DV2593

DevOps and Virtualization

Laboratory booklet

Scope: The purpose of Lab1 is to practice with process virtualization, i.e. Docker Container.

Individual work

Working environment: your own pc/laptop/server. You can work in the Lab room or where you prefer.

Assessment (U/UX/G): You will be evaluated on the basis of the submitted report.

Specific/punctual instruction are not given to let you master the concepts and strenghtening your critical thinking and problem solving capability.

1 Process virtualization

In this experience you will learn how to deploy your hown application with Docker containers and how to run Docker services.

The reference documentation is the Docker Documentation available at https://www.docker.com

1.1 Download and instal Docker

Dowload docker according to your host operating system. You can download Docker at https://www.docker.com/

Install docker and check if the environment is properly set up.

1.2 Build docker images and run containers

Create a simple client/server 3-tier application, accessible through a web or REST interface and that read and write data from the disk (from a file or a database). The data created by the application should persist in the system after the container is terminated.

Build the images for the application components layering the selected application and all the libraries and dependencies needed. To make an example, if your application is composed by a web server and a database, you should build two containers, one for the web server and one for the database server and then link/connect them (https://docs.docker.com/network/). To build an image you need to create the `dockerfile` that specifies the layers of the container and the dependencies.

Inspect all the layers of the built image to check that everything is in the right place. Upload your image in the Docker public registry.

Select and implement a method to persistently store the data, it could be volumes (https://docs.docker.com/storage/). or storage drivers (https://docs.docker.com/storage/storagedriver/)

Finally, you can run your application. After starting and networking the containers, test your application and verify the data written in the volume persist after the container is terminated.

1.3 Bind mount

Now, deploy and run your application using the host file system with a bind mount.

1.4 Container orchestration

The goal now is to deploy your application as a docker service and to orchestrate the service with docker swarm.

A service is described by a `docker-compose.yml` file, a YAML file defining how Docker containers behave when running. The `docker-compose.yml` file should include: from where to download the image and how to deploy the container, that is: how many replicas are need; the limits on resource usage; the restart policy; the load balancing policy; the communication ports and the network used by the containers.

When you have done with your docker-compose file you can init docker swarm and then you can deploy your service. Check the service is running and verify that the target number of instances (tasks in the Docker jargon) is running.

Finally, you can test your service: check that the load balancing work as expected and the resource limits and usage you configured are met. To monitor the container performance parameters you can use the Docker API (docker stat) or cAdvisor (https://github.com/google/cadvisor). Terminate a task to check that a new one is activated to maintain the target number of instances. Try to manually scale the service increasing or decreasing the target number of instances for the service changing the value in the docker-compose file and re-deploying the service.

1.5 Assessment

The student should produce a report, that **could be integrated with screen recording**, that could be a feaster way to show that the application is working properly or the swarm scale, and the like. In the screen recording, please add audio or text comments.

The student should produce a report describing:

1. The application created for the lab, and specifically what are the application components, what are the functionalities of each application component, how the application components interact.
2. How you built and run the container. Specifically:
    - provide the dockerfile (and eventually the dependencies file) with comments – please include the dockerfile in the report.
    - Show what are the layers of the container image (provide and comment the commands used for inspection and the results of the inspection)
    - Describe which networking method you used to let the containers communicate each other and to access the application (provide and comment the commands used for the configuration)
    - Show that the application works, and that data are stored persistently **(here for example you can use screen recording)**.

3. How you orchestrate a service. Specifically
    - Provide the docker-compose file with comments – please include the docker-compose file in the report.
    - Provide the list of commands (with comments) used to setup and run the service
    - Show that the service is up and running with the desired number of tasks and that the service is capable to keep the target number of also in case of failures, e.g. simulated with the termination of a task. **(Here, for example, you can use screen recording).**
    - Show that you can manually scale the service (provide also he commands used). **(Here, for example, you can use screen recording).**
    - Show that the limits imposed to the use of resources properly work.

In all the above points, always motivate your choises, discuss the issues encountered and how you solved those issues.

No specific template for the report is provided. You can use the BTH template for reports or your own template.

The assessment is pass or fail (U/Ux/G).