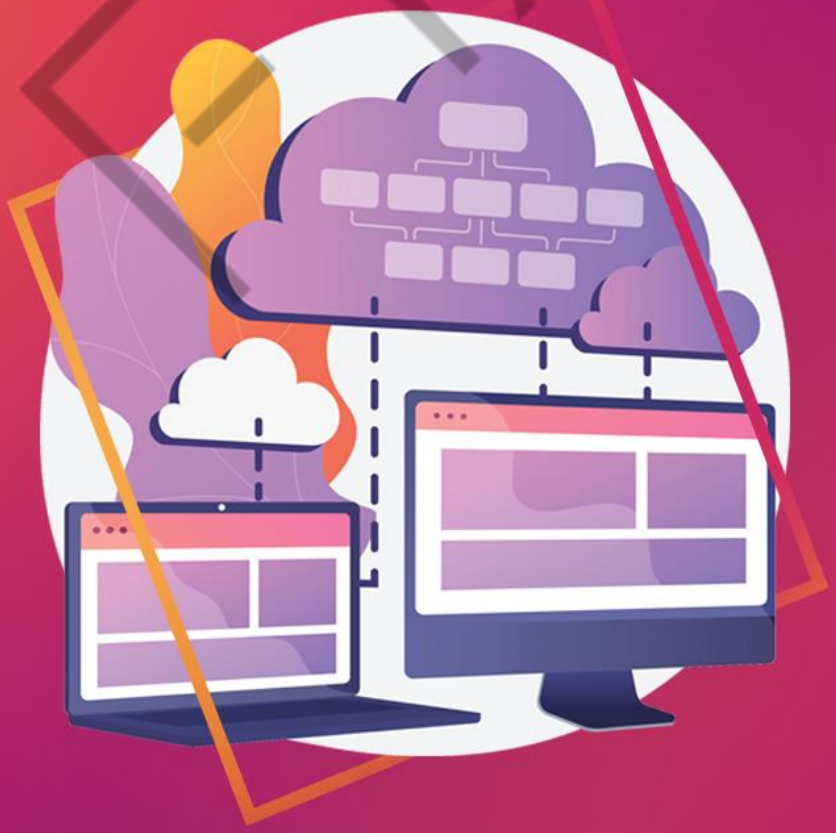
FIAP ON

CLOUD FUNDAMENTALS, ADMINISTRATION AND SOLUTION ARCHITECT

KUBERNATES, openshift e openstack

CHRISTIANE DE PAULA REIS



08

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

8 KUBERNETES, OPENSHIFT E OPENSTACK

Para gerir o complexo funcionamento dos diversos microsserviços de uma aplicação, existem **gestores de containers** e **gestores de recursos**.

- Gestores de containers são ferramentas responsáveis por gerir o que está em execução em cada *cluster*, viabilizando a escalabilidade e a disponibilidade dos serviços. Neste cenário, vamos falar sobre o Kubernates.
- Gestores de recursos são ferramentas utilizadas para orquestrar os componentes da aplicação, como um mecanismo de armazenamento de banco de dados, um serviço de registro em log, um servidor de integração contínua, um servidor web, componentes em nuvem, entre outros. Neste cenário, vamos falar sobre o OpenShift e o OpenStack.

8.1 Kubernetes

Kubernetes é uma plataforma de código aberto, que foi criada e desenvolvida pelos engenheiros da Google, para **orquestrar** e **gerenciar um *cluster* de containers** Linux como um único sistema. Suas principais características são:

- Organização em *clusters*.
- Automatização de execução de containers.
- Replicação de containers.
- Autoescalonamento dos containers.
- Suporte a containers Docker.
- Suporte a volumes persistentes remotos.
- Capacidade de integração com serviços de segurança, rede, armazenamento, monitoramento, medição e outros.

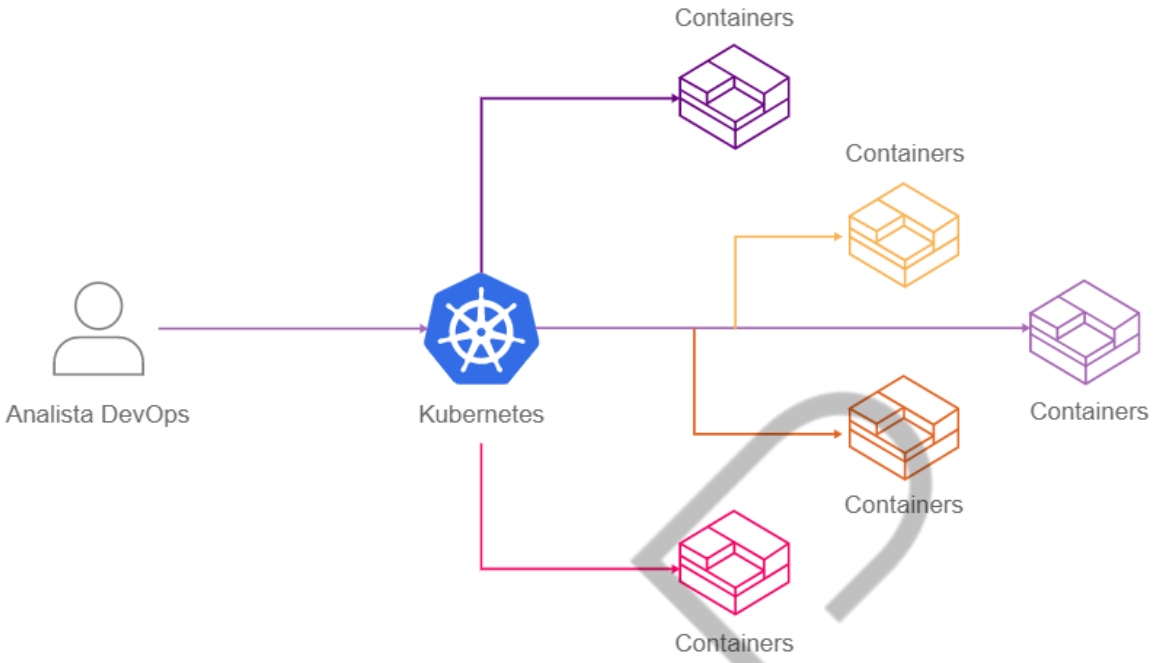


Figura 8.1 – Representação do Kubernetes
Fonte: Google Imagens, adaptado por FIAP (2019)

Dentre as funcionalidades desempenhadas pelo Kubernetes, temos:

- Gerenciamento e automatização de grande parte das implantações e atualizações de aplicativos.
- Otimização do uso do hardware, aumentando a disponibilidade dos recursos para executar os aplicativos.
- Capacidade de orquestrar containers em *clouds* privadas, públicas ou híbridas e também em múltiplos *hosts*.
- Garante a integridade e assegura que os aplicativos sejam autorrecuperáveis em seus containers, com reinício, posicionamento, escalonamento automáticos e replicação.
- É mais ágil para escalar aplicativos em containers e todos os recursos relacionados.

8.1.1 Arquitetura do Kubernetes

Vamos entender alguns conceitos importantes e termos específicos que permeiam o funcionamento do Kubernetes. Como você pode observar na Figura "Componentes da Arquitetura Kubernetes", a arquitetura do Kubernetes possui um servidor mestre – **Kubernetes Master**; um servidor de node – **Kubernetes Node**; e uma interface de comunicação – **Kubectl** entre DevOps e servidor.

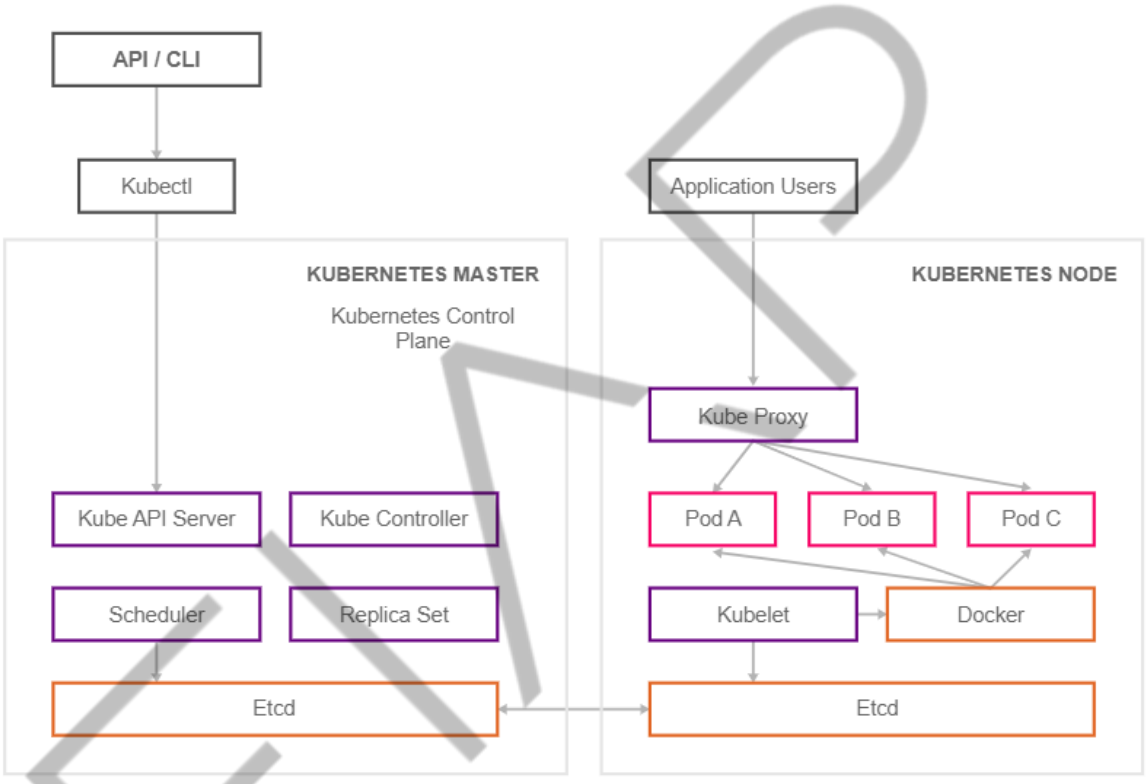


Figura 8.2 – Componentes da Arquitetura Kubernetes
Fonte: Google Imagens, adaptado por FIAP (2019)

Interface de comunicação – Kubectl

- **Kubectl:** é uma interface de linha de comando que se comunica com um *cluster* Kubernetes através de uma API a partir de um computador local (sob comando de um DevOps).

Componentes do Servidor Mestre

- **Etcd:** usado como armazenamento de backup do Kubernetes. É uma estrutura de armazenamento baseada em chave/valor e pode ser configurada e disponibilizada de forma global para vários *nodes*.

- **Kube API Server**: responsável por expor a API do Kubernetes através do uso do JSON sobre HTTP.
- **Scheduler:** realiza um trabalho de gerenciamento de *pod* recém-criado para designar em qual nó deve ser executado, baseando-se na disponibilidade de recursos.
- **Kube Controller ou Controller Manager:** é um processo que executa os controladores, como o Replication Controller, Node Controller, Endpoints Controller e o Service Account & Token Controllers. Cada um desses controladores é um processo, porém todos são compilados em um único binário e executados em um único processo.
- **ReplicaSet**: responsável por garantir a disponibilidade a qualquer momento de um número específico de réplicas de *pods*. O ReplicaSet veio como alternativa ao ReplicationController.
- **ReplicationController**: desempenha as mesmas funcionalidades e comportamentos do ReplicaSet, com exceção de que o ReplicationController não oferece suporte a requisitos de seletor de rótulos baseado em conjuntos. Por esse motivo, o ReplicaSet é o preferido para a função.

Componentes do Servidor de Node

- **Kubernetes Node**: é uma VM (*virtual machine*) ou uma máquina física em um determinado *cluster,* controlada no Kubernetes. É responsável pela execução dos seus aplicativos que são agrupados pelos *pods*. Um nó contém os serviços Docker, Kubelet e Kube-Proxy.
- **Kubelet**: é um pequeno serviço executado em cada nó no *cluster* e comunica-se com os componentes do servidor mestre para autenticar e receber comandos e trabalho.
- **Kube-Proxy**: como o próprio nome diz, este componente é um proxy de rede responsável por encaminhar solicitações e, assim como o Kubelet, também é executado em cada nó no *cluster.*

- **Docker**: responsável por rodar e executar os containers sob comando do Kubernetes.
- **Pod**: é um objeto simples que encapsula um ou mais containers controlados por uma única aplicação.

8.1.2 Funcionamento do Kubernetes

O Kubernetes baseia seu funcionamento em camadas: a camada mais alta desempenha um papel de *gateway* para acesso ao container; e os níveis mais baixos, denominados *pods*, criam uma camada extra de abstração que contém toda a complexidade necessária para trabalhar, por exemplo, com rede, armazenamento compartilhado, entre outros serviços.

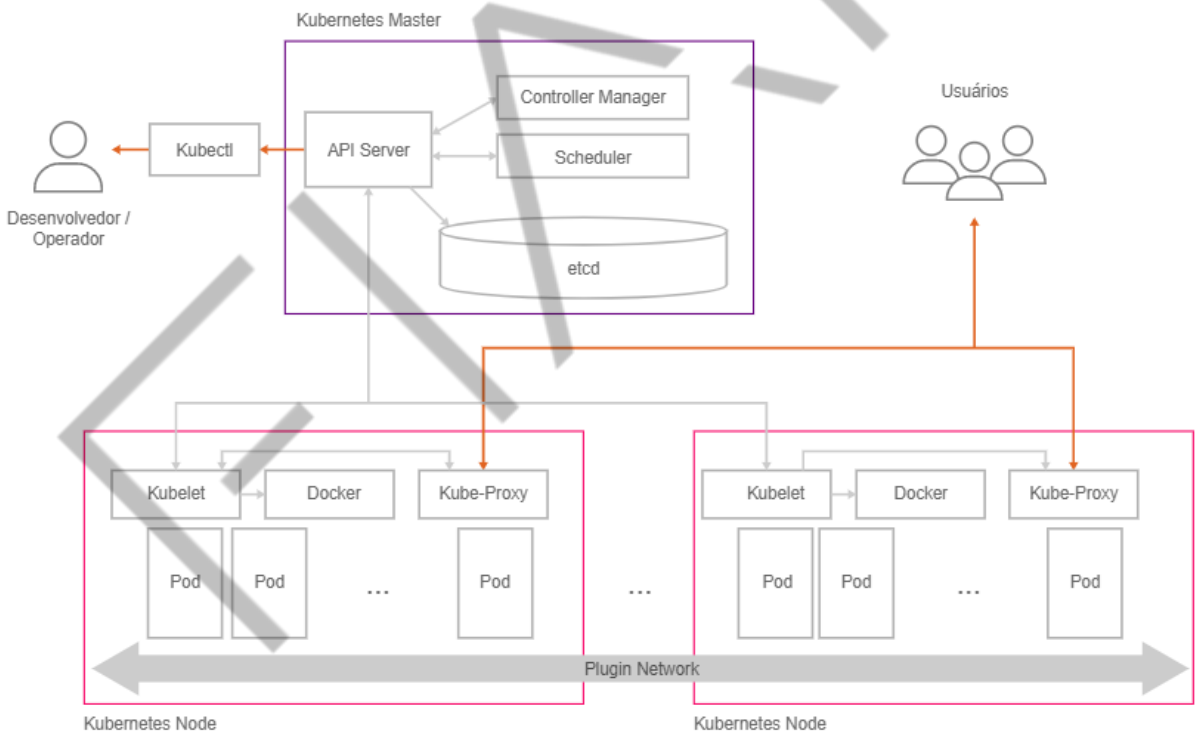


Figura 8.3 – Funcionamento do Kubernetes
Fonte: Google Imagens, adaptado por FIAP (2019)

Os DevOps interagem com a camada mais alta através da comunicação com a API Server, a principal API do servidor. O usuário final interage com os *pods,* através de *clients* e bibliotecas, os Kube-Proxy.

É necessário haver um plano declarativo de controle (Kubernetes Control Plane), em formato JSON ou YAML, para definir o que deve ser criado e como deve ser gerenciado.

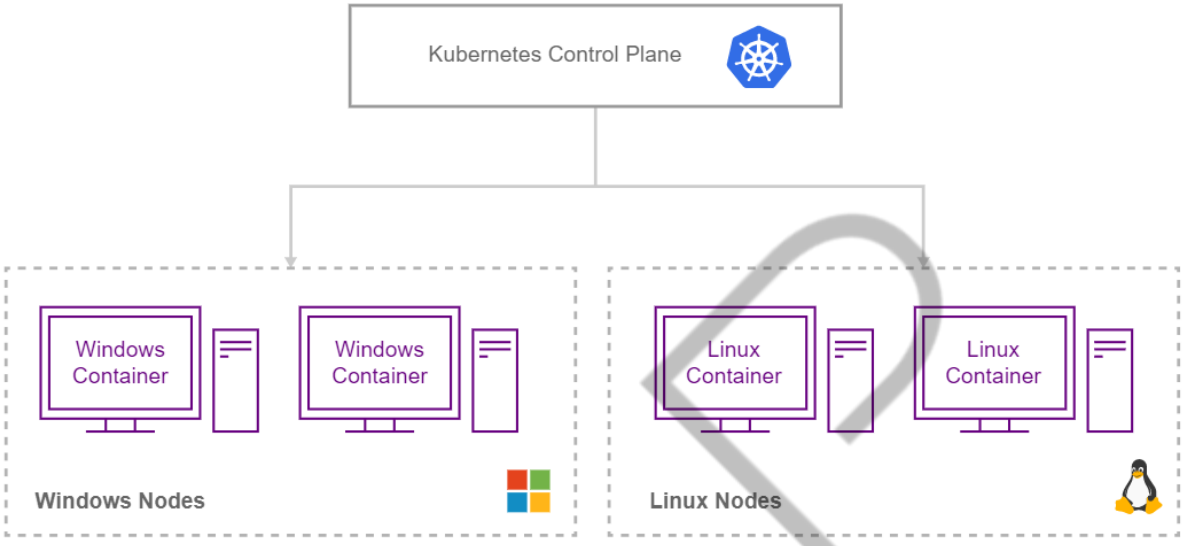


Figura 8.4 – Kubernetes Control Plane com Linux e Windows
Fonte: Google Imagens, adaptado por FIAP (2019)

O plano de controle é utilizado pelo servidor mestre para identificar qual é o estado atual do sistema e quais são os requisitos para executar o aplicativo ou serviço na infraestrutura (dentro do *pod*). Como você viu, os *pods*, que contêm o aplicativo ou serviço que foi definido pelo usuário, representam a última camada do Kubernetes.

8.2 OpenShift e Azure Red Hat Open Shift

O **OpenShift** tem grande adesão no mundo corporativo, porque oferece **componentes além do Kubernetes, ofertando-o** como projeto encapsulado**.** Surgiu da necessidade de trabalhar com **mecanismos de controle** e **processos mais burocráticos** deficientes no Kubernetes.

Ele tem como características importantes o fato de ser inerentemente um **sistema distribuído**, o que beneficia o desenvolvimento e a implementação de aplicativos na nuvem, e o de ser **fundamentado na arquitetura baseada em microsserviços**, que é muito importante para prover funcionalidades diversas, de acordo com a necessidade de que cada projeto.

OpenShift é uma **plataforma de código aberto** relacionada à nuvem, que usa a tecnologia de **conteinerização,** seguindo os principais conceitos do Kubernetes.

Além da **orquestração de containers** de forma independente da plataforma na qual os containers são executados, o OpenShift auxilia no processo de **gerenciamento de containers** e também disponibiliza uma **interface do usuário** baseada na web**.** Oferece **Plataforma como Serviço** (*PaaS – Platform as a Service*) e disponibiliza **integração facilitada com outras ferramentas e SDKs** (*Software Development Kit*) para diferentes linguagens.

Dentre a infinidade de serviços oferecidos pelo OpenShift, temos:

- Recurso de CI / CD (*Continuos Integration / Continous Delivery*) integrados com Jenkins, solução de software universal, genérica e madura que disponibiliza, por exemplo, autenticação OAuth e diversos outros recursos importantes que tornam todo o CI / CD mais “leves”.
- Políticas-padrão de segurança rígidas, tendo como parte integrante o Controle de Acesso Baseado em Função (*RBAC – Role Based Access Control*) para restringir o acesso a usuários autorizados.
- *Health check* das aplicações e containers que estão sendo gerenciados.
- Permite a execução em serviços de nuvem, como Google Compute Engine, AWS, entre outros.
- Gerenciamento de configurações e *logs*.
- Facilidades de implantação e testes, excluindo a necessidade de manuseio de servidores físicos ou virtuais.

A plataforma foi criada pela Red Hat Inc, que firmou parceria com a Microsoft em maio de 2019, para oferecer pela primeira vez ao mercado o serviço **Azure Red Hat Open Shift.** A novidade dispõe de uma *cloud* híbrida, neste caso, uma *cloud* pública operando ao lado da virtualização.

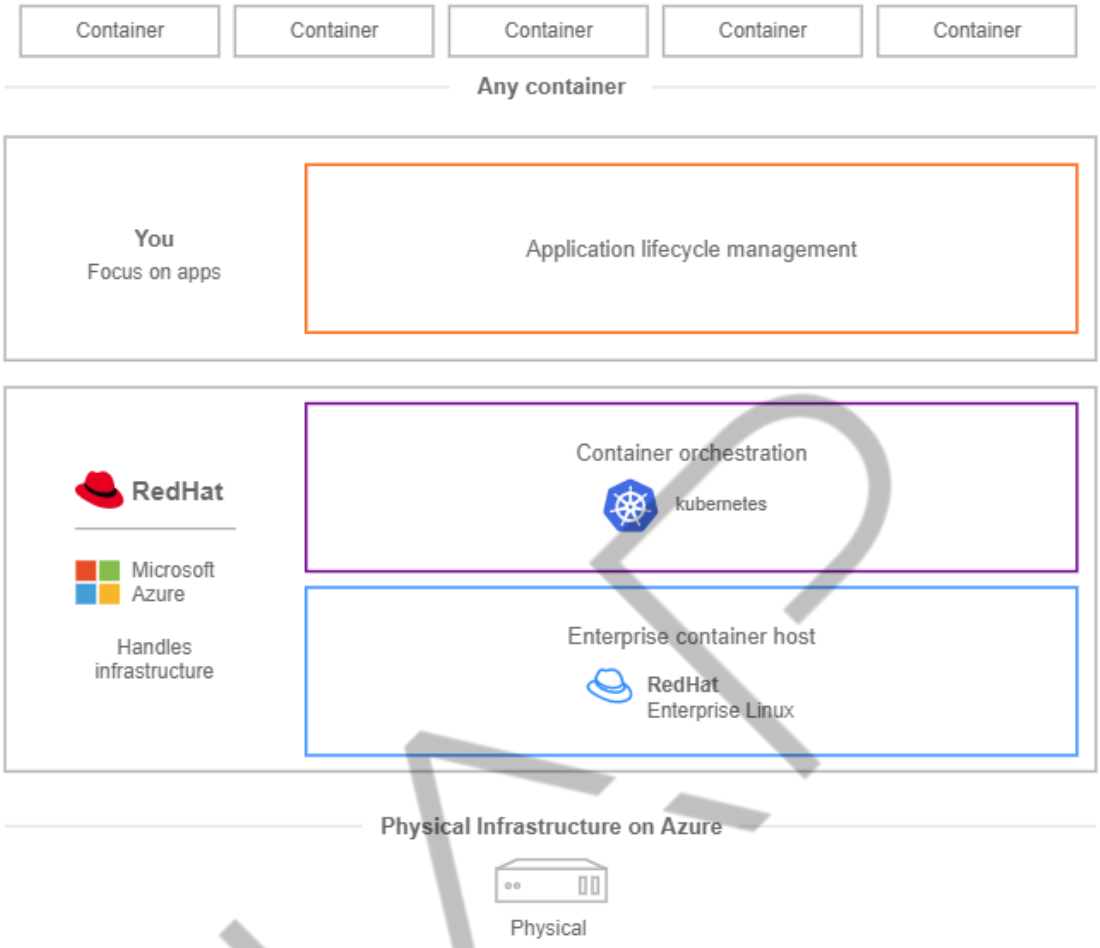


Figura 8.5 – Representação do Red Hat OpenShift do Azure
Fonte: OpenShift (2019)

A parceria disponibiliza o **Kubernetes de nível empresarial,** sendo desenvolvido, operado, suportado em conjunto e totalmente gerenciado na *cloud* pública Microsoft Azure. Possibilita às empresas de TI trabalhar com a Red Hat OpenShift Container Platform em seus *datacenters* e operar com a escala e a força do Microsoft Azure.

Você pode ver a seguir como se apresenta a Arquitetura do OpenShift.

8.2.1 Arquitetura do OpenShift

A infraestrutura do OpenShift contém um *node* para armazenar as aplicações. Como no exemplo da figura, temos uma estrutura baseada no Docker, em que um *node* contém aplicações que ficam dentro de seu respectivo container, agrupado pelo *pod*. Um *pod* fica hospedado em uma máquina (*node*), que faz parte do *cluster* Kubernetes, por sua vez, gerenciado pelo OpenShift.

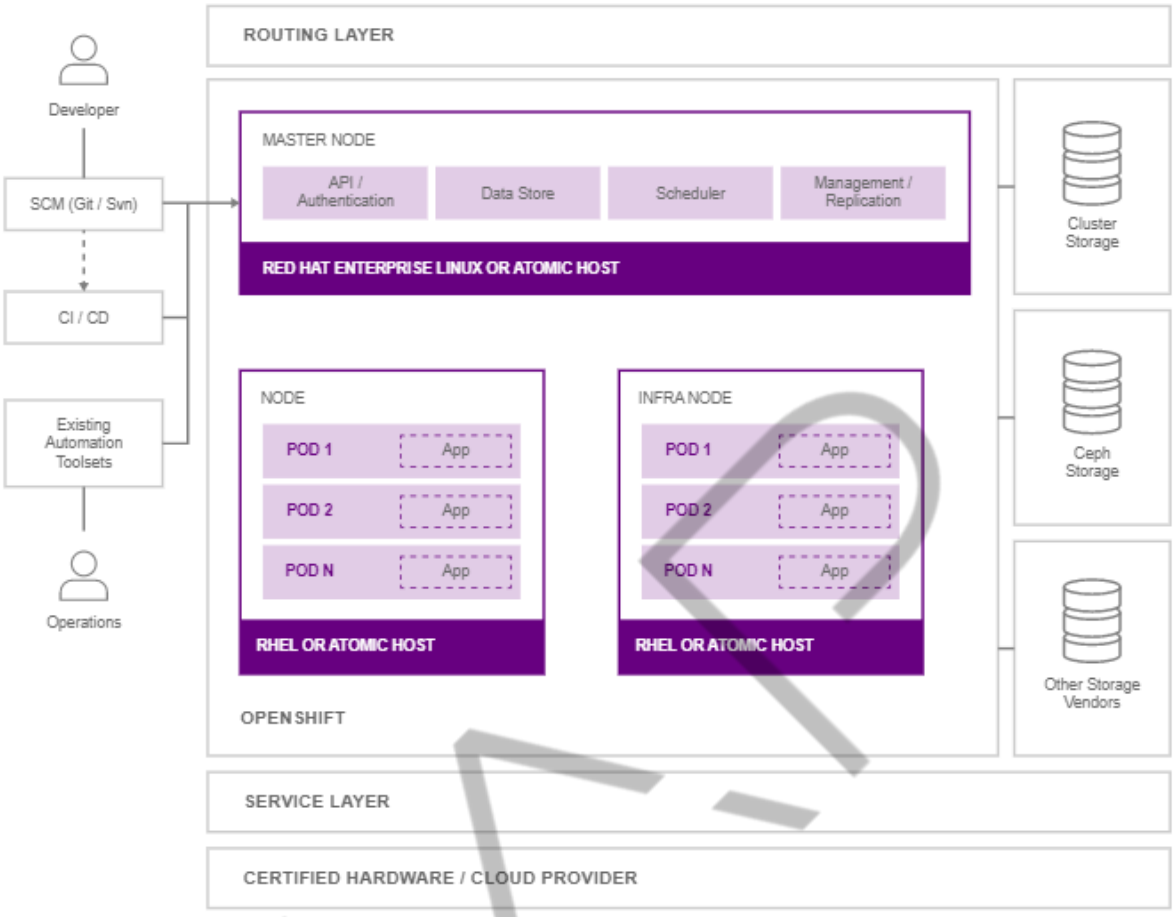


Figura 8.6 – Componentes da Arquitetura do OpenShift
Fonte: RedHat (2019)

Componentes do Nó Mestre (ou Node Master)

- **API/Authentication**: responsável pelo controle de acesso às APIs do OpenShift e do Kubernetes, com um processo de autenticação baseado em certificados SSL (*Secure Sockets Layer*) e no padrão OAuth.
- **Data Store**: armazena o estado e outras informações dos componentes do OpenShift. Geralmente, utiliza a estrutura de armazenamento etcd.
- **Scheduler**: faz a distribuição das cargas de trabalho entre os nós dos *clusters* de componentes.
- **Management/Replication**: é um processo que, além de realizar replicação, executa de tempos em tempos com a finalidade de coletar informações de estado dos elementos do *cluster* e dos demais componentes, atualizar o estado e armazenar estes estados no *Data Store* (etcd). O processo Management/Replication é gerenciado por *controllers*, sendo os principais

controllers: Replication Controller, Endpoint Controller, Namespace Controller e Service Account Controller.

Em seguida, você vai entender como funciona o OpenShift.

8.2.2 Funcionamento do OpenShift

Assim como o Kubernetes, o OpenShift também funciona sobre um sistema baseado em camadas, no qual, cada camada possui uma determinada responsabilidade. O conjunto de camadas é o que possibilita a disponibilidade das diversas funcionalidades do OpenShift.

Para rodar, o OpenShift utiliza um cluster baseado no Kubernetes, em que seus componentes são organizados como microsserviços e disponibiliza dentro de sua infraestrutura um nó master, que hospeda os principais componentes do OpenShift.

No diagrama abaixo, temos uma visão geral das ferramentas que são integradas e orquestradas pelo OpenShift:

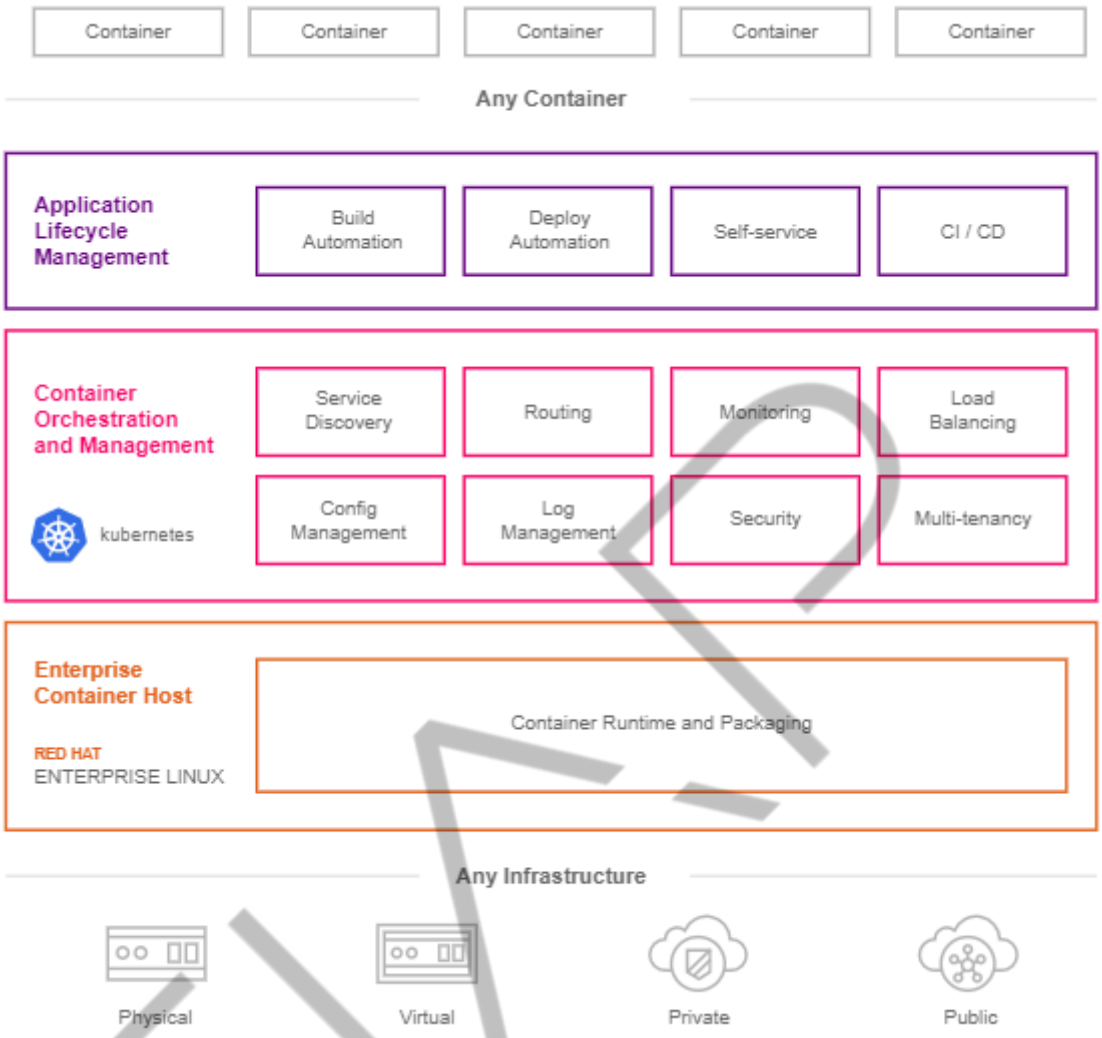


Figura 8.7 – Ferramentas do OpenShift
Fonte: RedHat (2019)

Como você pode observar no diagrama, o **OpenShift** apresenta-se como uma ferramenta qualificada para **controlar todo o ciclo de vida de uma aplicação** e **gerenciamento de containers**.

Em se tratando de tecnologia em nuvem, o gerenciamento não deve ocorrer apenas para containers, é preciso que haja também uma forma de gerenciar os recursos que compõem a infraestrutura. Para isso, existe o OpenStack.

8.3 OpenStack

OpenStack é um **gerenciador de recursos de ambiente de nuvem** que mostra aos seus usuários a possibilidade de gerenciar um container da mesma forma que uma máquina virtual. Oferece **Infraestrutura como Serviço** (*IaaS – Infrastructure as a Service),* possibilizando **compartilhamento de hardware** no ambiente, **escalabilidade**, **isolamento** e **segurança**. Assim como o OpenShift, o OpenStack é mantido pela Red Hat Inc.

"O OpenStack é uma plataforma de orquestração de recursos em nuvem computacional que, desde 2014, tem incorporado módulos e ferramentas para uso de containers" (OPENSTACK, 2018).

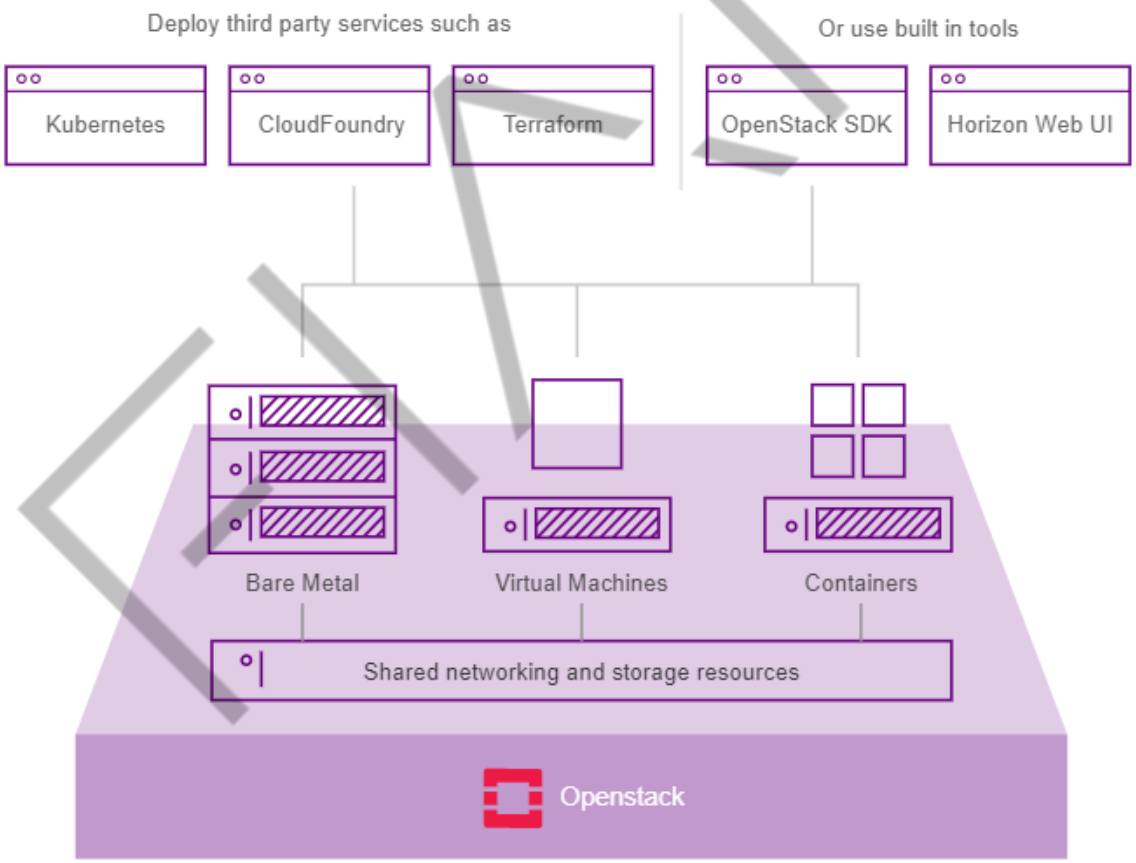


Figura 8.8 – Representação do OpenStack
Fonte: OpenStack (2019)

Suporta o provisionamento por três *hosts* diferentes:

- **Bare-metal (servidor físico):** sem virtualização, o hardware é totalmente dedicado, entregue sob demanda.

- **Máquina virtual**: virtualização tradicional em que a máquina aparece como um computador independente, inicializa kernel do SO padrão e executa um aplicativo.
- **Containers**: abordam isolamento de recursos, nos quais os aplicativos compartilham um kernel comum.

Disponibiliza para os administradores um painel com interface web que permite controle total e capacitação de seus usuários com o objetivo de atingir melhor provisionamento de recursos.

A arquitetura do OpenStack é composta por módulos que possibilitam a implantação por conteinerização.

8.3.1 Arquitetura do OpenStack

A plataforma do OpenStack foi projetada para ser escalável e flexível e seu *core* (núcleo) é formado por componentes e serviços. Fornece máquinas virtuais inicializáveis, rede, armazenamento em bloco, armazenamento de objetos e assim por diante.

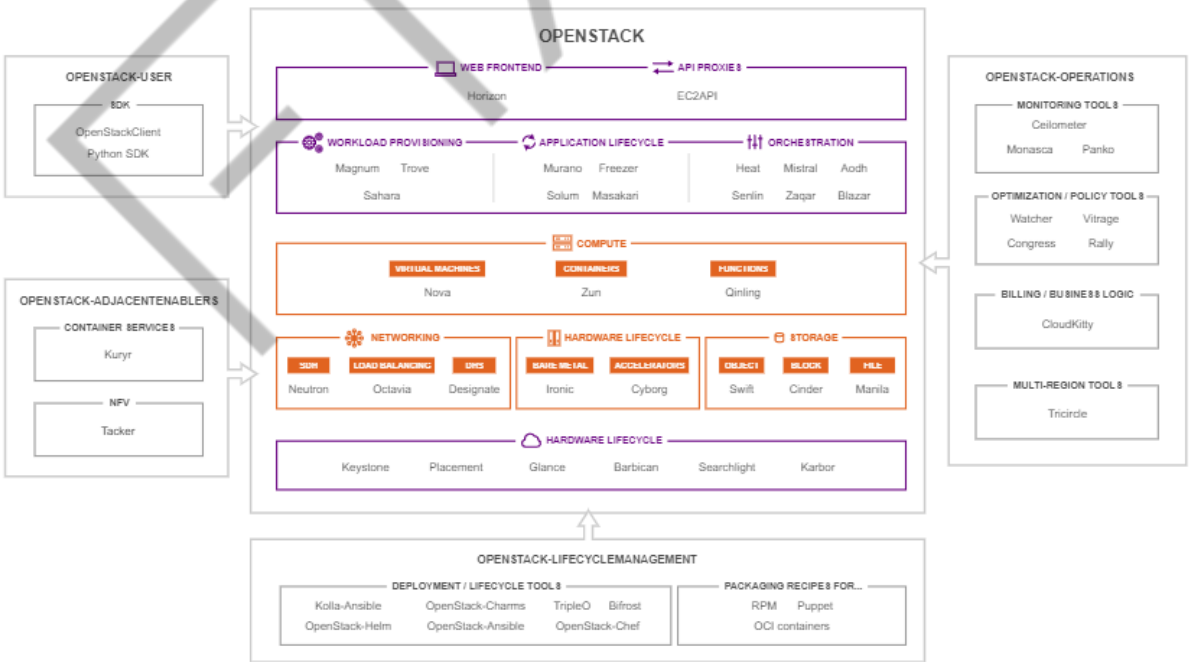


Figura 8.9 – OpenStack Landscape
Fonte: OpenStack (2019)

- **Componentes**: responsáveis pelo gerenciamento de falhas e de serviços, orquestração e por garantir alta disponibilidade dos aplicativos do usuário. Implementam as funcionalidades base da infraestrutura de nuvem.
- **Serviços**: responsáveis pela integração dos diversos componentes para fornecer uma plataforma de IaaS completa.

Módulo	Nome	Função
Compute	NOVA ZUN QINLING	Compute Service Containers Service Functions Service
Networking	NEUTRON OCTAVIA DESIGNATE	Networking Load balancer DNS service
Storage	SWIFT CINDER MANILA	Object store Block Storage Shared filesystems
Hardware Lifecycle	IRONIC CYBORG	Bare Metal Provisioning Service Lifecycle management of accelerators

Quadro 8.1 – Componentes e serviços do OpenStack
Fonte: OpenStack (2019)

Na página oficial do OpenStack é possível encontrar demais componentes e serviços, bem como suas funcionalidades e indicação de uso. Vamos ao funcionamento do OpenStack!

8.3.2 Funcionamento do OpenStack

OpenStack, como uma infraestrutura de nuvem, é sustentado pelos **pilares:** processamento (***compute***), rede (***networking***) e armazenamento (***storage***). Este SO faz uso de APIs com mecanismos comuns de autenticação para realizar tarefas de **gerenciamento** e **provisionamento**.

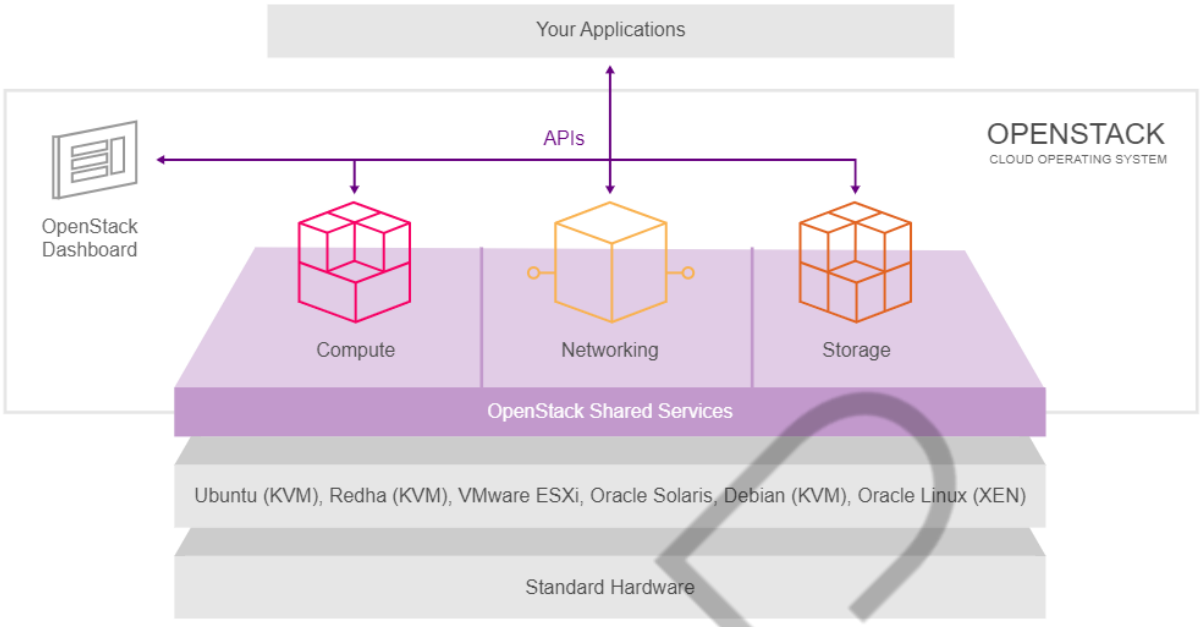


Figura 8.10 – Pilares da infraestrutura OpenStack
Fonte: OpenStack (2019)

Os serviços permitem a **personalização** de acordo com as necessidades, por exemplo, possibilita conectar e reproduzir componentes específicos. Configurando adequadamente o serviço *bare metal* com os serviços *Compute* e *Networking,* é possível provisionar máquinas virtuais e físicas por meio da API do serviço *Compute*.

8.4 Entendendo algumas comparações

8.4.1 Kubernetes e Docker

Muitos confundem os papéis e responsabilidades do Kubernetes e do Docker, e tentam fazer comparações, mas o correto é comparar Kubernetes com “Docker Swarm”, que são ferramentas de orquestração de *cluster*.

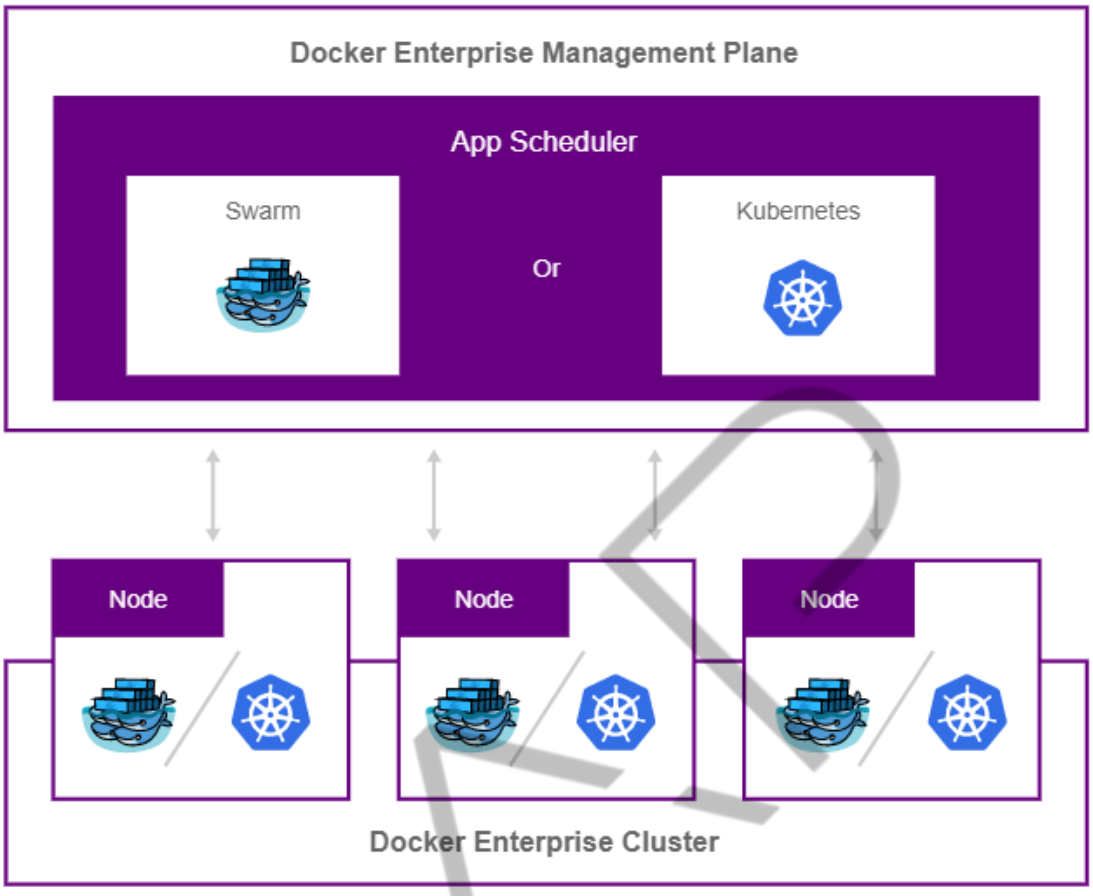


Figura 8.11 – Papel do Kubernetes e do Docker Swarm
Fonte: Docker (2019)

O papel do Kubernetes é utilizar o Docker para criar containers nos nós do *cluster* e tem como responsabilidades controlar, gerenciar e monitorar o estado dos containers Docker ao longo do *cluster*.

O Docker Swarm é nativo do Docker e desempenha as mesmas funcionalidades do Kubernates, mas apresenta algumas variações e diferenças técnicas e funcionais.

Os pontos de atenção ao utilizar o Docker Swarm são:

- Funcionalidade limitada.
- Tolerância a falhas limitada.
- Os serviços podem ser dimensionados manualmente.

Devido a essas e outras características, o Kubernetes é considerado mais confiável no mercado de tecnologia. O Kubernetes conta com recursos de dimensionamento automático e políticas de alta disponibilidade. Por esses motivos, é

muito usado por organizações de grande porte que possuem aplicativos complexos que necessitam utilizar centenas de milhares de containers no ambiente de produção.

8.4.2 Kubernetes e Microsserviços

Diante do que você já viu até aqui, é possível observar que a maioria dos recursos oferecidos pelo **Kubernetes** também é essencial para o contexto dos microsserviços. Por isso, o uso do Kubernetes **gerenciando um container Docker** tornou-se uma prática extremamente poderosa para a implantação de microsserviços, especialmente para os microsserviços de grande porte.

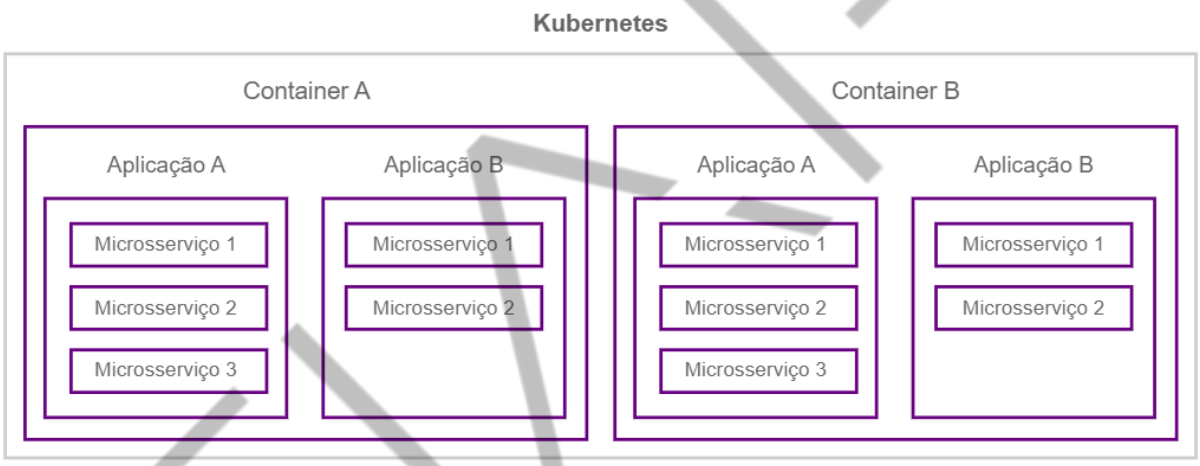


Figura 8.12 – Relação do Kubernetes e Microsserviços
Fonte: Elaborado pela autora, adaptado por FIAP (2019)

8.4.3 Kubernetes x OpenShift x OpenStack

A realidade é que Kubernetes, OpenShift e OpenStack não podem ser comparados por serem tecnologias independentes e diferentes em sua essência, e usados separadamente uns dos outros.

Então, veja um resumo das características de cada uma das tecnologias em questão:

- **Kubernetes:** desempenha o papel de sistema operacional no nível de containers com a responsabilidade de executar tarefas de orquestração de containers.

- **OpenShift:** incorporou o Kubernetes para oferecer funcionalidades além de gerenciamento de containers; e foi otimizado para rodar em infraestrutura de *cloud* pública ou privada.
- **OpenStack:** responsável por gerenciar grandes quantidades de recursos computacionais que envolvem processamento, armazenamento e rede; em suma, oferece uma plataforma para gerenciar a infraestrutura que é responsável por executar seus servidores.

CONCLUSÃO

Chegamos ao final deste módulo e você está capacitado para desenvolver e gerenciar containers Docker.

REFERÊNCIAS

AMAZON. **AWS documentation.** [s.d.]. Disponível em: <https://docs.aws.amazon.com>. Acesso em: 8 fev. 2021.

DOCKER. **Docker website.** [s.d.]. Disponível em: <https://www.docker.com/>. Acesso em: 8 fev. 2021.

KUBERNETS. [s.d.]. Disponível em: <https://kubernetes.io/>. Acesso em: 8 fev. 2021.

OPENSTACK. **Exploring opportunities: containers and OpenStack. technical report**. 2015. Disponível em: <https://object-storage-ca-ymq-1.vexxhost.net/swift/v1/6e4619c416ff4bd19e1c087f27a43eea/www-assets-prod/pdf-downloads/Containers-and-OpenStack.pdf>. Acesso em: 8 fev. 2021.

OPENSTACK. **Kolla documentation**. [s.d.]. Disponível em: <https://docs.openstack.org/kolla/4.0.0/>. Acesso em: 8 fev. 2021.

OPENSTACK. **Magnum user guide.** 2019. Disponível em: <https://docs.openstack.org/magnum/latest/user/>. Acesso em: 8 fev. 2021.

LISTA DE ABREVIATURAS

API	Application Programming Interface
BPM	Business Process Management
EAI	Enterprise Application Integration
HTTP	Hiper Text Transfer Protocol
IDE	Integrated Development Environment
Java EE	Java Enterprise Edition
RPC	Remote Procedure Call
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
TI	Tecnologia da Informação
WSDL	Web Service Description Language
XML	Extensible Markup Language
URL	Uniform Resource Locators
URI	Uniform Resource Identifiers
DDD	Domain-Driven Design
P&D	Pesquisa e Desenvolvimento
VM	Virtual Machine