Relatório 17 - Prática: Docker e Containers para Aplicações (III)

Lucas Augusto Nunes de Barros

Descrição das atividades

Na primeira etapa do curso são abordados tópicos básicos como instalação e introdução à sintaxe do Docker, além disso o autor traz conhecimentos teóricos e práticos sobre containers, imagens, volumes e etc, bem como explica os tipos de objetos do Docker e seu funcionamento. Executando o comando *docker –help* no terminal, é trazida uma lista de informações sobre o Docker e sua versão instalada.

```
Common Commands:
  run         Create and run a new container from an image
  exec        Execute a command in a running container
  ps          List containers
  build       Build an image from a Dockerfile
  pull        Download an image from a registry
  push        Upload an image to a registry
  images      List images
  login       Authenticate to a registry
  logout      Log out from a registry
  search      Search Docker Hub for images
  version     Show the Docker version information
  info        Display system-wide information
```

```
Commands:
  attach      Attach local standard input, output, and error streams to a running container
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  import      Import the contents from a tarball to create a filesystem image
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  rmi         Remove one or more images
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  wait        Block until one or more containers stop, then print their exit codes
```

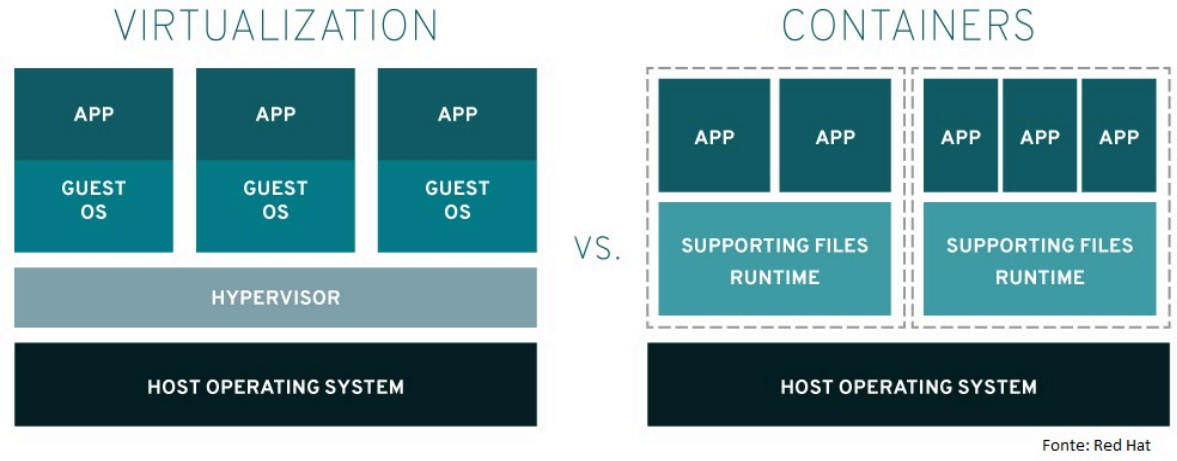
Nesse ponto vale ressaltar que durante a elaboração deste relatório a versão do Docker utilizada foi a 28.0.0. Verificar a opção *–help* do docker pode ser útil para verificar possíveis incompatibilidades. O formato de comando padrão é:

$ docker [OPÇÕES] [COMANDO] [ARGUMENTOS]

De forma simplista, um container é um sistema de arquivos isolado do sistema nativo. O container pode conter todo um sistema operacional ou apenas pacotes necessários para uma determinada aplicação, essa é uma das razões de sua popularidade, além de ser um sistema a parte da instalação nativa (podendo ser excluído e recriado sem maiores problemas) os containers são altamentes escalonáveis, funcionando bem para pequenas e grandes aplicações.

Devido a sua natureza escalonável, é altamente empregado em ambientes de desenvolvimento e de produção, sendo uma das opções mais comuns para disponibilização de serviços, seja de forma isolada ou fazendo parte de um ecossistema.

Abaixo um esquema para visualizar a diferença entre virtualização e containers. Enquanto na virtualização existe toda uma parte de *boot* envolvida, gerando um custo computacional maior, para usar containers o preço pago é o mesmo de carregar um processo na memória RAM, sendo bem menos custoso em termos computacionais.



Fonte: Red Hat

Alguns comandos docker utilizados no card11 (Apache Airflow) .

docker ps	Lista containers em execução.
docker ps -a	Lista todos os containers.
docker build -t airflow-basic	Constrói uma imagem personalizada.

sudo docker exec -it id_container /bin/bash	Executa o prompt do container Opções: ***-i*** ou ***--interactive*** *: permite interagir com processos dentro do container* ***-t*** ou ***--tty*** *: aloca um terminal para o container*

Comandos utilizados para manipulação de containers.

docker container stop <container>	Para o container
docker container start <container>	Inicializa o container
docker container cp <container>	Copia um container
docker container rm <container>	Remove um container
docker container attach <container>	Anexa-se ao container
docker container inspect <container>	Inspeciona containers
docker container rename <nome_antigo> <nome_novo>	Renomeia o container

Uma imagem docker é um arquivo independente que serve como modelo para criar um container. Essa imagem inclui todas as dependências, bibliotecas e arquivos necessários para a execução. Sua natureza portátil permite a implantação da mesma imagem em múltiplos ambientes, lembrando a distribuição de um arquivo executável.

Um container docker é um ambiente com todos os componentes necessários, sem precisar usar dependências da máquina *host*. Esse ambiente é executado em um servidor, físico ou em nuvem. Vários containers podem coexistir no mesmo servidor, fornecendo serviços independentes, esse ecossistema é gerenciado pelo orquestrador de containers, como o docker compose.

Uma das enormes vantagens dos containers é que eles empacotam todo *software* para execução em qualquer sistema operacional. Se antes era necessário empacotar *softwares* específicos para diferentes sistemas, os containers criam uma camada que faz a adaptação para o sistema *host*.

O compartilhamento de arquivos entre *host* e container é realizado através de volumes, que no contexto do docker funcionam como um sistema de arquivos compartilhados. Podendo ser utilizados para salvar estados de containers e também para compartilhar e sincronizar arquivos entre *host* e containers. Durante a criação do

container, um diretório do *host* é mapeado para dentro do container, de maneira que quaisquer alterações nesse diretório sejam sincronizadas.

Para definir um volume é utilizada a opção -v , que define a pasta do diretório *host* que será mapeada para dentro do container, sempre seguindo a seguinte sintaxe.

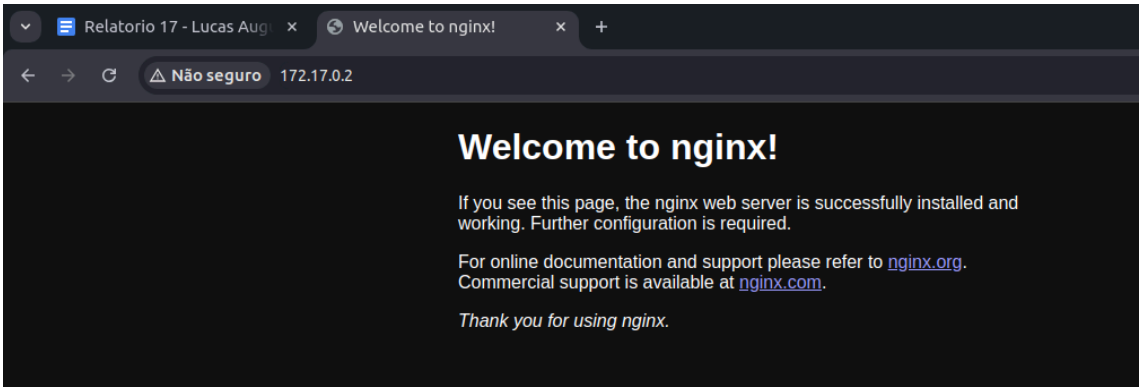
-v /diretorio/do/host/:/diretorio/do/container

Exemplo:

docker container run -v /home/augusto/bkp/:/bkp --name volume-teste -it alpine

É importante tomar cuidado ao utilizar volumes pois ao criar um diretório mapeado dentro do container, todas as alterações feitas através do container serão sincronizadas com o diretório do *host.*

Para disponibilizar serviços é preciso realizar também o mapeamento de portas, para então fazer o *deploy* das aplicações. Cada container é conectado a uma rede *bridge* (ponte) que o computador *host* tem acesso. Criando um container com a imagem do servidor Nginx é possível acessá-lo através da rede *bridge.*



E fazendo o comando:

docker container run -d -p 80:80 --name nginx nginx

É possível acessar o container através da porta 80 do *host,* fazendo assim o mapeamento de portas, que será usado no futuro para disponibilizar serviços em containers.

Uma função muito útil do docker é a de gerar imagens a partir de um container. Tornando possível criar um container, personalizá-lo da forma necessária, e então gerar uma imagem que torna possível outras pessoas executarem a aplicação, independente do sistema operacional em que a imagem vá ser montada como container. O processo de criação da imagem ainda pode ser acrescido de uma etapa de compactação, tornando a imagem docker um arquivo .tar.

A criação de uma imagem segue a sintaxe:

docker container commit [OPÇÕES] ID_CONTAINER IMAGEM[:TAG]

Um exemplo prático:

docker container commit nginx-allumy nginx-allumy-img

Cria a imagem *nginx-allumy-img* a partir do container de nome *nginx-alumy*.

Comando utilizados para manipular imagens e arquivos .tar

docker commit <container> <imagem>:<v1>	Cria uma imagem a partir do container
docker save -o <imagem.tar> <imagem>:<version>	Exporta uma imagem docker para um arquivo .tar
docker load -i <imagem.tar>	Carrega a imagem docker criado com *docker save*
docker export -o <imagem.tar> <container>	Exporta o sistema de arquivos de um container para um arquivo .tar
docker import <imagem.tar> <imagem-importada>	Cria uma imagem a partir de um arquivo .tar
docker build -t <imagem> .	Constrói uma imagem a partir de um *Dockerfile*
docker tag <imagem>:<version> <repositorio>:<v2>	Atribui uma nova tag a uma imagem existente
docker push <repositorio/imagem>:<v1>	Envia a imagem para um *registry,* seja o *Docker Hub* ou outro.
docker pull <imagem>:<version>	Faz o *download* de uma imagem docker.
docker <objeto> inspect <nome_objeto>	Exibe informações detalhadas sobre o objeto

Registry, no contexto do docker é como um GitHub das imagens, onde é possível criar novas e baixar imagens do repositório, facilitando o compartilhamento e o versionamento.

O autor traz também uma aula sobre *Dockerfile*, que é a melhor forma de criar imagens docker, pois garante reprodução simétrica em qualquer ambiente, além de otimizar o tamanho e facilitar o compartilhamento, torna o versionamento mais simples e evita problemas de inconsistência.

Em uma das novas versões do docker já é possível criar um volume e mapeá-lo antes de criar o container.

```
Usage:  docker volume COMMAND

Manage volumes

Commands:
  create      Create a volume
  inspect     Display detailed information on one or more volumes
  ls          List volumes
  prune       Remove unused local volumes
  rm          Remove one or more volumes
  update      Update a volume (cluster volumes only)

Run 'docker volume COMMAND --help' for more information on a command.
augusto@terminator:~$
```

Por fim são feitas algumas práticas sobre criação e importação de imagens utilizando arquivos .tar.

Criando o container com a imagem Ubuntu, esse container teve o nginx instalado para então gerar uma imagem a partir deste container.

```
augusto@terminator:~/Documentos/GitHub/LAMIA/bootcamp/card17$ sudo docker ps -a
CONTAINER ID   IMAGE                     COMMAND                  CREATED          STATUS                      PORTS     NAMES
4899e5a91e1a   ubuntu                    "/bin/bash"              12 minutes ago   Exited (127) 9 minutes ago            ubuntu-bootcamp
13f2ade84769   nginx                     "/docker-entrypoint.…"   4 hours ago      Created                               nginx
a7150b2cc445   airflow-section-4-airflow "/entrypoint.sh webs…"   7 days ago       Exited (0) 4 hours ago                airflow
cc6b82f2acc8   airflow-basic             "/entrypoint.sh"         9 days ago       Created                               lucid_wing
augusto@terminator:~/Documentos/GitHub/LAMIA/bootcamp/card17$
```

Imagem *ubuntu-bootcamp* criada a partir do container

```
augusto@terminator:~/Documentos/GitHub/LAMIA/bootcamp/card17$ sudo docker images
REPOSITORY                    TAG       IMAGE ID       CREATED          SIZE
ubuntu-bootcamp               latest    b790ad834d0d   11 minutes ago   78.1MB
airflow-section-4-airflow     latest    6ff6783c8fc0   7 days ago       1.69GB
<none>                        <none>    6ac23f32cecd   7 days ago       1.69GB
```

Arquivo .tar criado a partir da imagem docker.

```
augusto@terminator:~/Documentos/GitHub/LAMIA/bootcamp/card17$ ls
 Dockerfile  'Relatorio 17 - Lucas Augusto.pdf'   ubuntu-bootcamp.tar
```

Conclusão

O Docker revolucionou a forma como aplicações e serviços são disponibilizados, proporcionando isolamento, portabilidade e eficiência através de containers. Conhecer suas tecnologias é fundamental para compor qualquer equipe de desenvolvimento.

Referências

[1] card17 - Prática: Docker e Containers para Aplicações (III)

[2] AWS.AMAZON.COM. Qual é a diferença entre imagens e contêineres do Docker?

Disponível em: <https://aws.amazon.com/pt/compare/the-difference-between-docker-images-and-containers/#:~:text=Uma%20imagem%20do%20Docker%2C%20ou,cont%C3%AAiner%20precisa%20para%20ser%20executado.> Acesso em: 25 de mar. de 2025