Nivelamento & Docker

cfmcc@ufop.edu.br
Prof. Dr. Carlos Frederico MC Cavalcanti, 05/05/2023
versão 2024 @ 5/04/2024

Introdução

Bem vindo à disciplina de Sistemas Operacionais. Usaremos o conteúdo de um livro consagrado no mundo todo, Sistemas Operacionais Modernos, de Andrew Tanenbaum. Tanenbaum é um vetereno no desenvolvimento de sistemas operacionais assim como outro ícone, Abraham SIlberschatz, sendo este também escritor de livros de banco de dados.

Sistemas operacionais fazem a mágica da computação acontecer para o usuário, ofertando uma camada onde programas são executados. Geralmente não fazemos programas para rodar em "hardware" (tipo programa para rodar em i5 ou em um ARM com tanto de memória e disco de tanto) mas para rodar em Windows, Linux, MacOS ou uma variante desses, como Android ou IOS. Então Sistemas Operacionais provê um ambiente onde programas (para aquele SO) são executados, gerencia e abstrai o hardware da máquina.

Se olharmos este conceito, podemos observar que um navegador atual, provê um ambiente onde programas "turing complete" são executados. Observe que linguagens de marcação, como html e xml não são "turing complete" mas os veteranos PHP (criada em 1994) e Javascript (criado em 1993 e 1995) são, assim como as linguagens denominadas "de programação". Neste olhar, um browser, que é executado em cima de um SO, se comporta como um SO e acrescenta mais uma camada de abstração que, na prática, implementa uma máquina virtual. Em engenharia de software, considerando as aplicações WEB atuais, o "front end" é executado nos browsers. Esta questão de que programas são executados em *browsers* estão sendo levados cada dia mais a sério e por isto foi idealizado WebAssembly ou WASM, como é também conhecido, que define uma máquina virtual e suas instruções. WASM é implementado na maioria dos browsers usados atualmente e foi idealizado para ser usado também no "back-end". Veja mais neste simples tutorial WASM.

No capítulo 1 e 2, aprenderemos a importância da programação concorrente e paralela. as coisas não são tão simples como parece, não é verdade?

Quero lhe apresentar Docker. Existe um vídeo e um livro no Moodle., confira lá. Ele é importante porque praticamente TODOS os nossos trabalhos serão feitos usando docker e eu irei simplesmente "abaixar" uma imagem no docker hub.

Docker

É uma ótima opção para aumentar a segurança de sua aplicação, para dar agilidade no desenvolvimento e *deploy* de sua aplicação é o Docker. Docker em Linux/Ubuntu é bem fácil. use:

```
curl -fsSL https://get.docker.com/ | sh
```

Docker permite "contenizar" uma aplicação isolando-a do resto do ambiente computacional. Docker é a ferramenta ideal para a implementação de microsserviços, que é o paradigma do momento, juntamente com kubernetes e integração e entrega contínua, conhecido como CI/CD, *continuous integration/continuous delivery*. CI/CD é um método para entregar aplicações com frequência aos clientes (procure jenkins, que é a ferramenta mais popular para implementar CI/CD, no Google e Ansible.) O fato é que existe um bom conjunto de ferramentas que são usadas hoje com funcionamento similar para desenvolvimento de web applications modernas. Tudo isto afeta, inegavelmente, a segurança de uma aplicação.

Com o comando `curl` executado, veja se o docker está rodando:

```
#/etc/init.d/docker status
```

ou usando o seguinte comando `#service docker status` (tente usar `#service --status-all`)

e inicialize o docker `# service docker start`

rode sua primeira aplicação docker `# docker run hello-world`

e inicialize o docker uma instância do docker ubuntu `# docker run -it ubuntu`

veja que o prompt mudou.. Você está dentro do ubuntu, Se você executar `# docker images` observe que o comando não será encontrado.

Saia do container emitindo Control Z (^Z)

digite: `# docker ps -a` e você listará todas as instâncias das imagens, que são os contêineres, que estão executando ou que estão parados mas não finalizados.

Vá em https://www.kali.org/docs/containers/official-kalilinux-docker-images/

e entenda qual a imagem mais adequada para pulling. Em tempo de escrita deste texto, vou escolher a `kali-bleeding-edge`, assim

```
#docker pull kalilinux/kali-bleeding-edge
```

a imagem está lá:

```
root@ubuntu-s-1vcpu-1gb-nyc3-01:~# docker images
REPOSITORY                          TAG       IMAGE ID       CREATED         SIZE
kalilinux/kali-bleeding-edge        latest    4fdea723a015   6 days ago      126MB
ubuntu                              latest    d2e4e1f51132   3 weeks ago     77.8MB
hello-world                         latest    feb5d9fea6a5   8 months ago    13.3kB
```

Para sair no docker e deixá-lo rodando, mantenha pressionado tecla CTRL + tecla p + tecla q (ao mesmo tempo) dentro do prompt do shell

```
root@ubuntu-s-1vcpu-1gb-nyc3-01:~# docker run -it kalilinux/kali-bleeding-edge
┌──(root㉿948dbfccbbc8)-[/]
└─#

┌──(root㉿948dbfccbbc8)-[/]
└─#

┌──(root㉿948dbfccbbc8)-[/]
└─# root@ubuntu-s-1vcpu-1gb-nyc3-01:~# docker ps -a
CONTAINER ID   IMAGE                          COMMAND     CREATED          STATUS                          PORTS
 NAMES
948dbfccbbc8   kalilinux/kali-bleeding-edge   "bash"      21 seconds ago   Up 20 seconds
 beautiful_hofstadter
b3fbcbdb84f6   ubuntu                         "bash"      30 minutes ago   Exited (0) 30 minutes ago
 festive_banzai
8731d68bbab3   ubuntu                         "bash"      2 hours ago      Exited (130) 32 minutes ago
 reverent_poincare
5e9a89e0f00a   hello-world                    "/hello"    2 hours ago      Exited (0) 2 hours ago
 flamboyant_rosalind
root@ubuntu-s-1vcpu-1gb-nyc3-01:~#
```

Para voltar ao shell dentro do container… `docker attach <CONTAINER ID>`

No caso acima: `# docker attach 948dbfccbbc8`

Para visualizar o consumo computacional do container: `docker stats «CONTAINER ID>`

No caso acima: `# docker stats 948dbfccbbc8`

Experimente:

```
docker logs <CONTAINER ID>
docker top  <CONTAINER ID>>
```

Removendo um container: `docker rm -f <CONTAINER ID>`

Até agora estamos usando a imagem de terceiros e agora é o momento de construir a nossa imagem. Parte destes exemplos estão no livro “Descomplicando o Docker” de Jefferon Vitalino e Marcus Castro que deixei disponibilizado no Moodle e outros de outros locais e da minha experiência. Vamos lá…

A ideia é esta, construir uma imagem, em nosso caso, o programa que solicitei para baixar a partir da

sua conta dockerhub, Dockerhub é um repositório de imagens e a minha conta é cfred. **Abra a sua conta no dockerhub, se ainda não tiver**. Se você instalar o Docker no Windows (eu instalei tanto no Linux quanto no Windows), baixe o Docker Desktop for Windows .

Agora, vamos "construir" ("buildar", no jargão) uma imagem.

Na raiz, eu abri um diretório chamado cfredDockerfiles e dentro dele abri outro, chamado apache. Dentro deste diretório, abri o editor de texto e copiei o arquivo abaixo e salvei como Dockerfile

```
FROM debian
RUN apt-get update && apt-get install -y apache2 && apt-get clean
RUN apt-get install net-tools
ENV APACHE_LOCK_DIR="/var/lock"
ENV APACHE_PID_FILE="/var/run/apache2.pid"
ENV APACHE_RUN_USER="www-data"
ENV APACHE_RUN_GROUP="www-data"
ENV APACHE_LOG_DIR="/var/log/apache2"
LABEL description="Webserver"
VOLUME /var/www/html/
EXPOSE 80
```

```
root@ubuntu-s-1vcpu-1gb-nyc3-01:~/cfredDockerfiles/apache# ls -la
total 12
drwxr-xr-x 2 root root 4096 May 22 17:21 .
drwxr-xr-x 3 root root 4096 May 22 17:15 ..
-rw-r--r-- 1 root root  316 May 22 17:21 Dockerfile
root@ubuntu-s-1vcpu-1gb-nyc3-01:~/cfredDockerfiles/apache# cat Dockerfile
FROM debian
RUN apt-get update && apt-get install -y apache2 && apt-get clean
ENV APACHE_LOCK_DIR="/var/lock"
ENV APACHE_PID_FILE="/var/run/apache2.pid"
ENV APACHE_RUN_USER="www-data"
ENV APACHE_RUN_GROUP="www-data"
ENV APACHE_LOG_DIR="/var/log/apache2"
LABEL description="Webserver"
VOLUME /var/www/html/
EXPOSE 80
root@ubuntu-s-1vcpu-1gb-nyc3-01:~/cfredDockerfiles/apache#
```

"Buildei" esta image criando um container que nomeei como cfred/apache versão bcc264 versao 0, usando o comando

`#docker build -t cfred/apache:bcc264.0 .` estando no diretório onde está o arquivo Dockerfile (veja o "ponto" no final do comando).

o retorno `# Successfully built 9a78bc1c27bc`

```
Successfully built 9a78bc1c27bc
Successfully tagged cfred/apache:bcc264.0
root@ubuntu-s-1vcpu-1gb-nyc3-01:~/cfredDockerfiles/apache# docker ps -a
CONTAINER ID   IMAGE                              COMMAND    CREATED             STATUS
TS     NAMES
10b629bf118e   debian                             "bash"     10 minutes ago      Created
       frosty_goldberg
948dbfccbbc8   kalilinux/kali-bleeding-edge       "bash"     36 minutes ago      Up 36 minutes
       beautiful_hofstadter
b3fbcbdb84f6   ubuntu                             "bash"     About an hour ago   Exited (0) About an hour ago
       festive_banzai
8731d68bbab3   ubuntu                             "bash"     2 hours ago         Exited (130) About an hour ago
       reverent_poincare
5e9a89e0f00a   hello-world                        "/hello"   2 hours ago         Exited (0) 2 hours ago
       flamboyant_rosalind
root@ubuntu-s-1vcpu-1gb-nyc3-01:~/cfredDockerfiles/apache# docker images -a
REPOSITORY                    TAG         IMAGE ID       CREATED         SIZE
<none>                        <none>      7d717afde2d0   8 minutes ago   252MB
cfred/apache                  bcc264.0    9a78bc1c27bc   8 minutes ago   252MB
```

executo `# docker run -it  cfred/apache:bcc264.0`

e dentro do container:

```
/etc/init.d/apache2 start
/usr/sbin/apache2 -k start
ss -atn
ifconfig
```

IMPORTANTE: se fosse tivesse que "buildar" uma imagem para disponibilizar para seu professor, ele só digitaria isto:

```
# docker run -it  cfred/apache:bcc264.0
```

Sendo que o `cfred` é o username do usuário da conta no dockerhub e `apache:bcc264.0` é o nome do arquivo (imagem, para ser mais exato) e seu tag (versionamento)

IMPORTANTE-> No caso seria o username do aluno, o nome do TP (neste caso foi `apache` e o `bcc264 ou bcc423` seria substituído por `bcc264_19_1_12345:0`, sendo bcc264 o código da disciplina e `19_1_1234` o número do RA do aluno na UFOP e `0` o versionamento)

Você pode também fazer um `commit` e não usar o dockerfile.

Para "tagar" a imagem: `docker tag IMAGEID cfred/apache:bcc264.0`

O básico está neste texto . Há dois vídeos explicando sobre docker no Moodle feito por mim e muito material. Mas gostaria de deixar este tutorial (tem vários) além da documentação oficial.

https://blog.geekhunter.com.br/docker-na-pratica-como-construir-uma-aplicacao/

TP1 DOCKER e Linha de comando

Este é o primeiro TP de SO de 2023-1.

Todo SO tem o conceito de processo. Em Unix/Linux, temos os seguintes comandos:

1. `top`
   Pare verificar o uso de processos do sistema e constatar quais deles estão consumindo mais memória ou processamento. Note que os primeiros itens da lista são os que mais consomem recursos do computador. Para cancelar a execução e voltar à linha de comando, basta pressionar a tecla Q ou a combinação Ctrl+C.

2. `ps`
   O comando ps lista os processos em execução no sistema. Porém, diferentemente do top, ele não traz informações sobre o quanto de processamento ou de memória ele está consumindo. Apesar disso, o ps é uma maneira bem mais ágil de consultar o PID de um processo, principalmente ao ser usado em conjunto com o `grep`.

3. pstree
   Também há uma forma de visualizar os processos em forma de árvore, tornando mais visível as relações entre eles. Para isso, basta usar o comando pstree.

4. `kill`
   Se um software travou ou precisa ser interrompido de qualquer forma, o kill é a solução. Basta executar o comando seguido do PID do processo para que a aplicação "morra". Se mesmo depois disso você perceber que o processo ainda existe, tente acrescentar a opção -9 ao comando: kill -9 PID. Assim você força o processo a ser interrompido a qualquer custo.

   Se quiser matar mais de um processo ao mesmo tempo, basta listar os PIDs separando-os com um espaço, logo depois do comando kill. Exemplo: kill 3657 6785 3456

5. `killall e pkill`
   Caso prefira, você também pode matar de uma vez só todos os comandos selecionado ao nome de um programa. Para isso, basta usar o comando killall seguido do nome do software em questão, como killall vim.

Porém, o killall exige uma certa rigidez ao informar o nome do processo. Caso você não tenha certeza do nome completo, pode tentar o pkill, que faz diversas associações com a palavra-chave digitada.

6. `renice`
Todos os processos do Linux possuem prioridades de execução, variando em uma escala que vai de 19 (menos significativa) a -20 (mais significativa). Por padrão, os processos executados por um usuário ganham a prioridade 0, mas por meio do comando renice é possível alterar esse valor para algum nível entre 0 e 19. Apenas o usuário administrador (root) é capaz de ir além, alterando prioridades de qualquer processo e chegando até o nível máximo de -20.

   Para realizar esse tipo de operação, basta seguir a sintaxe renice novaprioridade -p PID. Se quiséssemos dar mais prioridade a um processo de PID 1516, por exemplo, usaríamos: sudo renice -10 -p 1516. Lembre-se que o sudo exigirá a senha do seu usuário antes de executar o comando.

Vamos objetivamente ao TP1:

1) Instale o Docker conforme explanado. Registre-se no Docker Hub. Lembre-se, o username é onde está a SUA imagem no Docker hub. Por exemplo, o meu é cfred.

2) Escolha uma distribuição Linux qualquer (sugiro o UBUNTU) e gere ("Build") uma imagem com
   a) gcc
   b) vim
   c) grep (se não estiver instalado nativamente na distribuição)
   d) um diretório com seu nome (todas as atividades abaixo devem estar neste diretório - ver item nomeado com "fechando" abaixo)

   Vc pode fazer o "Build" usando "`commit`" ou usando usando um dockerfile

3) Neste diretório com seu nome, gere (um arquivo TXT com o seu nome e turma e coloque o nome dele no formato `SEUNOME.TXT`. Veja que "`SEUNOME`" , para mim, seria CFRED

4) Faça o primeiro programa em C/C++ e coloque no diretório PRG1 onde o programa

simplesmente imprimirá:

a) a versão do sistema operacional (é só imprimirá um arquivo que está no ubuntu, descubra qual é)
b) imprimirá o arquivo TXT com seu nome e turma (veja acima)
c) Executará (via programa C/C++, veja o system), os 6 (na verdade 7) comandos relacionados com processos (veja acima).

5) Faça o SEGUNDO programa em C/C++ e coloque no diretório PRG2 onde o programa simplesmente terá o seguinte comportamento:
   a) você solicitará que o usuário digite o nome do programa para ser executado.
   b) Voce irá usar o FORK para invocar (chamar o programa) que o usuário digitou e
   c) retornará o prompt para que o usuário chame outro programa.
   d) Este programa (via programa C/C++, veja o system e FORK) funcionará parecido com CMD em Windows ou um shell BASH, mas bem primitvo, sem possibilidade de programar e outras coisas;
6) Mostre o que aconteceu (no vídeo) e nos comentários que poderão ser ecoados pelo programa.

7) Deixe tudo DOCUMENTADO nos arquivos e diretório onde está seu nome (veja etapa 3). Eu vou fazer um run -it <SUA IMAGEM> e a sua imagem deve baixar e compilarei e executarei usando a linha de comando que vc me indicou para executar. tipo “gcc …” (veja que se faz necessário fazer os passos abaixo..)
8) Para fechar (finalizar), **faça o commit de tudo**, ‘builde” a sua imagem e mande para o docker hub usando push (**SUBA a magem para o DOCKER HUB)** Sugiro conferir se realmente está tudo certo…(veja abaixo); Vc fazer um commit ou usar um docker file)
9) **Mande a ID da imagem (não a imagem)** que você gerou para eu possa baixar, nos comentários da atividade no Moodle. Por exemplo, eu gerei a imagem `cfred/apache:bcc264.1` se vc digitar `docker run -it cfred/apache:bcc264.1` vc baixará a imagem construída (“builtada”) por mim. Coloque isto nos comentários no Moodle.

```
root@ubuntu-s-1vcpu-1gb-nyc3-01:~/cfred# docker run -it  cfred/apache:bcc264.1
Unable to find image 'cfred/apache:bcc264.1' locally
bcc264.1: Pulling from cfred/apache
67e8aa6c8bbc: Already exists
d639fcb1969b: Already exists
cb8290f61dcc: Already exists
Digest: sha256:ff8c22a9d2f7cd959724f061fa51d67085edb0e1ccc969e3ff0348d92b8d851e
Status: Downloaded newer image for cfred/apache:bcc264.1
```

10) faça um vídeo de, no máximo, 5 minutos explicando o que vc fez. e explique bem o FORK (

muito importante) e coloque no Moodle. ISto é como se fosse falar comigo, mas faça um video.

a) Voce deve aparecer no video e se identificar
b) Não pode passar mais de 5 minutos com 1 min de tolerancia
c) Sugiro vc abrir uma reunião no Google Meet só voce.. GRAVE.. e mande a gravação (que fica no doogle drive) para mim via MOODLE.

Por fim:

1- Individual

2- Vc pode gravar quantas vezes quiser, mas faça perfeito!

3-Observe cada ponto que deve ser observado do trabalho.

4- Se quiser fazer um PDF de apoio, pode anexar ao trabalho, mas é opcional.

5-Tudo no Moodle, NÂO mande por email pois eu não consigo inserir em sua conta.