DESENVOLVIMENTO DE SOFTWARE E-COMMERCE UTILIZANDO ORQUESTRAÇÃO DE CONTAINERS COM KUBERNETES

Gabriel Altenhofen, Matias Riffel,Jonas Rodrigo Pacheco

RESUMO

Este trabalho descreve o processo de implementação de uma plataforma de e-commerce para docerias utilizando orquestração de containers com Kubernetes. O objetivo principal foi documentar e analisar a experiência de desenvolvimento de uma aplicação web utilizando tecnologias de conteinerização em ambiente de nuvem, especificamente Google Kubernetes Engine (GKE). A solução técnica proposta divide a aplicação em containers separados para frontend e backend, permitindo deployment independente e maior controle sobre o ambiente de execução. A implementação foi realizada utilizando o Google Cloud Platform (GCP), com Docker para containerização, PostgreSQL Database-as-a-Service para persistência de dados, e backend em Node.js. A interface de usuário foi desenvolvida com HTML, CSS, JavaScript e Bootstrap, com hospedagem de imagens delegada ao Cloudinary. O sistema integra a API do Mercado Pago para simulação de transações financeiras. Os resultados obtidos demonstram a viabilidade da abordagem de conteinerização para aplicações de pequeno e médio porte, evidenciando tanto os benefícios quanto os desafios específicos encontrados durante o processo de implementação. A experiência documenta lições aprendidas sobre orquestração de containers, configuração de clusters Kubernetes, e deployment em ambiente de produção.

Palavras-chave: Microserviços. SaaS. E-commerce. Docker. Orquestração.

INTRODUÇÃO

O cenário do comércio eletrônico tem demonstrado um crescimento contínuo e acelerado nos últimos anos. A digitalização das vendas, impulsionada por mudanças no comportamento do consumidor e pela necessidade de otimizar operações, tornou-se um fator crítico de competitividade, especialmente para pequenos negócios. Empresas como docerias, cafeterias e lanchonetes buscam soluções acessíveis e eficientes para gerenciar seus estoques, processar pedidos e oferecer uma experiência de compra online atrativa.

O desenvolvimento tradicional de aplicações web frequentemente enfrenta desafios relacionados à portabilidade entre ambientes, gerenciamento de dependências e escalabilidade. A conteinerização surge como uma alternativa para abordar esses problemas, oferecendo maior consistência entre ambientes de desenvolvimento, teste e produção, além de facilitar o processo de deployment e manutenção.

Diante desse contexto, o presente trabalho tem como objetivo implementar e documentar o desenvolvimento de uma plataforma de e-commerce utilizando tecnologias de conteinerização e orquestração com Kubernetes. A escolha dessa abordagem visa demonstrar na prática como tecnologias modernas de DevOps podem ser aplicadas no desenvolvimento de sistemas web, documentando os desafios encontrados e as soluções implementadas.

O projeto busca, assim, proporcionar um estudo de caso prático sobre a aplicação de conceitos de conteinerização em um cenário real de desenvolvimento, oferecendo insights sobre a viabilidade e os trade-offs envolvidos na adoção dessas tecnologias para aplicações de pequeno e médio porte.

FUNDAMENTAÇÃO TEÓRICA

A solução proposta fundamenta-se em um conjunto de tecnologias modernas de desenvolvimento e deployment, cada uma com um papel específico na construção de um sistema containerizado e escalável.

Conteinerização com Docker

A conteinerização é o processo de empacotar uma aplicação e suas dependências em um container, uma unidade de software leve, portátil e executável. Docker é a ferramenta líder de mercado para criar e gerenciar esses containers. Diferentemente da virtualização tradicional, os containers compartilham o kernel do sistema operacional hospedeiro, resultando em menor overhead de recursos e inicialização mais rápida (DOCKER, 2025).

No contexto deste projeto, a conteinerização oferece benefícios como:

Consistência de ambiente: Eliminação de diferenças entre ambientes de desenvolvimento, teste e produção

Isolamento: Separação de dependências entre diferentes componentes da aplicação

Portabilidade: Capacidade de executar em qualquer ambiente que suporte Docker

Versionamento: Controle de versões de imagens facilitando rollbacks e atualizações

Orquestração com Kubernetes

Kubernetes é uma plataforma de orquestração de containers de código aberto que automatiza a implantação, o escalonamento e o gerenciamento de aplicações conteinerizadas (KUBERNETES, 2025). A plataforma abstrai a complexidade da infraestrutura subjacente, permitindo que desenvolvedores foquem na lógica da aplicação. Os principais conceitos do Kubernetes utilizados neste projeto incluem:

Pods: Menor unidade de deployment, representando um ou mais containers que compartilham recursos de rede e armazenamento.

Deployments: Objeto que gerencia o estado desejado da aplicação, controlando a criação, atualização e remoção de pods.

Services: Abstração que define um conjunto lógico de pods e uma política de acesso, proporcionando descoberta de serviços e balanceamento de carga.

Ingress: Objeto que gerencia o acesso externo aos serviços dentro do cluster, tipicamente HTTP/HTTPS.

Database-as-a-Service (DBaaS)

Database-as-a-Service é um modelo de computação em nuvem onde o provedor gerencia a infraestrutura, configuração, backup e manutenção do banco de dados. No contexto deste projeto, PostgreSQL foi utilizado como DBaaS no Google Cloud Platform, oferecendo:

Gerenciamento automatizado: Backups, patches e monitoramento

Escalabilidade: Ajuste de recursos conforme demanda

Alta disponibilidade: Replicação e failover automático

Segurança: Criptografia e controle de acesso integrados

2.4 Google Kubernetes Engine (GKE)

O Google Kubernetes Engine é um serviço gerenciado de Kubernetes que facilita o deployment e gerenciamento de clusters. O GKE abstrai a complexidade de configuração e manutenção da infraestrutura Kubernetes, oferecendo:

Cluster gerenciado: Google gerencia o control plane

Auto-scaling: Escalabilidade automática de nodes

Integração nativa: Conectividade com outros serviços do GCP

Segurança: Patches de segurança automáticos

METODOLOGIA

A metodologia de desenvolvimento adotou uma abordagem prática e interativa, focada na implementação e documentação de uma aplicação real. O processo foi dividido nas seguintes etapas:

Definição de Requisitos e Arquitetura

Primeiro, definimos os requisitos funcionais, como o sistema de cadastro de produtos, carrinho de compras, interface administrativa e integração de pagamentos. A arquitetura foi projetada para separar frontend e backend em contêineres distintos, permitindo o desenvolvimento e o deployment de forma independente.

Desenvolvimento Local

O desenvolvimento começou localmente para validar a lógica e a interface do usuário. O frontend foi construído com HTML, CSS, JavaScript e Bootstrap, enquanto o backend foi feito em Node.js com Express.js, expondo uma API RESTful. Durante esta fase, configuramos a comunicação com o banco PostgreSQL e uma integração básica com a API do Mercado Pago para simulação de pagamentos.

Conteinerização

Após a validação local, a aplicação foi conteinerizada com Docker. Criamos Dockerfiles para o frontend (baseado em nginx) e para o backend (baseado em Node.js). As imagens foram testadas localmente com Docker Compose para garantir a comunicação entre os contêineres antes do *deployment* no Kubernetes.

Configuração de Manifesto Kubernetes

Criamos arquivos de manifesto YAML para definir a infraestrutura no Kubernetes, incluindo Deployments (para especificar o estado dos contêineres), Services (para a descoberta e exposição interna), Ingress (para roteamento externo) e ConfigMaps/Secrets (para configurações e credenciais).

Deployment em GKE

O *deployment* foi realizado em um cluster do Google Kubernetes Engine (GKE), utilizando o PostgreSQL como serviço de banco de dados. Este processo

incluiu o envio das imagens Docker para o Google Container Registry, a aplicação dos manifestos Kubernetes e a configuração de DNS e SSL.

Validação e Documentação

Na etapa final, fizemos testes funcionais no ambiente de produção e documentamos os desafios e as soluções. Também coletamos métricas de uso de recursos e registramos as lições aprendidas ao longo do processo.

APRESENTAÇÃO, ANÁLISE E DISCUSSÃO DOS RESULTADOS

A plataforma de e-commerce foi implementada com sucesso usando uma arquitetura de microserviços e foi testada no Google Cloud. Essa abordagem se mostrou modular e escalável, permitindo atualizações independentes do frontend e backend sem interromper o serviço.

Desempenho e Escalabilidade

O sistema apresentou um bom desempenho, com tempos de resposta rápidos. O uso do Kubernetes otimizou a alocação de recursos, garantindo que os serviços estivessem sempre disponíveis e responsivos. Os Deployments com réplicas asseguraram a alta disponibilidade, evitando que a falha de um contêiner causasse a indisponibilidade do serviço. A arquitetura está pronta para ser expandida com a adição de novos serviços, como um futuro sistema de gerenciamento de estoque ou de análise de dados.

Desafios e Lições Aprendidas

Apesar dos benefícios, a implementação com Kubernetes teve desafios. A curva de aprendizado inicial com arquivos YAML complexos e a comunicação entre os serviços exigiram bastante atenção para garantir a resiliência e a tolerância a falhas. A gestão do banco de dados também demandou cuidado para garantir a consistência dos dados.

A integração parcial com o Mercado Pago foi funcional para a simulação, mas um sistema de pagamento completo exigiria a gestão de webhooks e a segurança de dados sensíveis, que não foram abordados neste projeto. Futuros trabalhos deverão focar nesses pontos.

Impacto e Potencial de Negócio

O projeto mostra que pequenas empresas podem se beneficiar de tecnologias avançadas. Ao oferecer a plataforma como SaaS, a solução permite que negócios sem recursos técnicos utilizem um e-commerce moderno e flexível, pagando por um serviço em vez de investir em infraestrutura própria.

CONCLUSÃO

O desenvolvimento da plataforma de doceria online com microserviços e Kubernetes se mostrou uma abordagem eficaz para resolver os desafios de escalabilidade e manutenção em pequenos e-commerces. O protótipo construído tem uma arquitetura sólida e funcional, gerenciando produtos e pedidos de forma modular. Os objetivos do projeto foram atingidos, validando o uso de contêineres e orquestração para criar sistemas distribuídos e resilientes. Este trabalho reforça a

importância do planejamento arquitetural, mesmo com a curva de aprendizado acentuada.

As lições aprendidas sobre a complexidade de gerenciar serviços em um ambiente distribuído são valiosas para projetos futuros. A arquitetura criada serve como uma base robusta para a evolução do sistema, permitindo a fácil adição de novas funcionalidades, como um sistema de pagamentos completo ou recursos de monitoramento contínuo. O projeto contribui para a literatura acadêmica ao apresentar um caso prático da aplicação de tecnologias de nuvem em um contexto de negócio relevante.

REFERÊNCIAS

KUBERNETES. **Kubernetes Documentation**. Disponível em: https://kubernetes.io/docs/. Acesso em: 25 ago. 2025.

DOCKER. **Docker Documentation**. Disponível em: https://docs.docker.com/. Acesso em: 25 ago. 2025.

POSTGRESQL. **PostgreSQL Documentation**. Disponível em: https://www.postgresql.org/docs/. Acesso em: 25 ago. 2025. NODE.JS. **Node.js Documentation**. Disponível em: https://nodejs.org/en/docs/. Acesso em: 25 ago. 2025.

BOOTSTRAP. **Bootstrap Documentation**. Disponível em: https://getbootstrap.com/docs/. Acesso em: 25 ago. 2025. CLOUDINARY. **Cloudinary Documentation**. Disponível em: https://cloudinary.com/documentation. Acesso em: 25 ago. 2025.

MERCADO PAGO. **Mercado Pago Developers**. Disponível em: https://www.mercadopago.com.br/developers/pt. Acesso em: 25 ago. 2025.

GOOGLE CLOUD PLATFORM. **Google Cloud Documentation**. Disponível em: https://cloud.google.com/docs. Acesso em: 25 ago. 2025.

ATLASSIAN. **Arquitetura de microsserviços**. Disponível em: https://www.atlassian.com/br/microservices/microservices-architecture. Acesso em: 15 set. 2025.

MICROSOFT LEARN. **Arquitetura de microsserviços no serviço Kubernetes do Azure**. Disponível em: https://learn.microsoft.com/pt-pt/azure/architecture/reference-architectures/containers/aks-microservices/aks-microservices. Acesso em: 15 set. 2025.

ALURA. **Kubernetes: como usá-lo para gerenciar suas aplicações**. Disponível em: https://www.alura.com.br/artigos/kubernetes. Acesso em: 15 set. 2025.