Criando sistema de votação igual do **BBB** com **NodeJS** e **Serverless**

Sem ser eliminado

Evandro Pires da Silva

- Evangelista de Serverless
- AWS Community Builder
- Programador desde os 12 anos de idade (com Clipper)
- Filho de programador
- Head de Pesquisa e Plataforma na Senior Sistemas
- Founder e host do Sem Servidor, podcast dedicado ao tema Serverless
- Adoro estudar idiomas: inglês, francês e espanhol (WIP)
- Marido da Madi, e pai do Teodoro e da Olivia

AWS community builder

SEMSERVIDOR

Afinal, por que eu fiz isso?

Por que criar um sistema de votação igual ao **BBB?**

- Fenômeno de sucesso
- Teve picos de 1,5mi votos por minuto
- Recorde de 1bi votos em uma rodada
- Minha obsessão por *time-to-market*

O que as empresas buscam?

- Entrega de valor para o cliente
  - Entender melhor o problema
  - UX design
- Time to market
  - Grande vs Rápido
  - Metodologias ágeis
- Resultado (mais receita e menos custo)

Qual o papel da **pessoa desenvolvedora?**

- Mais proximidade com o negócio
- Mais agilidade
- Somos solucionadores de problema
- Melhor ferramenta para o problema
- Devemos resolver o problema com menos recurso possível
    - Nosso tempo
    - Infraestrutura
    - Manutenção

E o que **serverless** tem a ver com tudo isso?

O que é serverless

Definição

Serverless é uma metodologia para planejar, criar e implantar software de maneira a maximizar o valor, minimizando o trabalho pesado indiferenciado. Ele toca tudo na cadeia de valor, não apenas afetando a maneira como os engenheiros abordam o desenvolvimento, mas também influenciando a estratégia, o design, o orçamento, o planejamento de recursos e muito mais. ~ **Jeremy Daly**

Caraterísticas

Características

Como é a "anatomia" de um serviço serverless

- Paga pelo que usa
  - Não paga pelo tempo ocioso
  - Não paga por capacidade provisionada
- Não precisa se preocupar com o servidor
- Escalabilidade, Alta Disponibilidade e Segurança por *default*

Function as a Service (FaaS)

Características

- Foco no código, sem burocracia
- Muito granular
- Paga por uso memória e tempo de execução
- *Cold Start*

Prós e contras

Prós e contras

Prós

- Melhor *time-to-market* para desenvolvimento
- Redução de custos (TCO)
  - Paga somente pelo o que é usado
  - Time mais enxuto
- Disponibilidade, escalabilidade, segurança e compliance como commodity

Prós e contras

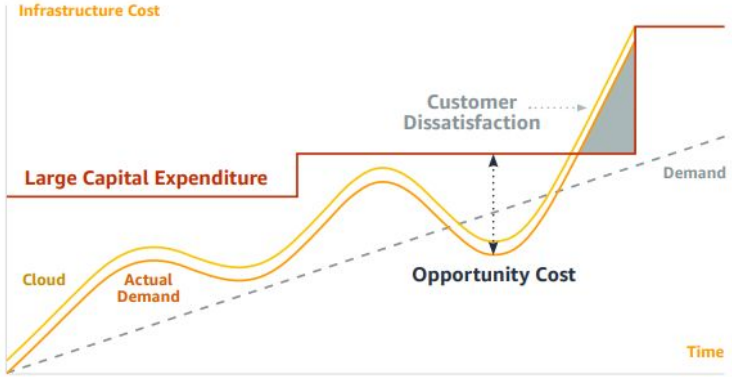
Contras

- *Cold start* (em alguns cenários)
- Vendor lock in (há opções para rodar on premise)
- Poucos cases e pessoas referências no Brasil (isso tem mudado rápido!)
- Custos lineares (pode ser bom)

Benchmark
Desenvolvimento tradicional vs Serverless

Benchmark

- Custo de infraestrutura
- Custo de desenvolvimento
- Custo de manutenção



Fonte: https://pages.awscloud.com/rs/112-TZM-766/images/AWS_MAD_Deloitte_TCO_paper.pdf

Benchmark

	Transportation Company		Banking Company	
	EC2	Serverless	EC2	Serverless
Infrastructure Cost ($/month)	$790	$1090	$296	$378
	Difference	**$300**	Difference	**$82**
Development Cost ($/month)	$640	$205	$640	$205
	Difference	$(435)	Difference	$(435)
Maintenance Cost ($/month)	$4096	$2240	$4096	$2240
	Difference	$(1856)	Difference	$(1856)
Total Cost ($/month)	$5526	$3535	$5032	$2823
	Difference	$(1991)	Difference	$(2209)

Fonte: https://pages.awscloud.com/rs/112-TZM-766/images/AWS_MAD_Deloitte_TCO_paper.pdf

Votação do BBB **#serverless**

Votação do BBB

Requisitos

- Simples, do ponto de vista funcional
- Complexo, por conta dos requisitos não funcionais
- Requisitos funcionais
  - Voto em algum nome
  - Ano da edição
  - Rodada
- Requisitos não funcionais
  - Alta escalabilidade
    - Já teve picos de 1,5mi votos por minuto
    - Recorde de 1bi votos
  - Apuração de votos em até 20 minutos

Votação do SSS

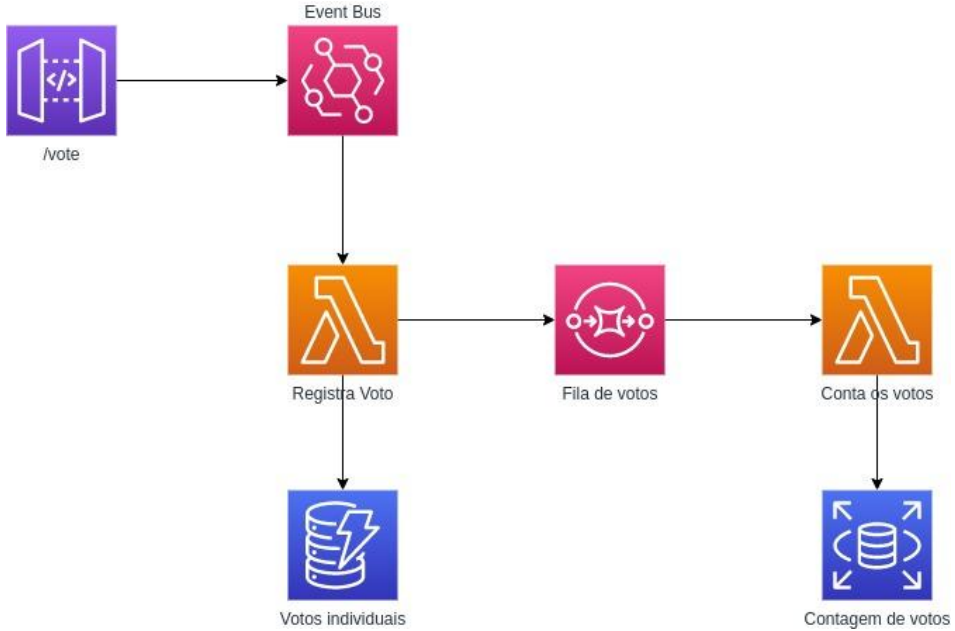
Super Serverless Sample (horrível, mas foi que consegui pensar)

- Arquitetura totalmente serverless
- Preparado para receber uma carga alta de requisições

Quem você quer eliminar?

- Demora para entrega
- Operação
- Custos

Arquitetura



Stack de código

- Infraestrutura como código
  - Serverless Framework
- Linguagem de programação
  - NodeJS

Principais dificuldades

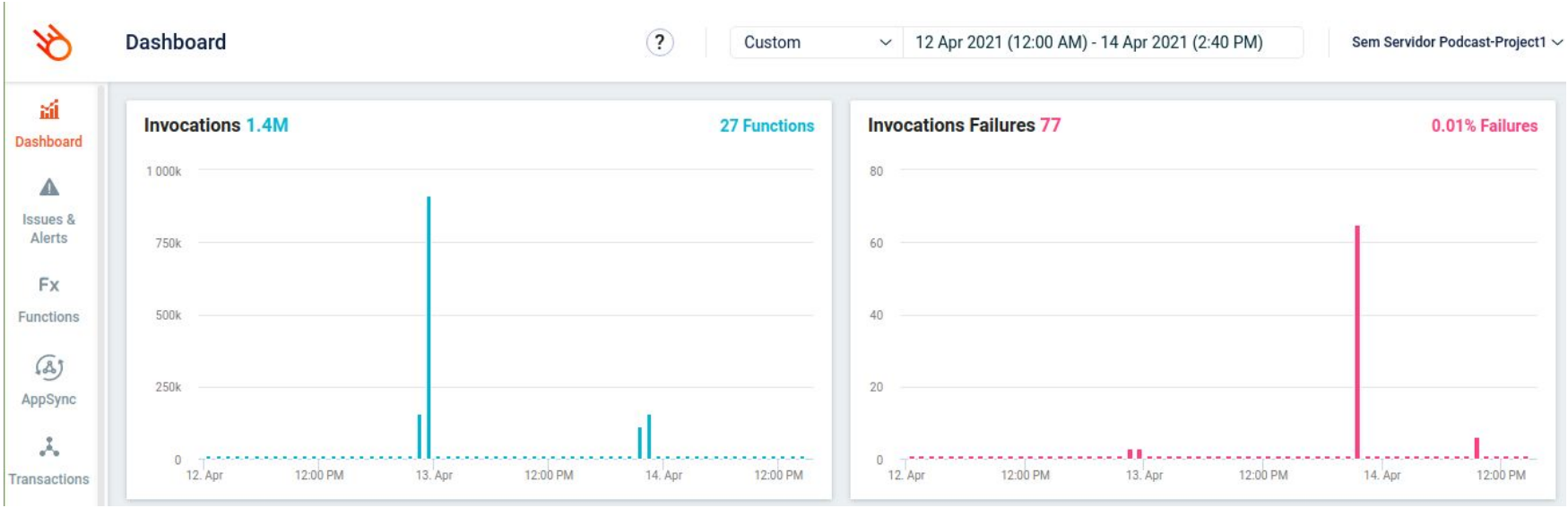
- Contabilizar os votos
  - Como processar em tempo viável?
- Técnicamente falando
  - Somar os votos sem perda
  - Limitação DynamoDB Stream
  - Capacidade de processamento de cada serviço da AWS: SQS <> EventBridge
    - Sem custos

Teste de carga

- Artillery (versão serverless)
  - 5k requisições por segundo
    - 10 segundos
    - 50k requisições
  - 20k requisições por segundo
    - 60 segundos
    - 1,2mi requisições
- JMeter

Monitoramento

- CloudWatch
- Lumigo



Show me the code

Estatísticas

- Desenvolvimento em 1 semana
  - 2 horas por dia útil
  - Total de 10 horas
- 1.2mi requisições em 1 minuto
  - Processamento do placar em 25 minutos
- Custos
  - RDS: US$ 2,90 / dia
  - VPC: US$ 1,44 / dia
  - Lambda: US$ 1,00 total
  - API Gateway: US$ 7,12 total
  - DynamoDB: US$ 2,46 total
  - SQS: US$ 0,21 total
  - **Total: US$ 52,57**

O que ainda poderia ser feito

- Abordagem mais robusta de dados
- *Near realtime* com Kinesis
- Pipeline para deploy automatizado
- Utilização de SecretsManager

Conclusões

Conclusões

- Várias outras formas de resolver o problema
  - Mas será que com o mesmo timing?
  - E ainda com os mesmo recursos?
- Inovação no SeniorLabs
  - Serverless para acelerar nossas PoCs
  - Redução de 80% em custos
    - Lightsail e EC2
      - ECS
      - CloudFront
      - S3
      - Lambda
- **Disclaimer:** Serverless nem sempre vai servir para todos os problemas

AWS community builder

SEMSERVIDOR

Referências

- Benchmark: https://pages.awscloud.com/rs/112-TZM-766/images/AWS_MAD_Deloitte_TCO_paper.pdf
- Serverless Artillery: https://github.com/Nordstrom/serverless-artillery
- Artillery: https://artillery.io
- Lumigo: https://lumigo.io
- SSS: https://github.com/epiresdasilva/super-serverless-sample

AWS community builder

SEMSERVIDOR

Obrigado!

Evandro Pires da Silva
@epiresdasilva at Linkedin, GitHub, Twitter