**H8: Docker**

*SSH deel moet niet gekend worden*

**Containervirtualisatie**

Op uw fysiek systeem download het zip-bestand en install vagrant(choco install vagrant) kan 20min duren. Cd in dockerlab en doe vagrant status en dan vagrant up kan 20min duren. Ga naar <http://192.168.56.20:9000> om portainer.io te accessen en maak een gebruiker aan. Gebruiker = admin en wachtwoord = verysecurepassword123.

Start de VM door te navigeren naar de dockerlab dir die gegeven is en dan via vagrant ssh dockerlab log je in.

We tonen de status van docker door systemctl comando en TCP server porten dat er gebruikt worden in foto hieronder.

Text

Description automatically generated

Docker ps en images om de runnende containers en images op te lijsten.

Graphical user interface

Description automatically generated

**Our first containers**

We checken of docker goed runt door de hello-world container te downloaden en op te starten.

Text

Description automatically generated

We zien dat de dockercontainer nu nog niet runt maar wel gedownload is. Docker ps geeft alle runnende containers en docker ps -a geeft alle containers ook die gestopt zijn.

Graphical user interface, application

Description automatically generated

We verwijderen de hello-world image en checken dit via docker ps -a. (docker rm containerID kan gebruikt worden om container te verwijderen als error voorkomt)

Graphical user interface, application

Description automatically generated

Alpine Linux is een kleine Linux distributie zeer handig voor in een Docker container en om apps op te runnen.

Om te launchen(interactive en open shell) -> docker run -i -t –name alpine alpine

Hierin zit je in een rootshell. Dit is de output van het help commando en toont dus alle commandos beschikbaar in de rootshell.

Text

Description automatically generated

Docker container ls is hetzelfde als docker ps. Docker inspect alpine geeft specifieke info over container in vorm van JSON-object.Graphical user interface, text, application

Description automatically generated

Docker top alpine geeft runnende processen in de gegeven container. Zoals process Ids en hun CPU en memory usage.



We exiten the Alpine shell en runnen de 3 commando’s weer. We zien dat alpine container niet meer runt en de status is veranderd naar exited en krijgen zelf een error bij docker top alpine.

Graphical user interface, text

Description automatically generated



Daarna runnen we de container op de achtergrond via onderstaande commando. Een error omdat we het al in een container gebruiken. Doe gwn docker start alpine wat automatisch de container op achtergrond runtA screenshot of a computer

Description automatically generated with medium confidence

Maar het heeft geen running services dus is niet direct reachable. Daarom docker exec -t alpine COMMANDO.

Bij /bin/hostname is de hostname is the same as containerID.

Text

Description automatically generated

Het ip address is 172.17.0.2 en we kunnen dit adres bereiken vanaf de dockerlab VM.

A screenshot of a computer

Description automatically generated with medium confidence

Door /bin/sh sshen we in de alpine container en als we hieruit gaan zien we dat het blijft draaien.

Graphical user interface, application

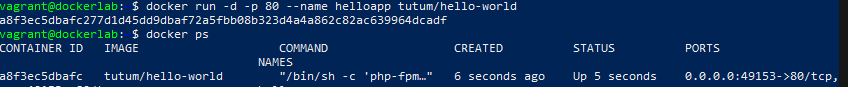
Description automatically generated

We stoppen de container en verwijderen het

Graphical user interface, text

Description automatically generated

We runnen een nieuwe gedownloade container tutum/hello-world en noemen die helloap. Runt op poort 80.



We zien de html pagina als we curl op de url gebruiken.

Text

Description automatically generated

De 80-poort wordt geforward naar de 49135 port op de dockerlab VM.

Graphical user interface, text

Description automatically generated

We kunnen dan ook de website gaan bezoeken op ons fysiek systeem

Graphical user interface, text, website

Description automatically generated

We halen een officele MySQL Docker image op en en creeren een volume om persistente data bij te houden in de container. Daarna checken we of het is gelukt. De mount point is de locatie op host systeem waar het volume opgeslagen wordt.

**Persistent data**

Text

Description automatically generated

We starten de MySQL container en bekijken de logs.

Text

Description automatically generated

In de container open een MySQL tekst console.

Graphical user interface, text

Description automatically generated

We maken een tabel en voegen wat data toe.

Text

Description automatically generated

Vanaf ons fysiek systeem of in de dockerlab VM kunnen we nu aan de database container.

Text

Description automatically generated

Hier zien we tabel die we hebben toegevoegd in de MySQL container.



We verwijderen de db container en herstarten het.

Text

Description automatically generated

We zien dat de data behouden blijft.



**Building a custom container image**

Alle files om een custom image te bouwen staat in de labs/static-website directory. We bouwen het en tonen alle images.

Text

Description automatically generated

Text

Description automatically generated

We starten een container en forwarden port 80 naar 8080 op ons fysiek systeem. We kunnen dit checken via curl

Text

Description automatically generated

**Layered file system**

A picture containing text

Description automatically generated

sha256:ded7a220bb058e28ee3254fbba04ca90b679070424424761a53a043b93b612bf

We halen de layers uit de JSON van alpine via jq

Text

Description automatically generated

We halen de layers uit de JSON van static-site. Hier zijn er 5. We zien dat de eerste layer hetzelde is als de alpine layer.

Text

Description automatically generated

In onderstaande dir kunnen we zien dat elke layer een file heeft met een sha256 checksum.

Text

Description automatically generated

We editen de Dockerfile van de static-site en rebuild de image en geef het de naam static-site-2. Nu zijn er 2 extra leyers. Alleen de eerste layer is hetzelfde.

Text

Description automatically generated

ADD files in de Dockerfile voegen waarschijnlijk lagen toe.



**Docker compose**

We builden de todo-app en starten het zoals de static-site. Je kan het zien werken door de poort te forwarden en te bezoeken.

Graphical user interface, text, application

Description automatically generated

In docker-compose.yml waarin 2 containers worden gedefineerd en hoe de image te bouwen en een volume. We runnen de containers via docker-compose up -d. We zien een mysql en een alpine container die aan het runnen zijn.

Graphical user interface, application

Description automatically generated

De app container heeft het ip adres van andere containers niet nodig om met ze te communiceren, alleen de containernamen. We controleren dit door ssh in de applicatie container te gaan en de mysql container te pingen. Getent ahosts mysql om DNS name resolution te zien.

Text

Description automatically generated

Je kan op de website TODO items toevoegen en als je de containers stopped door *docker-compose down*. Na het terug te starten met *docker-compose up -d* zien we dat de TODO items gebleven zijn.