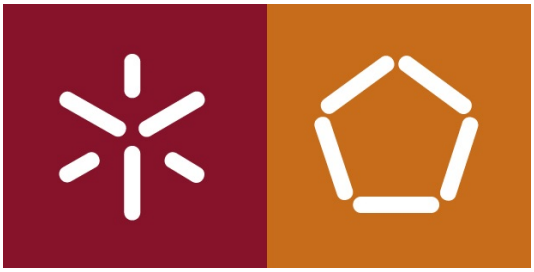

Virtualização de Redes TP2

Grupo 9
Bruno Chianca Ferreira PG33878
João Rui de Sousa Miguel A74237
Paulo Jorge Machado Guedes A74411

23 Março 2018

1 Introdução

No segundo trabalho prático, o objetivo era de aprofundar o conhecimento na criação de imagens *Docker* personalizadas e que poderiam fazer uso do *workflow* do *Docker Hub* em conjunto com repositórios do *Github*.

O projeto consiste em construir uma imagem nova baseada na imagem do *Ubuntu* à qual serão adicionadas algumas ferramentas como o *Mininet*, *Wireshark* e *tcpdump*. Quando uma alteração ao projeto for adicionada ao *GitHub* (*Push, Pull request, ...*), uma atualização automática é desencadeada no *Dockerhub*, esta também poderá ser desencadeada manualmente, diretamente nesta última plataforma.

2 Estrutura do projeto

Para criar e inicializar o projeto no ambiente *Docker*, foi feito uso de um arquivo de criação de imagens *Dockerfile*. O projeto consiste em dois ficheiros, o previamente mencionado *Dockerfile* e um ficheiro do *Docker compose* para orquestração.

- **Dockerfile** - Contém a imagem base, pontos de montagem para volumes, pré-instalação de programas e configuração de portas que serão expostas.
- **docker-compose.yml** - Contém as configurações necessárias para criar *containers* baseados numa imagem.

3 Desenvolvimento

Explicados todos os aspetos fundamentais do projeto, iremos proceder à explicação do desenvolvimento por parte do grupo de trabalho.

3.1 Dockerfile

Numa primeira fase, desenvolveu-se o ficheiro *Dockerfile* com o intuito de proceder à instalação das ferramentas anunciadas para este trabalho de forma automatizada.

Cumprindo os requisitos do enunciado, especificou-se um ficheiro capaz de instalar, aquando inicialização da imagem *docker*, as ferramentas *mininet*, *tcpdump* e *wireshark*. Abriram-se ainda as portas especificadas (6633 e 6653). A especificação final deste documento poderá ser observada na listagem a baixo.

```
1  FROM ubuntu:latest
2  RUN apt-get update && apt-get install -y apt-utils && apt-get install -y git\\
3  && apt-get install -y sudo && apt-get install -y net-tools \\
4  && apt-get install -y iputils-ping
5  RUN cd /root && git clone git://github.com/mininet/mininet \\
6  && mininet/util/install.sh -a
7  VOLUME /home/ubuntu
8  WORKDIR /home/ubuntu
9  RUN apt-get install -y tcpdump
10 RUN DEBIAN_FRONTEND=noninteractive apt-get install -y wireshark
11 EXPOSE 6653
12 EXPOSE 6633
```

Pode-se ver na primeira linha deste documento que iremos estar a realizar estas operações sobre um ambiente *linux*, mais concisamente *Ubuntu*. A 2ª, 3ª e 4ª linha demonstram quais as ferramentas genéricas necessárias de pé-instalação para que seja possível executar as ferramentas necessárias, estas poderão ser o motivo de problemas de instalação para as outras ferramentas, dado incorporarem na listagem de pré-requisitos destas mesmas. Na 5ª linha efetua-se o *clone* do repositório que contém a instalação do *mininet*, sendo que na 6ª se procede precisamente a esta instalação. Definem-se de seguida o volume e a diretoria de trabalho (neste caso ambas a */home/ubuntu*). Na 9ª linha instala-se a ferramenta *tcpdump*, utilizando a *flag* -y para que se aceitem todos os passos de instalação automaticamente. A 10ª linha demonstra precisamente a instalação do *wireshark* com a peculiaridade de existir a necessidade de definir que a instalação terá uma plataforma não interativa (previne bloqueios em *runtime*). Finalmente exportam-se os *ports* pedidos (6633 e 6653).

3.2 Docker compose

Após criada a imagem utilizando-se o comando *"docker build ."* foi necessária a criação de um arquivo *docker compose* para que a possamos utilizar e configurar mais facilmente. A seguir é exibida a listagem do ficheiro *docker-compose.yml* desenvolvido.

```
version: '2'
services:
  ubuntu_mini:
    image: brunobcf/vrtp2
    container_name: ubuntu_mininet
    volumes:
      - ubuntu_vol:/home/ubuntu
      - /tmp/.X11-unix:/tmp/.X11-unix
      - /lib/modules:/lib/modules
    ports:
      - 6653:6653
      - 6633:6633
    networks:
      - network1
    privileged: true
    environment:
     - DISPLAY
    #command: bash -c "service openvswitch-switch start && xmessage Container iniciado"
    #command: bash -c "service openvswitch-switch start && wireshark"
    command: bash -c "service openvswitch-switch start && tail -F /dev/null"
    #O comando tail -F /dev/null serve apenas para fazer com que o container continue a correr depois do
        startup.
volumes:
  ubuntu_vol:
    driver_opts:
      type: none
      device: $PWD
      o: bind

networks:
  network1:
  driver: bridge
```

De seguida tem-se a descrição de cada item deste ficheiro.

- **version** - Define a versão do formato do arquivo do docker-compose.
- **services** - Lista todas os *containers* que serão orquestrado por este ficheiro. Cada *container* contém uma lista de configurações individuais.

  image - nome da imagem do *dockerhub* a ser utilizada.

  container-name - Etiqueta que será associada ao *container* depois de iniciado, para facilitar qualquer referência a este.

  volumes - configuração de como serão utilizados os volumes no *container*, criando um par *"volume externo:diretoria interna"* do *container*.

  ports - mapeamento das portas do *container* que estarão disponíveis para o *host*.

  networks - configuração de quais interfaces de rede estarão disponíveis no *container*

  privileged - permite dar privilégios adicionais aos *containers*.

  environment - permite exportar variáveis de ambiente do ambiente *host* para o ambiente do *container*.

  command - permite executar um comando após a inicialização do *container*.
- **volumes** - Define volumes que podem ser usados por um ou mais *containers*.
- **networks** - Define interfaces de rede que podem ser usadas por um ou mais *containers*.

3.3 Valorização

Tendo concluído o projeto base, decidiu-se realizar as tarefas de valorização. Nestas era pedido que se procedesse à instalação da ferramenta *mininet* recorrendo ao seu código fonte. Esta alternativa foi explorada pelo trabalho base, sendo que o grupo efetuou, desde logo, a sua instalação recorrendo ao *GitHub* da própria ferramenta. O comando executado de seguida ao clone do repositório, *./mininet/util/install.sh -a* efetua o download de todos os pacotes necessários à compilação e execução de todos os módulos disponíveis no projeto *Mininet*.

O segundo aspeto importante no que toca à valorização deste trabalho prático, passa por executar aplicações com interface gráfica, *GUI*, dentro do próprio *container*. Para tal adicionaram-se as linhas já vistas na secção anterior, relativas ao documento *docker-compose.yml* que referem o *environment: -Display* juntamente com o comando *command: bash -c "..."*, utilizado para iniciar um comando dentro do *container*, este poderá ser relativo a um programa/ferramenta com a interface gráfica desejada para teste ou apenas um comando para permitir que o *container* continue correndo. Sendo assim, o utilizador poderá executar qualquer programa que necessite de uma interface gráfica e o mesmo irá conectar-se ao servidor X correndo no *host*. Tendo isto em mente, é então necessário permitir que conexões sejam feitas ao servidor X do *host*, e para tanto é necessário executar o comando *xhost +*. O grupo fez diversos testes e constatou que o procedimento funciona como esperado. Caso o *host* seja Windows ou *macOS*, provavelmente é necessária a instalação prévia de um servidor X como *Xming* (*Windows*) ou *XQuartz* (*macOS*).

Finalmente integrou-se a opção de criar volumes diretamente utilizando o *dockerfile* para que posteriormente estes possam ser mais comodamente mapeados para o equipamento *host*. Acabados todos estes elementos, verificou-se que o trabalho ficou realizado na integra. Este relatório especifica portanto não só o trabalho desenvolvido que era pedido, como também todos os elementos que compreendem a valorização extra.
Links para repositórios relativos ao trabalho desenvolvido:

- https://hub.docker.com/r/oluap18/virtualizacao-de-redes/
- https://github.com/Oluap18/Virtualizacao-de-Redes/

4 Conclusão

Este trabalho ajudou a aprofundar o conhecimento em criação de imagens personalizadas e utilizar o recursos de *workflow* do *Dockerhub* funcionando em conjunto com *GitHub* para a automatização do processo de criação de novas imagens. O conhecimento mostra-se bastante útil para a criação de ambientes virtuais específicos para desenvolvimento e testes de diversas aplicações, permitindo inclusive utilizar ferramentas com interface gráfica.