A blue cloud computing symbol

Description automatically generated with medium confidence

**Thierry van Boxel**

**Datum**

**Docker**

**En**

**Kubernetes**

# voorwoord

Met enthousiasme presenteer ik deze motivatie voor ons Docker en Kubernetes project. In een tijd waarin schaalbaarheid, flexibiliteit en efficiëntie cruciaal zijn, streven we ernaar om met deze technologieën een baanbrekende impact te realiseren.

Docker biedt de mogelijkheid om applicaties te containeriseren, waardoor we een consistente omgeving kunnen waarborgen, ongeacht de hostinginfrastructuur. Dit bevordert een naadloze ontwikkeling en implementatie, wat resulteert in kortere releasecycli en verbeterde samenwerking tussen teams.

Kubernetes, als ons orchestratieplatform, stelt ons in staat om deze containers op een geoptimaliseerde en schaalbare manier te beheren. Het dynamisch schalen van applicaties, automatische load balancing en failover-mogelijkheden dragen bij aan een betrouwbare en veerkrachtige infrastructuur.

Dit project heeft als doel niet alleen operationele efficiëntie te verhogen, maar ook innovatie te stimuleren door snellere iteracties en probleemloze implementaties mogelijk te maken. Met Docker en Kubernetes omarmen we een moderne aanpak die de complexiteit van infrastructuur vermindert en tegelijkertijd de prestaties verbetert.

Ik kijken ernaar uit om samen deze reis te maken en geloven dat dit project niet alleen onze technologische mogelijkheden zal verbeteren, maar ook de algehele prestaties zal optimaliseren.

Docker en Kubernetes

[**1. Docker** 4](#_Toc159419579)

[1.1 Wat is Docker? 4](#_Toc159419580)

[1.2 Waarom Docker gebruiken 4](#_Toc159419581)

[1.3 Gebruik van Docker 5](#_Toc159419582)

[**2. Docker netwerken** 6](#_Toc159419583)

[2.1 De 7 netwerken: 6](#_Toc159419584)

[2.2 Waar de netwerken voor dienen. 6](#_Toc159419585)

[2.3 Uitvoering van de voornaamste netwerken. 8](#_Toc159419586)

[Default bridge 8](#_Toc159419587)

[User-define bridge 8](#_Toc159419588)

[Host 10](#_Toc159419589)

[Mac-vlan 10](#_Toc159419590)

[Ip-vlan 11](#_Toc159419591)

[None 12](#_Toc159419592)

[**3. Kubernetes** 13](#_Toc159419593)

[3.1 wat is kubernetes? 13](#_Toc159419594)

[3.2 Hoe werkt het? 13](#_Toc159419595)

[3.3 Master processes van Kubernetes. 14](#_Toc159419596)

[3.4 componenten van kubernetes 14](#_Toc159419597)

[3.4.1 Pods 14](#_Toc159419598)

[3.4.2 Service 15](#_Toc159419599)

[3.4.3 Ingress 15](#_Toc159419600)

[3.4.4 ConfigMap 15](#_Toc159419601)

[3.4.5 Secret 15](#_Toc159419602)

[3.4.6 Volumes 15](#_Toc159419603)

[3.4.7 Deployment 16](#_Toc159419604)

[3.4.7 StatefulSet 16](#_Toc159419605)

[3.4.8 DeamonSet 16](#_Toc159419606)

[**4. Apache** 16](#_Toc159419607)

[4.1 Wat is apache 16](#_Toc159419608)

[4.2 Gebruik van apache 16](#_Toc159419609)

[**5. Uitwerking** 17](#_Toc159419610)

[5.1 uitleg vooraf 17](#_Toc159419611)

[**5.2 Docker** 17](#_Toc159419612)

[5.2.1 Docker installeren. 17](#_Toc159419613)

[5.2.2 image bekomen met Dockerfile. 18](#_Toc159419614)

[5.2.3 Apache-image 19](#_Toc159419615)

[5.2.4 ubuntu rdp image 20](#_Toc159419616)

[5.2.5 image uit repository halen. 21](#_Toc159419617)

[5.2.6 Apache container maken 23](#_Toc159419618)

[5.2.7 ubuntu rdp container maken 23](#_Toc159419619)

[**5.3 kubernetes** 25](#_Toc159419620)

[5.3 Kubernetes en toebehoren installeren. 25](#_Toc159419621)

[5.3.1 Cluster maken. 27](#_Toc159419622)

[5.3.2 pods maken. 29](#_Toc159419623)

[5.3.3 maken van de yaml-files voor de pods 30](#_Toc159419624)

[5.3.4 maken van de pods 35](#_Toc159419625)

# **1. Docker**

### 1.1 Wat is Docker?

Docker is uitgevonden door Solomon hykes en heeft de software gelanceerd in 2013 als een open-source platform voor het automatiseren van de implementatie van een applicatie.

Bij het ontwikkelen van applicaties zijn er vaak complicaties als er gewerkt word met verschillende besturingssystemen, de verschillen aan variabele kunnen een negatieve invloed hebben.

Docker is software dat gemaakt is voor systeembeheerders en ontwikkelaars. Het is een manier voor het ontwikkelen, verschepen en uitvoeren van software door het te beheren in een virtuele omgeving. Docker maakt gebruik van containerisatie-technologie die software gaat verpakken en isoleren waardoor de toepassing minder afhankelijk word van de host. Het is dus betrouwbaarder, eenduidiger en kan eenvoudiger worden verscheept en uitgevoerd op verschillende omgevingen.

Voor ontwikkelaars is dit een eenvoudige manier om applicaties te bouwen, distribueren en schalen ongeacht de host waar de container op draait.

Een container is een geïsoleerde omgeving die zelf over de nodige zaken bezit om de software in de omgeving uit te voeren en beheren, inclusief het besturingssysteem, de runtime, systeemgereedschappen, bibliotheken en instellingen.

### 1.2 Waarom Docker gebruiken

Docker is op verschillende sterke kanten. Door de populariteit heeft het al veel images en zijn deze eenvoudig in containers uit te voeren en beheren. Het is een open-source software die voor iedereen beschikbaar is.

Voordelen:

* Minder heisa met de installatie van de infrastructuur.
* Veilig en betrouwbaar door isolatie.
* Niet afhankelijk van de infrastructuur van de host.
* Gaat zuiniger om met het ram-geheugen.
* Snelle implementatie.
* Goedkope manier van virtualisatie

Nadelen:

* Complexiteit
* Leercurve
* Lichtgewicht
* beveiliging

Docker is niet zo vanzelfsprekend, het heeft een bepaalde hoeveelheid aan voorkennis nodig. Als Docker in een bedrijf gebruikt word is de leercurve groot en is een training soms noodzakelijk.

Het bied daarnaast ook een aantal mogelijkheden aan beveiliging, (wat later nog besproken word) maar dit sluit niet uit dat het niet nodig is om beveiligingsmaatregelen te nemen.

### A diagram of a docker Description automatically generated1.3 Gebruik van Docker

Ondanks Docker gebruikt kan worden op verschillende besturingssystemen word Docker het meest gebruikt bij Linux besturingssystemen en worden ze beheerd van uit de terminal. Voor Windows en MacOS is Docker-desktop gemaakt om het te kunnen beheren van uit een gui. (graphical user interface)

Er zijn twéé verschillende manieren om Docker te gebruiken:

* Docker run
* Docker-compose

Het zijn tools in de wereld van containerisatie die het makkelijker maken om applicaties te bouwen, implementeren en schalen.

Er zijn echter wel verschillen in het gebruik van deze tools. Docker run is volledig gebaseerd op command line terwijl Docker-compose YAML configuratiebestanden leest.

Een ander groot verschil is dat Docker-compose in staat is om meerdere containers te configureren en draaien terwijl Docker run dit maar één per één kan doen.

Het eenvoudigste is dat je Docker containers op een Linux host draait, maar het is mogelijk om de Linux containers op Windows te draaien met de Windows subsysteem voor Linux vanaf Windows 10 of hoger.

Docker is in de loop der tijd beter geworden dan dat men oorspronkelijk voor ogen had. Door de mogelijkheden zijn er optie gekomen om de containers sterker, sneller en efficienter te maken.

Note:

Door deze opties te implementeren word de container meer afhankelijk van het hardware van de hoste .

Hier een aantal opties die je kan toevoegen aan je container:

--add-host voeg een host toe aan ip-mapping

--annotation toevoegen van regels aan een container

--attach toevoegen aan STDIN? STOUT or STDERR

Cpu opties:

--cpu-count cpu count (enkel voor windows)

--cpu-percent cpu percent (enkel voor windows)

--cpu-period limiet cpu CFS (periode)

--cpu-rt-period limiteer cpu real-time in miliseconde.

# **2. Docker netwerken**

Waarom containers

Ook Docker netwerken worden als root uitgevoerd. Bij het toekennen van een netwerk aan een container is het belangrijk om te weten waarvoor de applicatie voor moet dienen en wat het doel is van de container. voorbeeld: als de container extern bereikbaar moet zijn ga je niet de “none” netwerk kiezen en ga je er voor zorgen dat je de nodige beveiligingen meegeeft en de aangepaste poorten. Het is dus belangrijk om eerst stil te staan bij de wwwwwh vragen. (wie, wat, waar, wanneer, waarom en hoe)

## 2.1 De 7 netwerken:

* Default bridge
* User-define bridge
* Host
* Mac vlan
  1. Bridge mode
  2. 802.1g
* Ip vlan l2 en l3 (l=level)
* Overlay
* None

## 2.2 Waar de netwerken voor dienen.

2.2.1 Default Bridge

Word ook wel docker0 genoemd. Dit netwerk bevind zich in een klasse B. wanneer er geen netwerk-type word meegegeven met de container zal het automatisch een IP-adres krijgen van de default bridge netwerk.

2.2.2 User-define Bridge

Dit netwerk heeft ook een klasse B netwerk. Het gebruik van dit netwerk is goed voor personaliseren en isoleren, je blijft in een klasse B maar gaat een ander netwerk aanmaken om het af te zonderen van de Docker default netwerk. Je kan deze netwerken zelf een naam toewijzen.

2.2.3 Host

Het host leeft van de ip-configuraties van de host, het is een soort van applicatie en gebruikt het IP-adres van de host. Voordeel hiervan is dat je geen poorten hoeft toe te kennen, nadeel is dat het afhankelijk is van de host en het niet beveiligd is.

2.2.4 Mac vlan (bridgemode)

Dit netwerk connecteert rechtstreeks met het netwerk die door de router word uitgedeeld. Bij dit netwerk moet je zelf het IP-adres, subnet en gateway meegeven, het is hier dus belangrijk om te weten wat jou netwerk is en welke IP-adressen zijn uitgedeeld. Bij het specifiëren moet je ook de netwerk interface meegeven. (NIC)

Als het netwerk aan een container word toegewezen zal het dan ook een eigen Mac-adress krijgen. Dit kan voor problemen zorgen bij de switch omdat je geen meerdere Mac-adressen om één poort kunt hebben. Dit kan verholpen worden door promiscuous mode aan te zetten.(bij een virtuele machine zet je het aan in de host “en” de machine.

Je kan, of een IP-adres geven als je maar één container in het netwerk wil of als je meerdere containers hebt kan je bij het maken van het netwerk een range meegeven. Let wel op dat je fysieke router geen IP-adressen uitdeelt binnen dat netwerk want de router houd geen rekening met de netwerk dat je hebt gecreëerd.

2.2.5 Mac-vlan (802.1g mode)

Je kan een nieuw sub-interface maken op het netwerk. Zo word er voorkomen dat je niet de zelfde IP-adressen gebruikt als dat de router uitdeelt aan je apparaten. Je maakt dus een eigen netwerk die over de link naar het fysieke netwerk gaat sturen.

2.2.6 Ip-vlan (l2 en l3)

L2

Dit netwerk werkt het zelfde als de mac-vlan bridge mode alleen hier is het voordeel dat het Mac-adres van de host gedeeld word met de containers.

L3

het netwerk gaat hier niet langer afhankelijk zijn van de fysieke router maar van de hoste. De hoste gaat als een router werken voor het netwerk. Dit zorgt er voor dat er geen Broadcast, Arp,… meer is en er dus minder verkeer gaat zijn. het nadeel is dat er geen communicatie meer is naar buiten en naar binnen. Om dit als nog mogelijk te maken moet je aan de router laten weten waar de IP-adressen zich bevinden. (stuur door naar host…)

2.2.7 Overlay

Hier word een laag boven het netwerk gemaakt waar je zelf de regels kunt specifiëren. Dit netwerk word gebruikt voor als je containers op meerdere hoste hebt draaien. Het netwerk zorgt er voor dat jij de orkest meester kan zijn voor de hosts. (soort van kubernetes in docker)

2.2.8 None

De none netwerk kan je gebruiken voor pure isolatie/beveiliging. Het gaat geen IP-adres, mac-adress,… uitdelen. Als je dit netwerk gebruikt zorg je er voor dat enkel de host kan communiceren met de containers.

### 2.3 Uitvoering van de voornaamste netwerken.

Er zijn drie netwerken die je zelf niet hoeft te creëren:

1. Default bridge
2. Host
3. None

Deze netwerken hoef je simpelweg mee te geven bij het maken van het netwerk.

### Default bridge

Voor dit netwerk is geen uitvoering nodig en zal, zoals eerder vermeld automatisch een ip-adres uitdelen als er zelf geen netwerk word meegegeven.

Door het commando “ip addres show” in te geven in de terminal zal je de configuratie te zien krijgen van het netwerk en zal het worden weer gegeven als Docker0

A computer screen shot of a program

Description automatically generated

### User-define bridge

De user-define bridge zit net als de default bridge in een klasse B ip-adres. Voor het maken en toekennen aan een container voer je de volgende stappen uit:

**Creëren**

docker network <naam van netwerk>

A screen shot of a computer

Description automatically generated**controleren**

docker network ls

A computer screen shot of a computer program

Description automatically generatedom te controleren in welk netwerk het netwerk zich bevind voer je volgende commando’s uit:

A screen shot of a computer

Description automatically generateddocker inspect user-defined-network

ip address show

**toewijzen aan een container**

docker run -itd --name mycentos --network user-defined-network centos

bij het maken van de container voeg het de “--network” parameter toe met daaropvolgend de naam van het netwerk.

### Host

**toevoegen aan container**

docker run -itd --name mycentos --network host centos

het commando is het zelfde als bij de user-define bridge. Je geeft bij de parameter het gewenste netwerk mee.

### Mac-vlan

voor dit netwerk te configureren is het belangrijk dat je het lan netwerk kent van je router. Je moet het subnet en gateway meegeven.

A black screen with white text

Description automatically generated**Creëren**

Docker network create -d macvlan --subnet <jou subnet> –gateway <jou gateway> -o parent=<jou Nic> macvlan-network.

Eerste en tweede parameter is om de netwerkconfiguraties mee te geven. Bij de derde parameter geef je mee aan welke fysieke netwerk interface je het gaat vastmaken. (o= options, parent= Nic)

Voor de 802.1g mode kan je achter de Nic de gewenste netwerk geven.

Bv :

A computer screen shot of a network

Description automatically generated**Controleren**

docker network ls

A black screen with white text

Description automatically generated**toewijzen aan een container**

docker run -itd –network macvlan --netwerk --ip 192.168.80.200 --name mycentos centos

bij het toewijzen aan en containergeef je nog een aantal parameters mee:

* Naam van gemaakte netwerk
* Een ip-adres dat niet gebruikt word
* Naam van de container
* Image

Het nadeel aan dit netwerk is dat elke container een eigen mac-adres heeft en de containers delen de zelfde switch poort van de host. Niet alle switchen kunnen meerdere mac-adressen op één port verwerken.

Je kan dit omzeilen door de promiscuous uit te schakelen op de host en in de instellingen in de virtuele machine.

### 

### Ip-vlan

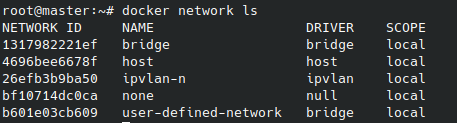
L2

De configuratie is het zelfde als mac-vlan alleen geef je bij de driver (-d) mee dat het over de ipvlan gaat.

A screenshot of a computer

Description automatically generated**Creëren**

Docker network create -d ipvlan --subnet <jou subnet> –gateway <jou gateway> -o parent=<jou nic> ipvlan-network

**controleren**

Docker network ls

A black screen with white text

Description automatically generated**Toewijzen aan container**

L3

A black background with white text

Description automatically generated**creëren**

docker network create -d ipvlan –subnet 192.168.38.0/24 -o parent=eth0 -o ipvlan\_mode=l3 <name>

meerdere netwerken maken kan enkel bij de configuratie door een extra parameter mee te geven met de gewenste netwerk.

Bij ipvlan l3 kunnen de containers wel met elkaar communiceren maar niet met de buitenwereld. Je kan dit mogelijk maken om in de router de nodige configuraties aan te passen.

### None

Dit netwerk werkt het zelfde als de default netwerk. je hoeft juist de parameter mee te geven als je de container aanmaakt.

**Creëren**

# **3. Kubernetes**

## 3.1 wat is kubernetes?

Kubernetes ook wel bekend als k8s is een Griekse benaming voor ‘stuurman’ van een schip. De software is tot stand gekomen bij google als een open-source systeem om grote groepen gecontaineriseerde applicaties te beheren. Engineers hadden de software ontwikkeld omdat er dringend behoefte was aan het ondersteunen van de groeiende hyperscale infrastructuur. De containers die op de hosts draaien worden beheerd vanuit één machine (meestal de server) die de containers gaat aansturen.

Door van kubernetes een open-source te maken waren ze in staat om meer kennis te vergaren en zo de software verbeteren.

## 3.2 Hoe werkt het?

kubernetes maakt orkestratie mogelijk, dit betekent een automatiseren en mannagment van containers. Eigen aan containers is dat het een autonoom stuk code bevat die zelden onveranderlijk zijn en kunnen binnen enkele seconden worden opgestart. (Manueel of autonoom)

Kubernetes start de containers op, zorgt voor load balans, failovers, routing van netwerkverkeer en upscaling/downscaling van alle resources.

De sterkte van kubernetes is dat het ingrijpt als o.a een container vastloopt, het zorgd dan tijdelijk voor een andere container zodat er weinig of geen probleem merkbaar is.

Wat kubernetes uniek maakt is dat het de menselijke factor als zwakste schakel uitschakelt. Stel dat je een paar honderd containers hebt draaien, Als je deze allemaal manueel zal beheren is een fout niet uit te sluiten. Door deze processen volautomatisch te laten lopen kan je zulke fouten voorkomen, daarnaast zal het je ook veel tijd besparen.

Door middel van clustering (nodes met elkaar verbinden) is de impact op systeemfouten veel minder en is het eenvoudiger om ze te schalen.

Voordelen:

* Eenvoudig uitvoeren en beheer van software
* Schaling
* Automatisering
* Probleemoplossend
* Updaten zonder downtime

Nadelen:

* De leercurve is lastig
* Het forceert je om met een bepaalde structuur te werken
* De applicatie moet geschikt zijn voor kubernetes

## 3.3 Master processes van Kubernetes.

De master processes draaien enkel op de masternode. Het zijn tools dat er voor gaat zorgen dat kubernetes aanspreekbaar/beheerbaar is en dat de data beheerd en opgeslagen kan worden.

* Api-Server
* Scheduler
* Controller manager
* etcd

## 3.4 componenten van kubernetes

Kubernetes heeft verschillende tools om te componenten (word later besproken) aan te spreken.

Tools:

* Kubectl
* Kubelet
* Kubeadm
* Kube-proxy

Al de componenten worden geconfigureerd in een yaml file. In deze file’s worden de componenten geconfigureerd en de specificaties van de pods en containers omschreven.

In de file heb je 3 verschillende secties:

* Metadata
* Specifications
* Status (word automatis geconfigureerd door Kubernetes)

Kubernetes haalt de status informaite uit de etcd master proces waar alle data van de kubernetes componenten worden omschreven.

### 3.4.1 Pods

Een pod is een omgeving of een laag bovenop een container met een aantal specificaties. De pod dient als middel om de container of image te beheren. Bv, als een container uit valt zal de pod de container herstarten of vervangen om er voor te zorgen dat de gebruikservaring optimaal blijft.

Een pod is gemaakt om één te beheren. Het is mogelijk om meerdere containers in één pod te steken maar dit is niet de bedoeling.

De pods moeten met elkaar en met de buitenwereld met elkaar kunnen communiceren. Kubernetes heeft hiervoor een virtueel netwerk. In Docker krijgt de container een ip-adress maar als we werken met kubernetes word de netwerk van de container vervangen door die van de pod. Elke pod krijgt dan een eigen Lan ip-adress.

Elke keer dat een pod defect is zal het ip-adress dat gelinkt is aan de pod ook verdwijnen. Om dit te voorkomen is het Service component uitgevonden.

### 3.4.2 Service

Het Service component is een statisch ip-adress/permanent ip-adress dat word toegewezen aan een pod. Op deze manier verhelp je dat als een pod niet meer functioneerd/defect de service of het ip-adress blijft.

Er zijn 2 soorten services:

* Internal
* External

Sommige applicaties moeten berijkbaar zijn via de webbrowser. Met de external service maak je dit mogelijk.

De URL ziet er dan als volgd uit. [http://192.168.30.204:24021](http://192.168.30.204:24021/) (ip-adress + poort) dit is niet veilig en ook niet handig. Om dit te beveiligen en een gepast domein naam toe te wijzen bestaat het Ingress Component.

### 3.4.3 Ingress

Ingress gaat als een soort proxy-server werken. Het gaar de domein namen, ssl/tls certificaten beheren en de portforwardings doen, dus als je het gewenste domeinnaam ingeeft zal het eerst binnenkomen via Ingress om het dan door te sturen naar Service als alle configuraties juist zijn.

### 3.4.4 ConfigMap

Het bevat externe configuratie zoals url’s, gebruikersnamen en wachtwoorden of andere zaken die mogelijks kunnen veranderen.

### 3.4.5 Secret

Configmap is niet beveiligd. Hiervoor bestaat het Secret component. Het word gebruikt voor data dat beveiligd moet worden. Het gebruikt een 64bit encryptie.

Omdat kubernetes niet voor beveiliging zorgd moet deze tool nog beveiligd worden met 3party tools.

### 3.4.6 Volumes

Als een pod defect gevallen is zal de data ook verloren gaan. Hiervoor heeft kubernetes de volume component in leven geroepen. De tool gaat er voor zorgen dat op de fysieke machine (waar containers draaien of de master) een opslag word voorzien waar de data in opgeslagen kan worden.

### 3.4.7 Deployment

Deplyment is een soort blauwdruk van een pod. Je gaat specifieren hoeveel replicas je hebt, opschalen of downschalen. Het is niet mogelijk om data de dupliceren met deployment. Dit zou willen zeggen dat het de opslag van de applicatie gaat delen. (dit is niet mogelijk zonder een tool)

### 3.4.7 StatefulSet

Deze tool word gebruikt voor het beheren van data en stateful apps als:

* mySQL
* elastic
* mongoDB

het word dus gebruikt voor database’s.

### 3.4.8 DeamonSet

Deamenset is een onderdeel dat er voor zorgd dat op elk knooppunt in een kluster altijd ene pod beschikbaar is. het verschil met Deployment is dat op basis van de vraag een aantal pods kan geverifieerd worden. Het zorgd er dus voor dat er altijd minstent 1 pod op elk knooppunt aanwezig is.

# **4. Apache**

### 4.1 Wat is apache

Apache is een http webserver uitgevonden door developers als een open-source cross-platform. Het was voornamelijk ontwikkeld om te draaien op linux besturingsystemen maar overtijd is het ook uitgerold voor andere besturingssystemen.

Gebaseerd op de ncsa httpd server begon de ontwikkeling van apache in 1995 na werk aan de ncsa code gesylleerd. Apache speelde een grote rol in de ontwikkeling van het www. (world wide web)

Samen met nginx had apache in 2022 het meeste marktaandeel.

### 4.2 Gebruik van apache

Het is mogelijk om Apache te gebruiken op Windows, maar zoals al eerder vermeld is Apache ontwikkeld om gebruikt te worden op Linux besturingssystemen. Apache kan men op verschillende manieren gebruiken. Het kan geïnstalleerd worden op een besturingssysteem of n een container. (Wat we in dit project gaan uitvoeren.

Het configureren en beheren van een Apache server met Docker word gedaan “in” de container. Daar kan men allerlei configuraties uitvoeren of de websites creëren.

# **5. Uitwerking**

## 5.1 uitleg vooraf

In dit project gaan we Docker installeren op nodes. nodes zijn de besturingssystemen waar Docker op gaat draaien en beheerd worden met Kubernetes, dit kan Windows, Chrome, Mac of Linux (met of zonder grafische omgeving) zijn maar in dit project werken we met Linux. In dit project gaan we 3 workernodes maken waar Docker op geïnstalleerd word met de containers.

Op de masternode zal Kubernetes draaien en worden de containers in pods beheerd.

De images van de Docker container gaan we op 2 verschillende manieren bekomen.

* Door ze zelf te maken met een Dockerfile.
* Door ze te halen uit de repository

## **5.2 Docker**

### 5.2.1 Docker installeren.

Het installeren van docker is niet op alle besturingssystemen het zelfde. Op linux verties kan de installatie verschillen maar in de meeste gevallen moet je de repository, certificaat en dependensies toevoegen. In dit project gaan we docker installeren op Ubuntu-studio, deze is al voorzien van de repository en certificaat. Je kan Docker snel en eenvoudig installeren.

Installeren van Docker:

* Apt update
* Apt install docker.io -y
* Systemctl enable docker

Controlleren of Docker actief is:

* A screen shot of a computer

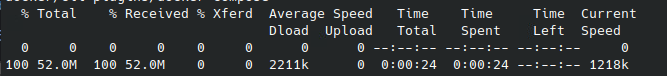
  Description automatically generatedSystemctl status docker
* Systemctl enable docker

Met dit commando kan je controleren of Docker actief is.

**Docker compose**

De installatie van docker-compose werkt een beetje anders.

* Mkdir –p ~/.docker/cli-plugins/
* curl -SL https://github.com/docker/compose/releases/download/v2.18.1/docker-compose-linux-x86\_64 -o ~/.docker/cli-plugins/docker-compose



* chmod +x ~/.docker/cli-plugins/docker-compose
* Docker compose version



### 5.2.2 image bekomen met Dockerfile.

Om een image te maken moeten we eerst 2 zaken voorzien:

* Een directory
* Een bestand genaamd dockerfile.

Voor de dockerfile is het belangrijk om te onthouden dat je maar slechts één dockerfile in één Directory ka hebben. Je kan wel een ander dockerfile maken in een ander directory maar dit is niet aan te raden.

We gaan een paar images aanmaken:

* Apache image
* Alpine image
* Ubuntu image

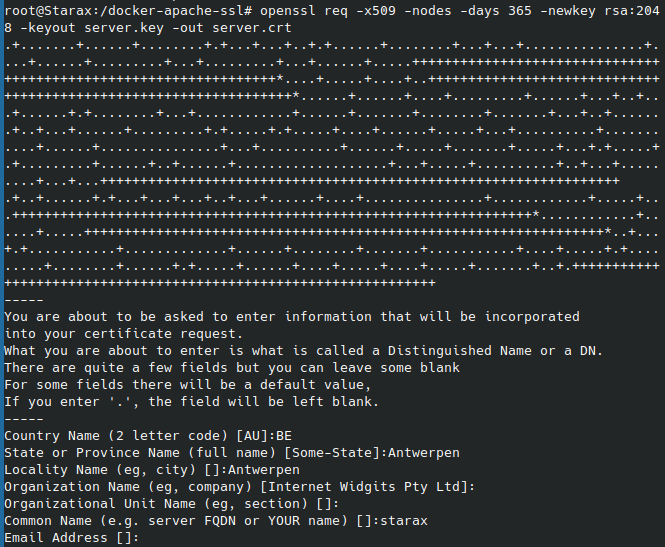
Directory maken:

* mkdir -p docker/docker-images/docker-apache-ssl
* cd docker/docker-images/docker-apache-ssl

voor dat we een dockerfile aanmaken gaan we eerst een ssl-certificaat genereren om deze nadien toe te voegen aan de image.

Voer dit uit in de directory waar je de dockerfile gemaakt hebt, dit gaat het makelijker maken voor het maken van de image.

* openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout server.key -out server.crt



geef de gevraagde informatie in.

### 5.2.3 Apache-image

geef volgende zaken in het bestand. (hoofdletter gevoelig)

dockerfile:

* nano dockerfile

# Gebruik een bestaande Apache image als basis

FROM httpd:latest

# Kopieer de aangepaste index.html naar de Apache-document root

COPY ./index.html /usr/local/apache2/htdocs/

# Kopieer SSL-certificaat en sleutel naar /usr/local/apache2/conf/

COPY ./server.crt /usr/local/apache2/conf/server.crt

COPY ./server.key /usr/local/apache2/conf/server.key

# Configureer Apache om SSL in te schakelen

RUN sed -i \

-e 's/^#LoadModule\ ssl\_module/LoadModule\ ssl\_module/' \

-e 's/^#LoadModule\ socache\_shmcb\_module/LoadModule\ socache\_shmcb\_module/' \

-e 's/^#Include\ conf\/extra\/httpd-ssl.conf/Include\ conf\/extra\/httpd-ssl.conf/' \

/usr/local/apache2/conf/httpd.conf

# Expose poorten 80 en 443

EXPOSE 80 443

Sla het bestand op.

Zorg er voor dat je met nano een index.html-file maakt of weizig het pad naar de locatie van jou script.

Voor een simpel script han je altijd beroep doen op chatgpt of github.

Nu gaan we de image bouwen:

* docker build -t mijn-apache-ssl-image .

### 5.2.4 ubuntu rdp image

je hoeft je dockerfile niet aan te passen. je kan een nieuwe dockerfile maken op voorwaarde dat deze zich in een andere directory bevind.

Eerst maken we een nieuwe directory aan:

* mkdir -p docker/docker-images/xrdp
* cd docker/docker-images/xrdp
* nano dockerfile

geef volgende input in de dockerfile:

# Gebruik de officiële Ubuntu-image als basis

FROM ubuntu:latest

# Installeer xrdp en xfce (een lichte desktopomgeving)

RUN apt-get update && \

   DEBIAN\_FRONTEND=noninteractive apt-get install -y lubuntu-desktop

RUN rm /run/reboot-required\*

RUN useradd -m thierry -p $(openssl passwd 1234)

RUN usermod -aG sudo thierry

RUN apt install -y xrdp

RUN adduser xrdp ssl-cert

# Installeer sudo voor de gebruiker 'thierry'

RUN apt-get update && \

   apt-get install -y sudo && \

   rm -rf /var/lib/apt/lists/\*

RUN sed -i '3 a echo "\

export GNOME\_SHELL\_SESSION\_MODE=Lubuntu\\n\

export XDG\_SESSION\_TYPE=x11\\n\

export XDG\_CURRENT\_DESKTOP=LXQt\\n\

export XDG\_CONFIG\_DIRS=/etc/xdg/xdg-Lubuntu:/etc/xdg\\n\

" > ~/.xsessionrc' /etc/xrdp/startwm.sh

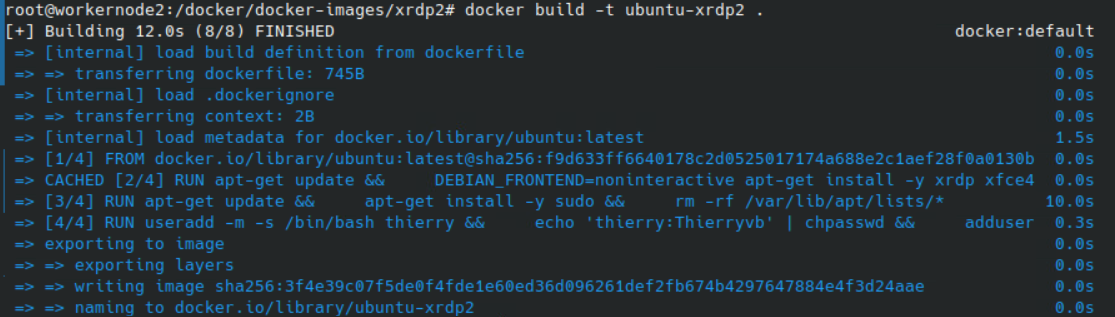
# Open poort 3389 voor xrdp

EXPOSE 3389

# Start xrdp bij het starten van de container

CMD service xrdp start ; bash

Nu gaan we de image maken:

* - docker build -t ubuntu-xrdp2 .

### 5.2.5 image uit repository halen.

Een image halen uit de Docker repository is vrij eenvoudig. Men kan de laatste verties van een besturingssystemen, webservers, apps en nog veel meer halen.

Gebruik volgende commando’s:

* docker pull mongo:latest
* docker images



als we image willen gebruiken moeten eerst onze image uploden naar onze docker repository. (als je geen repository hebt kan je in de link meer uitleg vinden)

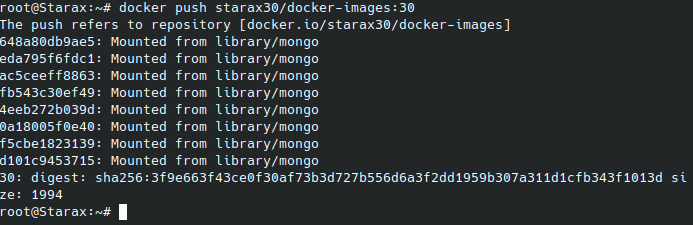
link: <https://docs.docker.com/docker-hub/repos/create/>

eerst moeten we inloggen. Daarna gaan we de image taggen en vervolgens uplode naar de reposytory.

uploaden naar de repository:

* docker login (geef gebruikersnaam en wachtwoord van de hub)
* docker tag image:tag docker hub/repository:tag

bv; docker tag mongo:31 starax30/docker-images:31

* docker push starax30/docker-images:31 (de repository in de tag van de image)

met dit pad kan je later kubernetes de image laten halen uit “jou” repository en kan je de image overal opvragen met het docker pull commando, hub, repository en de tag.

Nota:

Als je een image aan het magen bent met een docker-file kan je hier ook ineens een tag aan toevoegen door op het einden de “:tag” aan de string toe te voegen.

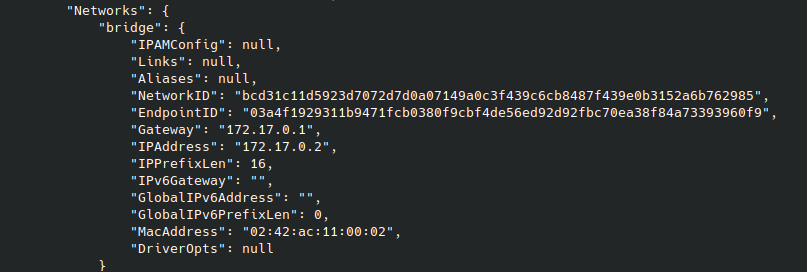
Bv; docker build -t alpine-image:40

### 5.2.6 Apache container maken

Voor het maken van een container is het belangrijk om eerst goed na te denken waarvoor het zal dienen, welke poorten je wil toekennen of aan welke specificaties de container moet voldoen.

Voor de apache-container gaan we wel poorten meegenen omm kubernetes naar de hostpoort door te verwijzen en aan te geven aan welk protocol er voldaan moet worden. We gaan geen netwerk meegeven, dit doen we in kubernetes. (de parameters kan je terugvinden in het hoofdstuk van docker.

Container maken en bekijken/controleren:

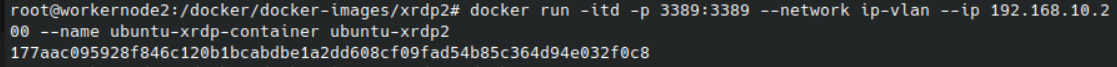
* docker run -itd -p 8080:443 --name apache-ssl-container mijn-apache-ssl-image
* docker ps
* docker inspect apache-ssl-container (specs van de container)

zoals we al weten heeft docker zelf een ip-adress toegekend aan de container omdat wij dit niet gedaan hebben. Je kiest je netwerk-configuraties naargelang de noden, veiligheid of beschikbaarheid.

Om de website in de container aan te spreken surf je naar https://”container-ip”, in de webbrowser.

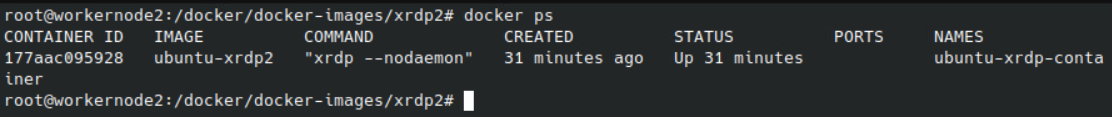
### 5.2.7 ubuntu rdp container maken

Bij het maken van de container gaan we een aantal parameters meegeven en gaan we een ipvlan ip-adres toevoegen aan de container, (zie hoofdstukuitwerking van de netwerken) de rdp poort, gewenste naam van container en de image:

* Docker run -itd -p 3389:3389 –network ip-vlan –ip 192.168.10.200 –name ubuntu-xrdp-container ubuntu-xrdp2

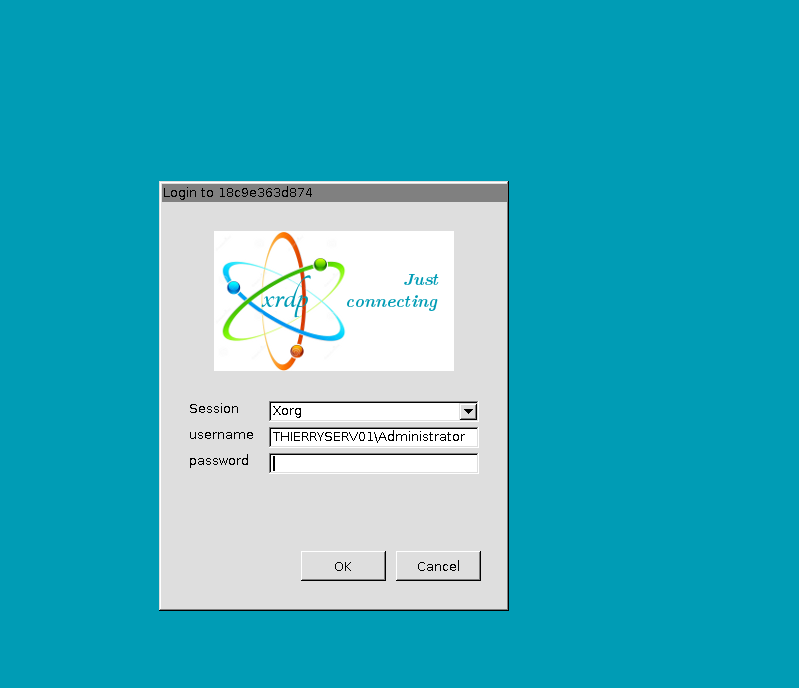
De container word gemaakt en het krijgt een je krijgt het unieke id te zien van de container.

Gebruik het Docker commando om te kijken of de container is gemaakt.

* Docker ps

Hier krijg je nog meer informatie te zien van de container

Het enige dat nog gedaan moet worden is de portforwardings doen in de firewall en op een client inloggen met extern bureaublad.

je krijgt dit scherm te zien. Log je in met je gebruikersnaam en wachtwoord en je hebt je lubuntu desktop omgeving. Wat je wel moet weten is dat niet alle gegevens worden opgeslagen en installaties beperkt zijn omdat docker een light-gewicht tool blijft.

## **5.3 kubernetes**

### 5.3 Kubernetes en toebehoren installeren.

Voor Kubernetes geinstalleerd kan worden moet eerst de swapfile uigeschakeld worden. (op alle node’s)

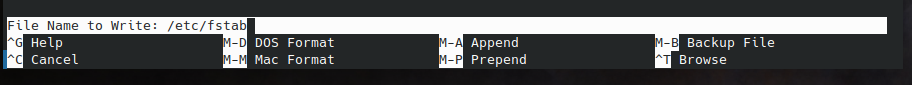
Voer volgende commando’s uit:

* Swapoff -a
* A screenshot of a computer program

  Description automatically generatedNano /etc/fstab

In het bestand zet je een # voor de swapfile.

Gebruik “ctrl + x” om op te slaan “y” om te bevestigen en vervolgens druk je op enter.



Voor Kubernetes geinstalleerd kan worden moeten we eerst de repository-key ophalen. Dit doen we met curl. (Als curl al geinstalleerd is kan je stap 1 overslagen) Deze commando’s voeren we enkel uit op de mastenode.

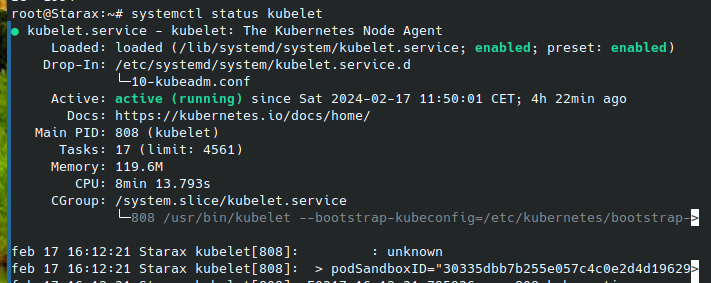
Voer volgende commando’s uit op alle nodes:

* apt install apt-transport-https curl -y
* curl -s <https://packages.cloud.google.com/apt/doc/apt-key.gpg> | apt-key add
* apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial mai  
  n"
* Apt update
* Apt install kubeadm kubelet kubectl kubernetes-cni -y

Controlleer en configureer met volgende commando’s de tools om hinder te voorkomen bij het

maken van de cluster:

* systemctl start kubelet
* systemctl enable kubelet
* systemctl status kubelet (status bekijken)



Kubelet is medeverantwoordelijk voor het aanmaken van de cluster samen met kube-proxy die in staat voor het netwerken van de cluster. (kube-proxy is in deze installatie standaard mee geinstalleerd)

### 5.3.1 Cluster maken.

We gaan er nu een join token kreëren om deze vervolgend in de workernode’s in te geven. Met de join-token kunnen de workernodes deel nemen aan de cluster.

Voer volgende commando’s uit:

* Kubeadm init
* Kubeadm <join-token> --ignore-preflight-errors=all

A screen shot of a computer

Description automatically generated

In de output krijg je het join-token te zien. Hiermee kan de de andere node’s die in het zelfde netwerk zitten deelnemen aan de cluster.

Je kopieert de token samen met het “kubeadm” commando en voert dit uit in de node die aan de cluster deel moet nemen.

Nota:

Installeer eerst Docker voordat je de node’s deel laat nemen.

Om gebruik te maken moeten volgende commando’s uitgevoerd worden als gewone gebruiker:

* mkdir -p $HOME/.kube
* sudo cp -i /etc/Kubernetes/admin.conf $HOME/.kube/config
* A screen shot of a computer

  Description automatically generatedsudo chown $(id -u):$(id -g) $HOME/.kube/config

A screenshot of a computer program

Description automatically generatedmet het “kubectl get nodes” kan je nu kijken naar de status van de kluster.

de nodes staan nu in de “NotReady” status. Dit is omdat we de containernetwerken moeten toevoegen. dit gaan we doen met calico. We geen de yaml-file van het internet trekken en installeren.

container netwerken toevoegen:

* kubectl apply -f <https://docs.projectcalico.org/manifests/calico.yaml>

A screen shot of a computer code

Description automatically generatednu kunnen we weer met het “kubectl get nodes” commando gaan kijken naar de status van de cluster.

### 5.3.2 pods maken.

We gaan starten met het maken van de Apache pod. Voor deze pod gaan we eerst met Kubernetes voor een aantal zaken voorzien.

Stel je hebt een web-pagina waar informatie moet worden opgeslagen, kubernetes gaat hier niet standaard voor zorgen. we hebben dus een database nodig om de gegevens weg te kunnen schrijven en op te slagen. Kubernetes zelf houd geen data bij maar dit kan je wel verwezelijken door een virtuele database in een pod te laten draaien. Je kan dit doen met: MySQL, elastic of mongodb. Wij gaan werken met mongodb.

Kubernetes word geconfigureerd op de api-server/control-plane met behulp van yaml-files waar we alle configuraties in gaan specifieren.

Eerst gaan we alle yamlfiles maken om ze nadien de deployen.

Installeer visual studiocode met de tools om een kubernetes yaml-file te kubben maken. Visual studiocode gaat helpen bij het voorkomen van fouten in de yaml-file’s. (yaml-files zijn gevoelig aan fouten)

Visualstudio installeren: (op de masternode)

* apt install snapd
* snap install --classic code
* apt update

open visual sudiocode en installeer de extenties. De extenties gaan helpen bij het maken van de yaml-file’s.

extentions installeren:

A screenshot of a computer

Description automatically generatedkubernetes

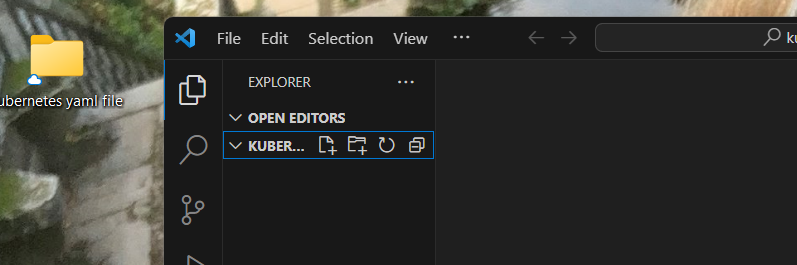
A screenshot of a computer

Description automatically generatedyaml

### 5.3.3 maken van de yaml-files voor de pods

Een yaml-file kan je maken op 2 manieren. Je kan een bestand aanmaken met nano of vi en de code plakken in het bestand of je kan met een absoluut pad doorverwijzen naar de yaml-files. Wij gaan voor optie 2 kiezen.

Om dit te doen maak je een map aan op het bureaublad en je sleept dit in de code editor. Als je dit gedaan hebt kan je in de editor bestanden aanmaken en deze zullen opgeslagen worden in de map die je hebt aangemaakt op je bureaublad.



**Secret component**

Voor dat we een yaml-file moor mongo service gaan maken gaan we er eerst voor zorgen dat een yaml-file voor Mongo-secret gemaakt is. dit gaar er voor zorgen dat de nodige data beveiligd word.

Yaml-file

apiVersion: v1

kind: Secret

metadata:

  name: mongo-secret

type: opaque

data:

  mongo-user: bW9uZ291c2Vy

  mongo-password: bW9uZ29wYXNzd29yZA==

de gebruiker en het wachtmoord gaan we genereren met de volgende commando’s in cmd als administrator:

* echo -n mongouser | base64
* echo -n mongopassword | base64

**Mongodb**

Het yaml-file voor mongodb. Hier gaan we de nodige forwardings doen en verwijzen naar mongo-secret.

Met dit yaml-file gaan wede mongo-database maken. We gaan maar 1 pod maken om conflicten te vermeiden. Het is perfect mogelijk om meerdere pods te magen maar dan moeten we de configuratie aanpassen en meegeven hoe de data weggeschreven moet worden.

apiVersion: apps/v1

kind: Deployment

metadata:

  name: mongo-deployment

  labels:

    app: mongo

spec:

  replicas: 1

  selector:

    matchLabels:

      app: mongo

  template:

    metadata:

      labels:

        app: mongo

    spec:

      containers:

        - name: mongodb

          image: mongo:5.0

          ports:

            - containerPort: 27017

          env:

            - name: MONGO\_INITDB\_ROOT\_USERNAME

              valueFrom:

                secretKeyRef:

                  name: mongo-secret

                  key: mongo-user

            - name: MONGO\_INITDB\_ROOT\_PASSWORD

              valueFrom:

                secretKeyRef:

                  name: mongo-secret

                  key: mongo-password

---

apiVersion: v1

kind: Service

metadata:

  name: mongo-service

spec:

  selector:

    app: mongo

  ports:

    - protocol: TCP

      port: 27017

      targetPort: 27017

**het maken van de apache pod**

hoor het maken van deze pod hebben we een aantal zaken nodig van de vorige yaml-files.

We gaan verwijzen naar de mongodb database zodat kubernetes weet dat de data die doorgevoerd word op de apache server naar de database moet weggeschreven worden. We gaan de dockercontainer specificaties meegeven en verwijzen naar de services.

In dit yaml-file kan je ook de gewenste aantal pods meegeven.

Pas volgende zaken aan:

* naam bij metadata
* naam app
* replicas (naar de gewenste pods
* containerport
* image (naar jou image)

apiVersion: apps/v1

kind: Deployment

metadata:

  name: webapp-deployment

  labels:

    app: webapp

spec:

  replicas: 3

  selector:

    matchLabels:

      app: webapp

  template:

    metadata:

      labels:

        app: webapp

    spec:

      containers:

        - name: webapp

          image: httpd:latest

          ports:

            - containerPort: 443

          env:

            - name: USER\_NAME

              valueFrom:

                secretKeyRef:

                  name: mongo-secret

                  key: mongo-user

            - name: USER\_PWD

              valueFrom:

                secretKeyRef:

                  name: mongo-secret

                  key: mongo-password

            - name: DB\_URL

              valueFrom:

                configMapKeyRef:

                  name: mongo-config

                  key: mongo-url

---

apiVersion: v1

kind: Service

metadata:

  name: web-service

spec:

  selector:

    app: webapp

  ports:

    - protocol: TCP

      port: 443

      targetPort: 30100

**service voor loadbalancer**

met dit yaml-file gaan we er voor zorgen dat we een loadbalencer hebben en de pod extern bereikbaar kunnen maken.

Pas volgende zaken aan:

* naam bij metadata
* port
* targetport
* naam van de app

apiVersion: v1

kind: Service

metadata:

  name: apache-service

spec:

  type: LoadBalancer

  port:

  - name: https

    port: 443

    protocol: tcp

    targetPort: 443

  selector:

    app: apache

  sessionAffinity: None

### 5.3.4 maken van de pods

Met het kubectl apply -f commando gaan we de pods aanmaken.

Met volgende commando’s gaan we de pods maken:

* kubectl apply -f "/home/masternode/Desktop/docker en kubernetes script folder/kubernetes yaml scripts/mongo.yaml"
* kubectl apply -f "/home/masternode/Desktop/docker en kubernetes script folder/kubernetes yaml scripts/webapp.yaml"
* kubectl apply -f "/home/masternode/Desktop/docker en kubernetes script folder/kubernetes yaml scripts/mongo-secret.yaml"
* kubectl apply -f "/home/masternode/Desktop/docker en kubernetes script folder/kubernetes yaml scripts/service.yaml"

om te zien of te pods hemaakt zijn kan je het volgende commando gebruiken en dan krijg je de nodige informatie te zien:

* kubectl get allA screenshot of a computer screen

  Description automatically generated

hier kan je zien dat de service, deployment en de pod met de apache image gemaakt zijn.

# Conclusie

**Realisatie van Docker en Kubernetes Project**

De voltooiing van het Docker en Kubernetes project markeert een mijlpaal in onze streven naar een moderne, schaalbare en efficiënte infrastructuur. Door het integreren van Docker-containers en Kubernetes-orchestratie hebben we niet alleen onze applicatie-implementatie gestroomlijnd, maar ook de flexibiliteit, betrouwbaarheid en prestaties van onze gehele omgeving verbeterd.

Met de containerisatie van applicatiecomponenten hebben we een geïsoleerde en consistente uitvoeringsomgeving gecreëerd, wat de ontwikkelingsworkflow heeft vereenvoudigd en versneld. Kubernetes heeft ons in staat gesteld om deze containers op een geoptimaliseerde en schaalbare manier te beheren, waardoor we dynamisch kunnen reageren op fluctuerende werklasten.

Het externe beschikbaar maken van onze applicatie met een specifieke domeinnaam heeft de toegankelijkheid vergroot, en door services te integreren en MongoDB-storage toe te voegen, hebben we een robuuste en duurzame infrastructuur neergezet. Het implementeren van continue integratie en continue levering (CI/CD) heeft geleid tot snellere releasecycli en een verbeterde samenwerking tussen ontwikkeling en operaties.

Deze technologische vooruitgang heeft niet alleen bijgedragen aan de verbetering van onze operationele efficiëntie, maar heeft ook de basis gelegd voor toekomstige innovaties en groei. Het vermogen om snel te schalen, de betrouwbaarheid te waarborgen en de onderhoudskosten te verminderen, positioneert ons bedrijf voor een competitief voordeel in een steeds veranderend technologisch landschap.

We hebben niet alleen technische competenties ontwikkeld, probleemoplossend vermogen en een dieper begrip van moderne DevOps-praktijken. Dit project is niet alleen een succesvolle implementatie van nieuwe technologieën, maar ook een bewijs van ons vermogen om te evolueren en te innoveren in lijn met de eisen van de markt.

De voltooiing van dit project markeert niet het einde van onze reis, maar eerder het begin van nieuwe mogelijkheden en uitdagingen. We kijken ernaar uit om voort te bouwen op dit fundament en blijven streven naar technologische excellentie en succes.

# Bibliografie

Ubuntu desktop.( <https://www.youtube.com/watch?v=0rl5145aEMk>)

Docker Documentation. (https://docs.docker.com/)

Kubernetes. <https://www.youtube.com/watch?v=7bA0gTroJjw&t=1204s>)

Kubernetes cource. (https://www.youtube.com/watch?v=s\_o8dwzRlu4&t=3653s)

Kubernetescluster. (https://www.youtube.com/watch?v=\_WW16Sp8-Jw)

Kubernetes Documentation. (https://kubernetes.io/docs/)

Apache HTTP Server Documentation. (https://httpd.apache.org/docs/)

MongoDB Documentation. (https://docs.mongodb.com/)