PROVAS DE TI
TUDO PARA VOCÊ PASSAR

Ferramentas DevOps

Prof. Rodrigo Macedo

Escopo do Curso

- Docker
- Arquitetura Docker
- Comandos Docker
- Dockerfile
- Kubernetes
- Questões de concursos

Docker - Conceitos Iniciais

- O Docker é uma plataforma de software que permite a criação, o teste e a implantação de aplicações rapidamente. O Docker cria pacotes de software em unidades padronizadas chamadas de contêineres que têm tudo o que o software precisa para ser executado, inclusive bibliotecas, ferramentas de sistema, código e runtime.
- Ao usar o Docker, é possível implantar e escalar rapidamente aplicações em qualquer ambiente e ter a certeza de que o seu código será executado.
- Docker é um conjunto de produtos de plataforma como serviço (PaaS) que usam virtualização de nível de sistema operacional para entregar software em pacotes chamados **contêineres**

Docker - Conceitos Iniciais

- Um contêiner é um formato de empacotamento que empacota todo o código e dependências de um aplicativo em um formato padrão que permite sua execução rápida e confiável em ambientes de computação.
- Um contêiner Docker é um popular contêiner leve, autônomo e executável que inclui tudo o que é necessário para executar um aplicativo, incluindo bibliotecas, ferramentas de sistema, código e tempo de execução.
- Docker também é uma plataforma de software que permite aos desenvolvedores criar, testar e implantar aplicativos em contêineres rapidamente.

Docker - Conceitos Iniciais

- Com o objetivo de permitir a execução independente dos processos, o Docker se utiliza do kernel (núcleo) do Linux e seus recursos – a exemplo do namespaces e do Cgroups – para isolar essas operações.
- Nessa perspectiva, o propósito dos containers é justamente este: **criar a independência para permitir a execução de múltiplas aplicações e processos de forma separada**. A partir daí, é possível ter um melhor aproveitamento da infraestrutura, assim como garantir a mesma segurança que se teria em sistemas isolados.
- É importante acrescentar que o Docker, tal como outros recursos de container, oferece um modelo de implantação baseado em imagem. Essa funcionalidade facilita o compartilhamento de serviços, aplicações e todas as suas dependências, como já ressaltamos acima.

Docker - Conceitos Iniciais

- Um Container é a forma de **empacotar** sua aplicação e suas dependências (bibliotecas) de forma padronizada.
- Podemos dizer que as palavras chaves para o Docker são: construir, entregar e rodar em qualquer ambiente (develop, ship and run anywhere).

docker

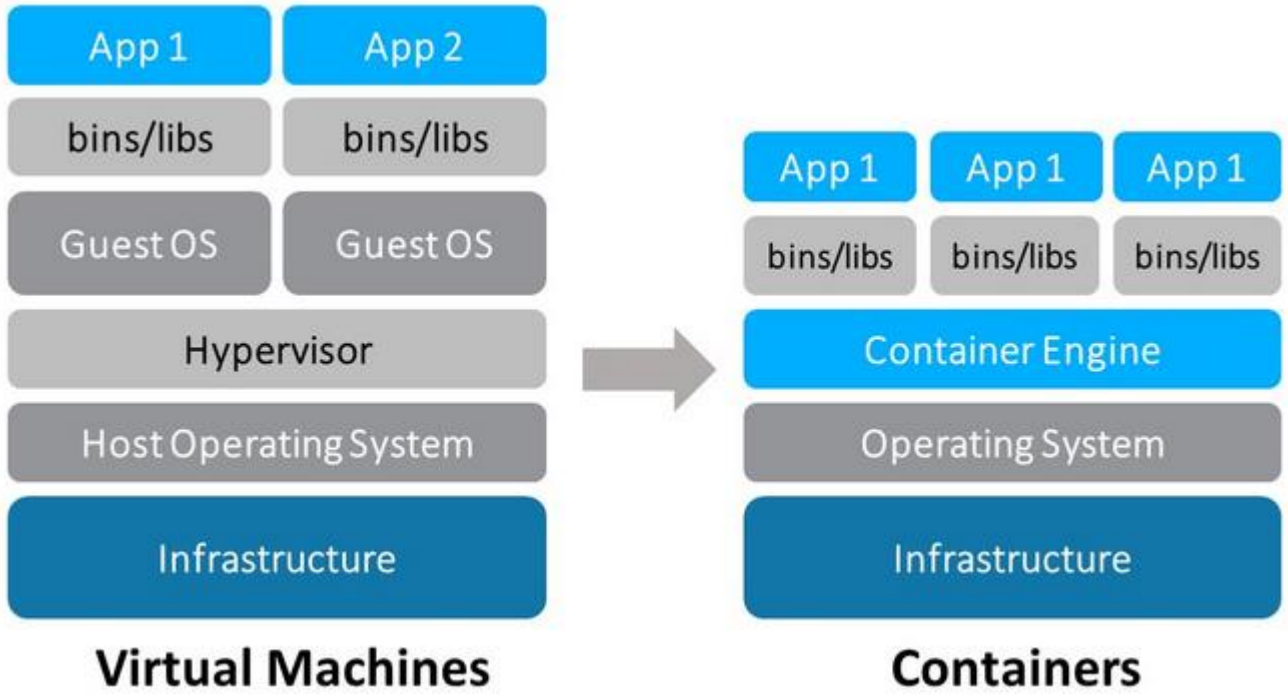
Docker x VM

Uma VM (máquina virtual) obriga a equipe a gerenciar todo o ambiente relacionado ao servidor. Isso abrange:

- a configuração de sistema operacional;
- atualizações do sistema operacional;
- patches de segurança;
- patches da aplicação;
- backup do sistema operacional;
- backup da aplicação.
- A diferença é que um contêiner no docker **compartilha** o Kernel com o sistema operacional do host. Isso faz com que o desempenho aumente e o consumo de memória do contêiner diminua.

Docker x VM

- Podemos notar que um contêiner Docker não precisa de um Hypervisor nem de um Sistema Operacional(SO) completo para funcionar; ele compartilha o Kernel a partir do SO do host.
- Para poder ser executado, um contêiner possui associado a ele um sistema de arquivos completo e isolado que contêm as dependências e bibliotecas necessárias.



Este sistema de arquivos somente-leitura são conhecidos como **Imagens**; e, a partir das imagens é que os contêiners são criados.

Docker - Benefícios

- Ao usar o Docker, é possível enviar o código com mais rapidez, padronizar as operações de aplicativo, mover o código com facilidade e economizar, melhorando a utilização de recursos. Com o Docker, você tem um único objeto que pode ser executado com segurança em qualquer lugar. A sintaxe simples e direta do Docker possibilita o controle total. A ampla adoção significa que o Docker disponibiliza um ecossistema reforçado de ferramentas e aplicações prontas para uso.

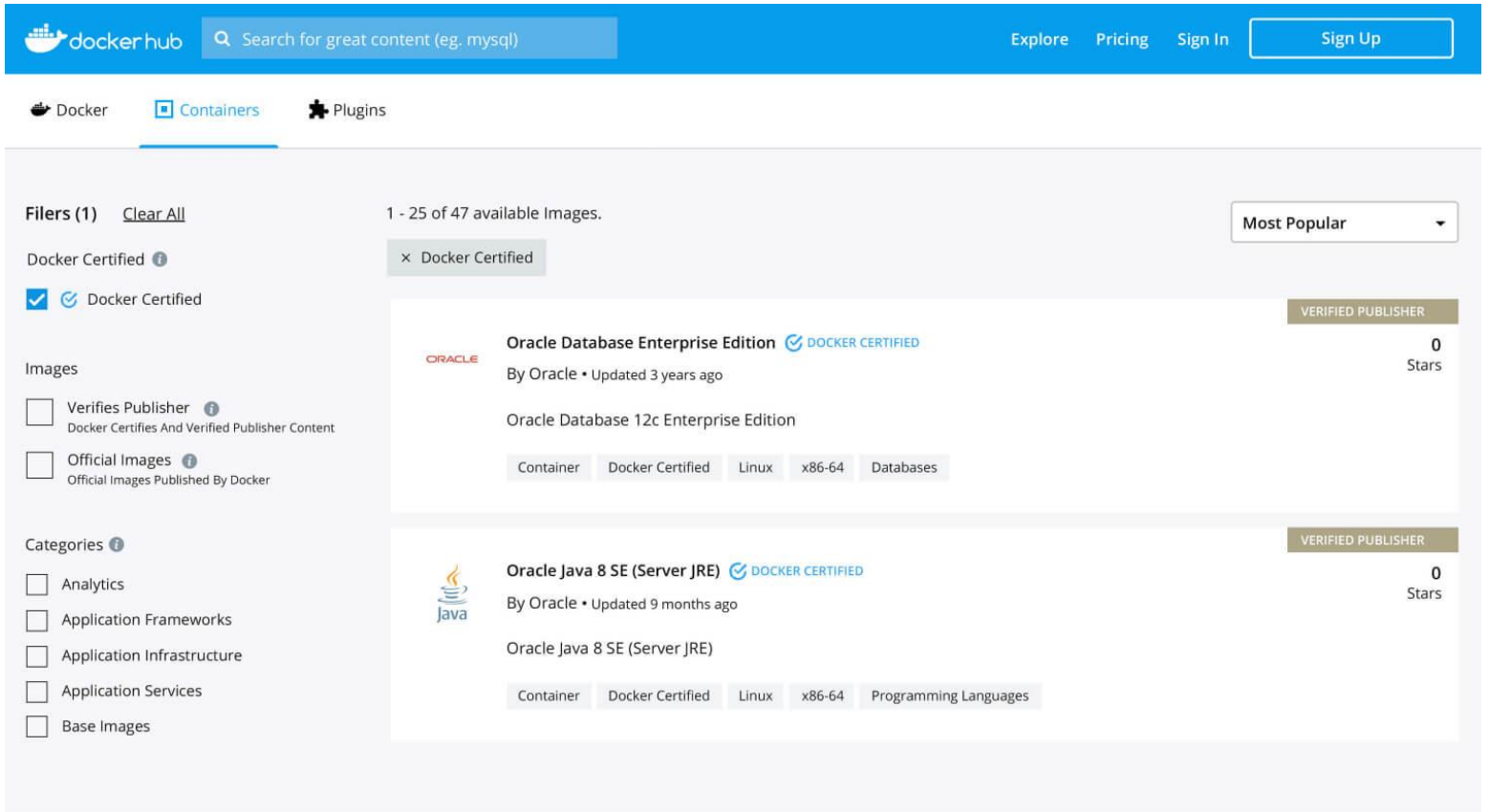
1. **Disponibilize mais software, com maior rapidez:** Em média, os usuários do Docker disponibilizam software com uma frequência 7 vezes maior do que os usuários de outras tecnologias. O Docker permite enviar serviços isolados sempre que necessário.

Docker - Benefícios

1. **Padronize operações:** Pequenas aplicações em contêineres facilitam a implantação, a identificação de problemas e o roll-back para correção.
2. **Mova com facilidade:** Aplicações com base em Docker podem ser transferidas de máquinas locais de desenvolvimento para implantações de produção na nuvem.
3. **Economize dinheiro:** Os contêineres do Docker facilitam a execução de mais códigos em cada servidor, melhorando a utilização e economizando seu dinheiro.
4. **Ambientes similares:** a possibilidade de transformar uma aplicação em imagem docker permite que ela seja alocada como container em ambientes diferentes, fazendo com que ela possa ser utilizada tanto no computador do desenvolvedor quanto no servidor da produção, por exemplo.

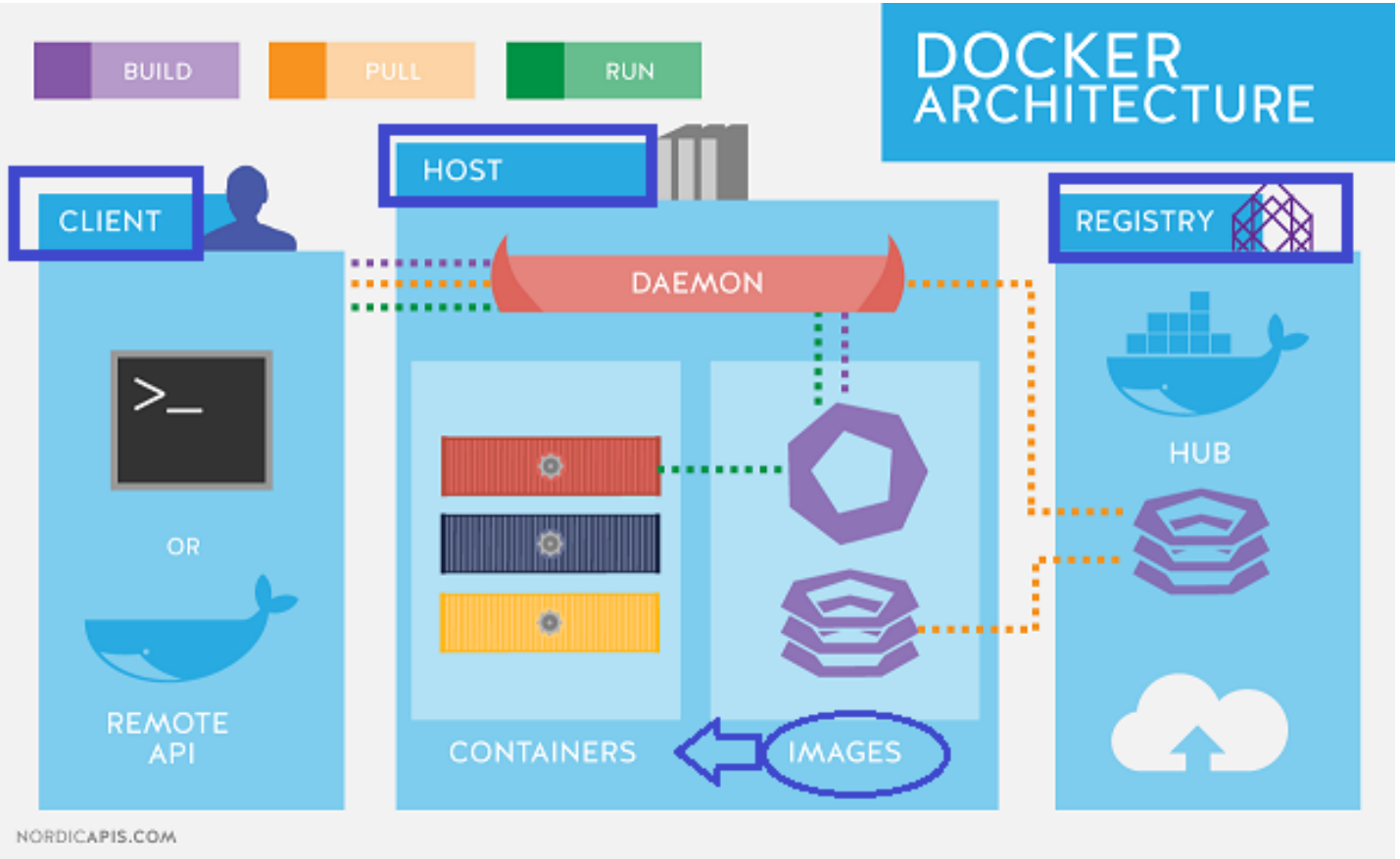
Docker - Benefícios

1. **Acesso a Comunidade:** outra vantagem do container docker é que ele torna possível conseguir modelos de infraestrutura prontos para integrações com alto grau de complexidade através do acesso ao repositórios de imagens docker. É possível usar as imagens do repositório e configurar os parâmetros para que ele se adeque ao ambiente.



Docker - Arquitetura

- Na figura abaixo temos uma visão resumida da arquitetura Docker:



1. **Docker Host ou Server** para Mac, Linux e Windows – versões que permitem instalar e executar containers nos sistemas operacionais de forma isolada.

Docker - Arquitetura

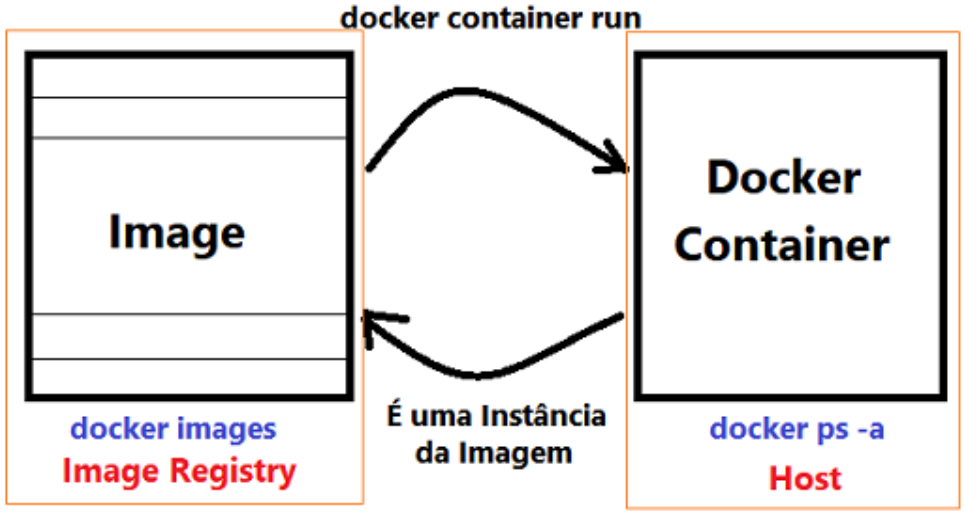
1. **Docker Daemon** – Software que roda na máquina onde o Docker está instalado. Usuário não interage diretamente com o daemon.
2. **Docker Client** – CLI ou REST API que aceita comandos do usuário e repassa estes comandos ao Docker daemon.
3. **Docker Image** – É um template. Uma imagem contém todos os dados e metadados necessários para executar containers a partir de uma imagem.
4. **Docker Container** – Detém tudo que é necessário para uma aplicação ser executada. Cada container é criado a partir de uma imagem. Cada container é uma aplicação isolada independente.

Docker - Arquitetura

1. **Docker Engine** – Usado para criar imagens e containers.
2. **Docker Registry** – Uma coleção de imagens hospedadas e rotuladas que juntas permitem a criação do sistema de arquivos de um container. Um registro pode ser público ou privado.
3. **Docker Hub** – Este é um registro usado para hospedar e baixar diversas imagens. Pode ser visto como uma plataforma SAAS de compartilhamento e gerenciamento de imagens.
4. **Dockerfile** – Um arquivo texto contendo uma sintax simples para criação de novas imagens.
5. **Docker Compose** – Usado para definir aplicações usando diversos containers.
6. **Docker Swarm** – É uma ferramenta que permite o agrupamento (clustering) de Containers Docker.

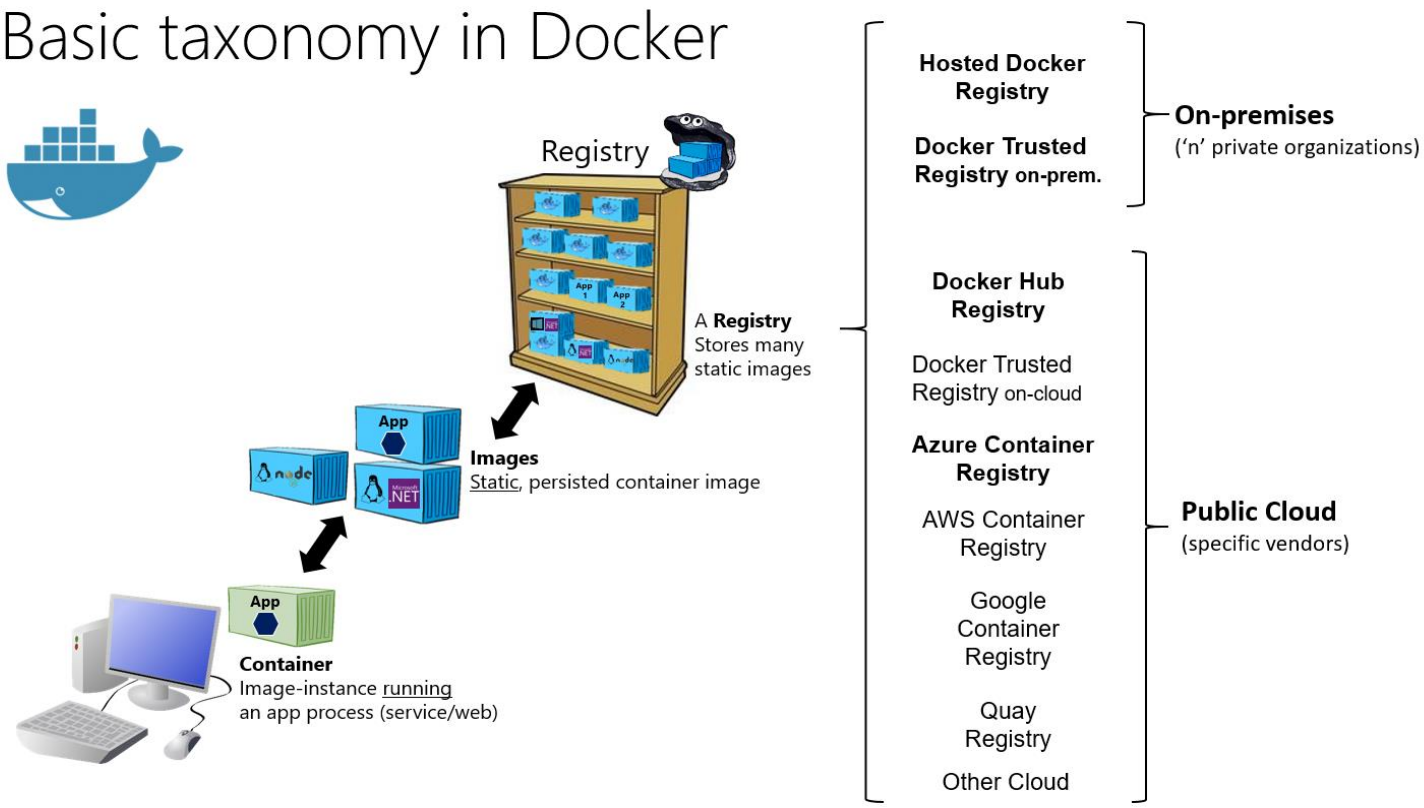
Docker - Arquitetura

- Imagens são arquivos que contém todo o conteúdo e estrutura de sistemas operacionais. Elas são a base de construção de containers no Docker.
- O servidor do Docker utiliza o AuFS, um file system em camadas que permite a separação do espaço do sistema em uma parte read-only e outra parte read/write.
- Ao executar um container de um cliente, o Docker utiliza a imagem de sistema operacional escolhida como base, e numa camada superior executa as configurações escolhidas pelo cliente.



Docker - Arquitetura

- O Registro do Docker é uma espécie de repositório para imagens. Com esse registro, um usuário pode construir, salvar e distribuir imagens com outros.
- O site do Docker fornece um sistema de registro chamado Docker Hub, que funciona como um git, permitindo ao usuário que construa localmente suas imagens em sua máquina, e então realize operações de commit e push.



Docker - Comandos

1. **docker info:** esse comando para verificarmos as informações do nosso Docker Host.

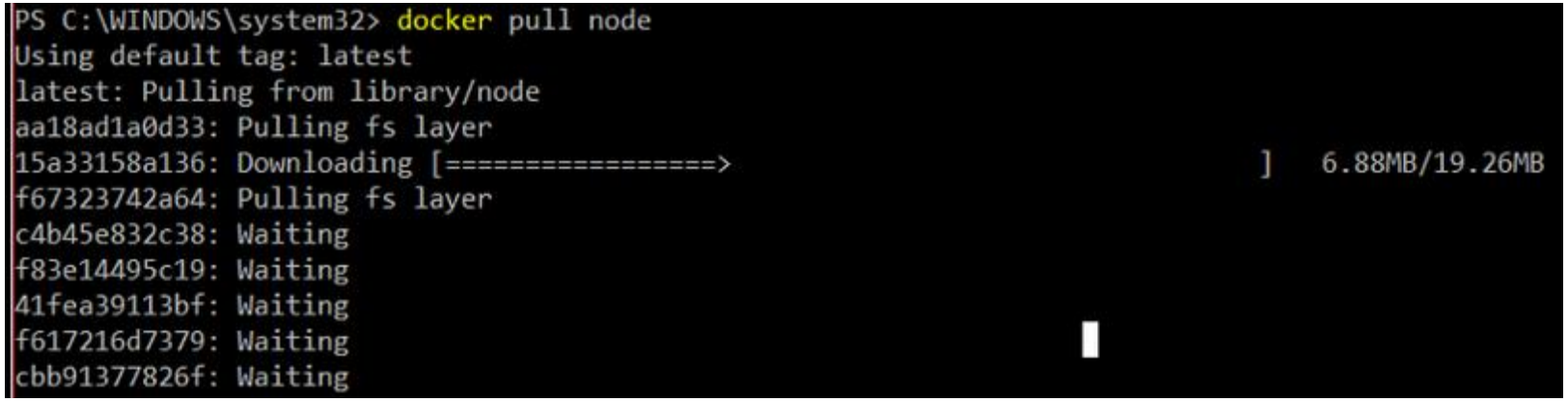
```
 Backing Filesystem: extfs
 Supports d_type: true
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host ipvlan macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file logentries splunk syslog
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 06b9cb35161009dcb7123345749fef02f7cea8e0
runc version: 3f2f8b84a77f73d38244dd690525642a72156c64
init version: 949e6fa
```

2. **docker search (parâmetro):** usado para procurar uma imagem, nós podemos utilizar o comando a baixo com o parâmetro definido.

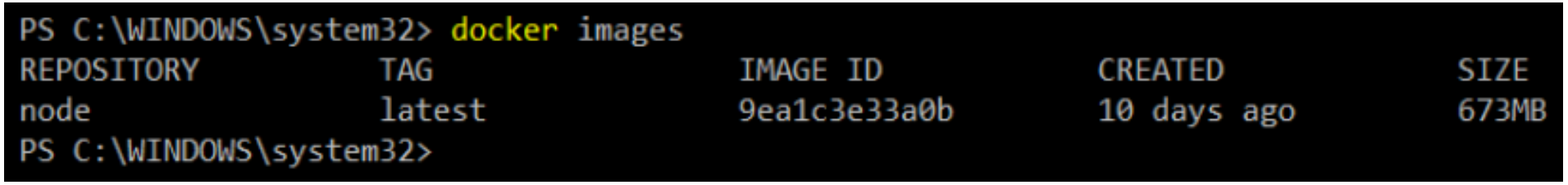
```
PS C:\WINDOWS\system32> docker search node
NAME                                DESCRIPTION                                     STARS               OFFICIAL
node                                Node.js is a JavaScript-based platform for...   4570                [OK]
mhart/alpine-node                   Minimal Node.js built on Alpine Linux           313
mongo-express                       Web-based MongoDB admin interface, written...   185                 [OK]
iojs                                io.js is an npm compatible platform origin...   121                 [OK]
selenium/node-chrome                                                                120
nodered/node-red-docker             Node-RED Docker images.                         80
prom/node-exporter                                                                  42
digitallyseamless/nodejs-bower-grunt    Node.js w/ Bower & Grunt Dockerfile for t...   41
readytalk/nodejs                    Node.js based off the official Debian Whee...   35
kkarczmarczyk/node-yarn             Node docker image with yarn package manage...   32
bitnami/node                        Bitnami Node.js Docker Image                    25
iron/node                           Tiny Node image                                 25
circleci/node                       Node.js is a JavaScript-based platform for...   16
calico/node                                                                         14
```

Docker - Comandos

1. **docker pull:** Quando encontrarmos a imagem que precisamos para a nossa aplicação, nós precisamos baixar ela para o nosso host.



2. **docker images:** lista as imagens que estão presentes no host.



Docker - Comandos

1. **docker run:** Cria e inicia um contêiner. Para que nós possamos criar um contêiner, nós precisamos de uma imagem. Caso você não tenha essa imagem no seu host ainda ele irá até o repositório central e irá baixar ela para o seu host.

```
PS C:\WINDOWS\system32> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
5b0f327be733: Pull complete
Digest: sha256:b2ba691d8aac9e5ac3644c0788e3d3823f9e97f757f01d2ddc6eb5458df9d801
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

2. **docker ps:** verifica o status do container. Quando nos executamos o comando docker run hello-world ele realizou todos os processos e morreu.

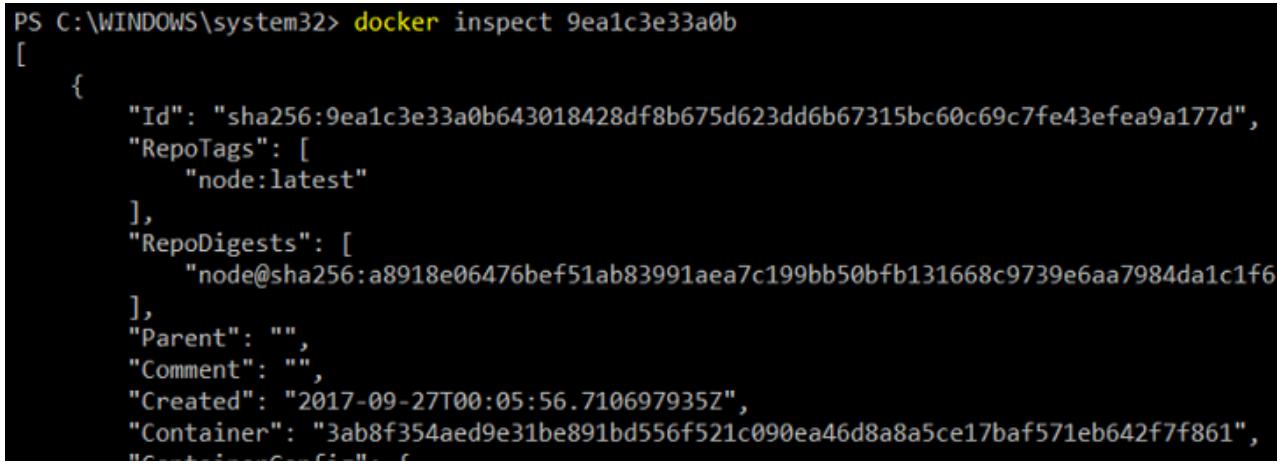
```
PS C:\WINDOWS\system32> docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
PS C:\WINDOWS\system32>
```

3. **docker ps -a:**

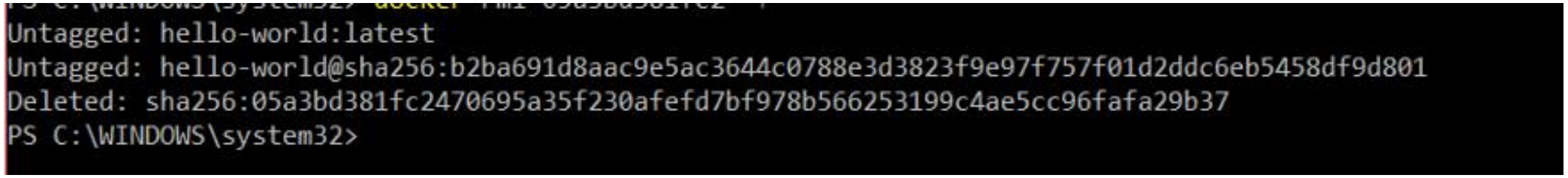
```
PS C:\WINDOWS\system32> docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
d6a7ea8ceeab        hello-world         "/hello"            4 minutes ago       Exited (0) 4 minutes ago                        compassionate_
lalande
```

Docker - Comandos

1. **docker inspect (id da imagem ou container):** Informa mais detalhes sobre a sua imagem ou o seu contêiner.



2. **docker rmi:** Remove uma **imagem** do host. O comando **docker rm** remove um **container**.



Docker - Comandos

1. **docker exec:** Execute um comando em um contêiner em execução.Ex: docker exec 8b54c76e81b7 mkdir /temp/
2. **docker attach (id_container ou nome_container):** Permite navegar em um container. A execução desse comando irá nos dar acesso de root no nosso contêiner. Para sair do contêiner execute o comando exit.

```
PS C:\WINDOWS\system32> docker attach 8b54c76e81b7
root@8b54c76e81b7:/#
```

3. **docker start (id do container):** Inicia um ou mais container parado. O comando docker stop para o container.

```
PS C:\WINDOWS\system32> docker start 8b54c76e81b7
8b54c76e81b7
PS C:\WINDOWS\system32>
```

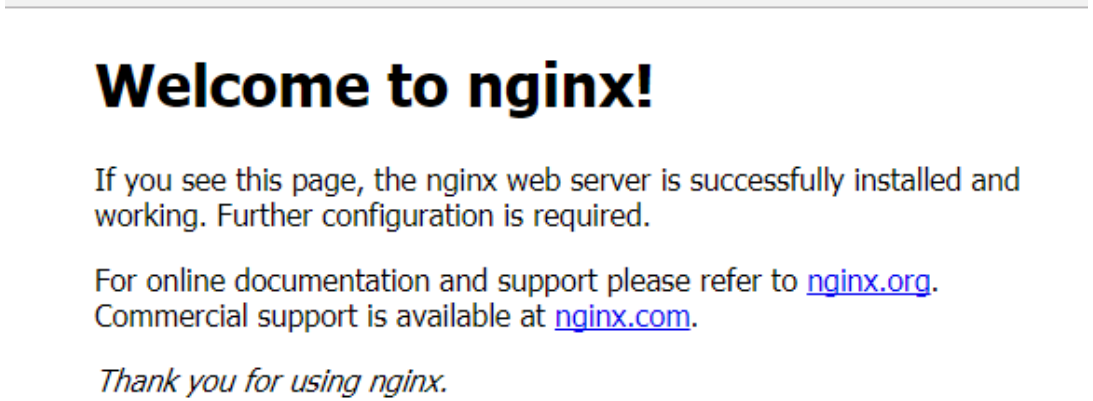
Docker - Comandos

1. **build:** constrói uma imagem a partir de um arquivo Dockerfile.
2. **diff:** inspecione as alterações no sistema de arquivos de um contêiner.
3. **history:** mostra o histórico de uma imagem.
4. **log:** buscar os registros de um contêiner.
5. **network:** gerenciar redes Docker.
6. **port:** Listar mapeamentos de portas ou um mapeamento específico para o CONTAINER.
7. **volume:** gerenciar volumes Docker. Comando para criar um volume: **docker volume create [name_volume].**
8. **expose:** A instrução EXPOSE informa ao Docker que o contêiner escuta nas portas de rede especificadas no tempo de execução. Você pode especificar se a porta escuta em TCP ou UDP e o padrão é TCP se o protocolo não for especificado.

Docker - Comandos

Para que possamos testar o mapeamento de portas iremos baixar a imagem do ngnix e passarmos para ele a porta 80.

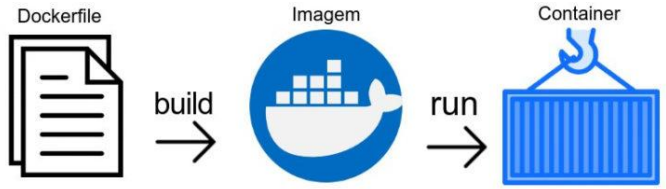
- **docker run -it -p 8080:80 nginx** (inicia o container em modo interativo, por isso o parâmetro -it).



- Um problema que temos ao executar o comando acima é que o console fica travado e não é possível executar novos comandos.
- Para resolver isso, pode executar o comando em background.
- **docker run -it -d -p 8080:80 nginx**

Dockerfile

- O Dockerfile nada mais é do que um meio que utilizamos para criar nossas próprias imagens.
- Em outras palavras, ele serve como a receita para construir um container, permitindo definir um ambiente personalizado e próprio para meu projeto pessoal ou empresarial.
- É com esse arquivo que podemos gerar o build e criar o container a partir dele.
- Para criar um Dockerfile é simples, basta criar um arquivo com o nome Dockerfile.
- Para gerar a imagem a partir do Dockerfile, executamos o comando no mesmo local que o arquivo se encontra: **docker build -t nome_da_imagem**



Dockerfile - Estrutura

1. **FROM:** É a instrução mais utilizada do arquivo, pois ela é obrigatória. Com essa instrução, pode-se definir qual será o ponto de partida da imagem que criaremos com o nosso Dockerfile, ou seja, se eu quiser utilizar a imagem do Java para produzir meu container, basta que eu especifique para utilizar a imagem do openjdk como base. Ficaria algo como: **FROM openjdk.** Pode ser especificado também a versão, exemplo: **FROM ubuntu:18.04.**
2. **RUN:** Ela pode ser executada uma ou mais vezes e, com ela, é possível definir quais serão os comandos executados na etapa de criação de camadas da imagem.

Quando o comando **docker build .** for executado, além de baixar a imagem do Ubuntu 18.04 para colocar na imagem, o processo de criação também executará os comandos para atualizar os repositórios do Ubuntu através do apt-get update, e para instalar o Java 8 utilizará o apt-get install openjdk-8-jdk-y

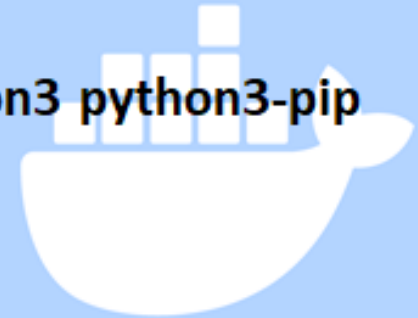
```
FROM ubuntu:18.04

RUN apt-get update

RUN apt-get install openjdk-8-jdk -y
```

Dockerfile - Estrutura

1. **ADD:** O papel do ADD é fazer a cópia de um arquivo, diretório ou até mesmo fazer o download de uma URL de nossa máquina host e colocar dentro da imagem.
2. **Copy:** permite apenas a passagem de arquivos ou diretórios, diferente do ADD, que permite downloads.
3. **EXPOSE:** divulga uma porta (TCP ou UDP) para os hosts externos à rede Docker.
4. **VOLUME:** Essa instrução cria uma pasta em nosso container que será compartilhada entre o container e o host.
5. **WORKDIR:** Essa instrução tem o propósito de definir o nosso ambiente de trabalho. Com ela, definimos onde as instruções CMD, RUN, ENTRYPOINT, ADD e COPY executarão suas tarefas, além de definir o diretório padrão que será aberto ao executarmos o container.

Dockerfile - Exemplo

```
FROM  ubuntu:16.04

MAINTAINER  Abhishek

RUN  apt-get update
RUN  apt-get install -y python3 python3-pip
COPY .  /app

WORKDIR  /app

EXPOSE  5000

ENTRYPOINT echo “Hello World !”
```



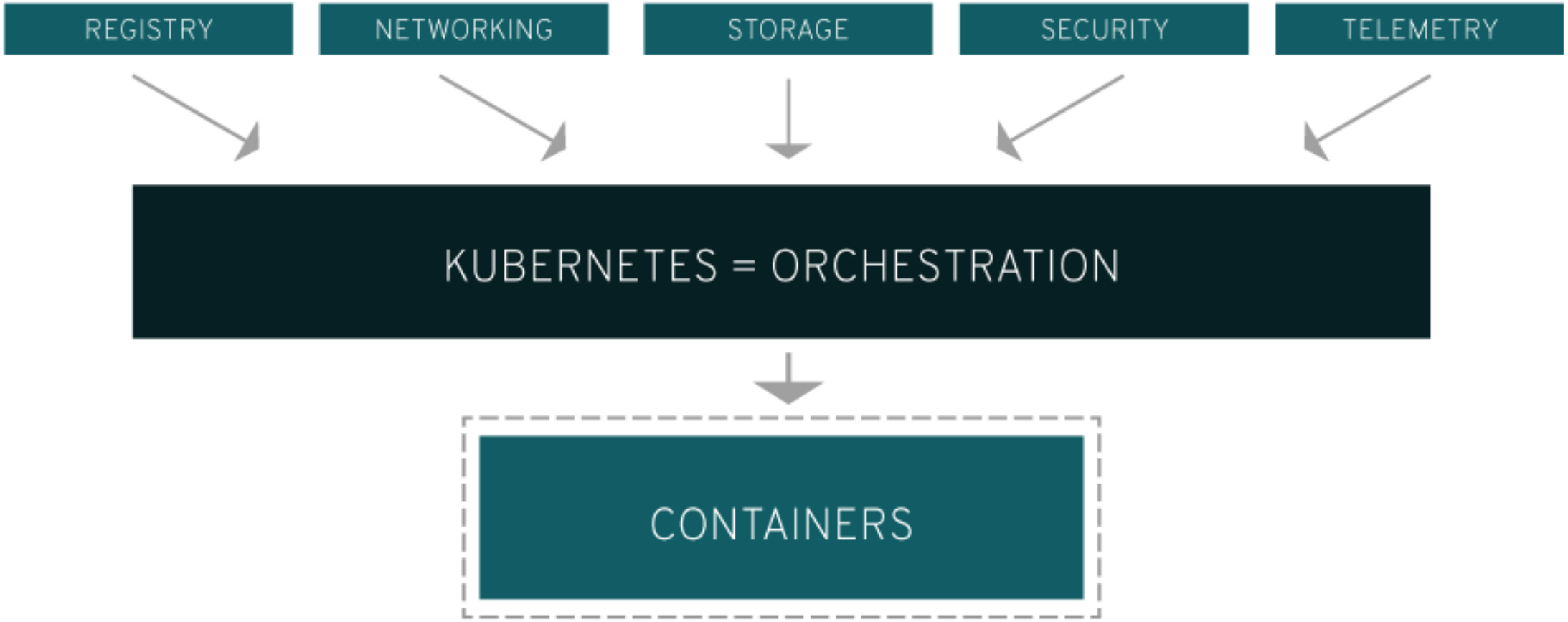
Kubernetes

- Kubernetes é um sistema de código aberto para automatizar a implantação, escalonamento e gerenciamento de aplicativos em contêineres.
- Ele agrupa os containers que constituem um aplicativo em unidades lógicas para fácil gerenciamento e descoberta.
- Atualmente mantido pela Cloud Native Computing Foundation, o Kubernetes gerencia os clusters que contêm os hosts que executam as aplicações Linux.
- Esses clusters podem incluir hosts em nuvem, por isso, o Kubernetes é a plataforma ideal para hospedar aplicações **cloud-native** que exigem escalabilidade rápida, como a transmissão de dados em tempo real por meio do Apache Kafka.
- Na prática, ele atua como um orquestrador (conjunto) de containers, similar ao que o Docker Swarm faria com o Docker.

Kubernetes

- Originalmente, o Kubernetes foi criado e desenvolvido por engenheiros do Google. O Google foi um dos pioneiros no desenvolvimento da tecnologia de containers Linux. Além disso, a empresa já revelou publicamente que tudo no Google é executado em containers (inclusive, essa é a tecnologia por trás dos serviços em cloud da empresa).
- O Kubernetes permite que o programador garanta exatamente onde e quando as aplicações baseada em contêineres serão executadas, bem como auxilia na busca dos recursos e ferramentas corretas para trabalhar.
- Ele ajuda a automatizar várias tarefas operacionais da gestão de contêineres, com comandos integrados para implantação de aplicativos.

Kubernetes


REGISTRY
NETWORKING
STORAGE
SECURITY
TELEMETRY
KUBERNETES = ORCHESTRATION
CONTAINERS

Kubernetes - Benefícios

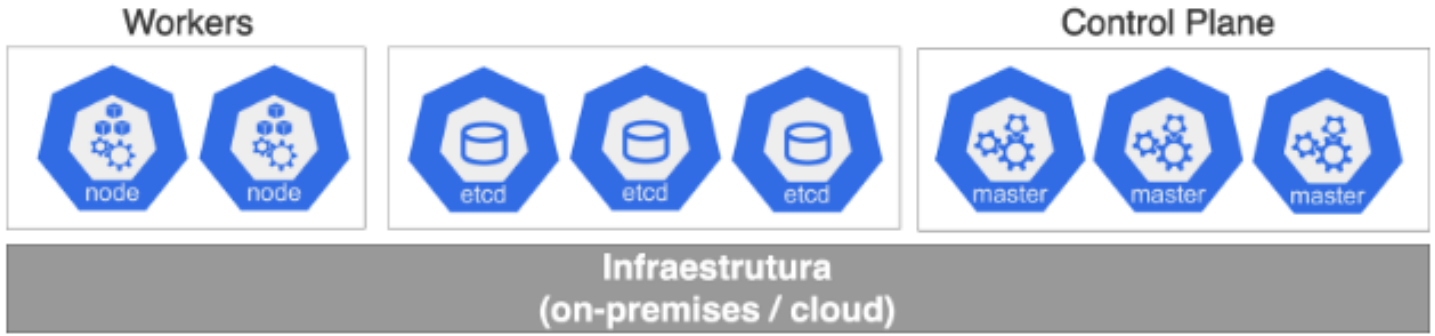
- A principal vantagem que as empresas garantem ao usar o Kubernetes, especialmente se estiverem otimizando o desenvolvimento de aplicações para a cloud, é que elas terão uma plataforma para programar e executar containers em clusters de máquinas físicas ou virtuais.
- Em termos mais abrangentes, com o Kubernetes, é mais fácil implementar e confiar totalmente em uma infraestrutura baseada em containers para os ambientes de produção. Como o propósito do Kubernetes é automatizar completamente as tarefas operacionais, ele permite que os containers realizem muitas das tarefas possibilitadas por outros sistemas de gerenciamento ou plataformas de aplicações.

1. Orquestrar containers em vários hosts.
2. Aproveitar melhor o hardware para maximizar os recursos necessários na execução das aplicações corporativas.

Kubernetes - Benefícios

1. Controlar e automatizar as implantações e atualizações de aplicações.
2. Montar e adicionar armazenamento para executar aplicações com monitoração de estado.
3. Escalar rapidamente as aplicações em containers e recursos relacionados.
4. Gerenciar serviços de forma declarativa, garantindo que as aplicações sejam executadas sempre da mesma maneira como foram implantadas.
5. Verificar a integridade e autorrecuperação das aplicações com posicionamento, reinício, replicação e escalonamento automáticos.
6. O Kubernetes oferece uma plataforma completa para aplicações conhecidas como cloud-native;
7. A API do Kubernetes é a porta de entrada de um cluster, sendo utilizada tanto pela linha de comando quando pela interface web.

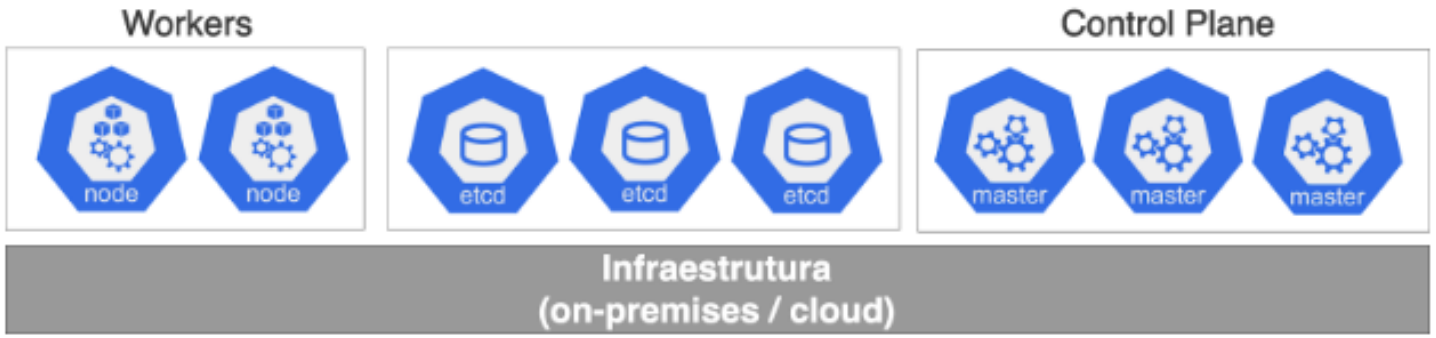
Kubernetes - Arquitetura



- O Kubernetes é composto por uma série de componentes, cada um com um propósito diferente. Para garantir que exista uma separação de responsabilidades e que o sistema seja resiliente, o Kubernetes utiliza um cluster de máquinas para ser executado.
- As máquinas de um cluster são separadas em três tipos:

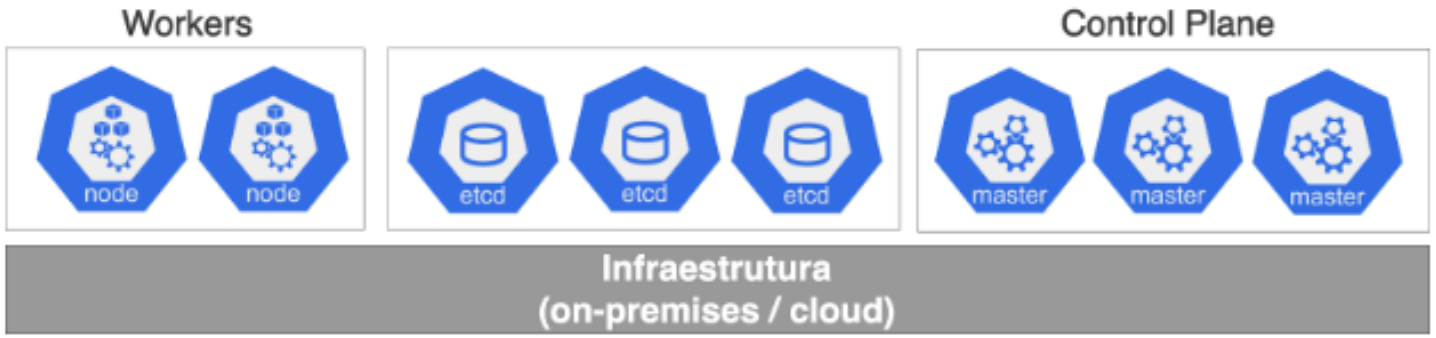
1. **Node:** O primeiro tipo é chamado de Node. O papel de um Node é executar os contêineres que encapsulam as aplicações sendo gerenciadas pelo K8s. Quando você faz o deploy de uma aplicação em um cluster K8s, essa aplicação vai ser executada em um dos Nodes do cluster. O conjunto de Nodes forma o que chamamos de **Workers**.

Kubernetes - Arquitetura



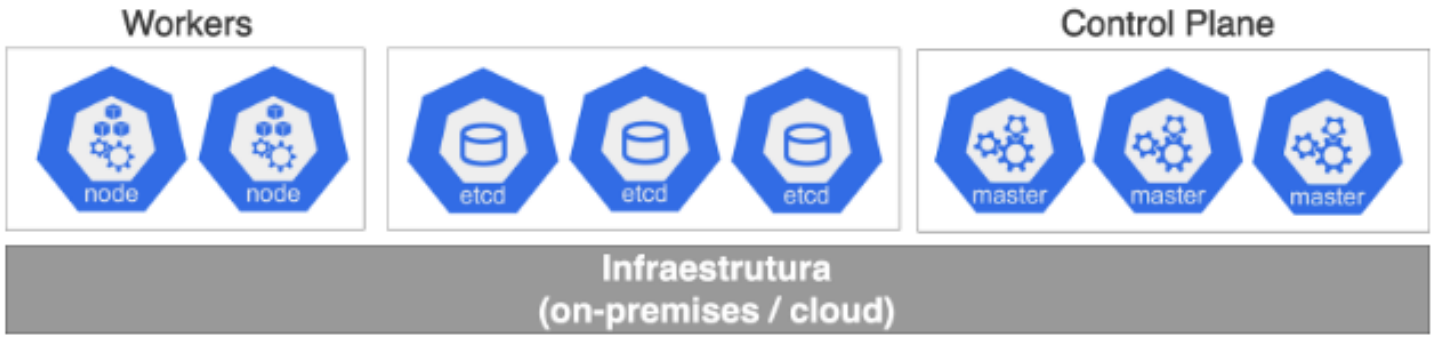
1. **etcd:** O segundo tipo de nó é o etcd. O etcd é, na verdade, o nome da base de dados distribuída que é utilizada para armazenar tudo o que está acontecendo dentro do cluster, incluindo o estado da aplicação. Em ambientes de produção, um bom gerenciamento desses nós é essencial para garantir que o cluster esteja sempre disponível.
2. **Master:** o último tipo de nó é o que chamamos de Master. É nesse tipo de nó que os principais componentes do Kubernetes são executados, como o Scheduler, o qual tem a responsabilidade de controlar a alocação de recursos no cluster. O conjunto de nós Master forma o que pode ser considerado o cérebro de um cluster Kubernetes: o **Control Plane.**

Kubernetes - Arquitetura



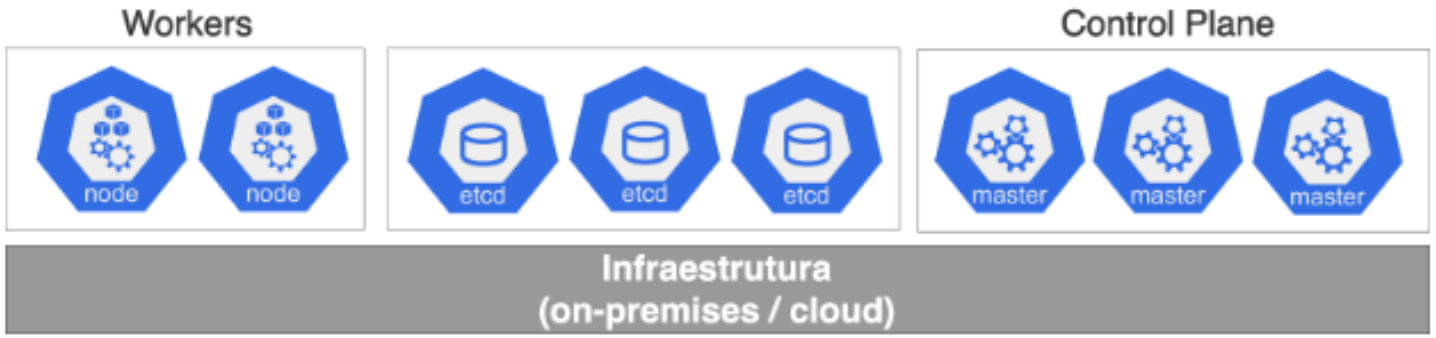
1. **Control Plane:** O Control Plane do Kubernetes pode ser considerado o cérebro de um cluster. Ele é responsável por gerenciar os principais componentes do sistema e garantir que tudo está funcionando de acordo com o estado desejado da aplicação. Para facilitar a representação desse estado, o K8s trabalha com uma abstração chamada de Object. Um Object representa parte do estado da aplicação e quando o seu estado atual não é o estado desejado, mudanças são aplicadas para que os dois estados se igualem novamente. Existem diversos tipos de Objects em um ambiente Kubernetes, mas alguns deles são essenciais para entendermos como um cluster funciona.

Kubernetes - Arquitetura



1. **Pod:** Pods são a unidade básica de um um cluster K8s. Elas encapsulam um ou mais contêineres de uma aplicação e representam um processo dentro do cluster. Quando fazemos o deploy de uma aplicação no K8s, estamos criando uma ou mais Pods. Elas são **criadas** e **destruídas** de acordo com as **necessidades** do cluster. Os pods separam a rede e o armazenamento do container subjacente. Isso facilita a movimentação dos containers pelo cluster.
2. **Service:** desacopla as definições de trabalho dos pods. Os proxies de serviço do Kubernetes automaticamente levam as solicitações de serviço para o pod correto, independentemente do local do pod no cluster ou se foi substituído.

Kubernetes - Arquitetura



1. **Kubelet:** um serviço executado nos nós que lê os manifestos do container e garante que os containers definidos foram iniciados e estão em execução.
2. **kubectl:** a ferramenta de configuração da linha de comando do Kubernetes.
3. **Controlador de replicações:** controla quantas cópias idênticas de um pod devem ser executadas em um determinado local do cluster.

Q1) [FUNDATEC CIGA-SC 2020] A Figura 3 abaixo representa o conteúdo de um arquivo Dockerfile:

```
FROM node:12
WORKDIR /app
COPY * ./
RUN npm install
EXPOSE 90
CMD [ "node", "server.js" ]
```

Figura 3

Assinale a alternativa INCORRETA sobre o arquivo.

a) É necessário executar o comando “docker build”, com os parâmetros apropriados, para criar uma imagem Docker a partir das definições do arquivo Dockerfile.

b) Com a imagem pronta, após a execução do comando “docker run”, com os parâmetros apropriados, é esperado que uma aplicação em NodeJS esteja sendo executada.

c) O comando “npm install” instala a aplicação que é definida na instrução “FROM”.

d) O código-fonte da aplicação que será executada no container deverá estar inicialmente no sistema de arquivos da máquina hospedeira (Container Host).

e) A instrução “WORKDIR” define um diretório para as instruções “RUN”, “CMD” e “COPY”.

Q1) [FUNDATEC CIGA-SC 2020 GAB C] A Figura 3 abaixo representa o conteúdo de um arquivo Dockerfile:

```
FROM node:12
WORKDIR /app
COPY * ./
RUN npm install
EXPOSE 90
CMD [ "node", "server.js" ]
```

Figura 3

Assinale a alternativa INCORRETA sobre o arquivo.

a) É necessário executar o comando “docker build”, com os parâmetros apropriados, para criar uma imagem Docker a partir das definições do arquivo Dockerfile.

b) Com a imagem pronta, após a execução do comando “docker run”, com os parâmetros apropriados, é esperado que uma aplicação em NodeJS esteja sendo executada.

c) O comando “npm install” instala a aplicação que é definida na instrução “FROM”.

d) O código-fonte da aplicação que será executada no container deverá estar inicialmente no sistema de arquivos da máquina hospedeira (Container Host).

e) A instrução “WORKDIR” define um diretório para as instruções “RUN”, “CMD” e “COPY”.

Q2) [UFMT UFMT 2021] Sobre Contêiner no Sistema Operacional Linux Ubuntu Server 18.04, analise o comando a seguir.

docker run --name web -p 80:80 -p 443:443 -p 65533:22 -ti ubuntu bash

Marque V para as afirmativas verdadeiras e F para as falsas.

( ) A opção --name indica o nome que o contêiner terá, nesse caso, web. ( ) Ao executar esse comando, o docker criará um contêiner baseado em uma imagem do ubuntu. ( ) A opção -p indica que o contêiner deve debugar essas range de portas apresentadas na frente do parâmetro.

Assinale a sequência correta.

a) F, F, V

b) V, F, F

c) V, V, F

d) F, V, V

Q2) [UFMT UFMT 2021] Sobre Contêiner no Sistema Operacional Linux Ubuntu Server 18.04, analise o comando a seguir.

docker run --name web -p 80:80 -p 443:443 -p 65533:22 -ti ubuntu bash

Marque V para as afirmativas verdadeiras e F para as falsas.

( ) A opção --name indica o nome que o contêiner terá, nesse caso, web. ( ) Ao executar esse comando, o docker criará um contêiner baseado em uma imagem do ubuntu. ( ) A opção -p indica que o contêiner deve debugar essas range de portas apresentadas na frente do parâmetro.

Assinale a sequência correta.

a) F, F, V

b) V, F, F

c) V, V, F

d) F, V, V

Q3) [IADES BRB 2021] Kubernetes é uma plataforma de código aberto, portável e extensiva para o gerenciamento de cargas de trabalho e serviços distribuídos em contêineres, que facilita tanto a configuração declarativa quanto a automação. Ele possui um ecossistema grande e de rápido crescimento. Serviços, suporte e ferramentas para Kubernetes estão amplamente disponíveis.

Disponível em: <https://kubernetes.io/pt-br/docs/concepts/overview/>. Acesso em: 25 jun. 2021, com adaptações.

Com base no texto apresentado e considerando o contexto do Kubernetes, assinale a alternativa que corresponde à ferramenta disponibilizada para realizar operações nos clusters Kubernetes, por meio de interface de linha de comando, pela qual é possível realizar a implantação de aplicações, inspecionar e gerenciar recursos do cluster e visualizar logs.

a) Kubeadm

b) Minikube

c) Kubectl

d) Kind

e) Kubelet

Q3) [IADES BRB 2021] Kubernetes é uma plataforma de código aberto, portável e extensiva para o gerenciamento de cargas de trabalho e serviços distribuídos em contêineres, que facilita tanto a configuração declarativa quanto a automação. Ele possui um ecossistema grande e de rápido crescimento. Serviços, suporte e ferramentas para Kubernetes estão amplamente disponíveis.

Disponível em: <https://kubernetes.io/pt-br/docs/concepts/overview/>. Acesso em: 25 jun. 2021, com adaptações.

Com base no texto apresentado e considerando o contexto do Kubernetes, assinale a alternativa que corresponde à ferramenta disponibilizada para realizar operações nos clusters Kubernetes, por meio de interface de linha de comando, pela qual é possível realizar a implantação de aplicações, inspecionar e gerenciar recursos do cluster e visualizar logs.

a) Kubeadm

b) Minikube

c) Kubectl

d) Kind

e) Kubelet

Q4) [FGV FUNSAÚDE-CE 2021] A plataforma Docker oferece a capacidade de empacotar e executar aplicativos em um ambiente isolado, denominado contêiner. A esse respeito, analise as afirmativas a seguir.

I. O isolamento e a segurança permitem que o usuário execute vários contêineres simultaneamente em um determinado host.

II. Os contêineres são leves e contêm todos os recursos necessários para executar um aplicativo, portanto, o usuário não precisa depender do que está instalado atualmente no seu host.

III. O usuário não pode compartilhar contêineres de ambientes de produção enquanto trabalha, evitando com que outros usuários interfiram na segurança do seu contêiner.

Está correto o que se afirma em

a) I, somente.

b) II, somente.

c) III, somente.

d) I e II, somente.

e) I e III, somente.

Q4) [FGV FUNSAÚDE-CE 2021] A plataforma Docker oferece a capacidade de empacotar e executar aplicativos em um ambiente isolado, denominado contêiner. A esse respeito, analise as afirmativas a seguir.

I. O isolamento e a segurança permitem que o usuário execute vários contêineres simultaneamente em um determinado host.

II. Os contêineres são leves e contêm todos os recursos necessários para executar um aplicativo, portanto, o usuário não precisa depender do que está instalado atualmente no seu host.

III. O usuário não pode compartilhar contêineres de ambientes de produção enquanto trabalha, evitando com que outros usuários interfiram na segurança do seu contêiner.

Está correto o que se afirma em

a) I, somente.

b) II, somente.

c) III, somente.

d) I e II, somente.

e) I e III, somente.

Q5) [COMPERVE TJ-RN 2020] Os volumes são mecanismos utilizados para persistir os dados gerados e usados pelos containers do Docker. Embora as montagens de ligação dependam da estrutura de diretórios da máquina host, os volumes são completamente gerenciados pelo Docker.

Considerando que um analista queira criar um volume de nome my-volume dentro de um docker, ele deve executar o comando

a) docker volume create my-volume

b) docker create volume my-volume

c) docker run create volume my-volume

d) docker create run volume my-volume

Q6) [CESPE SEFAZ-CE 2021] Julgue o item a seguir, referentes ao conjunto de práticas de desenvolvimento de software, operação e de apoio envolvidas (DevOps).

As alterações efetuadas em arquivos e diretórios copiados de uma camada base para dentro de um container docker, por padrão, são vistas pelos múltiplos containers do mesmo sistema de arquivos.

Q5) [COMPERVE TJ-RN 2020] Os volumes são mecanismos utilizados para persistir os dados gerados e usados pelos containers do Docker. Embora as montagens de ligação dependam da estrutura de diretórios da máquina host, os volumes são completamente gerenciados pelo Docker.

Considerando que um analista queira criar um volume de nome my-volume dentro de um docker, ele deve executar o comando

a) docker volume create my-volume

b) docker create volume my-volume

c) docker run create volume my-volume

d) docker create run volume my-volume

Q6) [CESPE SEFAZ-CE 2021] Julgue o item a seguir, referentes ao conjunto de práticas de desenvolvimento de software, operação e de apoio envolvidas (DevOps).

As alterações efetuadas em arquivos e diretórios copiados de uma camada base para dentro de um container docker, por padrão, são vistas pelos múltiplos containers do mesmo sistema de arquivos. ERRADO.

Q7) [FGV FUNSAÚDE-CE 2021] Pod é uma unidade atômica de escalonamento, implantação e isolamento na execução de um grupo de contêineres no Kubernetes, analise as afirmativas a seguir.

I. Um Pod garante uma mesma localização para os seus contêineres, graças a isso eles têm diversas formas de interagir com bom desempenho, por exemplo, através de troca de arquivos, uso de interface de redes ou de mecanismos de comunicação entre processos.

II. Um Pod tem um endereço IP, um nome e uma faixa de portas compartilhadas por todos os contêineres pertencentes a ele. Isso significa que os contêineres de um mesmo Pod devem ser configurados cuidadosamente a fim de evitar conflitos de portas.

III. Um Pod é um elemento persistente no tempo, ele resiste às operações de redimensionamento, falhas de verificação de sanidade de contêineres e migrações entre nós.

Está correto o que se afirma em

a) I, somente.

b) II, somente.

c) III, somente.;

d) I e II, somente.

e) I e III, somente.

Q7) [FGV FUNSAÚDE-CE 2021] Pod é uma unidade atômica de escalonamento, implantação e isolamento na execução de um grupo de contêineres no Kubernetes, analise as afirmativas a seguir.

I. Um Pod garante uma mesma localização para os seus contêineres, graças a isso eles têm diversas formas de interagir com bom desempenho, por exemplo, através de troca de arquivos, uso de interface de redes ou de mecanismos de comunicação entre processos.

II. Um Pod tem um endereço IP, um nome e uma faixa de portas compartilhadas por todos os contêineres pertencentes a ele. Isso significa que os contêineres de um mesmo Pod devem ser configurados cuidadosamente a fim de evitar conflitos de portas.

III. Um Pod é um elemento persistente no tempo, ele resiste às operações de redimensionamento, falhas de verificação de sanidade de contêineres e migrações entre nós.

Está correto o que se afirma em

a) I, somente.

b) II, somente.

c) III, somente.;

d) I e II, somente.

e) I e III, somente.

Q8) [INSTITUTO AOCP MJSP 2020] O Docker possibilita que uma imagem com todos os aplicativos e configurações realizadas em um contêiner sejam transferidos para outro host, bastando que este tenha o Docker instalado. Assinale a alternativa que apresenta o nome dessa operação.

a) Restore.

b) Dump.

c) Drill down Contêiner.

d) Portabilidade.

e) Drill up Contêiner.

Q9) [CESPE SEFAZ-CE 2021] Julgue o item a seguir, referentes ao conjunto de práticas de desenvolvimento de software, operação e de apoio envolvidas (DevOps).

Com a implantação do Kubernetes, é obtido um cluster com pelo menos um nó de trabalho (worker node); os nós de trabalho, por sua vez, hospedam vários componentes da carga de trabalho do aplicativo.

Q8) [INSTITUTO AOCP MJSP 2020] O Docker possibilita que uma imagem com todos os aplicativos e configurações realizadas em um contêiner sejam transferidos para outro host, bastando que este tenha o Docker instalado. Assinale a alternativa que apresenta o nome dessa operação.

a) Restore.

b) Dump.

c) Drill down Contêiner.

d) Portabilidade.

e) Drill up Contêiner.

Q9) [CESPE SEFAZ-CE 2021] Julgue o item a seguir, referentes ao conjunto de práticas de desenvolvimento de software, operação e de apoio envolvidas (DevOps).

Com a implantação do Kubernetes, é obtido um cluster com pelo menos um nó de trabalho (worker node); os nós de trabalho, por sua vez, hospedam vários componentes da carga de trabalho do aplicativo. CERTO.

Q10) [COMPERVE TJ-RN 2020] Uma imagem do Docker é criada a partir de uma série de camadas, onde cada uma representa uma instrução no Dockerfile da imagem. Considerando que um analista do Tribunal de Justiça queira listar as camadas (layers) da imagem docker mailserver, ele deve executar o comando a) docker expose mailserver

b) docker layers mailserver

c) docker history mailserver

d) docker image mailserver

Q11) [CESPE SERPRO 2021] A respeito do Kubernetes, julgue o próximo item.

Para obter o status de um node nomeado como node1 em um cluster, deve ser executado o comando a seguir.

Kubectl describe node node1

Q10) [COMPERVE TJ-RN 2020] Uma imagem do Docker é criada a partir de uma série de camadas, onde cada uma representa uma instrução no Dockerfile da imagem. Considerando que um analista do Tribunal de Justiça queira listar as camadas (layers) da imagem docker mailserver, ele deve executar o comando

a) docker expose mailserver

b) docker layers mailserver

c) docker history mailserver

d) docker image mailserver

Q11) [CESPE SERPRO 2021] A respeito do Kubernetes, julgue o próximo item.

Para obter o status de um node nomeado como node1 em um cluster, deve ser executado o comando a seguir. CERTO.

Kubectl describe node node1

Q12) [COMPERVE TJ-RN 2020] Uma imagem de container do Docker é um pacote de software leve, independente e executável que inclui tudo o que é necessário para executar uma aplicação. Na criação de um arquivo Dockerfile, a instrução EXPOSE

a) mapeia uma porta externa para uma porta interna à rede Docker.

b) divulga uma porta (TCP ou UDP) para os hosts externos à rede Docker.

c) expõe um serviço do container para a rede Docker default.

d) documenta quais portas se pretende publicar.

Q13) [CESPE SEFAZ-CE 2021] Julgue o próximo item, relativos ao Apache Kafka e ao Kubernetes.

No Kubernetes, kubelet é uma pequena aplicação localizada em um nó que se comunica com o plano de controle, assegurando que os containers estejam em execução em um pod, que consiste no menor e mais simples objeto do Kubernetes.

Q12) [COMPERVE TJ-RN 2020] Uma imagem de container do Docker é um pacote de software leve, independente e executável que inclui tudo o que é necessário para executar uma aplicação. Na criação de um arquivo Dockerfile, a instrução EXPOSE

a) mapeia uma porta externa para uma porta interna à rede Docker.

b) divulga uma porta (TCP ou UDP) para os hosts externos à rede Docker.

c) expõe um serviço do container para a rede Docker default.

d) documenta quais portas se pretende publicar.

Q13) [CESPE SEFAZ-CE 2021] Julgue o próximo item, relativos ao Apache Kafka e ao Kubernetes.

No Kubernetes, kubelet é uma pequena aplicação localizada em um nó que se comunica com o plano de controle, assegurando que os containers estejam em execução em um pod, que consiste no menor e mais simples objeto do Kubernetes. CERTO.

Q14) [CESPE SERPRO 2021] A respeito do Kubernetes, julgue o próximo item.

A camada de gerenciamento possui o componente etcd, cuja função é observar pods que foram criados sem nenhum node atribuído e selecionar um node para execução.

Q15) [CESPE SLU-DF 2019] No que diz respeito a ferramentas de desenvolvimento, julgue o item a seguir.

O Docker é uma ferramenta open source que permite a criação de ambientes virtuais por meio de Linux Containers, sendo uma das vantagens dos contêineres Docker fornecer uma virtualização em nível de sistema operacional, o que isola as aplicações em execução e não utiliza tantos recursos da máquina quanto as máquinas virtuais.

Q14) [CESPE SERPRO 2021] A respeito do Kubernetes, julgue o próximo item.

A camada de gerenciamento possui o componente etcd, cuja função é observar pods que foram criados sem nenhum node atribuído e selecionar um node para execução. ERRADO.

Q15) [CESPE SLU-DF 2019] No que diz respeito a ferramentas de desenvolvimento, julgue o item a seguir.

O Docker é uma ferramenta open source que permite a criação de ambientes virtuais por meio de Linux Containers, sendo uma das vantagens dos contêineres Docker fornecer uma virtualização em nível de sistema operacional, o que isola as aplicações em execução e não utiliza tantos recursos da máquina quanto as máquinas virtuais. CERTO.

Q16) [FEPESE Prefeitura Florianópolis - SC 2019] O Docker pode ler instruções através de um arquivo texto que contém instruções para montar uma imagem (dockerfile).

Nesse contexto, qual a palavra-chave ou instrução que indica ao docker a porta que o container deve utilizar em tempo de execução?

a) HTTP

b) PORT

c) EXPOSE

d) DOCKER

e) SERVERPORT

Q16) [FEPESE Prefeitura Florianópolis - SC 2019] O Docker pode ler instruções através de um arquivo texto que contém instruções para montar uma imagem (dockerfile).

Nesse contexto, qual a palavra-chave ou instrução que indica ao docker a porta que o container deve utilizar em tempo de execução?

a) HTTP

b) PORT

c) EXPOSE

d) DOCKER

e) SERVERPORT

Q17) [CCV-UFC UFC 2019] Dockerfile é um um arquivo de texto que contém todos os comandos, em ordem, necessários para construir uma determinada imagem Docker. Sobre as instruções contidas em um Dockerfile, assinale a alternativa correta.

a) A instrução VOLUME configura o tamanho da imagem.

b) A instrução ENV adiciona metadados para uma imagem.

c) A instrução WORKDIR permite a criação de um diretório no host onde ficam armazenados os dados do container.

d) A instrução EXPOSE informa ao Docker que o container escuta nas portas de rede especificadas em tempo de execução.

e) A instrução FROM configura qual será a aplicação principal do container, sendo executada após a inicialização do container.

Q17) [CCV-UFC UFC 2019] Dockerfile é um um arquivo de texto que contém todos os comandos, em ordem, necessários para construir uma determinada imagem Docker. Sobre as instruções contidas em um Dockerfile, assinale a alternativa correta.

a) A instrução VOLUME configura o tamanho da imagem.

b) A instrução ENV adiciona metadados para uma imagem.

c) A instrução WORKDIR permite a criação de um diretório no host onde ficam armazenados os dados do container.

d) A instrução EXPOSE informa ao Docker que o container escuta nas portas de rede especificadas em tempo de execução.

e) A instrução FROM configura qual será a aplicação principal do container, sendo executada após a inicialização do container.