



Desafios da transição de estado em um mundo serverless

Alex Coqueiro

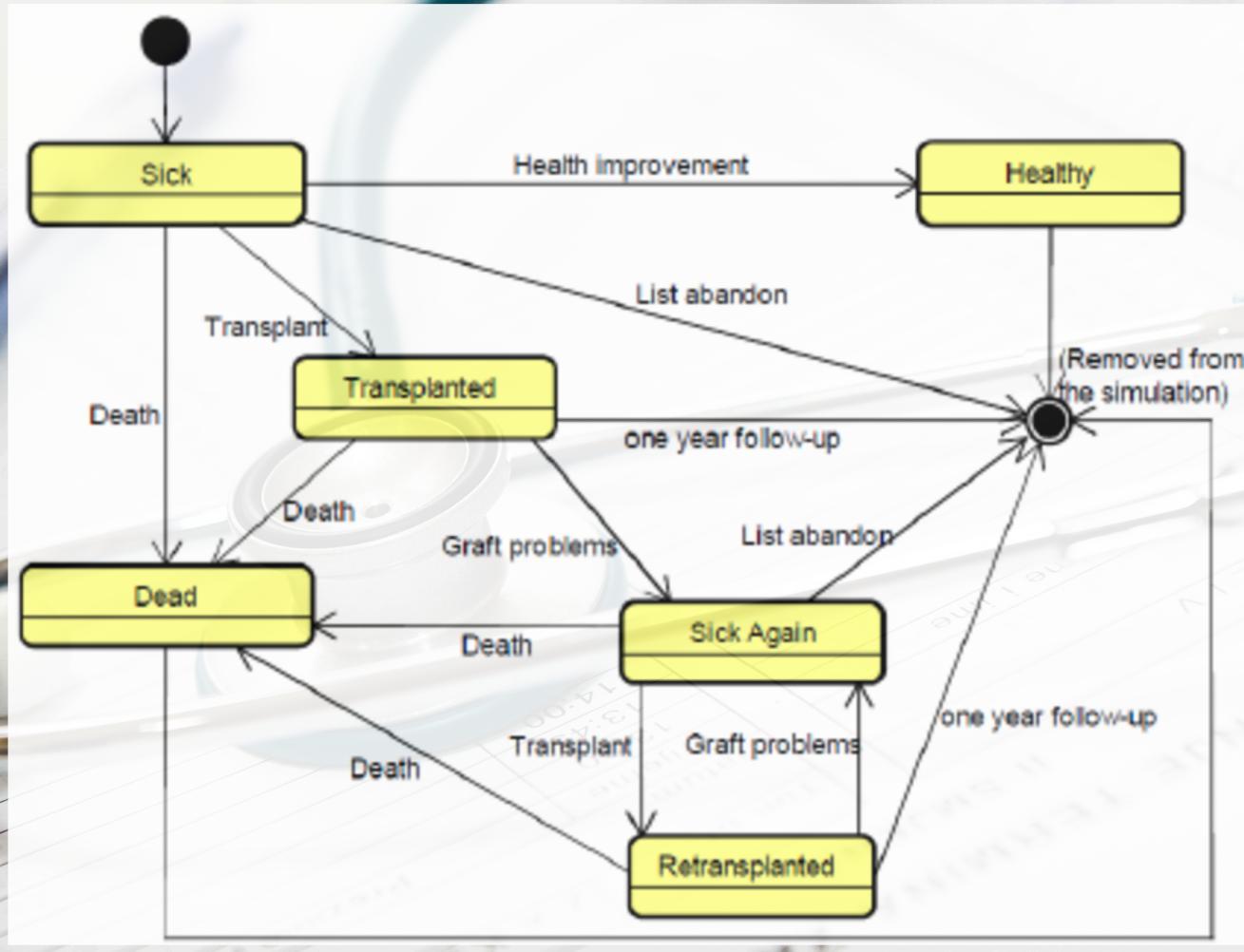
Head de Arquitetura de Soluções para América Latina, Canadá e Caribe
AWS Public Sector

 @alexbnbr

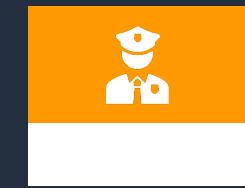
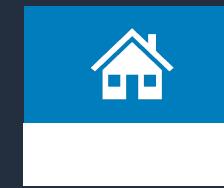
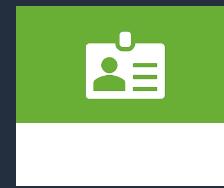
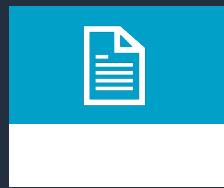
... Transição de estado em ação no dia-a-dia ...



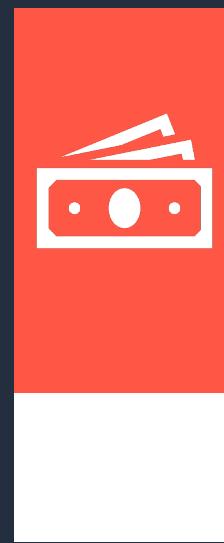
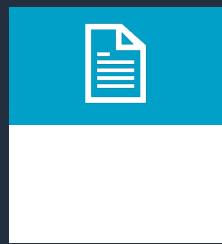
Transição de estado adiciona contexto de negócio



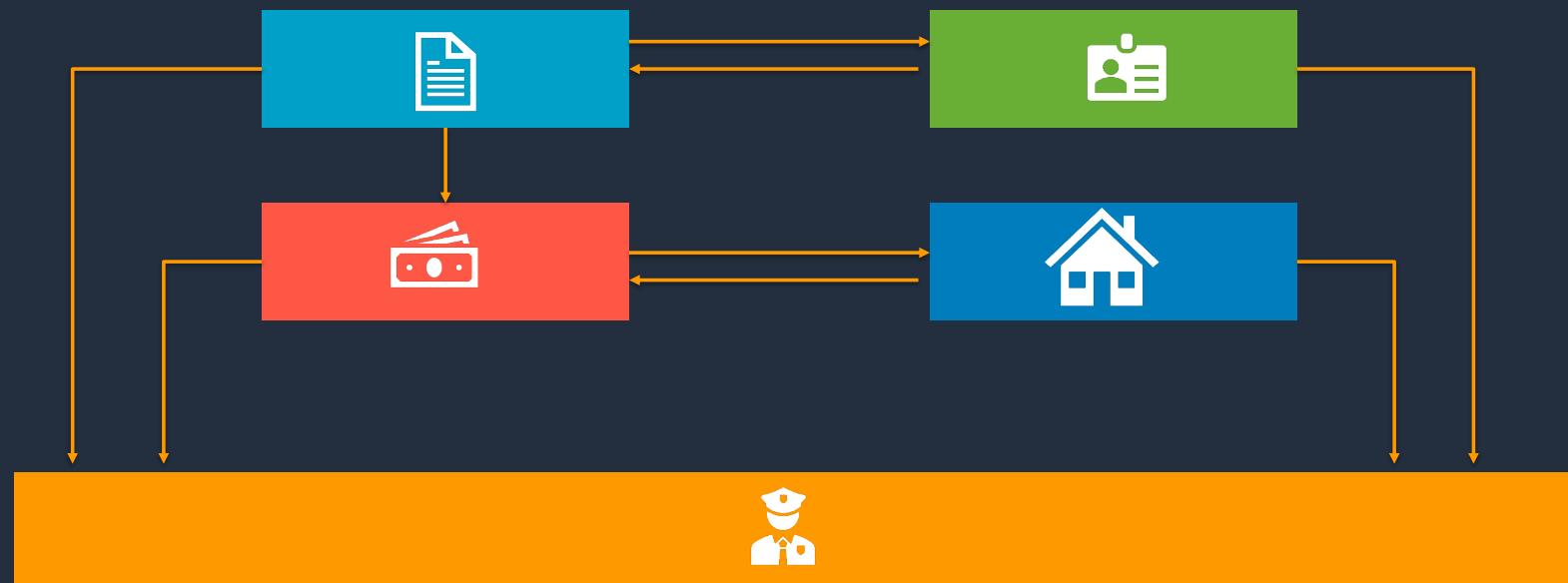
Mas como pensar em “Microserviços” & “Serverless” neste cenário



Mas como pensar em “Microserviços” & “Serverless” neste cenário



Mas agora quem é o responsável por controlar a transição de estado?



Microservices Patterns (<http://microservices.io>)

Microservice Architecture

Supported by Kong

Patterns Articles Presentations Resources Assessment Platform new Adopt new Refactoring new Testing new Other Languages

About Microservices.io

Microservices.io is brought to you by Chris Richardson. Experienced software architect, author of POJOs in Action, the creator of the original [CloudFoundry.com](#), and the author of Microservices patterns.



Chris helps clients around the world adopt the microservice architecture through consulting engagements, and training classes and workshops.

New virtual bootcamp: Distributed data patterns in a microservice architecture

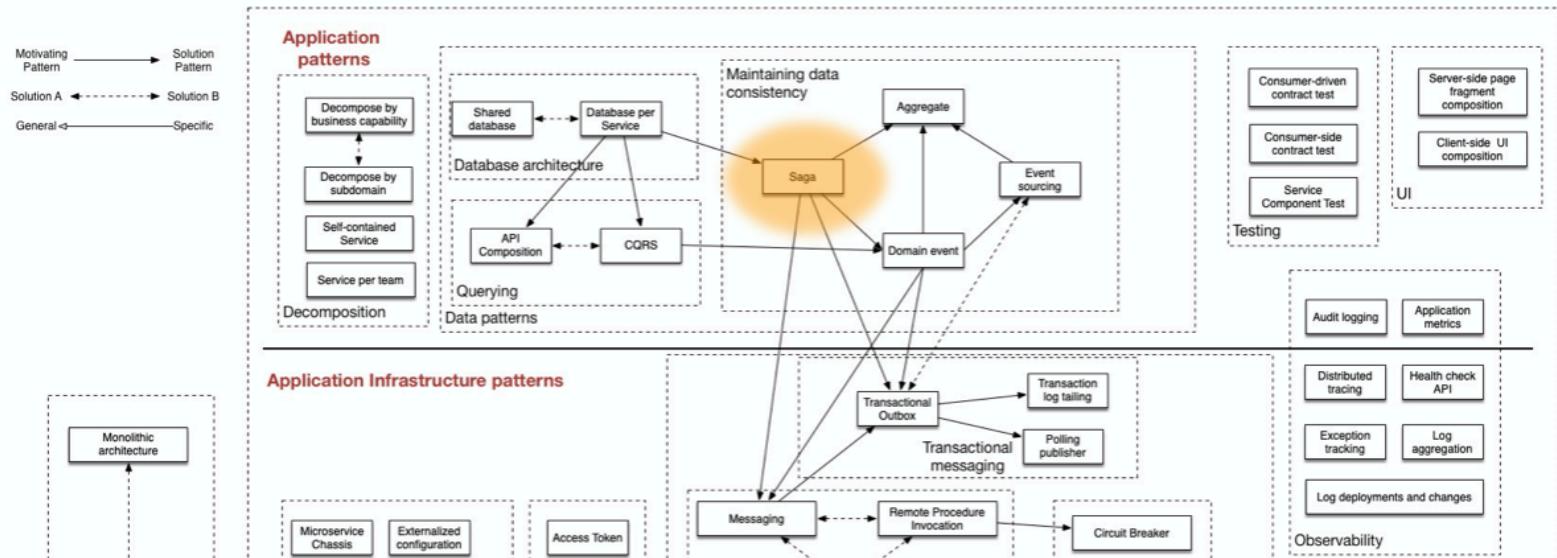


A pattern language for microservices

The beginnings of a pattern language for microservice architectures.

[点击这里，访问本系列文章的中文翻译](#)

[Click here for Chinese translation of the patterns](#)



Pattern aplicado a Transição de Estado

Saga

Event source



Result target



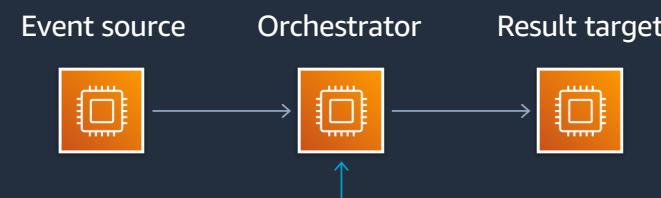
Pattern aplicado a Transição de Estado

Saga (Orquestração)



Pattern aplicado a Transição de Estado

Saga (Orquestração)

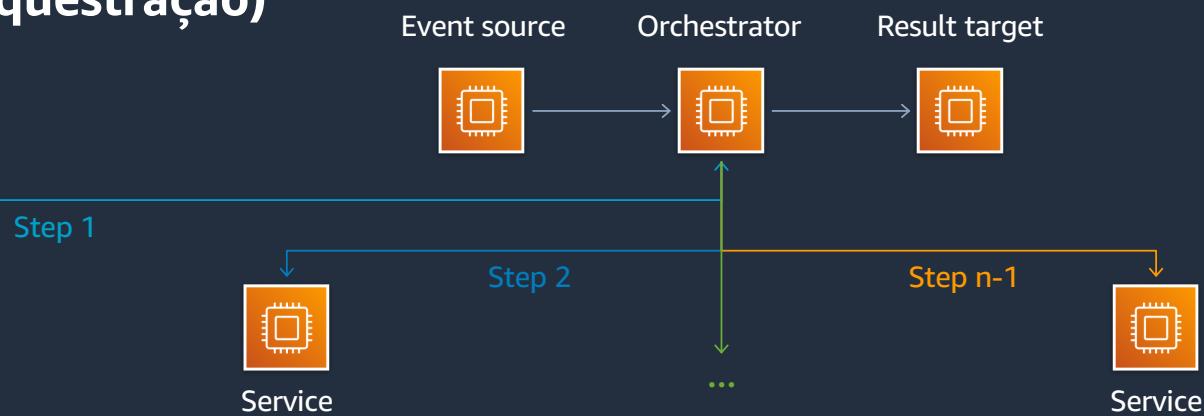


Service

Lógica externalizado pelo orquestrador (Saga Execution Coordinator)

Pattern aplicado a Transição de Estado

Saga (Orquestração)

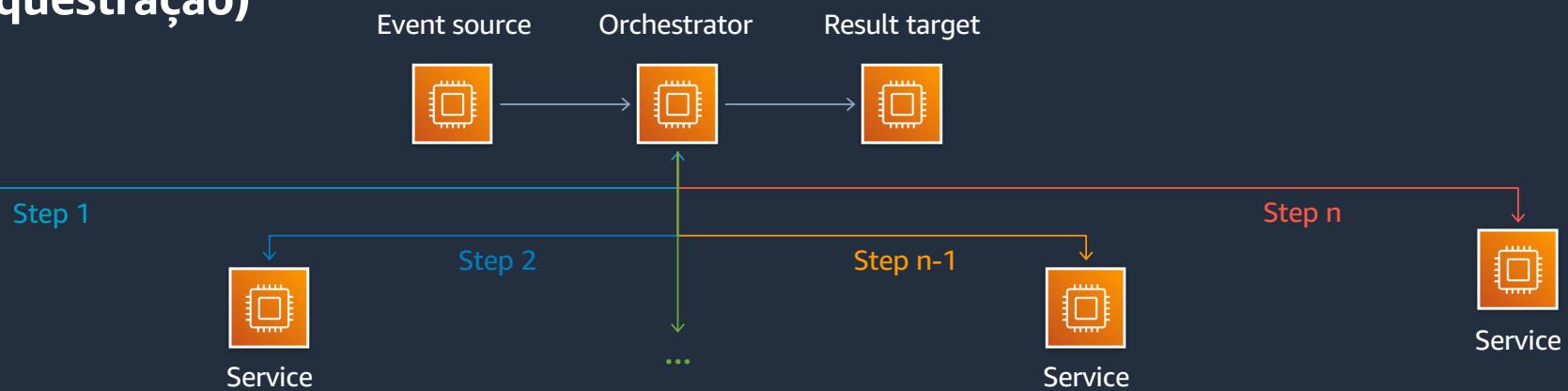


Lógica externalizado pelo orquestrador (Saga Execution Coordinator)

Participantes permanecem desacoplados mesmo em cenários de transações de longa duração

Pattern aplicado a Transição de Estado

Saga (Orquestração)



Lógica externalizado pelo orquestrador (Saga Execution Coordinator)

Participantes permanecem desacoplados mesmo em cenários de transações de longa duração

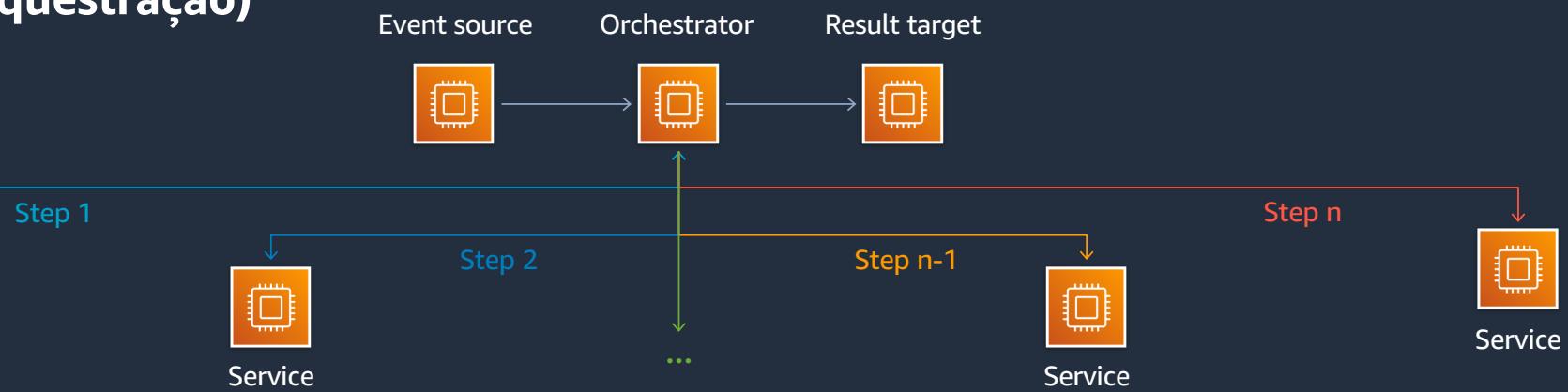
Falhas são tratadas usando o conceito de Compensation



...quero ver FUNCIONANDO...

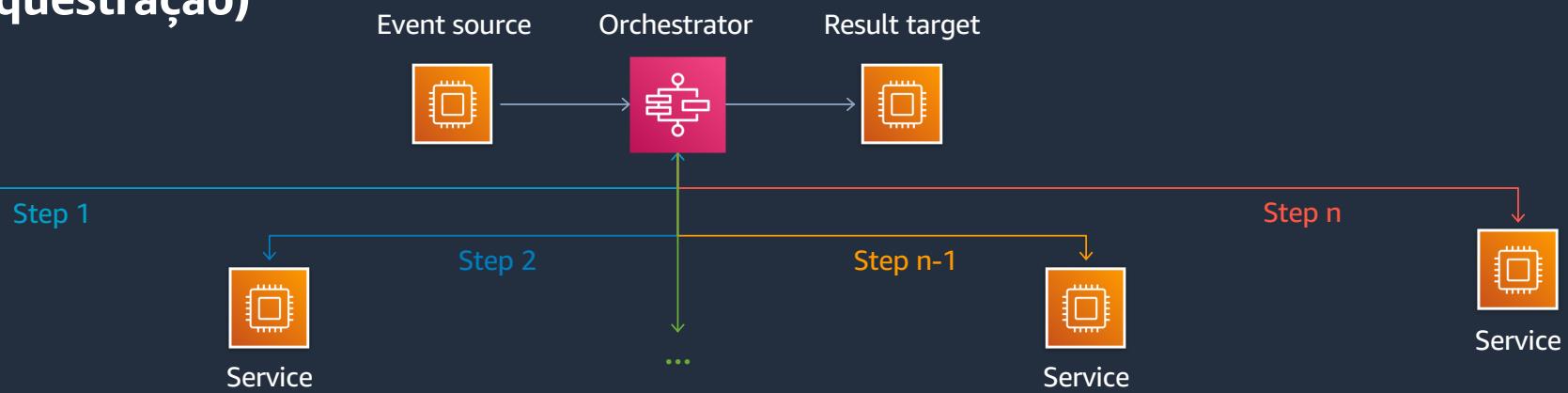
Pattern aplicado a Transição de Estado

Saga (Orquestração)



Pattern aplicado a Transição de Estado

Saga (Orquestração)



Serviço AWS para saga orchestration (serverless) >> AWS Step Functions

AWS Step Functions

Serviço gerenciado de máquina de estados na AWS

Automação de workflow resiliente

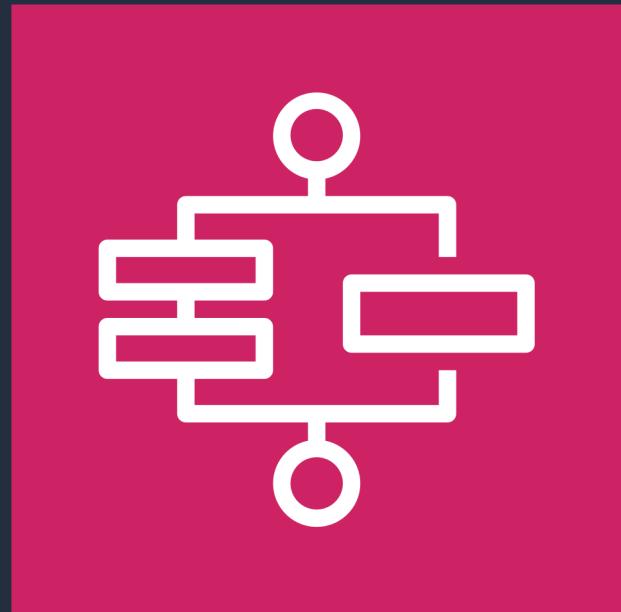
Suporte a manipulação de erros (Built-in)

Integração nativa com serviços AWS

Alta disponibilidade (HA) & escalar automaticamente

Histórico de execução auditável

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved. Amazon Confidential and Trademark



Caso de Uso: Triagem de pessoas infectadas pelo COVID-19



© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved. Amazon Confidential and Trademark

Fonte: Bloomberg / Getty Images



Operando com o AWS Step Functions

Definição em JSON

```
1  {
2    "Comment": "Manage opening an account",
3    "StartAt": "Perform Automated Checks",
4    "States": {
5      "Perform Automated Checks": {
6        "Type": "Parallel",
7        "Branches": [
8          {
9            "StartAt": "Check Identity",
10           "States": {
11             "Check Identity": {
12               "Type": "Task",
13               "Parameters": {
```

Monitorar Execuções

Visual workflow

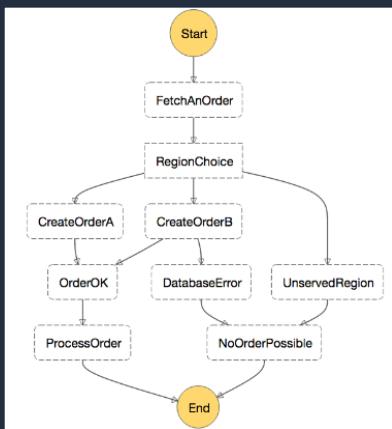
Code | Step details

Name	Automated Checks Choice	Type	Choice
Status	Succeeded		
Resource	-		
▶ Input			
▶ Output			
▶ Exception			

Execution event history

ID	Type	Step	Resource	Elapsed Time (ms)	Timestamp
▶ 1	ExecutionStarted		-	0	Sep 17, 2019 11:14:14.027 AM
▶ 2	ParallelStateEntered	Perform Automated Checks	-	41	Sep 17, 2019 11:14:14.068 AM
▶ 3	ParallelStateStarted	Perform Automated Checks	-	41	Sep 17, 2019 11:14:14.068 AM
▶ 4	TaskStateEntered	Check Identity	-	144	Sep 17, 2019 11:14:14.171 AM
▶ 5	LambdaFunctionScheduled	Check Identity	Lambda CloudWatch logs	144	Sep 17, 2019 11:14:14.171 AM
▶ 6	PassStateEntered	Check Fraud Model	-	157	Sep 17, 2019 11:14:14.184 AM

Visualizar a Console



© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved. Amazon Confidential and Trademark



Amazon States Language

<https://states-language.net/spec.html>

```
{  
  "Comment": "Verificar coleta de temperatura",  
  "StartAt": "coletar-temperatura",  
  
  "States": {  
    " coletar-temperatura ": {  
      "Type": "Task",  
      "Resource": "arn:aws:lambda...Triagem",  
      "End": true  
    },  
    [..]  
  }  
}
```



Adição de Lógica a Transição de Estado



Tasks



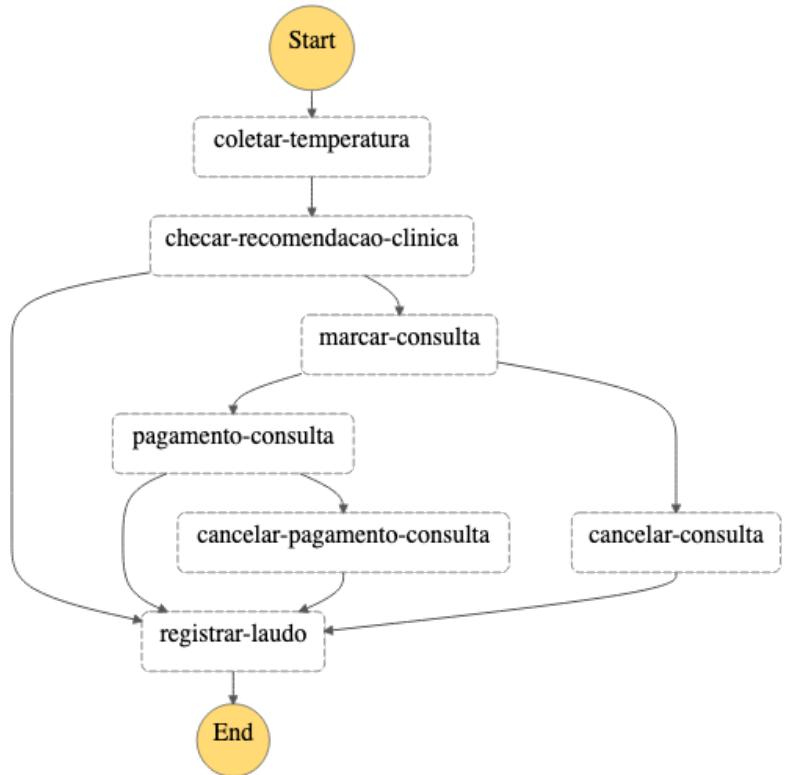
Parallel Steps



Branching Choice



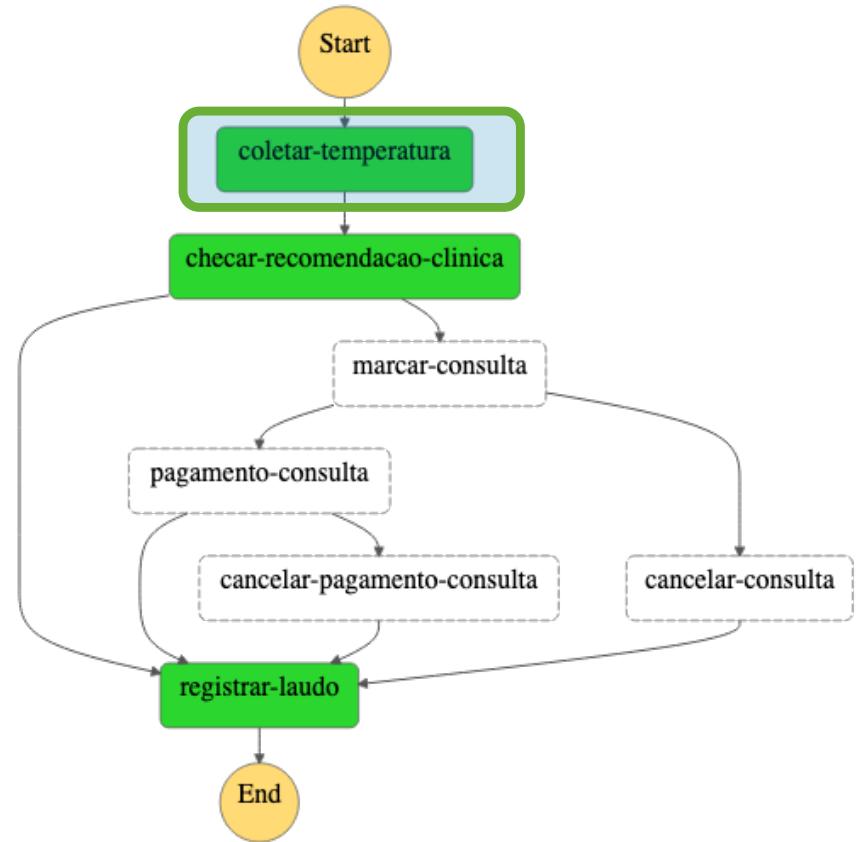
Wait for a callback





Executando uma Tarefa

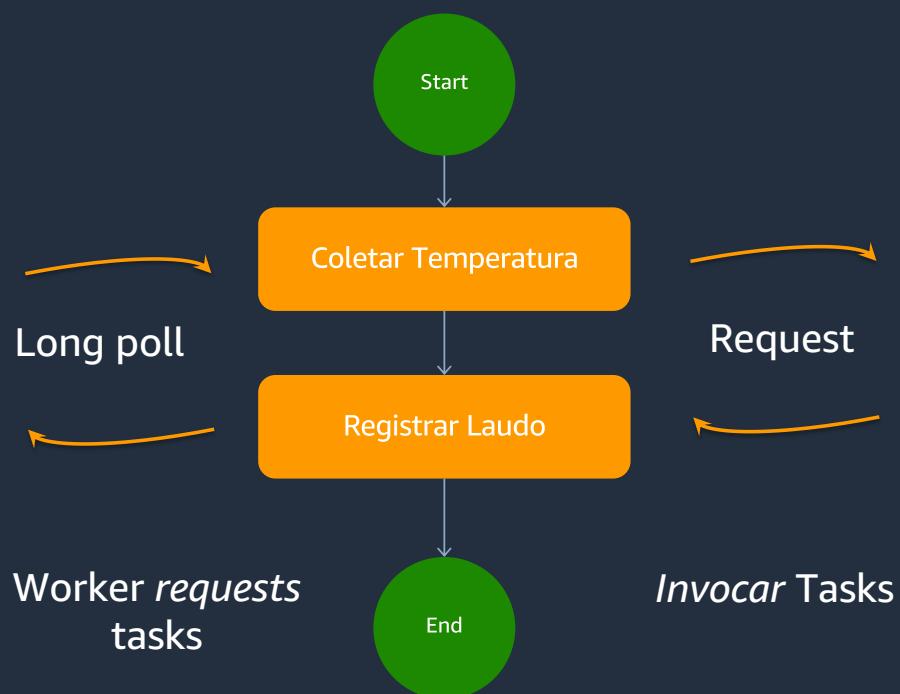
```
"coletar-temperatura": {  
    "Type": "Task",  
    "Resource": "arn:aws:lambda...triagem",  
    "Next": "checar-recomendacao-clinica"  
}
```



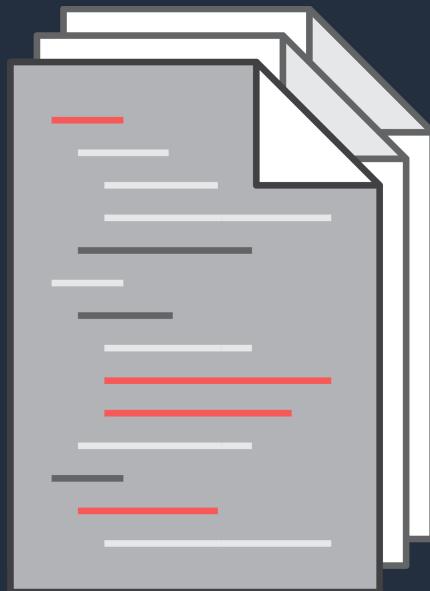
A photograph of three people in a modern office conference room. Two men and one woman are seated around a light-colored wooden conference table. The man on the left is wearing a dark blazer over a checkered shirt. The woman in the center has curly hair and is wearing a tan ribbed sweater. The man on the right is wearing glasses and a dark blazer over a blue shirt. On the table, there are two glasses of water, a bottle of white wine, a small black smartphone, a white bowl filled with green apples, and a black computer mouse. In the background, large windows offer a view of a city skyline at night. A large, semi-transparent orange text overlay reads "Que tal dar uma turbinada neste exemplo".

Que tal dar uma turbinada
neste exemplo

Integração com outros serviços na nuvem



Automação. Escolha o que melhor adequa-se a sua realidade



CHALICE



AWS CloudFormation



SAM



CDK



serverless



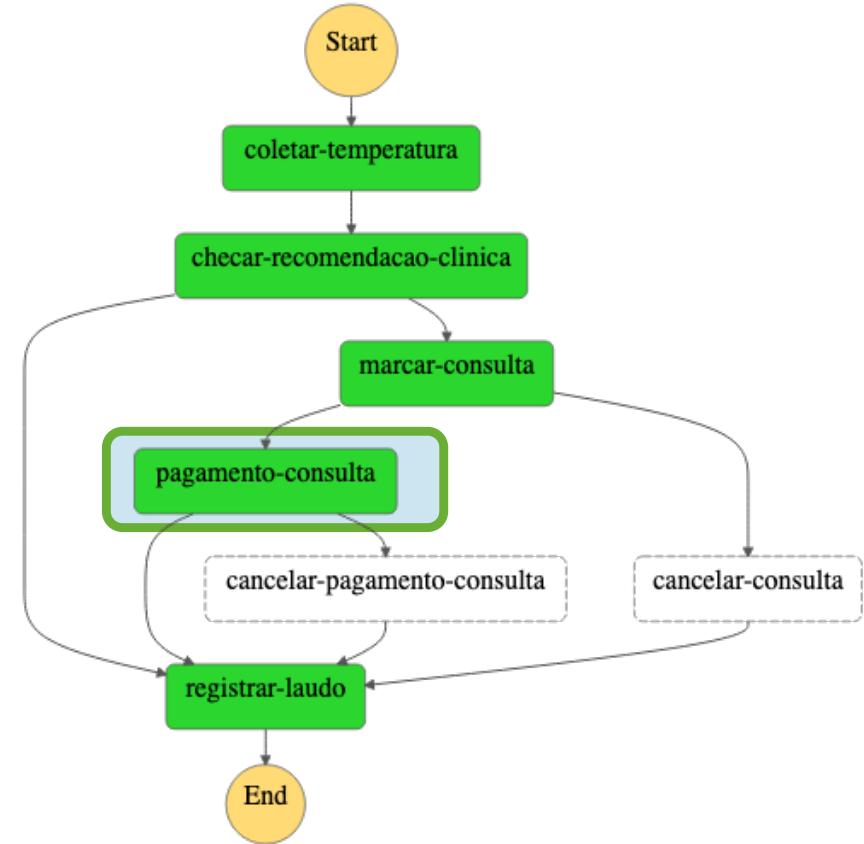
HashiCorp
Terraform





Uso de Retry e Compensation

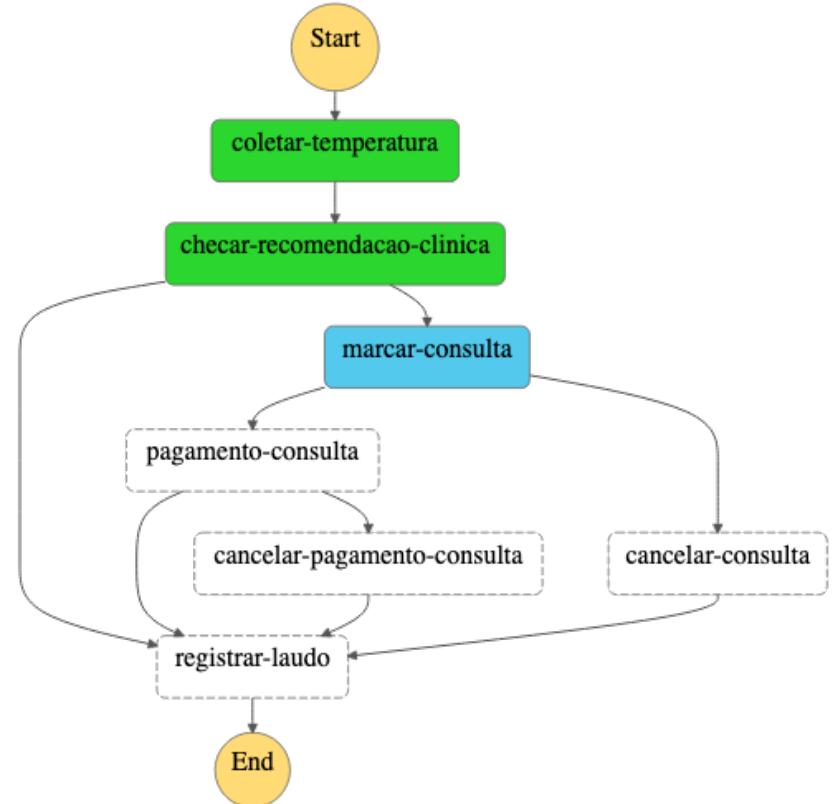
```
"marcar-consulta": {  
    "Type": "Task",  
    "Resource": "${MarcarConsultaFunctionArn}",  
    "ResultPath": "$.laudo",  
    "Next": "pagamento-consulta",  
    "Retry": [  
        {  
            "ErrorEquals": [  
                "States.ALL"  
            ],  
            "IntervalSeconds": 1,  
            "MaxAttempts": 3  
        }  
    ],  
    "Catch": [ {  
        "ErrorEquals": ["States.All"],  
        "Next": "cancelar-consulta"  
    }]
```



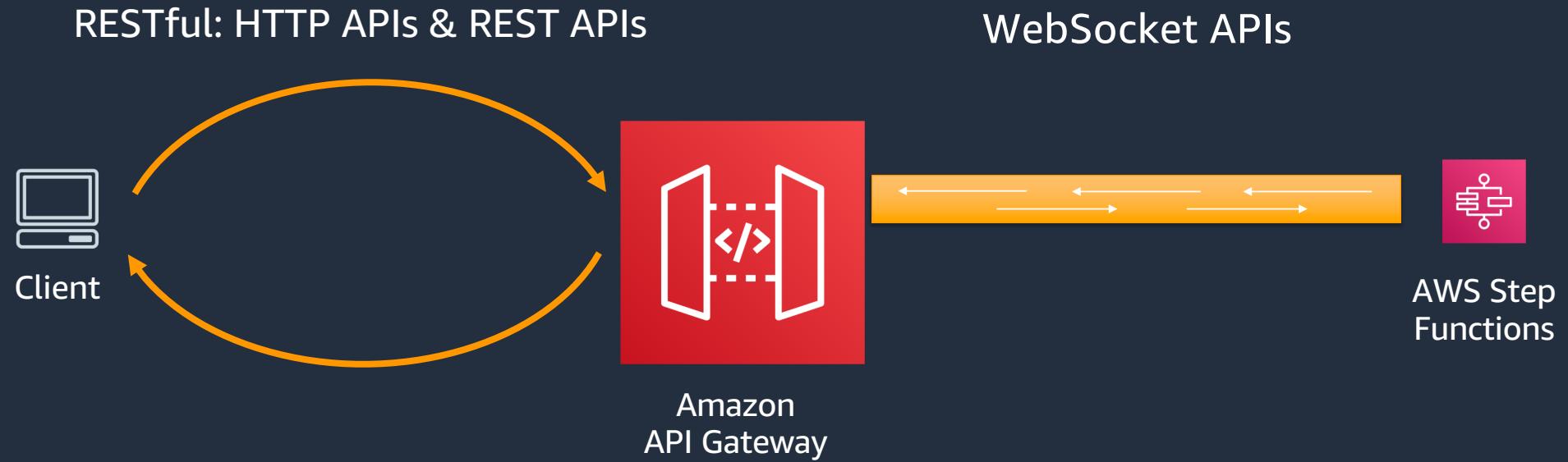


Aguardando callback no SQS

```
"Type": "Task",
"Resource": "arn:aws:states:::sns:publish",
"Parameters": {
    "QueueUrl": "https://sns.us-east-1.amazonaws.com/123/snsqueue",
    "MessageBody": {
        "Paciente": "$.paciente-id",
        "TaskToken.$": "$$.Task.Token"
    }
},
"ResultPath": "$.laudo",
"Next": "pagamento-consulta"
```



E no final tudo vira API com API Gateway ...



- Request / Response
- HTTP Methods like GET, POST, etc
- Short-lived communication
- Stateless

- Serverless WebSocket
- 2 way communication channel
- Long-lived communication
- Stateful



Em um mundo Cloud, considere
seriamente desacoplar a
transição de estado como parte
chave da sua arquitetura

Dicas

AWS Step Functions

<https://docs.aws.amazon.com/step-functions/latest/dg/sfn-local.html>

AWS blogs e outros conteúdos para integração de aplicações

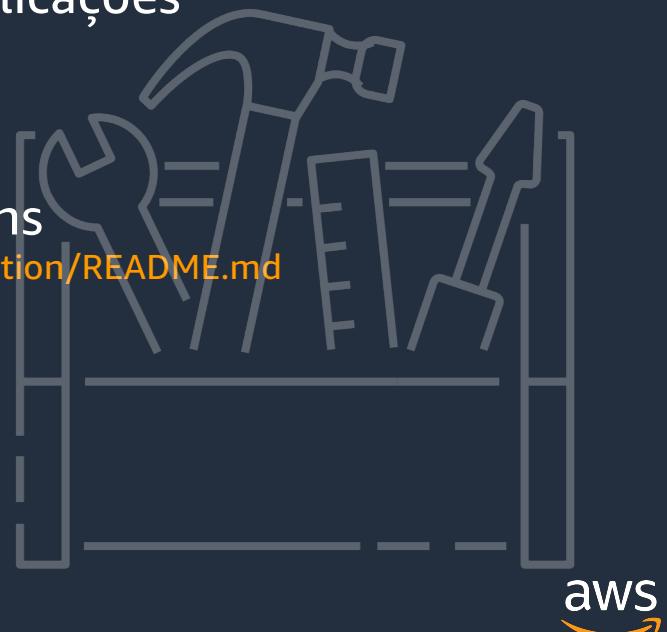
<https://bit.ly/aws-msgn>

Automação de Saga Pattern com CDK e Step Functions

<https://github.com/cdk-patterns/serverless/blob/master/the-saga-stepfunction/README.md>

Visual Studio Code - AWS Toolkit extension

<https://aws.amazon.com/visualstudiocode/>



Muito Obrigado

Alex Coqueiro

Head de Arquitetura de Soluções para América Latina, Canadá e Caribe
AWS Public Sector

 @alexbcbcbr