Pedro Felipe Santos Magalhães

Desenvolvimento de aplicações que usam o modelo *Serverless*

RELATÓRIO DE PROJETO FINAL

DEPARTAMENTO DE INFORMÁTICA
Programa de graduação em Ciência da Computação

Rio de Janeiro
mes de ano

Pontifícia Universidade Católica
do Rio de Janeiro

Pedro Felipe Santos Magalhães

Desenvolvimento de aplicações que usam o modelo *Serverless*

Relatório de Projeto Final

Relatório de Projeto Final, apresentado ao programa Ciência da Computaçãoda PUC-Rio como requisito parcial para a obtenção do titulo de Engenheiro de Computação.

Orientador : Noemi de La Rocque Rodriguez
Co-orientador: Maria Julia de Lima

Rio de Janeiro
mes de ano

Sumário

1
Introdução

Tradicionalmente o desenvolvimento de uma aplicação para a web envolve o desenvolvimento de dois componentes principais, a aplicação web(*frontend*) e o servidor web(*backend*) [Paulson, Linda Dailey. 2005]. O frontend executa no cliente enquanto o backend executa em um servidor físico gerenciado pelo desenvolvedor da aplicação.

Um dos principais desafios desta abordagem é dimensionar o servidor para atender todas as requisições dos clientes. Um superdimensionamento do servidor gera um gasto muito alto, enquanto o subdimensionamento limita o número de clientes que podem ser atendidos simultaneamente aumentando a latência das respostas [Iyengar, Arun, Ed MacNair, and Thao Nguyen. 1997].

O modelo de programação *Serverless* utiliza uma arquitetura em que o backend de uma aplicação não é administrado diretamente pelo desenvolvedor. Esse modelo vem ganhando força [Jamshidi, Pooyan. 2018] com o investimento pesado nos últimos anos das principais empresas de tecnologia de nuvem como IBM, Amazon, Google e Microsoft. A principal vantagem dessa abordagem de programação é que o desenvolvedor passa a não se preocupar com a infraestrutura de um servidor local, podendo implementar apenas a lógica que vai rodar em um servidor externo.

Neste projeto vamos estudar o desenvolvimento de aplicações com o modelo *serverless* em diferentes plataformas. Primeiramente, analisaremos as plataformas que oferecem esse modelo e selecionamos duas delas para um estudo mais aprofundado. Podendo assim, traçar um paralelo com o modelo tradicional de desenvolvimento em que a aplicação e o servidor são responsabilidade do desenvolvedor, com os prós e contras de cada.

2
Situação atual

O modelo *Serverless* é bem recente. O termo surgiu por volta de 2010. Em artigo de 2012, Ken Fromm esclarece que apesar do termo dar a ideia de que o servidor deixaria de existir, ele na verdade apenas indica que o servidor não seria mais gerenciado pelo desenvolvedor da aplicação [Fromm, K. 2012]. Ou seja, os servidores continuam existindo, mas são gerenciados por terceiros.

Hoje empresas como Amazon,Google, Microsoft, IBM oferecem algum tipo de serviço para implementação de ambientes *serverless*. A primeira delas a oferecer esse serviço foi a Amazon, com o "AWS Lambda" lançado em 2014, e as outras três lançaram suas implementações em 2016. Em geral, o pagamento para o uso dessas plataformas varia de acordo com o número de requisições recebidas e o tempo de computação utilizado. Todas elas oferecem uma versão grátis do serviço.

Além dos serviços em nuvem, existem implementações open source que são destinadas a rodar em um ambiente local, geralmente em clusters on premise. Como exemplos de implementação open source temos o Apache OpenWhisk, IronFunctions, Kubeless, OpenFaas, OpenLambda.

Em termos gerais, todas essas plataformas permitem que o desenvolvedor de aplicações não precise se preocupar com o desenvolvimento e configuração do servidor dedicando-se exclusivamente ao desenvolvimento da lógica da aplicação. Para desenvolver os serviços que atendem as requisições da aplicação, o desenvolvedor constrói funções stateless ou pequenos programas cadastrando-as na plataforma que fica responsável por ativá-las quando ocorre algum evento.

Por ser um modelo de programação recente e pouco explorado, achamos que pode ser interessante fazer uma análise para verificar os benefícios e desafios desta abordagem.

2.1 O modelo *serverless*

Inicialmente o termo *serverless* termo era usado para descrever aplicações que utilizavam grande parte ou todo o seu backend hospedado em uma plataforma de nuvem. Usualmente essas plataformas oferecem diversos serviços que são de uso comum a muitas aplicações, como serviço de autenticação de usuário, administração de banco de dados, armazenamento de dados em nuvem, push notification dentre outros serviços específicos de cada empresa que provê a plataforma de nuvem. Esse modelo é ficou conhecido como backend as a service (BaaS).

Mais recentemente o termo *Serverless* ficou mais ligado a aplicações que utilizam todo o servidor utilizando uma plataforma de nuvem. Em que, a lógica do servidor é escrita pelo desenvolvedor. No entanto, diferente da abordagem tradicional em que o servidor é uma aplicação monolítica, o desenvolvedor utiliza uma arquitetura de microsserviços [Namiot, Dmitry, and Manfred Sneps-Sneppe. 2014] escrevendo pequenas funções que cuidam de partes específicas da lógica do servidor. Essas funções rodam em contêineres sem estado em máquinas do provedor da nuvem e são ativadas através de eventos. Esse modelo também é conhecido como function as a service (FaaS).

Em geral as plataformas de nuvem oferecem os dois tipos de serviço em seu portfólio serverless, podendo ser usados em conjunto. Para tornar possível esse modelo de desenvolvimento dois conceitos foram primordiais: (i) tecnologia de contêineres. (ii) arquitetura de microsserviços.

A tecnologia de contêiner é uma alternativa às máquinas virtuais, e trazem o benefício de não possuírem uma camada do sistema operacional, compartilhando a camada do sistema operacional da máquina host. Desta forma um conteiner é muito mais eficiente tanto no tempo para inicialização da execução, quanto no tamanho da imagem de um contêiner [Bernstein, David. 2014]. A implementação mais utilizada é a do Docker. Discutiremos mais a fundo em 2.1.1

Com isso, qualquer máquina que seja capaz de rodar contêineres pode executar qualquer programa independente das dependência, basta o desenvolvedor criar um contêiner empacotando todas as dependências de ambiente que seu programa necessita. Assim, basta que a plataforma de nuvem utilize a imagem criada para rodar o programa desenvolvido.

As plataformas *serverless* ficam responsáveis por questões como a escalabilidade e elasticidade, disparando novas instâncias dos serviços de acordo com a demanda. O uso mais eficiente dos recursos é um dos principais benefícios deste modelo de programação. Uma vez que os serviços do *backend* são efême-

ros, ou seja, só ficam ativos durante uma execução após serem disparados por um evento.

Os serviços proprietários de nuvem oferecem alguns benefícios. Um deles é a redução do custo de recursos humanos e de máquinas para manutenção de um servidor. Outro é a menor latência de resposta para clientes afastados do local físico da empresa uma vez que em geral a nuvem possui servidores espalhados pelo mundo.

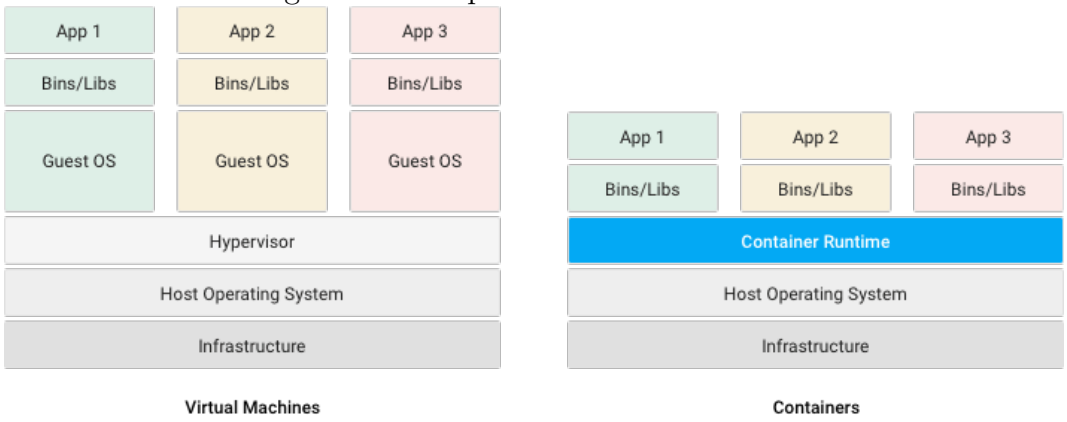
Em geral cada plataforma proprietária possui serviços extras que estão disponíveis apenas ao desenvolvedor que utiliza seus serviços, isso facilita o desenvolvimento da aplicação, mas tem o efeito de *Lock-in* [Zhu, Kevin Xiaoguo, and Zach Zhizhong Zhou. 2012] em que, com o passar do tempo, fica cada vez mais custoso migrar a aplicação para um outra plataforma *serverless*. Esse é um ponto fundamental na hora de avaliar qual plataforma utilizar.

Por outro lado, as plataformas open source oferecem um maior controle sobre os dados, o que pode ser interessante quando lidamos com dados sensíveis. Além disso o serviço é customizável, podendo ser estendido pelo desenvolvedor para atender a uma demanda específica da sua aplicação.

2.1.1 Contêineres

Os contêineres oferecem um mecanismo de empacotamento de aplicativos que permite abstrair o ambiente em que o aplicativo vai ser executado. Desacoplando assim, a plataforma de execução do desenvolvimento do software. São muito similares à maquinas virtuais, mas são mais otimizadas para ter um menor impacto de desempenho na maquina física que está rodando os contêineres.

Figura 2.1: Máquina Virtual x Contêiner



Dessa forma, basta o desenvolvedor definir o contêiner em que sua aplicação tem que executar para que ela passa rodar em qualquer outra

máquina que utiliza essa tecnologia. Na hora de executar a aplicação, a máquina executa o contêiner e não a aplicação diretamente. O contêiner de uma aplicação possui todas suas dependências, como pacotes e bibliotecas por exemplo, representado na figura 2.1 como "Bins/Libs".

Essa tecnologia foi essencial para o avanço da computação na nuvem, com ela a execução de aplicações de usuários é muito mais simples e independente da configuração física das máquinas que compõem a infra-estrutura da nuvem. Qualquer máquina pode rodar qualquer aplicação usando o contêiner definido por ela.

2.1.2 Microsserviços

3
Proposta e objetivos do trabalho

Com esse trabalho, pretendemos entender melhor as vantagens e limitações do modelo *serverless*. Para essa finalidade, pretendemos avaliar duas opções de plataforma *serverless* utilizando: (i) uma plataforma de nuvem (e.g, Google Cloud), (ii) uma solução open source em um cluster local.

Pretendemos definir duas aplicações que apresentem requisitos diferentes. Um dos cenários que nos interessa é o de serviços em pipeline. Esse cenário é caracterizado pelo encadeamento de serviços em que um evento inicial gera eventos para outros serviços desenvolvidos por nós. Um outro cenário é o de desenvolvimento de um serviço que utiliza outros serviços internamente.

Com o desenvolvimento dessas aplicações poderemos analisar a flexibilidade oferecida por cada plataforma, incluindo suas facilidade e limitações, tanto durante o desenvolvimento quanto durante seu funcionamento.

3.1
Aplicações propostas

3.1.1
Aplicação 1

3.1.2
Aplicação 2

3.2
Ambientes de desenvolvimento

3.2.1
Kubernetes

3.2.2
Google Cloud

4
Atividades realizadas

4.1
Estudo sobre os ambientes

4.2
Desenvolvimento de aplicações teste

4.2.1
Fórum serverless

Referências bibliográficas