Terceira Avaliação - 13/01/2025 : 20/01/2024 - Redes de Computadores II

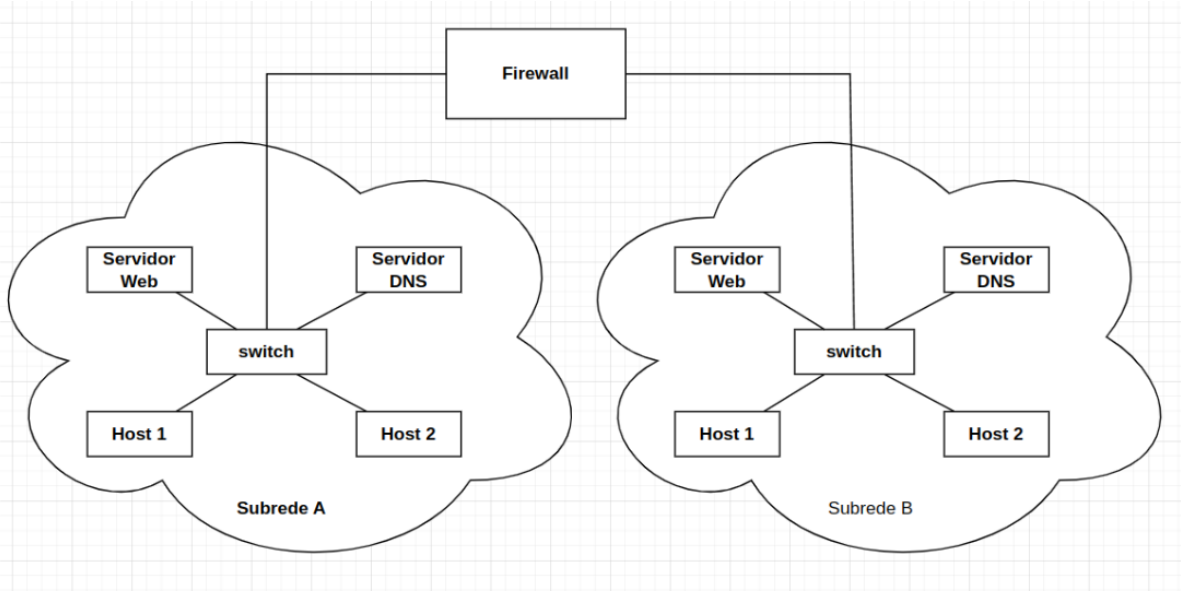
Equipe:

- MAURICIO BENJAMIN DA ROCHA : 20219016147
- PEDRO ANTONIO VITAL DE SOUSA CARVALHO : 20219029753

Objetivos

- Faça um relatório (how-to) descrevendo todos os passos.
    - Deixe bem claro o que você fez.
    - Descreva sequencialmente todos os processos requisitados nas questões.
    - Não basta colocar as imagens, descreva todas as figuras no relatório.
    - Relatório em PDF.
- Faça um vídeo explicando e demonstrando o desenvolvimento, publicar o vídeo no YouTube e adicionar o link no arquivo resposta.
    - verificar se o vídeo tem áudio.
- O que enviar no arquivo:
    - relatório;
    - link do vídeo

Questão Única: Utilizando Docker crie a infraestrutura abaixo e faça o que se pede a seguir



Descrição da figura: Duas subredes (A e B) interligadas por um roteador (firewall); Cada sub-rede com 4 computadores e um *switch* interligando todos os computadores dentro da subrede; Cada sub-rede contém um servidor web e um servidor DNS.

De acordo com a infraestrutura a cima, realize as operações numeradas a baixo

1. Instanciar a rede ilustrada na figura utilizando o docker. Utilize a imagem do ubuntu em todas os hosts; (1pt)

Para instanciar a infraestrutura apresentada, usaremos técnicas de engenharia de software para estruturar de forma organizada e eficiente todos os containers docker que usaremos.

Passo 1. Estruturar o Projeto

Crie uma pasta para o projeto, onde todos os arquivos relacionados ao projeto serão adicionados nela. Neste guia usaremos a pasta `infra` para isso, conforme ilustrado a baixo.

```
infra/
```

Dentro da pasta `infra` iremos criar um arquivo `docker-compose.yaml` para gerenciar nossos containers de forma mais organizada e reprodutível.

```
infra/
    docker-compose.yaml
```

Iremos dividir a infraestrutura em partes, usando o conceito de módulos, de forma que teremos 3 módulos principais: `Rede A` `Firewall` e `Rede B`. Para isso criaremos respectivamente as pastas `net-a`, `net-b` e `firewall` conforme apresentado a baixo.

```
infra/
    docker-compose.yaml
    firewall/
    net-a/
    net-b/
```

Conforme apresentando na figura, precisamos de 3 componentes essenciais em cada rede, sendo eles `servidor dns`, `servidor web` e `computadores hosts`. Iremos criar respectivamente em cada pasta de rede, 3 novas pastas, sendo elas `dns`, `web` e `host` respectivamente conforme apresentado a baixo.

```
infra/
    docker-compose.yaml
    firewall/
    net-a/
        dns/
        host/
        web/
    net-b/
        dns/
        host/
        web/
```

Visando ter um maior controle sobre as imagens docker para nosso containers, iremos usar um `dockerfile` customizado para componente da nossa rede.

Dentro de cada uma das pastas`host`, crie um `dockerfile` com o seguinte conteúdo:

```
FROM ubuntu:latest

RUN apt-get update && apt-get update -y && apt-get install -y curl
&& apt-get install iputils-ping -y && apt-get install net-tools -y

CMD ["sh", "-c", "sleep infinity"]
```

```
infra/
    docker-compose.yaml
    firewall/
```

```
    net-a/
        dns/
        host/
            dockerfile
        web/
    net-b/
        dns/
        host/
            dockerfile
        web/
```

O `dockerfile` irá garantir que os containers dos hosts serão construídos usando uma imagem do Linux `ubuntu` com algumas dependências customizadas de ferramentas para testar a rede como `ping`, `ifconfig` e etc.

Para o `servidor dns` iremos usar uma imagem do servidor dns `bind9` baseada o ubuntu, mas com alguns ajustes nossos que iremos fazer no futuro. Crie docker files para os servidores dns e os coloque em suas respectivas pastas com base nos exemplos a baixo:

Conteúdo do dockerfile

```
FROM ubuntu/bind9:latest

RUN apt-get update -y && apt-get install -y dnsutils

ENV TZ=UTC

EXPOSE 53/tcp 53/udp

CMD ["sh", "-c", "sleep infinity"]
```

Estrutura do projeto ao adicionar os docker files de `dns`

```
infra/
    docker-compose.yaml
    firewall/
    net-a/
        dns/
            dockerfile
        host/
            dockerfile
        web/
    net-b/
        dns/
            dockerfile
        host/
```

```
            dockerfile
        web/
```

Agora vamos ao ultimo componente que iremos customizar, o nosso `servidor web`. Para o servidor web foi escolhido o `nginx` por ser um servidor web muito usado no mercado de trabalho, tornando-se acessível a um grande volume de guias e tutoriais na internet para ajudar a configurá-lo. Se deseja saber mais sobre ele, recomendo que de uma olhada nas referencias usadas neste trabalho. Deixando as enrolações de lado, use o conteúdo a baixo para criar cada `dockerfile` para as pastas da web

```
FROM nginx:latest

CMD ["nginx", "-g", "daemon off;"]
```

Não se preocupe que iremos adicionar e configurar uma pagina web customizada para o dockerfile apresentar futuramente, mas por hora apenas adicione os docker files em suas respectivas pastas, obtendo o seguinte resultado.

```
infra/
    docker-compose.yaml
    firewall/
    net-a/
        dns/
            dockerfile
        host/
            dockerfile
        web/
            dockerfile
    net-b/
        dns/
            dockerfile
        host/
            dockerfile
        web/
            dockerfile
```

Terminamos de preparar temporariamente o necessário para nossas redes, agora vamos configurar o nosso `firewall`. Para configurar a imagem do firewall, iremos precisar de alguns recursos que serão abordados em questões futuras, portanto iremos apenhas preparar o mínimo e terminar a tarefa quando "sua hora chegar". Crie um `dockerfile` para o `firewall` com o seguinte conteúdo:

```
```

```
FROM ubuntu:latest

RUN apt-get update && apt-get install -y \
    iproute2 iptables iputils-ping \
    && echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf \
    && apt-get clean

CMD ["sh", "-c", "sysctl -p && tail -f /dev/null"]
```

Ao final teremos o seguinte resultado

```
infra/
    docker-compose.yaml
    firewall/
        dockerfile
    net-a/
        dns/
            dockerfile
        host/
            dockerfile
        web/
            dockerfile
    net-b/
        dns/
            dockerfile
        host/
            dockerfile
        web/
            dockerfile
```

Agora temos a estrutura base para trabalhar com todos os componentes da nossa rede!

Vamos estruturar nossa rede usando nosso arquivo `docker-compose.yaml` para instanciar os containers que iremos precisar, aproveitando os arquivos `dockerfile` para termos nossas imagens customizadas para facilitar nossas vidas.

```
networks:
  subnet-A:
    driver: bridge
    ipam:
      config:
        - subnet: 10.0.0.0/24

  subnet-B:
    driver: bridge
```

```
    ipam:
      config:
        - subnet: 20.0.0.0/24

services:
  firewall:
    build:
      context: ./firewall
    container_name: firewall
    cap_add:
      - NET_ADMIN  # Permissões para manipular as configurações de
rede
    sysctls:
      net.ipv4.ip_forward: "1"  # Habilitar roteamento de pacotes
    networks:
      subnet-A:
        ipv4_address: 10.0.0.5
      subnet-B:
        ipv4_address: 20.0.0.5
  # Rede A
  host1-net-a:
    build:
      context: ./net-a/host
    container_name: host1-net-a
    networks:
      subnet-A:
        ipv4_address: 10.0.0.2
    dns:
      - 10.0.0.20
  host2-net-a:
    build:
      context: ./net-a/host
    container_name: host2-net-a
    networks:
      subnet-A:
        ipv4_address: 10.0.0.3
    dns:
      - 10.0.0.20
  dns-a:
    build:
      context: ./net-a/dns
    container_name: dns-a
    ports:
      - "30051:53"
      - "30051:53/udp"
    networks:
      subnet-A:
        ipv4_address: 10.0.0.20
  web-a:
    build:
      context: ./net-a/web
```

```
    container_name: web-a
    networks:
      subnet-A:
        ipv4_address: 10.0.0.10
    ports:
      - "8051:80"
  # Computadores Subrede B
  host1-net-b:
    build:
      context: ./net-b/host
    container_name: host1-net-b
    networks:
      subnet-B:
        ipv4_address: 20.0.0.2
    dns:
      - 20.0.0.20
  host2-net-b:
    build:
      context: ./net-b/host
    container_name: host2-net-b
    networks:
      subnet-B:
        ipv4_address: 20.0.0.3
    dns:
      - 20.0.0.20
  web-b:
    build:
      context: ./net-b/web
    container_name: web-b
    networks:
      subnet-B:
        ipv4_address: 20.0.0.10
    ports:
    - "8052:80"
  dns-b:
    build:
      context: ./net-b/dns
    container_name: dns-b
    ports:
      - "30052:53"
      - "30052:53/udp"
    networks:
      subnet-B:
        ipv4_address: 20.0.0.20
```

Fique tranquilo que vamos discutir o `docker-compose.yaml` passo a passo!

- Redes
  - subnet-A:

        - Usa o driver bridge para criar uma rede de ponte.
        - Configurada com o intervalo de endereços IP 10.0.0.0/24.
    - subnet-B:
        - Também usa o driver bridge.
        - Configurada com o intervalo de endereços IP 20.0.0.0/24.
- Serviços
    - firewall:
        - Constrói a partir do contexto firewall.
        - Nome do container: firewall.
        - Adiciona a capacidade NET_ADMIN para manipular configurações de rede.
        - Habilita o roteamento de pacotes com net.ipv4.ip_forward: "1".
        - Conectado a subnet-A com o IP 10.0.0.5 e a subnet-B com o IP 20.0.0.5.
    - host1-net-a:
        - Constrói a partir do contexto ./net-a/host.
        - Nome do container: host1-net-a.
        - Conectado a subnet-A com o IP 10.0.0.2.
        - Usa o DNS 10.0.0.20.
    - host2-net-a:
        - Constrói a partir do contexto ./net-a/host.
        - Nome do container: host2-net-a.
        - Conectado a subnet-A com o IP 10.0.0.3.
        - Usa o DNS 10.0.0.20.
    - dns-a:
        - Constrói a partir do contexto dns.
        - Nome do container: dns-a.
        - Mapeia as portas 30051:53 e 30051:53/udp.
        - Conectado a subnet-A com o IP 10.0.0.20.
    - web-a:
        - Constrói a partir do contexto web.
        - Nome do container: web-a.
        - Conectado a subnet-A com o IP 10.0.0.10.
        - Mapeia a porta 8051:80.
    - host1-net-b:
        - Constrói a partir do contexto ./net-b/host.
        - Nome do container: host1-net-b.
        - Conectado a subnet-B com o IP 20.0.0.2.
        - Usa o DNS 20.0.0.20.
    - host2-net-b:
        - Constrói a partir do contexto ./net-b/host.
        - Nome do container: host2-net-b.
        - Conectado a subnet-B com o IP 20.0.0.3.
        - Usa o DNS 20.0.0.20.
    - web-b:
        - Constrói a partir do contexto web.

        - Nome do container: web-b.
        - Conectado a subnet-B com o IP 20.0.0.10.
        - Mapeia a porta 8052:80.
    - dns-b:
        - Constrói a partir do contexto dns.
        - Nome do container: dns-b.
        - Mapeia as portas 30052:53 e 30052:53/udp.
        - Conectado a subnet-B com o IP 20.0.0.20.

obs: Voce pode estar se perguntando "E os switch's?", e fique calmo que vou te explicar haha.

No Docker, o driver de rede bridge atua como um switch virtual que conecta os containers dentro da mesma rede. No nosso arquivo `docker-compose.yaml`, as redes subnet-A e subnet-B usam o driver bridge, o que significa que cada rede tem seu próprio switch virtual

- subnet-A: Todos os containers conectados a subnet-A (como host1-net-a, host2-net-a, dns-a, web-a) estão conectados a um switch virtual criado pelo driver bridge.
- subnet-B: Todos os containers conectados a subnet-B (como host1-net-b, host2-net-b, dns-b, web-b) estão conectados a outro switch virtual criado pelo driver bridge.

Portanto, o driver bridge do Docker atua como o switch que conecta os componentes da rede subnet-A internamente.