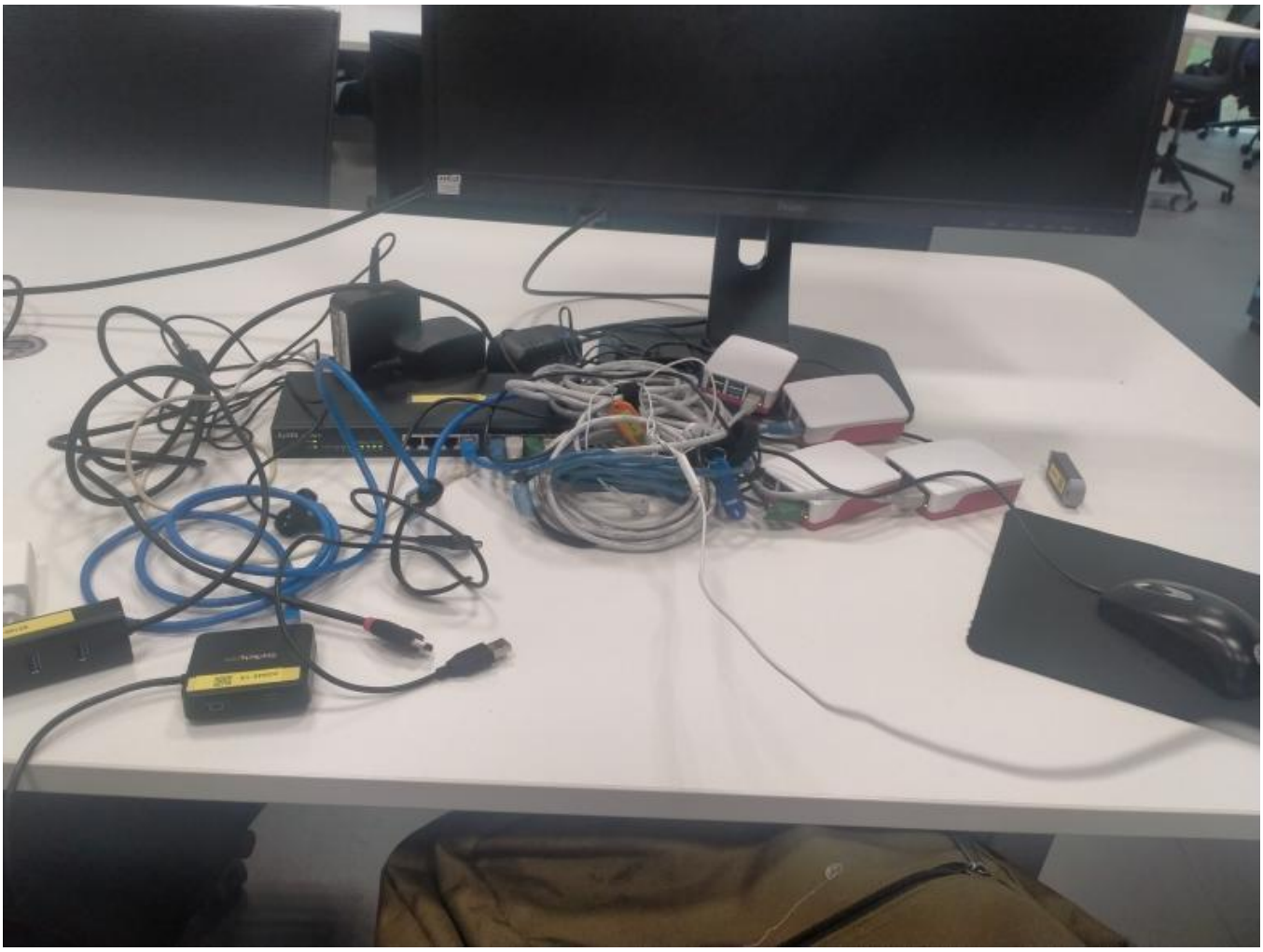
Adam Nunes

Challenge techniek cluster

Kubernetes Cluster



Inhoudsopgave

1. 4 Pi's
2. Omhulsing
3. micro sd x 4
4. Sd kaart installer
5. Voeding
6. Switch
7. UTP kabel
8. USB power hub
9. Virtuele router/ fysieke router

Stap 1: headless instelling

1. Download de Pi imager en selecteer Raspberry pi os lite → click vervolgens op write zodat de os in de sd kaart terecht komt.
2. Vervolgens plaats je de sd in de Pi en na 5 minuten haal je de pi van het stroom af en plaats je de SD kaart opnieuw in de Pc.i
3. Ga naar de boot driver op de SD → click het **cmdline.txt** bestandje → scroll naar het einde van de lange zin en plaats het volgende commando. **cgroup_memory=1 cgroup_enable=memory.**
4. Na het volgen van stap 3 moet aan het eind van diezelfde regel het ip adres van de pi staan en de default gateway. Vervolgens noteer je de naam van je Pi, de interface en zet je auto configuratie uit
5. Dit is een voorbeeld van de volledige zin die in het bestand gekopieerd dient te worden. " **cgroup_memory=1 cgroup_enable=memory ip=192.168.1.43::192.168.1.1:255.255.255.0:rpiname:eth0:off** "
6. Open het config.text bestandje. En aan de onderkant hiervan paste je het volgende commando **arm_64bit=1**

Toelichting Stap 1.2. Wanneer we dit doen maakt de pi een boot driver aan daarin willen we een command toevoegen die de Pi verteld dat het containers en K3s moet laten werken bij het opstarten.
Toelichting Stap 1.4. De interface laat de pi weten welk netwerk kaart gebruikt wordt. Verder staat auto configuratie uit omdat deze aanpassingen vanaf de boot moeten worden toegepast.

Toelichting Stap 1.6. Dit commando zorgt ervoor dat de 64 bit versie wordt gedownload

7. Open powershell en type in **I:** deze letter komt overeen met de letter dat genoteerd staat in de boot drive. Type **new-item ssh**
8. Plaats de sd kaart terug in de PI . Ping de ip addres die eerder was genoteerd in de config file.
9. Schrijf ssh pi@1.1.1.1 (gekozen Ip adres)

Stap 3 - Ssh verbindinding zonder fysieke router

De bovenstaande instrunties aangaande ssh verbinding zijn alleen mogelijke met een **fysieke router !**

Wanneer een digitale router gebruikt moet worden gaat dat als volgt.:

1. Verbind de Pi host met een utp kabel (als de laptop geen utp kabel aansluiting heeft kan een usb – utb converter gebruikt worde.)

2. Ga naar https://www.pfsense.org/download/ en download de pfsense ISO

3. Open VMWare → click op new virtuel machine

4. Vink aan Typical → Installeer de ISO → Geef 20 GB disk size → Finish

5. Voeg 1 netwerkadapter toe en zet het op vmnet0

5. In Pfsense, geef de wan kant DHCP en de lan kant een statische I

8. In deze tutorial word het volgende ip adres gebruikt 172.16.1.254. Dit ip is expres gekozen omdat het een privé ip adres is. Verder heb ik bewust niet voor 192.168 gekozen omdat dit anders in conflict kan komen met het seclab

9. Doe rechtermuisklik → removable devices → Selecteer utp kabel → Disconnect from host

10. Ga naar Edit → Virtuel Network editor → Change settings à Click vmnet 0 → Vink bridged → Bij bridged to selecteer de utp kabel.

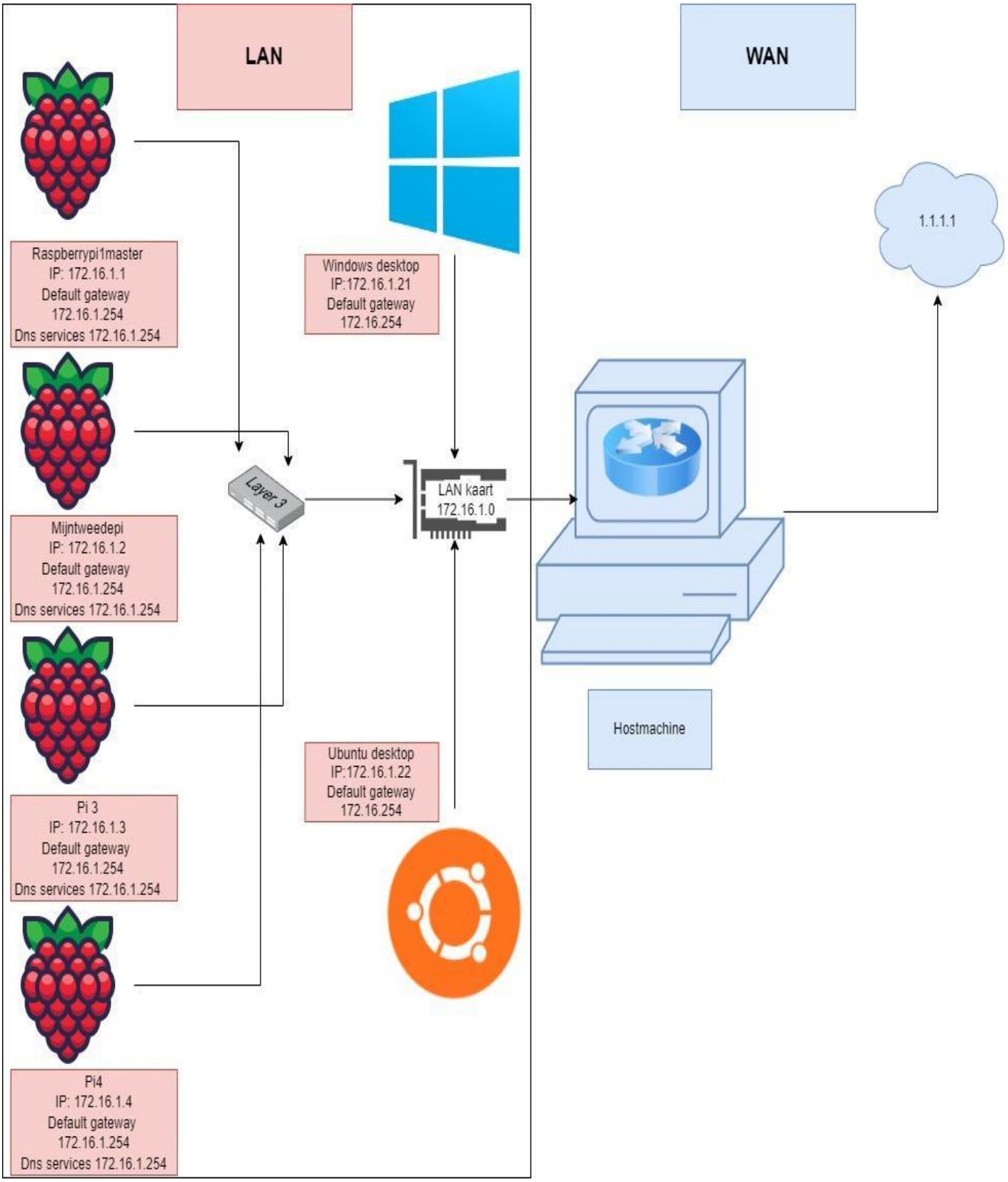
In mijn voorbeeld wil ik graag gebruik maken van putty dit betekent dat ik een windows machine wil verbinden aan de raspberry pi's die bridged verbonden zijn aan vmnet 0.

1. Download de windows enterprise ISO.
2. In vmware voeg een netwerkadapter toe zet die op bridged automatic en vink *"replica phyiscal network conection state"* aan.
3. Vul vervolgens het netwerk ip adres en bijbehorende dns servers en default gateway in de windows machine.
4. Download putty 64 bit

   Later zal rancher gebruikt worden, dit betekent dat er ook een windows machine verbonden moet worden aan het netwerk.

   1. Installeer de ubuntu iso
   2. En die vervolgens dezelfde stappen als hierboven genoemd opnieuw

Resultaat :


LAN
WAN
1.1.1.1
Raspberrypi1master
IP: 172.16.1.1
Default gateway
172.16.1.254
Dns services 172.16.1.254
Windows desktop
IP:172.16.1.21
Default gateway
172.16.254
Layer 3
LAN kaart
172.16.1.0
Mijntweedepi
IP: 172.16.1.2
Default gateway
172.16.1.254
Dns services 172.16.1.254
Hostmachine
Ubuntu desktop
IP:172.16.1.22
Default gateway
172.16.254
Pi 3
IP: 172.16.1.3
Default gateway
172.16.1.254
Dns services 172.16.1.254
Pi4
IP: 172.16.1.4
Default gateway
172.16.1.254
Dns services 172.16.1.254

Stap 4 - K3s voorbereiding

1. Wordt Root type in **sudo su -**
2. SSH naar je Pi
3. Schakel Ip tabellen in. Paste **sudo iptables -F**

   sudo update-alternatives --set iptables /usr/sbin/iptables-legacy

4. **sudo update-alternatives --set ip6tables /usr/sbin/ip6tables-legacy** daarna doe sudo reboot
5. Reboot (Na de reboot is de pi klaar voor een kubernetes installatie)

Stap 5 - K3s install

1. Type in **sudo su -**

   Bij het gebruiken van kubernetes in dit project is een soort hiërarchie aanwezig, waarin alle worker noods luisteren naar de master node. In stap 3 zullen we de master node configureren.

2. Installeer k3 (master setup)

   curl -sfL https://get.k3s.io | K3S_KUBECONFIG_MODE="644" sh -s -

3. Om ervoor te zorgen dat de “nodes” dus de verschillende Pi’s luisteren naar 1 pi de “master” moet er een token worden aangevraagd aan de master. Type in **sudo cat /var/lib/rancher/k3s/server/node-token**

Toelichting Stap 1.2. curl -sfL https://get.k3s.io | K3S_KUBECONFIG_MODE="644" sh -s -
curl = Download het script.
get.k3s.io = bevat kubernetes
K3S_KUBECONFIG_MODE="644" = Maakt kubernetes compaitibel met Rancher

Stap 4 - node instaleren

1. Paste dit commando in de pi's die gebruikt zullen worden als worker nodes.

curl -sfL https://get.k3s.io | K3S_TOKEN="JouwToken" K3S_URL="https:// [Ip Address van jouw master node:6443" K3S_NODE_NAME="Pi_2" sh -

Toelichting stap 4.1. Bij 'NODE_NAME' is het belangerijk dat elke Pi een andere naam krijgt.

Stap 5 - Rancher installeren

Rancher is een monitor service met een mooie gui die bedoeld is voor cluster het is dus heel handig om dit ook te installeren.

1. Maak een ubuntu 18.04 machine aan in Vsphere.
2. Maak een aantal directories aan

   mkdir /etc/rancher

   mkdir /etc/rancher/rke2

3. Maak een config file aan in de twee directory. Genaamd **nano config.yaml.**

   In de config file schrijf je:

   token : eenWachtwoord

   tls-san: Ipadres van nieuw aangemaakte ubuntu machine

4. Installeer Rancher

curl -sfL https://get.rancher.io | sh -

5. Check installatie

Type **rancherd --help**

6. Schakel Rancher in. (Type de volgende commando's in)

systemctl enable rancherd-server.service

systemctl start rancherd-server.service

journalctl -eu rancherd-server -f

7. Reset admin wachtwoord met het volgende commando.

rancherd reset-admin

Stap 6 - Rancher koppelen aan cluster

1. Click Add cluster → Other cluster → Naam geven → Create → Copy paste het tweede commando (Geen SSL).
2. Ga terug naar de master node en paste het commando

   Plaats een arm image in de Rancher Api.

3. Ga weer naar Rancher → click op de drie puntjes → View in Api → Edit → Bij agentImageOveride paste je dit commando : rancher/rancher-agent:v2.5.8-linux-arm64 → Scroll naar beneden → Send request → close

Op dit moment lukt het mij nog niet om de pi cluster robuust te maken.

De pi cluster werkt half soms kunnen pi1(master) en pi2 niet verbinding maken met het internet en soms wel. Beiden kunnen slechts de router pingen. En alle andere machines in het lan netwerk en ze kunnen ook terug worden gepingd.

Dit terwijl de andere twee pi's wel gewoon verbindingen kunnen maken met netwerk. Af en toe kunnen de pi's weer wel verbinding maken met het internet dus het werk half.

Dit had 4 mogelijk redenen.

Te lage voltage

De usb hub verwerft een lagere voltage dan dat de pi's gebruiken dit kan een mogelijke reden zijn dat de pi's minder goed werken.

In plaats van een usb hub heb ik een sterkere adapter geprobeerd echter heeft dit het probleem niet verholpen.

Firewall rule

Ik heb de volgende configuratie gedaan.

Ik wil ssh openzetten aan de wan poort

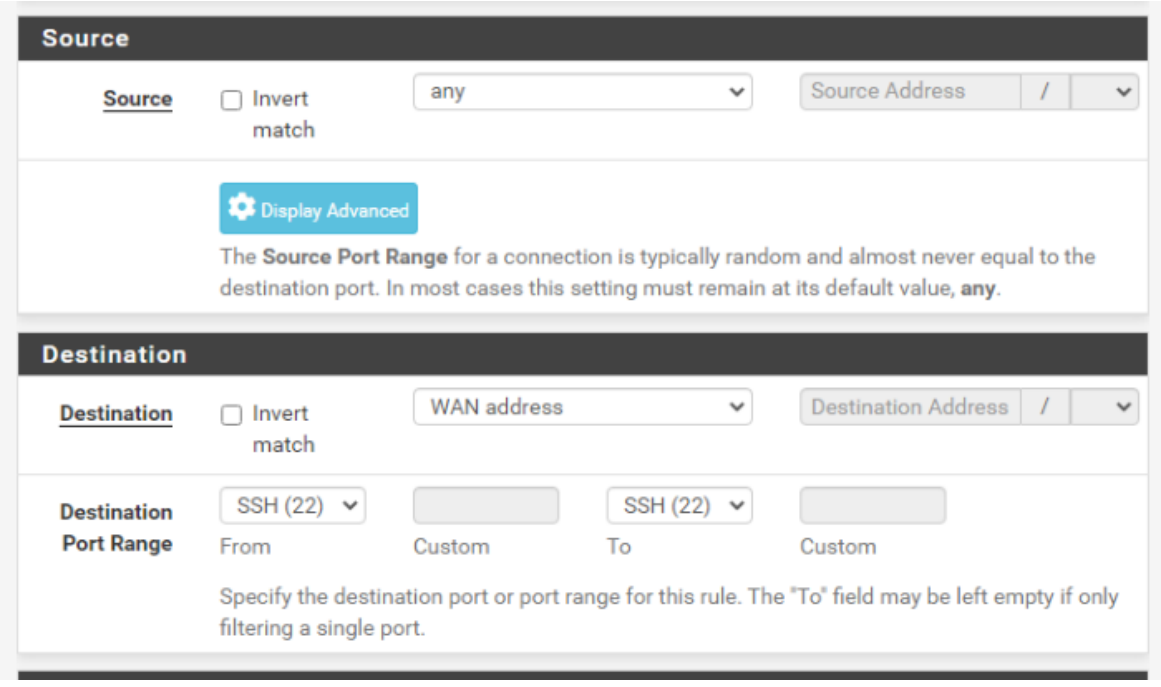
action = pass

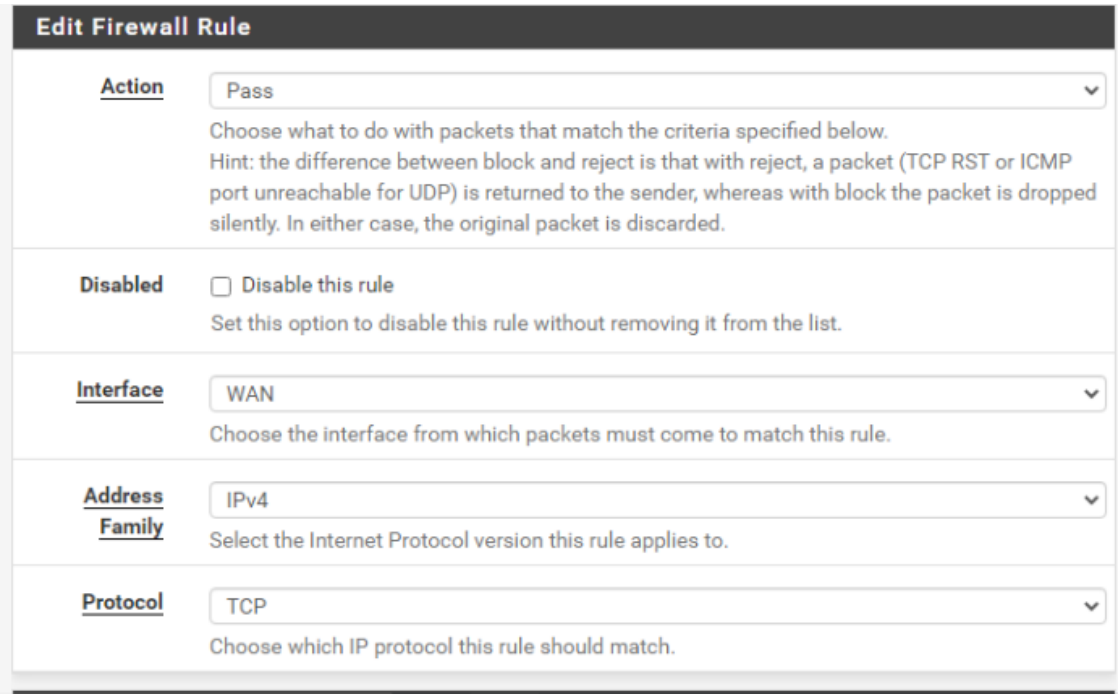
Interface = wan

Protocol = TCP

Source = any

Destination = From SSH (22) To SSH(22)





Netwerkapparatuur

Ik heb uitvoerig getest of de netwerk kabels en usb kabels wel goed werkte. En de pi4 en pi3 blijven goed werken terwijl de andere 2 pi's af en toe geen internet meer hebben

Verkeerde dns/ip ingevuld

De dns/ip configuratie is overal exact hetzelfde het enige verschil is de ingevoerde statische ip.

Verder heb ik geprobeerd om verschillende dns services te gebruiken op verschillende pi's maar dit had weinig succes.

Omdat het nogal lastig is om een remedie te vinden voor dit probleem zal ik de volgende cluster/kubernetes gerelateerde opdrachten maken in het seclab met de overgebleven tijd zal ik werken aan de fysiek kluster.

Als gevolg van deze complicaties ga ik eerst alle andere leerdoelen afwerken voordat ik terug keer naar dit probleem.

Oplossing

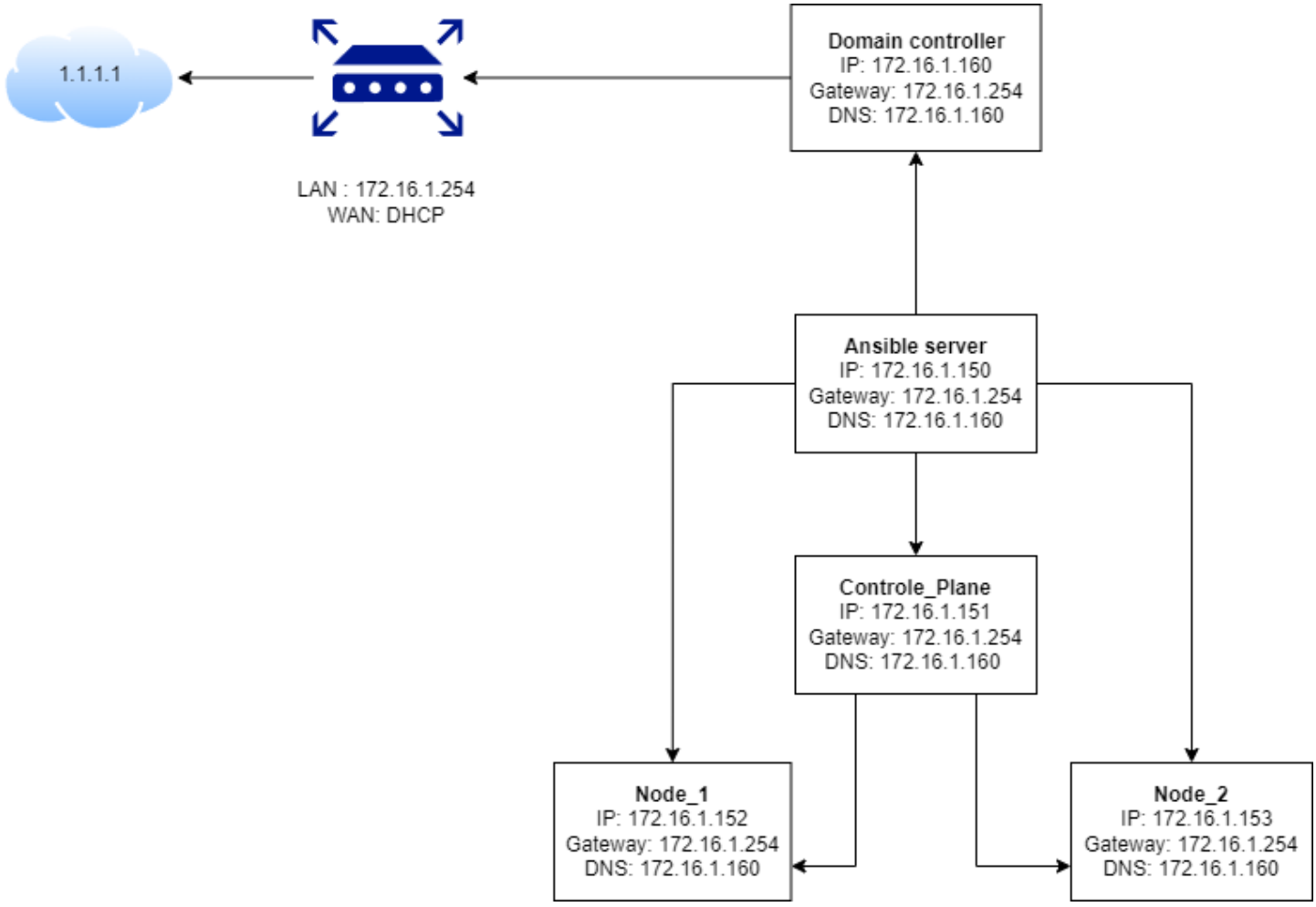
Het probleem zat hem in de dns configuratie. De oplossing is de domain controller ombouwen tot een dns server. En vervolgens wordt het ip adres van de DC gebruikt als dns server.

Ik ben erachter gekomen doordat een medestudent het zelfde probleem als ik had tegengekomen. Namelijk wanneer je veel nodes maakt treed dit probleem zich op. En moet je dus de dns configuratie aanpassen. Ik heb dit zelf uitgetest met het onderstaande project.

Kubernetes instellen met ansible (voorbereiding)

In dit gedeelte van het documenten wordt ansible gebruikt zodat bepaalde processen worden geautomatiseerd.

Netwerktekening:



1. **sudo apt-add-repository ppa:ansible/ansible**
2. **sudo apt update**
3. **sudo apt install ansible**

Ansible verbindt zich via ssh, dus zullen ssh verbindingen moeten worden opgezet.

1. Ga in de ansible server
2. Open terminal en type **ssh-keygen**
3. Wanneer je deze melding ziet

   Enter file in which to save the key (/home/demo/.ssh/id_rsa):

   druk op je op ENTER

4. Vull de passphrase in (dit is een soort wachtwoord die je zelf mag bedenken)

   Nu de sleutels zijn geïnstalleerd moet de public key op de andere nodes komen zodat ssh verbinding mogelijk is.

5. Open de node → terminal → type **sudo apt install openssh-server**

   Om de public key over te brengen naar de node zullen we wachtwoorden gebruiken. Om dit in te stellen zullen we een aanpassing moeten doen in de config files

6. type in **sudo nano /etc/ssh/sshd_config** En uncomment #PassworAuthenticaton door de hashtag we te halen
7. Vervolgens moet de ssh service opnieuw worden opgestart

   sudo service ssh restart

8. Ga terug naar de ansible server doe cd .ssh → type in

   ssh-copy-id node@172.16.1.152

   Dit kopieerd de key naar de node

9. Ga terug naar de node en type cd. ssh → cat authorized_keys

   Nu is de public key te zien.

10. Ga terug naar **sudo nano /etc/ssh/sshd_config** en plaats een comment voor PassworAuthenticaton.

    Dit zorgt ervoor dat je zonder wachtwoord verbindingen kan maken

Kubernetes instellen met ansible (uitvoering)

Maak een directory aan waarin de configuraties worden gedaan.

1. **mkdir ~/kube-cluster**
2. **cd ~/kube-cluster**

Maak een bestand waarin de hosts worden geconfigureerd. (de hosts waarmee ansible verbinding maakt)

3. **nano ~/kube-cluster/hosts**

[control_plane]

control1 ansible_host=**control_plane_ip** ansible_user=**root**

[workers]

worker1 ansible_host=**worker_1_ip** ansible_user=**root**

worker2 ansible_host=**worker_2_ip** ansible_user=**root**

[all:vars]

ansible_python_interpreter=/usr/bin/python3

Allen gekleurde teksten zijn punten die belangrijk zijn om benoemd te worden.

1. Oranje = de verschillende ansible groepen in dit project is dit belangrijk omdat het de controle plane onderscheid van de nodes.
2. Groen = dit zijn de ip adressen van de nodes
3. Rood = dit zijn de gebruikersnamen van de accounts op de nodes waar ansible aan verbindt.
4. Paars = geeft aan welke versie van python ansible moet gebruiken

Maak een yaml file aan die non root users maakt. Dit zorgt ervoor dat het monitoren van de virtuele machines kan gebeuren zonder dat er een risico bestaat dat belangrijke bestanden worden aangepast/verwijderd

4. **nano ~/kube-cluster/initial.yml**

hosts: all // installeer dit op alle hosts

become: yes // commando moet opnieuw worden uitgevoerd totdat het lukt

tasks:

- name: create the 'ubuntu' user

user: name=ubuntu append=yes state=present createhome=yes shell=/bin/bash // maakt de ubuntu gebruiker aan

- name: allow 'ubuntu' to have passwordless sudo // maak password less sudo aan

lineinfile:

dest: /etc/sudoers

line: 'ubuntu ALL=(ALL) NOPASSWD: ALL'

validate: 'visudo -cf %s'

- name: set up authorized keys for the ubuntu user // kopieerd ssh key naar de nieuw gecreerde gebruiker

authorized_key: user=ubuntu key="{{item}}"

with_file:
- ~/.ssh/id_rsa.pub

5. run de nano bestand(playbook)

   ansible-playbook -i hosts ~/kube-cluster/initial.yml

   installeer de kubernetes dependencies

6. Maak de volgend yaml bestand aan

   nano ~/kube-cluster/kube-dependencies.yml

Copy paste het onderstaande en verwijderd de comments

```
- hosts: all
  become: yes
  tasks:
   - name: create Docker config directory // Docker installatie
     file: path=/etc/docker state=directory


   - name: changing Docker to systemd driver //Zorgt ervoor dat docker vanaf boot opstart
     copy:
      dest: "/etc/docker/daemon.json"
      content: |
        {
        "exec-opts": ["native.cgroupdriver=systemd"]
        }


   - name: install Docker
     apt:
       name: docker.io
       state: present
       update_cache: true


   - name: install APT Transport HTTPS   // zorgt ervoor dat je externe https servers kan gebruiken
     apt:
```

```
    name: apt-transport-https
    state: present

- name: add Kubernetes apt-key
  apt_key:
    url: https://packages.cloud.google.com/apt/doc/apt-key.gpg
    state: present
- name: add Kubernetes' APT repository
  apt_repository:
   repo: deb http://apt.kubernetes.io/ kubernetes-xenial main
   state: present
   filename: 'kubernetes'

- name: install kubelet
  apt:
    name: kubelet=1.22.4-00
    state: present
    update_cache: true

- name: install kubeadm
  apt:
    name: kubeadm=1.22.4-00
```

```
      state: present


- hosts: control_plane
  become: yes
  tasks:
   - name: install kubectl
     apt:
       name: kubectl=1.22.4-00
       state: present
       force: yes
```

Copy paste het volgende commando

ansible-playbook -i hosts ~/kube-cluster/kube-dependencies.yml

Als laatste moeten we een tool installeren die de routing van de pods regelt.

nano ~/kube-cluster/control-plane.yml

```
hosts: control_plane
  become: yes
  tasks:
    - name: initialize the cluster
      shell: kubeadm init --pod-network-cidr=10.244.0.0/16 >> cluster_initialized.txt //Geeft het subnet aan waarin de pod zit
      args:
```

```
      chdir: $HOME
      creates: cluster_initialized.txt


  - name: create .kube directory // Directory word aangemaakt waarin admin key files worden geplaatst
    become: yes
    become_user: ubuntu
    file:
      path: $HOME/.kube
      state: directory
      mode: 0755


  - name: copy admin.conf to user's kube config
    copy:
      src: /etc/kubernetes/admin.conf
      dest: /home/ubuntu/.kube/config
      remote_src: yes
      owner: ubuntu
```

```
  - name: install Pod network

    become: yes

    become_user: ubuntu

    shell: kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml >>
pod_network_setup.txt

    args:

      chdir: $HOME

      creates: pod_network_setup.txt
```

! **Opmerking** het valt misschien op dat ik in de bovenstaand netwerktekening een domain controller heb toegevoegd.

De reden hiervoor is dat in een cluster soms een dns probleem kan ontstaan. Door de domain controller de functie te geven waarin het dns servers aanmaakt. Wordt het probleem opgelost dit is hoogstwaarschijnlijk ook de reden waarom mijn fysieke cluster niet werkte.

Services op de cluster

Dit script rolled meerdere nginx webservers uit met 1 configuratie.

apiVersion: apps/v1

kind: Deployment **// Type manifest is een deployment**

metadata:

  name: nginx-deployment

  labels:

```
    app: nginx
spec:
  replicas: 6  // de hoeveelheid applicaties die we willen
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx //naam van de container
        image: nginx:1.16 //Gebruikte image
        ports:
        - containerPort: 8080
```

Meerdere wordpress services uitrollen

Voor het uitrollen van meerdere wordpress websites moeten vier dingen gebeuren.

1. Mysql persistent volume claim yaml bestand.

   Dit maakt een pod aan die de volume claimed van de pod waarin de data wordt opgeslagen voor de mysql container.

2. Wordpress persisten volume claim yaml bestand.

   Dit maakt een pod aan die de volume claimed van de pod waarin de data wordt opgeslagen voor de mysql container.

3. Deploy mysql via het yaml script. In dit script is mysql ook verbonden met wordpress, volume claim en volume mount.
4. Deploy wordpress via het yaml script. In dit script is wordpress ook verbonden met volume claim en volume mount.

Mysql persistent volume

```
apiVersion: storage.k8s.io/v1beta1
kind: StorageClass
metadata:
  name: portworx-sc-repl3
provisioner: kubernetes.io/portworx-volume
parameters:
  repl: "3"
  priority_io: "high"
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pvc-1  // Naam claimed volume in de pod
  annotations:
```

```
    volume.beta.kubernetes.io/storage-class: portworx-sc-repl3 // portworx repareerd de node na een failure
spec:
  accessModes:
    - ReadWriteOnce // Data mag gelezen worden en bewerkt
  resources:
    requests:
      storage: 2Gi // grote die de volume moet hebben
```

Wordpress persistent volume

```
apiVersion: storage.k8s.io/v1beta1
kind: StorageClass
metadata:
  name: portworx-sc-repl3-shared
provisioner: kubernetes.io/portworx-volume
parameters:
  repl: "3"
```

```
  priority_io: "high"
  shared: "true"
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-pv-claim
  labels:
    app: wordpress
  annotations:
    volume.beta.kubernetes.io/storage-class: portworx-sc-repl3-shared
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

Wordpress image deployen

apiVersion: v1 **// service api versie**

kind: Service **// service type service**

```
metadata:
  name: wordpress
  labels: // zal verbinding maken met de selector (dan weet de deployment welke pods erbij horen)
    app: wordpress
spec:
  ports:
    - port: 80 // bereikbaar vanaf poort 80
      nodePort: 30303 // opent deze poort op alle nodes
  selector:  // maakt connectie met de onderstaande deployment file
    app: wordpress
    tier: frontend
  type: NodePort
---
apiVersion: extensions/v1beta1
kind: Deployment // type deployment
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  replicas: 3 //hoeveelheid websites
  strategy: // manier waarop de applicatie reageert bij een update
```

```
  type: Recreate // verwijder de oude versie en rol de nieuwe versie uit
 template:
  metadata:
   labels:
    app: wordpress
    tier: frontend
  spec:
   schedulerName: stork // stork achterhaald welke pod op welke node zit
   containers:
   - image: wordpress:4.8-apache
    name: wordpress
    imagePullPolicy: // Bevat de environment variables
    env:
    - name: WORDPRESS_DB_HOST
     value: wordpress-mysql
    - name: WORDPRESS_DB_PASSWORD
     valueFrom:
      secretKeyRef:
       name: mysql-pass
       key: password.txt
    ports:
    - containerPort: 80
```

```
        name: wordpress
      volumeMounts:
      - name: wordpress-persistent-storage
        mountPath: /var/www/html  // Geeft aan waar de volume zijn data opslaat
    volumes:
    - name: wordpress-persistent-storage
      persistentVolumeClaim:
        claimName: wp-pv-claim  // Verbind de volume aan de volume claim container deze naam is ook terug te vinden in de volume claim file van wordpress
```

Mysql image deployen

```
apiVersion: v1
kind: Service
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  ports:
    - port: 3306
  selector:  // Maakt verbinding met de wordpress  deployment
    app: wordpress
    tier: mysql
```

```
  clusterIP: None
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: mysql
    spec:
      # Use the stork scheduler to enable more efficient placement of the pods
      schedulerName: stork
      containers:
      - image: mysql:5.6
        imagePullPolicy:
        name: mysql
        env:
          # $ kubectl create secret generic mysql-pass --from-file=password.txt
```

```
      # make sure password.txt does not have a trailing newline
    - name: MYSQL_ROOT_PASSWORD
      valueFrom:
        secretKeyRef:
          name: mysql-pass
          key: password.txt
    ports:
    - containerPort: 3306
      name: mysql
    volumeMounts:
    - name: mysql-persistent-storage
      mountPath: /var/lib/mysql
  volumes:
  - name: mysql-persistent-storage
    persistentVolumeClaim:
      claimName: mysql-pvc-1
```