# Relatório do 2° trabalho de TDS

**Aluno: Matheus Rogerio Pesarini**

Segue-se meu passo a passo para fazer cada implementação dos containers na ordem de RunC, LXC BusyBox, LXC Debian, LXD Debian/12, Docker e Podman.

Repositório Codeberg

## Pacotes necessários para o trabalho

- Git
- RunC
- debootstrap
- LXC
- snapd
- snapdcore
- LXD
- libvirt-daemon-driver-lxc
- Docker
- Podman
- Docker nginx
- Docker-compose
- Virt-manager

## RunC FEITO

mkdir runc // criando a pasta runc

cd runc

mkdir rootfs // criando a pasta rootfs

sudo debootstrap stable .\rootfs // instalando o debootstrap na pasta rootfs

runc spec // criando o .json

sudo runc run hello // executando o container

Modifiquei o config.json no “capabilities”

Modifiquei as permissões para o usuário ter total liberdade:

```
"bounding": [
                    "CAP_CHOWN",
                    "CAP_DAC_OVERRIDE",
```

```
            "CAP_FSETID",
            "CAP_FOWNER",
            "CAP_MKNOD",
            "CAP_NET_RAW",
            "CAP_SETGID",
            "CAP_SETUID",
            "CAP_SETFCAP",
            "CAP_SETPCAP",
            "CAP_NET_BIND_SERVICE",
            "CAP_SYS_CHROOT",
            "CAP_DAC_READ_SEARCH",
            "CAP_SYS_ADMIN",
            "CAP_SYS_RESOURCE"
        ],
        "effective": [
            "CAP_CHOWN",
            "CAP_DAC_OVERRIDE",
            "CAP_FSETID",
            "CAP_FOWNER",
            "CAP_MKNOD",
            "CAP_NET_RAW",
            "CAP_SETGID",
            "CAP_SETUID",
            "CAP_SETFCAP",
            "CAP_SETPCAP",
            "CAP_NET_BIND_SERVICE",
            "CAP_SYS_CHROOT",
            "CAP_DAC_READ_SEARCH",
            "CAP_SYS_ADMIN",
            "CAP_SYS_RESOURCE"
        ],
```

```
"permitted": [
        "CAP_CHOWN",
        "CAP_DAC_OVERRIDE",
        "CAP_FSETID",
        "CAP_FOWNER",
        "CAP_MKNOD",
        "CAP_NET_RAW",
        "CAP_SETGID",
        "CAP_SETUID",
        "CAP_SETFCAP",
        "CAP_SETPCAP",
        "CAP_NET_BIND_SERVICE",
        "CAP_SYS_CHROOT",
        "CAP_DAC_READ_SEARCH",
        "CAP_SYS_ADMIN",
        "CAP_SYS_RESOURCE"
 ],
 "ambient": [
        "CAP_CHOWN",
        "CAP_DAC_OVERRIDE",
        "CAP_FSETID",
        "CAP_FOWNER",
        "CAP_MKNOD",
        "CAP_NET_RAW",
        "CAP_SETGID",
        "CAP_SETUID",
        "CAP_SETFCAP",
        "CAP_SETPCAP",
        "CAP_NET_BIND_SERVICE",
        "CAP_SYS_CHROOT",
        "CAP_DAC_READ_SEARCH",
```

```
                    "CAP_SYS_ADMIN",
                    "CAP_SYS_RESOURCE"
                ]
"noNewPrivileges": false
    },
    "root": {
        "path": "rootfs",
        "readonly": false
    }
```

sudo runc run hello

Agora o usuário tem permissões para fazer o que quiser dentro do container RunC

## Git

git init // fazendo o primeiro commit no codeberg e o envio dos arquivos para o repositório

git checkout –b main

git remote add origin https://codeberg.org/MatheusPato/TDSTrabalho.git

git status

git add runc

git commit –m "Pasta runc criada"

git config –global user.email mrogeriopesarini@gmail.com

git config –global user.name "MatheusPato"

git commit –m "Pasta runc criada"

Os comandos git não foram todos mostrados aqui o que resultou somente 3 commits realizados até o fim do trabalho. Muitas etapas tiveram que ser refeitas pois chegavam em caminhos sem saída, acontecia algum erro ou tinha sido feito errado.

## LXC:

## Busybox

sudo apt install lxc // instalando o LXC

sudo lxc-create –n containerbusy –t download // baixando a imagem do container busybox

sudo lxc-attach –n containerbusy // entrando no container

A partir daqui foram feitas tentativas de baixar sudo, faze conexão com a internet entre outras coisas, porém nada deu certo, então o BusyBox se mostrou um container muito limitado (que é a proposta dele), que para fazer a conexão com o virt-manager precisaria ser bastante modificado perdendo o objetivo pelo qual essa distro existe.

## Debian

sudo lxc-create –n container_debian –t download // fazendo a criação do container LXC debian

Opções escritas para o download:

debian

bullseye

amd64

sudo su // os próximos passos serão todos feito em modo root para ter as permissões para acessar a pasta do container

cd .. // voltar para o diretório inicial

cd /var/lib/container_debian // em modo root

mkdir LXC // criando a pasta LXC onde será configurado o .xml

cd LXC

nano container_debian.xml

Pelo editor de texto foi adiciona os seguintes comandos para virtualizar:

```
<domain type='lxc'>
  <name>container_debian</name>
  <memory unit='KiB'>4276800</memory>
  <vcpu placement='static'>1</vcpu>
  <os>
    <type arch='x86_64'>exe</type>
    <init>/lib/systemd/systemd</init>
  </os>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <devices>
    <emulator>/usr/lib/libvirt/libvirt_lxc</emulator>
    <filesystem type='mount' accessmode='passthrough'>
      <source dir='/var/lib/lxc/containerdebian/rootfs'/>
      <target dir='/'/>
    </filesystem>
    <interface type='network'>
```

```
    <source network='default'/>
   </interface>
   <console type='pty'>
    <target type='lxc' port='0'/>
   </console>
  </devices>
 </domain>
```

lxc-start –n container_debian // iniciando o container

lxc-attach –n container_debian // entrando no container

Passwd // criando uma senha para o usuário root

4435

4435

exit

lxc-stop –n container_debian // parando o container

apt install libvirt-daemon-driver-lxc // instalando a biblioteca libvirt

virsh –c lxc:// define container_debian.xml // definindo o documento xml criado para fazer a virtualização

virsh –c lxc:// start container_debian // iniciando o container de forma virtualizada

virsh –c lxc:// console container_debian // entrando no console do container de forma virtualizada

O acesso irá precisar do usuário e senha do container, por padrão o usuário é root e a senha foi criada por mim, sendo 4435.

root e 4435

Após isso caso não tenha sido instalado ainda o virt-manager

sudo apt-get install virt-manager

Agora é so abrir o virt-manager, adicionar uma conexão LXC sem marcar a checkbox e fazer a virtualização do container.

## LXD + LXC DEBIAN/12

apt install snapd // instalando o snapd que é necessário para instalar o LXD no meu caso

snap install core

snap install lxd // instalando o LXD

sudo lxd init

Enter em todas perguntas.

sudo lxc launch images:debian/12 c1 // criando um container lxc baseado na imagem do debian 12

sudo lxc list

```
+------+---------+--------------------+-----------------------------------------------+-----------+-----------+
| NAME |  STATE  |        IPV4        |                     IPV6                      |   TYPE    | SNAPSHOTS |
+------+---------+--------------------+-----------------------------------------------+-----------+-----------+
| c1   | RUNNING | 10.76.86.248 (eth0) | fd42:88ab:256f:25c3:216:3eff:feef:3ded (eth0) | CONTAINER | 0         |
+------+---------+--------------------+-----------------------------------------------+-----------+-----------+
```

Aqui é feito a execução do container e o delete.

sudo lxc start c1

sudo lxc exec c1 -- bash

exit

sudo lxc stop c1

sudo lxc delete c1

## Docker

sudo apt install docker.io // instalando docker no sistema

sudo docker run hello-world // verificando se o docker está funcionando

sudo systemctl start docker // iniciando o docker

sudo systemctl enable docker

sudo Mkdir docker // criando a pasta do docker para o dockerfile

sudo docker pull nginx // baixando a imagem do nginx para hosteamento web

Criado arquivo de texto Dockerfile e dentro dele escrito:

```
FROM nginx:stable-alpine

RUN rm -rf /usr/share/nginx/html/*

COPY ./dist/app /usr/share/nginx/html
```

sudo apt install docker docker-compose docker-doc docker-registry docker.io // instalar os requisitos para fazer o container rootless

sudo Docker –version

Docker version 24.0.5, build 24.0.5-0ubuntu1~22.04.1

whoami // para ver o nome de usuário do meu sistema para dar as permissões de root ao docker

** Fazendo algumas atualizações e downloads

sudo apt install apt-transport-https ca-certificates curl gnupg-agent software-properties-common

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

sudo echo deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu focal stable | sudo tee /etc/apt/sources.list.d/docker.list

sudo apt update

sudo curl -fsSL https://get.docker.com/rootless | sh // instalar docker rootless

sudo docker run -it --rm -d -p 8080:80 --name web nginx // Funcionando!!! O ip do localhost estava exibindo o html padrão do nginx

CONTAINER ID IMAGE COMMAND CREATED STATUS

3e08c5e133a6 nginx "/docker-entrypoint.…" 20 minutes ago Up 20 minutes 0.0.0.0:8080

PORTS NAMES

->80/tcp, :::8080->80/tcp youthful_sinoussi

** Tentarei modificar o html.index na pasta do nginx

Demorei mas como root consegui achar a pasta nginx contendo o html da pagina.

Criei na pasta docker uma pasta dist e dentro dela uma pasta app com o index.html modificado, na pasta docker também tem um dockerfile com as instruções para a atribuição do html ao nginx.

## ROOTLESS

sudo usermod -aG docker $USER // Adicionando as permissões do root para o usuário, assim não precisando de sudo

docker run hello-world // verificando se está de forma rootless

docker run -it ubuntu bash // entrando no container ubuntu de forma rootless

apt-get update // atualizando o container ubuntu para instalar vim

apt-get install -y vim

vim meu_arquivo2 // criando o arquivo vim

Digitei um texto qualquer nele

:wb // para salvar e sair

exit // saindo do container

## Podman

sudo apt-get install -y podman // instalando o podman

podman run -it --rm alpine /bin/sh // entrando num container alpine pelo podman

cat /etc/os-release // exibe a versão do alpine

apk add vim // instalando o vim

Vim meuarquivo.txt

Digitei um texto qualquer no txt e apertei esc, :wb enter para salvar e sair

exit