *CloudFormation Macro*, published in the Serverless Application Repository (SAR).

## Contents


- slic-watch
  - Contents
  - Getting Started with Serverless Framework
  - Getting Started with AWS SAM, CDK or CloudFormation
    - Deploying the SLIC Watch Macro
    - Adding the SLIC Watch Transform to SAM or CloudFormation templates
    - Adding the SLIC Watch Transform to CDK Apps
  - Features
    - Lambda Functions
    - API Gateway
    - DynamoDB
    - Kinesis Data Streams
    - SQS Queues
    - Step Functions
    - ECS / Fargate
    - SNS
    - EventBridge
    - Application Load Balancer


Install

```
> npm i serverless-slic-watch-plugin
```

Repository
github.com/fourTheorem/slic-watch

Homepage
github.com/fourTheorem/slic-watch#re...

Weekly Downloads
482

| Version | License |
|---|---|
| 2.1.2 | Apache |

| Unpacked Size | Total Files |
|---|---|
| 30.9 kB | 6 |

| Issues | Pull Requests |
|---|---|
| 17 | 3 |

Last publish
6 months ago

Collaborators

>_Try on RunKit

Report malware

# How SLIC Watch works 🛠️

Your app

**serverless.yml**

sls deploy

Your app

**template.yaml**

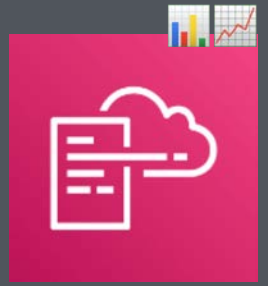
sam deploy

CloudFormation stack

**very-big.json**

SLIC Watch

👀 🛠️



CloudFormation stack ++

**even-bigger.json**

Deploy ☁️

# Configuration

🎀 SLIC Watch comes with **sane defaults**

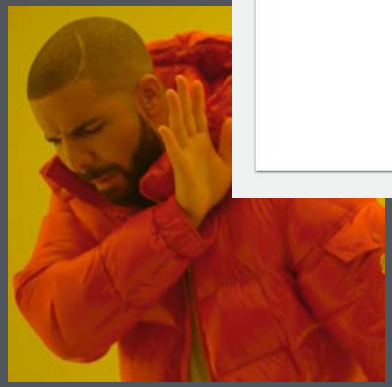
📝 You can **configure** what you don't like

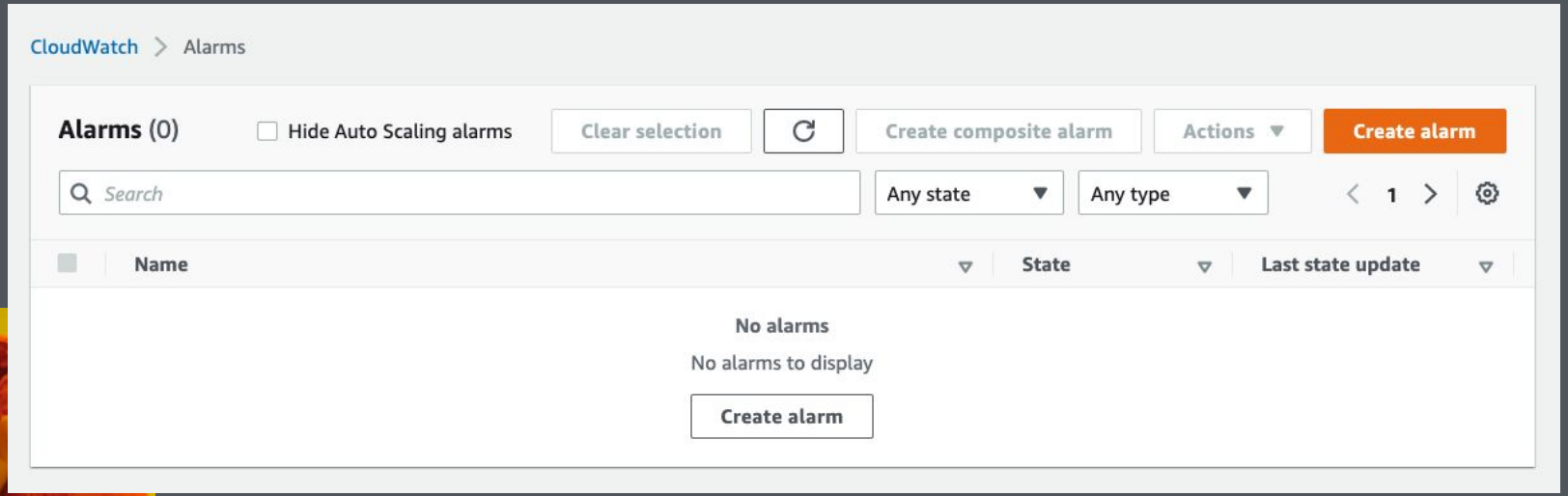
🔌 Or **disable** specific dashboards or alarms

fth.link/slic-watch

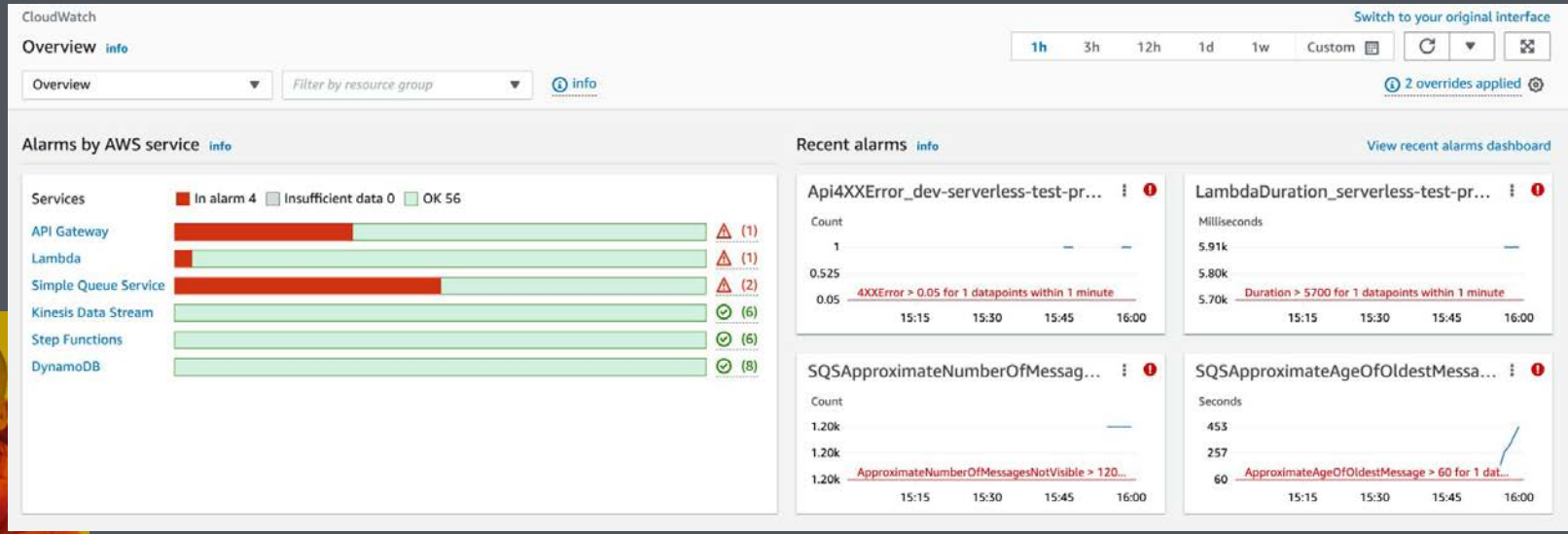
```yaml
custom:
  slicWatch:
    topic: SNS_TOPIC_ARN

    alarms:
      enabled: true
      Period: 60
      EvaluationPeriods: 1
      TreatMissingData: notBreaching
      ComparisonOperator: GreaterThanThreshold
      Lambda: # Lambda Functions
        Errors:
          Threshold: 0
          Statistic: Sum
        ThrottlesPc: # Throttles are evaluated as a percentage of invocations
          Threshold: 0
        DurationPc: # Duration is evaluated as a percentage of the function timeout
          Threshold: 95
          Statistic: Maximum
        Invocations: # No invocation alarms are created by default. Override threshold t
          enabled: false # Note: this one requires both `enabled: true` and `Threshold:
          Threshold: null
          Statistic: Sum
        IteratorAge:
          Threshold: 10000
          Statistic: Maximum
      ApiGateway: # API Gateway REST APIs
        5XXError:
          Statistic: Average
          Threshold: 0
        4XXError:
          Statistic: Average
          Threshold: 0.05
        Latency:
          ExtendedStatistic: p99
          Threshold: 5000
      States: # Step Functions
        Statistic: Sum
        ExecutionsThrottled:
          Threshold: 0
        ExecutionsFailed:
          Threshold: 0
        ExecutionsTimedOut:
```

# Before SLIC Watch

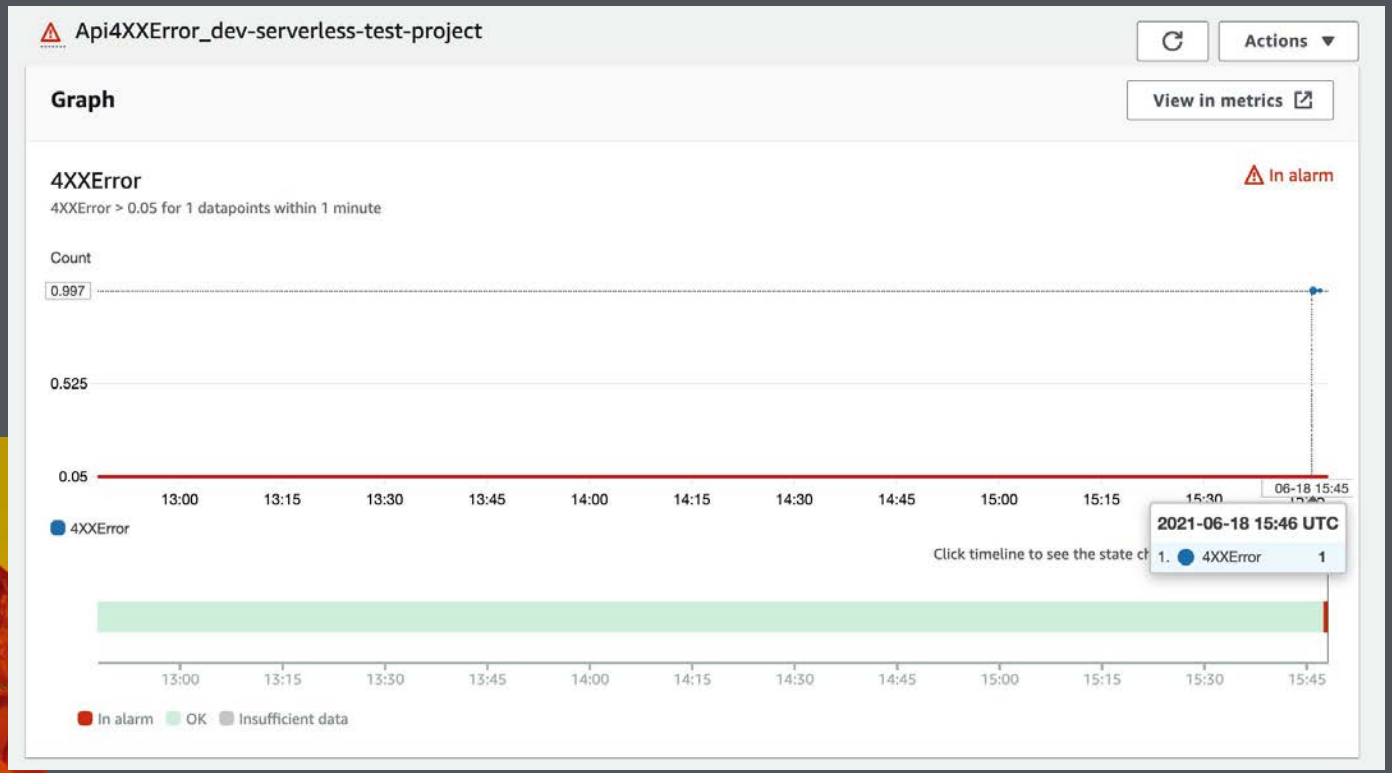




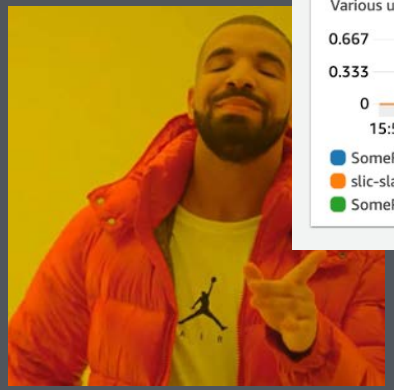
# After SLIC Watch

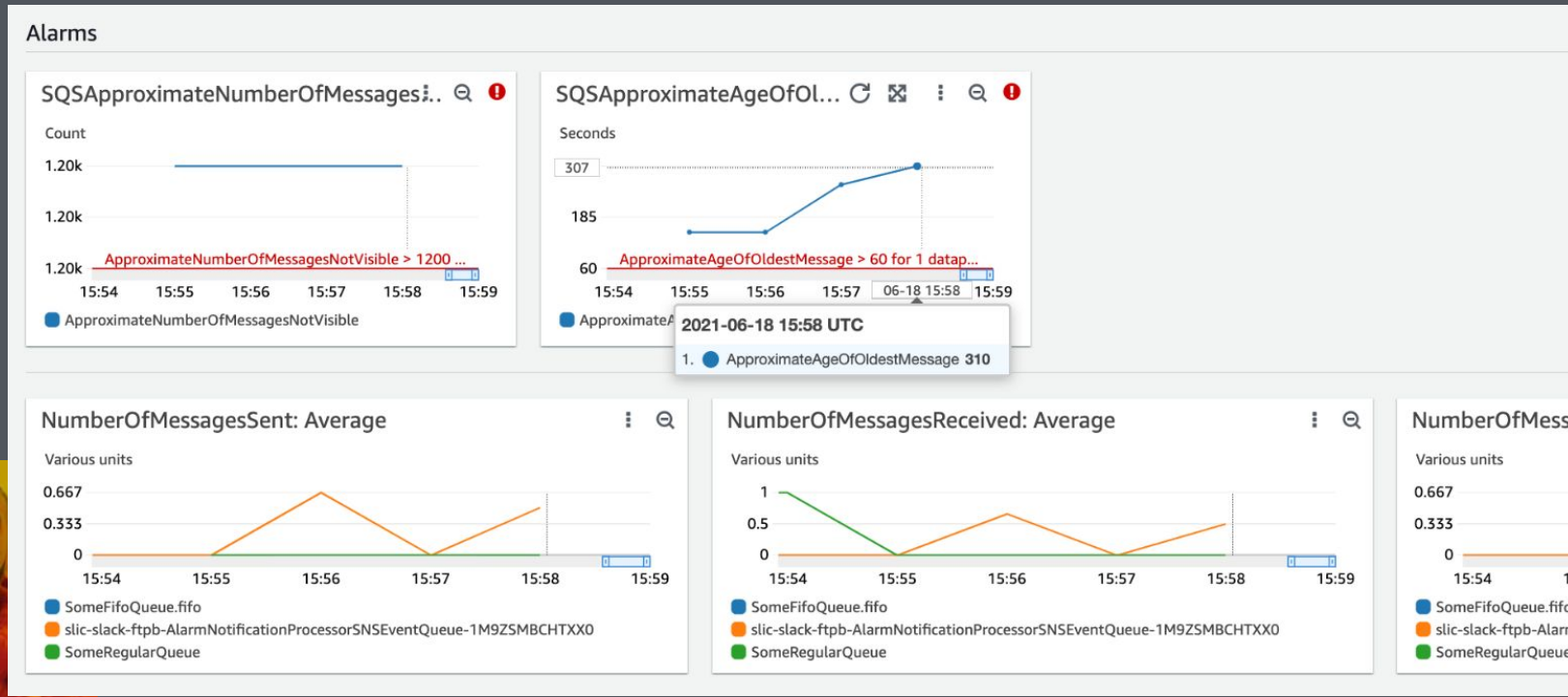


# After SLIC Watch



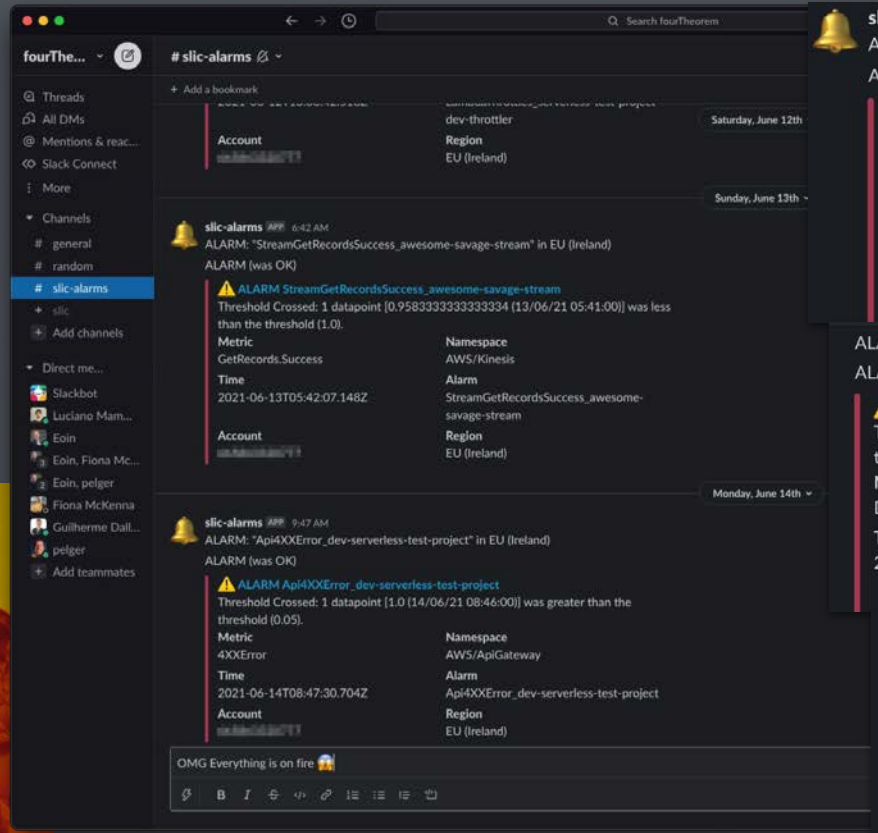
# After SLIC Watch

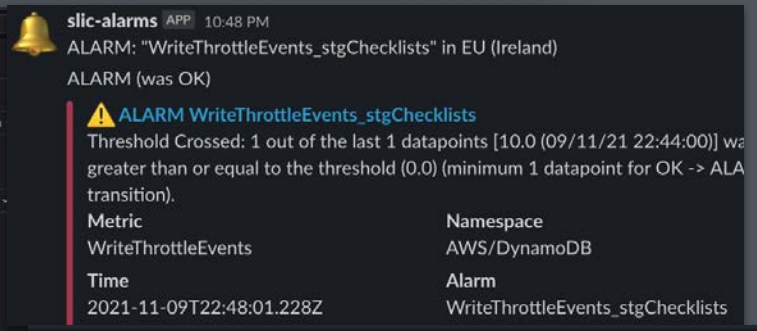


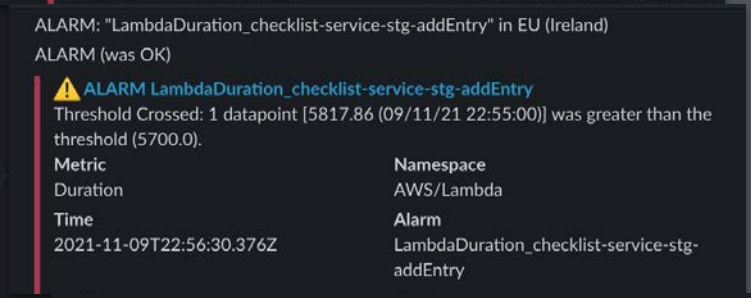


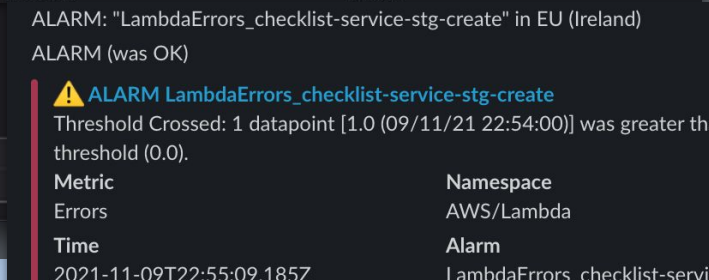
# After SLIC Watch

# After SLIC Watch: AWS Chatbot, Opsgenie, PagerDuty...



# Wrapping up 🎁

- ★ Observability is your friend from day one!
- ★ CloudWatch is very valuable if you configure it right!
- ★ Automation takes away the pain
- ★ SLIC Watch gives you this automation

🔬 Try it out! 🗣️ Give feedback! 🌈 Let's make it better!

fth.link/slic-watch

# Thank you!

fourTheorem

Cover picture by Radarsky 1984 on pexels