Serverless Computing

Apresentado por Wallan Melo

Definição do Modelo Serverless

A computação serverless é um modelo de computação em nuvem que permite aos desenvolvedores criar e executar código em servidores gerenciados pelo provedor de serviços em nuvem e disponíveis sob demanda. A computação serverless isenta os desenvolvedores da necessidade de gerenciar a infraestrutura de backend e fornece um ambiente escalável e flexível para as empresas.

Diferença entre Infraestrutura Tradicional, Containers e Serverless

Usos de recursos

- A infraestrutura tradicional, ou monolítica, utiliza recursos de forma menos eficiente, pois a aplicação inteira é executada mesmo que apenas uma parte esteja em uso.
- Já os containers, apesar de mais eficientes que a infraestrutura tradicional, ainda consomem mais recursos comparados ao serverless. Isso ocorre porque os containers precisam ser provisionados com antecedência, exigindo alocação de recursos como CPU e memória, mesmo que não estejam em uso constante.
- O serverless, por outro lado, escala automaticamente e compartilhar recursos entre múltiplas funções (modelo multi-tenant), o que resulta em um uso de recursos muito mais eficiente. Além disso, as funções serverless são menores e não carregam dependências desnecessárias, ao contrário dos containers, que precisam incluir todas as dependências do ambiente de execução.

Obs:

O que é Multi tenant:
O modelo multi-tenant é uma arquitetura de software que permite que um único software atenda múltiplos usuários. É uma arquitetura baseada na ideia de que vários clientes compartilham uma única instância de um aplicativo.

Diferença entre Infraestrutura Tradicional, Containers e Serverless

Custo

- A infraestrutura tradicional envolve custos fixos elevados, pois os servidores precisam estar sempre ativos, independentemente do uso.
- Já os containers, embora mais flexíveis, ainda exigem pagamento antecipado pelos recursos provisionados, o que pode levar a subutilização ou super provisionamento.
- O serverless, por sua vez, opera em um modelo pay-as-you-go, onde você paga apenas pelo tempo de execução do código, reduzindo custos antecipados e operacionais. No entanto, a escalabilidade automática do serverless pode dificultar o controle de custos em casos de tráfego imprevisível ou malicioso, como ataques de força bruta, onde cobranças podem disparar sem monitoramento adequado.

Diferença entre Infraestrutura Tradicional, Containers e Serverless

Desempenho

- A infraestrutura tradicional oferece desempenho estável, mas limitado pela capacidade do servidor, o que pode ser um problema para aplicações que exigem alta escalabilidade.

- Os containers, por outro lado, são ideais para processos longos, mas podem ser dispendiosos para workloads irregulares ou curtos. Além disso, após períodos de inatividade, os containers podem sofrer com "cold starts", causando atrasos de vários segundos ao serem reiniciados.

- O serverless também enfrenta o problema de "cold starts", mas é altamente eficiente para workloads esporádicos e de curta duração, escalando automaticamente conforme a demanda. No entanto, para aplicações de missão crítica que exigem baixa latência, esses atrasos podem ser inaceitáveis.

Diferença entre Infraestrutura Tradicional, Containers e Serverless

Segurança

- Na infraestrutura tradicional, a segurança é de responsabilidade total da equipe de TI, que deve gerenciar firewalls, patches, atualizações e monitoramento constante.

- Já os containers oferecem um maior isolamento entre aplicações, o que pode aumentar a segurança, mas ainda exigem gerenciamento ativo da infraestrutura de orquestração (como ex: Kubernetes).

- O serverless, por outro lado, transfere a responsabilidade da segurança do backend para o provedor de nuvem, deixando os desenvolvedores responsáveis apenas pela segurança do frontend e do código. No entanto, a arquitetura multi-tenant do serverless pode representar riscos se o provedor não isolar adequadamente as aplicações de diferentes clientes.

Diferença entre Infraestrutura Tradicional, Containers e Serverless

Agilidade

- A infraestrutura tradicional é menos ágil, pois qualquer alteração no código exige um novo deploy de toda a aplicação, o que pode ser demorado e arriscado.

- Os containers, por sua vez, permitem CI/CD (Integração Contínua e Entrega Contínua), implementação rápida e maior elasticidade, tornando-os mais ágeis que a infraestrutura tradicional.

- O serverless, no entanto, leva a agilidade a outro nível, com baixa barreira de entrada e escalabilidade automática, permitindo que os desenvolvedores foquem apenas no código. No entanto, a portabilidade pode ser um problema no serverless, já que as funções costumam ser altamente acopladas a serviços específicos do provedor, aumentando o risco de vendor lock-in.

Obs: O que é Vendor Lock In? Vendor lock-in, ou aprisionamento tecnológico, é uma situação em que uma empresa ou cliente se torna dependente de um único fornecedor de produtos ou serviços

Vantagens

- Aumento da produtividade dos desenvolvedores: Uma das maiores vantagens do Serverless é a libertação dos desenvolvedores das tarefas de gerenciamento de servidores.
- Facilitação das práticas de DevOps: A computação Serverless simplifica a adoção de práticas de DevOps. Sem a necessidade de descrever detalhadamente a infraestrutura para as equipes de operações, o ciclo de desenvolvimento torna-se mais ágil e integrado.
- Integração com BaaS (Backend as a Service): A capacidade de incorporar componentes de soluções BaaS de terceiros no desenvolvimento de aplicações é outra vantagem significativa. Isso pode acelerar o desenvolvimento e otimizar ainda mais os recursos.
- Redução de custos operacionais: No modelo Serverless, os custos estão associados ao tempo de uso efetivo da computação em nuvem. Isso contrasta com o modelo tradicional, onde os servidores permanecem ativos e geram custos contínuos, independentemente do uso.
- Desenvolva em qualquer linguagem: o serverless é um ambiente multilinguagem que possibilita aos desenvolvedores programar em qualquer linguagem ou estrutura, seja Java, Python, JavaScript, node.js, em que estejam familiarizados.

Desvantagens

- Limite de recursos: A computação Serverless não é adequada para algumas cargas de trabalho, como a computação de alto desempenho, devido aos limites de recursos impostos pelos provedores de nuvem.
- Dependência de fornecedor(Lock in): Utilizar intensamente os serviços de um provedor de nuvem pode aumentar o risco de dependência desse fornecedor. A mudança para um novo provedor pode envolver custos adicionais e a necessidade de adaptar os sistemas às especificações do novo ambiente.
- Inicialização lenta: Também conhecida como "cold start", essa latência devido a inicialização pode afetar o desempenho e a capacidade de resposta dos aplicativos serverless, especialmente em ambientes de demanda em tempo real.
- Teste e depuração complexos: a depuração pode ser mais complicada com um modelo de computação serverless, pois os desenvolvedores não têm visibilidade dos processos de back-end.
- Custo mais alto para a execução de aplicativos longos: Os modelos de execução serverless não foram projetados para executar códigos por períodos prolongados. Portanto, os processos de longa duração podem ter um custo mais elevado.

Principais Provedores do Mercado

- Microsoft Azure (Azure Functions)
- Google Cloud (Google Cloud Functions)
- IBM Cloud (IBM Cloud Code Engine).
- Amazon Web Services (AWS Lambda)

AWS Lambda

AWS Lambda no nível gratuito

Comece a usar o AWS Lambda gratuitamente »

PRODUTO	DESCRIÇÃO	DETALHES DA OFERTA DO NÍVEL GRATUITO	PREÇO DOS PRODUTOS
AWS Lambda	Serviço de computação que executa seu código em resposta a eventos e gerencia automaticamente os recursos computacionais.	**Sempre gratuito** **1 milhão** de solicitações gratuitas por mês **3,2 milhões** de segundos de tempo de computação por mês	Preços do AWS Lambda

Casos de Uso do AWS Lambda

- **Aplicativos web:** Ao combinar o AWS Lambda com outros produtos da AWS, os desenvolvedores conseguem criar aplicativos web.

- **Machine learning:** Você pode usar o AWS Lambda para pré-processar os dados antes de usá-los para alimentar seu modelo de machine learning.

- **Processamento de dados:** Execute códigos em resposta a acionadores, como alterações nos dados, mudanças no estado do sistema ou intervenções dos usuários. O Lambda pode ser acionado pelos serviços da AWS, como S3, DynamoDB, Kinesis ou SNS, e pode se conectar a sistemas de arquivos do EFS ou em workflows com o AWS Step Functions.

PRÁTICA na AWS Lambda

OBRIGADO