BULUT BİLİŞİM ve DEVOPS TEKNOLOJİLERİ

BÜTÜNLEME PROJE ÖDEVİ

Ali İhsan Kaya

Proje Danışmanı

02200201006

Samed Sonkaya

İÇİNDEKİLER

1. VirtualBox ile Linux Ubuntu Kurulumu

1.1 Gerekli dosyaların İndirilmesi

Google üzerinden bilgisayarımıza :

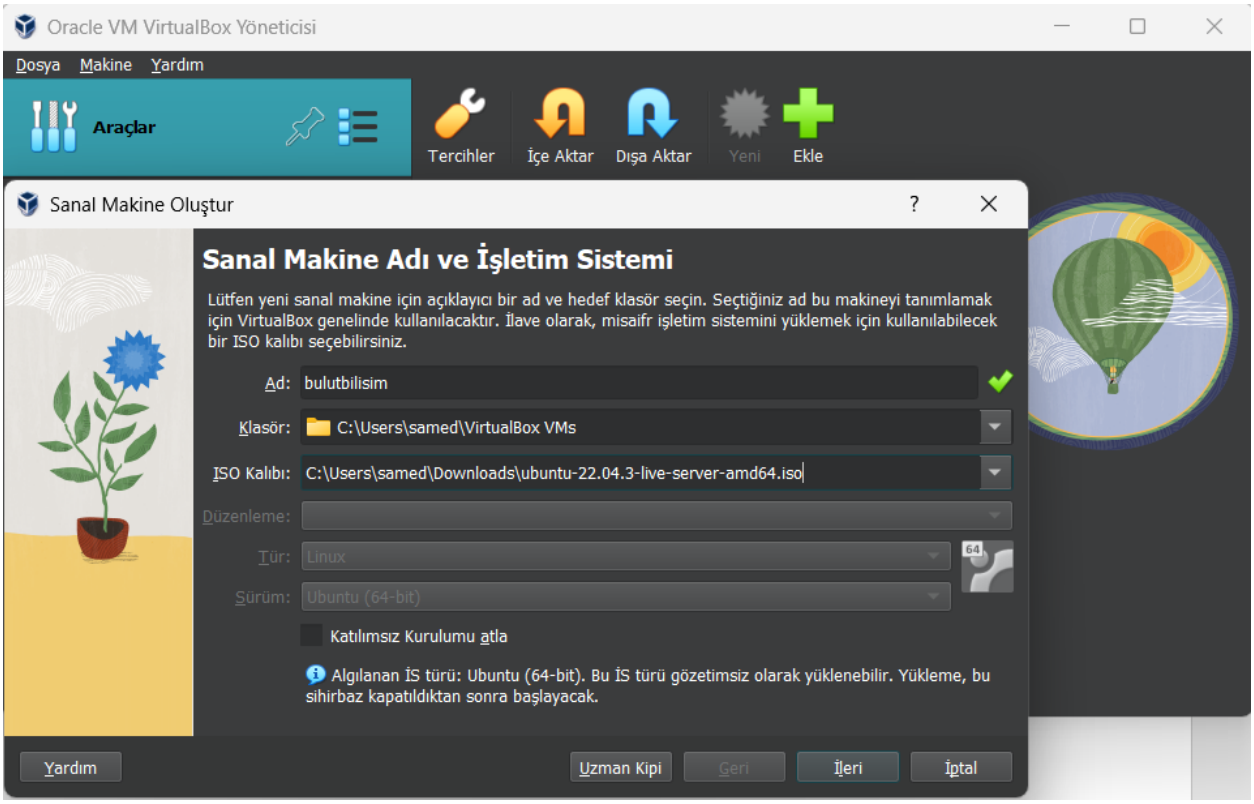
- Oracle VM VirtualBox 7.0.14 Windows host.

- Ubuntu Server 22.04.3

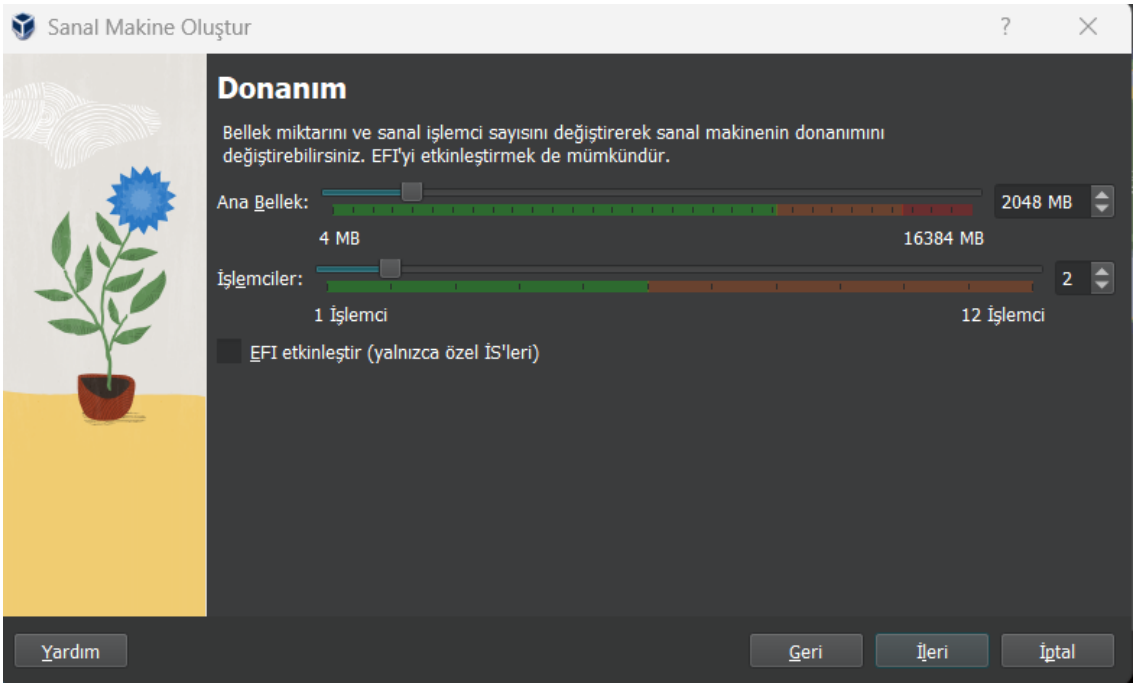
Dosyalarını indiriyoruz.

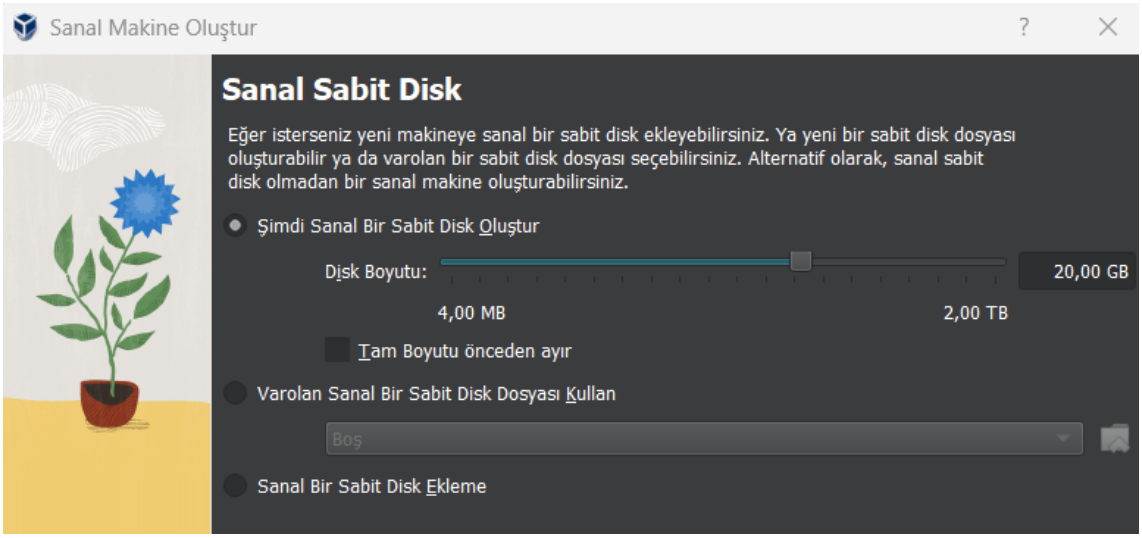
1.2 Ubuntu Kurulumu

Bilgisayarımıza kurduğumuz VirtualBox'da yeni kısmına basarak yeni bir sanal makine oluşturup ISO kalıbı olarakta indirdiğimiz ubuntu server dosyasını seçiyoruz ve ilerliyoruz.



Sanal bilgisayarım için ayıracağım bellek, işlemci miktarı ve disk boyutunu belirliyorum.



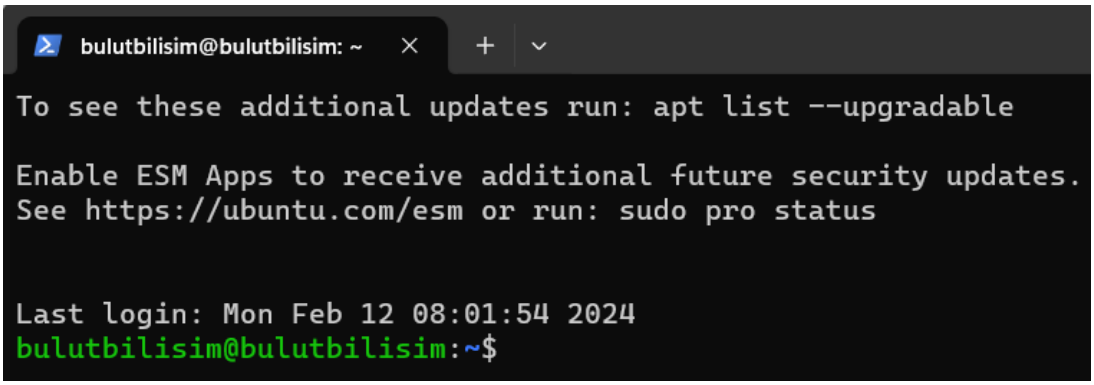


1.3 Ubuntu Kullanımı İçin Ön İşlemler

Oluşturduğum sanal bilgisayarı çalıştırarak kurulum sihirbazını başlattım. Kullanıcı adı ve şifre bilgilerini doldurduktan sonra sistemin kurulmasını bekledim. Kurulum bittikten sonra kullanıcı adı ve şifre ile giriş yaptım.

Sanal bilgisayarımda ssh toolu kullanmak için port ayarlarını yaptım ve Windows PowerShell'den sanal bilgisayarıma eriştim.

Ad	Protokol	Anamakine IP	Anamakine B.Noktası	Misafir IP	Misafir B.Noktası
Rule 1	TCP		2222	10.0.2.15	22



PowerShell üzerinden

- sudo apt update
- sudo apt upgrade
- sudo apt install curl
- sudo apt install net-tools

komutları ile temel işlemleri yaptım.

2. Ubuntu Kubernetes Master Node ve Docker Kurulumu

2.1 Swap Ayarı

-sudo swapoff -a

-sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab

2.2 Iptables Bridged Traffic Ayarı

-cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

br_netfilter

EOF

-cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

EOF

-sudo sysctl –system

2.3 Containerd Kurulumu

-sudo apt-get remove docker docker-engine docker.io containerd runc

-sudo apt-get update

-sudo apt-get install ca-certificates curl gnupg lsb-release

-sudo mkdir -m 0755 -p /etc/apt/keyrings

-curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

-echo \ "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \ $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

-sudo chmod a+r /etc/apt/keyrings/docker.gpg

-sudo apt-get update

-sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y

-containerd config default | sudo tee /etc/containerd/config.toml

-sudo su –

-sed -i 's/SystemdCgroup = false/SystemdCgroup = true/g' /etc/containerd/config.toml

-exit

-sudo systemctl enable docker && sudo systemctl start docker

-sudo systemctl enable containerd && sudo systemctl start containerd

-sudo docker run hello-world

```
bulutbilisim@bulutbilisim:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:4bd78111b6914a99dbc560e6a20eab57ff6655aea4a80c50b0c5491968cbc2e6
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

2.4 Kubeadm Kurulumu

-sudo apt-get update

-sudo apt-get install -y apt-transport-https ca-certificates curl gpg

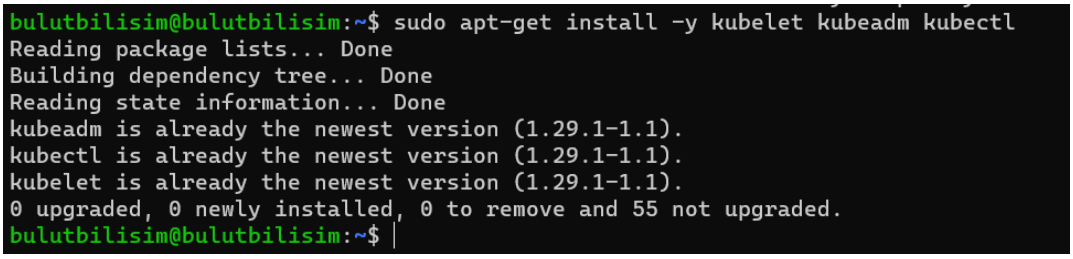
-sudo curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

-echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

-sudo apt-get update

-sudo apt-get install -y kubelet kubeadm kubectl

-sudo apt-mark hold kubelet kubeadm kubectl



Kubeadm kurulumunu yaptıktan sonra worker node'lar için sanal makinayı klonluyoruz.

2.5 Kubernetes Cluster Kurulumu

Master node içine yazılacak komutlar:

-sudo kubeadm config images pull

-sudo kubeadm init --pod-network-cidr=176.232.0.0/16 --apiserver-advertise-address=176.232.180.97 --control-plane-endpoint= 176.232.180.97

-mkdir -p $HOME/.kube

-sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

-sudo chown $(id -u):$(id -g) $HOME/.kube/config

-kubectl taint nodes --all node-role.kubernetes.io/master-

-kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/tigera-operator.yaml

-kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/custom-resources.yaml

-watch kubectl get pods -n calico-system

3. Ubuntu Kubernetes Worker Node Kurulumu

3.1 Etc/hosts Dosyalarının Hazırlanması

Sanal makinalarımızın &nano etc/hosts dosyasına alttaki komutları yazıyoruz.

127.0.0.1	localhost
176.232.180.97	bulutbilisim
176.232.180.98	worker1
176.232.180.99	worker2

3.2 Worker Node'ların Master Node'a bağlanması

Master node'da oluşturduğumuz cluster'ın join komutunu worker node'larda çalıştırarak birbirine bağlıyoruz.

sudo kubeadm join 176.232.180.97:6443 --token niuruv.uupht6jyubhohbgc \ --discovery-token-ca-cert-hash sha256:b203a0b6d4cc8e147937b243752efc42e7a133f5a2cee4dddba3877ddcae8180

4. Ubuntu Helm ile Nginix Kurulumu

4.1 Helm Kurulumu

-curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null

-sudo apt-get install apt-transport-https --yes

-echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable/debian/ all main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list

-sudo apt-get update

-sudo apt-get install helm

4.2Metallb Kurulumu

-export KUBE_EDITOR="nano"

-kubectl edit configmap -n kube-system kube-proxy” komutu ile açılan dosyada strictARP: true, mode: "ipvs" satırlarını değiştiriyorum”

-kubectl get configmap kube-proxy -n kube-system -o yaml | \ sed -e "s/strictARP: false/strictARP: true/" | \ kubectl diff -f - -n kube-system

-kubectl get configmap kube-proxy -n kube-system -o yaml | \ sed -e "s/strictARP: false/strictARP: true/" | \ kubectl apply -f - -n kube-system

-kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.13.9/config/manifests/metallb-native.yaml

-sudo apt install sipcalc

-sipcalc 176.232.180.97/24

-cat > metallb-config.yaml << EOF

apiVersion: metallb.io/v1beta1

kind: IPAddressPool

metadata:

  name: first-pool

  namespace: metallb-system

spec:

  addresses:

  - 176.232.180.97-176.232.180.97

---

apiVersion: metallb.io/v1beta1

kind: L2Advertisement

metadata:

  name: default

  namespace: metallb-system

spec:

ipAddressPools:

- default

EOF

-kubectl apply -f metallb-config.yaml

4.3 İngress-Nginx Kurulumu

-helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx

-kelm repo update

-helm install ingress-controller ingress-nginx/ingress-nginx

-helm ls

-kubectl get deployments

5. Docker ile Rancher Kurulumu

Kurmuş olduğumuz Docker için gerekli işlemleri yapıyoruz.

-sudo groupadd docker

-sudo gpasswd -a $USER docker

-newgrp docker

Ufw allow komutu ile kullanacağımız portlarını açtıktan sonra Rancher'ın 2.4.18 versiyonunu indiriyoruz. Bu versiyonunu indirmemizin sebebi latest versiyonunda hataların oluşması.

-docker run -d --restart=unless-stopped \

-p 80:80 -p 443:443 \

--privileged \

rancher/rancher:v2.4.18

--acme-domain 176.232.180.97.nip.io

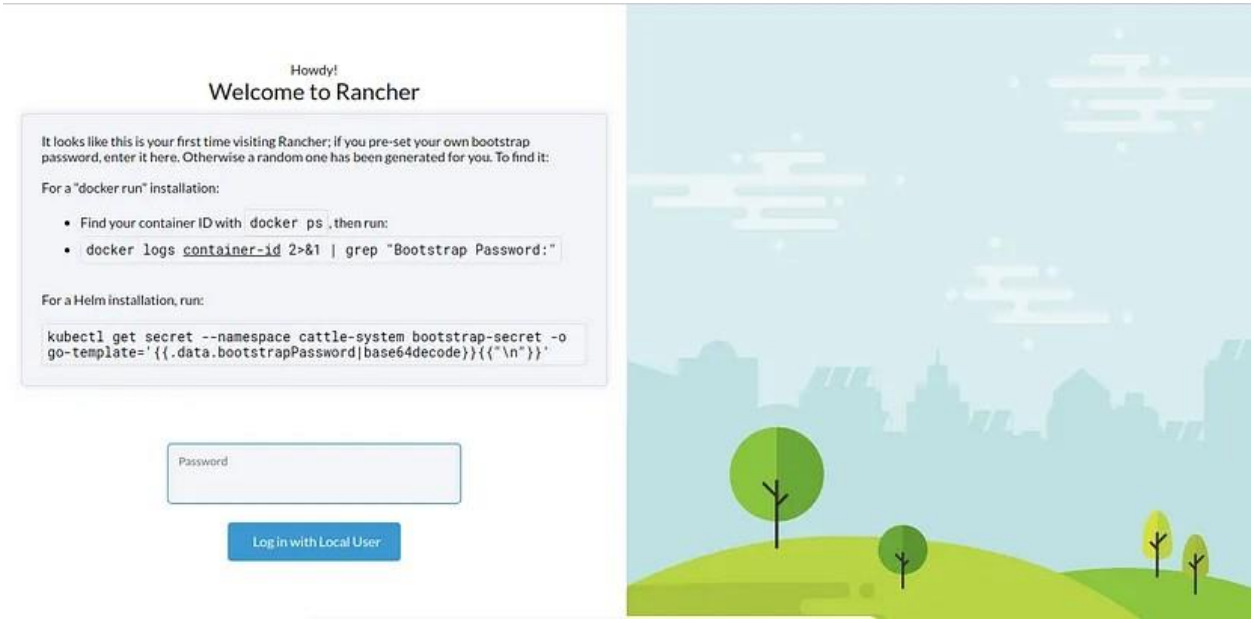
Kısa bir süre bekledikten sonra Docker ps ile Containerin çalışıp çalışmadığını control ediyoruz.

```
bulutbilisim@bulutbilisim:~$ curl ifconfig.co
176.232.180.97
bulutbilisim@bulutbilisim:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
2222/tcp                   ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
443/tcp                    ALLOW       Anywhere
22/tcp (v6)                ALLOW       Anywhere (v6)
2222/tcp (v6)              ALLOW       Anywhere (v6)
80/tcp (v6)                ALLOW       Anywhere (v6)
443/tcp (v6)               ALLOW       Anywhere (v6)

bulutbilisim@bulutbilisim:~$ docker ps -a
CONTAINER ID   IMAGE                     COMMAND                  CREATED          STATUS          PORTS
                                                 NAMES
981263856e05   rancher/rancher:v2.4.18   "entrypoint.sh --acm…"   25 minutes ago   Up 25 minutes   0.0.0.0:80->80/tcp, :
::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp   rancher-server
```

Daha sonrasında https://176.232.180.97.nip.io adresine giderek Rancher'ın adresine ulaşıyoruz.



Ekranda bulunan Docker komutunu çalıştırarak Password'ü öğrenip sisteme giriyoruz.

Daha sonrasında kendimize bir şifre belirleyip devam ediyoruz.

Rancher'in içine girdikten sonra oluşturduğumuz Kubernetes Cluster'ı dahil ediyoruz.

6. Rancher Üzerinden CD Süreci Başlatma

Rancher üzerinden Git Repo ayarlarını açıyoruz. Continuous Deployment sekmesini açıyoruz.

Git Repo adresimizi ve gerekli bilgileri dolduruyoruz.

Rancher da hangi dosyaların ve hangi değişin CD sürecini tetikleyeceğini belirliyoruz.

Daha sonrasında ayarları onaylayıp kaydediyoruz.