

Construindo sua própria **layer de observabilidade** com AWS Lambda telemetry API

Gabriel Prando

10/2024

Quem é Gabriel Prando?

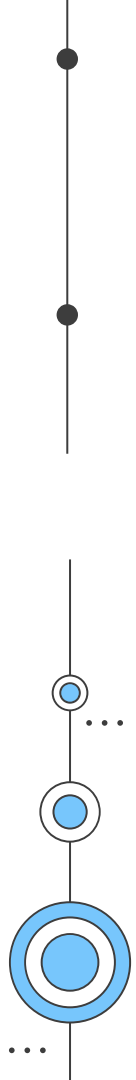


- ❖ Engenheiro de software/plataforma;
- ❖ Trabalhando com DevEx no iFood;
- ❖ Engenheiro de Computação (UTFPR-PB);
- ❖ Mestrando em Engenharia Elétrica e Computação (UTFPR-PB);
- ❖ Gosto de falar sobre Serverless, protocolos de comunicação e computação no geral;
- ❖ Em qualquer rede como @prandogabriel.

...



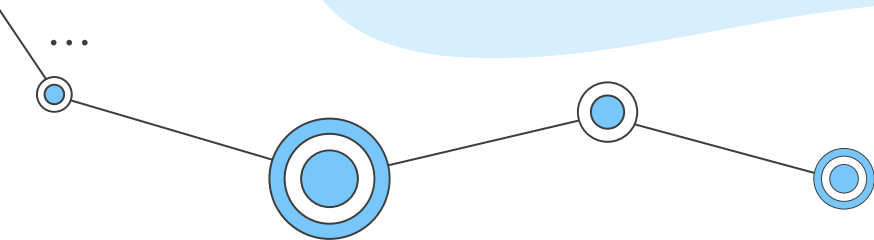
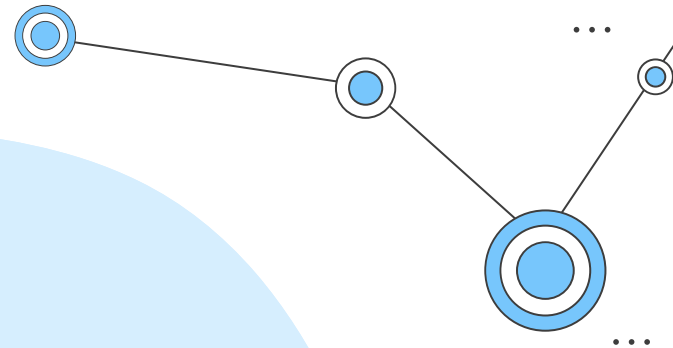
Agenda

- ❖ Alinhamento de conhecimento;
 - ❖ Funcionamento lambda;
 - ❖ Lambda Extensions;
 - ❖ Lambda Layers;
 - ❖ Telemetry API;
 - ❖ Como criar uma layer de observabilidade.
- 

...



Serverless



— **O que é serverless computing? (computação sem servidor)**



— Serverless

'Sem servidor' descreve a experiência. Os servidores são invisíveis **para o desenvolvedor**, que não os vê, gerencia ou interage, mas eles **existem sim**.



AWS Lambda

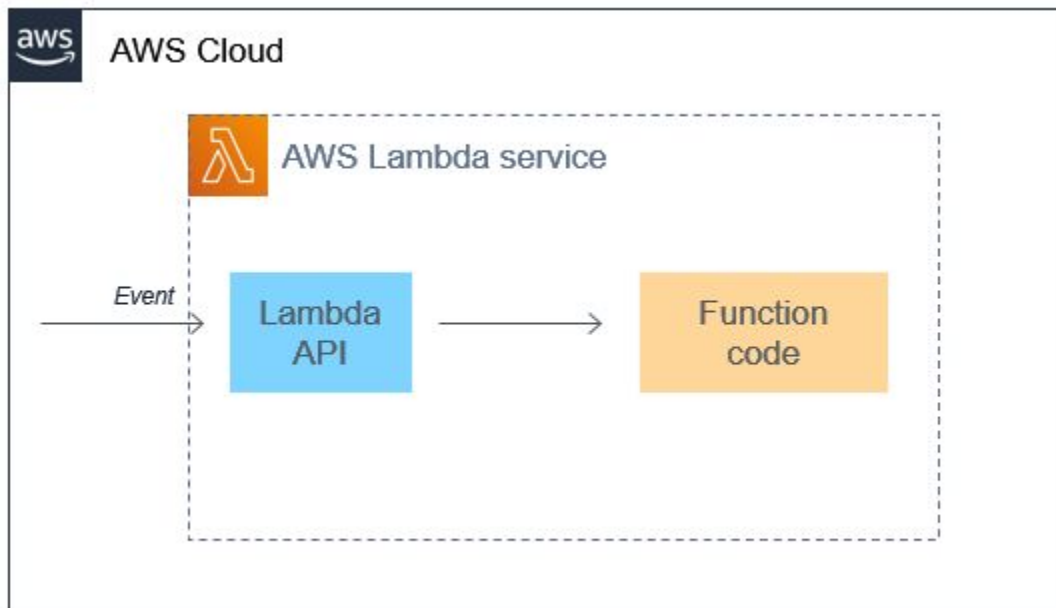
- ❖ Sem servidor para gerenciar;
- ❖ Escale com o uso;
- ❖ Orientado a eventos / triggers;
- ❖ Agnóstico a linguagem;
- ❖ Mais tempo para pensar no negócio (+dev -ops).



Casos de uso

- ❖ Rest APIs;
- ❖ Processamento assíncrono;
- ❖ Processamento de dados (ETL);
- ❖ Startups (economia, foco dev, fácil desenvolvimento e manutenção).

— Como uma lambda é chamada?

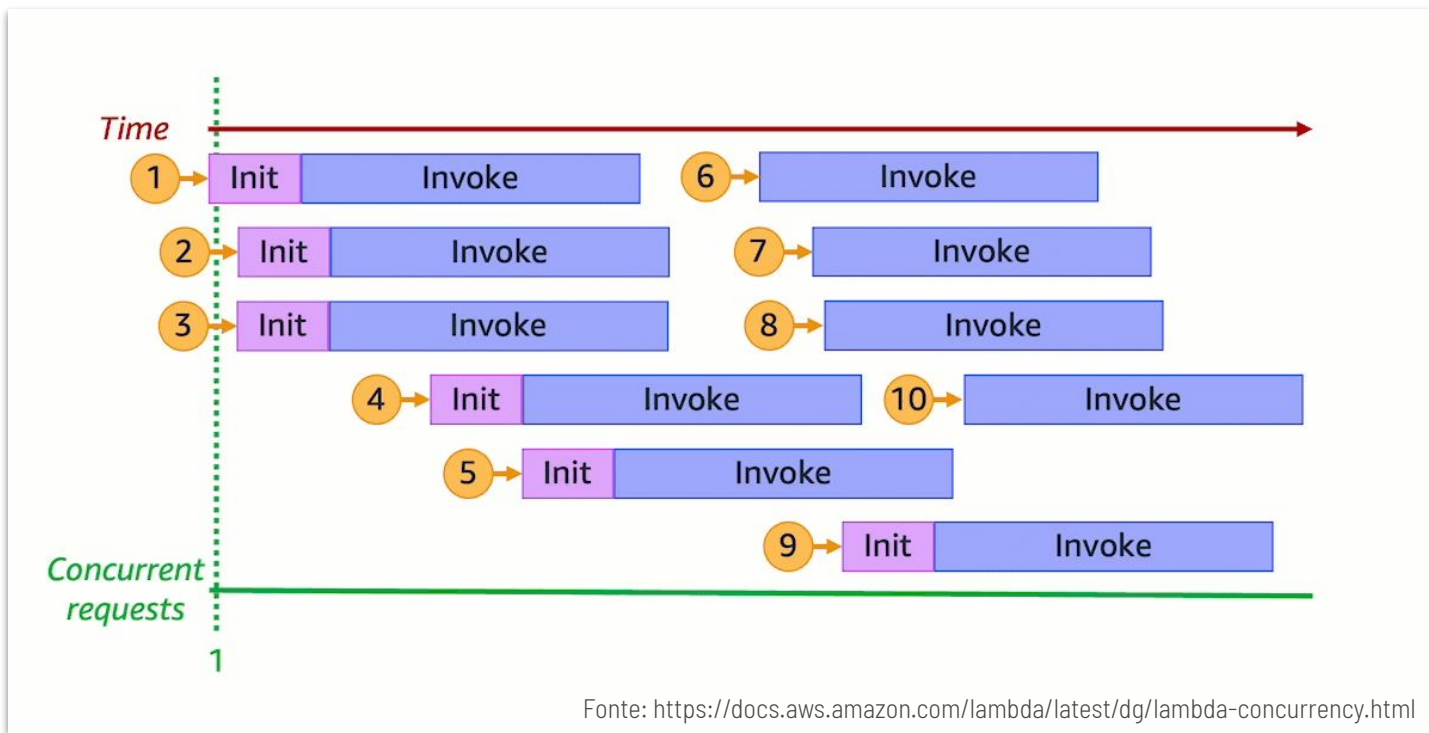


— Como functions escalam?

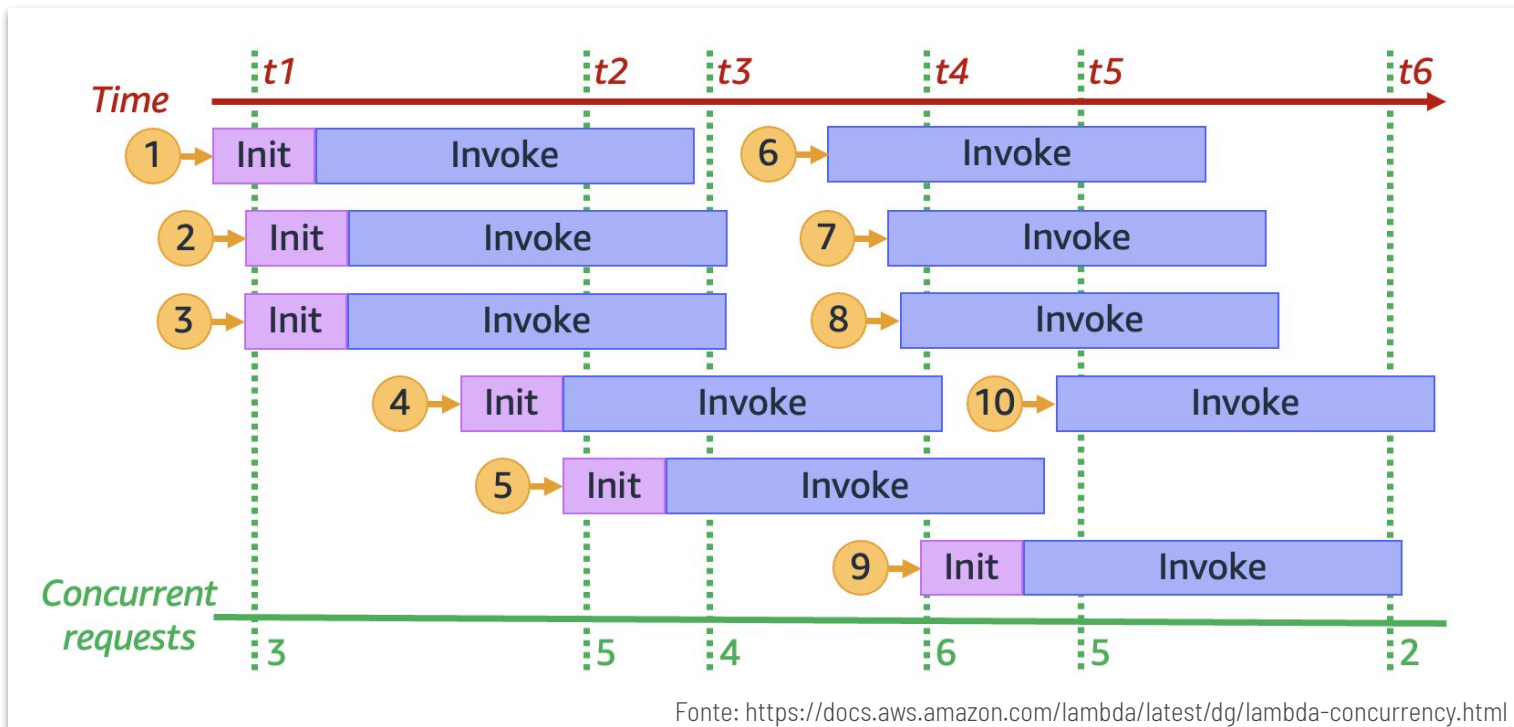


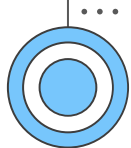
Fonte: <https://docs.aws.amazon.com/lambda/latest/dg/lambda-concurrency.html>

— Como functions escalam?



— Como functions escalam?





Como lambdas escalam?

❖ Ponto de atenção:

➤ Limite de 1000 instâncias simultâneas, por default:






Como lambdas escalam?

❖ Ponto de atenção:

➤ Limite de 1000 instâncias simultâneas, por default:

■ $1000\text{req} * 200\text{ms/req} * 60\text{s} * 60\text{min} * 24\text{h};$

- 5k req/s;
 - 300.000 req/min;
 - 18.000.000 req/hora;
 - 432.000.000 req/dia;
 - 12.960.000.000 req/mês.
- 

Como lambdas escalam?

❖ Ponto de atenção:

➤ Limite de 1000 instâncias simultâneas, por default:

■ $1000\text{req} * 200\text{ms/req} * 60\text{s} * 60\text{min} * 24\text{h};$

- 5k req/s;
- 300.000 req/min;
- 18.000.000 req/hora;
- 432.000.000 req/dia;
- 12.960.000.000 req/mês.

Acho que da pra começar
uma startup com isso...

E quanto custaria isso?

Unit conversions

Amount of memory allocated: $256 \text{ MB} \times 0.0009765625 \text{ GB in a MB} = 0.25 \text{ GB}$

Amount of ephemeral storage allocated: $512 \text{ MB} \times 0.0009765625 \text{ GB in a MB} = 0.5 \text{ GB}$

Pricing calculations

$12,960,000,000 \text{ requests} \times 200 \text{ ms} \times 0.001 \text{ ms to sec conversion factor} = 2,592,000,000.00 \text{ total compute (seconds)}$

$0.25 \text{ GB} \times 2,592,000,000.00 \text{ seconds} = 648,000,000.00 \text{ total compute (GB-s)}$

$648,000,000.00 \text{ GB-s} - 400,000 \text{ free tier GB-s} = 647,600,000.00 \text{ GB-s}$

$\text{Max}(647,600,000.00 \text{ GB-s}, 0) = 647,600,000.00 \text{ total billable GB-s}$

Tiered price for: $647,600,000.00 \text{ GB-s}$

$647,600,000 \text{ GB-s} \times 0.0000166667 \text{ USD} = 10,793.35 \text{ USD}$

Total tier cost = $10,793.3549 \text{ USD}$ (monthly compute charges)

Monthly compute charges: 10,793.35 USD

$12,960,000,000 \text{ requests} - 10,000,000 \text{ free tier requests} = 12,950,000,000 \text{ monthly billable requests}$

$\text{Max}(12,950,000,000 \text{ monthly billable requests}, 0) = 12,950,000,000.00 \text{ total monthly billable requests}$

$12,950,000,000.00 \text{ total monthly billable requests} \times 0.0000002 \text{ USD} = 2,591.80 \text{ USD}$ (monthly request charges)


Monthly request charges: 2,591.80 USD

$0.50 \text{ GB} - 0.5 \text{ GB (no additional charge)} = 0.00 \text{ GB billable ephemeral storage per function}$

Monthly ephemeral storage charges: 0 USD

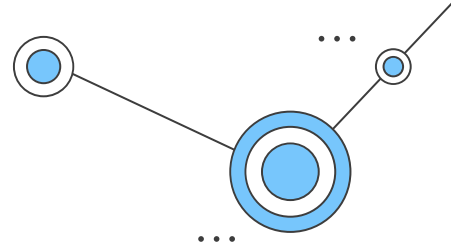
$10,793.35 \text{ USD} + 2,591.80 \text{ USD} = 13,385.15 \text{ USD}$

Lambda costs - With Free Tier (monthly): 13,385.15 USD

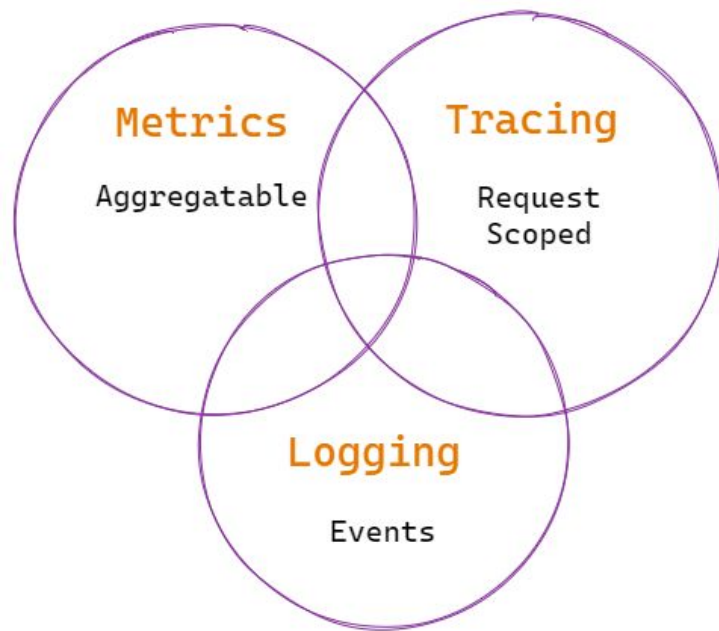


***Como funciona observabilidade nesse
econssistema?***

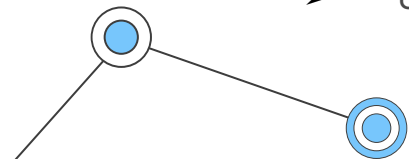
Definindo observabilidade

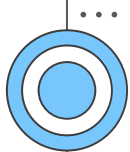


- ❖ Pilares:
 - Logs (evidências);
 - Métricas (feedback | alarmes-dashboards);
 - Trace (rastreamento);
- ❖ Compreender o estado de uma app;
- ❖ Identificar rapidamente (em tempo real):
 - Falhas;
 - Gargalos;



Fonte: <https://atatus.com/blog/observability-vs-monitoring/>



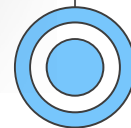


Anatomia de uma função lambda



```
console.log("Hello from function initialization");

exports.handler = async (event, context) => {
  console.log("Hello from function handler", {event});
}
```



Como vejo os logs da minha aplicação?

The screenshot displays the AWS CloudWatch Logs console interface. At the top, there's a breadcrumb navigation showing the path `/aws/lambda/my` with a 'Clear all' button. Below this, a 'Show more chosen log groups (+1)' link is visible. The main area contains a query editor with the following query:

```
1 fields @timestamp, @message
2 | sort @timestamp desc
3 # | filter @message like /minha-key/
4 # | stats count...
5 | limit 10000
```

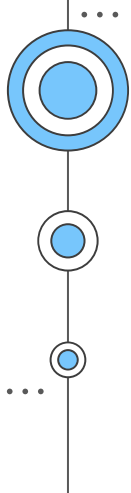
Below the query editor is a 'Query generator' button. To the right of the query editor are buttons for 'Run query' (highlighted in orange), 'Cancel', 'Save', and 'History'. A note states: 'Logs Insights query can run for maximum of 60 minutes.' Below this, a green checkmark and the word 'Complete' indicate the query status.

The interface has three tabs: 'Logs (8)', 'Patterns (-)', and 'Visualization'. The 'Logs (8)' tab is selected. On the right side of the 'Logs (8)' tab, there are buttons for 'Export results' (with a dropdown arrow), 'Add to dashboard', and a settings icon. Below the tabs, a summary bar shows 'Showing 8 of 8 records matched' with a help icon and a 'Hide histogram' link. Below this, a histogram shows the distribution of log records over time, with the x-axis labeled from 01 PM to Thu 22. The y-axis represents the count of records, ranging from 0 to 100. Below the histogram, a table displays the log records:

#	@timestamp	@message
▶ 1	2024-08-22T00:41:56.6...	END RequestId: 2438941a-9bf9-490c-bd42-5fa12d22913f
▶ 2	2024-08-22T00:41:56.6...	REPORT RequestId: 2438941a-9bf9-490c-bd42-5fa12d22913f Duration: 64.03 ms Billed Duration: 65 ms Memory Size: 128 MB Max Memory Used: 64 MB Init
▶ 3	2024-08-22T00:41:56.5...	2024-08-22T00:41:56.537Z 2438941a-9bf9-490c-bd42-5fa12d22913f INFO { source: 'invoke' }

— Como vejo os logs da minha aplicação?

#	@timestamp	@message
▶ 1	2024-09-15...	END RequestId: b0ee24d5-1414-47c1-83bd-138daf1fb25f
▶ 2	2024-09-15...	REPORT RequestId: b0ee24d5-1414-47c1-83bd-138daf1fb25f Duration: 1.22 ms Billed Duration: 2 ms Memory Size: 128 MB Max
▶ 3	2024-09-15...	2024-09-15T17:11:58.730Z b0ee24d5-1414-47c1-83bd-138daf1fb25f INFO Hello from function Hanlder
▶ 4	2024-09-15...	START RequestId: b0ee24d5-1414-47c1-83bd-138daf1fb25f Version: \$LATEST
▶ 5	2024-09-15...	END RequestId: a8581695-5a8f-46b2-92bc-c483e45dece6
▶ 6	2024-09-15...	REPORT RequestId: a8581695-5a8f-46b2-92bc-c483e45dece6 Duration: 2.76 ms Billed Duration: 3 ms Memory Size: 128 MB Max
▶ 7	2024-09-15...	2024-09-15T17:11:07.748Z a8581695-5a8f-46b2-92bc-c483e45dece6 INFO Hello from function Hanlder
▶ 8	2024-09-15...	START RequestId: a8581695-5a8f-46b2-92bc-c483e45dece6 Version: \$LATEST
▶ 9	2024-09-15...	2024-09-15T17:11:07.741Z undefined INFO Hello from function initialization
▶ 10	2024-09-15...	INIT_START Runtime Version: nodejs:20.v35 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:da38af670644eed7b5702c



Opções para se ter observabilidade em um sistema complexo?

- ❖ Cloudwatch by default;
- ❖ Métricas via EMF;
- ❖ Alarms no CW através de métricas do CW;
- ❖ AWS X-Ray (tracing);
- ❖ Export do cloudwatch para outros serviços como opensearch;
- ❖ Envio dos dados para fontes terceiras:
 - Em tempo de execução;
 - Via layer/extensão;



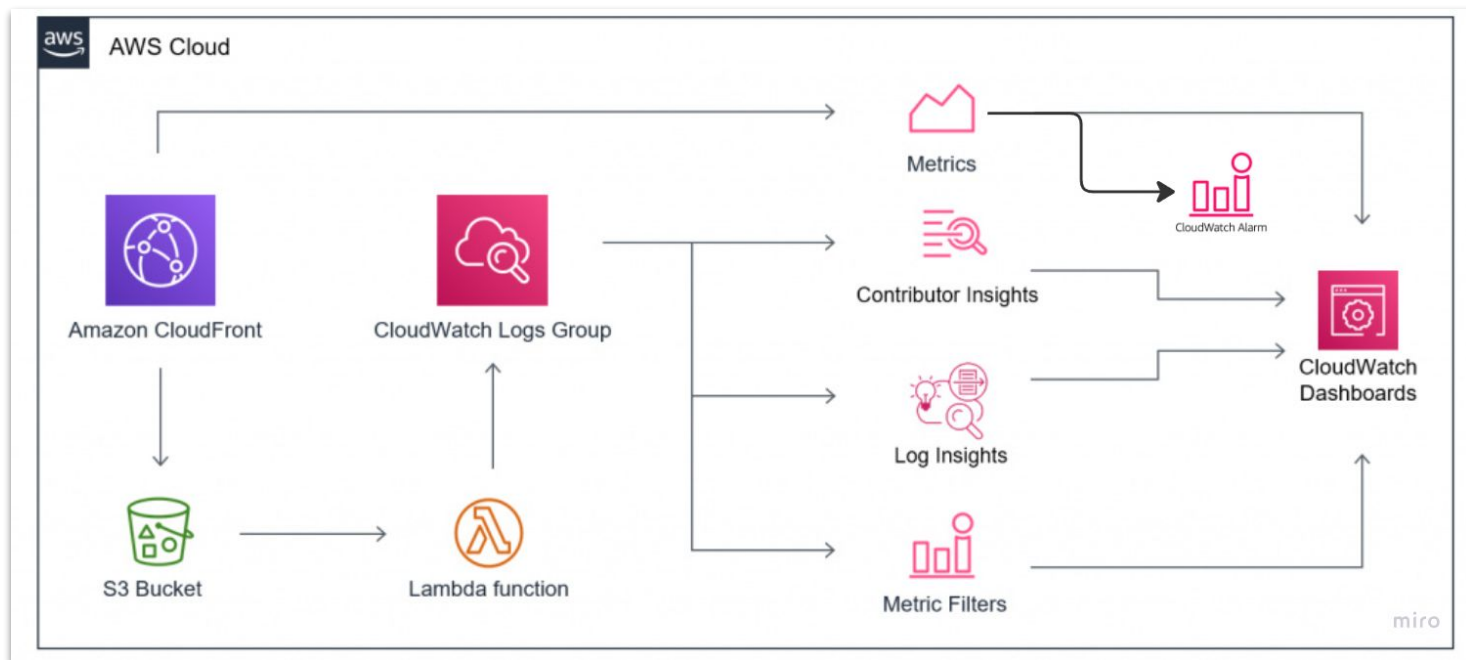
Métricas via EMF

- ❖ Log estruturado com pattern;
- ❖ Métricas custom automáticas publicadas no CW Metrics;
- ❖ Também é possível utilizar Metric Filter para algo similar.

```
{
  "_aws": {
    "CloudWatchMetrics": [
      {
        "Metrics": [{
          "Name": "Time",
          "Unit": "Milliseconds",
          "StorageResolution": 60
        }]
      }
    ]
  },
  "Time": 1
}
```

Alarmes do Cloudwatch

- ❖ Publicado em tópicos ou ações de lambdas.



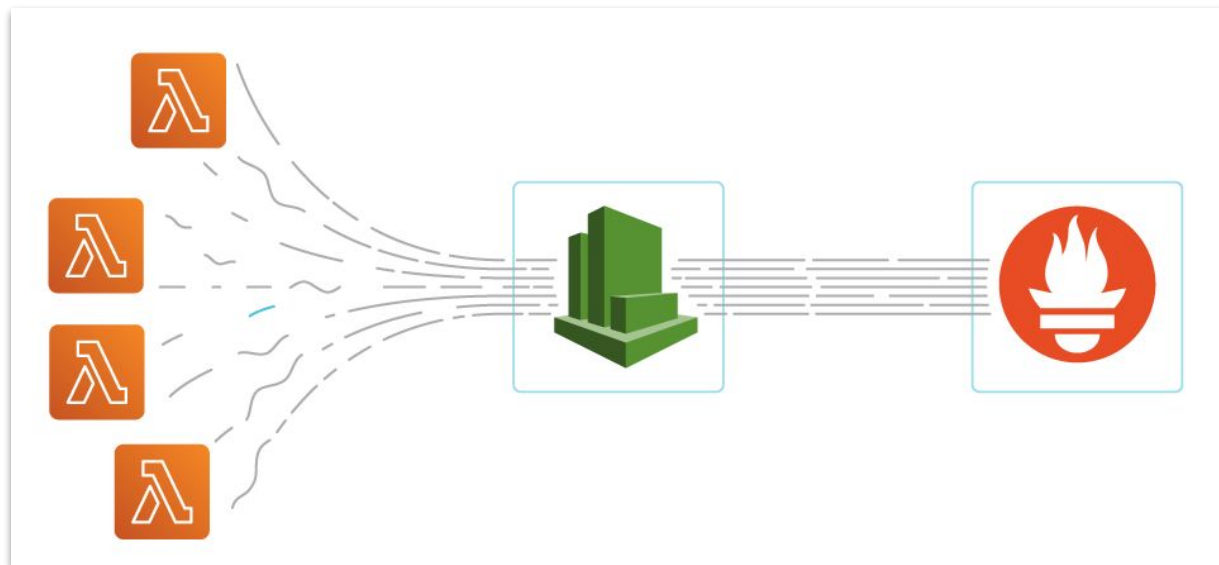
AWS X-ray

- ❖ Tracing/Tracking de requests;
- ❖ Instrumentação simples necessária.



— Export do cloudwatch

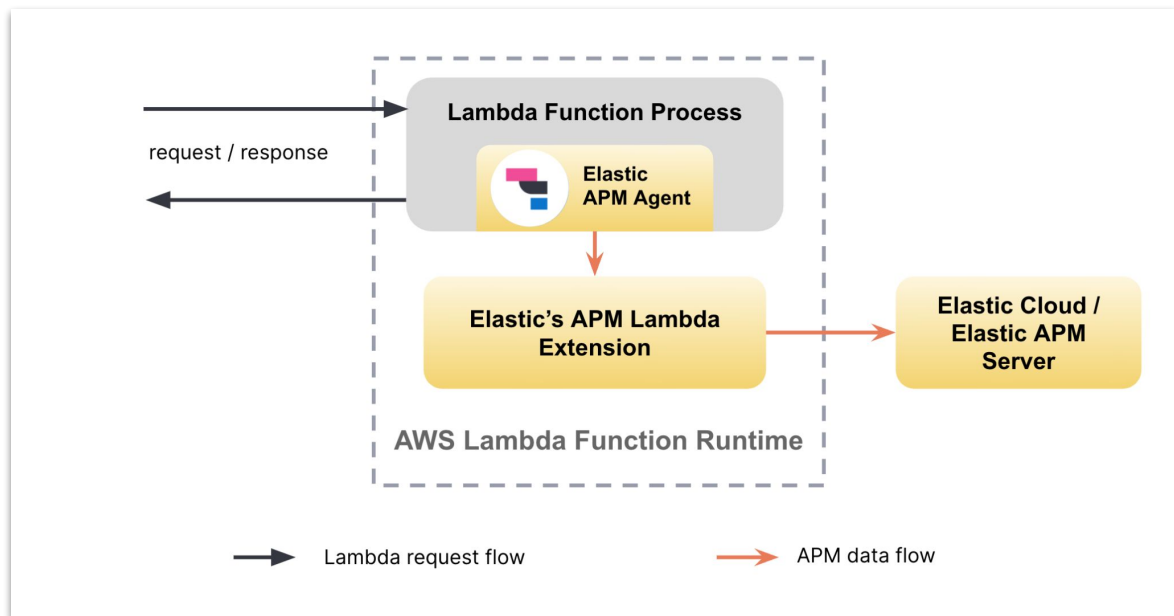
- ❖ "Plugável";
- ❖ Irá pagar por CW + Tráfego + Ferramenta (ELK, Prometheus...).



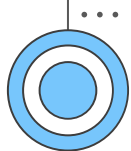
Fonte: <https://sysdig.com/blog/monitor-aws-lambda-prometheus/>

— Envio dados para fontes terceiras

- ❖ Envio em runtime ou background.

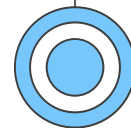


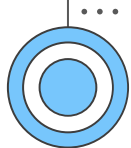
Fonte:
<https://www.elastic.co/guide/en/apm/lambda/current/aws-lambda-arch.html>



Dores do "by default AWS"

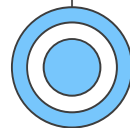
- ❖ Dificuldade de explorar logs:
 - Sintaxe própria do CW;
 - Limitação de 50 grupos de log por query;
 - Log descentralizado;
- ❖ Trace no X-Ray acaba sendo pouco eficiente;
- ❖ Limitação de não ter o serviço "em pé" direto limita alguns tipos de integrações;
- ❖ Dificuldade de sair a stack de serverless para outra arquitetura por tudo estar na AWS .





Como os grandes players de APM oferecem suporte a lambda?

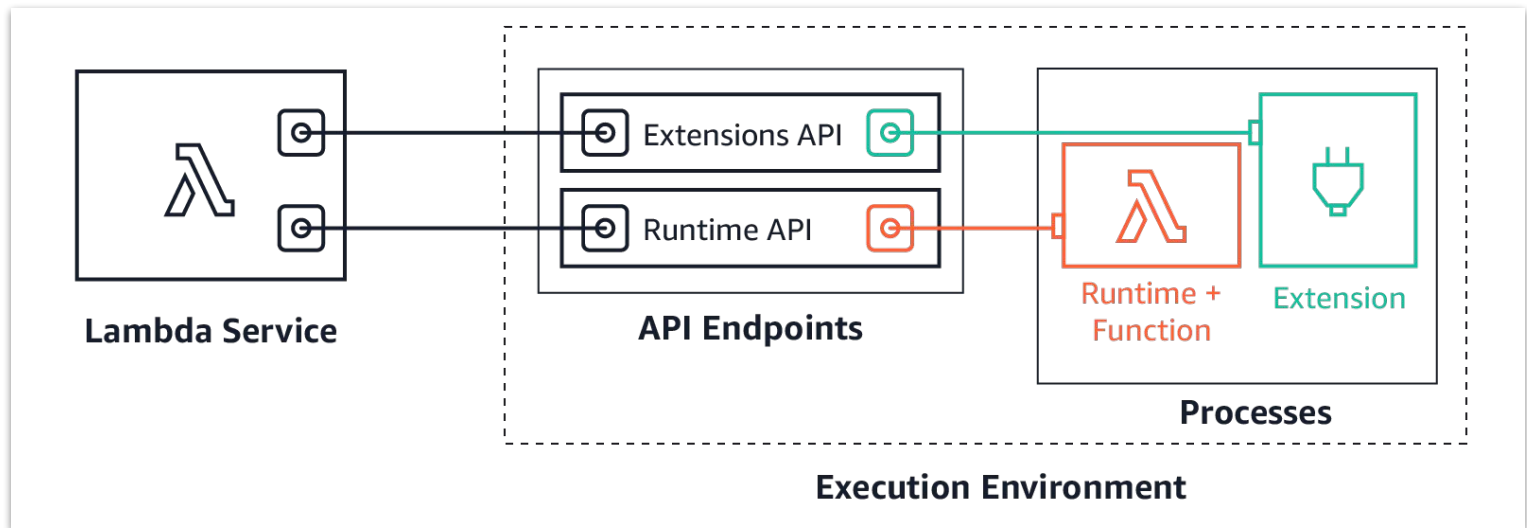
- ❖ Combinação de:
 - Lambda extensions;
 - Lambda Layer;
 - Telemetry API.
- ❖ Alguns disponíveis: ELK, Datadog, New Relic, Open telemetry...



O que são esses componentes?

Lambda Extensions

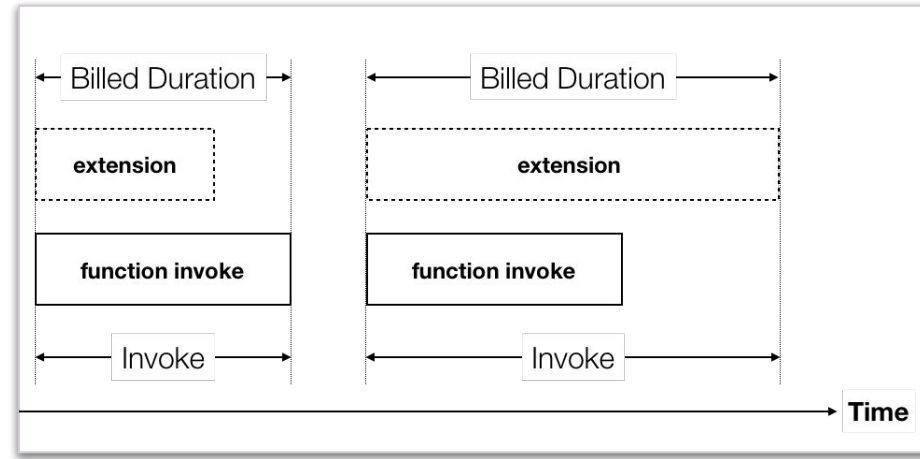
❖ Introduzida em 2020.



Fonte: AWS

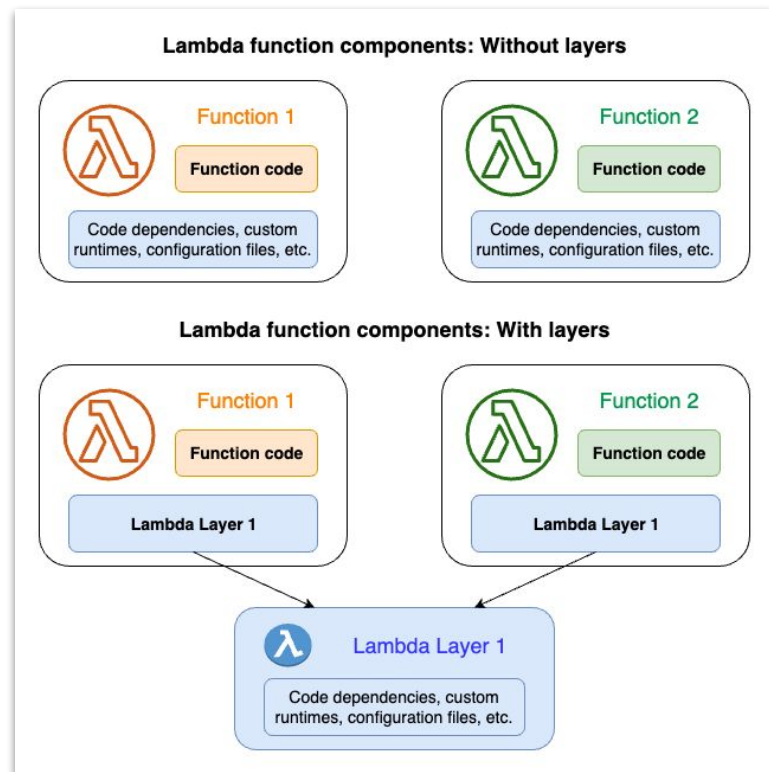
Lambda Extensions

- ❖ Facilitam a integração com outras ferramentas;
- ❖ Execução da função não depende da extension.



Lambda layers

- ❖ Arquivo .zip de conteúdo compartilhado;
- ❖ Empacote e compartilhe:
 - Dependências e bibliotecas;
 - Runtime personalizado;
 - Arquivos de configuração.



Fonte: AWS

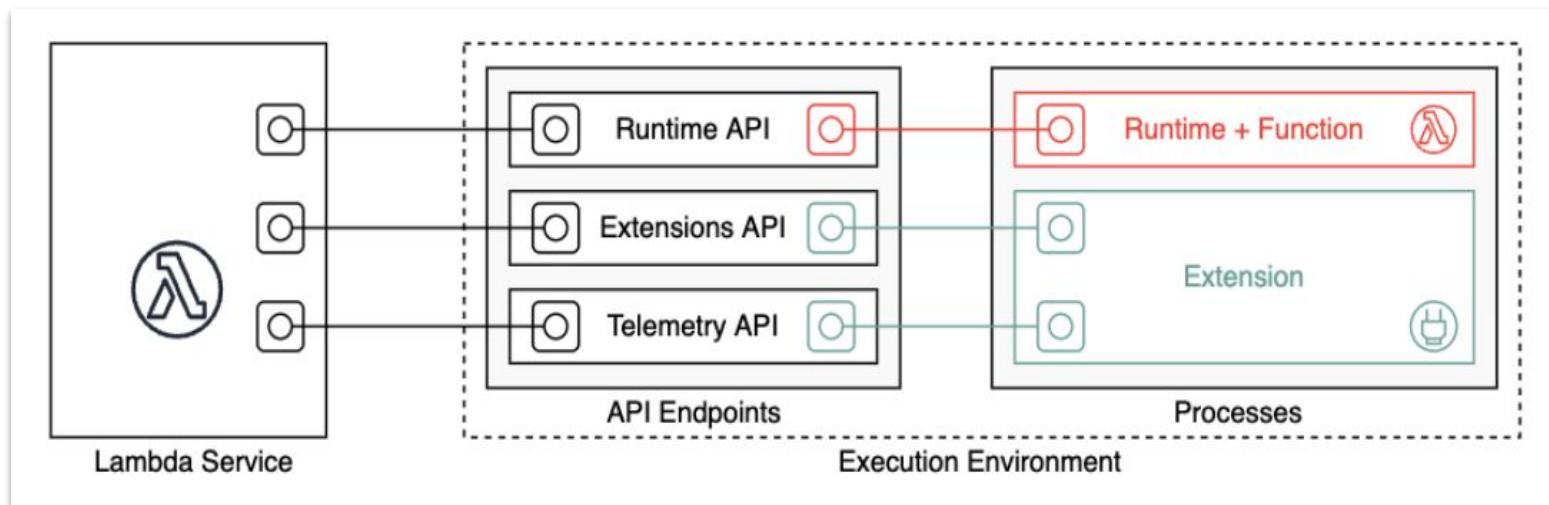
— Lambda layers

```
Description: CloudFormation Template for Lambda Function with Lambda Layer
Resources:
  MyLambdaLayer:
    Type: AWS::Lambda::LayerVersion
    Properties:
      LayerName: my-lambda-layer
      Description: My Lambda Layer
      Content:
        S3Bucket: amzn-s3-demo-bucket
        S3Key: my-layer.zip
      CompatibleRuntimes:
        - python3.9
        - python3.10
        - python3.11

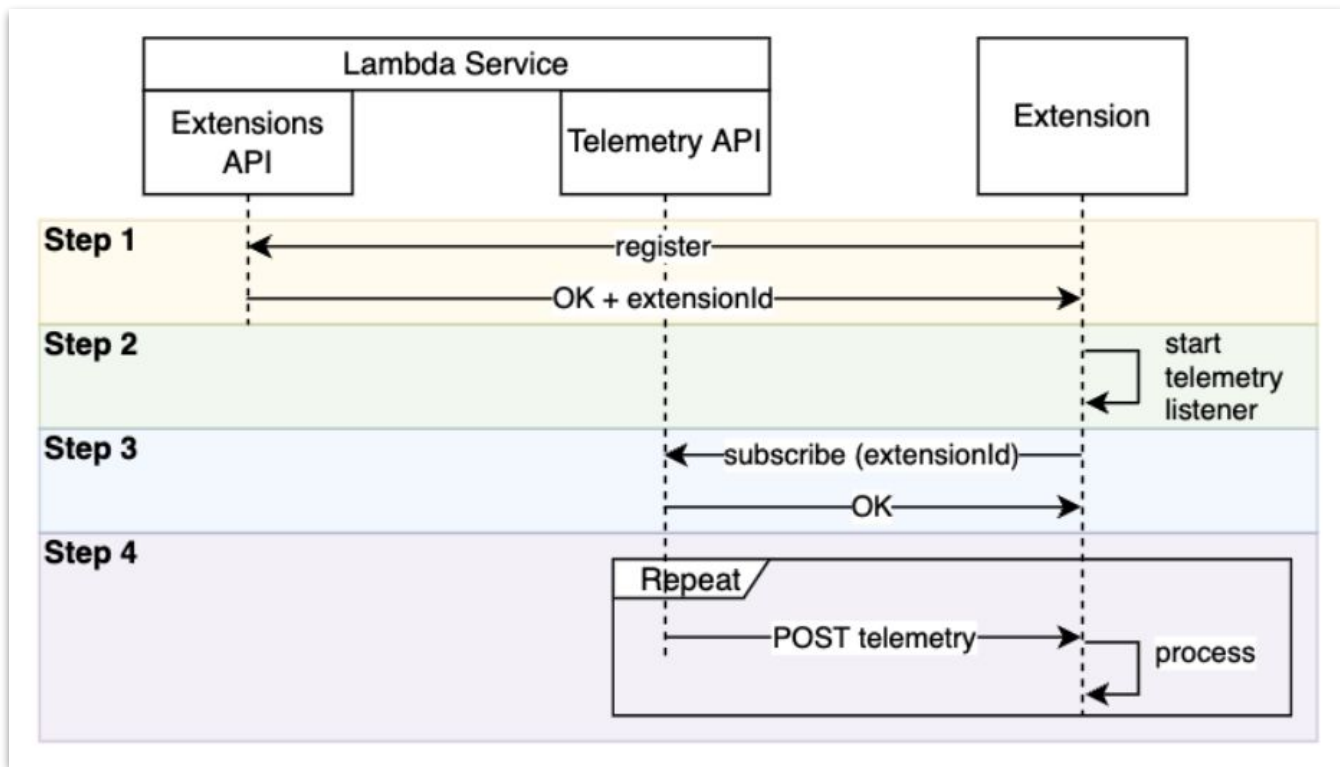
  MyLambdaFunction:
    Type: AWS::Lambda::Function
    Properties:
      FunctionName: my-lambda-function
      Runtime: python3.9
      Handler: index.handler
      Timeout: 10
      Policies:
        - AWSLambdaBasicExecutionRole
        - AWSLambda_ReadOnlyAccess
        - AWSXrayWriteOnlyAccess
      Layers:
        - !Ref MyLambdaLayer
```

— Telemetry API

- ❖ Possibilita extensões acessar dados de telemetria em tempo real;
- ❖ Logs, trace, métricas, métricas de plataforma;
- ❖ Possibilidade subscrever via extensão (infinitas possibilidades).

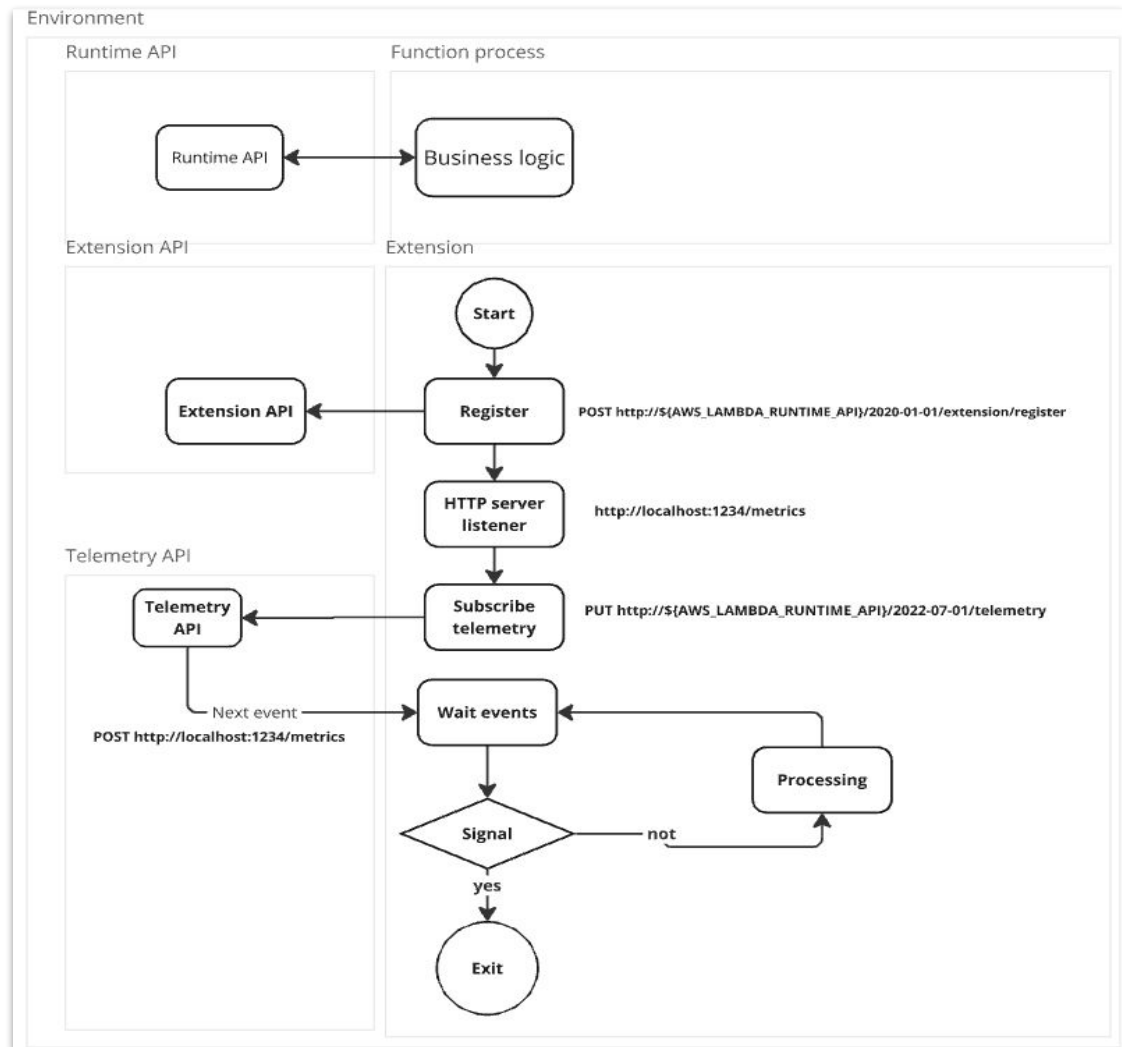


Como juntar tudo isso para criar sua layer de observabilidade?



Fonte: AWS

Como juntar tudo isso



Processamento dos eventos

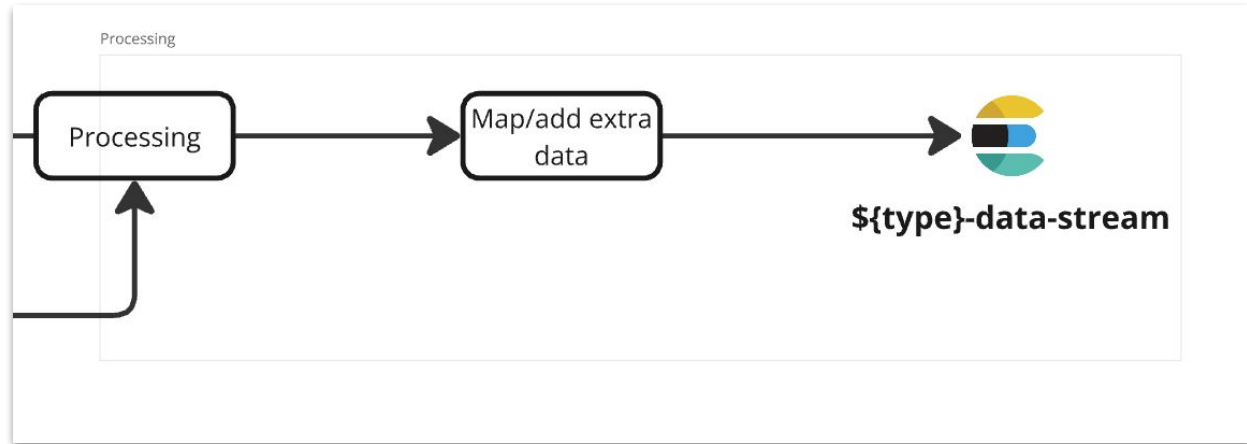
- ❖ Batches de uma ou mais execuções;

```
[
  {
    "time": "2022-10-12T00:03:50.000Z",
    "type": "function",
    "record": {
      "_aws": {
        "Timestamp": 1574109732004,
        "CloudWatchMetrics": [
          {
            "Namespace": "lambda-function-metrics",
            "Dimensions": [
              "functionVersion"
            ],
            "Metrics": [
              {
                "Name": "time",
                "Unit": "Milliseconds",
                "StorageResolution": 60
              }
            ]
          }
        ]
      },
      "functionVersion": "$LATEST",
      "time": 100,
      "requestId": "989ffbf8-9ace-4817-a57c-e4dd734019ee"
    }
  ]
}
```

```
[{
  "time": "2022-10-12T00:01:15.000Z",
  "type": "platform.report",
  "record": {
    "metrics": {
      "billedDurationMs": 694,
      "durationMs": 693.92,
      "initDurationMs": 397.68,
      "maxMemoryUsedMB": 84,
      "memorySizeMB": 128
    },
    "requestId": "6d68ca91-49c9-448d-89b8-7ca3e6dc66aa"
  }
},
{
  "time": "2022-10-12T00:03:50.000Z",
  "type": "function",
  "record": {
    "timestamp": "2022-10-12T00:03:50.000Z",
    "level": "INFO",
    "requestId": "79b4f56e-95b1-4643-9700-2807f4e68189",
    "message": "Hello world, I am a function!"
  }
}
]
```

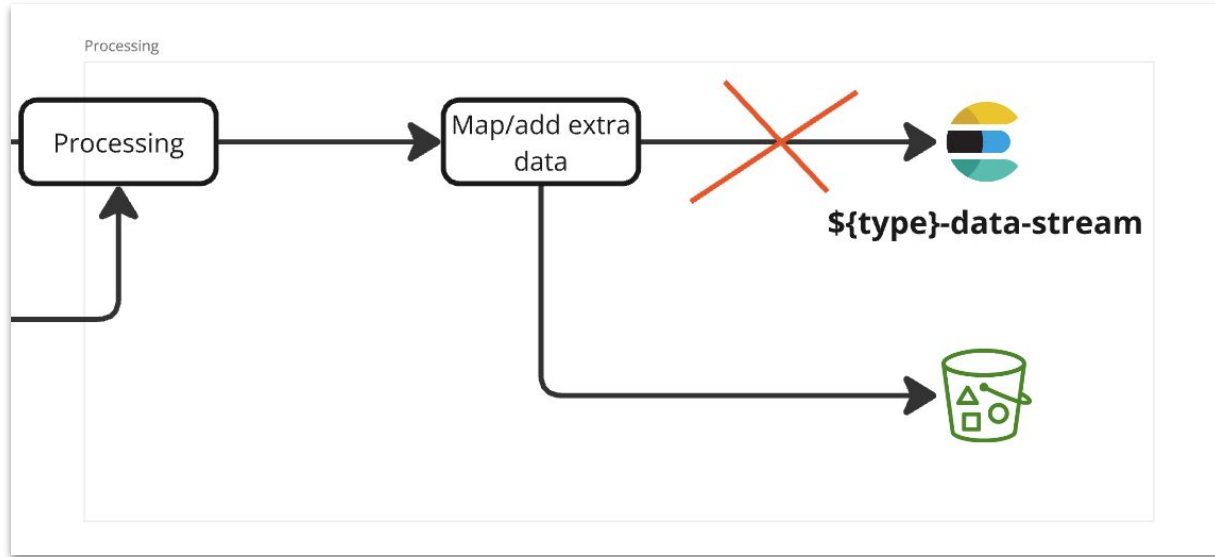
— Processamento dos eventos

- ❖ Bulk/batch add;
- ❖ Enriquecer dados com base no tipo;

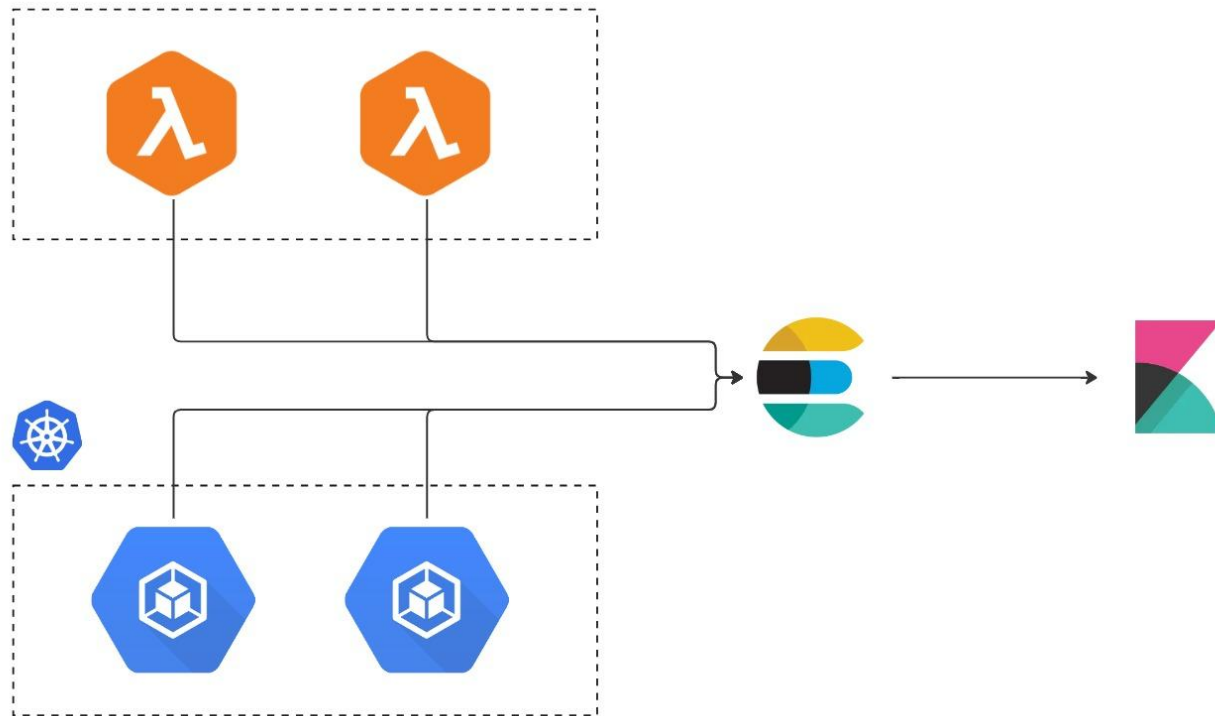


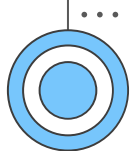
— Processamento dos eventos

- ❖ Fallbacks;
- ❖ Várias possibilidades.



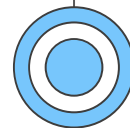
Vantagens de criar sua layer





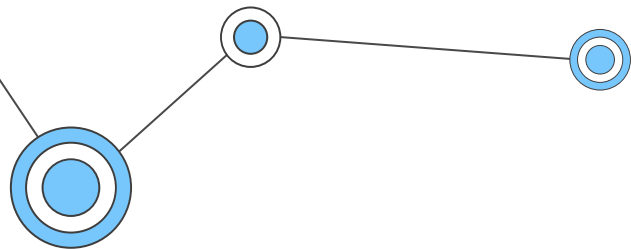
Vantagens de criar sua layer

- ❖ Log centralizado;
- ❖ Interoperabilidade entre sistemas;
- ❖ Gestão facilitada.



Referências

- ❖ <https://aws.amazon.com/lambda/sla/historical/>
- ❖ <https://docs.aws.amazon.com/lambda/latest/dg/lambda-runtime-environment.html>
- ❖ <https://docs.aws.amazon.com/lambda/latest/dg/lambda-concurrency.html>
- ❖ <https://catalog.workshops.aws/lambdaextensions/en-US>
- ❖ <https://docs.aws.amazon.com/lambda/latest/dg/runtimes-extensions-api.html>
- ❖ <https://docs.aws.amazon.com/lambda/latest/dg/chapter-layers.html>
- ❖ <https://docs.aws.amazon.com/lambda/latest/dg/telemetry-api.html>
- ❖ <https://github.com/aws-samples/aws-lambda-extensions>

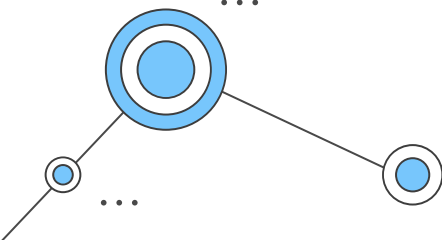


...

Perguntas?

Obrigado!

...



Aponte o celular e conecte-se :)



SCAN ME!