ANOTAÇÕES

DOCKER ESSENTIALS

**O que é Container?** É uma forma de você isolar recursos para uma determinada finalidade. Num servidor, por exemplo, é possível "dividí-lo" em pedaços colocando recursos em dependências diferentes. Uma aplicação em container não interfere em outras aplicações rodando na mesma máquina.

**Cgroups**: Também chamado de control groups, é um módulo o kernel Linux que é usado para fazer o isolamento de recursos da máquina como por exemplo CPU e memória RAM. Dizendo para cada processo quanto de memória RAM e de CPU estará disponível para executá-lo.

**Namespaces**: Outro recurso do kernel Linux também usado para fazer o isolamento de recursos como redes (network), processos (PID), filesystem (sistema de arquivos) e usuários (users).

**chroot**: O chroot somente altera o diretório raiz do sistema de arquivos Unix para outro diferente e serve também para isolar o filesystem.

**unshare**: O unshare é um utilitário da linha de comando do Linux que permite a um processo “desassociar” seus recursos do restante do sistema. O comando unshare cria novos namespaces (conforme especificado pelas opções da linha de comando) e então executa o que foi especificado.

**CoW - Copy-On-Write**: Uma imagem docker é formada por camadas quando eu quiser (ou precisar) modificar algo na camada primária imediatamente é criada uma cópia dessa camada e o que eu vou modifcar será uma cópia da camada primária, a camada primária será read-only (somente leitura), ou seja, só poderei ler (ou observar) o conteúdo presente lá, mas não posso modificá-lo já a cópia que foi gerada é read-write (leitura e escrita) isso significa que não só posso ler o conteúdo da camada como também posso modificar.

**Instalação do Docker (via CURL)**: É possível instalar utlizando executáveis .deb ou .rpm também o comando para instalar o Docker via CURL é o seguinte: *curl -fsSL https://get.docker.com | bash* **OBS**: No momento estou usando o Fedora 39 e por ser um sistema muito recentemente ainda não tem versão de Docker para ele, estou usando uma alternativa ao Docker que é o Podman mantido pela Red Hat que já vem instalado no sistema, o Podman tem os mesmos comandos do docker, a única diferença é que ao invés de usar a palavra "docker" no início de cada comando se usa a palavra "podman".

**Container rootless**: Um container rootless é um container que para ser executado não precisa ter acesso ao root da máquina onde está executado, ou seja, não precisa de privilégios de administrador.

**Comando para rodar o docker sem root**: *dockerd-rootless-setuptool.sh install*. Talvez apareça uma mensagem de erro, então é necessário executar o seguinte comando *sudo apt install -y uidmap*

## **Comandos Docker**

Abaixo serão listados comandos Docker apresentados no curso

*docker ps*: Comando mais antigo que é usado para mostrar os containers que estão em execução.

*docker container ls*: Comando atual que é usado para mostrar os containers que estão em execução. Uma alternativa ao docker ps apresentando uma sintaxe mais clara.

*docker version:* Comando que verifica a versão do Docker que está instalada no sistema.

*docker container run*: Cria e executa um container a partir de uma imagem, um exemplo disso é o container hello-world

*docker container run hello-world*: O Docker também tem o famoso Hello World, digite esse comando para mostrar o hello world no docker.

*docker container ls -a*: Este comando mostra todas as imagens que estão instaladas na sua máquinas, estando elas imagens em execução ou não.

*docker container run -it:* O -i indica que eu quero ter uma certa interatividade com meu container, já o t é de terminal, ou seja, interatividade de terminal isso significa que terei um terminal TTY para interagir com o container.

*docker container run -it ubuntu*: Comando para baixar a imagem Docker da última versão do Ubuntu

*CTRL + D*: Atalho usado para sair do cotainer e matar a execução, mas esse comando não funciona em todo e qualquer container. No caso do container do Ubuntu essa combinação de teclas vai parar a execução do container, porque o bash é o principal processo que mantém o container em execução.

*CTRL + P + Q*: Atalho utlizado para sair de dentro do container, mas não pará-lo. Pressione CTRL e enquanto estiver pressionado o CTRL vc pressiona o P e o Q.

*docker container run --name toskao -it ubuntu*: A flag *--name* permite que o desenvolvedor dê um nome ao container, no exemplo citado foi dado o nome do container de "toskao", mas poderia ser o nome que o desenvolvedor quisesse.

*docker container attach toskao*: Para entrar nesse container em execução também é possível usar o nome que foi dado a ele, nesse caso o nome do comando é "toskao"

*docker container pause toskao*: Esse comando pausa a execução do container toskao.

*docker container unpause toskao*: Esse comando vai "despausar" o container toskao.

*docker container stop toskao*: O container será parado.

*docker container rm toskao*: Remove o container toskao.

*docker container stats*: Mostra de forma dinâmica quais recursos o container está utilizando como por exemplo CPU e memóriam RAM. As informações são atualizadas em tempo real.

*docker container stats -a*: Mostra quais recursos todos os seus containers estão utilizando, esse comando mostra tanto os containers em execução quanto os cotainers que não estão em execução.

*docker container stats --no-stream*: Mostra quais recursos todos os seus containers estão utilizando, mas de forma estática sem atualização em tempo real como acontece.

*docker container top toskao2*: Usado para mostrar os processos estão sendo executados por um container específico, o nome do container nesse exemplo foi criado um container com o nome **toskao2**.

*docker container logs --details toskao2*: Exibe, em detalhes, as modificações feitas no container toskao2. Sendo necessário especificar o nome do container para que sejam mostrados os logs.

*docker container logs -f toskao2*: A flag -f é usada para mostrar em tempo real os logs do container em execução. Tudo o que acontece no container é mostrado em tempo real quando essa flag é utilizada.

*docker container prune*: Remove todos os containers que estão parados

*docker image ls ou docker images*: Lista as imagens que você baixou

*docker container run --name toskao4 -it ubuntu:20.04*: Esse comando especificando a versão da imagem do Ubuntu que será baixada, como essa imagem não existe, será feito o download e após o download ser concluído esse comando faz com que você entre automaticamente no container do Ubuntu 20.04

*docker image rm bf40b7bc7a11*: O parãmetro *rm*é usado para remover uma imagem que foi baixada, estou usando o **IMAGE ID** para dizer ao CLI qual imagem quero remover, pode ocorrer um erro onde não é possível remover a imagem, pois está sendo usada por um container, nesse caso o container em execução é o **toskao4**, portanto é necessário realizar um *docker container rm toskao4* para que o comando *docker image rm toskao4* seja executado com êxito.

*docker container rm -f toskao4*: A flag *-f* força a remoção de um container que está execução sem a necessidade de usar o parâmetro *stop*para parar a execução do container.

*docker container inspect toskao*: O *inspect*é usado para mostrar informações detalhadas de um determinado container, no caso o container foi o **toskao4**

*docker image inspect e4c58958181a*: O *inspect*também mostra informações sobre uma determinada imagem, basta usar o **IMAGE ID**

*docker container run -d --name meu-nginx ngnix*: O -d serve para baixar e executar o container em background sem a necessidade de entrar diretamente nele, ao contrário do -it que não só executa o container mais também entra no container que está em execução.

*docker container attach 2ef5*: Também é possível entrar no container digitando os quatro primeiros caracteres do **CONTAINER ID**.

*docker container exec -ti meu-nginx ls*: O parâmetro *exec*serve para executar um comando dentro de um container, no exemplo foi usado o comando *-it* ou *-ti*para que seja executado um terminal interativo dentro do container e também foi passado o comando*ls /* para ser executado.

*docker container exec -ti meu-nginx curl localhost*: Vai mostrar um código HTML da página inicial do Nginx

*docker container exec -ti meu-nginx bash*: Executa o bash dentro do container Nginx

*docker container run -d -p 8080:80 --name meu-ngnix ngnix*: A flag *-p* serve para mapear portas entre o host e o container. Isso permite que o container possa receber ou enviar conexões de rede usando as portas do host.

*docker pull cetos:7:* Baixa o container do CentOS versão 7 vindo do [Docker Hub](https://hub.docker.com/). Um repositório com várias imagens Docker

*docker container create --name opa nginx:* O comando *create*apenas cria um novo container, mas não o executa. O novo container pode ser acessado posteriormente