Intro

We recommend developing and running the following exercises on a recent Ubuntu Linux distribution.

Our preferred programming languages are Python and Go (Golang). But if you really must you can use another language as well :-).

Upload your solutions to GitHub and share the repository URL with us.

Don't worry if you don't make all of the exercises. We'll consider also partial solutions.

Exercise 1

1.1 Install and configure Docker

Create Ansible playbook `site.yml` that will:

- Install the Docker service
- Enable container logging to Docker host's syslog file

Example of expected result:

```
$ docker
The program 'docker' is currently not installed. You can install it by
typing:sudo apt install docker.io
$ ansible-playbook -i "localhost," -c local site.yml
<...snip...>
$ docker -v
Docker version 17.12.0-ce, build c97c6d6
$ docker info | grep 'Logging Driver'
Logging Driver: syslog
```

1.2 Write a weather reporting program

Create a simple script `getweather` that will get the current weather from https://openweathermap.org/api (you can use a library like pyowm). The script reads no CLI arguments. It is parametrized via the following environment variables: `OWM_API_KEY`, `OWM_CITY`. It prints to stdout.

Example of expected results:

```
$  export OWM_API_KEY='xxxx'
$ export OWM_CITY='Honolulu'
$ ./getweather
city="Honolulu", description="few clouds", temp=70.2, humidity=75
```

1.3 Dockerize and run the program

Create `Dockerfile` for building image for the getweather program you've written. Then build and run the image.

Example of expected result:

```
$ docker run --rm -e OWM_API_KEY="xxxx" -e OWM_CITY="Honolulu" getweather:dev
$ grep -i honolulu /var/log/syslog
Nov 30 11:50:07 ubuntu-vm ae9395e86676[1621]: city="Honolulu",
description="few clouds", temp=70.2, humidity=75
```

Exercise 2

2.1 Write a network scanner

Write a program for repetitive network scans that will display differences between scans. Scan can be executed either using external tools or dedicated libraries of the selected programming language.

- Target of the scan must be parametrized as CLI argument
- Target can be single IP address as well as network range

Example of expected result:

```
$ ./scanner 10.1.1.1
10.1.1.1
* 22/tcp open
* 25/tcp open
```

2.2 Kubernetize and deploy the scanner

Create Docker image containing the scanner program and push it to a public repository like Docker Hub. Create Kubernetes manifest that will run the image periodically every 5 minutes. Install local Kubernetes like minikube and apply the manifest.

Example of expected result:

```
$ kubectl apply -f scanner.yaml
$ kubectl get cj
NAME                  SCHEDULE      SUSPEND   ACTIVE   LAST SCHEDULE   AGE
scanner               0 5 * * *     False     0        7m              13m
```