UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO

**Roteiro de Aula Prática – Criação de Imagens e Containers Docker**

DISCIPLINA: DCA0132 – ENGENHARIA DE DADOS

PROFESSOR: Carlos Manuel Dias Viegas

Esta prática consiste em criar arquivos Dockerfile para a construção de imagens e posterior execução em containers Docker. Além disso, também deverá ser criado um arquivo Docker Compose para a execução.

O objetivo desta prática é criar duas imagens que serão executadas em containers diferentes, sendo uma cliente e outra servidora. O cliente enviará uma mensagem para o servidor e este exibirá em tela. Em seguida, após sucesso na criação e execução dos containers, as imagens serão enviadas para um repositório no DockerHub.

Ao final da tarefa, devem ser submetidos no SIGAA, em tarefa específica, o link para o repositório no DockerHub (contendo as imagens) e o arquivo docker-compose.yaml criado. A prática pode ser realizada em duplas.

**Tarefas**

1. [Criando a aplicação] Baixar os scripts clienteTCP.py e servidorTCP.py:

<https://www.dca.ufrn.br/~viegas/disciplinas/DCA0132/files/Sockets/clienteTCP.py>

<https://www.dca.ufrn.br/~viegas/disciplinas/DCA0132/files/Sockets/servidorTCP.py>

Estes scripts estão programados para que o cliente estabeleça uma conexão via socket TCP com o servidor e envie para o mesmo o seu hostname e endereço IP. Os códigos fonte de ambos não devem ser modificados. O cliente, ao ser executado, receberá por parâmetro (no comando de execução) o IP do servidor para que possa se conectar ao mesmo.

1. [Criando as imagens] Criar os arquivos Dockerfile contendo as informações necessárias para a criação das imagens. Caso deseje, pode fazer modificações no Dockerfile.

Arquivo 1: Dockerfile.client

|  |
| --- |
| FROM python:3-slim  WORKDIR /app  COPY clienteTCP.py /app  ENTRYPOINT ["python","clienteTCP.py"] |

Arquivo 2: Dockerfile.server

|  |
| --- |
| FROM python:3-slim  WORKDIR /app  COPY servidorTCP.py /app  CMD ["python","servidorTCP.py"] |

1. [Criando as imagens] Construir as imagens a partir dos arquivos Dockerfile.

Imagem cliente:

docker build -f Dockerfile.client -t <usuario-dockerhub>/engdados:clientetcp .

Imagem servidor:

docker build -f Dockerfile.server -t <usuario-dockerhub>/engdados:servidortcp .

PS: O ponto ao final indica a atual pasta em que estão os arquivos a serem utilizados na construção da imagem. Porém, é possível especificar o caminho para outra pasta, caso necessário.

Legenda:

|  |
| --- |
| Comando padrão para construção de imagem a partir de um Dockerfile:  docker build [opções] <tag-da-imagem> <pasta-local>  O campo opções permite uma série de parâmetros, tais como:  -f : para especificar qual o Dockerfile de entrada para a construção da imagem  -t : para definir a tag da imagem  Referência para consulta:  <https://docs.docker.com/engine/reference/commandline/build/> |

1. [Executando os containers] Executar o container servidor:

docker run --name servidor -it <usuario-dockerhub>/engdados:servidortcp

1. [Executando os containers] Executar o container cliente:

docker run --name cliente --restart always <usuario-dockerhub>/engdados:clientetcp <ip-do-servidor>

PS: O container cliente deve ser autossuficiente, de maneira que não é necessário acessar o terminal do mesmo para executar o script cliente. Basta apenas executar o container e passar o IP do servidor como parâmetro e aguardar o resultado na tela do servidor.

Legenda:

|  |
| --- |
| Comando padrão para execução de um container:  docker run [opções] <imagem-criada:tag> [parâmetros-se-necessário]  O campo opções permite uma série de parâmetros, tais como:  -it : para acesso direto ao terminal (interativo)  --name : para definir um nome ao container que será executado  --restart always : o container sempre será reiniciado após o término da execução  Referência para consulta:  <https://docs.docker.com/engine/reference/commandline/run/> |

É possível utilizar o comando docker logs <nome-do-container> para verificar o que foi enviado/recebido por cada um.

1. [Envio para o Dockerhub] Após sucesso na execução de ambos os containers, no qual o cliente enviou a mensagem para o servidor, deve-se enviar as imagens para o DockerHub (<https://hub.docker.com/>).

Por meio de um navegador web, acessar o DockerHub e criar um repositório público cujo nome deve ser engdados.

Em um terminal, fazer login no Dockerhub por meio do comando:

docker login

Após o login realizado, deve-se enviar as imagens.

* Enviar a imagem do cliente:

docker push <usuario-dockerhub>/engdados:clientetcp

* Enviar a imagem do servidor:

docker push <usuario-dockerhub>/enbgdados:servidortcp

Legenda:

|  |
| --- |
| Comando padrão para envio de imagens:  docker push <nome-do-usuario-dockerhub>/<tag-da-imagem-criada>  Referência para consulta:  <https://docs.docker.com/engine/reference/commandline/push/>  <https://docs.docker.com/docker-hub/> |

1. [Docker Compose] Criar um arquivo YAML (docker-compose.yml) que execute 3 aplicações, sendo um servidor e dois clientes. Cada cliente irá enviar mensagens para o servidor. O objetivo é que as 2 aplicações clientes enviem mensagens para o servidor e este as exiba em tela.

**Importante**: Não é pretendido que o docker-compose.yml faça o build da imagem. Deseja-se apenas executar containers baseados nas imagens previamente criadas e carregadas no Dockerhub.

A seguir segue um exemplo de docker-compose.yml como base:

|  |
| --- |
| # Serviços que serão executados nos containers  services:  # Nome do serviço a ser criado  servidor:  # Nome do container que será executado  container\_name: <nome-do-container>  # Imagem que será executada no container  # A imagem é obtida localmente ou a partir do Dockerhub  # image: <nome-do-usuario-dockerhub>/<nome-do-repositorio:tag>  image: <usuario-dockerhub>/enbgdados:servidortcp  # Permite a exibição de texto na tela do terminal  tty: true  ... (complete com o que achar relevante e importante)  # Nome do segundo serviço a ser criado  cliente:  container\_name: <nome-do-container>  # O depends\_on faz com que este serviço seja executado após outro  depends\_on:  - servidor  image: <usuario-dockerhub>/engdados:clientetcp  tty: true  # Quando o container terminar a sua execução, sempre irá reiniciar  restart: always  ... (complete com o que achar relevante e importante)  # Nome do terceiro serviço a ser criado  cliente2:  container\_name: <nome-do-container>  depends\_on:  - servidor  image: <usuario-dockerhub>/engdados:clientetcp  tty: true  restart: always  ... (complete com o que achar relevante e importante) |

1. No docker-compose.yml também deve ser definida uma rede com a faixa de IP específica na rede 172.20.0.0/24. Como sugestão, segue parte da especificação dessa rede:

|  |
| --- |
| networks:  minha-rede:  driver: bridge  ipam:  driver: default  config:  - subnet: 172.20.0.0/24 |

Em cada serviço deverá ser indicado qual rede o mesmo pertence. No caso acima, deve ser indicado para a minha-rede. É importante destacar que as configurações de networks não devem estar aninhadas em services (ou seja, devem estar sem indentação inicial). Consulte a documentação do Docker Compose para mais detalhes: <https://docs.docker.com/compose/> e <https://docs.docker.com/reference/compose-file/networks/>

Para executar os containers por meio do Docker Compose, basta executar no terminal:

docker compose up

Ou ainda, executar um serviço de forma específica:

docker compose up <nome-do-serviço>