NÃO PODE FALTAR

CONTEINERIZAÇÃO

Caique Silva Pereira

O QUE É CONTEINERIZAÇÃO?

A conteinerização envolve a ação de encapsular um aplicativo em um recipiente com o seu próprio ambiente operacional, pode ser considerado a emulação da aplicação. Ela é uma alternativa mais leve que a utilização de uma máquina virtual.

Fonte: Shutterstock.

Deseja ouvir este material?

Áudio disponível no material digital.

PRATICAR PARA APRENDER

Caro aluno, o desenvolvimento de um software, seja ele desktop, web ou um app para smartphone, envolve a utilização de diversos recursos, por exemplo, um sistema gerenciador de banco de dados, um (ou mais) ambiente integrado de desenvolvimento, uma (ou mais) linguagem de programação etc. Ao fazer suas escolhas e implementar a solução, o próximo passo é disponibilizá-la, mas, nesse momento, muitos problemas de incompatibilidade podem ocorrer. Uma frase típica é “mas no meu computador funcionou!”. Será que existem soluções para evitar esse tipo de problema? Existem e começaremos a explorá-las!

Nesta seção vamos ter a oportunidade de conhecer uma tecnologia muito requisitada no mercado de trabalho atualmente: os contêineres. Você já ouviu falar de uma das implementações mais utilizadas de contêineres no mundo, o Docker? Se não, aproveite a oportunidade. Sabia que há várias vagas de emprego requisitando conhecimentos dessa implementação aqui no Brasil e mundo afora? Não perca o foco!

Você se lembra da sua entrevista para a tão cobiçada vaga na área de DevOps? Avançando mais uma etapa, o coordenador pergunta se, na sua opinião, existem tecnologias similares, porém, mais eficientes que a virtualização, para rodar aplicações sem interferir no sistema operacional da máquina física, e você logo se

lembra dos contêineres. Sendo assim, deverá criar um relatório técnico indicando o uso de uma tecnologia atual que traga vantagens e diminua o consumo de recursos para a criação de um novo ambiente que execute um sistema ERP que será implementado daqui a alguns meses na empresa. Os ERPs são sistemas integrados de gestão empresarial, que interligam todos os dados e processos de uma empresa dentro do sistema.

Portanto é um dos mais importantes para um bom funcionamento de uma companhia, o que quer dizer que, quanto melhor for executado, mais pontos o profissional que planejou sua execução vai ganhar com os seus superiores. Então, chegou a hora de impressionar o coordenador em busca da vaga que você almeja, está pronto?

CONCEITO-CHAVE

Uma das tecnologias mais populares que temos atualmente é o uso de contêineres para a execução de sistemas dos mais variados tipos. Isso ocorre devido à facilidade e à flexibilidade que advêm do uso dos mesmos. O contêiner funciona como uma tecnologia que dá o suporte para o funcionamento de uma aplicação e pode ser considerado a emulação de nossa aplicação. Quando a aplicação é executada através de um contêiner, ela tem todas as bibliotecas e os elementos necessários para o funcionamento disponíveis dentro do contêiner. Uma maneira simples para entender o que são os chamados contêineres é imaginar que eles permitem a criação de ambientes virtuais isolados e independentes para serem utilizados por aplicações, similar ao resultado do uso de máquinas virtuais. Entretanto, um grande diferencial está no fato de que os contêineres são mais leves que as máquinas virtuais, por possuírem uma arquitetura mais otimizada.

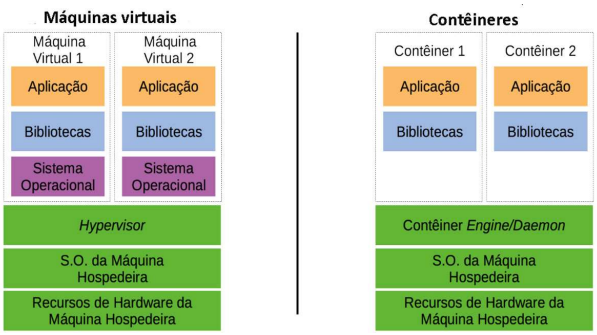
> **ASSIMILE**
>
> O contêiner traz muitas facilidades e é considerado uma das principais tendências de TI. Sua utilização simplifica a aplicação da metodologia DevOps e facilita o desenvolvimento. Grandes empresas, como a Google, usam essa tecnologia.

Se fosse necessário desenvolver um sistema em linguagem C para o cadastro de produtos do estoque de uma loja de varejo, poderíamos criar um contêiner que tivesse todas as bibliotecas essenciais para o funcionamento do sistema e, assim, nossa aplicação seria virtualizada através de um contêiner, sem a necessidade de um sistema operacional, pois, neste caso, já precisaríamos de uma máquina virtual.

Conforme pode ser visto na Figura 4.10, uma grande vantagem de contêineres em relação às máquinas virtuais é que não há, obrigatoriamente, a necessidade de instalar um sistema operacional completo, visto que as plataformas de conteinerização aproveitam bibliotecas compartilhadas com o sistema operacional hospedeiro. Por essa razão, os contêineres ocupam menos espaço em disco e

consomem menos RAM e processamento que as máquinas virtuais e, assim, possibilita a utilização de mais contêineres em uma mesma máquina física, favorecendo o uso de uma arquitetura mais modular para as aplicações.

Figura 4.10 | Comparação das arquiteturas de máquinas virtuais versus contêineres



Fonte: elaborada pelo autor.

REFLITA

Agora que conhecemos algumas vantagens dos contêineres em relação às máquinas virtuais, quais benefícios uma empresa que executa aplicações em máquinas virtuais poderia adquirir migrando a execução de suas aplicações para contêineres? Você acredita que seria benéfica essa migração?

Duas das principais características a favor da conteinerização são o baixo acoplamento entre os contêineres e a facilidade de migração entre provedores de *cloud computing*. Ambas se devem ao fato de que a ideia do contêiner é "empacotar" a sua aplicação em um módulo que é facilmente instalado em qualquer sistema operacional que suporte o uso de contêineres (e os principais sistemas operacionais utilizados em servidores possuem esse suporte).

No lugar do *hypervisor*, quando tratamos de máquinas virtuais, temos os chamados contêiner engines (por vezes chamados de contêiner *daemons*). Existem várias implementações para esses *engines*, como o Docker, o LXD, o Rkt, o Mesos e o Windows Server Containers, mas o mais popular entre eles é o Docker, sobre o qual aprenderemos na próxima seção. Porém, para começar a ter ideia do que se trata, quando falamos do Docker, estamos na realidade falando de uma empresa – chamada Docker – que foi responsável pela popularização dos contêineres, por meio de eventos e divulgação de material técnico. Essa companhia criou sua própria implementação que, coincidentemente, é chamada de Docker, cuja instalação, configuração e gerenciamento (e, consequentemente, curva de aprendizado) é relativamente mais simples comparando com outras implementações (DOCKER, 2018).

LINUX CONTAINER

Existem implementações que podem ser consideradas contêineres de sistema, por exemplo, os Linux. Essas tecnologias têm um comportamento muito parecido ao de uma máquina virtual, mas, na verdade, são equivalentes a novas instâncias do sistema utilizando todas as técnicas disponíveis para isso. Os contêineres Linux

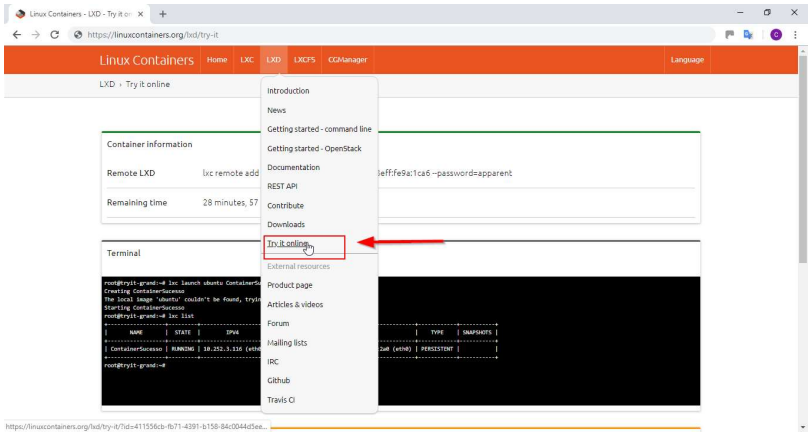
funcionam compartilhando o kernel da máquina real onde estão em funcionamento. Uma limitação encontrada nos contêineres é que, quando estão compartilhando o kernel da máquina física ou até mesmo virtual em que estão hospedados e em execução, tratam-se de contêineres Linux rodando em máquinas Linux com o mesmo kernel do host, ou seja, mesmo que você substitua todas as bibliotecas e frameworks utilizados pelo seu contêiner, o kernel será sempre o mesmo do host que o hospeda.

Segundo Linux Container (2018), o projeto Linux container é o guarda-chuva por trás do Linux Containers, que possuem as gerações LXC, LXD e LXCFS. O LXC é uma interface de espaço de usuário para os recursos de contenção de kernel do Linux. Por meio de uma API poderosa e ferramentas simples, permite que usuários do Linux criem e gerenciem facilmente contêineres de sistema ou aplicativo. O LXD é um gerenciador de contêineres de próxima geração, que oferece uma experiência de usuário semelhante às máquinas virtuais, mas usando contêineres do Linux. Já o LXCFS é um sistema de arquivos simples do *userspace* projetado para contornar algumas limitações atuais do kernel do Linux.

EXEMPLIFICANDO

Podemos até mesmo testar um ambiente com contêineres Linux pelo Portal *Linux Containers* (LINUX CONTAINER, [s.d.]). Esse portal disponibiliza toda a documentação com comandos utilizados no LXC e no LXD. Conforme mencionado, podemos praticar alguns comandos e testar ambos os ambientes através do menu "*try it online*", como podemos observar na Figura 4.11:

Figura 4.11 | Área de treinamento do portal Linux Containers



Fonte: captura de tela do portal *Linux Containers*.

Dentro da opção de treinamentos, é possível usar o "*Terminal*", disponível para executar alguns comandos e observar seu comportamento. Utilizamos o comando abaixo para criar o contêiner através da implementação Linux Containers:

```
lxc launch ubuntu MeuPrimeiroConteiner
```

Pode-se observar o contêiner com o nome "*MeuPrimeiroConteiner*" sendo criado no terminal e dando o retorno conforme é mostrado na Figura 4.12 a seguir.

Figura 4.12 | Criando um contêiner por meio do portal Linux Containers

```
root@tryit-romantic:~# lxc launch ubuntu MeuPrimeiroConteiner
Creating MeuPrimeiroConteiner

Starting MeuPrimeiroConteiner
```

Fonte: captura de tela do portal *Linux Containers*.

Feito isso, é possível consultar a lista de contêineres criados no terminal utilizando o comando abaixo:

```
lxc list
```

Pode-se observar na Figura 4.13 que o contêiner criado anteriormente já aparece como resultado em nossa listagem. Além disso, informações como o status da máquina e os números de seus respectivos IPV4 e IPV6 também aparecem, como podemos ver a seguir:

Figura 4.13 | Listando os contêineres criados através do portal *Linux Containers*

```
root@tryit-romantic:~# lxc list
+---------------------+---------+---------------------+----------------------------------------------+-----------+-----------+
|         NAME        |  STATE  |        IPV4         |                    IPV6                      |    TYPE   | SNAPSHOTS |
+---------------------+---------+---------------------+----------------------------------------------+-----------+-----------+
| MeuPrimeiroConteiner | RUNNING | 10.69.117.143 (eth0) | 2001:470:b368:1070:216:3eff:fe06:4d1f (eth0) | PERSISTENT |           |
+---------------------+---------+---------------------+----------------------------------------------+-----------+-----------+
```

Fonte: captura de tela do portal *Linux Containers*.

É possível executar comandos dentro de um contêiner utilizando o comando abaixo:

```
lxc exec nomedocontainer – comando
```

Há opções de iniciar, parar e excluir contêineres usando os comandos:

```
lxc start
```

```
lxc stop
```

```
lxc delete
```

Além das informações do contêiner exibidas através da listagem que foi vista anteriormente, podemos observar uma série de informações sobre nosso contêiner como consumo de memória RAM, consumo de redes, entre outros dados. Para isso, utilizamos o comando de sintaxe a seguir:

```
lxc info nomedocontainer
```

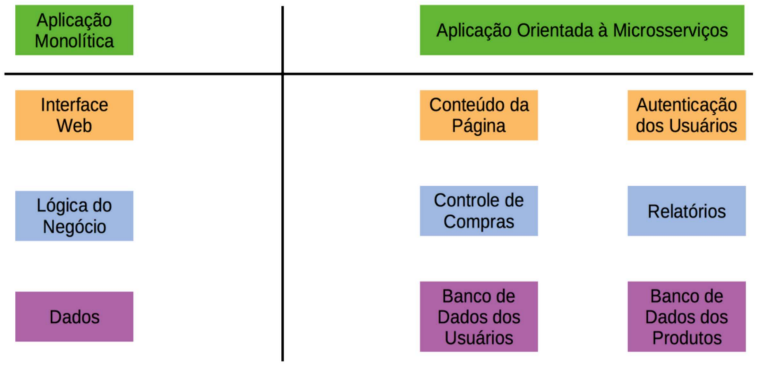
O PAPEL DA CONTEINERIZAÇÃO EM SISTEMAS DISTRIBUÍDOS

Os sistemas distribuídos fazem uso extensivo dos contêineres no contexto de microsserviços. A ideia dos microsserviços está associada a empresas que possuem sistemas altamente dinâmicos e ao termo modularidade. Um portal de notícias por exemplo, é composto por vários elementos, como um (ou mais) banco de dados, um (ou mais) *framework front-end* utilizado para desenvolver interfaces, *framework back-end* para desenvolver a parte dinâmica do sistema, frameworks para gerenciar mensagens entre servidores e clientes (por exemplo, o RabbitMQ) etc. Caso o sistema possua uma arquitetura monolítica, ou seja, uma forte dependência entre esses elementos, será muito difícil substituir alguns dos elementos citados anteriormente sem causar uma interrupção completa no sistema. Por outro lado, em uma arquitetura baseada em microsserviços, esses

componentes têm um baixo acoplamento entre si, ou seja, o grau de dependência entre os componentes é baixo, de forma que, caso você tenha de fazer uma substituição, terá um impacto bem menor na indisponibilidade do seu sistema, e os contêineres serão artefatos fundamentais para atingir esse baixo acoplamento, pois cada um dos elementos pode ser criado e implantado em um contêiner separado.

Dentre as vantagens de um sistema distribuído baseado em microsserviços, podemos apontar que quanto menores são as partes, mais fácil entendê-las. Além disso, cada microsserviço pode ser executado e escalado de maneira concorrente e independente entre si. Outra vantagem decorrente disso é que, como esses elementos possuem um baixo acoplamento entre si, um projeto de grande porte pode ser trabalhado de maneira razoavelmente independente entre as equipes de trabalho. A Figura 4.14 ilustra uma aplicação monolítica em relação a uma aplicação baseada em microsserviços.

Figura 4.14 | Exemplo de aplicações baseadas em microsserviços



Fonte: elaborada pelo autor.

ASSIMILE

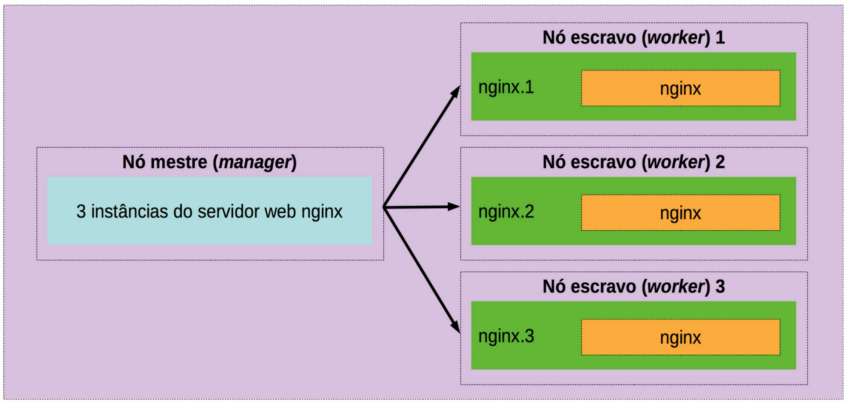
Atualmente, existe uma grande tendência de empresas migrarem de aplicações monolíticas para aplicações orientadas a microsserviços, por vantagens como escalabilidade e independência entre seus elementos (também conhecido como baixo acoplamento).

Como veremos na próxima seção (parte prática), apesar de a criação e o controle de um contêiner ser feita com simples comandos, podemos observar pela Figura 4.14 que uma aplicação é composta por não apenas um, mas vários microsserviços, cada um representando um ou mais contêineres. Gerenciar dezenas ou centenas de contêiner, de maneira isolada, pode ser uma tarefa muito trabalhosa, razão pela qual as empresas utilizam alguma ferramenta para criar, gerenciar e remover contêineres. Esse tipo de ferramenta é conhecido no mercado de trabalho como ferramenta de orquestração de contêineres. Acredita-se que a ideia esteja relacionada ao fato de que, em uma orquestra, um maestro coordena os músicos de maneira que o objetivo conjunto (obra tocada) seja o melhor possível. Nessa analogia, as ferramentas de orquestração realizam o mesmo papel do maestro, em que os músicos seriam os contêineres.

Atualmente, a ferramenta de orquestração mais popular e, portanto, mais solicitada no mercado de trabalho é o Kubernetes, da Google, que serve para orquestrar contêineres criados com o Docker. Importante notar que o próprio *framework* do Docker possui uma ferramenta de orquestração nativa, instalada automaticamente com o Docker. Essa ferramenta é chamada de Swarm. Em alguns aspectos, inclusive pelo fato de ser uma ferramenta totalmente integrada com o Docker, o Swarm é a mais indicada para um primeiro contato com o conceito de orquestração de contêineres, razão pela qual essa será a ferramenta adotada nesse livro.

O Swarm foi integrado ao Docker a partir da versão 1.12.0, com o chamado *swarm mode*, em junho de 2016, conforme noticiado no blog oficial do Docker (DOCKER, [s.d.]) A Figura 4.15 ajudará a entender os componentes do Swarm e como eles estão relacionados aos contêineres.

Figura 4.15 | Arquitetura do Swarm do Docker



Fonte: elaborada pelo autor.

Na Figura 4.15, o retângulo destacado em lilás representa o que o Docker chama de Swarm, que nada mais é que um *cluster* (conjunto de computadores interligados que funcionam como um grande sistema), formado por vários nós, que podem rodar uma aplicação – no exemplo, o servidor web Nginx – de maneira integrada e distribuída. Para que os nós possam estar integrados, de maneira que, caso uma instância do Nginx falhe por conta de um dos nós escravos estar indisponível, esta ser automaticamente instanciada em outro nó, é necessário que exista um nó mestre que faça essa gestão. Podem existir vários nós com a função de mestre do cluster (que o Docker chama de manager), para que, caso o nó mestre falhe, outro nó mestre assuma seu lugar. Os nós escravos (que o Docker chama de *worker*) rodam a quantidade de instâncias (frequentemente chamadas de réplicas) solicitadas pelo nó mestre e, para tal, é preciso que exista um contêiner "dentro" de cada nó escravo, o que está representado pelo retângulo laranja. Caro aluno, chegamos ao fim da seção e agora você já sabe o que é contêiner e logo mais estará apto para utilizar essa importante tecnologia! Bons estudos!

FAÇA VALER A PENA

Questão 1

Existem algumas ferramentas, chamadas de ferramentas de “orquestração de contêineres”, que facilitam e automatizam o gerenciamento de um conjunto de contêineres. Assim sendo, é de extrema importância a familiaridade com esse tipo de ferramenta.

Assinale a alternativa que apresenta somente ferramentas de orquestração de contêineres.

a. Docker e Kubernetes.

b. Mesos e Docker.

c. Mesos, Docker e Kubernetes.

d. Kubernetes e Swarm.

e. Docker e Swarm.

Questão 2

Os contêineres têm se popularizado cada vez mais devido a uma série de vantagens desse tipo de tecnologia, se comparados à tecnologia antecessor, de máquinas virtuais. Uma alternativa para implementação de contêiner é o Linux Container, embora existam outras implementações.

De acordo com os comandos LXC do Linux Container, assinale a alternativa que mostra corretamente a sintaxe para a criação de um novo contêiner ubuntu, com o nome `NovoConteiner`.

a. `lxc start NovoConteiner`

b. `lxc start ubuntu NovoConteiner`

c. `lxc launch ubuntu NovoConteiner`

d. `lxc create ubuntu NovoConteiner`

e. `lxc new NovoConteiner`

Questão 3

O Linux Container LXC e LXD tem seu uso se tornando cada vez mais popular, tanto que as ferramentas de orquestração de contêineres dão suporte a esta implementação. Sendo assim, é importante que conheçamos alguns comandos básicos dessa implementação.

Em relação ao seguinte comando, assinale a alternativa que apresenta a sintaxe correta do comando que traz informações do contêiner como seu status de ligado ou desligado, endereço de IPV4, endereço de IPV6, entre outras informações. Vamos utilizar um contêiner chamado `Conteiner_Kro`.

a. `lxc info Conteiner_Kro`

b. `lxc list`

c. `lxc launch ubuntu Conteiner_Kro`

d. `lxc view infos Conteiner_Kro`

e. `lxc Conteiner_Kro status,ipv4,ipv6 show`

Ver anotações

REFERÊNCIAS

DOCKER, C. **Docker Containerization Unlocks the Potential for Dev and Ops**. 2018. Disponível em: https://dockr.ly/2YTojVl. Acesso em: 10 dez. 2018.

DOCKER (Portal). **Swarm mode**. [s.d.] Disponível em: https://dockr.ly/3ryJY18. Acesso em: 10 dez. 2018.

LINUX CONTAINER. **Infrastructure for container projects**. Disponível em: https://bit.ly/3aKahe7. Acesso em: 10 dez. 2018.