

UNIVERSIDADE DE BRASÍLIA

FCTE

Programação para Sistemas Paralelos e Distribuídos

Atividade extra do projeto

**Cloud Native: Conceitos, Tecnologias e Impactos no
Desenvolvimento de Software**

Breno Henrique de Souza - 202015984

Giulia Domingues de Alcantara - 180121308

Samuel Alves Sato - 200069322

Valderson Pontes da Silva Junior - 190020521

Brasília, DF

2025

1. INTRODUÇÃO

Contextualização do tema Cloud Native

A computação em nuvem revolucionou a forma como as organizações desenvolvem, implantam e gerenciam aplicações. Nesse cenário, o conceito de Cloud Native emerge como uma abordagem que maximiza os benefícios oferecidos pelo modelo de entrega em nuvem. Aplicações nativas em nuvem são projetadas para explorar ao máximo as características da computação em nuvem, como escalabilidade, resiliência e flexibilidade. Elas podem ser construídas utilizando diferentes arquiteturas, incluindo tanto microsserviços quanto monolitos, dependendo dos requisitos específicos e objetivos do negócio. Além disso, essas aplicações frequentemente utilizam contêineres para garantir portabilidade e consistência, sendo orquestradas por plataformas como o Kubernetes.

Importância do Cloud Native no cenário atual da computação em nuvem

A adoção de práticas e tecnologias Cloud Native permite que as empresas desenvolvam aplicações mais resilientes e escaláveis, capazes de se adaptar rapidamente às mudanças nas demandas do mercado. Essa abordagem facilita a implementação de atualizações contínuas sem interrupções significativas, promovendo uma experiência aprimorada para os usuários finais. Além disso, ao utilizar arquiteturas modulares, compostas por componentes independentes que podem ser desenvolvidos, implantados e escalados separadamente, e serviços gerenciados, que reduzem a complexidade operacional ao oferecer funcionalidades prontas e otimizadas, as organizações conseguem otimizar recursos, melhorar a eficiência e reduzir custos operacionais.

Objetivo do documento

Este documento tem como objetivo explorar o conceito de Cloud Native, detalhando seus principais elementos e características. Serão discutidos os impactos dessa abordagem nas práticas de DevOps, incluindo a influência de contêineres e tecnologias de computação em cluster, com destaque para o

papel do Kubernetes. Além disso, será realizada uma comparação entre arquiteturas de aplicações baseadas em microsserviços e monolíticas, analisando o futuro das aplicações no paradigma Cloud Native. Por fim, será abordada a importância do monitoramento e da observabilidade em ambientes Cloud Native, apresentando desafios comuns e ferramentas de suporte.

2. CONCEITOS FUNDAMENTAIS DO CLOUD NATIVE

Definição de Cloud Native

O termo Cloud Native refere-se a uma abordagem de desenvolvimento e operação de aplicações que aproveita ao máximo as características e benefícios oferecidos pela computação em nuvem. Isso inclui a utilização de modelos baseados na nuvem para criar e executar aplicativos escaláveis e dinâmicos em ambientes modernos, como nuvens públicas, privadas e híbridas.

Principais elementos e características

As aplicações Cloud Native são caracterizadas por vários elementos e práticas que as tornam distintas:

- **Contêineres:** Empacotam o código e suas dependências em unidades portáteis e consistentes, garantindo que as aplicações funcionem de maneira uniforme em diferentes ambientes.
- **Microsserviços:** Arquitetura que divide a aplicação em serviços pequenos e independentes, cada um responsável por uma funcionalidade específica, facilitando o desenvolvimento, a implantação e a escalabilidade.
- **DevOps e Integração Contínua/Entrega Contínua (CI/CD):** Práticas que promovem a colaboração entre equipes de desenvolvimento e operações, automatizando processos de construção, teste e implantação para acelerar a entrega de software.
- **Infraestrutura Imutável:** Servidores e componentes de infraestrutura são substituídos em vez de modificados, garantindo consistência e

facilitando a recuperação em caso de falhas.

- **Orquestração e Automação Dinâmica:** Uso de ferramentas como o Kubernetes para gerenciar automaticamente a implantação, escalabilidade e operação de contêineres, otimizando a utilização de recursos.

Arquitetura e princípios fundamentais

A arquitetura Cloud Native baseia-se em princípios que promovem a eficiência e a resiliência das aplicações:

- **Desenvolvimento de Aplicações como Serviços Independentes:** Cada serviço é desenvolvido, implantado e escalado de forma autônoma, permitindo atualizações e manutenção sem impactar o sistema como um todo.
- **Automação de Processos:** Desde a provisão de infraestrutura até a implantação de código, a automação reduz erros humanos e aumenta a eficiência operacional.
- **Observabilidade:** Implementação de monitoramento, logging e tracing para fornecer insights detalhados sobre o comportamento da aplicação, facilitando a identificação e resolução de problemas.
- **Resiliência e Recuperação de Falhas:** Projetar sistemas que possam detectar, isolar e se recuperar automaticamente de falhas, minimizando o impacto nos usuários finais.

Esses princípios e elementos combinados permitem que as organizações desenvolvam e operem aplicações que são não apenas robustas e escaláveis, mas também ágeis e alinhadas às necessidades dinâmicas dos negócios modernos.

3. O PAPEL DO DEVOPS E A INFLUÊNCIA DAS TECNOLOGIAS CLOUD NATIVE

Adoção de Contêineres e sua Influência no DevOps

A integração de contêineres no ambiente DevOps tem transformado significativamente as práticas de desenvolvimento e operações. Contêineres

permitem empacotar aplicações e suas dependências em unidades portáteis e consistentes, garantindo que o software funcione uniformemente em diferentes ambientes. Essa consistência reduz discrepâncias entre ambientes de desenvolvimento, teste e produção, facilitando a colaboração entre equipes e acelerando o ciclo de entrega de software. Ferramentas como Docker têm sido fundamentais nesse processo, proporcionando uma plataforma robusta para criação e gerenciamento de contêineres.

Computação em Cluster e sua Relação com DevOps

A computação em cluster, especialmente quando combinada com orquestradores de contêineres como o Kubernetes, aprimora as práticas de DevOps ao fornecer uma infraestrutura escalável e resiliente. Clusters permitem que aplicações sejam distribuídas em múltiplos nós, garantindo alta disponibilidade e balanceamento de carga eficiente. O Kubernetes automatiza a implantação, o dimensionamento e o gerenciamento de aplicações containerizadas, permitindo que as equipes de DevOps gerenciem a infraestrutura e as configurações de maneira consistente em vários ambientes.

Benefícios do Kubernetes para Desenvolvimento e Manutenção de Aplicações

O Kubernetes oferece diversos benefícios para o desenvolvimento e a manutenção de aplicações. Ele facilita a automação de tarefas operacionais, como o dimensionamento automático, implantação contínua e recuperação de falhas, reduzindo a carga de trabalho manual e melhorando a eficiência operacional. Além disso, por ser uma plataforma de código aberto, o Kubernetes pode ser implementado em diversos ambientes, incluindo nuvens públicas, privadas e híbridas, proporcionando flexibilidade para as equipes de TI. Adicionalmente, o Kubernetes simplifica o monitoramento, o gerenciamento, a implementação e a configuração de aplicações em contêineres de qualquer tamanho ou escala, garantindo alta disponibilidade e desempenho consistente das aplicações.

Integração Contínua e Automação com Cloud Native

A adoção de práticas Cloud Native, combinada com ferramentas como o Kubernetes, potencializa a integração contínua (CI) e a entrega contínua (CD) ao automatizar processos de construção, teste e implantação de software. Essa automação reduz o tempo de entrega e aumenta a confiabilidade das aplicações. Além disso, o Kubernetes permite que as equipes de DevOps definam políticas de resiliência e escalonamento, minimizando a necessidade de intervenções manuais e melhorando a eficiência operacional.

4. ARQUITETURA DE APLICAÇÕES CLOUD NATIVE

A arquitetura de aplicações Cloud Native evoluiu para atender às demandas de escalabilidade, flexibilidade e manutenção no desenvolvimento de software moderno. Duas abordagens principais se destacam: a arquitetura monolítica e a de microsserviços.

Comparação entre Arquiteturas Monolíticas e Microsserviços

Na arquitetura monolítica, todas as funcionalidades de uma aplicação são integradas em um único bloco de código. Isso significa que a interface do usuário, a lógica de negócios e o acesso a dados estão interligados e dependentes uns dos outros. Embora essa abordagem possa simplificar o desenvolvimento inicial e a implantação, à medida que a aplicação cresce, torna-se mais complexa e difícil de manter. Atualizações ou modificações em uma parte do sistema podem impactar todo o aplicativo, exigindo testes e implantações abrangentes. Além disso, a escalabilidade é limitada, pois é necessário escalar toda a aplicação, mesmo que apenas um componente específico esteja enfrentando alta demanda.

Em contraste, a arquitetura de microsserviços divide a aplicação em componentes menores e independentes, cada um responsável por uma funcionalidade específica. Esses serviços comunicam-se entre si por meio de APIs bem definidas, permitindo que sejam desenvolvidos, implantados e escalados de forma autônoma. Essa modularidade facilita a manutenção e a introdução de novas funcionalidades sem afetar todo o sistema. Além disso, cada microsserviço pode ser desenvolvido com tecnologias e linguagens de

programação diferentes, conforme as necessidades específicas de cada um. No entanto, essa abordagem também traz desafios, como a complexidade na gestão de múltiplos serviços, necessidade de estratégias robustas de monitoramento e a garantia de comunicação eficiente entre os serviços.

Benefícios e Desafios dos Microsserviços no Contexto Cloud Native

A adoção de microsserviços no contexto Cloud Native oferece benefícios como escalabilidade, permitindo que cada serviço seja dimensionado de forma independente conforme a demanda, e agilidade no desenvolvimento, já que equipes podem trabalhar simultaneamente em diferentes serviços, acelerando a entrega de funcionalidades. Além disso, a resiliência é aprimorada, pois falhas em um serviço específico não comprometem toda a aplicação.

5. MONITORAMENTO E OBSERVABILIDADE EM CLOUD NATIVE

Em ambientes Cloud Native, o monitoramento e a observabilidade são cruciais para garantir a performance, a disponibilidade e a resiliência das aplicações. Com a crescente adoção de arquiteturas de microsserviços e infraestruturas dinâmicas, a complexidade dos sistemas aumenta, tornando indispensável uma visão abrangente e detalhada de todos os componentes envolvidos.

Importância do Monitoramento em Ambientes Cloud Native

O monitoramento em ambientes Cloud Native permite que as equipes de TI acompanhem, em tempo real, o desempenho e o estado de saúde das aplicações e da infraestrutura subjacente. Essa prática é essencial para a detecção proativa de problemas, possibilitando respostas rápidas a incidentes e minimizando o impacto nos usuários finais. Além disso, o monitoramento contínuo auxilia na otimização de recursos, garantindo que as aplicações operem de maneira eficiente e econômica.

Principais Desafios e Problemas Enfrentados

A transição para arquiteturas Cloud Native apresenta desafios específicos no

que diz respeito ao monitoramento e à observabilidade. A fragmentação de aplicações em múltiplos microsserviços pode dificultar a rastreabilidade e a compreensão do fluxo de dados. Além disso, ambientes que escalam automaticamente podem gerar volumes massivos de dados de telemetria, tornando desafiador o processamento e a análise em tempo hábil. A coexistência de diversas ferramentas, linguagens e plataformas requer soluções de monitoramento que integrem e correlacionam dados de fontes distintas de maneira coesa.

Ferramentas e Técnicas para Observabilidade e Monitoramento

Para enfrentar os desafios de monitoramento e observabilidade em ambientes Cloud Native, diversas ferramentas e técnicas têm sido adotadas. O Prometheus é uma solução de código aberto que coleta e armazena métricas em séries temporais, permitindo consultas e alertas em tempo real, sendo amplamente utilizada para monitorar a performance de aplicações e infraestrutura. Complementando o Prometheus, o Grafana é uma plataforma de análise e monitoramento que permite a visualização de métricas por meio de dashboards interativos, integrando-se facilmente com diversas fontes de dados. Outra ferramenta relevante é o Elastic Observability, parte do Elastic Stack, que oferece soluções para monitoramento unificado de logs, métricas e rastreamentos, facilitando a detecção e resolução de problemas em ambientes Cloud Native. Além disso, a tecnologia eBPF (Extended Berkeley Packet Filter) permite a execução de programas no kernel do sistema operacional, proporcionando monitoramento avançado e coleta de métricas com baixo overhead; empresas como Meta, Google e Netflix utilizam eBPF para aprimorar a observabilidade de seus sistemas.

Exemplos de Soluções Utilizadas na Indústria

Empresas líderes em tecnologia têm adotado soluções avançadas de monitoramento e observabilidade para aprimorar a eficiência e a confiabilidade de suas infraestruturas. Por exemplo, a Netflix utiliza o eBPF para obter visibilidade em nível de kernel, permitindo monitoramento detalhado e diagnóstico preciso de desempenho em sua infraestrutura de

streaming. Para otimizar o desempenho de programas eBPF, a empresa desenvolveu o bpftop, uma ferramenta de código aberto que fornece uma visão em tempo real dos programas eBPF em execução, exibindo métricas como tempo médio de execução e utilização de CPU.

Da mesma forma, a Meta implementou o eBPF por meio de seu balanceador de carga, o Katran, que gerencia eficientemente o tráfego para plataformas como o Facebook. O Katran é uma biblioteca em C++ que utiliza a infraestrutura XDP do kernel Linux para fornecer um plano de encaminhamento de alto desempenho para balanceamento de carga de camada 4.

Esses exemplos demonstram a importância de investir em ferramentas e práticas de monitoramento e observabilidade para manter a eficiência e a confiabilidade de aplicações em ambientes Cloud Native.

6. CONCLUSÃO

Ao longo deste artigo, foram explorados os conceitos fundamentais do paradigma Cloud Native, suas principais características e os benefícios que ele proporciona para o desenvolvimento e a operação de aplicações modernas. Foi discutido como essa abordagem, baseada no uso de contêineres, microsserviços e orquestração com Kubernetes, tem transformado o cenário de DevOps, permitindo maior automação, escalabilidade e eficiência operacional. Além disso, analisamos o impacto do monitoramento e da observabilidade, destacando ferramentas como Prometheus, Grafana e eBPF, que desempenham um papel essencial na garantia de desempenho, resiliência e disponibilidade das aplicações em ambientes dinâmicos.

Olhando para o futuro, a adoção de Cloud Native deve continuar crescendo, impulsionada por tendências emergentes como serverless computing, service mesh, edge computing e inteligência artificial aplicada à automação de operações (AIOps). Essas inovações prometem levar ainda mais eficiência e

flexibilidade às empresas, permitindo que aplicações sejam mais ágeis, resilientes e adaptáveis às constantes mudanças do mercado digital. Além disso, espera-se que novas práticas e ferramentas ampliem a segurança e a governança em ambientes de nuvem, tornando a adoção do modelo Cloud Native ainda mais robusta.

Por fim, a transição para um ambiente Cloud Native não se trata apenas de tecnologia, mas de uma mudança cultural e estratégica. Empresas que investem nessa abordagem precisam adotar práticas de DevOps bem estruturadas, investir em automação e monitoramento avançado, além de capacitar suas equipes para lidar com a complexidade e as oportunidades desse novo paradigma. À medida que essas tecnologias continuam evoluindo, organizações que souberem aproveitar seu potencial terão uma vantagem competitiva significativa, conseguindo entregar software de forma mais rápida, confiável e eficiente, atendendo melhor às demandas do mundo digital.

7. REFERÊNCIAS

GOOGLE CLOUD. What is Cloud Native? Disponível em:

<https://cloud.google.com/learn/what-is-cloud-native>. Acesso em: 19 fev. 2025.

DYNATRACE. Cloud Native Architecture. Disponível em:

<https://www.dynatrace.com/knowledge-base/cloud-native-architecture>. Acesso em: 19 fev. 2025.

AWS. What is Cloud Native? Disponível em:

<https://aws.amazon.com/what-is/cloud-native>. Acesso em: 20 fev. 2025.

Cursa. Containers e a sua Relação com DevOps: Docker e Kubernetes. Disponível em:

<https://cursa.app/pt/pagina/containers-e-a-sua-relacao-com-devops-docker-e-kubernetes>. Acesso em: 20 fev. 2025.

SERVICE. Como o Kubernetes Potencializa os Benefícios do DevOps. Disponível

em:

<https://service.com.br/como-o-kubernetes-potencializa-os-beneficios-do-devops/>.

Acesso em: 21 fev. 2025.

MARTITEC. Principais Benefícios do Kubernetes. Disponível em:

<https://martitec.com.br/principais-beneficios-do-kubernetes/>. Acesso em: 21 fev.

2025.

AWS. The Difference Between Monolithic and Microservices Architecture.

Disponível em:

<https://aws.amazon.com/pt/compare/the-difference-between-monolithic-and-microservices-architecture>. Acesso em: 21 fev. 2025.

MICROSOFT. Observability Patterns. Disponível em:

<https://learn.microsoft.com/pt-pt/dotnet/architecture/cloud-native/observability-patterns>. Acesso em: 21 fev. 2025.

NETFLIX TECH BLOG. Announcing Bpftop: Streamlining eBPF Performance Optimization. Disponível em:

<https://netflixtechblog.com/announcing-bpftop-streamlining-ebpf-performance-optimization-6a727c1ae2e5>. Acesso em: 19 fev. 2025.

FACEBOOK ENGINEERING. Open-Sourcing Katran: A Scalable Network Load Balancer. Disponível em:

<https://engineering.fb.com/2018/05/22/open-source/open-sourcing-katran-a-scalable-network-load-balancer>. Acesso em: 20 fev. 2025.