Sistemas Distribuídos

Anhanguera


AVALIE
SUA PROFISSÃO
QUANDO APARECER EM SEU PORTAL UMA AVALIAÇÃO SOBRE SEU CURSO, RESPONDA:
NOTAS
9 OU 10
SIGNIFICA QUE VOCÊ INDICA
NOTAS
7 OU 8
SIGNIFICA QUE VOCÊ NÃO INDICA
Anhanguera

Anhanguera

Simulando sistemas distribuídos com Docker

Anhanguera

O Docker é uma famosa plataforma genérica de conteinerização. Conforme já estudamos, o conceito de conteinerização é parecido com virtualização, porém é considerado “mais leve”. Contêineres são muito populares atualmente devido à facilidade e à flexibilidade que advêm de seu uso. Portanto, agora chegou a hora de colocar a mão na massa e aprender a utilizar essa famosa ferramenta.

Instalação do Docker

Vamos fazer a instalação do Docker em uma das distribuições populares GNU/Linux, o sistema operacional Ubuntu, cuja versão utilizada foi a 14.04.5 LTS. Para isso, devemos seguir os passos a seguir. Todo o procedimento deve ser feito com um usuário com permissões de administrador. Nesse caso, utilizaremos o root através do comando sudo su.

Você também pode instalar o Docker no sistema operacional Windows, fazendo o download da versão mais atual. O processo de instalação é bem simples, é preciso somente avançar as etapas do instalador.

Antes de instalar a ferramenta, devemos remover versões anteriores do Docker que possam estar instaladas, usando o comando:

sudo apt-get remove docker docker-engine docker.io

Caso não tenha nenhuma versão instalada, será exibida a mensagem que foi impossível encontrar o pacote docker-engine. Antes de instalar o Docker CE pela primeira vez em uma nova máquina host, você precisa configurar o repositório do Docker, atualizando os pacotes de sua máquina. Depois, você pode instalar e atualizar o Docker do repositório. Portanto, execute os comandos que vão atualizar a máquina:

sudo apt-get update

O comando acima tem o objetivo de realizar uma atualização de pacotes do Ubuntu. Na execução desse comando, o tempo pode variar de acordo com os pacotes que precisam ser atualizados, da velocidade da máquina e da conexão com a internet.

Através do comando abaixo atualizamos os pacotes necessários para a instalação do Docker. Para obter uma instalação bem-sucedida, é bom ter os programas utilizados nela em suas últimas versões:

sudo apt-get install \ apt-transport-https \ cacertificates
\ curl \ software-properties-common

Após atualizar os repositórios, vamos adicionar o repositório de instalação do Docker, usando o seguinte comando:

curl -fsSL https://download.docker.com/linux/
ubuntu/gpg | sudo apt-key add -

Nesse comando, estamos fazendo o apontamento do caminho de instalação oficial do Docker que o Ubuntu deve acessar quando queremos efetuar a instalação. Após o apontamento da URL indicando o local para download o Docker para Ubuntu, será possível adicionar o repositório no próximo comando:

sudo add-apt-repository \ "deb [arch=amd64]
*https://download.docker.com/linux/ubuntu \ $(lsb_*
release -cs) \ stable"

O comando utiliza a permissão considerada admin nos sistemas operacionais da família Linux, através da palavra sudo. Depois de usar a permissão de usuário do sudo, utilizamos o add-apt-repository para adicionar o repositório, que pode ser comparado a uma loja de aplicativos, responsável pelo download do Docker na versão Ubuntu. O restante do comando é o caminho do repositório.

Após adicionar o repositório para download do Docker, como mostra a Figura 3.17, devemos mais uma vez atualizar o apt-get, conforme o comando seguinte para aplicar as alterações:

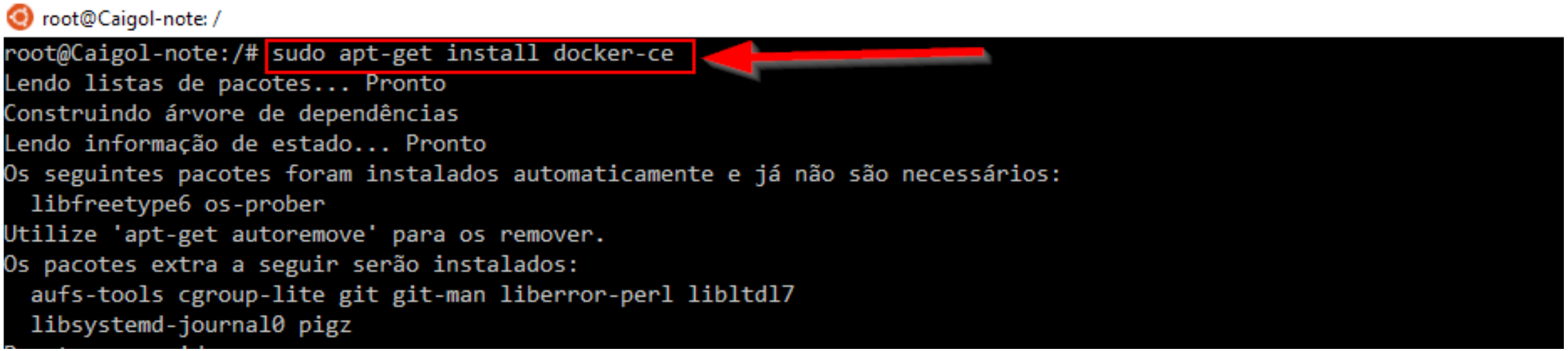
sudo apt-get update

```
root@Caigol-note:/# sudo add-apt-repository \
>    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
>    $(lsb_release -cs) \
>    stable"
root@Caigol-note:/# sudo apt-get update
```

Agora vamos utilizar o comando de instalação do Docker. Lembrando que, para que esse comando funcione, devemos seguir as etapas de apontar o repositório em que o Docker está disponível e adicionar esse repositório em nossa lista. No comando utilizamos o sudo para usar a permissão admin do sistema e o apt-get install para a fazer a instalação, por último o nome do programa que vamos instalar, no caso, o Docker (docker-ce).

sudo apt-get install docker-ce

Com todas as configurações feitas para atualizar os pacotes necessários e adicionar o repositório que contém o Docker, agora é possível fazer a instalação. Veja na Figura 3.18 o comando sendo aplicado no terminal e também o início da sua execução.


root@Caigol-note: /
root@Caigol-note:/# sudo apt-get install docker-ce
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
Os seguintes pacotes foram instalados automaticamente e já não são necessários:
libfreetype6 os-prober
Utilize 'apt-get autoremove' para os remover.
Os pacotes extra a seguir serão instalados:
aufs-tools cgroup-lite git git-man liberror-perl libltdl7
libsystemd-journal0 pigz

Anhanguera

Iniciando e testando o Docker

Para ver se o Docker foi instalado corretamente, devemos iniciar o serviço do Docker e verificar se ele está em execução. Podemos fazer isso com os seguintes comandos:

sudo service docker start

service docker status

O comando sudo utiliza a permissão de usuário administrador para execução do restante da linha, e as instruções service docker start iniciam o serviço Docker que foi instalado anteriormente. Após a inicialização do Docker, utilizamos o comando service docker status que mostra o status do serviço, ou seja, se ele está em execução ou parado.

Caso ocorra algum erro ao iniciar e testar o Docker, você deve executar os dois comandos seguintes e tentar novamente:

sudo apt-get update

sudo apt-get upgrade

Esses comandos utilizam a permissão de usuário administrador através do sudo e, além disso, executam uma atualização de pacotes essenciais para o bom funcionamento do sistema operacional, assim como o bom funcionamento dos serviços que rodam através dele, como o Docker.

Agora que instalamos e verificamos seu funcionamento, o sistema está apto a receber as especificidades que queremos criar. Para isso, é possível usar o Docker Swarm. Essa ferramenta é nativa e permite a criação de clusters de Docker. Nesse cenário, é possível agrupar vários hosts em um mesmo pool de recursos, o que facilita o deploy de contêineres (DIEDRICH, 2018). Esse framework é integrado ao Docker Engine a partir da versão 1.12.0, com o chamado “swarm mode”.

Uma alternativa à instalação do Docker é o portal Play With Docker, plataforma pela qual é possível testar o Docker utilizando seus comandos diretamente via navegador, mediante um cadastro.

Após a criação do login na plataforma, teremos acesso a uma interface para criação de instâncias como clusters e nós.

Através dos comandos listados abaixo podemos simular algumas características de sistemas distribuídos com Docker, como criação de nós dos tipos mestre (manager) e escravo (worker) dentro do nosso cluster, gerenciamento de contêiner, réplica de serviços etc.

Comando executado fora do(s) nó(s):

Agora vamos ver alguns comandos que são executados fora dos nós do ambiente Docker. Um dos mais importantes é o de criação de um novo manager para o cluster. Através do comando, criamos um manager chamado de mestre:

docker-machine create --driver virtualbox mestre

O começo desse comando significa que estamos criando uma nova máquina através do Docker. Utilizaremos docker-machine create como driver no VirtualBox e daremos o nome de mestre para nossa máquina.

Agora, vamos fazer a criação de uma nova máquina: o comando segue o mesmo padrão do anterior, mas desta vez vamos chamar a máquina de “escravo1”. Ela será um worker para o cluster, conforme comando abaixo:

docker-machine create --driver virtualbox escravo1

Podemos verificar o IP de qualquer uma das máquinas de nosso cluster utilizando o comando docker-machine ip e, a seguir, consultamos o IP de nossa máquina chamada “mestre”:

docker-machine ip mestre

Um dos comandos Linux mais populares utilizados para acesso de máquinas/servidores é o ssh, que significa “Secure Shell”, em que fazemos um acesso ou uma conexão através desse protocolo de rede criptográfica, que aplica mais segurança aos serviços de redes. Na utilização do Docker o mesmo comando é usado com o objetivo de acessar os nós criados que, em nosso ambiente, possuem os nomes “mestre” e “escravo1”.

docker-machine ssh mestre

docker-machine ssh escravo1

É possível verificar os nós (nodes) criados através do comando “ls”. Podemos criar um escravo para o cluster com hardware diferente do padrão. Também pode-se ver no comando a seguir que a máquina chamada “escravo5” que definimos a memória com o valor de 512 MB através da instrução virtualbox-memory “512” e um disco rígido de 5 GB através da instrução virtualbox-disk-size “5000”:

docker-machine create --virtualbox-memory “512”--virtualbox-disk-size “5000” --driver virtualbox escravo5

Comando executado dentro do(s) nó(s) manager:

Agora vamos ver os alguns comandos executados dentro do manager de nosso ambiente Docker. O comando a seguir pode ser utilizado para iniciar o cluster através do framework Swarm, de gerenciamento de contêineres.

docker swarm init --advertise-addr <IP do mestre>

Podemos consultar os nós que fazem parte do cluster utilizando o comando node ls:

docker node ls

Às vezes precisamos verificar informações sobre um nó específico. Para isso, utilizamos o comando Docker inspect. O exemplo a seguir contém informações sobre o nó chamado “escravo1”:

docker inspect escravo1

A seguir, é possível observar um exemplo de criação de um novo serviço (tarefa) para o cluster. Nesse caso, criamos um serviço Web através do popular servidor web nginx, que contém 3 réplicas:

d*ocker service create --name WEB --publish 85:80 --replicas=3 nginx:1.12.1*

No parâmetro name definimos um nome para a execução do serviço, em nosso caso, WEB. Através do parâmetro publish definimos uma porta de execução para o Docker e uma para o servidor web Nginx, no caso, “85” e “80” e, por último, definimos que o serviço terá 3 réplicas e utilizará o Nginx na versão 1.12.1.

Agora que criamos nosso serviço de internet chamado WEB, podemos utilizar o comando ps seguido do nome do serviço para verificar suas informações. Portanto, o comando a seguir mostra informações sobre um serviço específico aqui denominado como WEB:

docker service ps WEB

É possível usar os comandos a seguir para alterar a versão das instâncias do Nginx. No primeiro comando atualizamos, através do comando update, a versão do Nginx para 1.13.5 (anteriormente estávamos com a versão 1.12.1). Após isso, utilizamos o comando ps com o parâmetro –f que significa filter ou find, que pode ser considerado um filtro para a busca, em que usamos a palavra-chave Running para descobrir qual serviço está em execução e se sua versão foi mesmo atualizada. O resultado dos comandos está ilustrado na Figura 3.20.

docker service update --image nginx:1.13.5 WEB

docker service ps -f “desired-state=Running” WEB

Anhanguera

```
[manager1] (local) root@192.168.0.55 ~
$ docker service update --image nginx:1.13.5 WEB
WEB
overall progress: 3 out of 3 tasks
1/3: running
2/3: running
3/3: running
verify: Service converged
[manager1] (local) root@192.168.0.55 ~
$ docker service ps -f "desired-state=Running" WEB
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURR
ENT STATE           ERROR               PORTS
c9zc15td90at        WEB.1               nginx:1.13.5        manager1            Running             Runn
ing 18 seconds ago
qavj9bm8n70r        WEB.2               nginx:1.13.5        manager3            Running             Runn
ing 25 seconds ago
k3tmjqzwyvsg        WEB.3               nginx:1.13.5        worker2             Running             Runn
ing 11 seconds ago
```

Caso quiséssemos desfazer nossa atualização de versão das instâncias do Nginx, podemos utilizar o comando update como parâmetro rollback aplicado sobre o serviço WEB. Com isso, teremos a volta da versão das instâncias do Nginx para a versão anterior. Que pode ser consultado através do mesmo comando ps utilizado no exemplo anterior, conforme a seguir:

docker service update --rollback WEB

docker service ps -f "desired-state=Running" WEB

Caso seja necessário parar algum nó e seus serviços, podemos utilizar o comando update seguido do parâmetro availability drain e o nome do nó que desejamos parar, como em nosso exemplo, o nó denominado como escravo2:

docker node update --availability drain escravo2

Quando executamos esse comando, tanto o nó quantos os serviços que estão em execução serão parados.

Comando executado dentro do(s) nó(s) worker:

Agora vamos ver um dos comandos que são executados dentro dos nós escravos (worker) do ambiente Docker. O comando a seguir pode ser utilizado para adicionar um worker ao nosso cluster (Figura 3.21). Para isso, devemos acessar o nó escravo que queremos adicionar a um cluster e executar o comando a seguir, passando o token de segurança e o IP do nó mestre(manager) seguido por porta, conforme a sintaxe representada no comando:

docker swarm join --token <TOKEN> <IP do mestre:Porta>

```
[node1] (local) root@192.168.0.54 ~
$ docker swarm join --token SWMTKN-1-5hqliqpc6bo389i9mpxdvg435kxbqeorf3hslk156gkud0ulko-8ckf9h0o99nrdv3w
hqv4qwpfl 192.168.0.55:2377
This node joined a swarm as a worker.
```

Anhanguera

Comando executado dentro de cada um dos nós (managers e workers):

Agora vamos ver um dos comandos executados dentro dos nós tanto worker, quanto managers de nosso ambiente Docker. Quando executamos o comando Docker system prune com o parâmetro all dentro de um nó, ele será responsável por apagar/deletar tudo o que foi feito dentro do mesmo. Observamos a seguir sua utilização:

docker system prune --all

Agora que já conhecemos alguns comandos, vamos através da plataforma Play With Docker orquestrar um servidor web Apache em um cluster simples. Primeiramente, você deve estar logado na plataforma de playground do Docker, conforme orientações já apresentadas anteriormente. Uma vez que obteve acesso à plataforma, você deverá:

1. Criar um cluster com 3 nós, que serão suficientes para analisarmos nosso cluster sem comprometer a usabilidade da plataforma de testes do Docker. Sendo assim, você deve adicionar 3 nós, que farão parte do cluster, através do botão Add new instance.
2. No nó que você deseja que seja o mestre, digite o seguinte comando:

docker swarm init --advertise-addr <endereço IP desse nó>

Este comando define o nó como manager do cluster. Repare que, ao executá-lo, é apresentada uma saída com a mensagem: “Para adicionar um worker ao swarm, execute o seguinte comando”, conforme pode ser visto (em inglês) na Figura 3.22. Sendo assim, é necessário copiar a saída apresentada a você (que vai ser diferente da destacada em amarelo na Figura 3.22), pois esse comando deverá ser executado em cada um dos demais nós do cluster, adicionando-os como workers desse cluster

Anhanguera

```
[node1] (local) root@192.168.0.18 ~
$ docker swarm init --advertise-addr 192.168.0.18
Swarm initialized: current node (1v3ezy8z8ydd1jd1j8kcdjk8v) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-1eu6jq2a7j5w76ulzfevphb207kucl3nymzkgm317n4a3ohvqc-eexepp3i9erwvr
vjsdvk6tq52 192.168.0.18:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

3. Agora que os nós estão criados e seus papéis definidos, para criar o serviço que estará rodando (de maneira distribuída, replicada) do servidor web Apache, digite o seguinte comando no nó mestre:

docker service create --name WEB –publish 80:80 --replicas=5 httpd

Esse comando cria 5 instâncias de um servidor web Apache, que responderá na porta mapeada (80, nesse caso) e, para facilitar sua monitoração, demos um nome “amigável” a este serviço: WEB.

4. Para saber em quais nós as 5 réplicas desse serviço estão sendo executadas, digite o seguinte comando:

docker service ps WEB

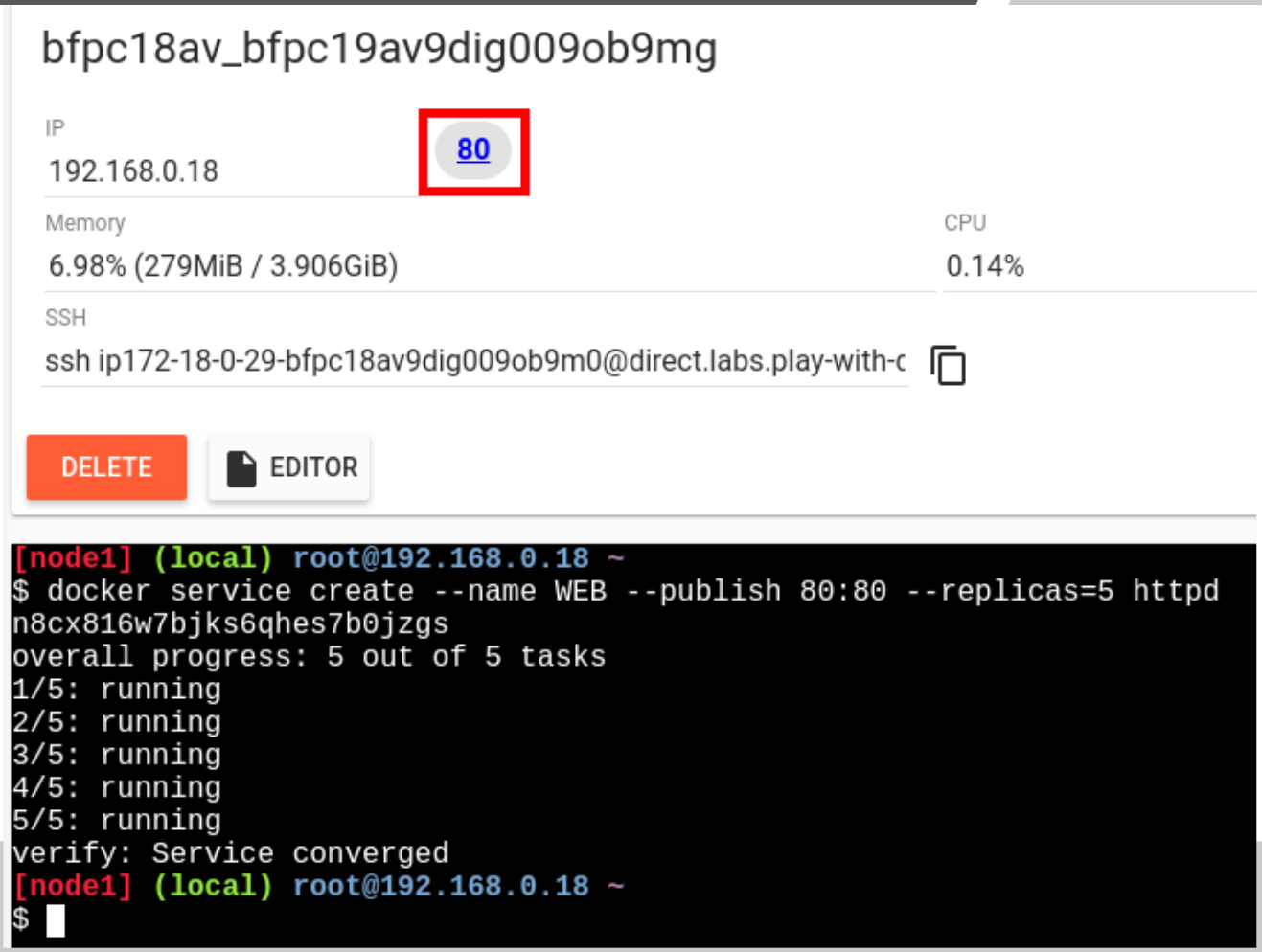
No caso da Figura 3.23 abaixo, podemos ver que 2 instâncias estão rodando no nó 1, e as outras 3 estão rodando no nó 2.

Anhanguera

```
[node1] (local) root@192.168.0.18 ~
$ docker service ps WEB
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE           ERROR
     PORTS
hy4vt3bspf9t        WEB.1               httpd:latest        node1               Running             Running 7 minutes ago

34f60ulp06x5        WEB.2               httpd:latest        node2               Running             Running 7 minutes ago

umf1me3zqdsx        WEB.3               httpd:latest        node1               Running             Running 7 minutes ago

olxz02bp40nh        WEB.4               httpd:latest        node2               Running             Running 7 minutes ago

w49ap6ndeosb        WEB.5               httpd:latest        node2               Running             Running 7 minutes ago
```

5. Por fim, precisamos acessar a página de boas-vindas desse servidor Apache através do(s) endereço(s) IPv4 de cada nó em que esse serviço web estiver rodando. Reparou que, ao criar o serviço, a porta que você mapeou aparece na parte superior, como um hyperlink?
Caso não tenha percebido, veja a porta 80, destacada em vermelho na Figura 3.24. Para acessar a página de boas-vindas, basta clicar nessa porta, em cada um dos nós onde esse serviço está rodando (no caso, nós 1 e 2 do cluster), para vermos a famosa mensagem “It works!” do Apache.

Anhanguera



Esta sequência de comandos que utilizamos para orquestrar um servidor web Apache em 3 nós é a configuração utilizada em sistemas distribuídos, para que os acessos a um website sejam balanceados e, caso ocorra algum problema em um dos nós que mantêm a aplicação, o outro nó assume a execução.

Com o Docker existe uma série de comandos que devemos utilizar para realizar determinadas tarefas dentro do cluster, por exemplo, criação de nós e definição de papéis de nós, como o nó mestre (manager) e o nó escravo (worker). Com isso, devemos estar atentos à sintaxe na utilização correta dos comandos.

De acordo com o texto, sabemos que há um comando para criação de nós em um cluster. Assinale a alternativa que corresponde à sintaxe correta deste comando para a criação de um nó chamado “mestre”:

a) docker-cluster create --driver virtualbox mestre

b) docker-machine create --driver virtualbox mestre

c) docker-machine create node --driver virtualbox mestre

d) docker-machine new --driver virtualbox mestre

e) docker cluster create --node mestre

Com o Docker existe uma série de comandos que devemos utilizar para realizar determinadas tarefas dentro do cluster, por exemplo: acessar nós via nome. Com isso, devemos estar atentos à sintaxe na utilização correta dos comandos.

De acordo com o texto sabemos, que há um comando para acesso de nó via nome. Assinale a alternativa que corresponde à sintaxe correta deste comando para acessar um nó chamado “mestre”:

a) docker ssh mestre

b) docker-machine acess mestre

c) docker-machine into mestre

d) docker-machine ssh mestre

e) docker-machine create mestre