Serverless

Apague os servidores, foque no desenvolvimento e deixe a AWS [ou outra nuvem] cuidar do resto para você.

Trigg

Milton Bertachini Neto <milton33b@hotmail.com>

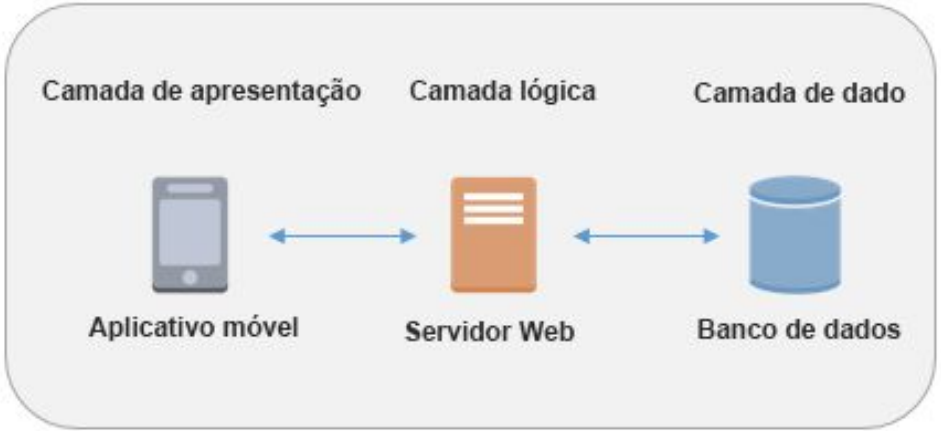
O que vamos abordar nesse encontro?

- Arquitetura cliente-servidor convencional;
- Arquitetura em containers;
- Arquitetura serverless (análise comparativa entre elas);
- Pontos Fortes;
- Pontos Fracos;
- Valores;
- Exemplo prático e interativo;
- IaC (infra as code) do exemplo.

Trigg

Arquitetura cliente-servidor convencional

Arquitetura cliente-servidor convencional



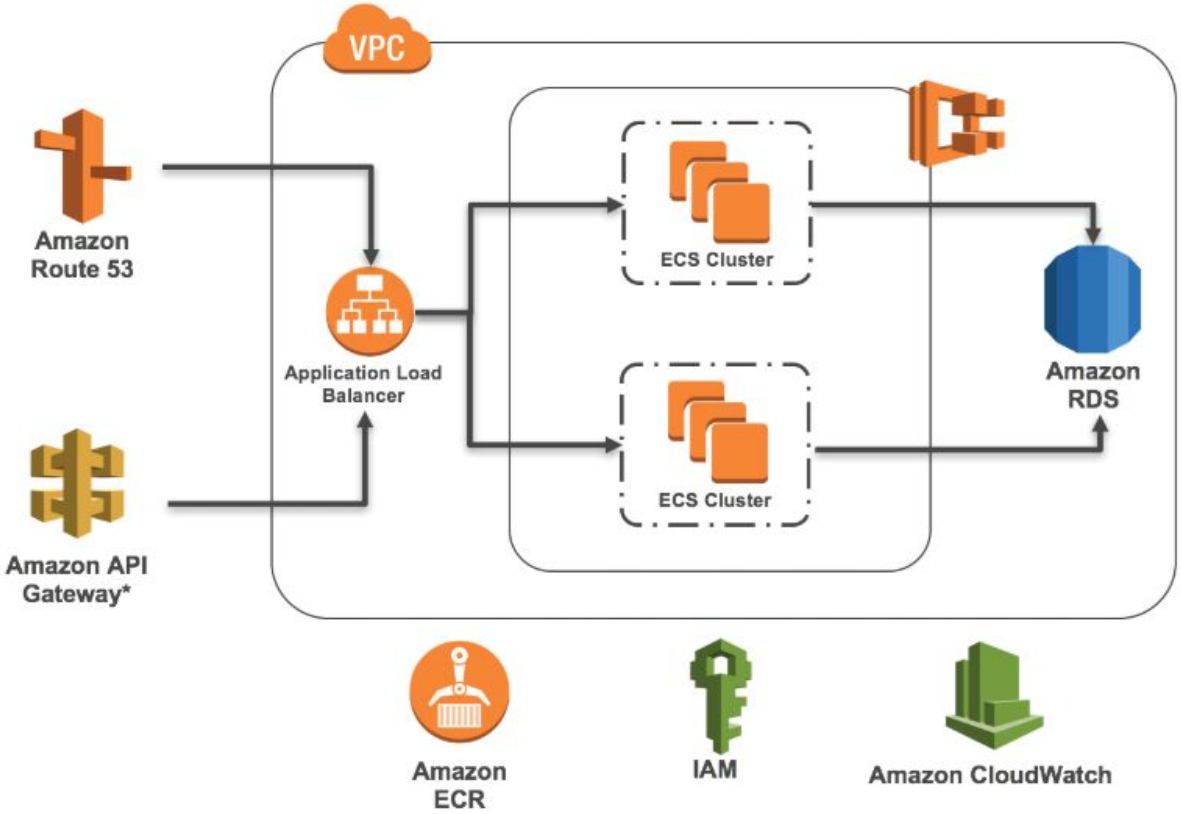
Problemas:

- Pontos únicos de falha;
- Alta ociosidade em baixa demanda;
- Baixa disponibilidade em alta demanda;

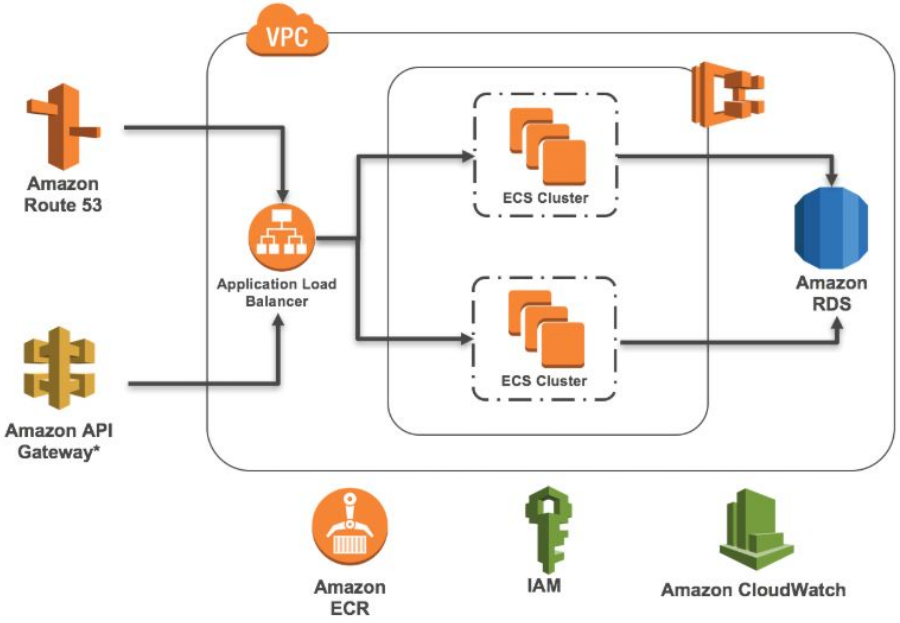
Arquitetura em containers

Com AWS

Arquitetura em containers



Arquitetura em containers



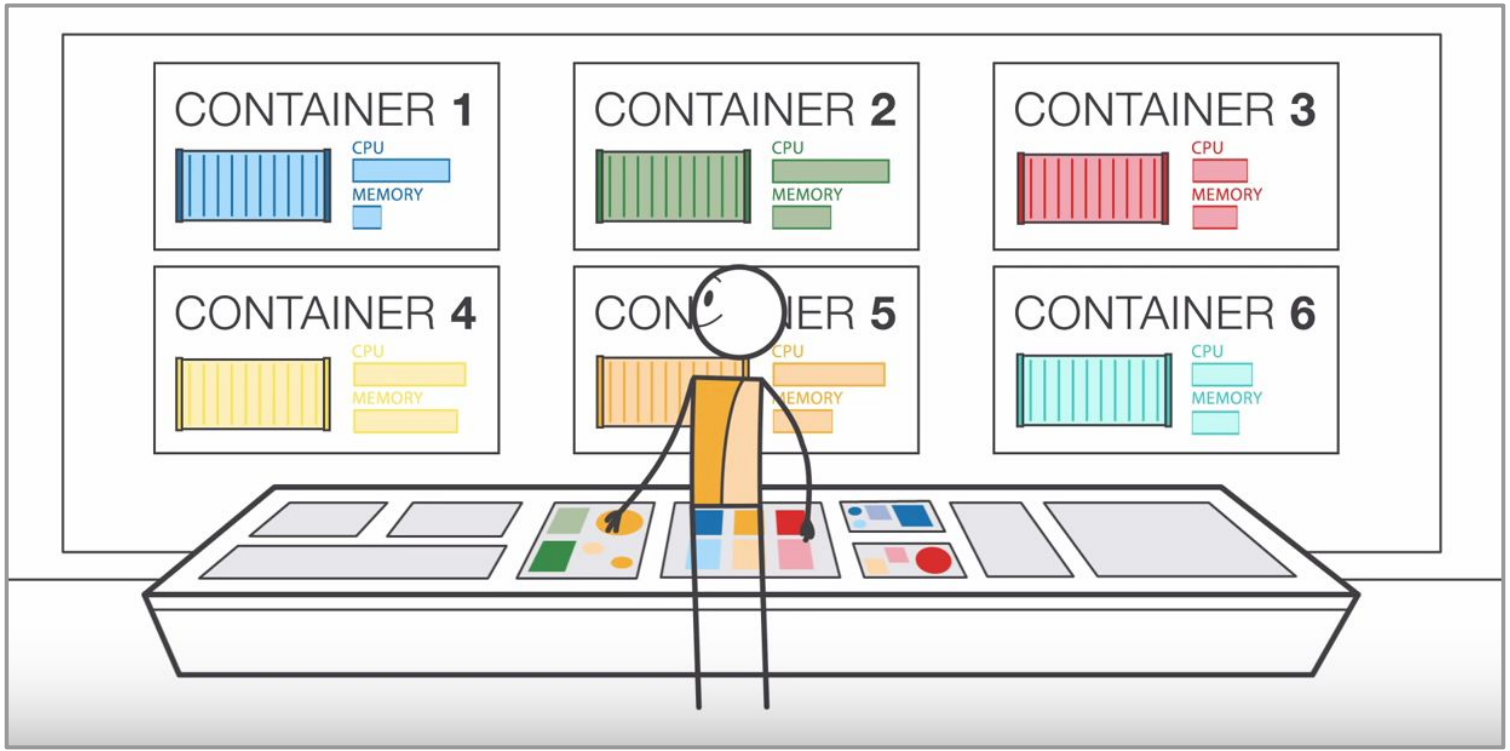
Pontos positivos

- Alta complexibilidade de construção e manutenção;
- Alto custo de máquinas e balanceadores ligados 24h;
- Arquitetura robusta e bastante utilizada pelo mercado.

Pontos Negativos

- Alta tolerância a falha;
- Fácil escalar conforme demanda;
- Baixa ociosidade (em relação ao modelo anterior).

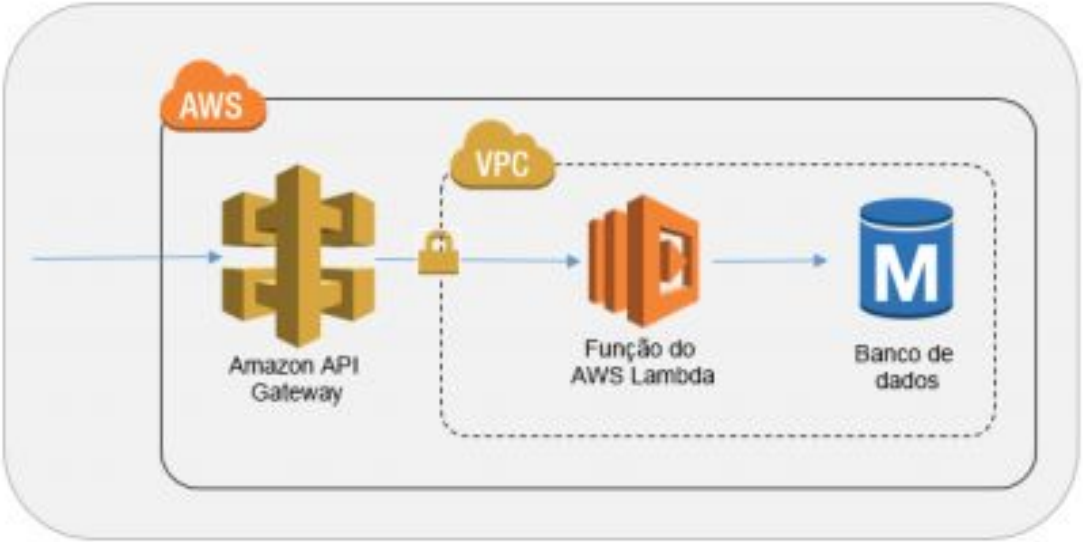
Arquitetura em containers



Arquitetura Serverless

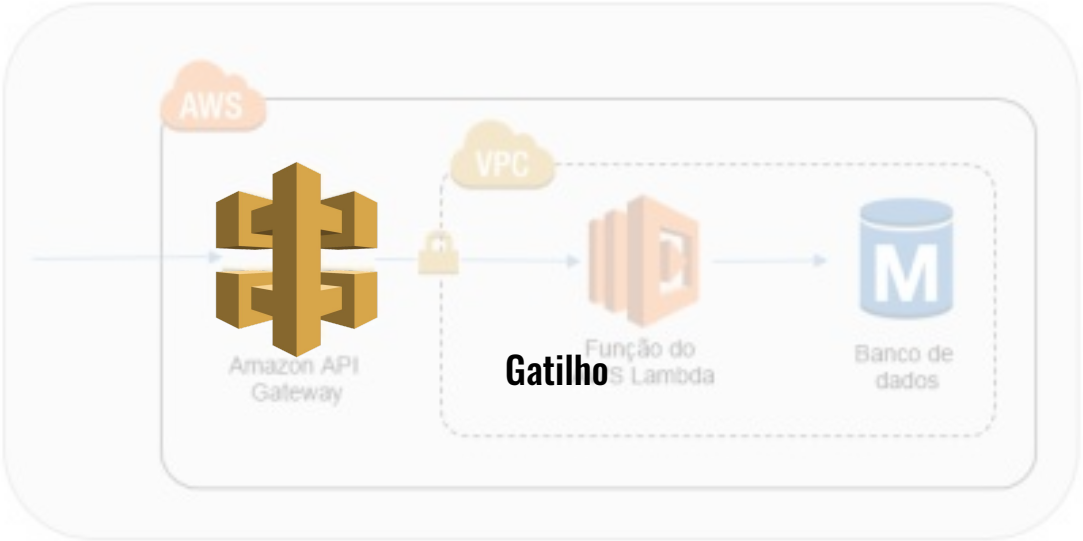
Com AWS

Arquitetura Serverless

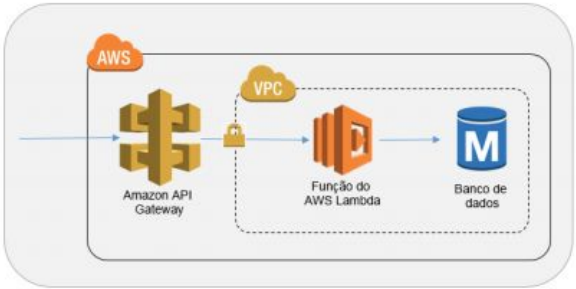


Arquitetura Serverless

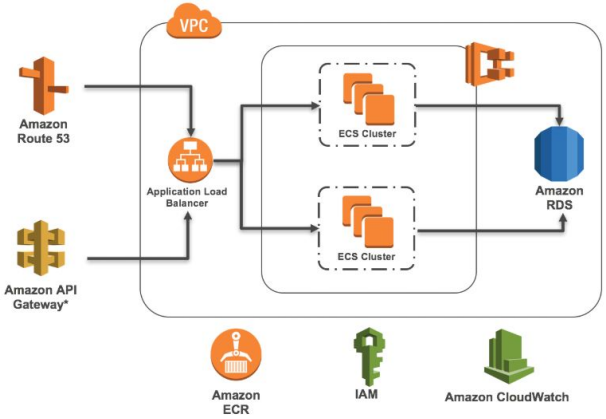
Arquitetura Serverless



Arquitetura Serverless



VS



Arquitetura Serverless

Pontos Positivos

- Integração nativa com outros produtos da nuvem;

20+

S3

Api Gateway

IAM

SQS

Arquitetura Serverless

Pontos Positivos

- Integração nativa com outros produtos da nuvem;
- Alta disponibilidade nativa;

Monthly Uptime Percentage	Service Credit Percentage
Less than 99.95% but greater than or equal to 99.0%	10%
Less than 99.0% but greater than or equal to 95.0%	25%
Less than 95.0%	100%

Compromisso comercial >= 99.95% de uptime

Arquitetura Serverless

Pontos Positivos

- Integração nativa com outros produtos da nuvem;
- Alta disponibilidade nativa;
- Escala nativa, limites iniciais robustos;

-> Limite inicial de 1.000 execuções simultâneas (esse valor pode aumentar conforme necessidade)
-> Memória disponível de 128mb até 3008mb por função
-> Até 15 minutos de limite por execução (c/ api gateway 28 seg)
-> Artefatos de até 50mb zipado
-> Armazenamento diretorio /tmp 512mb

Arquitetura Serverless

Pontos Positivos

- Integração nativa com outros produtos da nuvem;
- Alta disponibilidade nativa;
- Escala nativa, limites iniciais robustos;
- Zero gerenciamento de servidores;

Arquitetura Serverless

Pontos Positivos

- Integração nativa com outros produtos da nuvem;
- Alta disponibilidade nativa;
- Escala nativa, limites iniciais robustos;
- Zero gerenciamento de servidores;
- Zero ociosidade;

Arquitetura Serverless

Pontos Positivos

- Integração nativa com outros produtos da nuvem;
- Alta disponibilidade nativa;
- Escala nativa, limites iniciais robustos;
- Zero gerenciamento de servidores;
- Zero ociosidade;
- Pagamento conforme uso;

Arquitetura Serverless

Pontos Positivos

- Integração nativa com outros produtos da nuvem;
- Alta disponibilidade nativa;
- Escala nativa, limites iniciais robustos;
- Zero gerenciamento de servidores;
- Zero ociosidade;
- Pagamento conforme uso (+);
- Códigos mais simples pois não é necessária a utilização de frameworks tão robustos (ex spring boot);

Arquitetura Serverless

Pontos Positivos

- Integração nativa com outros produtos da nuvem;
- Alta disponibilidade nativa;
- Escala nativa, limites iniciais robustos;
- Zero gerenciamento de servidores;
- Zero ociosidade;
- Pagamento conforme uso;
- Códigos mais simples pois não é necessária a utilização de frameworks tão robustos (ex spring boot);
- Monitoramento detalhado integrado nativamente com alarmes e alertas.

Nem tudo são flores

Arquitetura Serverless

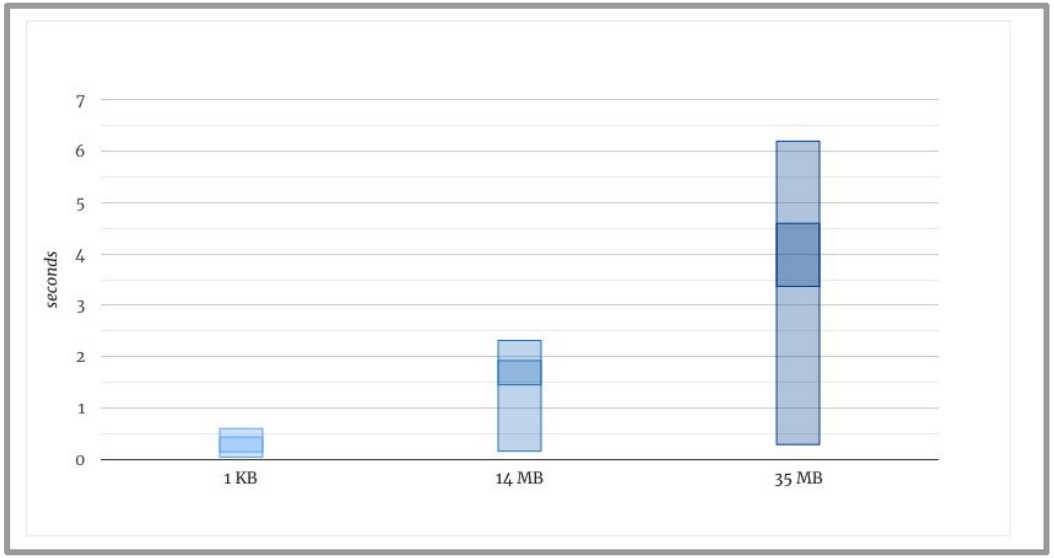
Pontos Negativos

- Alto acoplamento com o fornecedor de nuvem;

Arquitetura Serverless

Pontos Negativos

- Alto acoplamento com o fornecedor de nuvem;
- Artefatos precisam ser leves, quanto menos framework melhor;



Arquitetura Serverless

Pontos Negativos

- Alto acoplamento com o fornecedor de nuvem;
- Artefatos precisam ser leves, quanto menos framework melhor;
- Limitação de linguagens e versões;

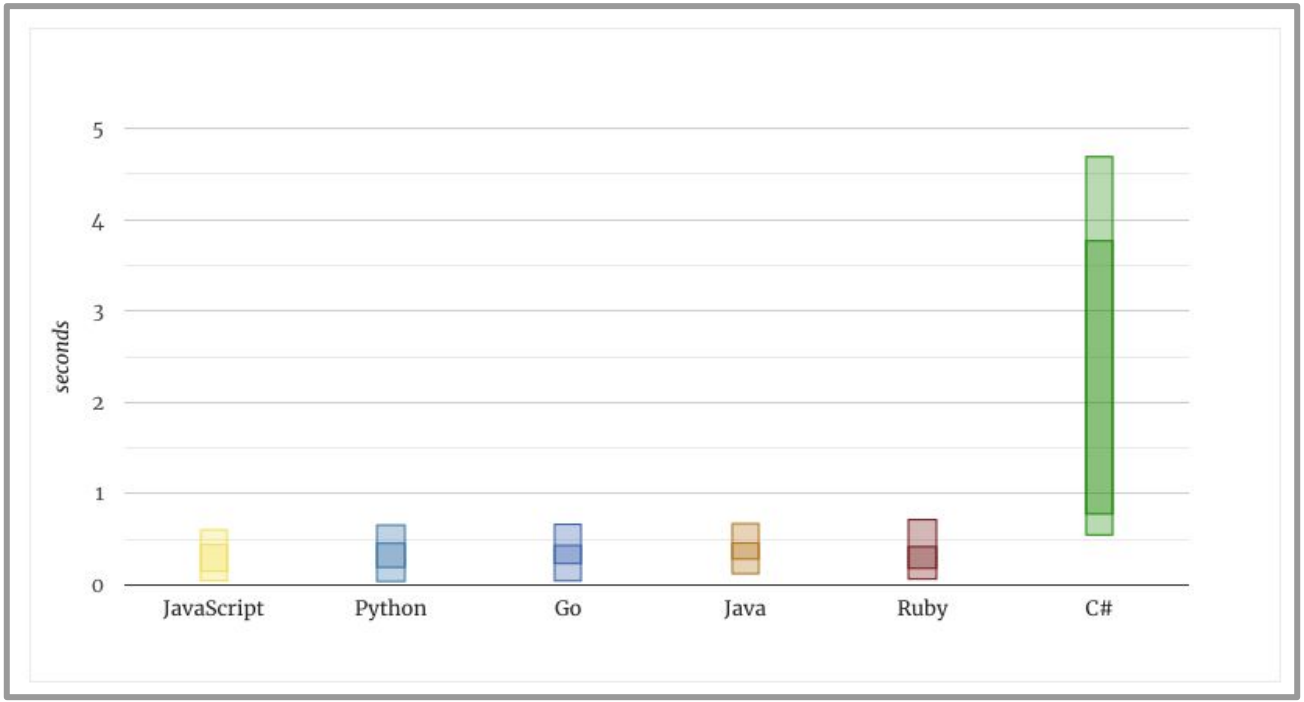
Node.js, Python, Java, Ruby, C#, Go e PowerShell

Arquitetura Serverless

Pontos Negativos

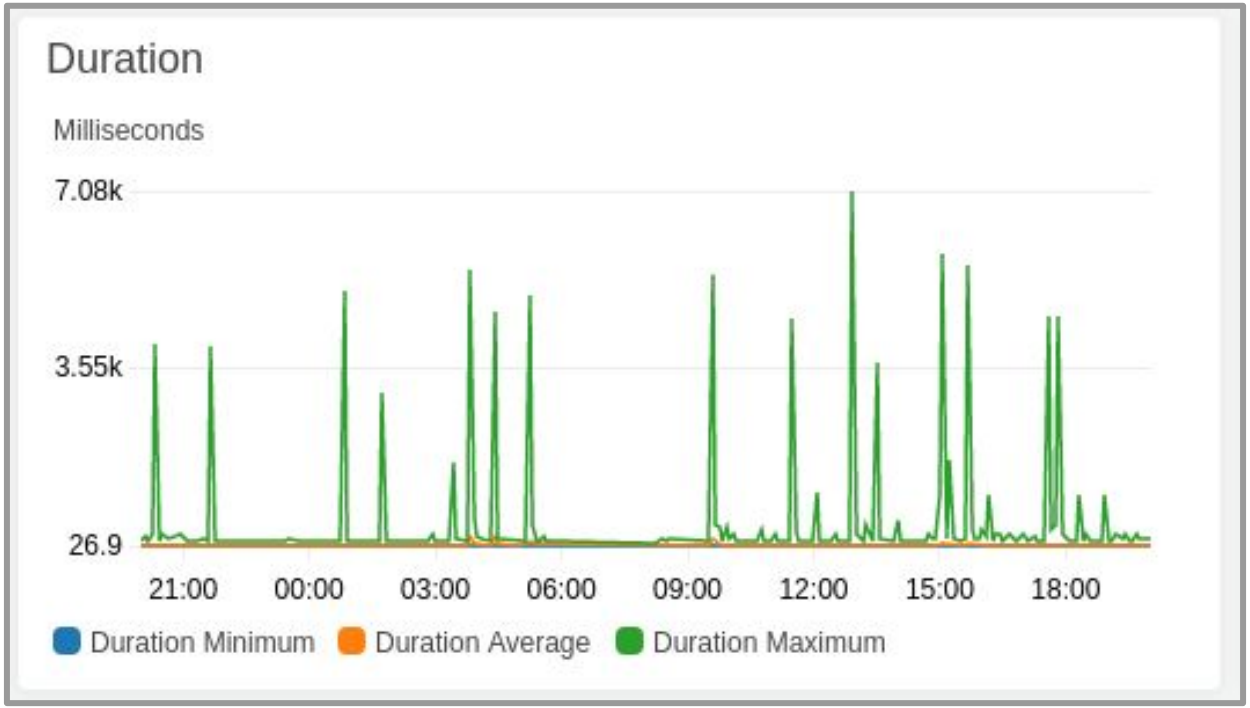
- Alto acoplamento com o fornecedor de nuvem;
- Artefatos precisam ser leves, quanto menos framework melhor;
- Limitação de linguagens e versões;
- O temido Cold Start;

Arquitetura Serverless



A mesma instrução executada em várias linguagens (cold start)

Arquitetura Serverless



Exemplo mundo real java8 (artefato zipado 13 MB)

Arquitetura Serverless

Pontos Negativos

- Alto acoplamento com o fornecedor de nuvem;
- Artefatos precisam ser leves, quanto menos framework melhor;
- Limitação de linguagens e versões;
- O temido Cold Start;
- Equipe capacitada para entregar código + infa nos deploys (devops) (+);

Cloudformation

Arquitetura Serverless

Pontos Negativos

- Alto acoplamento com o fornecedor de nuvem;
- Artefatos precisam ser leves, quanto menos framework melhor;
- Limitação de linguagens e versões;
- O temido Cold Start;
- Equipe capacitada para entregar código + infa nos deploys (devops) (+);
- Banco de dados [e demais serviços] robustos, pois não é possível trabalhar com pool de conexões

Valores (+)

Quer pagar quanto?

Arquitetura Serverless | Preços

Além do tempo gasto você ainda gasta USD 0,20 a cada 1 MM execuções

Memória (MB)	Segundos de nível gratuito por mês	Preço por 100 ms (USD)
128	3.200.000	0,000000208
192	2.133.333	0,000000313
256	1.600.000	0,000000417
320	1.280.000	0,000000521
384	1.066.667	0,000000625
448	914.286	0,000000729
512	800.000	0,000000834
576	711.111	0,000000938
640	640.000	0,000001042
704	581.818	0,000001146
768	533.333	0,000001250

~= USD10,00 a cada 10 MM de execuções

Exemplo 1

Se a função do Lambda@Edge executou 10 milhões de vezes em um mês, e executou durante 50 ms a cada vez, a cobrança seria calculada da seguinte forma:

Cobrança mensal de computação

O preço mensal de computação é 0,00000625125 USD por 128 MB-segundo

Total de computação (segundos) = 10 milhões * (0,05 s) = 500.000 segundos

Cobrança mensal de computação = 500.000 * 0,00000625125 USD = 3,13 USD

Cobrança mensal de solicitações

O preço mensal de solicitações é 0,60 USD por 1 milhão de solicitações.

Cobrança mensal de solicitações = 10 milhões * 0,6 USD/milhão = 6,00 USD

Total de cobranças mensais

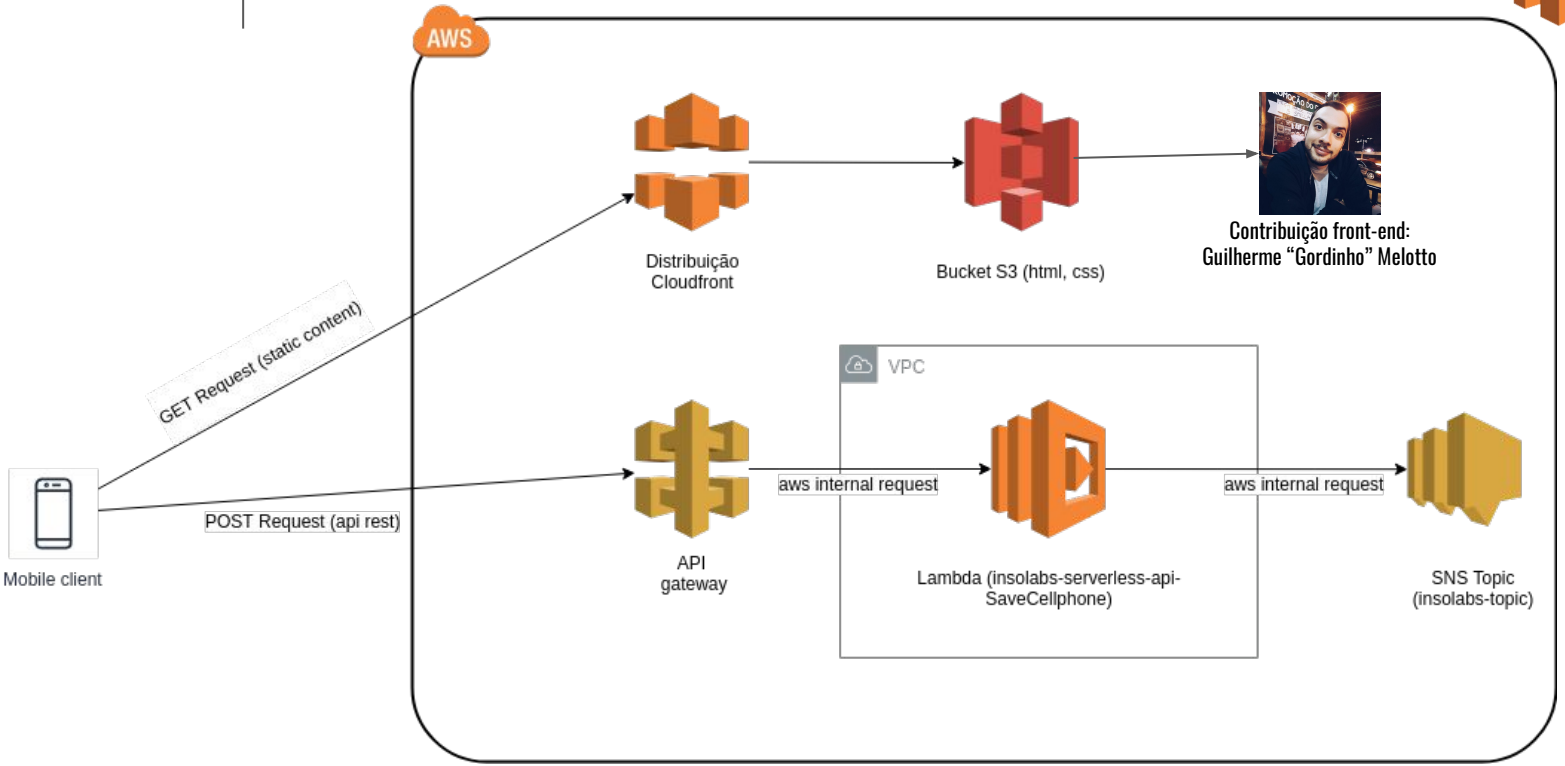
Cobrança totais = cobrança de computação + cobrança de solicitações = 3,13 USD + 6,00 USD = 9,13 USD por mês

Serverless vale a pena?

Serverless vale a pena?

https://d139c0m0hsg4pl.cloudfront.net

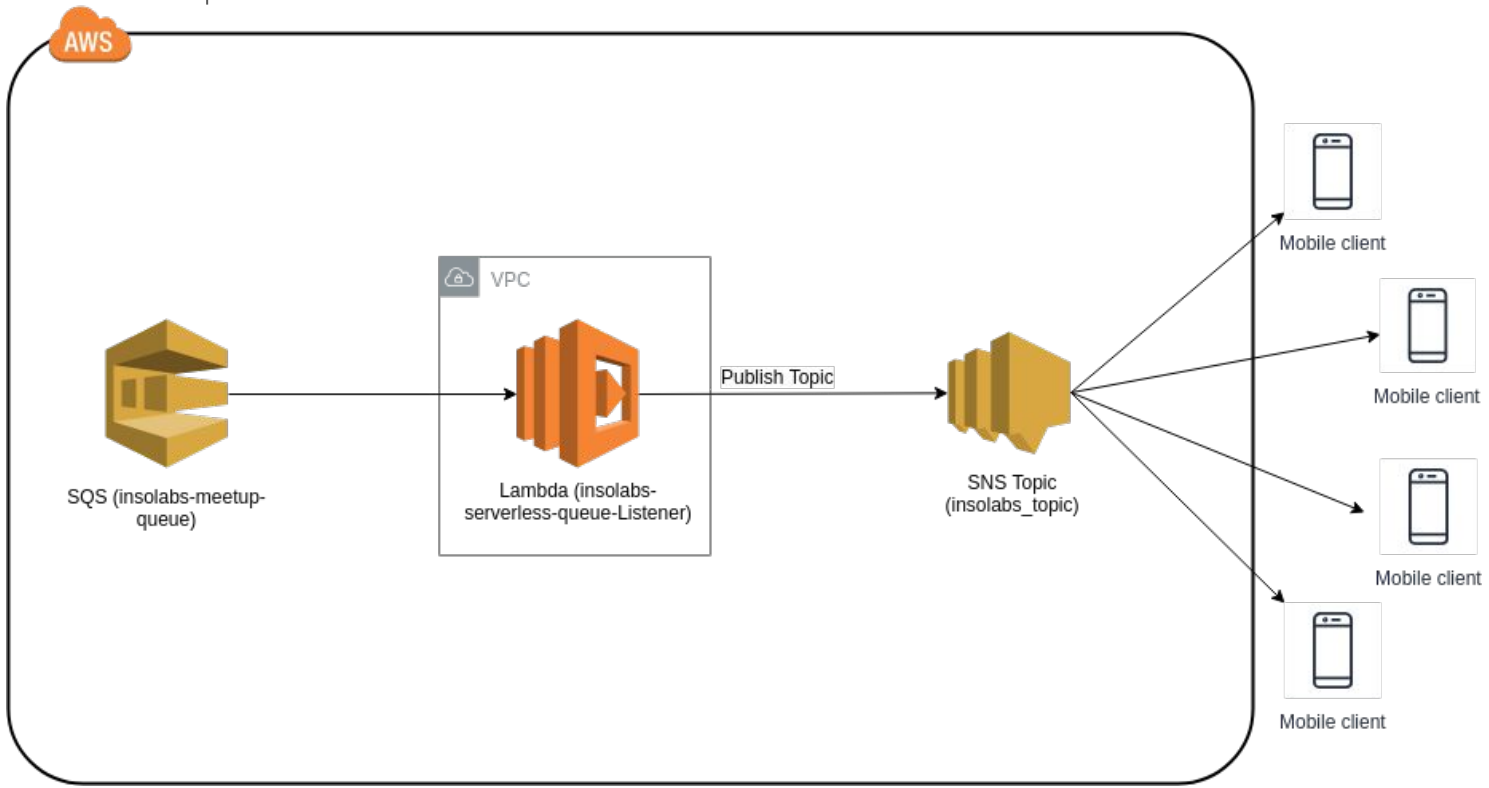
Arquitetura Serverless | Exemplificação Real



```java
public ServerlessApiResponse handleRequest(ServerlessApiRequest request, Context context) throws IOException {
    System.out.println(objectMapper.writeValueAsString(request));
    ServerlessApiResponse response = null;

    //
    //
    try {
        Client client = objectMapper.readValue(request.getBody(), Client.class);

        //
        //
        SubscribeResult subscribeResult = amazonSNS.subscribe(new SubscribeRequest()
                .withTopicArn(System.getenv("topicArn"))
                .withProtocol("SMS")
                .withEndpoint("+55" + client.getCellphone()));
        client.setId(subscribeResult.getSubscriptionArn());

        //
        //
        response = ServerlessApiResponse.builder()
                .statusCode(200)
                .body(objectMapper.writeValueAsString(client))
                .build();

    } catch(Exception exc) {
        ServerlessApiResponse.builder()
            .statusCode(500)
            .body(exc.getMessage())
            .build();
    }

    //
    //
    return response;
}
```

com.br.insolabs.serverless.handler.api.SaveCellphone

Arquitetura Serverless | Exemplificação Real



```
public void handleRequest(Object result, Context context) throws Exception {
    String jsonValue = result instanceof String ? (String) result : objectMapper.writeValueAsString(result);
    System.out.println(jsonValue);

    //
    //
    try {
        JSONObject jsonObject = new JSONObject(jsonValue);
        JSONArray records = jsonObject.getJSONArray("Records");

        for (int x = 0; x < records.length(); x++) {
            Message message= objectMapper.readValue(records.getJSONObject(x).getString("body"), Message.class);
            amazonSNS.publish(new PublishRequest().withTopicArn(System.getenv("topicArn")).withMessage(message.getText()));
        }

    } catch(Exception exc) {
        throw exc;
    }
}
```

com.br.insolabs.serverless.handler.queue.Listener

Infrastructure as Code (+)

IaC

Stack ChangeSet

Se você quer ir para o mundo serverless, precisa começar a tratar sua infraestrutura como código:

- Volte versão de um deploy defeituoso rapidamente;
- Faça o seu time pensar em como entregar soluções ponta a ponta, desde regra de negócio em código até quais permissões o sistema precisa ter dentro da infra;
- Segurança precisa ser um dever de todos e não só do “pessoal da infra”.
- Cada projeto com a sua stack de maneira organizada e nítida:

▼ Changes

The changes CloudFormation will make if you execute this change set.

Filter

Action	Logical ID	Physical ID	Resource Type
Add	APIGatewayInvokePolicy		AWS::IAM::Policy
Add	APIGatewayInvokeRole		AWS::IAM::Role
Add	ApiDeployment		AWS::ApiGateway::Deployment
Add	ApiGateway		AWS::ApiGateway::RestApi
Add	ApiGatewayMethodSaveCellphone		AWS::ApiGateway::Method
Add	ApiGatewayResourceCellphone		AWS::ApiGateway::Resource
Add	FunctionSqsSendMessageEventSourceMapping		AWS::Lambda::EventSourceMapping
Add	LambdaApiSaveCellphone		AWS::Lambda::Function
Add	LambdaApiSaveCellphoneRole		AWS::IAM::Role
Add	LambdaQueueListener		AWS::Lambda::Function
Add	LambdaQueueListenerRole		AWS::IAM::Role
Add	PublishTopicPolicy		AWS::IAM::ManagedPolicy
Add	Queue		AWS::SQS::Queue
Add	QueueListenerPolicy		AWS::IAM::ManagedPolicy
Add	SaveClientTopicPolicy		AWS::IAM::ManagedPolicy
Add	Topic		AWS::SNS::Topic

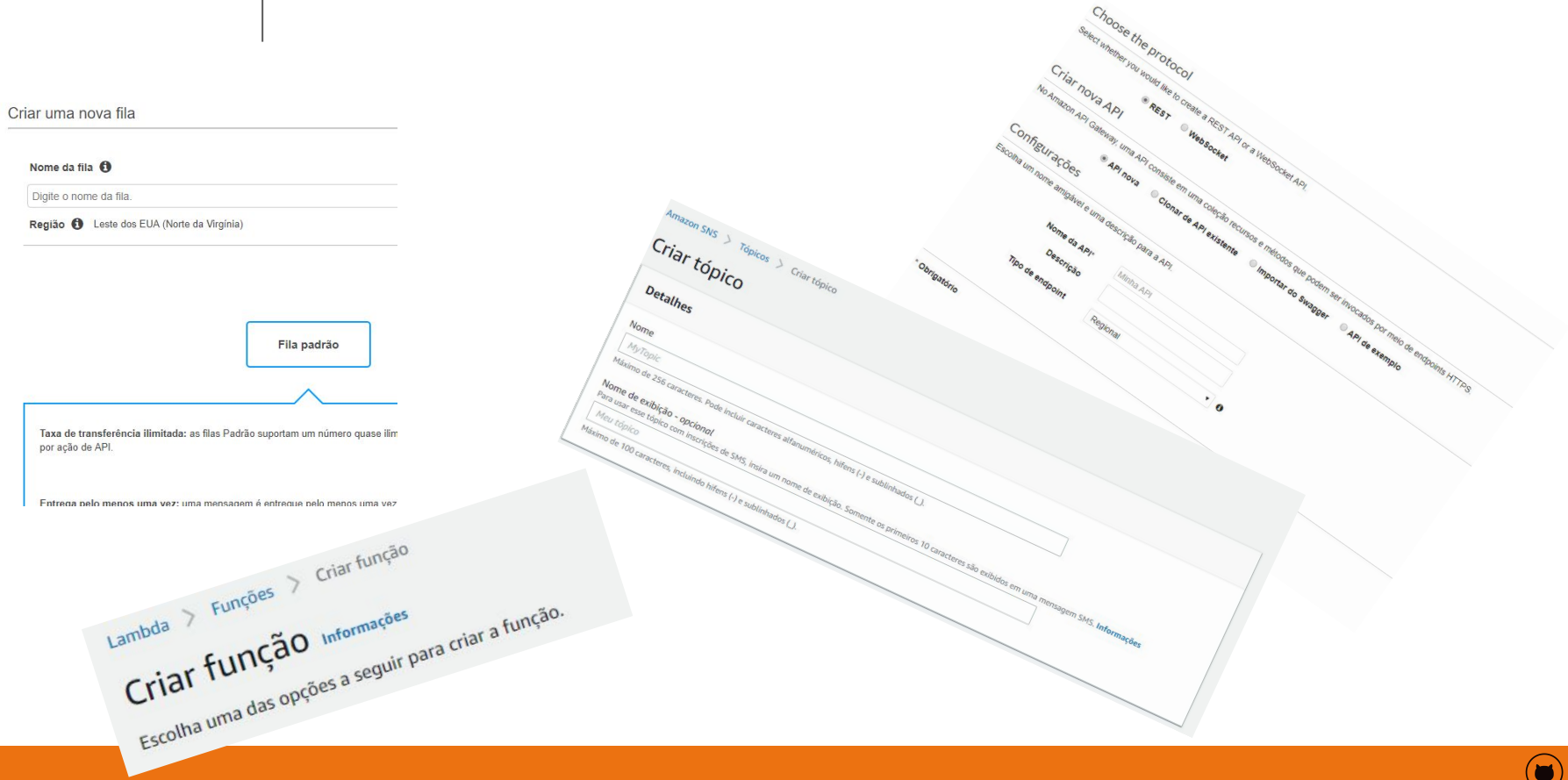
Trecho arquivo YAML

Dentro do nosso código fonte existe um arquivo chamado “template.yml” é lá dentro que toda infra necessária está sendo montada:

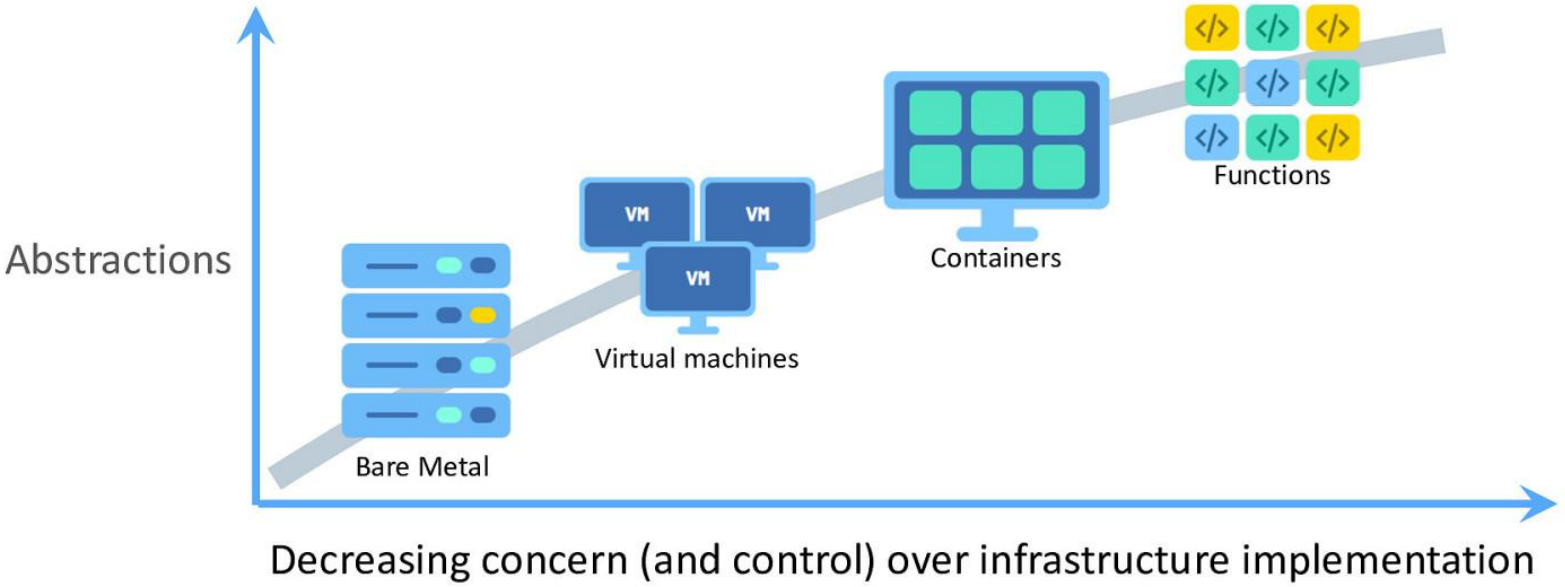
- 2 Lambdas
- Api Gateway (1 método POST)
- SQS (fila)
- SNS (tópico)
- Gerenciamento de permissões (policy)
- Gerenciamento de funções (roles)

```
LambdaApiSaveCellphone:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: build/distributions/app-0.1-SNAPSHOT.zip
    FunctionName: insolabs-serverless-api-SaveCellphone
    Handler: com.br.insolabs.serverless.handler.api.SaveCellphone::handleRequest
    Environment:
      Variables:
        topicArn: !Ref Topic
    Policies:
      - !Ref SaveClientTopicPolicy

LambdaQueueListener:
  Type: AWS::Serverless::Function
  Properties:
    Timeout: 60
    CodeUri: build/distributions/app-0.1-SNAPSHOT.zip
    FunctionName: insolabs-serverless-queue-Listener
    Handler: com.br.insolabs.serverless.handler.queue.Listener::handleRequest
    Environment:
      Variables:
        topicArn: !Ref Topic
    Policies:
      - !Ref QueueListenerPolicy
      - !Ref PublishTopicPolicy
```

Imagine ter que criar toda essa infra na 'unha'....



Trend towards Serverless



Referências

- https://aws.amazon.com/pt/lambda/
- https://mikhail.io/serverless/coldstarts/aws/
- https://aws.amazon.com/pt/api-gateway/
- https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-sam-template-basics.html
- https://aws.amazon.com/lambda/sla/

https://github.com/miltonhit/insolabs_meetup_serverless

OBRIGADO!!!