Chapter 4

Kubernetes - Orquestração de Containers

Angelo Resplandes Rodrigues, Caius Souza Ribeiro, Igor Santos, Souza Correa, Lucas Barros Mota, Luís Otávio de Siqueira Pimentel, Marcus Vinícius Rangel Coelho, Martiniano Gomes Barros, Cirqueira Neto, Michelangelo de Carvalho Costa and Vitória Leda Nogueira

Abstract

This paper discusses Kubernetes, an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. 1 The concepts of virtualization and containers are explored, highlighting the advantages of Kubernetes, such as auto-scalability, high availability, and portability. The architecture of Kubernetes, its main components (control plane and nodes), functionalities, and the challenges of its implementation, such as the learning curve and resource overhead, are presented. The article includes a practical guide for deploying a Node.js application using Docker and Kubernetes, demonstrating the creation of Docker images, Kubernetes configuration, and application deployment.

Resumo

Este trabalho aborda o Kubernetes, uma plataforma de código aberto para orquestração de containers, que automatiza o deploy, escalabilidade e gerenciamento de aplicações conteinerizadas. Explora-se o conceito de virtualização e containers, destacando as vantagens do Kubernetes, como auto escalabilidade, alta disponibilidade e portabilidade. Apresenta-se a arquitetura do Kubernetes, seus principais componentes (plano de controle e nós), funcionalidades e os desafios da sua implementação, como a curva de aprendizado e sobrecarga de recursos. O artigo inclui um guia prático de deploy de uma aplicação Node.js utilizando Docker e Kubernetes, demonstrando a criação de imagens Docker, configuração do Kubernetes e deploy da aplicação.

4.1. Introdução

Kubernetes é uma plataforma de código aberto (Open Source) que visa automatizar a implantação, escalabilidade e gerenciamento de aplicativos em contêineres. Seu nome é

derivado do termo grego que significa "timão" ou "piloto", analogia à sua principal função de orquestrar e dirigir a execução de contêineres em um ambiente distribuído.

A plataforma foi originalmente criada pelo Google, com base em mais de 15 anos de experiência no gerenciamento de clusters de servidores, utilizando o sistema Borg. Este sistema, embora de uso interno, forneceu a base tecnológica para a criação do Kubernetes, que foi projetado para superar as limitações do modelo tradicional de infraestrutura e permitir a automação de tarefas complexas de gestão de contêineres, como o escalonamento, o balanceamento de carga e a recuperação de falhas.

Atualmente, Kubernetes é mantido pela Cloud Native Computing Foundation (CNCF), uma organização sem fins lucrativos que visa promover e evoluir tecnologias de nuvem nativa.

4.2. Explicação teórica

4.2.1. Virtualização

Virtualização é a técnica que permite a execução de múltiplos sistemas operacionais ou ambientes isolados em um único hardware físico. Isso é feito por meio de uma camada de software chamada hipervisor ou monitor de máquina virtual (VMM), que cria e gerencia máquinas virtuais (VMs).[TANENBAUM 2016]

A virtualização traz várias vantagens, como melhor utilização dos recursos, isolamento entre sistemas, maior segurança e flexibilidade na execução de aplicações. Existem diferentes tipos de virtualização, como:

- **Virtualização total** – O sistema operacional convidado é executado sem modificações dentro de uma VM.
- **Paravirtualização** – O sistema operacional convidado é modificado para interagir melhor com o hipervisor.
- **Virtualização a nível** de sistema operacional (containers) – Compartilha o mesmo núcleo do sistema operacional, permitindo um isolamento mais leve.
- **Virtualização a nível** de sistema operacional (containers) – Compartilha o mesmo núcleo do sistema operacional, permitindo um isolamento mais leve.

Essa tecnologia é amplamente usada em computação em nuvem, servidores e desenvolvimento de software.

4.2.2. Container

Containers são uma forma de virtualização a nível de sistema operacional, na qual múltiplos aplicativos podem ser executados isoladamente dentro de um mesmo núcleo do SO. Diferente das máquinas virtuais (VMs), que simulam um hardware completo, os containers compartilham o kernel do sistema operacional, tornando-os mais leves e eficientes pois cada contêiner mantém seu próprio sistema de arquivos, enquanto compartilha CPU, memória e espaço de processos. Além disso, como são independentes da infraestrutura

subjacente, podem ser facilmente migrados entre diferentes ambientes, incluindo nuvens e distribuições de sistemas operacionais. [TANENBAUM 2016]

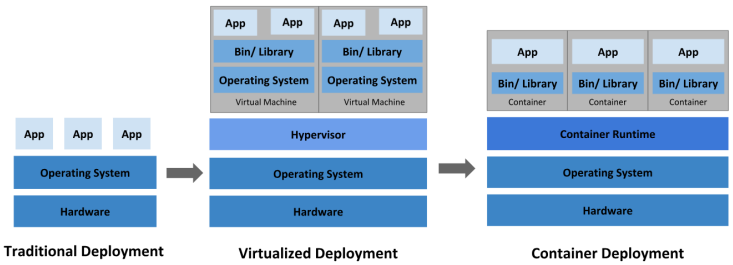

Figure 4.1. Comparativos dos três ambientes tradicional, virtualização e contêiner

4.2.3. Kubernetes

Com o aumento da adoção dos contêineres, surgiu a necessidade de ferramentas para automatizar sua gestão. Dessa necessidade, nasceram os orquestradores de contêineres, que permitem agrupar e gerenciar múltiplos contêineres em clusters, garantindo balanceamento de carga, escalabilidade e alta disponibilidade. Entre essas ferramentas, destaca-se o Kubernetes.

O Kubernetes oferece uma estrutura para executar sistemas distribuídos de forma resiliente. Ele cuida do escalonamento e da recuperação à falha de sua aplicação, fornece padrões de implantação e muito mais. Por exemplo, o Kubernetes pode gerenciar facilmente uma implantação no método canário para seu sistema. [KUBERNETES 2025]

4.3. Arquitetura do Kubernetes

4.3.1. Features

As funcionalidades disponibilizadas pelo Kubertenes são:

- **Descoberta de serviço e balanceamento de carga**: O Kubernetes pode expor um contêiner usando o nome DNS ou seu próprio endereço IP. Se o tráfego para um contêiner for alto, o Kubernetes pode balancear a carga e distribuir o tráfego de rede para que a implantação seja estável.
- **Orquestração de armazenamento**: O Kubernetes permite que você monte automaticamente um sistema de armazenamento de sua escolha, como armazenamentos locais, provedores de nuvem pública e muito mais.
- **Lançamentos e reversões automatizadas**: Você pode descrever o estado desejado para seus contêineres implantados usando o Kubernetes, e ele pode alterar o estado real para o estado desejado em um ritmo controlado. Por exemplo, você pode automatizar o Kubernetes para criar novos contêineres para sua implantação, remover os contêineres existentes e adotar todos os seus recursos para o novo contêiner.

- **Empacotamento binário automático**: Você fornece ao Kubernetes um cluster de nós que pode ser usado para executar tarefas nos contêineres. Você informa ao Kubernetes de quanta CPU e memória (RAM) cada contêiner precisa. O Kubernetes pode encaixar contêineres em seus nós para fazer o melhor uso de seus recursos.

- **Autocorreção**: O Kubernetes reinicia os contêineres que falham, substitui os contêineres, elimina os contêineres que não respondem à verificação de integridade definida pelo usuário e não os anuncia aos clientes até que estejam prontos para servir.

- **Gerenciamento de configuração e de segredos**: O Kubernetes permite armazenar e gerenciar informações confidenciais, como senhas, tokens OAuth e chaves SSH. Você pode implantar e atualizar segredos e configuração de aplicações sem reconstruir suas imagens de contêiner e sem expor segredos em sua pilha de configuração.

[KUBERNETES 2025]

4.3.2. Elementos Principais

Ao implantar o Kubernetes, você obtém um **cluster**.

Um cluster é uma estrutura de servidores de processamento, chamados de nós (worker node), interconectados trabalhando em harmonia. O cluster é dividido em duas partes distintas: o plano de controle e os nós. Os nós, tanto físicos quanto virtuais, formam a base operacional do cluster e cada um desses nós hospedam pods, que são componentes de uma aplicação. Esses pods, ao agruparem contêineres, facilitam a gestão eficiente de recursos, proporcionando uma abordagem flexível e escalável para a execução de aplicativos no ambiente do Kubernetes. [FREIRE 2021]

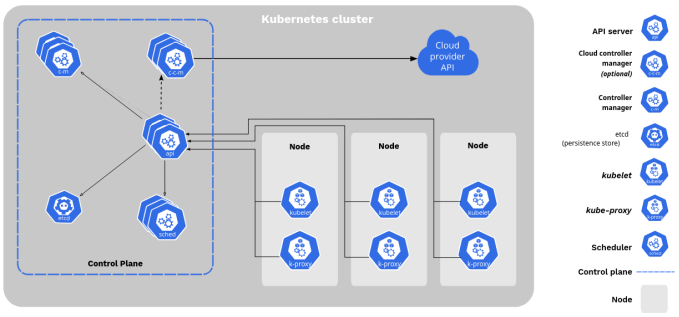

Figure 4.2. Componentes de um cluster do Kubernetes

4.3.2.1. Componentes da camada de gerenciamento

Os componentes da camada de gerenciamento tomam decisões globais sobre o cluster (por exemplo, alocação de Pods), bem como detectam e respondem aos eventos do cluster, são eles:

- **etcd**: Responsável por armazenar os dados de configuração acessíveis pelo servidor master da API do Kubernetes por meio de HTTP simples ou JSON API.
- **API Server**: Atua como o ponto central de gerenciamento do master node do Kubernetes, facilitando a comunicação entre os diferentes componentes e garantindo a integridade do cluster.
- **Controller Manager**: Assegura que o estado real do cluster esteja alinhado com o estado desejado, ajustando dinamicamente a escala dos workloads.
- **Scheduler**: Responsável por alocar workloads nos nodes apropriados, garantindo que, neste caso, todas as cargas de trabalho sejam executadas localmente no host do usuário.

4.3.2.2. Componentes do nó

Os componentes do nó são executados em todos os nós, mantendo os Pods em execução e fornecendo o ambiente de execução do Kubernetes.

- **Kubelet**: Recebe as especificações dos pods (grupos de containers com recursos compartilhados) do API Server e gerencia sua execução no host.
- **Kube-proxy**: O kube-proxy é responsável por manter as regras de rede no cluster Kubernetes, permitindo a comunicação entre diferentes serviços e pods. Ele gerencia o balanceamento de carga e a tradução de endereços de rede (NAT), garantindo que as requisições sejam encaminhadas corretamente dentro do cluster.
- **Container runtime**: (O tempo de execução do contêiner é o software responsável pela execução dos contêineres)

4.3.2.3. Complementos (addons)

Complementos (addons) usam recursos do Kubernetes (DaemonSet, Deployment, etc) para implementar funcionalidades do cluster.

- **DNS do Cluster**: Essencial para o funcionamento do Kubernetes, fornece registros DNS para serviços do cluster e é utilizado automaticamente pelos contêineres.
- **Web UI (Dashboard)**: Interface web para gerenciamento e solução de problemas de aplicações e do próprio cluster.
- **Monitoramento de Recursos**: Registra métricas dos contêineres em um banco central e disponibiliza uma interface para análise.
- **Logging a Nível do Cluster**: Armazena logs dos contêineres de forma centralizada, permitindo navegação e pesquisa.

[KUBERNETES 2025]

4.4. Vantagens e Desvantagens

Vantagem	Desvantagem
Escalabilidade automática: Kubernetes pode escalar aplicações automaticamente conforme a demanda de tráfego.	**Curva de aprendizado**: A complexidade do Kubernetes pode ser um desafio, especialmente para equipes sem experiência prévia.
Alta disponibilidade: Oferece ferramentas para garantir que as aplicações continuem funcionando, mesmo em caso de falhas de componentes.	**Sobrecarga de recursos**: Kubernetes pode exigir recursos significativos de hardware e rede, o que pode ser um fator limitante em algumas implementações.
Gerenciamento simplificado: Facilita o controle de múltiplos containers através de um único ponto de gerenciamento.	**Gestão de estado**: Embora Kubernetes facilite o gerenciamento de containers sem estado, a gestão de estado persistente pode exigir configurações adicionais.
Portabilidade: Kubernetes é agnóstico a provedores de infraestrutura, permitindo a execução de containers em qualquer ambiente.	

4.5. Atividade Prática

4.5.1. Códigos Utilizados na Atividade

- server.js
- Dockerfile
- deployment.yaml
- kind-config.yaml
- service.yaml

4.5.2. Pré-requisitos

Conheça os Requisitos:

- Sistemas Operacionais Windows cliente e servidor suportados (pode ser executado em sistemas operacionais mais antigos);
- PowerShell v2+;
- .NET Framework 4.8 (a instalação tentará instalar o .NET 4.8 se você não o tiver instalado);
- 8GB de memória RAM ou mais.

Antes de iniciar, certifique-se de instalar os seguintes pacotes:

- Chocolatey Abra o cmd como administrador e copie o comando no site a seguir:

https://chocolatey.org/install#individual

Antes do comando escreva: powershell.exe

Exemplo: powershell.exe Set-ExecutionPolicy Bypass...

Após a instalação do choco, é necessário fechar e abrir o cmd (administrador)

- Kind (Kubernetes in Docker)

```
choco install kind
```

Após a instalação do docker, é necessário reiniciar o computador.

- Kubectl (Kubernetes CLI)

```
choco install kubernetes-cli
```

- NodeJS

```
choco install nodejs.install
```

- Instalação do GIT

```
choco install git.install
```

4.5.3. Configuração do Projeto

- Clonar o Repositório

```
git clone https://github.com/Michelangelo-Costa/deploy-node-app.git
```

- Navegar até a pasta do Projeto

```
cd deploy-node-app
```

4.5.4. Instalar as Dependências

No diretório do projeto, execute:

```
npm install
# Instala Express e outras dependencias necessarias.
# A pasta node_modules sera criada.
```

4.5.5. Executar Aplicação

1. Inicie aplicação localmente:

```
node server.js
```

2. Teste a aplicação no navegador:

   - Página Inicial: `http://localhost:3000`
   - Simular Falhas: `http://localhost:3000/unstable`

4.5.6. Criação e Deploy no Docker + Kubernetes

- Criar imagem Docker da Aplicação

```
docker build -t node-app:v1 .
```

- Criar Cluster Kubernetes com Kind

```
kind create cluster --config kind-config.yaml --name my-kind-1
```

- Adicionar a Imagem Docker ao Cluster

```
kind load docker-image node-app:v1 --name my-kind-1
```

- Em outro terminal navegue até a pasta do Projeto

```
cd deploy-node-app
```

- Deploy no Kubernetes

```
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
```

- Inicie a aplicação localmente:

```
node server.js
```

4.5.7. Comandos Úteis

- Ver informações dos pods

```
kubectl get pods -o wide
```

- Ver logs de um pod

```
kubectl logs -p idpod
# Substituir "idpod" por id exibido no navegador
# Exemplo: kubectl logs -p node-app-6fc55684ff-8qz4n
```

- Ver informações dos nós do cluster

```
kubectl get nodes
```

4.5.8. Encaminhamento de Porta para Acessar a Aplicação nos Workers

```
kubectl port-forward svc/node-app-service 8080:3000
```

Agora você pode acessar a aplicação em: `http://localhost:8080`

4.6. Distribuição de Tarefas

Tarefa	Responsáveis
Organização do roteiro e abrangência teórica	Marcus Vinicius Rangel Coelho, Vitoria Leda Nogueira
Qualidade da apresentação teórica	Martiniano Gomes Barros Cirqueira Neto
Execução da Atividade Pratica	Michelangelo Costa, Igor Santos
Capacidade de solucionar dúvidas (Monitores)	Caius Souza Ribeiro, Luis Otavio de Siqueira Pimentel, Lucas Barros Mota
Conclusão e interação com o público	Ângelo Resplandes Rodrigues

References

[CHOCOLATEY 2025] CHOCOLATEY (2025). Instalação. https://chocolatey.org/install#individual. Acesso em: 06 fev. 2025.

[FREIRE 2021] FREIRE, J. E. L. (2021). Orquestração de containers usando kubernetes e docker swarm (mestrado) — universidade da beira interior (portugal), 2021. https://ubibliorum.ubi.pt/bitstream/10400.6/11091/1/7913_17373.pdf. Acesso em: 5 fev. 2025.

[IBM 2024] IBM (2024). Containerização. https://www.ibm.com/br-pt/think/topics/containerization. Acesso em: 5 fev. 2025.

[KIND 2025] KIND (2025). Quick start: Installation. kind - kubernetes. https://kind.sigs.k8s.io/docs/user/quick-start/. Acesso em: 06 fev. 2025.

[KUBERNETES 2025] KUBERNETES (2025). Documentação oficial do kubernetes. https://kubernetes.io/pt-br/docs/home/. Acesso em: 5 fev. 2025.

[TANENBAUM 2016] TANENBAUM, A. S. (2016). *Sistemas Operacionais Modernos*, chapter 7, pages 325–3456. São Paulo: Pearson.