exercise 5

For submission to the tasks:
As always, brief documentation of the individual steps and configurations and illustrations with screenshots of the inputs/outputs, configurations, etc.

1) Use the usual example network

host1 router host2

<192.168.10.10> ← → <192.168.10.2> <192.168.20.2> ← → <192.168.20.10>

The packages 'openssh-client' and 'proxychains' should be installed on host1, the packages 'openssh-server' and 'proxychains' on host2 and router.

a) Create the Dockerfiles for the corresponding Docker images 'simple_router_sshclient' and 'simple_router_sshserver', which are each based on the image 'simple_router'. For more convenient handling, you can create docker-compose.yml files and use docker-compose to create the images.

b) Change the previous docker-compose.yml for the network scenario to use the images with the ssh installations.

exercise 5

2) Generate a Makefile for the make tool, which in turn can use dockercompose, docker and others, allowing you to be comfortable and reproducible
- start the networks and containers,
- to stop and destroy the networks and containers,
- display the status or existence of networks, images and containers,
- start a bash on host1, host2 and router
- and sets the default routes of host1 and host2 during or after the start of the container in such a way that a ping between host1 and host2 can be run without any further action.

Other functions are conceivable.

exercise 5

simple_router/Dockerfile:

```
From simple

RUN apt-get update -y &&\
  apt-get install -qqy dnsutils net-tools iproute2 iptables \
      iputils-ping procps netcat-openbsd &&\ apt-get
  clean &&\
  rm -rf /var/lib/apt/lists/*

CMD ["/bin/bash"]
```

simple_router/docker-compose.yml:

```
versions: '2.1'
service:
    simple_router:
        build:
            context: .
            dockerfile: Dockerfile

        container_name: simple_router image:
        simple_router
        cap_add:
        - NET_ADMIN
        stdin_open: true
        tty: true
```

exercise 5

network_example_2/Makefile:

```
# for indentations use 'tab' DC =
`which docker-compose`

all: build

pear:
	@$(DC) --version

# make docker image
build:
	@$(DC) -f docker-compose.yml build

begin:
	@$(DC) -f docker-compose.yml up -d

Stop:
	@$(DC) -f docker-compose.yml down

status:
	@echo "Networks:\n=========\n"
	@docker network ls
	@echo "Images:\n=======\n"
	@docker images
	@echo "Container:\n==========\n"
	@docker container ls -a
```

exercise 5

3) Logging in from host1 to host2 - later the limited execution of individual commands - should be made possible and tested via ssh (see also 'SSH Hands On')

a) Attempting user/password based access... and making the necessary changes on host2 up to success

b) How does host2 identify itself to the requesting host1? See if there are any changes to the critical data (which ones?) when using ssh.

c) Generate a key pair on host1 and use it in such a way that it is possible to log in from host1 to host2 without a password.

d) How could the admin on host2 proceed if he wants to enable others – e.g. a user on host1 – to execute a single command on host2 without granting full access to an account at the same time?
Try this.